Human and Vehicle Trajectory Analysis

Trajectanalyse van mensen en voertuigen

Xingzhe Xie

Promotoren: prof. dr. ir. H. Aghajan, prof. dr. ir. W. Philips
Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Computerwetenschappen

Vakgroep Telecommunicatie en Informatieverwerking
Voorzitter: prof. dr. ir. H. Bruneel
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2015 - 2016

UNIVERSITEIT
GENT

ISBN 978-90-8578-904-8
NUR 958, 943
Wettelijk depot: D/2016/10.500/36

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Telecommunicatie en Informatieverwerking

Promotoren: Prof. Dr. Ir. Hamid Aghajan
Prof. Dr. Ir. Wilfried Philips

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur

Vakgroep Telecommunicatie en Informatieverwerking
Sint-Pietersnieuwstraat 41, B-9000 Gent, België

Tel.: +32-9-264.79.66
Fax.: +32-9-264.42.95

Voorzitter: Prof. Dr. Ir. Herwig Bruneel

Dit werk kwam tot stand in het kader van een specialisatiebeurs van het CSC (China Scholarship Council) en BOFcofunding-CSC (Special Research Fund - Cofunding for Chinese candidates holding a CSC-grant).

Proefschrift ingediend tot het behalen van de graad van
Doctor in de Ingenieurswetenschappen: Computerwetenschappen
Academiejaar 2015 - 2016

**Members of the jury**

Prof. Dr. Ir. Wilfried Philips (Ghent University, supervisor)
Prof. Dr. Ir. Hamid Aghajan (Ghent University, supervisor)
Prof. Dr. Ir. Peter Veelaert (Ghent University)
Prof. Dr. Ir. Sidharta Gautama (Ghent University, secretary)
Prof. Dr. Ir. Nico Van de Weghe (Ghent University)
Prof. Dr. Ir. Tinne Tuytelaars (University of Leuven)
Prof. Dr. Ir. Rik Van de Walle (Ghent University, chairman)

**Affiliations**

Research Group for Image Processing and Interpretation (IPI)
Independent Research Institute IMinds
Department of Telecommunications and Information Processing (TELIN)
Faculty of Engineering and Architecture
Ghent University

Sint-Pietersnieuwstraat 41
B-9000 Ghent
Belgium

# Acknowledgements

This dissertation would never have been accomplished without the guidance, help and support of many kind people. Especially I would like to express my gratitude to the following people:

Foremost, I would like to thank my advisors, Prof. Dr. Ir. Hamid Aghajan and Prof. Dr. Ir. Wilfried Philips, for giving the opportunity to conduct my doctoral research at Image Processing and Interpretation (IPI) group, Ghent University, and to spend 6 months doing my research at Ambient Intelligence Research (AIR) Lab, Stanford University. I would also like to thank them for numerous inspiring discussions, and specially for allowing me to work on trajectory analysis, which is more data mining than image processing. I also appreciate the help from Prof. Peter Veelaert for the very instructive discussions. Further, I am thankful to the China Scholarship Council (CSC) and BOF cofunding-CSC (Special Research Fund-Cofunding for Chinese candidates holding a CSC grant) for their fundings in this research.

My sincere gratitudes also go to all my colleagues and ex-colleagues at TELIN-IPI-IMINDS, Ghent University and AIR Lab, Stanford University for creating such a pleasant working atmosphere. Specially, I am very thankful to Filip Rooms for translating the summary into Dutch.

I am infinitely grateful to both of my Chinese and Italian families for their trust, love and support. Finally I would like to thank Riccardo Stara for his love, comfort and support.

*Ghent, February, 2016.*
*Xingzhe Xie*

# Samenvatting

De laatste jaren worden meer en meer volgsystemen gebruikt om mensen bij te staan in hun dagelijkse leven maar ook professioneel in domeinen als beveiliging en bewaking, bejaardenzorg, verkeersmonitoring, routeplanning en navigatie.

Eerst enkele definities: een verplaatsing van een object wordt voorgesteld door een pad, dat alle posities omvat die door het object ingenomen werden tijdens de verplaatsing. Het traject van het object bevat naast het afgelegde pad ook de tijdsinformatie wanneer het object zich waar op het pad bevond.

Dit genereert gigantische hoeveelheden data van trajecten van heel wat soorten bewegende objecten in allerlei soorten omgevingen, zoals bejaarden in bejaardenhuizen, passagiers in treinstations, arbeiders in een fabriek, voetgangers, fietsers en voertuigbestuurders in het verkeer, ...

Daarom is het nodig om met deze enorme hoeveelheden data om te kunnen gaan, en dit stimuleerde de ontwikkeling van trajectanalyse in data mining, machine learning en knowledge discovery.

Onderzoeksdomeinen rond trajectanalyse omvatten typisch (1) similariteitsmaten tussen verschillende trajecten, (2) trajectclustering en detectie van afwijkingen, en (3) trajectuitmiddeling. Deze drie topics worden hieronder verder beschreven.

- **Similariteitsmaten tussen verschillende trajecten:** Spatiale similariteit van twee trajecten geeft aan hoe gelijkaardig in de ruimte de paden waren van twee objecten tijdens twee verschillende verplaatsingen, en houdt geen rekening hoe snel of hoe traag de trajecten werden doorlopen. Temporele similariteit geeft aan hoe gelijkaardig de snelheden van objecten zijn langs het traject, met andere woorden hoe ze verschillen in snelheid langs het pad dat in de ruimte wel gelijkaardig is.

- **Trajectclustering en afwijkingsdetectie:** Zelfs in zeer grote datasets hebben veel trajecten de neiging gelijkaardig te zijn doordat een zelfde route werd gevolgd, en ze zelfs een gelijk snelheidsprofiel langs de gevolgde route vertonen.

  Trajectclustering streeft ernaar om gelijkaardige trajecten te groeperen in een klein aantal clusters, waarbinnen de trajecten ruimtelijk of zowel in ruimte als tijd zeer gelijkaardig zijn. Afwijkingsdetectie richt zich er dan op om trajecten te vinden die atypisch zijn en tot geen enkele cluster behoren.

- **Trajectuitmiddeling:** Trajectuitmiddeling bepaalt een representatief traject uit een verzameling trajecten, zoals een wandelroute uit een verzameling voetgangerstrajecten, of een weg uit een verzameling trajecten van voertuigen.

In deze thesis hebben we enkele technieken ontwikkeld binnen alle hierboven beschreven domeinen.

We stellen voor om spatiale en temporele similariteiten tussen paarsgewijze trajecten apart te meten gebruik makend van verschillende vormen van tijdsalignering.

Spatiale similariteit wordt berekend door middel van alignering op basis van Euclidische afstanden tussen de fysieke posities van punten langs de trajecten. Temporele similariteit wordt gemeten door middel van alignering op basis van de snelheidsverschillen voor spatiaal gelijkaardige posities.

We stellen ook een eigen nieuwe aanpak voor om een verzameling trajecten te clusteren die gebruik maakt van deze similariteitsmaten.

Dit afzonderlijk meten van de spatiale en temporele similariteiten is niet alleen nuttig om spatiale clusters te bepalen van bewegende objecten die gelijkaardige paden hebben gevolgd, maar helpt ook bij het detecteren van zowel spatiale als temporele afwijkingen. We hebben onze methode vergeleken met andere methoden om de similariteitsmaat te berekenen om trajecten van fabrieksarbeiders te clusteren, waarbij onze methode betere resultaten qua clusternauwkeurigheid oplevert dan de andere methoden.

De twee voorgestelde methoden voor gezamenlijke alignering van een grote aantal trajecten zoeken naar overeenkomsten tussen punten langs de verschillende trajecten, wat ons ook helpt om trajecten op een zinvolle manier uit te middelen. Dit uitmiddelen is niet triviaal want het gemiddelde van twee verschillende trajecten is niet eenduidig te definiëren. In onze aanpak middelen we punten uit die in de ruimte dicht bij elkaar liggen, en dit op een computationeel efficiënte manier. De eerste methode is gebaseerd op iteratieve paarsgewijze alignering gebruik makend van Dynamic Time warping (DTW). De andere methode is een zogenaamde *greedy* methode die erin slaagt om een deel van de complexiteit van DTW te vermijden door optimaal gebruik te maken van de statistieken van de data.

De resulterende uitgemiddelde trajecten zijn van hoge kwaliteit wanneer we ze vergelijken met de resultaten van andere bestaande methoden. Ze laten toe om eenvoudig maar toch nauwkeurig snelheidsvariaties langs een gemiddeld traject te kwantificeren.

De eerste methode zoekt naar *één-op-één* overeenkomsten tussen punten van de trajecten. Een "stretch and then compress" strategie wordt toegepast om de trajecten één na één te aligneren in volgorde van toenemende gemiddelde afwijking.

Tijdens paarsgewijze alignering van twee trajecten zal DTW lokaal de trajecten uitrekken (*stretch*) waar nodig om een optimale overeenkomst te bekomen. Dit uitrekken is niet altijd wenselijk, maar geeft zijn naam aan de eerste stap van de methode (stretch stap). De *compress* operatie verwijdert

de herhaalde punten uit de gerekte trajecten, waarbij we twee gecomprimeerde trajecten met *één-op-één* overeenkomsten tussen de punten bekomen.

Het laatste gecomprimeerde traject bepaalt de lengte van alle gewarpte trajecten. De *één-op-één* overeenkomsten worden dan gebruikt om voor alle andere trajecten de punten te bepalen die overeenkomen met de punten van de laatste comprimeerde trajecten.

De tweede methode zoekt naar *veel-op-één* overeenkomsten tussen punten van de verschillende trajecten en rekt de trajecten dus niet uit. In plaats van per paar trajecten te werken, worden alle trajecten samen gealigneerd langs de tijdsas. Om rekentijd te sparen stellen we een "greedy" procedure voor die direct alle punten van het gemiddelde traject één na één iteratief berekent. Daarvoor moeten we de punten lokaal clusteren, de best passende cluster selecteren en dan de punten in die cluster uitmiddelen om ruis uit te middelen. Terwijl onze eerste methode gebruik maakt van dynamic programming om op een optimale manier met ruis om te gaan, past de tweede methode uitmiddelen van veel trajectpunten toe om hetzelfde doel te bereiken.

Experimentele resultaten geven aan dat we er met onze beide nieuwe methoden in slagen om overeenkomsten tussen de punten van alle trajecten te vinden, waarbij gewarpte trajecten worden bekomen met de overeenkomstige punten voor dezelfde tijdindices langs de trajecten.

In deze thesis gebruiken we onze voorgestelde methoden voornamelijk voor twee toepassingen: optimalisatie van de werkcyclus uit de trajecten van fabrieksarbeiders en bepalen van een wegennetwerk uit GPS trajecten.

In onze toepassing voor werkcyclus optimalisatie, clusteren we de trajecten in verschillende soorten uitgevoerde werkcycli en bepalen daarbij de abnormale trajecten die niet horen bij de typische trajecten van arbeiders. Door de trajecten gezamenlijk te aligneren, kunnen we ze uitmiddelen tot een typerende route die hoort bij elke uitgevoerde werkcyclus, en houden daarbij typerende snelheid en doorlooptijd bij langs elke route. Deze resultaten kunnen dan worden gebruikt om de fabrieksarbeiders te helpen hun werkcycli efficiënter uit te voeren.

Bij onze toepassing om een wegennetwerk te bepalen, gebruiken we de drie voornaamste onderdelen van wegen: kruispunten, hoe de kruispunten verbonden zijn en geometrische voorstelling van elk wegsegment.

We stellen twee methoden voor om kruispunten te detecteren: de eerste is gebaseerd op het detecteren de lokaties waar weggebruikers hun beweegrichting verandert; de tweede is gebaseerd op het detecteren waar drie of meer wegsegmenten samenkomen.

We onderzochten hoe de kruispunten met elkaar waren verbonden door na te gaan of weggebruikers ooit reizen van een kruispunt naar een tweede zonder een van de andere kruispunten te passeren. GPS tracks worden dan opgesplitst in stukken wegsegment door deze direct verbonden kruispunten.

Voor elk wegsegment worden spatiaal abnormale tracks gedetecteerd door de tracks te clusteren volgens de spatiale dissimilariteitsmaat. Normale tracks worden gealigneerd gebruik makend van de voorgestelde gezamenlijke aligner-

ingsmethode, waarbij we gewarpte tracks bekomen. Deze gewarpte tracks worden dan uitgemiddeld om zo tot de geometrische voorstelling van ons wegsegment te komen.

We hebben onze methoden getest op twee datasets: de Chicago dataset met 889 GPS trajecten, en de Berlijn dataset met 26831 GPS trajecten. Experimentele resultaten tonen een grote nauwkeurigheid aan bij detectie van kruispunten, en een betere geografische nauwkeurigheid van de gevonden wegsegmenten in vergelijking met andere algoritmes.

Tenslotte beschrijven we in deze thesis nog een neventoepassing van trajectanalyse: het bepalen hoe een (vergader)ruimte is ingedeeld op basis van trajecten van mensen. In plaats van objecten te detecteren op basis van hun kenmerken in een beeld (zoals vorm, kleur en textuur), stellen we een indirecte manier voor om de aanwezigheid van stoelen, tafels en vrije wandelruimte af te leiden uit de trajecten van mensen in die ruimte. Bij deze toepassing gebruiken we geen van de hierboven voorgestelde methoden, maar maken gebruik van waarschijnlijkheidsrekening.

Eerst halen we de hoogte- en snelheidsinformatie uit de trajecten, en geven die als invoer van een SVM classifier, om zo verschillende klasses van de ogenblikkelijke activiteiten van mensen te bepalen. Deze klasses worden op een hoger niveau samengevoegd voor elk tijdstip. We berekenen dan twee bezettingsmappen, de ene voor waar personen zitten op elk tijdstip en de andere waar personen wandelen op elk tijdstip. De zitmap wordt bijgewerkt met de zitactiviteiten van de personen op elk moment, en de wandelmap wordt bijgewerkt met de wandelactiviteiten van de personen. Tenslotte wordt de aanwezigheid van de stoelen afgeleid uit de zitmap en die van de tafel uit de wandelmap. Experimentele resultaten tonen aan dat tafels en stoelen met succes worden gedetecteerd.

We vatten hier de belangrijkste bijdragen van deze thesis samen:

- een nieuwe methode voor trajectclustering die gebruik maken van zowel spatiale en temporele similariteitsmaten om paarsgewijs trajecten te vergelijken.

- twee nieuwe methoden voor gezamenlijke alignering van een groot aantal trajecten. De ene is gebaseerd op *één-op-één* overeenkomsten tussen punten, en de andere legt *veel-op-één* overeenkomsten tussen punten.

- twee nieuwe methoden om kruispunten te detecteren. De eerste is gebaseerd op het vinden van plaatsen waar de beweegrichting van weggebruikers verandert, en de tweede op het vinden van plaatsen waar minstens drie wegsegmenten samenkomen.

- een eerste toepassing van trajectanalyse: het optimaliseren van werkcycli, waarbij representatieve paden, snelheden en looptijden langs elk pad voor fabrieksarbeiders worden bepaald, om zo hun werkefficiëntie te verhogen.

- een tweede toepassing van trajectanalyse: bepaling van een wegennetwerk, gericht op het vinden van kruispunten, hoe deze kruispunten verbonden zijn en een geometrische voorstelling van elk wegsegment.
- de derde toepassing van trajectanalyse: bepalen hoe een ruimte is ingedeeld, gericht op het herkennen van stoelen, tafels en vrije wandelruimte in een slimme vergaderzaal.

In totaal resulteerde het onderzoek tijdens dit doctoraat in vijf publicaties in internationale peer-gereviewde journals: twee zijn reeds gepubliceerd [Xie 15b, Bo 14], en drie zijn nog in review [Xie 16d, Xie 16a, Xie 16b]. Verder hebben we ook elf artikels gepubliceerd in proceedings van internationale conferenties [Xie 12, Grünwedel 12, Xie 13b, Xie 14c, Xie 14a, Xie 14b, Bo 14, Eldib 14, Xie 15a, Eldib 15, Xie 16c].

# Summary

In recent years, tracking systems have become widely used to assist people's daily life and work in areas such as security and surveillance, elderly care, traffic monitoring and path planning and navigation. They generates massive trajectory data from a variety of moving objects in all kinds of environments, such as tracks of elderly people in care homes, passengers in train station, workers in the factory, pedestrians, cyclists, vehicles on the roads.... Automated processing of this data by trajectory analysis forms the basis of many other processing techniques, such as data mining, machine learning and knowledge discovery.

Research topics related to trajectory analysis typically include the definition of application-relevant trajectory similarity measures, techniques for trajectory clustering and/or for abnormality detection, techniques for computing representative trajectories and techniques for statistical analysis not only of the trajectories as a whole but also of local parameters of the trajectory. This PhD focuses on the following topics:

- **Trajectory similarity measures:** Spatial similarity between two trajectories is a measure of how similar the routes of the moving objects are in the two journeys irrespective of haw fast or slow the trajectories are traversed. Temporal similarity measures how similar the moving objects' speed profiles are in two trajectories, i.e., how differently they traverse trajectories which are otherwise spatially similar.

- **Trajectory clustering and abnormality detection:** Even in very large datasets, many trajectories tend to be similar, because they follow more or less the same route, perhaps even the same speed profile.

  Trajectory clustering aims to group similar trajectories into a small number of clusters, in which the trajectories are spatially or spatiotemporally very similar. Abnormality detection aims to find those trajectories which are not typical at all and do not belong to a specific cluster.

- **Trajectory averaging:** Trajectory averaging aims to extract a representative trajectory from a number of trajectories, such as a walking route from pedestrians trajectories or the road network from vehicle trajectories.

In this thesis, we deploy techniques to address all the above mentioned topics. We propose to measure spatial and temporal similarity between pairwise trajectories separately, using different forms of time alignment. Spatial

similarity is calculated as the residual error after aligning physical points on trajectories to minimize their Euclidean distance. Temporal similarity is calculated similarly, but this time after aligning the trajectories to minimize the velocity difference between corresponding spatial locations.

In this thesis, we also present a novel approach to jointly cluster many trajectories based on the similarity measures. Measuring the spatial and temporal similarities separately is not only helpful to find spatial clusters in which the moving objects follow similar routes, but beneficial to detect both spatial and temporal abnormalities. We compare our method with other methods and similarity measures in the problem of clustering factory worker trajectories. The results show that our method outperforms other methods and achieves higher clustering accuracy.

The two proposed methods for joint alignment of *many* trajectories establish point correspondences between trajectories, which also helps to average trajectories in meaningful ways. This problem is not trivial as the average of many distinct trajectories is not well and uniquely defined. In our approach we average points which are spatially close, and we do so in a computationally efficient way. One method is based on iterative pairwise alignment using dynamic time warping. The other one is a greedy method which manages to avoid some of the complexity of dynamic time warping by making optimal use of the data statistics.

The resulting average trajectories are high quality compared to those produced by existing methods. They allow to easily and accurately quantify velocity variance along the average trajectory.

The first method, i.e., the one based on DTW, first establishes *one-to-one* correspondences among the points of the trajectories. A "stretch and then compress" strategy is then applied to align the trajectories one by one in ascending order of average dissimilarity. During the pairwise alignment, the DTW locally *stretches* the trajectories where needed to achieve an optimal match. This stretching is not always desired, but lends its name to the first step of the method, the *stretch* step. A *compress* operation removes the repeated points from the stretched trajectories, producing two compressed trajectories with *one-to-one* correspondences between the points. The last compressed trajectory decides the length of all warped trajectories. The *one-to-one* correspondences are used to locate the points on every other trajectory, which are associated to the points of the last compressed trajectories.

The second method establishes *many-to-one* correspondences among the points of the trajectories, and so does not stretch trajectories. Instead of in a pairwise fashion, all of the trajectories are aligned simultaneously along the time dimension. In order to reduce the computational cost, we propose a "greedy" procedure which directly computes each point of the "average" trajectory iteratively, one at a a time. This involves local clustering of points and selection of the best matching cluster, followed by averaging of the points in a cluster to smooth out any noise. While the first method counts on dynamic programming to optimally handle the effects of noise, the second method relies

on the averaging of many trajectory points to achieve the same goal.

The experimental results in the thesis indicate that we successfully establish point correspondences among all of the trajectories using both proposed methods, producing warped trajectories with associated points at the same time index.

In this thesis, we deploy these proposed methods mainly in two applications: work cycle optimization using factory worker trajectories and road network inference using GPS trajectories. In our application of work cycle optimization, we cluster the trajectories into different types of executed work cycles and detect the abnormal trajectories which are unfit to any of the prototypical itineraries. Through joint trajectory alignment, we average the trajectories as a prototypical route for each type of executed work cycle, and build the prototypical velocity and dwell time along each route. These will be used to guide the factory workers to execute their work cycles more efficiently.

In the road network inference application, we detect the three main components of the road network: intersections, their connectivity and the geometric representation of the connecting road segments. We propose two methods to detect the intersections. One depends on detecting the turning points where the road users change their moving directions, the other one on detecting the connecting points where three road segments meet. The intersection connectivity is explored by examining whether the road users ever travel through every two intersections consecutively without passing by any other intersection. GPS traces are segmented to track pieces for individual road segments by the directly-connected intersections. For each road segment, spatially abnormal tracks are detected by clustering the tracks using the spatial dissimilarity measure. Normal tracks are aligned using the proposed joint alignment methods, producing warped tracks. The warped tracks are averaged as the geometric representation of the road segment.

We test our methods on two datasets: Chicago dataset with 889 GPS traces, and Berlin dataset with $26,831$ GPS traces. Experimental results show a high accuracy of intersection detection, and a better geographical accuracy of the extracted road segments, compared to other algorithms.

This thesis also presents a side application of trajectory analysis: room layout exploration from people trajectories. Instead of detecting the objects directly using their image features, such as color, shape and texture, we propose to recognize the presence of chairs, tables and walking areas in a smart meeting room indirectly from people's trajectories. In this application, we do not deploy any of the methods proposed above, but apply probability analysis. We first extract speed and height information from the trajectories, and input them to a SVM classifier, so as to categorize people's instantaneous activities. We then merge the instantaneous activities into higher-level activity at each time period. We build two occupancy maps, one is for sitting space, and the other one for walking space. The sitting map is updated using people's sitting activities at each time period, and the walking map is updated using people's walking activities. At last, chairs are inferred from the sitting map, and the table from

the walking map. Experimental results show that the table and chairs are successfully detected.

To summarize, the main contributions of this thesis are:

- a novel trajectory clustering method using both spatial and temporal similarity measures between pairwise trajectories.
- two novel methods for joint alignment of *many* trajectories. One builds *one-to-one* point correspondences, and the other one establishes *many-to-one* point correspondences.
- two novel methods for intersection detection. One is based on finding locations where road users change their moving directions, the other one on finding locations which connect three road segments.
- one application of trajectory analysis: work cycle optimization, focusing on extracting prototypical routes, and prototypical velocity and dwell time along each route for factory workers, to improve their work efficiency.
- another application of trajectory analysis: road network inference, focusing on extracting intersections, intersection connectivity and geometric representation of each road segment.
- the third application of trajectory analysis: room layout exploration, focusing on recognizing the presence of chairs, tables and walking areas in a smart meeting room.

In total, the research during this PhD resulted in five submitted publications in international peer-reviewed journals, two of which have already been published [Xie 15b, Bo 14], and three articles are under review [Xie 16d, Xie 16a, Xie 16b]. Furthermore eleven papers have been published in the proceedings of international conferences [Xie 12, Grünwedel 12, Xie 13b, Xie 14c, Xie 14a, Xie 14b, Bo 14, Eldib 14, Xie 15a, Eldib 15, Xie 16c].

