AccuSoft Corporation
71 Lyman Street
Northborough, MA 01532

Toll Free 800.525.3577
Phone 508.351.9092
Fax 508.351.9086

VisiQuest 4.1 Unix Quick-Start Guide

This document is meant to provide new users of VisiQuest with an overview of the layout of the product, how to use the visual environment, how to open and modify sample workspaces and use your own data within a predefined space, how to create a common program for image processing, and, for users who will be creating their own glyphs, how to import their code to become part of the VisiQuest environment. It is our intention that this is used as a guide to begin to explore the vast capabilities of VisiQuest. Should you have any questions, please call us, or enter an issue into our support database and someone will get back to you promptly.

We hope you enjoy your experience.

The AccuSoft VisiQuest Team

Introduction to Programming with VisiQuest

VisiQuest is used to develop programs visually by combining glyphs into a complete program.

VisiQuest is used to create customized software by combining a rich set of tools (in information processing, data exploration and data visualization) into a complete program. For users who like to write code, VisiQuest also provides users with a way to import their own functions written in C, C++, Perl, or other scripting languages, and add them to the existing tools. All tools in the VisiQuest environment are represented within the Visual Programming Environment(VPE) as visual objects called *glyphs*. To create a visual program, the user selects the desired glyphs from the Glyph dropdown menu, places the glyph on the main window of the VisiQuest application (called the *workspace*) and connects the glyphs to indicate the order in which they will be run (called *workflow*). Such workspaces can be executed, saved, and restored to be used again or modified later. Workspaces may also be encapsulated into stand-alone applications with a very simplified graphical user interface so that they may be treated as independent VisiQuest applications.. In section 1, we will walk through this process.

For advanced users, the visual hierarchy, iteration, flow control, and expression-based parameters make VisiQuest a powerful simulation and prototyping system. VisiQuest interprets the visual network dynamically to schedule glyphs and then dispatches them as processes. The VisiQuest visual programming environment extends the basic data flow paradigm to make it a more powerful application prototyping or simulation environment. Data and control-dependent program flow is provided by flow control glyphs such as if/else, while, count, and trigger. Visual subroutines, or procedures, are available to support the development of hierarchical data flow graphs. Variables may be set interactively by the user, or calculated at run time via mathematical expressions tied to data values or control variables.

This document is designed to help the VisiQuest user get started quickly by covering some of the more commonly used functionality of the Visual Programming Environment. It is broken into three sections: 1) an introduction to the layout of the VisiQuest VPE and how to do basic operations, including an exercise for importing sample workspaces, 2) creating your own image processing program using existing glyphs, and 3) an advanced exercise for creating a complex program that solves an image analysis problem using a new glyph that you create.

Section 1: VisiQuest VPE Basics

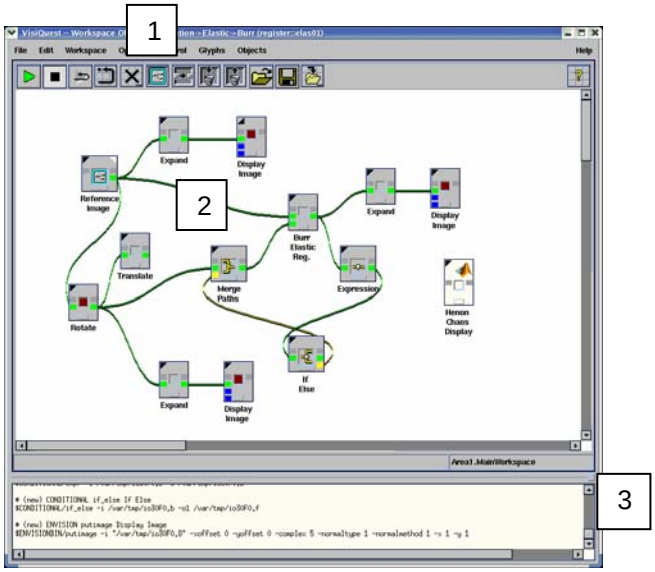
The best way to get started with VisiQuest is to use the software. This exercise starts with the assumption that the user has installed VisiQuest 4.1 on Unix, Linux, or Mac.

Invoking VisiQuest

In Unix, VisiQuest can be opened by typing in VisiQuest into the command shell.

