Noname manuscript No.
(will be inserted by the editor)

Data stream visualization framework for smart cities

Villanueva F.J. · Aguirre C. · Rubio A. · Villa D. · Santofimia M.J. · López J.C.

Received: date / Accepted: date

Abstract Monitoring smart cities is a key challenge due the variety of data streams generated from different process (traffic, human dynamics, pollution, energy supply, water supply,etc.). All these streams show us what, where and when is happening in the city. The purpose of this paper is to apply different types of glyphs for showing real-time stream evolution of data gathered in the city. The use of glyphs is intended to make the most out human capability for outperforming in detecting visual patterns.

Keywords Smart Cities · Data Visualization · Human behaviour understanding

1 Introduction

Visualization is the study of the transformation, from data to visual representations, intended to develop effective and efficient cognitive processes to gain insight into that data [17].

Precisely, one of the main issues in smart city development is to transform the great amount of data streams into information and finally, into strategic and tactical decisions. In a close future, data streams coming from different sources (energy metering, pollution monitoring, mobility, social media, etc.) will provide information in real-time about what is happening and where is it taking place in any smart city of the world.

Analyzing all these data flows represents a big challenge partially addressed by the Big Data paradigm. However, despite producing useful gathered from raw-data provided by sensors, citizen's smart phones, etc., Big Data fails when unusual circumstances take place. In other words, Big Data can only find what it was originally designed for.

Currently, only human mind can successfully infer new information from raw-data where something unusual is occurring, and only when this raw-data is showed in an appropriate manner. As stated in [19], the way how people perceive and interact with a visualization tool can strongly influence their data understanding as well as the system's usefulness.

This work is inspired in the *Instinctive Computing* term developed by Yang Cai [3] and defined as a computational simulation of biological and cognitive instincts. In this work, as it will be exposed later on, we adapt real-time streaming from smart city to help identifying anomalous patterns in such streams.

Our main motivation is to provide an effective visualization tool for assessing, in a day to day environment, tactical decisions under the smart city application field. For achieving this goal, we will use the excellent capacity of the human brain to identify visual patterns even under complex scenarios. Summarily, two are the main contributions of this paper: first, this work proposes a visualization method for real-time data streams using glyphs; and second, this work also proposes a distributed object-oriented architecture complying with all the needs of the considered visualization paradigm, including scalability, modularity, etc.

The remainder of this article is organized as follows. Next section describes the state of art in data stream visualization and it also establishes a set of requirements for an appropriate visualization method. Section 2 shows how previous approaches fail to handle these requirements Section 3 defines a glyph and proposes

Félix Jesús Villanueva
Paseo de la Universidad 4
Tel.: +34-926295300
Fax.: +34-926295354
E-mail: felix.villanueva@uclm.es

Fig. 1 CCTV surveillance cameras flood our cities (pictures from mylondonpics.com and [2])

serveral examples, nex we will see the proposed architecture along with interface specification and data management. Section 6 is devoted to glyph evaluation with an experiment done by several users. Finally, last section describes some of the most relevant aspects of the implemented prototype along with the main conclusions drawn for this work.

2 State of art

If we analyze current or ongoing methods for monitoring a city, one may find out that most of them are carried out with a forensic perspective. For example, Security cameras are useful, apart from their dissuasion effect, for analyzing the records of an event after it has concluded (figure 1). Pollution-sensor monitoring networks provide with a set of parameters recorded to be analyzed with statistical graphics.

Statistical graphics very often summarize very well the historical data but they are more oriented to strategic decisions, most of the time showing only two variables. For example, if we want to see the temperature evolution we can show a graphic with the evolution temperature over time. It is easy to elaborate real-time heat maps of the smart city and even programming alerts and warnings according to the temperature evolution.

For geo-localized streaming data, "heat" maps represents a very common visualization technique, an example of this type of maps is showed in [14] for urban noise. Combining 3D and heat maps also provide a good visualization tool when visualizing one variable. An example of this technique is showed, for urban air pollution, in [15]. Following the same principle, in [9] shows a Radial Pixel Visualization associated to just one variable and using a gradient color to express different types of waring level.

When we have more than two variables to consider and specially if the variables are correlated, 2D graphics are no so useful and 3D graphics also have their limitations. Sometimes if we want to identify a pattern in some real-time stream, common visualization techniques are very limited. For example, treemap is a good visualization method for seeing the relation between several streams about at most two variables. Streamgraphs show one variable evolution of several streams but it is difficult to introduce new variables.

In the smart city, the variety and quantity of streams encourage us to research in new visualization techniques. For example, if we want know the status of traffic flow in a street, a sensor can be deployed in order to detect velocity and separation between cars (e.g. number of vehicles per minute). The more velocity and separation between vehicles, the less collapsed is the street. However, according to the type of the street, the time of the day, the day of the week, etc. a traffic flow can be normal or abnormal.

The cognitive capacity of human being to detect visual patterns and visual shapes are far too complex to be emulated by machines. In the present work we want to improve the use of such capacity in detecting abnormal situations in the smart city.

3 Data stream visualization

Figure 2 shows the composition of a glyph devoted to represent pollution monitoring. Each variable represented in this graphic is associated to a part of the glyph or lobe with its own scale. The glyph adopts a different form accordingly to the values metered by pollution sensors. The main purpose of this work is to detect abnormal situations even when the surveillance worker does not have specific training. Effectively, as we can appreciate in figure 3, our mind detects easily the different shapes between a mosaic of glyphs in which a dissonant form appears. This is important because detecting an abnormal situation is the first step facing it.

The shape of the glyph has been chosen according to the type of data stream and variables involved in such stream. In this paper, we are going to use an asterisk as example of glyph due its simplicity. However, a research in depth is necessary to study which glyph is more appropriate to each data stream.

