3D Road Network Building & Visualization
Final Report

Course: CS 226 Big Data Management
Members (Group 9): Kapish Garg, Gyan Prakash, Tina Mirzaei, Zhuocheng Shang, Xinlong Yi

Abstract
There are a lot of applications for 3D road networks and there have been few tools which are able to convert the 2D data to 3D data. 2D road data is available in abundance and we require a technique to modulate the data into 3D by adding new parameters. In this project, we will merge two datasets: DEM data and OpenStreetMap data. Our team apply SparkRDD to complete data integration and apply Google Earth engine to visualize the output KML file.

1 Introduction & Problem Statement
There is a vast number of 2D road data which we implement in our everyday life for multiple reasons. This data is useful for applications such as navigating us from one place to our destination, increasing the efficiency of using and maintaining city infrastructure and also monitoring the environmental and social conditions of urban life. Therefore analysis and visualizing this 2D data is of great importance.
There are so many records and documentation of 2D openstreet map data that have been gathered during the past years, which can be used as a valuable source of data for the data analytics tasks. Applying analytical task to these data bring us valuable information that can be used to increase the efficiency of road networks.
There are only a few tools and methods that have been discovered to convert the data to 3D models; in our project, we specifically demonstrate an effective way to convert 2D road map into 3D network data which can be visualized by 3D road network visualization software to provide a better understanding of road networks.

In this project, we specifically work with the 2D data of Riverside as an example.

The project is basically divided into three parts:
- Data pre-processing
- Data integration
- Data visualization and evaluation

Data preprocessing includes analysing the format of DEM data (stored as a tiff file) and OpenStreetMap data. To do data integration, we process SparkRDD to generate <key, value>

pairs and output longitude, latitude, altitude for each road. Finally, we use Google Earth to visualize the data which stored as a KML file.

2 Related Work

Many researches have dug into this topic. In this section, we are going to summarize the literature survey we done for this project.

Data Integration

The core data of a geographic information system (GIS) is currently two dimensional. The 2D dataset for road network can be gathered from OpenStreetMap (OSM). And then we are fetching another dataset which contains the elevation of the various object points on the map from Digital Elevation Model (DEM). A DEM is a bare-earth raster grid referenced to a vertical datum. When non-ground points such as bridges and roads are filtered, we get a smooth digital elevation model. The built structures (like power lines, buildings and towers) and natural ones (trees and other types of vegetation) aren't included in a DEM [5]. The DEM data in this project is processed by using the GeoTrellis package to project X, Y to corresponding Z value through a matrix.

Visualization

There are many different types of big data visualization tools and frameworks. In this project, we focus on the tools and frameworks that can visualize the spatial data and 3D data.

One of the techniques used by Zheng and team for visualization involves dividing the task into three parts: 3D centerlines, 3D road surface, 3D road piles. Combination of all the inputs and GIS software ArcMap is used to visualize the model [1].

GeoSparkViz is a visualization framework based on GeoSpark. GeoSparkViz encapsulates the main steps of the geospatial map visualization process, e.g., rasterize spatial objects, aggregate pixels into a set of massively parallelized RDD transformations in Apache Spark. GeoSparkViz also proposes a map tile-aware data partitioning method that achieves load balancing for the map visualization workloads among all nodes in the cluster [9]. Hadoopviz, a MapReduce based framework designed for extensible visualization of spatial data , is also a useful tool. Hadoopviz supports single-level and multi-level visualization. A single level image is an image of a fixed resolution. A multi-level image is composed of many small image tiles generated for different regions at different zoom levels [10]. Da Vinci is a project aimed at providing a more flexible and efficient multi-level visualization of large spatial data. It supports visualization by Hadoop Viz and AID [11, 12].

Considering all the previous works, in this project, we will apply Google earth to complete visualization of the result.

