

### TABLE OF CONTENTS

- Editor's Note ..... 1
- Recent Releases ..... 1
- Visualizing Vector Fields Using LIC ..... 3
- Distributed Graphs in VTK Using PBGL ..... 5
- Kitware Quality Software Process ..... 8
- Fast Diffeomorphic Registration in ITK ..... 11
- In Progress ..... 13
- Kitware News ..... 14

### RECENT RELEASES

#### CDASH 1.2

CDash the open-source, web-based software testing server has had its second major release since May. CDash aggregates, analyzes and displays the results of software testing processes submitted from clients located around the world, to convey the state of a software system and continually improve its quality.

All of Kitware's software projects have been transitioned to CDash and the Dart server is no longer being used. The main CDash server is now hosting 12+ projects with accumulated data spanning 8 years. This accumulated data storage only uses about 60GB of disk space.

The 1.2 release fixes more than 30 bugs and adds more than 20 new features including:

- Support of PostGreSQL and MySQL as a backend database
- The Site page now lists the users who claimed a site
- The Site page shows the time spent to build and submit
- The configuration column now displays CMake configuration warnings, errors and time
- Variations in the number of errors, warnings, tests from the previous build are shown on the main dashboard page
- Users can recover passwords
- CDash can send summary email for nightly groups
- CDash sends an email when expected builds were not submitted the previous day
- Clicking on a graph landmark links to the results for that build
- Project administrators can send email to all site maintainers
- You can add a user or group of users without a file upload
- CDash now supports ViewVC, WebSVN and Loggerhead for source management
- CDash now supports multiple repositories per project
- Build notes are now compressed in the database

### EDITOR'S NOTE

The Kitware Software Developer's Quarterly Newsletter contains articles related to the development of Kitware projects in addition to a myriad of software updates, news and other content relevant to the open source community. In this issue, Utkarsh Ayachit illustrates the LIC implementations in VTKEdge, VTK and ParaView which will allow users to exploit the processing power provided by modern GPUs when utilizing these toolkits. Drs. Jeff Baumes and Doug Gregor introduce the ability to create and process distributed graph structures in VTK utilizing the Parallel Boost Graph Library. Bill Hoffman provides a brief tutorial for submitting test results to CDash utilizing the CTest component of CMake and explains how those test results can be used to create quality software. And Dr. Tom Vercauteren presents the first diffeomorphic image registration scheme introduced in ITK, diffeomorphic demons.

The Kitware Source is just one of a suite of products and services that Kitware offers to assist developers in getting the most out of its open-source products. Each project's website contains links to free resources including mailing lists, documentation, FAQs and Wikis. In addition, Kitware supports its open-source projects with technical books, user's guides, consulting services, support contracts and training courses. For more information on Kitware's suite of products and services, please visit our website at [www.kitware.com](http://www.kitware.com)



Site	Build Name	Upload			Configure			Build			Test			Build Time
		Files	Size	Errors	Warnings	Errors	Warnings	Time	Errors	Fail	Pass	Time		
kitware.com	Linux gcc 4.3.0 (TKL3.0)	9.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2008-09-15 04:21:28 EDT	
kitware.org	MacOSX 10.4 (TKL3.0)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2008-09-15 09:36:00 EDT	
kitware.org	MacOSX 10.4 (TKL3.0)	9.0	0.0	0.0	11.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2008-09-15 09:36:00 EDT	
kitware.org	MacOSX 10.4 (TKL3.0)	1.0	0.0	0.0	3.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2008-09-14 09:36:00 EDT	
kitware.org	MacOSX 10.4 (TKL3.0)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2008-09-15 09:36:00 EDT	

The new CDash layout includes errors and warnings for the configure stage as well as variations with the previous build

## Personal CDash

If you would like to use CDash for your software project, Kitware is now hosting a CDash server that allows anyone to quickly setup a new project. This exciting, free service allows projects to be administered entirely from the CDash web interface. To start submitting to your own personal CDash, register at <http://my.cdash.org>, and create your project.

## CMAKE 2.6.2

CMake has released version 2.6.2. This release mainly addresses compatibility issues and bug fixes for the 2.6 branch of CMake. Combined with 2.6.1, 2.6.2 addresses over 170 issues with 2.6.0. Some of the highlights are as follows:

- Builds Second Life Viewer on Windows, Linux and Mac OSX.
- Many small changes to find modules so that CMake can find installed software more reliably.
- The Mac OSX Framework creation support is complete
- A `-trace` option was added to CMake to enable easier debugging of CMake projects.
- Several documentation fixes.
- Added a new tree-view to the Qt cmake-gui.
- Allow `CMakeImportBuildSettings` to not match tools if `CMAKE_OVERRIDE_COMPILER_MISMATCH` is set.
- Bug fix in NSIS CPack where `CPACK_NSIS_MODIFY_PATH` did modify the path

For a full listing of the changes in 2.6.2 visit [cmake.org](http://cmake.org).

## ITK 3.8

ITK 3.8 was released on July 30. The major changes in this release include:

- Introduction of the Diffeomorphic Demons deformable registration family of classes contributed by Tom Vercauteren et. al, in their Insight Journal paper: <http://hdl.handle.net/1926/510>
- ImageIO classes to read Phillips/PAR/REC files and Bruker2D files, contributed by Don Bigler et. al, in their paper to the Insight Journal: <http://hdl.handle.net/1926/1381>
- Slice-by-Slice filter contributed by Gaetan Lehmann in his paper to the Insight Journal: <http://hdl.handle.net/1926/368>
- FFT & Helper classes for Complex image manipulations contributed by Simon Warfield in his Insight Journal paper: <http://hdl.handle.net/1926/326>
- ImageIO classes for CUB file format, contributed by Pablo Burstein et. al, in their Insight Journal paper: <http://hdl.handle.net/1926/303>
- Added support for the GCC 4.3 compiler
- Added support for Visual Studio 9 - 2008 (both 32- and 64-bit).
- Many bugs fixed thanks to the contributions of volunteers in the "Adopt-a-Bug" program. Thanks to all of them! [http://www.itk.org/Wiki/ITK\\_Adopt\\_a\\_Bug\\_Program](http://www.itk.org/Wiki/ITK_Adopt_a_Bug_Program)

More details about this release can be found by visiting [itk.org/Wiki](http://itk.org/Wiki) and searching for "ITK Release 3.8".

## PARAVIEW 3.4

The ParaView 3.4 release is now available for download from the ParaView website ([paraview.org](http://paraview.org)). It is also available through CVS; the tag is `ParaView-3-4`. Since the 3.2 release, we have been focusing on usability; 3.4 contains many improvements and bug fixes.

The major changes since 3.4 are:

- VTK and ParaView are now licensed under the BSD license as opposed to modified BSD.
- The multi-block and AMR support was improved significantly. Almost all filters, spreadsheet view and charts now support these datasets.
- The selection capabilities of ParaView were significantly improved. For details, see the July 2008 *Kitware Source*.

### New features

- Support for plotting multiple point/cell values over time.
- Support for picking end-points of line widgets using 'p'.
- Support for scene exporters. Supported formats are X3D (binary and ascii), VRML 2 and POV (Persistence of Vision Raytracer).
- Temporal statistics filter that can be used to find average, min, max and standard deviation of arrays over time.
- 2D views / slice representation for volumes (`vtkImageData`).
- Box and sphere widgets for slicing and clipping.
- Cube axes used to show scale of a dataset.
- Multiple selections using ctrl (command on Mac). When performing selection after the first time, hold ctrl to add to the existing selection.

### Improvements to Existing Features

- Plug-in improvements. (See the plug-in page on the wiki for more information.)
- Added an option to disable off-screen rendering while saving images. This option should be turned on if ParaView crashes when saving images.
- Changed the default width of the scalar bar; also tweaked the way scalar bar scaling works.
- Improved partial array support (when an array exists only in some of the blocks of a multi-block dataset).
- Fixed many widgets for floating point numbers, enabling higher precision.

### Reader and Writer Improvements

- Readers for MFIX and Fluent.
- Support for loading file series from the command line as well as from the recent file menu.
- Support for saving multi-block polygonal datasets as a collection of STL or PLY files.
- Support for file series of XML based VTK files.
- Updated XDMF reader to XDMF 2.

For a complete list of features and fixes, see [paraview.org/Wiki](http://paraview.org/Wiki) and search for "ParaView Release Notes".

## VTKEDGE ALPHA RELEASE

In October 2008, Kitware enabled public read access to the VTKEdge source code repository. VTKEdge is still in the alpha stage, with continued development expected over the next few months. Highlights of the initial release include:

- Advanced GPU volume ray casting
- Illuminated lines painter
- Surface LIC painter
- GPU accelerated filters for 2D LIC
- Feldkamp reconstruction (CPU and GPU versions)
- GPU accelerated array processing
- Powerful pixel / voxel editing widgets

Please visit the VTKEdge web site at [vtkedge.org](http://vtkedge.org) for additional information and download instructions.

## VISUALIZING VECTOR FIELDS USING LINE INTEGRAL CONVOLUTION

Line Integral Convolution (LIC) was introduced in 1993 [1]. Since then it has become one of the most powerful dense texture-based techniques for visualizing vector fields. These vector fields are obtained from various domains for example, from computational fluid dynamics. The reason for LIC's popularity is the fact that it provides a good understanding about the nature of the vector field over the entire domain without being dependent on factors such as a seed placement (as in the case of streamlines).

In spite of its tremendous usability, commercial application support for LIC is almost non-existent. One possible reason may be its computational complexity; line integrals are computed from each pixel in the output image. Also, until recently there were no robust techniques for application of LIC to non-uniformly sampled datasets.

The previous edition of the *Kitware Source* announced VTKEdge. VTKEdge is a new toolkit which allows Kitware to make the results of some of our Small Business Innovation Research (SBIR) and Small Business Technology Transfer Research (STTR) endeavors available to the open source community. With the introduction of VTKEdge, VTK (and a host of applications based on VTK, such as ParaView) now support LIC. The implementation tries to address performance issues by exploiting the processing power provided by modern GPUs. VTKEdge provides three implementations for applying LIC on 2D image data (uniform rectilinear grid) slices, 2D slices of a curvilinear grid, and vector fields over any arbitrary surface.

This article illustrates the LIC implementations and briefly describes the underlying implementation of these LIC extensions to VTK.

### LIC ON UNIFORM GRIDS

The original LIC paper by Cabral and Leedom [1] describes a method that takes in two datasets: a 2D vector field sampled uniformly over a rectilinear grid and a noise texture with the same resolution as the vector field. In cases where the vector field can be defined as point attributes of a 2D `vtkImageData`, VTKEdge provides the `vtkKWEImageDataLIC2D` filter. This filter can be used as follows:

```
vtkRenderWindow* renWin = vtkRenderWindow::New();
renWin->Render();
// test if all required OpenGL extensions are
// supported by the rendering context.
if (vtkKWEImageDataLIC2D::IsSupported(renWin)
    == false)
{
    cerr << "Required OpenGL extensions missing.";
    return;
}
vtkKWEImageDataLIC2D* licer =
    vtkKWEImageDataLIC2D::New();
// set the 2D vector field source as the input.
// This is required.
licer->SetInputConnection(0,
    image2Dsource->GetOutputPort());
// set the 2D noise texture input. This is optional.
// When not specified, the licer creates a internal
// vtkImageNoiseSource to generate the noise
// texture.
licer->SetInputConnection(1,
    noise2Dsource->GetOutputPort());
// specify the distance of the sample points
```

```
// as a fraction of the cell length.
licer->SetStepSize(0.3);
// specify the number of samples in each
// direction of the vector field.
licer->SetNumberOfSteps(40);
// provides for supersampling the result.
licer->SetMagnification(2);

// Now the result can be rendered using an image
// viewer.
vtkImageViewer2* viewer =
    vtkImageViewer2::New();
viewer->SetRenderWindow(renWin);
viewer->SetInputConnection(licer->GetOutputPort());
viewer->Render();
```

As is clear from the above code snippet, `vtkKWEImageDataLIC2D` is simply a two input image-data to image-data filter which produces the LIC image as the output. `vtkKWEImageDataLIC2D::IsSupported(vtkRenderWindow*)` is a static method that tests if all required OpenGL extensions are supported. This is essential since the implementation relies on recent additions to OpenGL such as frame buffer objects and shader programs.



Figure 1: Line Integral Convolution from a vector field sampled over a uniform rectilinear grid

### Implementation

The underlying implementation is surprisingly straightforward. We upload the input vector field as well as the noise image as textures to the GPU. A frame buffer the size of the expected output image is created (taking into consideration the input vector field dimensions and the magnification requested by the user). Using the vector field accumulating the noise value over all the passes, the texture coordinate space used to lookup the noise texture is warped in each rendering pass. The number of passes depends on the number of samples and equals  $(2 * \text{number of samples} + 1)$ .

