

Verification of Visualization: A Software Engineering Perspective

Robert S. Laramée
Visual and Interactive Computing Group
Computer Science Department
Swansea University, UK
r.s.laramée "at" swansea.ac.uk

9/9

0800 Antan started
1000 " stopped - antan ✓ { 1.2700 9.032847025
13:00 (033) MP-MC ~~1.98244000~~ 9.037846995 correct
 (033) PRO 2 2.130476415 (2) 4.615925059(-2)
 correct 2.130476415
Relays 6-2 in 033 failed special speed test
in Relay " 11.00 test "

1100 Started Cosine Tape (Sine check)
1525 Started Multi Adder Test.

1545  Relay #70 Panel F
(moth) in relay.

1630 Antan started.
1700 closed down.

Relay 3145
Relay 3372

Verification: Motivation

“Speed is irrelevant if you are traveling in the wrong direction.”

-Mahandas Gandhi

- intelligence, speed vs accuracy, (in)correctness

How do we know if what we see is correct?

Examples from the UK

August 2011: Riots in the UK -many photos

The screenshot shows the Guardian website's 'UK riots' section. The browser address bar displays 'www.guardian.co.uk/uk/london-riots'. The page features a navigation menu with categories like News, Sport, and Comment. A breadcrumb trail highlights 'UK riots'. The main content area is divided into several sections:

- Top stories:** Includes a photo of riot police and a headline 'Riot curfews for public proposed by Home Office'. The text states: 'Home secretary Theresa May backs police power to clear streets for first time since Riot Act was repealed'.
- Reading the riots:** Features a photo of a burning building and a headline 'A data-driven study into the causes and consequences of the riots'. The text asks: 'What really caused the August riots in the UK? A team of Guardian investigative and data journalists examine all the evidence'.
- Latest comment:** Includes a comment by Akala: 'Rap has become a news network of the downtrodden, oppressed and the socially conscious across the globe' with 138 comments.
- It's a hip-hop planet:** A sub-section header.
- Understanding the England riots from the perspective of those responsible:** A sub-section header.
- May seeks stronger police riot powers:** Includes a photo of riot police and a headline. The text states: 'Home secretary wants curfews to create "no-go" areas during serious outbreaks of disorder'.
- Coverage of the UK riots on Twitter:** A section with tweets from users like helenc1976 and PaulLewis.

On the right side, there is an advertisement for 'Economy Skycouch' with prices for Los Angeles (£160) and Auckland (£430) per couple, one way. Below the ad is a 'Book now' button and the website 'airnewzealand.co.uk'.

Examples from the UK

August 2011: Riots in the UK -many photos



Examples from the UK

August 2011: Riots in the UK -many photos



Examples from the UK

The “Pillage People”



Verification: Motivation

How do we know if what we are seeing is real, i.e., the truth?

- Verification and Validation (V&V) is vital but often neglected
- We advocate software engineering approach to V&V in visualization
- There are several software engineering related guidelines to V&V
 1. Visualize Tests and Comparisons
 2. Visualize Data Structure Traversal and Evolution
 3. Classify and Color Map
 4. Incorporate Algorithm Parameters into GUI
 5. Run Simple Error Checks
 6. Introduce a Step Function
 7. Make Use of Still-Image Driven Animation
 8. Test on a Variety of Data Sets
 9. And more

Verification in Visualization Challenges

V&V in visualization software is difficult! Why?

