Adaptive Composite Map Projections in D3

Sukolsak Sakshuwong
sukolsak@stanford.edu

Gabor Angeli
angeli@stanford.edu

ABSTRACT

Selecting a map projection for a visualization is a challenging problem, as different projections are better suited for different purposes, and in particular for different scales. We implemented as an extension to D3 a recent paper on the topic, which produces a composite projection between zoom levels and latitudes. The implementation ensures that a reasonable projection is chosen for any given reference frame, and that the transition between these projections is seamless. We extended the method in the paper in two key ways: (1) We interpolate paths and handle edge cases more robustly than the paper's proof-of-concept implementation, and (2) We implemented smooth animations between map locations which provide global context for the move. A demo can be found online at `http://jitouch.com/map`.

INTRODUCTION

A significant challenge in selecting an appropriate map projection, and parametrization for that projection, for visualizing geographic data. As no projection is good in every aspect, competing criteria include:

- **Equal area**: A projection should preserve the areas of a polynomial. For instance, the Mercator projection presents inaccurate areas near the poles. In contrast, projections such as the Hammer, Lambert Azimuthal, or the Albers Conic projection preserve area.
- **Conformal**: A projection should preserve local angles. Mercator is an instance of such a projection; in contrast to any of the equal-area projections, which must necessarily distort angles somewhere.
- **Equidistant**: Distances are preserved from a standard reference point. Although none of the projections in this paper have this property, an example of the projection is the azimuthal equidistant projection of the UN logo.

In addition, a projection should take into account to following criteria:

- **Surface**: The choice of projecting onto, e.g., a cylinder versus a cone is often relevant. For example, showing the entire globe on a cone or other uncommon projection may serve to confuse the user more then enlighten them.

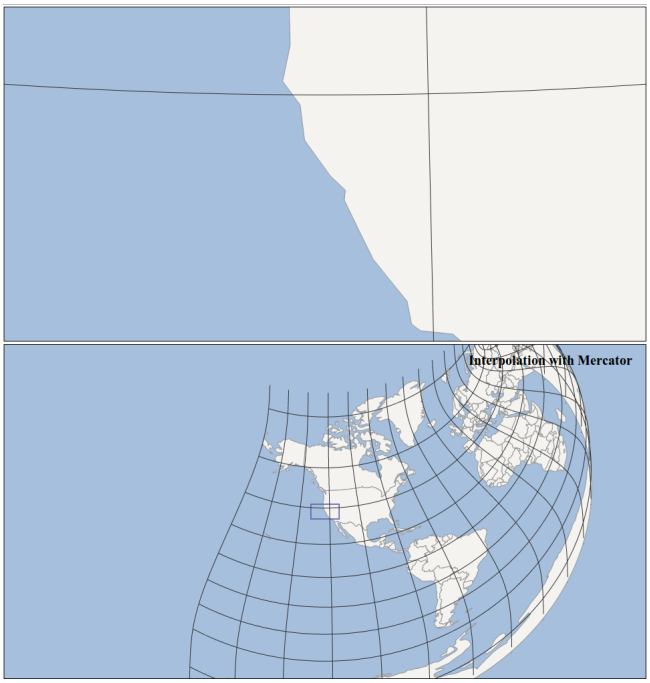

Figure 1. A screenshot of our demo. The top screen shows the view that the user of the map projection would see; the bottom screen shows the projection as it would look from a global perspective. The particular view being shown is an interpolated projection between Albers Conic and Mercator, centered on San Francisco.

- **Aspect**: Often it is beneficial to view the map from an oblique angle; for example, for viewing areas near the poles.
- **Integration with existing services**: In particular, many of the maps on the Internet use the Mercator projection, including importantly Google Maps and OpenStreetMap. It's beneficial to be able to interface with these services.

A recent paper by Bernhard Jenny [1] implemented an adaptive projection, which attempts to balance the criteria above depending on the user's absolute latitude and the scale at which they view the map. More details can be found in the Previous Work section.

We implement this paper as a proposed new projection in D3 (see Figure 1), and extend it to fit the production-ready standard and level of flexibility expected of D3. In particular, we address two key challenges:

Wrapping

The shape definitions used to render the globe come already split at the date line. However, as we're proposing a map which has far greater flexibility in its viewport; and, should

be able to handle user-defined polygons. In general these may wrap around the edge of the map, causing artifacts to appear on the projected map. We address this problem, described in more detail later, by interpolating paths and robustly detecting when such wrapping occurs. The reference implementation handles wrapping for the small-scale projections, but opts to hide invisible polygons in most cases.

