From Heterogeneous Web Streams to Personalized Situation Detection and Control

Mingyan Gao, Vivek K. Singh, and Ramesh Jain,

University of California, Irvine

Abstract

The Web now has enormous volume of heterogeneous data from different sensors and humans being continuously reported from different locations. These data flows can be considered as spatio-temporal-thematic streams. Combined effectively, these streams can be used for detecting situations and saving lives and resources in different applications like disaster mitigation, health, traffic, business planning, and social movements. We describe a framework to combine streams from heterogeneous data sources (e.g. weather data, twitter streams, and traffic information), process them to detect situations, and use the detected situations to aid millions of users. This system uses a unified data model to integrate different web streams, and provides a set of generic operators to detect spatio-temporal characteristics of individual or combined data streams to detect complex situations. The detected situations can be combined with user parameters to provide personalized information and action alerts. We demonstrate the efficacy of the framework and the developed system using multiple application examples.

1. INTRODUCTION

Detecting situations in time to take appropriate actions for saving lives and resources will impact multiple fields like disaster mitigation, health, security, traffic, business planning, and social movements. The data required for answering problems in most of these applications is *already* publically available as real-time web streams. Consider hurricane mitigation as an example. There are data streams for hurricane status (e.g. NOAA.gov), weather forecast (weather.com), population demographics (census.gov), rescue shelters (redcross.org), and traffic directions (maps.google.com). Combined effectively, these data sources can be used to detect and respond to various emerging situations.

With the proliferation of mobile phones, internet of things, planetary-scale sensing, location based computing, and social networks, the web is clearly moving away from its original underpinnings in cyberspace and merging into the physical space. Enormous amounts of data pertaining to different observed characteristics across space and time are being captured and shared over the web. Hence, mechanisms for integrating and reasoning with such spatio-temporal data streams will impact multiple life decisions.

We see two major challenges in this area:
1. **Data Fragmentation:** The data streams live in silos. Heterogeneous data sources (weather.com, traffic data), formats (e.g. KML vs. RSS feeds vs. images vs. maps), and spatio-temporal characteristics make the integration non-trivial. Efforts at combining data (when undertaken), tend to be closely coupled with the sources/formats and require computer science expertise.
2. **Going from Micro to Macro (Situations):** Currently there is a semantic gap between the high level concept space used by humans for decision-making and the torrent of low level data which is available on the Web. The decisions must be made based on situations, but the data is in attribute data streams.

The solution to both these problems lies in realizing the fundamentally different nature of these real-world data streams. Each new data nugget now comes inscribed with **space** and **time** meta-data. Hence, space and time are central to **integration** and **processing** of different data streams.

Looking beyond the media format details (e.g. KML, JSON, images, sensors, tweet streams), the proposed framework focuses on the commonality across different streams. By using a simple unified representation (based on space-time-theme) it indexes and organizes all data into a common representation. Similarly, for going from individual data nuggets (micro-events) to macro-situations it uses a set of generic spatio-temporal analysis operators. A basic assumption in this approach is that spatio-temporal situations are determined by evaluating a large number of data streams that represent different attributes measured by either physical sensors or observed by human-sensors. This means that an engine that is programmed using operators to detect complex events over vast number of data streams can be used to define and detect any arbitrarily complex **situation**.

We define a situation as: **An actionable abstraction of observed spatio-temporal descriptors** .

This definition emphasizes characterization of situations based on measurable spatio-temporal descriptors. Focus on spatio-temporal data (which is the most common connotation associated with situations), scoping of problem only to observable data, and an emphasis on actionable abstractions (as defined explicitly by human domain experts) allows development of a computational framework to define diverse situations and take appropriate actions.

Based on framework, we present a system called EventShop which provides operators for data stream ingestion, integration, situation characterization, and sending out alerts. The system can be graphically configured to detect different situations and undertake corresponding actions. A modular approach makes the system reconfigurable for different applications on the fly , and a simple GUI makes it accessible to large number of users. Hence, it provides non-technical users an opportunity to experiment with different data streams and integrate them for diverse applications, thus democratizing the process of app-building or even the area of situation-awareness.

We demonstrate the efficacy of the system using multiple example applications.

The main contributions of this work are:
1. Describe a generic approach to select, import, and combine real-time data streams and operate on them to detect real world situations for generating appropriate information and actions.
2. Providing a system which allows easy experimentation by users for diverse applications.

The rest of the paper is organized as follows. We review related work in section 2. Section 3 introduces the overall framework and our approach. We describe the system architecture and detailed design of EventShop in section 4. Section 5 demonstrates the utility of the framework and the system through multiple example applications. The paper concludes at section 6.

2. RELATED WORK

The proposed work lies at the intersection of multiple active areas including Geographical information systems (GIS), knowledge representation, web service choreography, media processing, event-based-computing, situation awareness, and social media analysis. Situation awareness is an active area of research. The most commonly cited definition of situations is the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future [Endsley 1988]. However other definitions like knowing what is going on so you can figure out what to do [Adam 1993] abound, and there is no clear consensus. In this work we propose a computational definition of situations which is closely tied to spatio-temporal analytics.

Our work is also inspired by media processing research for events and scene detection (e.g. [Oliva and Torralba 2001]). We borrow multiple ideas on raster data processing but the semantics of our work are very different (space-time). Similarly, we follow the layered notion of information and knowledge representation as frequently used in knowledge representation literature [Rowley 2007].

There has been a growing interest in web service choreography [Arkin et al. 2002] and semantic web services [McIlraith et al. 2001]. However, most of these efforts focus on well-structured (typically ontological) data. We on the other hand make the unified representation extremely simple (STT), which allows many web streams to integrate easily. Mashups and domain specific mashup tools (e.g. MashArt[Daniel et al. 2009]) have started integration of web content. However, most web mashups remain either very shallow (only visual integration, not deeper data integration or analysis) or require IT expert integrators [Chattiet al. 2011]. Efforts like Yahoo Query Language, Google Tables, Yahoo Pipes, and MashArt provide easy, modular, computational tools for users to integrate and analyze web data. This resonates deeply with our EventShop system. Following similar lead, we want to make *spatio-temporal analytics* and action taking modular, intuitive, and accessible to large number of users via a simple GUI.

Multiple recent attempts have used social web (e.g. blogs, Twitter, Facebook) [Bansal and Koudas 2007] [Sakaki et al. 2010] to detect real world events and situations. However our approach is broader and includes social media, sensors, and stream data.

Lastly, the concepts of raster algebra , and spatial analytics are similar to those described in GIS literature. Commercial systems like ArcGIS provide multiple operators on spatial analysis. However those tools have limited temporal analytics, or streaming data support. Some commercial systems (e.g. GeoIQ) have started to explore real-time geo-analytics, but they do not consider the control aspects or personalization of situations.

3. OVERALL FRAMEWORK

Our overall approach is shown in Figure 1. We are building towards Social Life Networks which is a way to connect people to the right resources based on the situations detected [Jain et al. 2010; Jain and Sonnen 2011].

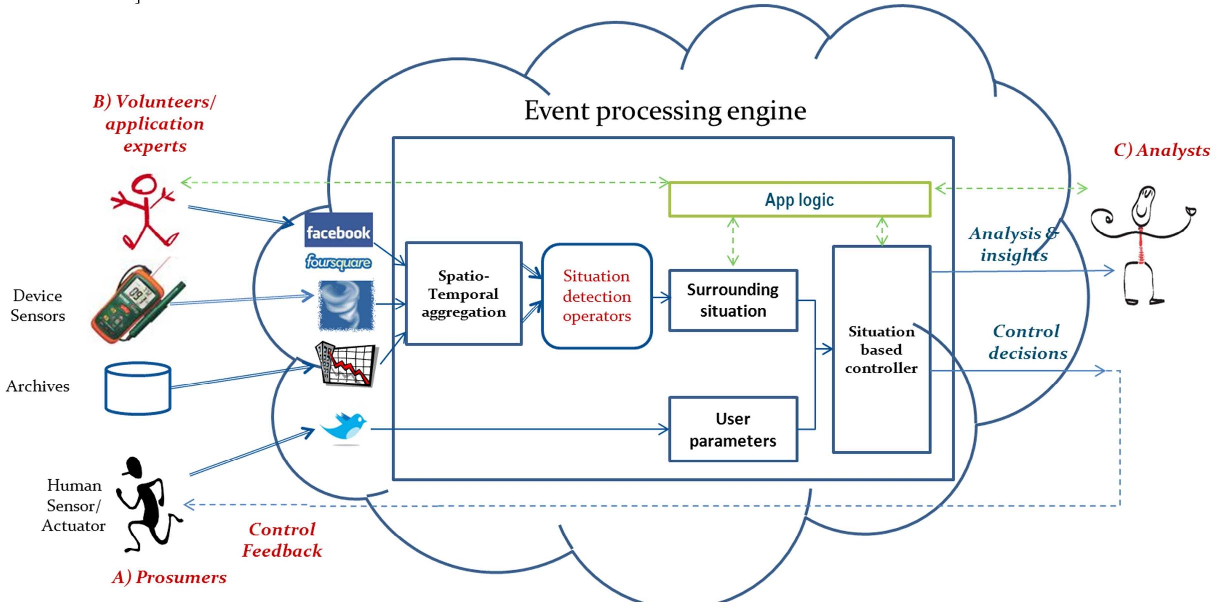


Figure 1. The Overall Framework

The proposed framework considers inputs from human-users as well as any sensor or archived data source. The human users can undertake different roles including data producers, consumers, volunteers, application experts, and analysts. The data streams over the web (e.g. tweets, weather.gov feeds) are translated into a unified format and made amenable for different situation detection operators. Based on application logic, different situation detection operators can be applied. The detected situations (flu outbreak in New England) can be combined with user parameters (e.g. high temperature , location) to send personalized action alerts (e.g. Report to CDC center on 4th street). The analytics are also available to a central analyst who can then take large-scale (state, nation, corporate, or world-wide) decisions.

