3D Augmentations of 2D Maps

Nir Hershko
3D Augmentations of 2D Maps

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Nir Hershko

Submitted to the Senate of the Technion — Israel Institute of Technology
Sivan 5775 Haifa June 2015
This research was carried out under the supervision of Prof. Gershon Elber, in the Faculty of Computer Science.

Some results in this thesis have been published during the course of the author’s research period, the most up-to-date versions of which being:


The generous financial help of the Technion is gratefully acknowledged.
Contents

List of Figures

Abstract

1 Introduction
   1.1 Background
   1.2 The Global Positioning System (GPS)
   1.3 The Open Street Map (OSM) project
   1.4 Digital Terrain Model (DTM)
   1.5 3D visualizations
   1.6 Overview

2 Previous Work
   2.1 Reconstruction of 3D highway interchanges
   2.2 Refinement and correction of Digital Terrain Models
   2.3 Estimating building height using smartphone

3 Augmentation of 2D highway interchanges to 3D using positional traces
   3.1 Introduction
   3.2 Problem Description
   3.3 The Method
      3.3.1 Registration
      3.3.2 Absolute average variant (AAV)
      3.3.3 Relative average variant (RAV)
   3.4 Application to Road’s Interchanges
      3.4.1 Input processing
      3.4.2 Boundary Conditions
   3.5 Results and Verification
   3.6 Conclusions and Future Work
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Amelioration and refinement of Digital Terrain Models (DTMs) from positional traces</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>The method</td>
<td>27</td>
</tr>
<tr>
<td>4.2</td>
<td>Implementation</td>
<td>28</td>
</tr>
<tr>
<td>4.3</td>
<td>Creating a DTM without augmentation</td>
<td>31</td>
</tr>
<tr>
<td>4.4</td>
<td>Results</td>
<td>31</td>
</tr>
<tr>
<td>4.5</td>
<td>Conclusions and Future Work</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>Estimating buildings’ heights using smartphones</td>
<td>35</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>35</td>
</tr>
<tr>
<td>5.2</td>
<td>The Tangent method</td>
<td>36</td>
</tr>
<tr>
<td>5.3</td>
<td>The four directions method</td>
<td>37</td>
</tr>
<tr>
<td>5.4</td>
<td>Comparison</td>
<td>39</td>
</tr>
<tr>
<td>5.5</td>
<td>Application Implementation</td>
<td>41</td>
</tr>
<tr>
<td>5.6</td>
<td>Conclusions</td>
<td>42</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions and future work</td>
<td>45</td>
</tr>
</tbody>
</table>

Hebrew Abstract i
List of Figures

1.1 GPS traces recorded while driving on Moshe Pliman road, Haifa, Israel, and a cross section at (32.78202N, 34.97626E). ................................................. 4
1.2 Elevation recordings of the same traces (left). The same recordings, each coerced to zero at the center (right). .................................................. 5
1.3 Several GPS traces recorded with the vehicle coming to a stop (before a traffic light), and their projection on the xy and yz planes. ................. 5
1.4 The map selection page of the OSM3D site. ........................................ 7

3.1 An example of (a) output and (b) inputs for our reconstruction method of 3D graphs. ................................................................. 14
3.2 Comparison of the two reconstruction variants on a synthetic example. 15
3.3 Overview of our 3D graph reconstruction method. ......................... 16
3.4 Construction of a candidate graph for the registration of a trace. ....... 17
3.5 Constructing $T_i^P$ using linear interpolation of the elevation of the registered trace $T_i^R$ on nodes along its path. .......................... 17
3.6 Example of the different types of constraints. .................................. 19
3.7 Side-view of a long road segment (bottom, in green), and its refinement (top, in blue) revealing some elevation curvature. ...................... 20
3.8 The result of applying the RAV reconstruction algorithm on some interchanges. ................................................................. 21
3.9 The OSM map and the available GPS traces at the ‘Mesubim’ interchange (32.038N, 34.830E). ................................................................. 22
3.10 The result of applying the RAV reconstruction algorithm on the interchange in Figure 3.9. ................................................................. 22
3.11 Plots of elevations along several roads of the interchange in Figure 3.9. 23
3.12 Histograms of the elevation difference between the tested method variants and the ground truth. ................................................................. 24

4.1 Refinement around a trace (blue). We only refine tiles that intersect the area of influence of the trace (green). ............................................ 29
4.2 An example of three weight functions. Left: the weight function after normalizing to \( f(3 \text{ meters}) \) and clamping to 1. Right: an example of applying the function in a synthetic example. The base DTM is at elevation 0 with \( W_{DTM} = 0.05 \).

4.3 An example of augmenting a DTM using GPS traces, in Nesher, Israel (32.77305N, 35.02980E).

4.4 An example of creating a DTM using only GPS traces, in Beit Oren forest, Israel (32.7275N, 35.0142E).

5.1 Panorama towers, Haifa, Israel (32.806N, 34.988E).

5.2 Diagram of the tangent method.

5.3 A diagram of the four direction vectors from the user to the corners of one of the building’s faces.

5.4 A scatter plot of the results of the different methods on the “Technion” dataset (32.77744N, 35.02155E).

5.5 Diagram of the synthetic dataset, top view.

5.6 Comparison between the methods – The results of the monte-carlo simulation on the synthetic dataset in a varying distance \( d \in [0 \text{m}, 140 \text{m}] \) and at a constant angle \( \theta \).

5.7 Comparison between the methods – The results of the monte-carlo simulation on the synthetic dataset in varying angles \( \theta \in [0^\circ, 90^\circ] \) and at a constant distance \( d = 45 \text{m} \).

5.8 Screenshots of our Android application.
Maps of the world around us are used daily for many purposes such as navigation. 3D maps, with terrain elevations and 3D objects, allow to model and simulate natural phenomena such as water flow or line-of-sight, and assist in orientation.

This dissertation presents several methods for augmenting 2D maps into 3D, each focusing on a different aspect of the map. The recurring theme in these methods is that the acquisition of the data can be done using a modern smartphone, and that they allow to use crowdsourcing (obtaining the data using a large number of people). In all three methods, the phone’s GPS sensor is used to acquire the global position of the user. However, the elevation reported by it is not as accurate as the horizontal location and this must be taken into account when using this data. In the methods’ implementations, we use data from the OpenStreetMap (OSM) project. We use the map provided by this project as the 2D map to augment to 3D, and the repository of GPS traces uploaded by OSM contributors.

The first method presented in this work reconstructs a roads interchange in 3D using a 2D map of the interchange and GPS traces recorded while driving. To overcome inaccuracies in the elevation data of the GPS traces, we only use the relative elevations along the trace. In the result, any road segment without GPS coverage is interpolated between the endpoints of that segment, where the relative elevation between those endpoints is inferred based on other paths between these points.

The second method reconstructs a digital terrain model (DTM) from GPS traces recorded in an area. This method can augment an existing DTM by refining it and integrating the data from the GPS traces (even in case the terrain has changed). To overcome errors in the elevations of the GPS traces, we add an offset factor to each trace, and calculate the offsets that achieve the best correspondence between the traces.

The third method calculates heights of buildings. The user selects a face of a building and captures the direction vectors between the user and the four corners of that face, using the smartphone’s sensors. This data, together with the 2D location of the building, is used to estimate the building’s height. As part of this method, we developed an Android application that allows the user to collect the data.

In order to visualize the results of all these methods, a web-based application was developed to display an interactive 3D map, including terrain, roads and buildings, using WebGL technology. It is accessible in http://osm3d.cs.technion.ac.il/. This
application was written in part by Eli Sherer as a part of a student’s project. The data displayed by this interactive map is based on OpenStreetMap data, with an existing globally available DTM, and augmented using the results of the methods presented in this work, that were uploaded to the server in advance.
Chapter 1

Introduction

In this chapter, we introduce the subject of this thesis work, and the concepts and frameworks shared by some of the following chapters.

1.1 Background

Maps of the world around us are used daily for many purposes such as navigation or location-based services. However, our experience with maps is usually limited to a 2D top-down projection. Using a 3D map instead have some advantages. To name a few:

- 3D visualization can help humans to interpret the maps – as elevational features can assist in orientation.
- When navigating, the elevation profile of a route may affect navigation decisions (as driving or walking up a hill and back down is sometimes less desirable than going around in a flat path) [SLNZ09].
- 3D road maps can support personal navigation assistants. Better online map matching can be achieved when facing ambiguities in the current location (after road bifurcations, for example), as the elevation change reported by the GPS may assist resolving the ambiguities [QON07]. On the other hand, using the elevation derived from the map can reduce the number of satellites required for a GPS position [TBS+01].
- 3D maps allow to model and simulate natural phenomena such as water flow [WG00], or line-of-sight analysis to detect object visibility [YPL07].

