EE558 : A Network Tour of Data Science

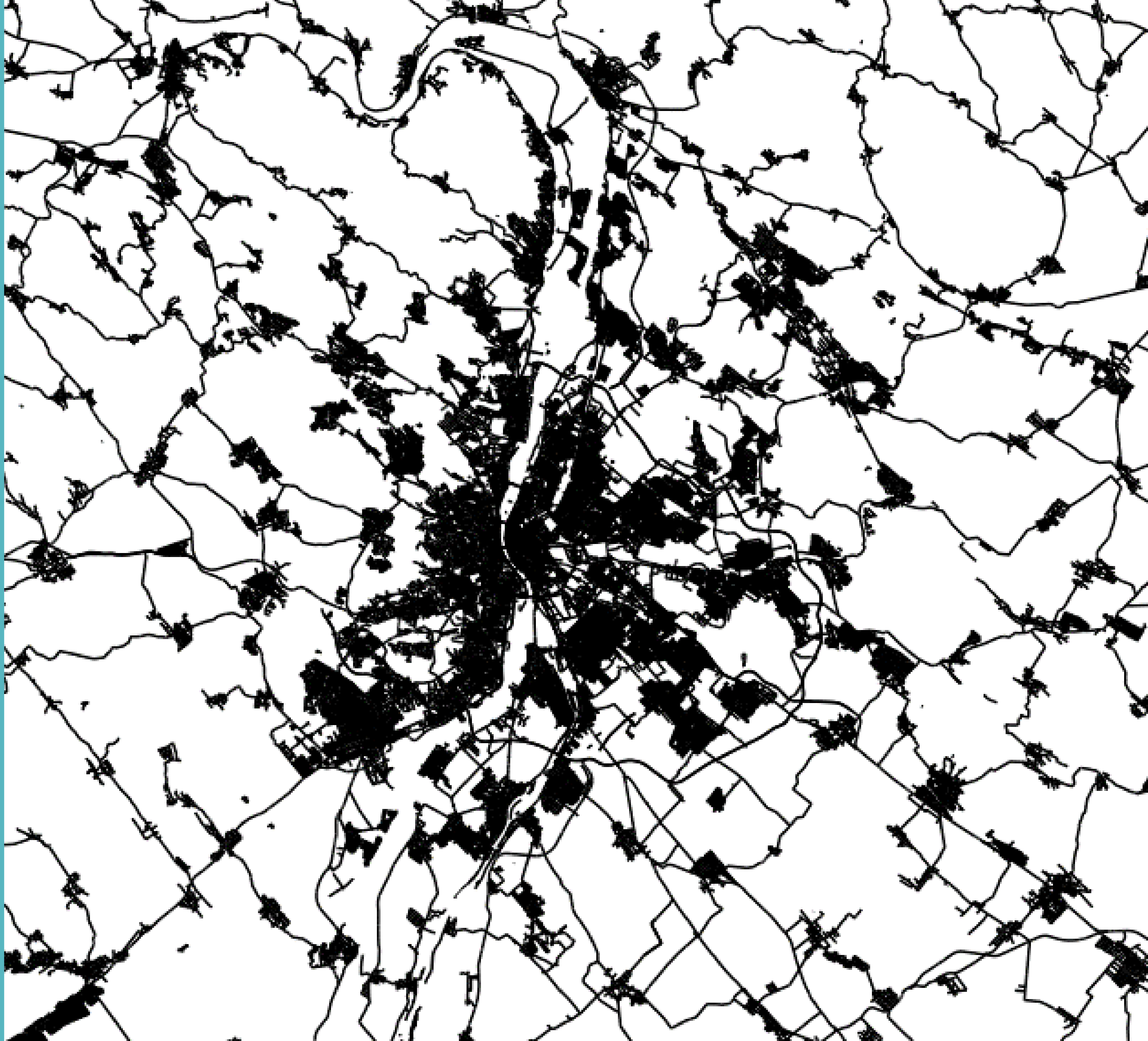
Professors : Frossard Pascal, Vandergheynst Pierre

BUDA + PEST = BUDAPEST

Dorian Herle, Giulio Masinelli

Silvio Zanoli, Sohyeong Kim

Jan 24th, 2018



1. Introduction

2. Data Collection
   - Getting the Data
   - Exploring the Data
   - Cleaning the Data

3. Building Features
   - Feature Extraction
   - Feature Creation

4. Graphing
   - Graph Creation
   - Graph Analysis

5. Results
   & Discussion

Introduction

We wanted to test a hypothesis……

> “ **Old parts** of a city can be distinguished
> from **new parts** of a city,
> solely based on **the road network.** ”

Data Collection

- Data Exploration

OSM(Open Street Map) Dataset

Format : Shapefiles (.shp)

- Geospatial vector data format for geographic information system(GIS) software
- Describes geometries as either 'points', **'polylines',** or 'polygons’

Roads

Source : Mapzen (https://mapzen.com/data/metro-extracts)

The following road networks were extracted for the project

- Budapest
- Bern
- Bologna
- New York
- Lama Mocogno

Data Collection

- Data Exploration

GeoPandas

- GeoPandas extends the datatypes used by pandas to allow spatial operations on geometric types.
- It combines the capabilities of pandas and shapely, providing geospatial operations in pandas and a high-level interface to multiple geometries to shapely.

	id	osm_id	type	name	tunnel	bridge	oneway	ref	z_order	access	service	class	geometry
0	1	2955839.0	rail	no_name	0	0	0	None	7	None	None	railway	LINESTRING (834180.7799455991 5913106.66249309...
1	2	3230982.0	residential	Hochfeldstrasse	0	0	0	None	3	None	None	highway	LINESTRING (826867.6168769542 5903563.51628549...
2	3	3728938.0	primary	Bernstrasse	1	0	0	1;12	-14	None	None	highway	LINESTRING (834386.4750492289 5913916.23293858...

BUDA + PEST
= BUAPEST

Data Collection

- Data Exploration

GeoPandas

- GeoPandas extends the datatypes used by pandas to allow spatial operations on geometric types.
- It combines the capabilities of pandas and shapely, providing geospatial operations in pandas and a high-level interface to multiple geometries to shapely.

	id	osm_id	type	name	tunnel	bridge	oneway	ref	z_order	access	service	class	geometry
0	1	2955839.0	rail	no_name	0	0	0	None	7	None	None	railway	LINESTRING (834180.7799455991 5913106.66249309...
1	2	3230982.0	residential	Hochfeldstrasse	0	0	0	None	3	None	None	highway	LINESTRING (826867.6168769542 5903563.51628549...
2	3	3728938.0	primary	Bernstrasse	1	0	0	1;12	-14	None	None	highway	LINESTRING (834386.4750492289 5913916.23293858...

Data Collection

- Data Exploration

Raw data of the roads

BUDA + PEST
= BUAPEST

Data Collection

- Data Cleaning

Avoid unrelated data

- Only keep relevant classes to roads ➔ **Highway**
- Within the class – Highway, choose the categories
  ; residential, primary, secondary, living_street, trunk, pedestrian, tertiary

v · d · e	Highways
Roads	motorway, trunk, primary, secondary, tertiary, unclassified, residential, service
Link Roads	motorway_link, trunk_link, primary_link, secondary_link, motorway_junction
Special	living street, pedestrian, bicycle road, track, bus guideway, raceway, road, construction, escape
Paths	footway, cycleway, path, bridleway, steps, escalator
Sidewalks	sidewalk, cycleway lane, cycleway tracks, bus and cyclists, lanes:psv, busway
Lanes	number of lanes, direction instructions ("turn lanes"), signposts
See also	conditional restrictions, turn restrictions, highway tagging samples

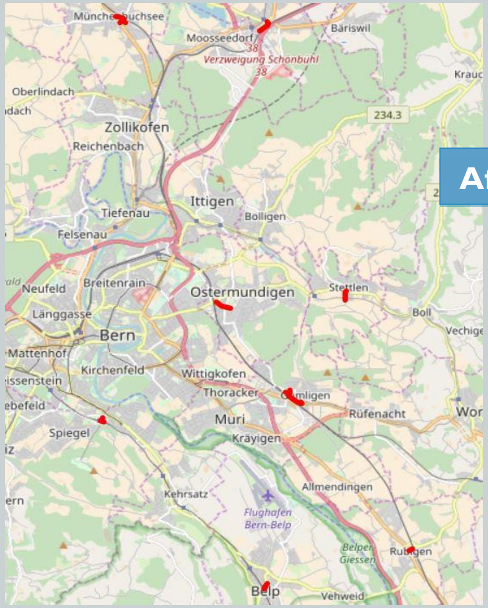
Extract from OSM Wiki: https://wiki.openstreetmap.org/wiki/Highways

