Investigating Meteorite Landings With Brushing, Filtering & Geovisualization

Erik Sven Vasconcelos Jansson[†]

Abstract—Exploring large datasets (visually), containing both spatial and temporal features, is a problem for which a lot of research has been developed. Here, an application is developed which uses some of these techniques to better visualize such a large dataset.

Index Terms—Information Visualization, Brushing, Filters, Geovisualization, Meteorite Landings

1 Introduction

Advances in hardware have allowed the collection and storage of vast amounts of data (cheaply too). However, exploring such gathered data to learn something from it isn't an easy task, as seen in *Keim et al.* [1], either because data are highly *multi-dimensional* or simply *numerous*.

Visual data exploration attempts to involve the human user in this exploration process; by providing an appropriate *visual representation* and set of *interaction techniques* for developing hypothesis on the data. Nowadays, several of the produced data are both *spatial* and *temporal*, however, attempting to explore these visually poses several interesting problems that require careful thought, as described by *Wood et al.* [6].

This article attempts to outline and thereafter implement techniques which relieve some issues of visual data exploration on large datasets containing both spatial (geographical usually) and temporal variables. As an example, Figure 1 displays the classic symptom of *visual clutter*.

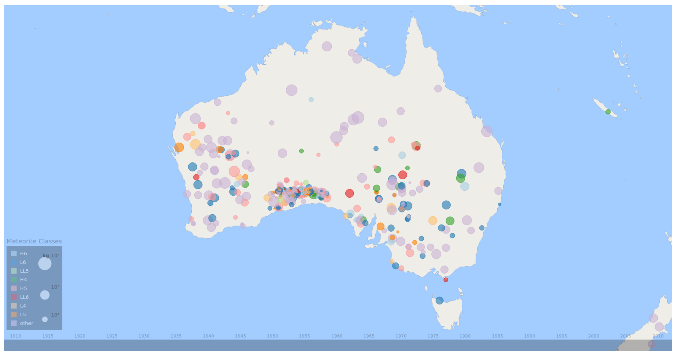


Fig. 1. Visual clutter caused by having large amounts of raw data points

2 Background and Related Work

Since the field being studied is fairly well researched but also broad, the summary within *Keim et al.* [1] provide a base for the techniques used when dealing with large-scale visual data exploration. Especially, their section on interactivity techniques cover the *filtering & zooming* approaches, which are described as essential by *Ben Shneiderman* [3] for fulfilling the so called "visual information seeking mantra", see [3].

Proposed in *Shneiderman's* [3] article are seven task taxonomies for presenting information rapidly while allowing controlled exploration: *Overview*, *Zoom*, *Filter*, *Details-on-Demand*, *Relate*, *History*, *Extract*. *Shneiderman* also describes methods to handle *spatio-temporal* data, however, these seem to be mostly suited for the small/normal datasets.

Discussed in *Wood et al.* [6] are, among others, methods for dealing with overplotting, such as collapsing spatially close symbols together, basically creating a hierarchy by considering similar/close data points.

† *Information Visualization Course at Linköping University, Sweden.*
Student contact email address: `<erija578@student.liu.se>`

3 Data

Searching for a reasonable dataset yielded the *Meteorite Landings* set, which is distributed under the public domain by NASA. It contains around 45 000 entries of all recorded impacts; each entry as in Table 1:

Name	Class	Mass	Seen	Year	Location ([5])
Worden	L5	1551	yes	1997	42.384, -83.611
Yanzhuang	H6	3500	yes	1990	24.566, 114.166
Haven	H6	6100	no	1950	37.964, -97.755
Jesenice	L6	3800	yes	2009	46.421, 14.052

Table 1. Excerpt rows from the NASA "Meteorite Landing" [2] data set

However, even though the data has been cleaned a lot by NASA, a large amount of location data was invalid. Therefore, some additional data cleaning was done, reducing the amount of datum to 26 821 rows.