Chapter 2 contains an overview of some of the existing 3D map types.

This work presents several independent approaches to augment 2D maps to 3D, possible using some of the capabilities of modern smartphones. Each approach is augmenting a different aspect of the map. We focus on the following aspects:

1. Roads network (viewed as a graph embedded in $\mathbb{R}^3$). In this part, in Chapter 3, we augment 2D roads’ networks into 3D.
2. Digital Terrain Models (DTMs) – assigning each location with its ground elevation. We present a method to refine and ameliorate existing DTMs by increasing the amount of details, and improving their accuracy, in Chapter 4.


1.2 The Global Positioning System (GPS)

To augment road networks and DTMs, we use positional traces — a sequence of points in 3-space — as a source of information. While these methods can be used with any type of positional traces, they were designed with the prevalent GPS or other Satellite Navigation Systems in mind – both by regarding only outdoor traces, and by overcoming the inherent inaccuracies in consumer-grade GPS receivers.

While only 2D data (Latitude and Longitude) is typically used in navigation and mapping applications, any GPS receiver calculates the elevation component as well, and it is usually also recorded. However, the elevation is not as accurate as the Latitude and Longitude. From Figure 1.1, it is clear that the horizontal readings of the GPS are significantly more precise than the elevation readings. Note that the blue and the black marks represent two opposite roads, each with two driving lanes, so some variance on the horizontal axis is expected.

![Figure 1.1: GPS traces recorded while driving on Moshe Pliman road, Haifa, Israel, and a cross section at (32.78202N, 34.97626E). The traces were recorded on the same device and under the same conditions over a period of several months.](image)

Although the elevation is not very accurate, local differences are quite accurately
measured while the positions tend to drift globally. Figure 1.2 presents an example of the observed behaviour of GPS traces, in the ‘z’ axis. As can be seen from the figure, while the variance of the positions is quite large, the relative differences are quite similar.

![Figure 1.2: Elevation recordings of the same traces (left). The same recordings, each coerced to zero at the center (right).](image)

During the recording, the user can change his/her speed and can even come to a stop. As the position of the GPS trace continues to drift, it results with a significant error over a short distance. This can be seen in Figure 1.3.

![Figure 1.3: Several GPS traces recorded with the vehicle coming to a stop (before a traffic light), and their projection on the xy and yz planes. Note again the large variance in z compared to xy.](image)

1.3 The Open Street Map (OSM) project

OpenStreetMap [HW08] – [http://www.openstreetmap.org/](http://www.openstreetmap.org/) – is an open (2D) mapping project that offers various services around a central database of geographic information (the map) — including map tiles’ rendering and hosting, map editors, and GPS traces repository. The project’s data set is maintained by a community of contributors who upload GPS traces and edit the map.
In this work, we are basing our implementation on the map provided by the OpenStreetMap project for road networks in Chapter 3 and buildings in Chapter 5. The map can be accessed as an XML file using the OSM API, as explained in http://wiki.openstreetmap.org/wiki/Download. In addition, we use this project as a source for 3D positional traces of roads, as many of the GPS traces uploaded by the users of OSM were recorded while driving. Traces in an area are queried using the OSM API (described in http://wiki.openstreetmap.org/wiki/API_v0.6#GPS_traces). Since elevation information is not provided as part of the API result, we then download the original traces files (trace files that were marked by their uploader as public, have their ID provided as part of the API result and can be downloaded from the OSM site).

1.4 Digital Terrain Model (DTM)

A Digital Terrain Model is a digital representation of a terrain’s surface. In this work, we regard DTMs that are represented as a raster – a square grid of terrain elevations. For the sake of simplicity – the refined DTM (see Chapter 4) will be represented as a two-level hierarchy of square grids.

There are a few world-wide DTMs that are publicly available. The most widely used was recorded on NASA’s Shuttle Radar Topography Mission (SRTM) [FRC+07] in February 2000, and processed by CIAT-CSI [Jar08]. It has a resolution of three arcseconds per pixel (∼90m). In this work, we use this DTM as a base DTM for refinement in the examples in Chapter 4 and for boundary conditions used in the verification of the implementation in Chapter 3.

An overview of other common methods of acquiring DTMs appears in Chapter 2.

1.5 3D visualizations

As part of this thesis, we built a system to visualize the 3D results of the algorithms presented in this work, called OSM3D. It is based on OSM and implemented as an interactive web site — publicly available on http://osm3d.cs.technion.ac.il/.

On the first page (shown in Figure 1.4), the user is requested to select the bounding box of the area to visualize in 3D. The server automatically collects the up-to-date map in that bounding box from OpenStreetMap and merges it with:

- The results generated for interchanges by the algorithm in Chapter 3.
- The buildings’ heights that were collected as described in Chapter 5.

The DTM is loaded from SRTM data, and merged with:

- The refined DTM resulting from the algorithm described in Chapter 4.
- A post-processing of the reconstructed 3D interchanges (i.e. Chapter 3) – lowering the terrain to below the roads, where necessary.
The visualization page (shown in the examples throughout this work) retrieves the aforementioned data from our server, and builds a 3D scene that is rendered in the browser using WebGL technology [Mar11]. This 3D visualization page was written by Eli Sherer, as a part of a student’s project.

1.6 Overview

The rest of this thesis is organized as follows. Chapter 2 provides an overview of previous works and the state of the art related to this work. In Chapter 3, we propose a method that allows one to reconstruct highway interchanges in 3D from positional traces, as a special case of 3D graph reconstruction. In Chapter 4, we describe how to use positional traces spread in some area to refine or update a DTM in that area, and compare the results to existing globally available DTMs. Chapter 5 puts to use some of the other capabilities of a modern smartphone - the accelerometer and magnetometer – to capture the height of a building. Finally, Chapter 6 concludes the thesis.
Chapter 2

Previous Work

Several types of 3D implementations of maps exist today:

1. A map consisting of a 3D textured meshes. This is implemented in Google Earth/Maps [Goo12], Nokia Maps [Nok12], and Apple iOS maps [App12]. In these implementations, the map is available only in selected cities. The 3D mesh is generated using algorithms operating on street-level, aerial and satellite photography. While providing great visual fidelity and photo-realism, this type of 3D map is very expensive to produce (hence its limited availability), and the 3D mesh data is separate from GIS features such as roads and buildings — so some applications like road navigation cannot use the data that is represented in this kind of map.

2. A surface elevated using a DTM, with superimposed 3D features such as buildings. Sometimes the surface has an overlay of 2D imagery such as a satellite image or a 2D rendered map. This type of map is implemented in Google Earth, OSM 3D Globe [USL⁺], and many others, including the visualization system we constructed as part of this work (see Section 1.5).

3. A flat 2D map with 3D extrusions of buildings according to their height (this type is sometimes also named ‘2.5D’). This 3D map type is implemented in the mobile Google Maps (in selected cities). This type of map is using the 3D buildings for simplified human orientation, but lacks all other 3D features.

Each of those map types might benefit from the reconstruction methods presented in this work, as either a direct improvement to the visualized map’s data, as an automatic mean to detect changes (a road or DTM that doesn’t match the 3D mesh), or as a data source for an accompanying service such as navigation.

2.1 Reconstruction of 3D highway interchanges

There are several approaches in the literature to the reconstruction of 3D graphs, differing mainly by the types of inputs available. In the medical field, Coste et al. [CVR99]
presented an algorithm to reconstruct a 3D network of blood vessels from several 2D images of projections of the network, acquired from angiographic imaging. Another general method, by ACC+12, reconstructs a 3D graph from 3D traces of paths on the graph. This method considers the input traces as a graph embedded in a metric space – a “metric graph” [Kuc04]. The metric graph is simplified and then re-embedded in $\mathbb{R}^3$. It should be noted that the conversion of traces to a (single connected) metric graph is not natural, and this approach disregards the connectivity of the points in the traces. Also, this method results with a graph that is guaranteed to be topologically-correct only for a certain limit of positional error.

