Eurographics Conference on Visualization (EuroVis) 2015
H. Carr, K.-L. Ma, and G. Santucci
(Guest Editors)

Volume 34 (2015), Number 3

Interactive Fusion and Tracking For Multi-Modal Spatial Data Visualization

M. Elshehaly[1], D. Gračanin[1], M. Gad[2], H. G. Elmongui [145], and K. Matković[3]

[1]Virginia Tech, USA [2]Ain Shams Univ., Egypt [3]VRVis Research, Austria [4]Alexandria Univ., Egypt [5]GIS Innov. Ctr., Umm Al-Qura Univ., KSA

Abstract

Scientific data acquired through sensors which monitor natural phenomena, as well as simulation data that imitate time-identified events, have fueled the need for interactive techniques to successfully analyze and understand trends and patterns across space and time. We present a novel interactive visualization technique that fuses ground truth measurements with simulation results in real-time to support the continuous tracking and analysis of spatio-temporal patterns. We start by constructing a reference model which densely represents the expected temporal behavior, and then use GPU parallelism to advect measurements on the model and track their location at any given point in time. Our results show that users can interactively fill the spatio-temporal gaps in real world observations, and generate animations that accurately describe physical phenomena.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques,

1. Introduction

For a long time, visualization has proven to be very useful in analyzing time-varying data. This is due to the ability of the human eye to quickly detect patterns and structural changes in images. An effective time-variant data visualization aims to help users answer a number of questions about: *(i)* whether a data element exists at a specific time, *(ii)* how frequent is the occurrence of a particular event or data item, *(iii)* what is the length of the time span from beginning to end of a subset of data, *(iv)* what is the rate of change, and *(v)* what is the sequence of events. Consequently, dynamic, time-varying data present a big challenge to visualization researchers. We need to enable the user to discriminate between the different data features and also to identify changes that those features undergo over time. These and other challenging tasks made visualization of time-variant data an active area of research. However, the bulk of research in the area is based on the crucial assumption that measurement data, covering the space-time domain densely enough, are available to represent the underlying phenomena. In these cases, the primary focus is on visual representation of a single source of data. The fact that this assumption is not satisfied by the majority of scanning modalities motivates the fusion of multiple sources to obtain more informative visualizations. In satellite data, single modality measurements are only available for a subset of space at any given time. Combining findings from multiple satellites and using simulation to fill the gaps can support timely decisions.

The main goal of data fusion is then to fill the gaps in data obtained from different sources, in order to present a more complete representation of the physical phenomena in question with the highest possible information content. Nevertheless, the generation of such representation is not a trivial task. For years, fusion research has adopted machine learning techniques to integrate multimodal information. However, researchers in the field have highlighted the shortcomings of these techniques alone in coping with the increased complexity of multimodal data, and have called for a human-in-the-loop fusion model [HHT00]. Consequently, research has witnessed a shift of interest toward "high level" fusion which addresses the problem from a human-computer interaction perspective [FN13].

This realization of how crucial visualization is to the fusion process strongly motivates us to propose a solution approach that aims at the interactive user-aided reconstruction of missing information in satellite data. In this paper, we present such an approach through describing a novel technique for the real-time correction, tracking, and prediction of ground truth data based on satellite measurements. Our technique starts by creating a reference model that captures

expected behavior of an extracted Region of Interest (ROI) from simulation, based on domain expertise. This ROI is then resampled to create a GPU-efficient data structure for real-time tracking. Our solution targets a specific problem in the atmospheric data science domain. However, the technique could be directly applied to the more general problem of particle tracing in a given flow field, for the purpose of reconstructing missing flow information within a ROI. Our contributions in this paper include: *(1)* A novel reference model construction technique that captures the expected behavior of physical phenomena, *(2)* An efficient data resampling strategy to support interactive fusion and tracking on the GPU, and *(3)* A GPU-based data integration and tracking method that projects measurements onto the constructed reference model to fill the spatio-temporal gaps and provide a coherent story of how measured points have evolved over time. Our results show that the proposed tracking technique is able to create a coherent representation from originally scattered detection points obtained from satellite readings.

Section 2 summarizes related work in data fusion, feature tracking and flow visualization. Section 3 describes the problem of missing satellite ash detections following the 2011 eruption of Puyehue-Cordón Caulle volcano in Chile — datasets made available by the IEEE Scientific Visualization Contest of 2014 [IEE14]. Section 4 describes our approach and explains the developed model construction and tracking techniques. Results are discussed in Section 5. Section 6 concludes the paper.

2. Background and Related Work

The visual fusion of multimodal time-variant datasets is motivated by research in the fields of data fusion and scientific visualization. In this section, we explore related work in data fusion, flow visualization, and feature tracking.

2.1. Data Fusion

Data integration is the process of merging information from heterogeneous sources with differing conceptual, contextual and typographical representations. A related term, "data fusion" involves the combination of unstructured or semi-structured information into a new set with the aim of reducing uncertainty. Over the years, the scope of applications that use data fusion techniques has increased from strictly military-related to a broad variety of domains. In the mid-1980s, the Joint Directors of Laboratories (JDL) introduced a model of data fusion that divides the various processes involved. Hall et al. [HHT00] proposed the inclusion of the Human Computer Interface (HCI) as a fifth level of the JDL Data Fusion Process Model. The Visual Data Fusion (VDF) Model was proposed by Karakowski [Kar98, BRW07] as an extension to the JDL model. In this model, the human is a central participant in the information fusion process, and the whole process becomes a creative problem-solving task. It is the closest to our developed framework (see [FN13] for a list of alternative models). Our approach builds on motivation for the VDF Model and targets similar goals of keeping the human-in-the-loop at different levels of operation.

