Interpolation - Geocode Software Pelias

Numerical Computation Project
Nicholas Pritchyk

Geocode Pelias Interpolation Service

The Pelias software is a geocoder that uses open data to turn addresses and place names into coordinates and the coordinates into addresses and places.

Open source and data software that provides ability to return street address interpolation queries.

Projects goal looks to fill the gaps in the geographic data to estimate where the house numbers are on the road.

Written in javascript for node.js, database engine of sqlite3.

Database Setup

field	type	description
rowid	`INTEGER`	the primary key column for this table, not generally referenced
id	`INTEGER`	a foreign key to the corresponding row in the `polyline` table
name	`TEXT`	the name of the street

`address` table

field	type	description
rowid	`INTEGER`	the primary key column for this table, not generally referenced
id	`INTEGER`	a foreign key to the corresponding row in the `polyline` table
source	`TEXT`	the datasource for this point
source_id	`TEXT`	a unique identifier for this point from the original datasource (if applicable)
housenumber	`REAL`	an exact or interpolated housenumber for this point
lat	`REAL`	the latitude of this address point
lon	`REAL`	the longitude of this address point
parity	`TEXT`	Either `L` or `R` if this is an exact address, denoting the side of the street the address is on
proj_lat	`REAL`	The projected point along the related street corresponding to this address
proj_lon	`REAL`	The projected point along the related street corresponding to this address

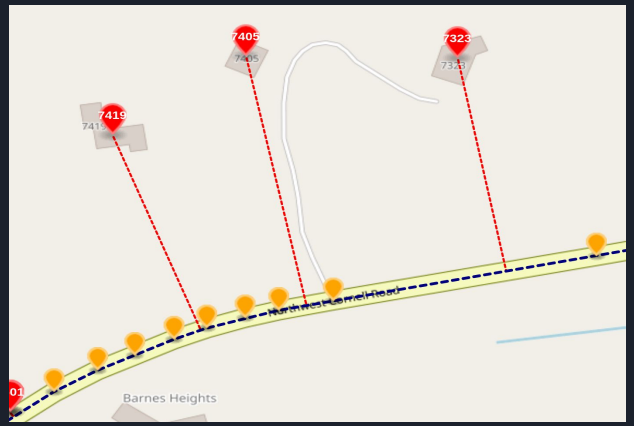
`polyline` table

field	type	description
id	`INTEGER`	the primary key for a given street
line	`TEXT`	polyline encoded linestring

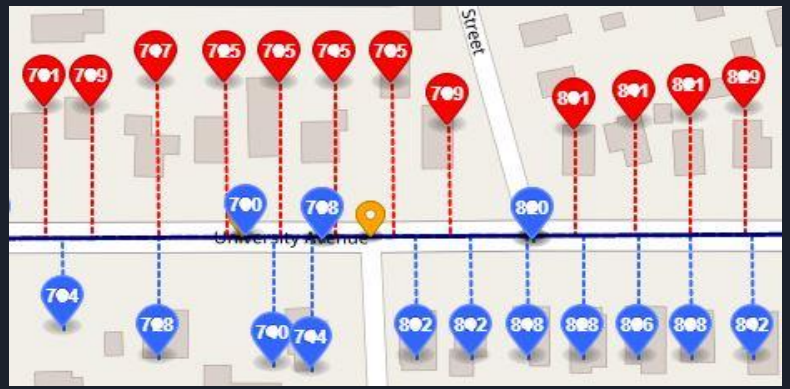
Software Real World Example

The red dots indicate the address point, the red line represents the address points to their projection on the street, orange points represent the vertex points.

Exact addresses use the proj_lat and proj_lon fields in the address table and represent the orthogonal projection of the address along the relevant street.



The interpolation engine plots addresses based on what side of the street they are



Interpolation

The interpolation of building addresses is based on numeric ranges of the house numbers.

Natural problems of interpolation include geometry of the area, road network or holes between building ranges.

Interpolating addresses:

```
// sort by miniumum housenumber difference from target housenumber ASC
segments.sort( function( a, b ){
  return Math.abs( a.diff.before + a.diff.after ) - Math.abs( b.diff.before + b.diff.after );
});

// select before/after values to use for the interpolation
var before = segments[0].before;
var after = segments[0].after;

// compute interpolated address
var A = { lat: project.toRad( before.proj_lat ), lon: project.toRad( before.proj_lon ) };
var B = { lat: project.toRad( after.proj_lat ), lon: project.toRad( after.proj_lon ) };
var distance = geodesic.distance( A, B );
```

```
  // if distance = 0 then we can simply use either A or B (they are the same lat/lon)
  // else we interpolate between the two positions
  var interpolatedPoint = A;
  if( distance > 0 ){
    var ratio = ((normalized.number - before.housenumber) / (after.housenumber - before.housenumber));
    interpolatedPoint = geodesic.interpolate( distance, ratio, A, B );
  }

  // return interpolated address
  return cb( null, {
    type: 'interpolated',
    source: 'mixed',
    number: '' + Math.floor( normalized.number ),
    lat: parseFloat( project.toDeg( interpolatedPoint.lat ).toFixed(7) ),
    lon: parseFloat( project.toDeg( interpolatedPoint.lon ).toFixed(7) )
  });
 } catch (err) {
  // an error occurred
  return cb(err, null);
 }
};
```

Interpolation

Interpolate function to interpolate a house number (fractional) based on a certain vertex distance.

```
// cycle through calculated addrPoints and interpolate a fractional housenumber
// value which would sit at this vertexDistance.
for( var x=0; x<addrPoints.length-1; x++ ){

  var thisAddr = addrPoints[x],
      nextAddr = addrPoints[x+1];

  // the vertex vertexDistance is less that the lowest housenumber
  // @extrapolation
  if( vertexDistance < thisAddr.dist ){ return null; }

  // vertex vertexDistance is between two house number vertexDistance
  if( nextAddr.dist > vertexDistance ){
    var ratio = ((vertexDistance - thisAddr.dist) / (nextAddr.dist - thisAddr.dis

    // invert ratio if the street was drawn from high house number to low
    if( thisAddr.housenumber > nextAddr.housenumber ){ ratio = 1 - ratio; }

    if( ratio >= 1 || ratio <= 0 ){ break; } // will result in a duplicate value
    var minHouseNumber = Math.min( thisAddr.housenumber, nextAddr.housenumber );
    var maxHouseNumber = Math.max( thisAddr.housenumber, nextAddr.housenumber );

    // house numbers are only a single number apart
    // see: https://github.com/pelias/interpolation/issues/6
    if( maxHouseNumber <= ( minHouseNumber + 1 )){ return null; }

    // return fractional housenumber
    return minHouseNumber + (( maxHouseNumber - minHouseNumber ) * ratio);
  }
```

Question

The softwares page does not expel any accuracy estimates or concerns about how close the interpolated addresses are to the actual. How can Pelias (creators) assess the validity or accuracy of the street address interpolation output?

Experiment

Due to there being no current public data from Pelias on validity of interpolated street and house addresses an interesting experiment would involve using accurate and credited data on local street and house addresses and try to find holes or inconsistencies in the interpolated buildings that Pelias has attempted to fill.