Kartta Labs: Collaborative Time Travel

Sasan Tavakkol
Google Research
tavakkol@google.com

Feng Han
Google Research
bladehan@google.com

Brandon Mayer
Google Research
bmayer@google.com

Mark Phillips
Google Research
embeepea@google.com

Cyrus Shahabi*
Google Research
cshahabi@google.com

Yao-Yi Chiang
University of Southern California
yaoyic@usc.edu

Raimondas Kiveris
Google Research
rkiveris@google.com

Abstract

We introduce the modular and scalable design of Kartta Labs, an open source, open data, and scalable system for virtually reconstructing cities from historical maps and photos. Kartta Labs relies on crowdsourcing and artificial intelligence consisting of two major modules: Maps and 3D models. Each module, in turn, consists of sub-modules that enable the system to reconstruct a city from historical maps and photos. The result is a spatiotemporal reference that can be used to integrate various collected data (curated, sensed, or crowdsourced) for research, education, and entertainment purposes. The system empowers the users to experience collaborative time travel such that they work together to reconstruct the past and experience it on an open source and open data platform.

1. Introduction

The ultimate goal of Kartta Labs is to create a collaborative time travel experience; think of Google StreetView (or Google Earth), but with the ability to go far back in time [1]. As with StreetView, our system needs to run on top of a map service; however, any map service we use must support a temporal dimension. Therefore the first step in this project is building a modular and scalable system to collect, process, and serve map data indexed by time and space. The Maps project consists of a stack of web applications that crowdsources collecting [2] and vectorizing historical maps. The vectorized spatiotemporal data are open sourced to promote the collaboration among the community. These vectorized data are also served online using a tile server[1] and visualized within a map renderer website. We previously introduced some parts of the Maps module in [3].

The second step in this project is to reconstruct the historical buildings as 3D models. To this end, we introduced an image processing pipeline in [4] where the first step was an image segmentation job to identify buildings facades. The identified facades were then fed to rectification [5] and inpainting [6] jobs. The output image was then applied on a face of a cuboid 3D mesh as a texture. In this paper, we introduce our improved pipeline which extracts 3D features of the facades and incorporates accurate footprints from historical maps. Our pipeline segments and parses a single view image of the building to procedurally reconstruct a 3D mesh of its facade. Subsequently, this facade is incorporated into one face of a 3D extrusion of the building footprint. The result is stored as a 3D mesh model in an online repository, accessible through a public API.

We follow the principles of systems design to layout the architecture of Kartta Labs and build a modular system. The modules and their sub-modules are primarily defined based on their input and output. The output of one sub-module becomes the input to another, creating an organic workflow. We also outline the storage and processing requirements of each module and briefly discuss their implementation. As we mentioned earlier, our system consists of two major modules: Maps and 3D models. Each module consists of smaller sub-modules. Figure 2 shows the major modules, their sub-modules, and the workflow. We briefly explain each sub-module in this paper. Most of the sub-modules are open sourced and as they mature, are added to our GitHub organization (https://github.com/kartta-labs). We use Kubernetes to deploy and manage our tools, which makes it easy for others to redeploy our suite of applications either for development or production purposes. We currently run a full instance of our system on Google Cloud Platform accessible at https://re.city.

*The work was performed when the author was a visiting researcher at Google.

[1]https://www.ogc.org/standards/wmts

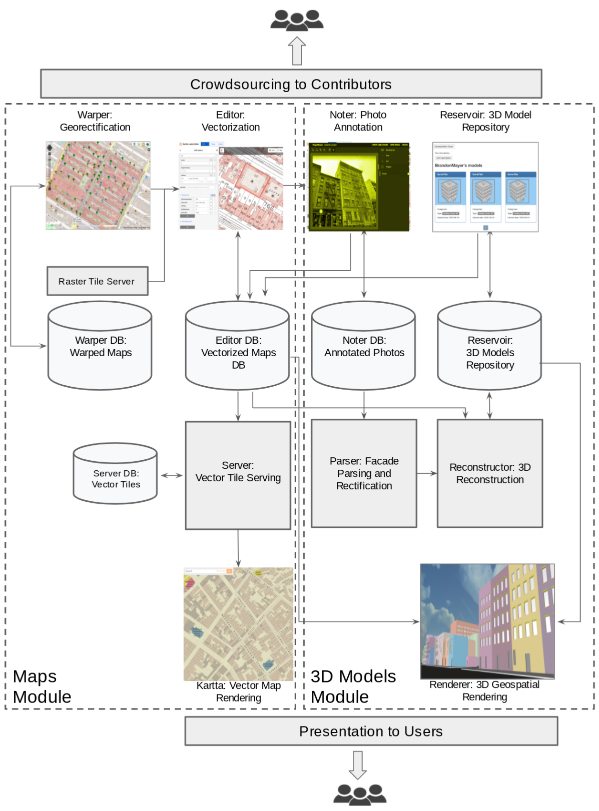

Crowdsourcing to Contributors
Warper: Georectification
Editor: Vectorization
Noter: Photo Annotation
Reservoir: 3D Model Repository
Raster Tile Server
Warper DB: Warped Maps
Editor DB: Vectorized Maps DB
Noter DB: Annotated Photos
Reservoir: 3D Models Repository
Server DB: Vector Tiles
Server: Vector Tile Serving
Parser: Facade Parsing and Rectification
Reconstructor: 3D Reconstruction
Maps Module
Kartta: Vector Map Rendering
3D Models Module
Renderer: 3D Geospatial Rendering
Presentation to Users

Figure 1. Sub-modules of the Kartta Labs software system.

