Cover Letter

In this revision I have endeavored to address all of the reviewers’ comments. Below is my response to each comment and a description of the changes made.

Reviewer: 1

It seems that the listing of keywords a bit sparse. Given that it is a survey paper I would expect there to be many more.

I’ve added keywords for most of the concepts introduced in the paper. Hopefully I have not gone overboard by introducing too many keywords.

The figures are very simple but certainly adequate. They could be polished for a better final paper. (This is optional, it's a stylistic preference.)

I replaced much of the high frequency boarders with light fills and subtle gradients. I also used color more to separate concepts of the pipeline structure and other features such as data or execution spaces (helping to separate basic pipeline structure from implementation detail as discussed in response to Reviewer 2).

I found the conclusion section to be less than satisfying. I was looking for a paragraph on future changes (e.g., effects of approaching exascale, etc.). As a survey paper this may not be appropriate but it would be nice to know what the author thinks about future research directions.

The conclusion is increased 670% and now contains wildly accurate predictions about the future.

I'm not sure that Figure 4 is correct. The box encapsulating the sub-pipeline "Reader 2->Filter 4->Filter 5" are the same for both T=t0 and T=t1. In the description in the text, the author indicates that Reader1 and Reader 2 begin executing together, followed by their downstream pipeline filters.

This is not a mistake. As explained in Section V.A.1: “Once the first reader completes, at time *t1*, both of its downstream modules may begin executing concurrently. The other reader and its downstream modules may continue executing at this time, or they may sit idle if they have completed.” For further clarity, I’ve appended the following parenthetical remark: “(Fig. 4 implies that the Reader 2, Filter 4, Filter 5 subpipeline continues executing after *t1*, which may or may not be the actual case.)”

Here are a couple of recommended changes to the text (nothing critical):

Change the sentence (by replacing the word "thing") from "Throughout this document we have considered the visualization pipeline as a static thing that transforms data" to "Throughout this document we have considered the visualization pipeline as a static construct that transforms data"

Changed.

Change the text: "...grown signiﬁcantly in recent years. Recent studies..." to "...grown signiﬁcantly in recent years. Studies..."

Changed.

Change the text: "However, the constrains imposed" to "However, the constraints imposed"

Changed.

Reviewer: 2

That said, it would be nice if the shortcomings of the pipeline approach were explicitly discussed. One problem revealed by the survey itself is that in actuality the simple pipeline model is inadequate to describe the recent advances in visualization systems. This is true even in the leading open-source toolkits that implement most of the features discussed (like the VTK). There seems to be no standard notation or pictorial representation of the modern visualization pipeline. An important feature of the pipeline architecture is the simplicity of its visual representation. Unfortunately, the notions of an executive, the multiple passes of information flow, flow of metadata (temporal, spatial and contextual) are always depicted in a non-standard fashion in much of software and literature. While this may not be the right place to discuss a new standard or proposal for this, I think the article would benefit from such a discussion.

After giving this some thought, I would like to nuance the comment a bit. I think the simple pipeline model is still a relevant model from a *user* perspective. To this day many applications (e.g. ParaView, Mayavi, VisTrails) use this simplified model to great effect. Even a coder using VTK rarely needs to consider more than this simple model.

The other notions of execution, flow, and concurrency are more of what I would consider implementation details. Diagrams of these features serve to explain the internal mechanisms of the pipeline. As these can change in subtle ways between implementations, not sure how to suggest a general diagram that can be effective. However, it seems to me that such meta-information about the pipeline operation is best presented when overlaid with the familiar basic pipeline representation. I added a paragraph stating this philosophy to Section II and remained consistent with it within the paper.

Minor typo:

Page 3 line 29: a executive -> an executive

Changed.

Minor comment on some references:

The reference for Mayavi is old and not easy to obtain, I believe a more

recent and accessible article is available here:

http://doi.ieeecomputersociety.org/10.1109/MCSE.2011.35

Changed.