Data Collection

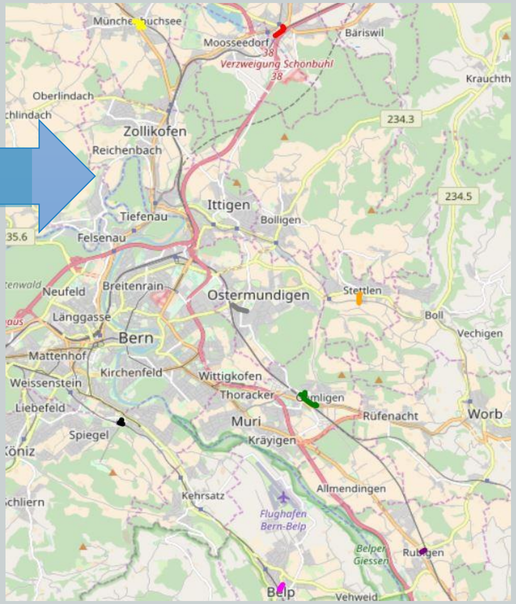
- Data Cleaning

Differentiating the roads

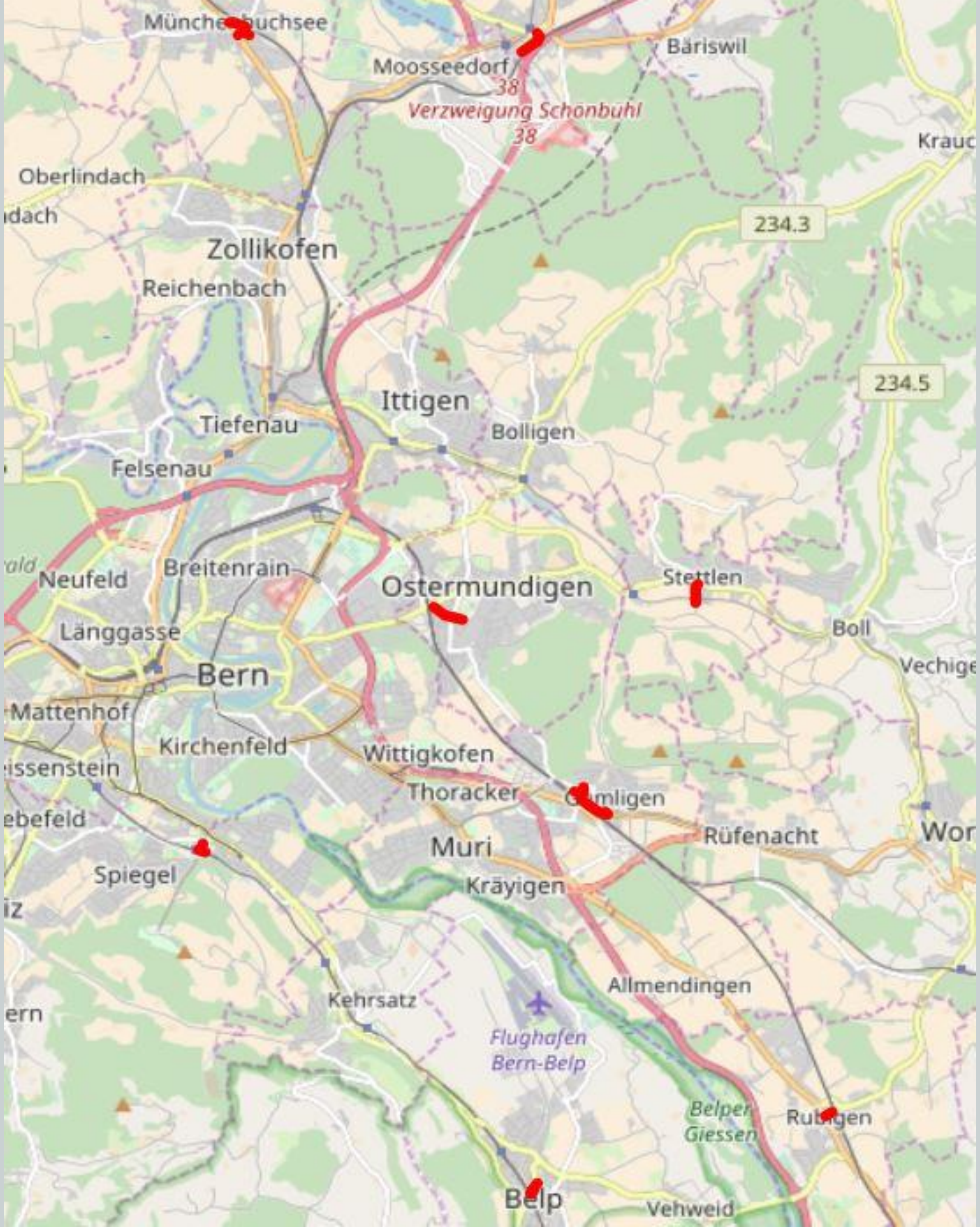
- Splitting different roads with same name and renaming them
- One road ➔ One name



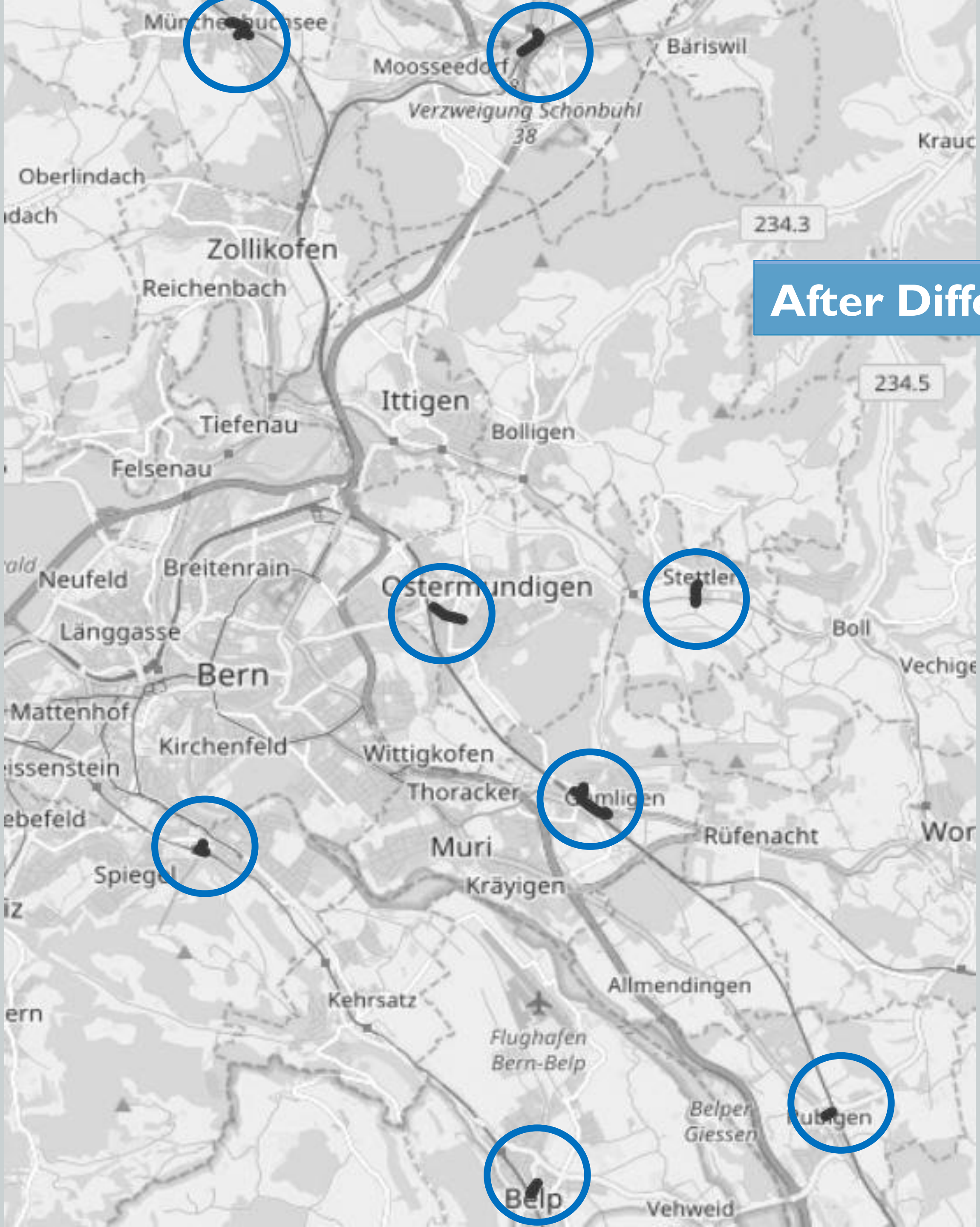
After Differentiating



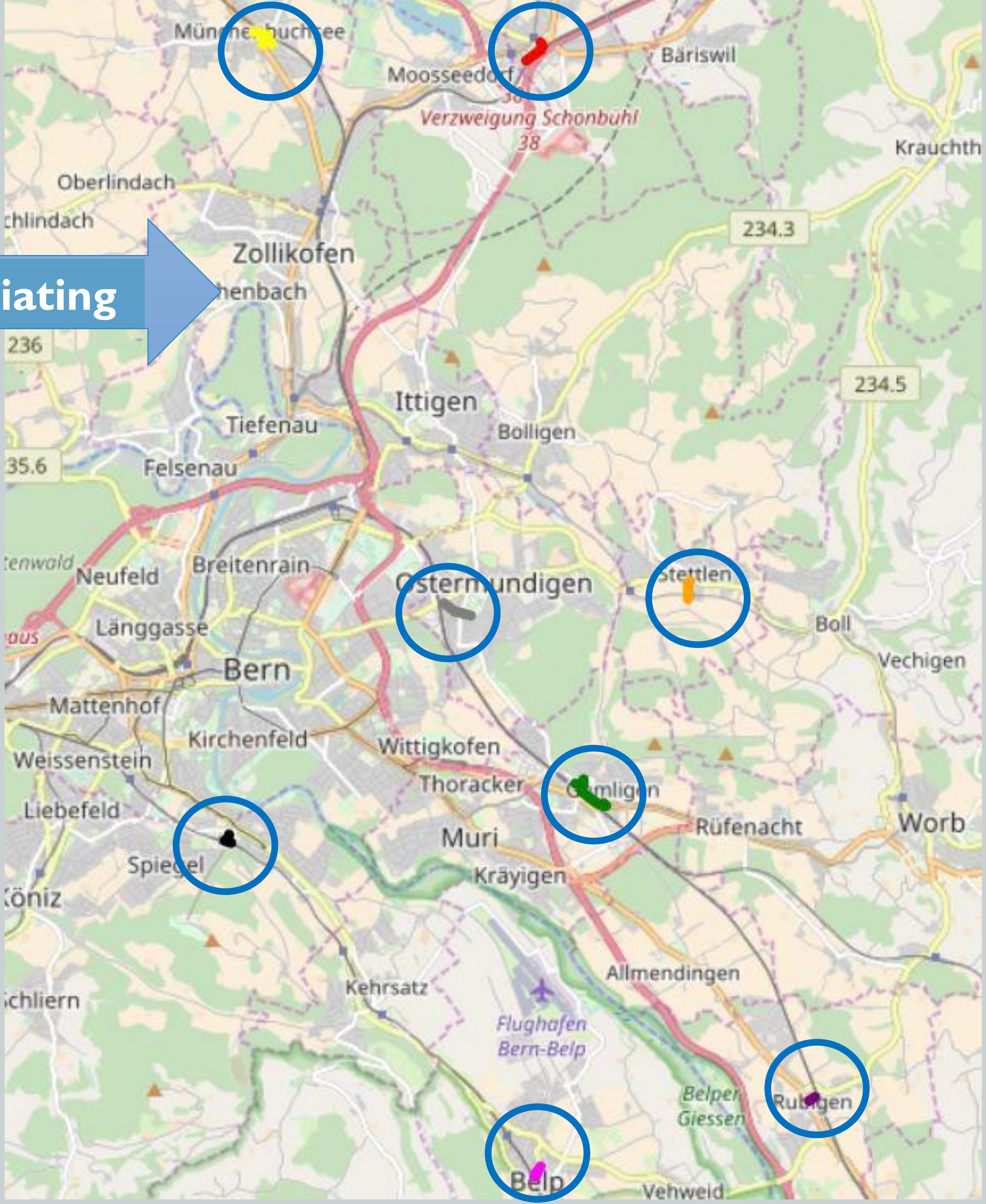
BUDA + PEST
= BUAPEST



Münchenbuchsee
Moosseedorf
38
Verzweigung Schönbühl
38
Bäriswil
Oberlindach
Zollikofen
Reichenbach
234.3
234.5
Ittigen
Bolligen
Tiefenau
Felsenau
Neufeld
Breitenrain
Ostermundigen
Stettlen
Boll
Länggasse
Bern
Mattenhof
Kirchenfeld
Wittigkofen
Thoracker
Rüfenacht
Muri
Spiegel
Kräyigen
Allmendingen
Kehrsatz
Flughafen Bern-Belp
Belper Giessen
Belp
Vehweid



After Differentiating



Building Features

How to deal with the data

- In the CLEANED DATASET there is a biunivocal correspondence between street name and the specific street.
- Each street has its coordinates stored in one or more LineString objects.
- To preserve the angles and the shapes of small objects, and to be able to obtain distances in meters, the whole dataset was re-projected using the Mercator projection.