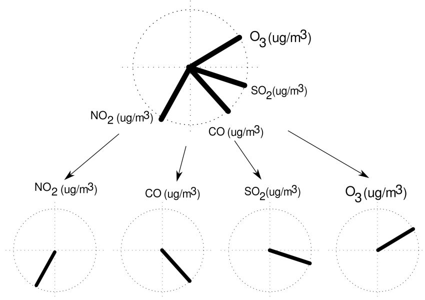


Fig. 2 Example of a glyph composition for pollution monitoring

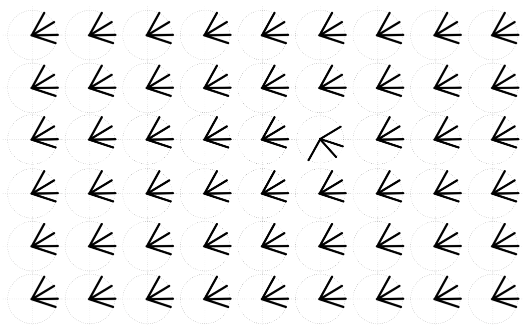


Fig. 3 Anomaly in pollution monitoring

According to the features of the data stream, we can "attach" specific glyphs, for example, in the position where the data stream is generated if the data stream is geo-located. There are more similar features to be analyzed:

- Geo-located: Most information requires to be geo-located in order to be useful. For example, traffic flow sensors provide information about velocity and distance between cars at specific points in the smart city.
- Granularity: what area represents the data stream shown through a specific glyph. Some data streams are representative of small areas whereas other types can represent the whole smart cities.
- Glyph arithmetic: glyph operations need further research in order to explore whether a set of operations could be representative of some city phenomenon.
- Privacy: probably one of the key issues in smart cities is privacy of data involved in different processes. Privacy is out of scope of this paper, however, we would like to point out that data stream distribution is sensible information so the platform should take care about who, when and how is accessing information.

In table 1 we analyze some of the candidate data streams for our visualization framework. We identify three main sources of information in a smart city, as they are citizens, companies and government, however, as the reader can see, we can fine-tune sources to streets, buildings, etc.

4 Framework Architecture

The architecture devoted to support this way of visualization is showed in Figure 4. Our research project has four main sources of data streams:

- From simulations: We can generate data from simulations in order to test the architecture, to explore the evolution/propagation of different parameters, to test different types of glyph or to see the most suitable for analyzing such data stream, etc.
- From databases: Similarly to data streams from simulations, we are using this source of data streams for studying how glyph take different forms according to different data streams recorded. From these records, we are more interested in records with abnormal behaviors in order to see what type of form adopts the glyph.
- From sensors: This is the final source of information and the main purpose of our architecture. Distributed in the smart city, real sensors will provide with data streams related with different physical parameters. We also collect information about logical sensors devoted to get information from different sources. For example, credit-card reader could be an excellent logical sensor, with an appropriate anonimization layer, to analyze and to study economic activity. The surveillance video-cameras are providing real-time information about the number of citizen, the dynamic of the city, traffic, etc., the challenge is to develop algorithms to extract such information.
- From social media: Lately, social media has become an interesting source of information. We believe that we can extract useful data streams for providing valuable information about what is happening in the city at each moment in time and in real-time. For example, the use of hastags associated with a city attached to the messages in Twitter can provide us with a valuable information about events, incidents, etc., in real time.

When we evaluated available platforms to be used on this work, we realized that existent platforms don't fulfill this job requirements, some of these requirements are:

- low foot-print platform and low overhead in order to embed the platform even in the cheapest sensors.

Source	Type	Variables	Source	Type	Variables
citizen	e-health	heart rate, blood pressure, temperature	companies	financial	billing, expenses
citizen	dynamic	velocity, distance, transport	companies	transport	frequency, passengers
government	energy	power, intensity, phase	companies	energy	power, intensity, phase
citizen	financial	number of operations, amount, type of expenses	companies	water	volume, intensity
citizen	food consumption	type, branch, amount	government	pollution	carbon, oxygen, nitrogen
buildings	energy consumption	power, intensity, phase	streets	traffic	intensity, velocity,..
citizen	social media	intensity of the activity, location	companies	social media	activity, location,..

Table 1 Data stream characterization

– enable developers to use the more appropriate programming language due different sources of information and different ways of representation
– high level interface mechanism to easily integrate different sources of information from social networks, wireless sensors, mobile phones, etc.

Most of the platforms devoted to the *Internet of Things*(IoT) [16] paradigm use IPv6 protocol stack and REST-like architectures [7] for modelling applications. Examples of this type of platforms are OpenIoT [11] or ThingSpeak [1]. The REST model works well on Internet due HTML goodness with security rules so you can send information from sensors to any other part in the Internet. When we are in smart city domain, the control we need is much more complex, for example we may need to access directly to sensors to configure frequency, with REST models we have to implement a web server in each sensor. Also the HTML (and the CoAP simplification [12]) has a great verbosity in the protocol so introduces unnecessary overhead. OpenMTC [21], a platform for Machine-to-Machine also has in the REST model its inspiration. Another key problem with these platforms is that they are designed for the restrictive scenario of IoT so fail to model other not so restrictive scenarios. We needed something more flexible and efficient in order to model a smart city.

We are modeling the architecture as a service-oriented distributed platform inside Civitas [20]. Civitas is specially devoted to support the service development process for the Smart City paradigm providing, among other mechanisms, methods for integrating devices ranging from small footprint devices to flexible hardware devices (e.g. FPGA) and a set of standards and tools that can be the backbone of the future smart city ecosystems.