### LIC ON CURVILINEAR GRIDS

Forsell extended the original LIC algorithm to vector fields defined over curvilinear grids [2]. Given a transformation to convert the vectors over the curvilinear grid and those over a uniform grid, one can simply use the standard LIC algorithm to generate the LIC image and then map the pixels back to the original grid. As described in [2], this transformation is given by the inverse Jacobian computed at each data loca-

tion. This is supported by the `vtkKWEStructuredGridLIC2D` filter. This is a two-input-two-output filter that takes in a 2D structured grid and noise texture and produces the original 2D structured grid with texture coordinates and the LIC image as output. The texture coordinates make it possible to render the output structured grid as geometry and apply the LIC image as a texture. The following code snippet illustrates the use of this filter:

```

vtkKWEStructuredGridLIC2D* licer =
    vtkKWEStructuredGridLIC2D::New();
// set the 2D vector field source as the input.
// This is required.
licer->SetInputConnection(0,
    grid2Dsource->GetOutputPort());
// set the 2D noise texture input. This is optional.
// When not specified, the licer creates a internal
// vtkImageNoiseSource to generate the
// noise texture.
licer->SetInputConnection(1,
    noise2Dsource->GetOutputPort());
// specify the distance of the sample points
// as a fraction of the cell length.
licer->SetStepSize(0.3);
// specify the number of samples in each
// direction of the vector field.
licer->SetNumberOfSteps(40);
// provides for supersampling the result.
licer->SetMagnification(2);
// create a texture using the LIC output
vtkTexture* tex = vtkTexture::New();
tex->SetInputConnection(
    licer->GetOutputPort(1));

// for rendering we need polydata, hence we extract
// the surface of the grid.
vtkDataSetSurfaceFilter* d2s =
    vtkDataSetSurfaceFilter::New();
d2s->SetInputConnection(licer->GetOutputPort(0));

// setup the mapper/actor
vtkPolyDataMapper* mapper =
    vtkPolyDataMapper::New();
mapper->SetInputConnection(d2s);
vtkActor* actor = vtkActor::New();
actor->SetMapper(mapper);
actor->SetTexture(tex); // assign texture to use

renderer->AddActor(actor);
renWin->Render();

```

As is the case with `vtkKWEImageDataLIC2D`, `vtkKWEStructuredGridLIC2D::IsSupported(vtkRenderWindow*)` is provided to test whether or not all required OpenGL extensions are supported.

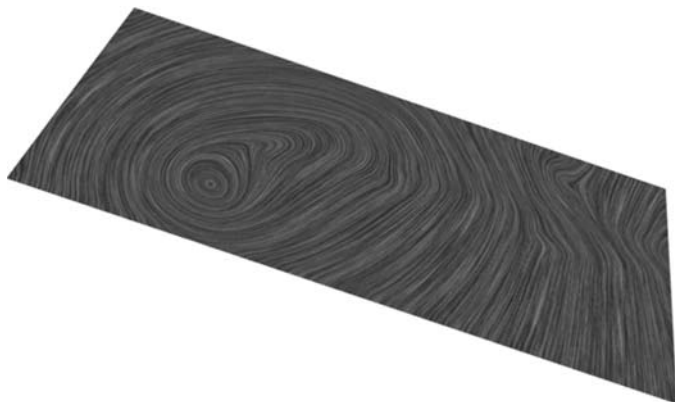


Figure 2: Line Integral Convolution applied over a curvilinear dataset

## Implementation

`vtkKWEStructuredGridLIC2D` uses a first rendering pass to compute the inverse Jacobian for each grid location and then transforms the vector field. This transformed vector field is used as the input vector field for the shader code used by `vtkKWEImageDataLIC2D` to generate the LIC over a uniform grid.

## LIC ON SURFACES

LIC on curvilinear grids makes it possible to generate LIC images on several real-world CFD datasets, yet its applicability remains limited. Several processing operations – such as clipping, cutting, thresholding, iso-surfacing – result in unstructured outputs when applied to curvilinear or uniform grids. Until recently, the only way to visualize vector flows on such arbitrary surfaces was through the use of streamlines or glyphs. Our implementation is based on work by Laramee, et. al [3]. Refer to [3] for the details about this algorithm. The major steps in our implementation can be summarized as follows:

- Surface polygons are rendered with the vector field passed as texture coordinates to the GPU.
- A vertex shader does the standard polygon model-view-projection transformation on the geometry while simultaneously projecting the vectors-to-image plane. Thus, we have valid vectors at all pixels where the geometry is projected and null vectors everywhere else.
- Next, the conventional LIC on is applied on these transformed vectors in the image space.
- The resultant image is blended with the surface rendering to retain surface shading, scalar coloring, etc.

Since the algorithm is dependent upon the camera position it must be rerun every time the camera changes angles. Hence, it is implemented as a PAINTER. In other words, surface LIC is a rendering style through which geometry with vector data can be rendered. The following code snippet illustrates its use:

```

vtkKWESurfaceLICPainter* licPainter =
    vtkKWESurfaceLICPainter::New();
// select the vectors to use for generating the LIC
licPainter->SetInputArrayToProcess(
    vtkDataObject::FIELD_ASSOCIATION_POINTS,
    "Velocity");
// specify the number of samples in each
// direction of the vector field.
licPainter->SetNumberOfSteps(40);
// step size in pixels.
licPainter->SetStepSize(0.5);
// specify LIC intensity in the result rendering
// obtained by blending the LIC image with the
// surface rendering.
licPainter->SetLICIntensity(0.8);
// enable/disable the painter
licPainter->EnableOn();

// insert the painter into the default painter
// chain used by the mapper.
vtkPainterPolyDataMapper* mapper =
    vtkPainterPolyDataMapper::New();
licPainter->SetDelegatePainter(
    mapper->GetPainter());
mapper->SetPainter(licPainter);

// set the data set input to the mapper.
mapper->SetInputConnection(
    dataSource->GetOutputPort());
vtkActor* actor = vtkActor::New();
actor->SetMapper(mapper);
renderer->AddActor(actor);
renWin->Render();

```

vtkKWESurfaceLICPainter also supports the static `vtkKWESurfaceLICPainter::IsSupported(vtkRenderWindow*)` method of testing for required OpenGL extensions.

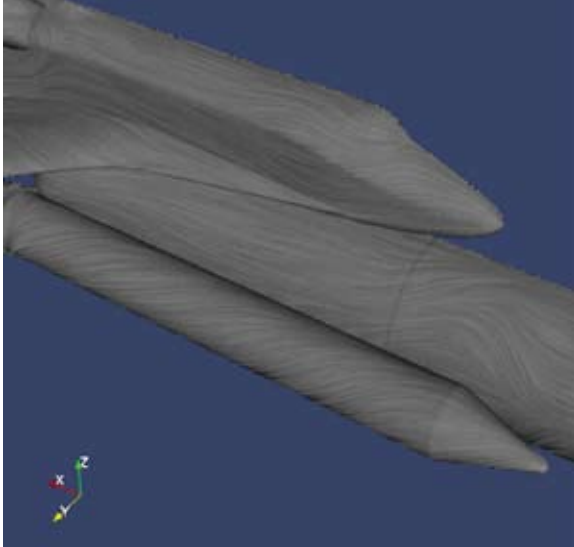


Figure 3: Line Integral Convolution applied to a vector field over a surface

## LIC IN PARAVIEW

ParaView 3.0 and later provide an elaborate plug-in API for writing extensions to bring advanced functionality into ParaView. VTKEdge uses the plug-in API to provide plug-ins for LIC on 2D images and surfaces so that they can be directly used in ParaView. To enable compiling of the ParaView plug-in, one must turn on the `BUILD_PARAVIEW_PLUGINS` CMake variable. Once the plug-in is loaded in ParaView, `vtkKWEImageDataLIC2D` filter is available under the filters menu as “LIC on 2D Image”. The `vtkKWESurfaceLICPainter` is now available as a new representation style (along with Outline, Points, Wireframe, and Surface) as “Surface LIC”. The parameters on the painter can be controlled using the display tab which includes choosing the vector field, number of steps, LIC intensity and so on.

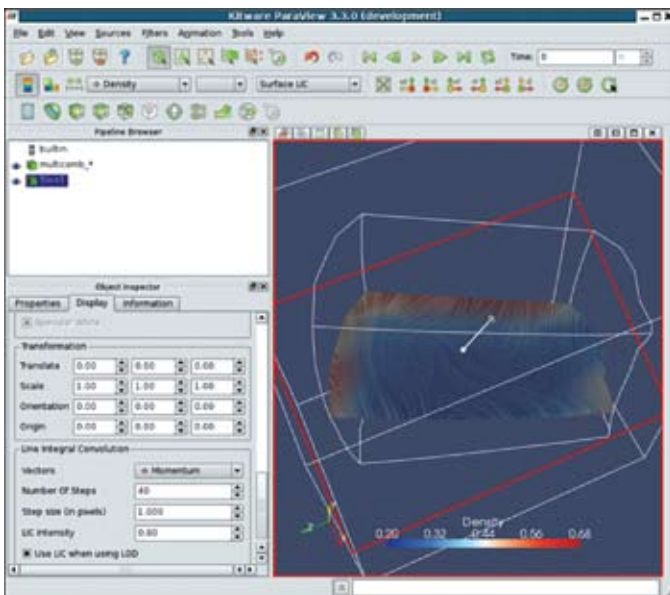


Figure 4: Surface LIC in ParaView using the plug-in for VTKEdge. The display tab shows the LIC parameters.

## REFERENCES

1. B. Cabral, C. Leedom, “Imaging Vector Fields Using Line Integral Convolution”, SIGGRAPH 93 Conference Proceedings, pages 263–270, ACM SIGGRAPH, 1993.
2. Forssell, L. K., “Visualizing flow over curvilinear grid surfaces using line integral convolution”, Visualization 94 Conference Proceedings, pages 240-247, IEEE Computer Society, 1994.
3. Laramée R. S., Jobard B., Hauser H., “Image space based visualization of unsteady flow on surfaces”, Visualization 03 Conference Proceedings, pages 131-138, IEEE Computer Society, 2003.

## ACKNOWLEDGEMENTS

This work was supported in part by the Department of Defense / Army Research Laboratory contract number W911NF-06-C-0179.



*Utkarsh Ayachit is an R&D Engineer in Kitware’s Clifton Park, NY office. Mr. Ayachit is the project lead for GPU-based extensions to VTK ([vtk.org](http://vtk.org)), and is also a developer on the ParaView project ([paraview.org](http://paraview.org)).*

## DISTRIBUTED GRAPHS IN VTK USING PBGL

This article will introduce you to creating and processing distributed graph structures in VTK. There are already many algorithms in VTK that work on distributed data. The ParaView application makes it possible to use this functionality to easily process very large data sets. Unlike images and other structured geometry used in VTK, real-world graphs such as social networks often have very high connectivity between most entities within the graph. The notion of “six degrees of separation”, which claims that any two people on Earth can be connected through a chain of about six acquaintances, highlights this connectivity. This greatly complicates the process of splitting a graph into various pieces, and care must be taken to avoid unacceptable communication overhead. Utilizing the Parallel Boost Graph Library (PBGL), VTK hides much of this complexity, making writing and using parallel graph algorithms a much simpler process.

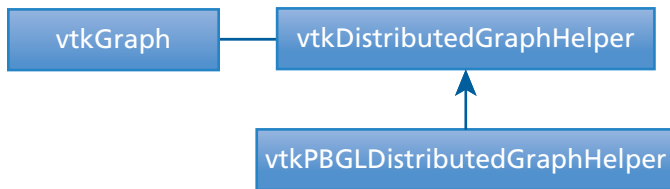
PBGL is a C++ library which uses generic programming techniques to provide a strong infrastructure for processing large networks (i.e. graphs) on multiple processors. It builds on the Boost C++ Libraries and in particular on the Boost Graph Library. The Boost Graph Library contains numerous algorithms for processing graphs to extract information such as shortest distances, clusters, connected components, and minimum spanning trees. It also provides versatile iterators for performing custom operations on graphs. PBGL expands on these capabilities to provide users of the Boost Graph Library with simple tools to parallelize graph algorithms and make them scalable to large datasets.

## HOW PBGL INTEGRATES WITH VTK

Previously, VTK has supported a non-distributed graph structure `vtkGraph`, which is integrated with the Boost Graph Library. Instances of this class may be readily adapted to the Boost Graph Library concepts by including a single header file (`vtkBoostGraphAdaptor.h`). Some VTK algorithm implementations such as `vtkBoostBreadthFirstSearch` make use of Boost algorithms instead of re-implementing them.

Through the efforts of the Open Systems Laboratory at Indiana University and the Titan project led by Sandia National Laboratories, PBGL functionality is now also available in VTK. While non-distributed graphs were adapted to Boost using a light wrapper, PBGL integration required some extensions of the `vtkGraph` structure itself to distribute its vertices and edges across multiple processes (often, on multiple nodes in a compute cluster).

