Computers & Graphics 63 (2017) 50–64

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Falcon: Visual analysis of large, irregularly sampled, and multivariate time series data in additive manufacturing☆,☆☆

Chad A. Steed[a,*], William Halsey[a], Ryan Dehoff[c], Sean L. Yoder[c], Vincent Paquit[d], Sarah Powers[b]

[a] Computational Sciences and Engineering Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA
[b] Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA
[c] Materials Science and Technology Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA
[d] Electrical & Electronics Systems Research Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA

ARTICLE INFO

Article history:
Received 11 October 2016
Revised 21 January 2017
Accepted 8 February 2017
Available online 16 February 2017

Keywords:
Visual analytics
Information visualization
Time series data
Additive manufacturing
Exploratory data analysis

ABSTRACT

Flexible visual analysis of long, high-resolution, and irregularly sampled time series data from multiple sensor streams is a challenge in several domains. In the field of additive manufacturing, this capability is critical for realizing the full potential of large-scale 3D printers. In this paper, we propose a visual analytics approach that helps additive manufacturing researchers acquire a deep understanding of patterns in log and imagery data collected by 3D printers. Specific goals include discovering patterns related to defects and system performance issues, optimizing build configurations to avoid defects, and increasing production efficiency. We introduce Falcon, a new visual analytics system that allows users to interactively explore large, time-oriented data sets from multiple linked perspectives. Falcon provides overviews, detailed views, and unique segmented time series visualizations, all with adjustable scale options. To illustrate the effectiveness of Falcon at providing thorough and efficient knowledge discovery, we present a practical case study involving experts in additive manufacturing and data from a large-scale 3D printer. Although the focus of this paper is on additive manufacturing, the techniques described are applicable to the analysis of any quantitative time series.

1. Introduction

The ability to find and thoroughly understand patterns in time series data is a fundamental requirement in many domains. With small to moderate size data sets involving a few variables and regular sampling intervals, basic graphs and statistical calculations are effective at revealing important features. However, these techniques are inadequate for analyzing time series data from multiple sensors that are long (multiple days), large (millions of data points), multivariate, and irregularly sampled. Such is the scenario faced daily by researchers in the field of additive manufacturing, where large-scale 3D printers are used to synthesize complex objects for industrial purposes.

Recent advances in additive manufacturing have improved production value by removing traditional manufacturing constraints and providing unprecedented geometrical freedom. The Oak Ridge National Laboratory (ORNL) Manufacturing Demonstration Facility (MDF) is at the forefront of this disruptive technology. Using multiple large-scale 3D printing systems, such as the Arcam Q10 system shown in Fig. 2, MDF researchers execute builds of complex objects and study both the printer log files and microstructure of the printed objects (see Fig. 1) to make fundamental scientific contributions related to the efficacy of the 3D printing process. The results of these investigations also enable the construction of unique prototypes, namely aerospace components, advanced robotics, and automobiles.

The key to realizing the full potential of additive manufacturing lies in providing researchers with intuitive tools that support exploratory analysis of the data produced by 3D printing systems. However, the size and complexity of the data exceed the

☆ This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

☆☆ This article was recommended for publication by Prof H Schumann.

* Corresponding author
E-mail addresses: csteed@acm.org, steedca@ornl.gov (C.A. Steed).

http://dx.doi.org/10.1016/j.cag.2017.02.005
0097-8493/

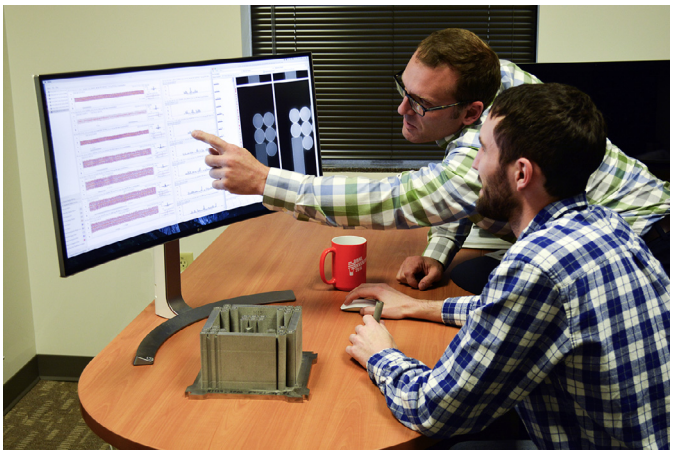

Fig. 1. Falcon is a visual analytics system used daily by additive manufacturing researchers to explore log and imagery data from 3D printers. Here two additive manufacturing researchers from the ORNL Manufacturing Demonstration Facility (MDF) are using Falcon on a curved, wide-aspect display to take advantage of the horizontally oriented time series visualizations. Having the printed objects on the desk, the researchers are able to supplement the data analysis with physical examinations.

capabilities of most general purpose data analysis systems. To address this capability gap, we have developed the Falcon visual analytics system using a collaborative, participatory design process with additive manufacturing experts. Although we apply Falcon to additive manufacturing in the current work, it is useful for any problem that involves the analysis of quantitative, time series data.

From a visual analytics perspective, Falcon combines several interactive data visualization techniques that extend the scalability of conventional time series analysis methods. The system design is inspired by the visual information seeking strategy [1] where zooming and filtering operations permit the descent from overviews to detailed data visualizations. Using both time-oriented and statistical views, Falcon coordinates user interactions in multiple visualizations to allow comparative visual analysis of long time series data that are sampled irregularly with subsecond precision. The system also utilizes the concept of information scent [2] where quantitative values from time series similarity and statistical algorithms are visually encoded within the user interface to highlight interesting relationships and reduce the search space. The coalescence of these capabilities into a visual analytics system helps researchers acquire a deeper understanding of 3D printing systems while reducing knowledge discovery timelines—the central promise of visual analytics.

1.1. Contributions

In the current work, we describe a system that addresses several key challenges faced in the exploratory analysis of data from 3D printers. This system has evolved based on an iterative design process with additive manufacturing researchers, who are also co-authors on this paper. The main contributions of the current work include the following:

- We describe the Falcon system, which goes beyond previous systems to support scalable exploratory analysis of long and complex time series data involving multiple variables and irregular sampling.
- We describe a new visualization technique, called the waterfall visualization, that combines overview and detail using miniaturized graphics.
- We describe a new segmented time series visualization technique that provides synchronized views of time series and imagery data with visual representations of information scent derived from time series similarity algorithms.
- We present a practical evaluation of Falcon in which an additive manufacturing researcher analyzes real 3D printer data. To the best of our knowledge, this paper documents the first application of visual analytics to the field of additive manufacturing and it represents an important success story of applying visual analytics to a challenging and significant domain. Key findings from this case study, which represents the daily workflow of an additive manufacturing researcher, have led to new knowledge about the 3D printing process.

1.2. Outline

After a survey of related work in Section 2, we provide background on additive manufacturing in Section 3 followed by an overview of key challenges for analyzing 3D printer data in Section 4. In Section 5, we describe the visual analysis techniques provided by the Falcon system. Then, in Section 6, we present a real case study in which 3D printer log data is analyzed. In Section 7, we reflect on key observations related to analytic workflows, our interdisciplinary development, and domain specific

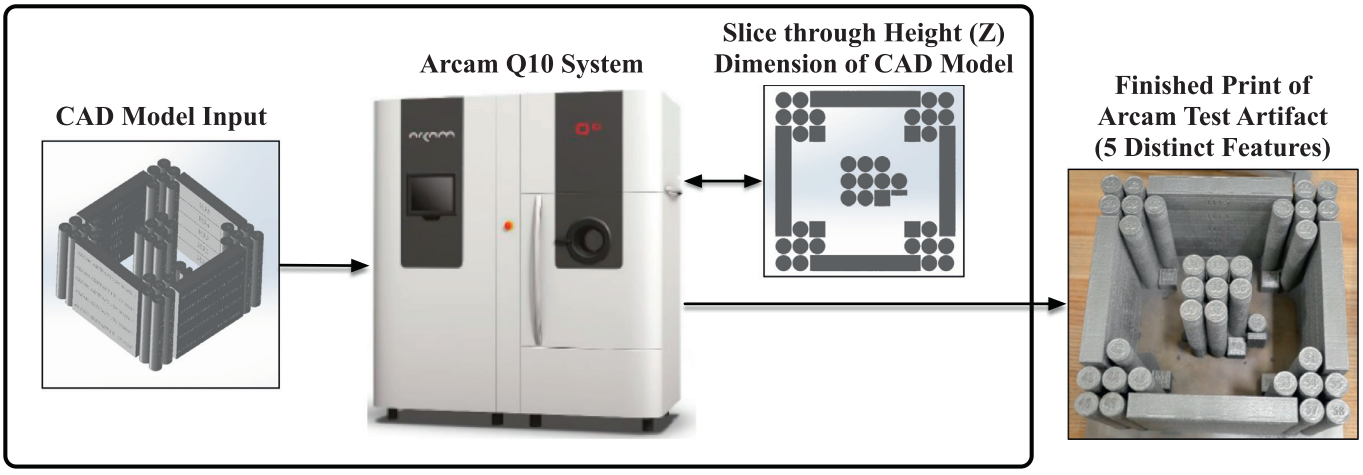

Fig. 2. The Arcam Q10 3D system is a large-scale 3D printer that uses electron beam melting to build metal objects. Each layer is melted using the geometry defined by slices taken through a CAD model's z dimension, progressing from the bottom to the top of the build. This technology is transforming the manufacturing industry by removing many traditional manufacturing constraints and giving designers greater geometrical freedom.

capabilities. Finally, in Section 8 we describe future work and in Section 9 we summarize our results.

2. Related work

Due to the copious nature of time-oriented data, a large body of work exists on time-based visualization techniques as evidenced by recent reviews of the field [3,4]. Some systems are designed with a very specific use case in mind to maximize the discovery of time-based insight. For example, many social media visual analytics systems (e.g. Leadline [5], Matisse [6], Visual Backchannel [7]) incorporate some form of time-based visualization as the central view with other linked views serving to supplement the temporal patterns. Social media visual analytics systems are a specific form of text visualization systems (e.g., ThemeRiver [8], EventRiver [9], and TextFlow [10]), which also tend to include a prominent time-based visualization component. But time-based analysis permeates nearly all domains, namely, climate [11,12], cyber security [13], and parallel computing performance monitoring [14]. Falcon can be classified as a domain specific system with general applicability to any quantitative time series data.

In addition to domain based systems research, a substantial segment of time-based visualizations are devoted to generic techniques that can be applied to a range of data sets. Most of these techniques represent time using line plots or bar charts with either variations on the graphical representation or enhancements that allow interactive visual queries. Examples of line plot variations that are designed to increase the number of comparable time series include small multiples [15] and sparklines [16], horizon graphs [17], and braided graphs [18].

To provide multi-scale views, van Wijk [19] used a calendar-based visualization to analyze time series data aggregated on a daily, weekly, or monthly basis via a similarity clustering method. Spiral layouts of the time axis are used in the SpiralGraph [20] and SpiralView [21] techniques. These spiral layouts perform poorly with long time series, but excel at showing recurring patterns. VizTree [22] presents a very unique representation of time series data that uses a sequence of symbols and a suffix tree. Although VizTree may be helpful for very large data sets, the view can be difficult to decode, especially for fledgling users. The TimeBench library [23] describes a general data model for time-oriented data to support data visualization through multi-scale data structures. In Falcon, common visual mappings are used for the visualizations to ease adoption among non-visualization experts. However, the waterfall visualization technique includes some subtle expressiveness similar to the more abstract visual representations mentioned above.