# Contents

# List of Abbreviations

| | |
|---|---|
| 2D | 2-Dimensional |
| ALBP | Assembly Line Balancing Problem |
| CRF | Conditional Random Field |
| DTW | Dynamic Time Warping |
| DBA | DTW Barycenter Averaging |
| GIS | Geographical Information Science |
| GPS | Global Positioning System |
| HMM | Hidden Markov Model |
| ICA | Independent Component Analysis |
| KDE | Kernel Density Estimation |
| LBS | Location-Based Service |
| LCSS | Longest Common Sub-Sequence |
| LIP | Locality In-between Polylines |
| MLN | Markov Logic Network |
| PCA | Principle Component Analysis |
| PSA | Prioritized Shape Averaging |
| RFID | Radio-Frequency Identification |
| RSS | Received Signal Strength |
| TDH | Trajectory Directional Histograms |
| TRPS | Trajectory Re-sampling Point Set |
| UWB | Ultra-Wide Band |

# 1

# Introduction

*The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny...'*
—Isaac Asimov

In recent years, decreasing cost and increasing performance of the tracking systems, for instance, cameras, Radio Frequency Identification (RFID), Ultra-Wide Band (UWB) and Global Positioning System (GPS), have lead to their widespread use in smart environments [Patel 09, Naftel 06, Chang 10], such as airports, railway stations, shopping malls, parking lots, banks, factories, etc. With the help of tracking systems, the trajectories of the moving objects can be easily recorded and then analyzed to find their motion patterns and detect suspicious events [Basharat 08,Morris 08,Morris 11], for instance, exiting the store without pausing at cash desks, running at the bank, driving in the wrong direction, etc. Besides, the tracking systems have also been applied in sport analysis, which provides technical support to the coaches by analyzing the trajectories of the players on the filed [Chavoshi 15, Faude 12]. Moreover, GPS has been popularly used in migration observation [Chapman 10,Shamoun-Baranes 10]. The change of the migration routes and the intermediate stations can be found from the trajectories of the animals and birds.

As the hand-held GPS units have been becoming popular in the last decade, geographical data is much more easily obtainable from not only cars, taxis and trucks, but also from cyclists and pedestrians. This abundance of GPS-derived geospatial data has stimulated intensive research activities in Geographical Information Science (GIS). Road networks are generated and updated from the trajectories of the road users, which are essential elements of the route planning [Syberfeldt 09, Schultes 08, Niehoefer 09, Morris 04]. The spatio-temporal patterns of urban traffic congestions have been unveiled to assist urban planning, traffic control, and Location-Based Services (LBS) [Herrera 10, Wang 13b, Castro 12]. Researchers in both industrial and academic fields have been analyzing the city mobility, so as to provide helpful information for public transportation systems, such as the bicycle-sharing system and the electric vehicle charing system [Semanjski 15,Lopez Aguirre 15,Zheng 08b].

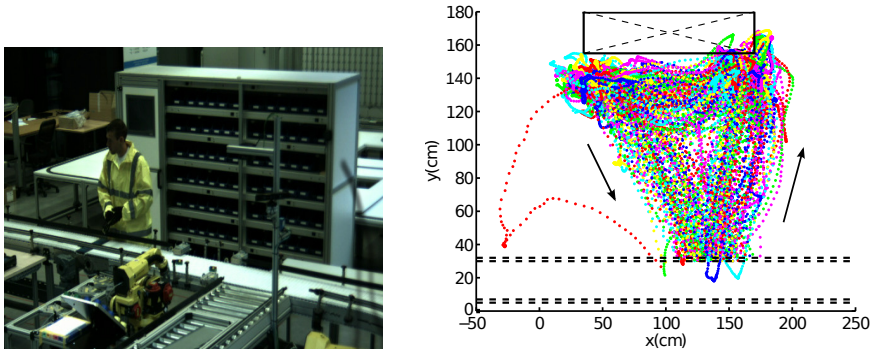
A trajectory is a time-ordered sequence of data points representing how the

physical locations of the moving object change over time. Current research on trajectory analysis focuses on trajectory similarity measures, trajectory clustering and abnormality detection, trajectory averaging, and so on:

- **Trajectory similarity measure:** Spatial similarity between two trajectories indicates how similar the routes of the moving objects are in two journeys. Researchers have proposed measures for spatial similarity using a variety of methods, such Dynamic Time Warping (DTW) [Niennattrakul 09], Longest Common Sub-Sequence (LCSS) [Smith 81], Principle Components Analysis (PCA) [Bashir 07], Distance measures [Atev 06, Zhou 07], etc. Temporal similarity measures have not been analyzed much; they describe how similar the moving objects' speeds and time duration are in two trajectories when they follow the same route [UETA 00, Pelekis 07, Zheng 10], or they find whether two trajectories intersect or not by checking their time instances [Hodgson 06].

- **Trajectory clustering and abnormality detection:** Trajectory clustering aims to find similar motion patterns of moving objects. In abnormality detection, one single typical motion pattern is expected and trajectories which do not conform to this pattern are considered as abnormal. Depending on how trajectories are processed, clustering methods can be broadly classified into feature-based methods and direct methods [Atev 10]. Feature-based methods first compute a feature vector for each of the trajectories. They then cluster or classify these vectors using techniques from pattern recognition. Their main downside is their sensitivity to feature selection [Li 06, Anjum 10, Zheng 10]. Direct methods operate on the data points of the trajectories directly. These methods compute a spatial similarity measure between pairwise trajectories by associating their data points optimally [Niennattrakul 09, Vlachos 02, Zhang 06]. These methods outperform the feature-based methods due to the accurate spatial similarity measure but their computational cost is expensive. Moreover, given the lack of temporal similarity measures in existing research, temporal abnormalities can not be detected easily with these methods.

- **Trajectory averaging:** "Trajectory averaging" is somewhat of a misnomer as it refers to the computation of a representative trajectory which is somehow "typical" for a set of trajectories, where "typical" means that most trajectories are either spatially or spatiotemporally similar to the selected representative trajectory. Through trajectory averaging, e.g., walking routes can be found from pedestrian trajectories, or the geometric representation of a road segment can be extracted from vehicle trajectories. Researchers have applied a variety of techniques to average trajectories, ranging from image processing to trajectory alignment [Junejo 08, Petitjean 11]. However none of them has established point associations among the trajectories, which is not only helpful to trajectory averaging, but also beneficial to statistical analysis. The associated points at the same




**(a)** Work station **(b)** Trajectories

**Figure 1.1:** *Work station and factory worker trajectories.* In (a), a factory worker stands at the work station assembling parts of a product. There is a storage rack behind him, where the tools and parts are kept. (b) shows 124 trajectories of factory workers in different colors, and the arrows indicate the directions of the trajectories. The location of the storage rack is shown as a black rectangle near the top of the figure; the dashed lines at the bottom show the location of the conveyor belt.

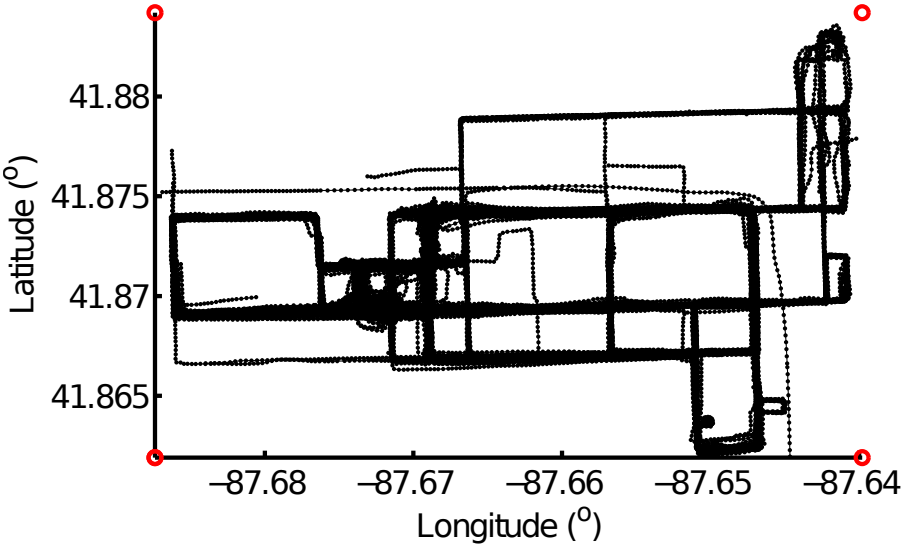
indexes can be directly averaged to form an average trajectory. The location and velocity variation along the average trajectory can be easily calculated using the associated points.

This thesis is related to all of the above topics, and addresses the unsolved problems: trajectory clustering using both spatial and temporal similarity measures and joint aligning more than two trajectories. Specifically, we will apply our methods in two applications: work cycle optimization using factory worker trajectories and road network inference from vehicle trajectories.

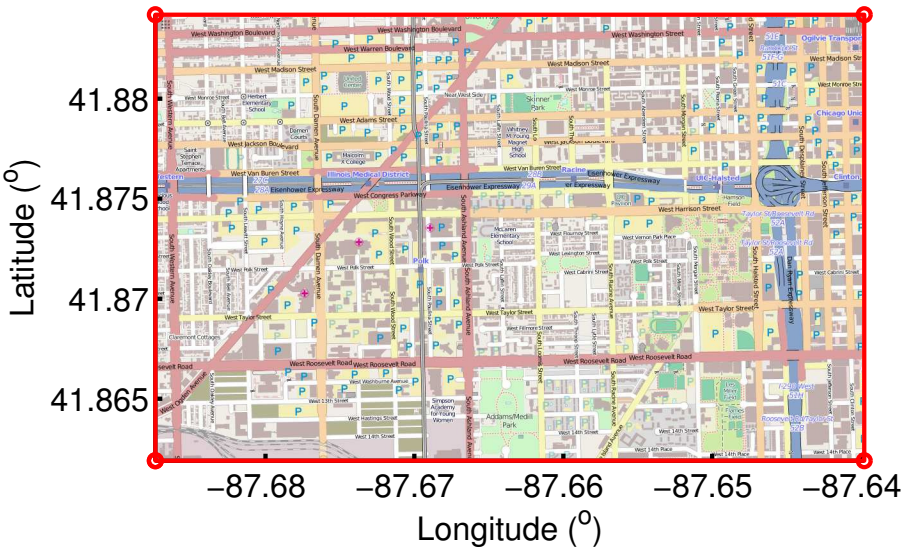
## 1.1 Problem statement

In the application of work cycle optimization, as shown in Fig. 1.1, the factory workers follow an assembly work cycle which involves going to the storage rack, picking up tools and objects in front of the storage rack, and coming back to the assembly platform along the conveyor belt. Although this work cycle is scheduled for the workers, how they execute it is different, which may lead to work inefficiencies. For instance, walking too much back and forth between the assembly platform and part storage racks, dwelling on the way to the storage rack or back to the assembly line, and strolling in front of the storage rack too slowly. The overall aim of this research is to analyze the factors leading to work efficiency and improve the work efficiency by building prototypical routes, and statistically analyzing prototypical velocity and dwell time along each route.

In the other application of road network inference, as shown in Fig. 1.2, we aim to extracting the road network from Global Positioning System (GPS)




**(a)** Trajectories




**(b)** Road map

**Figure 1.2:** *Road map and vehicle trajectories.* (b) shows 889 GPS trajectories of campus shuttles. (a) shows the road map of this area covered by the GPS trajectories. We aim to extract a road map from the trajectories.

trajectories. This road network extracted will allow mapping unexplored geographic regions (e.g., in developing countries), updating existing maps and planning traveling routes. As hand-held GPS devices have been used ubiqui-

tously in the last decade, geographical data can be obtained from a variety of road users, such as cars, taxis, cyclists, pedestrians, etc. The geographical locations recorded could be inaccurate because of GPS error, which is caused by inaccurate time-keeping by the receiver's clock, reflections from buildings and other large, solid objects, atmospheric disturbances, etc. GPS trajectories with high-error samples will result in spurious edges and inaccurate geometric representation for the road segment.

No matter whether extracting prototypical routes from the factory worker trajectories, or extracting road segments from vehicle trajectories, clustering techniques are needed to categorize the trajectories and remove outlier trajectories; trajectory averaging of the normal trajectories is needed as well in all cases, to extract the prototypical routes and their geometric representation. In this thesis, we measure both spatial and temporal dissimilarities between pairwise trajectories, and use them to cluster the trajectories and detect both spatial and temporal abnormalities.

As an important step in similarity measure computation and clustering, a large part of this dissertation focuses on establishing point correspondences. We propose two approaches to align many trajectories. The first method establishes *one-to-one* correspondences among the points of the trajectories, and the second method builds *many-to-one* point correspondences. The point correspondences are used to compute "average" trajectories and to statistically analyze the trajectories, e.g., regarding average speed at specific spatial positions.

In the road network inference application, we deal with GPS track pieces for each road segment, instead of the whole GPS traces which covers several road segments. Therefore, we first need to detect the intersections from GPS traces, then segment the GPS traces into track pieces corresponding to individual road segments, and finally align the track pieces. We detect intersections in two different (complementary) ways: one is by analyzing the connection pattern of road segments, and the other by analyzing changes in moving direction.

Moreover, we address the problem of inferring room layout of smart indoor environments from people's trajectories, in the context of a smart meeting room. For this purpose, we first detect people's instantaneous activities using their trajectories, fuse the instantaneous activities into higher-level activities over a period of time, and utilize them to update the occupancy maps. The occupancy maps are used to find the location of furniture, e.g., table and chairs.

## 1.2 Contributions and Publications

The main contributions presented in this thesis are as follows:

- We propose a novel trajectory clustering method based on a pairwise trajectory dissimilarity measure. Spatial dissimilarity between two trajectories is defined in terms of the residual Euclidean distance between trajectory points after optimally aligning the traversal order of both trajectories. Similarly, temporal dissimilarity is defined in terms of the residual

velocity difference between trajectory points. We apply a greedy method to group the trajectories directly based on a similarity criterion. This research has been published in the proceedings of International Conference on Distributed Cameras [Xie 14b, Xie 15a].

- We propose two novel methods to jointly align *many* trajectories. In the first method, we apply a "stretch and then compress" strategy based on the DTW algorithm to align the trajectories one by one in ascending order of average dissimilarity. This method produces *one-to-one* correspondences among the points of the warped trajectories, which contain fewer points than any of the original trajectories. One point in one of the warped trajectory corresponds to exactly one point in every other warped trajectory at the same time index.

  In the second proposed method, a greedy procedure directly locates a good warp path, by jointly traversing all trajectories in a way which keeps the "current" points on the trajectories close to each other. Concretely, this involves an iterative procedure to cluster successor points on the trajectories. This method does not use DTW, but rather relies on the statistics of many trajectories to compute a faithful association. This avoids the need for the back-tracking approach DTW, which becomes computationally intractable as the number of trajectories increases. The second method produces *many-to-one* point correspondences between trajectories.

  This research resulted in one publication in the International Journal of Geo-Information [Xie 15b], and several papers in the proceedings of international conferences [Xie 14a], [Xie 14c]. Furthermore, three journal publications have been submitted to the "Journal of Ambient Intelligence and Humanized Computing and another publication" , "Transportation Research Part C: Emerging Technologies" and "IEEE Transactions on Intelligent Transportation Systems."

- We propose two novel approaches to detect intersections from GPS traces. In the first method, intersections are defined as locations where the road users change moving direction. We first calculate the moving direction at each GPS point using the next point ahead located at least a minimal distance from it, and detect turning points, i.e., points where the moving directions changes. The turning points are grouped into intersection candidates depending on their distance. Finally, we remove simple road bends where road users always change their moving directions in the same way.

  In the second method, intersections are defined as junctions which connect at least three road segments. Common sub-tracks between each pair of GPS traces are found using Longest Common Sub-Sequence (LCSS). Their starting and ending points are collected as connecting points. Using Kernel Density Estimation (KDE), we then model the statistical distribu-

tion of these connecting points. The local maximums of this distribution are considered intersections.

This research resulted in one paper in the proceedings of the IEEE Intelligent Transportation Systems Conference [Xie 14c]. Furthermore, a paper has been accepted by 2016 IEEE International Geo-science and Remote Sensing Symposium (IGARSS 2016) [Xie 16c].

- We propose an approach to infer room layout in a meeting environment from people's trajectories in the room. We first extract speed and height information from the trajectories, and input them to a SVM classifier, so as to categorize people's instantaneous activities. We then merge the instantaneous activities into higher-lever activity at each time period. We build two occupancy maps, one for sitting space, and one for walking space. The sitting map is updated using people's sitting activities at each time period, and the walking map is updated using people's walking activities. Finally, the locations of chairs are inferred from the sitting map, and the locations of table from the walking map. This work has been published in the proceeding of 2012 International Conference on Distributed Smart Cameras (ICDSC) [Xie 12].

In total, the research during this PhD resulted in five publications in international peer-reviewed journals: two published [Xie 15b, Bo 14], and three articles under review [Xie 16d, Xie 16a, Xie 16b]. Furthermore ten papers have been published in the proceedings of international conferences [Xie 12, Grünwedel 12, Xie 13b, Xie 14c, Xie 14a, Xie 14b, Xie 13a, Eldib 14, Xie 15a, Eldib 15]. One paper is accepted by 2016 IEEE International Geo-science and Remote Sensing Symposium [Xie 16c].

## 1.3 Outline

The outline of the thesis is as follows:

Chapter 2 focuses on trajectory clustering using trajectory dissimilarity measures. Spatial and temporal dissimilarity are measured separately using two different trajectory alignments. The trajectories are first clustered based on the spatial dissimilarity, resulting in spatial clusters of normal trajectories, and outliers (abnormal trajectories) in which people follow very unusual routes. For every spatial cluster, temporal dissimilarity is used to recognize unusual speed profiles. We evaluate our method on the trajectories of factory workers, and compare with other dissimilarity measures.

Chapter 3 addresses the problem of jointly aligning *many* trajectories. Two methods are proposed to establish the point correspondences among the trajectories. The first method builds *one-to-one* correspondences based on pairwise trajectory alignment with a "stretch and then compress" strategy. The second method builds *many-to-one* correspondences by finding the a warp path through the local dissimilarity tensor using a "greedy" approach. We evaluate

the performance of our methods on both factory worker trajectories and vehicle trajectories. Moreover, we also compare the quality of the extracted average trajectories using the proposed alignments to those obtained with state-of-the-arts methods. Furthermore, we show how to utilize the established point correspondences in our two applications, work cycle optimization in Chapter 5 and road network inference in Chapter 6.

Chapter 4 presents a methodology to infer the presence and location of objects in the meeting environment from people's trajectories, such as chairs and table. We build two occupancy maps separately for sitting space and walking space using people's activities. Chairs are recognized from the sitting map, and a table from the walking map. We evaluate the performance of our method on several meeting experiments.

Chapter 5 details our first application: work cycle optimization, which aims to optimize the work cycle for the factory workers using their trajectories, and guide them to execute their work cycle efficiently. Using the trajectory alignment proposed in Chapter 3, we build a prototypical route and prototypical velocity and dwell time along the route for each spatial cluster detected in Chapter 2, so as to describe how the workers should execute each type of work cycle both spatially and temporally. Moreover, we also analyze the trajectories statistically with the help of the point correspondences established, such as location and velocity variance along the prototypical routes. We demonstrate that performance of trajectory alignment on averaging trajectories and statistically analyzing trajectories.

Chapter 6 focuses on the other application: road network inference, which aims to infer the road network from GPS trajectories. The road network in our work includes three elements, which are all automatically extracted: intersections, intersection connectivity and road segments. We propose two methods to detect intersections from GPS trajectories. One is based on detecting turning points where the road users change their moving directions. The other one is based on detecting connecting points where three road segments intersect. GPS traces are segmented into track pieces by the directly-connected intersections. The track pieces for each road segment are aligned using the methods presented in Chapter 3, so as to build the geometric representation of the road segment. We provide both qualitative and quantitative results on a GPS trajectory dataset and compare our approaches to state-of-the-art methods.

Chapter 7 presents the general conclusions of this dissertation.

# 2

# Trajectory Clustering

In the last decades, the use of visual sensors has grown substantially in a wide variety of applications [Zhou 07, Wang 13a]: sport games analysis, parking lot surveillance, smart home and other intelligent environments. It is possible to collect trajectories of moving objects over sufficient time using tracking techniques. During the last ten years, hand-held Global Positioning System (GPS) devices have become popular [Bellens 11, Lee 13], and GPS trajectories with geographical data are much more easily obtainable from cars, taxis and trucks but also from cyclists and pedestrians.

A trajectory is an itinerary that an object follows through space as a function of time. Moving objects following the same itinerary behave similarly, and their behavior patterns can be deduced by analyzing their trajectories [Morris 08]. A trajectory produced by following an unusual itinerary indicates abnormal behaviors. Therefore, it is possible to learn the behavior patterns and detect the abnormalities through trajectory clustering.

Trajectory clustering is a general machine learning technique to identify structure in unlabeled trajectory data [Zhang 06]. Trajectory clustering aims to group the trajectories into similar categories, where the trajectory dissimilarity within the same cluster is minimized, and the trajectory dissimilarity between different clusters is maximized. A fundamental issue in trajectory clustering is to measure the (dis)similarity between the trajectories.

In literature, Euclidean distance between coordinates of the corresponding points in the trajectories is the most common method to measure their dissimilarity. The corresponding points can be found by simply matching the data points at the same time instance on each raw trajectory [Fu 05], by matching points within a time window [Piciarelli 06], or by aligning the trajectories point by point using Dynamic Time Warping (DTW) [Niennattrakul 09] or Longest Common SubSequence (LCSS) [Vlachos 02]. Some other researchers calculate the Euclidean distance between trajectories in a Principle Component Analysis (PCA) subspace to indicate their dissimilarity [Bashir 07]. Besides, Hausdorff distance and Edit distance are also used to measure the dissimilarity [Zhou 07, Atev 06].

Most of the methods aforementioned actually calculate the spatial dissimi-

larity using the coordinates on the trajectories. Using the spatial dissimilarity, trajectories of mobile objects following the same itinerary are clustered into the same group. Trajectories, which are produced by the mobile objects following unusual itineraries, are detected as abnormalities.

In our application of road network inference from GPS trajectories, trajectory clustering using spatial dissimilarity can be applied to detect the abnormal trajectories, which deviate from the main roads. Only the normal trajectories spatially similar to each other will be used to infer the road's geometric representation. Removing the spatially abnormal trajectories from road network inference can improve the accuracy of the road's geometric representation.

In the other application of work cycle optimization for the factory workers, we aim to discover the factors leading to work inefficiency and optimize the work cycle using the trajectories produced during the work cycle execution. The work cycle is scheduled for the factory workers as: picking up required tools and parts from the storage rack and then assembling the parts using the tools at the work station next to the assembly line. However, the executed work cycles are different depending on how the workers explore the area in front of the storage rack. We can cluster the executed work cycles and detect abnormalities using the spatial dissimilarity between the trajectories. The spatial factors of work inefficiency can be inferred from these abnormality, such as: following an unusual itinerary and wandering in front of the storage rack. However, trajectory clustering using spatial dissimilarity is not able to discover the temporal factors leading to work inefficiency, for instance, following the same itinerary but at a very slow speed, dwelling somewhere on the way to or back from the storage rack, etc. Therefore, it is necessary to analyze the temporal dissimilarity between the trajectories as well.

In this chapter, we propose to measure trajectory dissimilarity both spatially and temporally using DTW. In DTW, the trajectories are first warped along the time index by finding the best match between the physical locations along the trajectories; only then, spatial dissimilarity is calculated – on these warped trajectories. Along the warp path, the associated data points are spatially close to each other, but their velocities may be very different. In our application of the work cycle optimization, the places in front of the storage rack, where the workers dwell to pick up tools and parts, are slightly different at different work cycle execution. The best location match makes the velocities mismatched at these places, producing inaccurate temporal dissimilarity. Therefore we propose to find the best velocity match between the trajectories and calculate the temporal dissimilarity on the newly warped trajectories. In this work, we first cluster trajectories according to spatial similarity. For each spatial cluster, we then detect temporal abnormalities by computing the temporal dissimilarity.

Our main contribution is twofold: 1) We calculate the temporal dissimilarity between trajectories using a best velocity match. 2) We propose a greedy clustering method using the dissimilarity measure to find patterns of the trajectories. The results show that the trajectories are clustered into different patterns successfully, and abnormalities are detected both spatial unfit to the

prototypical routes and temporally different from other trajectories in the same spatial cluster.