3.1 Key Concepts

a) Personalized Actionable Situations

Traditional situation detection and decision making has focused on single large scale (e.g. over city, state, country) decision-making. The decisions once made are broadcasted. This is true from health warnings, to weather alerts, to advertisements. Today, we have tools to individually access each user s inputs and combine this with the surrounding situation detected around her. Thus each user can now get a personalized (unicast) alert based on specific situation detected for her.

b) Use of Spatio-temporal-thematic Streams

Let us define a spatio-temporal-thematic data stream as: An uninterrupted flow of data points, each point of which can explicitly be assigned a spatio-temporal coordinate, and an application theme . We explicitly consider streams (and not static or on-disk data) to be our primary data model as practical situations need to be detected in real time to save lives and resources. The prominence of Twitter, Facebook, Stock-market, and satellite data streams in the last decade has convinced us of the importance of this approach. All our operators and representations work on streams and standing query notation.

c) Data Unification

Space, time, and theme (STT) (i.e. **where-when-what**) are the fundamental axes to organize different types of data [Perry 2008]. All incoming spatio-temporal-thematic data points to the framework are converted to, and represented in a common STT format. The framework uses a STTPoint as a fundamental data structure. A ***STTPoint*** is represented as:

STTPoint =¡ latitude, longitude, timeStamp, theme, value ¿.

A flow of STTPoints becomes a ***STT Stream***:

STTStream = (ST TPoint$_0$, ..., STTPoint$_i$, ...).

Values in STTPoints that are collected in a time window over STT stream can be combined and aggregated to form a two-dimensional data grid. The data grid together with related STT information is called ***E-mage***, represented as:

E-mage =¡ SWCoord,NECoord, latUnit, longUnit, timeStamp, theme, 2DGrid ¿.

The SWCoord and NECoord are the southwest and northeast spatial coordinate to which the value at the bottom-left and the top-right cell in the 2D grid corresponds. The unit of latitude and the unit of longitude specify the actual spatial distance, such as 40 miles or 0.1 latitude,of the width and length of each cell in the grid. timeStamp of an E-mage is the end time of the time window over which the STTPoints are collected, which is normally equal to the timeStamp of the last STTPoint in the time window. In the following discussion, the term E-mage may refer to the 2D grid in the E-mage, which should be clear from the context.

A flow of E-mages forms an ***E-mage Stream***, i.e.

E-mage Stream = (E-mage0‹ ▷▷▷‹E-mage*i*‹ ▷▷▷).

Building block of the 2D grid in an E-mage is a single cell. The cell together with STT information is called ***stel*** (spatio-temporal element),

stel =< SWCoord‹NECoord‹ latUnit‹ longUnit‹ timeStamp‹ theme‹ value >.

A stel is a special E-mage, where the size is 1 by 1. As described in section 4, some operators take input and generate output of this special E-mage stream, called ***stel stream***,

stel Stream = (stel0‹ ▷▷▷‹ steli‹ ▷▷▷).

Please refer [Singh et al. 2010] for more details on the data structure. Creation of each data structure and the complete life cycle of data are described in section 4.

3.2 Detecting Situations from Heterogeneous Streams

The process of moving from heterogeneous streams to situations is shown in Figure 2. The unified STT format employed (level 1) records the data originating from any spatio-temporal bounding box using its numeric value. Aggregating such data results in two dimensional data grids (level 2). At each level the data can also be characterized for analytics. The situational descriptor (level 3) is defined by the user (application expert) as a function of different spatio-temporal characteristics.

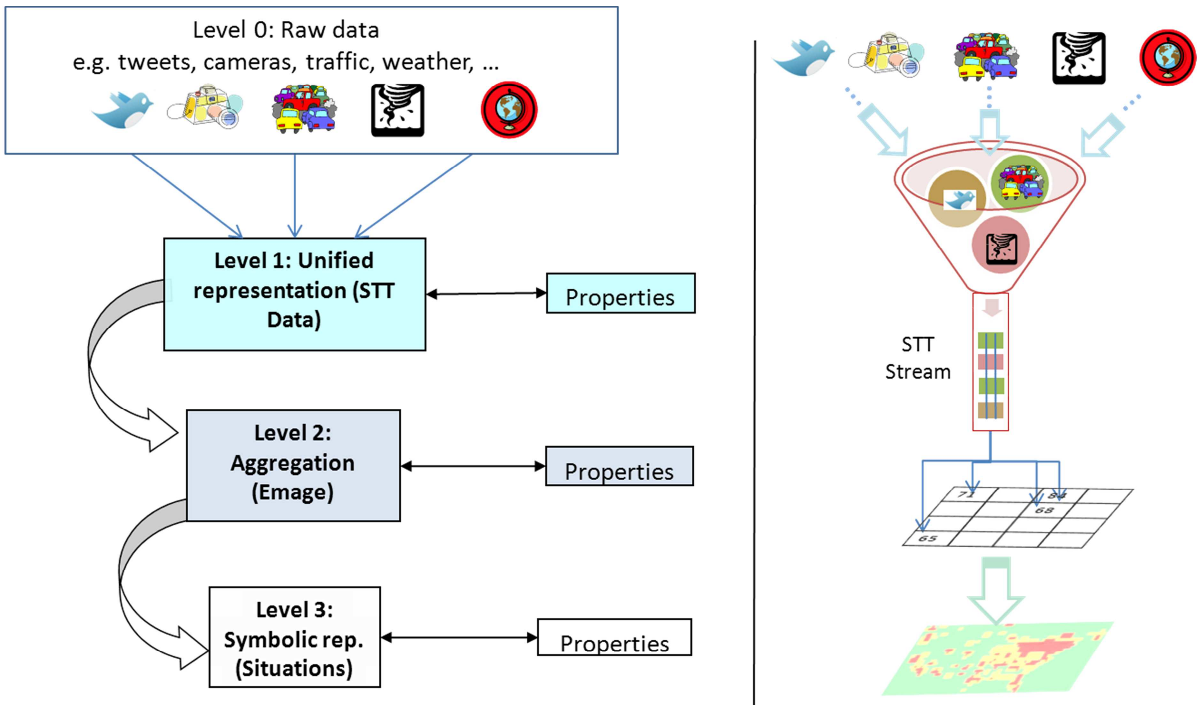

Level 0: Raw data
e.g. tweets, cameras, traffic, weather, …
Level 1: Unified representation (STT Data)
Properties
Level 2: Aggregation (Emage)
Properties
Level 3: Symbolic rep. (Situations)
Properties
STT Stream
71
84
68
65

Figure 2. Approach for detecting Situations

3.2.1 Data Representation Levels.

Level 0: Diverse Raw Data

We support data from different sources. Any sensor data can be associated to a stream based on its location and frequency of creation. Human sensor data, such as tweets and status updates can also be analyzed and converted to measurements related to a particular theme or attribute. Some data sources have tables or databases that are frequently updated to give certain sensory data collected by different agencies. Hence, we support as many different types of raw data as may be relevant. The types of data streams supported in the system will evolve as it is used for diverse applications. For computational purposes we normalize all data streams to numeric streams.

Level 1: Unified Representation

Heterogeneous data needs to be unified. Also, too much data can lead to high cognitive and data processing costs. This layer converts individual attributes into information in terms of what-when-where i.e. STTPoint, and facilitates aggregation of information in next (i.e. E-mage) level.

Level 2: Aggregation

Spatial data can be naturally represented in the form of spatial grids with thematic attributes. As explained, the framework considers ***E-mages***, and ***E-mage Streams*** as its data model. This image-like representation allows application of a rich collection of image and video processing operators ranging from segmentation, aggregation, detecting spatial and temporal patterns, and tracking patterns across space and time. Such a representation also aids easy visualization, and provides an intuitive query and mental model.

Level 3: Situation Detection and Representation

The situation at a location is characterized based on spatio-temporal descriptors determined by using appropriate operators at level 2. The final step in situation detection is a classification operation that uses domain knowledge to assign appropriate class to each stel. This classification results in a segmentation of an E-mage into areas characterized by the situation there. Once we know the situation, appropriate actions can be taken. (See Figure 11 for an example.)

3.2.2 Operators.

Multiple operators need to be provided for analysis and characterization of temporal E-mage streams. The operators considered include *Filter, Grouping, Aggregation, Spatio-temporal Characterization, and Spatio-temporal Pattern matching.* More details are provided in section 4.2.2.

3.3 Personalized Action Alerts

The situations detected can be combined with individual user parameters for customized action taking. We focus on action *recommendation* using the E-C-A (Event-Condition-Action) [Bailey et al. 2002] [Montanari et al. 2007] approach. The individual parameters can be spatio-temporal coordinates, as well as personal micro-events (e.g. sneezing) detected. The spatio-temporal coordinates can be used to direct users to nearest location satisfying certain conditions. Micro-events on the other hand, can be used to configure different E-C-A templates for different user events. Multiple such E-C-A templates can be registered to provide customized alerts to all recipients.

4. SYSTEM DESIGN

Based on the framework, we have built a system called EventShop. The system architecture is shown in Figure 3. EventShop includes both a front end GUI (Graphical User Interface) as well as a back end stream processing engine. In the front end, EventShop borrows the idea of PhotoShop by providing a user-friendly GUI that allows end users to register new data stream sources and formulate queries by combining a rich set of built-in operators. Users are also provided with a GUI tool which allows them to send personalized alerts to relevant people. In the back end, data sources and queries requested from the front end are stored into data source and query databases. Based on the information of registered data sources, EventShop continuously ingests spatio-temporal-thematic data streams and converts them to E-mage streams. Meantime, directed by the registered queries, EventShop pulls E-mage streams from data ingestors to query processor, which process the E-mage streams in each of the instantiated query operators. Besides being converted to E-mage streams, the raw data stream, (e.g. tweet stream) is also made persistent into raw data storage. Raw data together with query results provides necessary personal as well as local situation information to Personalized Alert Unit.