2. Related Work

A geographical backdrop (e.g., maps) to be used as a reference for integration of other datasests has always been of interest to researchers. This is evident by the numerous mashups developed on top of Google Maps. As a natural extension to this spatial reference, some use cases consider a dynamic spatiotemporal reference system. For example, Gapminder (www.gapminder.org), has a map feature that allows the user to geospatially visualize a statistical value (e.g., population) and navigate it through time using a time-slider feature. Another example is HistoryPin, a crowdsourced archive of historical media. HistoryPin enables users to ”pin” their media (e.g., images) to Google Maps and set the time to which they belong. Kartta Labs can act as a platform for such systems, providing accurate historical geospatial data over time as reference.

Endeavors to construct 3D worlds have been pursued for decades in academia and industry. Virtual worlds are examples of such endeavors that became relatively popular in 1990’s and are recently gaining traction again, thanks to the advances in virtual reality devices. Examples of virtual worlds are Active Worlds, Second Life, World of Warcraft among others [7]. The geography of these virtual worlds are often a fantasy world. For example, Active Worlds consists of hundreds of fantasy worlds, where users can explore 3D virtual environments built by others or build their own 3D content. Active Worlds has inspired several academic efforts in education [8, 9] and has served as a platform for data collection for various studies [10]. In contrast, Kartta Labs is meant to reconstruct the real world in time and in space.

Esri’s CityEngine[2] is another related work to Kartta Labs. CityEngine takes a procedural approach to construct 3D urban environments. It can procedurally generate 3D models given footprints of buildings. While the generated 3D models look compelling and consume metadata such as buildings height, they are not based on real world imagery and therefore the building facades are not detailed. CityEngine does not natively support a time dimension or tiling. Indeed, applications like CityEngine can be used to generate 3D models for Kartta Labs.

Another closely related work to Kartta Labs is 3DCityDB [11], a free 3D geo-database solution for 3D city models based on CityGML standard issued by the Open Geospatial Consortium (OGC). 3DCityDB does not natively support a historical time dimension. As we discuss in Section 8 we are considering using 3DCityDB to host city 3D models of Kartta Labs in the future.

Google Earth is perhaps the closest application to what we envision. Google Earth renders a 3D representation of Earth primarily based on contemporary satellite imagery. In addition to represent the surface of earth in 3D, Google Earth, also shows 3D building models in some cities. At the beginning, a community was formed around Google Earth that used applications such as SketchUp and Building Maker to manually create the 3D buildings, resembling our crowdsourcing approach to the problem. However, it now uses auto-generated 3D models. Google Earth also enables users to explore historical satellite imagery going back a few decades. However, it does not represent the historical satellite imagery in 3D, nor does vectorize them.

To the best of our knowledge, Kartta Labs is the only system that is capable of not only vectorizing historical maps, but also reconstructing them in 3D across time. Most of the similar solutions are focused on contemporary data. Others either deal with only maps or 3D reconstruction [4]. Furthermore the most compelling solutions are based on proprietary code

[2]https://en.wikipedia.org/wiki/CityEngine

and data. Kartta Labs differentiates itself from the prior work by combining the features of several similar applications and providing them as an open source and open data platform.

3. Design

We designed Kartta Labs following the principles of systems design to create a modular and scalable[3] software system. A modular design was required for Kartta Labs for several reasons. First, Kartta Labs mission is quite complicated. Therefore, as any software, a modular design let us divide the problem to smaller pieces and solve them independently. More importantly, our modular design enables us to adopt open source solutions for some of the modules. Furthermore, having a well defined interface between modules let us have more than one implementation for a module. For example, our photo annotation module has two implementations, one based on crowdsourcing and one based on artificial intelligence. Finally, a modular design makes Kartta Labs scalable.

We define our system and its modules based on their inputs and outputs, enabling us to define clean interfaces between modules. The input to Kartta Labs, as a system, is historical photos and maps. The output is a 3D representation of world with a time dimension. In order to process the input and create the output, Kartta Labs may rely on intermediate inputs such as geotagging and georefrencing of the input images and maps by humans.

Kartta Labs consists of two major modules: Maps and 3D models. In Section 4 we describe the Maps module and its sub-modules. The input of this module is a scanned historical map and the output is the same map, but in vector format. In Section 5 we layout the architecture of our 3D models module. The vector historical maps generated by the Maps module becomes the input to 3D models module. Furthermore, the 3D models module takes in historical urban photos as its input. The output of this module is the overall output of Kartta Labs: a 3D representation of world with a time dimension. We briefly explain the sub-modules of Maps and 3D models in their corresponding sections.

Karrta Labs is implemented in several different languages using different technologies and development frameworks. This is because we leveraged available open source solutions that are developed within different communities and perhaps for unrelated purposes. However, we unified the deployment of all these applications using Docker containers[4] and Kubernetes[5]. This deployment design not only makes our system a portable solution, such that it can be deployed locally or on different cloud platforms (e.g., Google Cloud), but also enables it to scale out[6] and scale up[7] on demand.

We use Google Cloud Platform (GCP) to deploy Kartta Labs. In addition to its Kubernetes Engine we use GCP's managed databases and storage to leverage its scalability, security, and reliability. We also use Google Clouds Functions, a serverless execution environment for running simple, single-purpose cloud services, for some of our simple services. Nevertheless, Kartta Labs can be deployed on other cloud platforms or locally on a single machine for development purposes.

