Contiguous Animated Edge-Based Cartograms for Traffic Visualization

Pedro Cruz, António Cruz, and Penousal Machado
University of Coimbra

Cartograms are a technique for displaying geographic information by resizing a map's regions according to a statistical parameter in a way that still preserves the map's recognizability.[1,2] Typically cartograms are contiguous and value-by-area; for example, for a world map, the areas of countries can be resized in proportion to their GDP or population[1] without breaking the countries' adjacencies. There are cases, however, when manipulating lengths is more natural to a dataset than manipulating areas. One such case is traffic information, where data is often reported in relation to a road (such as positions, velocities, and numbers of vehicles).

We are interested in discovering and generating cartograms for cities where slow traffic perceptually implies longer distances and better traffic flows would shorten those distances. Isochronic cartograms are a common technique that distorts nodes on a map in relation to a reference point to represent travel time,[3] velocities, or other variables. Edge-based cartograms can distort a map without a reference point, manipulating the edges' lengths to express an edge-dependent variable (such as velocities on a road network). These cartograms have been used occasionally to distort distances according to traffic velocity, but they use either small portions of the road network[4] or independent and disconnected edges.[5]

Our experimental model for animated contiguous edge-based cartograms shows traffic information by using a mesh of interconnected springs, generated from a city road map. We inject data in a system to alter the road's lengths according to their velocities, causing the city map to expand and compress accordingly. Spring-based approaches for value-by-area cartograms exist,[6] and just like in our value-by-length cartogram, they try to preserve a original map's shapes. Several implementations take seconds or hours to generate one static cartogram,[2,6] but our approach can continuously animate a cartogram through different information states with satisfactory frame rates and errors.

The visual analytics of movement has its own state-of-the-art techniques such as simultaneously representing trajectories[7] or the spatial aggregation into density surfaces,[8] but it has not generally utilized the cartogram technique. By applying the visualization of trajectories technique on top of our cartogram model, we are able to both distort the map and see which data caused the distortion.

From a graphical perspective, we are interested in exploring new languages not generally utilized in visual analytics. For example, in one animated cartogram, we use a graphical metaphor to depict a city as a system of pulsing blood vessels.[9] 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.[10] We argue that cartograms are evocative because they elicit the *peak shift effect*,[11] which is 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. In our case, the reference representation is a standard city road map. With this, we consider the vessels visualization on the cartogram as a caricatural approach to visualization because we are distorting our representation to emphasize a data dimension while using a highly figurative metaphor.

Spring Model for Edge Cartograms

Our model is a mesh of interconnected springs generated from a set of edges from a Lisbon city map. The springs' lengths are distorted to reflect the traffic velocities on the roads. First, we computed the overall average velocity for each road. If

 Published by the IEEE Computer Society

the current velocities are below that overall average, the road stretches as if distances were larger, and if the velocities are above, they shrink. The model has three types of springs:

- *backbone springs* make the roads themselves and are used to alter the roads' lengths (black roads in Figures 1 and 2);
- *inner springs* of each road further connect their points, trying to preserve the roads' shapes while adjusting to their new configurations (orange roads in Figure 1); and
- *connective springs* connect roads to each other, trying to preserve their relative distances and the overall appearance of the original map (blue roads in Figure 2).

Here, we describe how we generated the backbone mesh, inner springs, and connective springs from Lisbon's map; the nature of the traffic dataset being used; how this data operates on the springs' mesh; and how the mesh reacts.

Building the Spring Mesh

We developed a suitable road structure made up of unique points and edges that connect them. Roads of all types were extracted from OpenStreetMap (OSM) for Lisbon's area (latitudes [38.69°, 38.84°] and longitudes [–9.28°, –9.08°]) and mapped to meters on a plane using the Mercator projection and the WGS 84 standard equatorial radius. (See the "Technical Notes" sidebar for definitions of these and other technical terms.)