Buno et al. [24] introduce a visualization technique in the TimeSearcher 2 system that helps users see statistical summaries of time series data. The method renders the minimum and maximum range for each time record and a central line to show the mean value. This overview visualization is similar to the time-oriented summary visualizations presented by Brinton [25], Tufte [26], and Bade et al. [27]. In Falcon, we use an extended version of these statistical representations to visualize an overview of time series data. The TimeSearcher 2 system shows multiple time series simultaneously and provides the ability to search for similar patterns. Falcon also offers simultaneous views, pattern matching capabilities, and it provides additional support for visualizing irregularly sampled data.

Some time series visualizations use lens-based techniques to magnify time ranges of interest by distorting the time axis. Kincaid et al. [28] introduced SignalLens which magnifies an area of interest and compacts the areas on either side to maintain context. Walker et al. [29] introduced the RiverLens technique, which combines the SignalLens [28] and the River Plot [30]. Brushing time ranges expands the time series plot and shows an overlay on the River plot. A River Plot is shown on either side to provide context. Introduced by Zhao et al. [31], ChronoLens is an interactive visual analytics system that includes a lens-based technique to support elaborate analysis tasks. Traditional time series plots are augmented with lens distortions based on user selections. In addition, derived data, such as derivatives and moving averages, can be shown on the graphs and zooming, resizing, and movement operations are applied to the lens to alleviate occlusion.

Hao et al. [32] introduce an interest-based visualization using a layout of time series plots that reveals hierarchical relationships. Similarly, Stack Zoom is a multi-focus zooming technique described by Javed et al. [33] that maintains context and temporal distance during zoom operations. User selections produce a hierarchy that is represented in a nested tree layout, which also serves as a graphical history.

Instead of lens-based techniques, Falcon uses multiple views and multi-scale zooming due to problems that may occur with distorting the time axis. Plaisant et al. [34] introduce the concept of overview and detail displays, which provide simultaneous views of a focus visualization and an overview of the entire data series. The overview provides context for the focus visualization and users can brush areas of interest for detailed investigation. Falcon also provides zoom-and-filter interactions in the time series plots and fine-grained control over the time scale used in the visualization. As the zoom level is increased, the width of the time series graph is expanded while the display viewpoint remains constant. Scroll bars allow the user to freely navigate the expanded time series.

3. Background

To realize the full potential of additive manufacturing, researchers seek a deeper understanding of the 3D printing process (also known as a build). Specific goals include discovering patterns that indicate defects or system problems, optimizing build configurations to reduce the occurrence of defects, and reducing production costs. In addition to microscopic study of the actual 3D printed objects, researchers must carefully study the log files and imagery data captured by the 3D printer during builds to achieve these goals. Prior attempts using conventional tools (e.g., R, Excel, and proprietary log analysis software from printer manufacturers) have failed to provide efficient exploratory data analysis capabilities.

Our interdisciplinary team postulates that a visual analytics approach will reduce knowledge discovery timelines, improve the accuracy of additive manufacturing data analysis, and provide a deeper understanding of the 3D printing process. To practically test this postulation, we have executed a participatory design process to develop Falcon, which allows flexible data exploration with views and interactions that are designed to specifically address the data analysis challenges for this domain. In particular, we have focused on the Arcam Q10 3D printer system, which is shown in Fig. 2, and its unique electron beam melting process. In the remainder of this section, we provide background on the 3D printing process and data generated by the Q10 printer.

3.1. 3D printing process overview

As shown in Fig. 2, a build begins when a Computer-Aided Design (CAD) model is uploaded to the system. Beginning at the bottom, the system prints each layer by extracting slices through the height (or z) dimension of the model. For each layer, the system's rake mechanism prepares a layer of metal powder 50 microns (0.05 mm) thick. Then, the system completes a sequence of stages to synthesize the layer. During the melt stage, an electron beam melts the powder in the desired shape as defined by the

```
492258  2016-06-17 21:07:03.466|OPC.Temperature.ExtraTemp1|SuperUser (OPC)|9526586|618
492259  2016-06-17 21:07:03.466|OPC.Temperature.BottomTemperatureValidation|SuperUser
        (OPC)|9526586|618.19999999999982
492260  2016-06-17 21:07:03.486|Process.CathodeTuningControl.CathodePower|[OnChange(
        OPC.PowerSupply.Filament.VoltageFB)] Arcam.EBMControl.Process.
        CathodeTuningControl.OnCathodPowerChange() (Logic)|9526586|5.929202
492261  2016-06-17 21:07:03.516|Process.CathodeTuningControl.MeanCathodePower|[
        OnChange(Process.CathodeTuningControl.CathodePower)] Arcam.EBMControl.Process.
        CathodeTuningControl.MeasureMeanPower() (Logic)|9526587|5.905572
492262  2016-06-17 21:07:03.516|OPC.PowerSupply.HighVoltage.SafetySignal|[
        OnPositiveFlank(SafetySignalTimer.Timeout)] Arcam.EBMControl.Process.
        HighVoltageControl.OnTimeToSendSafetySignal() (Logic)|9526587|True
492263  2016-06-17 21:07:03.546|OPC.PowerSupply.SmokeDetector.Counts|SuperUser (OPC)
        |9526589|1
492264  2016-06-17 21:07:03.546|OPC.Temperature.ExtraTemp1|SuperUser (OPC)|9526589|598
492265  2016-06-17 21:07:03.546|OPC.Temperature.BottomTemperatureValidation|SuperUser
        (OPC)|9526589|598.09999999999991
492266  2016-06-17 21:07:03.666|OPC.Table.CurrentFeedback|SuperUser (OPC)|9526590|0.
        1153
492267  2016-06-17 21:07:03.666|OPC.Rake.CurrentFeedback|SuperUser (OPC)|9526590|0.
        1153
492268  2016-06-17 21:07:03.766|OPC.PowerSupply.SmokeDetector.Counts|SuperUser (OPC)
        |9526594|44
492269  2016-06-17 21:07:03.766|OPC.Temperature.ExtraTemp1|SuperUser (OPC)|9526594|544
492270  2016-06-17 21:07:03.766|OPC.Temperature.BottomTemperatureValidation|SuperUser
        (OPC)|9526594|544.40000000000009
492271  2016-06-17 21:07:03.786|OPC.InternalCooling.DifferentialPressureOverFilter|[
        OnChange(OPC.InternalCooling.PressureBeforeFilter_Unfiltered)] Arcam.
        EBMControl.Process.InternalCooling.UpdateDifferentialPressureOverFilter() (
        Logic)|9526594|0.02296734
```

Fig. 3. During a 3D print, the Arcam Q10 system collects sensor and diagnostic data in a textual log file. The file extract shown here, which is the subject of the case study in the paper, is from an 8-hr build and it contains 23,977 different variables and over 3 million data values that are recorded with subsecond accuracy and at different intervals. Each line in the file begins with a date/time stamp followed by the variable name, metadata, and the value. Although we currently focus on the quantitative values, categorical and Boolean data types are also stored.

CAD model slice. When finished, the printed objects are removed from the powder and unused material is recycled. This process enables the production of objects with complex geometries that require neither additional tooling or fixtures, and it avoids the production of significant amounts of waste material.

3.2. 3D printer data description

During a build, the Arcam Q10 system records sensor readings and diagnostic data in a textual log file (see Fig. 3). Reconstruction of a build for analytic purposes can only be accomplished using this data. A single 3D printer log file contains readings from thousands of different sensors and diagnostic modules. The readings are captured and stored during the build, which may take several hours and sometimes multiple days to complete. The data values are stored with time stamps at sub-second accuracy and irregular time sampling intervals.

When a build is initiated, an initial reading for each variable is recorded in the log. Thereafter, new variable values are only recorded as sensor and diagnostics report changes. As a result, except for the first time stamp, most time records will have only one variable value. Some variables contain tens of thousands of values, others contain hundreds, and some may have less than ten. Furthermore, although most of the variables are quantitative and we currently only focus on these values, some variables are Boolean and others are categorical. In addition to the textual log file, the printer captures a single near IR image at the completion of each build layer.

For a typical build, the total disk space that is required to store both the log and imagery data is approximately 50 GBs. Researchers commonly compare features between multiple builds, sometimes dozens and more recently hundreds. Therefore, the total size of data under investigation can easily reach multiple TBs and exceed the limits of most conventional analysis tools.

4. Key challenges of analyzing 3D printer data

Providing efficient exploratory analysis capabilities for 3D printer data is challenging for several reasons. Our interdisciplinary design team, which includes additive manufacturing experts who regularly use Falcon to analyze their data, have identified these challenges and used them to form design requirements that have guided the participatory design and development of Falcon. In the remainder of this section, the most critical challenges and associated requirements are described.

4.1. The exploratory nature of the field

Due to rapid technological advances in the additive manufacturing field, the questions that are asked of the data are often too exploratory for a completely scripted analytic workflow. Therefore, researchers require tools that allow them to interactively ask questions of their data, while not confining them to the original ideas that prompted the data collection. Indeed, historical reflections upon some of the most significant scientific discoveries show that profound findings are often unexpected [35]. Furthermore, the exploration process is iterative as new discoveries lead to new questions and the analytical discourse continues in a cycle until the researcher is satisfied.

Prior to the development of Falcon, researchers used a collection of analysis tools (e.g. R, Excel, and proprietary log viewers) to explore the data. Although powerful, limited interactive visual analysis capabilities forced researchers to switch between different static views and use scripting interfaces to ask questions of the data. Consequently, true exploratory capabilities were never achieved, analytics sessions stagnated, and knowledge discovery time cycles were delayed.

4.1.1. Requirement 1 (R1): support exploratory analysis

To provide true exploratory data analysis capabilities, human-centered, visual interactions are required that give the researcher control over the direction of the analysis process. These interactions must be coordinated across multiple views of the data to allow researchers to highlight patterns from multiple perspectives. Furthermore, the interactions should feed automated analytical algorithms to highlight similar temporal and statistical patterns automatically.

4.2. Finding patterns in long and complex time series data

Time is the primary dimension for analyzing log data, especially with 3D printers where known event sequences are studied for each layer of the build (e.g., initialization, material preparation, melting). Consequently, additive manufacturing researchers usually begin with time-based data visualizations. However, conventional systems cannot cope with the length and complexities of the 3D printer log data (see Section 3.2) without drastic data reductions, which often blur key patterns.

Additive manufacturing researchers are usually interested in patterns that are visible at different time scales. Macroscale patterns, such as the relative duration required to print each layer, are visible in overviews of the entire data set. Conversely, microscale features, such as system circuit resets, only appear at the finest levels of granularity.

4.2.1. Requirement 2 (R2): support multi-scale temporal visual analysis

To enable temporal data analysis, the user requires the ability to retrieve multiple visual encoding techniques that emphasize the sequential order and relative changes for temporal values, including detailed segmentation techniques. Overviews are necessary to show broad trends and help the user maintain context for selections in other detailed views. As the use of overviews implies some form of aggregation, the loss of information in these views should be offset by displaying variability measures with the aggregated visualization. In detailed temporal visualizations, the user requires

the ability to interactively adjust the time scale using the hierarchical nature of time (e.g., hours, minutes, seconds) to flexibly access the most appropriate detail level for particular features.

4.3. Supplemental statistical analysis

Descriptive statistics and their visual representations reveal broad trends in large data sets, and additive manufacturing researchers commonly employ statistical visualizations to study the 3D printer logs. They primarily rely on visualizations of value distributions and descriptive statistics. In addition to broad trends, visual statistical representations help scientists identify normal variation ranges, detect outliers, and form patterns for certifying the quality of 3D printed objects.