4. Maps

The Maps module aims to create a map server with a time dimension, we envision OpenStreetMap[8] (OSM) with a time slider to navigate the time dimension. We have developed and stacked a set of open source tools that are used to collect and vectorize scanned historical maps, via crowdsourcing, and serve them as vector tiles[9]. Maps is made up of a suite of tools that allow users to upload historical maps, georectify them to match real world coordinates, and then convert them to vector format by tracing their geographic features. These vectorized maps are then served on a tile server and rendered as maps in the browser.

The input of the Maps module is a scanned historical map and the output is the same map, but in vector format. The entry point of the Maps module is a web application, called Warper, that enables the users to upload historical images of maps and georectify them by finding control points on the historical map and corresponding points on a base map. Another web application, Editor, allows users to load the georectified historical maps generated by Warper as the background (through a raster tile server) and then trace their geographic features (e.g., building footprints, roads, etc.). These traced data are stored in OSM vector format. They are then converted to vector tiles and served from a vector tile server, dubbed as Server. Finally, our browser map renderer, called Kartta, visualizes the spatiotemporal vector tiles allowing the users to navigate space and time on historical maps. We briefly discuss the design of each Maps sub-module next.

[3]Scalability is the ability of a system to handle more work by adding more resources

[4]https://www.docker.com/resources/what-container

[5]https://kubernetes.io/

[6]Scaling out or horizontal scaling is adding more nodes (e.g., virtual machines.) to a system to handle more work

[7]Scaling up or vertical scaling is adding more resources to a single node by, for example, increasing its number of CPU's, memory, or disk storage.

[8]https://openstreetmap.org

[9]https://en.wikipedia.org/wiki/Vector_tiles

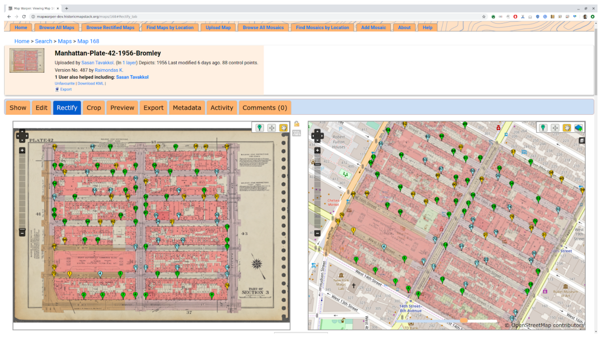


Figure 2. A screenshot of Warper showing how a map is georectified.

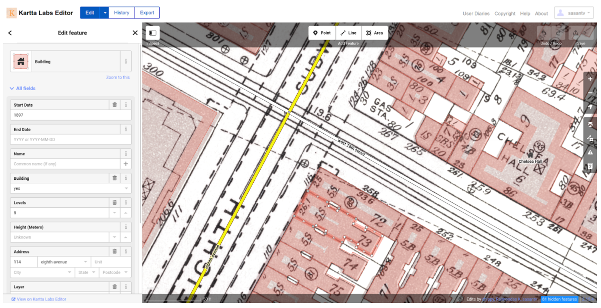


Figure 3. A screenshot of the Editor application used to trace footprints on a historical map.

4.1. Georectification

Warper is an open source web application that crowdsources collection and georectification of historical maps. It is based on the MapWarper[10] open source application. The input of Warper is scanned historical maps that users may upload. Warper makes a best guess of an uploaded map's geolocation by extracting textual information from the map and using algorithms outlined in [3, 12]. This initial guess is used to place the map roughly in its location and let the user georeference the map pixels by placing pairs of control points on the historical map and a reference map.[11] Given the georeferenced points, the application warps the image such that it aligns well with the reference map. This georectified map is the output of this sub-module. Warper also runs a raster tiles server that serves each georectified map at a tile URL. This raster tile server is used to load the georectified map as a background in the Editor application that is described next. Figure 2 shows a screenshot of Warper where a historical map of New York is georeferenced against a contemporary map of the same area from OSM.

4.2. Vectorization

The Editor is an open source web application from OSM[12] stack of tools that we have modified to fit in our system. Editor lets users extract vector geometries (output) from georectified images (input) and then stores them in a database. The vector data include information such as buildings footprints, roads, addresses, names and dates, as well as ”start date” and ”end date” fields which represent the time dimension; a feature is considered to exist in time between these two dates. A screenshot of the Editor web application is shown in Figure 3.

[10]https://github.com/timwaters/mapwarper

[11]Note that advanced 2D georectification processes, such as [13], can be used to automatically identify control point pairs and can be later added to Warper.

[12]https://github.com/openstreetmap

4.3. Tiling

To support the development of interactive map applications with a time dimension, we serve our spatiotemporal map data (input) as a collection of Mapbox vector tiles[13] (output) using the Tegola[14] vector tile server. We call this application Server, for short. This service makes tiles available using the standard OSM tile naming convention[15].

In our current implementation the time dimension is included as an attribute on the tile data; tiles are addressed by space (and zoom level) onlyClient applications can present a view of the data for a specific moment in time by using the ”start date” and ”end date” attributes to filter out features not present at that moment.

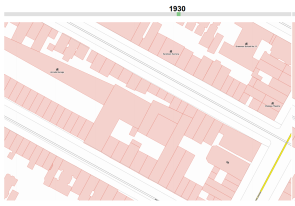
4.4. Visualization

The endpoint of the Maps module is a time-aware, interactive map application, called Kartta. Kartta works like any familiar map application (e.g., Google Maps), but also has a time slider so the user can choose the time at which they want to see the data. By moving the time slider, the user is able to see how features in the map such as buildings and roads have changed over time. The input to Kartta is a set of vector tiles and the output is rendered images showing those tiles in a given map style. Note that the images are rendered client-side, i.e., in the browser. Figure 4 shows two snapshots of this application in two different times around the Google NYC building (111 Eighth Avenue, New York, NY). Generating vector tiles, as opposed to raster tiles, was required to provide a seamless navigation of the time

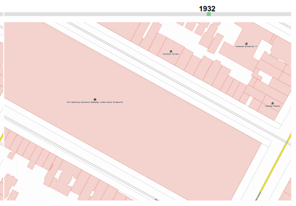
[13]https://docs.mapbox.com/vector-tiles/reference/

[14]https://tegola.io/

[15]https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames



(a) 1930



(b) 1932

Figure 4. Screenshots of the Kartta showing the area of the Google NYC building in Manhattan, before (a) and after (b) it was built.

dimension with any granularity.