Because the OSM's format can have one road defined through several data structures, we merged such structures when they referred to the same road name and when they had coincident endpoints. This resulted in 4,321 roads from the initial 5,131 data structures. We then guaranteed that any reference to a point in the OSM data was a single unique geographical point. This resulted in a scheme that describes a set of roads, where each road is a sequence of references to points (x, y) and each pair (x, y) is a unique reference. This means that roads can share points in common and are hence interdependent on one another. Furthermore, we also removed unnecessary complexity from the OSM data, removing points on the same road closer than 100 meters. This resulted in the removal of 20,315 points, leaving 17,664 unique points. Given our map scale of 1:60,000, this only results in loss of detail for distances shorter than 1.66 mm. Finally, roads that had only one reference to a point were removed, leaving 4,184 roads and 17,640 unique points, of which 8,387 (48 per-

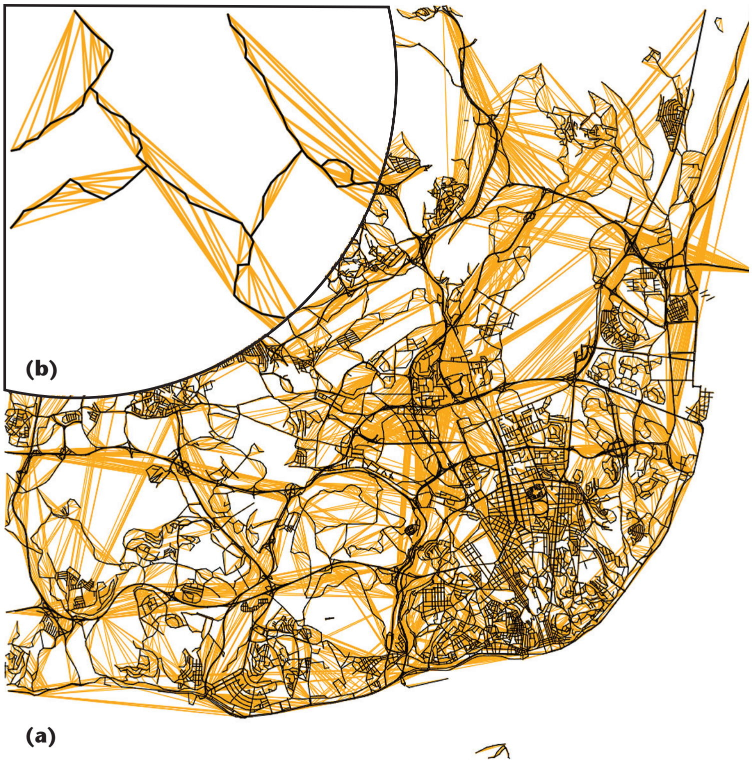

Figure 1. Spring model for edge cartograms. (a) The road map consists of the backbone edges in black and the inner edges in orange. (b) The enlarged section shows the detail map of the inner edges, which are related with the Delaunay graph of each road's points.

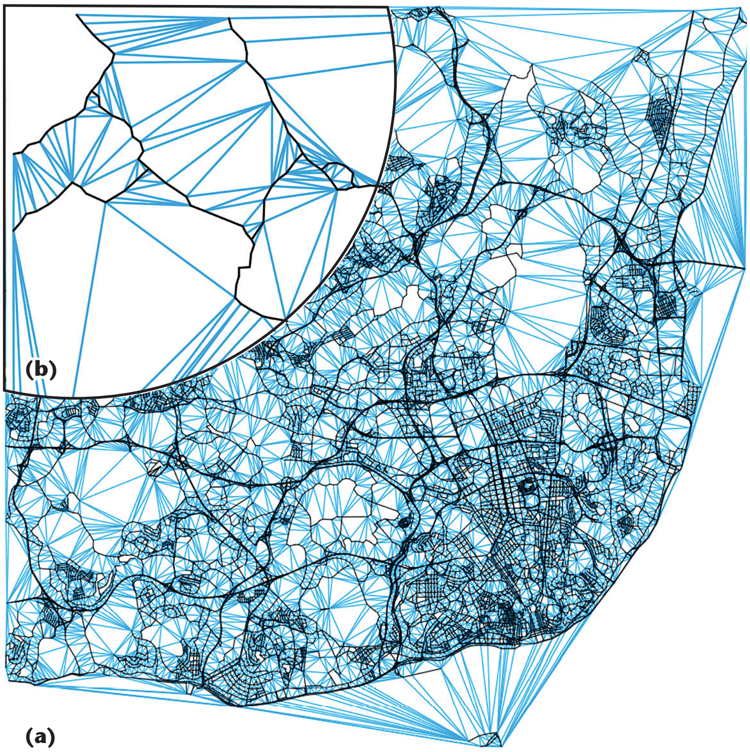

Figure 2. Connective springs in edge cartograms. (a) The set of connective edges are in cyan. (b) The enlarged section shows the detail map of the connective edges, which are related with the Delaunay graph of all the map points.

