SCIRun Haptic Display for Scientific Visualization

Lisa J K Durbeck, Nicholas J Macias, David M Weinstein, Chris R Johnson, John M Hollerbach

Introduction

The overall goal of this research is to enhance scientific visualization by the use of haptic feedback. We have chosen to approach this goal by incorporating a haptic interface into an existing full-featured scientific visualization tool.

People who use scientific visualizations are generally trying to analyze or solve a scientific problem. The visualizations are usually animations or interactive graphics that scientists create for themselves. The visualizations are intermediate representations intended to represent the problem or its solution graphically (1, 2). Ideally these visualizations are accurate, information-rich, interactive, and illustrative of key features of the phenomenon or data set. They should make the problem (or its solution) easier to understand. Toward this goal we augmented existing graphical visualizations with force feedback (3), as well as true 3-D interaction, in ways that highlight key features such as flow lines. The haptic interface we have developed allows the user to form a high-level view of his data more quickly and accurately.

Scientific visualization research has been underway at the University of Utah for much of this decade (4). One outcome of that research is a problem-solving environment for scientific computing called SCIRun (5,6). For this project, we interfaced SCIRun to a haptic device, and we then used SCIRun to create two displays, one graphic and one haptic, which operate together on a common visualization. The work described here uses the system to interact with vector fields, 3-D volumes in which every point in the volume has a vector associated with it. Examples of vectors are gravity, pressure, force, current, and velocity as well as scalar gradients.

Users of our new system can simultaneously see and feel a vector field. Haptic feedback is displayed on a SensAble PHANToM Classic 1.5 (7). The graphics are displayed on an SGI Octane. The user has a haptic display registered to a graphic display. She directs both displays by moving the PHANToM endpoint through the haptic representation of the data volume. The haptic display presents each vector as a force corresponding to the vector's magnitude and direction. The graphic display presents a subset of the vector field as lit directional line segments (8) or as the traditional arrow glyphs.

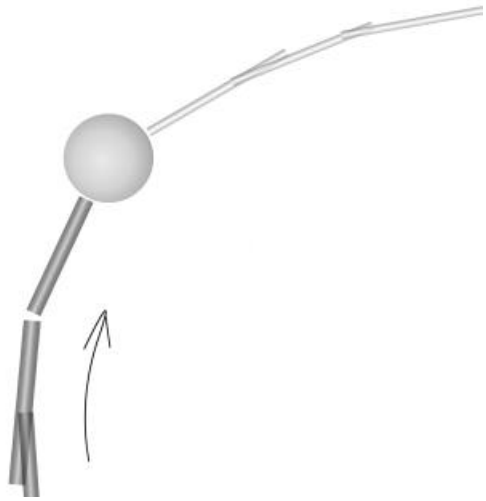

Fig 1. Illustration of user interaction with flow line. The display sweeps out the path to and from the current endpoint position. The pencilled arrow indicates the direction of travel. The sphere represents the current PHANToM endpoint position; the fatter, darker lit lines are the path already taken by the user; the thin, lighter lines are the flow line leading from the current position.

This haptic/graphic display is useful for displaying flow fields, vector fields such as fluid flow models for airplane wings in which the vectors tend to align into strong directional paths (9). The haptics feel as if you put your fingertip into a river: the vectors act upon your fingertip, drawing it in the

same direction as the local flow field. If the user does not apply any force, the forces displayed onto the PHANToM endpoint naturally draw his finger along the flow lines. This allows the user to trace his finger along the various flow lines within the vector field. The user can also move his finger onto a new path, and again the endpoint begins to follow this new path. This interface allows the vector field to act upon the PHANToM endpoint in a similar manner as a traditional visualization technique called seed point insertion or particle advection (1). The graphical display reinforces the haptic display by showing the endpoint moving along the flow line and by showing the part of the flow line that lies ahead of the current position. The user receives haptic and visual feedback which helps him stay on the path. Figure 1 shows a still image from a user's interaction with a flow line and Figure 2 shows an image of several traced flow lines composited over time.

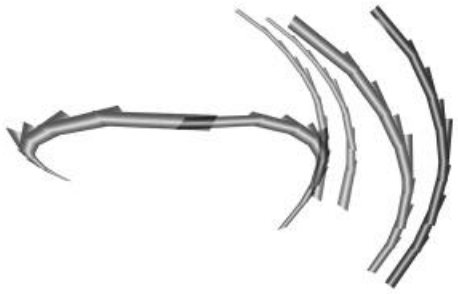

Fig. 2 Illustration of 3 separate flow line traces composited into one image. As the user interrogates the volume, the display forms an image of the flow lines within the field.

System Architecture

Figure 3 shows a high-level view of the software and hardware used for this system.