So following Civitas design principles, each flow in our platform is an *Internet Communication Engine* (Ice) service [10], Ice is an object-oriented middleware designed for massive distributed systems. Some issues related with highly distributed systems are mitigated starting from this middleware. For example, IceGrid is a location and activation service easing software management. Similar to IceGrid, IceBox makes easy software distribution, deployment and configuration and security issues are also considered using SSL connections.

We agree that the most successful visualization libraries to date have made it easy for programmers to develop and maintain algorithms that work on many data [6]. Our vision layer can be provided, as we will see in the next section, as an Ice service or as a programming library.

Every entity in *Civitas* exposes its functionality as a set of methods regardless its nature. According to this design guideline, our vision layer is composed of a set of distributed services offered to data streams sources. Every distributed service presents an specialized interface for the data stream.

In the following lines of code we can see a simplification of the pollution interface:

```
module Pollution {
    struct Coord {
        double latitude;
        double longitude;
        double altitude;
    };

    struct Gas {
        long sensorID;
        long timestamp;
        Coord point;
        double CO;
        double CO2;
        double NO2;
        double CH4;
    };

    interface PollutionListener {
        void report(Gas m);
    };
};
```

Similar interfaces are being defined for different domains in order to have a set of common interfaces for smart cities (with similar functionality to POSIX interfaces in Operating Systems Domain).

In the visualization layer, the service supporting that interface draws a glyph according to each received report. In the current implementation, a Gnome/GTK visualization tool shows the data stream received as we can see in the figure 6. Over a map from OpenStreetMap project [8] the pollution service paints and update the glyph associated to each stream identified by a sensorID. Every sensor update its glyph invoking the method *report* which is possible because every deployed sensor implements a low-footprint version of the Ice middleware developed in our previous work [13].

We define also a federation mechanism to distribute the data stream to several sinks of visualization in a broker-like behavior. We can see the PollutionAdmin interface:

```
interface PollutionAdmin{
 void addListener(PollutionListener* lst);
 void removeListener(PollutionListener* lst);
};
```

By using PollutionAdmin interface, any update in a service should be communicated to all services subscribed invoking the method *report* with the new data. With this mechanism we also enable a powerful and flexible mechanism for information federation, filter, processing, composition, etc.

Effectively, this federation mechanism is a key component for the scalability of our framework due we can forward the streams according to, for example, position of the source, we can also filtering the streams or understanding what information is given by the stream an simplify the stream reducing the amount of bytes transmitted.

Figure 5 shows a federation of several services, we identify three main roles in a federation graph:

– Access Services: this type of service is instantiated in nodes close to source streams and they are the first distribution point. According to the type of the stream, in this type of services we can perform some actions in the stream, for example, following with pollution example, this service is instantiated in the gateway between the Wireless Sensor Network and the local area network and we add a timestamp to each sensor value.
– Core Services: this type of role is the core of the scalability and flexibility of the framework, we can instantiate as much core services as we need according to the requirements of the scenario. For example, we could instantiate a core service per district.
– Visualization Service: Each entity of the city which requires to access data should instantiate one of the visualization services available. Police department, fire department, etc. are candidates to have a visualization service in their offices.

In figure 6 we can see the prototype of a Visualization Service implemented for check the framework, the main frame represents the map of Ciudad Real (Spain) from Openstreetmap project, the right lateral frame is