Technical Notes

Visualization in general uses many state-of-the-art techniques to enhance computational performance and solve graphical problems. These short notes elucidate some of the techniques used in this application.

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] To determine the Delaunay graph of a set of points, we used the QuickHull algorithm,[2] which 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 because 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 because it provides greater stability with a decreased amount of computation when compared with other methods (such as Euler).[5]

Mercator Projection

The Mercator projection maps the Earth's sphere on a plane as if it was projected on a cylinder. Invented by Gerardus Mercator in 1569,[6] 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.

References

1. B. Delaunay, "Sur la sphere vide" [On the Empty Sphere], *Bulletin de l'Academie des Sciences de lURSS, Classe des sciences mathematiques et naturelles*, no. 6, 1934, pp. 793–800.
2. C.B. Barber, D.P. Dobkin, and H. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," *Trans. Mathematical Software*, vol. 22, no. 4, 1996, pp. 469–483.
3. E. Catmull and R. Rom, "A Class of Local Interpolating Splines," *Computer Aided Geometric Design*, Academic Press, 1974, pp. 317–326.
4. L. Verlet, "Computer 'Experiments' on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules," *Physical Rev.*, vol. 159, 1967, pp. 98–103.
5. T. Jakobsen, "Advanced Character Physics," *Proc. Game Developers Conf.*, 2001, pp. 383–401.
6. J.P. Snyder, "Mercator Projection," *Map Projections—A Working Manual*, no. 7, US Geological Survey, 1987, pp. 37–47.

cent) are shared between two or more roads.

OSM data defines each lane direction separately for highways and main roads. This proves useful because vehicles can have different speeds depending on their travel direction, and they only affect the corresponding lane.

We generated the three types of springs in the mesh. From the OSM data, we have a set of unique points P and a set of roads $\mathcal{R}$. A spring that connects two points is also a set $\{a, b\} \subseteq P$, so 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 that 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 described earlier, the connective springs connect roads. They are generated based on the Delaunay graph of all the points of all the 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. We call the set of all backbone springs

$$\mathcal{B}\left(\mathcal{B} = \bigcup_i B_i\right)$$

and the set of all inner springs

$$\mathcal{I}\left(\mathcal{I} = \bigcup_i I_i\right), \text{ then } C = \mathcal{DG}(P) \setminus (B \cup I).$$

Data

The dataset contains traffic information for the city of Lisbon, consisting of about 2 million GPS traces of 1,534 vehicles for 30 days in October 2009. Each GPS trace is a (latitude, longitude) point in time with speed information in kilometers per hour. Considering the disparities in traffic density between weekends and weekdays, we decided to filter out the weekends and visualize the daily cycle of Lisbon's traffic by only aggregating its weekdays. This left us with 1.8 million GPS traces in the dataset. The data was animated for the times 0:00 to 23:59, iterating in one-minute increments 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), we can also space out the data injection rate to give time for the springs to better adapt to new data states.

Exciting Springs with Data

To generate a velocity cartogram, we deform each road based on the relation between the current traffic speed on that road and its overall average traffic speed. If for a given time of the day the current velocity in a road is below its daily average, the road length should stretch. Conversely, if it is above the average velocity, it should compress.

We first computed where road or lane each GPS trace is located. After that, we computed

- each road's average velocity for the whole day (dailyAvgVel) and
- the current average velocity and the current number of vehicles on each road R_i for each aggregation of (t – 30, t) minutes, where t is 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 that is compressed or stretched in relation to its rest length, and it exerts a force proportional to its change in length. The coefficient of proportionality is a constant that translates the spring's strength. The springs' initial rest lengths are the initial distances between the pair of particles that it connects, and hence 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 using a physics engine with a Verlet integration solver, with a 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:

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

To distort the length of a road R_i, we apply the distortionRatio(t, i) to its backbone and inner springs 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.

The connective springs exist to maintain the overall form of Lisbon's map, but if left unchanged, they 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 it connects. This causes regions of the map to extend or compress to accommodate the new road lengths in the area.

To generate a velocity cartogram, we deform each road based on the relation between the current traffic speed on that road and its overall average traffic speed.