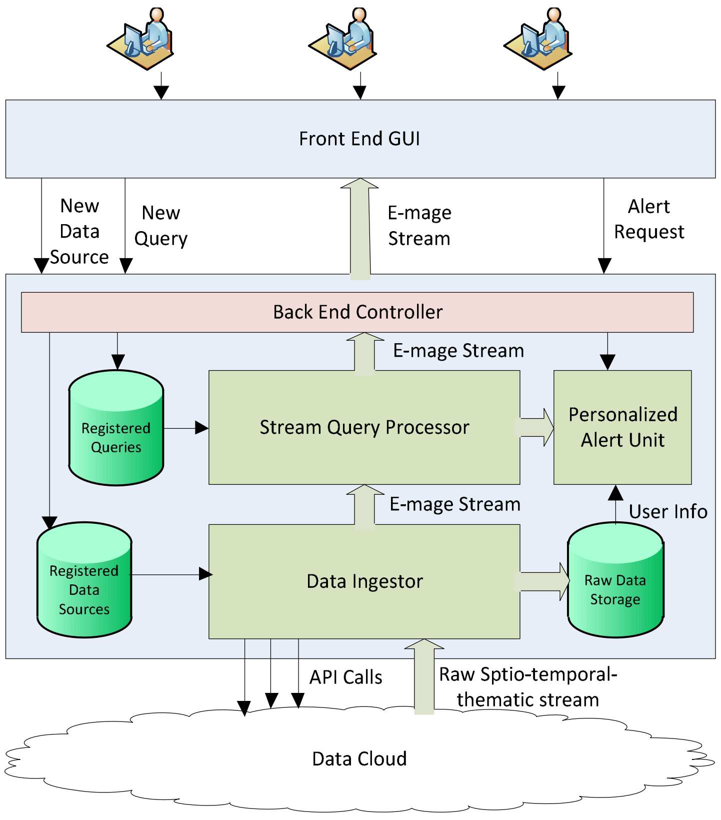


Figure 3. System Architecture of EventShop

In the following discussion, we focus on the description of back end design. The back end system consists of three major components, data ingestor, stream query processor, and personalized action alert unit.

4.1 Data Ingestor

After a new data source is registered and inserted into the data source database, back end controller creates new instances of STTPointIterator as well as EmageIterator (as shown in Figure 4) for this data source and add them to the data ingestor. Data ingestor then connects to the data source and takes the raw spatio-temporal-thematic data stream as input, and relies on these iterators to convert the raw data stream into an E-mage stream. Details of iterators are described below.

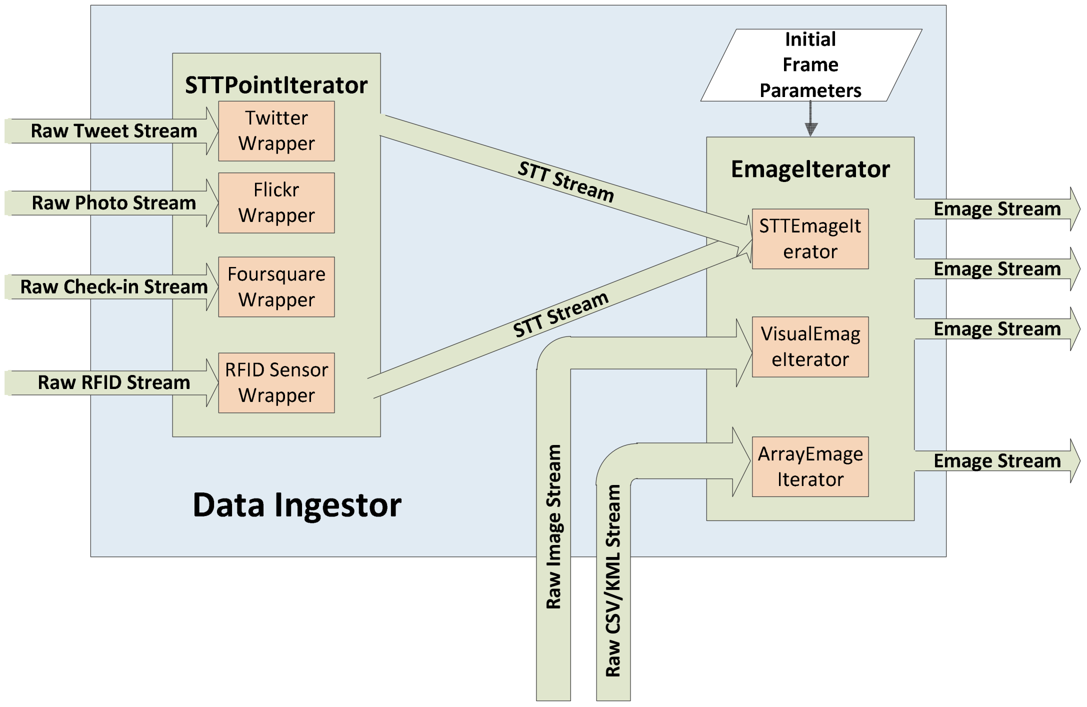


Figure 4. System Architecture of Data Ingestor

4.1.1 *Data Source.*
A data source registered by end users needs to include the following information to enable data ingestion process:
1. **Theme**, which is the central topic discussed in a data source. For example, hurricane, asthma, population, temperature, shelter locations, etc;
2. **URL**, which is the API access link of a data source. For instance, Twitter opens its Stream and Search API which allow users to query tweet stream. Sensor data, such as traffic sensors deployed by PeMS(Caltrans Performance Measurement Systems), is also often available online for user access.
3. **Data Type**. Data sources provide **raw data stream** in different formats. In the case of Twitter, Facebook, and varieties of sensors, single data point, such as a tweet, a status update, and a temperature reading, is the unit that is generated and appended to data stream. Some data sources aggregate their spatio-temporal-thematic data and provide them only in **geo image** format, e.g. pollen count data (http://pollen.com/images/usa map.gif). And some other data sources provide their data collected in a time window in **array** format, such as in CSV(comma-separated values) and KML(Keyhold Markup Language) structure. This type of data gets updated at each time window and essentially forms an array stream. A data source needs to declare itself as one of the types above.
4. **Type Specific Parameters**. Type specific parameters are also required in data collection process.
1) For raw spatio-temporal-thematic data stream, users need to specify the **attributes** that they are interested in. For Twitter and other social media sources, this is a bag of keywords that can cover a specific theme. For traffic and temperature sensors, attributes such as average speed, lane occupancy rate, max temperature and other measures recorded at sensors, need to be specified.
2) For geo image stream, users can either specify the geo coordinate system adopted by the original data sources or provide transformation matrices that can be used to convert original geo images to E-mages which follow the equirectangular projection system. To map image colors to values in E-mages, users can choose between converting images to grey-scale E-mages or assigning values to certain bins of image colors.
3) For array data stream, depending on the data format, users are required to specify column names or tag names from where the spatial coordinate and value of a data point can be extracted.

5. **Frame Parameters**. Output of data ingestor is E-mage Stream. Parameters that are sufficient for specifying the size, resolution and generation frequency of an E-mage (or a frame, borrowing the concept of frame from video) are necessary in the creation of E-mage stream. A set of frame parameters specifies the size of time window (e.g. 10 seconds, 1 hour, 1 day), the synchronization time point (e.g. creating an E-mage at every 10th second, or at 6AM everyday), unit of latitude (0.01 latitude, 0.1 latitude), unit of longitude, spatial boundary including southwest and northeast latitude and longitude values (e.g. for US, southwest point is (24, -125), and the northeast point is (50, -66)). That is,

FP = –window, sync, latUnit, longUnit, swLat, swLong, neLat, neLong˝.

This set of frame parameters which is used to guide the framing of raw spatio-temporal-thematic data stream to create E-mage stream for a data source is called Initial Frame Parameters. We will later introduce Final Frame Parameters as required by each query.

4.1.2 *System Design of Data Ingestor.*

As shown in Figure 4, a data ingestor is comprised of two types of iterators, STTPointIterator and EmageIterator.

1. **STT Point Iterator**

A STTPointIterator is a generic iterator that generates one single STTPoint for each raw data point in spatio-temporal-thematic data stream, and outputs the STTPoint at each *next*() function call. We call a specific STTPointIterator for a data source a wrapper. For example, we can create a wrapper for hurricane related tweet stream, or create a wrapper for Foursquare check-in stream at CDC. A wrapper converts each single data point in the raw spatio-temporal-thematic data stream into a STTPoint. For example, for each incoming tweet in the Twitter stream of a specific topic T, (e.g. hurricane), through the meta-data associated with the tweet, the Twitter wrapper finds out time *tweet time* and location *tweet location* of the tweet, and generates one STTPoint (*tweet location*, *tweet time*, *T*, 1). Value in the result STTPoint is decided by the wrapper. For example, it could be the count, e.g. 1, of the tweet, or the temperature value, depending on the applications. In future, we will open APIs that allow users to implement their own wrappers to ingest more data streams.

2. **E-mage Iterator**

Guided by the initial frame parameters, an EmageIterator generates one E-mage at each time window, and stores the E-mage in its internal buffer until query processor pulls the E-mage by calling *next*() function. As described above, based on the data source type, E-mages could be generated from STT stream, geo image stream and array stream.