2.2. Flow Visualization

The visual representation of flow data has been a central problem in scientific visualization. The main objective is to convey magnitude and direction of flow, while highlighting critical points, or regions of interest (ROIs). In order to fulfill that, four general classes of techniques exist in the literature [PL09]: *(i)* Direct representations [Lar08] place arrow glyphs at every sample point. *(ii)* Texture-based techniques [VW91] [LHD*04] aim to provide a complete, dense representation of the flow field with high spatio-temporal coherency. *(iii)* Geometric, or integration-based, techniques [MLP*10] advect streamlines for steady flow, or pathlines for unsteady flow, using different seeding strategies [MTHG03], for the purpose of occlusion reduction. Ma et al. [MWSJ14] defer the occlusion reduction process until after the streamlines are generated, by creating a graph-based representation. *(iv)* Feature-based [LWS13] methods extract flow features that are deemed interesting by the user in a preprocess. The visualization is then based on the extracted subsets rather than the entire dataset. Particle tracing methods can be used to reconstruct missing flow information for features of interest [COJ15]. Our approach combines ideas from geometric and feature-based flow visualization to reconstruct missing flow information within an extracted ROI and then use the reconstructed model for efficient real-time tracking of the extracted feature.

2.3. Feature Tracking

Walsum et al. [VWPSP96] define a feature as a region in a dataset that is of interest for interpretation. Feature tracking refers to the process of extracting dynamics, which seeks to identify and describe movement or transport of data. Feature tracking techniques can be broadly classified into: *(i)* pixel-based [RdE90] [Adi] [PPH*08], or *(ii)* feature-based. Pixel-based tracking methods have the disadvantage that they require small time steps for accurate matching. Feature-based tracking methods are further classified into ones that rely on *region correspondence* [SW96] [KS91] [YLC13], or *attribute correspondence* [SPY94] [SSZC94] [RPS01]. Feature-based methods require less computation than their pixel-based counterparts. They have the additional advantage of providing a higher level of abstraction to the data, and the ability to tolerate larger temporal gaps [SYM14].

We advocate a user-controlled dynamic fusion and visualization strategy that allows the expert to interactively select a time instance and obtain full information, based on multiple sources. Similar to a framework presented by Rattner et al. [RGJ12], we start by identifying regions of interest and

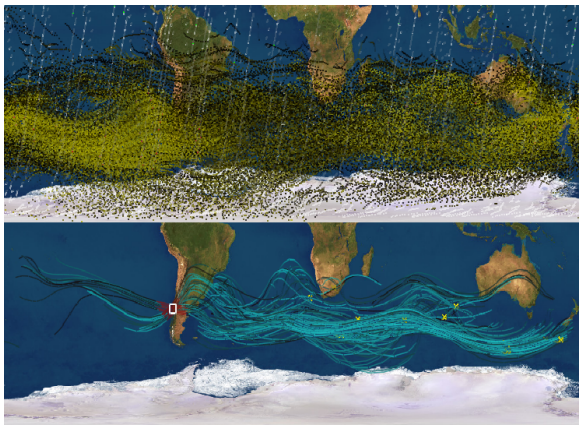

Figure 1: *CLaMS trajectories seeded at volcanic ash detections from* 0 *S to* 90 *S:* ***Top:*** *Point cloud filtered by time between June 12 at 13:00 and June 15 at 16:00.* ***Bottom:*** *Pathlines further filtered by StreamProbe technique [ESGG*14] around the eruption location with trajectory seeds marked by yellow X's.*

then track them through time. The main difference is that we target multi-sourced data containing large temporal gaps.

3. Problem: Atmospheric Data

We address the problem of missing data reconstruction in the context of an atmospheric science problem. The datasets represent ash detections in the atmosphere following the 2011 eruption of Puyehue Cordón Caulle volcano in Chile. Visualization aims at revealing the ash plume that resulted from the volcanic eruption and were injected into the atmosphere. Interest in visualizing such a plume and how it evolved over time is motivated by the fact that ash particles pose severe danger to aircrafts. Experts seek to understand the behavior of volcanic ash in the atmosphere to make decisions and find means to reduce the adverse effect that ash has on Air Traffic Management. To this end, our visualization aims to provide a coherent story about the ash plume ejected into the atmosphere, while enabling the experts to interact with it, view it at any user-defined time instant, and predict its future evolution. We start by describing the datasets that were used, and the challenges posed by the data.

3.1. Datasets and Challenges

The datasets made available through the IEEE Scientific Visualization Contest of 2014 [IEE14] contain both a simulation set and satellite measurements from June and July of 2011 following the eruption. Simulation data are obtained from The Chemical Lagrangian Model of the Stratosphere (CLaMS) [MKG*02]. It contains trajectories simulating individual air parcels including vertical transport. The trajectories are calculated using a fourth order Runge-Kutta scheme and are seeded at volcanic ash detections captured by The Michelson Interferometer for Passive Atmospheric Sounding (MIPAS) aboard the European Space Agency's Envisat satellite [GHSR14].

Figure 1 displays two subsets of CLaMS trajectories. The top view is filtered by time only while the bottom view is further filtered by an intersection with a region of interest around the eruption location and time. The seed points are marked in yellow but are hard to spot in the top view due to clutter. However, clutter is not the only challenging aspect of the CLaMS dataset. Like many simulation datasets, the trajectories are irregular in space and time which makes this dataset a poor candidate for interactive fusion. Correlating these trajectories to regions of interest in other datasets is computationally expensive.

The Atmospheric Infrared Sounder (AIRS) [HGM14] is one of the instruments aboard NASA's Aqua satellite. Figure 2 shows sample readings on two different 12-hour periods from AIRS data. The detected ash plume is shown in blue. Like the majority of satellite datasets, this set in its original form is of limited value for the analysis of the temporal behavior of the plume. This is primarily due to data discontinuity. In the particular case of the Puyehue Cordón Caulle eruption, the direction in which the satellite revolves around the Earth is opposite to the direction in which the plume is evolving. Therefore, detections found by AIRS at a certain location can only be captured again 12 hours later. A lot of change may have occurred during these 12 hours. Given AIRS data alone, there is no way for the expert to analyze such change. This challenge is clearly depicted by the discontinuity of the plume near Australia shown in the lower right corner of Figure 2. To address this problem and reconstruct the ash plume, Günther et al. [GSFT14] (the winning entry of the Contest) applied spatio-temporal interpolation of AIRS data in a pre-process. Since their interpolation is based on AIRS data alone, it does not readily provide altitude information. In a later step, they used CLaMS and MIPAS data to create a probability distribution of ash concentration on the z-dimension. Unlike their approach, our work focuses on speed and interactivity in the missing data reconstruction process to support timely decision making. We perform both spatio-temporal interpolation and altitude information reconstruction on the GPU by projecting raw AIRS data points on a synthetic plume model.