When we change a spring's length, the particles that it connects are shifted to accommodate its new length by the physics engine. Additionally, each particle's movement is constrained by having other springs connected to it, each one with its own respective length and force. Because of these constraints, each new spring length is not necessarily completely accommodated in the mesh, which results in representation errors. Those errors are a natural consequence of the shape constraints that any contiguous cartogram model has to incorporate.[2] Such models, just like ours, typically exhibit a tradeoff between representation and shape errors—that is, representation errors close to zero result in increased shape errors, whereas preserving more of the initial map's shapes means not appropriately distorting the map to represent the data.

For a good tradeoff between shape and representation errors, we had to find an adequate balance among the strengths k_B, k_I, and k_C. Through a set of preliminary experiments, we determined a good balance among such strengths for this application was $k_B = 1.20$, $k_I = 0.02$ and $k_C = 0.20$.

Figure 3 displays the average velocity of the vehicles being visualized while pinpointing four important simulation states for our model. Only the distorted backbone springs are drawn in Figure 3, forming the contiguous cartogram, animated through time. The figure shows that 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

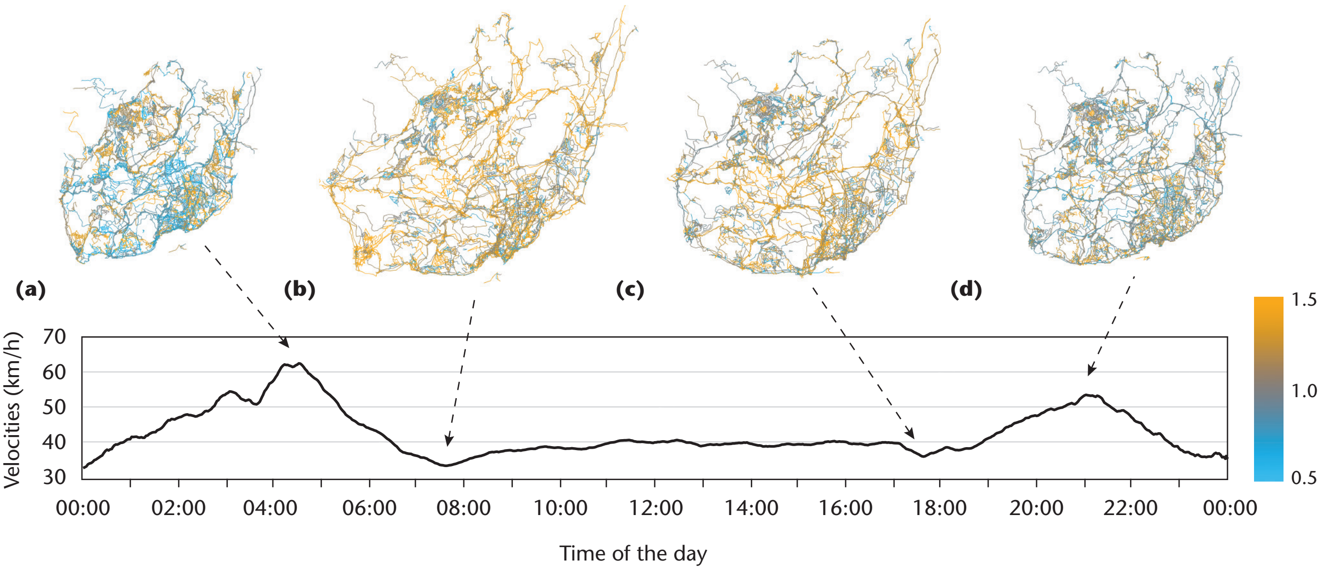

Figure 3. Average velocity (km/h) of the vehicles being visualized by time of day. There is an absolute maximum at 4:31 of 62 km/h and an absolute minimum at 7:35 of 35 km/h, followed by a local minimum at 17:36 of 37 km/h and a local maximum at 21:09 of 53 km/h. Each road has interpolated colors according to its current distortion ratios (see the color scale). (a) At dawn, most vehicles travel fast and thus the map is most compressed because distances are perceptually shorter (distortion ratios are generally below 1). (b) During morning rush hour, traffic is the slowest and the city's distances expand (distortion ratios are generally above 1). The pattern repeats itself (c) during the afternoon rush hour and (d) another velocity peak in the evening.

represented in Figure 3 at the designated frames. The animation can portray the rhythms of the city continuously, displaying both global and localized expansions and contractions in the distances. (See https://vimeo.com/91325884 for a visualization.)