3 Analytical Framework

In this section we will describe how we did the data analysis in our research. This section has three parts itself:

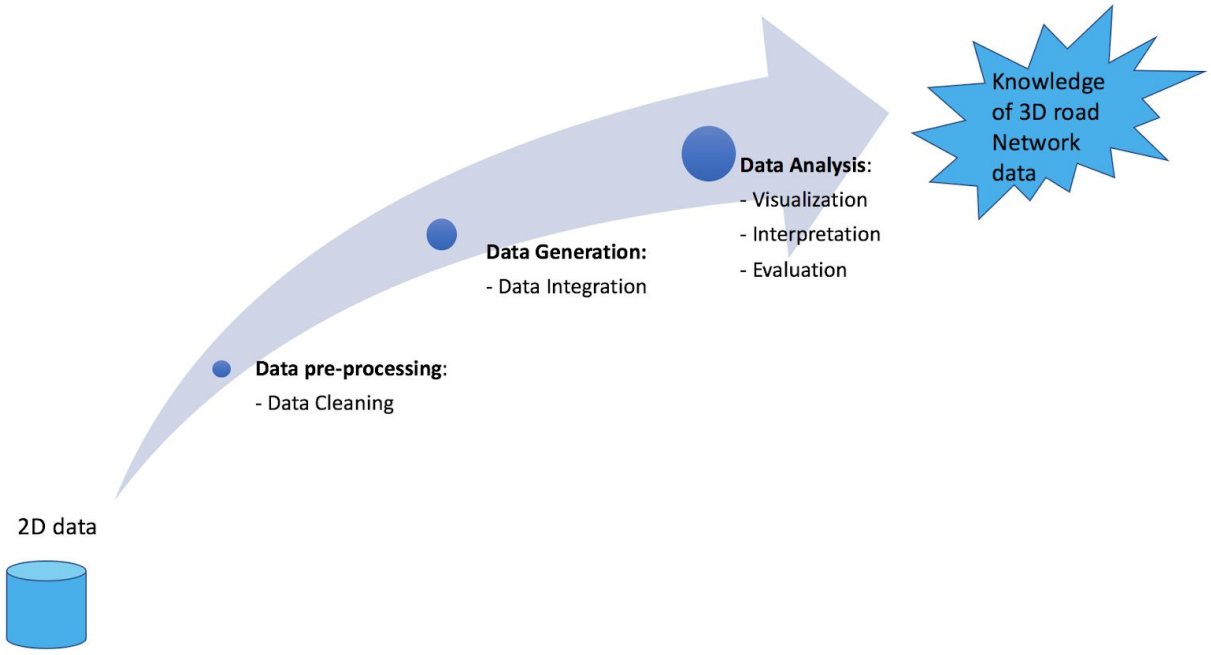

Figure 1.

3.1. Data Pre-processing

In this section, we are going to introduce the two datasets applied in the project. Despite describing how the dataset is collected for this project, we also explain the format of these dataset. Further, this section also presents how we clean the dataset and integrate the data.

3.1.1 Data description and preprocess

In this project, we generate two datasets. One of the datasets we apply is Digital Elevation Model (DEM) data. In our research, we use the package SinglebandGeoTiff designed in Geotrellis to extract the exact DEM data value from TIFF file. The Geotrellis is a geographic data processing engine for high performance applications [14]. Generally, the actual altitude value in DEM data is stored as a matrix, and each index in the matrix responds to a (longitude, latitude) pair. Firstly, it will gather the information from header which is located at the beginning of a tiff file. The header stores general information about the data matrix.

According to the header, XDIM and YDIM represent the size of geographic units by one pixel on x and y dimension [15]. SinglebandGeoTiff is designed to project each index in the matrix to the corresponding Longitude and Latitude value based on XDIM an YDIM. Therefore, we can get the actual three numeric values (longitude,latitude, altitude) for each index belongs to the matrix in our output file (stored as raster information).