Cleaned Data

BUDA + PEST
= BUAPEST

Building Features

How to deal with the data

- In the CLEANED DATASET there is a biunivocal correspondence between street name and the specific street.
- Each street has its coordinates stored in one or more LineString objects.
- To preserve the angles and the shapes of small objects, and to be able to obtain distances in meters, the whole dataset was re-projected using the Mercator projection.

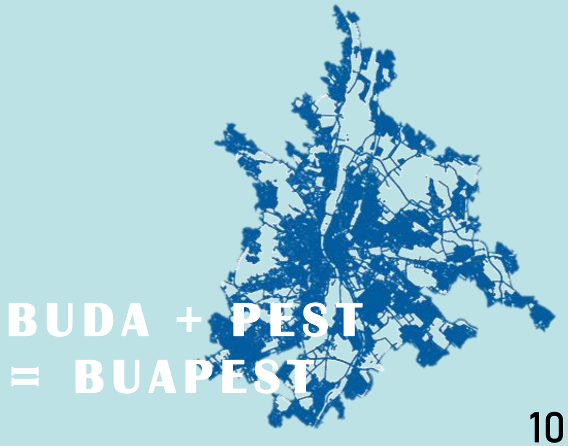
Cleaned Data

- Length
- Flight length
- Flight length – length Ratio
- Curvature
- One way
- Derivative

Building Features

- Length
- Flight length
- Flight length – length Ratio
- Curvature
- One way
- Derivative

The length of a street can be easily obtained exploiting the length() method of the LineString class and by summing up the lengths obtained for each of the LineStrings the street is made up.

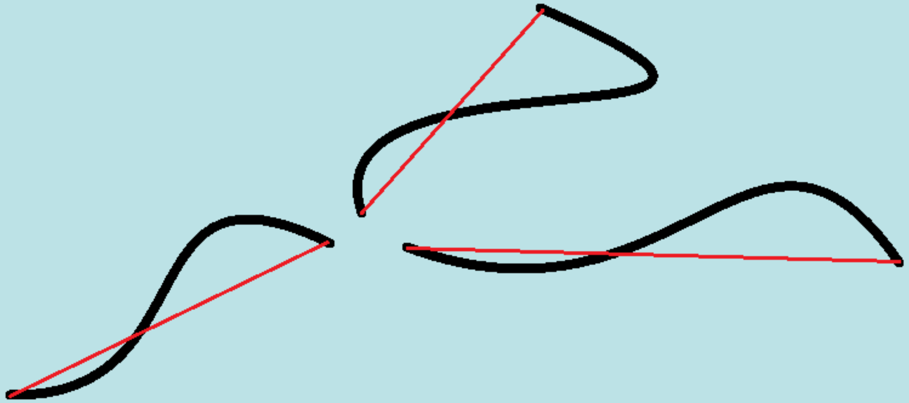


Building Features

- Length
- Flight length
- Flight length – length Ratio
- Curvature
- One way
- Derivative

The length of a street can be easily obtained exploiting the length() method of the LineString class and by summing up the lengths obtained for each of the LineStrings the street is made up.

The flight length is calculated as the Euclidean distance between the initial point of the street and the final one. In case the LineStrings that form the street are not contiguous (e.g. if some points are missing), the flight length is computed for each of the connected lines and the total one is the sum of all contributions.



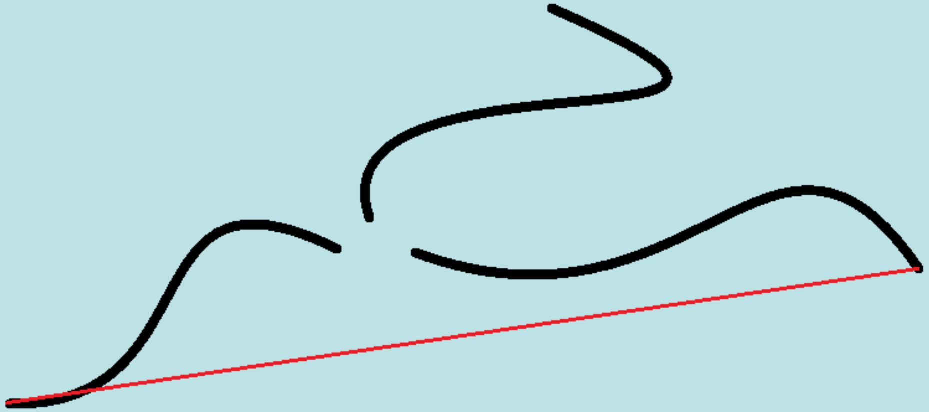
BUDA + PEST
= BUAPEST

Building Features

- Length
- Flight length
- Flight length – length Ratio
- Curvature
- One way
- Derivative

The length of a street can be easily obtained exploiting the length() method of the LineString class and by summing up the lengths obtained for each of the LineStrings the street is made up.

The flight length is calculated as the Euclidean distance between the initial point of the street and the final one. In case the LineStrings that form the street are not contiguous (e.g. if some points are missing), the flight length is computed for each of the connected lines and the total one is the sum of all contributions.



Building Features

- Length
- Flight length
- Flight length – length Ratio
- Curvature
- One way
- Derivative

The length of a street can be easily obtained exploiting the length() method of the LineString class and by summing up the lengths obtained for each of the LineStrings the street is made up.

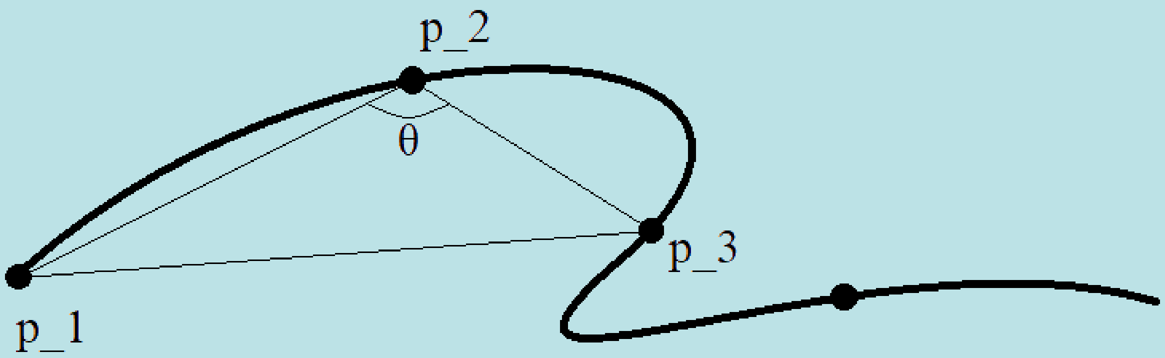
The flight length is calculated as the Euclidean distance between the initial point of the street and the final one. In case the LineStrings that form the street are not contiguous (e.g. if some points are missing), the flight length is computed for each of the connected lines and the total one is the sum of all contributions.

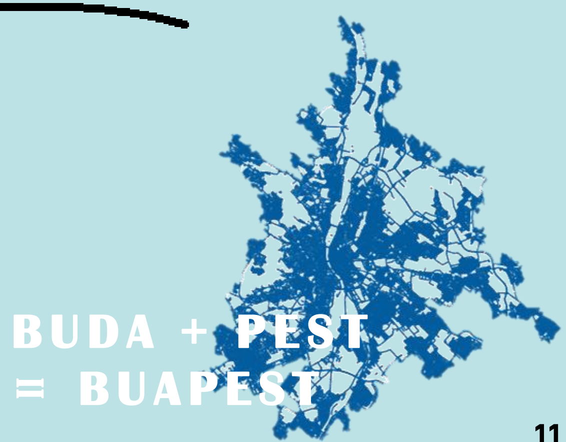
The flight length – length ratio is basically the quotient of the two previously computed distances. In many cases, it is an efficient indicator of the global curvature of the street.

Building Features

- Length
- Flight length
- Flight length – length Ratio
- Curvature
- One way
- Derivative

In order to measure the straightness of a street, we computed an angle for every consecutive triplets of points:

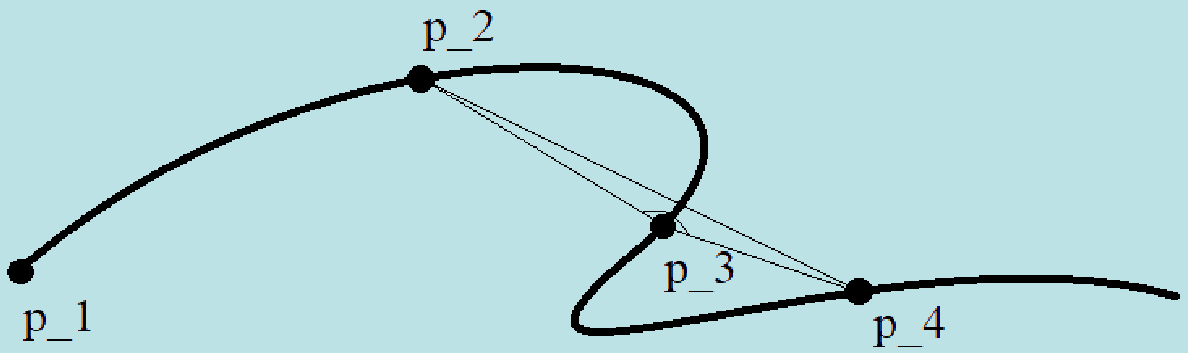




Building Features

- Length
- Flight length
- Flight length – length Ratio
- Curvature
- One way
- Derivative

In order to measure the straightness of a street, we computed an angle for every consecutive triplets of points:



BUDA + PEST
= BUAPEST

Building Features

Length

Flight length

Flight length – length Ratio

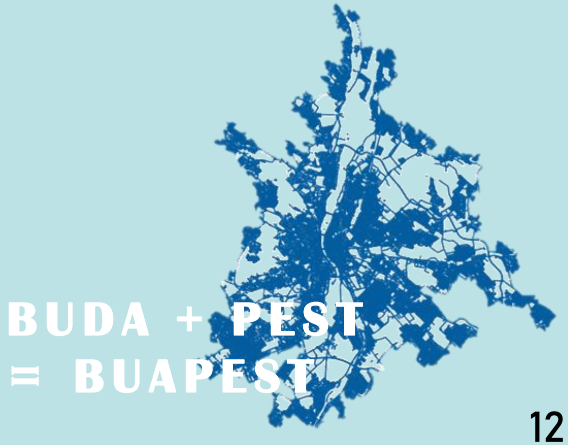
Curvature

One way

Derivative

By doing this, an angle between 0 and π is computed for every consecutive triplets of points. Since we need to compare the straightness of streets with different lengths, we have to aggregate these coefficients.

To have a fixed representation for each street, we exploit 7 summary statistics: minimum, maximum, median and the first 4 moments (mean, standard deviation, skew and kurtosis)



Building Features

- Length
- Flight length
- Flight length – length Ratio
- Curvature
- One way
- Derivative

One way is the Boolean value that is one in case the street traffic has to moves in a single direction.