Trajectories Visualization

This visualization application on top on the animated cartogram displays the recent trajectories of vehicles, creating a color-temperature map for roads' velocities. For each vehicle in the visualization at 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 lets us represent the recent trajectories of every vehicle, illuminating the roads in Lisbon's map. Each trail is colored according to the average velocity during the previous 30 minutes. For this, we use a color mapping inspired by traffic signals (see Figure 4). (See https://vimeo.com/91325884for a visualization.)

This visualization lets us visually group GPS traces, which provides a glimpse of traffic volume and speed. Thicker clusters of trails indicate higher volume and help identify some of Lisbon's highways where the lower velocities observed are yellowish green during rush hours. One discrepancy that became apparent was that areas with less traffic volume but slower velocities were not visually emphasized. For example, the downtown area with its narrow streets causes traffic to slow down, but it is not as distinguishable as the highways. To attenuate such discrepancies, every GPS point of every trail is also drawn with a circle with very low opacity and the same color of the trail. This confers a proper salience to low and slow traffic areas while also emphasizing spots with stationary vehicles.

We mapped the visualization to the animated cartogram by computing the closest road to a GPS point and storing its relative distance to the closest backbone spring. Those relative positions are then used to draw each GPS trace in relation to the distorted backbone spring. Figures 4a and 4c show two snapshots of this visualization.

Figurative Visualization of Blood Vessels

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

We implemented a pulsing motion for each vessel with a rate proportional to the velocities on

