Open Data Project: Traffic Violations

Douglas Garcia Torres (1036847)
M. Bilal Zahid (1035291)
Pradyumna Majumder (1036151)

06-04-2017

Contents

1 Introduction

This open data project is focused on creating a tools to analyze and get insights from traffic violation information. The result of this project could be a useful visualization application for a smart city solution that aims to prevent traffic problems, improve road security and optimize the urban flow of people. The main goal of this project is to incorporate the traffic data into a space-time effective user friendly visualization.

In order to develop the project, we selected the Traffic Violations dataset of the Montgomery County of Maryland, USA, to use it as a model for the structure of the data. This dataset can be obtained in the U.S. Government's Open Data Portal at catalog.data.gov. It contains traffic violation information from all electronic traffic violations issued in the specified County. The information in this dataset includes:

- Date and time of stop
- Location information (state, district, latitude, longitude)
- Description of the violation
- Arrest type
- Vehicle information (type, make, model, color)
- Driver information (race, gender, city and state of origin)
- Other boolean fields (accident, belt use, injuries involved, alcohol consumption, fatalities involved, etc.)

The resulting software helps to analyze geographical and time dimensional data through easy-to read charts and graphs. To develop this project, JavaScript was the main programming language used with the popular D3.js libraries, which allow us to bind the data to a Document Object Model (DOM), and then apply data-driven transformations to this document. In addition, to include appealing geographical visualizations, a very powerful JavaScript based library was used: CESIUM, an open source code for world class 3D globes and maps.

The code of this product is published in a GitHub repository. Additionally, a working version of the visualization is available online in through the GitHub Pages web hosting services pages.github.com (a free HTML/JavaScript website hosting).

2 Requirements

The main requirement for this project is to pick a non-trivial data set and create a tool for visually browsing that dataset for interesting information. It is also important (although not mandatory) that the problem involves a dataset with spatial-temporal data.

In addition, after defining the problem and selecting the working data set, another important requirement arose: because one is trying to build a data visualization tool for traffic information, it is natural to expect an integration of this tool with technologies like OpenStreetMap (OSM) or Google Street View with the intention to show the user real images of the geolocation data.

Another natural requirement for a visualization project like this, is that the tool should be highly interactive with the user. From there arises the consideration of the following features as requirements for the minimum viable product:

Visualization selection: the user can customize the type of visualization. Thanks to the CESIUM libraries, the user is table to select the layer of the map between multiple options like a satellite view, streets classic view, OpenStreetMap, National Geographic view, natural earth or black marble view (night view).

Hovering: the user can pop-up brief tooltip information about any part of the visualization by hovering over with the mouse. In addition, hovering a part of a specific visualization panel will **highlight** the related information in the main panel.

Drill-down: into detailed information by clicking on any part of the visual components with additional details available. After a left mouse click, a new panel will populate with the details of the traffic violations.

Filtering: providing the visualization tool with a panel with filtering controls by any of the available attributes for allowing the user to perform data analysis at any possible level. The filtering can be done by the following criteria:

- Time of stop (clustered by hours)
- Accidents involved
- Belt (use of)
- Personal injuries involved
- Property damage
- Fatalities involved
- Hazardous materials involved
- Alcohol consumption involved
- Vehicle type, make, color of vehicle, year
- Driver gender and race
- Charge and arrest type

Zoom in/out: the user is able to zoom in and out of the location of a specific incident or even on any random point on the main map graph.

Street View: the user is able to select the OpenStreetView as the visualization layer in the main graph with the possibility to zoom in and see nitid images of the locations of the incidents. In addition, if the time allows, it is possible to include an integration with the Google StreetView services to provide a closer look.

Temporal analysis: because the problem and the dataset used includes time dependent data, the tool should provide the means to analyze the evolution of data through the temporal dimension.

3 Analysis and Solution

3.1 Analysis

Considering the requirements stated in the previous section, the problem of analyzing traffic violation data requires spatial and temporal visualization tools. In addition, for any data analysis task it is mandatory to count with filtering and drill down capabilities without forgetting to provide ease of use to any kind of user.

The person who is performing a traffic violation analysis would be willing to visualize the evolution of the incidents over time, preferably mapped into a geographical layout view. In addition, this person would need to see the aggregated data of traffic violation records by any possible criteria. For example, analyze the evolution of the number of traffic violations of male drivers where alcohol consumption is involved.

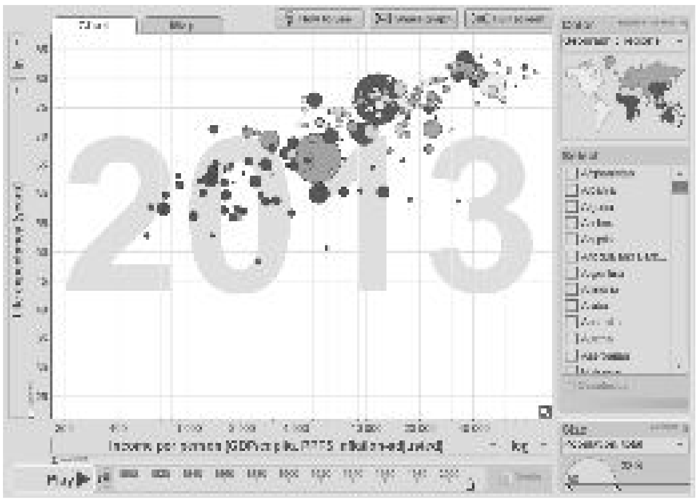
For this project, the idea is to cover all the stated requirements to provide the means for performing such kind of analysis by developing a straightforward visualization tool with the technologies that are more appealing to us as Data Scientist.