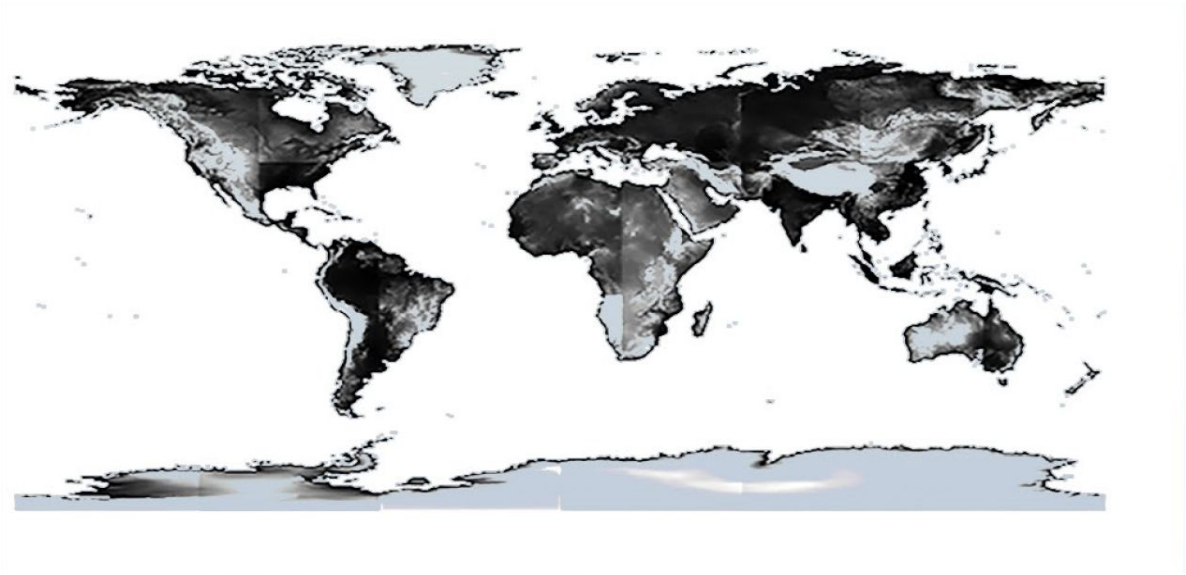

Figure 2. An example of DEM data

OpenStreetMap data is the other dataset collected in this project which includes following attributes and features that can be extracted:

- Road id : Every road has its unique id.
- Type: Linestring / Polygon
- Coordinates: An array of points consists in the road and each point in the array contains two values: longitude and latitude.

Figure 3. The difference between polygon and linestring

Linestring and polygon could be treated as the same type despise polygon has the same start and end point while consisting one road.
For handling data cleaning, we filter out the points with value -9999 or NULL which represents the Ocean area.

3.1.2 Data integration

In this project, we process data integration in Spark RDD.

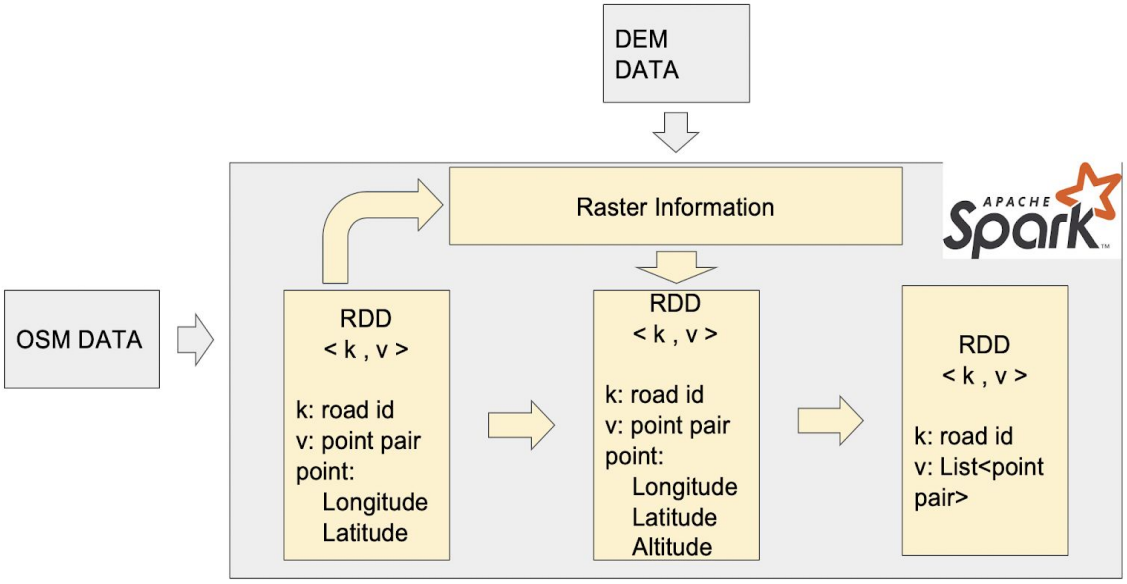

Figure 4. Spark RDD input and output

The first input dataset is OpenStreetMap data. Input format of this dataset is two strings: <String, String(Coordinates)>.

