Contiguous animated edge-based cartograms for traffic visualization

Pedro Cruz, António Cruz and Penousal Machado
CISUC, Department of Informatics Engineering, University Coimbra

Abstract: We describe an experimental model to animate contiguous computer cartograms by distorting the topological lengths of their edges. We use traffic information for the city of Lisbon to generate cartograms where the length of a road traduces the velocity in it by using mesh of springs that reacts to data injections. Furthermore we apply this model to two visualizations: the first is a trajectory visualization of vehicles, creating a temperature map for the roads' velocities; the second is a figurative approach where we visualize Lisbon as a system of pulsing blood vessels. Our model can efficiently generate and animate edge-based cartograms with low representation errors.

Keywords: information visualization, traffic visualization, cartogram

1. Cartograms and traffic visualization

Cartograms are a technique for displaying geographic information by resizing map's regions according to a statistical parameter, but in a way that preserves the map's recognizability [1], [2]. Typically cartograms are contiguous and *value-by-area*: for example in a world map, the areas of countries can be resized proportionally to their GDP or population [1] without breaking the countries' adjacencies. There are cases, though, when manipulating lengths is more natural to a dataset than manipulating areas. Such case is traffic information where data is often reported in relation to a road (e.g. positions, velocities and number of vehicles). Distorting a city's map for traffic representation has already many techniques if in relation to a certain geographic point: *isochronic cartograms* are able to distort nodes in a map in relation to a reference point in order, for example, to represent travel time from that point to those nodes [3]. Traffic data can be rich and complex, and thus we are interested in discovering and generating the overall shape of a city if provided that slow traffic implies longest perceived road-routes and better flowing traffic, shortest distances. Edge-based cartograms can distort the length of the edges on a map in order to represent some type of value over those edges (e.g. velocities on a road network). Those types of cartograms are under-explored, having only been applied in its non-contiguous version [4], where the lengths of road segments are used to display traffic congestion, but without one distorted segment affecting the others that are structurally connected to it.

In this work we describe an experimental model for animated contiguous edge-based cartograms of traffic information. Our model is a mesh of interconnected springs generated from a road map. Data is injected in the system in order to distort the length of the roads accordingly with traffic velocity in them, making the city expand and compress. Spring-based approaches for value-by-area cartograms exist [5], and just like in our *"value-by-length"* cartogram it tries to preserve the initial shapes of the map while distorting it accordingly with data. The cartogram is animated, being able to continuously adapt to new data states at satisfactory framerates, while in several cartogram approaches each frame can take several seconds or hours be generated [2], [5].

The visual analytics of movement has its own state-of-the-art techniques such as the simultaneous representation of trajectories [6] or the spatial aggregation into density surfaces [7], but not being based on cartograms. We apply the common visualization of trajectories technique to our animated cartogram, being able to represent the newly discovered shape of the city and the actual data that caused such distortions.

Furthermore, we are interested in exploring new graphical languages that are not usually seem in visual analytics. Just like Chernoff represents points in an n-dimensional space by using faces [8], or the fish-eye zoom [9] that resembles a physical lens, we intend to use more familiar languages for traffic depiction. In our second visualization application to our animated cartogram, we use a system of pulsing blood vessels to portray the dataset [10] with a strong graphical metaphor. We find cartograms an appropriate technique to experiment with such new languages because they distort reality and *"shock"* readers with an unusual and provocative perspective on common topics [11]. We argue that cartograms are more evocative due to the peak shift effect [12], the tendency to respond more strongly to exaggerated versions of a reference

stimuli than to the reference itself, provided that the exaggeration makes the reference more perceptible, where in our case the reference representation is the standard road map of the city. With this, we say the application of our second visualization model to the traffic cartogram is a *caricatural* approach to visualization, since we are distorting our representation in order to emphasize a data dimension while using a highly figurative metaphor.

2. Spring model for edge cartograms

The model to generate our animated cartogram is a mesh of interconnected springs that is generated from a set of edges based on Lisbon's map. The lengths of the springs are distorted in order to reflect the traffic velocities on Lisbon's roads. First, we computed the overall average velocity for each road, and if the current velocities are bellow the overall average of a certain road, that road tends to distend as if distances were larger, and if the velocities are bellow, it tends to shrink. Our model has three types of springs:

a) the *backbone springs* that make the roads themselves and are used to alter the roads lengths (they are the black roads on figures 1 and 2);
b) the *inner springs* of each road that further connect their points, trying the preserve the road's shape while adjusting to its new configuration (figure 1 in orange);
c) the *connective springs* that connect one roads among each other, trying to preserve their relative distances and the overall appearance of the original map (figure 2 in blue).

