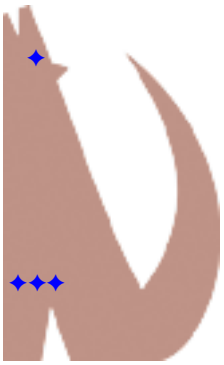
Active Contouring

✦ Discovering the Possibilities ✦✦✦



Active Contours in IDL

Overview

Active contours are also called “snakes”, because they are closed curves that writhe and move under the influence internal forces from the curve itself and external forces from the image environment. They are often used to identify regions of interest on images. The active contouring method employed here is a parametric method that uses an external force field called a *Gradient Vector Flow* as discussed by Chenyang Xu and Jerry Prince on their web page and in their March 1998 paper entitled “Snakes, Shapes, and Gradient Vector Flow” in the *IEEE Transactions on Image Processing*.

Parametric active contouring algorithms create parametric curves that move within an image domain towards features of interest, usually edges. The curves are drawn toward the edges by potential forces or gradients. There are also internal forces, such as elasticity forces that are designed to hold the curve together and bending or rigidity forces that prevent the curve from bending too much. Users of the *ActiveContour* program have the ability to adjust these external and internal forces to find the right conditions for optimal active contouring.

Installation

The *ActiveContour* program comes in a zip file. Create a new directory for these files, perhaps named “activecontour.” Extract all the files in the zip file into this new directory. You will find these five directories. Add the ***source*** directory to your path.

help	A directory containing the program help files.
images	A directory containing sample images.
source	A directory containing the *.pro or *.sav files.
parameters	A directory containing an example parameter definition files.
roi	A directory containing example final contours.

Running the Active Contour Program

Once installed, run the *ActiveContour* program from within IDL like this:

```
IDL> ActiveContour
```

You will be asked to select an example image file from the *images* directory. If you like, you can browse to any directory that holds image files. The program can work

with 2D images stored in BMP, DICOM, JPEG, PGM, PNG, and TIFF files. You see the image selection tool in Figure 1.

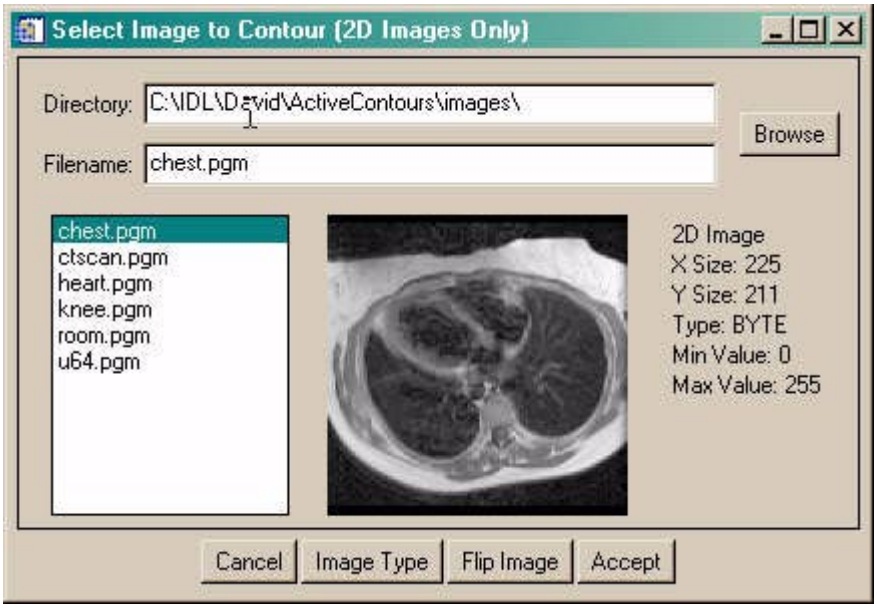


Figure 1: The image selection tool. Only 2D image can be selected for active contouring. Click the Browse button to select your own image stored in any of several standard image file formats.

Once you have selected an image, it will be displayed in a separate window and the *ActiveContour* control panel will appear, as shown in Figure 2. From the control panel, you can control the internal and external force parameters, other active contouring parameters, as well as manipulate the image before active contouring takes place. For example, the *Image Contrast* sliders will allow you to manipulate image contrast, and you can find smoothing functions listed under the *Adjustments* menu button.