### Class Diagram



This diagram shows a few of the classes that were added to VTK to allow distributed graphs to function. `vtkGraph` now has a member variable called `DistributedHelper`, which is initially empty but may be set to an instance of `vtkDistributedGraphHelper`. `vtkPBGLDistributedGraphHelper` is a concrete subclass of `vtkDistributedGraphHelper` which is only built when the `VTK_USE_PARALLEL_BGL` CMake flag is on. The helper hides many of the details about using MPI's to coordinate information among multiple processors. When a `vtkGraph` has a valid distributed helper, it may be adapted to PBGL's graph concepts by including a single header file (`vtkPBGLGraphHelper.h`), as was the case for the non-distributed BGL adapters.

### COMPILING VTK WITH PBGL SUPPORT

To obtain VTK with PBGL functionality, you will need the following on your system:

1. Boost, version 1.36.0 or newer ([www.boost.org](http://www.boost.org))
2. An MPI library (e.g. MPICH2 at [www.mcs.anl.gov/research/projects/mpich2](http://www.mcs.anl.gov/research/projects/mpich2) or OpenMPI at [www.open-mpi.org](http://www.open-mpi.org))
3. The latest development version of the Parallel Boost Graph Library ([www.osl.iu.edu/research/pbgl](http://www.osl.iu.edu/research/pbgl))
4. The latest development version of VTK from CVS ([www.vtk.org](http://www.vtk.org))

VTK must be compiled with `VTK_USE_64BIT_IDS`, `VTK_USE_BOOST`, `VTK_USE_PARALLEL`, `VTK_USE_MPI`, `VTK_USE_PARALLEL_BGL`. For detailed build instructions, see [kitware.com/InfovisWiki](http://kitware.com/InfovisWiki) and search for "Dev:Building Software" then click on Parallel Boost Graph Library in the Contents area or simply scroll down to the Parallel Boost Graph Library section.

### USING DISTRIBUTED GRAPHS

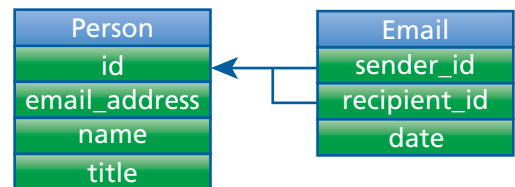
To create a distributed graph, use one of the following filters:

- `vtkPBGLGraphSQLReader` – Construct a distributed graph from a table in an SQL database.
- `vtkPBGLRandomGraphSource` – Construct an Erd s-Rényi random distributed graph.
- `vtkPBGLRMATGraphSource` – Construct a RMAT random distributed graph.

The following algorithms are currently implemented for processing distributed graphs. All of them make use of existing PBGL algorithms.

- `vtkPBGLBreadthFirstSearch` – Perform a network search from a starting vertex.
- `vtkPBGLConnectedComponents` – Find connected regions of a graph.
- `vtkPBGLMinimumSpanningTree` – Find the minimum set of edges to connect a graph.
- `vtkPBGLShortestPaths` – Determine the shortest paths from a starting vertex.
- `vtkPBGLVertexColoring` – Give vertices "colors" so no two adjacent vertices share the same color.

We will illustrate the capabilities of a distributed graph pipeline with an example. Suppose you have a large amount of information in a SQL database relating objects to one another in some way. For this example we will assume that you have a database with the following schema, where people are connected to other people through email messages:



From this information, you want to see a high-level view of how people with different jobs communicate with each other (e.g., how much do managers and lawyers communicate). In addition, we want to do this for a dataset with millions of emails in a reasonable amount of time and visualize the resulting communication graph. We can do all this with VTK and PBGL. We start by using `vtkPBGLGraphSQLReader` to read the data into a distributed graph structure.

```

vtkSQLDatabase* db = vtkSQLDatabase::CreateFromURL(
    "mysql://user@localhost/mydb");
db->Open("password");
vtkPBGLGraphSQLReader* reader =
    vtkPBGLGraphSQLReader::New();
reader->SetDatabase(database);
reader->SetVertexTable("Person");
reader->SetEdgeTable("Email");
reader->SetVertexIdField("id");
reader->SetSourceField("sender_id");
reader->SetTargetField("recipient_id");
  
```

When the reader executes, each process will execute queries to extract portions of the vertex and edge data tables in order to create a distributed graph. Every vertex in the graph has exactly one process that owns the vertex and stores all of its properties. A vertex, in turn, owns all of the edges originating from it. A configurable hash function determines the process where each vertex will live. The vertices and edges will automatically go to their assigned process during the communication synchronization phases of the algorithm. The resulting `vtkGraph` contains a record of all of the e-mail traffic in the database (the edges of the graph) from one person to another (the vertices of the graph), distributed across all of the processes.

Next, we wish to condense this graph into one where each vertex represents groups of people with the same title. To accomplish this, we use `vtkPBGLCollapseGraph`, which takes a distributed graph as input, along with a vertex data array used to group vertices which share values.

```
vtkPBGLCollapseGraph* collapse =
    vtkPBGLCollapseGraph::New();
collapse->SetInputArrayToProcess(0, 0, 0,
    vtkDataObject::FIELD_ASSOCIATION_VERTICES,
    "title");
collapse->SetInputConnection(
    reader->GetOutputPort());
```

The graph produced by this algorithm will still contain edges for all of the emails in the original dataset, but it drastically reduces the number of vertices, since now there will be only one vertex for each job title. In order to also condense the graph edges, we will group multiple edges between two vertices as a single edge with the number of original edges stored in an edge "weight" data array. For example, if there were 12,034 emails from managers to vice presidents, we want to replace these links with a single graph edge with the value 12,034 stored in a data array named "weight". The algorithm `vtkPBGLCollapseParallelEdges` accomplishes this.

```
vtkPBGLCollapseParallelEdges* collapseParallel =
    vtkPBGLCollapseParallelEdges::New();
collapseParallel->SetInputConnection(
    collapse->GetOutputPort());
```

Now that we have a much smaller summary graph, we can collect the graph into one process in order to connect it to a visualization pipeline. `vtkPBGLCollectGraph` takes a distributed graph as input and collects it into the full non-distributed graph on one or more processes.

```
vtkPBGLCollectGraph* collect =
    vtkPBGLCollectGraph::New();
collect->SetInputConnection(
    collapseParallel->GetOutputPort());
```

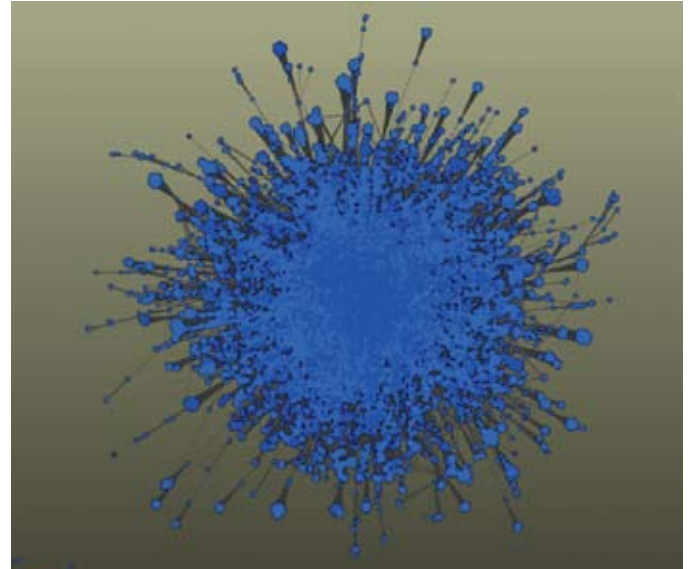
In order to execute this pipeline, or any parallel VTK pipeline, we must take care to request the appropriate portion of data for each process. Assuming "total" holds the number of MPI processes, and "rank" holds the current process index, the following will correctly setup the pipeline to execute in parallel:

```
collect->UpdateInformation();
vtkStreamingDemandDrivenPipeline* exec =
    vtkStreamingDemandDrivenPipeline::SafeDownCast(
        collect->GetExecutive());
exec->SetUpdateNumberOfPieces(
    exec->GetOutputInformation(0), total);
exec->SetUpdatePiece(
    exec->GetOutputInformation(0), rank);
collect->Update();
```

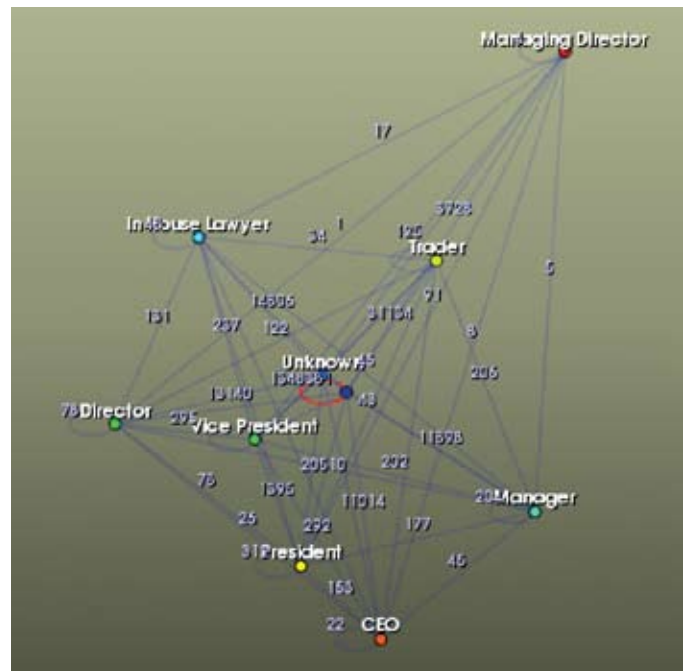
You may then take the output of the collect algorithm and plug it into a visualization pipeline. The simplest way to do this is to use `vtkGraphLayoutView`, which wraps graph layout, vertex glyphing, coloring, and labeling into one class. See `VTK/Parallel/Testing/Cxx/TestPBGLPipeline.cxx` for the full source code of this example.

The images below are the results of executing this pipeline on a database of emails made public by the Federal Energy

Regulatory Commission during the aftermath of the Enron scandal (see <http://www.isi.edu/~adibi/Enron/Enron.htm> for a raw version of this dataset, although additional processing is needed to get the schema described above). The database contains over 75 thousand unique email addresses and over 2 million emails. The following is the full graph which took over fifteen minutes to layout, render, etc. on a single process. Labels and curved edges were disabled to allow for full visualization. The graph also does not provide a lot of meaningful information.



The following is the result of running the pipeline with four processes on a single quad-core Linux computer. The processing took about two minutes, and since the processed graph is so small, graph layout and rendering are almost instantaneous.



## FUTURE WORK

While this project is still in its initial phases, it is quickly becoming a useful tool capable of seamlessly processing and visualizing very large graphs. We are currently working to make the algorithms more efficient, and to make all PBGL features fully available to VTK. Watch for new information

visualization plug-ins for ParaView which will wrap these operations in an easy-to-use interface. For more information or to become involved with this project, please contact Dr. Jeff Baumes at [jeff.baumes@kitware.com](mailto:jeff.baumes@kitware.com).

## ACKNOWLEDGEMENTS

This work was supported in part by Sandia National Labs through contract number #664044.



*Dr. Doug Gregor is Assistant Director of the Open Systems Laboratory at Indiana University, whose research involves generic programming, parallel programming, and large-scale graph analysis. He is the lead developer of the Parallel Boost Graph Library.*



*Dr. Jeff Baumes is an R&D Engineer in Kitware's Clifton Park, NY office. Dr. Baumes is the main developer for Titan, whose goal is to bring information visualization and informatics capabilities to VTK.*

as an administrator and edit your project, you will see a page like the one below (the project edit page for CMake itself). At the bottom of that page you can see the option to Download CTestConfig.cmake. If you click on that link you will get a copy of CTestConfig.cmake for the project with the correct Nightly start time for the project.

The screenshot shows a web interface for editing a CMake project. It includes a 'Documentation URL' field with 'www.cmake.org', a 'Logo' field with a 'Browse...' button, and a 'Current logo' section displaying a 3D pyramid logo. Below are several configuration options: 'Public Dashboard' (checked), 'Coverage Threshold' (70), 'Nightly Start Time' (21:00:00 EDT), 'Google Analytics Tracker' (UA-701656-6), 'Enable test timing' (checked), 'Test time standard deviation' (4.0), 'Test time standard deviation threshold' (2.0), and several email notification options (checked for 'Email broken submission', 'Email build missing', and 'Email test timing changed'). At the bottom, there is a 'Download CTestConfig:' link pointing to 'CTestConfig.php' and two buttons: 'Update Project' and 'Delete Project'.