In this section we describe how we generated the mesh of *backbone*, *inner* and *connective* springs from Lisbon's map; what is the nature of the traffic dataset that we used; how we used this data to excite the spring's mesh; and how it reacts to data.

2.1 Building the mesh of springs

In order to build our springs' mesh, we had to develop a suitable road structure made of unique points and edges that connect them. We extracted any type of road from OpenStreetMap (OSM) for Lisbon's area (latitudes [38.69°, 38.84°] and longitudes [−9.28°, −9.08°]) and mapped to roads to meters on a plane by using the Mercator projection and the WGS 84 standard equatorial radius.

Because the OSM's format can have one road defined through several data structures, we had to merge such structures for when they referred to the same road name and when they had coincident endpoints.

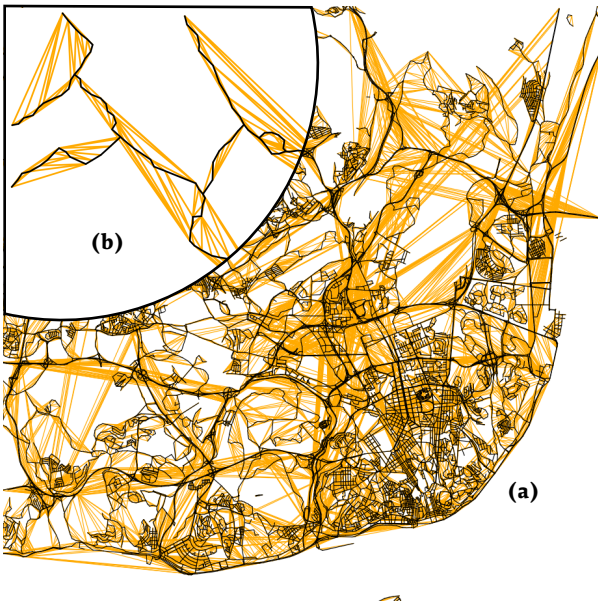

Figure 1. (a) The road map formed by the *backbone edges*, together with every *inner edge* in orange. (b) Detail of the inner edges that are related with the Delaunay graph of the points of each road.

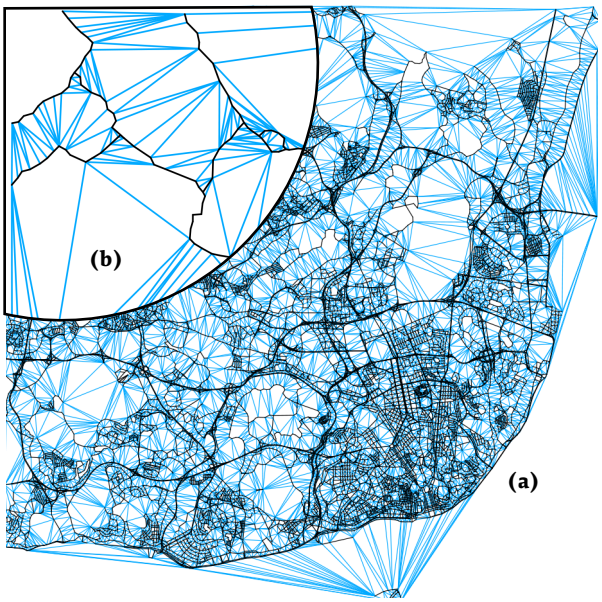

Figure 2. (a) The set of *connective edges C* in cyan. (b) Detail of the connective edges that are related with the Delaunay graph of all the points in the map.

Technical notes

Delaunay graph
The Delaunay graph of a set of points links the points in order to divide the region into triangular tiles such that no point is inside the circumcircle of any tile. This method tends to avoid skinny triangles by maximizing the minimum angle between any two edges. The concept was first introduced by Boris Delaunay in 1934 [1]. In order to determine the Delaunay graph of a set of points we used the QuickHull [2] algorithm that has a time complexity of $O(n\ log(n))$.

