EventShop: From Heterogeneous Web Streams to Personalized Situation Detection and Control

Mingyan Gao
University of California, Irvine
gaom@ics.uci.edu

Vivek K. Singh
University of California, Irvine
singhv@ics.uci.edu

Ramesh Jain
University of California, Irvine
jain@ics.uci.edu

ABSTRACT

The Web now has enormous volume of heterogeneous data being continuously reported by different sensors and humans 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. We describe a system to combine streams from heterogeneous data sources, 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.

1. INTRODUCTION

With the proliferation of mobile devices 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. These spatio-temporal data streams can be used to detect situations and help people in taking appropriate actions in time for saving lives and resources. Hence, designing mechanisms for integrating and analyzing such spatio-temporal data streams is of critical importance.

To integrate these heterogeneous data streams, we propose a system focusing on the spatio-temporal commonality across 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.

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, scoping of problem only to observable data, and an emphasis on actionable abstractions allows development of a computational framework to define diverse situations and take appropriate actions.

We present the system called 'EventShop' which provides operators for data stream ingestion, integration, situation characterization, and sending out alerts. The fundamental data structure being operated on EventShop for combining spatio-temporal data is ***E*-mage**, each cell of which shows the aggregated user interest on a topic from the corresponding geo-location. An example of E-mage is shown in Figure 1. The EventShop system can be graphically configured to detect different situations and undertake corresponding actions.

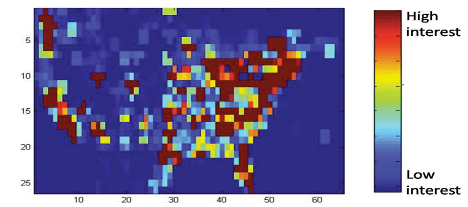


Figure 1: An E-mage showing user interest across mainland US in terms of number of tweets containing the term 'iphone' on 11th Jun 2009

2. SYSTEM DESIGN

2.1 Detecting Situations from Heterogeneous Streams

We first describe the general framework for detecting situations from heterogeneous streams. The process of moving from heterogeneous streams to situations is shown in Figure 2. The unified space-time-theme (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.

2.1.1 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 can also be analyzed and converted to measurements related to a particular theme. Some data sources have tables or databases that are frequently updated based on certain sensory data collected by different agencies. 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. 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.

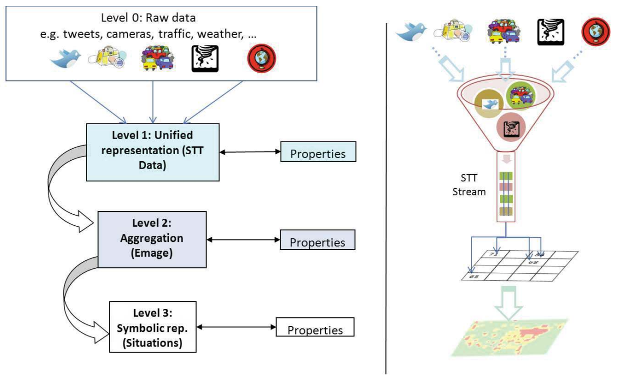

Level 0: Raw data
e.g. tweets, cameras, traffic, weather, ...
Level 1: Unified representation (STT Data)
Properties
Level 2: Aggregation (Emage)
Properties
Level 3: Symbolic rep. (Situations)
Properties
STT Stream

Figure 2: Approach for detecting Situations

Level 2: Aggregation
Spatial data can be naturally represented in the form of spatial grids with thematic attributes. 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 such as segmentation and detecting 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 cell. 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 7 for an example.)

2.1.2 System Architecture

The system architecture is shown in Figure 3. EventShop includes a front end GUI (Graphical User Interface) and a back end stream processing engine. In the front end, EventShop borrows the idea of PhotoShop by providing a 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 provided with a GUI tool that can be used to send personalized alerts to relevant people. In the back end, data sources and queries requested at the front end are stored into main memory databases for data sources and queries, respectively. Based on the information of registered data sources, EventShop continuously ingests spatio-temporal-thematic data streams by using STTPointIterators and converts them to E-mage streams in EmageIterators. Meanwhile, 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.

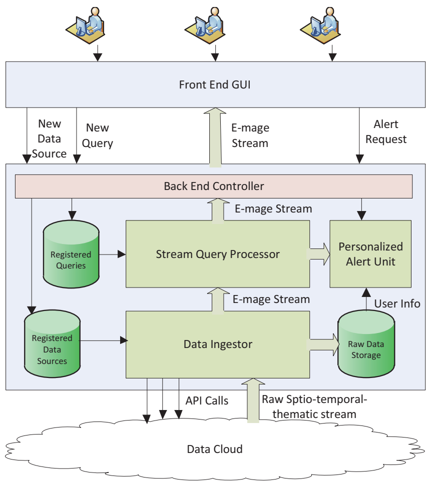


Figure 3: System Architecture of EventShop

2.2 Data Ingestor

After a data source is registered and inserted into the data source database, back end controller creates new instances of STTPointIterator as well as EmageIterator (as will be described below) 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.