5. 3D Models

The 3D Models module aims to reconstruct the detailed full 3D structures of historical buildings using the associated images and maps data, organize these 3D models properly in an online repository, and render them on the historical maps with a time dimension. The input to this module is historical images and vector historical maps, and the output is a 3D representation of an area across time.

In most cases, there is at most one historical image available for a building, which makes the 3D reconstruction an extremely challenging problem. To tackle this challenge, we developed a coarse-to-fine reconstruction-by-recognition algorithm as illustrated in Figure 5. The footprint of the building is extruded upwards to generate the coarse 3D structure, using any available metadata (e.g., number of floors) to set the extrusion height. Then, the historical image is annotated, either by crowdsourcing or automated algorithms, and the result is used to generate 3D details (e.g. windows, entrances, stairs) for the visible facades of the building from the street ground level. We discuss each sub-module next.

5.1. Photo Annotation

We need to annotate the historical photos to identify building facades and then to identify the structural details of each facade. We rely on crowdsourcing and machine learning algorithms. To crowdsource the annotation task, we developed a web application, called Noter. It consists of a frontend based on the open source tool MakeSense [16] connected to a backend we developed in Python. The application allows users to upload photos of historical buildings (input) or browse the photos uploaded by others. Users can then annotate (output) the

[16]https://github.com/SkalskiP/make-sense

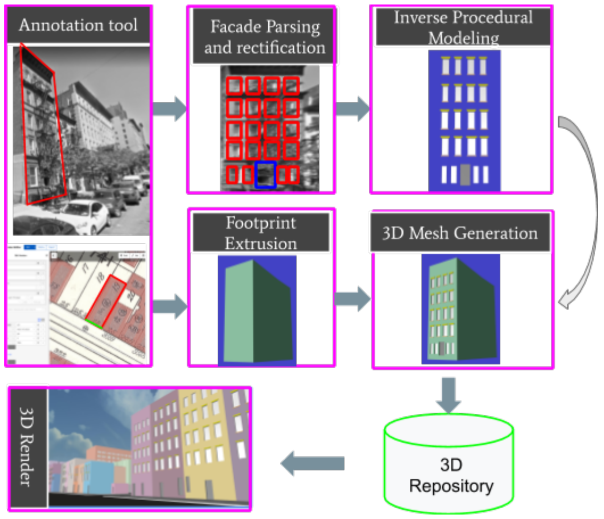


Figure 5. System diagram for the 3D models module.

photos given a preset of labels (facade, window, door, etc.). An ID is assigned to each annotation piece such as facades. The facade ID is used to associate that facade with part of a footprint in the Editor application. This process geotags that specific facade but it can also be used to roughly geolocate the rest of the facades in the same photo. If a facade is matched with another one in a different photo as being same, then the location information can be propagated between those photos. We are working on a spatial reasoning algorithm to construct a graph of facades and propagate the location information from one facade to others in the same sub-graph [14]. Such an algorithm can significantly facilitate geotagging historical photos.

5.2. Facade Parsing and Rectification

Facade parsing is the process of decomposing a facade into its constituent components, e.g., windows, entries, doors, and stair. We call our facade parsing sub-module Parser. The input to this sub-module is the photo of a building facade and the output is a rectified photo of the same facade with its components fully annotated. We take a supervised learning approach. A corpus of approximately 5,000 images were annotated by human annotators with over 500,000 boundary-level instance annotations.

We trained binary FasterRCNN neural networks using the facade component annotations for each target semantic class which are used to localize bounding-box level instances in new images. We used binary FasterRCNN rather than a single multi-class detector due to our observations of superior performance of a suite of binary classifiers compared to the multiclass version on held out data.

While extremely accurate, the FasterRCNN model is only capable of producing axis aligned (relative to the image frame) bounding box localizations requiring a rectification post-processing step. We have had success training and integrating semantic segmentation models including DeepLab [15] into the Kartta Labs Facade parsing pipeline but defer discussions of semantic segmentation for later publications. Figure 6 visualizes the output of the facade parsing pipeline prior to rectification and 3D reconstruction. The facade Parsing pipeline is written in C++ using the open-source MediaPipe[17] framework. The MediaPipe framework allows for parallelization and thread optimization of image processing routines on a per-process basis.

After parsing an image into facade components, the next step in the pipeline is to extract each facade primitive within the target (annotated) facade and normalize them with respect to camera viewpoint. We use a vanishing-point based rectification process to bring all components within each facade into frontal view. Man-made objects like facades have strong regularities and follow architectural principles. We use predefined grammar rules to regularize the parsing results on the rectified facade. For example, we organize windows in a grid and force them to share the same dimensions and appearance (e.g. number of panels, cornices, and sills), across each row.

5.3. 3D Reconstruction

As illustrated in Figure 5, the 3D reconstruction sub-module, dubbed as Reconstructor, consists of the following parts:

Footprint extrusion: The inputs to this part is a footprint polygon and its metadata (e.g. number of floors). We convert the geocoordinates of the footprint into Mercator coordinates and then into meters. We extrude the footprint vertically considering the height of the building to output the its coarse 3D mesh.