Catmull-Rom splines
Catmull-Rom splines [3] are smooth parametric curves that interpolate between a set of points, and are widely used in computer graphics. This method does not require the definition of additional control points for the curves since the original set of points also makes up the control vertices for the curve.

Verlet integration
Verlet integration is a numerical method to integrate Newton's equations of motion [4]. It is widely used for particle systems since it provides greater stability with decreased amount of computation when compared with other methods (e.g. Euler) [5].

Mercator projection
The Mercator projection maps the earth's sphere on a plane as if it was projected on a cylinder. It was invented by Gerardus Mercator in 1569 [6], and it is still the most often used map projection in the world. This projection considerably distorts the size and shapes of large objects that are closer to the poles. Given the large scale of our map and the distance of Lisbon from the North Pole, the distortions are negligible.

Additional references

[1] B. Delaunay, "Sur la sphere vide," *Bulletin de lAcademie des Sciences de lURSS. Classe des sciences mathematiques et naturelles*, no. 6, pp. 793–800, 1934.
[2] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *Transactions on Mathematical Software (TOMS*, vol. 22, no. 4, pp. 469–483, Dec. 1996.
[3] E. Catmull and R. Rom, "A class of local interpolating splines," *Computer aided geometric design. Academic Press.*, pp. 317–326, 1974.
[4] L. Verlet, "Computer 'Experiments' on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules," *Phys. Rev.*, vol. 159, pp. 98–103, Jul. 1967.
[5] T. Jakobsen, "Advanced character physics," presented at the Game Developers Conference, San Francisco, 2001, pp. 383–401.
[6] J. P. Snyder, "Mercator projection," in *Map projections--A working manual*, no. 7, Washington D.C.: U.S. Geological Survey, 1987, pp. 37–47.

This resulted in 4,321 roads from the initial 5,131 data structures. We then proceeded to guarantee that any reference to a point in the OSM data was indeed a unique geographical point. This resulted in a scheme that described a set of roads, being that each road is a sequence of references to points (x,y), having each pair (x,y) a unique reference and meaning that roads can share points in common. This is important since by sharing points roads are structurally interdependent from one another. Furthermore, we also removed unnecessary complexity from the OSM data, by removing points on the same road that were closer than 100 meters. This translated in 20,315 points orphaned of references that were removed, leaving 17,664 unique points. Given our map scale of 1:60,000, this only results in loss of detail for distances inferior to 1.66 mm. Finally, roads that had only one reference to a point were removed, leaving 4,184 roads, 17,640 unique points of which 8,387 (48%) are shared between two or more roads.

We now describe how we generated the three types of springs for our mesh. From the OSM data we have a set of unique points that we call P and a set of roads that we call $\mathcal{R}$. A spring that connects two points is also a set $\{a,b\} \subseteq P$, such as two points can only be connected by one spring. Each road $R_i \in \mathcal{R}$ is a sequence of n points $(p_k)_{k=0}^{n}$ where $p_k \in P$. Each road generates a set of *backbone springs* B_i such as each spring connects a consecutive pair of points (p_k,p_{k+1}) from R_i. Let $\mathcal{DG}(A)$ of a set of points $A \subseteq P$ be the set of edges in the Delaunay graph of A. The set of *inner springs* I_i for a road R_i connects the points of R_i just like in $\mathcal{DG}(R_i)$ except for the connections that are already part of its *backbone springs*. More formally: $I_i = \mathcal{DG}(R_i) \setminus B_i$.

As we previously described, the *connective springs* connect roads among themselves. They are generated based on the Delaunay graph of all the points the points of all roads. So, the *connective springs* C are all the springs that connect the points in $\mathcal{DG}(P)$ that are not already a *backbone spring* or a *inner spring*. If we

call the set of all *backbone springs* $\mathcal{B}$ ($\mathcal{B}=\bigcup_i B_i$) and the set of all *inner springs* $\mathcal{I}$ ($\mathcal{I}=\bigcup_i I_i$), then:

$$C = \mathcal{DG}(P) \setminus (\mathcal{B} \cup \mathcal{I}).$$

With this, having the sets of *backbone*, *inner* and *connective* springs, we can generate the mesh of springs for our cartogram model.