In the field of GeoInformatics, there have been done several works for 3D road network reconstruction from 3D imagery data (depth images, or point clouds). Chen et al. [CLST06] presented a method to reconstruct 3D road models using LIDAR data captured airborne, in two steps: 2D networking of the graph and augmenting the 2D graph to 3D. Such methods usually offer a relatively high quality model thanks to the accuracy of the LIDAR (LIght Detection And Ranging) scanner [WW02], but acquiring this kind of input requires the right equipment. In addition, the data may be partial due to concealments from the environment or from the model itself. For example, in the case of road networks, some roads might be concealed by higher levels of the road or nearby structures.

The second approach is the reconstruction of graphs from 3D traces of paths on the graph. This was studied primarily in the context of GeoInformatics — a result of the prevalence of low-cost GPS receivers, making it easy to crowd-source positional traces [Hei10]. While most works focus on reconstruction of 2D road networks [CK09, CGHS10], it has been demonstrated that GPS traces can also be used to reconstruct a 3D road network [GIK07]. However, the main problem of [GIK07] in 3D reconstruction of road networks is that it results, in many cases, with a topologically-inaccurate graphs that need to be manually corrected [FK10].

Our solution uses 3D GPS traces as inputs, which are very easy to acquire compared to aerial imagery. We use the curated topology of the map as a base model, and augment it using a relatively small number of traces compared to other methods that reconstruct the graph from scratch.

### 2.2 Refinement and correction of Digital Terrain Models

A common method of acquiring DTMs is an aerial photo capturing or LIDAR scanning — and the appropriate processing. This can provide a very-high resolution DTM but it is a rather expensive to acquire this data.

Koch and Heipke [KH06] suggested a method to *semantically correct* a DTM based on a 2D map by applying constraints on the semantics of a map, such as lakes that should be flat, and roads that are represented as tilted connected planes. This approach requires a pre-curated map, and is limited to where such features exist so for example it
cannot benefit parks and hiking trails.

Our solution introduces a new data source for DTMs, used in conjunction with an existing low-resolution DTM, allowing to achieve a higher-resolution DTM where the other methods either don’t have any terrain semantics available, and where an aerial capture is too expensive and impractical.

2.3 Estimating building height using smartphone

Several works have been made to recreate building models from satellite or aerial imagery \cite{Bre00} \cite{BZ00}. The main disadvantage of this approach is the high cost of acquiring the data. We will focus on methods that rely on data that can be collected easily by end-users.

Kim et al. \cite{KSS+08} exploited the reduction in signal to noise ratio when a building obstructs the line-of-sight from a receiver to the GPS satellite, to reconstruct a 3D model of a building. However, this method requires a significant amount of data, with the GPS satellites at the “correct” angles in order to provide an accurate model.

Debevec et al. \cite{DTM96} suggested a photogrammetric method to model architectural scenes by specifying constrained building blocks (cubes, prisms, etc.) and measuring them using a set of photographs. The user specifies the camera parameters and then matches each photograph to the model by marking the features in the image. The camera positions and model parameters are reconstructed using a non-linear optimization algorithm. Our approach, in contrast, employs the prior information available about the users’ locations obtained from the GPS sensor in the smartphone, and the prior information about the model obtained from a 2D map to reconstruct the single missing parameter that is the building’s height. Using this prior information allows us to have a much simpler user experience where only a single point of view is sufficient.

Simple trigonometric methods that are commonly used to estimate the height of trees, can be also employed to estimate the height of buildings. The tangent method estimates the height of a tree from the horizontal distance to the subject tree and the angles to the top and the base of the tree. These can be acquired using a clinometer and a tape. The sine method is using the distance to the top of the tree instead of the horizontal distance, which can be acquired using a laser rangefinder. Both methods are presented and compared in \cite{LML13}. Other similar methods exists, for example comparing to an object of a known size at the same horizontal position as the tree \cite{KT94}. Our approach, suited for smartphones, is also based on measuring angles (using the accelerometer) and employs the GPS to determine the horizontal distance to the building. Given these inputs, our approach also takes advantage of the premise that buildings have two-dimensional faces (we assume rectangular faces in this work) to achieve results that are more accurate than the tangent method.

Another possible method is to estimate the height of a building from the number of floors it has, as explained in \cite{oTBH15}. While simpler to implement, this method
requires the user to count the number of floors in the building and is generally not
guaranteed to be accurate, with deviation of up to 25% in some cases. In addition,
the assumptions presented in [oTBH15] are designed for tall buildings where the main
function is known – Office, Residential, or Mixed/Unknown.
Chapter 3

Augmentation of 2D highway interchanges to 3D using positional traces

We view the problem of reconstructing a network of 3D roads as the more general problem of reconstructing a 3D graph from its 2D projection and a set of inaccurate 3D positional traces on it. When using this general method to reconstruct road’s interchanges in 3D, an existing 2D road map serves as a 2D projection of the desired 3D road graph, and the 3D positional traces are provided by GPS traces recorded while driving through the interchanges.

This chapter presents a general approach for this reconstruction problem, and demonstrates that the implementation of this method for highway interchanges yields a robust and accurate solution compared to real ground truth data.

3.1 Introduction

A 3D graph is a graph in which every node is assigned a position in 3D. Edges on the graph are defined to be straight lines, so their coordinates are derived from the node’s coordinates. An example for a 3D graph can be seen in Figure 3.1a. The inputs to the proposed reconstruction method are shown in Figure 3.1b:

1. A 2D orthographic bijective projection of nodes of the graph – A graph with the same topology, but with nodes’ positions in 2D. This 2D graph can be self-intersecting.

2. A set of (imprecise) 3D positional univariate traces along the graph’s edges.

Our goal is to estimate the third coordinate of each node’s position — which will henceforth be referred to as an elevation. We discuss two variants of the elevation method, both based on registering the traces to the 2D graph:
1. **The absolute average variant** (AAV) averages the elevations of the traces at every node.

2. **The relative average variant** (RAV) deduces the elevations using a first-order differences of the traces’ elevations.

We will show that the RAV provides results that can be more robust and precise, under certain conditions. We also present a method to reconstruct the 3D graph even when some of it’s nodes are not covered with any 3D trace. This is achieved based on some additional assumptions on the structure of the 3D graph.

![Diagram showing a 3D graph](image1)

![Diagram showing a 2D projection of the 3D graph and set of imprecise 3D traces](image2)

Figure 3.1: An example of (a) output and (b) inputs for our reconstruction method of 3D graphs.

The rest of this chapter is organized as follows. Section 3.2 explains the reconstruction problem in more details and Section 3.3 describes our proposed reconstruction method. Section 3.4 outlines the application of the method to reconstruction of road’s interchanges from GPSs. In Section 3.5, we present results and verification of the implementation against real ground truth data, and finally, Section 3.6 discusses some future work and concludes this chapter.

### 3.2 Problem Description

Let $G = (N, E, p)$ be a 3D (possibly directed) graph, comprising of a set of nodes $N$, a set of (possibly directed) edges $E$, and position mapping $p : N \rightarrow \mathbb{R}^3$. We will regard the edges as the linear segment between the corresponding nodes’ positions. The graph’s 2D projection is denoted $\tilde{G} = (N, E, \tilde{p})$, with $\tilde{p} : N \rightarrow \mathbb{R}^2$. The unknown vertical component of some node $N_k$, will be referred to as the node’s *elevation* and denoted $N_e^k$. In the rest of this chapter, we will use $k$ and $l$ as indices of nodes.

A *trajectory* $p_i$ along the 3D graph is a continuous arc-length piecewise-linear parameteric curve $p_i(s) : [0, L_i] \rightarrow \mathbb{R}^3$ such that $p_i(s)$ is always on a graph’s edge, and its derivative $p_i'(s)$, if exists, adheres to the edge’s direction. A *trace* $T_i$ over the 3D graph $G$ is a piecewise-linear sampling $T_i = \{t_{i,j}\}$ such that $t_{i,j} = p_i(s_{i,j}) + e_i(s_{i,j}) \in \mathbb{R}^3$ for some trajectory $p_i(s) : [0, L_i] \rightarrow \mathbb{R}^3$ on the graph and some error function $e_i(s) : [0, L_i] \rightarrow \mathbb{R}^3$. 

We assume that the error function $e_i(s)$ is bounded and piecewise Lipschitz continuous. In the rest of this chapter, we will use $i$ as an index of a trace and $j$ as an index of a point in a trace. The word “node” (or “graph node”) will be exclusively used for elements in $N$, whereas the points along some trace will be referred to as “trace points”.