4. Approach

The ultimate goal of our visualization framework is to enable the user to answer from the data the questions about the spatio-temporal behavior of the physical phenomena it represents. Answering these questions usually requires the availability of measurements at all times and all spatial locations at which a given phenomenon occurred. Since this is not the case for any given modality, we propose a framework that incorporates expert knowledge with data obtained from

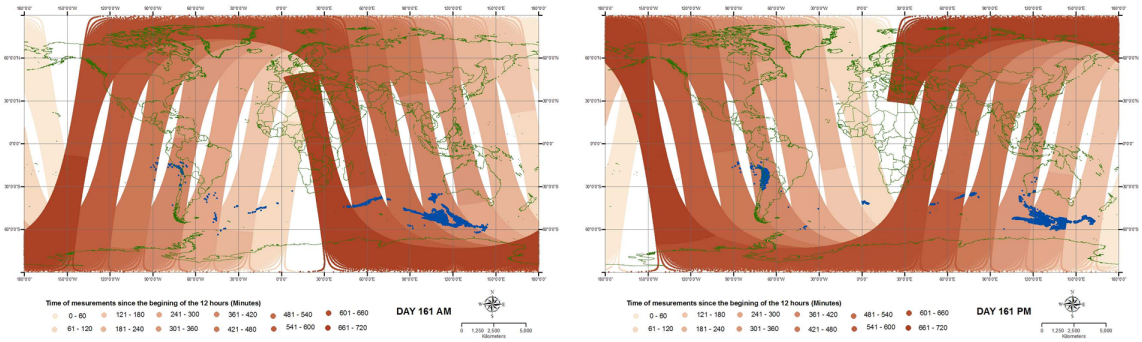

Figure 2: *Sample 12 hour AIRS coverage on June 10th 2011. Morning measurements are displayed on the left while afternoon readings are on the right. Shades of brown encode time while the ash plume is shown in blue.*

simulation to fill the gaps in the data and provide a coherent story to the user. Our solution accomplishes this through three data processing and visualization stages (Figure 3):

1. *Exploratory Analysis:* During this stage, the visualization seeks to integrate multiple data sets in a common reference frame while providing filtering and clutter reduction capabilities to enable the expert to understand the strengths and shortcomings of different modalities, and decide which ones can be most informative to include in a "reference model". One that captures the expected behavior of the studied phenomena, and acts as a prediction for the tracking of ground truth measurements. The resulting model can be further interpolated to increase its spatio-temporal resolution.
2. *Tracking:* The reference model output from the previous stage is fed into the tracking subsystem to act as a prediction for tracking. Given this model, the positions of ground truth measurements can be tracked at any given time instant. This effectively fills the gaps in the data and provides a more complete understanding of how the detected points must have evolved over time.
3. *Visual Feedback:* Tracked results are immediately fed to the rendering subsystem, possibly with special illustrative effects, to provide visual feedback. Both tracking calculations and visual mappings are performed in GPU shaders to maintain interactive rates (Section 4.2).

In all three stages, interactivity is crucial to capture human expertise in the process and to enable visual analytics tasks that guide adjustments to fusion parameters, in order to yield a visualization that tells an as accurate as possible story of what happened in the real world. In previous work [ESGG*14] we proposed a clutter reduction technique to support data exploration at interactive rates, as part of the exploratory analysis subsystem.

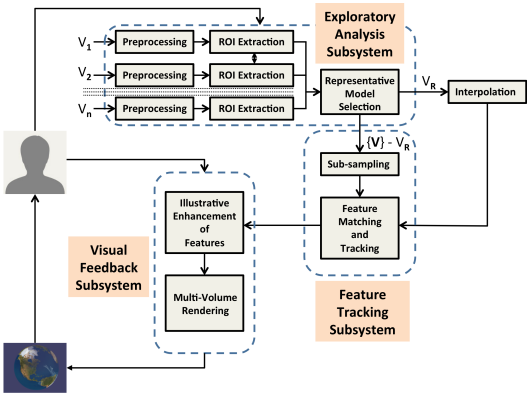

Figure 3: *Data visualization framework.*

In this paper, we focus on the model creation process and the tracking subsystem, while illustrative enhancement of our results is left for future work. We propose a reference model creation technique that samples the space-time domain at known locations, given human expertise and understanding of the origins of the studied phenomena. The generated sample points are advected through space and time by interpolating simulation data to create a synthetic plume, describing the expected behavior of the samples and covering their predicted motion paths as densely as possible. Visualizing this synthetic plume reveals that it is not representative of the ground truth but rather a dense enough prediction of motion pathlines. Our GPU-based approach uses massive parallelism to project real-world measurements onto model pathlines, effectively enabling interactive tracking that gives the user ability to navigate through time while inspecting every ash point's location at any given time instant. This supports visual analytics tasks that help the expert decide whether ad-

justments need to be made to the simulation to enhance their model based on real world measured phenomena.

4.1. Reference Model Creation

The problem of reconstructing missing data from sensor measurements can be theoretically insoluble, just like the problem of obtaining a 3D scene from a 2D image. However, in the case of a 2D image we use certain assumptions about, for example, the size of people, the shades, etc. to interpret the scene. These assumptions are inferred from the viewer's prior knowledge of what the world is like in real life. We apply the same concept to scientific data.

The main assumption we make is that expert knowledge can be combined with one or more of the available datasets to create a reference model that roughly describes the expected data behavior. One possibility is that given a simulation dataset that contains motion trajectories, one can advect ground truth measurements on the trajectories to fill in the gaps in the data and produce a coherent time-animated view. However, this is a naive approach that has several drawbacks. First, the simulation data covers spaces outside regions of interest which can be of little or no relevance to the correction and tracking of data points of interest. In fact, the inclusion of such irrelevant trajectories can increase uncertainty in our prediction model. Second, simulation data is usually irregular which makes it inefficient for GPU based neighborhood search –a process that is crucial to map ground truth measurements to the reference model.