1) **STT E-mage Iterator**. By iterating and combining collected STTPoints from the STT stream of a data source, STTEmageIterator generates an E-mage stream for the data source. Based on the initial frame parameters and the spatial and temporal coordinates stored in a STTPoint, a STTPoint is spatially mapped to a cell in the result E-mage. Suppose the original STTPoint is collected at (*lat*, *long*), and the target cell is *cell*(*i*, *j*). Now given the initial frame parameters *FP*,

$$i = \left\lceil \frac{long - FP.swLong}{longUnit} \right\rceil, \text{ and } = \left\lceil \frac{lat - FP.swLat}{latUnit} \right\rceil .$$

The value at the *cell*(*i*, *j*) is normally the sum of values of all the STTPoints that are mapped to the cell. Depending on the applications, however, the aggregate could also be *max*, *min*, *average*. The system defines a set of aggregates from which users can select to combine values.

2) **Geo Image Stream Iterator.** If data points from a data source are already aggregated as a geo image, E-mage can also be created or imported directly from the STTPoints in these geo image formats. The result E-mage has $\left\lceil \frac{neLat - FP.swLat}{latUnit} \right\rceil$ number of rows, and $\left\lceil \frac{neLong - FP.swLong}{longUnit} \right\rceil$ number of columns. And the value at a *cell*(*i*, *j*) in E-mage is computed from the original or normalized values at the pixels of the original geo image that are projected to this cell. Computation of projected area depends on the geo coordinate projection system.

3) **Array Stream Iterator.** Similar to STT E-mage Iterator, each single value in the array is associated with a spatial coordinate, which can be used to decide the E-mage cell to which the value is mapped.

4.2 Stream Query Processor

Different from traditional one-time query issued to database, query in EventShop is standing query. A standing query needs to be registered and instantiated in the system before related data streams flow into the system and get processed.

For each operator of each registered query, as specified by its parameters, back end controller creates an operator instance that performs the real computation. Next, back end controller connects these operator instances as defined in the logical operator tree of the query and forms a runtime operator tree in the stream query processor. Then, as required in the query, controller pulls E-mage streams from the EmageIterators of the appropriate data sources, and feeds the streams to the runtime operator tree. Each operator instance processes the E-mage streams pulled from upstream operators and buffers the results in its internal buffer. The E-mage stream output from the last operator instance is the output of the entire query. System architecture of query processor is shown in Figure 5.

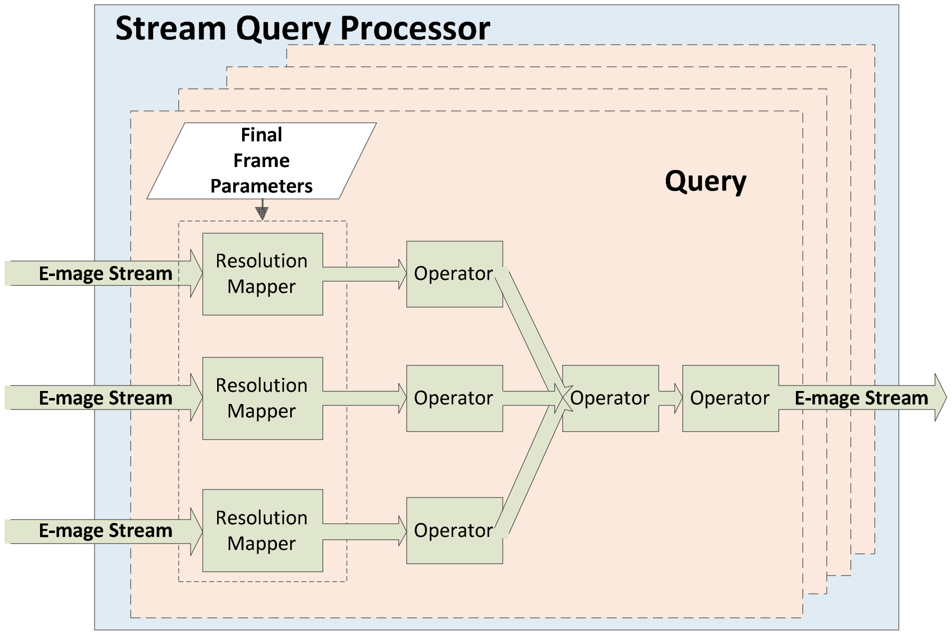


Figure 5. System Architecture of Query Processor

4.2.1 *Query*

A query registered by end users needs to specify two parts of information, a **Final Frame Parameters** and a logical operator tree.

Queries may need to access the same data source but combine them with other data sources at different spatial bounding box, resolution or time window. Therefore, every query specifies a set of final frame parameters, which should be followed by all input E-mages streams. E-mages pulled from data ingestors need to be first mapped to E-mages following the final frame parameters.

The system has a Resolution Mapper (RM) component, which takes an E-mage stream of a data source, the corresponding initial frame parameters, and final frame parameters as requested by a query as input, and generates a new E-mage stream following the final frame parameters. In the RM component, we allow users to select the strategy to perform frame mapping. If E-mages at coarser frame parameters are mapped to finer frame parameters, the set of strategies includes interpolation, repeat, and split. If the mapping is performed on the other way, users can choose strategy like sum, max, min, avg, and majority, for conversion.

The logical operator tree specifies the operator flow in this query. Nodes in the operator tree represent configured operators, and the directed edges between nodes denote the the E-mage flows from the upstream to downstream operators. In our system, users sequentially configure operators before actually issue the query. The complete sequence of operators forms the operator tree of this query. We show examples of logical operator tree in application section. This operator tree is parsed, instantiated and converted to a runtime operator tree by the back end controller. The runtime operator tree is then added to the stream query processor to do the actual processing.

4.2.2 *Operators*

Operators take E-mage streams from upstream operators as input, combine and process them, and create a new E-mage stream as output. Physically, operators are instantiated as EmageIterators. Each operator in the system maintains a buffer that stores the E-mages processed by it. Downstream operators continuously check and pull the next E-mage from the buffers. The final E-mages are stored in the buffer of last operator until they are pulled by the user front-end.

We have defined seven class of commonly used operations to perform on E-mages. For the complete description of data model and operation algebra, please refer to [Singh et al. 2010]. The operators are defined in a closed algebraic structure, which can be easily combined to form queries. In future, we plan to provide interfaces to allow users to define customized E-mage operators.

We provide detailed parameter settings of each operator in EventShop. While the list of options/ configurations for each operator is extensible, here we list the currently implemented options. A summary of operators is presented as Figure 6.

Operator	**Input**	**Output**	**Parameters**
Filter	E-mage Stream	E-mage Stream	• Value range predicate • Time interval predicate • Spatial bounding box predicate • Normalize Values? Y/N Yes, Target Value Range
Grouping	E-mage Stream	E-mage Stream	• Grouping Method ❖ K-Means ✓ Number of segments ❖ Thresholding ✓ Threshold of each segment • Split? Y/N • Color? Y/N ❖ Yes, Color code for each segment
Aggregation	K * E-mage Stream	E-mage Stream	• Aggregate: {max, min, sum, avg, sub, mul, div, and, or, not, xor, convolution} • Normalize Values? Y/N Yes, Target Value Range
Spatial Characterization	E-mage Stream	stel Stream	• Spatial Characteristics: {max, min, avg, sum, epicenter, coverage}
Spatial Pattern Matching	E-mage Stream	stel Stream	• Pattern Input Method ❖ From file ❖ Create a new one ✓ Number of rows ✓ Number of columns ✓ Distribution ▪ 2D Gaussian ▪ 2D linear • Normalize pattern size? Y/N • Normalize pattern value? Y/N
Temporal Characterization	stel Stream	stel Stream	• Time Window Size • Temporal Characteristics: {displacement, velocity, acceleration, periodicity, growth rate}
Temporal Pattern Matching	stel Stream	stel Stream	• Time Window Size • Pattern Input Method ❖ From file ❖ Create a new one

			✓ Sampling rate ✓ Pattern duration ✓ Distribution ▪ Linear ▪ Exponential ▪ Periodic • Normalize pattern size? Y/N • Normalize pattern value? Y/N

Figure 6. Summary of Operators

1. Filter
Filter takes an E-mage stream as input, and filters each E-mage in the stream based on
(a) a value range;
(b) a spatial bounding box;
(c) a time interval;
(d) Values of an E-mage can also be *normalized* into a new range of values by using Filter operator.

2. Grouping
Grouping operator segments each E-mage in an E-mage stream based on a grouping method. Grouping methods supported are K-means and Thresholding.
(a) For K-Means, users need to specify the number of groups. The converging threshold is automatically selected by the system. For thresholding, users provide the thresholds for each group.
(b) Users can specify whether to split the segmented E-mage into multiple E-mages or create a single E-mage with cell values corresponding to the assigned segment.
(c) If the result E-mage is not split, the users can select a color code for visualizing each segment.

3. Aggregation
Aggregation operator combines E-mages from two or more E-mage streams using aggregates. E-mages are aggregated per cell. For example, for each (*i*‹ *j*) pair, the sum aggregate adds *cell*(*i*‹ *j*) of E-mage from each input stream, and stores the sum value at *cell*(*i*‹ *j*) of the new E-mage. Note that the frame parameters followed by these E-mages need to be the *same* to be combined. In addition, the system allows one or more operands of aggregation operator to be scalar or 2D pattern, in which case, every E-mage in E-mage streams can be combined with the scalar or the pattern.
(a) Aggregates supported are *max‹min‹ sum‹ avg‹ sub‹ mul‹ div‹ and‹ or‹ not‹ xor‹ convolution*. Some aggregates, such as *sub* and *div*, can only be applied between two E-mages. *not* is only allowed on one E-mage.
(b) Users can choose to normalize values of the resulting E-mages.