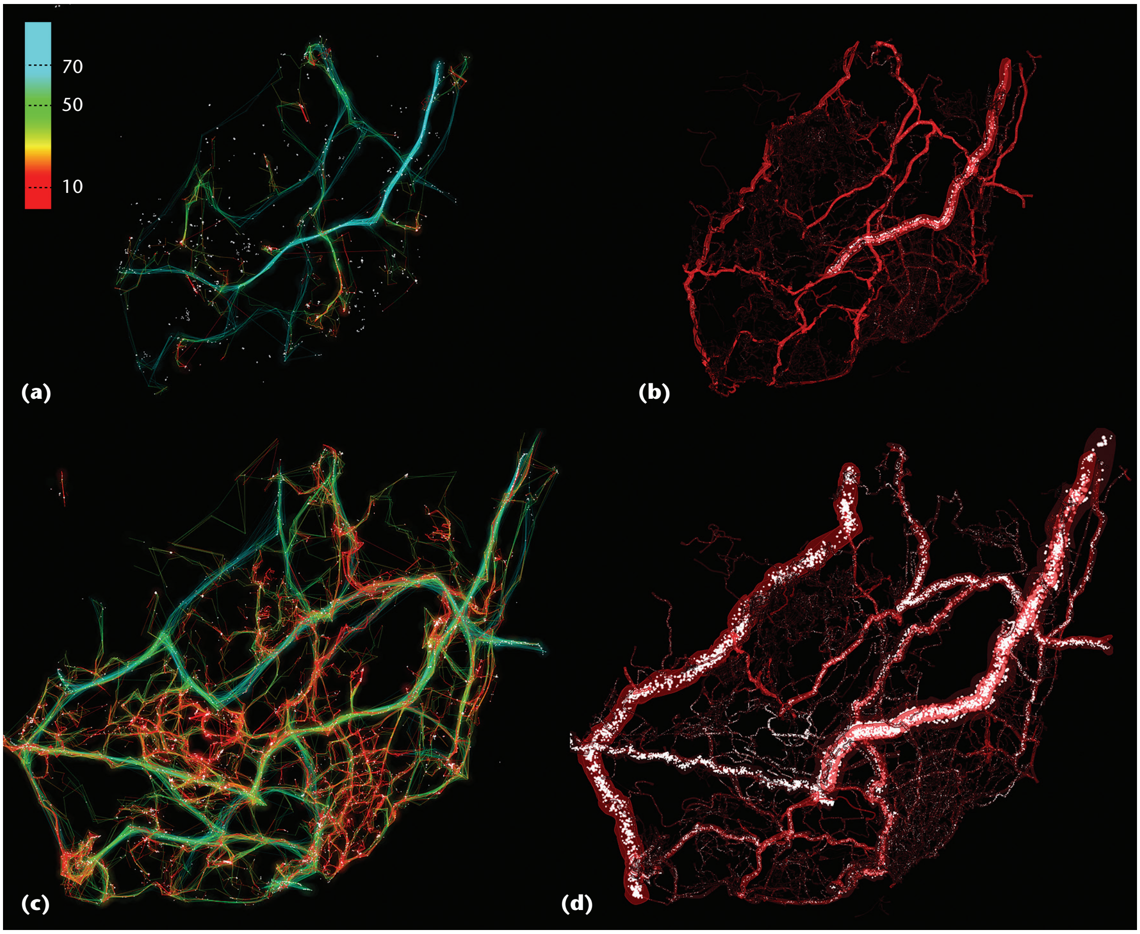

Figure 4. Snapshots of the visualization applications of our cartogram model. At time 4:31, we show the city at its most compressed state in (a) the visualization of trajectories of GPS traces and (b) a figurative visualization of blood vessels. At time 7:35, (c) the visualization of trajectories and (d) figurative visualization show the city when it is distended during the morning rush. The color scale is in kilometers per hour, and the main roads increase in thickness and in volume of cells/vehicles.

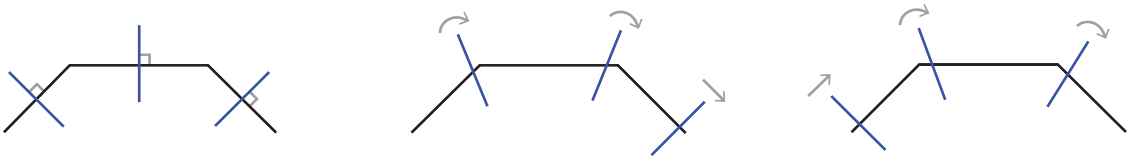

Figure 5. Interpolation of the segments used to implement a vessel's thickness and its pulsing motion. The thickness of the vessels corresponds to lengths of the segments, which are proportional to the current number of vehicles on the corresponding roads.

the corresponding road. For this, we compute a perpendicular segment for each backbone spring of a road. A perpendicular segment has its initial angle α expressed in relation to a world coordinate axis. Likewise, α_p and α_n represent the angle of the perpendicular to the previous and next backbone spring, respectively. 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. Given 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 $x = 1$, $\beta = (\alpha_n + \alpha)/2$. Figure 5 illustrates this behavior. The lengths of these segments are proportional to the current number of vehicles on the corresponding roads, altering the thickness of the vessels. The shape of each vessel is formed by uniting the segment's endpoints with Catmull-Rom splines.

Additionally, we added a "stream of cells" inside each vessel to represent the vehicle flow. The number of cells is proportional to the number of

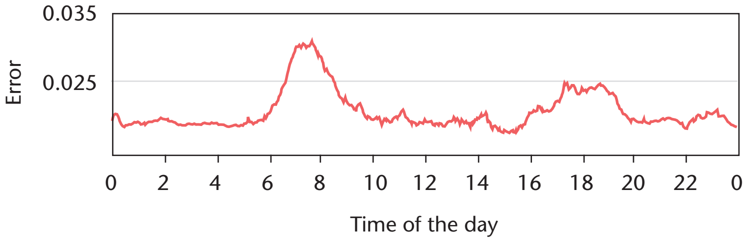

Figure 6. Representation error for each time frame. This resulted in a maximum of 0.031 and a minimum of 0.018.

vehicles, and their velocity is proportional to the average velocity on the road. This model emphasizes the roads with more traffic while also displaying their current velocity via the cells' motion and through brighter and darker reds. Figures 4b and 4d show two snapshots of this visualization.

Discussion

Generating contiguous cartograms for a given dataset involves some representation error, as we discussed earlier. We computed this error by adapting the area error function for lengths from previous research.[2] The function expresses the relative error between the actual lengths of roads and their expected lengths in our cartogram. Because each time frame represents a different set of data, the error has different results for each simulation step (see Figure 6). The error has a minimum of 0.018 at 15:00 (a period when the map has fewer tensions because 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 representation errors presented in other contiguous cartograms applications.[2,6] By testing different parameterizations of k_B, k_I, and 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.