Prior to the development of Falcon, researchers created static statistical plots disconnected from the time-based visualizations, which precluded holistic analysis due to multiple disjoint views. Furthermore, researchers struggled to dynamically create views that focus on different scales or time ranges of interest due to a reliance on scripted analysis frameworks.

4.3.1. Requirement 3 (R3): support supplemental statistical analysis

Additive manufacturing researchers require interactive statistical visualizations, which encode frequency and summary information. These visualizations should supplement the time series visualizations to provide additional insight, and they should include the ability to access multiple scales.

4.4. Multivariate comparisons across multiple views

Additive manufacturing researchers require the ability to quickly compare both univariate and multivariate patterns across multiple views. As researchers formulate specific criteria for certifying build quality, unique view combinations and interactive manipulation of view parameters are necessary, but difficult to construct and analyze, using conventional tools. For example, a researcher may wish to study time series and statistical views for dozens of variables at different time ranges to identify and understand patterns that indicate defects. As conventional tools lack the flexibility and linkages to quickly build these views, researchers are forced to view multiple static plots independently, which can delay the formulation of insight due to human memory limitations from one glance to the next [36].

Linking techniques alleviate these issues by interactively relating information between different views [37,38]. Brushing and linking is the most common view coordination strategy [39], in which interactive selections in one view are propagated to other views, where corresponding items are highlighted automatically. The linkage helps users form a more complete mental model of the data by combining input from different representations that emphasize particular features.

4.4.1. Requirement 4 (R4): support multivariate comparisons across multiple views

To support comparative analysis across different views and multiple scales, multiple visualizations of 3D printer build data are required where each view emphasizes a particular aspect (e.g. statistical versus temporal, different time ranges, different scales). In response to user interactions, each view should update its visual representation and propagate the changes to other linked views, which should also be updated.

5. Introducing Falcon

The Falcon visual analytics system combines interactive data visualization, automated statistical analytics, and data processing routines to form a human-centered interface that supports the formulation of hypotheses in complex time series. Falcon is an open source, standalone application implemented entirely in the Java programming language and it is suitable for use on a laptop or workstation computer. The system uses the JavaFX graphics API, which utilizes system GPUs, when present, for faster rendering performance. In the remainder of this section, we provide a technical description of the main features that comprise Falcon.

5.1. Scalable data processing and organization

Before analysis can begin, the 3D printer log data must be converted from its native text format into organized data structures. As described in Section 3.2, the data are large, irregularly sampled with sub-second accuracy, multivariate, and include near IR imagery captured for each build layer. Often researchers will compare data from multiple builds and the size of data under investigation can reach multiple TBs. Therefore, Falcon requires efficient memory management and indexing schemes to enable an interactive user experience.

Values for each variable are stored and indexed in a time series data structure, which extends the java.util.TreeMap class from the Java Standard Edition 8 library[1]. As an adaptation of the algorithms described by Cormen et al. [40], the Java TreeMap class is implemented as a red-black tree data structure with guaranteed searching in $\mathcal{O}(\log n)$ time. Raw data values are added to the TreeMap using the time stamp as the key and an array for the data values. Data values are stored in an array to deal with situations where multiple values are reported for the same variable at the same time. This data structure greatly increases the rendering performance for large time series as subtrees are quickly formed based on the current visible range of values.

To support the creation of overview visualizations in Falcon, a binned variation of the time series data structure is utilized. The bins cover equally spaced time ranges for the full time series and the Java TreeMap class is again used to store and index the bin objects. The start time instant is used as the key for the bins, and each bin stores references to the values that fall within its time range. Each bin also computes and stores a set of key descriptive statistics for its contents (e.g., mean, standard deviation, quantiles, min/max range). This same approach is used to create statistical visualizations in Falcon using the value space, instead of the time dimension, for the bin construction. Both the frequency-based and time-based data structures support variable scale temporal binning at different levels of detail. For example, the user can reconfigure the time axis to show the data at hour, minute, or second intervals. The variable scale summaries allow users to descend to more granular time measurements for greater fidelity as needed.

Most researchers confine their analyses to a particular set of variables. Except for metadata information, Falcon waits for the user to add a variable into the visualization panel before loading its data values. By loading information on-demand, Falcon requires a smaller memory footprint as compared to loading the entire file contents when the file is first opened. Also, by scanning the log file initially to gather metadata, the caching mechanism in most operating systems will ensure fast access on future read operations.

As shown in Fig. 4, Falcon lists the file variables in a tree panel on the left side of the window. With 3D printer logs, variables names are organized hierarchically by separating the categories with a period character. For example, the variable selected in the tree view shown in Fig. 4, is stored as *OPC.PowerSupply.Beam.BeamCurrent* in the log file. Falcon parses the

[1] Documentation on the Java SE 8 TreeMap class is available online at http://docs.oracle.com/javase/8/docs/api/java/util/TreeMap.html

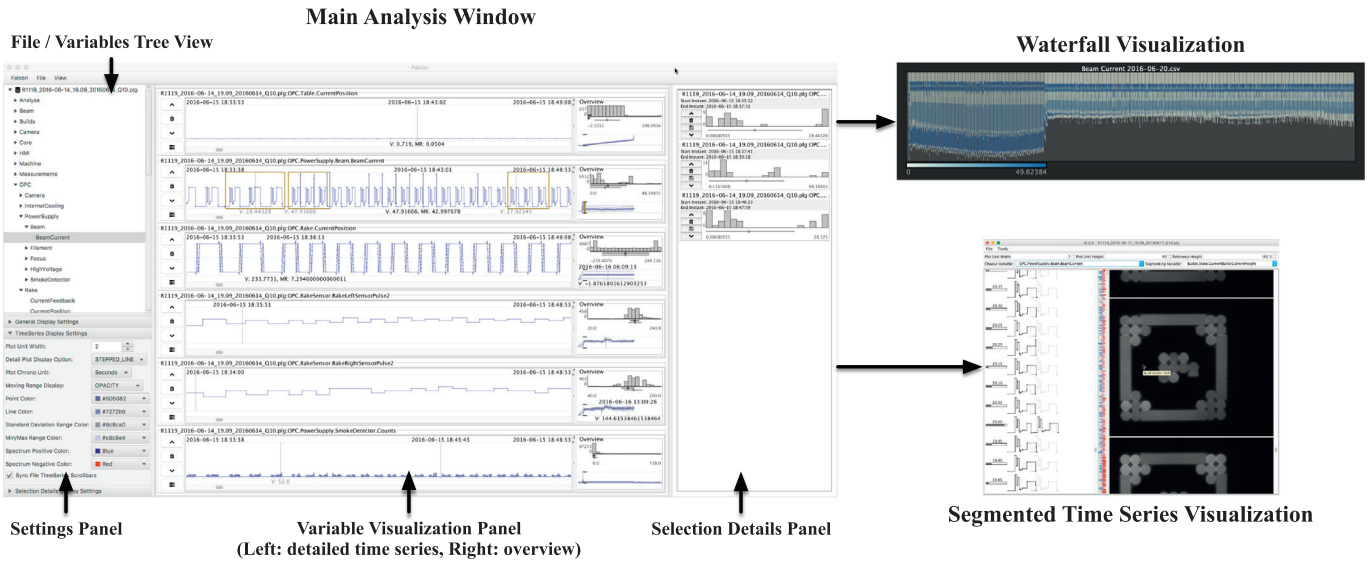

Fig. 4. The main analysis window (left) in Falcon consists of a set of interactive data visualizations that allow users to explore temporal and statistical patterns in time series data for multiple variables. The center variable visualization panel features both overview and detail visualizations. When time range selections are set (see the three rectangles with yellow borders in the *BeamCurrent* variable pane), a statistical summary pane is added to the selection details panel at right to summarize the selected data. From the main window, the user can access two additional analysis windows, which provide more detailed views of segments derived from the entire time series of a particular variable. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

variable names in this form to build the levels of the tree view. When analyzing thousands of variables, the hierarchical organization helps users search more efficiently. When variable names do not use this hierarchical organization, the tree will only have one level, below the file node. For comparative analysis of multiple data sets, Falcon also supports loading multiple files in the same session.

5.2. Falcon user interface layout

As shown in Fig. 4, Falcon consists of a main analysis window and two supplemental analysis windows. The main analysis window, which is designed for variable scale views, shows file variables and displays settings on the right, a variable visualization panel in the center, and a selection details panel on the left. For more detailed analysis, a segmented time series visualization window is available for a variable of interest. In addition to segmenting the time series for the variable into smaller visualizations according to layer heights, the segmented view provides information scent based on time series similarity algorithms. The waterfall visualization window is another segmented view that can be accessed for a variable of interest. The waterfall visualization shows an overview with details using miniaturized graphics. Together these linked views address R1 (see Section 4.1.1) enabling true exploratory data analysis. In the remainder of this section, the visual encodings and interactive techniques used in these analysis windows are described.

5.3. Variable visualization panel

The focal point of the main Falcon analysis window is the variable visualization panel (see Fig. 4), which provides multiple views for a specific variable. When a variable is selected, a new horizontally-oriented visualization pane is added to this panel. All variable panes currently under investigation are stacked vertically in the scrollable panel. We avoid alternative visualization techniques that superimpose the representations of multiple variables within a single plot area (e.g., Bertin's indexing method [41]) because additive manufacturing researchers prefer stacked, independent views that enable comparative analysis of different time ranges for multiple variables with varying units and scales. With the stacked layout, the number of variables that can be analyzed simultaneously is only limited by the resolution of the display device. The order of the panes and their removal from the panel is controlled using the buttons on the left side. This panel addresses R4 (see Section 4.4.1) by providing reconfigurable perspectives of multivariate patterns. As shown in Fig. 5, each pane consists of a zoomable detail time series visualization (left) and two overview visualizations (right). These interactive views are linked so that brushings and other manipulations are shared.

5.3.1. Variable overview visualizations

Fig. 5 shows the overview region of the variable pane, which consists of two visualizations: a statistical visualization (top) and a time series visualization (bottom). These visualizations address both the temporal overview requirement of R2 (see Section 4.2.1) and the statistical overview requirement of R3 (see Section 4.3.1). The statistical visualization displays the frequency distribution and summary statistics of the entire variable distribution. We use a standard histogram plot, which counts the number of values that fall within equally sized bins over the distribution space. From the settings panel, the user can change the number of bins, plot height, minimum and maximum values of the x-axis scale, and maximum count value of the y-axis scale.

As shown in Fig. 6, the bin counts are visually encoded as the height of the bars in the histogram plot. The result is an overview of the distribution for the variable of interest. To access detailed information, the user can hover over a bin (see Fig. 6) to see a tooltip with the numerical values and the bin's upper and lower limits. As the visible time range of the detail time series visualization changes, the histogram plot receives the updated set of visible values and shows darker bars on the histogram to indicate the percentage of the overall bin count.