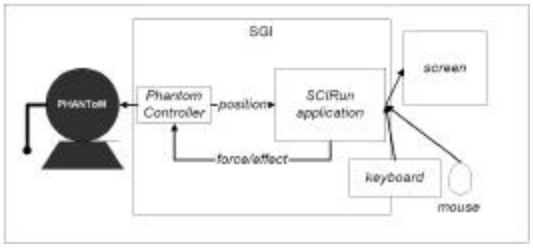

Fig. 3 High level view of system hardware & software

The software runs on an SGI, 02 or better, with a PCI or ISA slot for the PHANToM card.[1] SCIRun and the PHANToM controller are two separate client-server applications (10,11,12,13) which communicate via sockets. The PHANToM controller acts as the server and SCIRun acts as the client. The Appendix contains a full listing of our PHANToM control loop, written using a small subset of the Ghost SDK. As the size of this listing illustrates, the PHANToM controller is minimal: all the data and computations are handled by the program within SCIRun. Figure 4 shows a dataflow diagram representing the program we wrote within the SCIRun programming environment. The program consists of two loops, the haptic display loop and the graphic display loop. Within the haptic display loop, the program receives the latest PHANToM endpoint position, calculates a force based on that position, and sends out a force to the PHANToM controller. Within the graphic display loop, the program receives the latest PHANToM endpoint position, redraws the endpoint in the graphic display, and recalculates and redraws the vector field display.

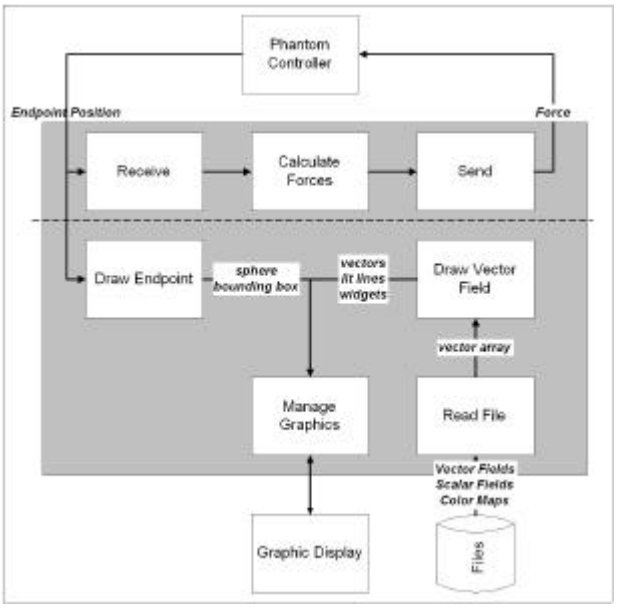

Fig. 4 Dataflow diagram of our software. Each labelled boxes represents a distinct component of the software

[1] Typically we use an SGI Octane with dual 195 MHz R10000 processors, 128 MB memory, and an SGI MXI graphics card.

system. All components within the shaded box run within SCIRun; the rest run outside of the SCIRun runtime environment. The components above the line are particular to the haptic display, while those below are particular to the graphical display.

Future Work

The user can change aspects of the graphic and haptic displays at runtime. One interesting visualization technique we would like to explore is user-defined transfer functions. Rather than mapping data vectors linearly to force vectors, we could map them based on a nonlinear or piecewise linear function defined by the user. Figure 5a) shows an example of a function that could be used to re-map one dimension of the vector field such as magnitude. Vectors with magnitude X along the x-axis are mapped to forces with magnitude Y. The transfer function in Figure 5a) amplifies the effect of small vectors and could be used to fine-tune the display for examining small vectors. The transfer function graphed in Figure 5b) effectively weeds out all vectors in the field except for those within the specified range. If the full range of force magnitudes is used within this smaller data range, then the effect is a haptic "zoom". Note that in 1 dimension, these transfer functions look like traditional force profiles (14) but are data-dependent, not position-dependent.

We also anticipate making use of multiprocessor scheduling on the Octane in order to maintain interactive rates for large visualizations or complex force calculations.

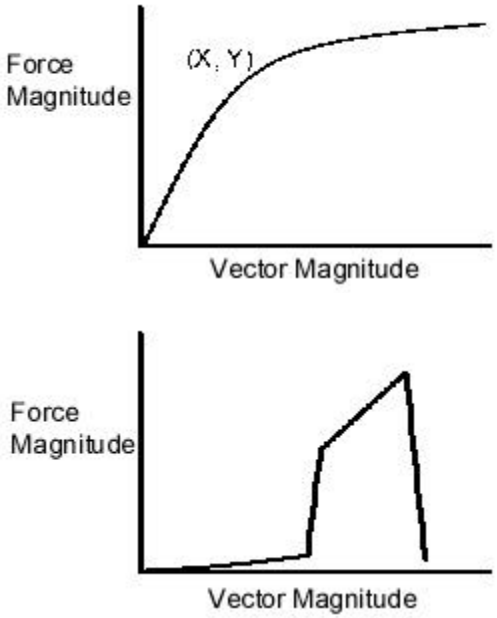

Fig 5a (top), 5b (bottom). Simple nonmonotonic transfer functions which map from data vector magnitude to force magnitude.

Acknowledgments

This work was supported in part by the Department of Defense through a DURIP equipment grant, the National Science Foundation, and the Department of Energy. Furthermore, the authors appreciate access to facilities that are part of the SGI-Utah Visual Supercomputing Center.