Extreme Distortion

Certain projections exhibit extreme distortion on the edges of the map. Particularly when taken jointly with wrapping, the effect is that there are occasionally phenomena which cannot be solved with interpolation alone. The reference implementation does not handle this case.

In addition to these extensions, we implemented smooth animation between local points on a map. The motivation is to give users context between two local areas, allowing them to orient themselves at a global, or near global scale, before transitioning to another local viewport. In addition to being a useful feature in its own right, this showcases the capability and applicability of the projection, in that it requires accurate projections at many scales, and requires the efficiency to transition between them at a high frame rate.

PREVIOUS WORK

The bulk of our project comes from the reimplementation of a recent paper [1]; we summarize the approach here.

Much of the insight from Jenny's paper, in turn, stems from an earlier work by Snyder [2], outlining a decision tree for which projections to use depending on the scale and location of the projection center. In particular, Snyder proposed the following projections for different scales. The projections used in our approach are distinguished in *italics*

- **World** *Hammer*, Mollweide, Eckert IV or VI, McBryde, Boggs Eumorphic, Sinusoidal, or miscellaneous pseudocylindricals.
- **Hemisphere** *Lambert Azimuthal*
- **Smaller Areas** We focus on square and landscape maps, for which Snyder proposes *Lambert Cylindrical*, *Albers Conic*, and *Lambert Azimuthal*.

Jenny outlines the criteria being optimized, which we in turn inherit:

- **Distortion** An emphasis is placed on equal-area projections, particularly at large scales.
- **Graticules** Graticules are constrained by the following criteria: (1) They should be outward-pointing and not unusually-shaped; (2) The equator should be a straight line when the map is centered on it; (3) Straight longitude and latitude graticules are preferred where possible; and (4) Straight meridians should be used in polar views.
- **Aesthetics** Aesthetically, maps with elliptical borders are preferred to those with rectangular shapes.
- **Continuity** There should be continuity in the transitions between projections – this is particularly important to our application, to help with smooth animations.

The final decision tree arrived at by the paper is summarized in Figure 2, and described in more detail below. The details of the projections used are summarized in the next section.

World and Hemisphere Views

At the world and hemisphere views, the paper suggests the Hammer and Lambert Azimuthal projections. These are, in fact, two instances of the same generalized projection; interpolation is done smoothly between them by varying the hyperparameter which differentiates the projections. At extreme latitudes, a polar aspect is used. This results in the initially strange, but actually fairly useful phenomena where the map centers and distorts around a pole. For example, the relatively small distance between North America and Europe over the North pole becomes far more salient.

Middle Views

Most of the differentiation in projections come from the middle scales – on the order of continents or countries. In particular, near the equator we select the Lambert Cylindrical projection, optimizing for straight graticules. Near the equator, this transitions seamlessly from Lambert Azimuthal. In the majority of the middle latitudes, we use the Albers Conic projection. Since we are sufficiently zoomed in, we do not see the aesthetically unappealing properties of the conic projection; furthermore, the transition from Lambert Azimuthal is fairly painless. Lastly, at extreme latitudes we maintain the Lambert Azimuthal projection.

Local Views

A key practical consideration, as mentioned above, is to allow the adaptive projection to interface with other web services. Importantly, to interface with tile-based services such as Google Maps and OpenStreetMap. We would thus like to transition to a Mercator projection, which, in addition, preserves angle and shape accuracy for the small objects we are likely to see at this scale. The transition is done by interpolating between the middle scale projections and Mercator with a simple linear interpolation.

An overview of the behavior of our program, zooming into San Francisco, is given in Figure 3 We proceed to discuss the details of each projection in greater depth.

PROJECTIONS USED

Some of the projections used in the previous sections are special cases of generalized projections, or reduce to each other in limiting cases. In this section we review each of the projections, and their relations to each other. In our discussion, we will use λ to denote the longitude and ϕ to denote the latitude of the unprojected point. The projected coordinates are denoted by x and y.