Once you have manipulated the image (it may not need any or much manipulation), you can move your cursor into the image display window and select the starting coordinates for the active contouring algorithm. You will need at least three non-linear points to create an initial contour. Use your left or middle mouse button to select the points. You can close the contour curve by either clicking with the right mouse button, or by double clicking either the left or middle mouse button inside the image.

When you close the initial contour curve, you will see the curve smooth itself slightly. This is normal and is required for subsequent active contour processing.

If you wish to choose another curve, just continue to click in the image window with your left or middle mouse button. The first curve will be discarded and the algorithm will use the curve you are drawing now as its initial active contour position. You can experiment with the *ActiveContour* parameters and the initial contour position to see how far away from the final contour you will need to position the initial contour. In general, you have be close but not so close that you have to be meticulous in selecting the initial contour. You can see an example of an initial active contour position in Figure 3.

Once you hit the *Apply Active Contour* button, you will see the active contour begin to snake around to its final position. You see an intermediate stage in Figure 4.

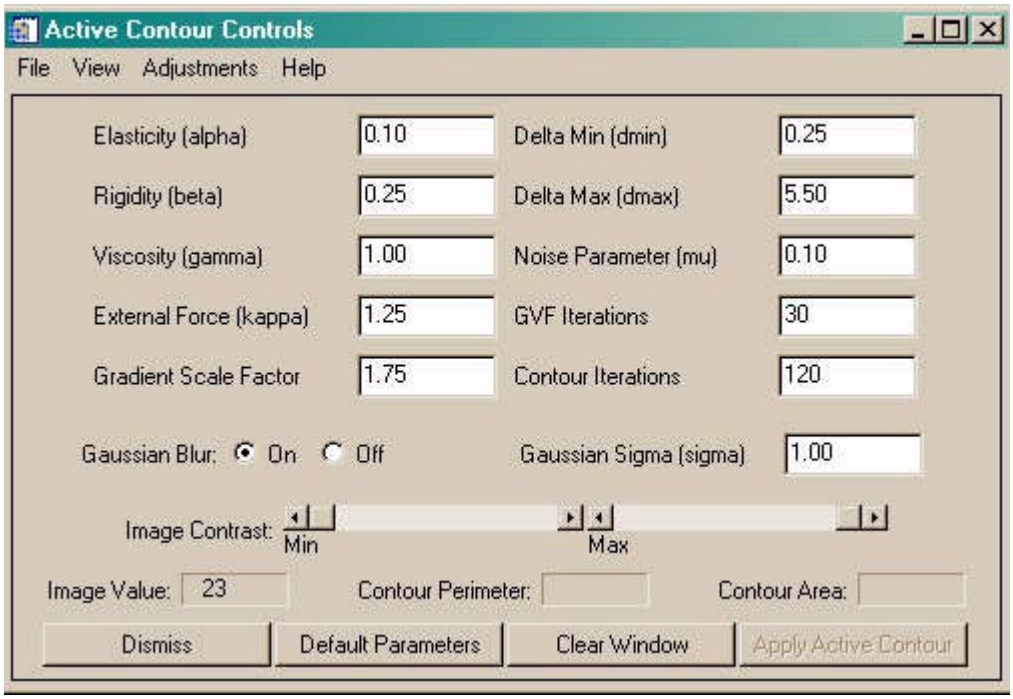


Figure 2: ***The ActiveContour Control Panel. The Apply Active Contour button does not become sensitive until an initial contour has been drawn in the image display window.***

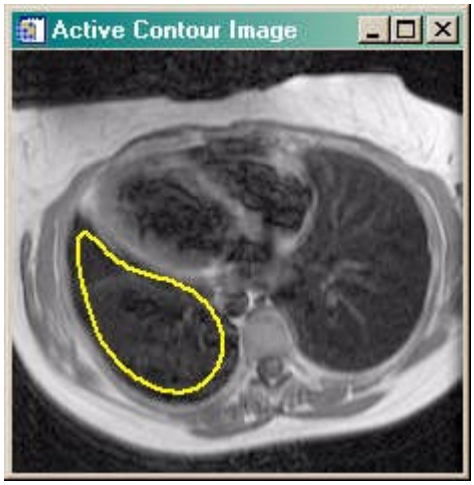


Figure 3: ***An initial contour drawn on an example chest image. The contour has been closed by double clicking in the window.***