4. Spatial Characterization
Spatial characterization operator works on a single E-mage stream and computes a spatially relevant property of each E-mage in the E-mage stream. The output is a stel stream. Each stel in the stream stores the spatial coordinate where the measure is taken, and the associated value.
(a) Following characterization measures are allowed, *max‹ min‹ sum‹ avg‹ epicenter‹ coverage*. For some characteristics spatial-location is not relevant (i.e. sum, avg, coverage) and is dropped.

5. Spatial Pattern Matching
Spatial pattern matching operator takes E-mages from an E-mage stream and a two-dimensional pattern as inputs, and tries to match the pattern in the E-mage. The output of this operator is a stel stream. The output stels record the location of highest match, and the corresponding similarity value.
(a) The 2D pattern can be input in two ways, either uploaded from an image file or generated by the system. Image files in most of the common image formats, such as bmp and png, are allowed as input. The system also allows users to create some commonly used 2D patterns. The users need to specify the resolution,(i.e. number of rows and number of columns of the pattern), and a spatial distribution including Gaussian2D and Linear2D. To generate a Gaussian pattern, the center point coordinate, x,y variance and the amplitude are needed. For 2D linear pattern, a starting point, starting value, directional gradient and value gradient are required.
(b) Users can choose to normalize the pattern resolution as well as values for matching purpose.

6. Temporal Characterization

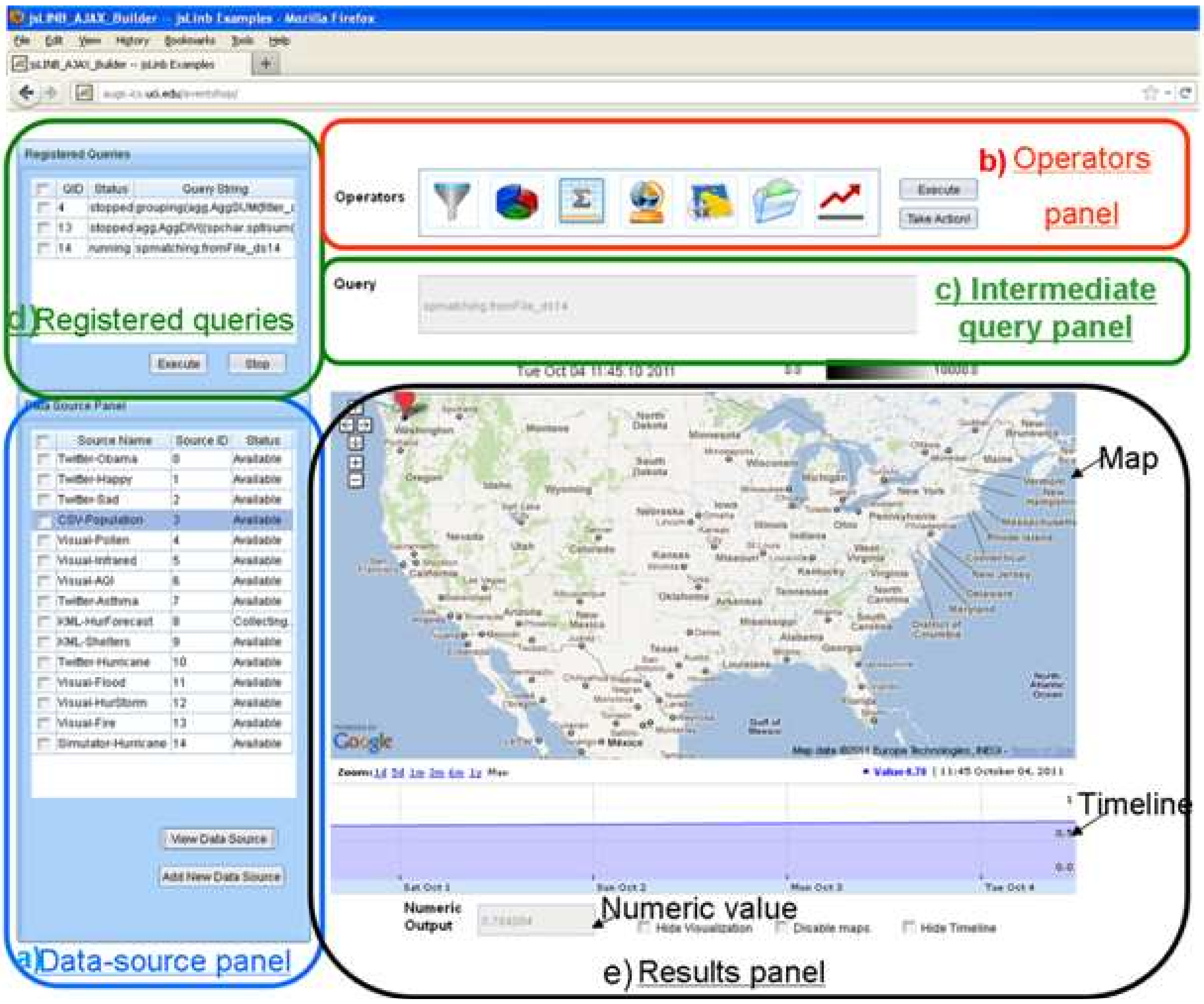
Temporal characterization operator takes a window of stels from a stel stream, and computes its characteristics like *displacement< velocity< acceleration< periodicity< growthrate*. The output of this operator is again a stel stream.
(a) Users need to specify the time window (in seconds), over which the measure is taken.

7. Temporal Pattern Matching
Temporal pattern matching operator takes 1D pattern and a window of stels from a stel stream as input, and tries to match the pattern over the window of stels. Output of this operator is a stel stream, where stel stores the center position of the sub window of stels where the pattern matches with the highest similarity value, and the similarity value.
(a) Users specify the window size in seconds, over which the pattern is to be matched.
(b)The pattern can be input from a file (CSV format) or generated in the system. Currently, we allow *linear, exponential, periodic* patterns. For generating a pattern, the users need to specify the sampling rate and the duration. For example, the sampling rate could be 1 value per 5 seconds, and the whole pattern duration is 30 seconds. For linear pattern, parameters include slope and Y-intercept. Exponential pattern need the base value and the scale factor and Periodic patterns use frequency, amplitude, and phase delay.
(c) Similar to spatial pattern matching, the users could choose to normalize the pattern size, pattern value, or both.

Note that in the current system, we handle spatio-temporal operators (e.g. temporal pattern matching on velocity of *epicenter* of a hurricane) by applying temporal operators on the outputs of spatial operators. See section 5.3 for an example.

4.3 Personalized Alert Unit
Our approach is based on E-C-As (Event-Condition-Actions) [Montanari et al. 2007]. If the user matches certain personal conditions AND lies in an area with prescribed situation, she can be directed to the nearest location matching certain other situation conditions AND / OR sent an alert message.
1) ***User conditions:*** Can be configured on a social media data-source using a bag-of-words approach. These user conditions are predicates of queries that are issued to the raw data storage for retrieving user information, such as user ID, for alerting purpose.
2) ***Surrounding Situation parameters:*** Thresholds on a situation detected at user s spatio-temporal coordinates. The situation is specified as a standing query in EventShop. Thresholds are applied on the result of this standing query.
3) ***Target Situation parameters:*** Thresholds for a desired situation where a user should be directed to. Similarly, the target situation is also specified as a standing query.
4) ***Message:*** The message can consist of:
a) Information about users current situation.
b) Information about nearest target location.
c) Additional information (e.g. Personal message, authority contact).
Currently, the system can send alerts to Twitter users.

5. APPLICATION STUDY
We have implemented the system as described in section 4, which is available at http://auge.ics.uci.edu/eventshop/. Front end GUI of EventShop is implemented in JavaScript, and it is able to send requests to back end controller through Ajax calls. The back end controller which receives requests and sends response to front end is written using Java Servlet. Data ingestor component is implemented in Java. Implementation of runtime operators makes use of OpenCV packages and is written in C++.



Registered Queries

	QID	Status	Query String
	4	stopped	grouping(agg.AggSUM(filter_(
	13	stopped	agg.AggDIV((spchar.spltsum(
	14	running	spmatching.fromFile_ds14

	Source Name	Source ID	Status
	Twitter-Obama	0	Available
	Twitter-Happy	1	Available
	Twitter-Sad	2	Available
	CSV-Population	3	Available
	Visual-Pollen	4	Available
	Visual-Infrared	5	Available
	Visual-AQI	6	Available
	Twitter-Asthma	7	Available
	XML-HurForecast	8	Collecting
	XML-Shelters	9	Available
	Twitter-Hurricane	10	Available
	Visual-Flood	11	Available
	Visual-HurStorm	12	Available
	Visual-Fire	13	Available
	Simulator-Hurricane	14	Available

Figure 7. The GUI of EventShop

A snapshot of EventShop is shown in Figure 7. The basic components are:
a) **Data-source Panel**: To register different data sources into the system.
b) **Operators Panel**: Different operators that can be applied to any of the data sources.
c) **Intermediate Query Panel**: A textual representation of the intermediate query currently being composed by the user.
d) **Registered Queries**: A list of configured queries registered with the system.
e) **Results Panel**: To see the output of the query (which can be presented on a map, timeline, as a numeric value or a combination).

We demonstrate three applications running on EventShop in the following sub sections. Users are free to create new applications by adding new data sources and registering new queries into the system.

5.1 Application for Thailand Flood

We demonstrate a real application of EventShop on suggesting safe locations to people who are trapped in Thailand flood. To realize the goal, we first segment the flooding areas into three groups based on flooding condition and shelter sufficiency. Then for those people who tweeted about flood from the dangerous areas, we send tweets to them and direct them to the nearest shelters in safe areas.