In addition to the frequency distribution, summary statistics are calculated for both the entire distribution and the visible set of values. As shown in Fig. 6, these summary statistics are visually represented in a statistical visualization technique that is inspired by the box plot [42]. Here the dot represents the typical value (mean or median) and the width of the line represents the dispersion range

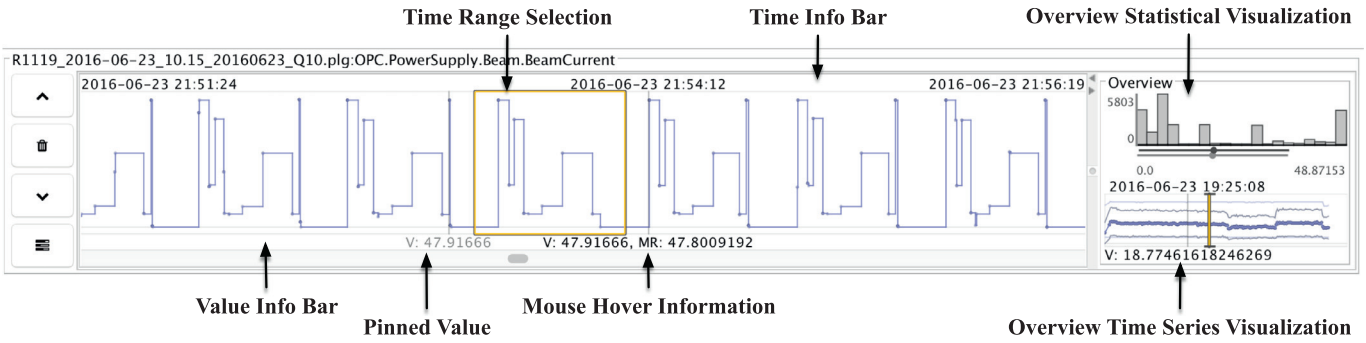

Fig. 5. This figure shows a single variable visualization pane for the *BeamCurrent* variable. At right, two overview visualizations provide statistical (top) and temporal (bottom) context for the main detail time series visualization at left. The detail time series visualization includes an interactive time range selection capability and details-on-demand features through mouse gestures. By interactively adjusting the time scale of the detail visualization, the user can zoom in or out.

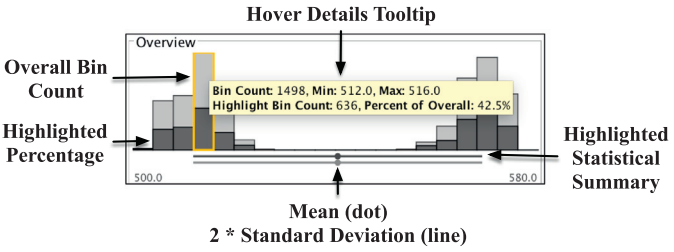

Fig. 6. The statistical view in Falcon encodes the frequency distribution of values using a histogram. The darker portions encode the percentage of values shown in the visible portion of the detail time series visualization. Below the histogram, the mean (dot) and two times the standard deviation range (line width) are represented for both the entire distribution (bottom) and the visible data values (top).

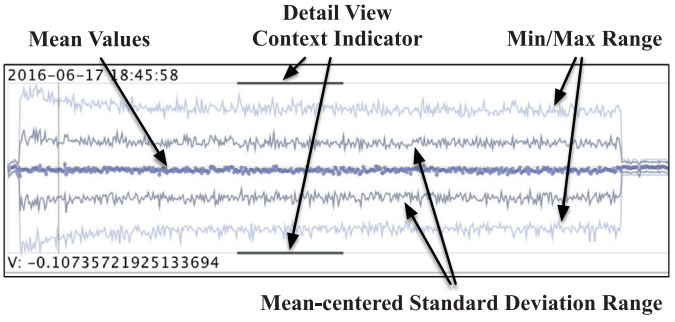

Fig. 7. The overview time series visualization creates an abridged visualization of the full time series. The visualization shows the mean values, and upper and lower standard deviation band, and the minimum and maximum value range. Two black horizontal bars show the relative position of the visible range in the detail time series visualization.

(two times the mean-centered standard deviation range or the interquartile range). The statistics for the set of visible values in the detail time series visualization are also shown using a dark gray color (see Fig. 6).

The overview time series visualization is rendered below the overview statistical visualization (see Fig. 5). In this view, the entire time series is condensed to fit the width of the variable pane overview region. To avoid overcrowding, a statistical aggregation of the full time series is shown. The system computes the maximum and minimum value range, dispersion range, and typical values for equally-sized time intervals (or bins) that cover the full time series. Using these summary values, the system creates an abridged time series visualization (see Fig. 7). By default, the mean values are shown as points that are connected with line segments. The dispersion range is shown by doubling the standard deviation, and centering it about the mean value. The upper and lower bounds for both the dispersion and minimum/maximum value range are rendered as polylines. Alternatively, the visualization can be modified to render the bin medians and interquartile ranges. The overview time series visualization provides context within the entire time series in a compact form. Although some details of the raw values are hidden, the range representations visually encode variability to compensate for the loss of information, thereby addressing requirements from R2 (see Section 4.2.1).

5.3.2. Zoomable detail time series visualization

In addition to the overviews, each variable pane includes a scrollable, detail time series visualization (see Fig. 5). Two user-defined settings control the time scale along the *x*-axis: the *Plot Chrono Unit*, which controls the chronological time unit (e.g., seconds, minutes, hours), and the *Plot Unit Width*, which determines the pixel width of each time unit. These controls permit the user to interactively drill-down, or conversely roll-up, to access different time scale representations. This visualization addresses the exploratory analysis requirements from R1 (see Section 4.1.1) and the multi-scale, temporal visualization needs from R2 (see Section 4.2.1).

Based on these settings, the system maps the time stamp for each data item to the *x* axis and renders the data points for the entire time series. If the width of the time scale is greater than the width of the panel, scroll bars appear to allow the user to virtually navigate forward and backward in time. To increase performance with long time series data, the system only renders the visible range of the time series, also known as the clip range.

As shown in Fig. 5, the detail visualization offers several interactive mechanisms for probing the data. Above the time series visualization, a time information bar shows the start and end time instants for the visible range, as well as the time instant that corresponds to the mouse hover position. Below the visualization, a value information bar shows both the value and moving range when the mouse hovers over a data item. The user can click on the value information bar to pin a value marker, which causes it to persist in the display for comparative purposes (see Fig. 5). The user may also drag a time range in the detail time series visualization. The selected time range is indicated by a rectangle with a yellow halo. It is also linked to the selection details pane, which is described in Section 5.4. The range selection can be translated and removed using mouse-based gestures.

The detail time series visualization offers four different modes for displaying the data items (see Fig. 8). The point mode simply represents each data item as an unfilled circle. In the line mode, the data items are shown as points that are connected with straight line segments based on the temporal sequence. With the stepped line plot, the value is assumed to remain constant until the time instant of the next value, which results in a horizontal

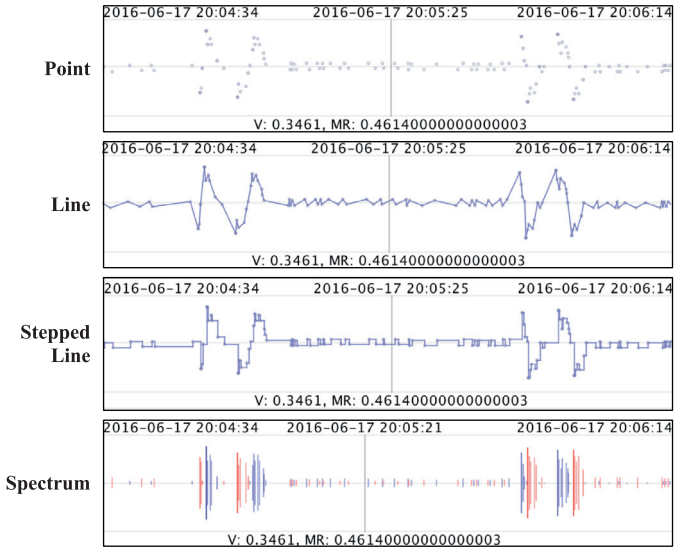

Fig. 8. The detail time series visualization can be configured to represent the data in one of four modes: point, line, stepped line, or spectrum. In addition to showing the data values, each mode can show the moving range values.

line segment between data points. When the next value is reached, a vertical line segment is drawn to connect the new value to the horizontal line segment of the previous value. This rendering option produces a stepped line configuration, which is the preferred representation for most sensor readings in the 3D printer logs.

The final representation option for the detail time series visualization is the spectrum mode (see Fig. 8). This mode is similar to a bar chart, except the bar is centered on the zero line, which passes through the middle of the *y*-axis. To indicate the sign, the bars are shaded differently for positive and negative values using user-defined colors. Inspired by audio spectrum plots, the spectrum mode emphasizes the magnitude of the values and the change between successive values in a more visually salient manner. Whereas a bar chart splits negative and positive values across the zero line, the spectrum plot keeps the values on the same plane for easier visual comparisons while preserving the sign with color encoding.

In addition to the data values, the detail time series visualization can optionally represent the moving ranges for the data values. Inspired by the process behavior chart introduced by Wheeler [43], the moving ranges are the differences between successive values and are used to measure routine variation. The user can choose to replace the data value with the moving ranges in the visualization or the moving ranges can be encoded in the opacity of the glyph (point, line, bar) representing the data value. When mapped to opacity, the smallest and largest range changes are mapped to the most and least transparent shadings, respectively. As shown in Figs. 14 and 8, mapping the moving range to the bar opacity in the spectrum plot mode enhances visual change detection. This approach emphasizes temporal variations while preserving the representations of individual values. The visualization can also be configured to not show the moving ranges.

As shown in Fig. 5, contextual information is preserved in the overview visualization by linking the scroll actions in the detail visualization to highlight the visible range with two black bars that surround the visualization. Coupled with the overview visualizations, the detailed time series visualization allows fine-grained, multi-scale data exploration while maintaining an awareness of temporal context in the whole data set.

5.4. Selection details panel

Located to the right of the variable visualization panel (see Fig. 4), the selection details panel shows statistical information for selections made in the detail visualization panel. This panel addresses the detailed statistical analysis requirements from R3 (see Section 4.3.1). When a time range selection is created, a selection details pane is added. The panes are stacked vertically with the most recent selection appearing at the bottom. To highlight the link between selections in the details panel and time series visualizations, clicking on a selection's name or time range in the details panel centers and highlights the selection range in the time series view, and hovering over a selection in the time series view highlights the corresponding selection visualization in the selection details panel using a color halo.

Each selection pane includes a statistical visualization similar to the one used in the overview region of the variable visualization panel (see Section 5.3.1). Reconfiguration of the pane layout and exporting of data associated with a selection can be accomplished via the button panel at right. The user can also adjust the *x* and *y* axis limits to normalize comparisons between selections. In Fig. 4, three selections are set in the *BeamCurrent* variable visualization pane. From left to right, the selections correspond to panes shown in the selection details panel from top to bottom. As the range selections are translated in the detail time series visualization, the corresponding selection pane visualizations are updated automatically. The selections provide the user with a lens for detailed comparisons.

5.5. Waterfall visualization

Shown in a separate window (see Fig. 4), the waterfall visualization provides both an overview and a detailed view of variable value changes for segments of the full time series. This visualization addresses both R1 (see Section 4.1.1) and R2 (see Section 4.2.1) design requirements. The waterfall visualization is constructed by partitioning the full time series into smaller segments, which represent build layers, using the build height variable (see Fig. 9). When a new build height value is recorded, the system begins printing a new layer.

As shown in Fig. 10, each layer is rendered as a color-shaded vertical line. Whereas a conventional time series line plot maps data values to the *y* axis, the waterfall visualization employs a color scale, which is mapped to the full range of values, to encode value changes on the line. The resulting vertical line condenses the layer information into a miniature form. Like the stepped line drawing mode for time series described in Section 5.3.2, a step encoding is used to color the line where the color representing the data value remains constant until a new value is encountered. An alternative approach is to interpolate between data values to produce a color gradient, however the step encoding is more appropriate for this application due to the sampling process of the 3D printer system.