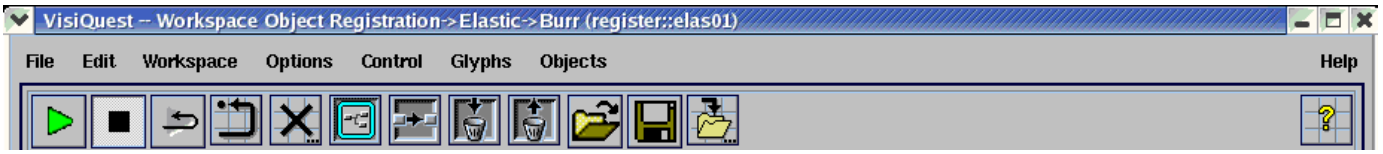
VisiQuest Layout

The VisiQuest 4.1 Visual Programming Interface displaying the Burr sample workspace.



1) The top set of Menus and controls, called the Toolbar, is used for managing and running your projects.
2) The main window of the product is the workspace. This is the location where you drag glyphs and connect them together to indicate workflow.
3) The console, or bottom text area of the workspace, provides the user with information returned while running the workspace.

Toolbar



File Menu: Contains the operations used to open, close and save your workspaces.

Edit Menu: Contains the commands to edit the glyphs on your workspace including Cut, Copy, Paste and Delete. Also contains the 'Find' function which is useful for locating glyphs that perform the functions you are searching for.

Workspace Menu: Contains the commands to run, stop, compile your workspace commands reside here. You can manage variables, and export your compiled workspace.

Options: Allows you to change the look and feel of your workspace. Choose the Preferences option on this menu to change the background color of the workspace, the connection types for the glyph, the way the data is transported between the glyphs, and the set of buttons which appear on your command bar.

Control: Contains all advanced programming controls.

Glyphs: Contains all of the 300+ glyphs for use in creating your new program. All glyphs are organized into categories and subcategories that loosely reflect their most common use. If you are having trouble navigating

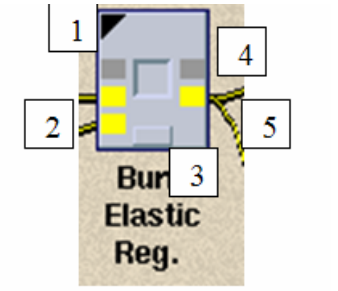
the glyphs, using the Find function available in the Edit menu will make it easier to understand the glyph locations.

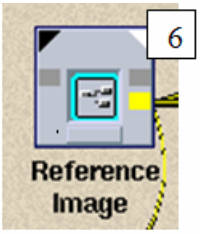
Objects: Contains all of the sample workspaces.

Help Menu: Contains standard help functions. By selecting the Tool Tips option, you will be able to see information on the buttons and workspace by hovering over the object.

Glyphs

Each program in VisiQuest has a graphical representation called a glyph. The glyphs can be brought into the workspace by browsing the Glyph menu from the Toolbar, and dragging the icon to the workspace. There are two main types of glyphs, glyphs that contain regular operators, and those that are procedures.





1) The upper-left section of each glyph gives you access to the glyph's pane. Many glyphs have different settings. These settings are viewed and updated by opening the glyph *pane*. A glyph pane is opened by clicking on the upper-left hand corner of the glyph.
2) The squares on the left side of the glyph represent input locations that are used when connecting glyphs to indicate workflow.
3) The rectangle at the bottom of the glyph can be selected to show the user any output from the glyph. If this rectangle is red, it indicates that there is an error, and the output should be reviewed.
4) The center of each glyph can be pressed to run or stop the glyph, independent of the rest of the workspace.
5) The squares on the right side of the glyph represent output locations that are used when connecting glyphs to indicate workflow.
6) Procedure glyphs are an iconic representation of a set of glyphs. The upper-right corner of Procedure glyphs can be selected to show the workspace represented by the procedure glyph. A procedure glyph can be created by selecting a group of glyphs on the workspace, and using the menu option Control→ Create Procedure.

Changing Glyph Options

Many glyphs have different settings. These settings are viewed and updated by opening the glyph *pane*. A glyph pane is opened by clicking on its upper-left hand corner of the glyph.

EXERCISE:

This exercise will open a glyph pane and change an option in the sample glyph Images (Misc).

1. Place the Images (Misc) glyph on the workspace by holding your right mouse button down and using the menupick Glyphs → Input/Output → Data Files → Images (Misc)
   Click on the workspace. The glyph should now show up on your workspace.

2. Open the pane associated with the Images (Misc) glyph by clicking on its upper-left hand corner. Notice that the direction of the triangle has changed.

3. In the pane, you will see a list of available test images. This glyph is meant to be used for demonstrations and provides users with 20 standard images to play with. For the purposes of this exercise, select ***MRI of Spine*** by selecting the checkbox.

4. Close the pane by clicking ***Close***. Your changes are automatically saved.

Leave the glyph on the workspace for the Connecting Glyphs exercise.