Sometimes the active contour will converge to its final position before the number of iterations set in the control panel is reached. If this happens, you can click the *Accept* button on the progress bar that appears on your display when active contouring is in progress. The program will stop within a few seconds and the contour on the display when you hit the *Accept* button will be considered the final contour.

Once the final contour is determined (either the number contour of iterations has been reached or the user hit the *Accept* button), the initial and final contours will blink four

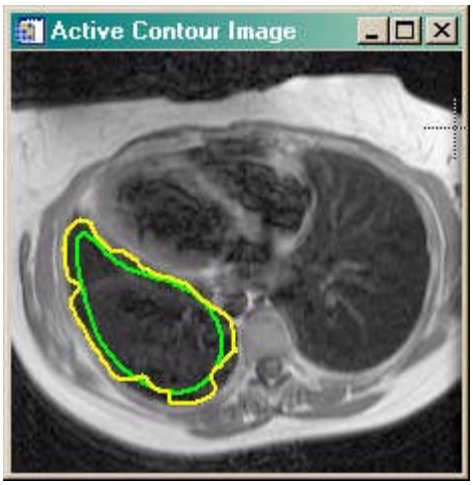


Figure 4: ***The intermediate active contour is shown in yellow, while the initial (now smoothed) contour is shown in green. This initial contour was well chosen, so the final contour converges quickly in about 25 iterations.***

times and the perimeter and area of the region of interest outlined by the final active contour is calculated and displayed in the control panel. You see an example of the final active contour in Figure 5.

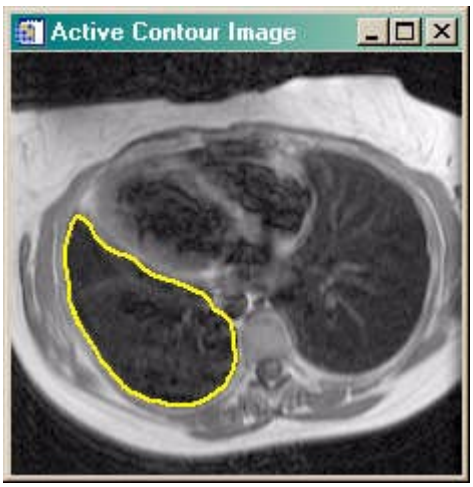


Figure 5: ***The final active contour. Once the contouring is finished, the area and perimeter of the contour is calculated and displayed on the control panel.***

If you hit the *Apply Active Contour* button again (perhaps after changing the active contour parameters), you will again start from the original contour points. Note that unless something has changed with the image, the GVF iterations are not re-calculated, since they are still current. This speeds experimentation significantly.

Active Contour Parameters

Active contours are like the little girl in the story who “when she was good, was very, very good, but when she was bad, was horrid.” That is to say, it is sometimes a challenge to get good active contour parameters in order to obtain acceptable results.

You get better with experience. The only good way to understand the parameters is to experiment with them and see how they affect the final contour. The *ActiveContour* program is designed to make this process as interactive and as easy as possible. Here are the parameters you can set interactively from the control panel shown in Figure 2.

Elasticity

The elasticity is known as the property *alpha* in the Xu and Prince paper cited in the *Overview*. Its value can be set with the **Alpha** keyword when calling the program from the IDL command line. The elasticity force acts to keep the curve from stretching. The default value is 0.1 units. Larger values make the contour harder to stretch along its length.

Rigidity

The rigidity is known as the property *beta* in the Xu and Prince paper. Its value can be set with the **Beta** keyword when calling the program from the IDL command line. The rigidity force acts to keep the curve from bending too much, as, for example, when turning a corner. The default value is 0.25 units. Larger values make the curve "stiffer" or harder to bend.