Generalized Hammer

A generalized formula can be used to unify both the Hammer and the Lambert Azimuthal projections, given below:

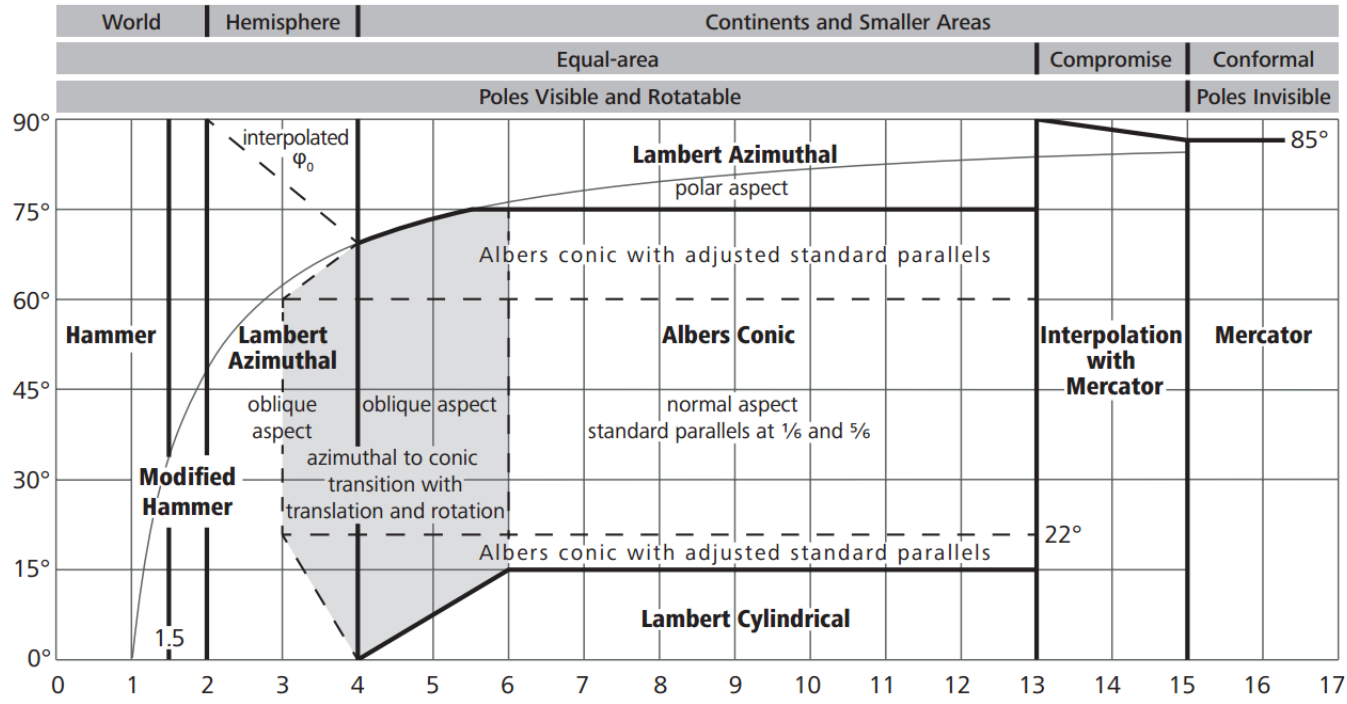

Figure 2. A summary of projections used at different scales and absolute latitudes. The x axis denotes scale; the value corresponds to inverse fraction of the map the current viewport occupies. Thus, at an x value of 5, the viewport is $\frac{1}{5}$ the size of the map. The y axis corresponds to the absolute latitude of the center of the map; 0 lies along the equator, which 90 is at the poles. This figure is copied from Bernhard Jenny's paper.

$$x = \frac{B\sqrt{2}\cos\phi\sin(\lambda/B)}{\sqrt{1+\cos\phi\cos(\lambda/B)}} \tag{1}$$

$$y = \frac{\sqrt{2}\sin\phi}{\sqrt{1+\cos\phi\cos(\lambda/B)}} \tag{2}$$

The free variable B can be used to select between the Hammer projection ($B = 2$) or the Lambert Azimuthal projection ($B = 1$). Importantly, the parameter can also be varied $1 \leq B \leq 2$ to interpolate smoothly between the two projections.