Bibliography

1 R. S. Gallagher, ed. Computer Visualization: Graphics Techniques for Scientific and Engineering Analysis. CRC Press, Boca Raton, 1995.

2 W. Schroeder, K. Martin, & B. Lorensen. The Visualization Toolkit, 2nd edition. Prentice Hall, New York, 1998.

3 J.P. Fritz & K.E. Barner. *Haptic Scientific Visualization*. Proceedings of the First PHANToM User's Group Workshop, 1996.

4 see http://www.cs.utah.edu/~sci/publications

5 S.G. Parker, D.M. Weinstein, & C.R. Johnson. *The SCIRun computational steering software system*. Modern Software Tools in Scientific Computing, E. Arge, A.M. Bruaset, & H.P.Langtangen, eds. Birkhuaser Press, 1997: 1-44.

6 S.G. Parker & C.R. Johnson. *SCIRun: A Scientific Programming Environment for Computational Steering*. Supercomputing '95, 1995.

7 T.H. Massie. *Design of a Three Degree of Freedom Force-Reflecting Haptic Interface*. SB Thesis, Department of Electrical Engineering and Computer Science, M.I.T. May 1993

8 M. Zockler, D. Stalling, H.C. Hege. *Interactive Visualization of 3D-Vector Fields Using Illuminated Stream Lines*. Visualization '96, 1996: pp 107-113.

9 J. Helman & L. Hesselink. *Visualizing Vector Field Topology in Fluid Flows*. IEEE Computer Graphics and Applications, May 1991.

10 W.R. Mark, S.C. Randolph, M. Finch, J.M. Van Verth, R.M. Taylor II. *Adding Force Feedback to Graphics Systems: Issues and Solutions*. Siggraph '96 Computer Graphics Proceedings, 1996: 447-452.

11 W. Plesniak & J. Underkoffler. *SPI Haptics Library*. Proceedings of the First PHANToM User's Group Workshop, 1996.

12 S. Vedula & D. Baraff. *Force Feedback in Interactive Dynamic Simulation*. Proceedings of the First PHANToM User's Group Workshop, 1996.

13 S.W. Davidson. *A Haptic Process Architecture using the PHANToM as an I/O Device in a Virtual Electronics Trainer*. Proceedings of the First PHANToM User's Group Workshop, 1996.

14 A.J. Kelley & S.E. Salcudean. *The Development of a Force Feedback Mouse and its Integration into a Graphical User Interface*. In Proceedings of the International Mechanical Engineering Congress and Exposition. Chicago USA 1994. DSC-Vol. 55-1: 287-294

Appendix

Program listing for PHANToM Controller

```
// PHANToM_controller.cpp - main loop for PHANToM Controller program
// derived from hello.cpp provided by SensAble
#include <stdlib.h>
#include <gstBasic.h>
#include <gstSphere.h>
#include <gstPHANToM.h>
#include <gstSeparator.h>
#include <gstScene.h>
#include "ljdForceInput.h"
#include "server.c"

main() {
gstScene *scene = new gstScene;
gstSeparator *root = new gstSeparator;
gstPHANToM *PHANToM = new gstPHANToM("PHANToM.ini");

root->addChild(PHANToM);
scene->setRoot(root);

ljdForceInput * forces = new ljdForceInput; // force from SCIRun
PHANToM->setEffect(forces);
PHANToM->startEffect();

printf("put the PHANToM in neutral position and hit return...\n");
getchar();

scene->startServoLoop();
gstPoint pos;  // holds current PHANToM position
double u,v,w;
```

```
sock_init();
while (!scene->getDoneServoLoop()) {

   pos = PHANToM->getPosition_WC();
   write_triple(pos.x(), pos.y(), pos.z());   // send position to client, SCIRun
   if (read_triple(&u,&v,&w) == 0) {   // read resulting force from SCIRun
     // copy to readable location
     scirun_force = gstVector(u,v,w);
     // the next time calcEffectForce() happens, it will see this new force.
   }
   else {
     printf("client scirun has shut down.\n");
     sock_deinit();
    }
}
PHANToM->stopEffect(); // quit my force input
}

-------------------------------------------------------------
// ljdForceInput.h derived directly from Ghost SDK CalcEffect.h
#include <math.h>
#include <gstBasic.h>
#include <gstEffect.h>

gstVector scirun_force; // set by main loop

class  ljdForceInput:public gstEffect
{
public:
       ljdForceInput():gstEffect(){} //Constructor
       ~ljdForceInput() {}  // Destructor

       virtual gstVector     calcEffectForce(void *PHANToMN)
           {
                if (PHANToMN);  // To remove compiler warning
                if (!active) return gstVector(0.0, 0.0, 0.0); // check first
                gstPoint pos;
                double xc, yc, zc;  // force vector components
                xc = scirun_force.x(); yc = scirun_force.y(); zc = scirun_force.z();
                return gstVector(xc, yc, zc);
           }
};
```