The remainder of this chapter is organized as follows. In Section 2.1, we introduce the related work. In Section 2.2, we detail the procedure of Dynamic Time Warping. Section 2.3 elaborates how to calculate the spatial and temporal dissimilarity separately. In Section 2.4, we explain our trajectory clustering approach based on the dissimilarity measurement. Section 2.5 shows our experimental results. Finally, we conclude this chapter in Section 2.6.

## 2.1 Related Work

In this section, we provide an overview of state-of-the-art algorithms for trajectory clustering and techniques for measuring the dissimilarity of the trajectories.

### 2.1.1 Clustering Algorithms

Various algorithms have been developed to cluster unlabeled trajectories data [Atev 10]. Depending on how the trajectories are processed, the approaches can be broadly classified into two groups:

**Feature-based methods.** These methods first extract *features* from the trajectories, e.g., average speed and directional histogram, and use these features to group the trajectories into similar categories.

Li *et al.* extract two kinds of features after smoothing the trajectories: Trajectory Re-sampling Point Sets (TRPSs) and Trajectory Directional Histograms (TDHs) [Li 06]. They interpolate all smoothed trajectories at the same space interval, producing a set of re-sampling points for each trajectory, which is called TRPS in their work. All TRPSs contain the same number of data points. For each re-sampled trajectory, they calculate the moving direction at each point, and create a $N$-bins of histogram from the moving directions of all points, which rang from $-\pi$ to $\pi$. The histogram describes statistical moving directional characteristics of the trajectory, which is called TDH consequently. TDHs are used to obtain coarse trajectory clusters, in which the trajectories share a similar directional distribution. Subsequently, trajectories in each coarse cluster are grouped into fine clusters using TRPS.

Anjum and Cavallaro extract features like average speed and "directional distance," and utilized these features to cluster trajectories using the Mean-shift algorithm [Anjum 10].

The accuracy of feature-based methods depends on the feature selection. Methods using average speed as a feature are sensitive to changes in speed over time within each trajectory. For instance, they cluster two trajectories with the same average speed into the same category even when the two speed profiles are very different. In our application of work cycle optimization, the workers' work cycle is scheduled as: walk to the storage rack, and then dwell in front of the storage rack to pick up tools and parts, finally walk to the work station to

assemble the parts. During the journey, the moving speed of the workers goes high (walk), then low (dwell), then back to high (walk). Methods using average speed as a feature are not able to distinguish the executed work cycles in which the workers walk to the storage rack at a lower speed and then rush back to the work station without picking up any tools or parts, if their average speed is the same as that in the scheduled work cycle. Although some features do contain the shape of the trajectory, such as TDH [Li 06], they do not capture the fine details of the timing. Also, TDH does not cope with the object staying stationary. In our application, the factory workers dwell in front of the storage rack to pick up tools and parts. No matter how long the workers dwell there, their trajectories will be clustered into the same group using TDH as a feature, as far as their moving directional distributions are the same. Therefore feature-based methods are not suitable to our application of work cycle optimization, which need fine timing information.

**Direct methods.** These methods operate on the data points of the trajectories directly, without first converting them into feature sets. They first measure the dissimilarity between two time series of data point using a variety of approaches [Niennattrakul 09, Vlachos 02, Zhang 06], such as Dynamic Time Warping (DTW), Longest Common Sub-Sequence (LCSS), etc. Then they group the trajectories, which are highly similar to each other, into the same category. In literature, two types of trajectory clustering methods using dissimilarity measurement are popular: *agglomerative hierarchical clustering* and *spectral clustering*.

Agglomerative hierarchical clustering is a bottom-up clustering method which starts by placing each trajectory in its own cluster, and then merge pairs of newly-formed clusters into larger clusters until a hierarchical tree is formed. The resulting tree can be analyzed at different levels to group trajectories over a variety of scales. Fashandi and Moghaddam define a dissimilarity measure based on LCSS and apply agglomerative hierarchical clustering to group vehicle trajectories into different behaviors, such as overtaking, zigzag, lane changing, returning, turning and circling [Fashandi 05]. They apply a stopping criterion for the clustering procedure: when the number of clusters is equal to the specific number of behavior types. Biliotti *et al.* apply Independent Component Analysis (ICA) to create trajectory representations [Biliotti 05, Antonini 06]. They use the Hausdorff distance between the estimated independent components to measure trajectory dissimilarity, and agglomerative hierarchical clustering to group the trajectories. Their stopping criterion is when the distance between two clusters exceeds the predefined threshold.

Spectral clustering operates on the matrix of pairwise similarity. It relies on eigenvalue decomposition of the similarity matrix. The $K$ largest eigenvectors are used to cluster the trajectories into $K$ groups. Fu *et al.* use average Euclidean distance between two vehicle trajectories as their dissimilarity measure, and spectral clustering to classify the trajectories into motion patterns [Fu 05]. Atev *et al.* apply modified Hausdorff Distance to measure the trajectory dissimilarity, and spectral clustering to learn the traffic patterns at

the intersections from vehicle trajectories [Atev 06]. Spectral clustering has become popular recently because of its efficient computation and good clustering results. However, spectral clustering may be sensitive to the choice of parameters for dissimilarity measure. Besides, it requires to specify the number of clusters $K$ in advance. But this information is usually unknown, and extra effort is needed to determine the optimal choice of $K$.

We apply a greedy clustering method similar to agglomerative hierarchical clustering. Instead of merging pairwise similar clusters hierarchically, we group the trajectories directly based on a similarity criterion. Without merging sub-clusters at lower layers to clusters at higher layers as the agglomerative hierarchical clustering does, our method is more computationally efficient.

### 2.1.2 Dissimilarity Measure

In the past decades, there has been much progress on detecting patterns of the moving objects by measuring their trajectory similarity. The main difficulty of similarity measure is the unequal duration of the trajectories: Objects may move at different speeds, leading to differences in number of samples for spatially similar trajectories and difficulties in finding corresponding points on trajectories. A related problem is that different types of recording devices may lead to various sampling rates as well. Some researchers simply select the first $N$ points in both trajectories, and calculate the average Euclidean distance between them to measure the dissimilarity [Fu 05]. This measure may be very inaccurate considering their differences in sampling rate. In literature, some techniques have been proposed to measure the (dis)similarity between two trajectories with differing sampling.

Niennattrakul and Ratanamahatana compute the similarity between pairs of trajectories using DTW [Niennattrakul 09]. The trajectories are warped along the time axis by finding the optimal match between the physical locations along the trajectories with certain restrictions. The warped trajectories contain the same number of data points, and the data points at the same time index in the warped trajectories are matched to each other. The overall Euclidean distance between the matched points is used as the representation of their dissimilarity. DTW allows comparing trajectories with differing numbers of samples.

Vlachos *et al.* propose an alignment tool based on finding the longest common sub-sequence between trajectories recorded by Globe Positioning System (GPS) equipments [Vlachos 02]. A trajectory subsequence is a sequence of data points that appears in the same relative order within the original trajectory, but not necessarily occupies consecutive positions. A common subsequence of two trajectories is a subsequence present in both of them. A longest common subsequence is a common subsequence of maximal "length". The "length" here refers to the number of data points in the subsequence. Using LCSS, the trajectories are aligned only at the data points in their longest common subsequence. The other data points in each trajectory, which not appear in the longest common subsequence, are called *unmatched points*. The ratio of the length of the longest common subsequence to the minimal length of two trajectories is used

to measure their dissimilarity. Different from DTW, LCSS does not associate all of the points on the trajectories, but allows some points unmatched, making the dissimilarity measurement more robust to noise.

Piciarelli and Foresti design a dissimilarity measurement between a trajectory and a given cluster of trajectories [Piciarelli 06]. The given cluster is represented as a list of vectors, including position data and temporal information. The distance of a trajectory from a given cluster is calculated as mean Euclidean distance of every point of the trajectory to its nearest point in the cluster. The nearest point of one point is found by searching its neighbor points whose time indexes are within a temporal window. The distance calculation starts from the first point of the trajectory, and ends at the last point. As the time instance of the point grows, the size of the temporal window increases as well. The given cluster indicates the temporal information that the objects arrive at a specific position. In the trajectories to be clustered, the object may move at a different speed, leading to arrival at the same position at a very different time from the scheduled time in the given cluster. The time difference if exists will accumulate as the time instance of the point grows in the trajectory. Therefore they apply a temporal window with increasing size to deal with the temporal difference.

Bashir *et al.* use Principle Components Analysis (PCA) to transform the coordinates of the trajectories into a lower-dimension subspace and calculate Euclidean distance between the PCA coefficients to express the dissimilarity [Bashir 07]. Benefit from the reduced dimension, it is more computationally efficient to measure the dissimilarity using the first $K$ coefficients. But PCA-based dissimilarity measure can not distinguish the speed variation in the trajectories.

Pelekis *et al.* propose the Locality In-between Polylines (LIP) distance function upon the (projected on the Cartesian plane) routes of the trajectories [Pelekis 07]. LIP calculates the area of the shape formed by two 2D polylines; it is used by authors for measuring the spatial similarity of vehicle GPS trajectories. They also extend LIP to spatiotemporal LIP using a weighting function, which is dependent on the local temporal distance, so as to measure the spatiotemporal similarity between two trajectories. Modified Hausdorff distance and Edit distance are also used in trajectory similarity measurement by some researchers [Atev 06, Zhou 07].

Zhang [Zhang 06] and Morris [Morris 09] evaluated the methods above using different datasets. We will compare our dissimilarity measure with some of the methods above by applying the same clustering method on these measurements.

## 2.2 Dynamic Time Warping

Dynamic Time Warping (DTW) is an algorithm for finding the optimal alignment of two time series by warping them non-linearly in the time dimension. DTW was originally developed for speech recognition [Sakoe 78], to estimate similarities in acoustic features between pairs of words irrespective of speed

variations, and later has been employed in other various fields, such as data mining [Keogh 05], and behavior analysis [Suzuki 07]. In these applications, the elements of the time series thus vary from acoustic features, over locations, velocities, behavior features, to financial data, etc. In our application of work cycle optimization, a time series is a trajectory belonging to a factory worker, which consists a sequence of data points. In our application of road network inference, a time series is a trajectory belonging to a road user, which consists of a sequence of GPS data points.

We adopt the following notations. Suppose we have two trajectories $\mathbf{r}_1(t_1)$, $t = 1 \dots T_1$ and $\mathbf{r}_2(t_2)$, $t_2 = 1, \dots T_2$. The numbers of samples $T_1$ and $T_2$ of the two trajectories need not be the same. DTW minimizes the overall dissimilarity between the two trajectories, i.e. the sum of the local dissimilarity between two individual points $\mathbf{r}_1(w_1(k))$ and $\mathbf{r}_2(w_2(k))$, which are associated by the *warp path* $(w_1(k), w_2(k)$, $k = 1, \dots K$. Not all such associations are allowed. Specifically we consider only warp paths which belong to a subset $\Gamma$ of all possible associations.

$$D_\Gamma(\mathbf{r}_1, \mathbf{r}_2) \stackrel{\triangle}{=} \min_{(w_1, w_2) \in \Gamma} \frac{1}{K} \sum_{k=1}^{K} \sqrt{(\mathbf{r}_1(w_1(k)) - \mathbf{r}_2(w_2(k)))^2} \tag{2.1}$$

where $w_1(k)$ and $w_2(k)$ associate corresponding time instances $t_1$ of the first trajectory and $t_2$ of the second trajectory, and Euclidean distance, $\sqrt{(\mathbf{r}_1(w_1(k)) - \mathbf{r}_2(w_2(k)))^2}$, is used to measure the dissimilarity between the two associated points.

The associations are typically subject to the following constraints:

- **Boundary** The warp path must start at the beginning of the two trajectories, and end at the ending of the two trajectories: $(w_1(1), w_2(1)) = (1, 1)$ and $(w_1(K), w_2(K)) = (T_1, T_2)$.
- **Monotonicity** The warp path must be monotonically nondecreasing along both time axes: $w_1(k-1) \leq w_1(k)$ and $w_2(k-1) \leq w_2(k)$. This constraint guarantees that the warp path will not roll back on itself.
- **Continuity** The warp path does not jump in either of the time indexes: $(w_1(k), w_2(k)) - (w_1(k-1), w_2(k-1)) \in \{(0,1), (1,0), (1,1)\}$. This constraint guarantees the alignment does not omit any point of either trajectory.

There are a large number of warp paths that satisfy all of the above constraints. The most naive approach to find the optimal warp path would be to enumerate all possible paths, and select the path that gives the smallest overall dissimilarities between the two trajectories. This will result in enormous computational cost. To overcome this challenge, Dynamic Programming is employed to find the path efficiently.

Dynamic Programming is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those

subproblems firstly, and then using them to find the solution to the complex problem. In our case, rather than finding the warp path, along which the overall dissimilarity between two whole trajectories is minimized, we first find the sub-path, along which the overall dissimilarity between two smaller sub-trajectories are minimized, and use them to find the optimal warp path for the larger sub-trajectories, until we find an optimal warp path for the whole trajectories.

The first stage of implementing Dynamic Programming is to create a two-dimensional $T_1 \times T_2$ local dissimilarity matrix $C_1$. The value at each cell, $C_1(t_1, t_2)$, is the Euclidean distance between point $\mathbf{r}_1(t_1)$ of the first trajectory and point $\mathbf{r}_2(t_2)$ of the second trajectory. In the second stage, we build a two-dimensional $T_1 \times T_2$ overall dissimilarity matrix $C_2$. The value at each cell, $C_2(t_1, t_2)$, is the minimal overall dissimilarity between two sub-trajectories $\mathbf{r}_1(i)$, $i = 1, \ldots t_1$ and $\mathbf{r}_2(j)$, $j = 1, \ldots t_2$. $C_2$ is recursively calculated from $(1, 1)$ to $(T_1, T_2)$ as:

$$C_2(t_1, t_2) \triangleq C_1(t_1, t_2) + \min \begin{pmatrix} C_2(t_1 - 1, t_2) \\ C_2(t_1, t_2 - 1) \\ C_2(t_1 - 1, t_2 - 1) \end{pmatrix} \tag{2.2}$$

where $t_1 \in [1, T_1]$ and $t_2 \in [1, T_2]$.

Once the overall dissimilarity matrix $C_2$ is built, the optimal warp path $(w_1(k), w_2(k))$, $k = 1, \ldots K$ can be found using a "backtrack" procedure of searching adjacent cells from $(T_1, T_2)$ to $(1, 1)$ through $C_2$, as described by Algorithm 1. Given the $k^{th}$ element of the warp path $(w_1(k), w_2(k)) = (t_1, t_2)$, one of its adjacent cells $\{(t_1 - 1, t_2), (t_1, t_2 - 1), (t_1 - 1, t_2 - 1)\}$, whose value is the smallest, is selected as the next element $(w_1(k - 1), w_2(k - 1))$. Because $(w_1(k), w_2(k)) - (w_1(k - 1), w_2(k - 1)) \in \{(1, 0), (0, 1), (1, 1)\}$, the warp path satisfies the Continuity and Monotonicity constraints.

Fig. 2.1 shows an example of aligning two GPS trajectory pieces for the same road. The two trajectories are spatially similar to each other. Therefore DTW establishes only three two-to-one correspondences between the points, warps the trajectories to 20 points.

Fig. 2.2 shows another example of aligning factory worker trajectories. The trajectories have been obtained with a multi-camera system. For clarity, the figure only shows small pieces of the trajectories. In this example, the worker dwells at different locations in the two trajectories. This results in many-to-one correspondences between the two trajectories, and warped trajectories with 54 points.

## 2.3 Dissimilarity Measure

The dissimilarity measure between pairwise trajectories can be used to cluster a set of trajectories into groups, so as to analyze the objects' moving patterns. DTW warps two trajectories for optimal point associations, which minimize the

**Algorithm 1** FindPath

**Input:** $C_2$
**Output:** $(w_1(k), w_2(k))$,$k = 1, \ldots K$
Initialization $(w_1(1), w_2(1)) \leftarrow (T_1, T_2)$, $k \leftarrow 2$
**while** $t_1 + t_2 > 2$ **do**
  **if** $t_1 = 1$ **then**
    $t_2 \leftarrow t_2 - 1$
  **else if** $t_2 = 1$ **then**
    $t_1 \leftarrow t_1 - 1$
  **else**
    **switch** $\min(C_2(t_1 - 1, t_2), C_2(t_1, t_2 - 1), C_2(t_1 - 1, t_2 - 1))$
      **case** $C_2(t_1 - 1, t_2)$
        $t_1 \leftarrow t_1 - 1$,
      **case** $C_2(t_1, t_2 - 1)$
        $t_2 \leftarrow t_2 - 1$
      **case** $C_2(t_1 - 1, t_2 - 1)$
        $t_1 \leftarrow t_1 - 1$, $t_2 \leftarrow t_2 - 1$
  **end if**
  $(w_1(k), w_2(k)) \leftarrow (t_1, t_2)$
  $k \leftarrow k + 1$
**end while**
$(w_1(k), w_2(k)) \leftarrow (1, 1)$, $K \leftarrow k$
$w_1 =$reverse$(w_1)$, $w_2 =$reverse$(w_2)$

overall dissimilarity between individual points of the trajectories. By associating the points using their physical locations, the spatial dissimilarity between the two trajectories will be measured. By associating the points using their velocities, the temporal dissimilarity will be measured.

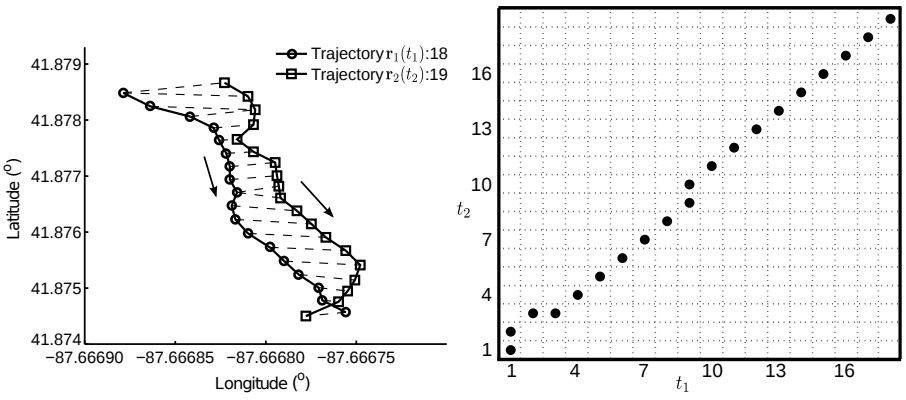
### 2.3.1 Spatial Dissimilarity

DTW takes two trajectories (sequences of points), $\mathbf{r}_1(t_1)$, $t_1 = 1, \ldots T_1$ and $\mathbf{r}_2(t_2)$, $t_2 = 1, \ldots T_2$, and uses the physical locations of the points $\mathbf{r}_1(t_1) = (x_1(t_1), y_1(t_1))$ and $\mathbf{r}_2(t_2) = (x_2(t_2), y_2(t_2))$ as its inputs. The local dissimilarity between two individual points $\mathbf{r}_1(t_1)$ and $\mathbf{r}_2(t_2)$, is calculated as the Euclidean distance between their physical locations:

$$C_1(t_1, t_2) = \sqrt{(x_1(t_1) - x_2(t_2))^2 + (y_1(t_1) - y_2(t_2))^2} \tag{2.3}$$

where $t_1 \in [1, T_1]$ and $t_2 \in [1, T_2]$.

Using Euclidean distance between physical locations as local dissimilarity, the overall location dissimilarity matrix $C_2$ is calculated as shown in Equation 2.2. The warp path, $(w_1(k), w_2(k))$, $k = 1, \ldots K$ is produced by "backtracking" the minimal overall dissimilarity through $C_2$ from cell $(T_1, T_2)$ to cell $(1, 1)$, as




**(a)** Point associations **(b)** Warp path

**Figure 2.1:** *Example* 1 *of trajectory alignment.* In (a), the associated points are connected using black dashed lines. The arrows represent the moving directions of the road users in each trajectory, respectively. (b) shows the optimal warp path through the overall dissimilarity matrix.

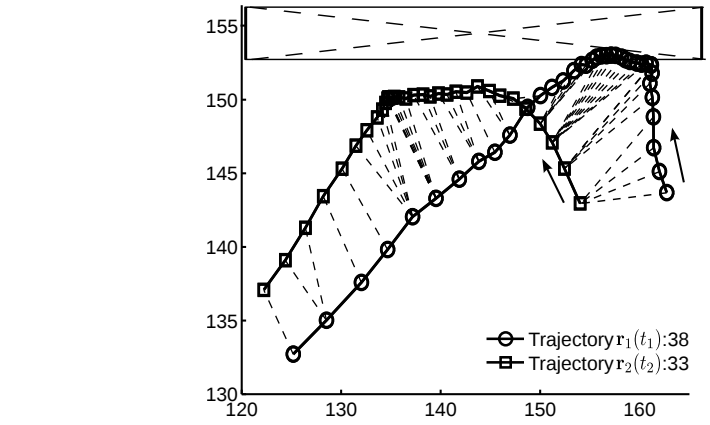
described in Algorithm 1. We now define the *spatial dissimilarity* $D_s(\mathbf{r}_1, \mathbf{r}_2)$ between two trajectories as follows:

$$D_s(\mathbf{r}_1, \mathbf{r}_2) \triangleq \frac{1}{K} \sum_{k=1}^{K} \sqrt{(x_1(w_1(k)) - x_2(w_2(k)))^2 + (y_1(w_1(k)) - y_2(w_2(k)))^2} \tag{2.4}$$

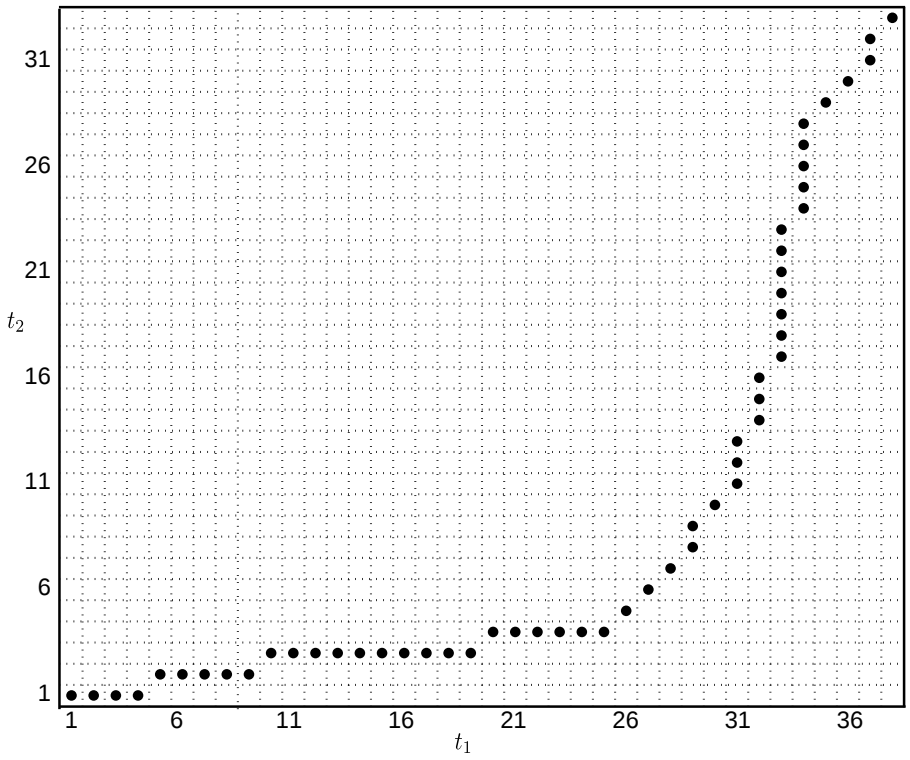
In other words: this is mean Euclidean distance between the physical locations of the points in the trajectories after optimally warping them in the time dimension to minimize their overall location dissimilarity. The spatial dissimilarity between two trajectories which traverse the same or similar spatial locations will be low, even if the traversal is at very different speed. Therefore the spatial dissimilarity is useful to cluster the trajectories into groups in which the moving objects follow different paths, and detect the abnormal trajectories caused by unusual paths or inaccurate data. However it is not an adequate measure to distinguish the trajectories in which the moving object travels at an unusual speed.