Rotation Of Origin

For these projections, it becomes important to rotate the earth such that it is centered on the center of the viewport. While the longitudinal rotation is trivial, rotating towards the poles requires some spherical geometry. Given a rotated longitude λ, a latitude ϕ, and a delta δ by which we would like to move, we compute our new longitude and latitude as:

$$\lambda' = \tan^{-1}\frac{\sin\lambda\cos\phi}{\cos\lambda\cos\phi\cos\delta - \sin\phi\sin\delta} \tag{3}$$

$$\phi' = \sin^{-1}(\cos\lambda\cos\phi\sin\delta + \sin\phi\cos\delta) \tag{4}$$

Note that the argument to $\sin^{-1}$ in ϕ' is constrained to be between -1 and 1 in practice. Figure 4 shows the Lambert Azimuthal projection rotated to center on Europe. Again, we see that the distance between Asia and North America is rather small over the North pole – an insight which is almost entirely opaque in the cylindrical and even conic projections.

Albers Conic

The Albers Conic projection, given an origin point (ϕ_0, λ_0) and parametrized by standard parallels ϕ_l and ϕ_u, is defined by the formulæ below [3]:

$$x = \rho\sin(n\,(\lambda - \lambda_0)) \tag{5}$$

$$y = \rho_0 - \rho\cos(n\,(\lambda - \lambda_0)) \tag{6}$$

Where:

$$n = \frac{1}{2}(\sin\phi_l + \phi_u)$$

$$C = \cos^2\phi_l + 2n\sin\phi_l$$

$$\rho = \frac{\sqrt{C - 2n\sin\phi}}{n}$$

$$\rho_0 = \frac{\sqrt{C - 2n\sin\phi_0}}{n}$$

Note that the Albers Conic projection can be coaxed into the Lambert Azimuthal and Lambert Cylindrical projections by changing the location of the standard parallels.

Lambert Cylindrical

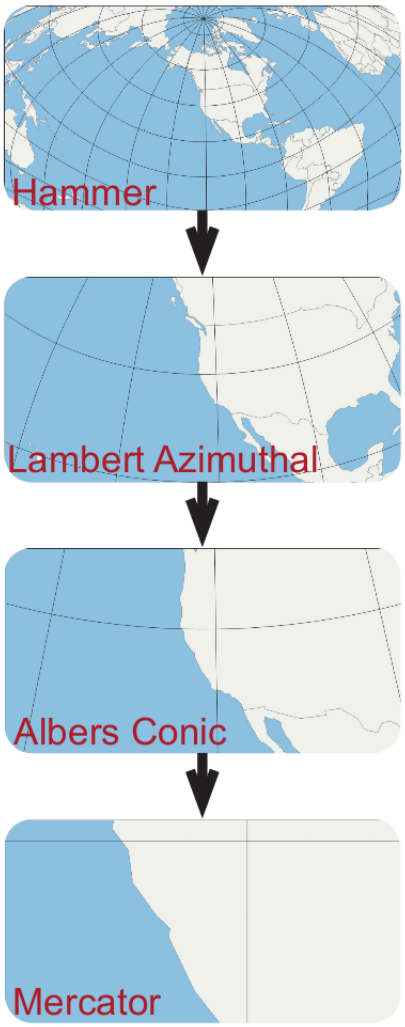

Figure 3. An example of the algorithm, zooming in on San Francisco. Note that no sharp jumps are noticeable (beyond those caused by zooming). Nonetheless, note that the graticules are straightening as we zoom in, and by the end we reach the Mercator projection, allowing us to potentially overlay tiles.

Perhaps the simplest projection, the Lambert Cylindrical projection is defined as:

$$x = \lambda - \lambda_0 \tag{7}$$
$$y = \sin\phi \tag{8}$$

IMPLEMENTATION

The code was designed to behave as much as a plug-and-play replacement for other projections as possible. In particular, the public-facing interface for the composite projection is very similar to that of other projections, and no functions were introduced (beyond the animate function) that did not exist in at least some other projection. The top-level interface is:

`d3.geo.composite([viewport])` Create a new composite projection. If a viewport is specified, the composite projection will calibrate itself according to that viewport; otherwise, it will create a default projection of width and height 500px.

`composite(location)` Project the (λ, ϕ) point defined by `location` into (x, y) coordinates.

`invert(point)` Invert the projection from (x, y) coordinates to longitude and latitude.

`origin([location])` Set the origin of the projection to the given point, or return the origin of the projection. This is one of the key functions which will change the projection.

`scale([factor])` Set the scale of the projection, or get the scale. This is the other key function which will change the projection.

`interpolate(origin, destination, λ)` Interpolate between an origin and a destination position, backing off to a more holistic view as $\lambda \to \frac{1}{2}$. This is the key function used to animate transitions.