We adopt a reference set construction technique that samples the space at regular time steps within an extracted region of interest (ROI). At each time step, a fixed number of sample points are generated and advected through time by using an interpolation scheme based on simulation trajectories. The technique consists of two modules, the along trajectory adjustment (ADJUSTT) and the spatio-temporal interpolation (SPATIOT). For every generated sample, motion information is obtained from the nearest neighboring points in the simulation dataset by following two steps:

1. The neighboring points are adjusted in time and space to match the time of sample generation,
2. Spatial interpolation is used to obtain translation information to the un-gauged location under consideration based on its relative location to the nearest simulation points.

The technique starts by pre-calculating coefficients for a polynomial regression of a second degree at each simulation point, using the least squares method [LH74]. Regression uses time difference as predictor to calculate the spatial shift as a response. ADJUSTT uses the regression coefficients to calculate the location for each of the neighboring points constituting the nearest simulated neighborhood N_{X_i} to a generated sample point X_i at time t_i (Figure 4*a*).

SPATIOT determines the motion vector during a certain

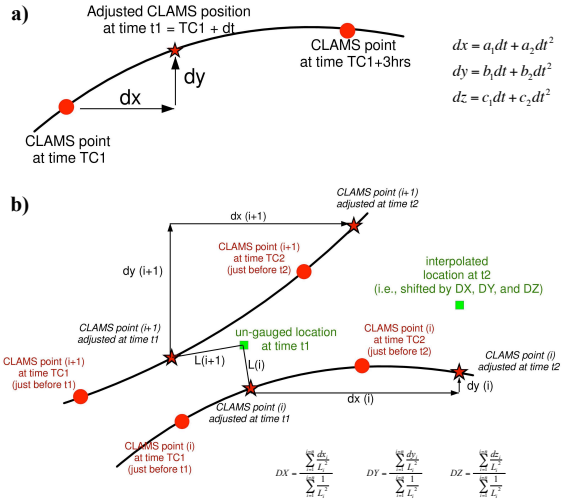

Figure 4: *Data processing: CPU-based spatio-temporal interpolation technique.*

period at any un-gauged location (i.e., not available in the simulation data) by interpolating the corresponding shifts of the neighborhood N_{X_i} to the un-gauged location. The spatial interpolation implemented is based on the inverse weighted principle [She68] in which closer points are given higher weights (Figure 4*b*). Neighborhood size affects the accuracy of the prediction. Considering more neighboring points yields more accurate interpolation results. However, no relevant gain in accuracy was observed with neighborhood sizes of more than 8 points. We found a neighborhood size of 4–16 points to be locally descriptive of the flow field. Full evaluation of different methods of interpolation with varying neighborhood sizes is an interesting topic for future research.

The resulting dataset is a customizable time-regular model grid that can be accessed efficiently on GPU for nearest neighbor search and interactive measurement tracking. The developed technique accepts user input to determine sample generation locations (e.g., based on a known ROI), time frequency of sample generation (e.g., every hour, day, etc.) and the number of points generated at each time step.

4.2. GPU-Based Tracking

The study of temporal evolution of data points that belong to regions of interest is no simple task, especially when data is obtained through multiple complementary sources and the acquired measurements are sparsely positioned in space and time. To address this issue, we created a hypothesized data model that provides a set of assumptions about how the physical phenomenon under examination behaves. Once this model is calculated (Section 4.1), it is passed to GPU memory in a Shader Storage Buffer Object (SSBO). OpenGL's vertex shader operates on a per vertex level and

parallelizes among all vertices that are traveling down the rendering pipeline. In our technique, detection points (AIRS data) are fed as vertex arrays to the shader program and are tracked in parallel. Every shader instance tracks exactly one detection (query point) through time by performing k-nearest neighbor (k-NN) search to extract model points that are in close proximity to the point in question. An aggregate displacement measure is calculated from these neighboring points, by fetching their updated locations at a target time step $t_{destination}$, which is set by user's interaction with a time slider and passed to the shader as a uniform variable. This displacement measure is then used to advect the location of the data point being tracked to time $t_{destination}$. The fact that this process is executed in parallel for each individual point in the dataset being tracked makes it possible to interactively obtain a complete instantaneous image at every time instance, effectively filling gaps in the data, and creating a meaningful time continuum that the user can scroll through to understand and analyze temporal behavior.

Algorithm 1 Vertex Shader algorithm for Point Tracking.

```
t_source = round_vertex_time_to_nearest_step();
t_dest = round_selected_time_to_nearest_step();
offset_source = calculate_buffer_offset(t_source);
offset_dest = calculate_buffer_offset(t_dest);
TF = define_tracking_time_frame(t_dest);
if vertex_inside(TF) then
    [NN] = find_nearest_neighbors(offset_source);
    [NPos_source] = get_point_positions([NN], t_source);
    if tracking_points_exist_at(t_dest) then
        [NPos_dest] = get_point_positions([NN], t_dest);
        vertex_Pos_dest = calculate_IDW(
                          vertex_Pos_source,
                          [NPos_source], [NPos_dest]);
    end if
else
    discard_vertex();
end if
```

Algorithm 1 describes the proposed GPU-parallel approach. The main goal is to allow the user to freely navigate through time with a slider. The time step selected by the slider is target time destination ($t_{destination}$). All detections captured within a time frame TF are time-corrected to the time selected by the slider. We set TF to 12 hours of AIRS readings in the absolute time domain (independent of any given detection's time stamp). The algorithm starts by rounding the time stamp of an input vertex (source) and that of the time slider position as selected by the user (destination) to the nearest time steps in the model plume, and then calculates offsets into the model dataset based on these time steps. Since the number of model points generated at each time step is controlled by the user, $calculate\_buffer\_offset()$ computes the offset at which model points corresponding to a particular time step start in the buffer according to Equation 1, where $sample\_size$ is the number of model points generated at the user-specified spatio-temporal ROI and resolution.

$$offset = \frac{(step)(step+1)}{2} sample\_size \tag{1}$$

The results described in Secion 5 are obtained from a plume with $sample\_size$ of five points generated every hour at the eruption location. Larger $sample\_size$ can give us more dense coverage of the flow field at the price of increased cost for neighborhood lookup. Once obtained, model points within the source time step are searched for the nearest spatial neighborhood of the input vertex. To extrapolate the location of the input vertex through time, the shader program tests if the neighborhood points exist at the destination time step. If so, it uses the same inverse weighted distance interpolation scheme described in Section 4.1 (Figure 4*b*) to calculate the new AIRS location at time $t_{destination}$.