2.2.1 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, etc;

2. **URL**, which is the API access link of a data source. For instance, Twitter opens its Stream and Search API which allows users to query tweet stream. Sensor data, such as traffic sensors deployed by PeMS(Caltrans Performance Measurement Systems), is also available online for user access.

3. **Data Type**. Data sources provide **raw data stream** in different formats. In the case of Twitter and many sensors, single data point, such as a tweet, is the unit generated and appended to data stream. Some data sources aggregate STT data and offer them in **geo image** format, e.g. pollen count data (http://pollen.com/images/usa_map.gif). Other data sources provide their data collected in a time window in **array** format, such as CSV(comma-separated values) and KML (Keyhole Markup Language) structure. This type of data gets updated at each time window and forms an array stream. A data source needs to declare itself as one of the types above.

4. **Type Specific Parameters**. 1) For raw STT stream, users need to specify **attributes**. For Twitter and other social media sources, this is a bag of keywords that can cover a specific theme. For traffic and temperature sensors, these are measures recorded at sensors, such as average speed and lane occupancy rate. 2) For geo image stream, users can either specify the geo coordinate system of the original data sources or provide transformation matrices that convert original geo images to equirectangular projected E-mages. To map image colors to values in E-mages, users can convert images to grey-scale E-mages or assign values to certain bins of image colors. 3) For array data stream, depending on the data format, users need 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 for specifying the size, resolution and generation frequency of an E-mage are needed in the creation of E-mage stream. This set of frame parameters includes the length 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)). The frame parameters used to guide the framing of raw STT stream to create E-mage stream is called **Initial Frame Parameters**. We will later introduce **Final Frame Parameters** as required by query.

2.2.2 System Design of Data Ingestor

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 STT 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. A wrapper converts each data point in the raw STT 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 gets time $tweet\_time$ and location $tweet\_location$, 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.

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**. STTEmageIterator generates an E-mage stream by iterating and combining collected STTPoints from the STT stream of a data source. Based on the initial frame parameters and the spatial and temporal coordinates in a STTPoint, the 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, $i = \left\lceil \frac{long - FP.swLong}{longUnit} \right\rceil$, and $j = \left\lceil \frac{lat - FP.swLat}{latUnit} \right\rceil$.

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 $max, min, average$.

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 $\left\lceil \frac{neLat - swLat}{latUnit} \right\rceil$ number of rows, and $\left\lceil \frac{neLong - swLong}{swLong} \right\rceil$ 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.

2.3 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.

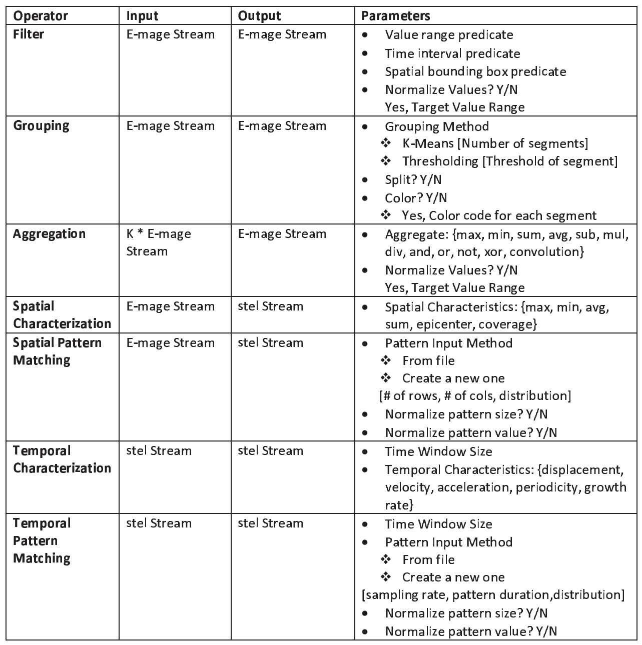
For each operator of each registered query, as specified by its parameters, back end controller creates an operator instance that performs the real computation. Next, back end controller connects these operator instances as defined in the logical operator tree of the query and forms a runtime operator 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.

2.3.1 Query

A query registered by end users needs to specify two parts of information, a **Final Frame Parameters** and a logical operator tree. The final frame parameters are used to specify the data property needed in the query, and it should be followed by all input E-mage streams. The system also 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. 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.