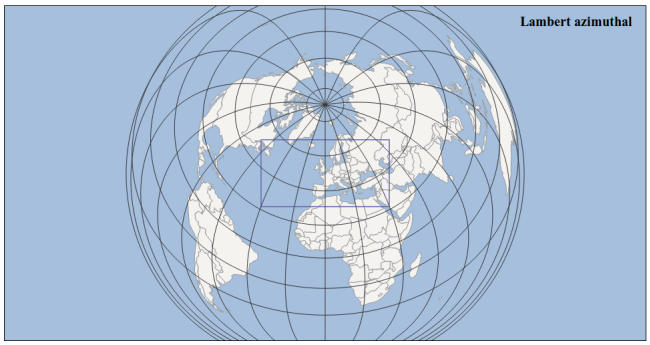

Figure 4. Lambert Azimuthal projection rotated to be centered over Europe.

Furthermore, the changes made to `d3/geo/path.js` are entirely backwards-compatible with current projections. In fact, it fixes a bug in the live version of D3 where viewing the world countries using the Albers Conic projection over the United States will cause Antarctica to cross over the viewport. Beyond cases like these, however, the implementation should neither be incorrect or slower for maps which do not need interpolation (e.g., the Mercator projection).

EXTENSIONS

In addition to implementing [1] in D3, we create a number of extensions to the approach, largely focused on refining it to be appropriate for a production-ready environment. These improvements are described in the following sections:

Removing Visual Artifacts

Several techniques are employed to minimize visual artifacts that appear from using D3. These techniques are chosen to preserve the fidelity of the data without compromising the rendering speed too much.

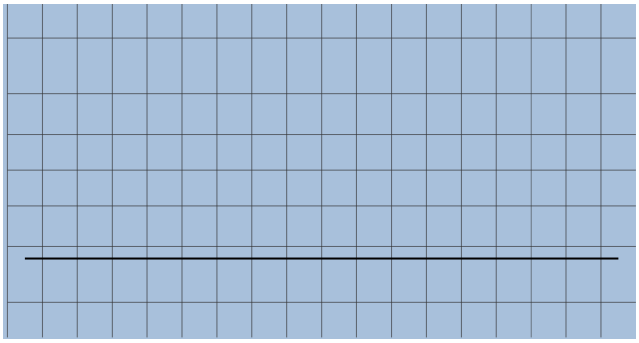

Figure 5. A straight line in the Mercator projection.

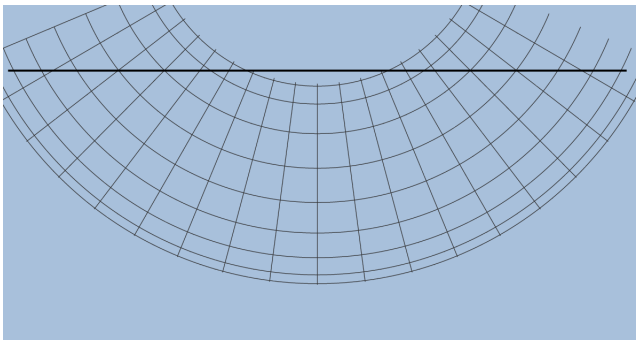

Figure 6. The same line in the Albers conic projection drawn without interpolation.

Coarse Granularity

In D3, after the map is projected onto a plane, lines are drawn from point to point using a straight line in the Cartesian coordinate system. To draw more accurately, we need to draw along the great-circle arc between points, but this is computationally expensive as it involves the use of trigonometric functions and many lines drawn. Drawing a straight line in the Cartesian coordinate system is visually acceptable as long as points are not too far apart from each other. However, this becomes a problem when points are too far apart. For example, in Figure 5, a straight line is drawn in the Mercator projection between two endpoints. When the map transforms to the Albers conic projection, the same line is still drawn between the two endpoints, causing a visual artifact.

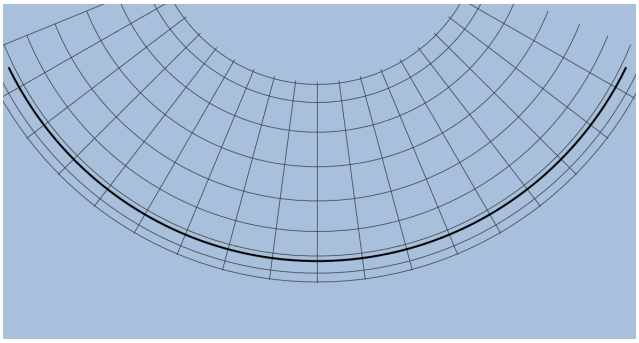

Figure 7. The same line in the Albers conic projection drawn with interpolation.