Connecting Glyphs

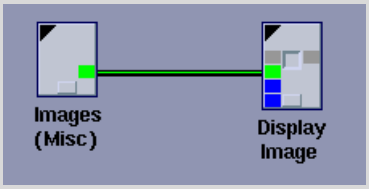
Glyphs must be associated with each other to determine the order in which they will be run. To associate glyphs, you must connect an output square on the right of a glyph with an input square on the left side fo the glyph.
This section of the Exercise will show you how to associate glyphs with each other.

EXERCISE contd:

5. Place the Display Image Glyph on the workspace by using the Menupick Glyphs → Visualization → Non-Interactive Image Display → Display Image
Click on the desktop and place the glyph to the right of Images (Misc).

6. Open the Pane of the Display Image glyph. Observe that the parameter Input File is empty. This will automatically be filled in with the name of the Image File selected in the Image (Misc) Once we connect the glyphs. Close the pane.

7. Connect the output of the Images (misc) glyph to the Display Images glyph by clicking on the output box of the Images (misc) glyph followed by another click on the Yellow input box of the Display Image glyph.



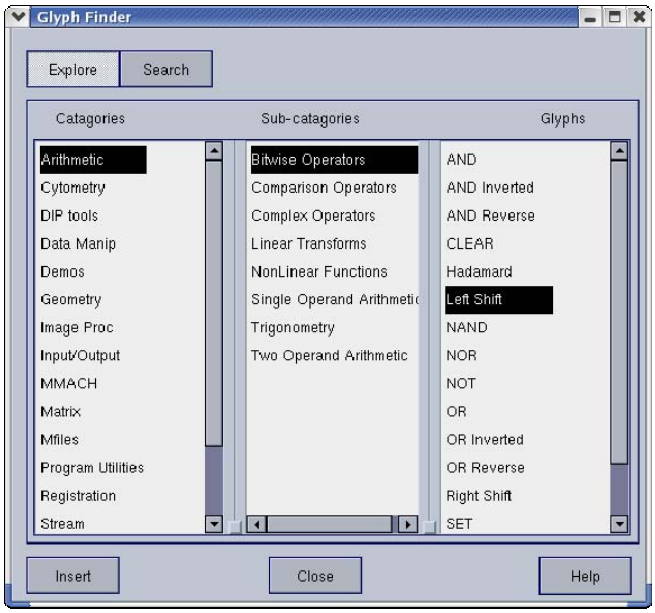
8. To see how the glyphs have now been associated, you can open the pane of the Display Image glyph and see that the input file is now filled in. (Note that the color of the connection is green, indicating that data is available. A yellow connection means that the glyphs must be run before the data is available.)

Finding Glyphs by Keyword

If you do not know where the glyph is located that you would like to use, you should take advantage of the Glyph Finder. The Glyph finder lets you locate and place glyphs on the workspace in one of two ways, by Keyword or by location.

Invoke the Glyph Finder by using the menupick Edit → Find …

This will open a new window. In the Finder tool, the two views are Routines and Finder.



Explore View

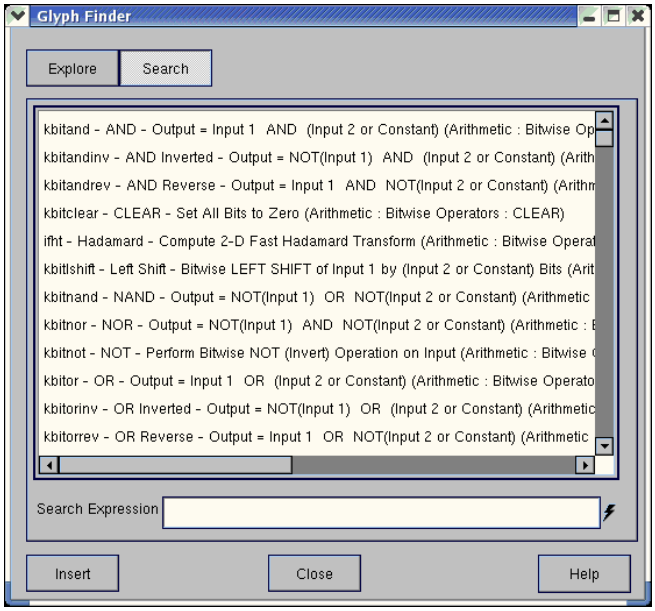
The Explore view is available when the Explore button is highlighted.

This view enables the user to simply walk through the Categories and Subcategories of glyphs. It is an exact duplicate of the Glyphs Menu.

When the Category on the left pane is selected, the middle pane will show all available Subcategories within that category.

When a subcategory in the middle pane is selected, the right pane will show all available glyphs.

Click on a glyph, drag it to the workspace, and it will be placed on the workspace.



Search View

The Search view is available when the Search button is highlighted.