Inverse procedural modeling: The inputs to this part is the parsed sub-components (e.g. windows, entries, stairs, etc.) within a rectified facade. For each sub-component category, we first extract a set of parameters (e.g. width/height ratio for windows) and then use procedural modeling to generate a 3D instance of this category to provide a realistic 3D experience consistent with the given image.

3D mesh generation: With the help of the annotation tool, each annotated facade is also linked to one side of the footprint and thus linked to a 3D plane of the footprint extrusion. With this correspondence, we can compute a transformation that maps a point on the rectified facade to its corresponding point on the face of the footprint extrusion. Using this transformation, we map each reconstructed 3D sub-component to the proper location on the footprint extrusion. At the end, we can merge these transformed 3D sub-component and footprint extrusion into one single mesh as the final 3D reconstruction of the target building.

5.4. 3D Model Repository

The Kartta Labs 3D Model Repository, called Reservoir, hosts and serves the geolocated 3D models for downstream rendering. It is an open-sourced web service, based on the 3DMR[18] project, that hosts the reconstructed 3D assets which can be inspected, modified, pushed, and fetched either through a user interface or programmatically through a REST API. An ID is associated with each 3D model uploaded to the Reservoir which can be used to link it to a building footprint in Editor. Unlike other sub-modules in Kartta Labs, Reservoir does not process its input (3D models) to generate an output.

Reservoir is a centralized location for federated researchers to push their temporal and geolocated reconstructions with corresponding metadata to a common platform for uniform downstream rendering. This decoupling extends to the rendering process as the open-sourced 3D assets served by the model repository can be accessed and rendered by multiple, potentially independent rendering projects.

5.5. 3D Rendering

The 3D renderer of Kartta Labs, called Renderer for short, is our user facing web application that visualizes the reconstructed 3D models on their geolocation. Renderer is a client-side application that fetches the map features, including building footprints, from our database. It then extrudes a footprint if a 3D model is not available for that building, otherwise it downloads the associated 3D model from the Reservoir and renders it. The input to Renderer is the vector map tiles and the 3D models, and the output is 3D visualization of an area. Renderer uses THREE JS library to display the 3D models. To provide a fast and seamless transition in time, Renderer downloads the 3D models for all the buildings disregarding their start and end dates. It then deactivates the buildings not present in a given time set by a slider. First-person street level view and birds-eye view are available.

[17] https://github.com/google/mediapipe

[18] https://gitlab.com/n42k/3dmr

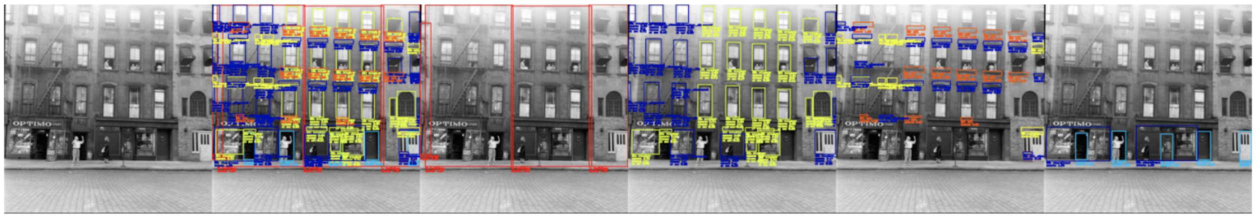


Figure 6. Kartta Labs' facade parsing output. The input image (far left) is parsed to detect facade sub-components such as windows, window sills, cornices, roof cornice, storefronts, entries, and stairs.

5.6. Data

Collecting: Data plays a major role in this project. Even though we rely on our users to collect historical data, we are actively looking for available resources for historical maps and urban photos. To bootstrap our historical maps database, we are discussing possible collaborations with archives, libraries, municipalities, etc. to load their archived maps and photos into our pipeline. Furthermore, some parts of the contemporary OSM data are relevant. For example, most of the streets in large cities have not changed in the past decades or there are many century-old buildings in Manhattan, New York. This kind of data is readily available in the OSM database.

Quality control: Quality control often becomes a critical issue in any crowdsourcing project. Furthermore, any data generated using machine learning approaches also needs proper quality control as these methods are inherently meant not to be perfect. Since Kartta Labs uses both crowdsourcing and machine learning to generate its output data, it needs to have a procedure for quality control.

Quality is a subjective issue in general. The expectations for different aspects of quality such reliability, accuracy, relevancy, completeness, and consistency [16] can significantly vary for different projects. For example Kartta Labs tolerates incomplete data with the expectation that it will eventually achieve completeness. As an another example, we do not need to precisely know the dates the historical photos are taken. This is because buildings life often spans several decades and it is usually enough to know the approximate snapped time of a historical photo to associate it with a set of certain buildings.