BUDA + PEST
= BUAPEST

Building Features

- Length
- Flight length
- Flight length – length Ratio
- Curvature
- One way
- Derivative

One way is the Boolean value that is one in case the street traffic has to moves in a single direction.

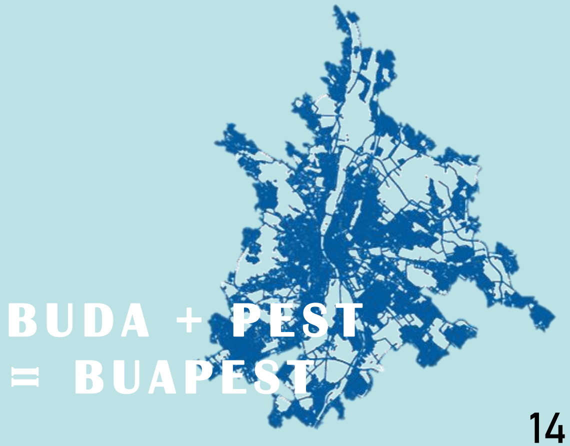
It is also possible to compute the derivative of the coordinate points of each street. Again, as the derivate of each line has the same number of points of the line itself, to be able to compare derivates of different streets, we aggregate the coefficients by using the previous described 7 summary statistics (minimum, maximum, median and the first 4 moments).

BUDA + PEST
= BUAPEST

Building Features

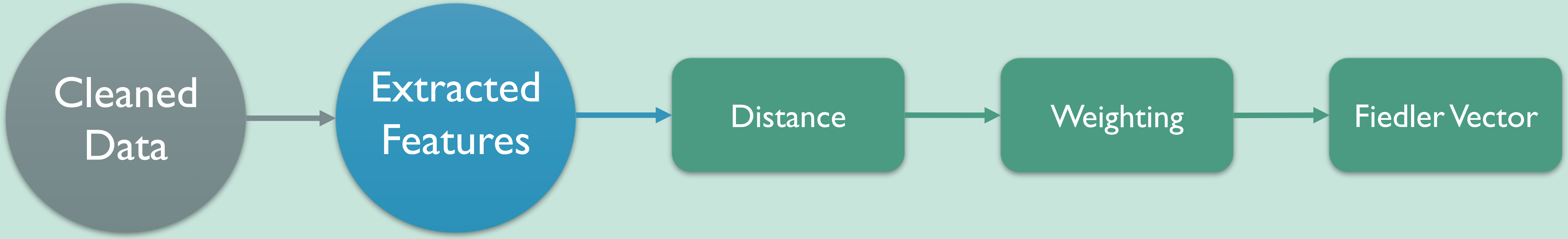
- Length
- Flight length
- Flight length – length Ratio
- Curvature
- One way
- Derivative

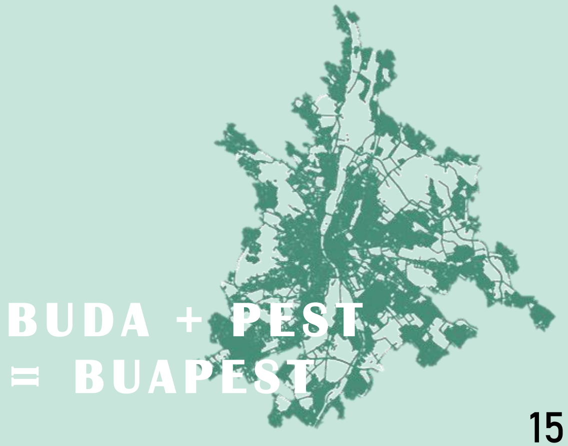
25 features in total!



Graphing

Once obtained the said features we can use them to find if there are some clusters present.
And we need a GRAPH !





Graphing

The first step is to find a similarity measurement between data points and define weighting coefficients for each parameter.

First Guess : Use classical Euclidian distance

Distance

Weighting

Fiedler Vector

```
#tuning parameters
l=2          #coeff. for length
c=2          #coeff. for mean curvature
f=2          #coeff. for flight-distance
f_r=3.5      #coeff. for ratio flight-distance
o=1.5        #coeff. for one-way
s=1.5        #coeff. for std_curvature
mi=0.25      #coeff. for min curvature
ma=3.5       #coeff. for max curvature
skew=1       #coeff. for skew curvature
kurt=0.5     #coeff. for kurtosis curvature
med=1.5      #coeff. for median curvature
dexstd = 1   #coeff. for std_derivative_x
dexmed = 1   #coeff. for median_derivative_x
dexskew=0.5  #coeff. for skew_derivative_x
dexkurt=0.25 #coeff. for kurtosis derivative x
dexmi = 1    #coeff. for min derivative x
dexma = 1.5  #coeff. for max derivative x
dexmean = 1  #coeff. for mean derivative x
deymean = 1  #coeff. for mean derivative y
deymi = 1    #coeff. for min derivative y
deyma = 1.5  #coeff. for max derivative y
deystd = 1   #coeff. for std derivative y
deymed = 1   #coeff. for median derivative y
deyskew=0.5  #coeff. for skew derivative y
deykurt=0.25 #coeff. for kurtosis derivate y
bias=0.01    #generic bias
```

```
dfp['length']=dfp['length']*l
dfp['std_curvature']=dfp['std_curvature']*s
dfp['min_curvature']=dfp['min_curvature']*mi
dfp['max_curvature']=dfp['max_curvature']*ma
dfp['mean_curvature']=dfp['mean_curvature']*c
dfp['flight_distance']=dfp['flight_distance']*f
dfp['percentage_flight_distance']=dfp['percentage_flight_distance']*f_r
dfp['oneway']=dfp['oneway']*o
dfp['std_derivative_x'] = dfp['std_derivative_x']*dexstd
dfp['mean_derivative_x'] = dfp['mean_derivative_x']*dexmean
dfp['max_derivative_x'] = dfp['max_derivative_x']*dexma
dfp['min_derivative_x'] = dfp['min_derivative_x']*dexmi
dfp['std_derivative_y'] = dfp['std_derivative_y']*deystd
dfp['mean_derivative_y'] = dfp['mean_derivative_y']*deymean
dfp['max_derivative_y'] = dfp['max_derivative_y']*deyma
dfp['min_derivative_y'] = dfp['min_derivative_y']*deymi
dfp['skew_curvature'] = dfp['skew_curvature']*skew
dfp['kurtosis_curvature'] = dfp['kurtosis_curvature']*kurt
dfp['median_curvature'] = dfp['median_curvature']*med
dfp['skew_derivative_y'] = dfp['skew_derivative_y']*deyskew
dfp['kurtosis_derivative_y'] = dfp['kurtosis_derivative_y']*deykurt
dfp['median_derivative_y'] = dfp['median_derivative_y']*deymed
dfp['skew_derivative_x'] = dfp['skew_derivative_x']*dexskew
dfp['kurtosis_derivative_x'] = dfp['kurtosis_derivative_x']*dexkurt
dfp['median_derivative_x'] = dfp['median_derivative_x']*dexmed
```

Graphing

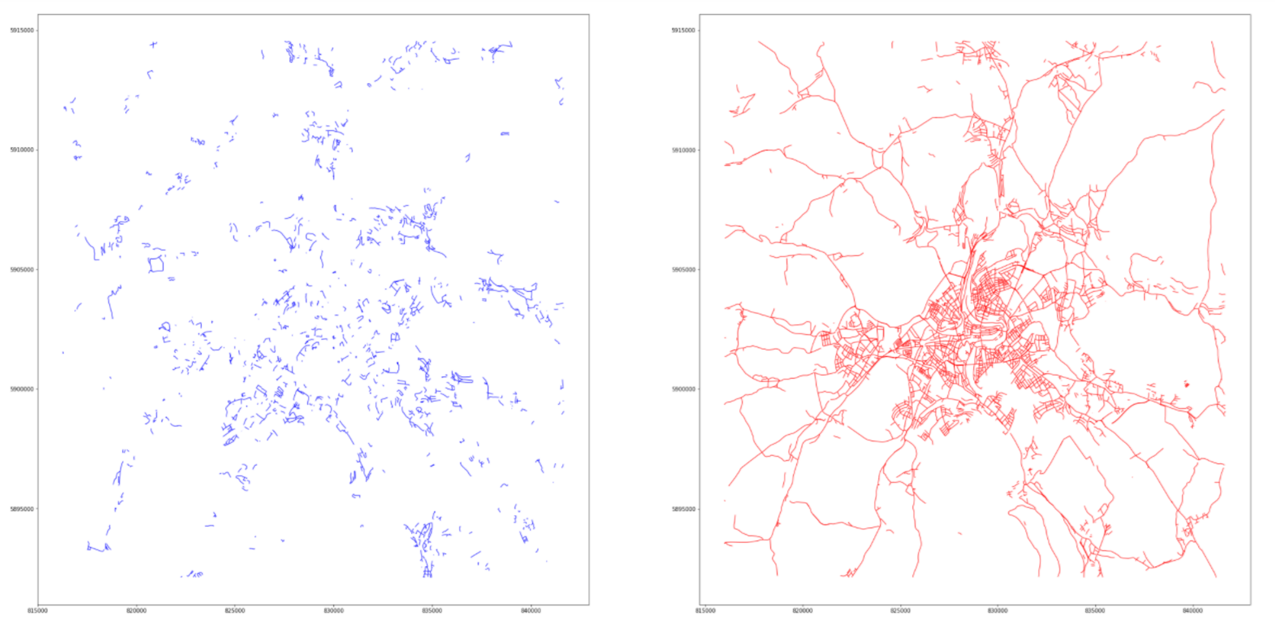
Distance

Weighting

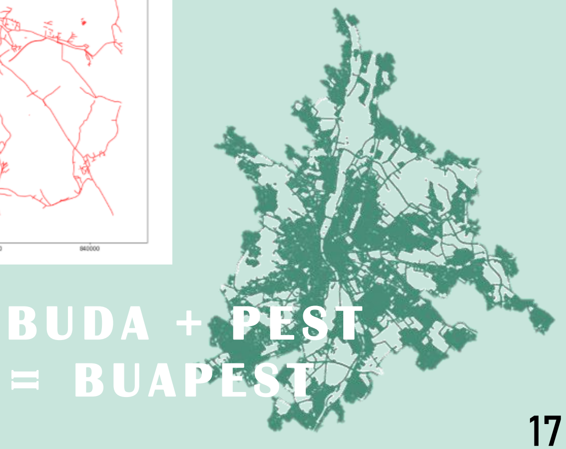
Fiedler Vector

The first step is to find a similarity measurement between data points and define weighting coefficients for each parameter.

First Guess : Use classical Euclidian distance



➔ Leads to this separation: not so good…



Graphing

The first step is to find a similarity measurement between data points and define weighting coefficients for each parameter.