Viscosity

The viscosity is known as the property *gamma* in the Xu and Prince paper. Its value can be set with the **Gamma** keyword when calling the program from the IDL command line. The viscosity controls how quickly and how far the curve can be deformed between iterations. The default value is 1.0. Larger values make the curve harder and slower to deform. If you see the curve "jumping around" a lot, increase the viscosity.

External Force

The external force is known as *kappa* in the Xu and Prince paper. It's default value is 1.25. Larger values cause a stronger force toward the image edges. The value can be set with the **Kappa** keyword.

Gradient Scale Factor

The external force field "pushes" the curve toward edges in images. Edges are where image values change quickly from dark to light and visa versa. The force is calculated in terms of a gradient or first derivative operator. This factor simply multiplies the gradient force fields by this factor before they are used to calculate the Gradient Vector Flow (GVF). The default value is 1.75. The value can be set with the **GradientScale** keyword.

Delta Min/Delta Max

As the curve is deformed, points are either added to the curve or subtracted from it (so the curve can expand and contract). These factors determine the minimum and maximum pixel distance between adjacent points in the curve. If two adjacent points are further apart than the maximum, a point is added between them. If two adjacent points are closer together than the minimum, then one of the points is eliminated. You might have to play with these parameters depending upon the size of your original image and initial contour. These parameters are knows as *dmin* and *dmax* in the Xu and Prince paper, and can be set with the **DMin** and **DMax** keywords.

GVF Iterations

The Gradient Vector Flow fields are derived from the image by minimizing a pair of decoupled linear partial differential equations that diffuses the gradient vectors (in X and Y) of the image. This minimization must be done only once for each image (assuming the image doesn't change in appearance). The solution is arrived at iteratively. The default number of 30 is usually sufficient to arrive at a minimized solution, although it could be that 10 or 15 iterations will suffice. In any case, the *ActiveContour* program only performs

the calculation when necessary to improve interactivity. Set this value with the **GVF_Iterations** keyword.

Contour Iterations This is the number of times the active contour is deformed according to the internal and external force fields and then "processed" by the program to obtain the next starting contour location. The default is 120, but most active contours will settle to a stable configuration in far fewer iterations. You can click the *Accept* button on the *Contour Iteration* progress bar to accept the current configuration, even if you haven't reached the end of the iterations. Set this parameter with the **Iterations** keyword.

Noise Parameter The noise parameter is known as the property *mu* in the Xu and Prince paper. Its value can be set with the **Mu** keyword when calling the program from the IDL command line. The noise parameter governs the way the GVF field is calculated. Basically, this parameter should be set to higher values in noisy images and to lower values in normal images. The default value is 0.1.

Gaussian Blur Often the image gradients are more distinct, and provide a better GVF external force field, if a Gaussian smoothing function is applied to the image prior to calculating the image gradients. The default is to include this image smoothing.

Gaussian Sigma The sigma or overall shape of the Gaussian smoothing is set with this factor, called *sigma* in the Xu and Prince paper. A larger value of sigma results in smoother image than with a small value of sigma. The default is 1.0 and the parameter can be set with the **Sigma** keyword.

Image Contrast The image contrast sliders can be manipulated to control the visual contrast in the image. Quite often good separation of features can be obtained, or at least improved, by manipulating image contrast prior to applying the active contour algorithm. Moving the sliders will remove any preliminary active contour currently on the image, so perform image contrast enhancement before drawing the initial active contour. Note that these controls are only available if the *ActiveContour* program created its own image display window.

Active Contour Buttons

There are four buttons at the bottom of the *ActiveContour* control panel. They look similar to the buttons in Figure 6.

Figure 6: The buttons along the bottom of the ActiveContour control panel.

Dismiss This button destroys the control panel. If the *ActiveContour* program has created the image display window (as it does in the free demo), this window will be destroyed, also.

Default Parameters This button will load the default parameters into the control panel. This is a quick shortcut for loading the default parameter file under the *File->Load Parameters* menu item.

Clear Window This button removes any initial or final active contour from the image display window.

Apply Active Contour This button applies the GVF active contour algorithm to the initial contour currently on display in the image display window. The button is only active when an initial contour is ready to be processed by the algorithm.