4 Method

Given the above fairly large dataset, we seek methods to best represent and present the entire data, with all variables, visually to a human user. Instead of starting from scratch, the taxonomies from *Shneiderman* [3] up to the fifth, are used to outline the primary tasks for the application:

- **Topological overview:** since the data is *geographical* in nature, a *world map* is suitable for gaining rough information on where meteorites have fallen and where "clusters" (visually) may exist. Also, by applying *visual variables* to each data point, some of the features can quickly be visualized by the user. Some candidate visuals are *point size* for *mass* and *point color* for *class*, common choices, presented in *Jaques Bertin's Sémiologie Graphique* [4].

- **Translating and scaling:** because the dataset contains locations around the world, being able to *move* & *zoom in* is beneficial for looking around a specific area (reducing a user's search context).

- **Filtering and brushing:** none of the above techniques explicitly relieve visual clutter, but by filtering *mass, classification & time* the user can arbitrarily reduce the scope of the visual exploration. However, these methods need domain knowledge to be effective.

- **Details-on-demand:** since some variables are hard to represent, such as the *meteorite name* or also if it was *found* or *seen falling*, a *textual representation* was appropriate when a point is selected. Additionally, also allowing precise *mass* & *time* to be displayed.

- **Relating datum:** the *meteorite mass* is measured in *grams* and points are different by *orders of magnitude* $\rightarrow$ *logarithmic scale*.

5 Implementation

Building the outlined application was done with *JavaScript*, *D3* and *TopoJSON* since these are well documented and familiar to the author. Following the prepared guideline in Section 4, the following was done:

- **Drawing map:** by converting *Natural Earth's* vector data to the TopoJSON format, one can use the built-in D3 JSON loader and thereafter project data points using the *Mercator transform*. The generated points are then drawn using D3 and SVG `path` nodes.

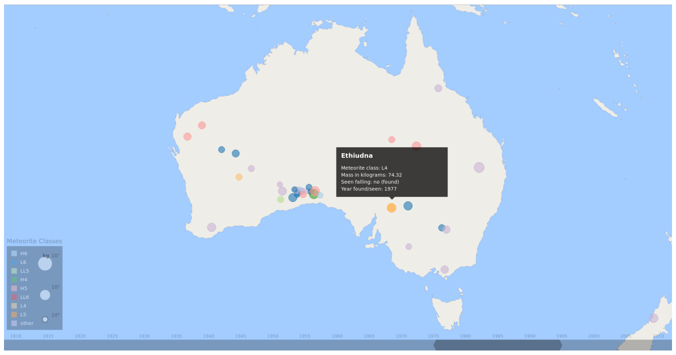
- **Map navigation:** re-drawing the map from scratch each time an interaction (move or zoom) is made wasn't an option, since it significantly reduced performance. By grouping together `path` nodes in a SVG `g` node, one case use `translate` to batch apply movement and scaling operations. All transformations caused by interacting with the map are derived from the built-in D3 `zoom`.

- **Spawning points:** the dataset is bundled in a CSV file, here, the built-in D3 CSV loader was used. After this, the points are placed on top of the map, also by using the Mercator projection. Thereafter, each of these data are given a radius which scales *logarithmically* with the mass and colored by their *classification*. To keep the color palette manageable, only the eight most common classes have unique colors, all others are group colored "others". These are also transformed with the map navigation interactions.

- **Filtering points:** is handled by having three control parameters: *minimum mass*, *chosen meteorite class* and a *time interval*. When any of these are updated, all data are checked to see if they satisfy all three conditions, if not, the SVG node is set to not be shown. This is triggered by brushing the timeline or using the "legend".

- **The interface:** constructed by using SVG and *not* applying any navigation transformations. There exists three main components: a *legend "filter"*, *brushing timeline* and an *information overlay*. Whenever the user hovers over a data point of interest, additional details on that particular data point is displayed, including the *meteorite name*, if it was *seen* or *found* and precise *mass* & *year*.

Perhaps a more visual depiction of the application is in order, see the Figure 2 below for an overview of how the web app is organized. Also, see the *attached appendices* for larger versions of these images.