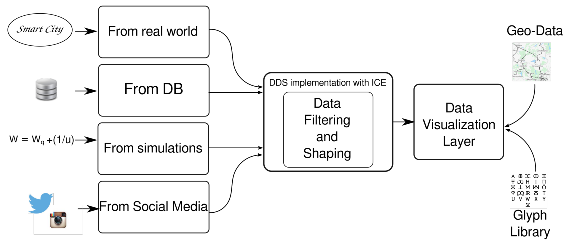


Fig. 4 Architecture

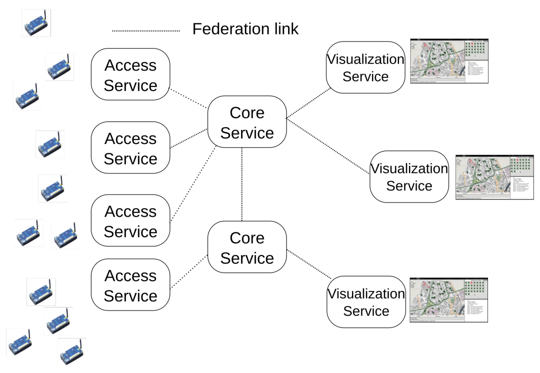


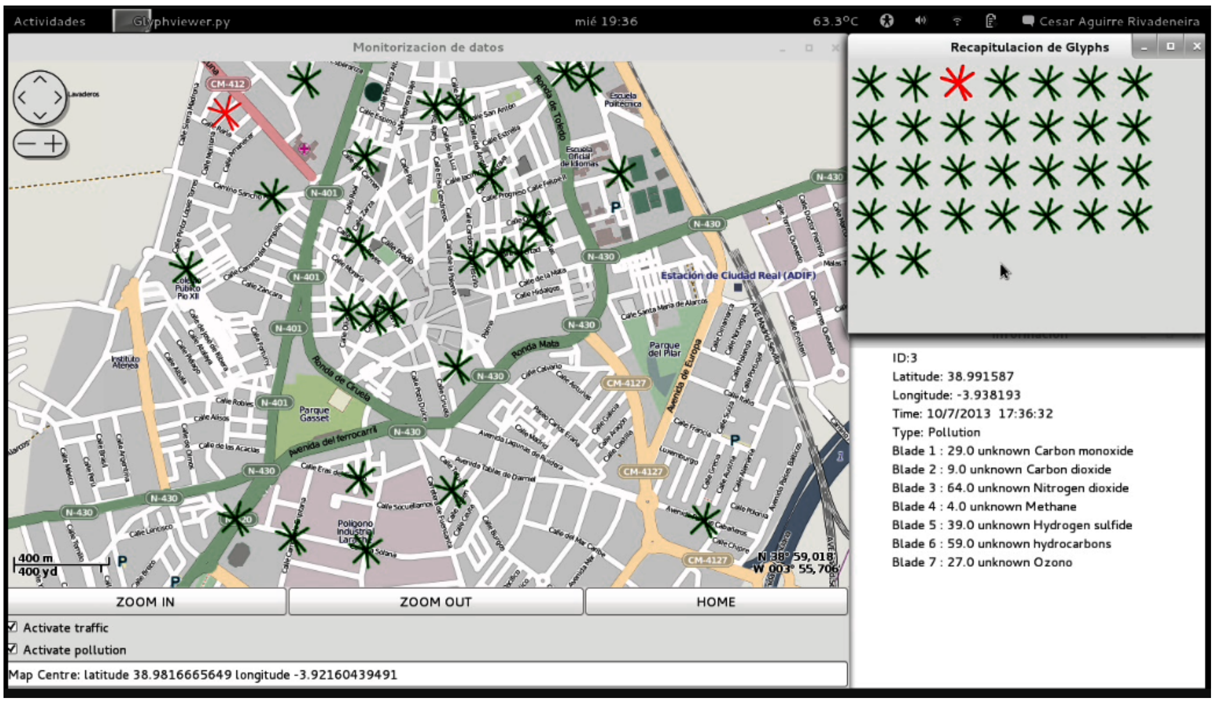
Fig. 5 Federation

Fig. 7 Pollution sensor ready to be deployed

divided into two parts. In the top we can see all glyphs together forming a mosaic to detect pollution anomalies meanwhile in the botton we see specific values of a selected glyph, which is highlighted in red in both frames, the map and the mosaic.

Despite glyphs shown in the 6 are from simulations, real data collected by a test sensor (Figure 7) was used for generating the virtual data-streams. This pollution sensor is one example of the type of sensors which we are going to deploy in a future real scenario.

Pollution example is a good example which show glyphs goodness due to it is difficult to parametrize (i. e. average with threshold) due to it depends of the season of the year, traffic conditions, variables show a



ID:3
Latitude: 38.991587
Longitude: -3.938193
Time: 10/7/2013 17:36:32
Type: Pollution
Blade 1 : 29.0 unknown Carbon monoxide
Blade 2 : 9.0 unknown Carbon dioxide
Blade 3 : 64.0 unknown Nitrogen dioxide
Blade 4 : 4.0 unknown Methane
Blade 5 : 39.0 unknown Hydrogen sulfide
Blade 6 : 59.0 unknown hydrocarbons
Blade 7 : 27.0 unknown Ozono

☑ Activate traffic
☑ Activate pollution
Map Centre: latitude 38.9816665649 longitude -3.92160439491

Fig. 6 Screenshot of the demo app visualizing pollution related glyphs

complex relation among them, etc. Even the individual variables that can be parameterized clearly sometimes cannot be used due to big cities usually exceed individual thresholds established by health authorities. So we are interested in exceptional events that it could represent a risk for citizens. Glyphs are excellent showing that type of anomalies.

Some variables are related and according to specific situations the relation between them could be significant or no if, for example, only is produced in one part of the city. Meanwhile in a non-glyph based scenario to interpret this type of relations requires of high-qualified worker with a glyph tool the identification of unusual events is much more intuitive.

Finally, our current effort is to periodically match the form adopted by a glyph at a specific moment in time to a set of forms adopted by the same glyph associated with specific states of the data stream. Let's take an example with a glyph associated with a data stream of traffic flow in a street involving velocity and separation between cars. We can identify several states of the traffic flow (fast, fluent, slow, traffic jam, etc.) with a specific glyph form (figure 8). Then, we use libraries (the libraries used for recognizing handwritten symbols works fine) to match what is happening in a street, or in other words, what form is taking its glyph and what form from recorded data set look like the current glyph form so we can extract complex states even if the current glyph does not exactly match to any of the previously stored glyphs.

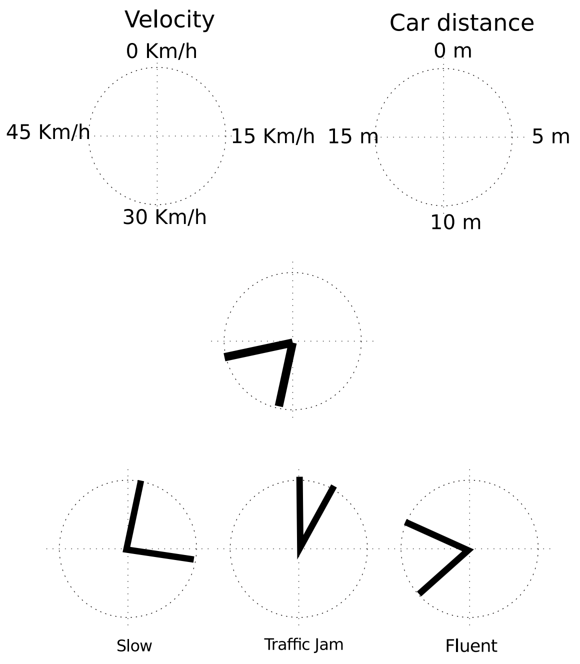


Fig. 8 Glyphs associate with different states of traffic flow

5 Interpreting human actions

Probably one of the key elements in understanding the smart city is to interpret human dynamics. Our target is to predict anomalies in citizen behaviour in order to detect accidents, catastrophe, etc. Extending the visualisation framework to the interpretation of citizen actions implies to monitor even millions of data streams from their smart phones.