Active Contour Menu Selections

File Menu

The *File* menu is where you will find the ability to open new image files and to save and restore previously used parameters and active contours. A brief description of each menu item shown in Figure 7.

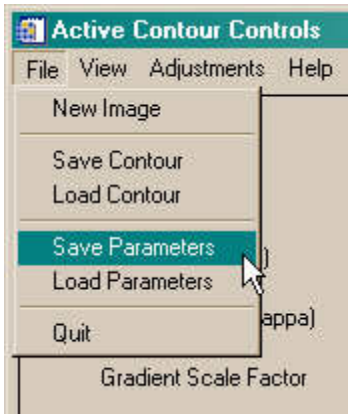


Figure 7: The File menu selections

New Image This button allows you to select and open a new image file for display in the image display window. BMP, Dicom, JPEG, PICT, PPM, PGM, PNG, and TIFF files can be opened. Note that only 2D images are allowed.

Save/Load Contour These buttons allow you to save and restore the final contour values. The area and perimeter of the final active contour are stored as well as the contour points themselves.

Save/Load Parameters It is an effort occasionally to find the right parameters to perform the active contouring adequately. Once you have them, you often want to save them so you can use them again on similar images. These menu buttons allow you to do so.

Quit This button quits the *ActiveContour* program and has exactly the same effect as the *Dismiss* button discussed above.

View Menu

The *View* menu allows you to view some of the intermediate results of the program as an aid to understanding active contouring.

Edgemap At the heart of the GVF active contouring method is an edgemap. This is the gradient of the image scaled into the range of 0 to 1. What you should see in the edgemap image are the edges of the image. The edgemap is used to calculate the external forces on the contour in the GVF algorithm The active contour is “pushed” or “pulled” toward the image edges. .

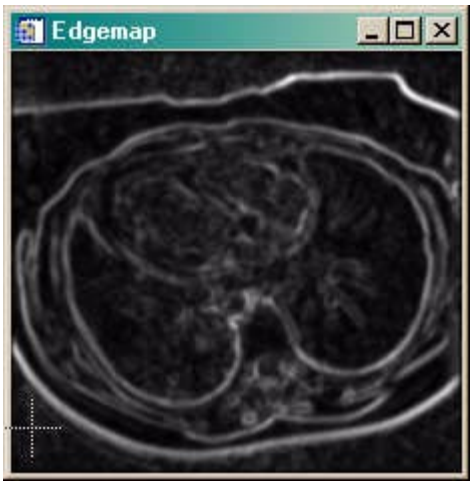


Figure 8: The edgemap image for the chest image used in the examples.

Gradient Fields The forces used in GVF active contouring are produced by gradient fields. To be useful, these fields need to be calculated in the X and Y directions. This menu item will show you the *Fx* and *Fy* gradient images.

Vector Field This button will allow you to view the vector force field (i.e., a plot of *Fx* vs. *Fy*) that is pushing the contour toward the image edges. Note that the vectors are strongest near the image edges and nearly perpendicular to the edges, as well.

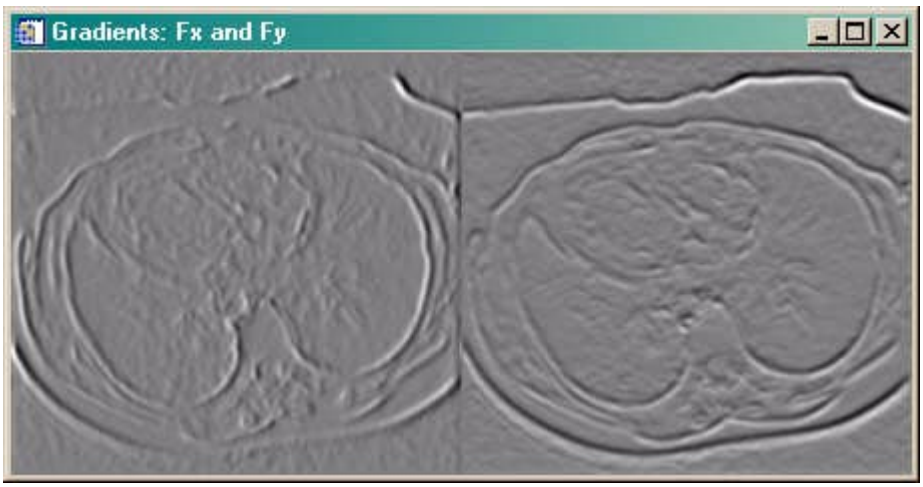


Figure 9: The gradient force fields Fx and Fy.

