Algorithms for Geospatial Data Matching

Progress Report

In fulfillment of the requirements for the

NU 302 R&D Project

At NIIT University

Submitted by

Surajnath Sidh(U101114FCS146)

Shubham Singh(U101114FCS192)

Mehak Bhatia(U101114FCS193)

Kartik Bhadada(U101114FCS078)

Area

NIIT University ,Neemrana

Rajasthan

CERTIFICATE

This is to certify that the present research work entitled "Algorithms for Geospatial Data Matching” being submitted at NIIT University, Neemrana, Rajasthan, for the fulfillment of requirements of the course, embodies authentic and faithful record of original research carried out by Surajnath Sidh (U101114FCS146), Shubham Singh (U101114FCS192), Mehak Bhatia (U101114FCS193) and Kartik Bhadada (U101114FCS078), student/s of B Tech (CSE) at NIIT University, Neemrana,. She /He has worked under our supervision and that the matter embodied in this project work has not been submitted, in part or full, as a project report for any course of NIIT University, Neemrana or any other university.

Dr. Prosenjit Gupta

LIST OF FIGURES

LIST OF TABLES

CONTENTS

Topic Overview

- The field of Geographical Information Systems (GIS) experienced a rapid growth of available sources of geospatial datasets.

- More and more data integration has been demanded by this growth in order to explore the benefits of these data further. However, the same phenomena, that is, geospatial features is being implied by the points of views for many data providers.

- Due to recent trends in Geographic Information Science (GIScience) the demand of integration of an unceasingly growing size of geospatial datasets have been spotted. Among these trends the popularization of crowdsourced data can be cited and this crowdsourced data is also referred to as Volunteered Geographic Information (VGI).

- Data Matching : aka linking/alignment/reconciliation is a process of gathering different data sources with pronounced similarity in a geospatial or semantic way.

- In layman terms, we consider it as a process of finding correspondences between two spatial datasets, like concepts, objects or their components.

Rationale Of Work

❑ Rapid growth of data sets resulted in the need to plot them.

❑ Since most of the data has been plotted, now there is a need to detect the errors and correct/update them by matching the data with the crowdsourced data.

❑ Crowdsourced data also known as retrieved data is the data collected by sources such as check-ins etc. Relevant data or authoritative data is the actual data that needs to be verified/corrected.

❑ Since huge amount of retrieved and relevant data is available, matching it becomes a rather hectic task.

❑ Hence certain specific algorithms needs to be implemented in order to lighten the job.

❑ In our research, we will be working on the Foursquare check-In dataset and using it as a test dataset for our algorithm.

LITERATURE REVIEW

1. A Survey of Measures and Methods for Matching Geospatial Vector Datasets.

OVERVIEW

In the literature, one can find many different approaches to solve the geospatial matching problem.

Two key concepts to organise these approaches are:

1) Measures

2) Methods

- Measures can be single, or multiple, combined using normalized score, a weighted combination, the probabilistic theory, optimization processes or brief theory.

  The matching measures are organised on the basis of nature of measured quantity:-

1) Geometry (shape, location, length, area)
2) Topology (geographic space, for instance network matching)
3) Attributes (non-geometric properties; numeric, list and text measures)
4) Context (geographic context, similarity on the basis of presence of another object)

- Methods on the other hand have been classified according to different perspectives. They have evolved from being simpler methods to increasingly complex methods which include a more diverse perspective.

  Matching methods are classified according to-

1) Level of actuation

- Schema (highest level, that is, modelling)
- Feature (object level)
- Internal (internal components)

2) Case of correspondence

- One to One
- One to Many
- Many to Many

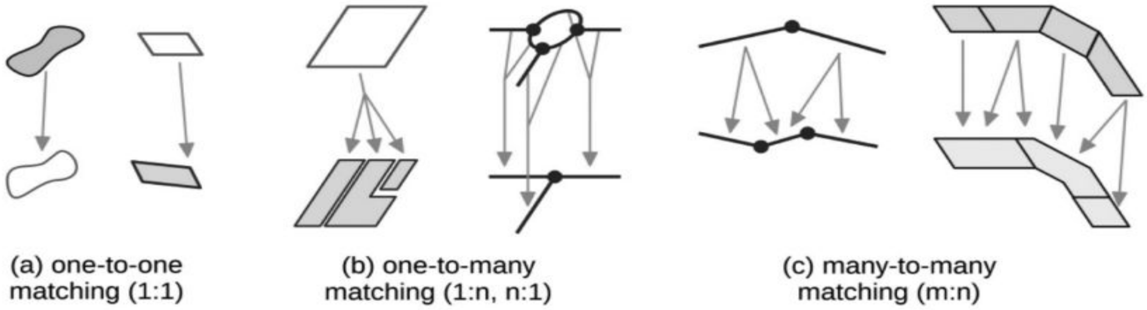

(a) one-to-one matching (1:1)
(b) one-to-many matching (1:n, n:1)
(c) many-to-many matching (m:n)

Fig 1:

CONCLUSION-

This paper presented measures and methods applied to matching geospatial datasets. The paper discusses

1) Classification of measures and methods
2) Issues choosing measures and methods.

2. Similarity of Spatial Scenes:

- Similarity of scenes is a causal judgement people make in everyday life. It is intuitive, subjective and displays no strict mathematical models.
- This paper focusses on similarity measures of spatial scenes that are contained in geographic databases.
- A spatial scene is basically a set of geographic objects together with their spacial relations. These spatial relations are being discussed in this entire paper.

OVERVIEW

The paper deals with three major similarity measures of spatial relations

1) Similarity of Topological Relations-

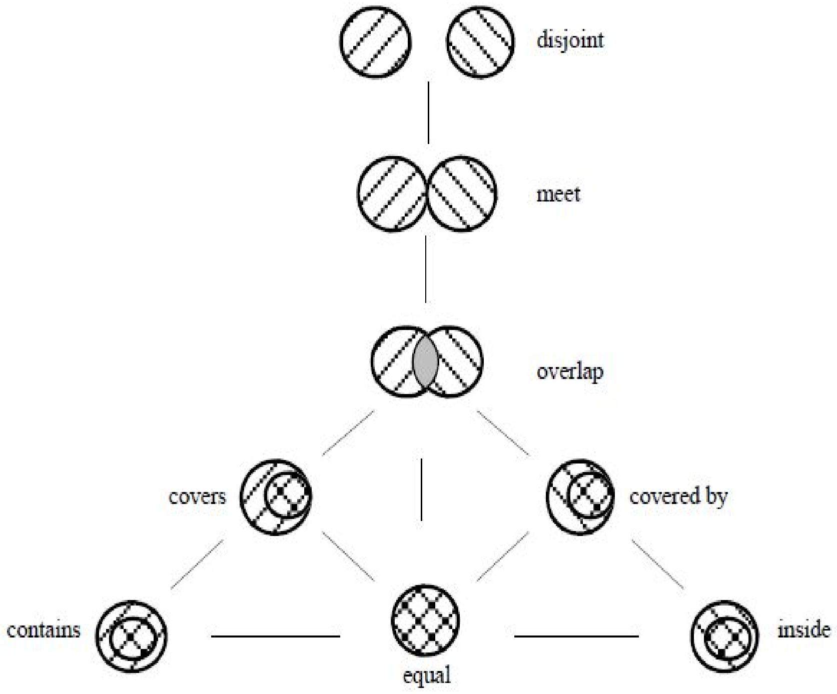


Fig 2:

2) Similarity of Distance Relations (Distance relations are matched according to the radial distance between two objects. Here one of the objects is assumed to be a reference object and the distance is measured by moving the other object.)

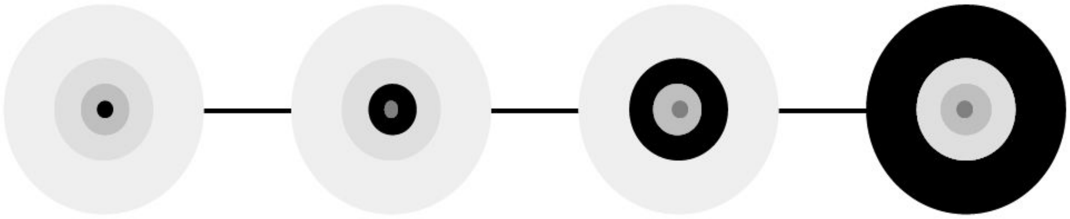


Fig 3:

3) Similarity of Direction Relations

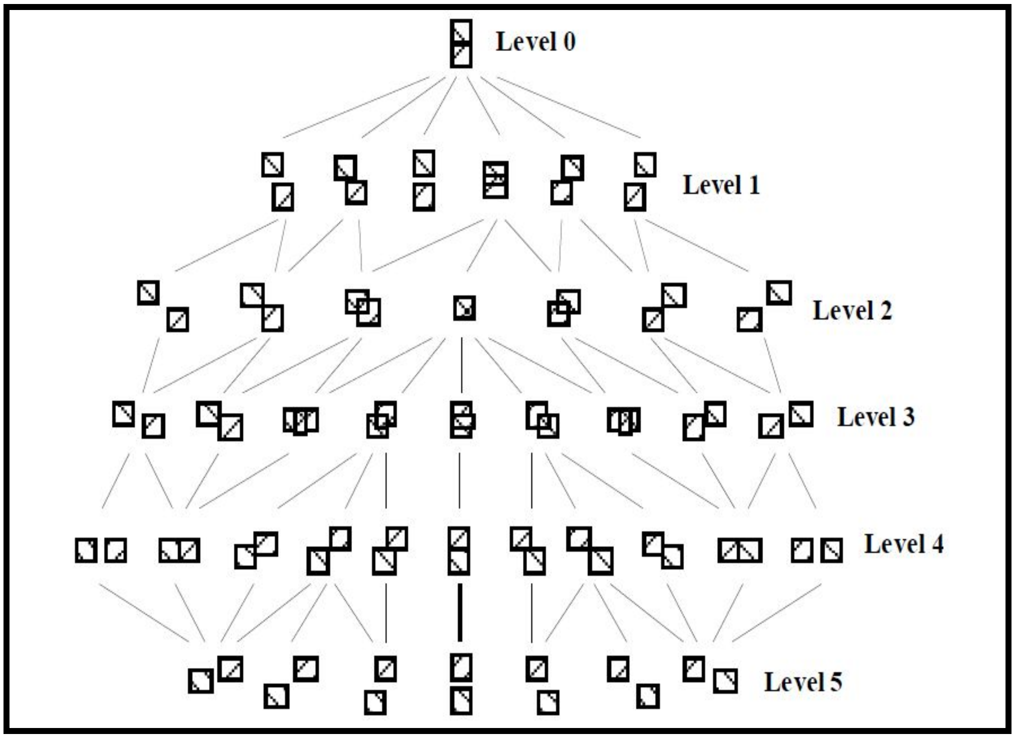


Fig 4:

↓

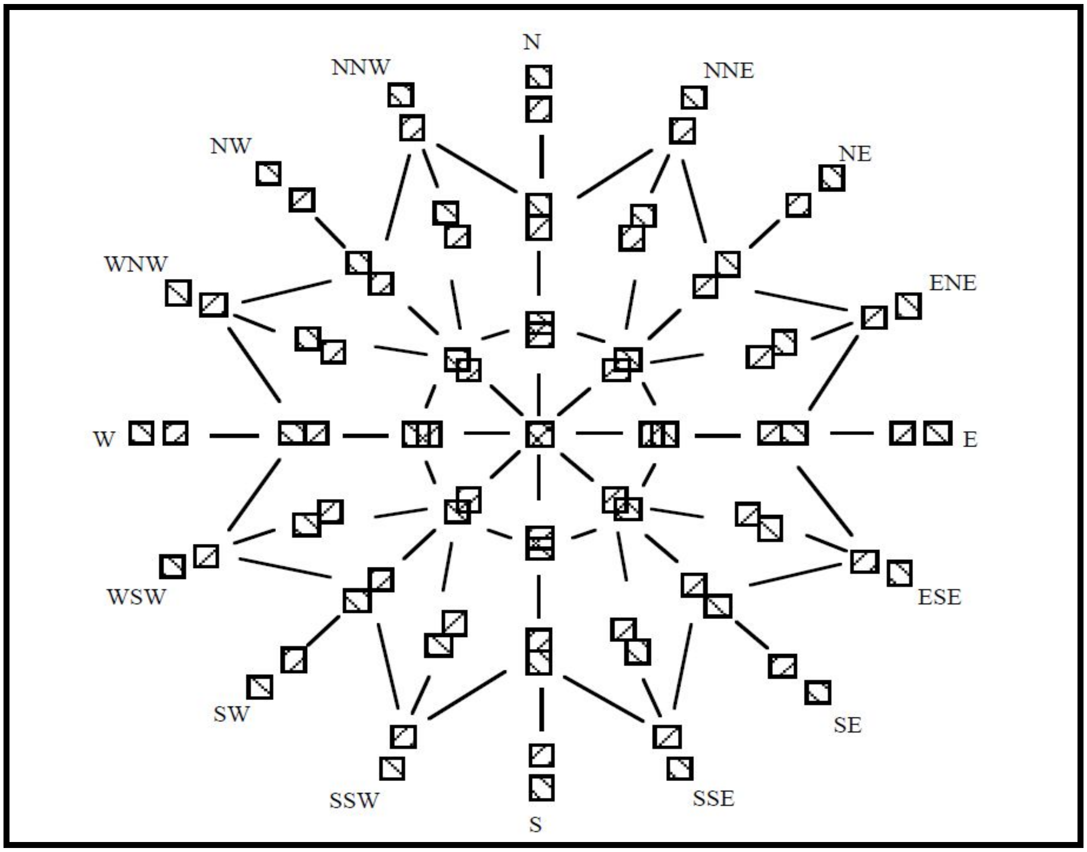


Fig 5:

ASSESSING SPATIAL SIMILARITY-

- The three models of topology, direction and distance form the basis for the assessment of spatial similarity of scenes.
- The relative similarity of the sets of scenes can also be determined by assessing the spatial similarity. For instance-

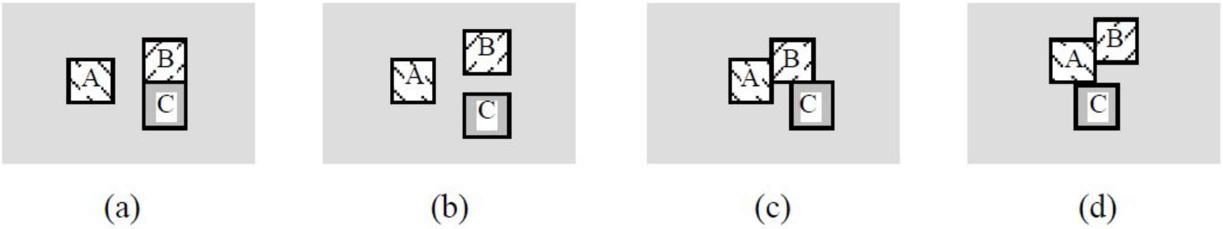


Fig 6:

CONCLUSION-

This paper presents a method for assessing spatial similarity. It is based on the topological, directional and distance relations in the spatial scenes and their conceptual neighborhoods.

3. Geo MatchMaker: Automatic and Efficient Matching of Vector Data with Spatial Attributes in Unknown Geometry Systems.

This paper discusses about the large amount of geospatial data that is available through different sources these days and integration and fusion of data from different sources.

- Geospatial data fusion has been one of the central issues in GIS.
- There have been a number of efforts to automatically or semi-automatically detect matched features across different road vector datasets
- The paper proposes several approaches to handle the matching of diverse data sets automatically and efficiently.
- Vector to vector conflation.
- The initial focus was to remove spatial inconsistency between two data sets. Once the inconsistency is removed it is easier to compare attributes and fuse datasets.

- Multiple attempts to solve this problem have been there.

Methods-

1. PPM(point pattern matching)

This uses the brute force method comparing each attribute of dataset 1 (s1) to all the attributes of dataset 2 (s2) to find the matching point(intersection). It is accurate but very complex for big data sets and very time consuming as well.

2. Geo-PPM

Geo PPM focuses on eliminating the comparison of point pairs that can't be the candidate for matching pattern by using network properties. It is not possible for big data sets especially when datasets follow a pattern like grid or manhattan etc.

3. Prioritized Geo-PPM

The intuition in prioritized Geo-PPM is that the matching feature in one data set is likely to be a matching feature in another data set.This means that if we find the matching feature in one data set we can find it very quickly in another data set and hence it will be easy to find intersections.Prioritized Geo-PPM can provide a substantial improvement over Geo-PPM for large networks.
Matching features - angle between the points, distance between the points etc.

CONCLUSION-

In this paper, we learned about problems with data integration of different data sets and more issues with GIS and the solutions to solve these problems .

- PPM
- Geo-PPM
- Prioritized Geo-PPM.

4. Geospatial Information Integration for Authoritative and Crowd Sourced Road Vector.

This Paper explores the technical issues associated with integrating unstructured crowd sourced data with authoritative national mapping data.

Introduction

- Specially abled people have difficulties getting around due to obstacles found in our infrastructure.
- Accessibility mapping is mapping with knowledge of obstacles.
- That knowledge map of obstacle data is used to build an Navigation system for specially abled people
- This route calculation takes obstacles into account while suggesting route.

Methods

- Tactile Maps
- Notification system in Public Transit and places
- Using GIS Data and network Analysis for Routing
- Using VGI(volunteered geographic information) for Improved Routing

Routing Scenario Example-

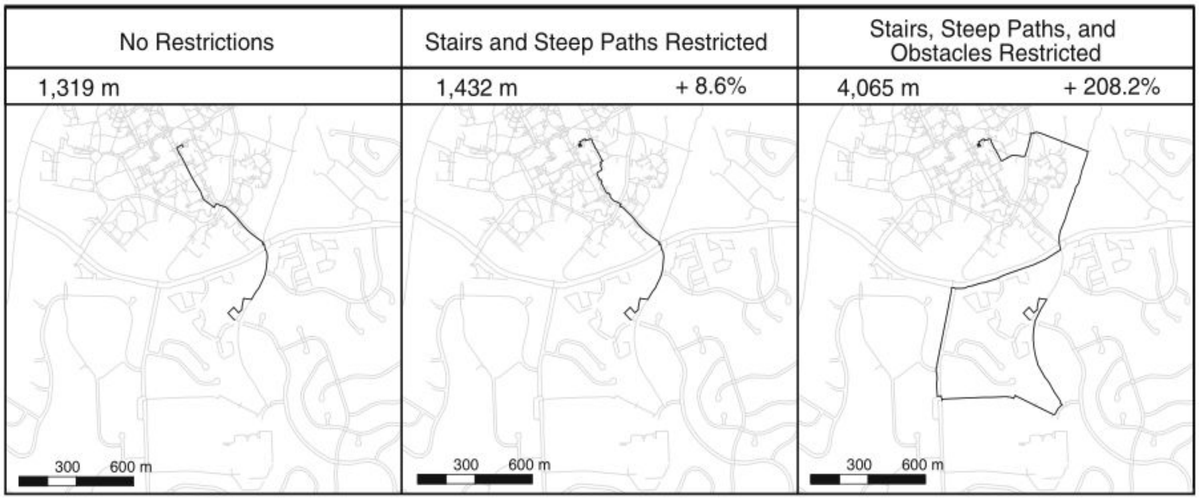

No Restrictions
Stairs and Steep Paths Restricted
Stairs, Steep Paths, and Obstacles Restricted
1,319 m
1,432 m
+ 8.6%
4,065 m
+ 208.2%
300
600 m
300
600 m
300
600 m

Fig 7:

Conclusion-

This paper discusses the development of techniques in Routing for Specially Abled people and Use of GIS and VGI data to improve life of Specially abled people

Objectives

- To bring attention towards data integration and data matching.
- To increase focus towards correcting erroneous data.
- To create an efficient algorithm for dynamic error detection, correction and updating the location of different places(hotels, restaurants etc) using retrieved or crowdsourced data.