The CTestConfig.cmake file for the CMake project looks like this:

```
set(CTEST_PROJECT_NAME "CMake")
set(CTEST_NIGHTLY_START_TIME "21:00:00 EDT")
set(CTEST_DROP_METHOD "http")
set(CTEST_DROP_SITE "www.cdash.org")
set(CTEST_DROP_LOCATION
    "/CDash/submit.php?project=CMake")
set(CTEST_DROP_SITE_CDASH TRUE)
```

## EXPERIMENTAL DASHBOARDS

The Experimental submission is the easiest to understand and create. CTest simply performs a build and runs the tests for the current copy of the software on the disk with no query of the version control system. The only thing that is required is the CTestConfig.cmake file that points to the project name, drop site and location. To run an experimental dashboard for a project all you need is an existing build tree for the project. If you run the following command from the command line while in the build:

```
ctest -D Experimental
```

CTest will run CMake, build the project using the native build tool, and run any tests added with the `add_test` command. CTest will compile the results into XML files and send them to the CDash server specified in the `CTEST_DROP_SITE` variable of the CTestConfig.cmake file. This is a great place to start to make sure the process is setup and working. After CTest is done, the results should be immediately available on the CDash dashboard for the project in the Experimental section. Experimental dashboards are useful not only for testing the setup of the project for CDash, but for sharing build and test issues with other developers. For example, if you had some test that was failing, but you did not want to commit the code into your version control software, you could do an experimental submission and discuss the results with other developers that don't have that access to the machine or don't have that version of the software.

## KITWARE QUALITY SOFTWARE PROCESS

Over the past ten years Kitware has been refining and developing tools, in collaboration with many others, that help to produce quality software. Those tools include the CMake/CTest/CPack build system, and CDash, the web-based software testing server. The process relies on the methods from test-driven development (TDD) and extreme programming. This article will cover how to use the CTest component of CMake to submit test results to CDash and how to use those results to create quality software. It will begin by describing the various types of submissions that CDash supports and how to view them, then cover the details of how to setup clients to submit the test results. Once a developer has used and understands how to effectively use CDash on a project, developing software any other way is like coding with one hand tied behind your back.

Although CTest and CDash do not require the use of CMake as the build system, CMake does make using these programs much easier. This article will assume that you have a project that has CMakeLists.txt files with at least one `add_test` command that runs a test. It will also assume that you have a CDash server set up and working. If you do not, you can create a dashboard hosted by Kitware. See [cdash.org](http://cdash.org) for more information about Kitware's free CDash hosting options. CDash supports three main types of submissions: Experimental, Nightly, and Continuous, as well as the display of dynamic analysis and code coverage results. We'll discuss how to enable a CMake-based project for CDash and how to interpret and create all five of those dashboard submissions in this article as well.

## CDASH ENABLING OF A PROJECT

Before a project can use CTest to submit results to CDash, the project must have a file called CTestConfig.cmake in the top level source tree. The entire file CTestCoinfig.cmake can be generated automatically by CDash. If you log into CDash

## NIGHTLY DASHBOARDS

The Nightly submission shows the state of the software at a given point in time for the repository. This is done by picking an arbitrary point in time during the 24 hour day that will be the “nightly” time for the project. When CTest creates a Nightly submission for CDash it will use this time and the version control system to get the correct copy of the software to test. Each CTest client can be run at any point during the day to create a Nightly dashboard entry, and CTest will pick the closest Nightly time that has already passed, as version control systems are not yet able to get software from the future. This will assure that all Nightly submissions are building and testing the exact same version of the software by using that point in time. The time setting is stored in the CTestConfig.cmake in the CTEST\_NIGHTLY\_START\_TIME variable. In the example shown for CMake’s dashboard the nightly time is 21:00:00 EDT.

The Nightly dashboard is very important for a project, and provides a status on the system giving advanced users of the project “a release every day”. If you look at the dashboard and all systems are building and passing tests, then it is okay to assume that the software is safe to update to that point in time. If instead the dashboard is full of test errors and build problems, you would know that it is not a good time to do an update until things look better.

To create a nightly dashboard for a project, follow the same steps that you took for creating an experimental dashboard but change the command line for CTest to look like this:

```
ctest -D Nightly
```

In practice, this is not sufficient enough for a reliable nightly submission from a client because it requires that you have an existing build tree. If anything goes wrong during the checkout of the software or the CMake run, the dashboard gets no submission at all. To get around these issues, scripting was added to CTest in order to make the process more robust. The following script will drive a nightly dashboard for CMake:

```
set (CTEST_SITE           "MySite")
set (CTEST_BUILD_NAME     "WindowsXP")
set (CTEST_NOTES_FILES    "${CTEST_SCRIPT_DIRECTORY}/${CTEST_SCRIPT_NAME}")
set (CTEST_SOURCE_DIRECTORY "c:/CMake")
set (CTEST_BINARY_DIRECTORY "c:/CMakeNightlyBuild")
set (CTEST_UPDATE_COMMAND "svn")
set (CTEST_CMAKE_GENERATOR "Unix Makefiles")
set (CTEST_PROJECT_NAME  "CMake")

ctest_empty_binary_directory (
  ${CTEST_BINARY_DIRECTORY})
# this is the initial cache to use for the binary tree,
# be careful to escape any quotes inside of this string
# if you use it.
file(WRITE
  "${CTEST_BINARY_DIRECTORY}/CMakeCache.txt"
  "")
CMAKE_RUN_LONG_TESTS:BOOL=TRUE
)

ctest_start (Nightly)
ctest_update (SOURCE
  "${CTEST_SOURCE_DIRECTORY}")
ctest_configure (BUILD
  "${CTEST_BINARY_DIRECTORY}")
ctest_build (BUILD "${CTEST_BINARY_DIRECTORY}")
ctest_test (BUILD "${CTEST_BINARY_DIRECTORY}")
ctest_submit ()
```

If the above script were saved in a file called BillsMachineNightly.cmake, the script is run like this:

```
ctest -S BillsMachineNightly.cmake
```

That command should be put into a cron job or Windows Scheduled Task. There is more information on how to do that please visit: [cmake.org/Wiki](http://cmake.org/Wiki) and search for “CMake Scripting Of CTest”. Then select the hyperlink “Setting Up Cron/Scheduler” from the contents menu in the article.

The script will remove the entire binary tree each night and submit results to the dashboard even if the CMake step fails because of a bad commit that breaks the build system.

## CONTINUOUS DASHBOARDS

Continuous dashboards are used to monitor the version control system as developers commit to the dashboard. Clients are setup to monitor the version control system (VCS) for changes. As code is committed to the VCS, CTest will perform an incremental update, build and test. The results are then sent to CDash. If errors or test failures are detected, CDash will send email to the developers that are responsible for that commit. The goal is to avoid errors during the work day so that the nightly builds are error free. This is best summarized in the figure below which shows the Kitware Quality Software Process. Although the graphic refers to all types of dashboard submissions, the Continuous submission is a much tighter loop with information being sent to the developer, often times, within minutes of the commit.



Again the continuous submission can be run with a simple command in the build tree:

```
ctest -D Experimental
```

However, a CTest script is a much better way of automating this task:

```
set (CTEST_SITE           "MySite")
set (CTEST_BUILD_NAME     "WindowsXP")
set (CTEST_NOTES_FILES    "${CTEST_SCRIPT_DIRECTORY}/${CTEST_SCRIPT_NAME}")
set (CTEST_SOURCE_DIRECTORY "c:/CMake")
set (CTEST_BINARY_DIRECTORY "c:/CMakeNightlyBuild")
set (CTEST_UPDATE_COMMAND "svn")
set (CTEST_CMAKE_GENERATOR "Unix Makefiles")
set (CTEST_PROJECT_NAME  "CMake")

ctest_empty_binary_directory (
  ${CTEST_BINARY_DIRECTORY})
```

```

while (${CTEST_ELAPSED_TIME} LESS 36000)
  set (START_TIME ${CTEST_ELAPSED_TIME})
  ctest_start (Continuous)
  ctest_update (SOURCE)
  "${CTEST_SOURCE_DIRECTORY}" RETURN_VALUE res)
# force a build if this is the first run
# and the build dir is empty
if(NOT EXISTS
"${CTEST_BINARY_DIRECTORY}/CMakeCache.txt")
set(res 1)
message("First time build")
# this is the initial cache to use for the binary
# tree, be careful to escape any quotes inside
# of this string if you use it.
file(WRITE "${CTEST_BINARY_DIRECTORY}/CMakeCache.
txt" "
CMAKE_RUN_LONG_TESTS:BOOL=TRUE
")
endif(NOT EXISTS
"${CTEST_BINARY_DIRECTORY}/CMakeCache.txt")
if("${res}" GREATER "0")
message("Found changes run build")
ctest_configure (BUILD
"${CTEST_BINARY_DIRECTORY}")
ctest_read_custom_files
("${CTEST_BINARY_DIRECTORY}")
ctest_build (BUILD "${CTEST_BINARY_DIRECTORY}")
ctest_test (BUILD "${CTEST_BINARY_DIRECTORY}")
ctest_submit ()
endif("${res}" GREATER "0")
# loop no faster than once every 5 minutes
message("wait for 5 minutes")
ctest_sleep( ${START_TIME} 300 ${CTEST_ELAPSED_
TIME})
endwhile(${CTEST_ELAPSED_TIME} LESS 36000)

```

This script should be run as a cron job or Windows Scheduled Task by following the same steps that you took for creating a Nightly build. However, this build should be run during the daytime when developers are active on the project.

## DYNAMIC ANALYSIS

CDash and CTest also support dynamic analysis. CTest currently supports valgrind, purify, and bounds checker for dynamic analysis. Dynamic analysis tools are very powerful tools that check for memory leaks and array bounds checks. These tools are often able to find hard to track problems. The downside of these tools is that they use a great deal of computing resources. As a result, developers tend to only run them when they suspect a problem. However, problems like these are tricky and often do not manifest themselves as soon as the bad code is committed. If one or more dynamic analysis tools are run on a project on a regular basis, it becomes much easier to pinpoint the code that started the problem and fix it early on.

Again, the dynamic analysis dashboard can be created with the simple CTest command, however this time the type of dashboard has the word MemCheck appended to it. One of the following can be done:

```

ctest -D ExperimentalMemCheck
ctest -D NightlyMemCheck
ctest -D ContinuousMemCheck

```

Of course a CTest script is better suited for this task as well. The script should replace the `ctest_test` with a `ctest_memcheck` command:

```

ctest_memcheck(BUILD "${CTEST_BINARY_DIRECTORY}")

```

This will run all tests with memory checking enabled. You also need to tell CTest how to find the memory checking tool, command line options, and an optional suppression file:

```

set(CTEST_MEMORYCHECK_COMMAND
"/home /valgrind-3.2.3-install/bin/valgrind")
set(CTEST_MEMORYCHECK_SUPPRESSIONS_FILE
"${CTEST_SCRIPT_DIRECTORY}/CMake.supp")
set(CTEST_MEMORYCHECK_COMMAND_OPTIONS
"--gen-suppressions=all --trace-children=yes
-q --leak-check=yes --show-reachable=yes
--workaround-gcc296-bugs=yes
--num-callers=100 -v")

```

## COVERAGE ANALYSIS

CTest supports coverage information build types. For coverage analysis, CTest supports GCC's open source gcov and the commercial Bullseye coverage tool ([www.bullseye.com](http://www.bullseye.com)). These tools will show developers which parts of the code are run during the testing of the software. This is a critical part of software testing, and has the motto "If it is not tested, it doesn't work". What this means is that when a user discovers a bug in software you can almost always look at the current testing coverage and find out that part of the code was not being run. The July 2008 *Kitware Source* covered how to setup a Bullseye dashboard script. To use gcov, you can use a script much like the ones we have been using, but you add a `ctest_coverage` command into the build script. You also need to change the flags used to compile your project. The best way is to set the environment variables `CXXFLAGS`, `CFLAGS`, and `LDLFLAGS` in the CTest script before the project is configured with CMake with something like the following:

```

SET (ENV{CXXFLAGS}
"-g -O0 -fprofile-arcs -ftest-coverage")
SET (ENV{CFLAGS}
"-g -O0 -fprofile-arcs -ftest-coverage")
SET (ENV{LDLFLAGS} "-fprofile-arcs -ftest-coverage")

```

This will cause GCC to inject coverage information into the software as it is built. Then when gcov is run it will create coverage files that CTest can find to compile the information about the code coverage during software testing.

## SUMMARY

The software testing process that is possible with the CTest/ CMake/CDash system enables software to be developed more quickly and with greater confidence that bugs will be found early. As a warning, developers will become addicted to this style of development and demand that all projects use this style of testing. Once it is setup, the process is lightweight and does not interfere with the developer, but rather allows the developer to work faster and with greater confidence that each commit of source code works as expected on all supported platforms for the project.

## ACKNOWLEDGEMENTS

This work has been partially funded by the National Library of Medicine ITK project, and by the NA-MIC National Center for Biomedical Computing, NIH Roadmap for Medical Research, Grant # U54 EB005149.



