Geospatial data management project

Pietro Barone
975940

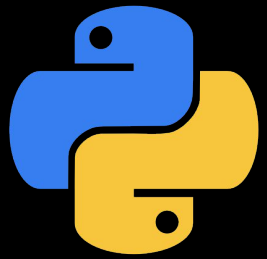
2020/2021

The assignment

Two main things to do:

1 - Collect a trajectory with a tracker, with a visual evidence of at least one stop

2 - Implement the staypoint algorithm to detect stops in the trajectory



The implementation (phase 2)

Algorithm StayPoint_Detection(*P*, *distThreh*, *timeThreh*)

Input: A GPS log *P*, a distance threshold *distThreh*
and time span threshold *timeThreh*
Output: A set of stay points $SP=\{S\}$

1. i=0, *pointNum* = |P|; //the number of GPS points in a GPS logs
2. **while** $i <$ *pointNum* **do**,
3. j:=i+1;
4. **while** $j <$ *pointNum* **do**,
5. *dist*=Distance(p_i, p_j); //calculate the distance between two points
6. **if** *dist* > *distThreh* **then**
7. $\Delta T = p_j.T - p_i.T$; //calculate the time span between two points
8. **if** ΔT > *timeThreh* **then**
9. *S.coord*=ComputMeanCoord($\{p_k \mid i <= k <= j\}$)
10. $S.arvT = p_i.T$; $S.levT = p_j.T$;
11. *SP*.insert(*S*);
12. i:=j; break;
13. j:=j+1;
14. **return** *SP*.

```
def stayPoint( distThres = 20, timeThres = 120):
    cur.execute('select * from walk3857')
    points = cur.fetchall()
    usedpointsID=1
    pointNum = len(points)
    i = 0
    while i < pointNum-1:
        j = i+1
        while j < pointNum:
            pi = points[i]
            pj = points[j]
            dist = getDistance(pi[2],pj[2])
            if dist > distThres:
                t_i=getEpoch(pi[1])
                t_j=getEpoch(pj[1])
                deltaT = t_j - t_i
                if deltaT > timeThres:
                    coords = computMeanCoord(points[i:j+1])
                    geom = getGeom(coords)
                    arriveTime=getTimestamp(t_i)
                    leaveTime=getTimestamp(t_j)
                    insertStaypoint((geom, arriveTime, leaveTime))
                    for ps in points[i:j+1]:
                        cur.execute('INSERT INTO usedPoints VALUES\
                            (%s, %s, %s)',[ps[0],ps[2],usedpointsID])
                    conn.commit()
                    usedpointsID+=1
                break
            j += 1
        i = j
```

The implementation (phase 2)

I of course use a database to store all the points.

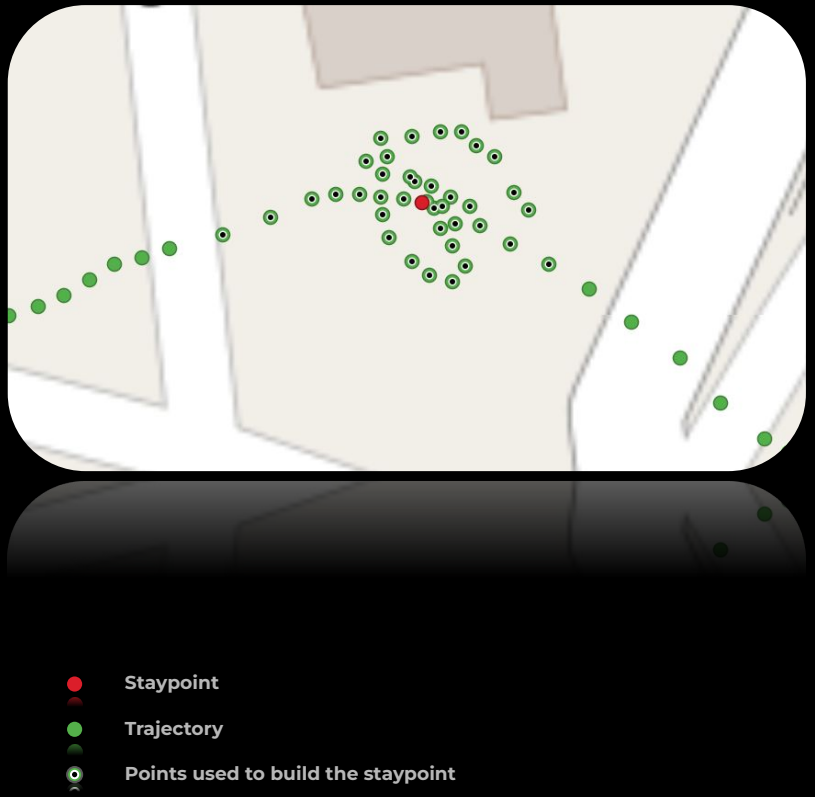
Note also that instead of python libraries I use the operations offered by postgis.