3.2 Solution

The solution proposed is a visual web application developed using JavaScript language, the D3.js library and the CESIUM library to map the data into a spatial dimension. The first version of this application aims to be connected to a local data source in JSON format.

The web application provides a straightforward single perspective visualization with a main geographical view along with three additional panels: one panel to visualize the evolution of data over time, another panel to view histogram-like aggregated information, and a third panel to verify the detailed information of the traffic violations. This layout can be reviewed in more detail on the next section.

The single perspective is complemented with 2 widgets: a filtering widget to allow the user to analyze the data by any possible specific criteria, and a play-time widget to provide an effective solution to visualize time-dependent data. This play-time widget is inspired on the famous Gapminder charts which makes possible to visualize the behaviour of 4 different dimensions/measures over the dimension of time (see figure below).

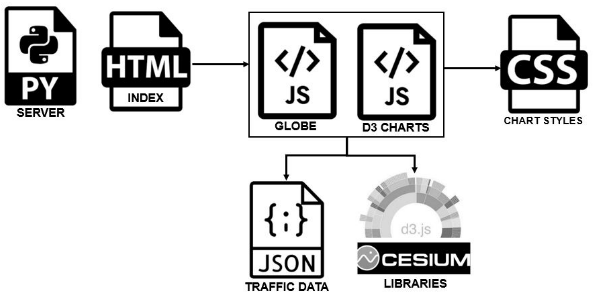


The Gapminder chart shows 5 dimensions of data. The life expectancy in years (Y axis), the income per person as GDP per capita (X axis), the countries (each country is a bubble), the population of a country (bubble size), the region of the country (bubble colour) and the time in years changing when the play-time widget is triggered.

With this tool any user could easily analyze the behaviour of traffic violations over space and time, while selecting specific criteria of available attributes to filter data. This tool is now available online through the GitHub Pages website hosting services pages.github.com.

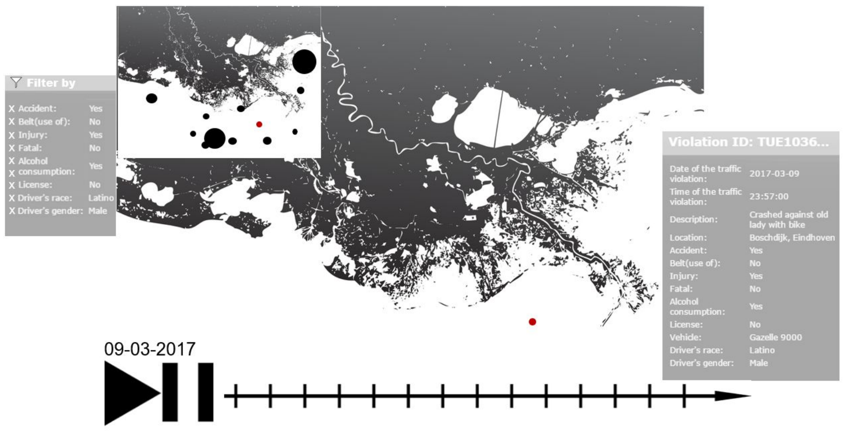
3.3 System Design

The solution built is mainly composed by seven components between HTML, CSS, Javascript, Python and JSON files along with the already mentioned JavaScript third party libraries. The scheme of the backend of the application is illustrated below:



- The main file is Index.html, and makes reference to the two main JavaScript files: Globe.js and the D3 chart file.
- The Globe.js file loads and maps the traffic data (from a JSON file) into the main visualization provided by CESIUM libraries. In addition, defines the functions for the user interaction with the main graph.
- The D3 Chart file loads the data into the D3 developed visualizations, also defining the interactive functions and the interoperability with the main visualization. It makes use of the CSS styles file for the graphs.
- The CSS styles file provides the D3 visualization with proper style formatting.
- All the traffic data is included into one only JSON file.
- The Server.py file is used to generate local web services in order to run the application locally.
- The rest of the files are included in two folders corresponding to the third party libraries used: D3.js and CESIUM.

On the other hand, the visualization is basically composed by five components: a main geographical visualization, the D3 developed visualizations with aggregated information, a detailed information panel, the filtering panel and the play-time widget. The design of this layout is shown below:



Filter by

X Accident:	Yes
X Belt(use of):	No
X Injury:	Yes
X Fatal:	No
X Alcohol X consumption:	Yes
X License:	No
X Driver's race:	Latino
X Driver's gender:	Male

Violation ID: TUE1036...

Date of the traffic violation:	2017-03-09
Time of the traffic violation:	23:57:00
Description:	Crashed against old lady with bike
Location:	Boschdijk, Eindhoven
Accident:	Yes
Belt(use of):	No
Injury:	Yes
Fatal:	No
Alcohol consumption:	Yes
License:	No
Vehicle:	Gazelle 9000
Driver's race:	Latino
Driver's gender:	Male

- The main geographical visualization show the location points of the traffic violations. It is synchronized with the other four elements:
  - If the user interacts (clicks on a specific part) with the D3 visualization of the top left side of the screen, the related traffic violations will be highlighted.
  - When the user executes a filtering function, just the traffic violations that meet the filtering criteria will appear.
  - When the user clicks on a specific point (traffic violation), the detailed information panel will pop-up in the right side of the screen.
  - The points shown will correspond only for the traffic violations occurring in the specific period of time (day) that is running on the play-time widget.