Applications

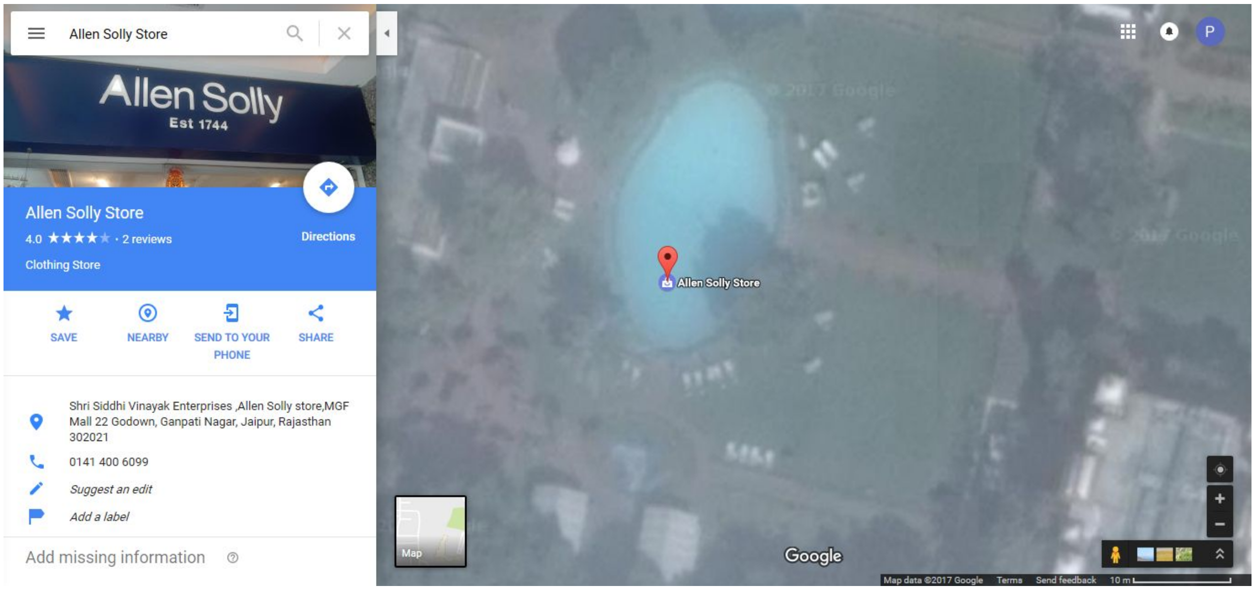
- Disaster Relief
- Route Filtering
- Toll Plaza Markings
- Identity Verification
- and many-many more…

Our problem:

- We are working on a dataset (Foursquare) . It contains location of Hotels and Restaurants attained through check-ins of various visitors.
- Through this data set we are trying to correct or update locations of hotels that are marked inappropriately as of now.

Problem

Real life Example of Problem



Shri Siddhi Vinayak Enterprises ,Allen Solly store,MGF Mall 22 Godown, Ganpati Nagar, Jaipur, Rajasthan 302021

0141 400 6099

Fig 8:(a)

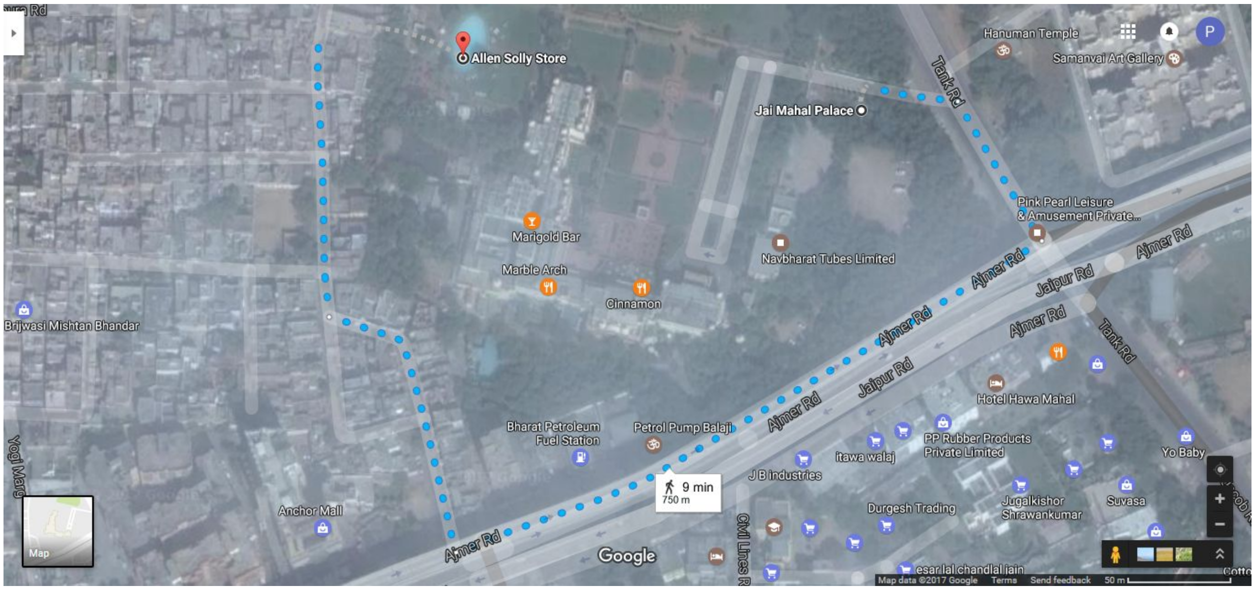
A clothing store in a swimming pool, *Allen Solly Store Google Maps accessed on 19 Mar 2017*

Problem: wrong location data
Current Solution: Manually edit these places by crowdsourcing this editing process
Problem with Current approach:

1.too much data to edit by hand

2. By the rate maps changes with time it’s hard to keep up with it

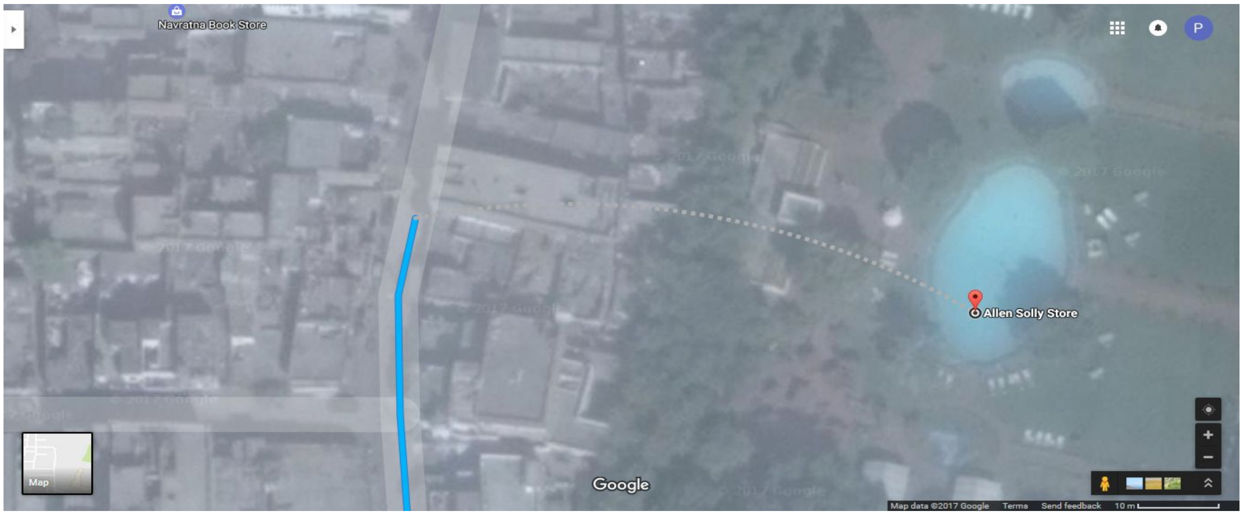
3. Not so reliable



Let’s see how we can reach this store (b)

Side effects of Wrong location data

It tried to approximate the and suggested the way to nearest road and then it does not know what to do

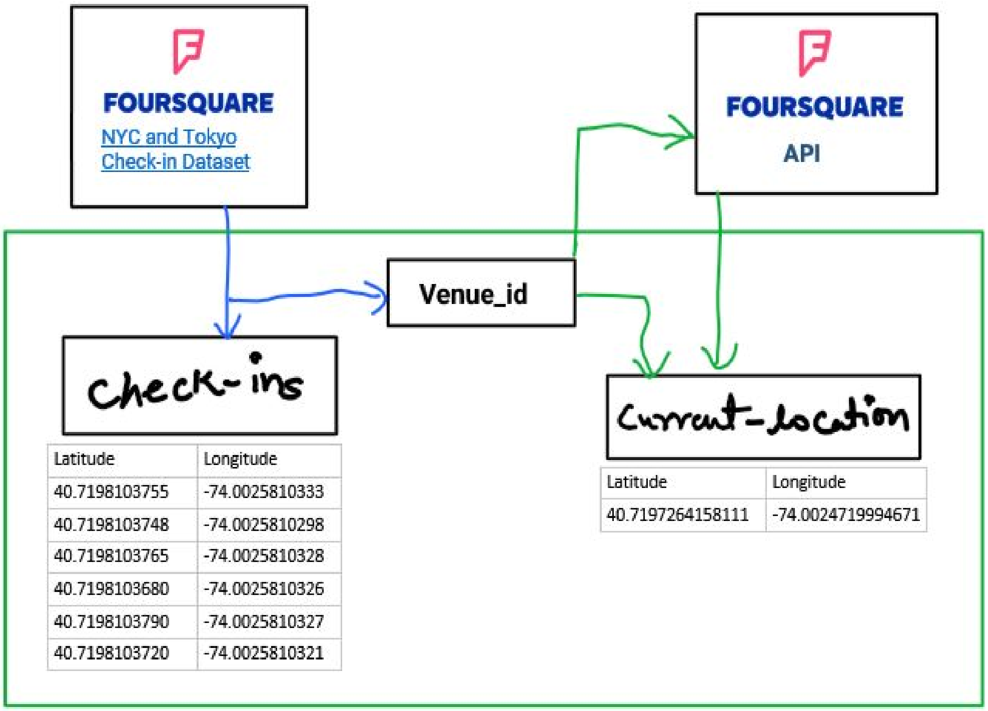


So we have to go through these buildings to reach our desired location

(c)

Dataset

- 783511 Checkins and 97395 Unique locations
- Have Foursquare venue_id which can be used to get more data if required