Approaches to human action recognition range from intrusive ones such as body sensor networks [4] to non-intrusive ones such as those based on information retrieved from smart phones. However, the visualisation of the information retrieved from different smart phone sensors is not always possible nor feasible using glyphs. For example, the gyroscope sensor provides three values: x, y, and z coordinates. Using glyphs for visualisation this information would not be very useful.

In order to overcome this visualisation problem, information must be turned into a greater granularity. To this end, rather than using glyphs to represent raw sensor data, high level human activities such as walk, run, or kick will be used as glyphs inputs.

The problem of recognising human activities has been faced here as machine learning problem. To this end, several modules have been developed, run, and evaluated. First, an Android application has been developed to collect information from the smartphone sensors. Then a sliding window approach has been implemented in order to segment the signal. The segmented signals are sent to a server in which a data reduction is carried out, employing to this end a Discrete Wavelet Transfor (DWT). This transformation reduces noise and eliminates redundant information. Once the feature vector characterizing signal segments have been constructed out the reduced and segmented signal, these feature vectors are subjected to an additional dimensionality reduction. To this end, feature vectors are then clustered. After that, each feature vector is described as an histogram measuring the distance to each of the identified clusters. This process is known a Bag of Words. Histograms will be provided to a Support Vector Machine (SVM) classifier[18], that will compute a model for the training dataset and will use that model for later recognition processes. An outline of the system is depicted in Figure 9.

The data used for the SVM classifier to generate a model has been recorded using a Samsung S4 smart phone. Out of the 9 sensors available in this smart phone, only the gyroscope and accelerometer were used during the data gathering stage. 16 volunteers participate in the experiment from which 154 actions were

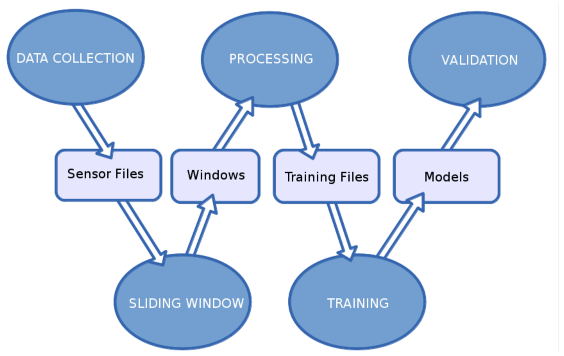


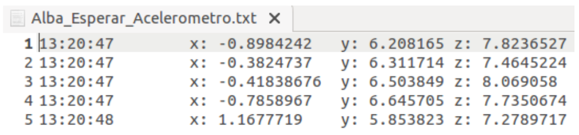
Fig. 9 Outline of the different stages involved in the process

recorded altogether. Each action was manually labelled with a number between 1 to 14 as follows:

1. Wait: The actor is still and therefore no action is being carried out.
2. Walk: The actor is walking.
3. Run: The actor is running.
4. Sit down: The actor lower down and sat in a chair.
5. Stand up: The actor is previously sat in a chair and stands up.
6. Lay down: The actor is previously standing up and lays down in a flat surface.
7. Fall down: The actor is walking and suddenly lower down to the floor level pretending a fall down.
8. Go upstairs: The actor is going upstairs.
9. Go downstairs: The actor is going downstairs.
10. Elevator moving up: The actor gets into the elevator and moves up.
11. Elevator moving down: The actor gets into the elevator and moves down.
12. Punch: The actor moves his/her arms as though he/she was punching something or someone.
13. Kick: The actor moves his/her legs upwards as though he/she was kicking something or someone.
14. Push: The actor moves violently his/her arms as though he/she was pushing something or someone.

One file is generated per action, actor, and sensor. Since a total of 154 actions were recorded using two sensors per action, 308 files were used as the SVM classifier input. The content of the file were organised as depicted in Figure 10. Sensor values are timestamped, and printed one value per line. In this case, since the accelerometer has three values per measure, x, y, and z are used to identified the collected values.

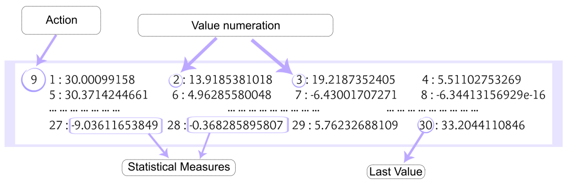
The obtained files out of the data collection process are organised in two sets: One for the accelerometer model and a different one for the gyroscope. Two models will be obtained so therefore two training stages need to be carried out. For every training stage, data need



Alba_Esperar_Acelerometro.txt

```
1 13:20:47          x: -0.8984242   y: 6.208165 z: 7.8236527
2 13:20:47          x: -0.3824737   y: 6.311714 z: 7.4645224
3 13:20:47          x: -0.41838676  y: 6.503849 z: 8.069058
4 13:20:47          x: -0.7858967   y: 6.645705 z: 7.7350674
5 13:20:48          x: 1.1677719    y: 5.853823 z: 7.2789717
```

Fig. 10 Extract of a file containing data of the acceleromenter, for the waiting action carried out by actor Alba



```
9 1 : 30.00099158    2 : 13.9185381018    3 : 19.2187352405    4 : 5.51102753269
  5 : 30.3714244661  6 : 4.96285580048    7 : -6.43001707271   8 : -6.34413156929e-16
  ... ... ... ... ... ... ...       ... ... ... ... ... ... ... ... ...       ... ... ... ... ... ... ... ... ...
  27 : -9.03611653849  28 : -0.368285895807  29 : 5.76232688109  30 : 33.2044110846
```

Fig. 11 Example of data stated in the SVM format

to be labelled using the format stated by the Libsvm library[5] as depicted in Figure 11.

Table 2 summarizes the obtained accuracy rates for the activity recognition system. The diagonal of the confusion matrix represents the number of actions that have been correctly recognized, whereas the other cells represent the actions that were wrongly recognized and also, with which actions were they mistaken.