- The D3 developed visualization on the left side of the screen shows 3 dimensions of data: latitude (Y axis), longitude (Y axis) and a custom selected dimension (color). As the time passes (by the play-time widget), the bubbles will accumulate as the number of incidents increases in a specific sector. This way, the user can detect the hot points where most of the traffic violations occur.

- The detailed information panel on the bottom right side of the screen show all the available information of the traffic violation selected on the main visualization.

- With the filtering panel, the user will be able to decide what data to visualize according to the constraints selected. The filtering options will be populated dynamically according to the loaded data values.

- The play-time widget is provided by the CESIUM library and it features forward and backwards linear execution of time. The speed can be tuned in the application code.

3.4 Functioning of the System

This section is aimed to explain the functioning of the system and illustrate our contribution beyond the use and integration of CESIUM libraries with the Traffic Violation dataset. All the visualization and interaction functions (that are not part of the provided ready-to-use CESIUM functionalities) are explained from a technical point of view with pseudo-code snippets of the implemented object oriented algorithms. All the implemented code is available in the following GitHub repository: https://github.com/garciatorres/D3Cesium.

To begin with, the main file **index.html** invokes the scripts contained in the Globe.js and D3 charts JavaScript files. It is relevant to point out that it makes use of a shared object that contains all the needed global variables to synchronize these two files.

```
var SharedObject = {
    Date Date,
    Array Data,
    Array Filter,
    String GroupBy,
    Function Dispatch,
    Function DispatchFilter,
}
```

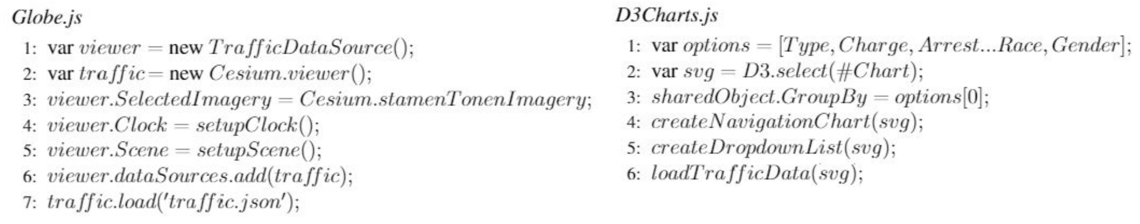
The *Date* variable contains the current date processed by the visualization at any given moment. The *Data* and *Filter* variables are two arrays containing the data records of traffic violations and the filters applied by the user respectively. The String variable *GroupBy* contains the dimension selected by the user as the criteria to group the data (the default value is the Traffic Violation type).

Additionally, the functions *Dispatch* and *DispatchFilter* are D3 mechanisms to trigger and handle the events related to the pop up of additional panels (the filtering and the detailed information panels).

3.4.1 Visualization

There are two main scripts that builds the visualizations of this Javascript application: the Globe script and the D3 Chart script. The image below shows an overview of the pseudo-code for each one of these main scripts.

NOTE: to facilitate the understanding of the pseudo code, most of the functions are simplified. Moreover, the order of execution for the auxiliary functions may not be the same order in which the real script code is written. The functions that does not start with the Cesium or D3 prefixes, are not part of the pre-built Cesium or D3.js libraries, this means, they were developed for this project or copied from another code snippet available on the web.



Globe.js

```
var viewer = new TrafficDataSource();
var traffic= new Cesium.viewer();
viewer.SelectedImagery = Cesium.stamenTonenImagery;
viewer.Clock = setupClock();
viewer.Scene = setupScene();
viewer.dataSources.add(traffic);
traffic.load('traffic.json');
```

D3Charts.js

```
var options = [Type, Charge, Arrest...Race, Gender];
var svg = D3.select(#Chart);
sharedObject.GroupBy = options[0];
createNavigationChart(svg);
createDropdownList(svg);
loadTrafficData(svg);
```

From the *Globe.js* script, the functions *setupClock()* and *SetupScene()* are simplifications of several lines of code to set up parameters of the Cesium features like: clock start date, clock end date, clock pace, show the sun in the scene, show the moon, initial position of the camera, etc. The *TrafficDataSource* object is a defined class that contains a pointer to the selected traffic violation, the start date of the traffic violations, the end date, among other attributes.

From the D3Charts script, the functions *createNavigation()* Chart and *createDropDownList()*, are in the same way, simplifications of several lines of code for creating and appending the corresponding D3 elements to the *svg* variable for the drop down UI element and the navigation panel chart.

As the programming language is not only object-oriented, but function-oriented, most of the developed code is declared as functions and assigned to the defined visualization objects (i.e. drop-down menus, chart points, legend elements, etc.). The rest of this section lists all the visualization functions (related to visualization of elements, not user interaction) with their associated technical name.

Load Data: most of the implemented code relies within the load data functions of each one of the scripts. It is important to notice that both scripts open and read the *json* datafile. This code can be improved and this can be a aspect to optimize.

In the Globe script, the data load function is within the *TrafficDataSource* object, the technical function definition is: *TrafficDataSource.prototype.load*.
In the D3Charts script, the data load function definition is within the statement:
d3.json("traffic.json", function(violations)...

Display Year: this function relies on the D3Charts script. In addition to update the current year in execution by the play-time widget, it invokes the required functions to update all the shown data points of the navigation panel. Its is defined as the function *displayYear(year).*

Build Navigation Panel: in the D3Charts script, the code for the construction of the navigation panel is defined before the data load over the *svg* variable. Moreover, within the data load function the navigation panel chart gets populated after defining the *dot* object variable with the *position()*, *order()* and *interpolatedData(year)* functions.

Build Histogram: the extension of the navigation panel to build and show the Histogram for the aggregated data, is performed by the functions *build_histogram()* and *show_histogram()* within the D3Charts script.