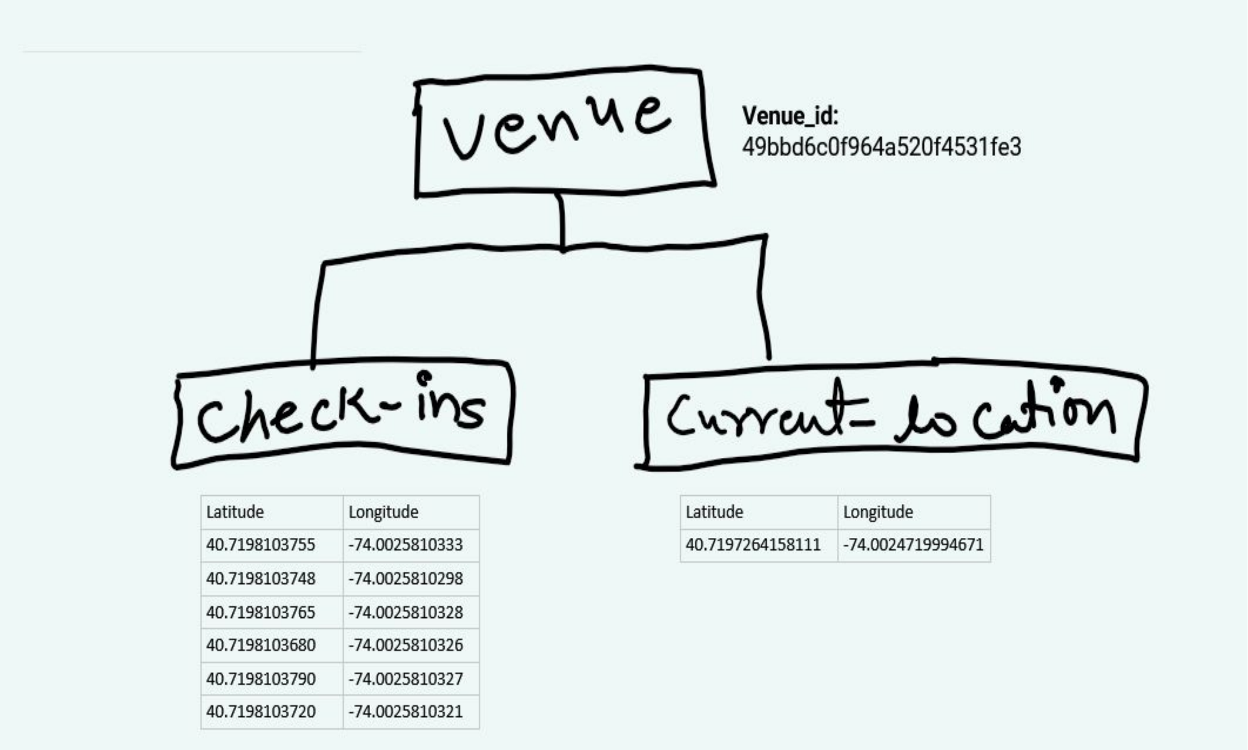


Latitude	Longitude
40.7198103755	-74.0025810333
40.7198103748	-74.0025810298
40.7198103765	-74.0025810328
40.7198103680	-74.0025810326
40.7198103790	-74.0025810327
40.7198103720	-74.0025810321

Latitude	Longitude
40.7197264158111	-74.0024719994671

Fig: creation of dataset using NYC and Tokyo Check-in dataset and Foursquare API

Fig 9:



Venue_id:
49bbd6c0f964a520f4531fe3

Latitude	Longitude
40.7198103755	-74.0025810333
40.7198103748	-74.0025810298
40.7198103765	-74.0025810328
40.7198103680	-74.0025810326
40.7198103790	-74.0025810327
40.7198103720	-74.0025810321

Latitude	Longitude
40.7197264158111	-74.0024719994671

Fig 10:

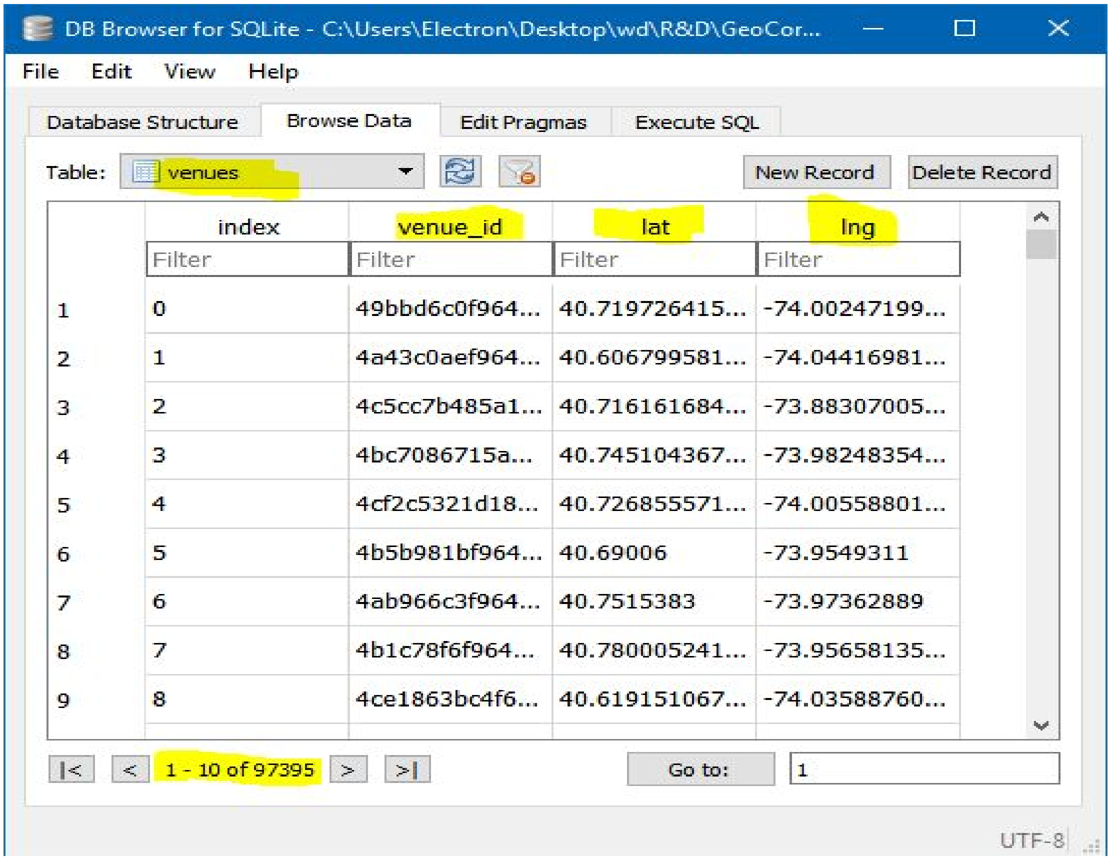


Table: venues

	index	venue_id	lat	lng
	Filter	Filter	Filter	Filter
1	0	49bbd6c0f964...	40.719726415...	-74.00247199...
2	1	4a43c0aef964...	40.606799581...	-74.04416981...
3	2	4c5cc7b485a1...	40.716161684...	-73.88307005...
4	3	4bc7086715a...	40.745104367...	-73.98248354...
5	4	4cf2c5321d18...	40.726855571...	-74.00558801...
6	5	4b5b981bf964...	40.69006	-73.9549311
7	6	4ab966c3f964...	40.7515383	-73.97362889
8	7	4b1c78f6f964...	40.780005241...	-73.95658135...
9	8	4ce1863bc4f6...	40.619151067...	-74.03588760...

1 - 10 of 97395

Go to: 1

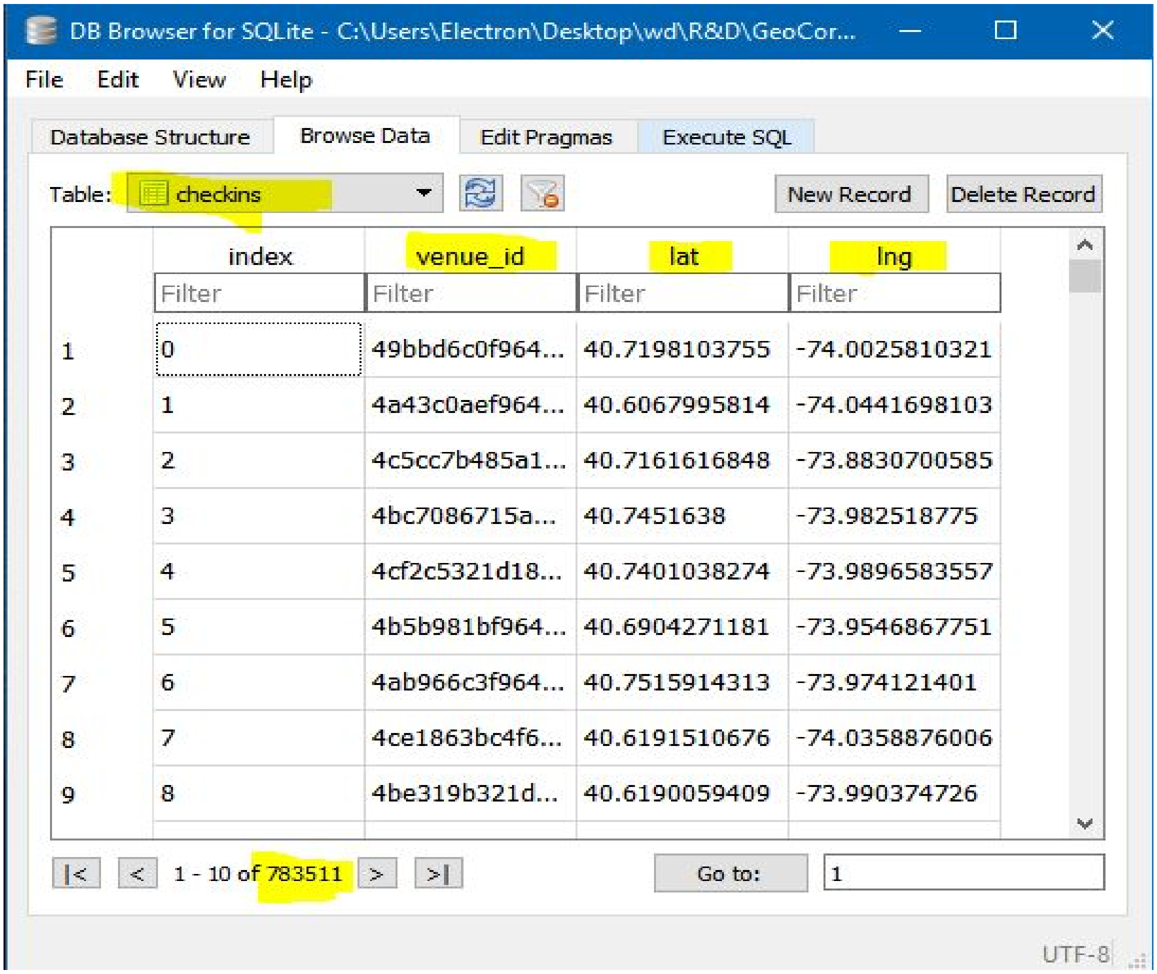


Table: checkins

	index	venue_id	lat	lng
	Filter	Filter	Filter	Filter
1	0	49bbd6c0f964...	40.7198103755	-74.0025810321
2	1	4a43c0aef964...	40.6067995814	-74.0441698103
3	2	4c5cc7b485a1...	40.7161616848	-73.8830700585
4	3	4bc7086715a...	40.7451638	-73.982518775
5	4	4cf2c5321d18...	40.7401038274	-73.9896583557
6	5	4b5b981bf964...	40.6904271181	-73.9546867751
7	6	4ab966c3f964...	40.7515914313	-73.974121401
8	7	4ce1863bc4f6...	40.6191510676	-74.0358876006
9	8	4be319b321d...	40.6190059409	-73.990374726

|< < 1 - 10 of 783511 > >|

Go to: 1

Table 1:

Algorithm

Projection

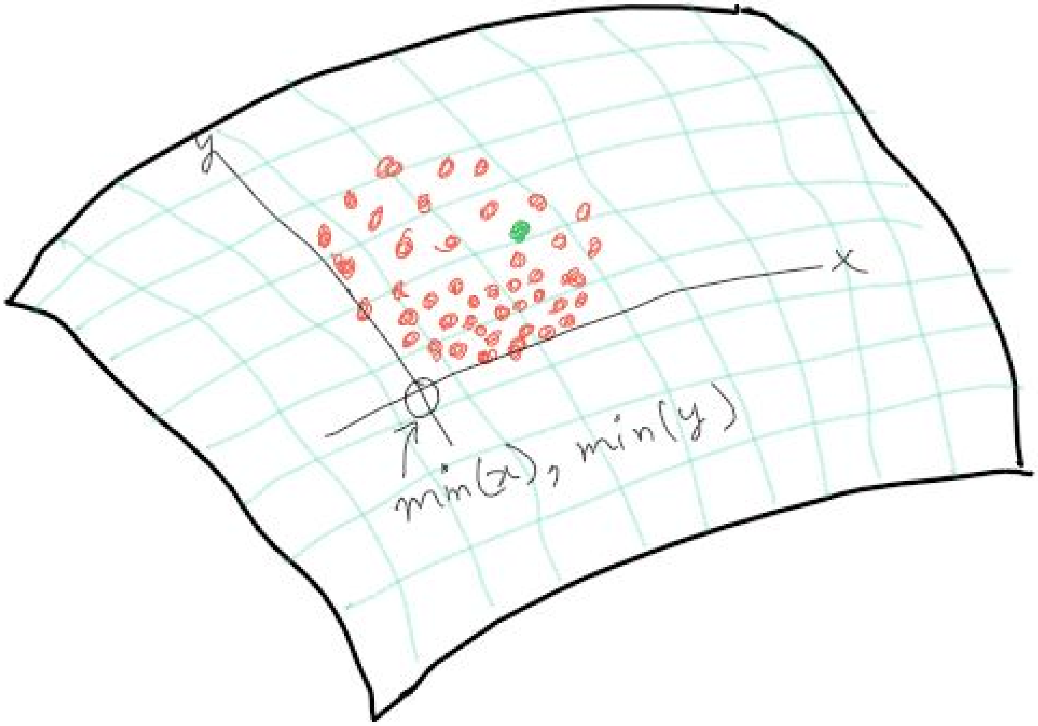


Fig: 3D cross-section of area is divided in quadrants based on current-location(center- *min(x), min(y)*

Fig 11:

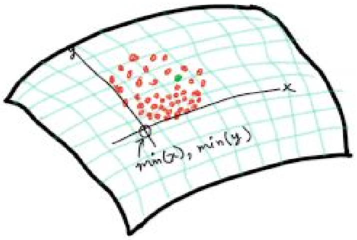
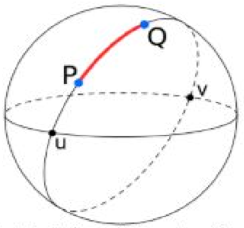
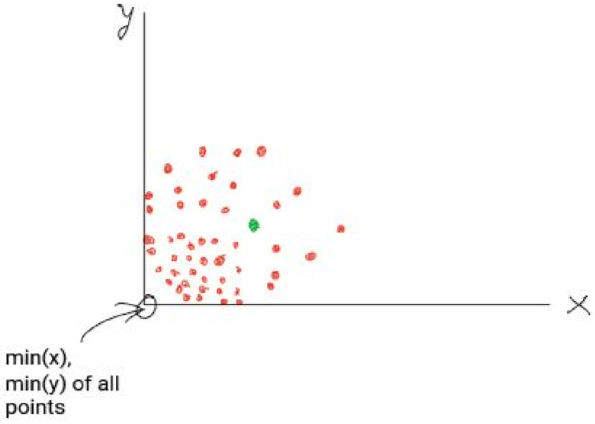


Fig: 3D cross-section of area is divided in quadrants based on current-location(*center-min(x), min(y)*)



great-circle distance (drawn in red) between two points on a sphere, P and Q



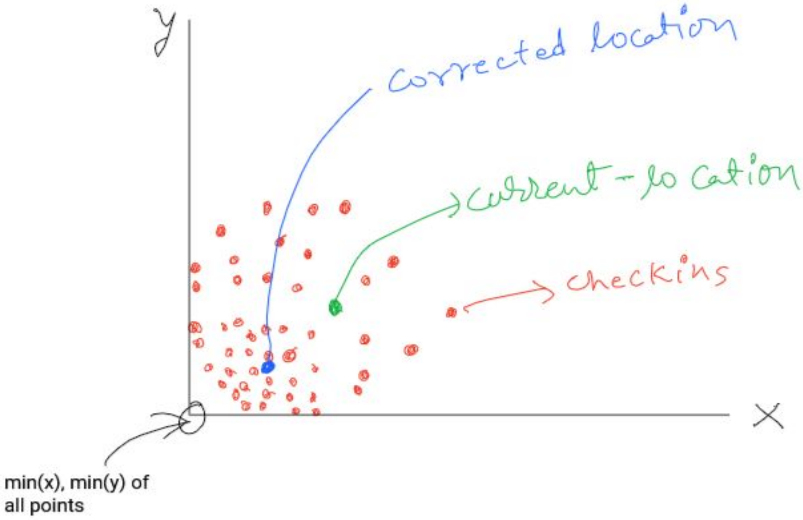


Fig 12:

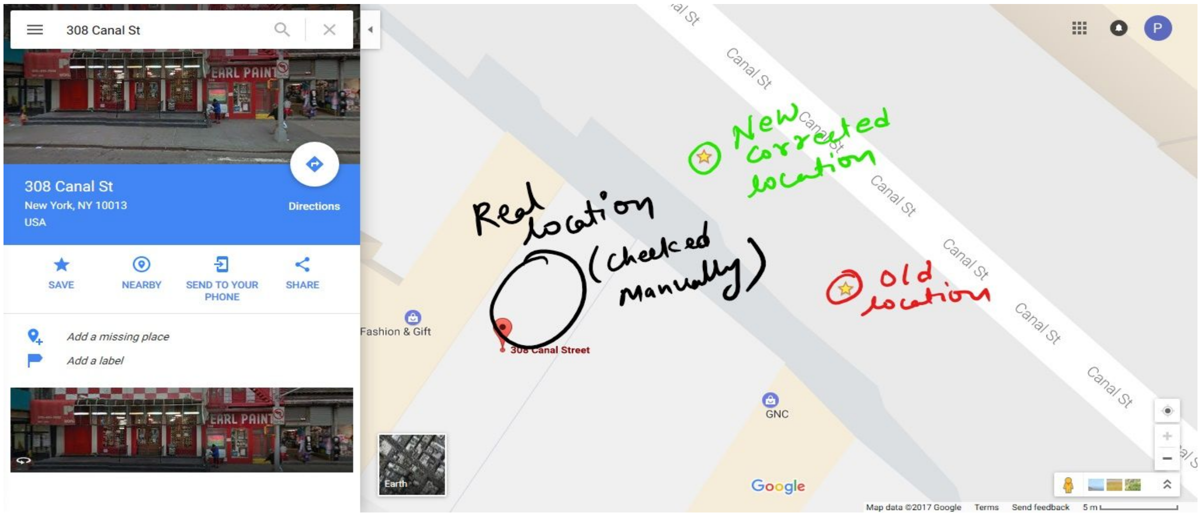
Methodology for Algorithm

1. Select an id from Venues Table in Dataset
2. Select all the check-in points for that id from Check-ins table
3. Find mode of all the check-in points for a given id.
4. The mode will be new corrected location
5. Find the great circle distance between two points.

Three cases for calculation of Mode:-

- ID doesn’t exist in check-ins table.
    - If there are no check-in points for an id
    - Present location is final.
- ID exists and mode can be calculated.
    - If there are check-in points and mode can be calculated
    - Calculated location using mode is final.
- ID exists but multiple mode exist for it.
    - If multiple modes are found for an id
    - Location closer to current one is treated as final.