*Bill Hoffman is currently Vice President and CTO for Kitware, Inc. He is a founder of Kitware, a lead architect of the CMake cross-platform build system and is involved in the development of the Kitware Quality Software Process and CDash, the software testing server.*

## FAST DFFEOMORPHIC REGISTRATION IN ITK

Image registration, the process of determining the spatial transformation that maps points from one image to the corresponding points in the second image, is one of the two main topics addressed by ITK. As such, a wealth of image registration algorithms have been implemented in ITK. The flexibility of ITK's design allows the user to build a complete registration pipeline by picking the right building blocks among a large set of spatial transformation types, image interpolators, similarity metrics and cost function optimizers. ITK's focus on biomedical applications is illustrated by the number of biomedical image analysis oriented features available for registration, for example multi-modality similarity metrics and deformable spatial transformation models.

In many biomedical applications where deformable registration is used, there is a need to enforce some properties of the resulting spatial transformation. For example, it is often desirable to preserve the topology of the objects being deformed by the registration process. This can be obtained by ensuring that the spatial transformation is a one-to-one mapping. This process is called diffeomorphic (or invertible) registration. Prior to release 3.8, no diffeomorphic registration algorithm was available in ITK.

This article presents the first diffeomorphic image registration scheme introduced in ITK: diffeomorphic demons [1]. This contribution to the 3.8 release should help ITK keep up with the fast pace of research in deformable registration. Furthermore we show that, contrary to popular belief, diffeomorphic registration can be fast. Typical 3D MR images can indeed be registered in less than 3 minutes with the diffeomorphic demons running on a Mac Pro Dual Quad Core.

### DIFFEOMORPHIC DEMONS BACKGROUND

Since Thirion's seminal paper [2], the demons algorithm has become a popular method for the problem of intra-modality deformable image registration. The demons algorithm has successfully been used by several teams and an open source implementation of it is available in ITK. The success of this method in the field of biomedical imaging can largely be explained by its efficiency. Thirion introduced demons that push according to local characteristics of the images. The forces are inspired from the optical flow equations and the method alternates between computation of the forces and their regularization by a simple Gaussian smoothing.

Looking more closely at the first of these two steps, it is shown that the computation of the demons' forces is an optimization of the image similarity criterion over the space of dense displacement fields. The incremental displacement provided by the demons' forces is thus an update step that is simply added to the current spatial transformation, which is also represented as a dense displacement field.

Such an additive update is efficient but disregards the fact that we work with spatial transformations that do not necessarily live on a vector space. The most natural operation we can endow a space of spatial transformations with is actually the composition and not the addition. In fact, most of the spatial transformations used in biomedical applications live

on some curved space where addition may have no geometric meaning. Looking at rotations is a good example. Rotation can be represented in many ways such as rotation matrices or quaternion. Adding two rotations matrices does not necessarily provide a rotation matrix, whereas the composition of two rotations is a rotation that can be represented by the multiplication of the input rotations' matrices.

An intrinsic approach to perform optimization on such curved spaces of spatial transformations is illustrated in Fig. 1. The idea is to do the computations of the update step on the (flat) tangent space at the current transformation and to project the result back onto the curved space of interest. The projection operator is classically referred to as the exponential mapping.

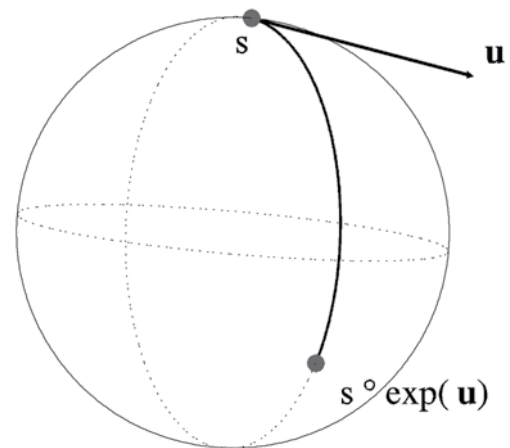


Figure 1: Schematic view of the intrinsic optimization update step on a manifold of spatial transformations. From the current spatial transformation  $s$ , an unconstrained update  $u$  is computed on the tangent space and is projected back onto the manifold through the exponential.

This is exactly what is done in the diffeomorphic demons algorithm [3]. Diffeomorphic transformations are still represented as some displacement fields. In this case, adding invertible transformations could lead to non-invertibility. This often happens in the classical demons scheme. In contrast, in the diffeomorphic demons, the demons' forces (the update step) are first projected onto the space of diffeomorphisms and then composed with the current spatial transformation. The nice property of this approach is that it basically only replaces one step in the classical demons algorithm. Therefore, its ITK implementation fit nicely with the generic Finite Difference Solver Hierarchy.

### SOME RESULTS

Before providing some details about the implementation let us first look at a simple toy example that illustrates the power of diffeomorphic registration compared to a classical approach. Figure 2 clearly shows that, on the standard Circle-to-C example, diffeomorphic demons are able to register the images and provide a smooth one-to-one mapping whereas the classical demons fail and introduce foldings in the transformation.

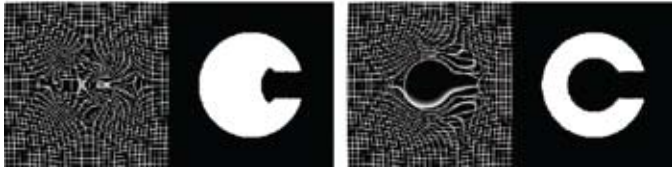


Figure 2: Classical Circle to C registration example. With the same set of parameters the additive demons fails to converge and shows foldings in the registration results whereas the diffeomorphic demons converges with a smooth invertible transformation.

Figure 3 shows the diffeomorphic demons in use in MedINRIA [4]. Two different anatomies were downloaded from Brainweb [5] and registered through MedINRIA's GUI. Using the command line tool the registration of these two 256 x 256 x 181 images runs in 2 minutes and 30 seconds on a 2 x 2.8 GHz quad-core Intel Xeon Apple Mac Pro computer. Such computation time makes the algorithm a good fit for clinical problems.

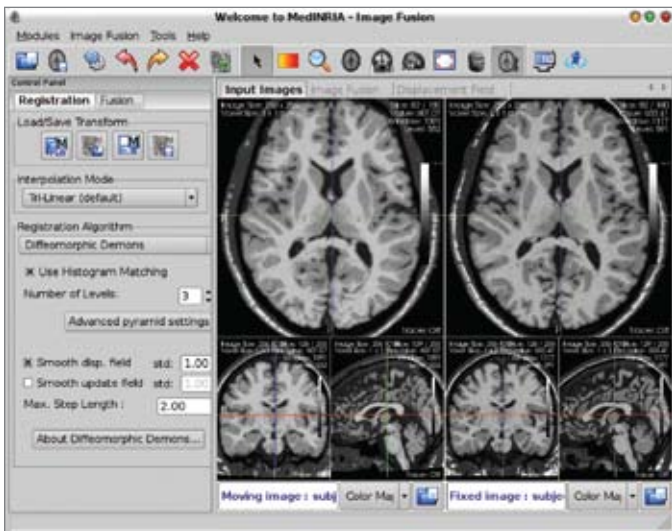


Figure 3: Registration of two BrainWeb images through MedINRIA diffeomorphic demons GUI

## IMPLEMENTATION NOTES

In order to implement the diffeomorphic demons within ITK, the first building block we needed was a code that computes the exponential of a displacement field. Following ITK's spirit of reusable code, this is implemented as an `itk::ImageToImage` filter templated over the types of the input and output images. To ease the creation of this filter two convenient filters have been introduced: `itk::VectorLinearInterpolateNearestNeighborExtrapolateImageFunction` and `itk::DivideByConstantImageFilter`.

The most important filter we implemented is the one that encapsulates the diffeomorphic demons: `itk::DiffeomorphicDemonsRegistrationFilter`. For this filter, a helper class was also introduced: `itk::MultiplyByConstantImageFilter`.

To extend to the demons' forces capabilities already available in ITK, we implemented `itk::ESMDemonsRegistrationFunction`. This function allows for easy control over the maximum step length and the type of gradient used to compute the demons' forces:

```
forces->SetMaximumUpdateStepLength(2.0);
forces->SetUseGradientType(Symmetric);
```

A helper class, `itk::VectorCentralDifferenceImageFunction`, appeared to be useful in this setup.

In order to evaluate the registration results, several filters have also been implemented:

- `itk::DisplacementFieldJacobianDeterminantFilter`
- `itk::GridForwardWarpImageFilter`
- `itk::WarpHarmonicEnergyCalculator`

These classes should help ITK users evaluating the spatial transformations from any registration scheme.

Once the implementation of the diffeomorphic demons was completed, code profiling was performed with Nicolas Savoie from Mauna Kea Technologies. We realized that quite a lot of time was spent in some tiny pieces of code, namely in `itk::LightObject::Register()`, `itk::LightObject::Unregister()` and `itk::TimeStamp::Modified()`. We started using some atomic operations to avoid mutex locks and achieved a large performance gain. Thanks to the community, these efforts have been made fully cross-platform and have been integrated in ITK's trunk. This will be beneficial to every ITK application.

## HELP FROM THE COMMUNITY

Soon after the initial submission of the diffeomorphic demons to the Insight Journal, we received some great feedback from users in the community. This helped us make our software truly cross-platform and helped us fix a few bugs.

We also benefited from the work of Marius Staring who developed a filter to generate a deformation field from an input parametric spatial transformation that was more advanced than the one we initially proposed. This multi-threaded filter allows for the initialization of demons-like algorithms with the results of any prior registration such as a rigid-body registration algorithm.

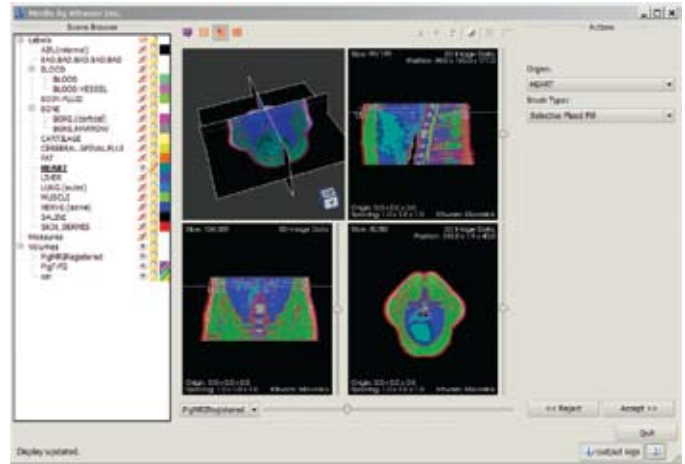
Luis Ibáñez has been of a major help in introducing our code within ITK's trunk. Thanks to his work, a large set of unit tests are now supporting our code and increasing its testing coverage. He also expanded the number of helper classes we proposed with other variants and modified our code to fully match ITK's style.

## CONCLUSION

We have presented the first diffeomorphic registration method to be included in ITK. We believe that this should be a great step in maintaining ITK's leader position in the field of biomedical image registration. The great support and feedback received from the community provides a strong incentive to pursue the open-science goal. The next step for us could be integrating the symmetric extension we proposed at MICCAI this year [6].

## REFERENCES

1. Tom Vercauteren, Xavier Pennec, Aymeric Perchant, and Nicholas Ayache. Diffeomorphic Demons Using ITK's Finite Difference Solver Hierarchy. Insight Journal , ISC/NA-MIC Workshop on Open Science at MICCAI 2007, <http://hdl.handle.net/1926/510>, October 2007
2. Jean-Philippe Thirion. Image Matching as a Diffusion Process: An Analogy with Maxwell's Demons. Medical Image Analysis, 3(2), 1998
3. Tom Vercauteren, Xavier Pennec, Aymeric Perchant and Nicholas Ayache. Non-parametric Diffeomorphic Image Registration with the Demons Algorithm. Proceedings of the 10th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI 2007), 319-326, October 2007
4. MedINRIA, <http://www-sop.inria.fr/asclepios/software/MedINRIA/>
5. BrainWeb, <http://www.bic.mni.mcgill.ca/brainweb/>
6. Tom Vercauteren, Xavier Pennec, Aymeric Perchant, and Nicholas Ayache. Symmetric Log-Domain Diffeomorphic Registration: A Demons-based Approach. Proceedings of the 11th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI 2008), 754-761, September 2008



Paintbrushes are shown within a Maverick-derived application for anatomical segmentation and biological simulation. Paintbrushes have, in part, changed anatomic segmentation from a tedious 2D task that was prone to error, to a robust and intuitive 3D task involving significant computer assistance.



*Dr. Tom Vercauteren is a research engineer at Mauna Kea Technologies where he works on biomedical image processing and analysis with a focus on image registration methods. Dr. Vercauteren's code from his Insight Journal paper on "Diffeomorphic Demons" was one of the most significant contributions to the ITK 3.8 release. His current research interests are biomedical image analysis, statistical signal processing and Sequential Monte Carlo methods.*

## KITWARE NEWS

### VOLVIEW 3.0

In late October Kitware will be releasing VolView 3.0. VolView is an intuitive, interactive system for volume visualization that allows researchers to quickly explore and analyze complex 3D medical or scientific data on Windows, Mac and Linux computers. Users can easily load and interactively explore datasets using 2D and 3D display methods and tools. 3D tools include volume rendering, maximum intensity projections, and oblique reformatting. The ability to save an entire visualization session allows users to easily stop and start a session. Advanced users can perform custom data processing using a simple plug-in API.