In future work, we plan to do a comprehensive parametric study of k_B, k_I, and k_C. Our tests were run on a 2.7 GHz IntelCore i5 with 8 Gbytes of RAM, and the cartogram generation without the visualization applications was computed and rendered at a rate of 6 to 7 frames per second, which potentially deems this model suitable for real-time applications. (Other models can take several seconds or hours to generate a cartogram.[2,6])

The oscillatory motion of the road network is a natural consequence of our spring model, but it can be exacerbated by accelerated animations and high data-injection rates. Therefore, we tested the model with a lower data-injection rate (once per each 90 frames), allowing the system more iteration steps to adapt. Although the representation error decreased, the difference was not significant. This rate eliminates the oscillatory motion at the expense of creating longer animations. If we were to maintain the previous duration by compressing these frames, the animation would become nervous and unnatural. Nonetheless, this option can be used to generate static cartograms with better accuracy and observe how the cartogram model would react to real-time data injections.

The produced visualizations are able to capture geographical and temporal patterns of city traffic, showing the varying behaviors of different regions. For example, Lisbon's downtown area has slower and lower traffic levels than major highways. On a temporal level, rush hours are properly emphasized by the city's expansion, while also observing that the downtown has a later rush hour because it expands after major roads.

During the cartogram simulation, some roads exhibit sharp shape distortions. The system does this to accommodate expanded roads that are constrained by other springs, not allowing the region to resize uniformly. In this way, the sharp shape distortions become a method to graphically identify roads with slow traffic. The appearance of this and other less natural artifacts is a compromise that we cannot circumvent, regardless of the technique we choose. That is, like contiguous value-by-area cartograms, contiguous edge-based cartograms will have a shape error, some with more pleasing shape distortions than others.

Acknowledgments

Our data was provided by the MIT Portugal CityMotion project. Pedro Cruz received funding from FCT (Fundação para a Ciência e Tecnologia), under grant SFRH/BD/77133/2011.

References

1. W. Tobler, "Thirty Five Years of Computer Cartograms," *Annals Assoc. of Am. Geographers*, vol. 94, no. 1, 2004, pp. 58–73.
2. D.A. Keim, S.C. North, and C. Panse, "CartoDraw: A Fast Algorithm for Generating Contiguous Cartograms," *IEEE Trans. Visualization and Computer Graphics*, vol. 10, no. 1, 2004, pp. 95–110.
3. J. W. Clark, "Time–Distance Transformations of Transportation Networks," *Geographical Analysis*, vol. 9, no. 2, 1977, pp. 195–205.
4. D. Offenhuber, "Wegzeit: The Geometry of Relative

Distance," *ACM SIGGRAPH, Conf. Abstracts and Applications*, 2002, p. 280.
5. Y.H. Wu, "Non-connective Linear Cartograms for Mapping Traffic Conditions," *Cartographic Perspectives*, no. 65, 2010, pp. 33–50.
6. D.H. House and C.J. Kocmoud, "Continuous Cartogram Construction," *Proc. Conf. Visualization*, 1998, pp. 197–204.
7. N. Andrienko and G. Andrienko, "Visual Analytics of Movement: An Overview of Methods, Tools and Procedures," *Information Visualization*, vol. 12, no. 1, 2013, pp. 3–24.
8. N. Willems, H. Van De Wetering, and J.J. Van Wijk, "Visualization of Vessel Movements," *Computer Graphics Forum*, vol. 28, no. 3, 2009, pp. 959–966.
9. P. Cruz and P. Machado, "Visualizing the Circulatory Problems of Lisbon," *ACM SIGGRAPH 2011 Posters*, 2011, article no. 92.
10. D. Dorling, "Area Cartograms: Their Use and Creation," Dept. of Geography, Univ. of Bristol, 1996, p. 4; www.dannydorling.org/wp-content/files/dannydorling_publication_id1448.pdf.
11. V.S. Ramachandran and W. Hirstein, "The Science of Art: A Neurological Theory of Aesthetic Experience," *J. Consciousness Studies*, vol. 6, no. 7, 1999, pp. 15–51.

Pedro Cruz *is a PhD student in the Department of Informatics Engineering at the University of Coimbra. Contact him at pmcruz@dei.uc.pt.*

António Cruz *is a master's student in the Department of Informatics Engineering at the University of Coimbra. Contact him at antonioc@student.dei.uc.pt.*

Penousal Machado *is a teacher and scientific director of the Computational Design and Visualization Lab at the University of Coimbra. Contact him at machado@dei.uc.pt*

Contact department editor Mike Potel at potel@wildcrest.com.