The first “string” is the Road ID, the second “string” stores the points array in dataset.

$< ID,\ "Point_1, Point_2, ..., Point_n" >$

Each point is separated by commas, and each contains two values: longitude and latitude and separated by one space.

$< ID_i,\ "Longitude_1\ Latitude_1, Longitude_2\ Latitude_2, ..., Longitude_n\ Latitude_n" >$

We applying SparkRDD to split each point pair from one whole road and map to the format of <String, Tuple2<Double, Double>>.

The following example shows how to map the point array to separate points consists on one road. If the road has a unique ID and contains n points in the list:

$< ID_1, < Longitude_1,\ Latitude_1 >>$

$< ID_1, < Longitude_2,\ Latitude_2 >>$

……

$< ID_1, < Longitude_n,\ Latitude_n >>$

The input format of DEM dataset is a raster information which was described in section 2.1.1.

Using the RDD pairs we generated, we could find corresponding points in the DEM raster information. The output of this combination should be in this format: <String, tuple3<Double,Double, Double>>.

$< ID_i,\ < Longitude_i\ Latitude_i\ Altitude_i >>$

Finally, we invoke SparkRDD to map points belong to the same Road ID. The final output is also <String, String>.

$< ID_i, \text{"}Longitude_1\ Latitude_1\ Altitude_1, Longitude_2\ Latitude_2\ Altitude_2, ..., Longitude_n\ Latitude_n\ Altitude_n\text{"} >$

3. 2 Visualization

The Final spatial data at this point has all x, y, and z coordinates for each point on the road network. These points can be used to create a 3D visualization where the z coordinate represents the height of that point from sea level.

The data has been generated incrementally for different levels, first we have created data for city level, ie. Riverside in our case, then we have processed it for bigger areas like state-wide, country-wide and finally for the world-wide.

Considering the amount of data we need to process, We have assessed different tools of visualization like ArcGIS, Google Earth, Three.js, CesiumJs, Ploymaps, etc. Among which we have finalized upon the Google Earth and Three.Js. Predominantly we are using Google Earth Pro to render the 3D data, but it has its own limitations. Google earth uses a XML based representation called KML file (Keyhole markup language) which is an XML notation for expressing geographical annotation and visualization. The KML file contains the entire data passed as the value of coordinates attributes in XML.

The data is fetched and converted from a Json file. Hence, when the data becomes huge, the conversion gets difficult and expensive. Also, Google earth uses its own overlay in the maps, which can not be removed or replaced. It can only be overlapped with another overlay. Hence, for huge data, the visualization gets difficult, this is where we can use Three.js, which is an open-source javascript library. It is used to create and display animated 3D computer graphics in a web browser. It comes with a good support of maps and 3D visualization and can be used for independent simulation of 3D roads altogether from scratch. Three.Js scales really well for huge datasets.

Below is the example of the KML file we are using to send the input 3D points into Google Earth. List of points are passed into <coordinates/> tag inside <LineString/> attribute of Placemark.