3.4.2 Interaction

The functions described below are the ones in charge of the main interactive features of this application. They are all part of this project development and can be found on either the Globe script or the D3Charts script.

Mouse Over (Hover): the mouse over handler function that displays the information panel data is implemented in the Globe script after being defined within the global object *SharedObject.*
sharedObject.dispatch.on("nationMouseover.cesium", function(violationObject)...

Group By: the function that handles the trigger for changing the grouping dimension after the a user selection on the corresponding drop down list, is defined in the D3Charts script with the statement: *dropDown.on("change", function()...*

Filter Data: the code implementations supporting the filtering features are all included in the D3Charts script. They are the following functions: *filter_function()*, *filter_dots()* and *filter_date()*.

Clear Filter: this function is performed in the D3Charts script by the *clear_filter()* function.

Show Histogram: the functions *build_histogram()* and *show_histogram()* defined in the D3Charts script act as the handlers of this feature.

Fly to Location: this function is implemented in the Globe script as part of the global object SharedObject. The technical name is *sharedObject.flyTo().*

4 Motivation of Choices

4.1 Data: Traffic Violations

To select this traffic violation data, our motivation was to find a large and complex enough dataset which can fulfill all the requirements for this project. After reviewing different options, we selected this traffic violations dataset because it contains a large number of interesting visualization aspects, it is complex enough and also gives us various ideas to develop. Moreover, it also contains space-temporal information of a particular incident, which is required for this open data project.

4.2 Technologies: JavaScript, D3 and Cesium

The main reason to choose JavaScript as the programming language for developing this application was that it is the base language of the famous D3.js libraries. The use of this technologies was recommended to us by various colleagues, the professor of this course and the professor of the Visual Computing course, Michel Westenberg.

Further, the D3.js and Cesium libraries together, helps us to easily generate contents while providing a variety of appealing visualization aspects. A considerable number of the search results for visualization tool options, recommended to use these libraries. In particular, with the CESIUM libraries one can fulfill the OpenStreetMaps integration requirement while providing multiple additional features:

- Visualize high-resolution global terrain.
- A 3D globe, 2D map view.
- Layer imagery from multiple sources.
- Timeline and animation widgets for controlling simulation time.
- Base layer picker widget for selecting imagery and terrain.
- Selection and info box widgets for highlighting objects and displaying information.
- Geocoder widget for flying to addresses and landmarks.
- Home view widget to fly to the default camera view.
- Scene mode picker widget to morph between 3D, 2D, and Columbus view.
- Full-screen widget for toggling fullscreen mode.

4.3 Functionality and Interactive features

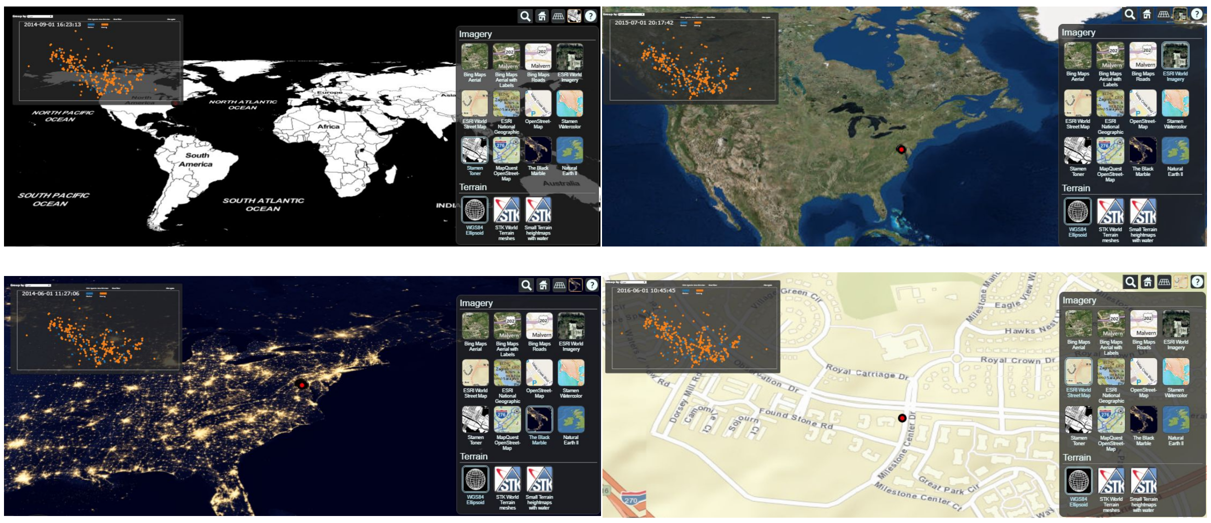
The choice to show specific visualization aspects are based on past experiences (Data Visualization courses) and interests. Moreover, the resulting visualization and functionalities implemented are influenced by the flexibility and appeal of popular visualizations like Hans Rosling's Gapminder, or the Radio Garden open data project (developed with CESIUM libraries), among others examples from the D3 homepage.