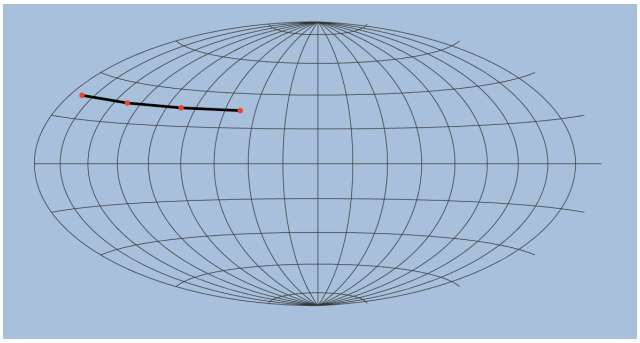

Figure 8. A line formed by four points in the Hammer projection.

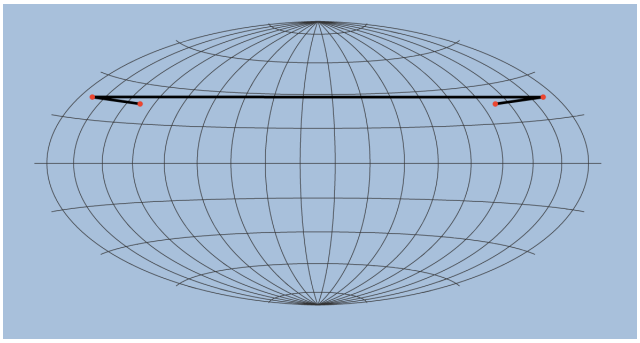

Figure 9. As we rotate the map, two points are moved to the right. The line between the second point and the third point is incorrectly drawn.

We solve this problem by interpolating lines, attempting to use as few additional points as possible. To interpolate a line, we find the middle point between the two original endpoints using the average latitude and longitude, project the middle point onto a plane, and then calculate the distance between the projected middle point and each of the projected endpoints. If the distance is within a certain threshold, we draw that part. If not, we recursively divide the line until the length of each part is within the threshold.

Discontinuity

Map projection transformations are not continuous functions. Therefore, it is possible that a straight line, after being projected, might be cut into multiple parts and each part appear at different sides. For example, in Figure 8, a line is formed by four points. After we rotate the map to the left, two points are moved to the right. But the line between the second point and the third point is still drawn, as in Figure 9. This can be easily fixed in the interpolation by using the fact that if the line is continuous, the length of each part after being cut will converge to zero. But if the line is not continuous, one of the parts will not converge to zero. Therefore, we limit depth of the recursion and cut the line if, after reaching the limit, the length of the line is still greater than the threshold.

We summarize our algorithm for interpolating points in Algorithm 1

Crossing the Antimeridian

The demo that we created allows the user to draw a great-circle arc between any two points. However, when the arc crosses the antimeridian (the 180th meridian), a line will

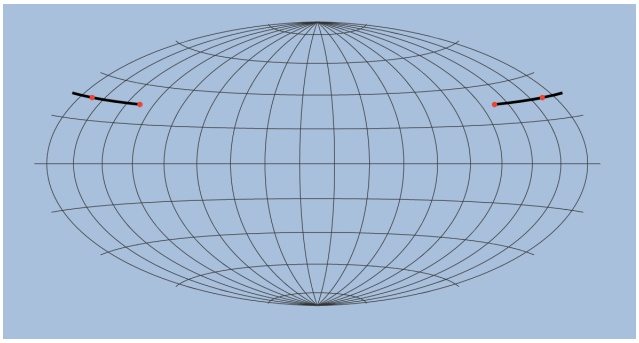

Figure 10. Using interpolation, the line between the second point and the third point is correctly drawn.