Reviewer: 3

My main problem with the "word" pipeline is that different systems, libraries and authors use it to mean substantially different software infrastructure. This can bee seen by the organization of the paper itself. For example, an "event-driven" pipeline is written in a completely different way than a "demand-driven" pipeline. Extending these two models in different directions requires fundamentally different design and software engineering decisions. As a concrete example, one can examine the difference between the VTK pipeline and the "shading pipeline" pioneered by Abram, Whitted and Cook in their publications. The first actually involves passing data around, some of which coming from disk, some coming from processing modules. Abram, Whitted and Cook, on the other hand, use an abstract-syntax tree that ends up being compiled into a shading description of some sort. These are literally nothing like each other aside from being "boxes and connections". VisTrails, as an additional example that this reviewer is more intimately familiar with, encodes boxes as arbitrary Python objects and connections and specific methods of these python objects.

If pretty much anything can become a pipeline, then, the crucial question is: what is this survey really about? It is too much to ask from the manuscript that the survey become about all of visualization, but it seems that most visualization software can be fairly described as being "pipeline-based". To give another example, consider recent infovis libraries such as Protovis and D3. The authors of these libraries would not describe them as pipeline-based, but they do closely follow the model.

To borrow from Protovis's website, read the following snippet of Javascript code:

vis.add(pv.Panel)

.data(data)

.add(pv.Dot)

.left(function(d) x(d.x))

.bottom(function(d) y(d.y))

.strokeStyle(function(d) c(d.z))

.fillStyle(function() this.strokeStyle().alpha(.2))

.size(function(d) d.z)

.title(function(d) d.z.toFixed(1));

vis.render();

It is pretty clearly separating the classic pipeline steps ("load data", "define visual mapping", "assign data properties to visuals", "render").

My assessment, then, is that it might be worthwhile for the author to try an articulate more clearly what separates "visualization pipelines" from "visualization libraries" in general.

To repeat, I do believe a survey of such visualization libraries is welcome and that it would be useful. At the same time, I think the technical contribution of a survey paper is to meaningfully categorize the field, or at least articulate why such a categorization is not possible. I don't quite see that in this paper.

To help clarify the scope of the survey, I added a formal definition of what I mean by a visualization pipeline to the end of Section II. I won’t repeat the whole thing here, but what it really amounts to what you refer as “boxes and connections.” Specifically, the system has to be an embodiment of the pipeline. You have to be able to instantiate objects that represent modules and connect them. And data has to flow through this network that is created. The network cannot be just the structure of the data like a scene graph. I also less formally state that the system must be self-declared as a scientific visualization system. (I am not going to attempt formally defining visualization. You could write an entire survey paper just on papers that try to do only that.)

To elaborate, let’s look at the systems declared in this comment.

The shading pipeline work of Abrams, Whitted, and Cook is not really discussed because it is not technically a visualization pipeline. (It is, however, mentioned in the Introduction as a “similar pipeline structure… in [a] related field.”) But apart from differing target domains, the key difference as you mentioned is that Abrams’ system does a static analysis of the pipeline and builds a static imperative flow. Were this or something like it in the scope covered by this survey, I would probably add a note about this static analysis in Section III.C as a potential way to implement centralized control. Regardless, I would not spend too much time on this as it is fairly non-specific to visualization (whereas much of the other discussion directly relates to the data, structures, metadata, and operations common in visualization).

VisTrails *is* considered a visualization pipeline application. Its python objects behave as modules; they are directionally connected; data flows through the network, and the system manages the execution. VisTrails is clearly identified as an implementation in the Introduction and is the primary implementation behind Section VI.A Provenance.

Protovis does not qualify as a pipeline because although it builds up a network, it does not build up a dataflow network. Potovis builds a tree that represents a hierarchical relationship of data and then marks, which can provide operation on elements in the hierarchy. However, no data flows through this network and the flow of the “classic pipeline steps” are invoked through the imperative commands of JavaScript, not a control flow in the pipeline.

At any rate, I believe that the more formal definition of visualization pipelines added does a reasonable job of scoping the paper.