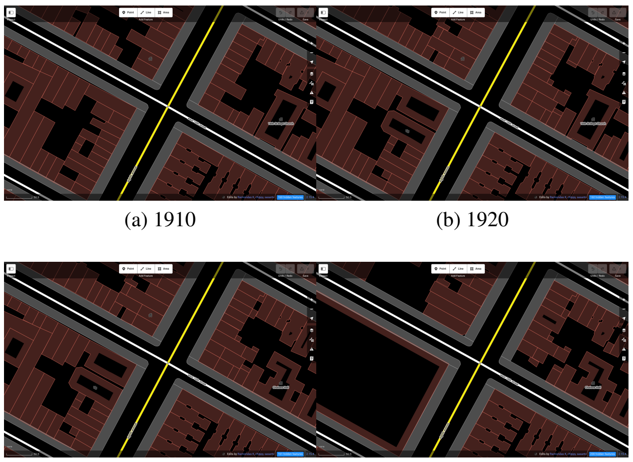
Similar to projects such as OpenStreetMap and Wikipedia, the quality control in Kartta Labs heavily relies on crowdsourcing itself. For example, users can leave "notes" on the map to describe discrepancy or correct the flawed data themselves. We also rely on automated tools to ensure the quality of our output. For example, the Editor has a feature to detect overlapping buildings. We are extending this feature to take the time dimension into account. The result is that the editing user receives a warning if a building overlaps another one at the same time period. Another example is our regularization sub-module that applies a certain set of predefined rules to ensure the reconstructed facades follow expected grammars.

Several crowdsourcing projects rely on reputation [17] of users to ensure the quality of their work. We took a similar but simpler approach by defining pipelines to ban users with malicious activity and making a small subset of users as admins with more authority. We intent to expand our quality control after we launch and collect more data.

License: To encourage the collaborative nature of our project, we use the Open Database License (ODbL) on our Maps data. Other generated and crowdsourced data, such as 3D reconstructions and photo annotations are also open sourced under appropriate licenses.

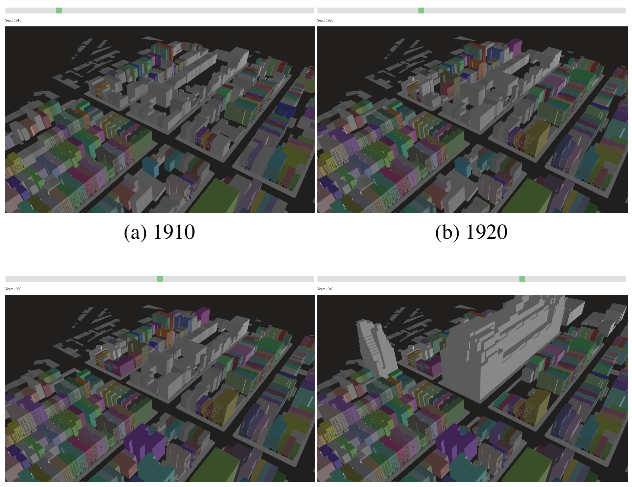
6. Results

To evaluate our system, we are running an experimental instance of the Kartta Labs applications on an internal network. We reconstructed 8 blocks of Manhattan around the Google NYC building. More specifically we reconstructed the blocks between 7th and 9th avenues and W. 14th and W. 18th streets. The time was limited between 1900 to 1960. More than 1000 building footprints were traced from historical maps of different years. We were able to reconstruct the 3D models of 333 buildings from historical photos. Figure 7 shows the map of the area north-east of the Google NYC building (intersection of 8th Ave and W. 16th Street) in 1910, 1920, 1930, and 1940. The vectorized data are extracted from scans of historical maps. Figure 8 shows an area around the Google NYC building during the same years but in 3D and from Renderer. We have added man-made and more accurate 3D models for a couple of buildings, including the Google NYC building, to Reservoir as a reference as well as to show the capability of the system to incorporate external


(a) 1910 (b) 1920

(c) 1930 (d) 1940

Figure 7. Vectorized maps of part of Manhattan around Google NYC building in different years.


(a) 1910 (b) 1920

(c) 1930 (d) 1940

Figure 8. Part of Manhattan around Google NYC building reconstructed in 3D in different years from birds-eyeview of Renderer.

3D models. Finally, Figure 9 shows a reconstructed street view of the 15th street, south of Google NYC building, in 1910, 1920, and 1940 from our Renderer and compares it with the modern Google StreetView of the same location. Reconstructed buildings from photos are shown in vivid colors to distinguish them from those that are only extrusions of footprints. Note that our results shown in this paper are considered preliminary. We are working on rendering our results in a photorealistic mode by generating textures for buildings facades and sub-components.

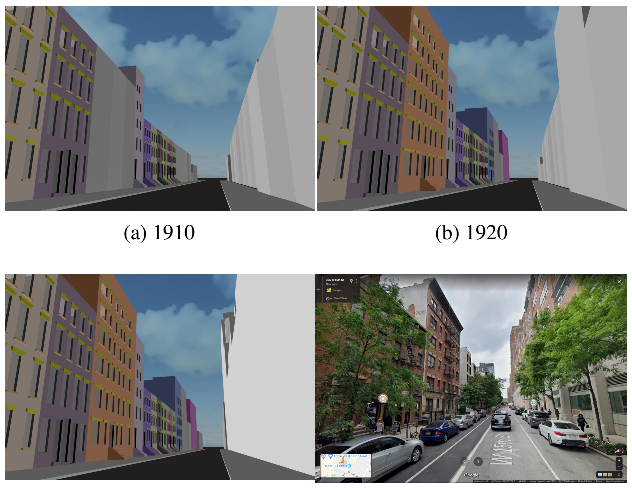
7. Proposed Use Cases and Applications

The Kartta Labs system provides a valuable platform and resource for research and education. First and foremost, we would like to build a community that not only utilizes our historical datasets and open-source codes, but also contributes to both. As a platform that collect, integrate, and visualize detailed historical information about places, Kartta Labs can be used to facilitate numerous educational and research applications and use cases, such as topics in sociology (e.g., [18]), cancer and environmental epidemiology (e.g., [19]), urbanization, biodiversity (e.g., [20]), human disease (e.g., [21]), and biology (e.g., [22, 23, 24]). (See [25] and [26] for examples on using historical geographic datasets and historical Geographic Information System in scientific studies.)