The output of the training process is the generation of two models, one per sensor used in the training stage. This output turns into the input of the following stage, as it is the testing or recognising stage. At the stage, data are being retrieved from the accelerometer and gyroscope and combined with the aforementioned models in order to obtain a patter match. Smart phone sensors are queried periodically and the retrieved information is provided to the SVM classifier along with the model corresponding to the sensor being analysed. The classifier will output a number, from 1 to 14, corresponding to the action with a higher confidence degree. This action is afterwards provided to the visualisation tool that translate the action in a characteristic glyph.

The use of glyphs for representing the activities carried out by the citizens of the smart city opens up many more opportunities. For example, a higher level analysis could be carried at the glyph level in order to determine the evolutionary patterns. In park area, people is expected to walk, approach a bench and sit down for a while. This sequence will give rise to a characteristic change pattern in glyphs that can be used to model normality. However, if punching or kicking activities are being recognised in a context under which these are not normal activities, alerts can be triggered. Visual patterns can be easily recognised once sensor information has been translated into high level human activities.

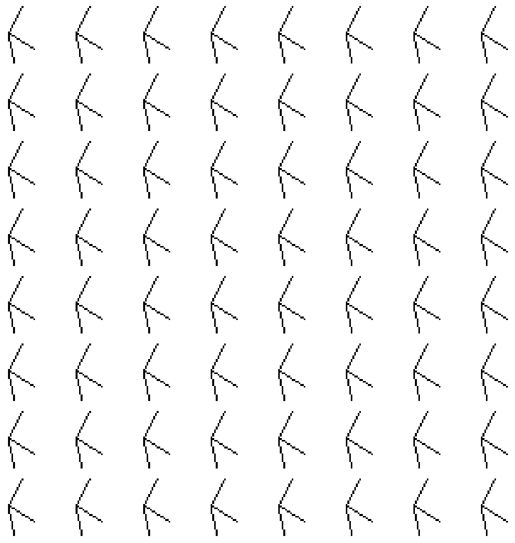


Fig. 12 Test number 7

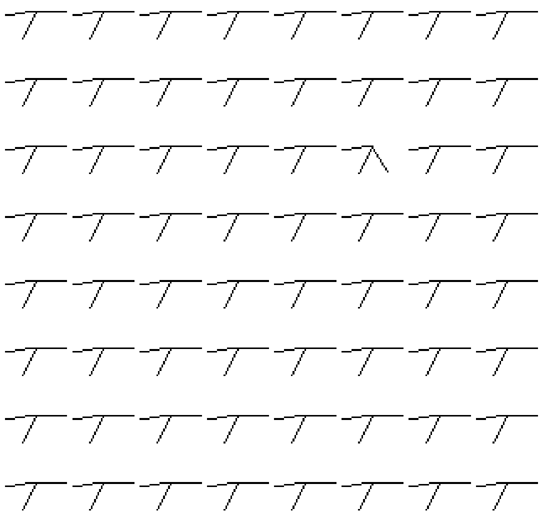


Fig. 13 Test number 14

6 Evaluating human sensibility to glyph

Visualizing information through glyph help users to detect anomalies, but, how users interact with the information? which type of variations are more sensible from user point of view?

To explore this, we test a set of snapshot of sixteen glyph panels with ten users to check the "sensibility" of users to specific glyph forms and changes. Each panel has 64 glyph of three lobs each one of them. An example of one of these panels is shown in Figure 13.

The sixteen glyph panels are different, in some of them all glyph are identical meanwhile other panels have one glyph different. In table 3 we can see which

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	**192**	197	0	0	0	0	0	0	0	69	0	0	0	0
2	59	**5825**	10	0	0	20	0	0	0	67	0	0	0	0
3	2	280	**183**	0	0	1	0	0	0	3	0	0	0	0
4	3	295	1	**0**	0	12	0	0	0	10	0	6	0	0
5	2	173	1	0	**0**	3	0	0	0	5	0	0	0	0
6	8	212	3	0	0	**34**	0	0	1	31	1	5	0	0
7	2	121	4	0	0	6	**0**	0	0	9	1	0	1	0
8	5	370	6	0	0	1	0	**0**	0	7	1	1	0	0
9	5	350	16	0	0	4	0	0	**0**	2	0	0	0	0
10	88	266	1	0	0	4	0	0	0	**425**	1	0	0	0
11	97	149	0	1	0	2	0	0	0	94	**1**	0	0	0
12	24	218	4	0	0	2	0	0	0	9	0	**0**	0	0
13	12	192	18	0	0	1	0	0	0	5	0	0	**1**	0
14	3	123	0	0	0	1	0	0	0	16	0	0	0	**0**

Table 2 Confusion matrix for accelerometer sensor

Table 3 Number of test and variability of the glyph in each test

Nº Test	1	2	3	4	5	6	7	8
Variability	0	68°	0	5.7°	0	11.4°	0	23°
Nº Test	9	10	11	12	13	14	15	16
Variability	0	34, 4°	46.8°	0	0	57.2°	0	0

test shows no variability (e.g In Figure 12 we can see the test number 7 with value 0) and the degree of variability in degrees of the glyph when there is one different (e.g Figure 13).

The setup of our test was an application where the glyph panels was shown to the user sequentially from test number one to test number two, asking them, all glyph of of this panel are identical? with two buttons one for identical another for different. The application took the time used by the user in answer since each panel is shown. Ten users did the experiment.

The results are shown in form of boxlots. For each data set, a yellow box is drawn from the first quartile to the third quartile, and the median is marked with a thick line. Additional whiskers extend from the edges of the box towards the minimum and maximum of the data set, but no further than 1.5 times the interquartile range. Data points outside the range of box and whiskers are considered outliers and drawn separately, as small circles.