Distance

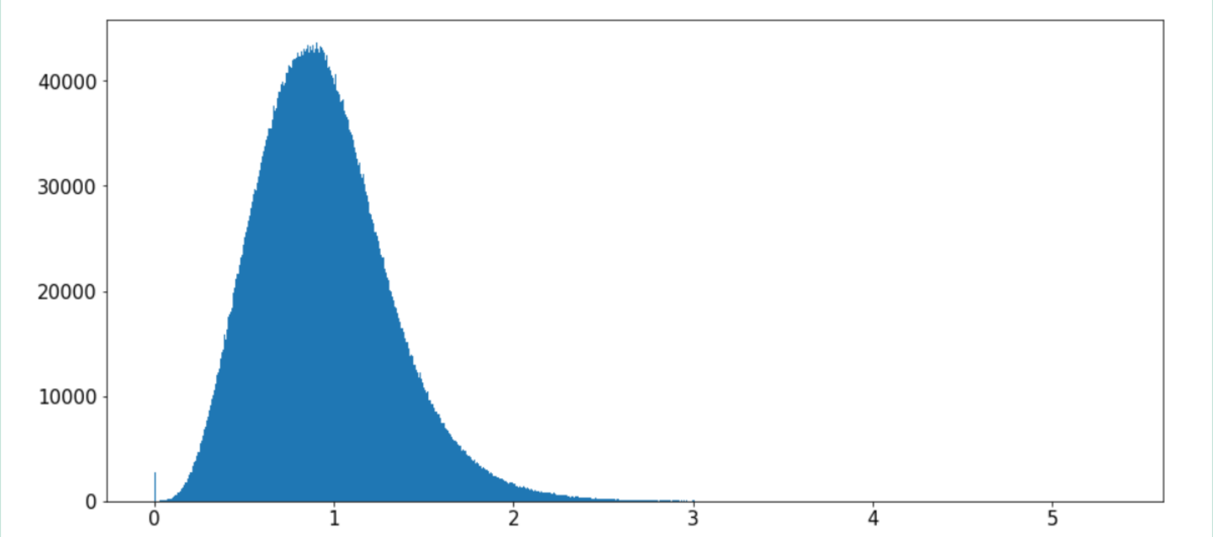
Weighting

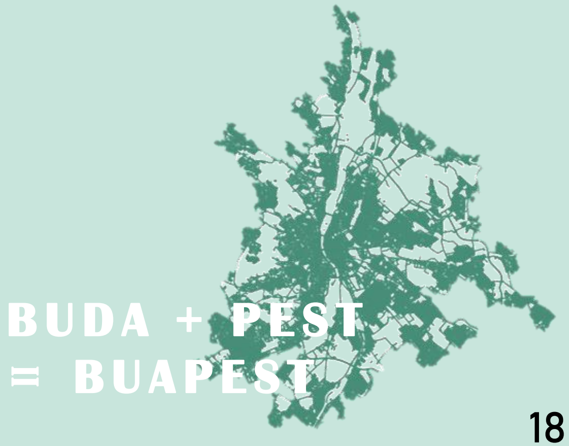
Fiedler Vector

Second Guess : Use Bray-Curtis distance

Given 2 vector u and v in R^n: $$d(u,v) = \frac{\sum_{i=1}^{n} |u_i - v_i|}{\sum_{i=1}^{n} |u_i + v_i|}$$

Distance distribution using the Bray-Curtis distance





Graphing

The second step is to map distance values in a weight matrix.
To do so, we need a decreasing function having maximum at 0.

Distance → Weighting → Fiedler Vector

Exponential Kernel

$$W(u,v) = e^{-\frac{d(u,v)^2}{\sigma^2}}$$

Graphing

Distance

Weighting

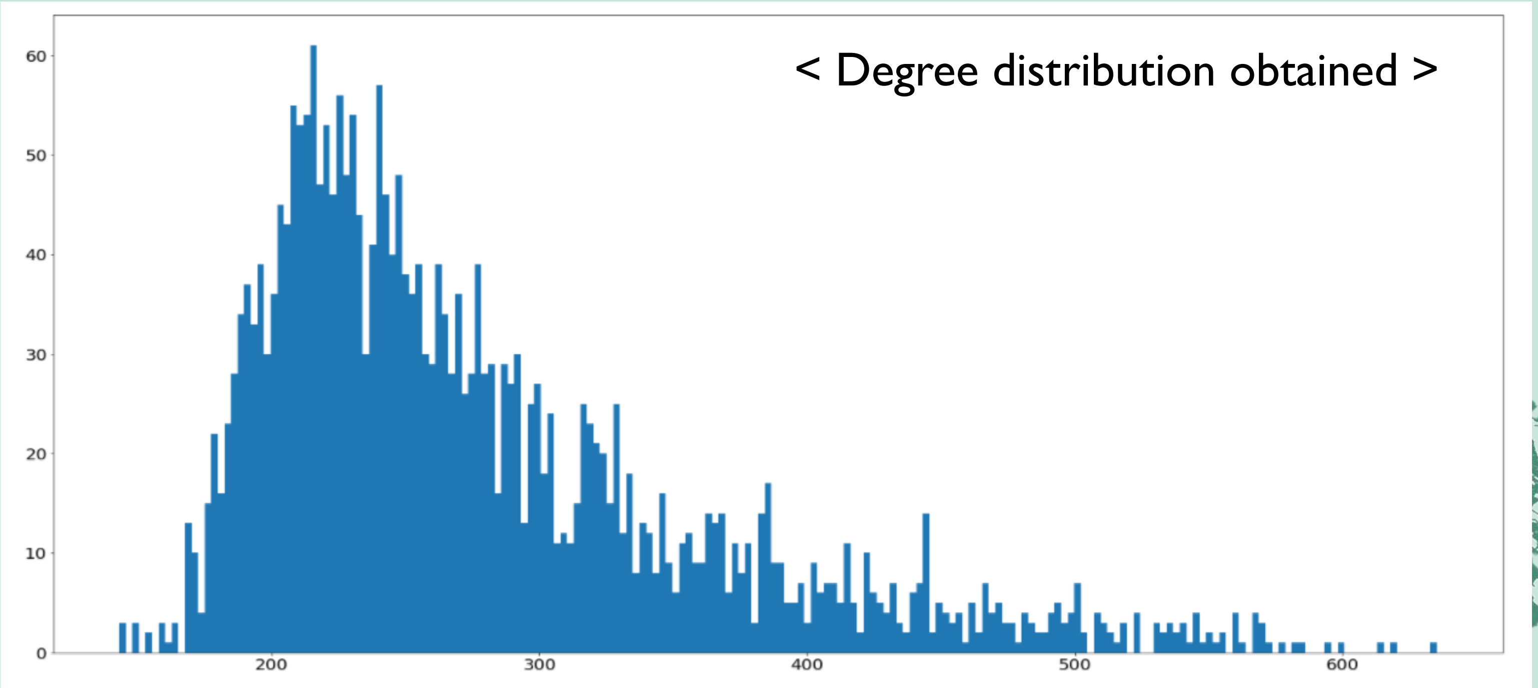
Fiedler Vector

The second step is to map distance values in a weight matrix. To do so, we need a decreasing function having maximum at 0.

To work with a sparse matrix we opted to sparsify the weight-matrix using a **KNN algorithm** and keeping only around 10% of the original entries.

Exponential Kernel

KNN algorithm



Graphing

The final step is to compute the Fiedler vector as the second eigen-vector of the Laplacian matrix.