5. Results

The reference model was created by generating a synthetic plume using the interpolation technique described in Section 4.1. A set of sample points is generated at the eruption location on an hourly basis. This plume model is then fed to the GPU SSBO (Section 4.2). The results of our tracking technique produce a continuously evolving plume of AIRS detections, starting at the eruption location, and can be continuously animated using a time slider. In Section 5.1, we evaluate the correctness of both the interpolation scheme used to generate the synthetic plume model and the AIRS corrected plume to assess their accuracy. We then discuss the efficiency of the application on our specified hardware with a summary of performance measures in Section 5.2.

5.1. Accuracy Analysis

We compare our results with raw archived VIS (Visible) and IR (Infra Red) images from geostationary weather satellites at a 3-hour temporal resolution, as obtained from NCDC (National Climatic Data Center) at NOAA (National Oceanic and Atmospheric Administration). The raw images are from seven geostationary satellites (GOES11, GOES12, GOES13, METEOSAT7, METEOSAT9, FEN YUNG 2E, and MTSAT). For the sake of comparison, we pre-processed the raw images to have a common reference frame with our results. First, the raw images, obtained in image coordinates, are geo-referenced in the corresponding satellite vertical perspective projection. The images are then re-projected in the common coordinate system (i.e., geographic WGS1984 system). Next, a mosaic of the projected images is produced in the common coordinate system to generate one image covering the globe at every time step. Figure 5 depicts two sample geo-referenced images from GEOS12 and MTSTAT after reprojection in WGS1984.

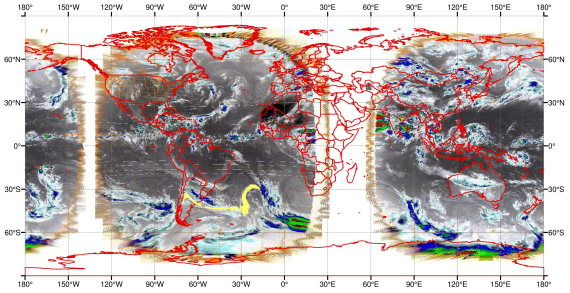

Figure 5: *Two sample IR images at 20110606114500 UTC after re-projection in the World Geodetic System 1984 (WGS1984) common geographic coordinate system.*

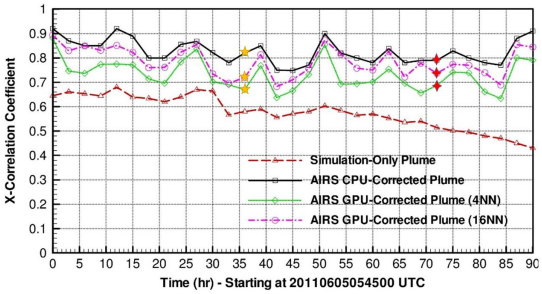

Figure 6: *Cross correlation over 3 consecutive days between geostationary satellites' delineated plumes and four reconstructed plumes: the simulated model plume (red), CPU-corrected AIRS data using CLaMS for interpolation (black), and GPU-corrected AIRS data using plume for interpolation at neighborhood sizes of* 4 *(green) and* 16 *(violet).*

The next step in our validation process is to establish a ground truth plume to which the results of our tracking technique can be numerically compared. To this end, we asked a domain expert to manually delineate the plume regions on the pre-processed weather satellite images. Although this is subject to human judgment, it constitutes the best possible definition of the true shape of the plume. This expert-guided approach is similar to the validation process described in [MVVW05] for DTI bundles extraction and clustering evaluation. The delineated plume is then rasterized into a binary image with pixel value of 2 where the plume exists and 1 elsewhere. On the other hand, the time enabled scatter points produced by our techniques (i.e. the synthetic plume and AIRS corrected plume) are also converted to raster format to facilitate the comparison. A focal window (circular neighborhood with radius 0.1 degrees) is used to smooth the point scatters and reproduce our results in raster format. Again, plume pixels are set to a value equal 2 and global non-plume pixels are set to 1.

The accuracies of the synthetic plume and the AIRS corrected plumes, and their ability to emulate the exact shape of the volcanic plume are evaluated by calculating the cross correlation coefficient (as a measure of pattern matching) with the delineated plume on weather satellite images. Figure 6 plots the cross correlation over three consecutive days from June 5th to 8th, 2011, during the Puyehue Cordón Caulle eruption. The plumes represented by the curves are: *a* the reference plume model created from simulation only (red) *b* a tracked AIRS plume that uses the original CLaMS data for neighborhood lookup (black). This plume is constructed on the CPU using both temporal adjustment and spatiotemporal interpolation (see Figure 4 for details). *c* and *d* represent two tracked AIRS plumes that use the synthetic plume for neighborhood lookup with neighborhood sizes 4 and 16 (green and violet, respectively). These plumes are constructed on the GPU with spatiotemporal interpolation as explained in Section 4.2.

The generated plume model (red curve) has the least accuracy. This is attributed to the fact that the simulated plume is obtained from interpolating CLaMS simulation only. Therefore, errors are progressively accumulated. Our technique updates AIRS corrected plumes every 12 hours with the newly acquired AIRS data (i.e. any errors are reset with every new AIRS data acquisition which is approximately 12 hours apart from the time stamp of every detection). The CPU-based correction has the highest correlation with ground truth. This is attributed to the temporal adjustment step made per detection (Figure 4*a*). That is, detections are not rounded to the nearest time step as is done in the GPU-based tracking. However, we believe that the compromise made to the accuracy of the GPU implementation is an acceptable sacrifice for interactive performance. It should be noted that the accuracy analysis performed here constitutes an assessment of both the CLaMS model and the spatiotemporal interpolation. Inaccuracies are attributed to the whole system of calculation including the CLaMS model itself.