Data Sources

Data sources employed in this application include the map of flood affected areas across Thailand, and shelter map, which are updated every a few hours. We also collect tweets sent from southern Thailand with keywords #ThaiFlood , #Flood , and #ThaiFloodEng . Figure 8 shows the parameters used in ingesting the three data sources. Sample E-mages created from data sources are shown in Figure 9.

Theme	URL	Type	Time Window	Synchronization Point	Spatial Bounding Box	Spatial Resolution
Flooding Area	http://www.thaiflood.com/floodmap	KML Stream	6 hours	0 ms	Southern Thailand	0.01 Lat x 0.01 Long
Shelter Map	http://shelter.thaiflood.com/webservice/request.kml	KML Stream	6 hours	0 ms	Southern Thailand	0.01 Lat x 0.01 Long
Thai Flood Tweet	Twitter Search API	Text Stream	6 hours	0 ms	Southern Thailand	0.01 Lat x 0.01 Long

Figure 8. Thailand Flood Application: Data Sources

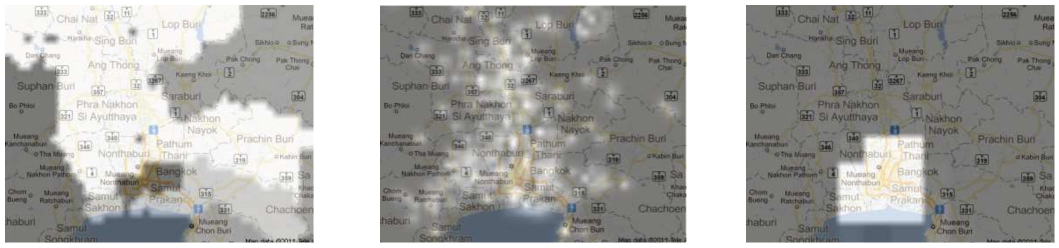
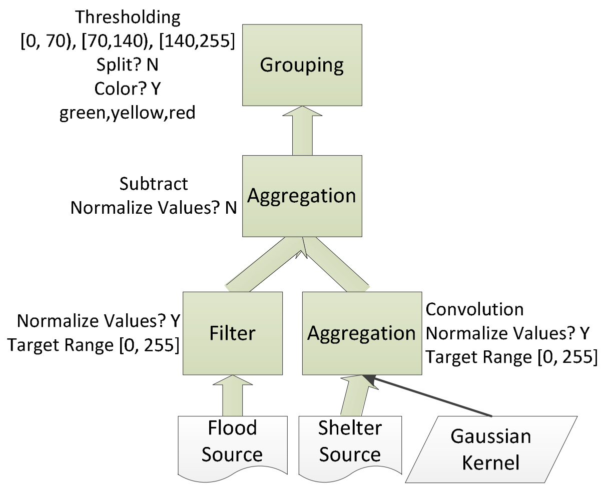


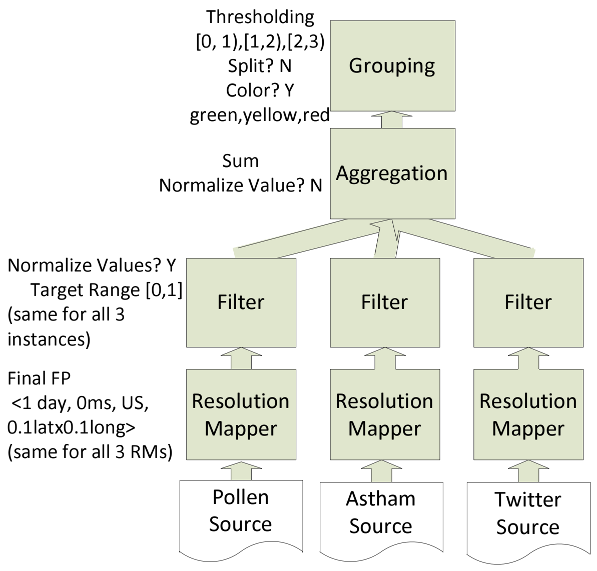
Figure 9. Thailand Flood Application: Sample E-mages

Queries
Final frame parameters of this query are ¡6 hours, 0, Southern Thailand, 0.01 lat x 0.01 long¿, same as the initial frame parameters of all three data sources.

1. Grouping Query
The logical operator tree of the grouping query and parameters of each operator are presented in Figure 10(a).

2. Personalized Alerts
As described in section 4.3, four parameters should be specified to send personalized alerts. We identify people who sent tweets about flood from southern Thailand as satisfying user conditions. If an area is segmented as group 0 or group 1, we identify that area as under threat of flood while lacking shelter protection. The target is to move people to the nearest shelter which is located in an area segmented to group 2. We send information about the nearest shelter location to users.



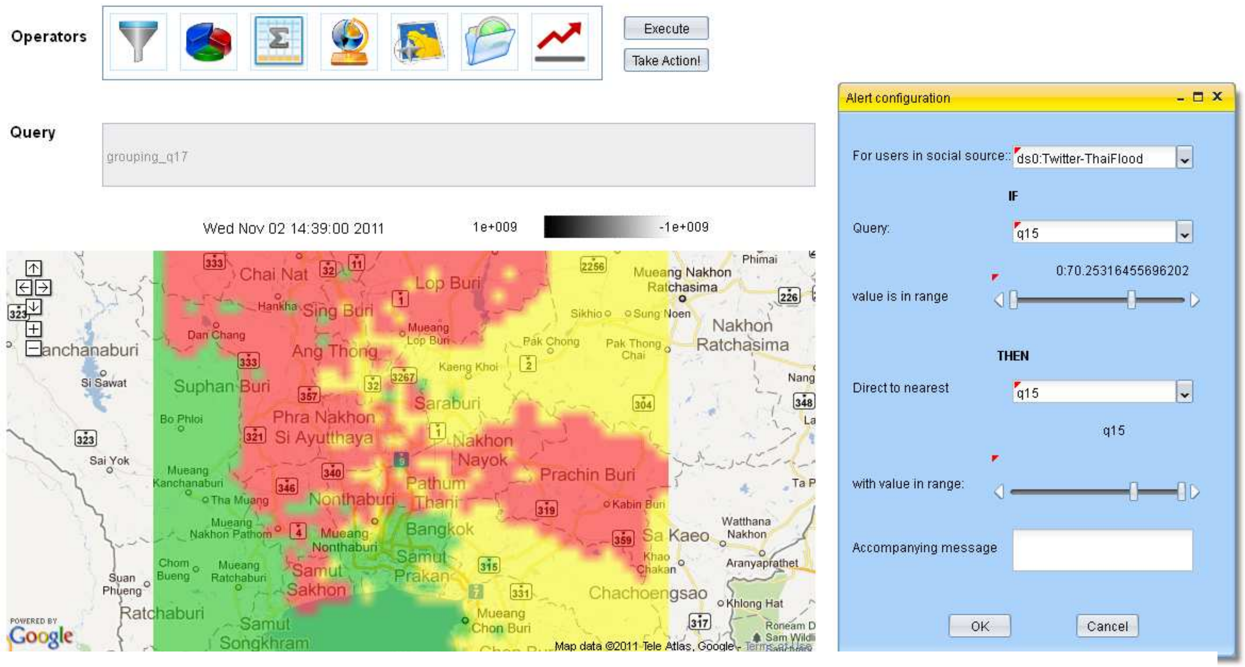


(a) Flooding Area Grouping Query (b) Asthma Index Query

Figure 10. Logical Operator Tree for Queries

Results

Grouping query result as well as alert settings as shown in Figure 11. A snapshot of our twitter account sending tweets is shown in Figure 12. Some of our tweets are being re-tweeted by tweet receivers. Our Twitter account is @SocLifeNetworks.



Query: grouping_q17

For users in social source:: ds0:Twitter-ThaiFlood

Query: q15

value is in range 0:70.25316455696202

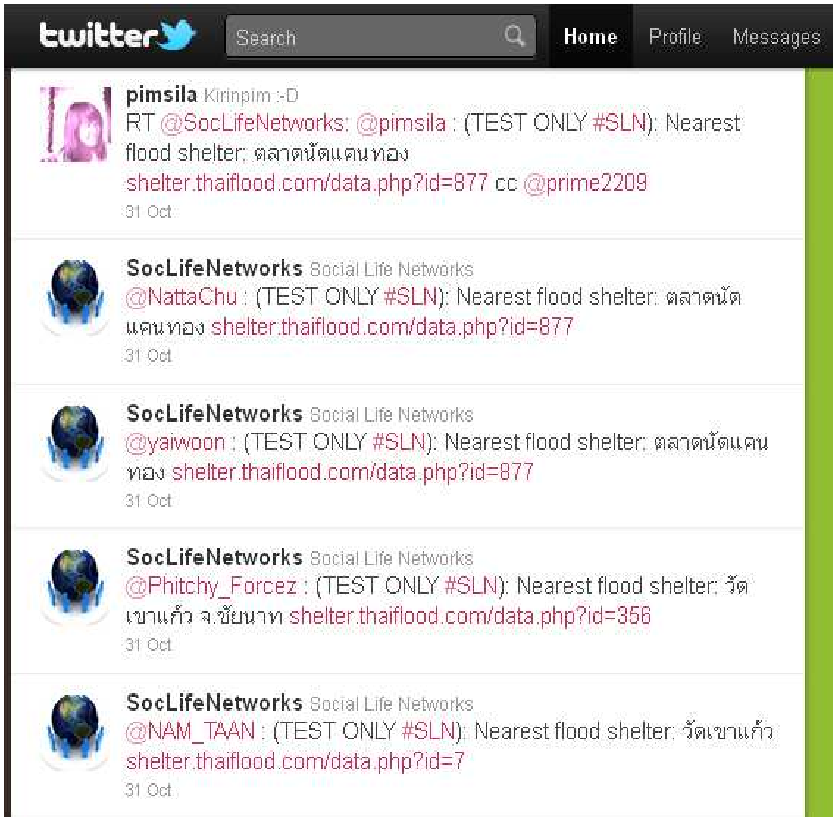
Direct to nearest q15

q15

with value in range:

Accompanying message

Figure 11. GUI of Alert System and E-mage Result of Grouping Query



pimsila Kirinpim :-D
RT @SocLifeNetworks: @pimsila : (TEST ONLY #SLN): Nearest flood shelter: ตลาดนัดแดนทอง shelter.thaiflood.com/data.php?id=877 cc @prime2209
31 Oct

SocLifeNetworks Social Life Networks
@NattaChu : (TEST ONLY #SLN): Nearest flood shelter: ตลาดนัดแดนทอง shelter.thaiflood.com/data.php?id=877
31 Oct

SocLifeNetworks Social Life Networks
@yaiwoon : (TEST ONLY #SLN): Nearest flood shelter: ตลาดนัดแดนทอง shelter.thaiflood.com/data.php?id=877
31 Oct

SocLifeNetworks Social Life Networks
@Phitchy_Forcez : (TEST ONLY #SLN): Nearest flood shelter: วัดเขาแก้ว จ.ชัยนาท shelter.thaiflood.com/data.php?id=356
31 Oct

SocLifeNetworks Social Life Networks
@NAM_TAAN : (TEST ONLY #SLN): Nearest flood shelter: วัดเขาแก้ว shelter.thaiflood.com/data.php?id=7
31 Oct

Figure 12. Alert Tweets sent from EventShop

5.2 Application for Asthma Relief

In this experiment, we combine data from three data sources related to asthma study, and then segment the aggregated data over entire US into three danger zones based on the values in E-mages.

Data Sources

The severeness of an environment to an asthma patient is related to the pollen count and air quality in that area. From crowd sourcing aspect, if many people from a specific area discuss about asthma, it usually suggests that the situation in that area is not very friendly to asthma patients. Parameters of each data source used in this study are shown in Figure 13. The keywords selected for the Twitter source include asthma and allergy . Figure 14 shows sample E-mages from the three data sources.

Theme	URL	Type	Time Window	Synchronization Point	Spatial Bounding Box	Spatial Resolution
Pollen Count	http://pollen.com/images/usa_map.gif	Geo Image Stream	1 day	0 ms	US	0.1 Lat x 0.1 Long
Air Quality	http://www.epa.gov/airnow/today/forecast_aqi_20111101_usa.jpg	Geo Image Stream	1 day	0 ms	US	0.1 Lat x 0.1 Long
Asthma Tweet	Twitter Search API	Text Stream	6 hours	0 ms	US	0.1 Lat x 0.1 Long

Figure 13. Asthma Application: Data Sources

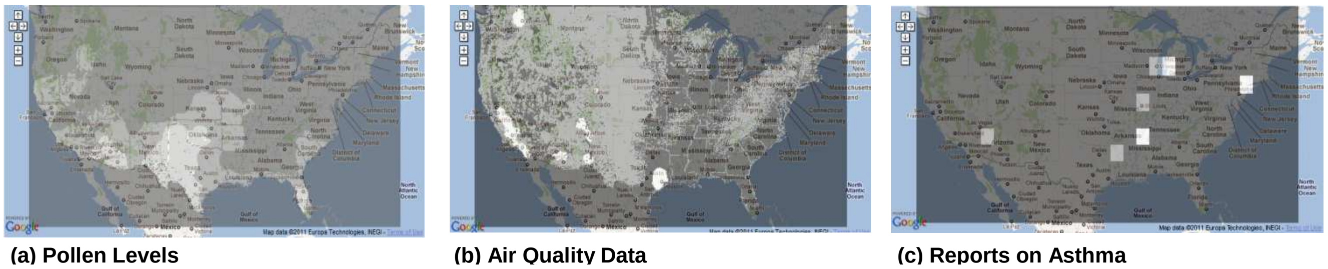

(a) Pollen Levels (b) Air Quality Data (c) Reports on Asthma

Figure 14. Asthma Application: Sample E-mages

Query
The final frame parameters specified for this query are ¡1 day, 0, US, 0.1 lat x 0.1 long¿. E-mages coming from the three data sources with distinct initial frame parameters are mapped to E-mages following the same final frame parameter by Resolution Mapper. The final E-mages from each data source will be generated every day at 12AM with the spatial resolution as 260 x 590.

Logical operator tree of the Asthma Index query and operator parameters are shown in Figure 10(b). Note that the actual settings for data sources, combination operations and threshold values should be guided by domain experts.

Results
One result E-mage of asthma index query is shown in Figure 15.

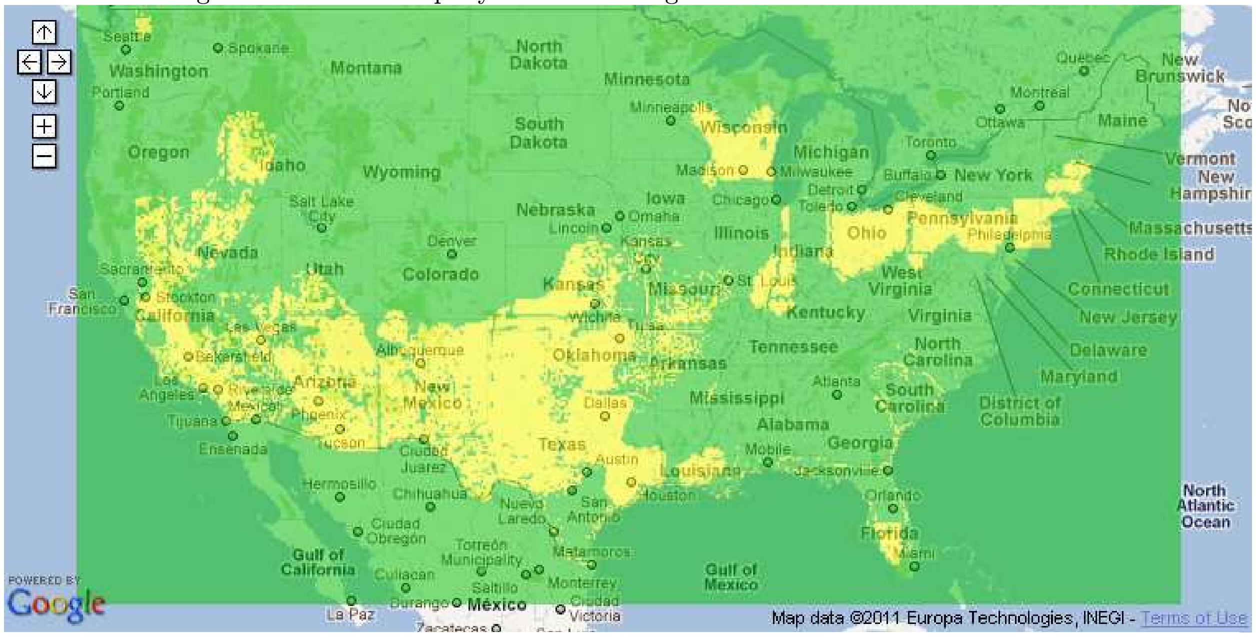


Figure 15. Result E-mage of Asthma Index Query

5.3 Application on Simulated Hurricane Data

In this experiment, we use simulated hurricane data to demonstrate the power of EventShop in performing spatio-temporal characterization and pattern matching queries, as well as the ability in continuously updating query results in real time.

Data Sources
We implemented a STTPoint simulator which generates STTPoints as specified by an initial set of frame parameters. We select 4 locations as centers of 4 2D-Gaussian distributions. We also specify amplitude and variance for each of the distribution. In addition, at each time window, we introduce a hurricane pattern at some random selected location in the generated E-mage. The parameters of the data sources are listed in Figure 16. A sample E-mage can be seen as in Figure 18.

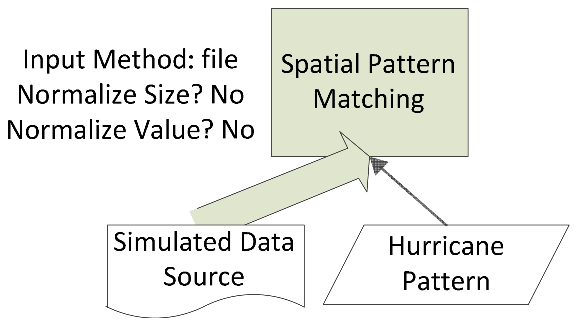
Theme	URL	Type	Time Window	Synchronization Point	Spatial Bounding Box	Spatial Resolution
Simulated Hurricane Data	Local Memory Buffer	STT Stream	10 sec	0 ms	US	0.1 Lat x 0.1 Long

Figure 16. Hurricane Application: Data Sources

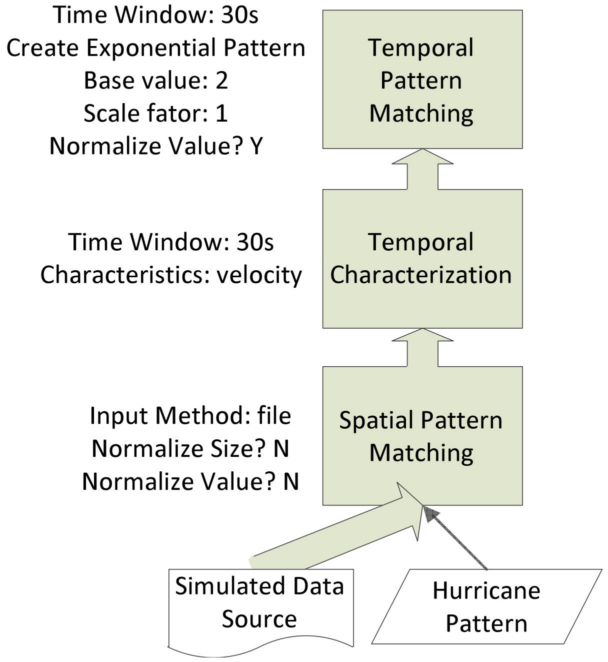
Queries

We demonstrate two queries in this application. The aim of the first query is to search for hurricane in E-mages. Since our E-mages are generated every 10 seconds, the query result also gets updated accordingly every 10 seconds. Based on the first query, the second query studies the velocity of the hurricane being followed. To be specific, the following question is asked: does the velocity increase exponentially with base value 2 and scale value 1?