Ethiudna

Meteorite class: L4
Mass in kilograms: 74.32
Seen falling: no (found)
Year found/seen: 1977

Fig. 2. Demonstration of the additional information overlay for one point

6 Results

Applying filtering to the large spatio-temporal dataset proved to be an effective method to reduce the amount of visual clutter, leading to a more manageable visual data exploration. See Figure 3 for the effects of applying both the *minimum mass* and the *time interval* filter with the values larger than 10^3 grams and time range between 1975 and 1995. Looking back at Figure 1 demonstrates that indeed the clutter has been reduced, but still not completely eliminated, some locations still have overlapping data points. Further discussion on possible improvements have been given later in this article, which should hopefully reduce the visual clutter further if applied; empirical tests would need to be done.

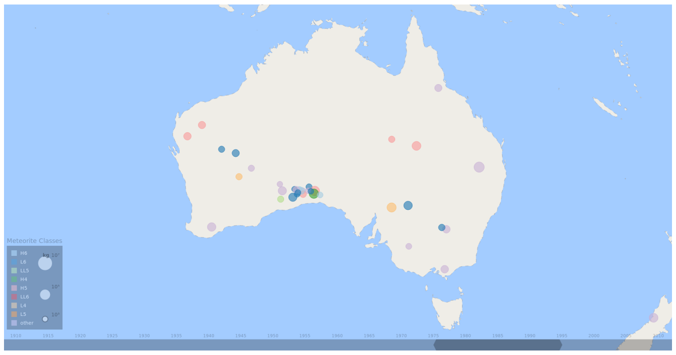


Fig. 3. Filtering reduces the clutter by removing any uninteresting datum

7 Evaluation

Reduces number of uninteresting data points by $|D \cap M \cap C \cap T| \div |D|$, where $d_i \in D$ is a data point and M, C, T the set of matching d_i for the *minimum mass*, *meteorite class* and *time interval* filtering parameters. Implicitly, this method reduces the number of overlapping data points. Techniques that reduce clutter better would produce less intersections.

8 Conclusions and Future Work

Integrating filtering in the large spatio-temporal dataset for the visual data exploration application, reduced the total amount of visual clutter. However, it did not succeed in removing everything, as points still overlap. Additionally, filtering requires *domain knowledge* for being truly effective, as shown in *Wood et al.* [6], which is a bit unfortunate since it would require an experienced user to properly explore the data. Nevertheless, the results imply that filtering is useful to reduce clutter.

Below follow some ideas and reflections on what could have been done differently, and what could be done in the future to improve them:

- **Meteorite clustering:** reducing the amount of initial clutter would have been desirable, and that could have been done with *DBSCAN*, a clustering algorithm for spatially grouping locations. Instead of having many points, one could hierarchically put these together, similar to how *Google Earth* does, as in *Wood et al.* [6]. Most clutters still present in Figure 3 would have been removed.

- **Clustering classifications:** there are a lot of different meteorite classes, as can be seen in *Weisberg et al.* [5], limiting the amount of them would have been desirable. By using the *Levenshtein distance* of the classes, and then clustering them together, one would be able to group similar classifications (e.g. H5, H6), and therefore remove some of the "noise" in the exploration process.

- **Integrating population:** an interesting relation that could have been nice to analyze, is the relationship between the population density of an area, and the amount of meteorites found and seen.

- **Improving performance:** the implementation is interactive, but could become better by speeding up the filtering process, e.g. by limiting the scope with *quad trees* or offload GPU computations.

References

[1] D. A. Keim, C. Panse, and M. Sips. Information visualization: techniques and opportunities for geovisualization. *Exploring Geovisualization*, 2004.