For removing the Google Earth's Default overlay, we are placing a plain colored image on top of it for better visualization as shown in Figure 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
    <Document>
        <name>Paths</name>
        <description>Big data project.</description>
        <Style id="yellowLineGreenPoly">
            <LineStyle>
                <color>7f00ffff</color>
                <width>4</width>
            </LineStyle>
            <PolyStyle>
                <color>7f00ff00</color>
            </PolyStyle>
        </Style>
        <Placemark>
            <name>Absolute Extruded</name>
            <description>Transparent green wall with yellow outlines</description>
            <styleUrl>#yellowLine</styleUrl>
            <LineString>
                <extrude>0</extrude>
                <tessellate>1</tessellate>
                <altitudeMode>absolute</altitudeMode>
                <coordinates> -117.4095937,34.0880706,378.0
                -117.4112566,34.0880795,378.0</coordinates>
            </LineString>
        </Placemark>
        <Placemark>
            <name>Absolute Extruded</name>
            <description>Transparent green wall with yellow outlines</description>
            <styleUrl>#yellowLine</styleUrl>
            <LineString>
                <extrude>0</extrude>
                <tessellate>1</tessellate>
                <altitudeMode>absolute</altitudeMode>
                <coordinates>
                    -117.2597693,34.077561,299.0
                    -117.2597482,34.0758706,299.0
                    -117.2597512,34.0743249,313.0
                    -117.2597801,34.0735512,313.0
                    -117.2600513,34.0728645,313.0
                    -117.260586,34.0722261,313.0
                    -117.2607879,34.0720068,313.0
                    -117.2609561,34.0717784,313.0
                    -117.2610609,34.07155,313.0