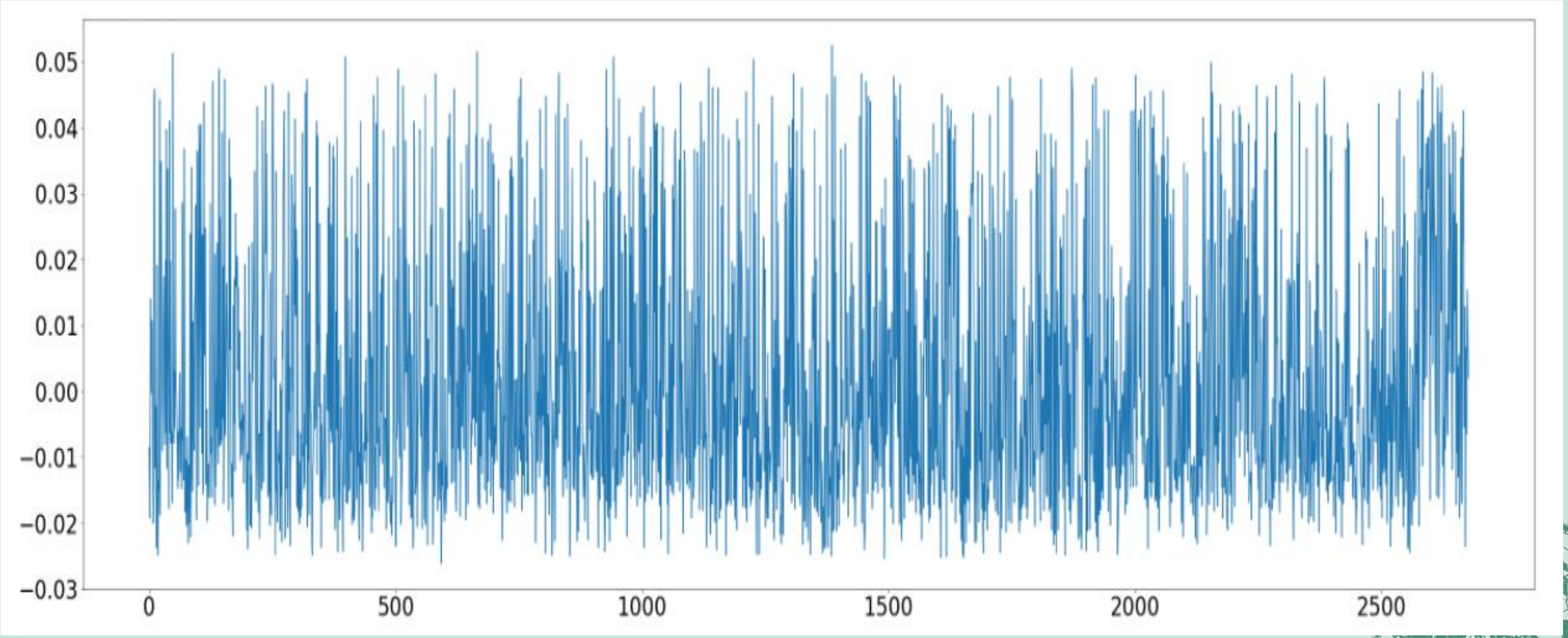
Distance

↓

Weighting

↓

Fiedler Vector



BUDA + PEST
= BUAPEST

Graphing

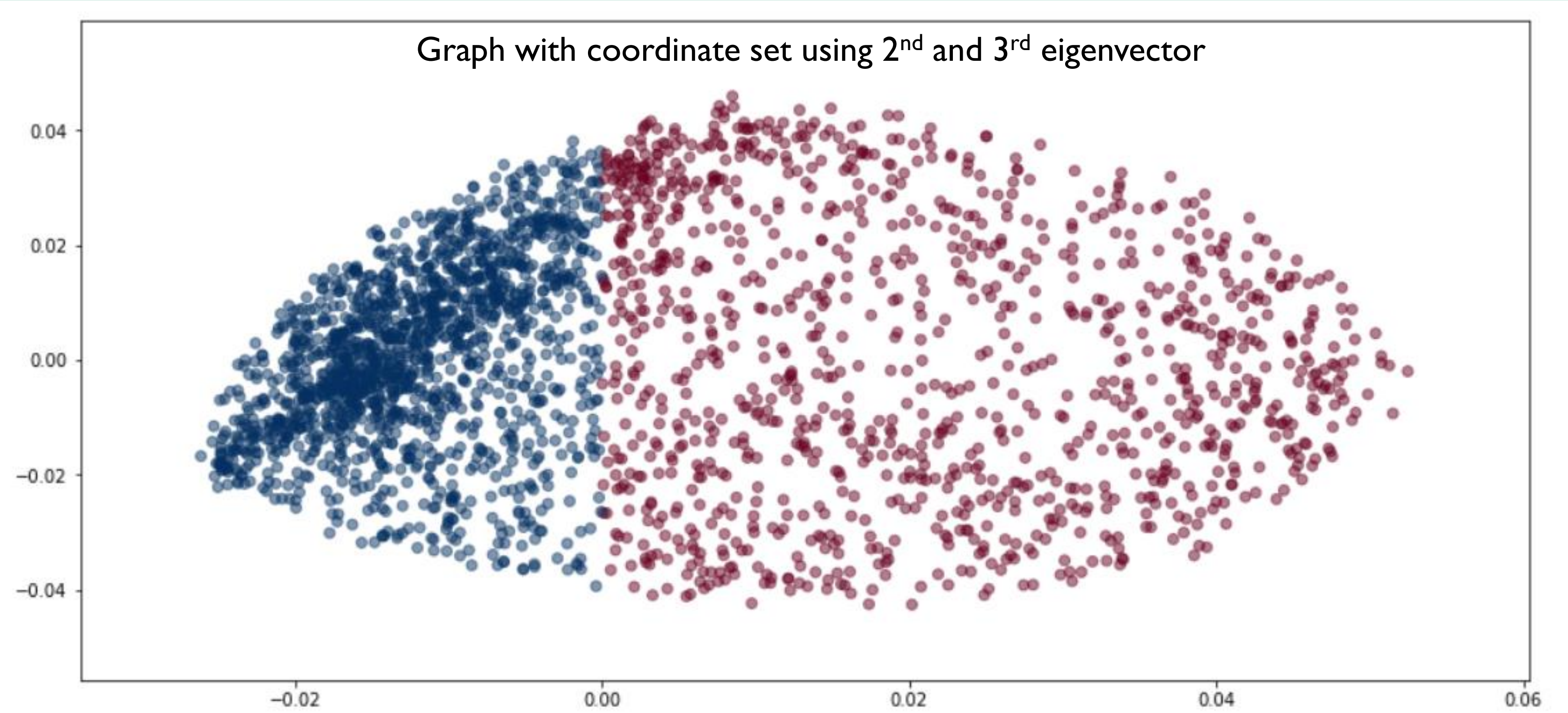
The final step is to compute the Fiedler vector as the second eigen-vector of the Laplacian matrix.

Fiedler Vector : Using the 0 as discriminant we can label the streets with different colors.

Distance

Weighting

Fiedler Vector



Graphing

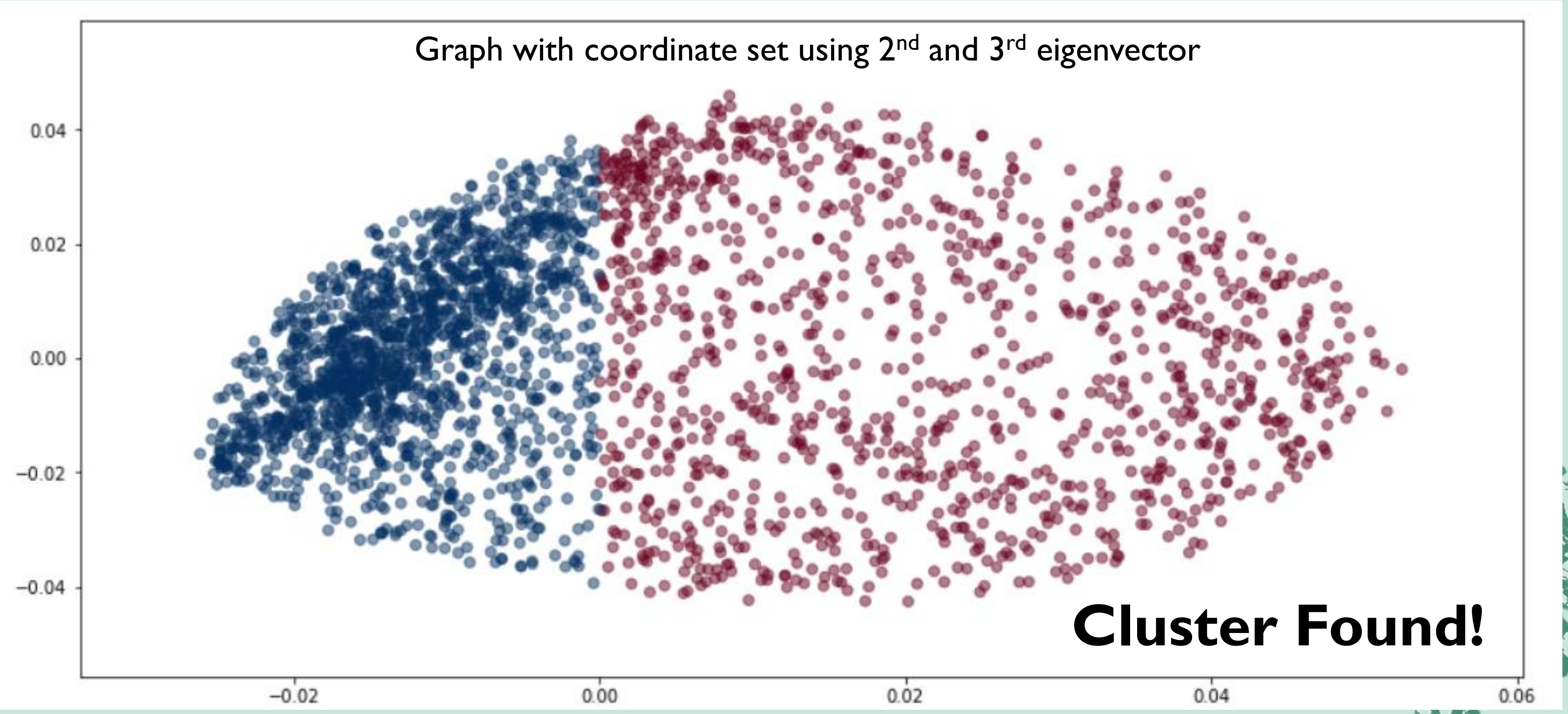
Distance

Weighting

Fiedler Vector

The final step is to compute the Fiedler vector as the second eigen-vector of the Laplacian matrix.

Fiedler Vector : Using the 0 as discriminant we can label the streets with different colors.

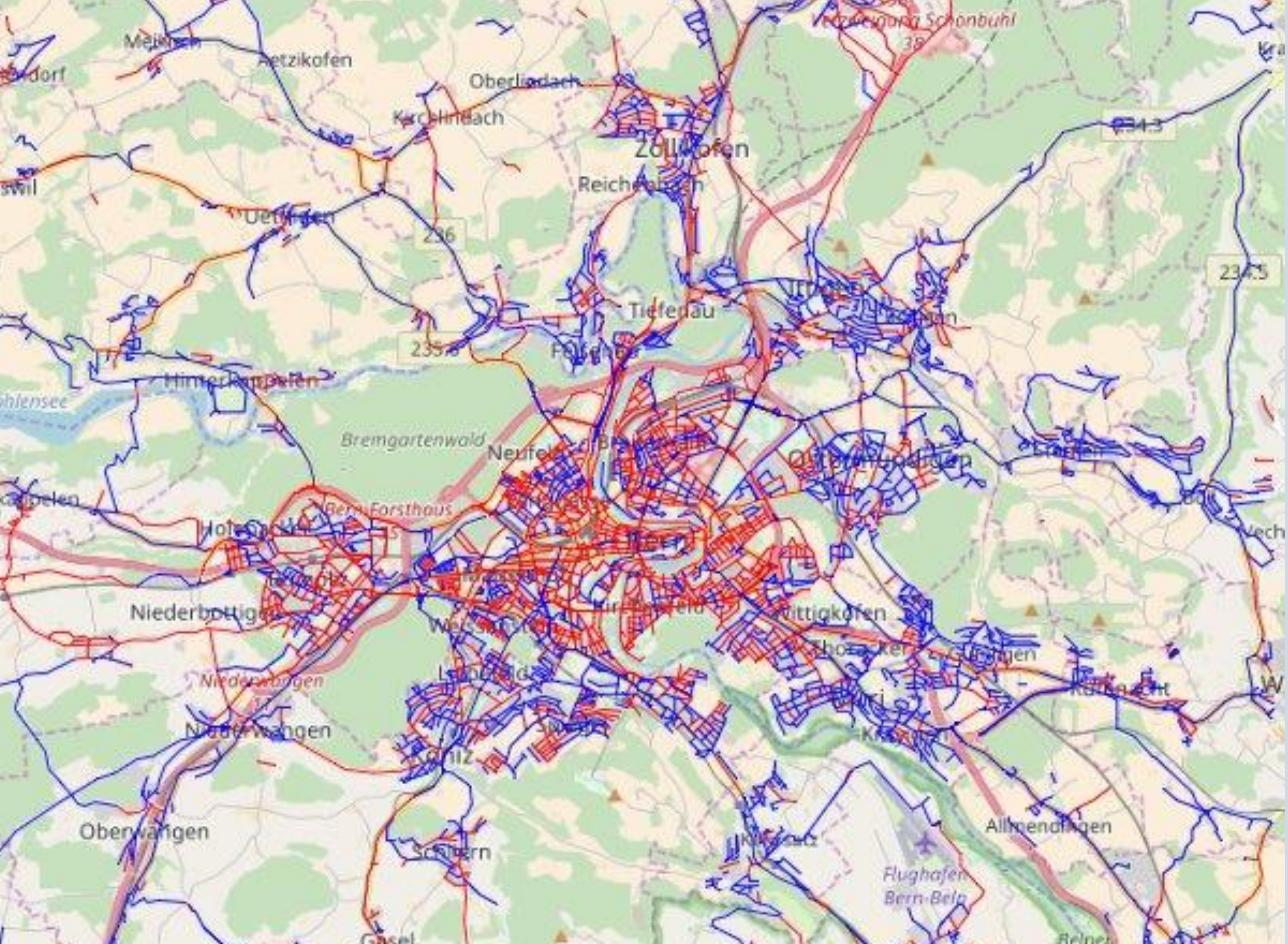


Results & Discussion

Results 1. Classifying the areas in one city

Choose some cities and try our algorithm if it can classify the area (i.e., city center – city outskirt, old part – new part) within one city.