5 Results and Limitations

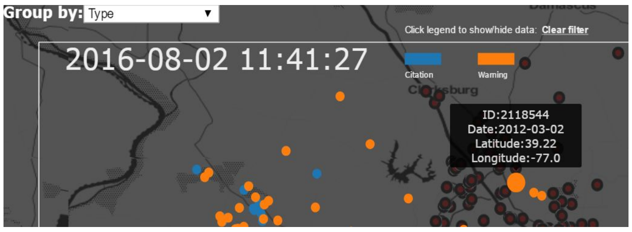
5.1 Results

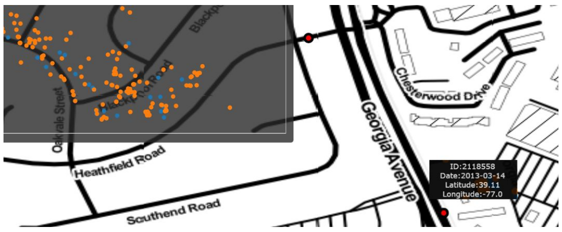
At the end of this project, it is possible to claim that all the features planned in the concept report were implemented and the initial requirements were fulfilled. Between the features planned before the beginning of the implementation phase, there were visualization selection possibilities, hovering effects, drill-down and filtering for information insights, zoom-in/out from a geographical location, integration with street views and tools to allow temporal analysis. This section shows how the solutions implemented satisfies each one of these requirements.

Visualization selection: the Imagery widget provided by CESIUM and integrated into this application, allows the user to select from a variety of layers and apply them to the geographical visualization. The images below show various of the possible layers to select (from left to right and top to bottom: Stamen Toner, ESRI World Imagery, The Black Marble and ESRI World Street Map). The Imagery widget is shown extended on the right side of the screen.

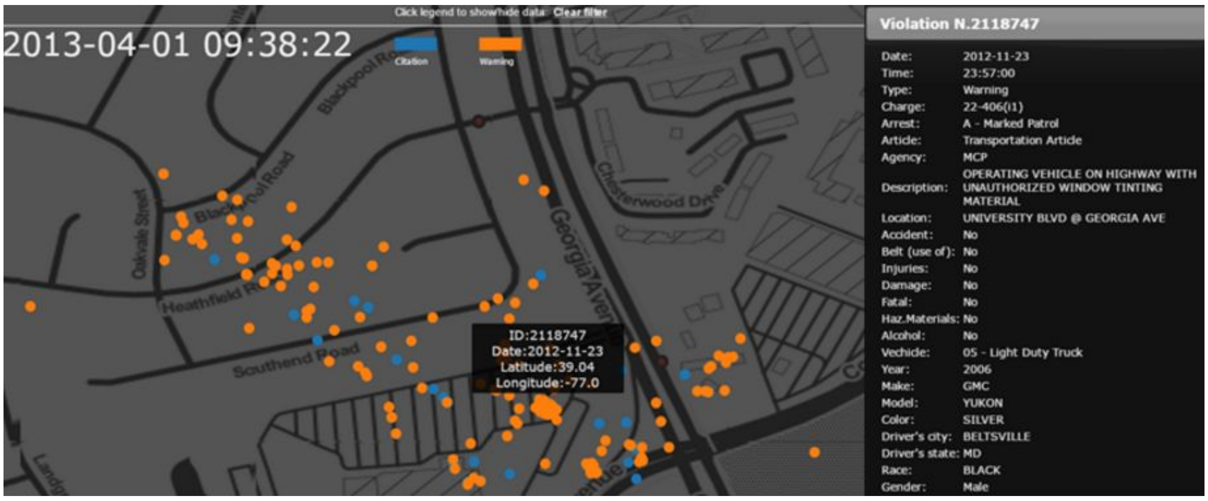


Hovering: this feature was implemented using Tipsy, a JQuery plugin that provides a very efficient and easy to implement tooltip functionality. Whenever the user hovers on a specific data point, a tooltip will pop-up showing brief information of the associated Traffic Violation. In the image shown below it is possible to see brief information (ID, Date and geo-location data) of the specific point hovered on the navigation panel.





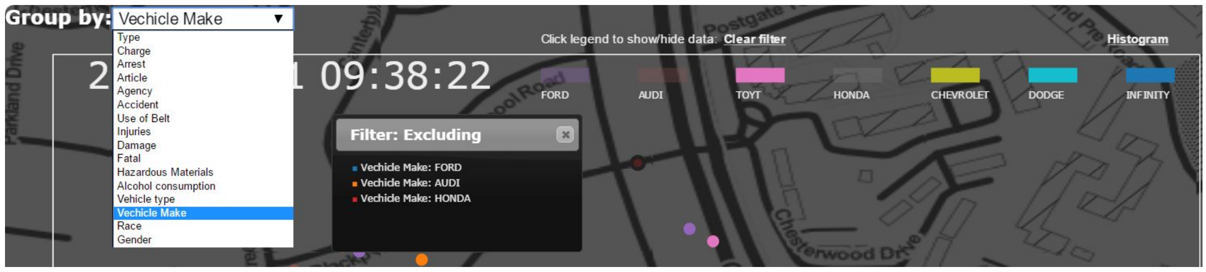
Drill-down: in addition to the hovering features that shows brief information of a specific data point, when the user triggers a left mouse click, an additional panel shows up (on the right side of the screen) with all the available information of the Traffic Violation.