                    -117.2611437,34.0713284,313.0
                    -117.2611961,34.0711091,313.0
                    -117.2612142,34.0700013,313.0
                    -117.261202,34.0689843,313.0
                    -117.2612302,34.0678092,313.0
```

Figure 5. KML file sample used for visualizing in Google Earth

We began with visualization on the small scale of dataset of road network on Google Earth and tried to evaluate our outputs.

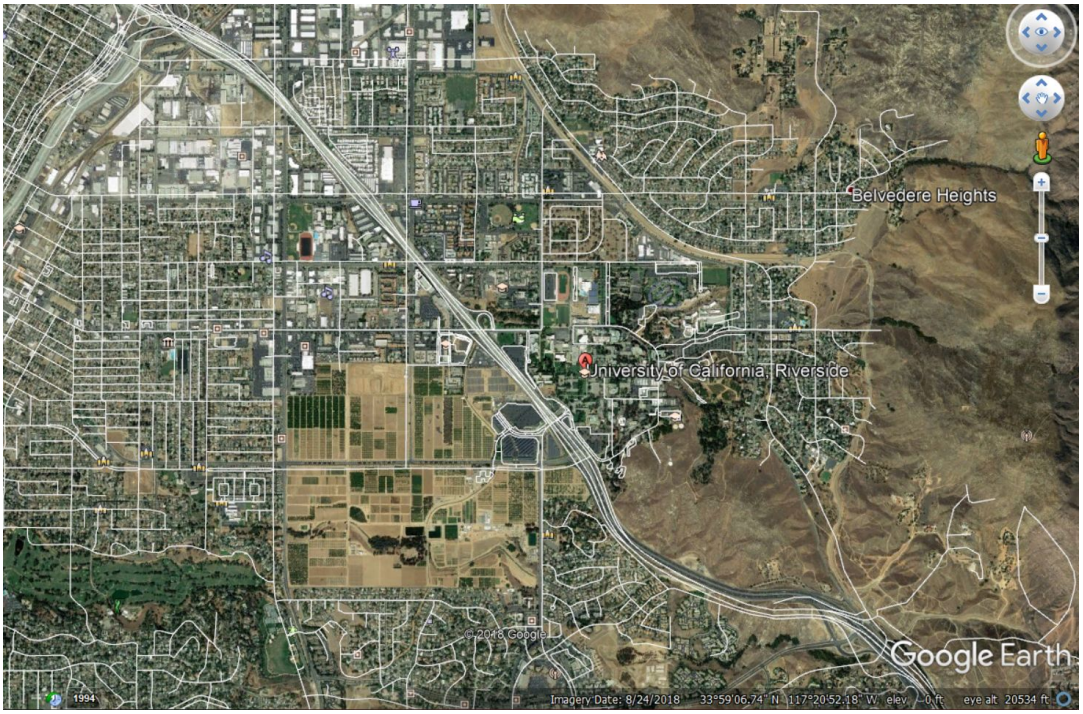

Figure 6. Riverside map output with terrain

To show and evaluate the altitude of the points, we are showing the road network in absolute elevation mode in Google earth, where it shows the height as shown in the below screenshot.

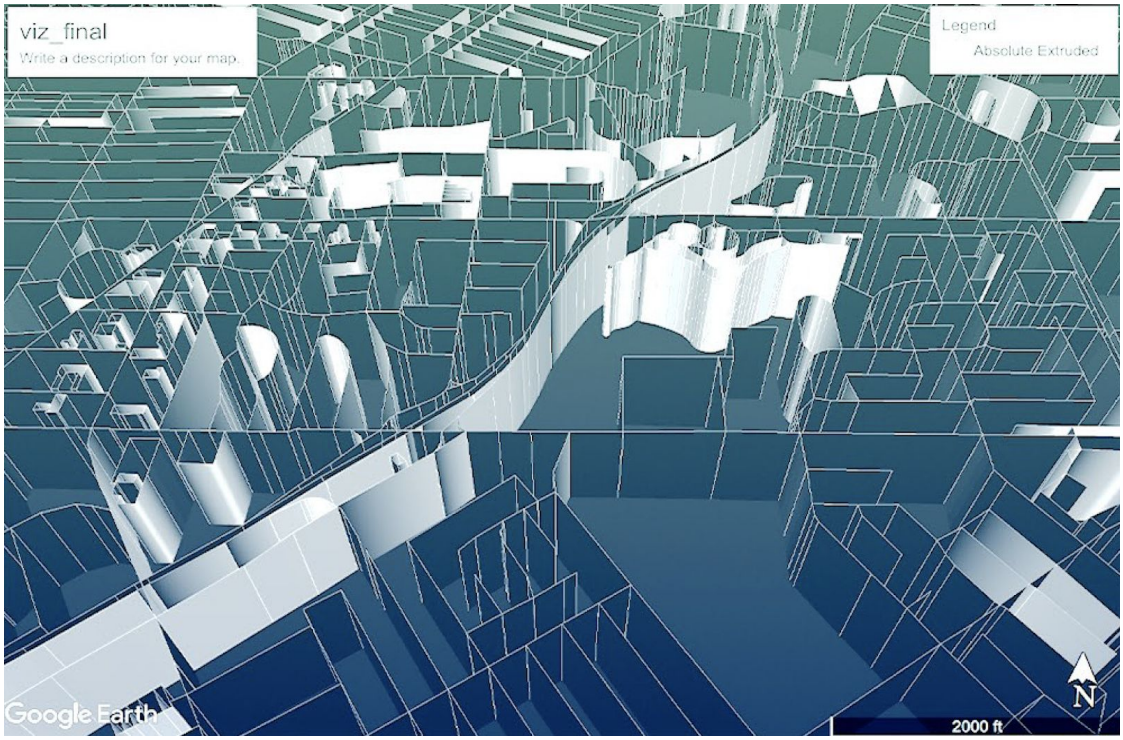

Figure 7. visualization with altitude on Google earth

With this we can also consider the slope of the road along with their widths. the slope can be calculated by delaunay triangulation. Firstly the terrain data is stored as a Triangular Irregular network, with the height for each point. Here every triangle initially different inclination. After buffering,the left and right side of the road contains as much points as the centre axis.Then, the object is triangulated in such a way that the cross sections situated at the points of the road center axis, borders the triangles of the object TIN and smooth slope can be interpolated on it.

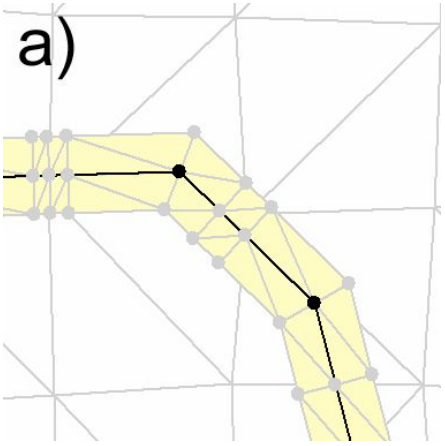

Figure 8. Original DEM TIN and object "road" after buffering [5]

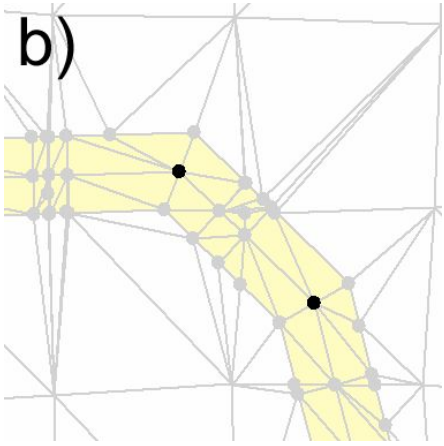

Figure 9. After applying delaunay triangulation [5]

3.3. Evaluation & Experiments

This part will provide some details about the prediction and how accurate our 3D road network model is.

In the project, we extract dataset which represents Riverside as sample data. After evaluating the sample model, we expand our dataset to large scale such as the United States and global map. We have an assumption while combining two datasets:

- If DEM dataset does not contain some points in the OpenStreetMap: We first use delaunay triangle to implement the missing area. Then we estimate the height based on the new data.

Evaluating the model based on the accuracy of the visualization result:

- Compare the visualization result with Google Earth: the actual geologe location and its height in reality.

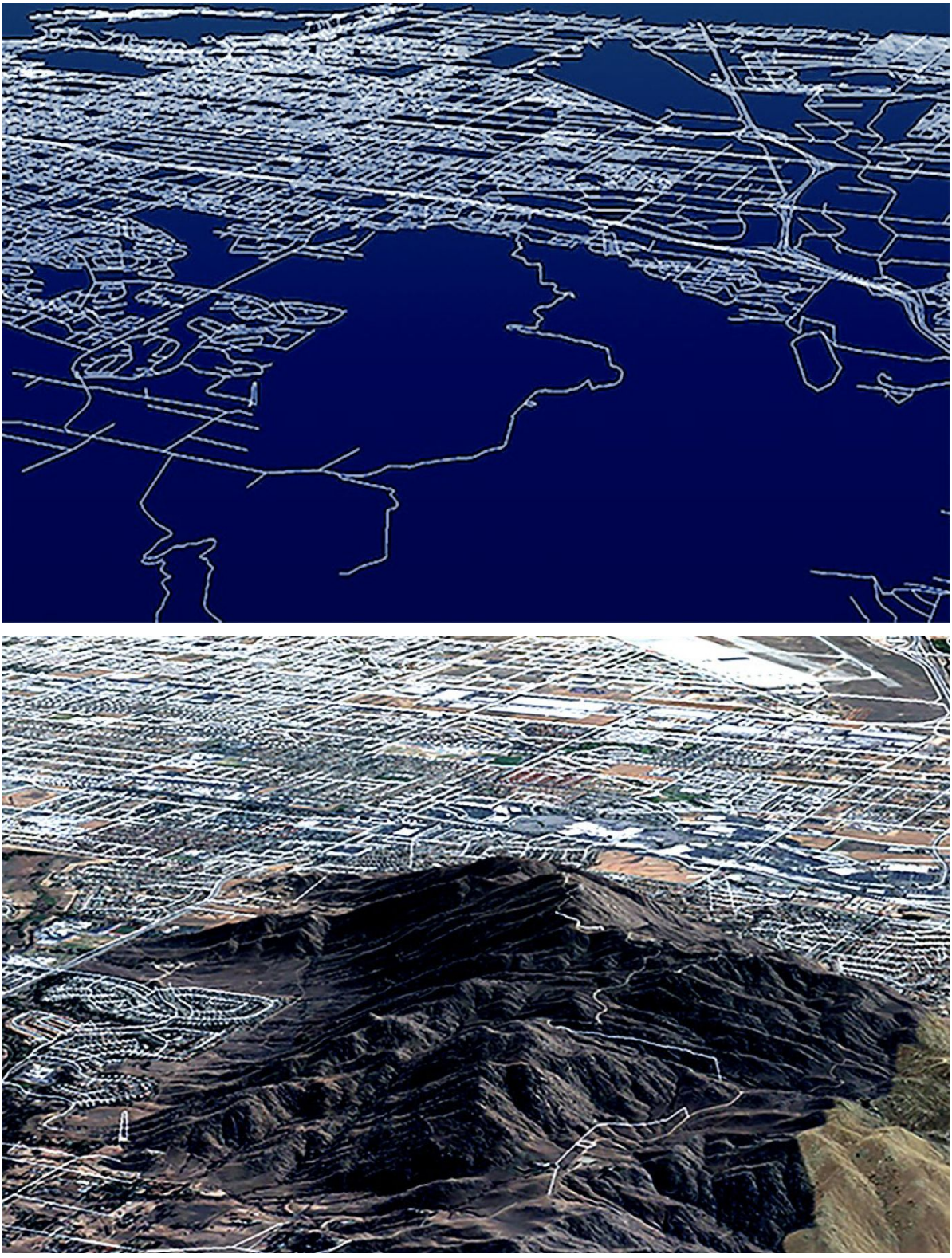

Figure 10,11. applying visualization with terrain

4. Future Work

There are some limitations in our project. Firstly, The DEM data we collected does not contain the altitude of all the points on the road. It means that the output data we generated contains missing data. In order to find these missing altitude points we can use Delaunay Triangulation method which gives us the approximate value of altitudes. Here we mention some other model data that can be used to enhance our results in different aspects:

Digital Terrain Model (DTM): A DTM is a vector data set composed of regularly spaced points and natural features such as ridges and breaklines. A DTM augments a DEM by including linear features of the bare-earth terrain.

DSM (Digital surface model) data: This data is more refined which includes heights of bridges, tunnels and other man-made structures as part of the road. Hence, it can be used to include their altitudes as well, which is missing in the DEM data presently.

Moreover, the current project implementation focuses on Riverside but this can be expanded to a global level.

References

[1]. Zheng, Y., Li, X., Li, M., Li, X., & Tang, D. *"Modeling road surface and network from a 3D perspective,"* in *IEEE*, 2010

[2]. T. Tang, J. Z. *"Terrestrial laser scan survey and 3D TIN model construction of urban buildings in a geospatial database,"* in *Geocarto International*, 259-272, 2008