## IN PROGRESS

### PAINTBRUSHES IN VTKEDGE

Paintbrushes are a powerful set of extensible editing tools for pixel /voxel values. A brush is an arbitrary 2D or 3D stencil that can be interactively painted onto a labelmap image. The shape of the stencil corresponds to the shape of the brush. For example, the stencil and corresponding brush shapes can be ellipsoids or cubes. Furthermore, an "underlying" image, companion to the labelmap image, can also be used by the paintbrushes. In particular, the paintbrushes can perform computations on that underlying image to modify the painting process. For example, it is easy to create a paintbrush for which the paint is only deposited in the labelmap image when the underlying image's intensities are within a specific range. More complex operations can also be performed, such as region growing using the underlying image to constrain how the paint spreads out from the brush.

The paintbrushes were developed as part of Kitware's Maverick Library for application development. They were funded, in part, by the Air Force Research Laboratories (AFRL), SBIR Phase II Contract No. FA8650-07-C-6756. The paintbrushes have been transitioned to VTKEdge to promote their use outside of Kitware.



VolView 3.0 is a major upgrade that provides many new features including support for loading of multiple datasets simultaneously, support for creating and applying 2D and 3D display presets, and the ability to make image measurements within a user defined contour. A simpler and easier to use interface allows for rapid navigation of the user interface. Users can download a free trial of this software at [kitware.com/volview3](http://kitware.com/volview3). Please visit [kitware.com](http://kitware.com) for more information on purchasing or upgrading to the new VolView release.

## ITK TCON 2.0: SECOND LIFE

The development of the Insight Toolkit (ITK) started in 1999 with a geographically distributed group of about 50 developers. In order to coordinate the development activities, several mechanisms were put in place: mailing lists, a weekly telephone conference and regular meetings held about twice a year. The weekly phone conference was referred to as "the tcon" and hosted many interesting discussions, and even some occasional C++-based fights that we will not mention here. The tcon offered an agile mechanism for discussing difficult design issues and their implementation, and provided a way of monitoring the progress of the entire project.

The ITK weekly phone conference has been ongoing for the past nine years and is one of the main vehicles of communication and coordination among developers. However given the growth, and particularly the internationalization, of the development team the phone conference is no longer inclusive enough to fulfill its original purpose effectively. We are therefore experimenting with new technologies hoping to find a new method of communication that can effectively include developers from different geographical zones. With this goal in mind we have started hosting the ITK weekly meetings in the Virtual Environment of "Second Life" (<http://secondlife.com/>). Staff from Linden Lab (<http://lindenlab.com/>), the makers of Second Life, have kindly allowed us to use one of the theaters at the Hippotropolis Island (<http://wiki.secondlife.com/wiki/Hippotropolis>) to host our weekly meetings. We are calling this new event, the "ITK Tcon 2.0".



*The Hippotropolis region was established to provide a meeting place for developers and residents interested in improving the open source code of the Second Life Viewer.*

*Image and caption courtesy of the Second Life Wiki (<http://wiki.secondlife.com/wiki/Hippotropolis>), Copyright © 2008 Linden Research, Inc. Licensed under the Creative Commons Attribution-Share Alike 3.0 License (see the complete license terms at <http://creativecommons.org/licenses/by-sa/3.0>).*

Participants simply need to install the free Second Life Viewer application. The Second Life Viewer is an open source project configured using CMake. It runs on Linux, Windows and Mac. In order to join the meetings, participants start their viewer and then teleport to the Hippotropolis location. During the meeting they can use voice communication, with a sound quality superior to that of a typical phone connection, while at the same time exchanging text information in a chat window. Several 3D primitives have been created for connecting directly to the ITK Wiki, the Dashboard and the Web portal to the CVS repository.

The weekly meetings are open to the public, and you are welcome to join us every Friday at 1pm Eastern Time. This is a great opportunity for users and developers to interact using modern technologies, and to make the toolkit a more useful resource for all of us. You can find more details at [itk.org/Wiki](http://itk.org/Wiki) by searching for "ITK in Second Life".

## MIDAS-JOURNAL HOSTS MICCAI WORKSHOP PROCEEDINGS

I recently returned from MICCAI 2008, the premier conference in my field of medical image analysis. As was the case last year, I co-organized a successful, intermediately-sized workshop on evaluating medical image analysis methods against real clinical study datasets. In contrast to last year's cumbersome review and non-permanent publication process, this year our workshop used Kitware's Midas-Journal as an online publication system for its proceedings. What a difference in so many ways! For me, this was a perfect partnership. Our workshop whole-heartedly endorses open access and open reviews. These online journal systems simplified the submission and reviewing process significantly and, more importantly, provided a permanent location for the proceedings. As a result, papers in the workshop proceedings can easily be cited and disseminated. I am quite convinced that the use of this online publishing system contributed substantially to the higher interest and participation at the workshop as compared to last year. Additionally appealing for a comparison workshop such as ours, is the ability for authors to provide updated revisions of their manuscript in the case of bug fixes or updated evaluations. In summary, the Midas-Journal allowed our workshop to provide high-quality proceedings and to improve over previous years without the need of a complicated and costly partnership with a traditional publishing house. This is the way to publish for workshop and small conference proceedings! I will definitely use it again for any of my future workshops.



*Dr. Martin Styner is a research assistant professor in the Department of Computer Science with a joint appointment in the Department of Psychiatry at the University of North Carolina at Chapel Hill (UNC). Dr. Styner has co-authored 23 papers in peer reviewed journals and 63 papers in peer reviewed conferences. His main field of expertise is in medical image processing and analysis for neuro-imaging and computer assisted surgery.*

## INSIGHT-JOURNAL AND MIDAS-JOURNAL 2.0

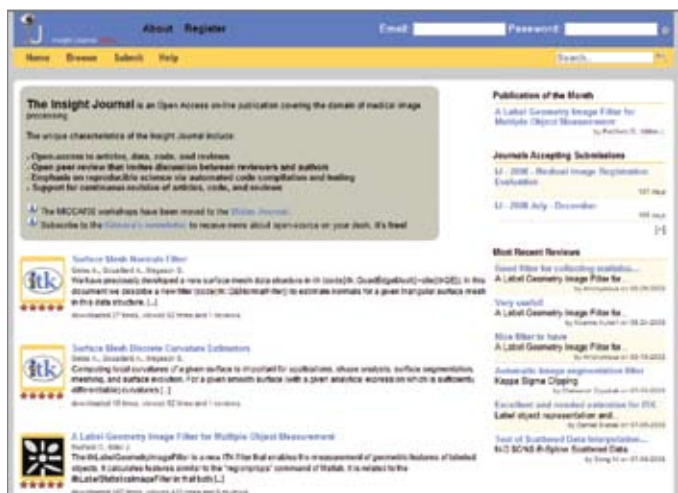
The Insight-Journal is an electronic publication which supports the verification of reproducibility in technical and scientific medical image analysis papers. Papers submitted to the Insight-Journal include the source code, data and param-

eters that are needed for reproducing the work described in the paper. An automatic system builds executables from the submitted source code and runs them using the data and parameters provided by the papers' authors.

Papers and their associated source code, data and parameters are publicly distributed in Open Access under a Creative Commons Attribution License. All Journal readers can make reviews which are posted publicly. Readers can also rate reviewers and in this way create a circle of checks-and-balances.

The Insight-Journal is designed to satisfy the needs of the readers and in particular to materialize the rapid dissemination of information by taking advantage of web technologies and the open collaboration of a community. In that spirit, authors are allowed to post revisions of the papers, and to upload additional material at any moment. This is done with the goal of providing readers with the best quality of information possible.

After three years, the Insight-Journal ([insight-journal.org](http://insight-journal.org)) is host to more than 250 publications and approximately 1,000 registered users. The new version of the website went live in early September with a completely new design.



Along with the Insight-Journal and the new website, the Midas-Journal ([midas-journal.org](http://midas-journal.org)) also launched. Both websites are running on the same web platform; however, the Midas-Journal doesn't require the submission of any

source-code or open-data associated with a publication. The Midas-Journal is perfect for streamlining the review process, accepting papers, assigning reviewers and publishing an article online. If you are interested in using the Midas-Journal for your next workshop or conference feel free to contact us at [info@midasjournal.org](mailto:info@midasjournal.org).

The Midas-Journal hosts papers from several workshops presented at the recent Medical Image Computing and Computer-Assisted Intervention (MICCAI) conference. These papers are free to download and open for public review.

Starting in 2009, articles from both the Insight-Journal and Midas-Journal will be published in the *Kitware Source* newsletter which is distributed to more than 300 community members worldwide. This is an excellent opportunity for open-source developers to share their current advances with the community.

Stay tuned for more information on these publication resources in the near future and check out the following video which fully highlights all of the new features and capabilities of the Insight- and Midas-Journals' 2.0 release: [insight-journal.org/blog](http://insight-journal.org/blog).

### Acknowledgments

The Insight-Journal has been initiated by Dr. Luis Ibáñez. The new Insight-Journal and Midas-Journal websites have been designed and implemented by Julien Jomier and Charles Marion.

### ITK 3.8 IS NOW AVAILABLE FOR UBUNTU 8.04

Thanks to Dr. Paul Novotny ITK 3.8 is now available for Ubuntu 8.04. Dr. Novotny works at the Children's Hospital in Boston where they are using ITK to prototype a tool for ultrasound-guided heart surgery. His department is pioneering the use of 3D ultrasound techniques to operate on the beating heart. ITK and Python are used to read images and prototype algorithms in order to write custom software for surgeons to implement these surgical techniques.

Currently, Dr. Novotny creates his own test packages and the developers of the Vascular Modeling Toolkit ([vmtk.org](http://vmtk.org)) who are one of his largest user bases of ITK/Ubuntu packages help test the software.

Kitware would like to thank Dr. Novotny for his work on this package. Such volunteer acts allow our toolkits to be more easily adopted by a broad set of users, and we are greatly appreciative of these types of efforts.

If you are interested in installing the ITK 3.8 Ubuntu package please visit: <http://paulnovo.org/repository>. If you are interested in helping Dr. Novotny test ITK Ubuntu packages please contact him via email at [paul@paulnovo.org](mailto:paul@paulnovo.org).

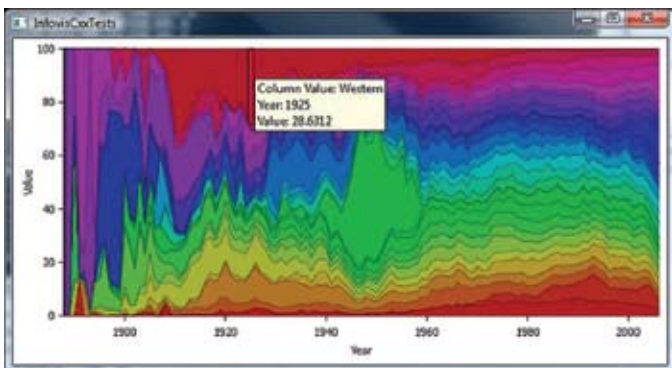


**Paul M. Novotny, Ph.d.** is an Associate Scientific Researcher in the Department of Cardiac Surgery at Children's Hospital Boston, and an Instructor of Cardiac Surgery at Harvard Medical School. His research focuses on real-time image processing of high-data rate medical imaging, such as 4D Ultrasound, and applying these methods to cardiac surgery.

## SANDIA AND KITWARE PRESENT INFORMATION VISUALIZATION WITH VTK AT VISWEEK 2008

The merging of information visualization and scientific visualization is an active area of research, development and discussion. Recently the Visualization Toolkit (VTK) has been expanded with new data structures and a large set of filters, views, database adapters and other components that support the processing of informatics data. This tutorial, organized by Brian Wylie of the Sandia National Labs and taught by Jeff Baumes (Kitware), Timothy Shead (Sandia) and John Greenfield (Sandia), provided a detailed overview of VTK's new components and demonstrated their use within the flexible pipeline architecture. Attendees learned the basics of using VTK for information visualization with a focus on building quick, functional applications. Additionally the tutorial also demonstrated how users can extend the current capabilities to include their own functionality.

If you missed this tutorial at IEEE Vis and are interested in finding out more about the information visualization capabilities of VTK please contact us at [kitware@kitware.com](mailto:kitware@kitware.com). Kitware training courses offer the best opportunity to gain an in-depth understanding of our open source products. To find out about upcoming training courses on information visualization with VTK please contact us at [courses@kitware.com](mailto:courses@kitware.com).



Stacked chart showing the number of movies in each genre over time.