As shown in Fig. 9, layer lines are rendered by aligning the start of each line along the top of the visualization. The layer start time is also mapped to the *x* position of the vertical line in a descending chronological layout from left to right. Layer segments are shaded using a semi-transparent color, which reveals portions of the build with shorter layer build times and more layers. That is, when the width of the window compresses the *x* axis time scale and causes layers with short durations to overplot one another, the semi-transparent shading will reveal the density of lines.

The waterfall visualization provides an important component for understanding layer patterns at different scales. It provides both an overview of the entire build and details about individual stages within each build layer using an approach reminiscent of

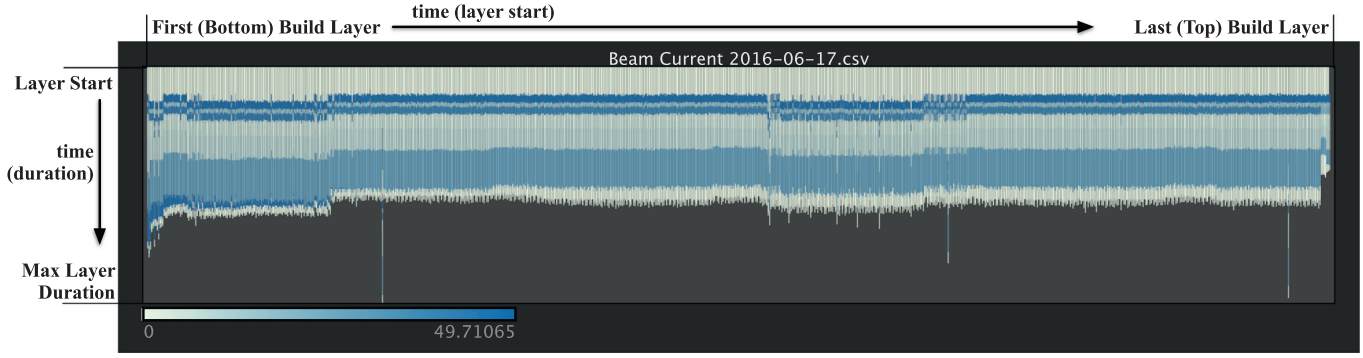

Fig. 9. The waterfall visualization shows a segmented overview of an entire time series using detailed micrographics. Each vertical line represents a build layer from a 3D print. The layer start time determines the x location of the line. The line length shows the time required to print the layer. Values are encoded in the color of the line segments using the color scale shown below and a step encoding. In this figure, three layers with significantly longer durations are visible as well as subtle variations in the print process near the middle of the build.

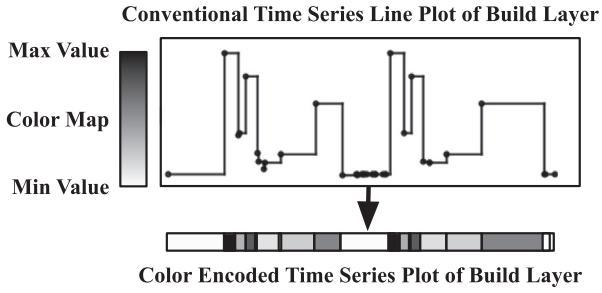

Fig. 10. Each vertical line in the waterfall visualization encodes data values for a particular variable during a build layer. Instead of using distance offsets on the x or y axis like conventional time series plots, the values are encoded using color based on a color scale that covers the min/max value range.

the micro/macro readings described by Tufte [15]. Therefore, both broad trends and details are displayed together and researchers gain a unique perspective of a build.

5.6. Segmented time series visualization

The segmented time series visualization (see Fig. 11) is also displayed in a separate window accessible through the main Falcon user interface. Similar to the waterfall visualization, the segmented visualization partitions the full time series for a variable into multiple time series plots for detailed comparative analysis. This detailed view addresses R2 (see Section 4.2.1) and R4 (see Section 4.4.1) design requirements. Although this visualization allows the user to specify the segmenting variable, the build height variable is usually used in 3D printer data analysis. As shown in Fig. 11, the segmented view consists of three main visualization panes: the segmented time series pane (left), the similarity/dissimilarity indicator pane (middle), and the image view pane (right).

5.6.1. Comparative visual analysis and information scent

The time series visualization panel stacks the segmented time series plots vertically from the top to the bottom of the build in a scrollable view (see Fig. 11). The numerical labels shown to the left of the time series indicate the segment build heights. To select a reference segment, the user clicks on its label (see Fig. 12). Then, the time series for the reference segment is drawn beneath the other segments' time series plots using a light gray color to support comparative visual analysis, which addresses requirements mentioned in R4 (see Section 4.4.1).

When a reference segment is selected, it is also compared numerically to the other segments using an implementation of the dynamic time warping (DTW) algorithm [44]. The DTW technique finds the optimal alignment between two times series by stretching or shrinking one time series. This warping is then used to determine the similarity between the two time series yielding a distance metric. Because the original DTW technique has $\mathcal{O}(n^4)$ time and space complexity, we use a Java-based implementation of FastDTW[2]. FastDTW is an approximation of DTW that has linear time and space complexity. Falcon applies the FastDTW technique to each pairwise combination of the reference with the other segments. For each segment, the distance value is normalized and inverted (to emphasize the smaller more similar values) yielding a similarity metric that is visually encoded in the width of a bar displayed below each segment label. The more filled the bar, the more similar the segment is to the reference segment. For example, in Fig. 11, the segment at height 20.00 mm is most similar to the reference height of 102.5 mm.

The middle similarity/dissimilarity indicator panel visually encodes the FastDTW distance values for all segments to guide the user to similar or dissimilar segments over the entire build. As shown in Fig. 12, the graphical indicators are arranged vertically and correspond to the layout of the time series segment pane. Since the height of the window typically prevents displaying all segment indicators individually, a binning algorithm groups neighboring segments to fit the available display space. Then, each bin is summarized by finding the largest and smallest FastDTW distance values and normalizing the values. The largest distance value is visually encoded with the red dissimilarity bar. The smallest distance value of the bin is inverted (to emphasize the smaller more similar values) and visually encoded with a blue similarity bar. The visual encoding uses both the width and color saturation to represent the distance values whereby wider and more saturated bars denote bins that are either more similar (blue bars) or dissimilar (red bars).

The resulting array of binned distance indicators, which are aligned with the time series panel scroll bar, provide information scent [2] in the context of the entire build. This scent guides the user to layers of interest, relative to the reference layer, without requiring manual scrolling and visual inspection of each individual time series segment. This method for providing information scent addresses the interactive feedback requirements from R1 (see Section 4.1.1). The user can hover the mouse over individual indicator bars to see the distance values as tooltips, and click on the bars to center the left time series panel on the appropriate segments.

[2] Falcon currently uses Dave Moten's FastDTW implementation available online at https://github.com/davidmoten/fastdtw

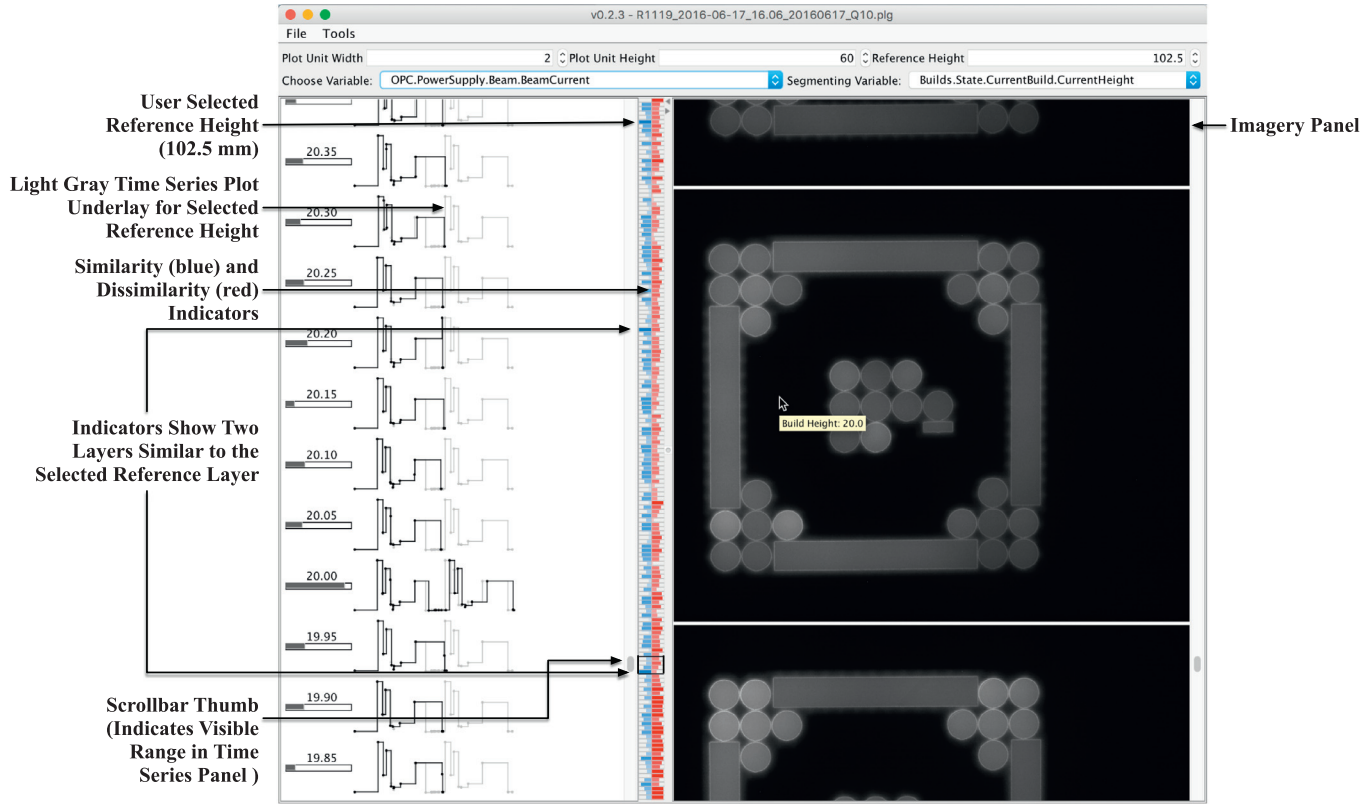

Fig. 11. The segmented time series view partitions the full time series for a variable of interest into multiple time series plots using a user-defined segmenting variable. Here the segmenting variable is the layer build height. The view consists of three main visualization panes: the segmented time series pane (left), the similarity/dissimilarity pane (middle), and the image view pane (right).

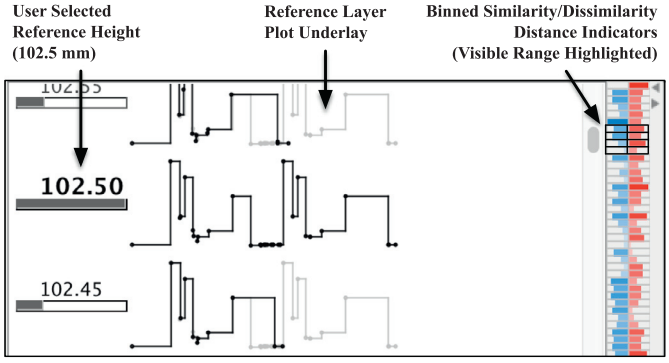

Fig. 12. When a build layer segment is selected in the segmented time series visualization, the layer is used as a reference height. The time series plot for the reference height is then rendered below the other time series plots for comparison. Also, similarity/dissimilarity with the reference height is visually encoded in the binned distance indicators. Here the black outlines indicate those bins whose time series plots are visible in the time series plot panel at right.