Given the graph’s 2D projection $\tilde{G}$ and a set of such traces $T = \{T_i\}$, in the next section we present our approach to reconstruct the graph in 3D — namely, assign each node its elevation.

### 3.3 The Method

Assuming the error functions $e_i$ are small enough, we will be able to register each trace to the route that represents it on the 2D graph, and use averages of the traces’ elevations as an estimator to the 3D graph position (The AAV approach). However, if the error $e_i$ drifts globally, the entire traces might shift, while the derivatives will remain quite accurate. In this case, one rather use the traces’ derivatives as an estimate to the corresponding edges’ derivative, and globally resolve them with proper boundary conditions over the graph to estimate the elevation (the RAV approach).

Hence, the method we propose to resolve the problem of 3D reconstruction of graphs includes the following steps:

1. Registration of the 3D traces to the 2D graph (a.k.a. map-matching).

2.a. (AAV) Averaging the elevations of the nodes of the 2D graph across all the traces, or;

2.b. (RAV) Averaging the elevations’ differences on the edges across the traces, and solving for the absolute elevations using some known boundary conditions.

![Figure 3.2: Comparison of the two reconstruction variants on a synthetic example. The original input traces are shown in black.](image)

Figure 3.2 compares the AAV and RAV in a typical reconstruction case for 3D graphs: two of the three traces (in black) are partial (a result of splitting or merging with other paths in the graph), and they don’t cover all this path. While the AAV results with discontinuities, the RAV results with a smooth solution. Figure 3.3 provides an overview of the different steps in our method, and the different steps are now discussed in detail.
3.3.1 Registration

The registration of the traces correlates every trace $T_i$ with the 2D graph $\tilde{G}$, resulting with the registered trace $T_i^R = \{t_i^{R, j}\}$, where every point on the trace, $t_i^{R, j}$, contains the corresponding position on the graph $\tilde{G}$, and the edges in the graph on the way to the next point of the trace, $t_i^{R, j+1}$. The registered trace’s path is the graph nodes along the trace. The input trace $T_i$ is of the same topology (i.e. number of points) as its graph-registered counterpart $T_i^R$. The registration can be done in any of the various methods that consider the traces’ positions and derivatives in relation to the 2D graph, account for the graph’s topology, or use other domain knowledge or information that is available in a specific application.

In this work, we use a simple Map-Matching method from the field of Geographic Information Systems, known as Weight-based Map Matching [YW04]. For every point $t_{i,j}$ along the trace, we find all possible candidate matched points on the graph, and weigh them based on the distance from the trace. We then build a candidate graph: The nodes are the set of all candidate points, and the edges are the set of shortest paths between pairs of adjacent candidate points, as seen in Figure 3.4. The least-cost path on the graph, calculated using a Dijkstra’s algorithm, corresponds to a valid matching of the trace with the minimal total distance from the graph. For more information about various Map-Matching methods, [YZZ+10] provide a recent overview, together with their own approach.
3.3.2 Absolute average variant (AAV)

In this reconstruction variant, every node’s elevation is calculated as a simple average of the elevations of all traces through it. First, every registered trace $T_i^R$ is used to calculate the elevations of the graph nodes in the trace’s path, using some interpolation or approximation scheme, resulting with $T_i^P = \{t_{i,j}^P\}$, such that $N[t_{i,j}^P]$ is the corresponding node in the graph and $e[t_{i,j}^P] \in \mathbb{R}$ is the node’s interpolated/approximated elevation according to this trace. It should be noted that $T_i^P$ is usually not of the same topology as the registered trace $T_i^R$ that it is based upon. Our implementation is using a linear interpolation of the elevations on the distance along the trace’s path in the graph, as is shown in Figure 3.5.

![Figure 3.5: Constructing $T_i^P$ using linear interpolation of the elevation of the registered trace $T_i^R$ on nodes along its path.](image)

After calculating the elevations of the nodes along the traces, $e[t_{i,j}^P]$, we simply average the elevations of each node in the graph. For the sake of brevity, we will use $E(k)$ to denote the set of all the elevated nodes (based on the different traces) corresponding to the node $k$:

$$E(k) = \{t_{i,j}^P : N[t_{i,j}^P] = k\}.$$  \hspace{1cm} (3.1)
In the AAV, the final elevation of the node k will then be:

$$AAV_k = \frac{1}{\|E(k)\|} \sum_{t_{i,j} \in E(k)} e[t_{i,j}] .$$

(3.2)

### 3.3.3 Relative average variant (RAV)

In this reconstruction variant, the interpolated traces \(\{T_i^P\}\) are calculated as in the AAV, but are used only for the first-order elevation differences between adjacent nodes. The elevation differences are averaged and then globally resolved in the least-squares sense, over an over-constrained weighted linear system, in which the unknowns are the nodes’ elevations \(\{N^e_k\}\), and boundary conditions provide absolute elevation anchors. Weights are used to amplify the effect of some constraints, such that a constraint \(C = 0\) with a weight of \(w\) contributes \(w^2C^2\) to the total squared-residuals sum of the system.

Let \(E(k, l)\) be the set of all pairs of adjacent elevated nodes:

$$E(k, l) = \{(t_{i,j}^P, t_{i,j+1}^P) \in E(k) \times E(l)\} .$$

(3.3)

Each such averaged elevation difference over a graph’s edge \(N_kN_l\) is expressed as the constraint in the linear system:

$$N^e_k - N^e_l = \frac{1}{\|E(k, l)\|} \sum_{(t_{i,j}^P, t_{i,j+1}^P) \in E(k, l)} e[t_{i,j}^P] - e[t_{i,j+1}^P] .$$

(3.4)

This type of constraint is exemplified in Figure 3.6 as yellow lines.

In addition to the relative elevations, some absolute elevation are also prescribed as boundary conditions.

$$N^e_k = BoundaryElevation_k .$$

(3.5)

A different weight could be assigned to some constraints, representing our confidence in these elevations’ accuracy. This type of constraint is exemplified in Figure 3.6 as red dots.

The RAV also allows us to assign elevation to nodes that are not covered by any trace. Under the assumption that the graph’s elevations tend to change smoothly and gradually, one can assign a zero-elevation-difference contraints to all edges that have no elevation difference information:

$$N^e_k - N^e_l = 0 \text{ if } E(k, l) = \emptyset .$$

(3.6)

Let \(k, l, \) and \(n\) be three consecutive nodes in the graph, such that node \(l\) is connected only to nodes \(k\) and \(n\), and there is no trace that traverses any of the edges \(kl\) and \(ln\). To obtain a linear elevation gradient over these edges, we assign a weight of \(1/\sqrt{d(k, l)}\) to the constraint in Equation (3.6), where \(d(k, l)\) is the xy length of the graph edge \(kl\).
The explanation for this choice of weight follows. As no other constraint depends on \( N^e_i \), minimizing the total energy (in \( L_2 \) sense) of the two weighted constraints

\[
E = \left( \frac{N^e_k - N^e_i}{\sqrt{d(k,l)}} \right)^2 + \left( \frac{N^e_i - N^e_n}{\sqrt{d(l,n)}} \right)^2 ,
\]

with respect to \( N^e_i \) (i.e. differentiating \( E \) with respect to \( N^e_i \) and equating to zero), results with

\[
0 = \frac{\partial E}{\partial N^e_i} ,
\]

or (\( d(\cdot, \cdot) \) represents the lengths in the 2D graph and does not depend on \( N^e_i \)),

\[
0 = \frac{1}{d(k,l)} \frac{\partial}{\partial N^e_i} \left( N^e_k^2 - 2N^e_k N^e_i + N^e_i^2 \right) + \frac{1}{d(l,n)} \frac{\partial}{\partial N^e_i} \left( N^e_l^2 - 2N^e_l N^e_i + N^e_i^2 \right)
\]

\[
= \frac{1}{d(k,l)} (-2N^e_k + 2N^e_i) + \frac{1}{d(l,n)} (2N^e_l - 2N^e_i) ,
\]

or (assuming \( d(\cdot, \cdot) \) is not zero),

\[
0 = d(l, n) (-N^e_k + N^e_i) + d(k, l) (N^e_i - N^e_n) ,
\]

and

\[
N^e_i = \frac{N^e_n d(k, l) + N^e_k d(l, n)}{d(k, l) + d(l, n)} ,
\]

which describes the linear dependency of \( N^e_i \) on its two neighbors. Repeating the process on pairs of edges along an “uncovered” path explains the linear change of the elevation between the two terminal nodes by induction. This type of constraint is exemplified in Figure 3.6 as a blue line.