Since we have placed our results and the geostationary data in a common reference frame, we render both point sets together for visual comparison. Figure 7 shows a sample time of the generated plume model (yellow) and the original AIRS data (red) during the first 12 hours on June 8th, 2011. Both sets are displayed on top of IR data captured at 5:54 AM that same day. Differentiating between ash plumes and convective clouds in IR data could be difficult for the untrained eye so feedback from a subject matter expert is essential. Clearly, the plume model fills the gaps in the data but generates more points than is actually needed to represent the true ash plume. Figures 8 and 9 show two different cases at two sample time steps to compare our tracking results with IR images, Part *a* displays raw AIRS data captured over 12 hours of the day, with no correction. Parts *b*, *c*, and *d* display the same corrected plumes from Figure 6.

Sample case 1 (Figure 8): tracking results compared at time 5:45PM, June 6th, 2011, with the GEOS-12 IR image at the

same time. In this case, the differences among the raw data and the corrected plumes are not significant as the raw AIRS data is captured within a small time interval. However, we note that the pattern shaped like a right-to-left question mark has a visible shift in raw AIRS data (as indicated by yellow arrows), while in the CPU corrected plume and the 16-*NN* GPU-corrected plume match IR data perfectly. More importantly, This sample case provides us with a visual sense of the calculated correlation coefficients. The cross correlation coefficients between IR delineated plume and the plumes shown in Figures 8*b*, *c*, and *d* are 0.82, 0.67, and 0.72 respectively (marked in yellow in Figure 6, time 36 hours).

Sample case 2 (Figure 9): tracking results are compared at time 5:45AM, June 8th, 2011, with the METEOSTAT-7 data at the same time. The difference between the raw AIRS data and the corrected plumes is significant because the raw data was acquired over a long time span (almost 12 hours time difference between the patterns on the left and the ones on the right in the raw data panel). The corrections in Figures 9*b*, *c* and *d* were able to reproduce the patterns in proper locations while filling the gaps. The cross correlation coefficients between IR delineated plume and the results shown in Figures 9*b*, *c*, and *d* are 0.79, 0.68. and 0.73 respectively (marked in red in Figure 6, time 72 hours). This case highlights the strength of our tracking technique in detecting locations and times of dangerous corridors in the atmosphere that would otherwise go undetected by raw AIRS data.

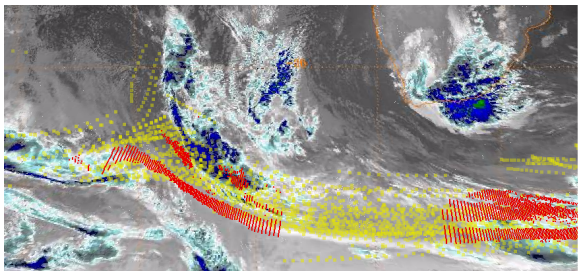

Figure 7: *Generated plume (yellow) shown with raw AIRS data (red) from midnight to noon on June 8th, 2011.*

5.2. GPU Performance

We used a laptop computer with Intel Core i7 CPU and NVIDIA GeForce 740M GPU with 4GB total graphics memory. The GPU tracking runs in OpenGL's vertex shader (GLSL version 4.50). When generating a reference model, the sample size (the number of points generated at each time step) has a major effect on the tracking algorithm's efficiency in terms of memory usage and speed. Since the model generation algorithm creates *sample_size* points at every time step, the number of points a time step t of the plume model is equal to $(t+1) \times sample\_size$, where $t = 0, \ldots, t_{max}$. We use linear nearest neighbor search within a time step which is $O(t \times sample\_size)$. This can be improved upon with a spatial indexing structure, such as a *kd*-tree. An advantage of the generated reference model, over *kd*-tree based neighborhood lookup of irregular simulation trajectories [COJ15], is that looking up a neighbor point's location at the destination time step is straight forward in our plume as we store the point at the same offset within every time step. The indices of the neighborhood points are directly used to fetch their updated locations at the destination time step. In the case of irregular simulation data, we would need auxiliary data structures to store trajectory IDs, and the offset at which each trajectory's vertices exist in memory. After experimenting with different sample sizes, we have found that generating five points at a time resolution of one hour yields satisfactory results in terms of accuracy and performance.

Average frame rates are 17.7, 15.7, 10.2 and 6.2 for 1-NN, 4-NN, 8-NN and 16-NN, respectively. The system performs best with small neighborhoods because it maintains a list of k-NNs sorted by distance from query point. Upon inserting a new neighbor, it shifts all neighbors that are farther away from the query point upward in the list. This sorting step affects algorithm performance. The use of a binary heap could help address this problem to support larger neighborhoods.

6. Conclusions and Future Work

We presented a novel technique for the interactive tracking and fusion of time variant data from multiple sources. This technique is, to the best of the authors' knowledge, the first interactive, GPU-based technique to reconstruct missing data by advecting the sensor measurements to the simulation model. A spatio-temporal interpolation scheme was adopted to re-sample irregular simulation data within a region of interest in order to create a detailed model of expected behavior based on the simulation. The created model is stored in a regular data structure for efficient nearest neighbor search on the GPU. Shader's storage buffers were used to store the created reference model, tracking is performed interactively as the user slides through time. Interactive and accurate results were obtained using the proposed approach. One limitation is the construction time of the reference model on the CPU (order of minutes), but this is done in a pre-process so it doesn't affect interactivity. The GPU algorithm is limited by the number of generated sample points since the grid size grows exponentially, increasing the cost of neighborhood lookups. In the future, we plan to explore other GPU-efficient acceleration data structures for interactive use of more generic reference models.

Acknowledgements

This work was supported by a grant from the National Institute of Mental Health (R21MH100268). Part of this work was done in the scope of the K1 program at the VRVis Research Center.