Violation N.2118747

Date:	2012-11-23
Time:	23:57:00
Type:	Warning
Charge:	22-406(i1)
Arrest:	A - Marked Patrol
Article:	Transportation Article
Agency:	MCP
Description:	OPERATING VEHICLE ON HIGHWAY WITH UNAUTHORIZED WINDOW TINTING MATERIAL
Location:	UNIVERSITY BLVD @ GEORGIA AVE
Accident:	No
Belt (use of):	No
Injuries:	No
Damage:	No
Fatal:	No
Haz.Materials:	No
Alcohol:	No
Vechicle:	05 - Light Duty Truck
Year:	2006
Make:	GMC
Model:	YUKON
Color:	SILVER
Driver's city:	BELTSVILLE
Driver's state:	MD
Race:	BLACK
Gender:	Male

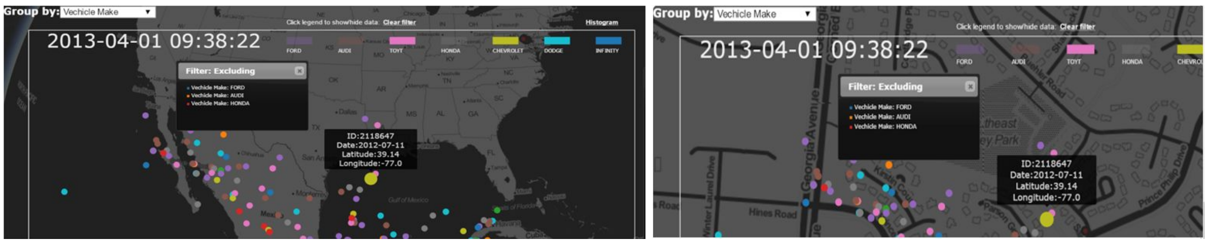
Filtering: this functionality was implemented within the navigation panel. On the image below one can see all the elements of the filtering feature: a drop-down list to select the grouping attribute, an interactive legend to show/hide data based on the attribute values, a pop-up panel showing the selected filters and a "Clear filter" option that can be used to reset the filtering function.



Filter: Excluding

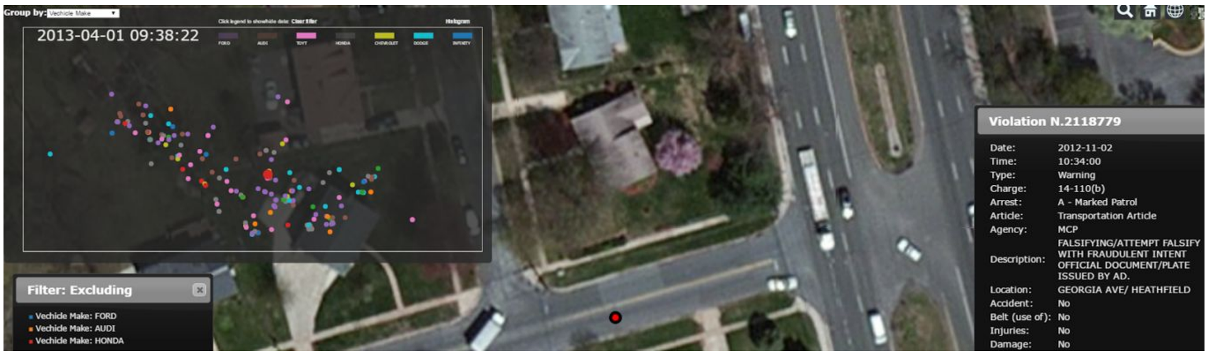
- Vechicle Make: FORD
- Vechicle Make: AUDI
- Vechicle Make: HONDA

Zoom in/out: the CESIUM library provides a built in option to zoom in/out from any location of the CESIUM provided globe. In addition, our application includes a developed function that zooms in to the specific location of any data point in the navigation panel. That function is triggered when the user executes a left click with the mouse. The image below shows the visualization panel status before (left) and after (right) the user triggers a left-mouse click on a specific data point.



ID:2118647
Date:2012-07-11
Latitude:39.14
Longitude:-77.0

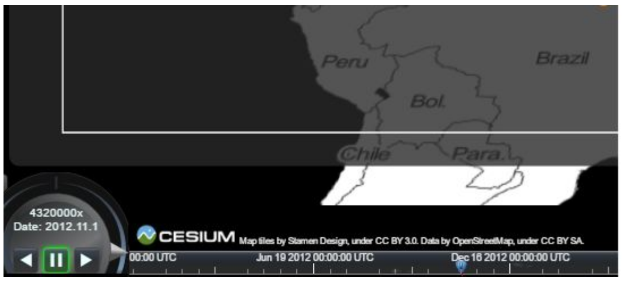
Street View: the image below shows a closer real imagery street view available when the user selects the proper layer and clicks on a specific data point. The red point shows where the traffic violation took place while the information panel shows all the details of the incident.