## IGSTK: BUILDING HIGH QUALITY ROADS WITH OPEN SOURCE SOFTWARE

Luis Ibáñez gave a presentation on the following paper "IGSTK: Building High Quality Roads with Open Source Software" at the Systems and Architectures for Computer Assisted Interventions workshop at MICCAI 2008.

The paper is a position statement on the benefits that an Open Source toolkit can bring to the field of image guided surgery. The particular example discussed is the IGSTK toolkit, an open source software project designed for supporting the development of image guided surgery software applications.

The goal of the workshop was to establish a forum for discussing open source toolkits and open interfaces and how they have been applied in the development of computer assisted intervention systems. The intent was to establish a dialogue between the open source community, industry, and researchers in the field.

The workshop included presentations and panel discussions about: 1) existing open source toolkits, 2) open (research) interfaces to medical products, and 3) integrated systems for computer assisted interventions. One outcome of the

workshop will be a technical paper (white paper) that will be submitted for review to the International Journal of Medical Robotics and Computer Assisted Surgery. If the paper is completed by October 2, 2008, the paper will be considered for a special issue on New Technologies.

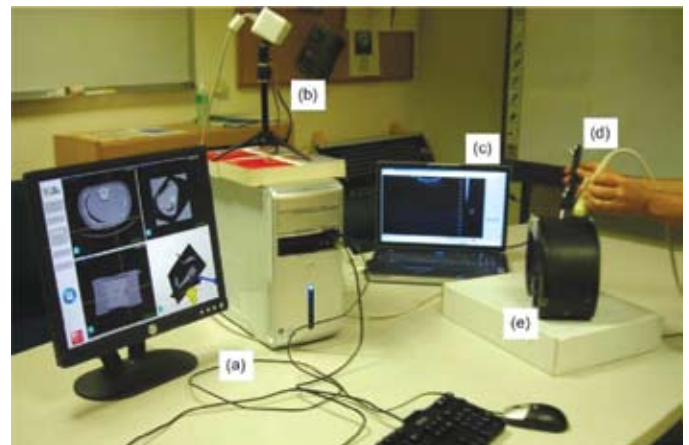
To contribute to this ongoing dialog, please join the IGSTK mailing list at <http://public.kitware.com/mailman/listinfo> and select "IGSTK-Users". To access the full paper presented at MICCAI please visit the MIDAS Journal at [midasjournal.org](http://midasjournal.org) and search for "Building High Quality Roads".

L. Ibáñez, A. Enquobahrie, M. Turek, J. Jomier, R. Avila, P. Cheng, Z. Yaniv, F. Lindseth, K. Gary, K. Cleary "IGSTK: Building High Quality Roads with Open Source Software", MIDAS Journal, July 31, 2008.

## IGSTK AND OPENIGTLINK: INTERFACING PROPRIETARY HARDWARE WITH OPEN SOURCE SOFTWARE

A demonstration of a CT-Ultrasound Guided system for needle biopsy using IGSTK was presented at MICCAI 2008 by Sebastian Ordas from the Computer Aided Interventions and Medical Robotics (CAIMR) at ISIS Center at Georgetown University's Medical Center. The presentation addressed the key technical challenges in developing an extensible image-guided navigation system that interfaces with external proprietary hardware.

CAIMR has been working to interface proprietary hardware with the Image-Guided Surgery Toolkit (IGSTK) in order to extend its CT-based navigation system for needle biopsies to incorporate ultrasound as a complementary real-time imaging modality. The original navigation system was implemented using the IGSTK, a platform independent toolkit for rapid prototyping of IGS applications. The only external hardware component on which the system is dependent was the Terason T2000 portable US tracking system.



CT/US navigation system components: (a) navigation system computer; (b) MicronTracker localizer, (c) Terason T2000 US system, (d) tracked tools (pointer and US probe), and (e) abdominal phantom

Incorporating the Terason system into the existing navigation system presented two challenges: an additional bus was needed to concurrently run the Terason system and due to Terason's ActiveX controls the navigation system could only run under the Windows OS.

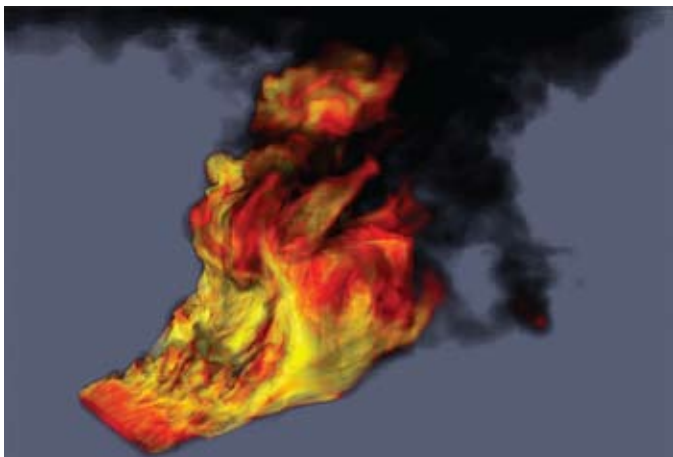
Utilizing the recently proposed OpenIGTLink communication protocol solved both of these problems by allowing the navigation system to run as two processes, either on the

same computer under the Windows OS, or on two separate computers connected via an intranet with US acquisition performed on a computer running the Windows OS.

Kitware has been working with CAIMR to develop IGSTK through an NIH R01 grant. For more information on the IGSTK framework or on the work presented at this MICCAI workshop specifically, please contact us at [kitware@kitware.com](mailto:kitware@kitware.com) or the project manager, Patrick Cheng at [cheng@isis.georgetown.edu](mailto:cheng@isis.georgetown.edu).

### **SANDIA AND KITWARE PRESENT ADVANCED PARAVIEW VISUALIZATION AT VISWEEK 2008**

ParaView is a powerful open-source turnkey application for analyzing and visualizing scientific datasets ranging from small desktop-sized problems to the world's largest simulations. ParaView is used by numerous government, educational, and commercial institutions throughout the world. Designed to be configurable, extendible, and scalable, ParaView is built upon the Visualization Toolkit (VTK) to allow for the rapid deployment of visualization components. Through this tutorial, organizer Kenneth Moreland of the Sandia National Labs brought together David Thompson (Sandia), Timothy Shead (Sandia), John Biddiscombe (CSCS) and Utkarsh Ayachit (Kitware), whom are all designers and builders of ParaView, as means of giving researchers and developers detailed guidance on the behavior and abilities of the ParaView application.



*Cross Wind Fire Simulation. Image courtesy of Sandia Labs.*

This allowed the tutorial participants to solve their unique visualization problems, modify the ParaView application to their specific problem domains, or leverage the design into their own applications. Participants were taught how to customize their visualization with selection, scripting, plugins, and vertical applications. Additionally the tutorial covered some of ParaView's new and advanced features including higher order elements, particle visualization, and information visualization.

If you missed this tutorial at IEEE Vis and are interested in finding out more information on the advanced features of ParaView please contact us at [kitware@kitware.com](mailto:kitware@kitware.com). Kitware training courses offer the best opportunity to gain an in-depth understanding of our open source products. To find out about upcoming training courses on ParaView please contact us at [courses@kitware.com](mailto:courses@kitware.com).

### **LARGE SCALE VISUALIZATION WITH PARAVIEW TUTORIAL AT SUPERCOMPUTING 2008**

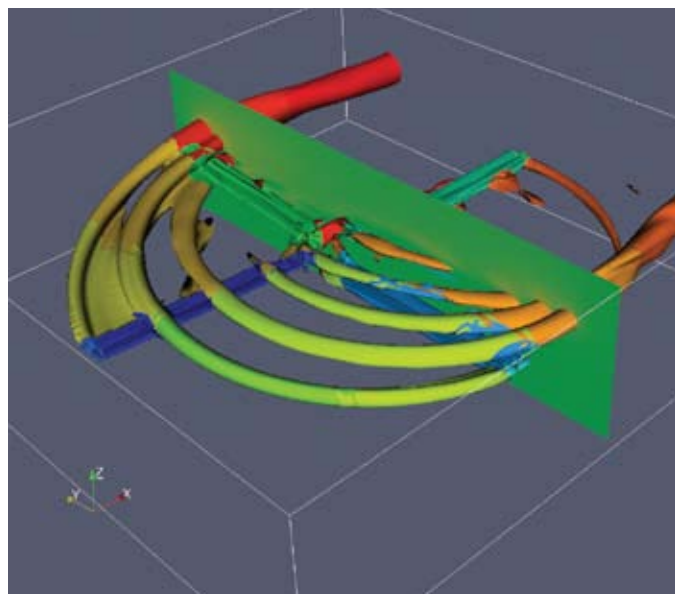
Berk Geveci, Utkarsh Ayachit and Dave DeMarle from Kitware and Ken Moreland and John Greenfield from Sandia National Labs will be presenting a tutorial on "Large Scale Visualization with Paraview" on Monday November 17 during the Supercomputing 2008 conference. This tutorial presents the architecture of ParaView and the fundamentals of parallel visualization. Attendees will learn the basics of using ParaView for scientific visualization with hands-on lessons. The tutorial will provide an introduction to scripting and extending ParaView and provide attendees with detailed guidance for visualizing the massive simulations run on today's supercomputers. Attendees should bring laptops to the tutorial so that they may install ParaView and follow along with the demonstrations.

ParaView is a powerful open-source turnkey application for analyzing and visualizing large data sets in parallel. Regularly used by Sandia National Laboratories analysts to visualize simulations run on the Red Storm and ASC Purple supercomputers and by thousands of other users in worldwide, ParaView is designed to be configurable, extensible, and scalable. In order to allow for the rapid deployment of visualizations components, ParaView is built upon the Visualization Toolkit (VTK).

If you are unable to attend this tutorial at SC08 and are interested in finding out more information on the Large Scale Visualization with ParaView please contact us at [kitware@kitware.com](mailto:kitware@kitware.com). Kitware training courses offer the best opportunity to gain an in-depth understanding of our open source products. To find out about upcoming training courses on ParaView please contact us at [courses@kitware.com](mailto:courses@kitware.com).

### **PROCESSING VISUALIZATION INFORMATION DURING CFD SIMULATION CONTRACT**

Kitware was recently awarded a Phase II SBIR from the Department of Defense. The purpose of this project is to develop tools for co-processing unsteady CFD codes for the purpose of visualization and analysis.

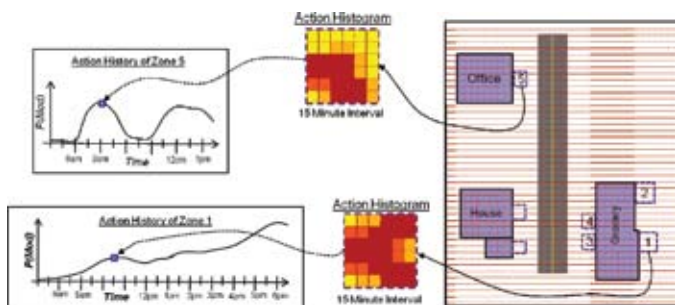


As the compute capability of our computational resources grows, we are facing a discrepancy between our ability to produce results and store them for analysis. We are quickly outstripping the capability of IO resources to store results from time-dependent simulations and the capability of post-processing tools running on small- to medium-sized clusters to analyze such data. The main objective of this SBIR is to integrate core data processing with the simulation to enable scalable data analysis.

The Phase I effort focused on developing a prototype of a distributed co-processing system capable of processing, at run-time, results of large and distributed simulations. During Phase I, we developed a co-processing library prototype and successfully tested it with two CFD codes: Overflow from NASA and Phasta from RPI. The specific goals of the Phase II are to develop: an extensible and flexible co-processing library; configuration tools for co-processing configuration; CFD specific analysis and visualization algorithms; and tools for visualizing, analyzing and managing extracts. The estimated completion date for this Phase II contract is September 30, 2009.

## BLUE

Kitware was awarded a DARPA STTR Phase I \$100K contract to develop algorithms for Building Labeling in Urban Environments (BLUE). The goal of the effort is analyze persistent video in order to determine the types of buildings in the scene based on the activities of people and vehicles surrounding them. We are partnering with the GRASP Lab at the University of Pennsylvania on this contract for their expertise in video analysis and urban structure recognition. The eight month phase I effort will result in a proposal for a two-year phase II effort which will develop the initial concepts into a mature prototype. The algorithms developed in this contract will be used by the military for urban mapping and can be commercially applied to the retail industry in the future.