Adjustments Menu

The *Adjustments* menu allows you to smooth the image in various ways prior to calculating the active contour. This menu item is only available when the *ActiveContour* program creates its own image display window.

Undo This button removes either a *Smooth* or a *Median* filter operation in the opposite order in which they have been applied to the image. The button is only sensitive when there is a process to undo.

Smooth Filter This button allows you to apply a *Smooth* filter of various sizes to the image. Note that once a filter is applied, it can be removed by the *Undo* button.

Median Filter This button allows you to apply a *Median* filter of various sizes to the image. Note that once a filter is applied, it can be removed by the *Undo* button.

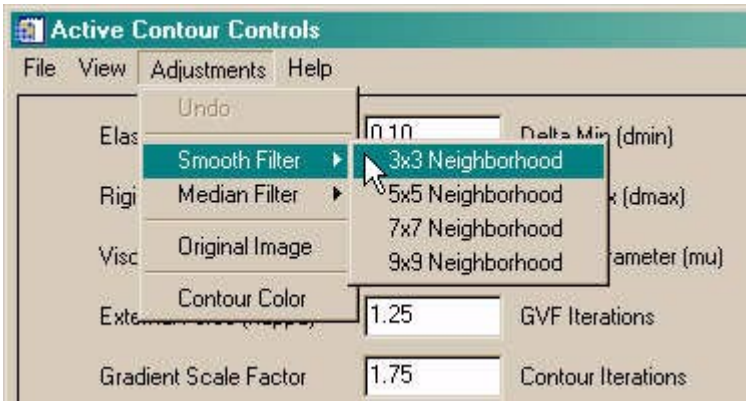


Figure 10: The Adjustments menu selections.

Original Image This button restores the original image in the image display window and discards any filter operations that have previously been applied to the image.

Contour Color This button will allow you to change the color used to draw the initial and final contours.

Help Menu

The *Help* menu spawns the Adobe Acrobat Reader to read the *ActiveContour User Guide* that is supplied with the *ActiveContour* program (this document), or you can look at an HTML version of the program documentation produced with in IDL command *MK_HTML_HELP*.

Active Contour Advanced Features

The *ActiveContour* program has been written in such a way that it can be a stand-alone application for learning about active contouring, or it can be a tool that is easily incorporated into your already-written IDL software. The stand-alone application is available for free downloading. It requires IDL 6.0 to run, or if you don't have that version of IDL, you can run it on the IDL Virtual Machine, available for free from Research Systems, Inc. The stand-alone version has features that make it possible to run the software from your own programs disabled.

To obtain a fully-functional version of the software, please contact Fanning Software Consulting, Inc.

Active Contour Written as an Object

The majority of the *ActiveContour* program is written as an object, which makes it convenient and simple to use in conjunction with your own IDL programs. Simply create the object and call its various methods to perform the active contouring in your own image display windows. You can have multiple versions of the object open at any

one time. For example, here is how *ActiveContour* might be called and used from within your application:

```
snake = Obj_New('ActiveContour', brainImage, $
   DrawID=theDrawWindow, Notify_Event='myprogram_snake', $
   Spatial_Scale=[0.04, 0.05])
```

The image *ActiveContour* is working with is the 2D image identified as *brainImage*, and the initial contours are going to be selected in the draw widget identified as *theDrawWindow*. The *ActiveContour* program will take over event handling for that particular draw widget, and will restore it to its normal event handling, when it is finished. The **Notify_Event** keyword is used to identify a user written procedure that will be notified or called when the initial and final contours are created. This is essentially an event handler, although the normal IDL event handling framework is bypassed in favor of a direct call to the procedure. The **Spatial_Scale** keyword allows you to specify the spatial X and Y scale of the image, so that the perimeter and area calculations can be done in image units rather than in the pixel units of the demo program. .