### 2.3.2 Temporal Dissimilarity

The spatially-similar trajectories may be different temporally. For instance, the object follows the same path at different speeds. We use the instantaneous velocity of the trajectories to measure their dissimilarity temporally. The instantaneous velocity is defined as $d\mathbf{r}/dt$ and can be estimated using various numerical techniques, the simplest one being $\frac{d\mathbf{r}}{dt} \approx (\mathbf{r}(t+1) - \mathbf{r}(t-1))/2$. However, some temporal smoothing can provide temporal robustness against




**(a)** Point associations




**(b)** Warp path

**Figure 2.2:** *Example* 2 *of trajectory alignment.* In (a), the associated points are connected using black dashed lines. The arrows indicate the direction of the motion along these trajectories separately. (b) shows the optimal warp path through the overall dissimilarity matrix.

data noises. In the following we will focus primarily on the magnitude $|\frac{d\mathbf{r}}{dt}|$ of the velocity, but omit objects' moving directions because they are similar in two spatially-similar trajectories.

The instantaneous velocities $v_1(t_1)$, $t_1 = 1, \ldots T_1$ and $v_2(t_2)$, $t_2 = 1, \ldots T_2$ of two trajectories are input to DTW procedure to align the two trajectories optimally. The local dissimilarity between two individual points $\mathbf{r}_1(t_1)$ and $\mathbf{r}_2(t_2)$, is calculated as the absolute difference between their instantaneous velocities:

$$C_1(t_1, t_2) = |v_1(t_1) - v_2(t_2)| \tag{2.5}$$

where $t_1 \in [1, T_1]$ and $t_2 \in [1, T_2]$.

Using absolute differential velocity as local dissimilarity, the overall velocity dissimilarity matrix $C_2$ is calculated as:

$$C_2(t_1, t_2) \triangleq \lambda(t_1, t_2) * C_1(t_1, t_2) + \min \begin{pmatrix} C_2(t_1 - 1, t_2) \\ C_2(t_1, t_2 - 1) \\ C_2(t_1 - 1, t_2 - 1) \end{pmatrix} \tag{2.6}$$

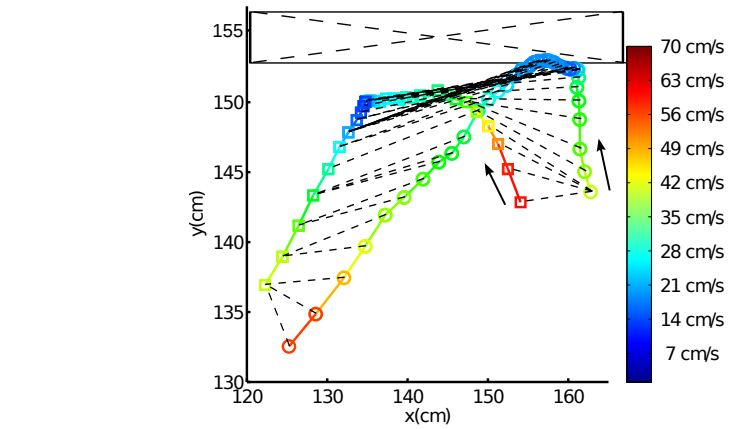
where $\lambda(t_1, t_2)$ is a weighting function, which is dependent on the local distance between the physical locations of the points $\sqrt{(x_1(t_1) - x_2(t_2))^2 + (y_1(t_1) - y_2(t_2))^2}$. This guarantees that one point will not be matched to another point too distant from it [Jeong 11].

The warp path, $(w'_1(k), w'_2(k))$, $k = 1, \ldots K'$, is found by "backtracking" the minimal overall dissimilarity through $C_2$ from cell $(T_1, T_2)$ to cell $(1, 1)$, as described in Algorithm 1. We now define the *temporal dissimilarity* $D_t(\mathbf{r}_1, \mathbf{r}_2)$ between two trajectories as the mean difference between the velocities of the points on the trajectories, after optimally warping them in the time dimension to minimize the overall velocity dissimilarity.

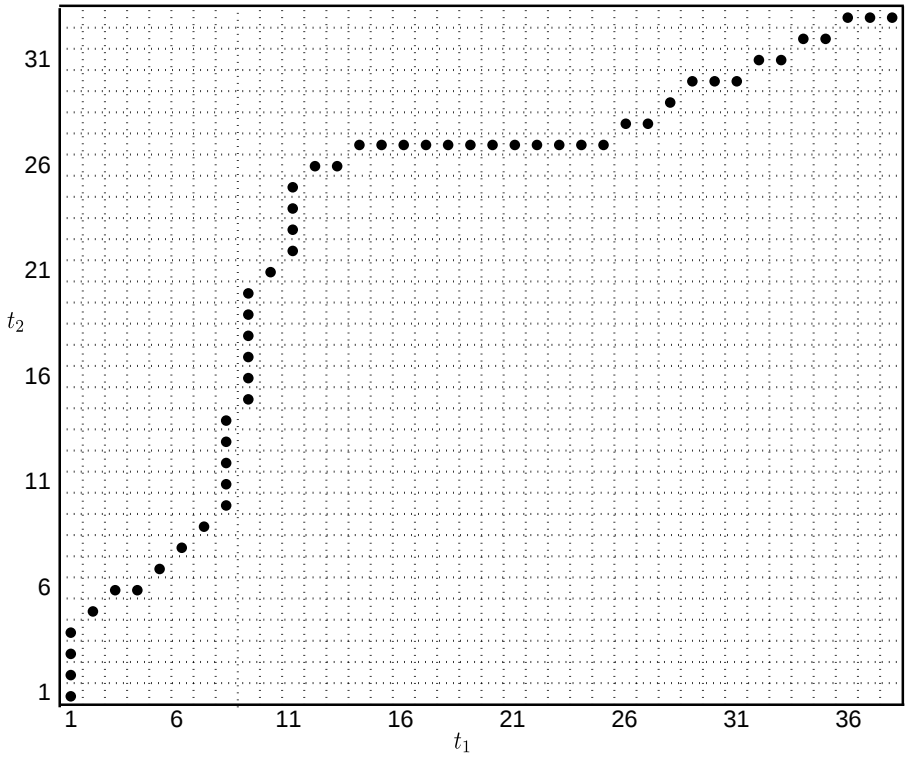
$$D_t(\mathbf{r}_1, \mathbf{r}_2) \triangleq \frac{\sum_{k=1}^{K'} |v_1(w'_1(k)) - v_2(w'_2(k))|}{\sum_{k=1}^{K'} \lambda(w'_1(k), w'_2(k))} \tag{2.7}$$

Although we already get the optimal alignment of the trajectories using the physical locations of their points in Section 2.3.1, they are not suitable to measure the temporal dissimilarity because they are based on minimizing the overall Euclidean distance between the physical locations of the points, rather than the velocities at the points. The point of one trajectory is matched to the point of the other trajectory with the smallest distance, as shown in Fig. 2.2a. The velocities at these matched points may be very different. Temporal dissimilarity aims to describe how differently the moving object changes its velocity in spatially similar trajectories, rather than how far one trajectory deviates from another spatially.

Fig. 2.3 shows an example of aligning two trajectory pieces using the the velocities of their data points. In these two pieces, the factory worker stays in front of the storage rack to pick up tools and parts. Part of the storage rack is shown as a rectangle. Although he or she stays at the right corner in both

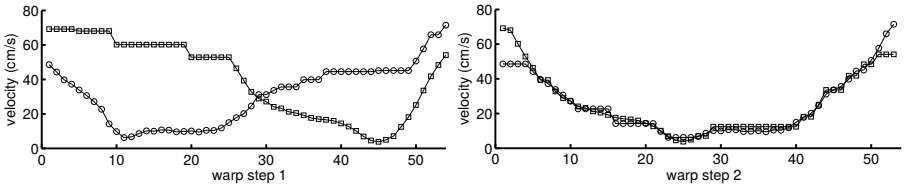



**(a)** Point associations




**(b)** Warp path

**Figure 2.3:** *Example 3 of trajectory alignment.* The same trajectories as shown in Fig. 2.2 are now aligned using DTW based on their point velocities. (a) shows the associated points and their velocities. (b) shows the new warp path.




**(a)** Velocities along warp path 1 **(b)** Velocities along warp path 2

**Figure 2.4:** *Point velocities along different warp paths.* (a) depicts the velocity of the data points on the trajectories along the optimal warp path $(w_1(k), w_2(k))$, $k = 1, \dots K$, which are produced using physical locations of the points as inputs to DTW, while (b) shows the velocity along another optimal warp path, $(w'_1(k), w'_2(k))$, $k = 1, \dots K'$, produced based on velocity difference.

**Algorithm 2** Cluster trajectories

**Input:** $\{\mathbf{r}_n, n = 1, \dots V\}$
**Output:** $\{G_i : i = 1, \dots I\}$

Initialization $G_{un} \leftarrow \{\mathbf{r}_n, n = 1, \dots V\}$ $i \leftarrow 1$
**for** each $\mathbf{r} \in G_{un}$ **do**
  $G_i^{(1)} \leftarrow \{\mathbf{r}\}$, $k \leftarrow 1$
  **for** each $\mathbf{r}' \neq \mathbf{r} \in G_{un}$ **do**
    **if** $\frac{1}{|G_i^{(k)}|} \sum_{\mathbf{r}'' \in G_i^{(k)}} D_s(\mathbf{r}', \mathbf{r}'') \leq d_{thre}$ **then**
      $G_i^{(k+1)} \leftarrow G_i^{(k)} \cup \{\mathbf{r}'\}$
      $k \leftarrow k + 1$
    **end if**
  **end for**
  $G_{un} \leftarrow G_{un} \setminus G_i^{(k)}$
**end for**

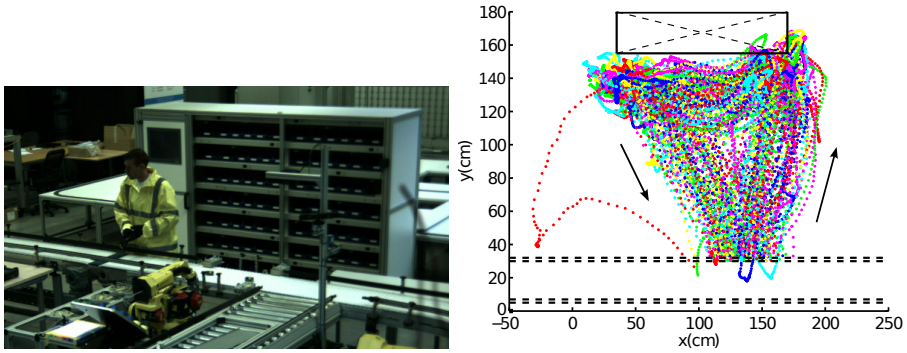
trajectories, the staying places are slightly different. In Fig. 2.3a, the points of two trajectories at the staying places are associated.

The velocity of data points along different warp paths are shown in Fig. 2.4. The alignment produced using the point velocities as inputs, indeed minimize the velocity dissimilarity of the trajectories much better than the spatial alignment produced using the physical locations.

## 2.4 Trajectory Clustering

The spatial dissimilarity is calculated as explained in Section 2.3.1 and then used to cluster the trajectories as shown in Algorithm 2.

An initial cluster $G_i^{(1)}$ is grown from a single seed trajectory $\mathbf{r}_i^{(1)}$ by iteratively adding trajectories to the cluster $G_i$ if they are spatially similar to all the trajectories in $G_i$. The most recently tested trajectory is added to the cluster




**(a)** Work station **(b)** Trajectories

**Figure 2.5:** *Work station and the trajectories.* In (a), a factory worker stands at the work station assembling the parts. There is a storage rack behind him, where the tools and parts are kept. (b) shows 124 trajectories of factory workers in different colors. The location of the storage rack is shown as a black rectangle near the top of the figure; the dashed lines at the bottom show the location of the conveyor belt.

$G_i^{(k+1)} \leftarrow G_i^{(k)} \cup \{\mathbf{r}'\}$ and used with other trajectories in this cluster to test whether a new trajectory belongs to this cluster or not. This procedure is then repeated on the trajectories which have not yet been clustered. The outputs of this step are $I$ spatial clusters, $G_i = \{\mathbf{c}_i^{(n)}, n = 1, \dots |G_i|\}$, $i = 1, \dots I$, with $|G_i|$ the number of turning trajectories per cluster. Clusters with small number of trajectories are considered as spatial abnormalities.

Although the trajectories in each cluster $G_i$ are spatially similar to each other, the velocities of the trajectories may be different. To distinguish the trajectory in which the moving object travels at an unusual speed, we group the trajectories in each spatial cluster $G_i$ using Algorithm 2, but with measuring the temporal dissimilarity $D_t(\mathbf{r}', \mathbf{r}'')$ between pairwise trajectories, instead of the spatial dissimilarity $D_s(\mathbf{r}', \mathbf{r}'')$. The outputs of this step are sub-clusters for each spatial cluster $G_i$. The sub-cluster with the largest number of trajectories is considered as temporal normalities, and other sub-clusters are temporal abnormalities.

## 2.5 Results

In order to evaluate the accuracy of the presented dissimilarity measures, we evaluate them on a data set of factory worker trajectories at an assembly line (Factory dataset). In this data set, the worker repeats his or her work by going to the storage rack, picking up tools, coming back to the assembly platform and doing his operation, which we call an executed work cycle. We recorded two workers repeating the procedure separately at a work station of $3 \times 2$ $m^2$ as shown in Fig. 2.5a, and obtained 124 trajectories in total from our multi-

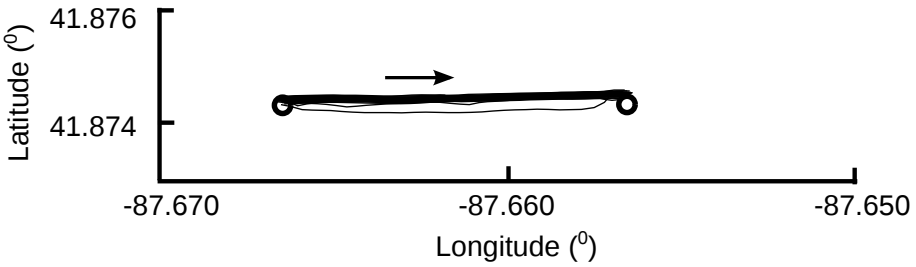



**Figure 2.6:** *Trajectories on one road segment* 80 trajectories (black lines) connect the left intersection to the right intersection (black circle). The arrow indicates the direction of the motion along these trajectories separately.

camera vision system. Fig. 2.5b shows all of the trajectories in different colors. The storage rack is shown in a rectangle at the top, and the conveyor belt in dashed lines at the bottom.

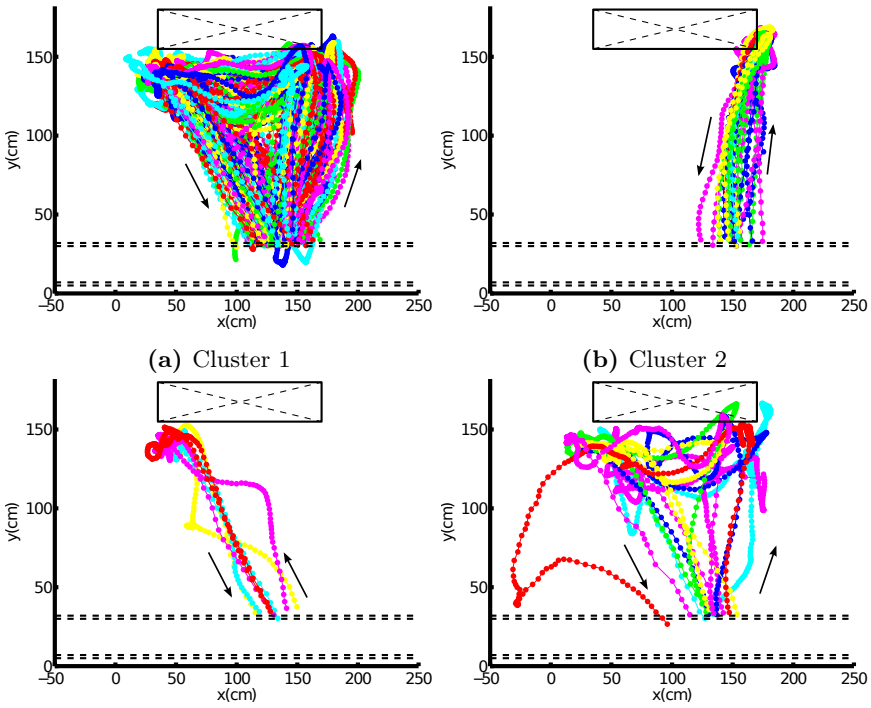
Besides the Factory dataset, our algorithm is also tested on detecting spatial abnormalities of vehicle trajectories for road network inference. We will show the clustering results for trajectories of one road segment, as shown in Fig. 2.6. The results of the full data set will be shown in Section 6.6 after the road segmentation.

## 2.5.1 Our Results

### 2.5.1.1 Results of Factory Worker Trajectories

By manually checking the videos of the Factory dataset, we find the workers mainly execute the work cycle in three types of ways by visiting different area of the storage rack: visiting the left and then the right corner of the storage rack; visiting just the left corner; and visiting just the right corner. This results in three spatial clusters of trajectories. Occasionally the workers pick up the tools and parts in a very different way, for instance, visiting other area of the work station, visiting the storage rack from the right to the left corner, etc. This unusual executed work cycles lead to spatially abnormal trajectories. By clustering the trajectories, we can find the workers' moving patterns and their abnormalities of visiting the storage rack, so as to analyze the factors leading to work inefficiency, and eventually build a prototypical work cycle for them to follow.

Our algorithm groups the 124 trajectories into 10 clusters based on their spatial dissimilarity. As shown in Fig. 2.7a, there are 100 trajectories in the first cluster, in which the workers pick up tools and parts by walking from right to left through the whole area in front of storage rack. In the second cluster with 13 trajectories depicted in Fig. 2.7b, the workers walk to the right corner of the storage rack and back to the operation place directly, while only the left corner of the storage rack is visited in the third cluster with 4 trajectories in Fig. 2.7c. The three types of executed work cycles are detected successfully




**(a)** Cluster 1 **(b)** Cluster 2

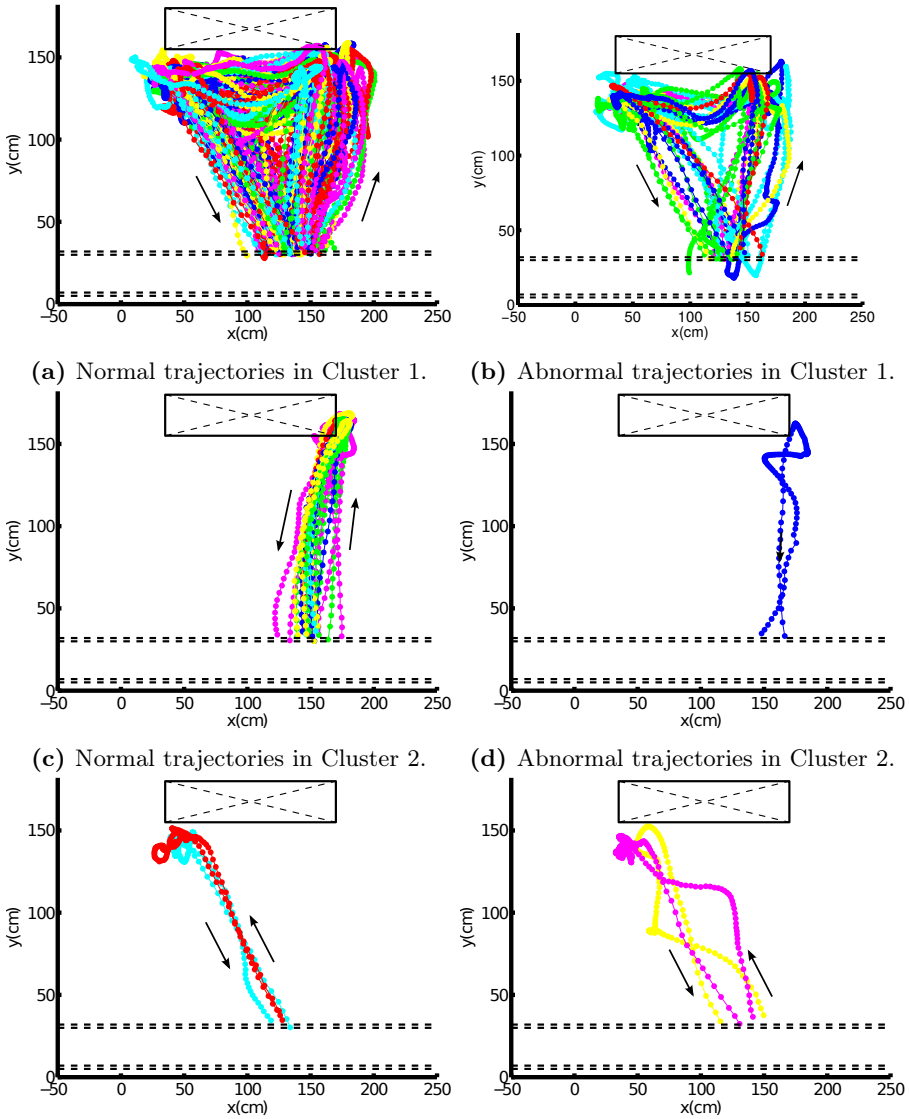
**(c)** Cluster 3 **(d)** Spatial abnormalities

**Figure 2.7:** *Trajectory clustering based spatial dissimilarity.* 124 trajectories, shown in different colors, are clustered into three work cycle clusters corresponding with 100, 13 and 4 trajectories separately, and 7 abnormalities. The main difference between these three types of executed work cycle is how factory workers traverse the area in front of the storage rack. The abnormalities includes trajectories in which factory workers follow unusual paths.

using our algorithm. Because there is only one trajectory in each of other 7 clusters, these executed work cycles are considered as abnormalities.

As shown in Fig. 2.7d, the workers follow very different paths to pick up the tools and parts in these 7 trajectories. The worker once makes a big detour on his or her way back to the operation place. This trajectory is shown as a red line. In another trajectory shown in yellow, he or she walks through the whole area in front of the storage rack, but from left to right and back to left. These unnecessary detours will leave inadequate time to do the assembly operations. Most other abnormalities are caused by the flexuous manner of walking through the area in front of the storage rack, which is different from that the worker walks directly from the right to the left corner of the storage rack in the first type of executed work cycle.

Because the conveyor belt moves at a fixed speed, it gives the workers a fixed time duration to finish their journey of picking up tools and parts. If the journey




**(a)** Normal trajectories in Cluster 1.

**(b)** Abnormal trajectories in Cluster 1.

**(c)** Normal trajectories in Cluster 2.

**(d)** Abnormal trajectories in Cluster 2.

**(e)** Normal trajectories in Cluster 3.

**(f)** Abnormal trajectories in Cluster 3.

**Figure 2.8:** *Normal and abnormal trajectories detected using temporal dissimilarity.* 18 of 100 trajectories in Cluster 1, 1 of 13 trajectories in Cluster 2, and 2 of 4 trajectories in Cluster 3 are detected as abnormalities based on temporal dissimilarity measure.

is finished too slow, the workers will miss the next assembly operation. If too fast, the workers will have to wait for the next assembly operation. In either way, the workers' trajectories are considered as temporal abnormalities. We use temporal dissimilarity between pairwise trajectories to find the temporal abnormalities for each spatial cluster in Fig. 2.7, and analyze the factors leading to the temporal abnormalities.

Fig. 2.8 shows the temporal abnormalities for each cluster. Most of these abnormalities in the first cluster are caused by the the worker dwelling in front of the storage rack in an unusual manner, such as staying too long at the left corner of the storage rack, strolling in front of the storage rack etc. He or she once walks to the right corner and continues to the left corner immediately without taking any parts. Our algorithm successfully detects these abnormalities, as shown in Fig. 2.8b. In the second cluster, the worker once makes a "S" detour at a very slow speed before he reaches the storage rack. Our algorithm successfully detects this abnormal trajectory, indicated in blue line as shown in Fig. 2.8d. Two trajectories in the third cluster are successfully detected as abnormalities because the worker does not go back to the storage rack directly, but stays in the middle of his or her way to pick up tools.