- Challenging algorithms
- Large amounts of data, iterations, comparisons etc.
- Complex data structures
- Traditional software engineering tools de-couple information they report with spatio-temporal domain in which unexpected problems occur, e.g. `printf()`, setting breakpoints

Guidelines inspired by development experience in both industry and academia

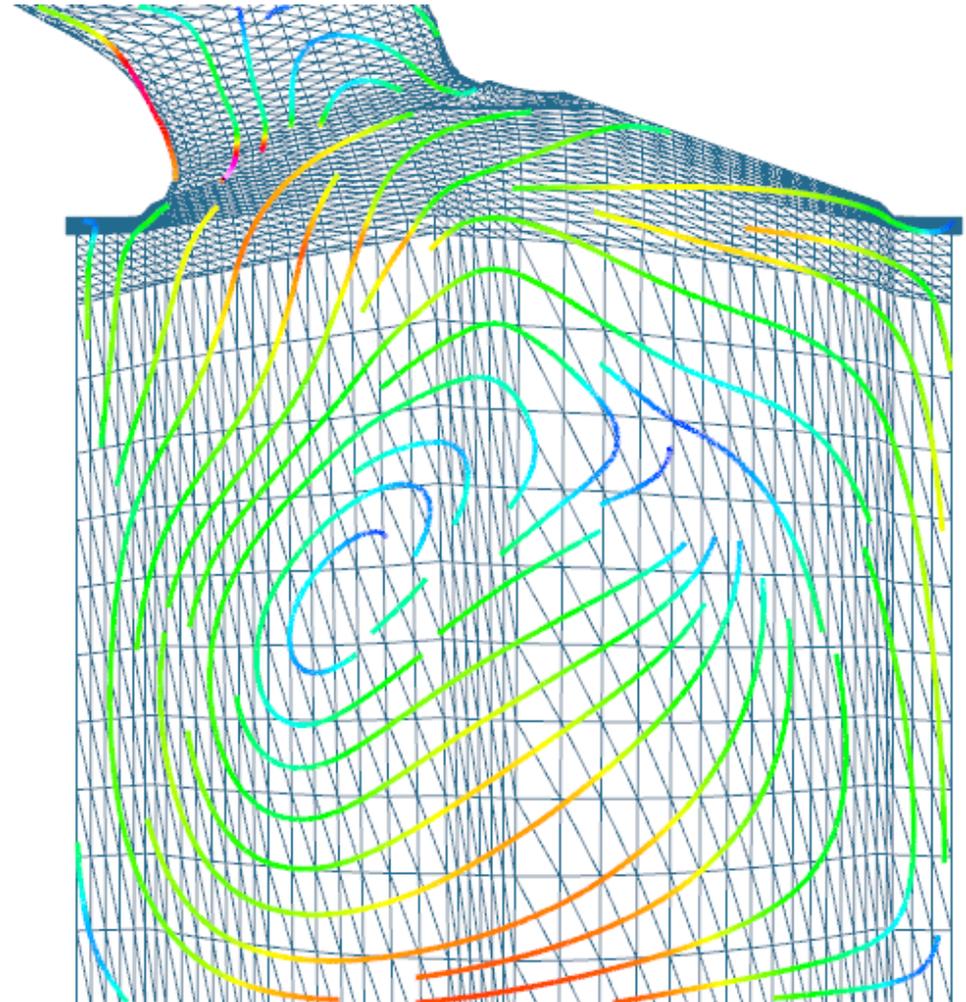
Key: (1) exploit strengths of visualization itself + (2) combine with traditional good software engineering practices

1. Visualize Tests and Comparisons

- Vast majority of algorithms involve tests or comparisons between visualization primitives: points, line segments, polygons, voxels, texels, tetrahedra etc.
- Visualize tests between two or more of these primitives by highlighting focus object, a , and another object, b , at run time.
- This :
 - (1) informs the developer if a and b are expected primitives to test and
 - (2) can verify results of comparison