To add accuracy all the coordinates are transformed in EPSG 3857.

```python
def getDistance(geom1,geom2):
    cur.execute("select st_distance(%s,%s)",(geom1,geom2))
    return cur.fetchone()[0]
def getLat(geom):
    cur.execute("select st_y(%s)",[geom])
    return cur.fetchone()[0]
def getLong(geom):
    cur.execute("select st_x(%s)",[geom])
    return cur.fetchone()[0]
def getEpoch(timestamp):
    cur.execute("select extract(epoch from timestamp %s)",[str(timestamp)])
    return cur.fetchone()[0]
def getTimestamp(epoch):
    cur.execute("select to_timestamp(%s)",[epoch])
    return cur.fetchone()[0]
def getGeom(coordinates):
    cur.execute("select st_geomfromtext('Point (%s %s)')",coordinates)
    return cur.fetchone()[0]
def computMeanCoord(gpsPoints):
    long = 0.0
    lat = 0.0
    for point in gpsPoints:
        long += float(getLong((point[2])))
        lat += float(getLat((point[2])))
    return (long/len(gpsPoints), lat/len(gpsPoints))
```

Customization

To add some customization to the project I've also created a table to store all the points that were useful to calculate the staypoints. Those are the ones on which I calculate the mean coordinates of the new staypoint.



Results

Despite the accuracy of the tracking technology (GPS) the results are pretty accurate.

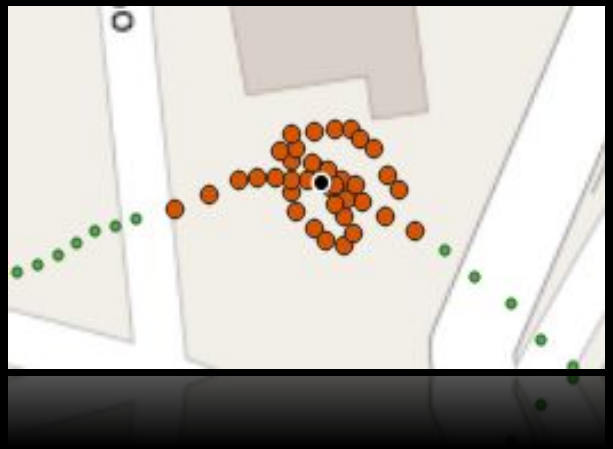
Staypoint

Trajectory

Point used to build each staypoint

A comparison with DBSCAN

I thought to apply this kind of clustering technique to find each staypoint by creating the clusters and then for each one make a centroid out of them.



How I did it

1 - First I had to create the clusters using the tool

2 - The second thing to do was to categorized the new table to visualize it better, giving different colors to each cluster

3 - I had to create a multipoint from the points of each cluster.

4 - Then I had to find the centroid of each cluster

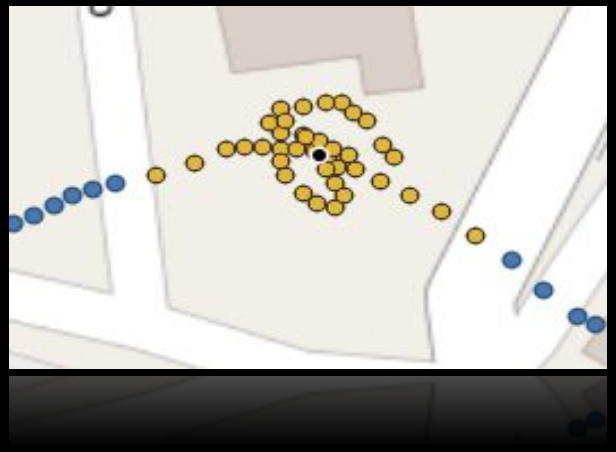
A comparison with DBSCAN

DBSCAN

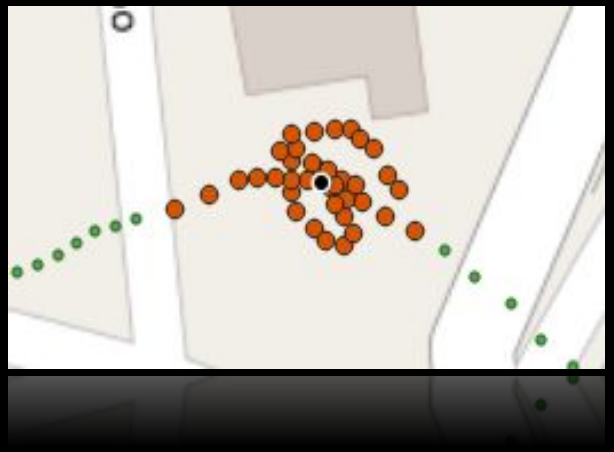
Staypoint algorithm

A comparison with DBSCAN

DBSCAN



Staypoint algorithm



Again we can see how similar and comparable are the two results.

Thanks for the attention