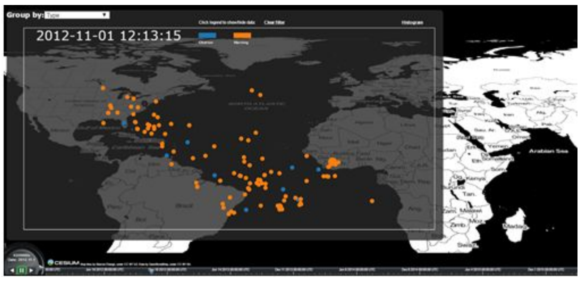


Violation N.2118779

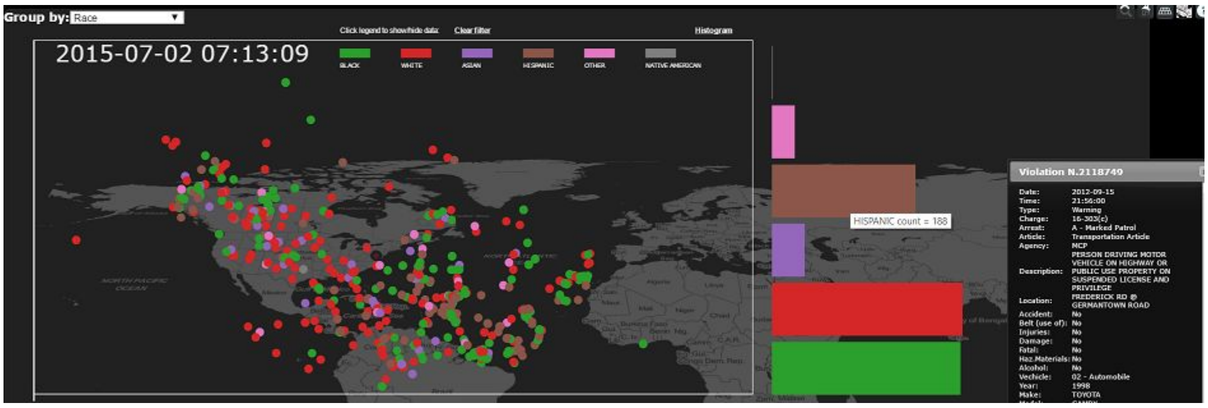
Date:	2012-11-02
Time:	10:34:00
Type:	Warning
Charge:	14-110(b)
Arrest:	A - Marked Patrol
Article:	Transportation Article
Agency:	MCP
Description:	FALSIFYING/ATTEMPT FALSIFY WITH FRAUDULENT INTENT OFFICIAL DOCUMENT/PLATE ISSUED BY AD.
Location:	GEORGIA AVE/ HEATHFIELD
Accident:	No
Belt (use of):	No
Injuries:	No
Damage:	No

Temporal analysis: the CESIUM libraries provide a play-time widget where it is possible to see the evolution of the data on a temporal dimension. In order to integrate that widget with our dataset, it was necessary to develop additional functions to show/hide the data point appropriately according to the current date of the widget. The data points shown are the ones with a date minor or equal to the date that appears on the widget (and on the panel). Additionally it is also possible to stop the time, play it slower, faster or play it in reverse. On the images below, one can see the components of this widget (left), these are; the buttons for playing the time onwards and backwards, the pause button and a handle to set the time speed. The image on the right shows the synchronization of time and data on the other panels developed from scratch.





Other features: in addition to the features included in the concept report, an histogram was developed to be embedded in the navigation panel. As can be seen below, it shows the number of incidents related to each one of the attribute values of the chosen grouping dimension. In this particular example, the user has selected the attribute Race as the grouping attribute, and after clicking the Histogram option in the navigation panel, a bar chart shows the number of incident for each one of the Races (Black, White, Asian, Hispanic, etc.). This bar chart is synchronized with the play-time widget and it gets updated as the time runs.



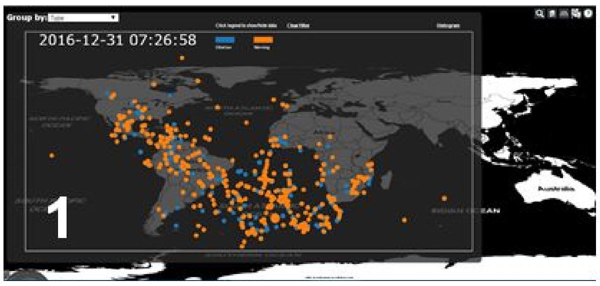
Violation N.2118749	
Date:	2012-09-15
Time:	21:56:00
Type:	Warning
Charge:	16-303(c)
Arrest:	A - Marked Patrol
Article:	Transportation Article
Agency:	MCP
Description:	PERSON DRIVING MOTOR VEHICLE ON HIGHWAY OR PUBLIC USE PROPERTY ON SUSPENDED LICENSE AND PRIVILEGE
Location:	FREDERICK RD @ GERMANTOWN ROAD
Accident:	No
Belt (use of):	No
Injuries:	No
Damage:	No
Fatal:	No
Haz.Materials:	No
Alcohol:	No
Vechicle:	02 - Automobile
Year:	1998
Make:	TOYOTA

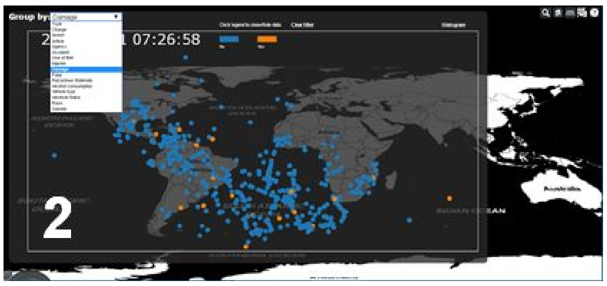
5.2 A Use Case Scenario

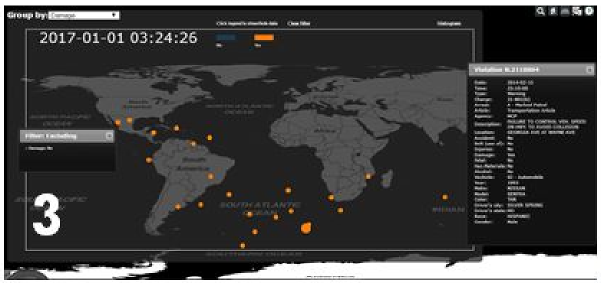
A Traffic Analyst of the Montgomery County of Maryland, USA, requires to determine the spots where most of the Traffic Violations with property damage involved take place. The requirement involves taking pictures of these spots for further analysis. To accomplish the task, the Traffic Analyst could follow these steps:

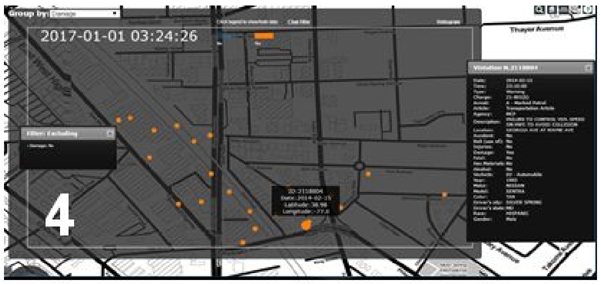
1. Stop the clock at the end of the time line
2. Group by the *Damage* dimension
3. Hide the data points with Damage value equals to *No*
4. Determine the spots with the highest concentration of data points and click on each one of these spots to fly to their specific location
5. Change the layer to the required visualization
6. Take a screenshot

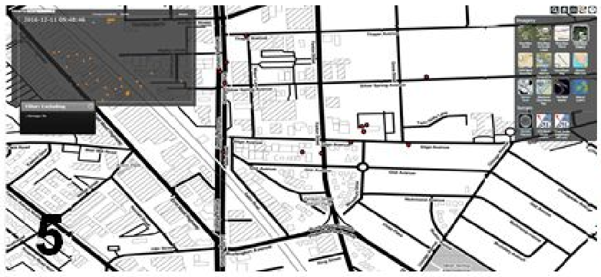
The following images show the sequence of the steps to be performed:

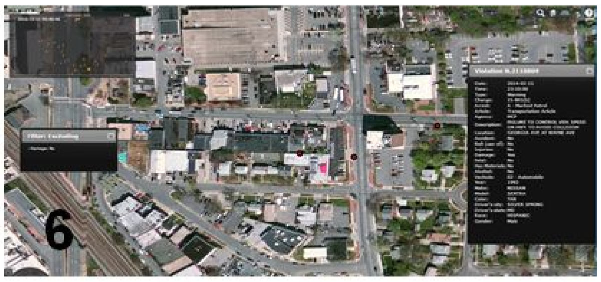












It is important to point out that there is a lot of space for improvement on this application. In the following section most of the detected limitations are described.

5.3 Limitations

There are various limitations on this application as a consequence of is considered out of scope for this project. For the moment, the following can be mentioned:

- The geographical scope of this application lies on the limits of the Montgomery County of Maryland, USA. For the moment we have not found a more complete traffic violations dataset on another location. Therefore, this dataset's structure is taken as a model. To include the data for any other region, it has to be pre-processed to fulfill the structure and format requirements.

- The application runs over a local dataset. In order to perform analysis on updated data, it has to be manually downloaded, cleaned and preprocessed (if necessary). Although the development of a web crawler API would be of great value for this project, it is considered out of the scope.

- The ability to process large datasets ended up being out of scope for this project. At the moment of the delivery, the application has been tested to handle datasets up to 50 MB of JSON data. Unfortunately, bigger datasets are not well handled.

- There are some detected functions that need to be implemented in order to improve the actual features and facilitate the traffic analysis. Some of them are:

  - Data clean option: if the user would like to analyze the data starting from one specific date, could place the timeline pointer at that date, select the data clean option (to clear all the accumulated data points) and press play on the play-time widget.

  - Camera: once the user has placed the visualization focus on a required place, an available camera option could be very useful to automatically place the screenshot of a specific place on the user’s clipboard.

  - Density detection: instead of expecting the user to “subjectively” determine the places with the higher density of incidents by looking at the navigation panel, a function can be implemented to automatically determine and propose these density spots.

- Other features not covered (neither in the design or in the implementation phase) like Traffic Violation forecasting or Traffic Violations Search Engine. In the Future Work section, there is a list of features with which this visualization can be potentially enhanced.

6 Conclusions and Future Work

A conclusion one can mention results from the surprising ease to create highly powerful visualizations with available open source tools like the D3.js and CESIUM libraries. Having discovered that most of the difficulties of creating visualizations come from the data processing tasks.

There is still a lot of space for fixing and improving the features in the actual version of this application. Probably more space of improvement that one could want. However, there is a famous quote from Reid Hoffman, founder of Linkedin, that says: ***"If you are not embarrassed by the first version of your product, you've delivered too late"***.

The future work related to this project is going to be driven by the interests of the members of this team. At this moment one can think of three possibilities: enhancing the features of the traffic violations visualization, combining and complementing traffic violations data with other traffic related dataset (e.g. violations without police intervention), or applying this visualization layout and code to another unrelated topic.

With the first possibility, this visualization could be strongly enhanced with a wide variety of useful features. At the end of this project, specially after receiving the professor's and peer's feedback, the following possibilities were written down:

- Drag and drop capabilities,
- Additional panels showing different graphs and perspectives of traffic violations aggregated information, or by giving the user the possibility to personalize the already developed features (e.g. adding the possibility to change the dimensions involved in the spatial bubble graph of the traffic violation count).
- A web scraper, i.e. a web API to provide pre-processed and cleaned traffic violation data transformed to fit the data format requirements of this application.
- A traffic violations Search Engine, to easily search for a specific record by the date of the incident, the place, vehicle attributes, etc.
- Machine Learning features, to provide an intelligent forecast of traffic violations that can occur in the future based on the already processed factual data.

With the second possibility the current visualization could be enriched with other traffic related data to give more value to this open data project. One possibility could be to find (or define) a unpunished traffic violations dataset and develop a combined visualization of punished against unpunished records. Such visualization could be very useful to traffic authorities.

The third possibility in mind is to adapt the developed visualizations for this project to World Development Indicators data. Having in mind that the popular World Bank dataset, does not count with an appealing geographical data visualization. In addition, such project can include the use of the World Bank data catalog API for a real-time visualization of data and adapt the play-time widget used for this project, to simulate the famous Hans Rosling's Gapminder bubble charts with any of the World Development Indicators.