5.6.2. Near IR imagery panel

The rightmost imagery panel shows the near IR images for each build layer in a scrollable view similar to the time series visualization panel. The user can change the image scale by adjusting the panel bounds. To maintain a consistent view, scrolling in all of the panels can be synchronized. The image view provides a different perspective for visually examining build quality and identifying defects such as porosity, swelling, and temperature variation in a layer. In the future, the image panel will be augmented with information from computer vision algorithms to highlight porosity, hot spots, and other potential problems. Currently, these problems are detected manually through visual inspection by the user.

6. Case study: analyzing 3D printer build data

In this section, a case study of Falcon is presented to demonstrate its practical capacity to reveal multivariate patterns at multiple scales in complex time series data. By documenting the exploration and discovery of significant patterns from an actual build by an additive manufacturing researcher, who is also a co-designer of Falcon and co-author of this paper, this case study validates the effectiveness of Falcon. The effectiveness of Falcon is further corroborated by the fact that these experts have adopted the system as their primary tool for daily analysis work at the ORNL MDF.

A photograph of a typical analysis scenario is shown in Fig. 1. The geometrical configuration of this test, which is shown in Fig. 2, is designed to ensure that the Arcam Q10 system is functioning properly by printing a structure that is created entirely from the build platform without support structures. This configuration includes four distinct geometrical layouts as well as five specific features. The features include 4 – 104 × 104 × 15 mm blocks, 58 – 105 × 15 mm cylinders, 5 – 15 mm cubes, 1 – 15 × 30 × 5 mm block, and 1 – 30 × 15 mm cylinder. The data generated from a build of this test configuration is the subject of the analysis that is described in the remainder of this section. We present the analysis session as a narrative that is told from the perspective of the additive manufacturing researcher.

6.1. Overview first

We begin the analysis session by loading the printer data (a log file and near IR images) into Falcon and creating an overview of the entire build for the set of key variables. Based on experience acquired through the analysis of other builds and background knowledge of additive manufacturing, we have determined that

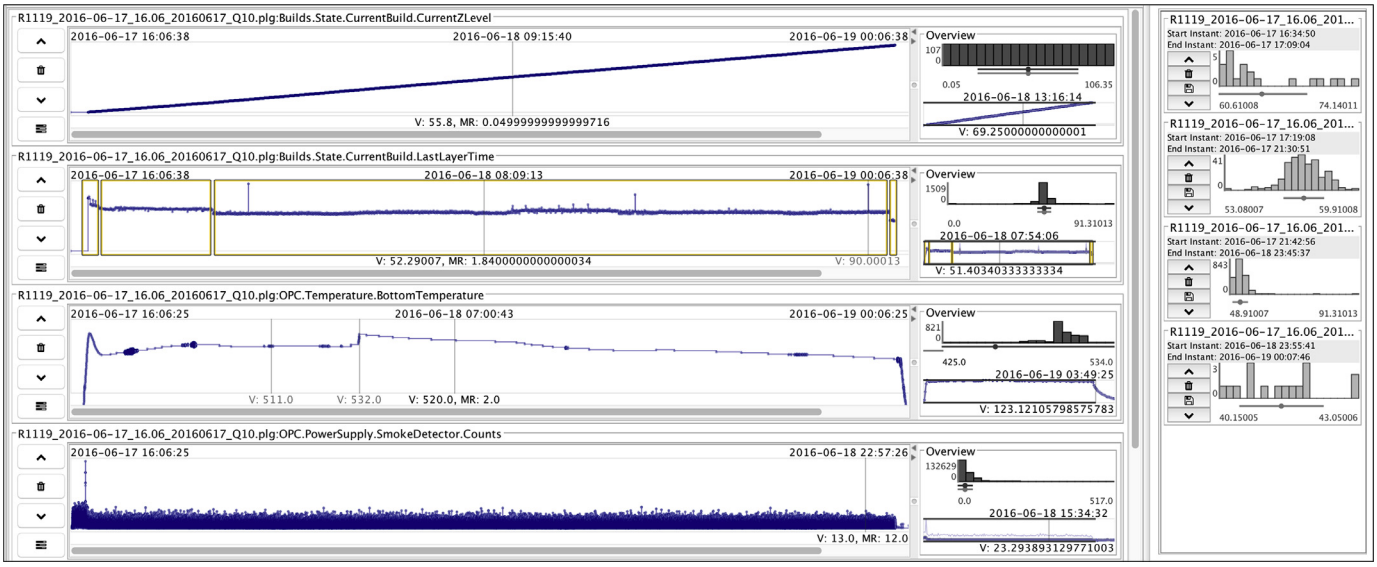

Fig. 13. We begin analysis of the 3D printer log by visually inspecting overview visualizations for a set of key variables. Here we show only four of the key variables due to space restrictions, but a typical session would involve dozens. The three outliers in the *LastLayerTime* visualization and a spike in the *BottomTemperature* visualization require further investigation as they may indicate problems in the build.

these variables are prime indicators of the overall quality of the build. Although this case study focuses on a few variables from this limited set due to space restrictions, it is important to note that many other variables are studied during a normal analysis session.

As shown in Fig. 13, we construct the visual overview by adjusting the parameters to show the entire time series in the detail time series visualization for each variable. In this case, the overview is created by setting the *Plot Chrono Unit* to hours and the *Plot Unit Width* to 30 pixels, which yields a 2-minute resolution. The figure shows four from the full set of key variables, but a typical analysis session may include dozens of variables from multiple files.

6.1.1. Build height analysis

In Fig. 13, the top variable visualization panel shows the *CurrentZLevel* variable, which indicates the height in mm from the bottom layer. We recognize the steady increase in value for this variable over the duration of the build as an artifact of the manner by which the system prints from bottom to top of the model. The mouse hover query on the *CurrentZLevel* detail time series visualization shows that the height is 55.8 mm at 9:15:40 and the moving range is approximately 0.05 mm, which is the expected layer thickness. At this coarse view, it appears that the build height plot is consistent with a normal build.

6.1.2. Bottom temperature analysis

Our eye is drawn next to the third variable panel, which shows the *BottomTemperature* variable. This variable indicates the temperature at the base plate, which should steadily decrease after the build initialization phase as the print layer height increases. In the detail time series visualization, we note a sharp increase in temperature near the middle of the build process (see the second pinned value of 532 °C in Fig. 13). This abnormal pattern warrants further investigation as it may indicate a change in processing parameters due to the build geometry.

6.1.3. System rake analysis

As shown in Fig. 14, we change to the spectrum time series plot mode to check the printer rake sensor readings. These sensors measure the consistency of the powder bed by raking powder over the sensor flaps and recording the time that they are open. Since a consistent powder bed is vital to an optimal melt process, the gaps and value spikes that appear in the highlighted ranges of this visualization are troubling. The spectrum visualization excels at revealing patterns of this nature. In addition to prompting us to inspect the consistency of the printed object, we are compelled to physically inspect the powder (e.g., distribution and chemical composition) and rake blade (e.g., bends, cracks, or holes) mechanisms within the printer.

6.1.4. Layer build time analysis

Turning our attention back to Fig. 13, we study the second variable pane, which shows the *LastLayerTime* variable. This variable is directly related to each layer's melt area as more time is required to melt a larger area, and vice versa. From this view, we identify and highlight four distinct stages of the overall print process for the test configuration using the time range selection tool (see the yellow rectangular highlights in Fig. 13). These stages correspond to four regions of the melt area for the build starting with the largest per layer melt area at the bottom and progressing to the smallest at the top.

In the bottom region (see the far left selection that corresponds to the top statistical selection view at right), the build layers require the most melt time per layer (between 60 and 74 sec). In the next selected region (see the second selection from the left and second statistical selection view from the top at right), the build layers require less melt area and exhibit a more normally distributed range of melt times (between 53 and 60 sec). The smaller dispersion of values and apparent lack of significant outliers in this region (see the statistical summary histogram) increases our confidence in the quality. Jumping ahead in the build time series, we note that the fourth region (see the far right selection and bottom statistical selection view at right) is also characterized by relatively small deviations in layer build times and requires the least amount of time to complete (approximately 13 min).

We now turn our attention to the third selection range in the *LastLayerTime* detail time series visualization (see the third selection from the left and the third statistical selection view from the top in Fig. 13). We observe three outliers occurring near the beginning, middle, and end of this range, the largest of these being approximately 91.3 sec (see the max value shown in the statistical view). These outliers skew the histogram plot, which reduces the perceivable structure of values within the range of normal layer

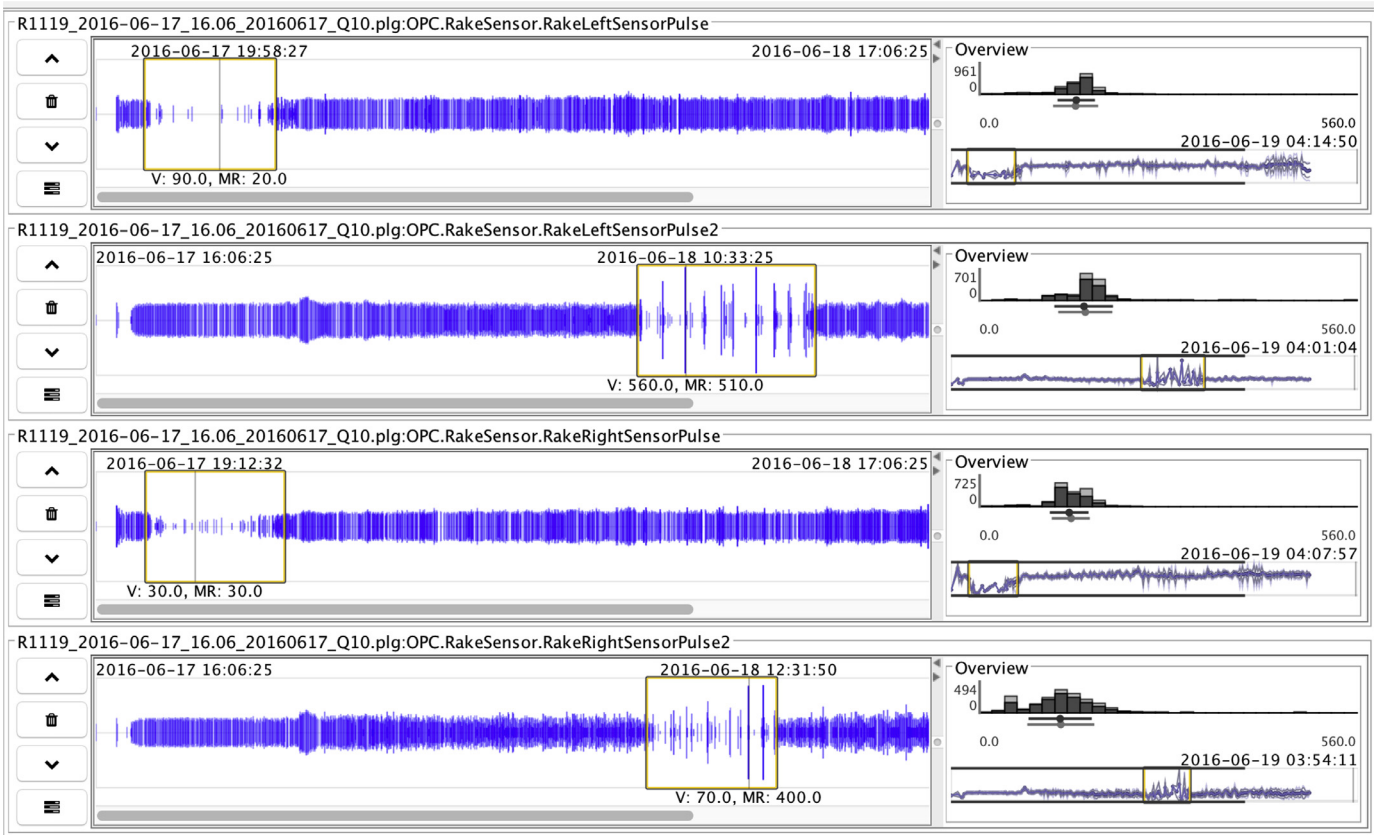

Fig. 14. We use the spectrum time series plot mode to analyze four rake sensors over the entire build. In a normal build, these plots should show a minimal variation without large gaps. Here the gaps and spikes that appear may indicate problems with the powder bed consistency or rake blade. The spectrum plot mode excels at revealing such conditions.

times. As shown in Fig. 13, we set a pinned value marker on the last outlier showing the value is about 90 sec. These aberrations suggest that something occurred in the preceding layer to cause a significantly longer time for its completion, which may indicate problems in the build. Therefore, we must investigate the corresponding layers in more detail.