As we can see in figure 15, there is a variety of users from the time they take in choose the answer to each panel (e.g if all the glyph are equal or not) so we have a set of users faster than others in the answer.

In each panel, the users spend time looking if all glyph are equal or not, as it is shown in figure 14, the average time in answer is lower when the difference is obvious. For example, in figure 13 the panel fourteen is shown, the different glyph is quickly detected by the user and answer. However, when there is a panel with

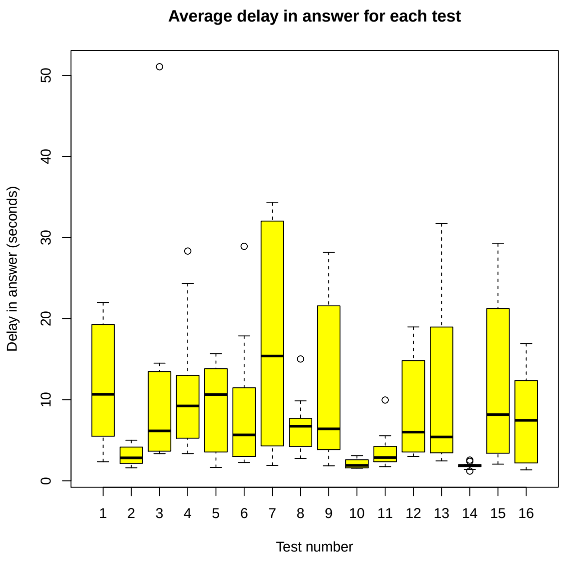


Fig. 14 Delay in answer per test

all glyph equals, users tend to look up for the difference and delay the answer, this is the case of test seven, which has the longest time to answer in average, that we can see in figure 12. As we expected, panels with variation in glyph formed by right angles are more easy to detect, for example, the test number ten has one of the lowest delay in answer for all users (Figure 16).

In spite all panels show the same type of glyph (with 3 lobs), of course, the human sensibility to the change in one of the glyph is detected for values around to 30 degrees. As we can see in figure 17, all users fail in detect the variation in the test number 4. In this test, a glyph is different but only with a variation of 5.7 degrees in one of its lobs. Six users also fail the test number six and four the test number eight with a variation of 11 degrees and 23 degrees respectively. Finally, the rest of the test have one or none failure in the human perception.

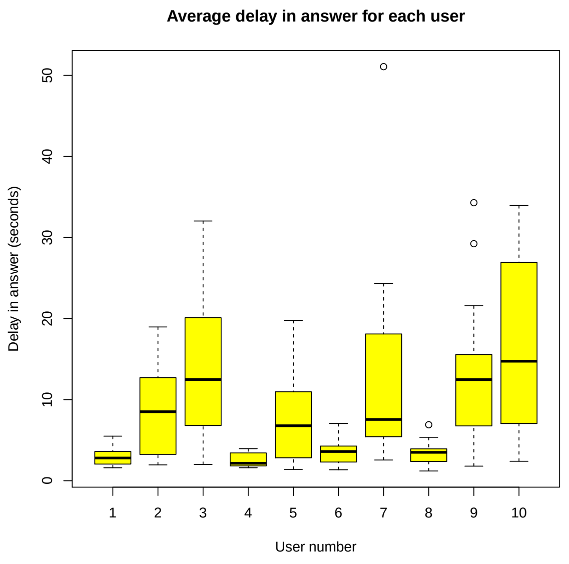


Fig. 15 Delay in answer per user

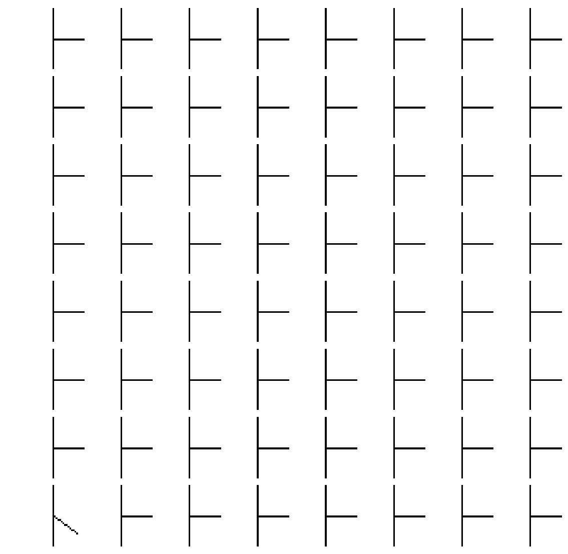


Fig. 16 Test number 10

Again It is necessary to point out that by passing this experiment, users are emulating to monitor 64 variable streams with 3 variables in each stream detecting anomalies in few seconds (Figure 15)

7 Conclusions

One of the key problems in smart cities is analyzing and understanding the great amount of data genereated by different processes in the city (traffic, pollution monitoring, human dynamics, etc.). In this work we analyze glyphs like tool for data stream visualization

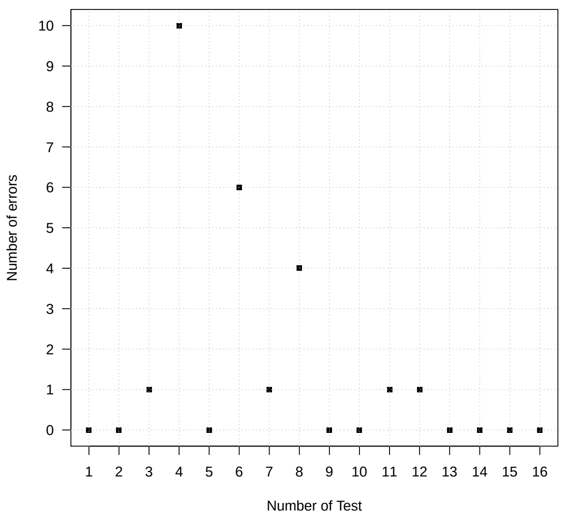


Fig. 17 Errors in human perception of glyph

First we describe how we can use the glyphs for visualization of multi-variable streams, then we build a flexible and scalable distributed framework for collect, fusion, filter and visualize the information. Unfortunately, not all data stream can be presented raw to users, for example accelerometers from mobile phones do not show valid information withouth a previous understanding process. For this reason, we build a SVM approach to human action recognitions for visualization of human dynamics process inside smart cities.