Algorithm 1 Interpolation

procedure INTERPOLATE(p_0, p_1, ρ_0, ρ_1) ▷ Projected p; unprojected ρ
  if $||p_0 - p_1||^2 < 25$ **then**
    return $[p_0, p_1]$
  end if
  $m \leftarrow \text{mean}(\rho_0, \rho_1)$
  $pm \leftarrow$ projection(midpoint)
  $d_{a2m} \leftarrow ||p_0 - pm||^2$
  $d_{m2b} \leftarrow ||p_1 - pm||^2$
  if $d_{m2b} > d_{a2m}^2$ **then**
    return [INTERPOLATE$(p_0, pm, \rho_0, m), p_1$]
  else
    $a \leftarrow$ INTERPOLATE(p_0, pm, ρ_0, m)
    $b \leftarrow$ INTERPOLATE(pm, p_1, m, ρ_1)
    return $[a, b]$
  end if
end procedure

be drawn through a longer path that does not cross the antimeridian, causing a visual artifact. For example, the great-circle arc between $(170W, 0)$ and $(170E, 0)$ is drawn through $(170W, 0), (160W, 0), \ldots,$ and $(170E, 0)$ instead of being drawn through $(170W, 0)$, $(180W, 0)$, and $(170E, 0)$.

This problem cannot be solved by using interpolation alone because when we find the middle point between two points, we use the average latitude and the longitude as the middle point, which is not the actual middle point of the great-circle arc. For example, the middle point that we calculate between $(170W, 0)$ and $(170E, 0)$ is $((-170 + 170)/2, (0 + 0)/2) = (0, 0)$, which is not the actual middle point $(180W, 0)$. Hence, the distance of each part converges to zero as if the transformed line was continuous.

One way to solve this problem is to interpolate lines using the actual middle point of the great-circle arc of two endpoints, but this, again, is computationally expensive as we need to use trigonometric functions and it is a function that is called extremely frequently. Another way to solve this problem is to cut the data along the antimeridian. This ensures that when we find the middle point between any two points using the average latitude and longitude, the calculated point will not be too far from the actual middle point. For example, we

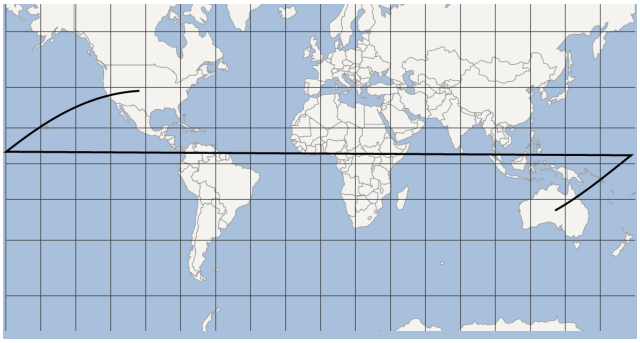

Figure 11. The great-circle arc between the United States and Australia. The arc crosses the antimeridian and is drawn with incorrect interpolation.

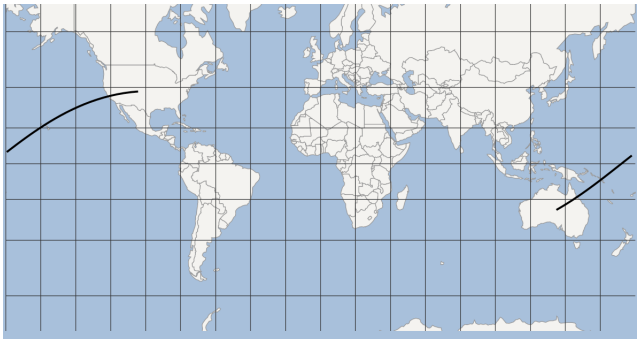

Figure 12. The great-circle arc between the United States and Australia drawn with correct interpolation.

cut the line $(170W, 0)$–$(170E, 0)$ into $(170W, 0)$–$(180W, 0)$ and $(180E, 0)$–$(170E, 0)$. Since $(180W, 0)$ and $(180E, 0)$ are the same point, the line will appear continuous when the user rotates the map.

Extreme Distortion

A nuanced problem arises in the Lambert Azimuthal projection, and the Hammer variants which approach it. When a polygon of nonzero width is projected onto the edge of the map, it begins to distort into a crescent shape, eventually becoming a half-circle. In itself, this can be faithfully rendered by interpolating the polygon. However, in certain cases the splitting line goes down the center of the polygon, and we get as a result two opposing half-circles, one on each side of the map. The effect is that our interpolated path becomes a full circle, without large jumps or visible discontinuities, as the location of the jump is at the pole. Since we naively fill any polygon we render, the visual result is a colored circle overlaying the rest of the map. This is relevant as Antarctica distorts into the bottom of the map, or as continents distort off of the sides (e.g., South America or Africa).