6.2. A more focused overview

We create a more focused overview using a waterfall visualization of the *BeamCurrent* variable (see Fig. 9). As described in Section 5.5, the waterfall visualization is constructed by segmenting the variable time series into individual layers using the build height variable. Again we see the four distinct regions that characterize the test pattern. Moreover, we immediately recognize the three outlier layers, as the vertical lines are significantly longer than the others. In addition, we see more subtle variances that are difficult to see in the initial overview. We detect slight variances in the initial preheat stages of the layers (see the slight shift in the length of the white segments at the top of each vertical line). We also see smaller variations in layer duration and the beam current values during the middle of the build. Specifically, seven build layers show some variation within the normal range. During a typical analysis session these variations would warrant deeper investigations, but, for the sake of brevity, we do not discuss them in the current work.

6.3. Zoom and filter

At this point, we have visually identified potential issues at specific times/heights in the build using overview visualizations. Next, we increase the detail of the visualizations to conduct more thorough analysis of the abnormal build layers. In the detail time series visualization, we increase the level-of-detail by setting the *Plot Chrono Unit* to seconds with a *Plot Unit Width* of two pixels. Fig. 15 shows the resulting view after scrolling to the time range associated with the third outlier, which is indicated in Fig. 13 by the rightmost pinned value marker of approximately 90 sec.

In Fig. 15, the black context bars, which halo the overview time series plot for each variable, show the relative position of the visible time range in the detail time series visualization. In the *LastLayerTime* detail time series visualization, we highlight the outlier using a time range selection. We use the mouse hover query to confirm that the preceding layer print time value is approximately 90 sec, which is approximately 38 sec more than the previous reading of approximately 51.9 sec (see the pinned marker value to the left of the mouse selection range in the detail time series visualization).

Since *LastLayerTime* values describe the previous layer, we glance back one layer on the other three variable plots shown in Fig. 15. In the detail time series visualization for the *CurrentZLevel* variable, we notice the longer time gap between *z*-level changes (see the highlighted range in the top variable plot). We pin a value marker on this plot to show that the outlier layer occurs at a height of approximately 102.5 mm from the bottom layer. Likewise, we see that the outlier layer is also visible in the wider gap between the *Rake.CurrentPosition* variable readings (see the highlighted range in the bottom variable plot). This variable captures the position of the rake (a mechanism that moves metal powder across the platform before the melting process) relative to the center of the build. The normal rake pattern is visible in the detail time series visualization—the rake traverses three times from one

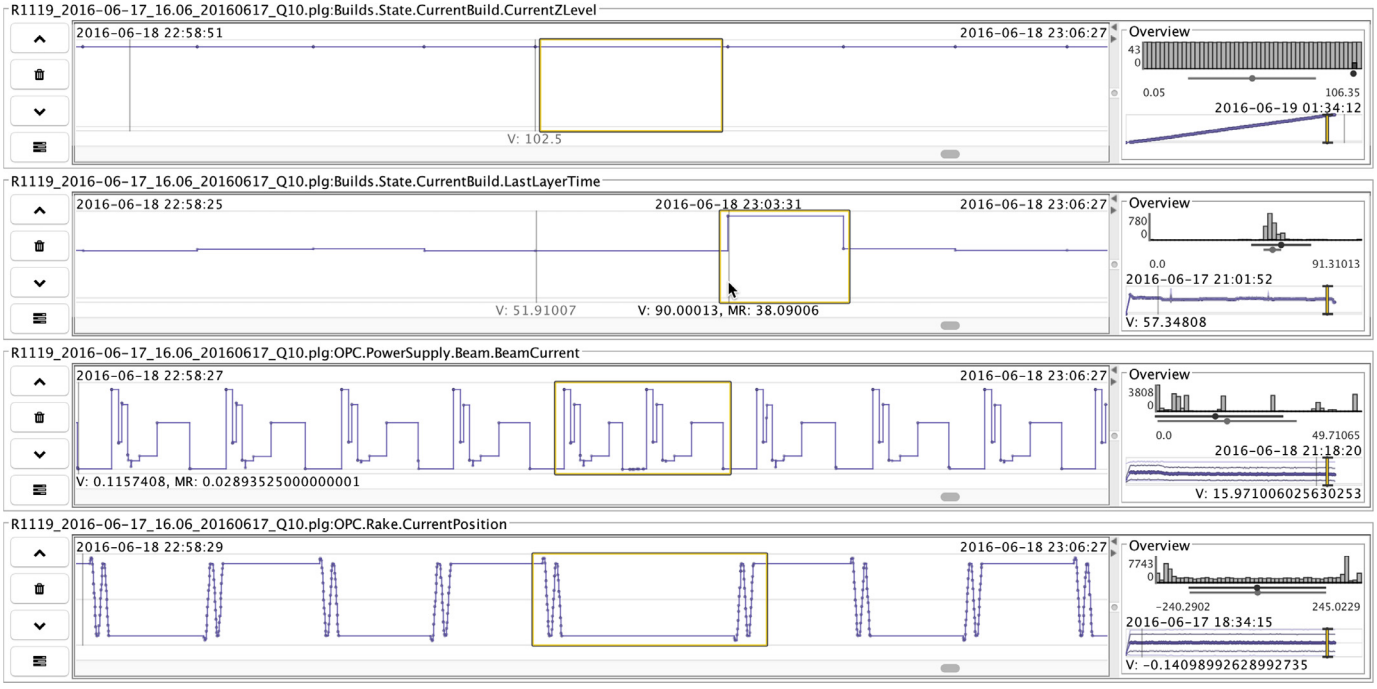

Fig. 15. We zoom into the detail time series visualization by changing the time scale settings to investigate one of the outliers detected in the overview visualization. We see a repeat pattern in the *BeamCurrent* detail visualization that is caused by a system arc trip. The arc trip may indicate problems in the microstructure of the printed objects. Falcon helps find such issues and precisely locate regions of the printed object that may be affected.

side to the other depositing and smoothing the powder over the print area as it moves.

The most telling discovery is found in the *BeamCurrent* detail time series visualization. In Fig. 15, we highlight the time range for the outlier layer using the time range selection. The normal *BeamCurrent* pattern can be observed in the other layers as it modulates between different predictable levels during each of the five stages the system goes through as it prints a layer. However, in the outlier layer, we see that the pattern is repeated, which narrows down the cause of the outlier. We look at the last peak of the first five stage pattern where the process seems to stop and restart the print pattern.

6.4. More detail, on demand

To study the abnormal layers in more detail, we open the segmented visualization panel as shown in Fig. 11. We segment the full *BeamCurrent* variable time series into smaller time series visualizations using the *CurrentHeight* variable. The resulting view shows the individual time series visualizations for each build layer. As shown in Fig. 12, we set the outlier layer at 102.5 mm as the reference segment by clicking on its time series. This action causes the panel to render the time series for the reference segment as an underlay, using a light gray color, on all the other time series plots. The reference plot highlights the repeated print pattern. At right of the time series panel scroll bar, the blue similarity indicators reveal the locations of the other two outlier build layers.

In Fig. 11, we scroll to the bottom of the build near 20 mm, where one of the other outliers appears. The 20 mm layer's indicator suggests that it is similar to the reference layer. In the time series plot, we see the same repeated pattern although the reference signal (at 102.5 mm) is slightly shifted to the left. Despite the slight shift, this evidence suggests that the outliers at 102.5 mm and 20 mm are caused by the same condition.

Finally, the right image panel shows a scrollable view of the near IR images for each layer. This image panel is synchronized with the layer time series panel so that we see the image for the layer at 20 mm. This view aids in the discovery of microstructure problems such as porosity, hot spots, or swelling that may occur during the stages of each layer build. Together with microscope images (not shown), the image views help explain the issues that are seen in the other visualizations and ascertain the quality of the material. In this instance, we see no microstructure problems in the image at layer 20 mm.

6.5. Understanding the cause of the outliers

From experience, we know that such abnormal layer times are usually caused by either an arc trip or a smoke detection event. Both conditions cause us to question the build quality. When the beam is shut down during the melting stage, which is the case in this instance, the microstructure in that layer can be altered. Consistent microstructure of the material is essential for creating parts that are dependable.

An arc trip occurs when the beam forces too much energy density through the filament. The system responds to the arc trip by immediately shutting off the beam and completely restarting the layer build process. In the time series visualization for the *BeamCurrent* (see Fig. 15), an arc trip appears as a repeated signal of the electron beam current as it progresses through the five layer stages.

The detection of radiation from ionized powder particles in the vicinity of the electron gun is characterized as a smoke event. This condition causes the powder to scatter, forming a dust cloud in the build chamber, which can harm the beam filament. As with the arc trip condition, a smoke detection event causes the beam to shut off and the layer print is restarted, which also manifests in the variable visualization as a repeated signal.

By separately analyzing the smoke sensor readings, we determine that the outliers are caused by an arc trip. The bottom variable visualization pane in Fig. 13 shows the *SmokeDetector.Counts* variable. Except for a spike during the initialization phase, which

is expected, the remainder of the detail time series visualization shows normal variations.

The ability to discover and thoroughly investigate such complex patterns will help us formulate new algorithms to automatically detect these and other conditions that impact the quality of the print—a critical capability for certifying the quality of 3D printed objects for applications where failures are not tolerable, such as printing aircraft parts. Additional study of this data is also helping us fine tune the print parameters to avoid such conditions in future builds. Moreover, we have engaged in supplemental investigations of this particular print to understand how changes in the melt pool affect microstructures. These investigations involve image processing algorithms to detect porosity using the near IR imagery, viewing cross sections of the material through optical microscopes, and using a scanning electron microscope to view extremely fine resolution structure. Falcon helps researchers determine the defect conditions and related sensor patterns, as well as the precise location to physically examine the microstructure of the 3D printed objects.

7. Discussion

Additive manufacturing researchers have formed an analysis workflow based on several months of experience using Falcon to analyze hundreds of builds. The workflow closely follows the stages of Shneiderman's visual information seeking mantra: "Overview first, zoom and filter, then details-on-demand" [1]. Although the design of Falcon may include some indirect guidance to proceed from an overview to details during analysis, the researchers have adopted this workflow based on their own independent explorations. The fact that they have naturally gravitated to this strategy corroborates the notion that it is a favorable strategy for designing new data visualization systems.