![Figure 3.6: Example of the different types of constraints.](image)

The system of linear constraints formed out of Equations (3.4), (3.5) and (3.6) is solved in the least-squares sense. By experimentation, the weights we use for these types of constraints are 1 (for each pair of nodes), 10 (for each boundary node), and 0.1 (for a distance of 1 meter), accordingly. As this is a sparse system, a sparse least-squares solver such as LSQR [PS82] can be employed.

19
3.4 Application to Road’s Interchanges

This section describes the implementation of the presented method in the specific application of reconstruction of road’s interchanges in 3D from a 2D road map and GPS traces. We used data from the OSM project for both the tagged 2D road map and GPS traces uploaded by the users (see details in Section 1.3).

3.4.1 Input processing

In the Introduction chapter, we observed that GPS traces have the property of a global drifting error as defined in the problem description (with time as the curve parameter), but with the exception of when the user comes to a stop. Our proposed remedy is to remove and split the traces where the speed is too low – making the error as a function of the distance small – satisfying the problem’s requirements. One should also note that in practice this difficulty is not manifested in interchanges, in general, as cars usually move at some minimal speed.

We pre-process the 2D road map as well. Long straight roads are usually represented in OSM as a single segment (consisting of the two nodes at the extremes). However, in 3D it is often the case that the road is not straight but has some vertical alterations. To allow the reconstruction to express these higher frequency details, we refine all roads so the maximum segment is of bounded length – as demonstrated in Figure 3.7.

![Figure 3.7: Side-view of a long road segment (bottom, in green), and its refinement (top, in blue) revealing some elevation curvature.](image)

3.4.2 Boundary Conditions

As boundary conditions in the RAV, we use the circumference of the approximated region – where the calculated road network is stitched to the rest of the world. We seek to process (i.e. render) the algorithm’s result together with parts of the road’s network that don’t have an approximated elevation. Hence, all roads that enter/leave the region under approximation, will be set with an elevation boundary condition that will force the stitching to the surroundings, to be continuous.

3.5 Results and Verification

Figure 3.8 presents a pair of results of the algorithm using maps and traces from the OSM project. The road network in these examples was refined based on a maximum segment length of 30 meters.
In order to validate our approach, we use the ‘Mesubim’ interchange in Israel, at (32.038N, 34.830E), for which ground-truth data was kindly provided by Armi Grinstein – Geodetic Engineering Ltd (http://www.armig.co.il/). Figure 3.9 shows its map and a set of traces over it and Figure 3.10 shows the result of applying the RAV algorithm to the interchange.

The ‘Mesubim’ map consists of 601 nodes and 633 edges. 118 traces were processed into a linear system of 642 constraints, including 274 relative-elevations constraints (Equation (3.4)), 359 zero-elevation-difference constraints (Equation (3.6)) and 9 boundary conditions (Equation (3.5)). On a single 2.8GHz CPU core, matching the traces to the map took 6 seconds, after which constructing the constraints is of negligible time. Solving the least-squares problem took half a second. The “ground truth” elevation data was derived from a photogrammetric measurement and has an elevation accuracy of 0.3 to 0.5 meters, and horizontal sampling interval along the roads is in the range of 8 to 15 meters.

Since the OSM 2D data does not perfectly match, in xy, to the ground truth, the roads’ 2D locations were registered manually (with changes of up to 12 meters – perpendicular to the road direction) for the sake of this verification. The OSM roads at this interchange were then resampled at intervals of one meter, and the elevations...
resulting from our algorithms were compared to the elevations at those uniformly-sampled ‘ground truth’ points.

We compared both presented variants. Note that while the RAV provides elevation for roads that are not covered with any trace, the AAV does not. Figure 3.11 presents several of the road sections at the interchange, with the elevations along them, including the ground truth, the results of both the AAV and the RAV, and elevations derived from a publicly available Digital Terrain Model (DTM), for comparison. From Figure 3.11, several points can be noted:

- Figure 3.11(a), which depict elevations on the bridge, have the least agreement between the ground truth and the DTM (a difference of 4 meters). It seems that due to the low resolution of the DTM, it only partially captures the bridge’s elevation details. This can also be seen in Figure 3.11(b), which depicts the road under the bridge.

- Since the road in Figure 3.11(a) have the most GPS traces, and it is uninterrupted (as there are no traffic lights or other stops along those roads) both variants yield
Figure 3.11: Plots of elevations along several roads of the interchange in Figure 3.9.

good agreements with the ground truth.

- Since Figures 3.11(b) and 3.11(c) have low and intermittent coverage, the AAV behaves poorly. However, the RAV continues to perform well in these cases.

- In the right side of Figure 3.11(b), it is visible that where the AAV has no information, the RAV falls back to a linear interpolation.

In Figure 3.12, we see a histogram of the differences between the ground truth and the results of applying both variants (in road segments where the AAV provides a result). These differences’ aggregates are specified in Table 3.1. From these results, it is clear that the RAV yields elevations that are closer to the ground truth than the AAV — both in terms of mean and variance of the difference.

<table>
<thead>
<tr>
<th>Variant</th>
<th>Data</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAV</td>
<td>all points</td>
<td>+0.59</td>
<td>1.51</td>
</tr>
<tr>
<td>RAV</td>
<td>coverage</td>
<td>+0.47</td>
<td>1.63</td>
</tr>
<tr>
<td>AAV</td>
<td>coverage</td>
<td>+3.56</td>
<td>6.88</td>
</tr>
</tbody>
</table>

Table 3.1: A comparison of the different methods’ results to the ground truth. “coverage” refers to all the points with at least one trace covering it.
Figure 3.12: Histograms of the elevation difference between the tested method variants and the ground truth. The histograms are partially transparent, allowing to see the sections common to both variants (in brown).

One should note that since the GPS receiver is typically not located at the ground level while recording, it is expected that the AAV will results with some offset in z above the ground truth, but this should not affect the variance of the comparison with the ground truth.

3.6 Conclusions and Future Work

In this chapter, we presented an approach for the reconstruction of 3D graphs from 3D positional traces and 2D graphs, and shown that this method can provide good results in the application of road interchanges.

We believe that the RAV method might also be applied to other applications such as medical reconstruction, cave mapping, or flight traffic. In order to adapt this method to other applications, we suggest a variation to the problem as follows. Since we use in this method the graph’s 2D positions only for the purpose of registering the traces, we could think of a variation of the problem in which the nodes’ 2D positions are not known, and instead other information is available to allow us to register the traces to the graph, or the 2D positions are not accurate but can still be used for registration. In that case, instead of calculating the nodes’ elevations, every coordinate of the nodes’ positions is calculated separately and independently in the same manner.

For the application of reconstruction of 3D road network from traces, we suggest a semi-automatic method to provide more topologically-accurate results than existing methods that reconstruct a road network from traces alone. With the evident success of OSM and other commercial 2D maps, we believe that using our method on existing 2D datasets to ameliorate them to 3D can provide with results that are more robust, due to the high quality of these existing datasets. In general, the system would consist of the following steps:

1. Reconstruction of a 2D graph from the univariate traces using an existing 2D
reconstruction method, such as [GIK07, CK09, CGHS10].

2. Augmentation of the 2D graph into 3D using the method presented in this work. This proposed approach also allows one to make use of traces that have only 2D information in the first step, and to achieve reasonably good results with a small amount of 3D traces.
Chapter 4

Amelioration and refinement of Digital Terrain Models (DTMs) from positional traces

In this chapter, we present a method to ameliorate an existing DTM by refining it and augmenting it with data from positional traces. Global DTMs such as the SRTM [FRC+07] are commonly used for 3D maps. However, the lack of details in the DTMs of hilly areas is especially noticeable, as can be seen in Figure 4.3. The method presented here generates a DTM that is based on a low-resolution DTM and reflects the elevation changes in the provided traces. Such traces can be collected easily in residential environments or in highly-travelled parks, where the augmentation is most needed.

4.1 The method

Overall – Our proposed method is comprised of:

1. A refinement (Subdivision) step of the given DTM grid.

2. An augmentation step – assigning elevation to every point in the refined DTM using an approximation of the original DTM and the positional traces around that point.