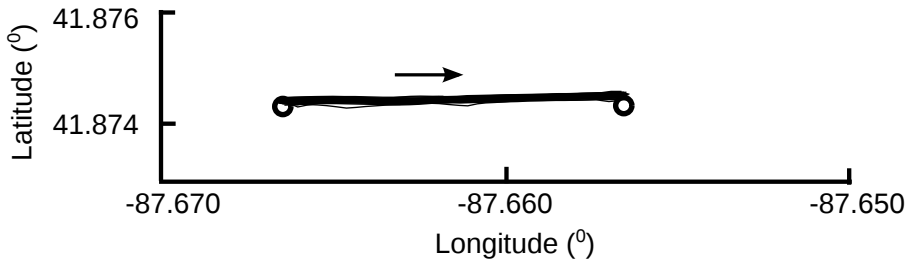
We conclude that there are mainly three factors leading to temporal abnormalities: unusual walking speed during the journey, unusual dwelling time at the left corner of the storage rack, and unnecessary dwell at other places except the two corners of the storage rack.

#### 2.5.1.2 Results of Vehicle Trajectories

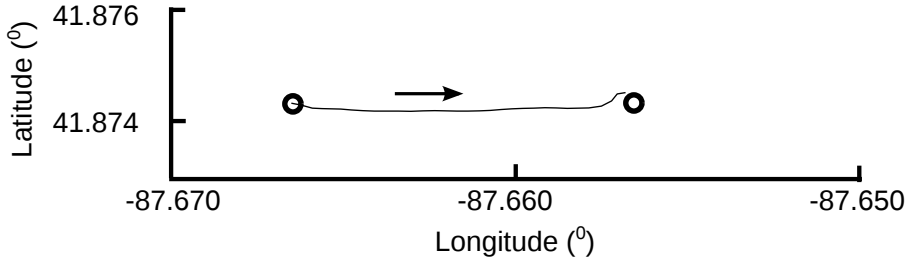
As shown in Fig. 2.6, there are 80 trajectories recorded on this road segment from the left intersection to the right intersection. One of them is detected as an spatial abnormality, as shown in Fig. 2.9b, because it is at a lower latitude than other trajectories. GPS trajectories are used to generate the road network in Chapter 6. 79 normal trajectories which are spatially similar to each other, as shown in Fig. 2.9a, will be aligned together to infer the geometric representation of the road segment. Without removing this abnormal trajectory, the inferred geometric representation will be inaccurately dragged down to a lower latitude by it.

### 2.5.2 Comparison with other similarity methods

We check the recorded videos of the Factory dataset and find three types of executed work cycles. In each of them, the workers follows a different path to pick up tools and parts. We manually group the trajectories into clusters and abnormalities, depending on the path that the workers follow. We also manually count the time that the worker spends on his or her journey to pick up tools and parts for each trajectory in each of the three clusters, and divide the trajectories into normalities and abnormalities, depending on the time the worker spends on the journey. At last, 124 trajectories are manually grouped




**(a)** Normal trajectories



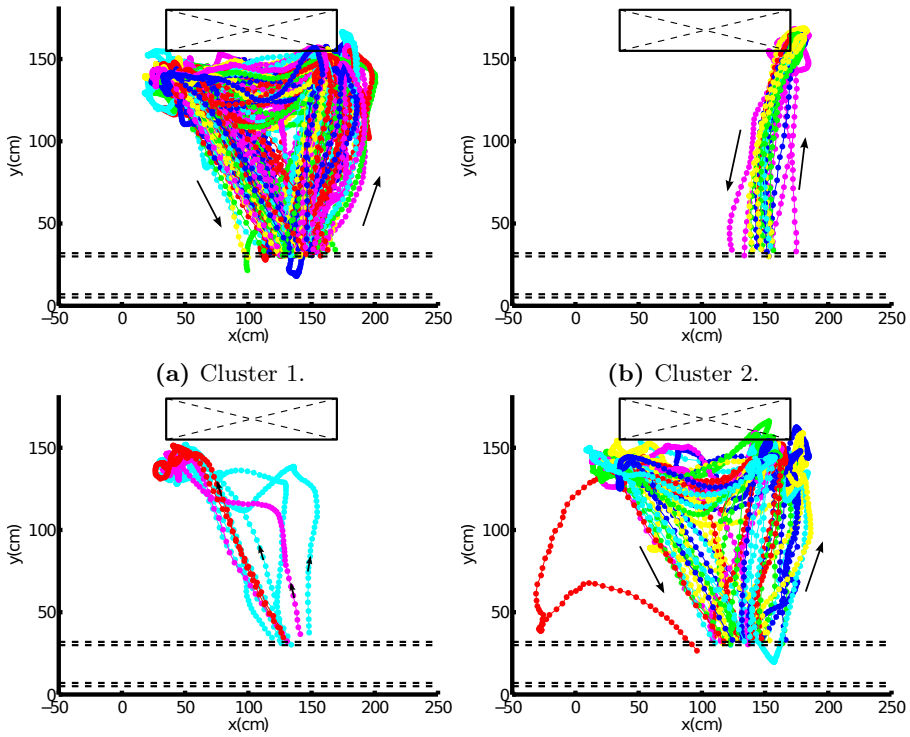

**(b)** Abnormal trajectory

**Figure 2.9:** *GPS trajectory clustering.* 1 of 80 GPS trajectories in Fig. 2.6 is detected as an abnormality based on spatial dissimilarity measurement.

into three clusters. There are 89, 12 and 2 trajectories in each cluster separately and 21 abnormal trajectories. We will use this as ground truth to test the performance of different similarity measures on trajectory clustering. The accuracy of clustering results is calculated as the ratio of the number of trajectories correctly-clustered and the total number of the trajectories.

Vlachos *et al.* defined the dissimilarity between two trajectories as the ratio of the length of the longest common subsequences and the smaller length of two trajectories. Using this measure, 124 trajectories are grouped into 4 clusters. As shown in Fig. 2.10, there are 85, 10 and 5 trajectories separately in each cluster. During the journey, the workers spend more time on staying stationery so as to pick up tools and parts from the storage rack, than walking to and back from the storage rack. Therefore, the clustering results using this dissimilarity measure are sensitive to the behavior of staying stationery. For instance, two of the cyan trajectories in cluster 3, as shown in Fig. 2.10c, actually belong to the first cluster because the workers also visit the right corner of the storage rack. However, because they spend more time staying stationery at the left corner, and come back to the assembly line in a similar route as the other two trajectories, the ratio of the number of their common points and their whole length is very high, leading to a wrong classification. For the same reason, there are as much as 24 abnormal trajectories, as shown in Fig. 2.10d.

Fig. 2.11 shows the clustering results using a trajectory dissimilarity measure base on calculating Euclidean distance between PCA coefficients of the



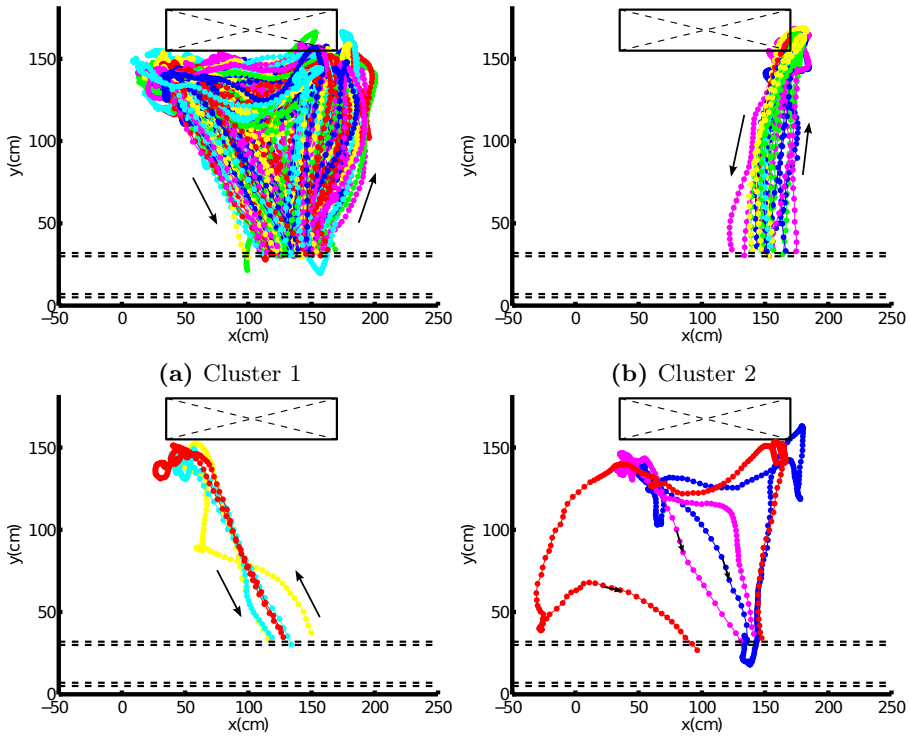

**(a)** Cluster 1. **(b)** Cluster 2.

**(c)** Cluster 3. **(d)** Abnormalities

**Figure 2.10:** *Trajectory clustering based on LCSS.* 124 trajectories, depicted using different colors, are clustered into three types of work cycle with 85, 10 and 5 trajectories separately, and 24 abnormalities.

trajectories. The trajectories are successfully grouped into three work cycles. Only three abnormalities are detected, as shown in Fig. 2.11d. But there are 21 abnormalities in our ground truth. This measure is not able to distinguish the temporal abnormalities, such as dwelling unnecessarily on his or her way to the storage rack, strolling in front of the storage rack, etc.

As shown in Table 2.1, the dissimilarity measure based on PCA + Euclidean distance shows the lowest clustering accuracy, compared to other methods. The dissimilarity measure based on LCSS works slightly better than on LCSS, but not as good as our proposed spatial dissimilarity measure based on physical location alignment, and temporal dissimilarity measure based on velocity alignment.




**(a)** Cluster 1 **(b)** Cluster 2

**(c)** Cluster 3 **(d)** Abnormalities

**Figure 2.11:** *Trajectory clustering based on PCA + Euclidean distance.* 124 trajectories are clustered into three types of work cycle with 105, 13 and 3 trajectories separately, and 3 abnormalities.

## 2.6 Conclusions

In this chapter, we presented new methods for measuring the spatial and temporal dissimilarity of pairwise trajectories using different forms of time alignment. Spatial dissimilarity is calculated using alignment based on Euclidean distance between the physical locations of the points. Temporal dissimilarity is measured using the alignment based on velocity difference. We also presented a greedy approach to efficiently cluster a set of trajectories using these dissimilarity measures.

We evaluated our proposed methods using a data set of 124 factory workers' trajectories. The experimental results showed a higher accuracy of trajectory clustering based on our dissimilarity measures than for other methods.

Although, the trajectory with the same dissimilarity measure to two clusters is always grouped into the first cluster found. Future work could on focus on discovering this kind of trajectories and assigning them with considering all of the clusters.

| | Accuracy |
|---|---|
| LCSS | 0.83 |
| PCA+Euclidean distance | 0.84 |
| The proposed measure | 0.94 |

**Table 2.1:** *Clustering results using different dissimilarity measures.* Our proposed spatial and temporal dissimilarity measure outperforms other dissimilarity measures in trajectory clustering.

This research resulted in two papers in the proceedings of International Conference on Distributed Cameras [Xie 14b] and [Xie 15a].

# 3

# Joint Alignment of Many Trajectories

Chapter 2 treated the problem of aligning trajectories two at a time. This allowed defining a spatial and temporal similarity measure between trajectories which could be used to cluster the trajectories. This approach has the following limitations:

1. Pairwise trajectory alignment is not able to solve the problem of averaging the psychical locations of *many* trajectories. In our application of the road network inference, we need to average the GPS trajectories with different number of data points to obtain the road's geometric representation. In the other application of work cycle optimization, the factory workers' trajectories need to be averaged to extract the prototypical routes, so as to improve their work efficiency. When computing the spatial similarity measure in Chapter 2, every two trajectories are aligned temporally by finding the optimal match between the physical locations along these two trajectories. The optimal matches differ for different pairs of trajectories. Therefore, one trajectory will be warped to new trajectories with different number of data points when it is paired and aligned with different other trajectories. Therefore, the newly warped trajectories can not be averaged directly.

2. Pairwise trajectory alignment is not very suitable for statistical analysis, e.g., analyzing speed variance at a specific location, or timing variances of objects arriving at a specific location. In Chapter 2, only pairwise trajectory alignments are produced, whereas statistical analysis requires joint associations between all trajectories. Some researchers may argue that one trajectory can be chosen as a center trajectory, and align it with every other trajectory. Statistical analysis of trajectories can be done using the data points on in this center trajectory and their associated points on other trajectories. However, DTW creates *many-to-one* correspondences during the alignment. Each point in this center trajectory may be associated with several points on other trajectories. Pairwise

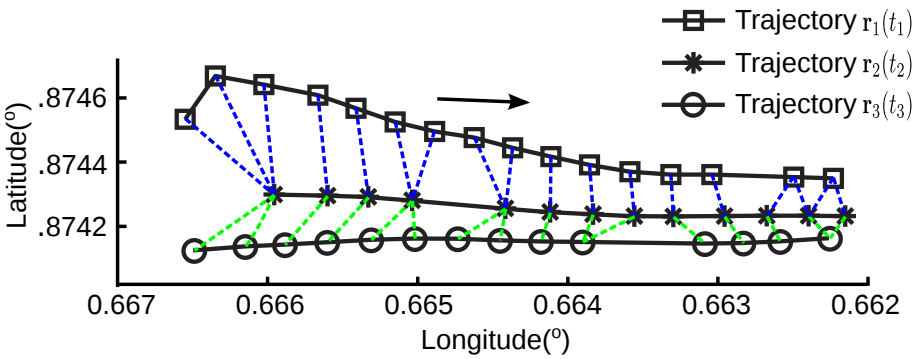



**Figure 3.1:** *Pairwise trajectory alignment.* Trajectory $\mathbf{r}_1(t_1)$, $t_1 = 1, \ldots T_1$ is aligned with trajectory $\mathbf{r}_2(t_2)$, $t_2 = 1, \ldots T_2$. Trajectory $\mathbf{r}_2(t_2)$, $t_2 = 1, \ldots T_2$ is aligned with trajectory $\mathbf{r}_3(t_3)$, $t_3 = 1, \ldots T_3$. The coordinates are relative to (-87,41). The users move from the lower-longitude area to higher-longitude area in all of these three trajectories.

trajectory alignment gives no knowledge about how to select the associated points from all trajectories. However, aligning *many* trajectories together produces warped trajectories with associated points at the same time index. Another drawback of this argument is that the statistical results rely very much on the choice of this center trajectory. Therefore, pairwise trajectory alignment is useful for similarity measure, but it is not capable of averaging *many* trajectories, neither analyzing *many* trajectories statistically. In order to solve these problems, we propose the idea of aligning more than two trajectories.

In this chapter, we aim to align *many* trajectories by building their point associations and warp all of the trajectories into new trajectories of the same length using the point associations. The data points at the same position of the newly warped trajectories are associated to each other, i.e. spatially close to each other. The associated points can be averaged easily to create geometric representation for road network inference and prototypical routes for work cycle optimization. The location variance and speed variance at a specific location can be analyzed directly using the associated points there.

In literature, there have been some effort on averaging *many* trajectories based on pairwise trajectory alignment using DTW [Junejo 07, Junejo 08, Niennattrakul 09, Petitjean 11]. In their work, they have been concentrating on averaging *many* trajectories using some strategies based on DTW, such as averaging the two boundaries of all trajectories, averaging two trajectories with the largest similarity hierarchically until only one trajectory is left, etc. However, they have not established the point correspondences among all trajectories. Therefore, these strategies are only useful to trajectory averaging, but not to statical analysis of the trajectories.

A naive DTW-based method to align more than two trajectories starts by first sorting trajectories in some well defined order, according to a suitable chosen criterion, e.g. an order depending on the similarity of the trajectories, or in random order. Then trajectory $\mathbf{r}_1(t_1)$, $t_1 = 1, \dots T_1$ is aligned with trajectory $\mathbf{r}_2(t_2)$, $t_2 = 1, \dots T_2$ using DTW; Next trajectory $\mathbf{r}_2(t_2)$, $t_2 = 1, \dots T_2$ is aligned with trajectory $\mathbf{r}_3(t_3)$, $t_3 = 1, \dots T_3$ and so on. However, the trajectories will get longer and longer when aligning more and more trajectories, which increases the computational cost, especially for the big data set. Using this naive method, every two adjacent trajectories are aligned, for instance, $\mathbf{r}_1(t_1)$, $t_1 = 1, \dots T_1$ with $\mathbf{r}_2(t_2)$, $t_2 = 1, \dots T_2$, creating point associations between them. However, point correspondences between *non-adjacent* trajectories are *not* established, e.g., between trajectories $\mathbf{r}_1(t_1)$, $t_1 = 1, \dots T_1$ and $\mathbf{r}_3(t_3)$, $t_3 = 1, \dots T_3$. Without point associations between all of the trajectories, it is difficult to analyze them statistically.

Fig. 3.1 shows an example of three GPS trajectories aligned using pairwise trajectory alignment. Trajectory $\mathbf{r}_1(t_1)$, $t_1 = 1, \dots T_1$ is aligned with trajectory $\mathbf{r}_2(t_2)$, $t_2 = 1, \dots T_2$, producing point associated indicated in blue lines. The green lines are for the point associations between trajectories $\mathbf{r}_2(t_2)$, $t_2 = 1, \dots T_2$, and $\mathbf{r}_3(t_3)$, $t_3 = 1, \dots T_3$. Because of the *many-to-one* point correspondences built by DTW, the first three points of the first trajectory, $\mathbf{r}_1(t_1)$, $t_1 = 1, 2, 3$, are associated to the first two points of the third trajectory, $\mathbf{r}_3(1)$ and $\mathbf{r}_3(2)$, indirectly, through the first point of the second trajectory, $\mathbf{r}_2(1)$. This three-to-two correspondence complicates the statistical analysis which needs collect one associated point from each trajectory, and analyze the speed and location variance of these associated points, because there is no knowledge about how to choose the associated point in the first trajectory from its first three points, and the associated point in the third trajectory from its first two points, given the first point of the second trajectory.

In this chapter we propose two new methods to extend the DTW for establishing the point correspondences among *many* trajectories, instead of pairwise trajectories. In our first method, we aim to find *one-to-one* correspondences among the physical locations along the trajectories. We apply a "stretch and then compress" strategy based on the DTW algorithm to align the trajectories one by one in ascending order of average dissimilarity.

Firstly, the first two trajectories are aligned using DTW in the *stretch* operation, producing two stretched trajectories. The *compress* operation then removes repeated points from the stretched trajectories, producing two compressed trajectories with *one-to-one* correspondences between the points. Establishing the *one-to-one* correspondences is beneficial to easily locate corresponding points in the nonadjacent trajectories. The third trajectory is then aligned with the second compressed trajectory using the "stretch and then compress" strategy again. The procedure is repeated until the last trajectory is processed.

Secondly, the points on other trajectories associated to the point on the last compressed trajectory are located through the *one-to-one* correspondences

between pairwise adjacent trajectories, creating warped trajectories of the same number of points.

In the second proposed method, all of the trajectories are aligned simultaneously along the time dimension, instead of in a pairwise fashion. Instead of using the "backtrack" procedure through the overall dissimilarity tensor to find the optimal warp path as DTW does, we propose a greedy procedure to locates a good warp path, by jointly traversing all trajectories in a way which keeps the points associated by the path element spatially close to each other. This involves an iterative procedure to cluster successor points on the trajectories. This method does not use DTW, but rather relies on the statistics of *many* trajectories to compute a faithful association. This procedure produces *many-to-one* point correspondences among the trajectories.

Experimental results indicate that we successfully establish the point correspondences among all of the trajectories using both proposed methods, producing warped trajectories with associated points at the same time index. The average trajectory calculated based on the point correspondences is more representative than using other algorithms in literature.

The chapter is structured as follows: In Section 3.3, we introduce the first algorithm to align *many* trajectories one by one using a "stretch and then compress" strategy. In Section 3.4, we explain the trajectory alignment using the greedy method based on successor classification. We show the experimental results in Section 3.7. Finally, we conclude this chapter in Section 3.8.

## 3.1 Related Work

Chapter 2 concentrates on point association between pairwise trajectories for similarity measure. In this chapter, we establish the point correspondences among all trajectories, so as to provide a tool for trajectory averaging and statistical analysis along the averaged trajectory.

In literature, some researchers have proposed some strategies to average the trajectories directly using pairwise trajectory alignment [Junejo 07, Junejo 08, Niennattrakul 09, Petitjean 11]. Without associating the points among all trajectories, their strategies can not be used to analyze the trajectories statistically.

Junejo and Foroosh simply align the boundaries of pedestrians' trajectories using DTW, and take the mid-points of the lines joining the matched corresponding points in the warped trajectories as the common path [Junejo 07, Junejo 08]. A boundary trajectory is a sequence of points along one edge of the pedestrian road. This method is very efficient and the computational cost is very low since only two boundary trajectories are warped and averaged. However, the common path is irrelevant to the spatial distribution of the points of all trajectories [Sester 12], because the points of the boundary trajectory are selected only from the trajectories at the edge of the road surface, and the trajectories in the middle of the road surface are ignored. Therefore the common path does not show the preference of the pedestrians on the road, for instance, which side of the road the pedestrians walk on more frequently. This common

path is also not very suitable as the prototypical route for the workers in our application of work cycle optimization. Simply averaging two boundary trajectories but ignoring the trajectories between these two boundary trajectories leads to the prototypical route not representative. Besides, the common path does not give any information about the dwell time of the factory workers when they pick up parts and tools from the storage rack, because the points of the boundary trajectories are unrepeated.

Niennattrakul and Ratanamahatana propose a bottom-up hierarchical clustering approach called Prioritized Shape Averaging (PSA) based on DTW [Niennattrakul 09]. This method first computes the similarity between all pairs of trajectories. In the first phase of the algorithm, the two most similar trajectories are averaged after warping and then removed from the set of available trajectories, whereas the newly created average trajectory is added to this set. In the second phase, from this new set the two most similar trajectories are selected and averaged once more. This procedure is repeated until a single "overall average" trajectory is left in the set. At each phase of the algorithm, a weight is assigned to each of the two most similar trajectories, depending on whether it is an original trajectory (weight: 1), or an averaged trajectory from the previous phase (weight: the number of trajectories that the averaged trajectory is formed from). This ensures that each original trajectory play equally in creating the "overall average" trajectory. This method is slow due to the need of computing a similarity measure between pairwise trajectories at each phase of the algorithm.

Petitjean *et al.* propose a DTW-based global averaging method, called DTW Barycenter Averaging (DBA) [Petitjean 11]. The DBA algorithm starts by picking one trajectory and using it as the initial estimate of the prototypical trajectory. The time association between each individual trajectory and the prototypical trajectory is found using DTW. The prototypical trajectory is refined based on the computed time association. Each point of the prototypical trajectory is then replaced by the mean of the points associated to it on all of the trajectories. The procedure is repeated until the association between each individual trajectory and the prototypical trajectory no longer changes. This method takes all of the trajectories into account jointly and is efficient, but it is sensitive to the initialization of the prototypical trajectory. Specifically, this initialization fixes the length of the final prototypical trajectory, which cannot change during the algorithm. As a result this length may not be optimal.

The above mentioned DTW-related methods aim to average multiple trajectories using different strategies and do not focus on establishing time correspondences between the "average" trajectory and each individual trajectory, neither on building point associations among all trajectories. Obtaining these correspondences would require additional DTW alignment computations. Instead of building the point correspondences after averaging *many* trajectories, in our methods we first establish the point correspondences during the procedure of aligning *many* trajectories, and then utilize them to average the original trajectories and statistically analyze the variance along the averaged trajectory.