Results



Results from manual testing

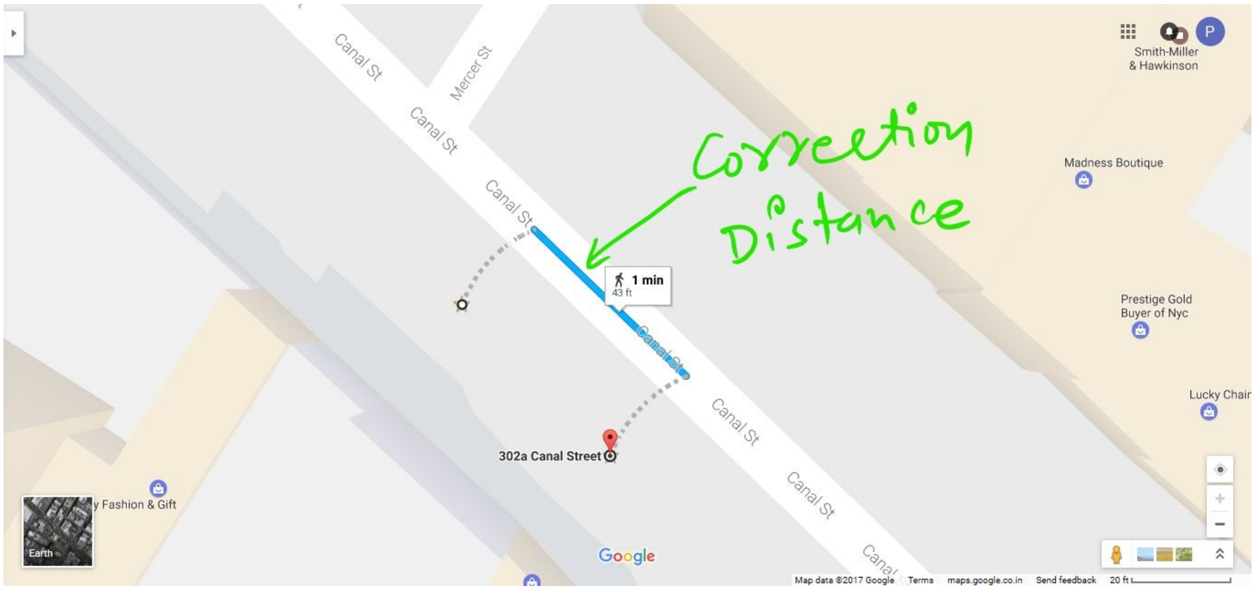
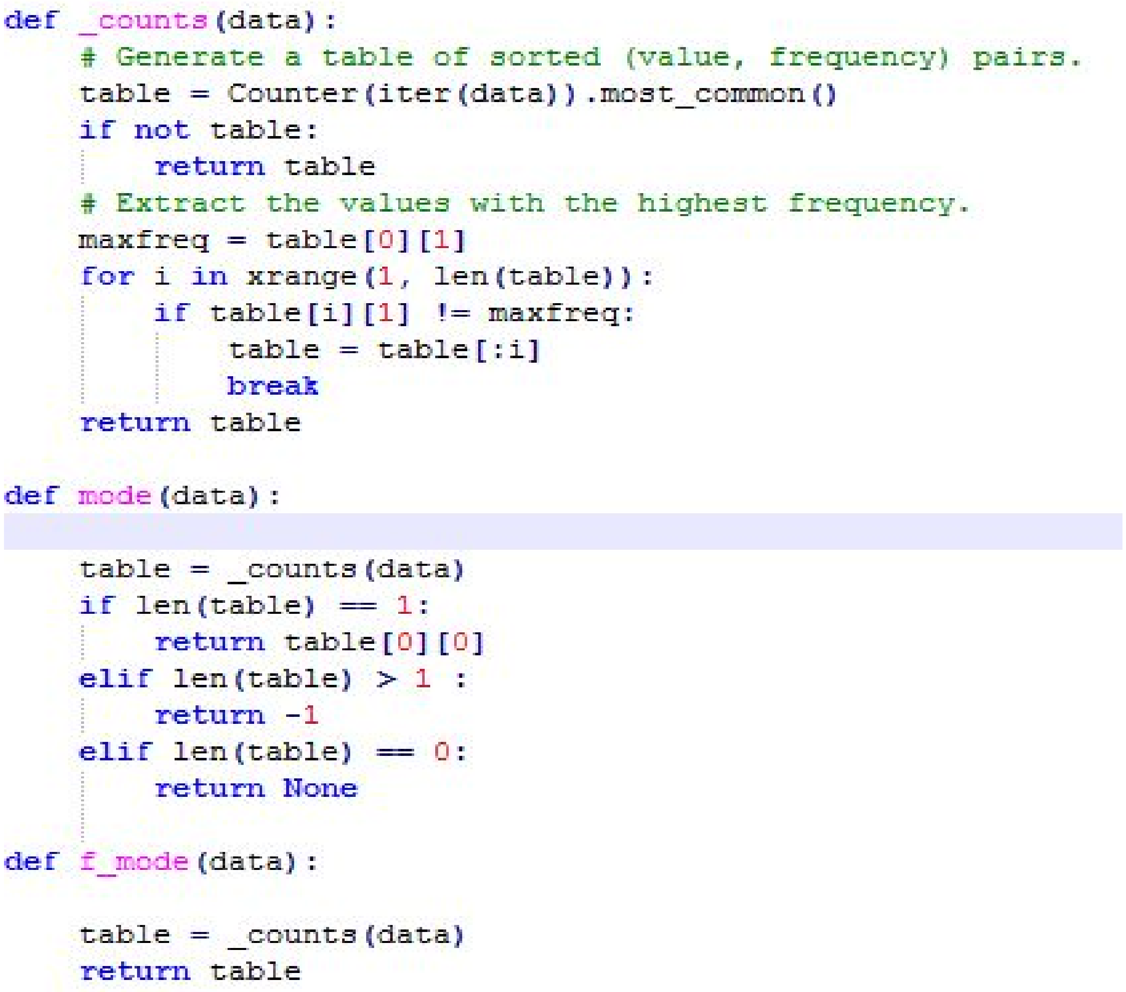


Fig 13:

Screenshot Of Algorithm(Mode calculation)



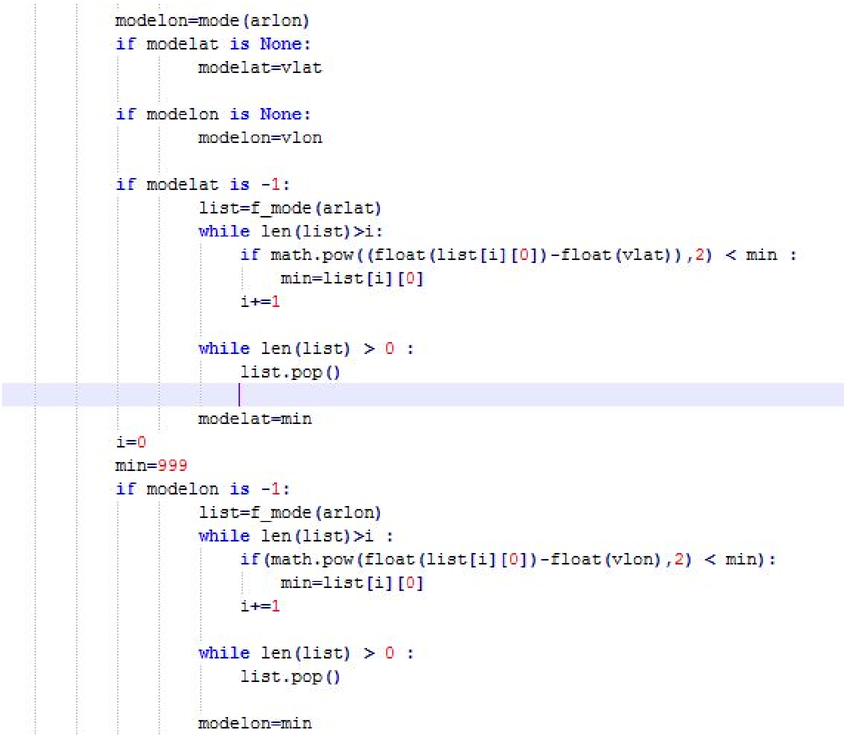
```
def _counts(data):
    # Generate a table of sorted (value, frequency) pairs.
    table = Counter(iter(data)).most_common()
    if not table:
        return table
    # Extract the values with the highest frequency.
    maxfreq = table[0][1]
    for i in xrange(1, len(table)):
        if table[i][1] != maxfreq:
            table = table[:i]
            break
    return table

def mode(data):

    table = _counts(data)
    if len(table) == 1:
        return table[0][0]
    elif len(table) > 1 :
        return -1
    elif len(table) == 0:
        return None

def f_mode(data):

    table = _counts(data)
    return table
```



```
modelon=mode(arlon)
if modelat is None:
        modelat=vlat

if modelon is None:
        modelon=vlon

if modelat is -1:
        list=f_mode(arlat)
        while len(list)>i:
            if math.pow((float(list[i][0])-float(vlat)),2) < min :
                min=list[i][0]
            i+=1

        while len(list) > 0 :
            list.pop()

        modelat=min
i=0
min=999
if modelon is -1:
        list=f_mode(arlon)
        while len(list)>i :
            if(math.pow(float(list[i][0])-float(vlon),2) < min):
                min=list[i][0]
            i+=1

        while len(list) > 0 :
            list.pop()

        modelon=min
```

Analysis:-

- Total number of locations (id's) in database = 97395
- Total number of locations with no change = 9070
- Total number of locations with a change = 88325
- Roughly 90% locations in the dataset were found out to be inaccurate.

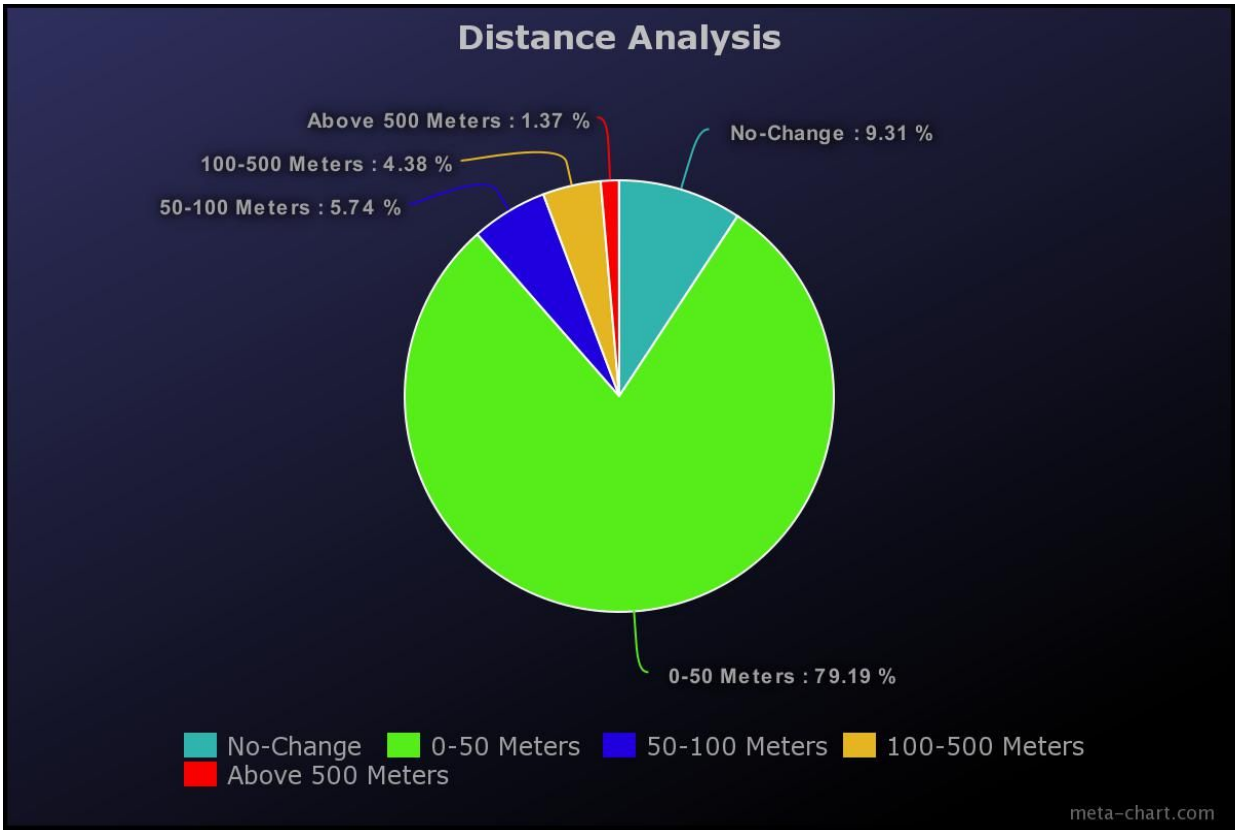


Fig 14:

Outcome

- Even If we don’t consider the change in distance of less than 100 meters, still around 6% of location need to be changed which is quite a significant proportion.
- And when we want to build maps for self-driving cars the locations with less than 100 meters change need to be considered as well.

Screenshot of Results database

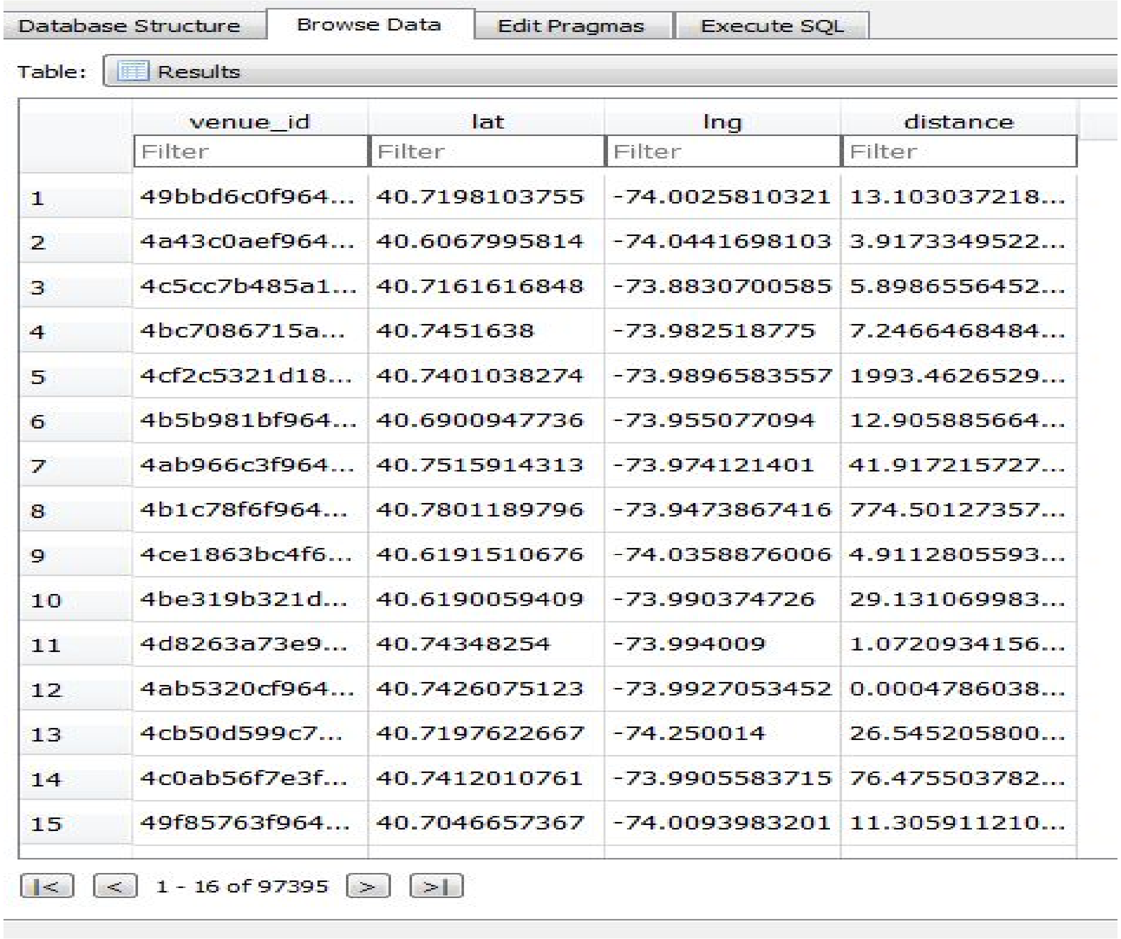


Table: Results

	venue_id	lat	lng	distance
	Filter	Filter	Filter	Filter
1	49bbd6c0f964...	40.7198103755	-74.0025810321	13.103037218...
2	4a43c0aef964...	40.6067995814	-74.0441698103	3.9173349522...
3	4c5cc7b485a1...	40.7161616848	-73.8830700585	5.8986556452...
4	4bc7086715a...	40.7451638	-73.982518775	7.2466468484...
5	4cf2c5321d18...	40.7401038274	-73.9896583557	1993.4626529...
6	4b5b981bf964...	40.6900947736	-73.955077094	12.905885664...
7	4ab966c3f964...	40.7515914313	-73.974121401	41.917215727...
8	4b1c78f6f964...	40.7801189796	-73.9473867416	774.50127357...
9	4ce1863bc4f6...	40.6191510676	-74.0358876006	4.9112805593...
10	4be319b321d...	40.6190059409	-73.990374726	29.131069983...
11	4d8263a73e9...	40.74348254	-73.994009	1.0720934156...
12	4ab5320cf964...	40.7426075123	-73.9927053452	0.0004786038...
13	4cb50d599c7...	40.7197622667	-74.250014	26.545205800...
14	4c0ab56f7e3f...	40.7412010761	-73.9905583715	76.475503782...
15	49f85763f964...	40.7046657367	-74.0093983201	11.305911210...

1 - 16 of 97395

Table 2:

Observations for algorithm:

- Algorithm works best when we have massive amount of check-in data
- Algorithm is tolerant to outliers and simple to implement in comparison with Algorithm we designed at first stage
- Algorithm will never introduce more error then current error, provided that check-in data is correct

Visualisation Of Results

Visualisation demo can be seen live at http://geocorrect.herokuapp.com/

Technical Stack of Visualisation

Flask
web development,
one drop at a time

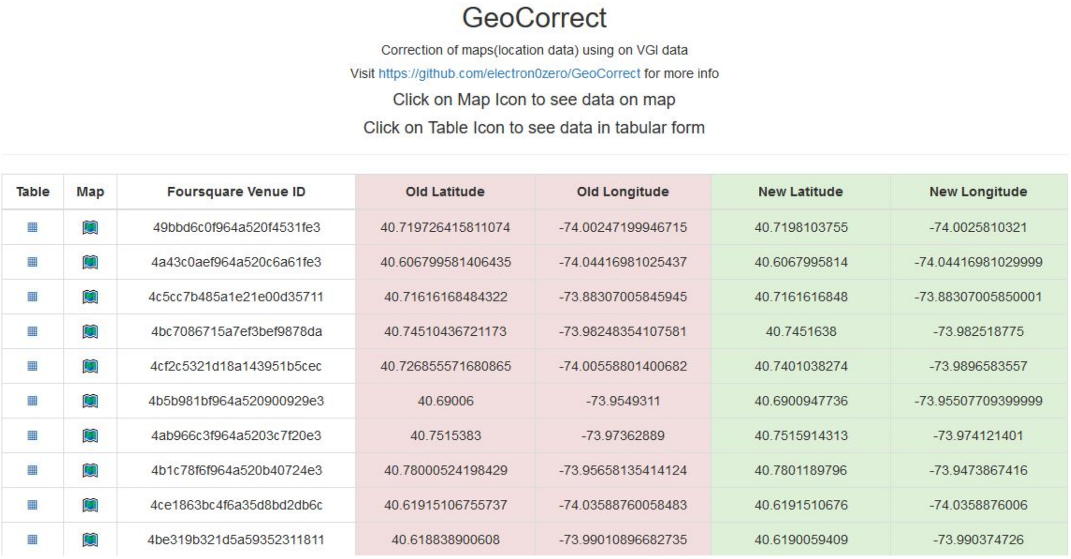
python™

HEROKU

Google Maps

FOURSQUARE

Visualisation View



GeoCorrect

Correction of maps(location data) using on VGI data

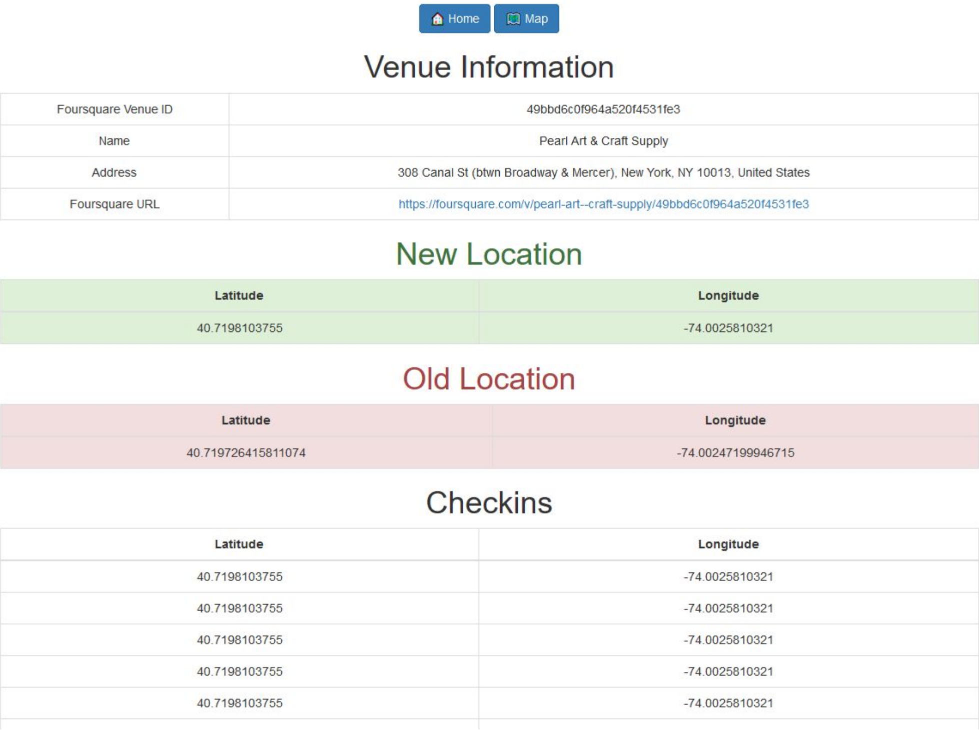
Visit https://github.com/electron0zero/GeoCorrect for more info