The view enables the user to easily look for glyphs whose short description contains a keyword.

To enter a keyword, hover the mouse over the textbox, enter in a keyword (wildcard is "*"), and hit enter. This will filter the glyphs to show only those which meet the keyword criteria.

Double click on a row which contains the glyph description that meets your needs and the glyph will automatically appear on the workspace.

Running a workspace

VisiQuest can be in one of three main modes: Stop, Run Continuous or Run Once.

There are three buttons on the command bar which are shortcuts for the Stop and Run commands. When you execute a workspace by clicking on the Run icon button, VisiQuest will enter the Run mode. When you stop the execution of the workspace by selecting the Stop option or by clicking on the Stop icon, VisiQuest enters the Stop mode. To toggle between Run Once and Run Continuous, use the Button with a Grey arrow and a black wrapping arrow.

 **Stop Mode.** This is the default behavior of the workspace. In this mode, you can execute each glyph individually by clicking on the glyph's middle square button. In the stop mode, any changes to the glyphs parameters will not rerun the workspace. For the changes to take effect, the workspace must Run.

 **Run Once Mode.** In the run once mode, VisiQuest executes all of the glyphs in the workspace in the order they are connected until they reach a stable condition. Changes made in glyph parameters will not be re-executed.

 **Run Continuous Mode.** In the run continuous mode, glyphs as parameters are changed in glyphs, VisiQuest will re-execute the updated glyphs and all downstream glyphs.

When a glyph is executing, you will notice that the square in the center of the glyph changes to red. After execution, the center of the glyph changes back to grey and the small colored boxes change from yellow to green. These small boxes are the input and output boxes.

EXERCISE contd:

This section will show you how to run and stop your program.

9. Click the Run button and watch your program go! Note that you now have a display window open showing you a picture of a spine.
10. Stop the program by clicking on the Stop button. The picture will disappear.

Congratulations! You have just created your first VisiQuest Visual Program that consists of a network of Glyphs.

Saving, Closing and Restoring a Workspace

It is important to be able to re-use your workspaces. VisiQuest contains all of the standard ways to save your work. You can save your workspace with the same name, or choose Save As to avoid overwriting your previous version.

EXERCISE contd:

This section will save your workspace with a new name, close and reopen the workspace.

11. Save this program to a file by accessing the *Save File* option located in the File menu. (Remember where you save it!)
12. Clear the workspace by clicking the button with a Giant "X" on the toolbar.
13. Now verify that you have saved the workspace properly by accessing the *Open File* option in the File menu.

Opening a Sample Workspace and running it with your own data

The menu Objects contains a large number of sample workspaces. It may be helpful to use one of these samples as a starting point for creating your own application.

EXERCISE:

For this example, we will open the sample workspace Labeling and Shape Analysis, run and stop the workspace, then put in a glyph that allows you to import your own image to be run on the workspace.

1. To open a sample workspace, menupick Objects → General → Image Processing → Labeling and Shape Analysis

If you did not clear your workspace previously, you may be prompted to clear the workspace, Click Yes.

Your workspace now contains a network of glyphs which have been put together to extract individual shapes from an image.

2. Note that there are four glyphs which are named "Display Image", you can change the name of the glyphs by clicking on the name displayed under the glyph. This will also change the name of the Display Image window.
3. Change the name of the Display Image glyph directly attached to the "Images (Misc)" glyph by clicking on the words Display Image. This will bring up a dialog box "Query Dialog for VisiQuest". Enter a new icon name "Original Image". Click "OK"

Note that the name of the glyph is now Original Image.

4. Click the Run button and view the results. Note that the original image is displayed in a window called Original image, and the other images are in a box titled "Display Image".

5. To clear the boxes, stop the workspace by clicking on the second button on your toolbar.

IMPORT YOUR OWN DATA TO RUN ON THE WORKSPACE

Watching the product do interesting things on our sample data is fun, but it is likely that you have images of your own to analyze. To run the template workspace on your own data, simply add your data to the workspace.

6. Place a "User Defined" glyph on the workspace by using menupick
Glyphs → Input/Output → Data Files → User Defined
Click on the workspace

7. First delete the connections to Images (Misc) by clicking on the lines and selecting Delete Connection. Repeat this for both connections to Labeling and Original Image.
8. To create the new connections, click on the right hand output box of the User Defined glyph, then click on the upper left blue box on the Labeling glyph. A new green line will appear and connect the glyphs. Release this and connect the User defined glyph to the Original Image glyph.
9. Point the User Defined glyph to one of your own images by opening the User defined glyph pane (the black triangle in the upper left corner of the glyph.)

10. Click the Input File button in the pane and using the browser, select an image from your local machine.
11. Once you have an image, click OK on the browser window, and click close on the Pane. You can now run the workspace and see the results with your own image!