It is time-consuming to align trajectories in pairs for very big data set repetitively using DTW procedure. In Petitjean's work, DTW is used many times in the Barycenter Averaging: during each iteration, the association of each individual trajectory to the current average trajectory is computed using DTW. Therefore, the DTW procedure is repeated $N \times I$ times, given $N$ trajectories and $I$ iterations. In Niennattrakul's work, the two trajectories with the biggest similarity are aligned using DTW at each of $N-1$ phases of the algorithm. In order to find the two trajectories most similar to each other, DTW is used to calculate the distance between each pair of trajectories as the description of similarity. In total, $\sum_{n=2}^{N-1} n(n+1)/2$ DTW computations are needed.

In our first method, $N$ trajectories are aligned one by one in a specific order, so pairwise trajectories are aligned using DTW $N-1$ times. With this reduced computational complexity, our method is much faster than the methods mentioned above.

Our second method does not use DTW, but a greedy procedure to locate a good warp path through the local dissimilarity tensor, in a way which keeps the points associated by each element of the warp path spatially close to each other. This involves an iterative procedure to cluster successor points on the trajectories. Although it takes time to cluster the successors, avoiding the back-tracking procedure for optimal alignment reduces the computational cost enormously. Therefore, our method is faster than the methods mentioned above, including our first proposed method.

In Section 3.7, we will show the results of our proposed methods. We successfully establish the point correspondences among all trajectories, and produce warped trajectories of the same length. The points of each warped trajectory appear in the same relative order within the original trajectory. The points at the same time index of the warped trajectories are spatially close to each other. The associations among the points of all trajectories can be used to extract the average trajectory, and analyze the trajectories along the average trajectory statistically. Since no such correspondences are established in other DTW-related methods mentioned above, we cannot compare our method with other methods on this point. These methods aim to compute average trajectories, and we will compare our methods with them on this point.

## 3.2 Problem Statement

In this chapter, we aim to align many trajectories through establishing the associations among the physical locations on all the trajectories. The point associations will be used to create an average trajectory and analyze the location and speed variance along the average trajectory in our applications.

DTW is a good method to optimally align two trajectories by minimizing the physical distance between corresponding points on the trajectories. During the implementation phase, a two-dimensional local dissimilarity matrix is calculated, and then a two-dimensional overall dissimilarity matrix is calculated based on the local dissimilarity matrix. As elaborated in Section 2.2, the

optimal warp path through the overall dissimilarity matrix, which represents the optimal alignment between two trajectories, is found by backtracking the minimal overall dissimilarity.

Theoretically DTW can be extended to find the globally optimal alignment of *many* trajectories directly by minimizing an overall distance criterion which measures the overall similarity between more than two associated locations. Given $N$ trajectories $\mathbf{r}_n(t_n)$, $t_n = 1, \dots T_n$, $n = 1, \dots N$, a N-dimensional local dissimilarity tensor $C_1$ and a N-dimensional overall dissimilarity tensor $C_2$ need to be calculated, where the subscript 1 and 2 refers to *local* dissimilarity and *overall* dissimilarity, respectively. Once $C_2$ is calculated, the optimal warp path through tensor $C_2$ can be found by backtracking the minimal overall dissimilarity from the last cell $(T_1, \dots T_n)$ to the first cell $(1, \dots 1)$. However it is prohibitively computationally expensive to calculate the overall dissimilarity tensor $C_2$ and to backtrack the overall dissimilarity through tensor $C_2$.

Both local dissimilarity tensor $C_1$ and overall dissimilarity tensor $C_2$ contains $\prod_{n=1}^{N} T_n$ elements. The value at each element of $C_2$ needs to be calculated recursively using the local dissimilarity at this cell and the values at its $2^N - 1$ adjacent cells, as shown in Equation 3.1.

$$
\begin{aligned}
C_1(t_1, \dots t_N) &\triangleq \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\mathbf{r}_n(t_n) - \frac{1}{N}\sum_{n=1}^{N}\mathbf{r}_n(t_n))^2}, \\
C_2(t_1, \dots t_N) &\triangleq C_1(t_1, \dots t_N) + \min_{j_n \in [0,1] \sum_{n=1}^{N} j_n \neq 0} C_2(t_1 - j_1, \dots t_N - j_N),
\end{aligned}
\tag{3.1}
$$

where $C_1(t_1, \dots t_N)$ is the local dissimilarity at cell $(t_1, \dots t_N)$, which is measured using the standard deviation of its corresponding points $\mathbf{r}_n(t_n), n = 1, \dots N$.

The computational cost of calculating $C_2$ is strongly dependent on the trajectory lengths $T_n$, $n = 1, \dots N$, and also increases exponentially with increasing $N$, which is the number of the trajectories. For instance, the value at each element of $C_2$ will be calculated using the values of its $2^{20} - 1 = 1048575$ adjacent cells, suppose that $N = 20$ trajectories need to be aligned.

During the procedure of backtracking the minimum overall dissimilarity through $C_2$, assume the current element of the warp path is cell $(t_1, \dots t_N)$ of $C_2$, one of its $2^N - 1$ adjacent cells, $(t_1 - j_1, \dots t_N - j_N), j_n \in [0, 1], \sum_{n=1}^{N} j_n \neq 0$, whose value is the smallest, will be selected as the next element of the warp path. The computational cost of identifying the optimal warp path through $C_2$ also highly dependent on the trajectory lengths $T_n$, $n = 1, \dots N$ and the number of the trajectories.

Therefore it is impractical to find the globally optimal alignment of *many* trajectories using DTW. Rather than the optimal alignment, we aim to find a good alignment in this chapter, which has the following properties:

1. It produces warped trajectories with the same number of points.

2. The points at the same index of the warped trajectories are associated to each other, i.e. spatially close to each other.

3. The points of each warped trajectory appear in the same relative order in the original trajectory.

We propose two methods to identify such good alignments. In the first method, we utilize pairwise alignments with a "stretch and then compress" strategy to establish the *one-to-one* point correspondences among all trajectories, as described in Section 3.3. Instead of pairwise alignment, all of the trajectories are aligned at once in the second method, as elaborated in Section 3.4. We build the *many-to-one* correspondences among the points of all trajectories by forward-tracking the relative minimum local dissimilarity.

## 3.3 Stretch and Compress Trajectory Alignment

For a data set of many trajectories, the optimal matches are diverse for different pairs of trajectories, producing different warped trajectories with different lengths. This can not be directly used to obtain an average trajectory, neither analyze all trajectories statistically. We propose a "stretch and then compress" strategy to establish the *one-to-one* correspondences among the points of all trajectories through pairwise trajectory alignments, and create warped trajectories of the same length. DTW is used to perform pairwise alignment of two trajectories.

### 3.3.1 "Stretch and then compress" Strategy

This strategy is composed of two different operations: a *stretch* operation followed by a *compress* operation as described below.

**1) Stretch.** Trajectories $\mathbf{r}_1(t_1)$, $t_1 = 1, \dots T_1$ and $\mathbf{r}_2(t_2)$, $t_2 = 1, \dots T_2$ are aligned using DTW, creating new trajectories $\mathbf{r}_1^{(s)}(j_{1,2})$ and $\mathbf{r}_2^{(s)}(j_{1,2})$, $j_{1,2} = 1, \dots J_{1,2}$, where the superscript $(s)$ means "once-stretched" and $j_{1,2}$ as the joint index in the two warped trajectories. The original time indexes of the points in the two warped trajectories are denoted $w_1^{(s)}(j_{1,2})$ and $w_2^{(s)}(j_{1,2})$, $j_{1,2} = 1, \dots J_{1,2}$. In other words: $\mathbf{r}_1^{(s)}(j_{1,2}) = \mathbf{r}_1(w_1^{(s)}(j_{1,2}))$ and $\mathbf{r}_2^{(s)}(j_{1,2}) = \mathbf{r}_2(w_2^{(s)}(j_{1,2}))$, for all indexes $j_{1,2} = 1, \dots J_{1,2}$. This Stretch operation involves a DTW operation, to align two trajectories using their physical locations as shown in Section 2.3.1. The new trajectories can contain more data points than either of the original trajectories, i.e., $J_{1,2} \geqslant T_1$ and $J_{1,2} \geqslant T_2$. Therefore this operation is called *stretch*, even though the stretching is actually an undesired side effect: the true goal is to align the trajectories. In any case, the new trajectories are called *stretched* trajectories, to differentiate them from the trajectories resulting from the next operation *compress*. It is possible that

one point on one trajectory is matched with multiple points on the other trajectory. These *many-to-one* correspondences create difficulties in associating the data points between nonadjacent trajectories.

**2) Compress.** The *many-to-one* correspondences can be simplified by keeping only one of the *many-to-one* correspondences per pair. Specifically we keep only the correspondence pair with the shortest distance between points. In this way, the stretched trajectories $\mathbf{r}_1^{(s)}(j_{1,2})$ and $\mathbf{r}_2^{(s)}(j_{1,2})$, $j_{1,2} = 1, \dots J_{1,2}$ with $J_{1,2}$ points are shortened to trajectories $\mathbf{r}_1^{(c)}(k_{1,2})$ and $\mathbf{r}_2^{(c)}(k_{1,2})$, $k_{1,2} = 1, \dots K_{1,2}$ with $K_{1,2}$ points, where the superscript $(c)$ means "once-compressed." Similarly, $w_1^{(c)}$ and $w_2^{(c)}$ denote the original time indexes of the points of the new, compressed trajectories, i.e. $\mathbf{r}_1^{(c)}(k_{1,2}) = \mathbf{r}_1(w_1^{(c)}(k_{1,2}))$ and $\mathbf{r}_2^{(c)}(k_{1,2}) = \mathbf{r}_2(w_2^{(c)}(k_{1,2}))$, for all indexes $k_{1,2} = 1, \dots K_{1,2}$. Because the new trajectories can contain fewer points than either of the original trajectories, i.e., $K_{1,2} \leqslant T_1$ and $K_{1,2} \leqslant T_2$, this operation is called *compress* and the new trajectories are called *compressed* trajectories. One point in one of the compressed trajectory $\mathbf{r}_1^{(c)}(k_{1,2})$ now corresponds to exactly one point in the other compressed trajectory, and this point $\mathbf{r}_2^{(c)}(k_{1,2})$ has the same new time index $k_{1,2}$, which is called *one-to-one* correspondence. With the *one-to-one* correspondences between points of pairwise compressed trajectories, the point associations between nonadjacent trajectories can be easily established.

## 3.3.2 Details of the Alignment Procedure

We align *many* trajectories in an ascending order of average dissimilarity. The average dissimilarity of each trajectory is calculated as the mean of the dissimilarities between this trajectory and every other trajectory. We first apply the "stretch and then compress" strategy to process all the trajectories pairwise from the first two trajectories to the last two trajectories sequentially. **Phase 1** in Fig. 3.2 illustrates the procedure presented below:

### 3.3.2.1 Phase 1: Iterative Stretch and Compress

The "stretch and then compress" operation is applied to trajectory $\mathbf{r}_1(t_1)$, $t_1 = 1, \dots T_1$ and trajectory $\mathbf{r}_2(t_2)$, $t_2 = 1, \dots T_2$. This results in the once-compressed trajectories $\mathbf{r}_1^{(c)}(k_{1,2})$ and $\mathbf{r}_2^{(c)}(k_{1,2})$, $k_{1,2} = 1, \dots K_{1,2}$ and the associated indexes $w_1^{(c)}(k_{1,2})$ and $w_2^{(c)}(k_{1,2})$, $k_{1,2} = 1, \dots K_{1,2}$, where the superscript $(c)$ means once-compressed here.

The once-compressed trajectory $\mathbf{r}_2^{(c)}(k_{1,2})$, $k_{1,2} = 1, \dots K_{1,2}$ and a new input trajectory $\mathbf{r}_3(t_3)$, $t_3 = 1, \dots T_3$ are input to the "stretch and then compress" module, resulting in a twice-compressed trajectory $\mathbf{r}_2^{(2c)}(k_{2,3})$ and a once-compressed trajectory $\mathbf{r}_3^{(c)}(k_{2,3})$, $k_{2,3} = 1, \dots K_{2,3}$, where the superscript $(2c)$ means twice-compressed. The associated indexes are $w_2^{(2c)}(k_{2,3})$

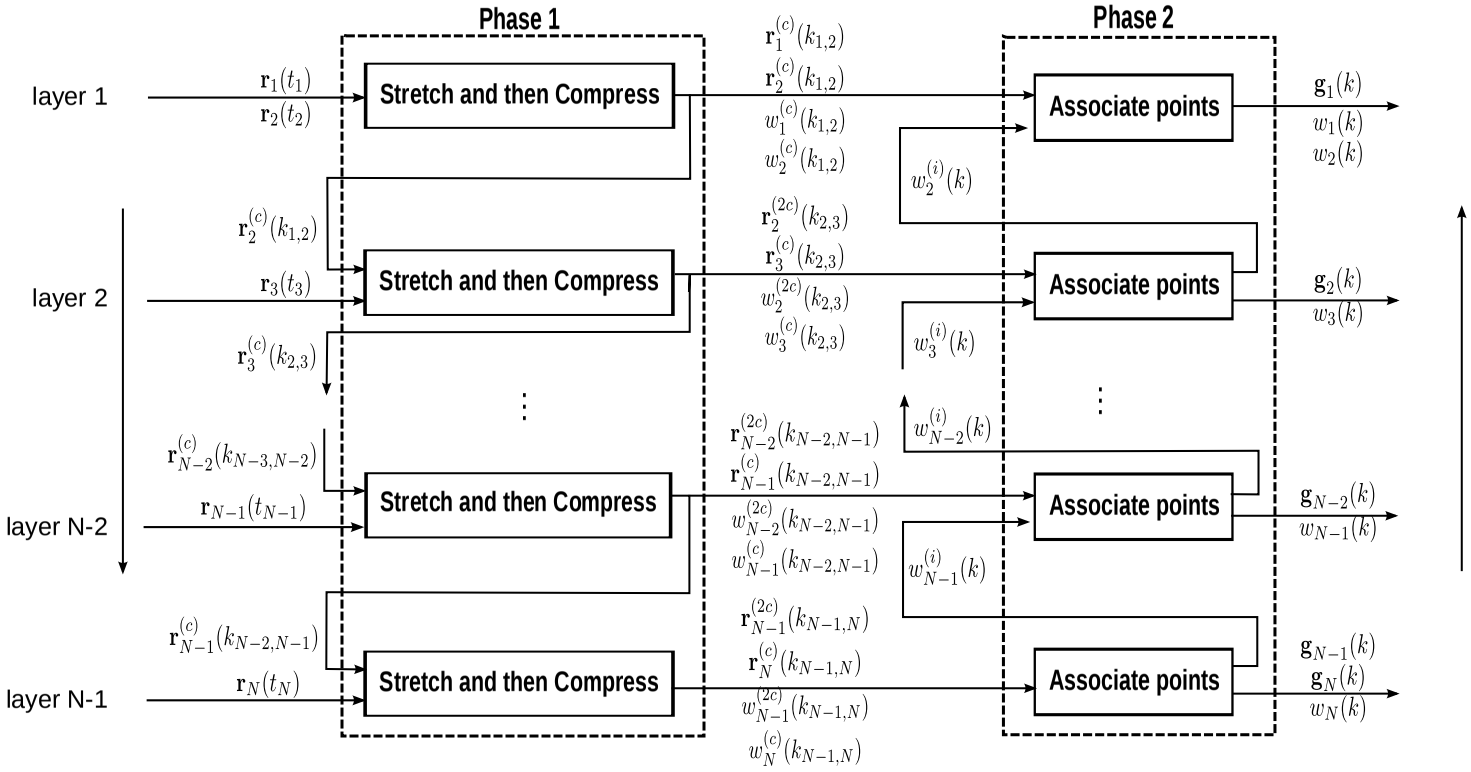



**Figure 3.2:** *Aligning many trajectories.* In **Phase** 1, the trajectories are aligned pairwise using a "stretch and then compress" strategy in an ascending order of average dissimilarity. **Phase** 2 produces warped trajectories $\mathbf{g}_n(k)$, $k = 1, \ldots K$, $n = 1, \ldots N$ with the same number of points and with all points associated to a single time index $k$. The functions $w_n(k)$ indicates the original time index of the point in trajectory $n$ corresponding to time index $k$.

of the points of the twice-compressed trajectory $\mathbf{r}_2^{(2c)}(k_{2,3})$, $k_{2,3} = 1, \dots K_{2,3}$ in the once-compressed trajectory $\mathbf{r}_2^{(c)}(k_{1,2})$, $k_{1,2} = 1, \dots K_{1,2}$, and $w_3^{(c)}(k_{2,3})$ for time indexes of the points of the once-compressed trajectory $\mathbf{r}_3^{(c)}(k_{2,3})$, $k_{2,3} = 1, \dots K_{2,3}$ in the original trajectory $\mathbf{r}_3(t_3)$, $t_3 = 1, \dots T_3$.

This process is repeated for $n > 2$. Each time, a once-compressed trajectory $\mathbf{r}_n^{(c)}(k_{n-1,n})$, $k_{n-1,n} = 1, \dots K_{n-1,n}$ and a new trajectory $\mathbf{r}_{n+1}(t_{n+1})$, $t_{n+1} = 1, \dots T_{n+1}$ are input to the "stretch and then compress" module, until the last trajectory $\mathbf{r}_N(t_N)$, $t_N = 1, \dots T_N$, is processed.

The outputs of **Phase 1** are:

- the once-compressed trajectories $\mathbf{r}_n^{(c)}(k_{n-1,n})$, $n = 2, \dots N$, with $K_{n-1,n}$ points, and $\mathbf{r}_1^{(c)}(k_{1,2})$ with $K_{1,2}$ points;
- the twice-compressed trajectories $\mathbf{r}_n^{(2c)}(k_{n,n+1})$, $n = 2, \dots N-1$, with $K_{n,n+1}$ points;
- the time indexes $w_1^{(c)}(k_{1,2})$ and $w_n^{(c)}(k_{n-1,n})$, $n = 2, \dots N$, of the points of the once-compressed trajectories in the original trajectories, i.e. $\mathbf{r}_n^{(c)}(k_{n-1,n}) = \mathbf{r}_n(w_n^{(c)}(k_{n-1,n}))$;
- the time indexes $w_n^{(2c)}(k_{n,n+1})$, $n = 2, \dots N-1$, of the points of the twice-compressed trajectories in the once-compressed trajectories: these satisfy $\mathbf{r}_n^{(2c)}(k_{n,n+1}) = \mathbf{r}_n^{(c)}(w_n^{(2c)}(k_{n,n+1}))$.

Each of the "stretch and then compress" blocks outputs two trajectories with the same number of points: a twice compressed trajectory $\mathbf{r}_n^{(2c)}(k_{n,n+1})$ and a once-compressed trajectory $\mathbf{r}_{n+1}^{(c)}(k_{n,n+1})$, $k_{n,n+1} = 1, \dots K_{n,n+1}$. Each point of one output trajectory corresponds to one single point of the other output trajectory, and both points have the same time index $k_{n,n+1}$. Because of the *compress* operation, the output trajectories of the module are shorter than either of the input trajectories, i.e. $K_{n,n+1} \leqslant K_{n-1,n}$ and $K_{n,n+1} \leqslant T_{n+1}$, where $n = 2 \dots N-1$.

The main advantage of the compression steps in **Phase 1**, is that they ensure there is only one point in every other trajectory, which is associated to each point in the last once-compressed trajectory $\mathbf{r}_N^{(c)}(k_{N-1,N})$, $k_{N-1,N} = 1, \dots K_{N-1,N}$. Without the compression step, there would be *many-to-one* point correspondences instead between the adjacent trajectories, and *many-to-many* correspondences between the nonadjacent trajectories. It is difficult to utilize them to obtain an alignment of all trajectories, which produces warped trajectories of the same length, and associates the points at the same position of the warped trajectories. Also the stretched trajectories will get longer and longer, leading to an increased computational cost.

**Algorithm 3** AssociatePoints

**Input:** $w_1^{(c)}(k_{1,2})$, $k_{1,2} = 1, \dots K_{1,2}$
$w_n^{(c)}(k_{n-1,n})$, $k_{n-1,n} = 1, \dots K_{n-1,n}$, $n = 2, \dots N$
$w_n^{(2c)}(k_{n,n+1})$, $k_{n,n+1} = 1, \dots K_{n,n+1}$, $n = 2, \dots N-1$
**Output:** $w_n(k)$, $k = 1, \dots K_{N-1,N}$, $n = 1, \dots N$

$w_N(k) \leftarrow w_N^{(c)}(k)$, $k = 1, \dots K_{N-1,N}$
$w_{N-1}^{(i)}(k) \leftarrow w_{N-1}^{(2c)}(k)$, $k = 1, \dots K_{N-1,N}$
$w_{N-1}(k) \leftarrow w_{N-1}^{(c)}(w_{N-1}^{(i)}(k))$, $k = 1, \dots K_{N-1,N}$
**for** $n = N-2$ to 2 **do**
    $w_n^{(i)}(k) \leftarrow w_n^{(2c)}(w_{n+1}^{(i)}(k))$, $k = 1, \dots K_{N-1,N}$
    $w_n(k) \leftarrow w_n^{(c)}(w_n^{(i)}(k))$, $k = 1, \dots K_{N-1,N}$
**end for**
$w_1(k) \leftarrow w_1^{(c)}(w_2^{(i)}(k))$, $k = 1, \dots K_{N-1,N}$

#### 3.3.2.2 Phase 2: Final Alignment

**Phase 2** associates the points of all trajectories to those of the last once-compressed trajectory $\mathbf{r}_N^{(c)}(k_{N-1,N})$, $k_{N-1,N} = 1, \dots K_{N-1,N}$ with $K_{N-1,N}$ points. This not only mutually aligns all of the trajectories but also produces warped trajectories with the same number of points, and with corresponding points having the same time index. We adopt the notation $\mathbf{g}_n(k)$, $k = 1, \dots K$, for the warped trajectory, and $w_n(k)$, $k = 1, \dots K$, for the time indexes of the points of the warped trajectory on the original trajectory, where $n = 1, \dots N$.

The number of the points on output trajectories of **Phase** 1 is different. There are $K_{n-1,n}$ points in the once-compressed trajectories $\mathbf{r}_n^{(c)}(k_{n-1,n})$, $n = 2 \dots N$, and $K_{1,2}$ data points on the first once-compressed trajectory $\mathbf{r}_1^{(c)}(k_{1,2})$. There are $K_{n,n+1}$ data points on the once twice-compressed trajectories $\mathbf{r}_n^{(2c)}(k_{n,n+1})$, $n = 2 \dots N-1$. Among all of the once-compressed and twice-compressed trajectories, the last once-compressed trajectory $\mathbf{r}_N^{(c)}(k_{N-1,N})$ is the shortest one. Therefore, it is considered as the warped trajectory for the last trajectory, i.e. $\mathbf{g}_N(k) = \mathbf{r}_N^{(c)}(k)$, $w_N(k) = w_N^{(c)}(k)$, $k = 1, \dots K_{N-1,N}$. In each of other trajectories, there is a unique point associated to every point of this warped trajectory $\mathbf{g}_N(k) = \mathbf{r}_N^{(c)}(k)$. The length of all warped trajectories is determined as $K = K_{N-1,N}$. If any of other longer compressed trajectory is taken as the warped trajectory, it is possible that no associated points can be found in the shorter compressed trajectories. For each point in the last warped trajectory $\mathbf{g}_N(k)$, a unique associated point in each of other trajectories can be located using a bottom-up strategy through the associated indexes $w_n^{(c)}(k_{n,n+1})$, $n = 1, \dots N-1$, and $w_n^{(2c)}(k_{n,n+1})$, $n = 2, \dots N-1$.

To clarify the procedure of **Phase 2**, which computes the final warping, we introduce the notation $w_n^{(i)}(k)$, $k = 1, \dots K_{N-1,N}$, for the time indexes of data points of the warped trajectory in the once-compressed trajectory, where $(i)$

means *intermediate* time indexes, and $n = 1, \dots N-1$.