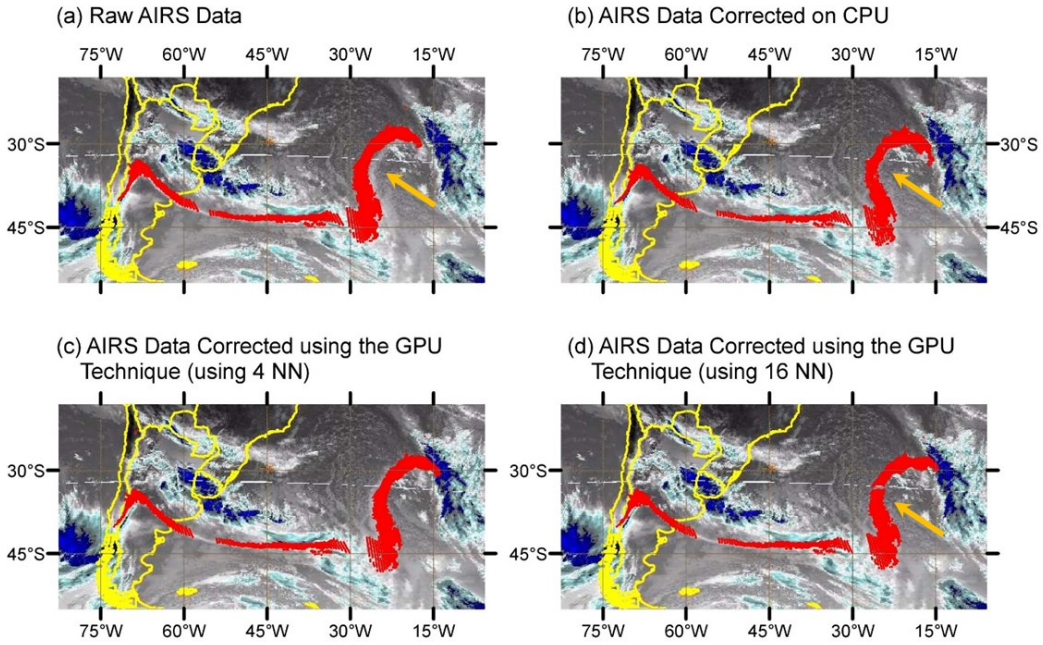

Figure 8: *Comparison at 20110606174500 UTC between GOES-12 IR image, raw AIRS data, and three types of AIRS corrected plumes. (a) raw AIRS (no correction), (b) CPU corrected plume, (c) GPU plume calculated from 4 NN (nearest neighbors) , and (d) GPU plume calculated from 16 NN.*

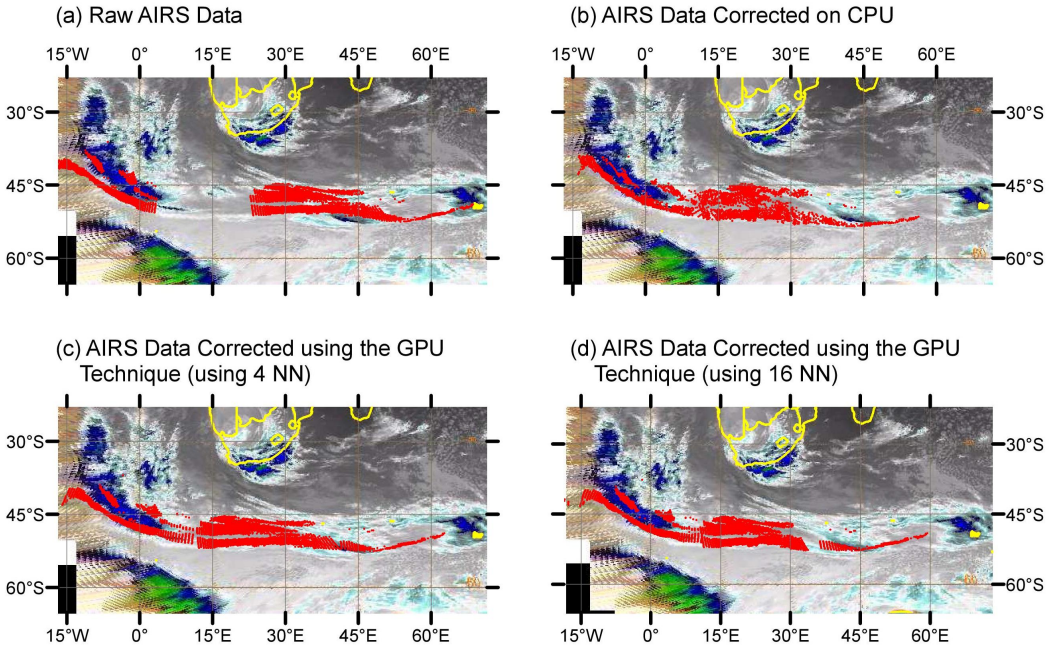

Figure 9: *Comparison at 20110608054500 UTC between METEOSAT-7 IR image, raw AIRS data, and three types of AIRS corrected plumes. (a) raw AIRS (no correction), (b) CPU corrected plume, (c) GPU plume calculated from 4 NN (nearest neighbors), and (d) GPU plume calculated from 16 NN.*

References

[Adi] ADIV G.: Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2

[BRW07] BOSSÉ É., ROY J., WARK S.: *Concepts, models, and tools for information fusion*. Artech House, 2007. 2

[COJ15] CHANDLER J., OBERMAIER H., JOY K.: Interpolation-based pathline tracing in particle-based flow visualization. *IEEE Transactions on Visualization and Computer Graphics 21*, 1 (Jan 2015), 68–80. 2, 8

[ESGG*14] EL-SHEHALY M., GRACANIN D., GAD M., ELMONGUI H., MATKOVIC K.: Streamprobe: A novel GPU-based selection technique for interactive flow field exploration. In *IEEE Visualization Poster* (2014), IEEE. 3, 4

[FN13] FOO P. H., NG G. W.: High-level information fusion: An overview. *Journal of Advances in Information Fusion 8*, 1 (2013), 33–72. 1, 2