We consider Kartta Labs as the underlying frame of reference to integrate various sources of spatiotemporal historical data such as traffic [27], census, weather, crime [28], pollution [29] and other environmental, sensed [30], or crowdsourced [31] data with location and time dimensions. Imagine Kartta Labs as a generalization of Google Maps where instead of showing the current state of affairs (e.g., current traffic, current population), can show the same information for past historical time frames. For example, transportation authorities can study the impact of building certain freeways in Los Angeles on its traffic or pollution. This spatial integration of data to its historically relevant underlying infrastructure (buildings and roads) can revolutionize the way we do research and educate [32]. Beyond its educational and research applications it can be used for journalism [33] and entertainment to tell better and more visually accurate stories.

Kartta Labs can be used for change detection [34] in various application domains from urban planning to transportation and public health [35] policy making. The decision makers can visualize seamlessly how the urban structure has changed over time and study the impact of these changes on the city infrastructure and public. For example, how often and in which locations new hospitals were built, the rate of increase (or decrease) in parks, schools, shops and restaurants in certain neighborhoods.

Finally, entertainment can be a major use case of Kartta Labs. For example, location-based games such as Ingress can extend their maps in the time-dimension, augmented reality games such as Minecraft Earth can pull in historical 3D buildings, etc. Movie industry can use Kartta Labs to recreate accurate and photorealistic historical scenes.



(a) 1910 (b) 1920

(c) 1940 (d) 2019

Figure 9. Part of Manhattan south to the Google NYC building reconstructed in 3D in different years from Renderer (a,b,c) and the same area in 2019 from Google StreetView.

8. Conclusion and Future Work

In this paper we introduced Kartta Labs, an open source platform to reconstruct historical cities in 3D. In order to make the system open source, we designed Kartta Labs in a modular way with clear interface design (e.g., input and output) for each module, so that each module can be developed independently, potentially by extending existing open-source components, or be replaced easily in future by alternative implementations and designs. Moreover, by deploying each module in a Docker container managed by Kubernetes, we empowered Kartta Labs to both scale out and up with the ability to be deployed locally on a single machine or on different cloud platforms (e.g., Google Cloud). We also described the two main modules of the system: Maps and 3D Models. The main challenge in developing these modules is the lack of sufficient historical data, especially historical photographs from which 3D models of historical buildings can be constructed. Therefore, we are relying on an active community that can contribute data (and code) and help with geotagging historical buildings and georectifying historical maps. We developed several tools to facilitate these crowdsourced activities. The final outcome has the potential to revolutionize how we teach history and geography, how we research urban planning, transportation, and public health and how we tell stories in journalism and for entertainment.

We are working on developing a better database schema to share our 3D models. Currently our 3D models are hosted individually on an online repository. This is useful as it enables users to view and possibly edit individual 3D models. However, it is not the most efficient solution when it comes to rendering these 3D models on a map. We are considering 3D tiling technologies such as 3DCityDB [11].

We intend to develop a number of new tools to help with automatic geotagging of historical buildings. This is challenging as the facade of the historical buildings may have changed over time and hence image-matching approaches such as PlaNet [36] cannot work on this dataset. The ultimate goal is to allow users to upload any historical photograph of buildings and automatically use the facade of the buildings in the picture to improve the 3D models at the correct time frame. We are also interested in expanding the community around Kartta Labs and supporting new applications and use-cases.

9. Acknowledgements

We thank Amol J. Kapoor for his contributions to this project and for his thorough review of this paper.

References

[1] P. J. Ethington, "Placing the past: Groundwork for a spatial theory of history," *Rethinking History*, vol. 11, no. 4, pp. 465–493, 2007.

[2] M. F. Goodchild, "Citizens as sensors: the world of volunteered geography," *GeoJournal*, vol. 69, no. 4, pp. 211–221, 2007.

[3] S. Tavakkol, Y.-Y. Chiang, T. Waters, F. Han, K. Prasad, and R. Kiveris, "Kartta labs: Unrendering historical maps," in *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery*, pp. 48–51, 2019.

[4] A. Kapoor, H. Larco, and R. Kiveris, "Nostalgin: Extracting 3d city models from historical image data," in *Proceedings of the 25th ACM SIGKDD*, pp. 2565–2575, 2019.

[5] Z. Zhang, "Single-view geometry of a rectangle with application to whiteboard image rectification," 2013.

[6] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Free-form image inpainting with gated convolution," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4471–4480, 2019.

[7] Y. Sivan, "3d3c real virtual worlds defined: The immense potential of merging 3d, community, creation, and commerce," *Journal For Virtual Worlds Research*, vol. 1, no. 1, 2008.

[8] H. Holmstrom and M. Jacobsson, "Using models in virtual world design," in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, pp. 9–pp, IEEE, 2001.

[9] M. D. Dickey, "Three-dimensional virtual worlds and distance learning: two case studies of active worlds as a medium for distance education," *Br J Educ Technol*, vol. 36, no. 3, pp. 439–451, 2005.

[10] I. Naper, "System features of an inhabited 3d virtual environment supporting multimodality in communication," in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, pp. 10–pp, IEEE, 2001.

[11] Z. Yao, C. Nagel, F. Kunde, G. Hudra, P. Willkomm, A. Donaubauer, T. Adolphi, and T. H. Kolbe, "3dcitydb-a 3d geodatabase solution for the management, analysis, and visualization of semantic 3d city models based on citygml," *Open Geospatial Data, Software and Standards*, vol. 3, no. 1, pp. 1–26, 2018.