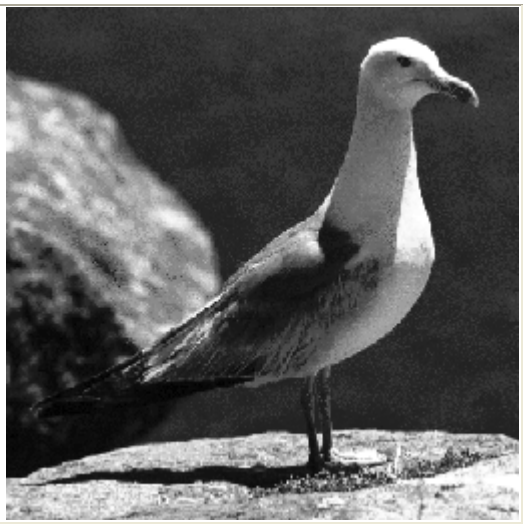
Section 2: Programming a Correlation Workspace using VisiQuest glyphs

Now that you know how the basics of how VisiQuest works, this example will show you how to perform complex analysis of an image by putting together glyphs that already exist in VisiQuest. While this may not directly apply to your work, it is meant to introduce the power of the VisiQuest operators by introducing more advanced workspace methods including while loops and decision trees. This example is taken from Chapter 6, section 5 of our Digital Image Processing (DIP) course. If you are interested in seeing additional examples, you can view the entire course. If your VisiQuest installation is at "c:\vq35", the course can be opened by viewing "C:\vq35\Contrib\dip2001\index.html"

Experiment: Overview Template Matching by Normalized Correlation

One application of correlation is to perform template matching. The idea is to find the position where a best match exists between a small pattern (template) and a set of patterns in a larger image.

In this experiment we will use the gull image as the larger image and a small portion (10x10) around its eye as the pattern image or template.



a) ***b)***

a)larger image; b)zoomed pattern or template (region of the eye)

The correlation of the image with the 10x10 template does not yield the expected result. The maximum points occur at the location width=199 and height=100 and do not correspond to the position where the eye is. Note that the template behaves as a lowpass filter blurring the image. Also, the maximum point will depend on the amount of white (high pixel values) present. You can see the output of the correlation in the image below:

Correlation of the gull image and its eye template

To solve this problem we have to use the **normalized correlation** procedure defined below for the 1D case

$$g_c(x) = \frac{f(x) \circ h(x)}{\sqrt{f^2(x) \circ i}}$$

where **i** is a constant kernel where each pixel has value 1 and with the same size as the correlation pattern. The denominator is known as the normalization factor for correlation.

Now, the maximum of the normalized correlation is at the gull's eye. You can also see that the detected point is not very robust as there are many other high value points that correlates with the template.

a)

b)

a)normalized factor for correlation; b)normalized correlation: maximum at the eye

To help you determine exactly the location of the maximum points use the statistic operators.

Lab: Correlation

Purpose: The purpose of this lab exercise is to experiment with template matching, correlation, and normalized correlation using spatial convolution using the VisiQuest glyphs.

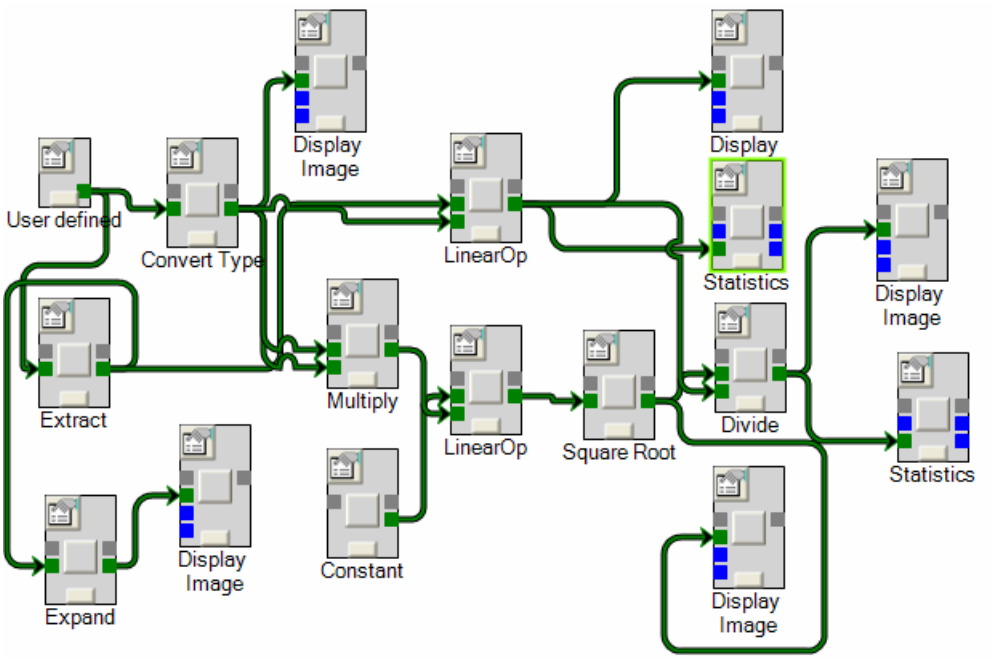
Activities:

1. Read and display the image $DIP/data/gull.kdf. Convert it to floating point using the operator Convert Type.
    1. Glyphs:Input/Output:Data Files:User defined
    2. Glyphs:Visualization:Non-Interactive Display:Display Image
    3. Glyphs:Data Manip:Data Conversion:Convert Type
2. Extract a small (10x10) pattern from the image using the Extract operator. Determine the coordinates of the region you wish to extract by moving the cursor over the pattern in the displayed image. Display the extracted region (first expand it by a factor of 10).
    1. Glyphs:Data Manip:Size & Region Operators:Extract
    2. Glyphs:Data Manip:Size & Region Operators:Expand
    3. Glyphs:Visualization:Non-Interactive Display:Display Image
3. Perform the correlation between the image and the pattern using the LinearOp (Linear Operator) operator. Set its parameter option to perform the "Correlation". Display the result with the Display Image glyph.
    1. Glyphs:Arithmetic:Linear Transforms:LinearOp (Linear Operator)
    2. Glyphs:Visualization:Non-Interactive Display:Display Image
4. Use the Statistics operator to find the coordinates of the maximum point of the resulting image. Open the Statistics pane and select the "maximum", and the width and height coordinates of the maximum.
    1. Glyphs:Data Manip:Analysis & Information:Statistics

    Notice that the maximum does not occur where you would expect it to, which is over the pattern that you extracted.

5. Perform the normalized correlation.
    1. Multiply converted $DIP/data/gull.kdf image for it.
        1. Glyphs:Arithmetic:Two Operand Arithmetic:Multiply
    2. Create a constant kernel of value 1 that is the same size as the pattern (10x10) using the Constant operator. Perform correlation between the multiplyed image and this kernel. Set up the LinearOp (Linear Operator) operator as you did in step 3.
        1. Glyphs:Input/Output:Generate Data:Constant
        2. Glyphs:Arithmetic:Linear Transforms:LinearOp (Linear Operator)
    3. Take the square root of the result of the correlation in step B, using the Square Root operator.
        1. Glyphs:Arithmetic:Single Operand Arithmetic:Square Root
    4. Divide the original correlation image (step 3) by the normalization factor image just created, and display the result.
        1. Glyphs:Arithmetic:Two Operand Arithmetic:Divide
        2. Glyphs:Visualization:Non-Interactive Display:Display Image
    5. Find the maximum of the normalized correlation using the Statistics operator as you did in step 4.
        1. Glyphs:Data Manip:Analysis & Information:Statistics

Your results may resemble this:



Section 3: Software Development within VisiQuest

You have created a complicated workspace, but you have operations that you would like to add to the product. Using our powerful software development tools, Craftsman, Composer, and GUISE, you can create new glyphs in C, C++, PERL or other scripting language.

This section will walk you through creating a new toolbox to hold your new functions, coding a new glyph in C, giving it a user interface, and adding it to the VisiQuest Visual Programming Environment.

Create a new Toolbox

For organization purposes, you need to create your own toolbox to add glyphs to. To create a new toolbox:

1. Open Craftsman
On Unix, type in craftsman in a shell window, in Windows, navigate Start → Program Files → VisiQuest → Craftsman

2. Select Toolbox → Create Toolbox
For the purpose of this example, we are creating a toolbox called DEMO_TOOLBOX

   a. Toolbox Name: DEMO_TOOLBOX
   b. Toolbox Path: //C/ VQ35/DEMO_TOOLBOX/
   c. Toolbox Title: DEMO_TOOLBOX
   d. Update email address if incorrect
   e. Click “Create Toolbox.” Toolbox will be created and fields will be cleared. Click Close.
   f. In craftsman, you will now notice the new toolbox name.

Create a New Glyph & Add it to the Visual Programming Environment

* For illustrative purposes, this glyph will perform the simple operation of reading a data file and printing to stdout the pixel value of a specified coordinate. The name of this object will be kpixel.

You develop software using VisiQuest by creating software objects. When creating an operator software object, you will:

- Design its GUI pane interface.
- Use the code generator to create the user interface code.
- Add the code that implements the operator.
- Compile and install the operator.
- Test the operator using cantata or the CLUI interface.