- Bern
- Budapest

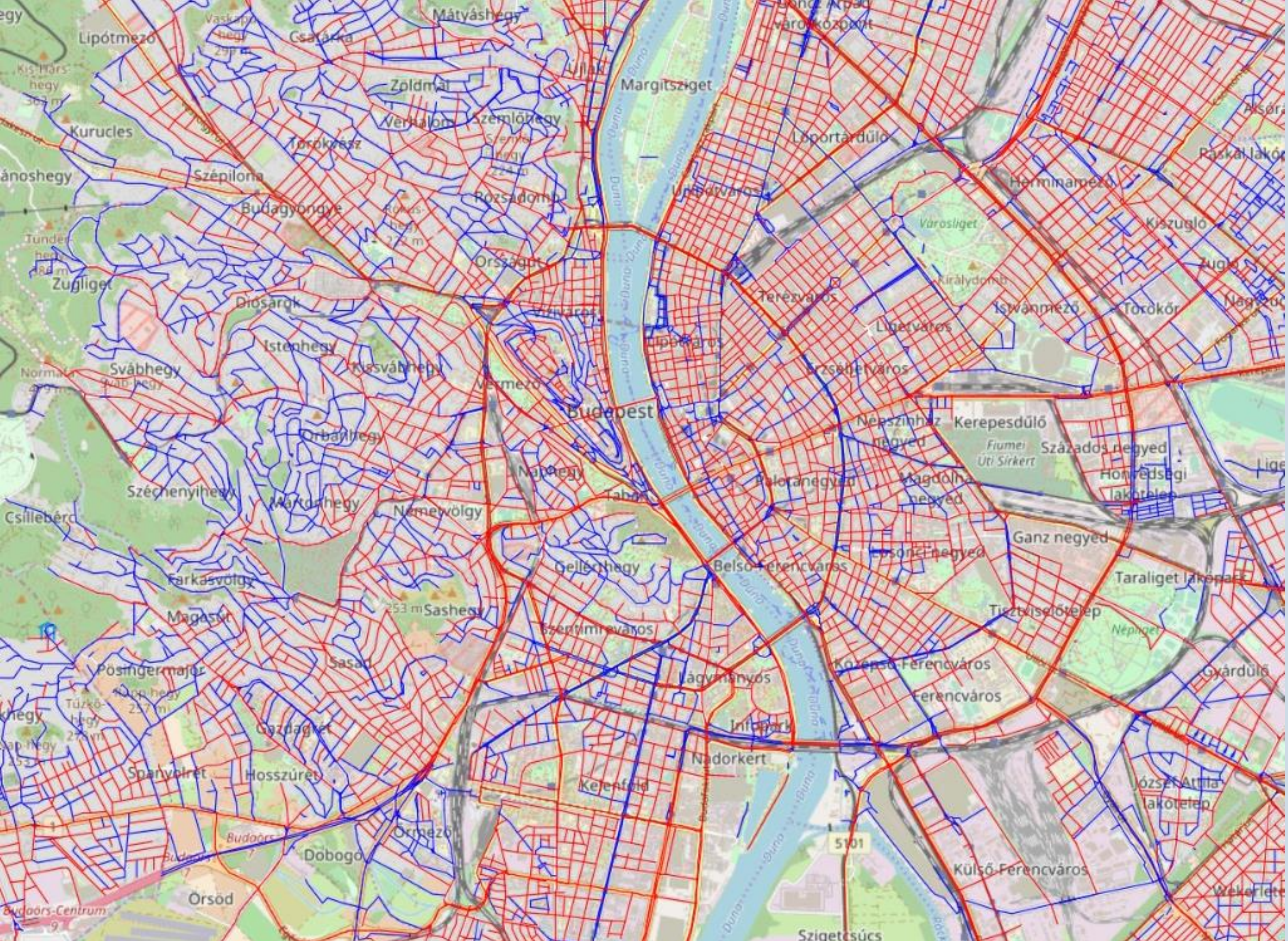


Example I

Bern

- City Center → Red
- Suburban area → Blue

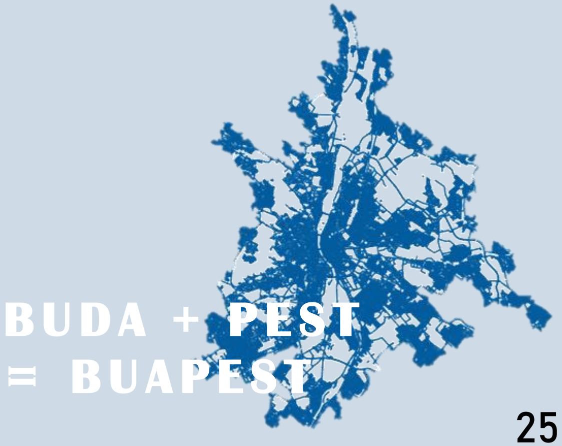
BUDA + PEST = BUAPEST



Example 2

Budapest

- Pest → **Red**
- Buda → **Blue**

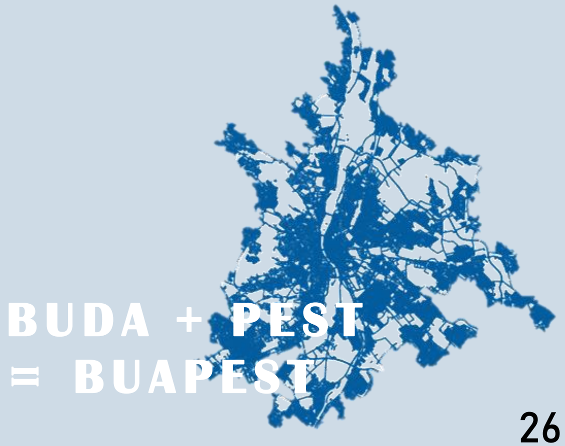


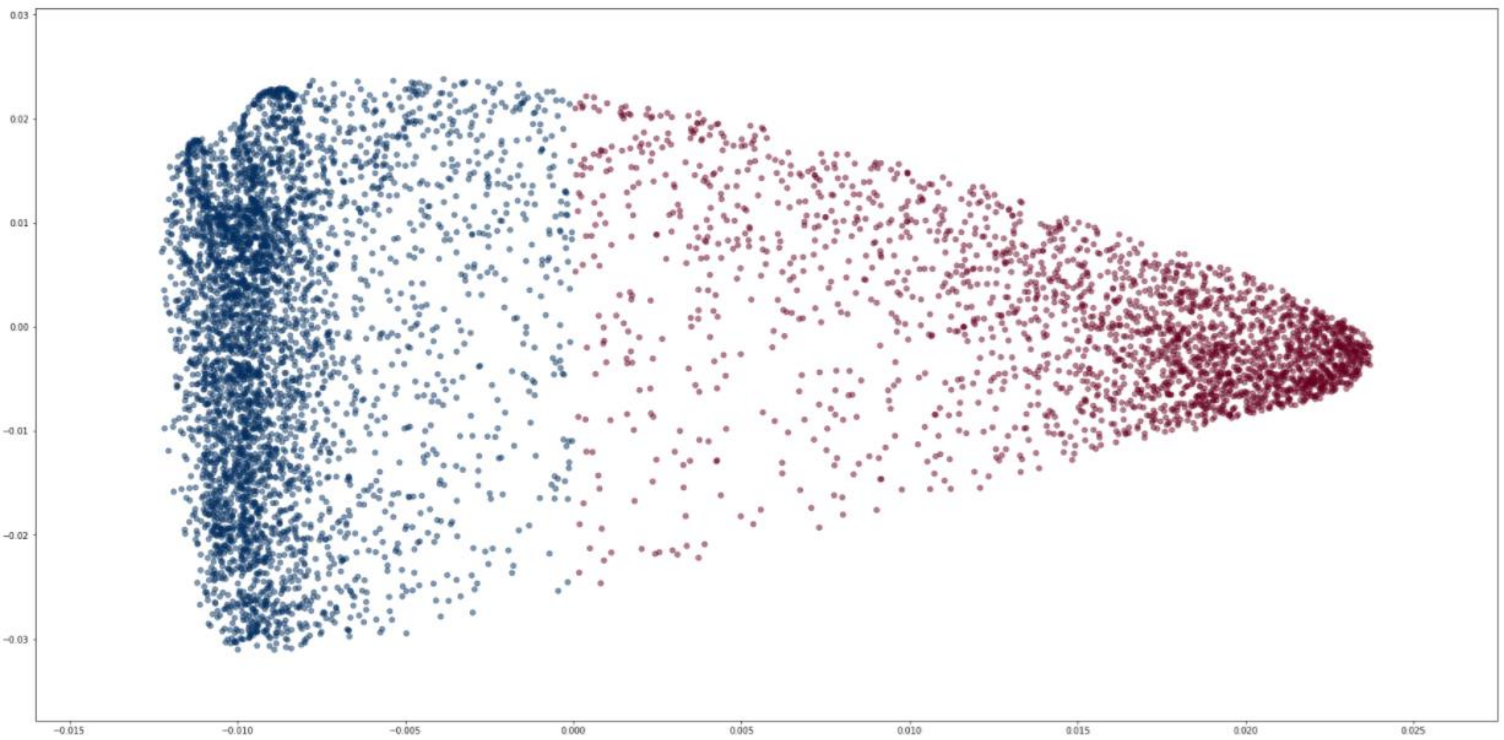
Results & Discussion

Results 2. Identifying two different cities

Choose two distinct cities and try our algorithm to see if it can identify which road belongs to which city.

- New York (United States): Metropolitan, straight roads
- Lama Mocogno (Italy): Mountain area village , curvy roads