[12] Z. Li, Y.-Y. Chiang, S. Tavakkol, B. Shbita, J. H. Uhl, S. Leyk, and C. A. Knoblock, "An automatic approach for generating rich, linked geo-metadata from historical map images," in *Proceedings of the 26th ACM SIGKDD*, 2020.

[13] C.-C. Chen, C. A. Knoblock, C. Shahabi, Y.-Y. Chiang, and S. Thakkar, "Automatically and Accurately Conflating Orthoimagery and Street Maps," in *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems*, GIS '04, (New York, NY, USA), pp. 47–56, ACM, 2004.

[14] S. Tavakkol, C. Shahabi, F. Han, and R. Kiveris, "Piaget: A probabilistic inference approach for geolocating historical buildings," in *2020 IEEE International Conference on Big Data (Big Data)*, IEEE, 2020. in press.

[15] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[16] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. Motahari-Nezhad, E. Bertino, and S. Dustdar, "Quality control in crowdsourcing systems: Issues and directions," *IEEE Internet Computing*, vol. 17, no. 2, pp. 76–81, 2013.

[17] L. Kazemi, C. Shahabi, and L. Chen, "Geotrucrowd: trustworthy query answering with spatial crowdsourcing," in *Proceedings of the 21st ACM SIGSPATIAL*, pp. 314–323, 2013.

[18] L. Kurashige, "Rethinking anti-immigrant racism: Lessons from the los angeles vote on the 1920 alien land law," *Southern California Quarterly*, vol. 95, no. 3, pp. 265–283, 2013.

[19] T. M. Mack, *Cancers in the Urban Environment*. Elsevier Science, 2004.

[20] A. W. Hill, R. Guralnick, P. Flemons, R. Beaman, J. Wieczorek, A. Ranipeta, V. Chavan, and D. Remsen, "Location, location, location: utilizing pipelines and services to more effectively georeference the world's biodiversity data," *BMC bioinformatics*, vol. 10, no. Suppl 14, p. S3, 2009.

[21] K. Yoshida, H. A. Burbano, J. Krause, M. Thines, D. Weigel, and S. Kamoun, "Mining herbaria for plant pathogen genomes: back to the future," *PLoS pathogens*, vol. 10, no. 4, p. e1004028, 2014.

[22] C. C. Davis, C. G. Willis, B. Connolly, C. Kelly, and A. M. Ellison, "Herbarium records are reliable sources of phenological change driven by climate and provide novel insights into species' phenological cueing mechanisms," *American journal of botany*, vol. 102, no. 10, pp. 1599–1609, 2015.

[23] C. Lavoie, "Biological collections in an ever changing world: Herbaria as tools for biogeographical and environmental studies," *Perspectives in plant ecology, evolution and systematics*, vol. 15, no. 1, pp. 68–76, 2013.

[24] M. Vellend, C. D. Brown, H. M. Kharouba, J. L. McCune, and I. H. Myers-Smith, "Historical ecology: using unconventional data sources to test for effects of global environmental change," *American journal of botany*, vol. 100, no. 7, pp. 1294–1305, 2013.

[25] Y.-Y. Chiang, W. Duan, S. Leyk, J. H. Uhl, and C. A. Knoblock, *Using Historical Maps in Scientific Studies: Applications, Challenges, and Best Practices*. Springer, 2020.

[26] I. N. Gregory and P. S. Ell, *Historical GIS: Technologies, Methodologies, and Scholarship*, vol. 39 of *Cambridge Studies in Historical Geography*. Cambridge University Press, 2007.

[27] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[28] S. Chainey and J. Ratcliffe, *GIS and crime mapping*. John Wiley & Sons, 2013.

[29] L. Perez, F. Lurmann, J. Wilson, M. Pastor, S. J. Brandt, N. Künzli, and R. McConnell, "Near-roadway pollution and childhood asthma: implications for developing win–win compact urban development and clean vehicle strategies," *Environmental health perspectives*, vol. 120, no. 11, pp. 1619–1626, 2012.

[30] S. He, F. Bastani, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, and S. Madden, "Roadrunner: improving the precision of road network inference from GPS trajectories," in *Proceedings of the 26th ACM SIGSPATIAL, Seattle, WA, USA, November 06-09, 2018* (F. B. Kashani, E. G. Hoel, R. H. Güting, R. Tamassia, and L. Xiong, eds.), pp. 3–12, ACM, 2018.

[31] Y. Tong, Z. Zhou, Y. Zeng, L. Chen, and C. Shahabi, "Spatial crowdsourcing: a survey," *VLDB J.*, vol. 29, no. 1, pp. 217–250, 2020.

[32] S. Shekhar, S. K. Feiner, and W. G. Aref, "Spatial computing," *Commun. ACM*, vol. 59, p. 7281, Dec. 2015.

[33] H. Wei, J. Sankaranarayanan, and H. Samet, "Enhancing local live tweet stream to detect news," *GeoInformatica*, vol. 24, no. 2, pp. 411–441, 2020.

[34] M. Hussain, D. Chen, A. Cheng, H. Wei, and D. Stanley, "Change detection from remotely sensed images: From pixel-based to object-based approaches," *ISPRS Journal of photogrammetry and remote sensing*, vol. 80, pp. 91–106, 2013.

[35] H. G. Basara and M. Yuan, "Community health assessment using self-organizing maps and geographic information systems," *International journal of health geographics*, vol. 7, no. 1, p. 67, 2008.

[36] T. Weyand, I. Kostrikov, and J. Philbin, "Planet-photo geolocation with convolutional neural networks," in *European Conference on Computer Vision*, pp. 37–55, Springer, 2016.