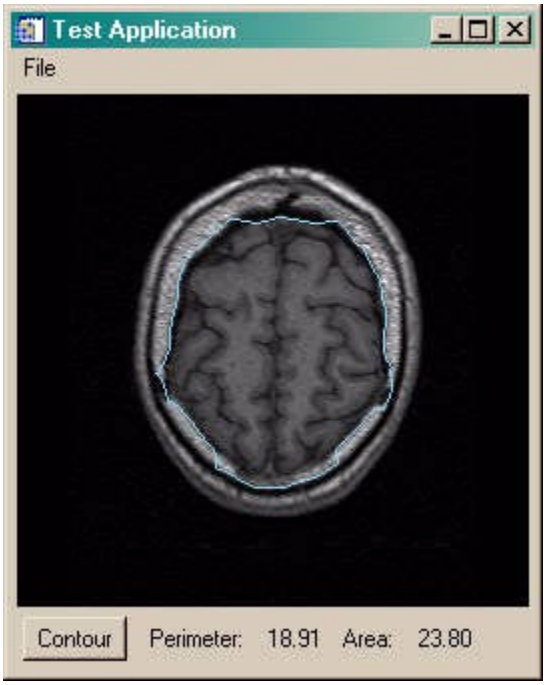


Figure 11: The TestApplication that comes with the ActiveContour code. This test program illustrates how to incorporate the ActiveContour object into your own program code.

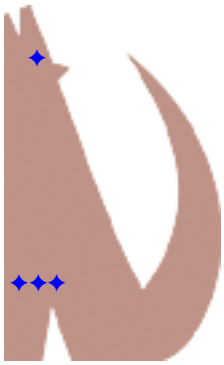
There is a *TestApplication* program distributed with the *ActiveContour* file distribution that illustrates how to use the ActiveContour program from within your own code. You can find the IDL code for *TestApplication* in Appendix A of this document.The program looks similar to the results in Figure 11, after the *ActiveContour* algorithm has been applied.

Contact Information

To report bugs or to discuss obtaining a fully-functional copy of the *ActiveContour* program, please contact the author, David Fanning. Here is his contact information.

David W. Fanning, Ph.D.
Fanning Software Consulting
1645 Sheely Drive
Fort Collins, CO 80526 USA
Phone: 970-221-0438
Fax: 970-221-4762
E-Mail: david@dfanning.com