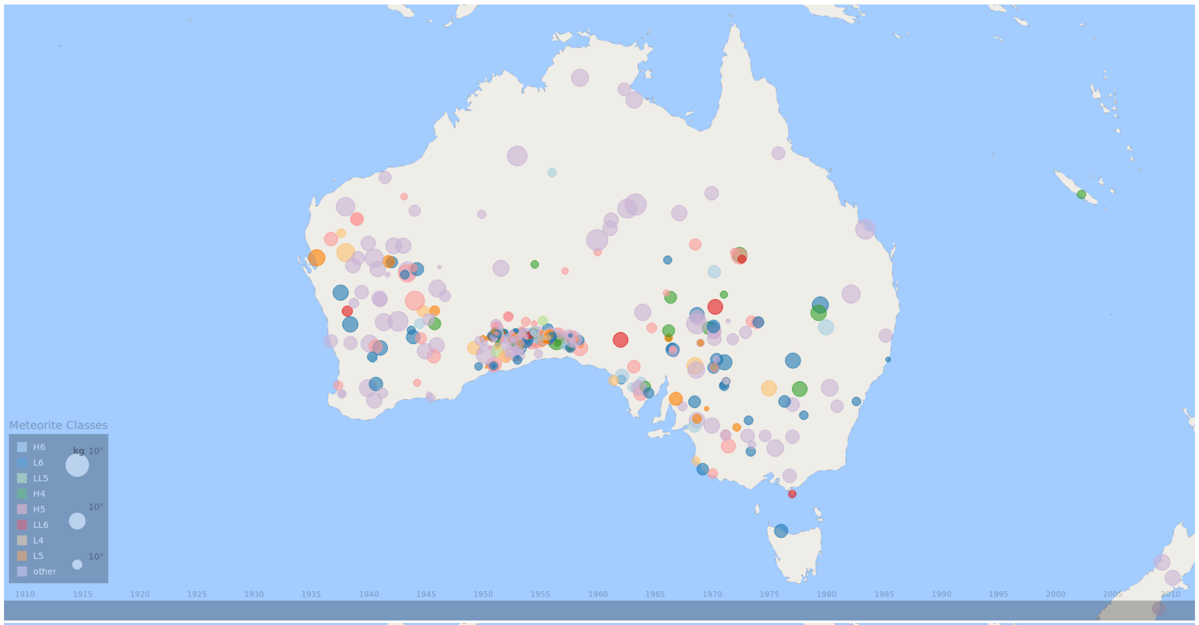
[2] NASA. *Meteorite landings*. `www.data.nasa.gov`, 2015.

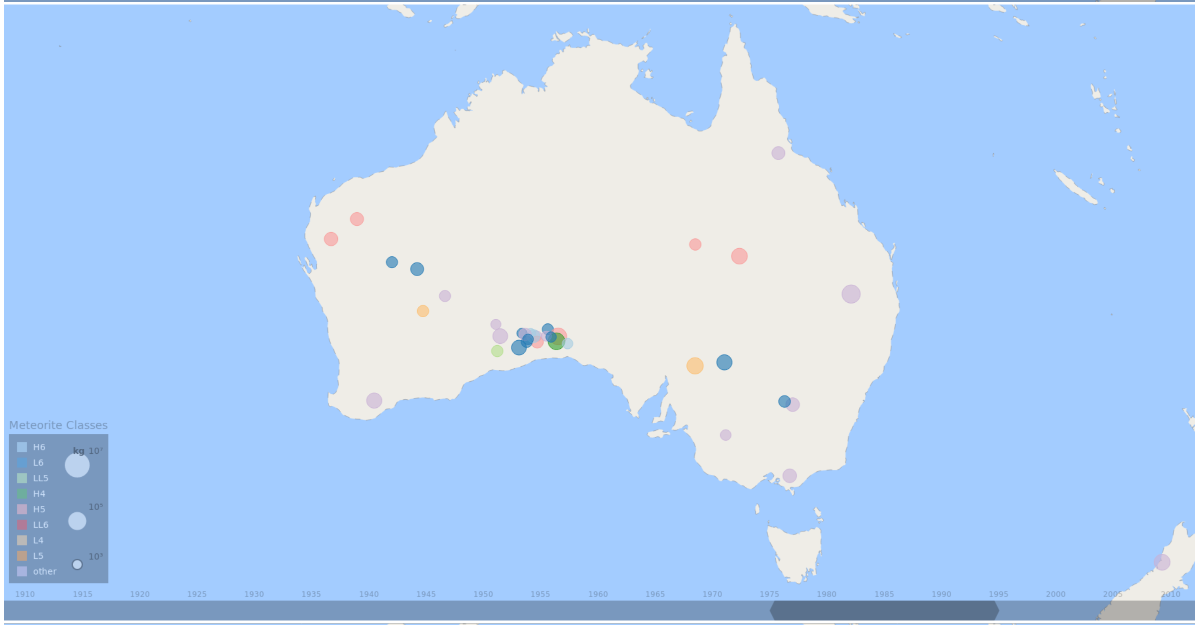
[3] B. Shneiderman. A task by data type taxonomy for information visualizations. In *Proceedings On Visual Languages*, pages 336–343. IEEE, 1996.