Finally, we have analyzed with users their responsiveness and sensibility to glyph changes by exposing them to a testbed of sixteen panels. This study will help us to design more appropriate glyph for data stream visualization.

As main conclusion, we strongly believe that a glyph based tool can improve the monitoring task and reduce time reaction to abnormal situations in future smart cities.

As future work, we are involved now in several research lines extending this work. First we are planning to extend the study with users perception to different type of glyph forms in order to improve our glyph design skills. Also simulation of a smart city generating data streams is necessary to test the scalability of the framework and also see how different situations are translated to glyph visualization. Protest, panic situations, traffic accident, are examples of events which we are interested in. Only simulation can provide us with enough datastream flows to test this type of situations at affordable cost.

Finally, in spite our tool is devoted to smart city monitoring, we are also interested in automatic smart city understanding.

Acknowledgements This research was supported by the Spanish Ministry of Economy and Competitiveness under the project DREAMS (TEC2011-28666-C04-03).

References

1. Thingspeak. https://thingspeak.com/
2. Blog, S.C.: Video 'must haves' for active surveillance. Http://security.americandynamics.net/
3. Cai, Y.: Instinctive computing. In: Artifical Intelligence for Human Computing, *Lecture Notes in Computer Science*, vol. 4451, pp. 17–46. Springer Berlin Heidelberg (2007)
4. Cavallari, R., Martelli, F., Rosini, R., Buratti, C., Verdone, R.: A survey on wireless body area networks: Technologies and design challenges. Communications Surveys Tutorials, IEEE **16**(3), 1635–1657 (2014)
5. Chih-Wei Hsu, C.C.C., Lin, C.J.: A practical guide to support vector classification. National Taiwan University (2010)
6. Childs, H., Geveci, B., Schroeder, W., Meredith, J., Moreland, K., Sewell, C., Kuhlen, T., Bethel, E.: Research challenges for visualization software. Computer **46**(5), 34–42 (2013)
7. Colitti, W., Steenhaut, K., De Caro, N., Buta, B., Dobrota, V.: Rest enabled wireless sensor networks for seamless integration with web applications. In: Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on, pp. 867–872 (2011). DOI 10.1109/MASS.2011.102
8. Haklay, M.M., Weber, P.: Openstreetmap: User-generated street maps. IEEE Pervasive Computing **7**(4), 12–18 (2008)
9. Hao, M., Marwah, M., Mittelstadt, S., Janetzko, H., Keim, D., Dayal, U., Bash, C., Felix, C., Patel, C., Hsu, M., Chen, Y.: Exploring cyber physical data streams using radial pixel visualizations. In: Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on, pp. 225–226 (2012). DOI 10.1109/VAST.2012.6400541
10. Henning, M.: A new approach to object-oriented middleware. IEEE Internet Computing **8**(1), 66–75 (2004)
11. Kim, J., Lee, J.W.: Openiot: An open service framework for the internet of things. In: Internet of Things (WF-IoT), 2014 IEEE World Forum on, pp. 89–93 (2014). DOI 10.1109/WF-IoT.2014.6803126
12. Lerche, C., Laum, N., Golatowski, F., Timmermann, D., Niedermeier, C.: Connecting the web with the web of things: lessons learned from implementing a coap-http proxy. In: Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on, vol. Supplement, pp. 1–8 (2012). DOI 10.1109/MASS.2012.6708525
13. Moya, F., Villa, D., Villanueva, F.J., Barba, J., Rincn, F., Lpez, J.C.: Embedding standard distributed object-oriented middlewares in wireless sensor networks. Wireless Communications and Mobile Computing **9**(3), 335–345 (2009). DOI 10.1002/wcm.545
14. Park, H.K., Lee, Y.W., Jang, S.I., Lee, I.P.: Online visualization of urban noise in ubiquitous-city middleware. In: Advanced Communication Technology (ICACT), 2010 The 12th International Conference on, vol. 1, pp. 268–271 (2010)
15. Park, J.W., Yun, C.H., Jung, H.S., Lee, Y.W.: Visualization of urban air pollution with cloud computing. In: Services (SERVICES), 2011 IEEE World Congress on, pp. 578–583 (2011)
16. Perera, C., Liu, C., Jayawardena, S., Chen, M.: A survey on internet of things from industrial market perspective. Access, IEEE **2**, 1660–1679 (2014). DOI 10.1109/ACCESS.2015.2389854
17. Rhyne, T., Chen, M.: Cutting-edge research in visualization. Computer **46**(5), 22–24 (2013)
18. Sapankevych, N., Sankar, R.: Time series prediction using support vector machines: A survey. Computational Intelligence Magazine, IEEE **4**(2), 24–38 (2009)
19. Tory, M., Mller, T.: Human factors in visualization research. IEEE Transactions on Visualization and Computer Graphics **10**, 72–84 (2004)
20. Villanueva, F., Santofimia, M., Villa, D., Barba, J., Lopez, J.: Civitas: The smart city middleware, from sensors to big data. In: Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on, pp. 445–450 (2013)
21. Wahle, S., Magedanz, T., Schulze, F.: The openmtc framework: M2m solutions for smart cities and the internet of things. In: World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a, pp. 1–3 (2012). DOI 10.1109/WoWMoM.2012.6263737