The final frame parameters for the two queries are the same as the initial frame parameters, i.e. ¡10s, 0, US, 0.1 lat x 0.1 long¿. Logical operator trees and operator parameters of the two queries are shown in Figure 17(a) and 17(b).



(a) Query 1

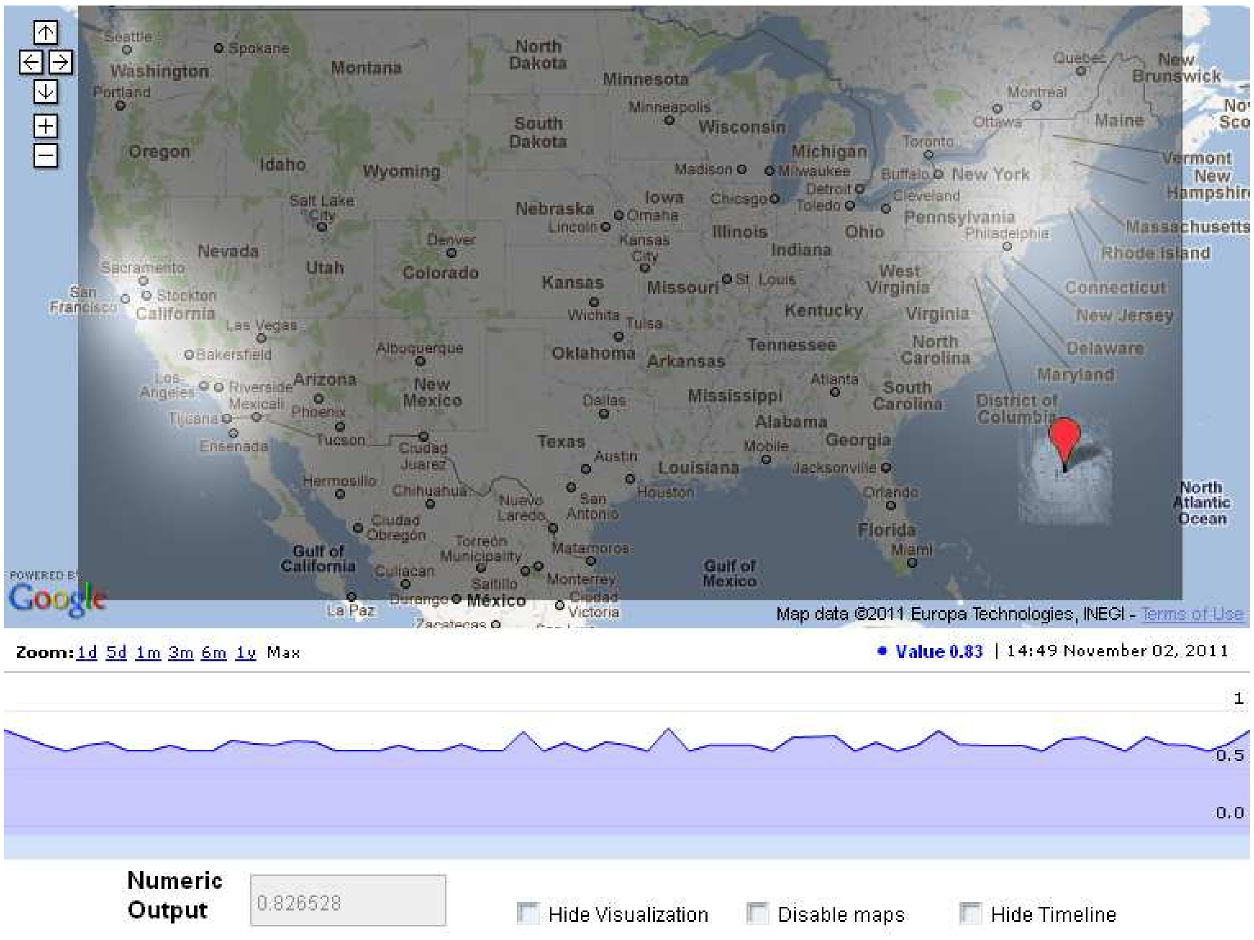


(b) Query 2

Figure 17. Logical Operator Tree for Queries

Results

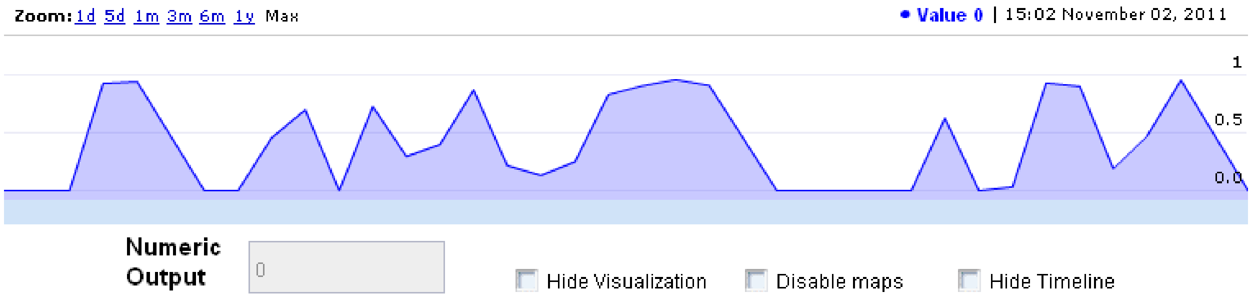
Result of query 1 includes a geo coordinate as well as the similarity value, which are updated every 10 seconds. Figure 18 shows the map, timeline and numeric output view of this query. Result of the second query is a similarity value which gets updated every 10 seconds. Figure 19 shows the results in timeline. Given that the hurricane location is randomly generated at each time window, the velocity of the moving hurricane also shows randomness.



Numeric Output 0.826528

Hide Visualization Disable maps Hide Timeline

Figure 18. Hurricane Application: Result of Query 1


Zoom: 1d 5d 1m 3m 6m 1y Max
• Value 0 | 15:02 November 02, 2011
1
0.5
0.0

Numeric Output 0

Hide Visualization Disable maps Hide Timeline

Figure 19. Hurricane Application: Result of Query 2

6. CONCLUSIONS

Spatio-temporal data streams with data about different attributes are becoming increasingly common on the Web. We need tools to ingest, integrate, and analyze such streams for real-time situation awareness. We have described a generic approach for heterogeneous stream integration, situation detection, and personalized decision making from these streams. The presented system provides an easy, modular way for different users to detect different situations and send personalized alerts to millions of users. The applications abound in areas like emergency mitigation, health care, traffic, and business analytics. Through three different applications we have demonstrated the efficacy of the proposed approach for detecting situations, and taking actions.

Future work remains in defining a declarative language for all the operators defined, building a multi-user, multi-query optimized system, and supporting a generic reasoning and control action mechanisms.

REFERENCES

Adam, E. 1993. Fighter cockpits of the future. In *DASC*.
Arkin, A., Askary, S., Fordin, S., Jekeli, W., Kawaguchi, K., Orchard, D., Pogliani, S., Riemer, K., Struble, S.,
Takacsi-Nagy, P., et al. 2002. Web service choreography interface (wsci) 1.0. *Standards proposal* .
Bailey, J., Poulovassilis, A., and Wood, P. 2002. An event-condition-action language for xml. In *WWW*.
Bansal, N. and Koudas, N. 2007. Blogscope: a system for online analysis of high volume text streams. In *VLDB*.
Chatti, M., Jarke, M., Specht, M., and Schroeder, U. 2011. Model-driven mashup personal learning environments. *Int.*
Journal .
Daniel, F., Casati, F., Soi, S., Fox, J., Zancarli, D., and Shan, M. 2009. Hosted universal integration on the web: The
mashart platform. *SOC*.
Endsley, M. 1988. Situation awareness global assessment technique (sagat). In *NAECON*. IEEE.
Jain, R., Singh, V., and Gao, M. 2010. Social life networks.
Jain, R. and Sonnen, D. 2011. Social life networks.
McIlraith, S., Son, T., and Zeng, H. 2001. Semantic web services. *IS*.
Montanari, M., Mehrotra, S., and Venkatasubramanian, N. 2007. Architecture for an automatic customized warning
system. In *ISI*.
Oliva, A. and Torralba, A. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*.
Perry, M. 2008. A framework to support spatial, temporal and thematic analytics over semantic web data. Ph.D. thesis,
Wright State University.
Rowley, J. 2007. The wisdom hierarchy: representations of the dikw hierarchy. *Journal of Information Science*.
Sakaki, T., Okazaki, M., and Matsuo, Y. 2010. Earthquake shakes twitter users: real-time event detection by social sensors.
In *WWW*.
Singh, V., Gao, M., and Jain, R. 2010. Social pixels: genesis and evaluation. In *ACM MM*.