As shown in Fig. 3.2, **Phase 2** starts from the bottom block. The points of trajectory $\mathbf{r}_{N-1}(t_{N-1})$, $t_{N-1} = 1, \dots T_{N-1}$, which are associated to the points of $\mathbf{g}_N(k)$, $k = 1, \dots K_{N-1,N}$, are computed by pairwise alignment. This corresponds to the bottom block of **Phase 1**. Both of output trajectories of the "stretch and then compress" module at the bottom block, i.e., $\mathbf{r}^{(2c)}_{N-1}(k_{N-1,N})$ and $\mathbf{r}^{(c)}_{N}(k_{N-1,N})$, $k_{N-1,N} = 1, \dots K_{N-1,N}$, contain the same number $K_{N-1,N}$ of points, and are already associated with the same time index. Since one of the output trajectory $\mathbf{r}^{(c)}_{N}(k_{N-1,N})$, $k_{N-1,N} = 1, \dots K_{N-1,N}$ is the warped trajectory for the last trajectory, $\mathbf{g}_N(k)$, $k = 1, \dots K_{N-1,N}$, the other output trajectory $\mathbf{r}^{(2c)}_{N-1}(k_{N-1,N})$, $k_{N-1,N} = 1, \dots K_{N-1,N}$ is the warped trajectory for the second last trajectory, $\mathbf{g}_{N-1}(k)$, $k = 1, \dots K_{N-1,N}$. $w^{(2c)}_{N-1}(k)$ indicates the time indexes of the points of the second last warped trajectory $\mathbf{g}_{N-1}(k)$, $k = 1, \dots K_{N-1,N}$ in the second last once-compressed trajectory $\mathbf{r}^{(c)}_{N-1}(k_{N-2,N-1})$, $k_{N-2,N} = 1, \dots K_{N-2,N-1}$, which is one of the input trajectories to the bottom block of **Phase 1**.

The "associate points" module at the bottom block of **Phase 2** outputs two warped trajectories $\mathbf{g}_{N-1}(k)$ and $\mathbf{g}_N(k)$, the time indexes of the points of the last warped trajectory in the last original trajectory, $w_N(k)$, the time indexes of the points of the second last warped trajectory in the second last once-compressed trajectory, $w^{(i)}_{N-1}(k) = w^{(2c)}_{N-1}(k)$. They satisfy $\mathbf{g}_N(k) = \mathbf{r}_N(w_N(k))$ and $\mathbf{g}_{N-1}(k) = \mathbf{r}^{(c)}_{N-1}(w^{(i)}_{N-1}(k))$,$k = 1, \dots K_{N-1,N}$. $w^{(i)}_{N-1}(k)$ will be used to locate the associated points in trajectory $\mathbf{r}_{N-2}(t_{N-2})$, $t_{N-2} = 1, \dots T_{N-2}$.

Through the pairwise alignment of $\mathbf{r}_{N-1}(t_{N-1})$ and $\mathbf{r}^{(c)}_{N-2}(k_{N-3,N-2})$ at the second block from the bottom of **Phase 1**, and the intermediate time indexes of the points of $\mathbf{g}_{N-1}(k)$ in $\mathbf{r}^{(c)}_{N-1}(k_{N-2,N-1})$, $w^{(i)}_{N-1}(k)$, which are obtained from the last block of **Phase 2**, the warped trajectory for the third last trajectory is computed as $\mathbf{g}_{N-2}(k) = \mathbf{r}^{(2c)}_{N-2}(w^{(i)}_{N-1}(k))$, $k = 1, \dots K_{N-1,N}$.The time indexes of the points of $\mathbf{g}_{N-2}(k)$, $k = 1, \dots K_{N-1,N}$ in the once-compressed trajectory $\mathbf{r}^{(c)}_{N-2}(k_{N-3,N-2})$, $k_{N-3,N-2} = 1, \dots K_{N-3,N-2}$ are indicated by $w^{(i)}_{N-2}(k) = w^{(2c)}_{N-2}(w^{(i)}_{N-1}(k))$.

At each block of **Phase 2**, a warped trajectory is computed through the pairwise alignment of **Phase 1** at the same layer, and the intermediate time indexes from a lower block of **Phase 2**. Algorithm 3 explains how to compute the time indexes $w_n(k)$, $n = 1, \dots N$ of the points of the warped trajectories $\mathbf{g}_n(k)$, $n = 1, \dots N$ in the original trajectories $\mathbf{r}_n(t_n)$, $n = 1, \dots T_n$, through the intermediate time indexes $w^{(i)}_n(k)$, $n = 2, \dots N-1$ in **Phase 2**. They satisfy $\mathbf{g}_n(k) = \mathbf{r}_n(w_n(k))$, $k = 1, \dots K_{N-1,N}$, $n = 1, \dots N$.

Fig. 3.3 illustrates this procedure with an example of aligning four GPS trajectories on one road segment. Each trajectory is shown in different markers: squares for $\mathbf{r}_1(t_1)$, stars for $\mathbf{r}_2(t_2)$, circles for $\mathbf{r}_3(t_3)$, and crosses for $\mathbf{r}_4(t_4)$. The markers indicate the location of the points on the trajectories. The trajectories have 16, 13, 14 and 10 points, respectively.

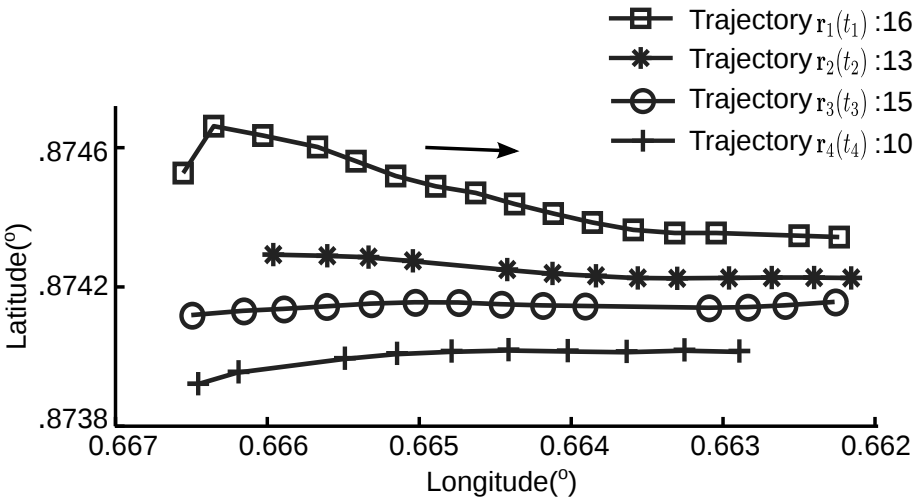
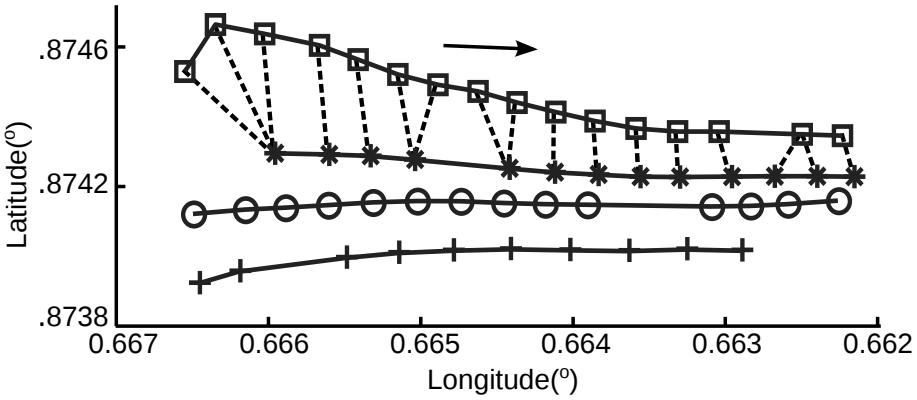


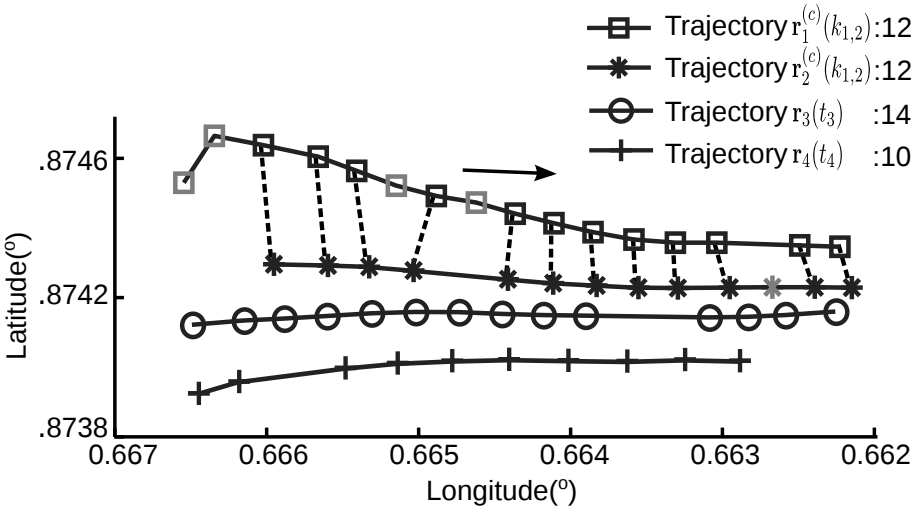

**Figure 3.3:** *Four GPS trajectories one one road segment.* Each trajectory is shown in different markers; the markers indicate the location of the points on the trajectories. The coordinates are relative to (-87,41).

Fig. 3.4, 3.5 and 3.6 illustrate successive steps of the "stretch and then compress" alignment strategy. If a point is removed from the trajectory by the *compress* operation, it is indicated using a **gray** marker, instead of a colored marker. In Fig. 3.4a, $\mathbf{r}_1(t_1)$ with 16 points and $\mathbf{r}_2(t_2)$ with 13 points are aligned using DTW. The matched points are connected using black dashed lines. In Fig. 3.4b, *many-to-one* correspondences are transformed to *one-to-one* correspondences by keeping the pair of points with the shortest distance. The removed points are shown in gray markers. 4 of 16 points are removed from $\mathbf{r}_1(t_1)$ and 1 of 13 points is removed from $\mathbf{r}_2(t_2)$, resulting in the once-compressed trajectories $\mathbf{r}_1^{(c)}(k_{1,2})$ and $\mathbf{r}_2^{(c)}(k_{1,2})$, $k_{1,2} = 1, \ldots 12$. The once-compressed trajectory $\mathbf{r}_2^{(c)}(k_{1,2})$ with 12 points and a new non-aligned trajectory $\mathbf{r}_3(t_3)$ with 14 points are input to the "stretch and then compress" module at the second step, as shown in Fig. 3.5, resulting in a twice-compressed trajectory $\mathbf{r}_2^{(2c)}(k_{2,3})$ and a once-compressed trajectory $\mathbf{r}_3^{(c)}(k_{2,3})$, $k_{2,3} = 1, \ldots 10$. 2 of 12 points are removed from $\mathbf{r}_2^{(c)}(k_{1,2})$, and 4 of 14 points from $\mathbf{r}_3(t_3)$. At the last step, $\mathbf{r}_3^{(c)}(k_{2,3})$ with 10 points and $\mathbf{r}_4(t_4)$ with 10 points are stretched and then compressed, producing a twice-compressed trajectory $\mathbf{r}_3^{(2c)}(k_{3,4})$ and a once-compressed trajectory $\mathbf{r}_4^{(c)}(k_{3,4})$, $k_{3,4} = 1, \ldots 9$. The first point of $\mathbf{r}_4(t_4)$ and the last point of $\mathbf{r}_3^{(c)}(k_{2,3})$ are removed.

The warped trajectories for the last two trajectories are $\mathbf{g}_4(k) = \mathbf{r}_4^{(c)}(k)$ and $\mathbf{g}_3(k) = \mathbf{r}_3^{(2c)}(k)$, $k = 1, \ldots 9$, which are indicated using crosses and circles in Fig. 3.7. The points of the other two warped trajectories, $\mathbf{g}_2(k)$ and $\mathbf{g}_1(k)$, $k = 1, \ldots 9$, can be located through the pairwise trajectory alignment using



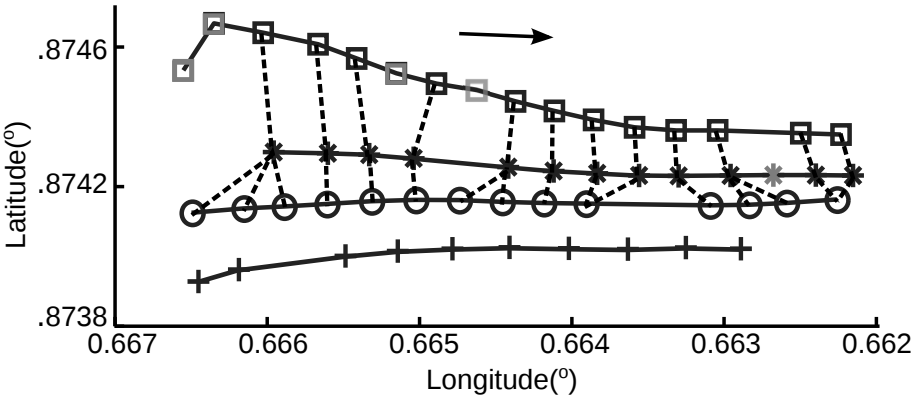

**(a)** Stretch



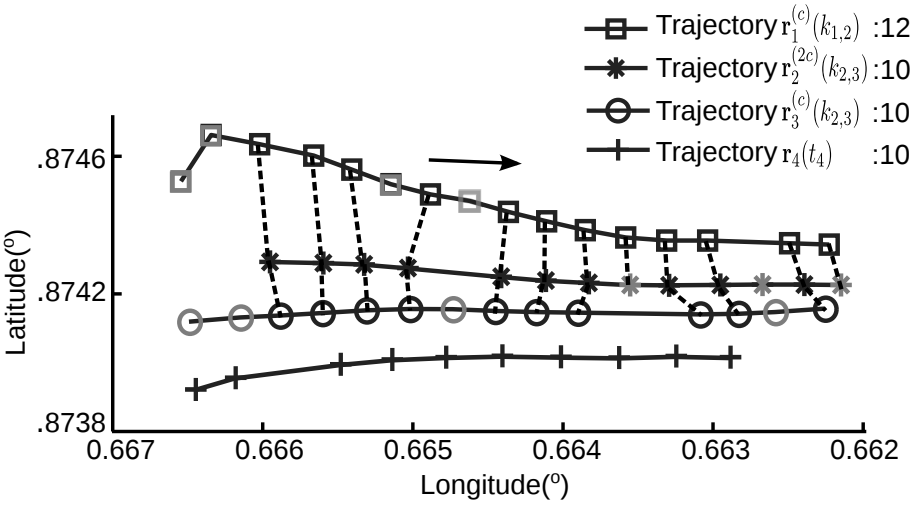

**(b)** Compress

**Figure 3.4:** *Four-trajectory alignment, Step 1:* $\mathbf{r}_1(t_1)$ with 16 points and $\mathbf{r}_2(t_2)$ with 13 points are stretched (depicted in (a)) and then compressed (depicted in (b)), resulting in $\mathbf{r}_1^{(c)}(k_{1,2})$ and $\mathbf{r}_2^{(c)}(k_{1,2})$ with 12 points associated at the same time index. 4 points in $\mathbf{r}_1(t_1)$ and 1 point in $\mathbf{r}_2(t_2)$ are removed by the *compress* operation. The coordinates are relative to (-87,41).

a bottom-top strategy. By following the arrows in Fig. 3.7, 9 of 10 points in $\mathbf{r}_2^{(2c)}(k_{2,3})$ are associated to the points of $\mathbf{g}_3(k)$. The last point of $\mathbf{r}_2^{(2c)}(k_{2,3})$, which is indicated using a black star but without arrow to it, is removed because there is no point in $\mathbf{g}_3(k)$ matched to it. 3 of 12 points in $\mathbf{r}_1^{(c)}(k_{1,2})$ are removed for the same reason, producing the warped trajectory $\mathbf{g}_1(k)$ with 9 points. Fig. 3.8 depicts the geographical locations of these four warped trajectories in different black markers.



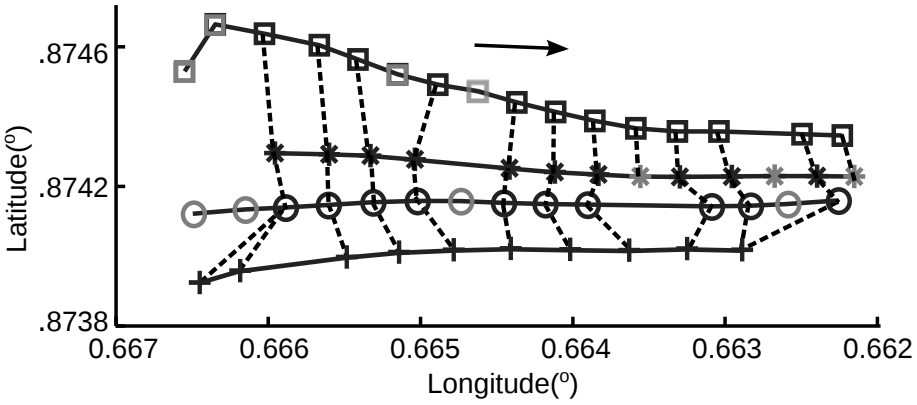

**(a)** Stretch



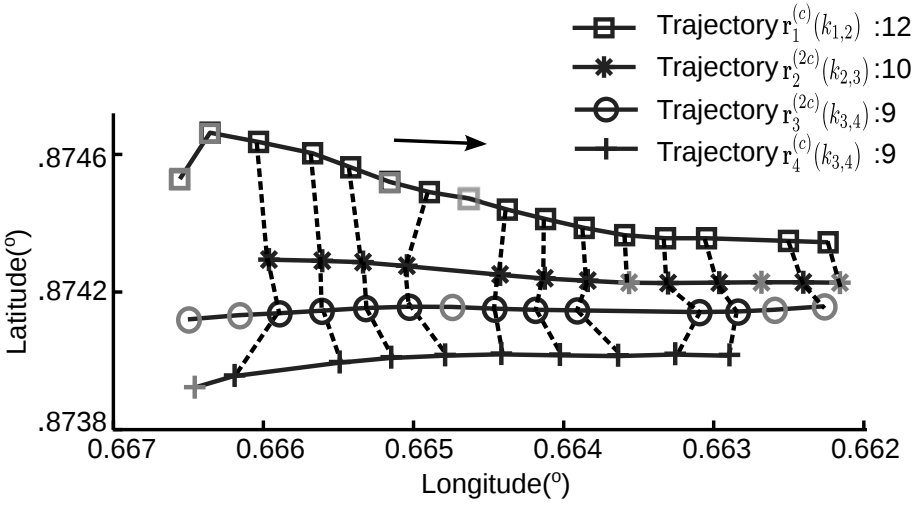

**(b)** Compress

**Figure 3.5:** *Four-trajectory alignment, Step 2:* $\mathbf{r}_2^{(c)}(k_{1,2})$ with 12 points and $\mathbf{r}_3(t_3)$ with 14 points are stretched and compressed, resulting in $\mathbf{r}_2^{(2c)}(k_{2,3})$ and $\mathbf{r}_3^{(c)}(k_{2,3})$ with 10 points associated at the same time index. 2 points in $\mathbf{r}_2^{(c)}(k_{1,2})$ and 4 points in $\mathbf{r}_3(t_3)$ are removed by the *compress* operation.

Table 3.1 shows the time indexes of the points of the warped trajectories in the original trajectories. The points at the same index $k$ of the warped trajectories are associated to each other. Each warped trajectory contains 9 points respectively.




**(a)** Stretch




**(b)** Compress

**Figure 3.6:** *Four-trajectory alignment, Step* 3*:* $\mathbf{r}_3^{(c)}(k_{2,3})$ with 10 points and $\mathbf{r}_4(t_4)$ with 10 points are stretched and compressed, resulting in $\mathbf{r}_3^{(2c)}(k_{3,4})$ and $\mathbf{r}_4^{(c)}(k_{3,4})$ with 9 points associated at the same time index. 1 point in $\mathbf{r}_3^{(c)}(k_{2,3})$ and 2 point in $\mathbf{r}_4(t_4)$ are removed by the *compress* operation.

## 3.4 Successor Classification based Alignment

Different from the pairwise alignment in Section 3.3, we propose a greedy procedure to align *many* trajectories at once through finding a *good* warp path through the local dissimilarity tensor in a way which keeps the points associated by each path element spatially close to each other. To reduce the computational cost, we propose to limit the candidate cells in the local dissimilarity tensor for each path element by classifying its *successors*. We will elaborate the

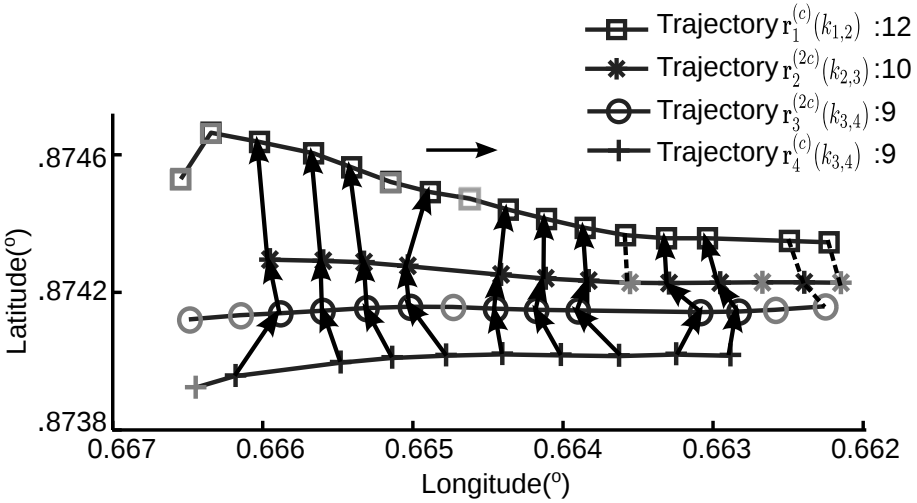



**Figure 3.7:** *Four-trajectory alignment, point association:* The points in other trajectories, which are associated to the points of $\mathbf{g}_4(k) = \mathbf{r}_4^{(c)}(k)$, $k = 1, \ldots 9$, are found by following the arrows from the bottom trajectory to the top trajectory.

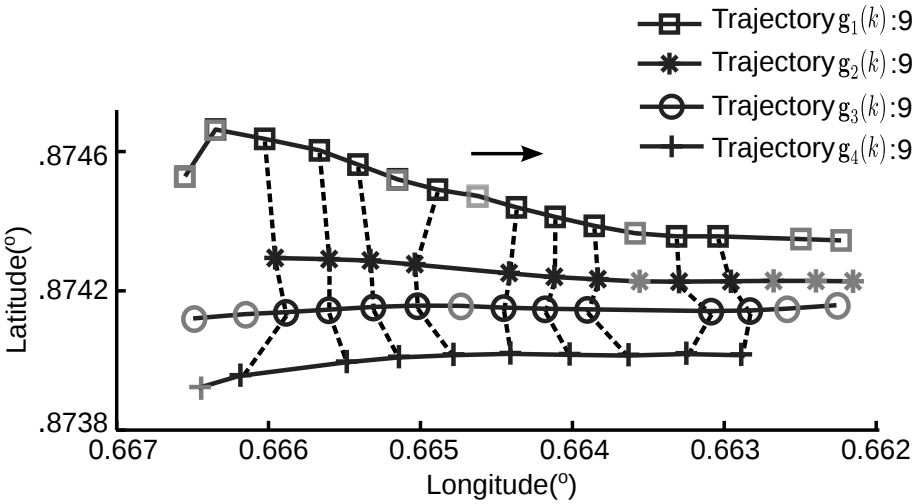



**Figure 3.8:** *Four-trajectory alignment, final result:* There are 9 data points left in each warped trajectory, which are connected using dashed lines. The removed points are indicated using gray markers.