We calculate the elevation of a point $p$ in the refined DTM as an approximation of both the heights of the traces near $p$ and of the original DTM – manifested as a weighted average. Generally, the closer the trace to $p$, the more weight this traces contributes to the elevation of $p$, while the original DTM contributes a constant weight. Since the contribution of a trace from a great distance is insignificant, we limit the radius of influence of traces. This way, only a limited area around the positional traces needs refinement, and the DTM in “uncovered” areas remains unchanged.
Elevation offset correction

As shown in Chapter 1, GPS traces have global elevation offset. The remedy is to raise or lower the elevation of each trace uniformly, such that the traces will agree on the elevation at each location with other traces and with the original DTM – as much as possible. In Section 4.2 we explain how to find these offsets.

Variations

This method can be modified in several aspects without changing its underlying concepts:

- Instead of using a weighted average of trace points, another approach could be to use a weighted average of the traces, where the elevation (and weight) of each trace is based on the closest point of that trace.

- Since the elevation offsets drift slightly over the course of time, long traces can be fragmented to short sections, with a separate offset for each fragment.

4.2 Implementation

Area of influence

To optimize the run time of the algorithm as well as the representation of the refined DTM, the weight functions are truncated at a minimal value (corresponding to a maximal distance), which beyond it the contribution of a trace point is discarded completely. This maximal distance defines an area of influence for every trace – the set of points within that maximal distance to the trace.

DTM representation

We used the SRTM DTM [FRC+07], which is represented as a rectangular grid. For the function DTM(p), we use a bilinear interpolation on the four corners of the tile p is in. The refined DTM is represented as a two-level hierarchy of rectangular grids – where each level-0 tile that falls under the area of influence of one of the traces is refined to a level-1 sub-grid of a constant size, as can be seen in Figure 4.1. We refined the SRTM DTM from a resolution of 1,200 level-0 tiles per degree of latitude or longitude (tile size of \(\sim 92\) meters on each side, when near the equator) to a sub-grid of size 17*17 level-1 tiles – a resolution of 20,400 level-1 tiles per degree (tile size of \(\sim 5.4\) meters).

Other DTM representations could have been used instead, but this simple approach proved to be enough for our purposes — the results can be rendered in real-time on modern desktop computers, as demonstrated in the interactive web-site mentioned in Section 1.5.
Figure 4.1: Refinement around a trace (blue). We only refine tiles that intersect the area of influence of the trace (green).

Weight scheme

In order to prevent a sharp change in the augmented DTM, that might result from errors in the trace, we enforce a maximum weight value to traces up to a certain distance. A large value for that distance parameter will smooth the augmented DTM, and a small value for that distance parameter will allow noise in the traces to affect the augmented DTM. Other weight values for a weight function $f$ are normalized according to the value at this minimal distance. The examples below use a minimal distance of 3 meters.

$$F(d) = \min \left( 1, \frac{f(d)}{f(3 \text{ meters})} \right)$$

(4.1)

Several weight functions were considered as candidate for this algorithm. In Figure 4.2 we compare some on a synthetic simplified case of several traces that run orthogonally to the 2D plane of the diagram. Every weight function has its own behaviour, but we noticed that a “long tail” of a function is not a desirable property. This can be seen for example with the function $d^{-2}$ where it is possible to see that each cluster of traces affects the other cluster considerably, while the Gaussian function has a smoother result and the clusters of traces are separate.

Elevation offset correction

Let $\text{DTM}(p)$ be the DTM function that maps a position to its elevation, and $T = \{T_i\}$ a set of positional traces as defined in Chapter 3. We will denote $E = \{E_i\} \in \mathbb{R}^{|T|}$ as the elevation offsets of the traces. The elevation of a trace point $t_{i,j} \in T_i$ will be denoted $e[t_{i,j}]$. Given a weight scheme $W(t_{i,j}, p)$ and $W_{\text{DTM}}$ (as the one to be presented in Section 4.2), the augmented elevation of a point $p$, denoted $E_A(p)$, is dependent on the elevation offsets of the trace. It is defined as the following weighted average of the DTM
(a) Gaussian — $\exp(-0.018d^2)$.

(b) Power — $d^{-2}$.

(c) Linear — up to 15 meters.

Figure 4.2: An example of three weight functions. Left: the weight function after normalizing to $f(3\text{ meters})$ and clamping to 1. Right: an example of applying the function in a synthetic example. The base DTM is at elevation 0 with $W_{DTM} = 0.05$.

and the trace points:

$$E_A(p) = \frac{W_{DTM}DTM(p) + \sum_{T_i \in T} \sum_{t_{i,j} \in T_i} W(t_{i,j}, p) (e[t_{i,j}] + E_i)}{W_{DTM} + \sum_{T_i \in T} \sum_{t_{i,j} \in T_i} W(t_{i,j}, p)}. \quad (4.2)$$
To find the optimal elevation offsets $E$, we minimize the (vertical) distance between the traces and the augmented elevations at the trace points:

$$
\sum_{t_{i,j} \in T} \frac{e[t_{i,j}]}{\text{elevation on this trace}} + E_i - \frac{E_A(t_{i,j})}{\text{augmented elevation}} = 0,
$$

for each trace $T_i$ (note that the augmented elevation depends on the elevation offsets $E$). This set of constraints is a linear system of $|T|$ variables and $|T|$ constraints, with a density that depends on the proximity between the traces. The solution to this system provides us with the offsets. Finally, once the $E_i$’s are known, Equation (4.2) is evaluated for every point on the refined grid, providing us with the augmented DTM.

### 4.3 Creating a DTM without augmentation

In cases where an original DTM is not available or has obsolete data due to changes in the environment since the global DTM was captured, it is also possible to use a variation of the proposed method without such original DTM. The changes to the method include setting $W_{DTM}$ in Equation (4.2) to zero, as $DTM(p)$ is not available. The system used to determine the $E_i$ variables, composed of the constraints in Equation (4.3) no longer has a unique solution. When adding a constant value $C$ to each of the elevation offsets $E_i$, it is cancelled in Equation (4.3). Using the LSQR algorithm to solve the system provides the result in which $\sum E_i^2$ is minimal. However, the system becomes numerically unstable as weights tend to decrease to zero between each pair of separate traces. To boost the stability, it is possible to explicitly add another constraint or set of constraints to enforce a unique solution. For example, it is possible to add the following set of constraints: $E_i = 0$.

### 4.4 Results

We present the results of this method in two cases. The first case is using the SRTM as an existing low-resolution DTM and several traces recorded while walking in Nesher, Israel. The results for this case are shown in Figure 4.3.

In the second case, we didn’t use an existing DTM, and only used traces recorded while walking in Beit Oren forest, Israel and driving on Road 721 next to it. The results for this case are shown in Figure 4.4. In addition, these results demonstrate the effect of the trace elevation offset correction. For example, without the elevation offset correction, it is visible that the road is raised unnaturally, due to several traces that leave the road at this point.
4.5 Conclusions and Future Work

In this chapter we introduced a method to augment existing DTM to a higher resolution using GPS traces, or creating a DTM using only GPS traces.

As future work, it might be useful to use signal processing techniques together with deeper knowledge of the GPS receiver algorithms in order to process and clean the GPS traces before using them. This should allow to overcome some of the current problems with GPS traces such as noise and inaccuracies.
(a) 2D map of the area and the traces used for generating the DTM.

(b) 3D render of the generated DTM without trace elevation offset correction.

(c) 3D render of the generated DTM with trace elevation offset correction.

(d) 3D render of the SRTM DTM. This DTM was not used for the generated DTMs in (b) and (c).

Figure 4.4: An example of creating a DTM using only GPS traces, in Beit Oren forest, Israel (32.7275N, 35.0142E). To see the elevations, the 3D renders have iso-lines every 1 meter of elevation with bold lines every 10 meters.
Chapter 5

Estimating buildings’ heights using smartphones

5.1 Introduction

In this chapter, we aim to reconstruct 3D models of buildings, by extruding their top-view 2D outlines from the 2D map, using their heights (see Figure 5.1). Naturally, this scheme is suitable only for buildings that have a linearly extruded shape – with a strictly vertical walls and a uniform height.

As detailed in Section 2.3, there exist many methods to estimate the height of a building. We will constraint ourselves to methods in which the input can be acquired easily by a user with a smartphone – the phone’s accelerometer and magnetometer can be used to acquire the direction to an object, and the GPS receiver can be used to acquire the position of the user. We also assume we have online access to a 2D map.
(where we have the 2D outline of the building to extrude). Without aerial photography, LIDAR, or a laser rangefinder, most of the methods presented in Section 2.3 are not available. Others require more effort from the user - capturing the building from several points of view, or having a direct access to the building.