1. Visualize Tests and Comparisons, Example

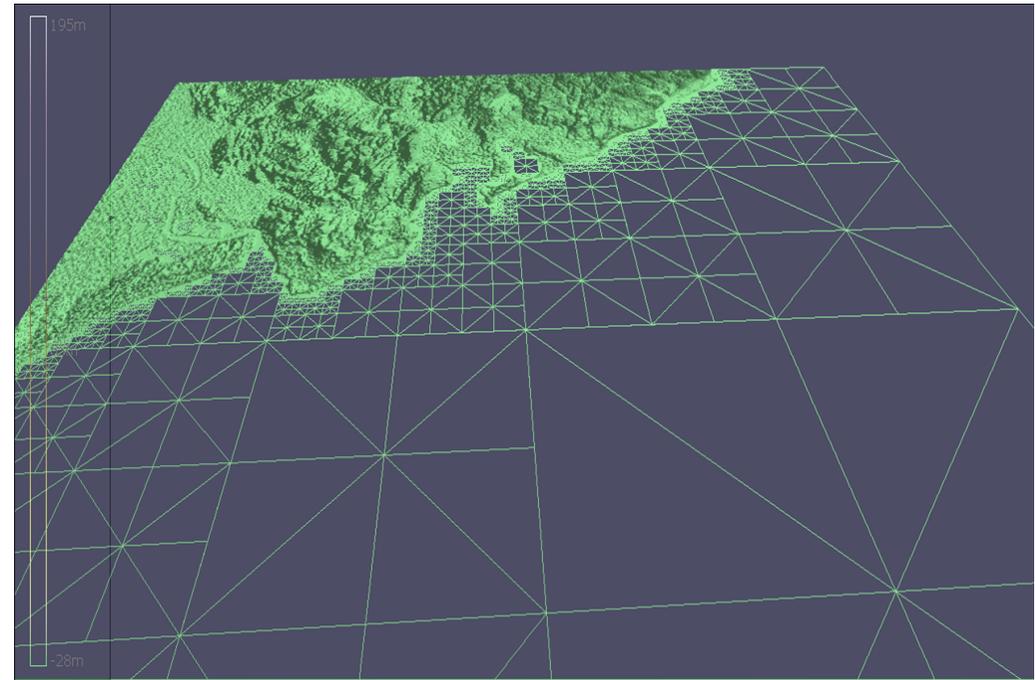
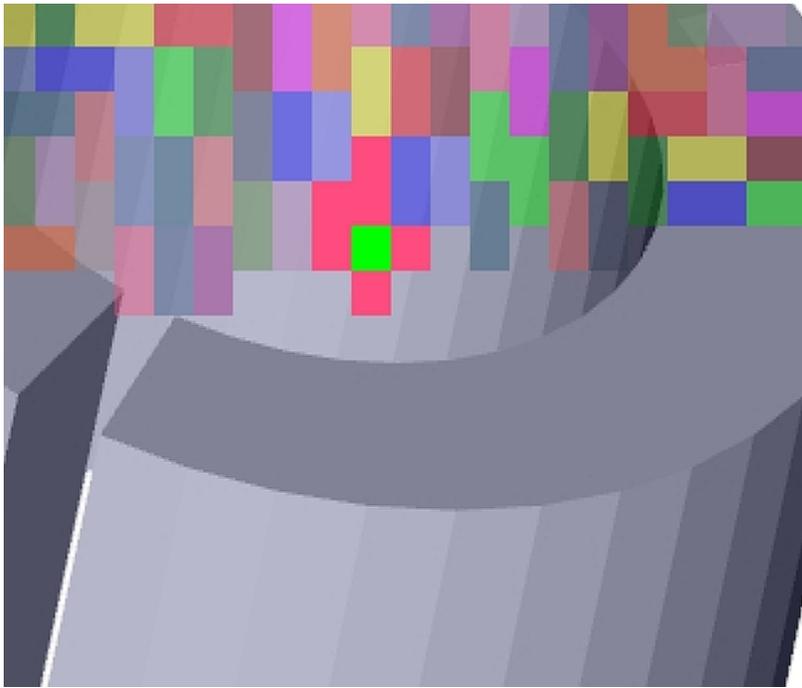
- Example, streamline tracing on unstructured grid.
- Involves three basic computations: (1) point location, (2) integration, (3) interpolation.
- Point location can be source of bugs.
- Test and visualize: (1) current line segment and (2) current mesh edge at run time