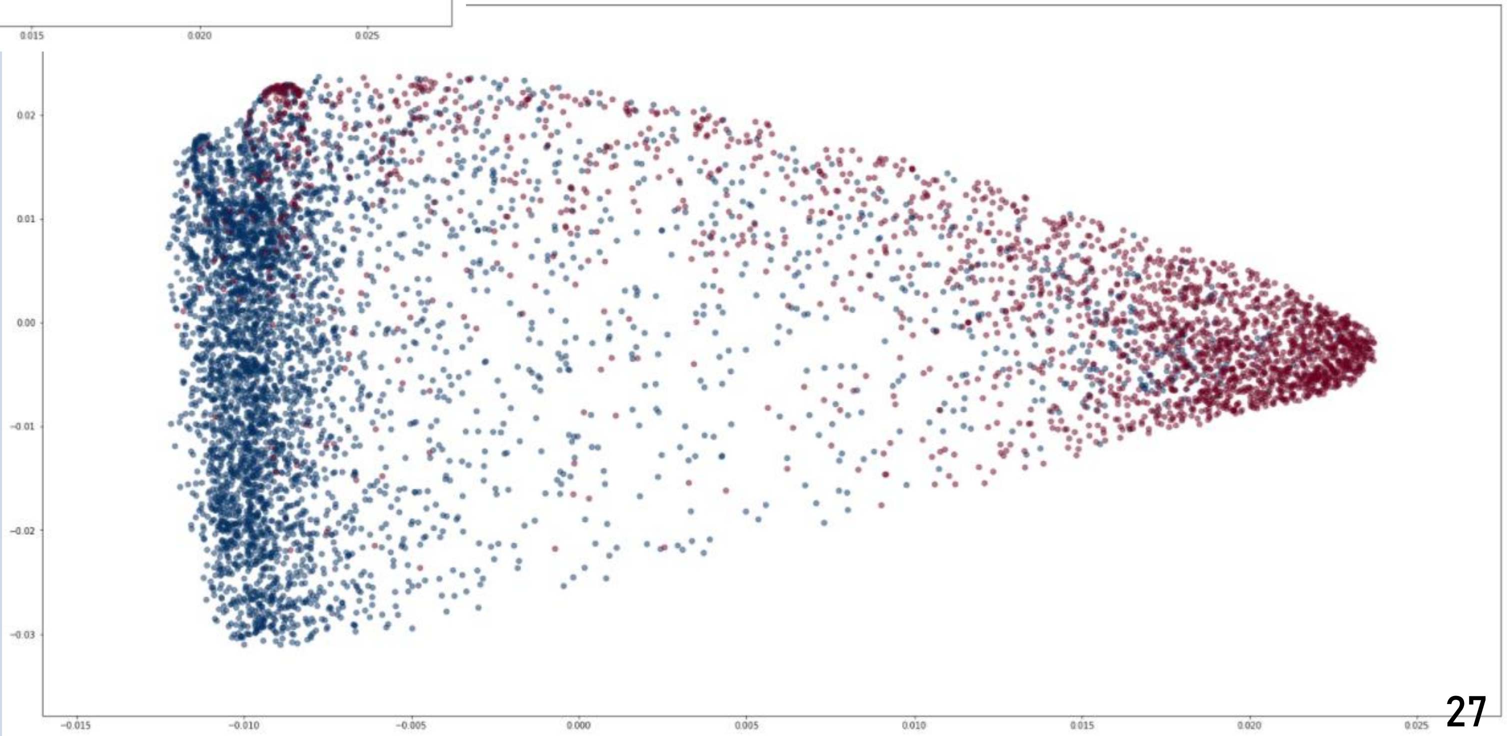


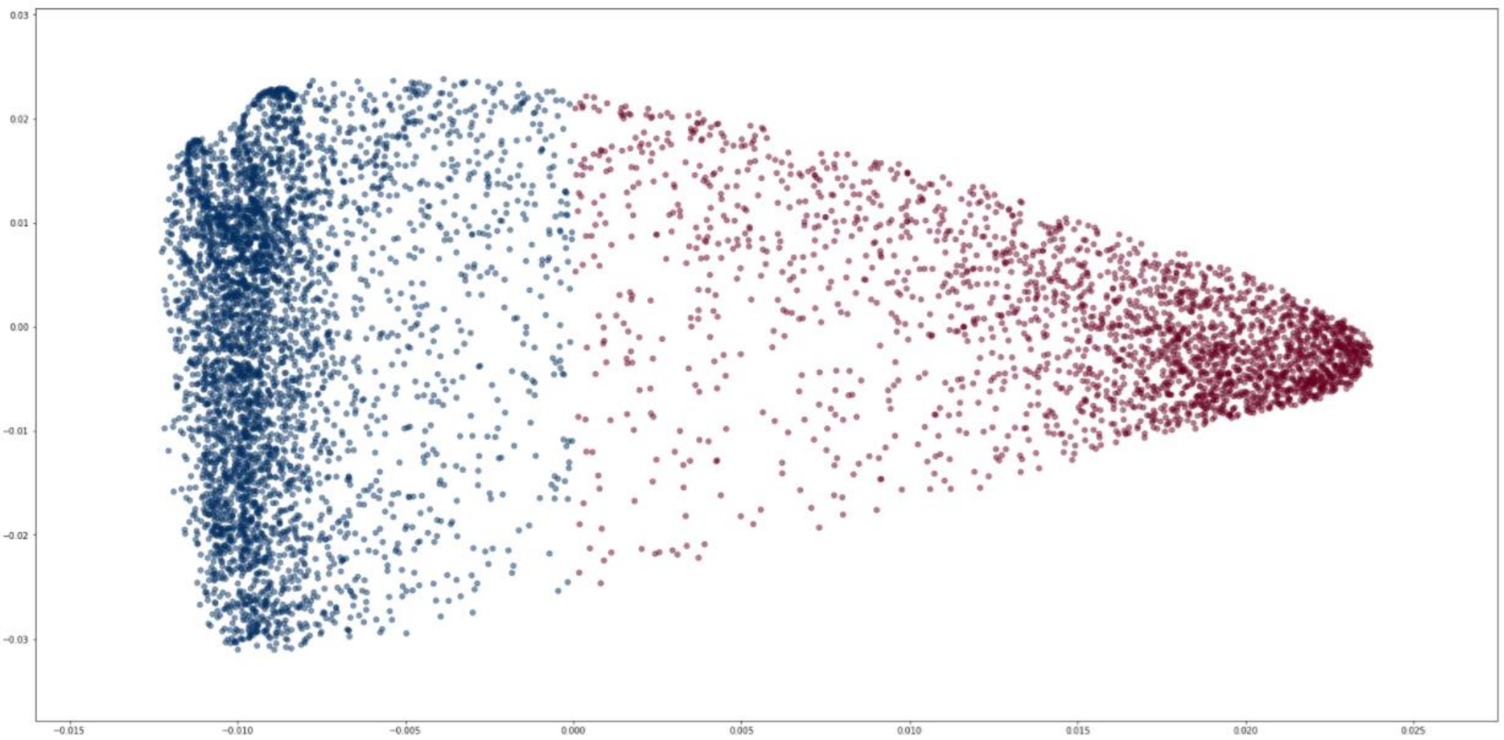
<Classified Result> (left)

- Roads classified as the one in Lama Mocogno is colored in Blue.
- Roads classified as the one in New York is colored in Red.

<Ground Truth> (right)

- Roads in Lama Mocogno is colored in Blue.
- Roads in New York is colored in Red.



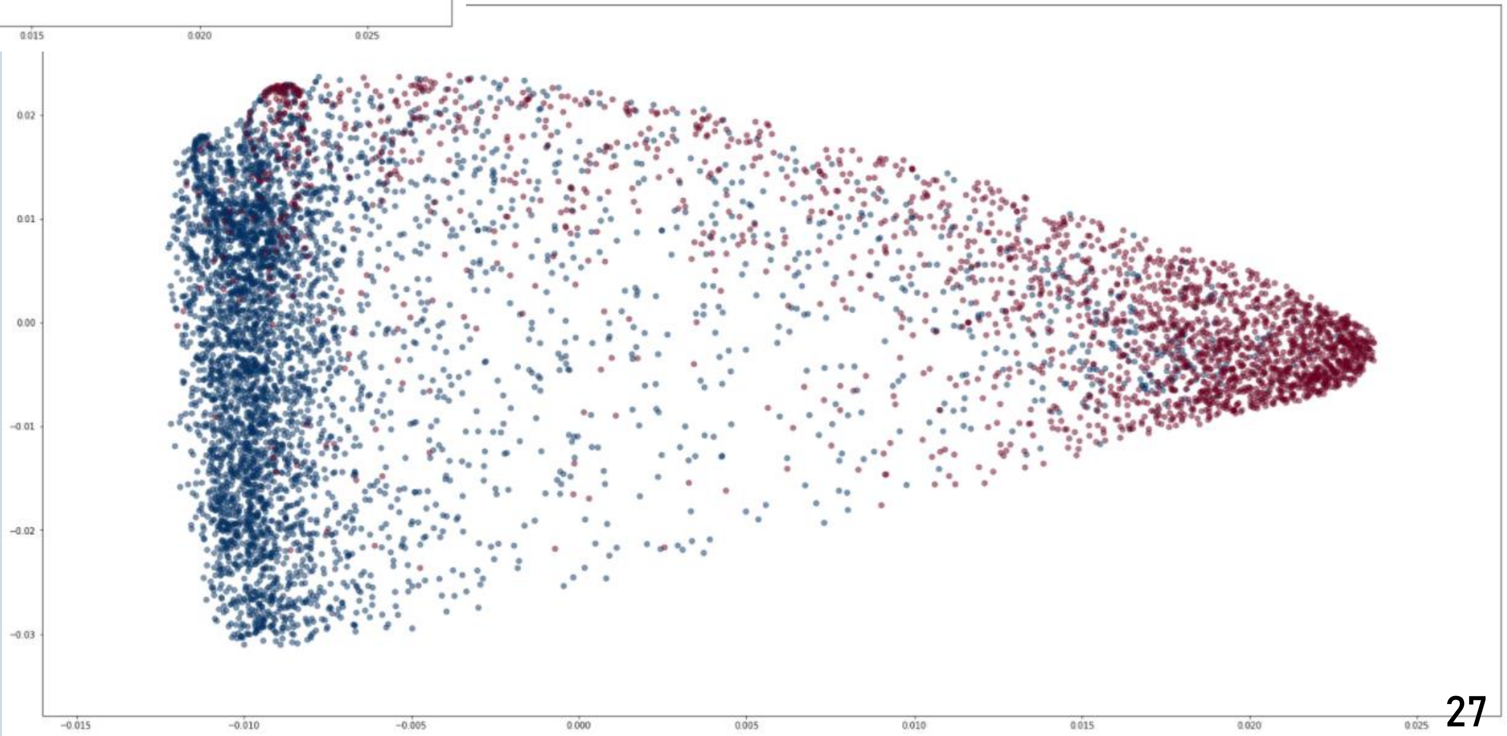


<Classified Result> (left)

- Roads classified as the one in Lama Mocogno is colored in Blue.
- Roads classified as the one in New York is colored in Red.

Classification Error

12.95%



<Ground Truth> (right)

- Roads in Lama Mocogno is colored in Blue.
- Roads in New York is colored in Red.

New York → **Red**

Blue ← Lama Mocogno

Results & Discussion

Future improvements

Add more features of the roads

- Such as bridge, road’s width, number of intersections with other roads.

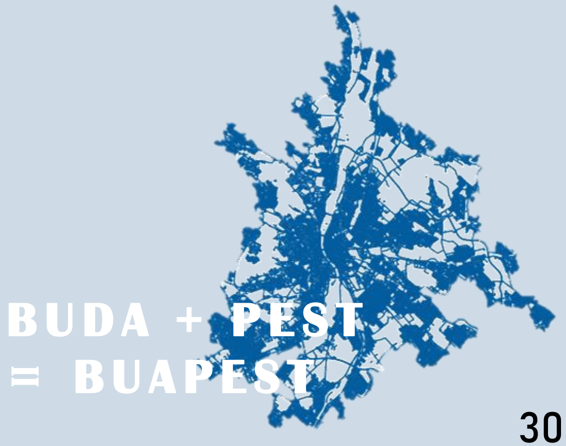
Test with more cities

- Such as cities in Asia which have old and new part.

Results & Discussion

In conclusion….

> “ **The parts** of a city can be distinguished
> based on **the road network.** ”



THANK YOU

FOR

YOUR ATTENTION!

BUDA + PEST
= BUAPEST