If you have the Visual Programming Environment open, close it, as it will need to be closed to refresh to see the new Glyph.

1. Open Craftsman

   Select Demo_toolbox on left
   Select object pane on right
   Go to menu Object → Create Object

   NOTE: *You can create software objects of other class types by selecting the desired class type from the Select Classtype pulldown menu. The default, and the choice we will use is C. By selecting Khoros Routines → Khoros Scripts, you can choose to work in PERL, csh, or sh.*

2. Enter the following info to create your glyph (You will need to check the checkboxes to activate the windows):
   Toolbox Name demo_toolbox
   Object Name **kpixel**
   Binary Name **kpixel**
   Icon Name **Pixel**
   Author **your name**
   Email Address **your email**
   Short Description **Reads a pixel value**
   Generated Language **C**
   Install in VisiQuest? **TRUE**
   Category **demo_toolbox** (you will need to type this in)
   Sub-category **Information** (you will need to type this in)
   Continuous Run Driver **No**
   Strict Arguments **True**

3. Click “Create ANSI C/C++ Routine”. This will generate a status window with messages.
4. Once the operation has completed, Click Close. Note that the object has been created.
5. Now you can edit the object
   a. With the kpixel object selected in the craftsman subform, select Edit Object (Composer) in the Object pulldown menu. This executes composer.
   b. Create the Glyph Pane
      1. Click on the + under “uis” to reveal the kpixel.pane UIS file.
      2. Double-click on the kpixel.pane _le to execute guise.

   In a few moments GUISE will create two windows. The first is the guise subform where you select the appropriate commands in the object pane. The second is the pane of the object in the edit mode. Note the gridded background in the pane object. Also note that the pane object, by default, has already one Input and one Output file parameter, and Run, Help, and Close buttons.

EDIT THE GUI

As the kpixel operator requires a single file input parameter, delete the output file parameter.

- Select the Output parameter in the pane object by clicking on it with the left mouse button. Its four corners are highlighted indicating the selection made.
- In the guise subform, select Delete in the Edit pulldown menu. This will delete the Output parameter from the object pane.

o Creating five integer parameters (The object must have five integer parameters that correspond to the width, height, depth, time, and element coordinates.

- Select Integer in the Simple Variables pulldown menu in the guise sub-form. This brings up an integer parameter to the pane object.
- Now change this input parameter to reflect the desired title, variable, and description.
  - Click with the middle mouse button on the Integer parameter in the object pane. This shows the Integer Selection subform for editing the parameter.
  - Set the following parameters in the Integer Selection subform.
    - Parameter value
    - Bounds Value >= 0
    - Title Width
    - Variable w
    - Desc width pixel coordinate
  - Close the Integer Selection subform.
- Create the height parameter by duplicating the width integer parameter.
  - With the Width selected, select Edit →Copy in the guise pulldown menu.
  - You can move this field around by holding down the left mouse button.
  - Click on the duplicated parameter with the middle mouse button and change the parameters.
    - Parameter Value
    - Title Height
    - Variable h
    - Desc height pixel coordinate
  - Create the last three parameters, Depth, Time, and Element, by duplicating and then editing each one. Use the following titles, variables, and description for each parameter.

Title	Variable	Desc
Depth	d	depth pixel coordinate
Time	t	time pixel coordinate
Element	e	element pixel coordinate

c. Save the UIS _le by selecting Save from the guise File menu.
d. Exit guise by selecting Quit from the File menu.

Source code generation

1. On the Composer subform, select Options →Commands
2. On the Commands subform, select Generate Code from the Make menu.
3. Close the Commands subform.

Editing the source C files

This object will read the input file and will pick the pixel value at the specified pane coordinate parameters.

1. Click on the src directory in the Composer list. Three source code files appear in the list.

   To start the editor, double click on the kpixel.c file. This displays the kpixel.c source file in your preferred text editor. Note that a good deal of the file has already been filled in. Based on the GUI created, the variables are defined as :

```
clui_info->w_int, /* w input parameter */
clui_info->h_int, /* h input parameter */
clui_info->d_int, /* d input parameter */
```

```
clui_info->t_int, /* t input parameter */
clui_info->e_int, /* e input parameter */
```