2. Visualize Data Structure Traversal and Evolution

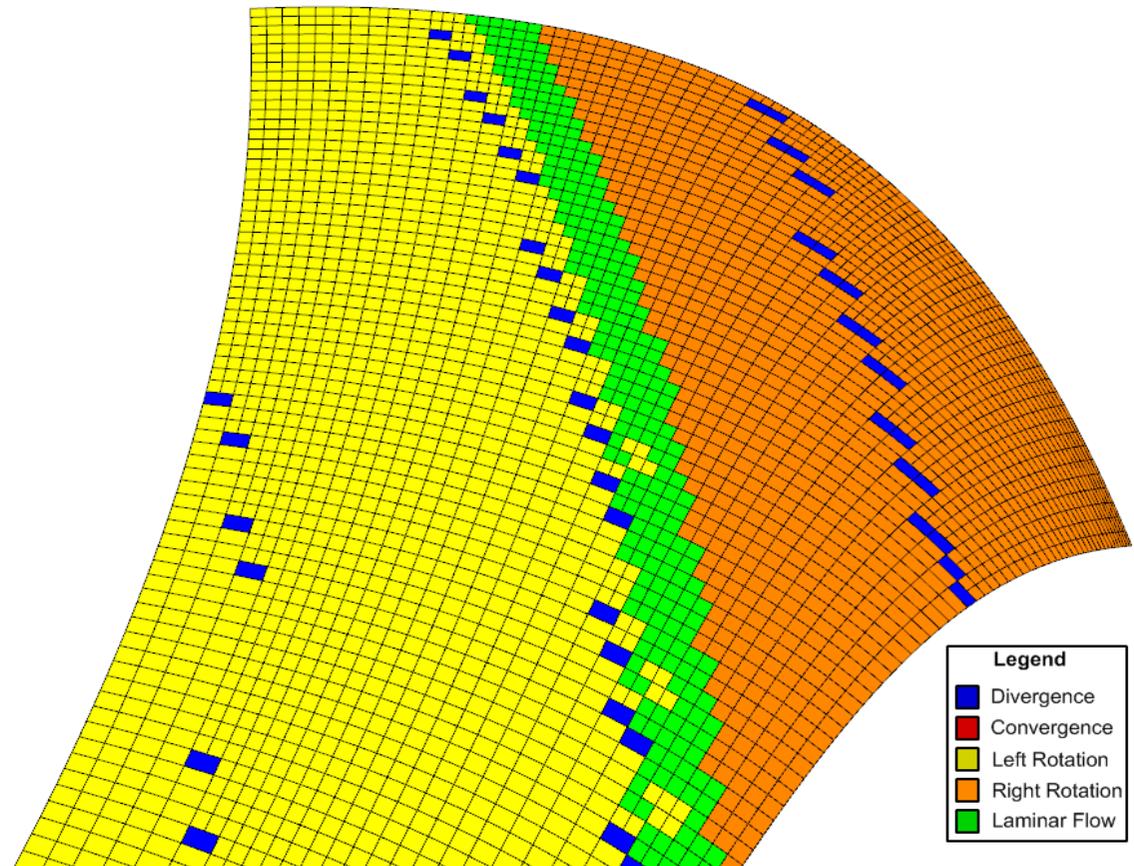
Almost all algorithms involve one or more data structures

- Visualize data structure traversal and evolution, i.e., as it's being built
- Examples: binary tree for clustering, quadtree for AR height map data