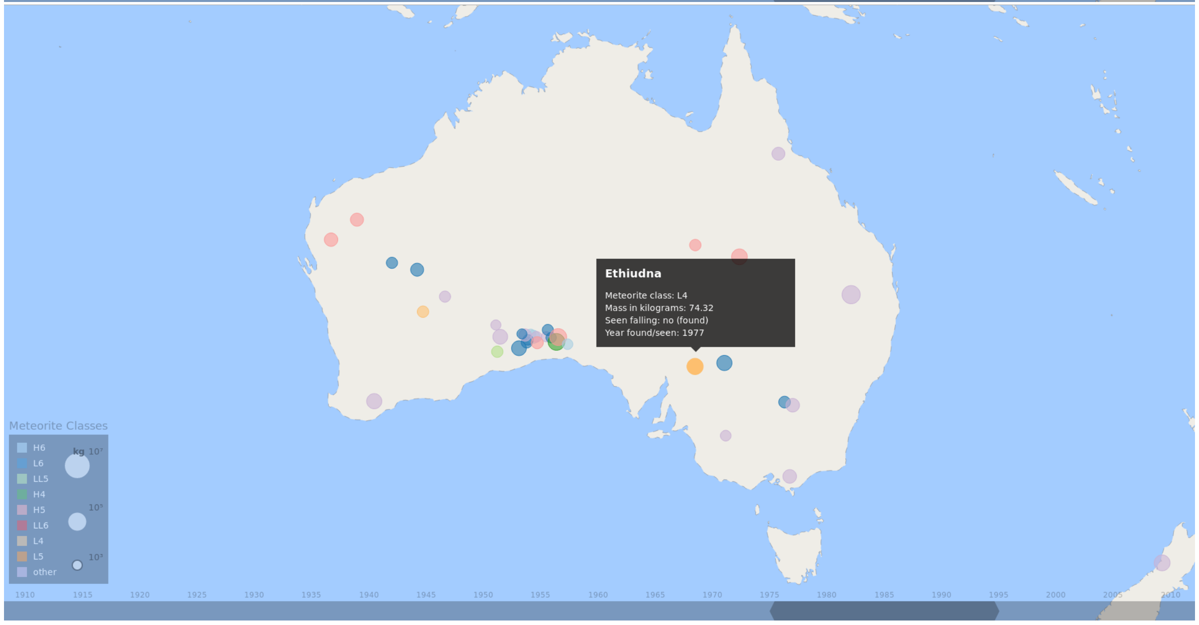
[4] R. Spence. *Information visualization*, volume 1. Springer, 2001.

[5] M. K. Weisberg, T. J. McCoy, and A. N. Krot. Systematics and evaluation of meteorite classification. *Meteorites and the early solar system*, 19, 2006.

[6] J. Wood, J. Dykes, A. Slingsby, and K. Clarke. Interactive visual exploration of a large spatio-temporal dataset: Reflections on a geovis mashup. *IEEE tr. on visualization and computer graphics*, 13(6):1176–1183, 2007.






Ethiudna

Meteorite class: L4
Mass in kilograms: 74.32
Seen falling: no (found)
Year found/seen: 1977