The success of Falcon is largely attributed to the assembly of an interdisciplinary team of motivated individuals each bringing a specific area of expertise to the project. Pictured in Fig. 1, additive manufacturing experts at the ORNL MDF push the limits of manufacturing technology to build 3D printed vehicles, complex aircraft parts, and even sustainable houses. We employ a participatory design process where these experts are co-designers and the primary users of our tool, and we meet with them weekly to discuss new developments, evaluate their use of the tools, and plan milestones.

New feature development often begins with the additive manufacturing researchers expressing a challenging analysis task, ideas for addressing the task, and expected outcomes. Next, the visual analytics researchers create mock-up sketches, which the team discusses and refines. The sketches are transformed into prototype solutions, which are field tested by the additive manufacturing researchers. The team continues refining the resulting techniques until a satisfactory solution is reached.

We have sacrificed some generality in Falcon to implement features that are specific to additive manufacturing, such as segmentation according to build heights and porosity detection. Although these decisions make Falcon less compatible with all possible scenarios, it has enabled discoveries that would be far more difficult, if not impossible, without them. That said, we have striven to modularize such domain specific capabilities to permit future utilization in fields that face similar challenges, such as cyber security and climate science.

8. Future work

Since Falcon is an ongoing development, we have identified areas for improvement to increase its effectiveness in additive manufacturing and other similar domains. Although Falcon supports the ability to save the current display parameters to allow sharing and restarting a previous analysis session, it lacks the ability to capture provenance information. We plan to incorporate provenance capabilities by integrating existing frameworks, such as VisTrails [45]. In addition to provenance, we plan to develop annotation capabilities where notes persist with the data and are shareable with multiple users.

Interactive similarity/dissimilarity methods for time series data have demonstrated a promising capability that we are currently expanding into an automated process. In particular, we are investigating active learning techniques that automatically learn features of interest by monitoring the user's interaction. We are also incorporating computer vision algorithms to automatically detect and highlight porosity, swelling, and hot spots based on the near IR imagery. The integration of these and other machine learning algorithms will help reduce the search space, recommend hidden features in the data, and increase the overall efficiency of the analysis process.

We are integrating new visualization techniques into Falcon. Additional multivariate visualizations, such as parallel coordinates [46], are under development. We are studying different visualization strategies for representing Boolean and categorical information in time series visualizations. We also plan to investigate new focus + context techniques for time series visualizations to prevent perceptual issues related to the analysis of multiple separate views [36]. Another practical visualization feature that we plan to develop is the capability to difference and integrate multiple time series streams into a single view for more direct comparative analysis.

9. Conclusion

In this paper, we present Falcon, a new system that follows a visual analytics approach to improve knowledge discovery in long and complex time series data with practical applications to the field of additive manufacturing. Falcon leverages a human-centered design grounded in the visual information seeking strategy [1]. Falcon provides linked visualizations from both temporal and statistical orientations with automated analytics to highlight interesting features. In addition, Falcon offers intuitive mechanisms to access multiple levels-of-detail as necessary.

From our informal evaluations of the applied use of Falcon in additive manufacturing, we have learned that non-visualization experts can be vital members of interdisciplinary design teams as they help design new capabilities that respond to their actual needs, and they quickly employ new visual analytics techniques in creative ways to solve problems. The parallels between the analytical goals in additive manufacturing and other domains suggest that these capabilities are broadly applicable to many domains as they help users develop and refine a more complete mental model of complicated and large-scale time series data.

Acknowledgments

This research is sponsored by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy, Advanced Manufacturing Office, under contract DE-AC05-00OR22725 with UT-Battelle, LLC and Oak Ridge National Laboratory LDRD project No. 7409. The authors wish to express our appreciation to both the reviewers and editor for clear and constructive feedback.

References

[1] Shneiderman B. The eyes have it: a task by data type taxonomy for information visualizations. In: Proceedings of the IEEE symposium on visual languages; 1996. p. 336–43.

[2] Willett W, Heer J, Agrawala M. Scented widgets: improving navigation cues with embedded visualizations. IEEE Trans Vis Comput Graph 2007;13(6):1129–36.

[3] Aigner W, Miksch S, Schumann H, Tominski C. Visualization of time-oriented data. Springer; 2011.
[4] Bach B, Dragicevic P, Archambault D, Hurter C, Carpendale S. A review of temporal data visualizations based on space-time cube operations. In: Proceedings of eurovis; 2014.
[5] Dou W, Wang X, Skau D, Ribarsky W, Zhou MX. LeadLine: Interactive visual analysis of text data through event identification and exploration. In: Proceedings of the IEEE conference on visual analytics science and technology; 2012. p. 93–102.
[6] Steed CA, Drouhard M, Beaver J, Pyle J, Bogen PL. Matisse: a visual analytics system for exploring emotion trends in social media text streams. In: Proceedings of the IEEE international conference on big data; 2015. p. 807–14.
[7] Dörk M, Gruen D, Williamson C, Carpendale S. A visual backchannel for large-scale events. IEEE Trans Vis Comput Graph 2010;16(6):1129–38.
[8] Havre S, Hetzler B, Nowell L. ThemeRiver: visualizing theme changes over time. In: Proceedings of the IEEE symposium on information visualization; 2000. p. 115–23.
[9] Luo D, Yang J, Krstajic M, Ribarsky W, Keim D. EventRiver: visually exploring text collections with temporal references. IEEE Trans Vis Comput Graph 2012;18(1):93–105.
[10] Cui W, Liu S, Tan L, Shi C, Song Y, Gao Z, et al. TextFlow: towards better understanding of evolving topics in text. IEEE Trans Vis Comput Graph 2011;17(12):2412–21.
[11] Li J, Zhang K, Meng ZP. Vismate: interactive visual analysis of station-based observation data on climate changes. In: Proceedings of the IEEE conference on visual analytics science and technology; 2014. p. 133–42.
[12] Steed CA, Evans KJ, Harney JF, Jewell BC, Shipman G, Smith BE, et al. Web-based visual analytics for extreme scale climate science. In: IEEE international conference on big data; 2014. p. 383–92.
[13] Harrison L, Hu X, Ying X, Lu A, Wang W, Wu X. Interactive detection of network anomalies vis coordinated multiple views. In: Proceedings of the IEEE symposium on visualization for cyber security; 2010. p. 91–101.
[14] Haugen B, Richmond S, Kurzak J, Steed CA, Dongarra J. Visualizing execution traces with task dependencies. In: Proceedings of the 2nd workshop on visual performance analysis. ACM; 2015. p. 1–8.
[15] Tufte ER. Envisioning information. Cheshire, CT: Graphics Press; 1990.
[16] Tufte ER. Beautiful evidence. Graphics Press; 2006.
[17] Heer J, Kong N, Agrawala M. Sizing the horizon: the effects of chart size and layering on the graphical perception of time series visualizations. In: Proceedings of the SIGCHI conference on human factors in computing systems; 2009. p. 1303–12.
[18] Javed W, McDonnel B, Elmqvist N. Graphical perception of multiple time series. IEEE Trans Vis Comput Graph 2010;16(6):927–34.
[19] Wijk JJV, Selow ERV. Cluster and calendar based visualization of time series data. In: Proceedings of the IEEE symposium on information visualization; 1999. p. 4–9.
[20] Weber M, Alexa M, Müller W. Visualizing time-series on spirals. In: Proceedings of the IEEE symposium on information visualization; 2001.
[21] Bertini E, Hertzog P, Lalanne D. SpiralView: towards security policies assessment through visual correlation of network resources with evolution of alarms. In: Proceedings of the IEEE symposium on visual analytics science and technology; 2007. p. 139–46. doi:10.1109/VAST.2007.4389007.
[22] Lin J, Keogh E, Lonardi S, Lankford JP, Nystrom DM. Visually mining and monitoring massive time series. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining; 2004. p. 460–9.
[23] Rind A, Lammarsch T, Aigner W, Alsallakh B, Miksch S. Timebench: a data model and software library for visual analytics of time-oriented data. IEEE Trans Vis Comput Graph 2013;19(12):2247–56.
[24] Buono P, Aris A, Plaisant C, Khella A, Shneiderman B. Interactive pattern search in time series. In: Proceedings of SPIE visualization and data analysis conference, 5669; 2005. p. 175–86.
[25] Brinton WC. Graphic methods for presenting facts. New York, NY: The Engineering Magazine Company; 1914.
[26] Tufte ER. The visual display of quantitative information. Cheshire, CT: Graphics Press; 1983.
[27] Bade R, Schlechtweg S, Miksch S. Connecting time-oriented data and information to a coherent interactive visualization. In: Proceedings of the SIGCHI conference on human factors in computing systems; 2004. p. 105–12.
[28] Kincaid R. SignalLens: focus+context applied to electronic time series. IEEE Trans Vis Comput Graph 2010;16(6):900–7.
[29] Walker J, Borgo R, Jones MW. TimeNotes: a study on effective chart visualization and interaction techniques for time-series data. IEEE Trans Vis Comput Graph 2016;22(1):549–58.
[30] Buono P, Plaisant C, Simeone A, Aris A, Shmueli G, Jank W. Similarity-based forecasting with simultaneous previews: a river plot interface for time series forecasting. In: Proceedings of the international conference information visualization; 2007. p. 191–6.
[31] Zhao J, Chevalier F, Pietriga E, Balakrishnan R. Exploratory analysis of time-series with chronolenses. IEEE Trans Vis Comput Graph 2011;17(12):2422–31.
[32] Hao MC, Dayal U, Keim DA, Schreck T. Importance-driven visualization layouts for large time series data. In: Proceedings of the IEEE symposium on information visualization; 2005. p. 203–10.
[33] Javed W, Elmqvist N. Stack zooming for multi-focus interaction in time-series data visualization. In: Proceedings of the IEEE pacific visualization symposium; 2010. p. 33–40.
[34] Plaisant C, Carr D, Shneiderman B. Image-browser taxonomy and guidelines for designers. IEEE Softw 1995;12(2):21–32.
[35] Beaveridge WIB. The art of scientific investigation. Caldwell, NJ: The Blackburn Press; 1957.
[36] Rensink RA. Change detection. Annu Rev Psychol 2002;53:245–577.
[37] Wang Baldonado MQ, Woodruff A, Kuchinsky A. Guidelines for using multiple views in information visualization. In: Proceedings of the working conference on advanced visual interfaces; 2000. p. 110–19.
[38] North C. Multiple views and tight coupling in visualization: a language, taxonomy, and system. In: Proceedings CSREA CISST workshop of fundamental issues in visualization; 2001. p. 626–32.
[39] Becker RA, Cleveland WS. Brushing scatterplots. Technometrics 1987;29(2):127–42.
[40] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to algorithms. 2nd. Cambridge, MA: The MIT Press; 2001.
[41] Bertin J. Semiology of graphics: diagrams, networks, maps. Madison, WI: University of Wisconsin Press; 1983.
[42] Tukey JW. Exploratory data analysis. Reading, MA: Addison Wesley; 1977.
[43] Wheeler DJ. Making sense of data. Knoxville, TN: SPC Press; 2003.
[44] Salvador S, Chan P. Fastdtw: toward accurate dynamic time warping in linear time and space. In: Proceedings of the ACM KDD workshop on mining temporal and sequential data; 2004. p. 70–80.
[45] Bavoil L, Callahan SP, Crossno PJ, Freire J, Scheidegger CE, Silva CT, et al. Vistrails: enabling interactive multiple-view visualizations. In: Proceedings of the IEEE visualization conference; 2005. p. 135–42.
[46] Inselberg A. The plane with parallel coordinates. Vis Comput 1985;1(2):69–91.