2.2 Data

The dataset has traffic information for the city of Lisbon, consisting of about 2 million GPS traces of 1534 vehicles for the 30 days of October 2009. Each GPS trace is a (*latitude,longitude*) point in time with speed information in km/h. Considering the disparities in traffic density between weekends and weekdays we decided to visualize the daily cycle of Lisbon's traffic by only aggregating its weekdays, filtering out the weekends and being left with 1.8 million GPS traces. The data was animated from 0:00 to 23:59, iterating one by one minute and considering, for each minute, the GPS traces occurring in a time-window of the previous 30 minutes.

Although the data can be injected in our system for each simulation step (making animations of 1 minute data per frame), there is also the option to space out this data injection rate to give time for the springs to better adapt to new data states. The results of using such feature are discussed in the final section.

2.3 Exciting springs with data

In order to generate a velocity cartogram we deform each road in function of the relation between the current traffic speed on that road and its overall average traffic speed. So, if for a given time of the day, the current velocity in a road is below its daily average velocity, the road length should stretch. Conversely, if it is above the average velocity it should compress. We first computed in which road each GPS trace is located and after this, we computed:

a) the average velocity of each road for the whole day (*dailyAvgVel*);
b) the current average velocity and the current number of vehicles in each road R_i for each aggregation of $[t-30,t]$ minutes, being t the time currently being visualized (*currentAvgVel(t,i)* and *currentAvgVeh(t,i)* respectively).

For every point in P, we have a corresponding particle of unitary mass that is connected to other particles through our mesh of springs. A spring is an elastic device such as when compressed or stretched in relation to its rest length, it exerts a force proportional to its change in length. The coefficient of proportionality is a constant that traduces the spring's *strength*. The initial rest lengths of the springs are the initial distances between the pair of particles that it connects, such as no force is exerted. Each type of spring can have a different strength: k_B for the *backbone springs*, k_I for the *inner springs* and k_C for the *connective springs*. The behavior of the particles and springs is simulated with a physics engine with a Verlet integration solver, with unitary step and 50 iterations per step.

When running the simulation, for the time t, we compute a distortion ratio *distortionRatio(t,i)* for each road R_i:

$$distortionRatio(t,i) = \frac{dailyAvgVel}{currentAvgVel(t,i)} .$$

In order to distort the length of a road R_i we apply the *distortionRatio(t,i)* to its *backbone* and *inner* springs in order to sustain the road's shape. This is done by multiplying the initial spring's length by the distortion ratio, "scaling" the road inversely to the current velocity in it.

As previously mentioned, the *connective springs* exist to maintain the overall form of Lisbon's map, but if left unchanged can vastly constrain the distortions that we apply to the roads. Therefore, the length of each *connective spring* is also distorted by the average of the distortion ratios from all the roads that connects. This causes regions of the map to extend or compress in order to accommodate the new roads' lengths in the area.

When we change a spring's length, the particles that it connects are shifted in order to accommodate its new length by the physics engine. Nevertheless, each particle's movement is constrained by having other springs connected to it, each one with their respective lengths and forces. Due to these constraints each new spring length is not necessarily completely accommodated in the mesh, meaning that each road velocity has a *representation error*. Those errors are a natural consequence of the shape constraints that any contiguous cartogram model has to incorporate [2]. Those models, just like ours, typically exhibit a tradeoff between representation error and shape error, such as one can have representation errors close to zero but with increased shape errors, or by preserving more the shapes of the initial map, having the cost of not appropriately distorting the map to represent the data.

We observed that in order to have a good tradeoff between shape error and representation error, we also

had to find adequate balances among the strengths k_B, k_I and k_C. Through a set of preliminary experiments we found out a good balance among such strengths for this application and thus will be discussing the results for $k_B = 1.20$, $k_I = 0.02$ and $k_C = 0.20$.

Figure 3 displays the average velocity of the vehicles being visualized while pinpointing four important states of the simulation for our model. Only the distorted *backbone springs* are drawn in Figure 3, forming the contiguous cartogram, animated through time. As it can be observed the city compresses in the evening when traffic velocities are higher, and expands when the velocities are lower. The distortion ratios for each road are also represented in figure 3 at the designated frames. The animation can portray the rhythms of the city in a continuous fashion, displaying global expansions and contractions in the distances as well as more localized ones.