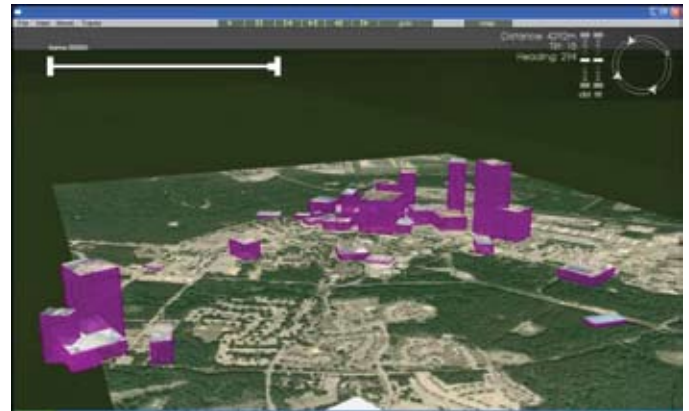
*We expect that the activities at the entrances of different buildings will have patterns that correlate with the type of building. We plan to model these patterns and use them to recognize the building type.*

## KITWARE AWARDED \$6.7M DARPA CONTRACT

On September 11, 2008, Kitware was awarded a \$6,735,503 Phase I contract for the Video and Image Retrieval and Analysis Tool (VIRAT) program by the Defense Advanced Research Projects Agency (DARPA). Kitware competed with 19 other companies for the contract.

Kitware's team proposed building a revolutionary video analyst workstation called Video and Image-Based Retrieval and ANalysis Tool (VIBRANT). VIBRANT will leverage the most promising technologies in computer vision, video data indexing, and content-based retrieval in an integrated system that

will filter and prioritize massive amounts of archived and streaming video. The most high-value intelligence content will be clearly and intuitively presented to the video analyst, resulting in substantial reductions in analyst workload per mission as well as increasing the quality and accuracy of intelligence yield.



*A visualization of a video query result where each purple tower is a video clip retrieved from a video archive. Height indicates the level of similarity to a query video clip supplied by a video analyst. The analyst can browse the result clips in 3D+time by watching all clips play simultaneously on the towers.*

Currently, video analysis for Predator and other aerial video surveillance platforms is very labor intensive, and limited to metadata queries, manual annotations, and "fast forward" examination of clips. The ability to quickly search large volumes of existing video data and monitor real-time video data for specific activities or events will provide a dramatic new capability to the US military and intelligence agencies.

"Kitware is thrilled to be part of VIRAT. DARPA has put together a highly ambitious program that addresses the most critical problems in video analysis, which is the hottest topic in the ISR community today. This project will really make a difference to the warfighter," said Anthony Hoogs Kitware's Director of Computer Vision and the project leader for this contract. "The whole team – including our industry and academic partners – did a terrific job in formulating a highly innovative solution to a very challenging problem."

Kitware's Computer Vision group has extensive expertise developing robust solutions for real-world problems, outstanding research community credentials and a demonstrated ability to deliver on DARPA programs. To meet the VIRAT program's needs Kitware has assembled a world-class team including two leading defense technology companies, Honeywell and General Dynamics, and multiple internationally-renowned academic researchers.

General Dynamics Advanced Information Systems brings the systems integration, technology transfer and domain expertise of a major defense contractor that is a leader in operational video exploitation systems. Honeywell Labs is a market leader in commercial video surveillance systems with a substantial presence in the defense industry and an outstanding research group in video indexing.

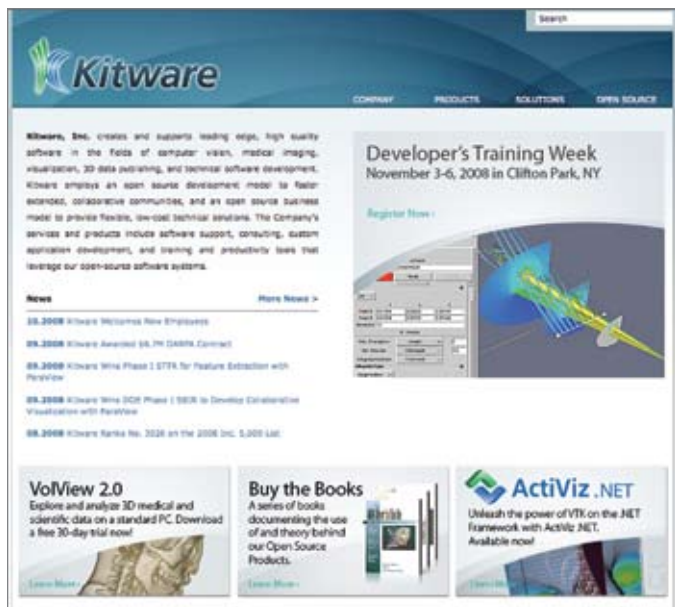
Academic team members were selected based on their demonstrated ability to solve real problems and their outstanding research credentials. They are Carnegie Mellon University; the International Computer Science Institute affiliated with

UC Berkeley; California Institute of Technology; Columbia University; the University of Texas at Austin; Rensselaer Polytechnic Institute; and the University of Maryland. Together, their academic partners form a complementary group spanning all of the critical VIRAT technology elements.

The VIRAT program includes Phases 2 and 3 for one year each, which are options on the awarded contract. The estimated completion date for this Phase 1 contract is March 11, 2010.

## NEW KITWARE WEBSITE

The Kitware website, [www.kitware.com](http://www.kitware.com), has been restructured in order to better support Kitware's growing line of products and services. The new site, which went live in June, allows users to easily access Kitware's open source projects, commercial products, find out information on upcoming training courses, and get information on our myriad of consulting packages. Users may also order books directly through the new site. One of the new features of [www.kitware.com](http://www.kitware.com) are the solutions areas which, we hope, will help educate our user community about the full potential of Kitware's products and services.



## NEW EMPLOYEES

### Chuck Atkins

Chuck joined Kitware in September 2008 where he is currently involved as an R&D Engineer for the Computer Vision team. Chuck's current research is in scalable software systems for video processing. Prior to joining Kitware Chuck worked as a Senior Systems Analyst for Hutchinson Technology Inc. (HTI) in the Advanced Vision Development group where he was one of the three primary software developers on a high resolution multi-camera, multi-node visual inspection system. Chuck received his BS in Computer Science from Oklahoma State University.

### Matt Bowman

Matt joined Kitware in September 2008 as a Systems Administrator. His role at Kitware is to maintain all Information Technology aspects of the company. Prior to joining Kitware Matt worked as an IT manager for Informz Inc., an email marketing company. Matt received his B.S. in Computer Science degree from State University of New York in Oneonta.

### Katie Cronen

Katie joined Kitware in September 2008 as a Human Resources Generalist. Her role is to participate in all Human Resources functions throughout the company including recruitment, performance management, employee relations, and the creation and compliance of employment policies and decisions. Katie received her B.S. in Business Administration, with a concentration in Human Resource Management, from Ramapo College of New Jersey. She is certified as a Professional in Human Resources (PHR) and is a member of the Society for Human Resource Managers (SHRM).

### Yvette Fitzgerald

Yvette joined Kitware in September 2008 as a contracts administrator. Her role at Kitware is to develop and oversee contract review, writing and negotiation procedures. Prior to joining Kitware, Yvette was a corporate litigation associate with a large law firm in Delaware. She handled various corporate matters including contract disputes, shareholder actions, derivative actions, and partnership dissolutions. Yvette received her BA with a concentration in political science and communications from the State University of New York at Geneseo and her Juris Doctor from William and Mary Law School.

### Zhanping Liu

Dr. Liu joined Kitware in September 2008 as an R&D Engineer. Dr. Liu is currently working on VTK and ParaView. Prior to joining Kitware Dr. Liu worked as a post-doctoral associate with the Micro-CT Lab in the Department of Radiology at the University of Iowa and then as a research scientist with the Visualization, Analysis and Imaging Lab in the High-Performance Computing Collaboratory (HPC2, formerly the NSF ERC --- Engineering Research Center for Computational Field Simulation) at Mississippi State University. He received his BS in Mathematics from Nankai University and PhD in Computer Science from Peking University.

### Pat Marion

Pat joined Kitware as a student intern in 2007 where he contributed to the ParaView project. In 2008, he received his BS in Computer Science from Rensselaer Polytechnic Institute and moved to North Carolina to join Kitware's Chapel Hill office as an R&D Engineer. Pat's current research is in landmark detection of algorithms for polygonal data and prioritized, streaming pipelines.

### Hua Yang

Dr. Yang joined Kitware in August 2008 as an R&D Engineer with a focus on computer-guided surgery and medical image segmentation. Prior to joining Kitware Dr. Yang worked at the University of North Carolina at Chapel Hill as a member of the Augmented Reality Group where he contributed to developing 3D guidance systems that improve surgeons' ability to accurately perform minimally invasive surgeries (MIS). Dr. Yang received his BE in Electronic and Information Science from Huazhong University of Science and Technology, his MS in Pattern Recognition and Artificial Intelligence from the Institute of Automation, Chinese Academy of Sciences and his PhD in Computer Science from the University of North Carolina at Chapel Hill.

## NEW OPEN SOURCE PROJECT SITES

In late October Kitware will launch the new ParaView ([paraview.org](http://paraview.org)) and CDash ([cdash.org](http://cdash.org)) websites. The new CMake ([cmake.org](http://cmake.org)) site went live at the beginning of September. Additionally, a brand new site for the new VTKEdge toolkit ([vtkedge.org](http://vtkedge.org)) will go live in late October as well. The new site design is focused on providing researchers and developers with streamlined access to information on Kitware's open source toolkits. These sites will provide users with a multitude of resources to assist in development and create easier access to licensing information, related publications, ways to get involved with the projects, and resources for development and support. Additionally, the new sites are uniformly designed so that users will find consistency when accessing information from project to project. We are expecting to release new sites for VTK ([vtk.org](http://vtk.org)) and ITK ([itk.org](http://itk.org)) by the end of this year.



## EMPLOYMENT OPPORTUNITIES

### Computer Vision Software Engineer

Qualified candidates will have a broad understanding of surveillance applications such as: action, event and activity analysis; motion segmentation; and tracking; strong software skills (C++ or Java); excellent communications skills; and an enthusiasm for progressive innovation. Candidates should have experience in video processing, video surveillance and/or computer vision. US Citizens/Green Card holders preferred; permanent resident status is a plus.

### Medical Imaging Software Engineer

Qualified candidates will be experienced in developing medical image analysis software in at least one of the following specialties: volume rendering, image processing or 3D segmentation; medical image processing algorithms such as: registration, feature recognition and motion analysis; strong software skills (C++); a solid foundation in object-oriented analysis and design; and excellent math skills. Candidates should have experience in image processing, medical image analysis, DICOM, ITK and/or VTK.

### Software Developer

Qualified candidates will be expert C++ software developers with an in-depth knowledge of the software build process including a variety of compilers (and their quirks) and software quality processes such as TDD and extreme programming. CMake expertise is a definite plus; demonstrated experience and accomplishment is a must. Position entails working with external customers and internal projects establishing, enforcing and extending Kitware's quality software process.

### Advanced OpenGL Programmer

Qualified candidates will be expert OpenGL programmers with CUDA and GPU experience. MS or BS preferred. Candidates will work with Technical Leadership to develop advanced visualization and image processing solutions.

In addition to providing readers with updates on Kitware product development and news pertinent to the open source community, the *Kitware Source* delivers basic information on recent releases, upcoming changes and detailed technical articles related to Kitware's open-source projects. These include:

- The Visualization Toolkit ([www.vtk.org](http://www.vtk.org))
- The Insight Segmentation and Registration Toolkit ([www.itk.org](http://www.itk.org))
- ParaView ([www.paraview.org](http://www.paraview.org))
- The Image Guided Surgery Toolkit ([www.igstk.org](http://www.igstk.org))
- CMake ([www.cmake.org](http://www.cmake.org))
- CDash ([www.cdash.org](http://www.cdash.org))
- KWWidgets ([www.kwwidgets.org](http://www.kwwidgets.org))
- BatchMake ([www.batchmake.org](http://www.batchmake.org))
- VTKEdge ([www.vtkedge.org](http://www.vtkedge.org))

Kitware would like to encourage our active developer community to contribute to the *Source*. Contributions may include a technical article describing an enhancement you've made to a Kitware open-source project or successes/lessons learned via developing a product built upon one or more of Kitware's open-source projects. Authors of any accepted article will receive a free, five volume set of Kitware books.

Kitware's Software Developer's Quarterly is published by Kitware, Inc., Clifton Park, New York.

Contributors: Lisa Avila, Rick Avila, Stephen Aylward, Utkarsh Ayachit, Jeff Baumes, Kevin Cleary, Andinet Enquobahrie, Berk Geveci, Doug Gregor, Bill Hoffman, Anthony Hoogs, Luis Ibáñez, Julien Jomier, Paul Novotny, Sebastian Ordas, Niki Russell, Martin Styner, Will Schroeder, and Tom Vercauteren.

Design: Melissa Kingman, [www.elevationda.com](http://www.elevationda.com)

Editor: Niki Russell

Copyright 2008 by Kitware, Inc. or original authors.

*No part of this newsletter may be reproduced, in any form, without express written permission from the copyright holder. Kitware, ParaView, and VolView are all registered trademarks of Kitware, Inc.*

To contribute to Kitware's open source dialogue in future editions, or for more information on contributing to specific projects, please contact the editor at [kitware@kitware.com](mailto:kitware@kitware.com).