Dynamic geometry (Spline json) using virtual spline (Spline pools) for making Maps.

Abstract:

In maps, we always have geometries with data precision and limitations, now we need something different which should provide the ideal maps. With real time data precision, it's tough to provide large geometries in rest calls, can we create a system where these geometries are created at run time, with best precision? This complete idea tries to come up with the unique approach to solve these problems. We propose a technique which takes splines attributes as the input and create geometry(roads) on fly over the maps. Depending upon the smoothness and the precession the roads are carved. Each spline will be given the confidence score, so that the values can be improved. There is further possibility of the duplicate spline definition as the geometries are repeated (mostly in roundabouts at different functional classes and the highways which generally follow the same law) and hence the spline definition can be taken from pools of splines. For junctions, we may add further attribute in the splines where the tolerance of junction is subdivided with multiple spline definitions.

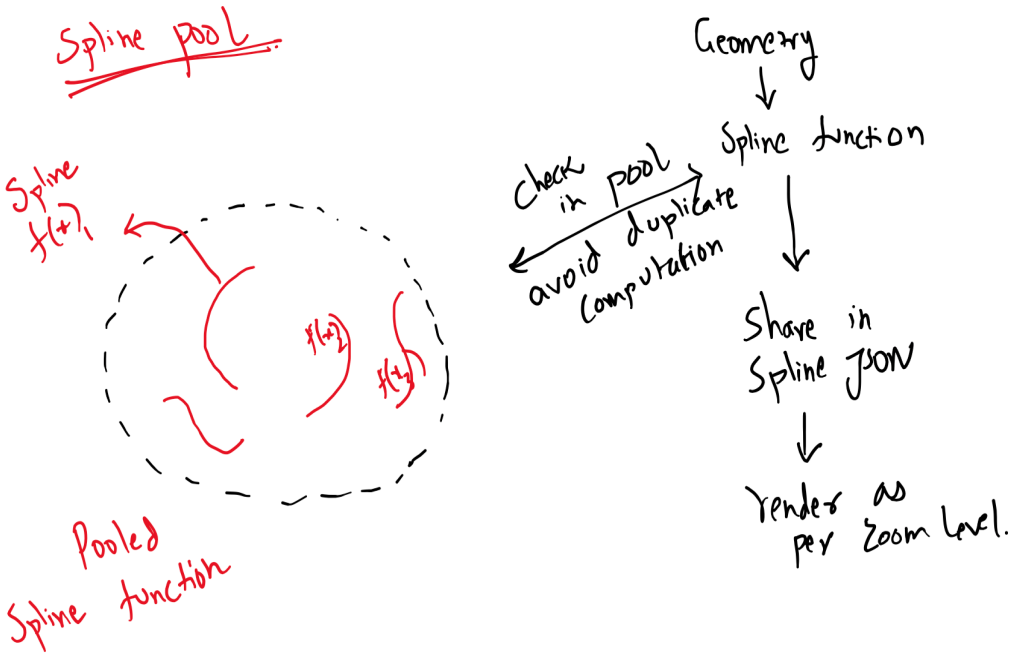

Fig 1: Spline Pool (ALGO)

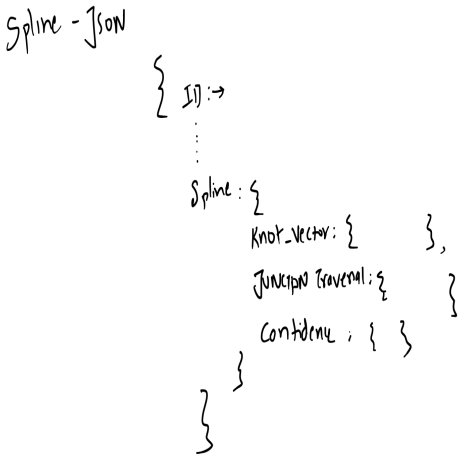

Fig 2: Spline Json(format)

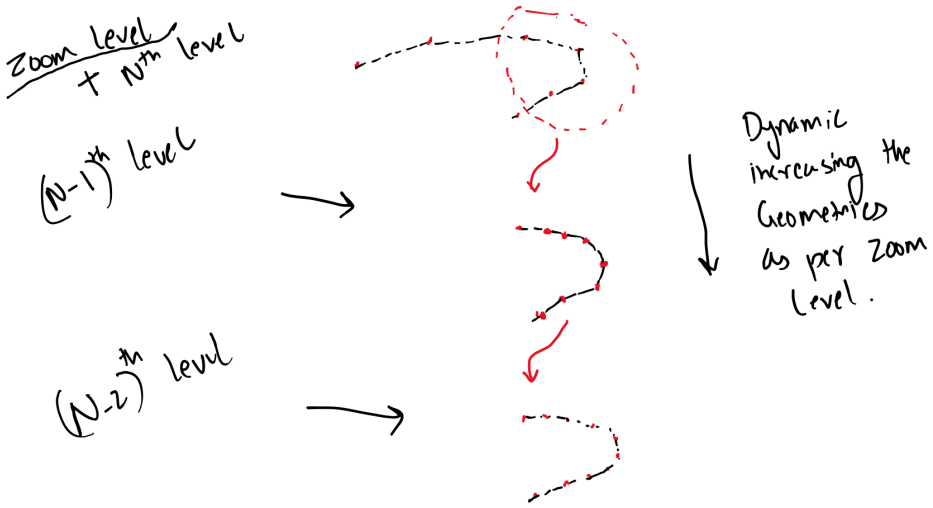

Fig 3: runtime creation of geometry as per zoom level.

Detail Description:

Currently representation of road is in Geojson format and hence have fixed number of Geometries. Perfect/Smooth Road representation is directly proportional to number of geometries (fig 3) and hence the storage space increase and so do the network load. Representation of huge geometry is also challenging over the browser. This novel technique provides the functions (Splines) which can generate the geometries in fly. **The idea is to store the functions and not the geometries.**

As Describe in the fig1, we will be creating the spline functions from the geometries (local system), for the duplicate shifted geometry fetch the function from the spline pools so to avoid the recalculation computation. Representation will be in the form of **Spline json** fig2**,** it will include the traversal differences. From fig3, depending upon the smoothness of the road as per the zoom level the geometries are created from the spline functions. **In theory we can create infinite geometries, and hence ideal Maps.**

Algorithms (bird eyes):

Backend side:

1. With the real data the splines are created (including the junction definitions)
2. The spline id is stored as per region which can be extracted as per BB-BOX or any other geo-spatial queries
3. There is another column in the database which should store the repetitive spline distance

Frontend:

1. Like the geo-json data, the spline json data is passed with the all the definition
2. At frontend, it should be able to derive the geometries from it on runtime.

Benefit and Improvements

1. At runtime, creation of geometries
2. No extra storage
3. High precision data
4. No kinkiness, more smoothness
5. Pooling is possible (repetition can be avoided)
6. We will be having best MAPS.