Click on Map Icon to see data on map

Click on Table Icon to see data in tabular form

Table	Map	Foursquare Venue ID	Old Latitude	Old Longitude	New Latitude	New Longitude
▦	🗺	49bbd6c0f964a520f4531fe3	40.719726415811074	-74.00247199946715	40.7198103755	-74.0025810321
▦	🗺	4a43c0aef964a520c6a61fe3	40.606799581406435	-74.04416981025437	40.6067995814	-74.04416981029999
▦	🗺	4c5cc7b485a1e21e00d35711	40.71616168484322	-73.88307005845945	40.7161616848	-73.88307005850001
▦	🗺	4bc7086715a7ef3bef9878da	40.74510436721173	-73.98248354107581	40.7451638	-73.982518775
▦	🗺	4cf2c5321d18a143951b5cec	40.726855571680865	-74.00558801400682	40.7401038274	-73.9896583557
▦	🗺	4b5b981bf964a520900929e3	40.69006	-73.9549311	40.6900947736	-73.95507709399999
▦	🗺	4ab966c3f964a5203c7f20e3	40.7515383	-73.97362889	40.7515914313	-73.974121401
▦	🗺	4b1c78f6f964a520b40724e3	40.78000524198429	-73.95658135414124	40.7801189796	-73.9473867416
▦	🗺	4ce1863bc4f6a35d8bd2db6c	40.61915106755737	-74.03588760058483	40.6191510676	-74.0358876006
▦	🗺	4be319b321d5a59352311811	40.618838900608	-73.99010896682735	40.6190059409	-73.990374726

Table 3:



Venue Information

Foursquare Venue ID	49bbd6c0f964a520f4531fe3
Name	Pearl Art & Craft Supply
Address	308 Canal St (btwn Broadway & Mercer), New York, NY 10013, United States
Foursquare URL	https://foursquare.com/v/pearl-art--craft-supply/49bbd6c0f964a520f4531fe3

New Location

Latitude	Longitude
40.7198103755	-74.0025810321

Old Location

Latitude	Longitude
40.719726415811074	-74.00247199946715

Checkins

Latitude	Longitude
40.7198103755	-74.0025810321
40.7198103755	-74.0025810321
40.7198103755	-74.0025810321
40.7198103755	-74.0025810321
40.7198103755	-74.0025810321

Table 4:

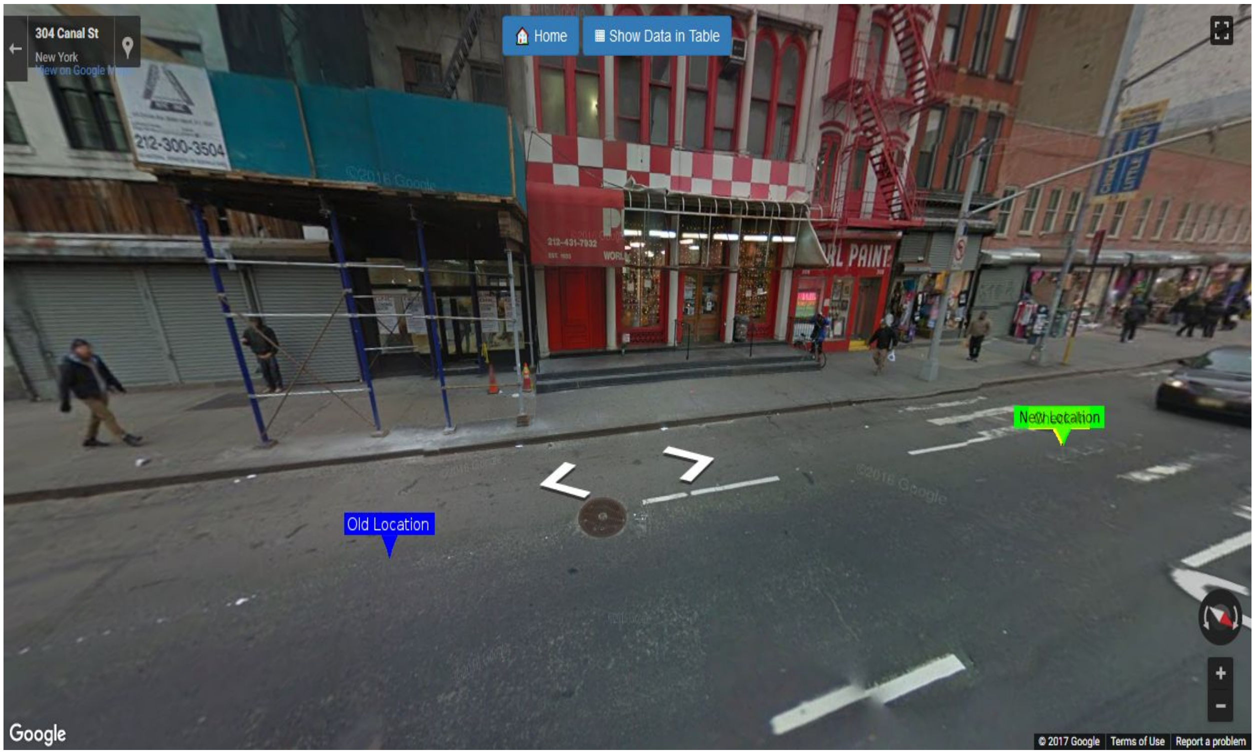


Fig 15:

Future Work

- On 4th April 2017, Google shut down its current procedure for map corrections.
- We have an idea about how our algorithm can be used to solve this problem in an efficient and sustainable way.
- The only requirement for our algorithm to work is a correct dataset and google has all the resources required to come up with one.
- So , how can Google create a dataset with correct locations of all erroneously marked location.
- Google can launch an app in which any individual can apply for a change in location for any given location.
- Google keeps track of location of G-Maps users.So,whenever there is a request for a location change Google can send out survey form to 15-20 random people who live near that location for authentication of the query.
- If all people agree to the change in location then google should proceed and update the dataset for a change.
- Once the locations to be updated are in the dataset, our algorithm can be used to update the locations.
- And this approach can be used to attain a dynamic map correction and this can make Google map as accurate as possible.

Why should people file a query??

- Google can provide people who post a correct location change with incentives like space in Google Drive or benefit on google products.
- They can provide them with some artificial money say credit points which can be used to get benefit from any of google services.
- We think If google uses this approach Google Maps will never ever have any competitor.

Difficulties

- Had to throw away all the work done in first few weeks due to anonymized foursquare dataset
- Not able find a non-anonymized dataset with all the attributes required,Had to build a dataset using the attributes from NYC and Tokyo Check-in Dataset and Used the Foursquare API to fetch other Required attributes
- Algorithm we designed at first was prone to outliers and hence not so accurate, that resulted in redesign of algorithm
- During Deployment of Visualisation ToolKit, we have to use Mobile data because SSH, Git & FTP ports are blocked on University Network

References

Xavier, Emerson, Francisco J. Ariza-López, and Manuel A. Ureña-Cámara. "A Survey of Measures and Methods for Matching Geospatial Vector Datasets." *ACM Computing Surveys (CSUR)* 49.2 (2016): 39.

Bruns, Tom, and Max Egenhofer. "Similarity of spatial scenes." Seventh international symposium on spatial data handling. Delft, The Netherlands, 1996.

Kolahdouzan, M. R., et al. "GeoMatchMaker: automatic and efficient matching of vector data with spatial attributes in unknown geometry systems." *Proc., UCGIS 2005 Summer Assembly* (2005).

Du, Heshan, et al. "Geospatial information integration for authoritative and crowd sourced road vector data." *Transactions in GIS* 16.4 (2012): 455-476.

Geocrowdsourcing and accessibility for dynamic environments H Qin, RM Rice, S Fuhrmann, MT Rice, KM Curtin… - GeoJournal, 2016

Zook, Matthew, et al. "Volunteered geographic information and crowdsourcing disaster relief: a case study of the Haitian earthquake." World Medical & Health Policy 2.2 (2010): 7-33.

"API Endpoints". *Developer.foursquare.com*. N.p., 2017. Web. 20 Mar. 2017.

Dingqi Yang, et al. "Modeling User Activity Preference By Leveraging User Spatial Temporal Characteristics In Lbsns". *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.1 (2015): 129-142. Web.

"Earth Ellipsoid". *En.wikipedia.org*. N.p., 2017. Web. 20 Mar. 2017.

"Earth Radius". *En.wikipedia.org*. N.p., 2017. Web. 20 Mar. 2017.

"Electron0zero/Geocorrect". *GitHub*. N.p., 2017. Web. 20 Mar. 2017.

"Geopy/Geopy". *GitHub*. N.p., 2017. Web. 20 Mar. 2017.

Illustration Of Great-Circle Distance.. 2017. Web. 20 Mar. 2017.

“Calculating Latitude/Longitude X Miles From Point?". *Gis.stackexchange.com*. N.p., 2017. Web. 20 Mar. 2017.

"Understanding Latitude And Longitude". *Learner.org*. N.p., 2017. Web. 20 Mar. 2017.

"Google Maps Embed API | Google Developers". Google Developers. N.p., 2017. Web. 26 Apr. 2017.

"Welcome | Flask (A Python Microframework)". Flask.pocoo.org. N.p., 2017. Web. 26 Apr. 2017.

"Mbr/Flask-Bootstrap". GitHub. N.p., 2017. Web. 26 Apr. 2017.

"Flask-Googlemaps 0.2.4 : Python Package Index". Pypi.python.org. N.p., 2017. Web. 26 Apr. 2017.