3. Visualization applications

In this section we describe two visualization applications of our spring based cartogram, the first one is a common trajectory visualization, and the second experiments with new languages, being figurative and caricatural.

3.1 Trajectories visualization

This first visualization applied to our cartogram displays the recent trajectories of vehicles, creating a color-temperature map for roads' velocities. For each vehicle in the visualization time t, we draw a small white dot at the vehicle's current position. Then, we draw a trail for its t−30 minutes GPS positions. This enables to get a representation of the recent trajectories of every vehicle, illuminating the roads in Lisbon's map. Each trail is colored accordingly with the average velocity of its traces during the previous 30 minutes. For this, we used a color mapping that is inspired in traffic signals: red is "an almost stopped" corresponding to velocities below 10 km/h; green is a "you are good to go", corresponding to 50km/h; for velocities above 50km/h we use bluish greens and cyans (see Figure 4).

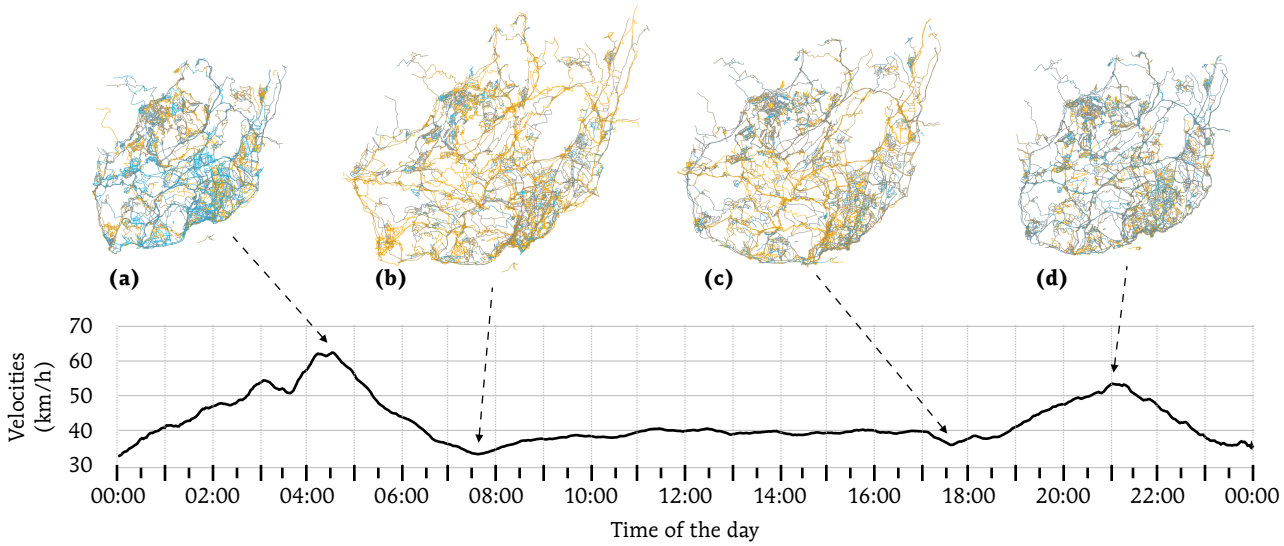

Figure 3. Average velocity (km/h) of the vehicles being visualized by time of the day. There is an absolute maximum at 4:31 of 62km/h, and absolute minimum at 7:35 of 35km/h, followed by a local minimum at 17:36 of 37km/h and a local maximum at 21:09 of 53km/h. Each road has interpolated colors accordingly with their current distortion ratios such as blue for 0.5 and bellow, gray for 1.0 and orange for 1.5 and above (0.5 ... 1.0 ... 1.5). The different the distortion ratio from 1, the higher is the tension that the spring exerts on the system. (a) refers to dawn time when most vehicles travel fast and thus the map is most compressed since distances are perceptually shorter (distortion ratios are generally bellow 1); (b) refers to the morning rush hour when traffic is the slowest and the city's distances are expanded (distortion ratios are generally above 1). The pattern repeats itself in the afternoon rush hour (c) and at another velocity peak in the evening (d). The results can be better visualized in https://vimeo.com/91325884.

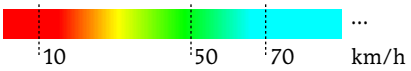

Figure 4. Color scale used to map the velocities and illuminate the visualization of GPS traces.