[GHSR14] GRIESSBACH S., HOFFMANN L., SPANG R., RIESE M.: Volcanic ash detection with infrared limb sounding: MIPAS observations and radiative transfer simulations. *Atmospheric Measurement Techniques 7*, 5 (2014), 1487–1507. 3

[GSFT14] GÜNTHER T., SCHULZE M., FRIEDERICI A., THEISEL H.: Visualizing the aftermath of volcanic eruptions. IEEE Visualization 2014. Winning entry. 3

[HGM14] HOFFMANN L., GRIESSBACH S., MEYER C. I.: Volcanic emissions from AIRS observations: detection methods, case study, and statistical analysis, 2014. 3

[HHT00] HALL M., HALL S., TATE T.: Removing the HCI bottleneck: how the human computer interface (HCI) affects the performance of data fusion systems. In *Proceedings of 2000 Meeting of the MSS, Nat. Symp. Sensor and Data Fusion* (2000), pp. 89–104. 1, 2

[IEE14] IEEE: http://www.viscontest.rwth-aachen.de/index.html, 2014. 2, 3

[Kar98] KARAKOWSKI J.: Towards visual data fusion. *Military Sensing Series National Symposium on Sensor and Data Fusion International Open Session* (1998). 2

[KS91] KALIVAS D. S., SAWCHUK A. A.: A region matching motion estimation algorithm. *CVGIP: Image Understanding 54*, 2 (1991), 275 – 288. 2

[Lar08] LARAMEE Z. P. R. S.: Vector glyphs for surfaces: A fast and simple glyph placement algorithm for adaptive resolution meshes. *Proceedings of Vision, Modeling, and Visualization 2008, October 8-10, 2008, Konstanz, Germany* (2008), 61. 2

[LH74] LAWSON C. L., HANSON R. J.: *Solving least squares problems*, vol. 161. SIAM, 1974. 5

[LHD*04] LARAMEE R. S., HAUSER H., DOLEISCH H., VROLIJK B., POST F. H., WEISKOPF D.: The state of the art in flow visualization: Dense and texture-based techniques. In *Computer Graphics Forum* (2004), vol. 23, Wiley Online Library, pp. 203–221. 2

[LWS13] LI Y., WANG C., SHENE C.-K.: Streamline similarity analysis using bag-of-features. In *IS&T/SPIE Electronic Imaging* (2013), International Society for Optics and Photonics, pp. 90170N–90170N. 2

[MKG*02] MCKENNA D. S., KONOPKA P., GROOSS J., GUNTHER G., MÜLLER R.: A chemical lagrangian model of the stratosphere (clams). In *Proceedings of the 12th Conference on the Middle Atmosphere* (2002). 3

[MLP*10] MCLOUGHLIN T., LARAMEE R. S., PEIKERT R., POST F. H., CHEN M.: Over two decades of integration-based, geometric flow visualization. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 1807–1829. 2

[MTHG03] MATTAUSCH O., THEUSSL T., HAUSER H., GRÖLLER E.: Strategies for interactive exploration of 3d flow using evenly-spaced illuminated streamlines. In *Proceedings of the 19th spring conference on Computer graphics* (2003), ACM, pp. 213–222. 2

[MVVW05] MOBERTS B., VILANOVA A., VAN WIJK J.: Evaluation of fiber clustering methods for diffusion tensor imaging. In *IEEE Visualization 2005. VIS 05* (Oct 2005), pp. 65–72. 7

[MWSJ14] MA J., WANG C., SHENE C.-K., JIANG J.: A graph-based interface for visual analytics of 3D streamlines and pathlines. *IEEE Transactions on Visualization and Computer Graphics 20*, 8 (2014), 1127–1140. 2

[PL09] PENG Z., LARAMEE R. S.: Higher dimensional vector field visualization: A survey. In *TPCG* (2009), pp. 149–163. 2

[PPH*08] PARK S. I., PONCE S. P., HUANG J., CAO Y., QUEK F.: Low-cost, high-speed computer vision using nvidia's CUDA architecture. In *37th IEEE Applied Imagery Pattern Recognition Workshop. AIPR'08.* (2008), IEEE, pp. 1–7. 2

[RdE90] ROSSOW W. B., DEL GENIO A. D., EICHLER T.: Cloud-tracked winds from Pioneer Venus OCPP images. *Journal of Atmospheric Sciences 47* (Sept. 1990), 2053–2084. 2

[RGJ12] RATTNER A. S., GUILLEN D. P., JOSHI A.: *Generalized Framework and Algorithms for Illustrative Visualization of Time-Varying Data on Unstructured Meshes*. Tech. rep., Idaho National Laboratory (INL), 2012. 2

[RPS01] REINDERS F., POST F. H., SPOELDER H. J.: Visualization of time-dependent data with feature tracking and event detection. *The Visual Computer 17*, 1 (2001), 55–71. 2

[She68] SHEPARD D.: A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference* (1968), ACM, pp. 517–524. 5

[SPY94] SETHI I. K., PATEL N. V., YOO J. H.: A general approach for token correspondence. *Pattern Recognition 27*, 12 (1994), 1775 – 1786. 2

[SSZC94] SAMTANEY R., SILVER D., ZABUSKY N., CAO J.: Visualizing features and tracking their evolution. *Computer 27*, 7 (1994), 20–27. 2

[SW96] SILVER D., WANG X.: Volume tracking. *Proceedings of Visualization '96.* (1996), 157–164. 2

[SYM14] SAUER F., YU H., MA K.-L.: Trajectory-based flow feature tracking in joint particle/volume datasets. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (Dec 2014), 2565–2574. 2

[VW91] VAN WIJK J. J.: Spot noise texture synthesis for data visualization. In *ACM SIGGRAPH Computer Graphics* (1991), vol. 25, ACM, pp. 309–318. 2

[VWPSP96] VAN WALSUM T., POST F., SILVER D., POST F.: Feature extraction and iconic visualization. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (1996), 111–119. 2

[YLC13] YU L., LU A., CHEN W.: Visualization and analysis of 3D time-varying simulations with time lines. *Journal of Visual Languages & Computing 24*, 5 (2013), 402–418. 2