2.3.2 Operators

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 [2]. While the list of options/ configurations for each operator is extensible, here we list the currently implemented options, as shown in Figure 4.



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 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 [# of rows, # of cols, distribution] • 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] • Normalize pattern size? Y/N • Normalize pattern value? Y/N

Figure 4: Summary of Operators

2.4 Personalized Alert Unit

Our approach is based on E-C-As (Event-Condition-Actions) [1]. If the user matches certain personal conditions AND lies in an area satisfying surrounding situation, she can be directed to the nearest location matching certain other situation conditions AND / OR sent an alert message. **User conditions** are used as predicates in queries that are issued to the raw data storage for retrieving user information, like user ID. **Surrounding situation parameters** define the thresholds on a situation detected at user's spatio-temporal coordinates, and **target parameters** are thresholds of desired situation. Additional **messages** are added with alerts. Currently, EventShop sends alerts only to Twitter users.

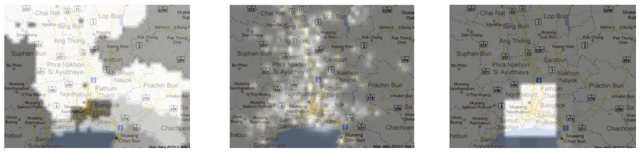
3. APPLICATION STUDY

We recently used EventShop [1] for suggesting safe locations to people who were trapped in Thailand floods (Oct-Nov 2011). As shown in Figure 6, we 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 sent tweets directing 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 (www.thaiflood.com/floodmap, KML format), and shelter map (shelters.thaiflood.com, KML format), which were updated every 6 hours. We also continuously collected tweets sent from southern Thailand with keywords "#ThaiFlood", "#Flood", and "#ThaiFloodEng". Sample E-mages created from data sources are shown in Figure 5.

Final Frame Parameters 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]EventShop: http://auge.ics.uci.edu/eventshop/
Hurricane Demo: http://www.youtube.com/watch?v=Y0pr3x66rvU.



(a) Flood Affected Areas (b) Shelter Map (c) Tweets on Thai Flood

Figure 5: Thai Flood: Sample E-mages

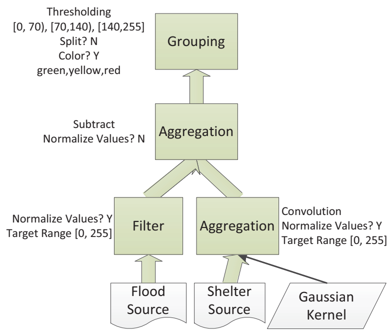
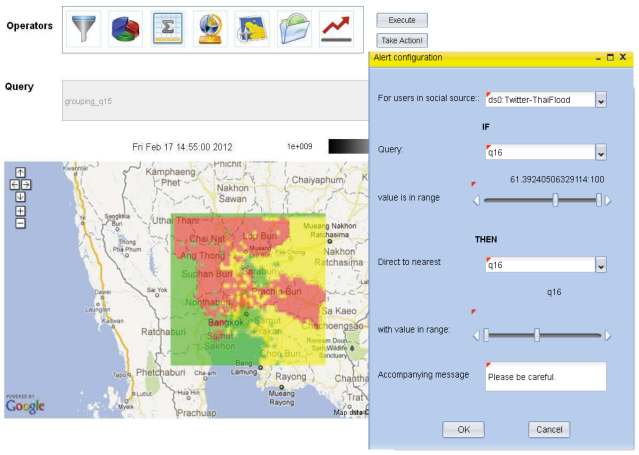


Figure 6: Flooding Area Grouping Query

Personalized Alerts People who sent tweets about flood from southern Thailand are the target users. We send information about the nearest shelter in area of group 2 (safe) to those users located in area classified as group 0 or 1 (unsafe).

Results Grouping query result as well as alert settings are shown in Figure 7. Some of our tweets are re-tweeted by the receivers. Our Twitter account is @SocLifeNetworks.



Query: grouping_q15
Fri Feb 17 14:55:00 2012
1e+009

For users in social source:: ds0:Twitter-ThaiFlood
IF
Query: q16
61.39240506329114:100
value is in range
THEN
Direct to nearest: q16
q16
with value in range:
Accompanying message: Please be careful.

Figure 7: GUI of Alert System and E-mage Result

4. CONCLUSIONS

We 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.

5. REFERENCES

[1] M. Montanari, S. Mehrotra, and N. Venkatasubramanian. Architecture for an automatic customized warning system. In *ISI*, 2007.

[2] V. Singh, M. Gao, and R. Jain. Social pixels: genesis and evaluation. In *ACM MM*, 2010.