We address this problem by querying the projection regarding whether a proposed path will undergo extreme distortion. If every vertex in the path is beyond a threshold, the path is not rendered. This causes polygons near the edge of the map to disappear after a certain point; however, by then they are thin enough that the practical effect is minimal.

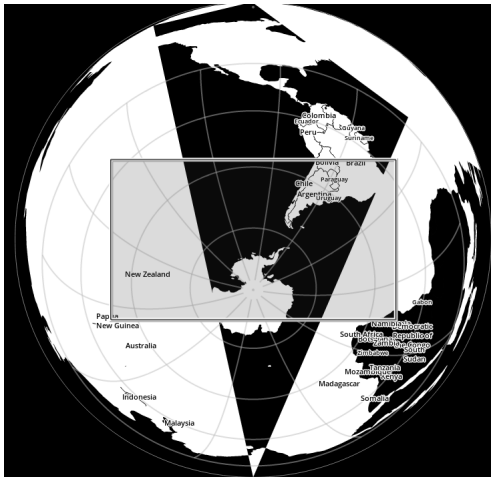

Figure 13. A snapshot of the reference implementation from Jenny, found at `http://cartography.oregonstate.edu/demos/CompositeMapProjection/`. The coordinates are $(100W, 65S)$ at a zoom of 2.1. Note that Russia has distorted around the map, causing polygons to close themselves in front of the globe.

Importantly, we note that the proof-of-concept implementation does not handle this case, despite having the benefit of hiding polygons as they exit the viewport. See Figure 13, or center the demo at zoom level 2.1 and coordinates $(100W, 65S)$.

Animation

Another extension that we added to the map is animation. The user can drag the mouse cursor on the global map to create a great-circle arc. Then the user can click the Animate button to move the map along the arc. To implement this, we first convert the Cartesian coordinates of the mouse cursor into the spherical coordinates. Then we convert the spherical coordinates into normal vectors, and use spherical linear interpolation (Slerp) to find points on the arc with an equal distance between points. We also make use of an ease function to make the animation appear smooth. This animation is useful for visualizing, for example, the path an airplane should fly to get to the destination in the shortest time.

Furthermore, when animating, we zoom out for a wide view of the globe first to give the user the context of where he or she is. As we get close to the destination, we zoom in again. This has the advantage that a visualization developer can move to significantly different areas of the globe without worrying that the user would get confused. For example, we can use animation to visualize a typhoon path in fine-grained detail and coherently move the user to different parts of the globe to visualize another typhoon.

DISCUSSION AND FUTURE WORK

The largest bit of work which we intend to undertake is submitting a pull request to D3, in hopes that the extensions will actually be incorporated into the distribution. More conceptually, however, a number of improvements could be made.

Largest among them: the current D3 framework views projections as a mapping from points to points. In this framework, drawing a closed polygon will always draw the inside of the region. This results in the error discussed above, wherein an extremely distorted polygon will cover the entire globe. The principled solution to this, as is being implemented in the most recent version of D3, is to specify a polarity for the polygon and shade only the region inside.

Relatedly, handling wrapping of polygons would be more elegantly done if the polygons are treated as atomic units, rather than a collection of points. This is also, we hear, in the works for newer D3 versions.

More on the aesthetics side, integration with OpenStreetMap would make for a compelling demo. The challenge there is mapping accurately from their notion of scale to ours, and smoothly rescaling tiles as we change zoom levels, as their resolution is more fine-grained than the resolution we allow.

CONCLUSION

We implemented a recent paper on adaptive composite map projections, and integrated the technique into D3. Furthermore, we have made a number of improvements in order to robustly handle data viewed at non-standard rotations and scales. Lastly, we have implemented a technique for animating transitions between local views which gives the user global context in a visually appealing and useful way. A demo to showcase our approach is online at `http://jitouch.com/map`.

REFERENCES

1. Jenny, B. Adaptive composite map projections. *Visualization and Computer Graphics, IEEE Transactions on 18*, 12 (2012), 2575–2582.
2. Snyder, J. *Map projections–A working manual*. No. 1395. USGPO, 1987.
3. Weisstein, E. W. Albers equal-area conic projection. Last visited on 12/12/2012.