[3]. Yuzhong Shenb, J. G. *"Automatic high-fidelity 3D road network modeling based on 2D GIS data,"* in *Advances in Engineering Software,Elsevier*, 86-98, 2014

[4]. Chen, M.-B. *"A parallel 3D Delaunay triangulation method. IEEE Ninth International Symposium on Parallel and Distributed Processing with Applications,"* in *IEEE,* 2011

[5]. Koch, A. *"An approach for the semantically correct integration of a DTM and 2D GIS Vector data,"* 2002

[6]. Christen M., Nebiker S. *"Large Scale Constraint Delaunay Triangulation for Virtual Globe Rendering. In: Kolbe T., König G., Nagel C. (eds) Advances in 3D Geo-Information Sciences,"* Lecture Notes in Geoinformation and Cartography. Springer, Berlin, Heidelberg

[7]. Yuxing Chen, Peter Goetsch, Mohammad A. Hoque, Jiaheng Lu and Sasu Tarkoma. *"d-Simplexed: Adaptive Delaunay Triangulation for Performance Modeling and Prediction on Big Data Analytics,"* in *IEEE,* 2019

[8]. Malcolm Sambridge, Jean Braun and Herbert McQueen, *"Geophysical parametrization and interpolation of irregular data using natural neighbours,"* in *Geophysical Journal International*, Volume 122, Issue 3, Pages 837–857, 1995

[9]. Jia Yu, Zongsi Zhang and Mohamed Sarwat. *"GeoSparkViz: A Scalable Geospatial Data Visualization Framework in the Apache Spark Ecosystem,"* in *SSDBM*, 2018

[10]. Ahmed Eldawy, Mohamed F. Mokbel and Christopher Jonathan. *"HadoopViz: A MapReduce Framework for Extensible Visualization of Big Spatial Data,"* in *ICDE*, 2016

[11]. Saheli Ghosh, Ahmed Eldawy and Shipra Jais. *"AID: An Adaptive Image Data Index for Interactive Multilevel Visualization,"* in *IEEE*, 2019

[12]. Da Vinci Project. Homepage: https://davinci.cs.ucr.edu/

[13]. CesiumJS. Documentations: https://cesium.com/docs/

[14]. GeoTrellis. Documentations: https://docs.geotrellis.io/en/latest/index.html

[15]. GTOPO30 Documentations: https://prd-wret.s3-us-west-2.amazonaws.com/assets/palladium/production/s3fs-public/atoms/files/GTOPO30_Readme.pdf