With this visualization, GPS traces can be visually grouped, providing a glimpse on traffic volume and speed: thicker clusters of trails indicate higher volume. These thick clusters identify some of Lisbon's highways where the lower velocities observed are around yellowish greens during rush hours. One discrepancy that became apparent was that areas with less traffic volume but where traffic is much more slower would not be as visually emphasized – for example downtown with its narrower streets causes traffic to slow down but it is not as distinguishable as the highways are. In order to attenuate such discrepancies in the emphasis, every GPS point of every trail is also drawn with a circle with very low opacity, having the same color at the trail. This confers a proper salience to low traffic areas while also emphasizing spots with stationary vehicles.

As previously mentioned, this visualization is mapped to the animated cartogram previously described. In order to do so, when computing the closest road to a GPS point, we also store its relative distance to the closest *backbone spring*. Those relative positions are used to draw each GPS trace in relation to the distorted *backbone springs*. Two snapshots of this visualization can be seen in figure 5 (a) and (c).

3.2 Figurative visualization of blood vessels

The continuous animation of the cartogram is highly complex, presenting variable behaviors depending on the city's region but also exhibiting emergent ones such as the city compressing and expanding as a whole. This complexity inspired us to use the metaphor of a *"living organism"* on a graphical and conceptual level. This figurative visualization of Lisbon portrays the city as a system of blood vessels with streams of vehicles. Every vessel is red, having brighter reds for high velocities and darker reds for lower ones, as if "blood" was clotting. On top of that, the thickness of the vessels is directly proportional to the current number of vehicles, and since the vessels derive from our springs' model, their lengths also vary with the velocities on the corresponding road.

We also implemented a pulsing motion for each vessel with a rate proportional with the velocities at the corresponding road. For this, we compute a perpendicular segment to each *backbone spring* of a road. A perpendicular segment has its initial angle α expressed in relation to a world coordinate axis. Likewise, there is α_p and α_n, respectively the angle of the perpendicular to the previous and next *backbone spring*. Each segment travels along the respective *backbone spring* and restarts when finished. The angle that a segment makes with its spring is not constant as it travels. Considering that $x \in [0,1]$ is the position of the segment on a spring, and that β is the angle of that segment, β is calculated by linearly interpolating the following conditions: when $x = 0$, $\beta = (\alpha_p+\alpha)/2$; when $x = 0.5$, $\beta = \alpha$; and when $x = 1$, $\beta = (\alpha_n+\alpha)/2$. This behavior is illustrated in figure 6. The lengths of these segments are proportional to the current number of vehicles on the corresponding roads, making the thickness of the vessels. The shape of each vessel is formed by uniting the segment's endpoints with Catmull-Rom splines.

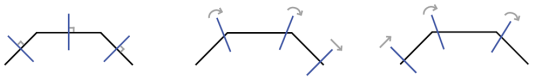

Figure 6. Diagram of the interpolation of the segments used to implement both the thickness of a vessel and its pulsing motion.

We also added a *"stream of cells"* inside each vessel to represent the flow of vehicles. The number of cells is proportional do the number of vehicles, and their velocity is proportional to the average velocity on the road. This model brings a great emphasis to the roads with more traffic while also displaying their current velocity through the motion of the *"stream of cells"* inside each vessel and also through brighter and darker reds. Two snapshots of this visualization can be seen in figure 5. (b) and (d).

4. Discussion

As previously mentioned, generating contiguous cartograms for a given dataset comes with a representation error. We computed this error by adapting the area error function in the work of Keim [2] for lengths. The function expresses the relative error between the actual lengths of roads and their expected lengths in our cartogram. Since each time-frame represents a different set of data, the error has different results for each simulation step (see figure 7). The error has a minimum of 0.018 at 15:00 (a period when the map has fewer tensions since the velocities are close to the daily averages) and a maximum of 0.031 at 07:40 (the morning rush hour when the tensions are greater). These results are consistent with the rep-