This chapter presents a new method for estimating the height of a building, and compare it to the tangent method, as presented in [LML13]. We use the tangent method for comparison because it complies with all the requirements above. The rest of this chapter is organized as follows. Section 5.2 reviews the tangent method used for estimating an object’s height – using two measured angles and the distance to the building. Section 5.3 describes two versions of our proposed estimation method – based on four vectors, and in Section 5.4, we compare the results from our methods to the tangent method, and verify them using the ground truth. Section 5.5 presents the implementation of our method in Android-based smartphones, and finally, Section 5.6 discusses some future work and concludes this chapter.

5.2 The Tangent method

The Tangent method, also presented in [LML13], calculates the height of an object using the measured angles and the distance to that object, by means of simple trigonometry, as seen in Figure 5.2. Given two angles $\alpha, \beta$ to the top and to the base of the object above and below the horizontal plane, and the user’s distance from the object $d$, the object’s height $h$ is calculated as follows.

$$h = d \cdot (\tan(\alpha) + \tan(\beta))$$  \hspace{5cm} (5.1)

![Figure 5.2: Diagram of the tangent method.](image)
Many implementations of this method \(^1\) allow to calculate the user’s distance \(d\), based on the user’s height \(u\) and the angle to the ground \(\beta\), assuming horizontal surroundings, as described in Equation (5.2).

\[
d = \frac{u}{\tan(\beta)}
\] (5.2)

However, in our implementation, to avoid the (usually incorrect) assumption that the user is standing at the same elevation as the base of the building, we use the user’s GPS location and the building’s map information to infer the distance.

### 5.3 The four directions method

In this method, we estimate a building’s height based on a 2D map of the building and four direction vectors from the user to the corners of the building’s face. In this chapter and in the implementation, we limit ourselves to a single planar face for the sake of simplicity, but this method can be applied to more generalized cases such as where the two right-side points are not on the same face as the two left-side points.

![Diagram of the four direction vectors](image)

Figure 5.3: A diagram of the four direction vectors from the user to the corners of one of the building’s faces. The drawn lines reach the corners for illustrative purposes - but the actual distances to the corners are unknown.

We construct a linear system with the following constraints (for each corner \(i\) – top/bottom left/right):

\[
U + d_i D_i = P_i \quad i \in \{tl, tr, bl, br\}
\] (5.3)

where \(U\) is the user position (a 3D vector variable), \(d_i\) is the distance of the user from

---

\(^1\) Examples include Smart Measure by Smart Tools co. (available in [http://play.google.com/store/apps/details?id=kr.sira.measure](http://play.google.com/store/apps/details?id=kr.sira.measure)).
the face corner (one variable for each corner), $D_i$ are the directions towards the corners (known vectors), and $P_i$ are the corners’ positions. The positions of the bottom corners are known (see below) and the top corners depend on another variable – the building’s height $h$. Each corner constraint is represented as 3 linear constraints - one for each positional component. Specifically:

$$U^x + d_{tl}D_{tl}^x = P_{tl}^x$$
$$U^y + d_{tl}D_{tl}^y = P_{tl}^y$$
$$U^z + d_{tl}D_{tl}^z = h$$

$$U^x + d_{tr}D_{tr}^x = P_{tr}^x$$
$$U^y + d_{tr}D_{tr}^y = P_{tr}^y$$
$$U^z + d_{tr}D_{tr}^z = h$$

$$U^x + d_{bl}D_{bl}^x = P_{bl}^x$$
$$U^y + d_{bl}D_{bl}^y = P_{bl}^y$$
$$U^z + d_{bl}D_{bl}^z = 0$$

$$U^x + d_{br}D_{br}^x = P_{br}^x$$
$$U^y + d_{br}D_{br}^y = P_{br}^y$$
$$U^z + d_{br}D_{br}^z = 0$$

(5.4)

where the variables are $h$, $U^x$, $U^y$, $U^z$, $d_{tl}$, $d_{tr}$, $d_{bl}$, $d_{br}$. This system has a total of 8 real variables, and 12 linear equations and is thus over-constrained. The least-squares solution to this linear system provides us with the building’s height $h$.

In order to coerce all element into a single coordinate system, the 2D locations of the building’s face are converted from the map’s coordinate system to a local canonical Euclidean space where “East” corresponds to $(1, 0, 0) \in \mathbb{R}^3$, “North” corresponds to $(0, 1, 0) \in \mathbb{R}^3$, and “Up” corresponds to $(0, 0, 1) \in \mathbb{R}^3$. The X and Y coordinates of the corners are deduced from the geodetic coordinates (Latitude and Longitude) of the face, based on the 2D map. The building is fixed to be at elevation of 0 (Z component of the bottom corners) in this canonical space, so the user’s elevation $U^z$ is relative to the building. The Z component of the top corners is the height of the building – $h \in \mathbb{R}$.

**Adding a location constraint**

As will be seen in the comparisons below in Section 5.4 — the tangent method is sensitive to errors in short distances (due to GPS location error), and the 4 directions method is sensitive to errors in long distances (due to a larger influence of angular error). In order to provide a solution that doesn’t have these drawbacks, we propose to add a positional constraint to the linear system, based on the GPS location of the user. Effectively, this adds two more constraints to the system – in $x$ and $y$. Specifically:

$$U^x = U_x$$
$$U^y = U_y$$

(5.5)

A constraint for the elevation of the user is not added because the elevation of the base of the building is not known, and because the elevation of the GPS is very inaccurate, as was discussed in Section 1.2. We prefer to introduce this extra information to the
system in the form of more constraints and not less variables, in order to allow the solver to assign errors to these constraints.

5.4 Comparison

We compared the methods that were presented: The tangent method, Four directions, and Four directions with location constraint. For the comparison, the tangent method was applied twice in each test: on the left side and on the right side of the building’s face. The two resulting heights were averaged for a combined height that is based on the same 4 direction vectors.

First, we compare these methods on a real dataset of 36 samples, captured at the Technion campus, in front of the south-east face of the computer science faculty building (32.77744N,35.02155E) at various distances and angles. The ground truth of the building height is 23.2m, measured manually. For the purpose of this experiment, the building’s face is from the bottom of the 1st floor to the bottom of the 7th floor, also shown in Figure 5.8b. The results can be seen in Figure 5.4 and in Table 5.1.

![Figure 5.4: A scatter plot of the results of the different methods on the “Technion” dataset (32.77744N,35.02155E). The ground truth is marked as a dashed line.](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tangent method</td>
<td>23.39m</td>
<td>15.56m²</td>
</tr>
<tr>
<td>Four directions</td>
<td>27.07m</td>
<td>4.72m²</td>
</tr>
<tr>
<td>Four directions + Location constraint</td>
<td>23.04m</td>
<td>11.43m²</td>
</tr>
</tbody>
</table>

Table 5.1: A comparison of the results from the different methods on the “Technion” dataset (32.77744N,35.02155E). The ground truth is a height of 23.2m.

Next, we tested the methods on synthetic datasets – comprising of a virtual building face with a height of 30 meters and a width of 30 meters, with the ‘user’ in front of the center of the face at an elevation of 0 (as the base of the building), in a variable distance $d$ and at some angle of $\theta$.

For each sample of $d$ and $\theta$, a Monte-Carlo simulation was applied. The clean data was randomly jittered 5000 times. Each jittered sample has a location error of up to 5 meters in the xy plane and a 3° rotation of each direction vector around a random
rotation axis, independently. Figures 5.6 and 5.7 show the results of the comparison. From Figure 5.6, it is visible that the four directions + location constraint method provides the lowest variance, and a consistent and correct mean value. In contrast, the tangent method’s results deviate from the correct value at small distances, and the basic four directions method’s results deviates from the correct value at larger distances. Also, from Figure 5.7 it is visible that while the basic four directions method behaves poorly when capturing the building from its side (at $\theta \to 90$), the four directions + location constraint method keeps at the correct mean value and at a small variance.

![Diagram of the synthetic dataset, top view.](image)

Figure 5.6: Comparison between the methods – The results of the monte-carlo simulation on the synthetic dataset in a varying distance $d \in [0m, 140m]$ and at a constant angle $\theta$. Each pair of graphs presents the mean and variance calculated over each data point.

(a) $\theta = 0^\circ$.