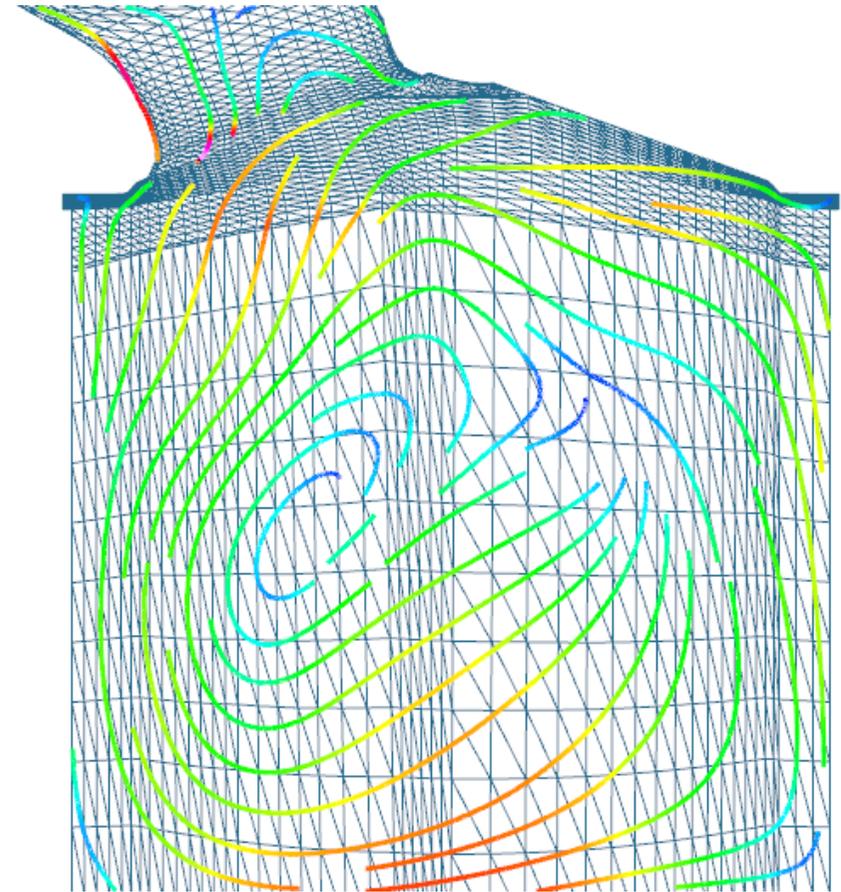
3. Classify and Color Map

- Classify and color map geometric primitives in visualization
- Data has different attributes and values, can be used for debugging.
- Example, quads in stream surface
- Example: Particle color to (1) energy level, (2) type, (3) amount of repulsion, etc. [Crossno and Angel '99]



4. Incorporate Algorithm Parameters into User Interface

- New algorithms introduce new parameters: e.g., threshold values, min and max values, alpha values, special distances, etc.
- Identify, discuss, and visualize these new parameters over range of values.
- Best value of parameter is generally unknown *a priori*, thus they are incorporated into user interface.
- Example, d_{sep} (Jobard and Lefer '97)



5. Run Simple Error Checks

Don't forget to run simple, sanity checks on visualization primitives and data structures.

- A simple, generic error checking procedure can run through data objects and check basic properties, e.g., point locations, edge lengths, minimum, average, and maximum data values, and boundary conditions
- Testing to see if all of object's attributes are within reasonable, expected bounds.
- A general error checking function can be invoked at any time throughout vis pipeline to catch updates that cause unexpected changes.

5. Run Simple Error Checks, Example

Example 1: an isosurfacing algorithm for adaptive resolution data:

Cracks appeared (only) at 128^3 resolution data or greater.

- To track down error, we implemented simple, generic, error-checking procedure that examined:
 - (1) locations of triangle vertices—testing to see if they fell outside of their associated cube and
 - (2) maximum and minimum data values of each node in octree to ensure that all child values fell within this range.
- Values were re-computed and compared to stored values.
- This function traversed entire octree and could be called at any time.

Example 2: Render normals, discover lighting and shading bugs

6. Introduce a Step Function

Algorithms involve computation that iterates several times, perhaps over each data item, and again over each pass over data structure.

- Incorporate feature in user-interface that interrupts (or pause) execution between each iteration of algorithm processing.
- User can pause current scene and look in more detail. Pressing pause button again then executes exactly one loop of algorithm.
- User may then step through program execution at run time, one iteration (or loop) at a time.