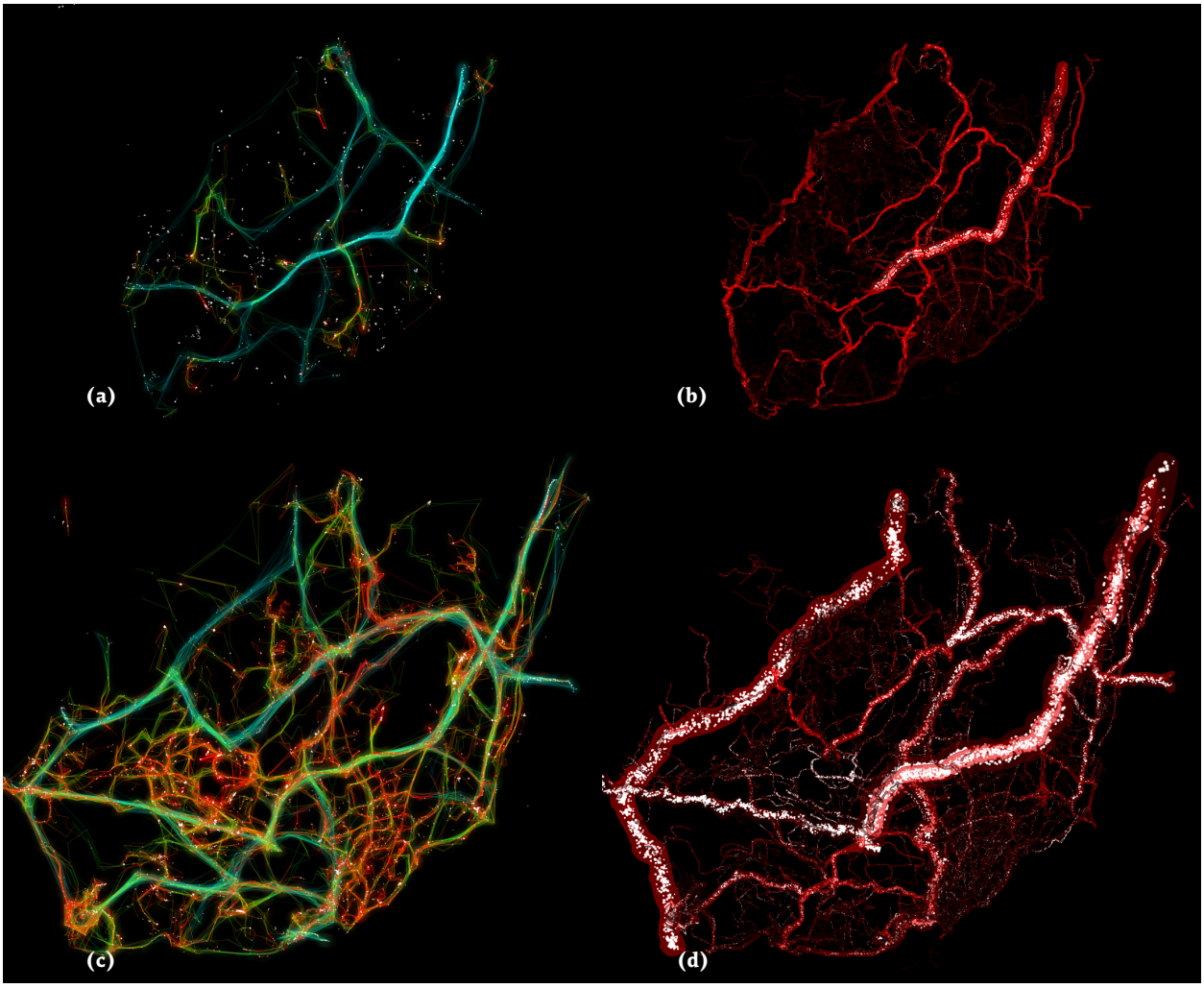

Figure 5. Snapshots of the visualization applications of our cartogram model. The direct visualization of GPS traces in (a) when the city is most compressed at dawn at 4:31; and in (c) when the city is more distended at the morning rush hour at 7:35. The figurative visualization of blood vessels in (b) at 4:31; and in (d) at 7:35. Please notice how the main roads increase in thickness and in volume of cells/vehicles. The results can be better visualized in https://vimeo.com/91325884.