Active Contouring



✦ Discovering the Possibilities ✦✦✦

Appendix A: IDL Code for a Test Application

```
PRO TestApplication_SnakeEvents, event
   Widget_Control, event.top, Get_UValue=info

   ; Two types of events can come in here. POINTS_COLLECTED indicates that
   ; the original contour has been drawn by the user. CONTROLS_KILLED
   ; indicates the the ActiveContour control panel was killed. This may
   ; have implications for the application program, so a notification
   ; occurs.
   CASE event.type OF

      ; OK, initial points have been collected.
      'POINTS_COLLECTED': BEGIN

           ; Apply the GVF Active Contour algorithm
           roiInfo = (*info).snake -> ApplyGVFSnake(Cancel=cancelled)

           ; Make the Contour button sensitive.
            Widget_Control, (*info).contourID, Sensitive=1

            ; Restore our own event handling and destroy the Control Panel
           (*info).snake -> ResetDisplay
           (*info).snake -> DestroyControls

           ; If the user cancelled, re-display the image.
            IF cancelled THEN BEGIN
               WSet, (*info).wid
               TV, BytScl((*info).image)

           ; If the Active Contour finished, print the results and draw the final contour.
            ENDIF ELSE BEGIN
               Widget_Control, (*info).plabel, Set_Value='   Perimeter: ' + $
                  String(roiInfo.perimeter, Format='(F8.2)')
               Widget_Control, (*info).alabel, Set_Value='   Area: ' + $
                  String(roiInfo.area, Format='(F8.2)')
               WSet, (*info).wid
               TV, BytScl((*info).image)
               PlotS, roiInfo.x, roiInfo.y, /Device, Color=FSC_Color('sky blue')
            ENDELSE
         END

       ; If the Control Panel was destroyed, redisplay the image and reset the
       ; event handling.
       'CONTROLS_KILLED': BEGIN
           WSet, (*info).wid
           TV, BytScl((*info).image)
           (*info).snake -> ResetDisplay
           Widget_Control, (*info).contourID, Sensitive=1
         END
      ELSE: ok = Dialog_Message('Unrecognized event.')
```

```
   ENDCASE
END; ----------------------------------------------------------------------------


PRO TestApplication_Contour, event

   ; Set up for contouring with the ActiveContour program.
   Widget_Control, event.top, Get_UValue=info

   ; Display the image.
   WSet, (*info).wid
   TV, BytScl((*info).image)

   ; Set the display to allow ActiveContour to take over draw widget control.
   (*info).snake -> SetDisplay

   ; Display the control panel (not required here, but gives the
   ; user a chance to change ActiveContour parameters.
   (*info).snake -> Controls

   ; Make the Contour button insensitive.

   Widget_Control, (*info).contourID, Sensitive=0
END ; ----------------------------------------------------------------------------


PRO TestApplication_DrawEvents, event

   ; Have to click the Contour button to draw initial contour.
   ok = Dialog_Message('Click the CONTOUR button to draw the initial contour.')
END ; ----------------------------------------------------------------------------


PRO TestApplication_Quit, event
   Widget_Control, event.top, /Destroy
END ; ----------------------------------------------------------------------------


PRO TestApplication_Cleanup, tl

   ; Be sure to destroy all objects, pointers, etc.

   Widget_Control, tlb, Get_UValue=info
   Obj_Destroy, (*info).snake
END ; ----------------------------------------------------------------------------


PRO TestApplication

   ; Error handling.

   Catch, theError
   IF theError NE 0 THEN BEGIN
      Catch, /Cancel
      ok = Error_Message(/Traceback)
      RETURN
   ENDIF

   ; Create the TestApplication widgets.
   tlb = Widget_Base(Title='Test Application', Column=1, MBar=menuID)
   fileID = Widget_Button(menuID, Value='File')
   button = Widget_Button(fileID, Value='Quit', Event_Pro='TestApplication_Quit')
   drawWindow = Widget_Draw(tlb, XSize=256, YSize=256, Button_Events=1, $
      Event_Pro='TestApplication_DrawEvents')
   labelBase = Widget_Base(tlb, Row=1)
   contourID = Widget_Button(labelBase, Value='Contour', $
      Event_Pro='TestApplication_Contour')
   plabel = Widget_Label(labelBase, Value='   Perimeter:       ', /Dynamic_Resize)
   alabel = Widget_Label(labelBase, Value='   Area:       ', /Dynamic_Resize)
   Widget_Control, tlb, /Realize

   ; Read the image data file.
   brainFile = Filepath(Subdir=['examples', 'data'], 'mr_brain.dcm')
   brainImage = Read_Dicom(brainFile)
```

```
   ; Create the ActiveContour object.
   snake = Obj_New('ActiveContour', brainImage, $
      DrawID=drawWindow, Notify_Event='TestApplication_SnakeEvents', $
      Spatial_Scale=[0.04, 0.05])
   IF Obj_Valid(snake) EQ 0 THEN BEGIN
      ok = Dialog_Message('Cannot create valid ActiveContour object. Returning.')
      Widget_Control, tlb, /Destroy
      RETURN
   ENDIF

   ; Get the window index number and display the image.
   Widget_Control, drawWindow, Get_Value=wid
   WSet, wid
   TV, BytScl(brainImage)

   ; Create an info structure and store it.
   info = { snake:snake, wid:wid, drawWindow:drawWindow, contourID:contourID, $
            plabel:plabel, alabel:alabel, image:brainImage }
   Widget_Control, tlb, Set_UValue=Ptr_New(info, /No_Copy)

   ; Start the program.
   XManager, 'testapplication', tlb, /No_Block, Cleanup='TestApplication_Cleanup'
END ; ---------------------------------------------------------------------------
```