Example: while developing a flow visualization algorithm we introduced step function that can stop program execution after each time-step.

7. Make Use of Still-Image Driven Animation

Some bugs occur very infrequently, e.g., only after several time-steps have been processed and visualized.

You may be watching an animation of your visualization in action and notice bug(s) only after several seconds (or even minutes).

- Point in time at which the error is recognized may come too late in order to slow down or interrupt algorithm, e.g., by invoking step function.
- Stopping process and starting all over is painful and time-consuming.

Use feature that saves still images of the visualization each time frame buffer is updated.

7. Make Use of Still-Image Driven Animation, Example

We recommend a user option to your software or system that automatically:

- (1) re-sizes the viewer to 512² pixels and
- (2) saves each frame as a still image in JPEG (or PNG) format.

The still images are used as input to an application which can play them back. We use Adobe Premiere or Video-Mach

(<http://www.gromada.com/videomach.html>).

- We consider this as a standard feature: also used to make movies.

8. Test Algorithms on a Variety of Data Sets

- Test algorithm on small, familiar data sets first
- If possible, create synthetic data with known characteristics
- Find smallest data set that creates problem
- Test algorithm on big, complex data sets second.
- Test on a variety of data sets.

Obvious, but still not carried out.

9. Exploit and Compare with Previous Literature

Chances are, you are not first person to visualize a given data set.

- Compare visualizations you create with your predecessors.
- Simple and obvious, however, we witness colleagues overlook this strategy fairly regularly.
- Communicate with those who have already worked with a given data set.
- Colleagues are happy to share their experiences and share important information that was not published.

10. Make Exclusive Use of Accessor Methods

All class member variables are accessed through accessor methods, i.e. `Get()` and `Set()` methods, e.g., `GetClassVariable()`, `SetClassVariable(int newValue)`.

We advocate no exceptions to this rule. The use of accessor methods:

- Enforces encapsulation, makes implementation easy to change,
- If class variable assignment is performed exclusively through `Set()` methods, then objects are always in valid state.
- `Set()` methods perform error and bounds tests and sanity checking on parameters passed to the procedure.

Following this convention leads to very robust code.

11. Follow Coding Conventions

Big projects require multiple, coordinated developers over years. And, applications should not generally be started from scratch.

- But, we in software development start projects from scratch over and over again—repeatedly re-inventing wheel.
- Source code that does not follow any conventions and is not very legible. → Quickly turns into legacy code.
- Writing illegible code is easy and is generally default.
- We have encountered numerous instances of programmers who cannot read their own code.

11. Follow Coding Conventions, Example

Bob's Concise Coding Conventions (available online) are:

- Influenced by and drawn from other coding standards and guidelines including VTK, Sun Microsystems, Meyers, and Dickheiser.
- Concise so they can be printed out and hung up for ease of use.
- Basic philosophy: code legibility should be maximized.
- Maximum legibility leads to minimum number of bugs.
- Maximizing legibility also helps maximize code re-use, good design, and flexibility.

Summary and Conclusions

V&V in visualization is vital but often neglected

Traditional debugging tools de-couple information from spatio-temporal domain in which unexpected problems occur.

We discuss a software engineering approach to V&V inspired by development experience in both industry and academia

For more information, please see:

R. S. Laramée, **Using Visualization to Debug Visualization Software**, in *IEEE Computer Graphics and Applications (IEEE CG&A)*, Vol. 20, No. 6, Nov/Dec 2010, pages 67-73 (available online)

Acknowledgements

We thank the following people for their contributions:

- Eugene Zhang of Oregon State University for valuable discussions.
- Dan Lipsa of Armstrong Atlantic State University for valuable proof reading.
- Zhenmin Peng, Tony McLouglin, and Edward Grundy of Swansea University for images.
- Research supported partly by EPSRC Grant EP/F002335/1

Thank you for your attention.
Any questions?