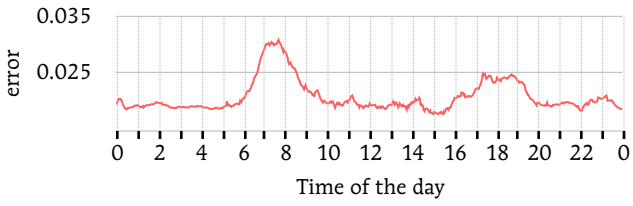

Figure 7. The representation error for each time-frame, with a maximum of 0.031 and a minimum of 0.018).

resentation errors presented in other applications of contiguous cartograms [2], [5]. By testing with different parameterizations of (k_B, k_I, k_C) we are able to get representation errors close to 0, but at the expense of increasing the shape error and distorting the roads in a less recognizable way. The results presented here are a compromise between the two errors. As future work we plan to do a comprehensive parametric study for (k_B, k_I, k_C). Our tests were run on a 2.7 GHz IntelCore i5 with 8GB of RAM and the cartogram generation without the visualization applications was computed and

rendered at a rate of 6-7 frames per second which potentially deems this model suitable for realtime applications – other models can take several seconds or hours to generate a cartogram [2], [5].

Our model was also tested at a lower data injection rate (once per each 90 frames), meaning that the system has more iteration steps available to adapt itself to new data states. Although the representation error decreased, such difference was not significant. Nevertheless, this option can be used to generate static cartograms with better accuracy and simulate how the cartogram model would react to real world data injections.

With this work we presented a new model to generate contiguous edge-based cartograms for cities. In addition to its map representation, we also explored how other two concrete visualizations work when mapped over an animated cartogram. This way, we were able to portray the richness of the dataset and not only aggregated velocities. The first visualization is a common trajectory visualization, and the second we experimented with new graphical languages, such as representing the city as a system of pulsing vessels. With this second model traffic volume and velocity are displayed in a less abstract way with organic aesthetics. In the future we intend to study this model and approach against other traffic datasets and cities.

Acknowledgments

Data provided in the context of MIT Portugal CityMotion project. Pedro Cruz has a grant from FCT, ref SFRH/BD/77133/2011.

References

[1] W. Tobler, "Thirty five years of computer cartograms," *ANNALS of the Association of American Geographers*, vol. 94, no. 1, pp. 58–73, 2004.

[2] D. A. Keim, S. C. North, and C. Panse, "CartoDraw: a fast algorithm for generating contiguous cartograms," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 10, no. 1, pp. 95–110, 2004.

[3] J. W. Clark, "Time–Distance Transformations of Transportation Networks," *Geographical Analysis*, vol. 9, no. 2, pp. 195–205, 1977.

[4] Y. H. Wu, "Non-Connective Linear Cartograms for Mapping Traffic Conditions.," *Cartographic Perspectives*, no. 65, pp. 33–50, 2010.

[5] D. H. House and C. J. Kocmoud, "Continuous cartogram construction," *VISUAL-98*, pp. 197–204, 1998.

[6] N. Andrienko and G. Andrienko, "Visual analytics of movement: An overview of methods, tools and procedures," *Information Visualization*, vol. 12, no. 1, pp. 3–24, Jan. 2013.

[7] N. Willems, H. Van De Wetering, and J. J. Van Wijk, "Visualization of vessel movements," *Computer Graphics Forum*, vol. 28, no. 3, pp. 959–966, Jun. 2009.

[8] H. Chernoff, "The use of faces to represent points in k-dimensional space graphically," *Journal of the American Statistical Association*, pp. 361–368, 1973.

[9] G. W. Furnas, "Generalized fisheye views," presented at the CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems, 1986.

[10] P. Cruz and P. Machado, "Visualizing the circulatory problems of Lisbon," presented at the ACM SIGGRAPH 2011 Posters, 2011, p. 92.

[11] D. Dorling, "Area cartograms: their use and creation," *Concepts and Techniques in Modern Geography series*, vol. 59, p. 4, 1996.

[12] V. S. Ramachandran and W. Hirstein, "The science of art: A neurological theory of aesthetic experience," *Journal of Consciousness Studies, 6*, vol. 6, no. 7, pp. 15–51, 1999.

Bios

Pedro Cruz is a data visualization specialist and explorer. He is a Ph.D. student at University of Coimbra.

António Cruz is a M.Sc. student in Design and Multimedia at the University of Coimbra.

Penousal Machado, is a teacher and scientific director of the Computational Design and Visualization Lab. of the University of Coimbra. He is the author of more than 100 refereed journal and conference papers and the recipient of several scientific awards.