(b) $\theta = 45^\circ$. 
Figure 5.7: Comparison between the methods – The results of the monte-carlo simulation on the synthetic dataset in varying angles $\theta \in [0^\circ, 90^\circ]$ and at a constant distance $d = 45m$. Each pair of graphs presents the mean and variance calculated over each data point.

### 5.5 Application Implementation

Our implementation is an Android application that augments buildings in OpenStreetMap. Figure 5.8a shows how the application is using GPS and OSM data to detect buildings nearby, and the orientation to select a specific face of a building. As the user points the smartphone at the building, the application highlights the nearest intersecting 2D edge on the map. Then the user clicks an on-screen button to confirm the selection. Figure 5.8b shows how the application is using the camera to provide feedback to the user about the captured directions. The user points the smartphone towards one of the corners such that it is visible on the virtual crosshairs, and taps on the appropriate quarter of the image. A thumbnail image appears on the corner of the screen to indicate the user that this direction was acquired. When the user clicks the *Calculate* button, the data is sent to our server, and the calculated height is made available online. To summarize, our scheme is as follows:

1. The user selects an edge of a 2D-mapped building.
2. The user captures direction vectors to the corners of that building’s face.
3. A calculation takes place, which results with an estimate of the building’s height.
4. The building’s height is fed back into the map.
5. Later, when viewing the 3D map (in our OSM3D site introduced in Section 1.5), the building’s 3D model is recreated by extrusion of its 2D polygon.

**Acquiring a direction vector in smartphones**

To acquire the directions, we exploit the accelerometers and magnetometers that are present in modern smartphones. When at rest, the accelerometer measures a 3D
vector pointing upwards, due to the gravitational field of the earth. The magnetometer measures the direction (and strength) of earth’s magnetic field. Assuming that magnetic interferences are negligible, a magnetic declination correction is applied to the “magnetic north”, obtaining the “true north” direction. While using the accelerometer and magnetometer allows us to detect the current orientation, we only need the forward direction.

5.6 Conclusions

In this chapter, we introduced a new method for estimating a building’s height from a nearby point of view by capturing the angles to the building’s corners. We compared our method to a commonly used existing method and formulated a hybrid method that enjoys the benefits of both – as demonstrated in the synthetic simulation and evidenced by real recordings.
As a future work, we suggest to extend the implementation to support non-planar and complex faces. This can be done because we only need two points on the building’s 2D map, where the building has the same height and the same base level. This will allow to capture the height in more cases where the building’s geometry is complex or where there are occlusions. In addition, it is also possible to use more than two such points, in order to improve the accuracy.
Chapter 6

Conclusions and future work

In this work, we presented three independent approaches to augment 2D maps to 3D, using data that can be easily collected using a smartphone. These methods cover different aspects of the map: roads networks, terrain, and buildings. In each chapter, we discussed the possible ways to improve the method, apply it to different subjects, or add additional features to it.

In addition to the conclusions of each chapter, we would like to also present a suggestion for integrating those methods into a unified system that reconstructs all aspects of a 3D map. Such system will subscribe to a data source of recorded GPS traces and smartphone measurements, and reconstruct a fully-featured 3D map based on an existing 2D map such as OpenStreetMap. This work can be viewed as the foundation for such a system.

Such system will have to identify locations of interest that should be subjected to augmentation. Regarding the methods presented in this work - those locations would be interchanges and bridges with enough trace coverage, as well as hilly areas with dense-enough coverage. For a long-term use, this system will also need to consider the date of recording of each trace, as well as the date of changes in the map, in order to ignore outdated traces. Finally, method-specific improvements could also be beneficial, such as outlier removal when aggregating several measurements. This can be done in several points in the presented algorithms. For example: in Chapter 3 – when calculating the average of the elevation difference (Equation 3.4) ; in Chapter 4 – when averaging the elevations of points $e[t_{i,j}]$ around a single $p$ ; and in Chapter 5 - when averaging a building’s height from several users. The implementation details of the above ideas currently remain as open questions.
Bibliography


Kihwan Kim, Jay Summet, Thad Starner, Daniel Ashbrook, Mrunal Kapade, and Irfan Essa. Localization and 3d reconstruction of urban


[Nok12] Nokia 3D maps. http://here.com/0,0,3,0,0,3d.day, 2012. [Online; accessed Jan-2013].


Matthias UDEN, Arne SCHILLING, Ming LI, Marcus GÖTZ, and Alexander ZIPF. Creating a worldwide 3d globe from user-generated data.


The building

Some of the most important data for the visualization of the results of the methods he employs, is displayed in a table that includes all the buildings of the city. The first screen allows the selection of an area in the map to view. After selecting an area, the server sends the map, and it is displayed in WebGL technology. The code that displays the 3D environment is written in a part of the project that is presented at the Technion - Computer Science Department - M.Sc. Thesis.

OpenStreetMap

SRTM

also global height map.

The SRTM elevation data is used to render the height of the buildings in the map.
The project provides a service that allows the receipt of data from a certain area on a map.

API

Internet and API methods for applications.

GPS

In addition to maps, use is also made of a database of recorded GPS tracks, which are also available for free use.

The first method presented in this narrative describes an algorithm for retracing roads in three dimensions, using GPS tracks recorded at the junction. In this method, GPS data from a two-dimensional map is used. The altitudes of GPS correspond to the track points on the road on the map. Despite this, the altitude is not accurate as it is compared to the actual position, and therefore we cannot rely on it absolutely.

Therefore, the data from the track is used to calculate the altitude relative to the same track on itself. Despite the fact that each track is recorded on a separate track in the junction, the method uses the fact that the junction is connected between the different tracks.

The solution provides the altitude along the track, but even if there are gaps in the recorded data, the solution still works. The method is linear for segments, and the height difference is calculated from the stations on the track. In this way, the solution is more accurate than just the map.

The second method describes the possibility of retracing a height map from GPS data in an area. In contrast to the first method, this method also allows the use of an existing height map to display the changes in the area.

The main drawback to the first method is that it requires a more expensive and complex equipment. This method also has a drawback, but it provides more accurate results.

The third method calculates the altitude of buildings. It is similar to an older method for calculating the height of trees using trigonometry, but it uses a different approach. It calculates the height of the building using the GPS data from the map.

This solution offers a better solution compared to the existing method, and it is also more accurate. In the end, an application was developed for Android that allows the user to display the height of buildings on the map.
M.Sc. Thesis

To summarize, maps are a world of connected nodes, serving as tools for navigation, both on roads and in space. A map can be either a physical map or a digital one, allowing for algorithmic calculations of routes from one point to another. When a map is digital and presented in three-dimensional format, it includes information about the terrain and objects on the ground, such as buildings. This enables virtual tours in places that are otherwise inaccessible, and helps in analyzing geographical data, such as the flow of water, movement of bodies, and more. Using a digital map, the three-dimensional display can help the user to understand their path.

A common method for creating three-dimensional maps is based on aerial photography and automatic building models, including ground and surrounding objects, such as buildings, trees, and even cars. However, this method, although based on high-resolution photos, has its limitations, such as the high cost of production. The method also has limitations in terms of navigation and searching.

To overcome these limitations, other methods for creating three-dimensional maps have been developed, including the use of geographical databases such as OpenStreetMap (OSM). This project aims to provide free geographical data to anyone in the world. The data is updated based on the contribution of the volunteers. The map is three-dimensional and consists of a collection of geographical points and three-dimensional objects.

In the implementation of the methods described in this thesis, we use existing sources of information, such as the OSM project, which provides geographical data without any restrictions. The data is provided by volunteers around the world, with updates based on the activities of these volunteers. The map is three-dimensional and consists of a collection of geographical points and three-dimensional objects.

The methods described in this thesis rely on modern devices, with the help of GPS and other devices, such as magnetometers (magnetometer) and accelerometers (accelerometer) and gyroscopes (gyroscope) for orientation. GPS can provide data about the current location, and the magnetic field for orientation. The combination of these devices can provide accurate and up-to-date geographical information.
המחקר בוצע בנחייתו של פרופסור גרשון אלבר, באוניברסיטת תל אביב.
שדרוגים תלת מימדיים
של מפות דו מימיות

היבר על מחקר

לשם مليו חלקי של הדרישותכבכל התואר
פגיסרטERM adultery במודיעי המחשב

ניק הרשק

רותם למטני תכני – מוכן טכנולוגי לישראל
סיןומ скороלה חפץ יוני 2015
שדרוגים תלת מימדיים של מפות דו מימדיות

 Nir Hershkow