2. For simplification purposes, you can copy and paste the entire text to replace the existing text, or review each section and note the code to be added:

```
/*
  * Khoros: $Id$
  */

#if !defined(__lint) && !defined(__norcsid)
static char rcsid[] = "Khoros: $Id$";

#endif

/*
 * Copyright (C) 1993 - 2002, Khoral, Inc., ("KI").
 * All rights reserved.  See $BOOTSTRAP/repos/license/License
 * or run klicense.
 */

/* >>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<<
   >>>>
   >>>>  Driver code for kpixel
   >>>>
   >>>>  Private:
   >>>>  run_kpixel()
   >>>>  kpixel_extra_usage_additions();
   >>>>         kpixel_extra_free_args();
   >>>>   Static:
   >>>>   Public:
   >>>>
   >>>>>>>>>>>>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<<<<<<<<< */


#include "kpixel.h"
kobject src_obj;
double *data = NULL;
/*-------------------------------------------------------------
|
|  Routine Name: run_kpixel() - Reads a pixel Value
|
|       Purpose: program driver code for kpixel
|
|       Returns: TRUE (1) on success, FALSE (0) on failure
|    Written By: ACCUSOFT.COM+NicoleR
|          Date: Dec 07, 2004
|
-------------------------------------------------------------*/



int run_kpixel(void)

{
if ((src_obj = kpds_open_input_object(clui_info->i_file)) == KOBJECT_INVALID)
{

 kerror("meigen","main", "Cannot open input object.");
```

```
return FALSE;

}


   if (!kpds_set_attributes(src_obj,
KPDS_VALUE_DATA_TYPE, KDOUBLE,
KPDS_VALUE_OFFSET,
clui_info->w_int, /* w input parameter */
clui_info->h_int, /* h input parameter */
clui_info->d_int, /* d input parameter */
clui_info->t_int, /* t input parameter */
clui_info->e_int, /* e input parameter */
NULL))
{
kerror("main","kpixel",
"unable to set source object value attr.");
return(FALSE);
}
data = (double *)kpds_get_data(src_obj,
KPDS_VALUE_POINT,
data);
kprintf("Pixel[%d,%d,%d,%d,%d]=%g\n",
clui_info->w_int,
clui_info->h_int,
clui_info->d_int,
clui_info->t_int,
clui_info->e_int,
data[0]);

   return TRUE;
}

/*-------------------------------------------------------------
|
|  Routine Name: kpixel_usage_additions
|
|       Purpose: Prints usage additions in kpixel_usage routine
|
|    Written By: ACCUSOFT.COM+NicoleR
|          Date: Dec 07, 2004
|
-------------------------------------------------------------*/
void kpixel_extra_usage_additions(void)
{
   /*-- Put Your Code Here --*/
   return;
}

/*-------------------------------------------------------------
|
|  Routine Name: kpixel_free_args
|
|       Purpose: Frees CLUI struct allocated in kpixel_get_args()
|
|         Input: status      - The exit status for the program.
|         client_data - Client data needed to by the exit handler
|    Written By: ACCUSOFT.COM+NicoleR
```

```
|          Date: Dec 07, 2004
kfree_and_NULL(data);
kpds_close_object(src_obj);
|
------------------------------------------------------------*/
/* ARGSUSED */
void kpixel_extra_free_args(
   kexit_status status,
   kaddr        client_data)
{
   kfree_and_NULL(data);
kpds_close_object(src_obj);

   return;
}
```

3. Once you have copied the above into the kpixel.c file, save the file and exit the editor.

Compiling your code

4. Select the Commands button from the Options menu in composer .
5. In Commands, choose Kmake Install from the Make pulldown menu.
6. If everything works, your object has been compiled and installed successfully. There may be several warning messages, but you can verify that the object has been installed if the line before the last in the Commands subform is the install command or a message saying that kpixel is installed.
7. Close the Commands window, quit composer and craftsman.

Use your new glyph in a Visual Program

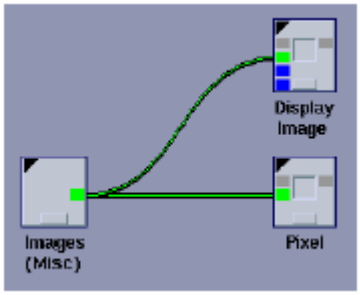
1. If you have the VPE running, exit the program and invoke it again so the program will recognize the new toolbox and glyph.

We will now create our own workspace and program. That uses a combination of existing glyphs and the new one we created:

Put the following glyphs on your workspace (from under Glyph menu):

a. Input/Output →Data File→ Images (Misc) glyph
b. Demo_toolbox→ Information→ Pixels glyph. Note that this is the new glyph you have just created using composer.
c. Visualization → Non-Interactive Image Display→ Display Image glyph

2. Connect them together:



.

3. Select the Spanish Sea Gull image in the Images (Misc) glyph.
4. Run the Pixel glyph and verify its output. The first pixel of the gull image has value 41.
5. Test the Pixel glyph in other pixel coordinates by opening the glyph pane and changing the options. Compare the output values with the Display Image glyph.