**Table 3.1:** Time indexes $w_n(k)$

| $k$ | $w_1(k)$ | $w_2(k)$ | $w_3(k)$ | $w_4(k)$ |
|---|---|---|---|---|
| 1 | 3 | 1 | 3 | 2 |
| 2 | 4 | 2 | 4 | 3 |
| 3 | 5 | 3 | 5 | 4 |
| 4 | 7 | 4 | 6 | 5 |
| 5 | 9 | 5 | 8 | 6 |
| 6 | 10 | 6 | 9 | 7 |
| 7 | 11 | 7 | 10 | 8 |
| 8 | 13 | 9 | 11 | 9 |
| 9 | 14 | 10 | 12 | 10 |

algorithm in Section 3.4.1, and detail the procedure of finding the warp path using the proposed algorithm in Section 3.4.2.

### 3.4.1 Algorithm Elaboration

Because the computational cost of identifying the optimal alignment of *many* trajectories using DTW is prohibitively expensive, we try to find a close to optimal alignment in this section. Instead of finding the *optimal* warp path by back-tracking the overall dissimilarity, we propose to find a *good* warp path by forward-tracking the local dissimilarity. Although the alignment defined by this warp path is not optimal, it is acceptable as far as it satisfies properties of a good alignment described in Section 3.2. Without calculating the overall dissimilarity tensor, the computational cost is reduced extraordinarily.

Given $N$ trajectories $\mathbf{r}_n(t_n)$, $t_n = 1, \dots T_n$, $n = 1, \dots N$, we adopt the notation $(w_1(k), \dots w_N(k))$ for the warp path, where $k = 1, \dots K$, and $\mathbf{g}_n(k)$, $k = 1, \dots K$ for the warped trajectory, where $n = 1, \dots N$. Each element of the warp path $(w_1(k), \dots w_N(k))$ corresponds to the alignment of the points $\mathbf{r}_n(w_n(k))$,$n = 1, \dots N$. $w_n(k)$, $k = 1, \dots K$ denote the time indexes of the points of the warped trajectory in the original trajectory, $\mathbf{g}_n(k) = \mathbf{r}_n(w_n(k))$, $k = 1, \dots K$, $n = 1, \dots N$.

The forward-tracking procedure starts at cell $(1, \dots 1)$ of the local dissimilarity tensor $C_1$, and ends at cell $(T_1, \dots T_N)$ according to the Boundary constraint, i.e. $(w_1(1), \dots w_N(1)) = (1, \dots 1)$ and $(w_1(K), \dots w_N(K)) = (T_1, \dots T_N)$. Given the current element of the warp path: cell $(w_1(k), \dots w_N(k))$, the next element $(w_1(k+1), \dots w_N(k+1))$ is determined from the adjacent cells of the current path element in tensor $C_1$, depending on the local dissimilarities stored at these cells, i.e. the spatial dissimilarities of the corresponding points.

In a straightforward extension, we would consider all $2^N - 1$ adjacent cells which satisfy the Monotonicity and Continuity constraints as descried in Section 2.2.

$$
\begin{aligned}
(w_1(k+1), \dots w_N(k+1)) &\triangleq (w_1(k), \dots w_N(k)) + (j_1(k), \dots j_N(k)) \\
&= (w_1(k), \dots w_N(k)) + \underset{\substack{(j_1(k), \dots j_N(k)) \\ j_n \in [0,1] \\ \sum_{n=1}^{N} j_n \neq 0}}{\arg\min} C_1(w_1(k) + j_1, \dots w_N(k) + j_N)
\end{aligned} \tag{3.2}
$$

where $C_1$ is the local dissimilarity tensor calculated by Equation 3.1. The 0-or-1 steps $j_n(k)$, $n = 1, \dots N$, $j_n(k) \in [0, 1]$ describe how the time indexes of the points of the trajectories increase along the warp path: for $j_n(k) = 0$ the path does not advance in the $n^{th}$ dimension of tensor $C_1$, whereas for $j_n(k) = 1$ it advances by one time unit. This corresponds, respectively to $w_n(k+1) = w_n(k)$ and $w_n(k+1) = w_n(k)+1$ or to $\mathbf{r}_n(w_n(k+1)) = \mathbf{r}_n(w_n(k))$ and $\mathbf{r}_n(w_n(k+1)) = \mathbf{r}_n(w_n(k)+1)$.

However, it is still computationally expensive to calculate the dissimilarity of the corresponding points for all $2^N - 1$ adjacent cells.

There are $2N$ points involved in finding the next path element: $N$ points corresponding to the current path element $\mathbf{r}_n(w_n(k))$, $n = 1, \dots N$ and its $N$ *successors*. The current points are associated by the current path element, which means they are spatially close to each other. Depending on the speeds of the moving objects, the *successors* may be close to the current points, or distant from them. On the trajectories with similar moving velocities, the *successors* tends to appear at similar locations. Fig. 3.9 shows three examples of current points and their *successors*.

For clarity, the figure only shows small pieces of the trajectories in gray lines with circles. Fig. 3.9a and Fig. 3.9b show 10 pieces of factory worker trajectories, respectively. The points associated by the current path element are shown in circles, and the successors of the current path element are depicted in squares. The associated points are spatially close to each other. In the trajectory pieces shown in Fig. 3.9a, the factory workers move in the same direction at two different speeds, thus the successors gather at two different locations: one is around 5 cm to the current points, and the other one is around 10 cm away in y-direction. In Fig. 3.9b, the factory workers arrive at the right corner of the the storage rack. In 3 of the trajectory pieces, the workers stay stationary to pick up tools and parts from the storage rack, leading to that the successors are very close to the current points. In the other trajectory pieces, the factory workers are moving at two different speeds, the successors therefore are of different distances away from the current points. Fig. 3.9c show 10 pieces of vehicle trajectories with similar moving speeds. The trajectory sampling interval varies from 2 second to 10 seconds. The distance between the successor and the current point in each trajectory piece is very different. On the trajectory pieces with the same sampling interval, the successors are spatially close to each other.

In this section, we propose to classify the *successors* of the current path element, $\mathbf{r}_n(w_n(k)+1)$, $n = 1, \dots N$ for extending the trajectory into $M$ groups,

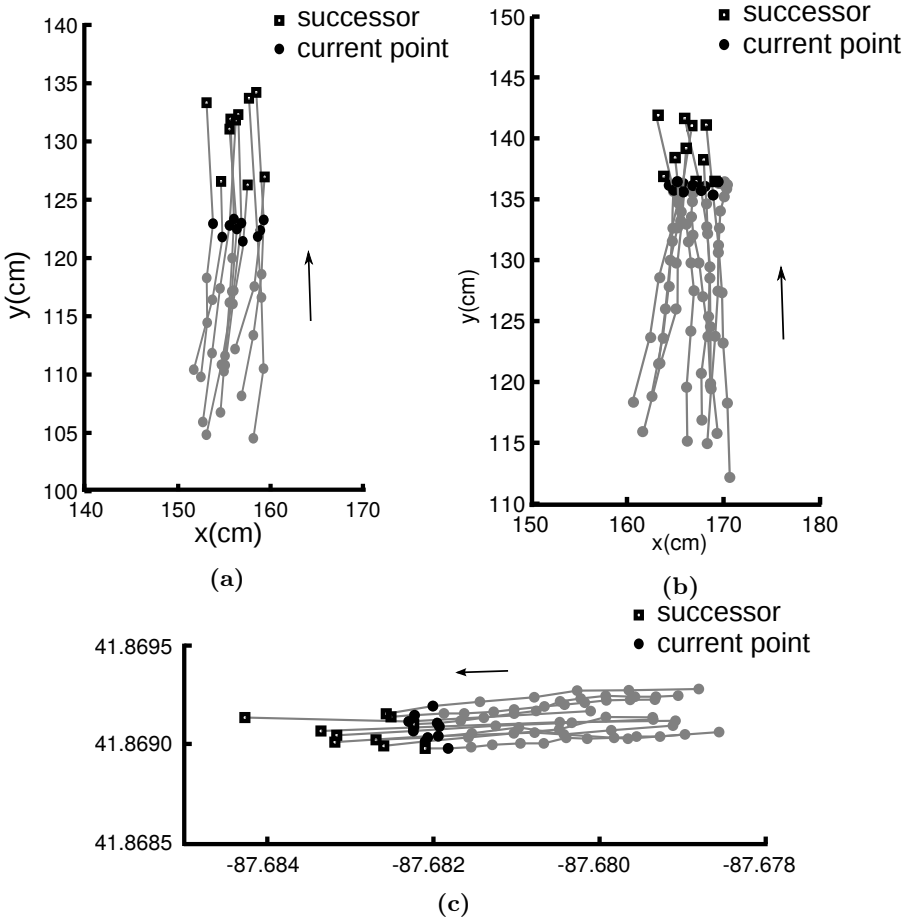



**Figure 3.9:** *Points corresponding to the current path element and its* successors. (a) and (b) show pieces of factory worker trajectories. (c) depicts pieces of vehicle trajectories. Given the points corresponding to the current path element, the *successors* gather at different locations depending on the speeds of the moving objects.

and enforce a Continuity constraint on the time indexes of each *group* of trajectories. This limits the number of candidate cells to $2^M - 1$. The corresponding points of these candidates cells have low dissimilarity, because they include the *successors* in the same class, which are similar to each other. One of the $2^M - 1$ candidate cells with low dissimilarities, whose corresponding points have the lowest dissimilarity, is taken as the next element of the warp path.

Because the current points $\mathbf{r}_n(w_n(k))$, $n = 1, \ldots N$ are not involved in limiting the candidates cells from $2^N - 1$ to $2^M - 1$, but only in finding the next path element from the $2^M - 1$ candidate cells, the next path element we find may be not be locally optimal, leading to lower alignment quality. It is

**Table 3.2:** *Allowable time index increases.* The *successors* of each path element $\mathbf{r}_n(w_n(k)+1)$, $n = 1, \dots N$ are grouped into $M$ clusters using K-means algorithm. When the warp path advances to the next element in the local dissimilarity tensor $C_1$, the time indexes of the points of the trajectories in the same group increase in the same way along the warp path. Under the Continuity constraint, the time indexes of the points can only increase by one or zero, $a_m \in [0,1]$, $m = 1, \dots M$, resulting in $2^M = 1$ allowable types of time index increases $b_i$, $i = 1, \dots 2^M - 1$, i.e. $2^M = 1$ candidate cells for the next path element.

| | $a_M$ | ... | $a_3$ | $a_2$ | $a_1$ |
|---|---|---|---|---|---|
| $b_1$ | 0 | ... | 0 | 0 | 1 |
| $b_2$ | 0 | ... | 0 | 1 | 0 |
| $b_3$ | 0 | ... | 0 | 1 | 1 |
| $b_4$ | 0 | ... | 1 | 0 | 0 |
| $b_5$ | 0 | ... | 1 | 0 | 1 |
| $b_6$ | 0 | ... | 1 | 1 | 0 |
| $b_7$ | 0 | ... | 1 | 1 | 1 |
| ... | ... | ... | ... | ... | ... |
| $b_{2^M-1}$ | 1 | ... | 1 | 1 | 1 |

possible that one of the other cells, rather than any of the $2^M - 1$ cells, has the lowest dissimilarity among all $2^N - 1$ adjacent cells. However, we have to compromise the alignment quality in order to reduce the computational cost.

We apply K-means algorithm to cluster the physical locations of the *successors* of the current path element $\mathbf{r}_n(w_n(k)+1)$, $n = 1, \dots N$, as shown in Algorithm 5. First, $M$ successors are randomly chosen from $N$ successors as initial cluster centroids. Second, each successor is assigned to the cluster whose centroid is closet to it. Third, the centroid of each cluster is recalculated using successors assigned to it. This procedure is repeated until the cluster centroids do not change any more. This results in the cluster label for each successor, $l(n,k)$, $l(n,k) \in [1, \dots M]$, $n = 1, \dots N$, where $n$ refers to the $n^{th}$ trajectory and $k$ refers to the $k^{th}$ element of the warp path. The successors of each path element are different.

Given the current path element $(w_1(k), \dots w_N(k))$ and the cluster labels of its *successors* $l(n,k)$, $n = 1, \dots N$, the joint move $(j_1(k), \dots j_N(k))$, which decides the next path element as shown in Equation 3.2, is restricted to 1 out of $2^M - 1$ possible moves:

$$(j_1(k), \dots j_N(k)) = \underset{\substack{(a_{l(1,k)}, \dots a_{l(N,k)}) \\ a_{l(n,k)} \in [0,1] \\ l(n,k) \in [1,M] \\ \sum_{n=1}^{N} a_{l(n,k)} \neq 0}}{\arg\min} C_1(w_1(k)+a_{l(1,k)}, \dots w_N(k)+a_{l(N,k)}) \quad (3.3)$$

where $C_1$ is the local dissimilarity tensor, which is calculated using Equation 3.1. The 0-or-1 steps $a_{l(n,k)}$, $n = 1, \dots N$, $a_{l(n,k)} \in [0,1]$, $l(n,k) \in [1,M]$

**Algorithm 4** AlignTrajectories

**Input:** $\{\mathbf{r}_n(t_n),\, t_n = 1, \dots T_n,\, n = 1, \dots N\}$; $b_i,\, i = 1, \dots 2^M - 1$
**Output:** $(w_1(k), \dots w_N(k)),\, k = 1, \dots K$

Initialization $k \leftarrow 2$, $(w_1(1), \dots w_N(1)) \leftarrow (1, \dots 1)$
**while** $(w_1(k), \dots w_N(k)) \neq (T_1, \dots T_N)$ **do**
    $(l(1,k), \dots l(N,k)) \leftarrow ClusterData(\mathbf{r}_n(w_n(k)+1),\, n = 1, \dots N)$ Algorithm 5
    **for** $i = 1$ to $2^M - 1$ **do**
        **for** $n = 1$ to $N$ **do**
            $j_n^{(i)} \leftarrow \begin{cases} b_i(l_n(k)) & \text{if } w_n(k) < T_n \\ 0 & \text{if } w_n(k) = T_n \end{cases}$
        **end for**
        $\mu_i \leftarrow \frac{1}{N} \sum_{n=1}^{N} \mathbf{r}_n(w_n(k) + j_n^{(i)})$
        $\sigma_i \leftarrow \sqrt{\frac{1}{N} \sum_{n=1}^{N} (\mathbf{r}_n(w_n(k) + j_n^{(i)}) - \mu_i)^2}$
    **end for**
    $i_0 \leftarrow \arg \min_{i \in [1,7]} (\sigma_i)$
    $(w_1(k+1), \dots w_N(k+1)) \leftarrow (w_1(k), \dots w_N(k)) + (j_1^{(i_0)}, \dots j_N^{(i_0)})$
    $k \leftarrow k + 1$
**end while**

describe how the time indexes of the points of the trajectories increase along the warp path: for $a_m = 0$ the time indexes of the points of all trajectories $\{n : l(n,k) = m, n \in [1,N]\}$, whose *successors* are in the same cluster $m$, stay the same, whereas for $a_m = 1$ they increase by one. This guarantees the time indexes of the points of the trajectories in the same group increase in the same way when the warp path advances to the next cell. Table 3.2 shows the $2^M - 1$ possible time index increases, $b_i,\, i = 1, \dots 2^M - 1$.

By classifying the *successors* of each path element, the number of the candidate cells for the next path element is limited from $2^N - 1$ to $2^M - 1$, where $N$ is the number of the trajectories, and $M$ is the number of the categories of the *successors*. During the procedure of forward-tracking the local dissimilarity through $C_1$, we just need calculate the values of the $2^M - 1$ adjacent cells in $C_1$ for each path element, to determine the next path element. The computational cost is reduced enormously. Although the successor classification takes some time, it is considerably less than calculating the values of all $2^N - 1$ adjacent cells, especially for very big data set. Thus this trajectory alignment algorithm can be executed in a reasonable amount of time.

### 3.4.2 Aligning Trajectories

As shown in Algorithm 4, the warp path can be deduced as followed, given $N$ trajectories $\mathbf{r}_n(t_n)$, $t_n = 1, \dots T_n$, $n = 1, \dots N$:

The warp path starts at cell $(1, \dots 1)$ of the local dissimilarity tensor $C_1$,

**Algorithm 5** ClusterData

**Input:** $\mathbf{r}_n(w_n(k)+1),\ n=1,\dots N$
**Output:** $(l(1,k),\dots l(N,k))$

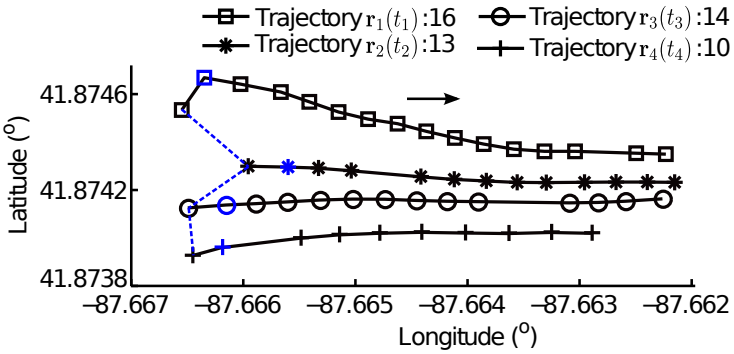
Initialize $M$ cluster centroids $\mu_m$, $m=1,\dots M$ randomly from the input
**while** there are still changes in the centroids **do**
  **for** $n \leftarrow 1, N$ **do** **Update cluster assignments**
    $l(n,k) \leftarrow \arg\min_{m\in[1,M]} \|\mathbf{r}_n(w_n(k)+1)-\mu_m\|^2$
  **end for**
  **for** $m \leftarrow 1, M$ **do** **Update cluster centroids**
    $\mu_m \leftarrow \frac{\sum_{n=1}^{N} 1\{l(n,k)=m\}\mathbf{r}_n(w_n(k)+1)}{\sum_{n=1}^{N} 1\{l(n,k)=m\}}$
  **end for**
**end while**

i.e. $(w_1(1),\dots w_N(1)) = (1,\dots 1)$. The *successors* of the first path element $(w_1(1),\dots w_N(1))$, $\mathbf{r}_n(w_n(1)+1) = \mathbf{r}_n(2)$, $n=1,\dots N$, are grouped into $M$ clusters using $K$-means algorithm, resulting in the group labels for the *successors* $l(n,1)$, $n=1,\dots N$. According to Table 3.2, there are $2^M-1$ candidate cells for the second path element, $(w_1(1),\dots w_N(1)) + (b_i(l(1,1)),\dots b_i(l(N,1)))$, $i=1,\dots 2^M-1$. One of the candidate cells with the lowest dissimilarity is chosen as the second element of the warp path $(w_1(2),\dots w_N(2))$.

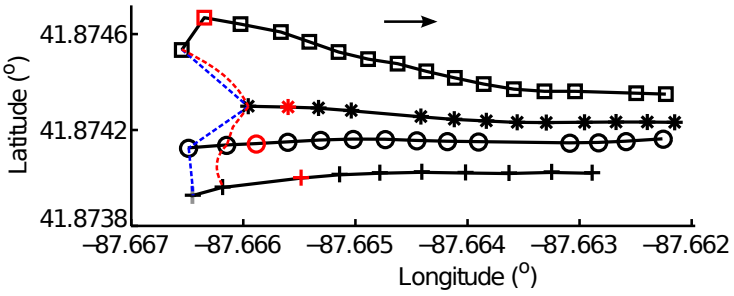
This process is repeated until the warp path ends at cell $(T_1,\dots T_N)$. Each time, one of the $2^M-1$ candidate cells, which is chosen from $2^N-1$ adjacent cells of the current path element by classifying its *successors* $\mathbf{r}_n(w_n(k)+1)$, $n=1,\dots N$, is determined as the next path element $(w_1(k+1),\dots w_N(k+1))$, if its corresponding points have the lowest dissimilarity.

The outputs of the procedure are: a warp path through the local dissimilarity matrix $C_1$, $(w_1(k),\dots w_N(k))$, $k=1,\dots K$, and the warped trajectories along the warp path $\mathbf{g}_n(k)=\mathbf{r}_n(w_n(k))$, $k=1,\dots K$, $n=1,\dots N$ with points associated at the same time index.

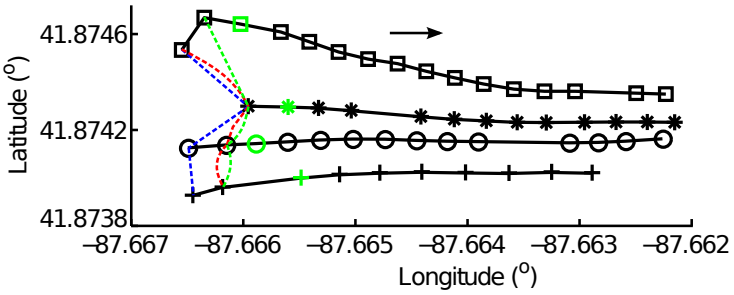
The same four GPS trajectories shown in Fig. 3.3 are aligned using the greedy method based on classifying the *successors* into three groups as an example. The warp path starts at the first cell of the local dissimilarity matrix $C_1$, i.e. $(w_1(1),w_2(1),w_3(1),w_4(1)) = (1,1,1,1)$, which associates the first point of each trajectory. The associated points are connected using blue dashed line in Fig. 3.10a. The *successors*, $\mathbf{r}_1(1+1)$, $\mathbf{r}_2(1+1)$, $\mathbf{r}_3(1+1)$, and $\mathbf{r}_4(1+1)$, which are indicated using blue markers, are classified into three groups using Algorithm 5, resulting in group labels $(l(1,1),l(2,1),l(3,1),l(4,1)) = (1,2,3,3)$. Because the *successors* in the third and fourth trajectories belong to the same group, the time indexes of the points of these two trajectories increase in the same way when the warp path advances to its second element.




**(a)** Step 1




**(b)** Step 2




**(c)** Step 3

**Figure 3.10:** *Example of four-trajectory alignment.* In (a), the warp path starts at cell $(1, 1, 1, 1)$ of the local dissimilarity tensor $C_1$, i.e. $(w_1(1), w_2(1), w_3(1), w_4(1)) = (1, 1, 1, 1)$. The points associated by the first path element are connected using blue dashed lines, and the *successors* are indicated using blue markers. The *successors* are classified into three groups, creating 7 candidate cells in $C_1$ for the second path element. The candidate cell with the lowest dissimilarity, whose corresponding points are connected using red dashed lines as shown in (b), is chosen as the second path element, $(w_1(2), w_2(2), w_3(2), w_4(2)) = (1, 1, 2, 2)$. Its *successors* are indicated using red markers. (c) shows the points associated by the third path element $(w_1(3), w_2(3), w_3(3), w_4(3)) = (2, 1, 2, 2)$ using green dashed lines, and its *successors* in green markers.

7 candidate cells for the second path element are listed:

$$\begin{aligned}
&\text{candidate 1} \quad (1,1,1,1)+(0,0,1,1)=(1,1,2,2)\\
&\text{candidate 2} \quad (1,1,1,1)+(0,1,0,0)=(1,2,1,1)\\
&\text{candidate 3} \quad (1,1,1,1)+(0,1,1,1)=(1,2,2,2)\\
&\text{candidate 4} \quad (1,1,1,1)+(1,0,0,0)=(2,1,1,1)\\
&\text{candidate 5} \quad (1,1,1,1)+(1,0,1,1)=(2,1,2,2)\\
&\text{candidate 6} \quad (1,1,1,1)+(1,1,0,0)=(2,2,1,1)\\
&\text{candidate 7} \quad (1,1,1,1)+(1,1,1,1)=(2,2,2,2)
\end{aligned}$$

Because the corresponding points of the first candidate cell have the lowest dissimilarity, this cell is chosen as the second element of the warp path, $(w_1(2), w_2(2), w_3(2), w_4(2)) = (1,1,2,2)$.

As shown in Fig. 3.10b, the associated points by the second path element are connected using red dashed line, and the *successors*, $\mathbf{r}_1(1+1)$, $\mathbf{r}_2(1+1)$, $\mathbf{r}_3(2+1)$, and $\mathbf{r}_4(2+1)$, are indicated using red markers. Using Algorithm 5, these *successors* are classed into three groups, resulting in group labels $(l(1,2), l(2,2), l(3,2), l(4,2)) = (1,2,3,2)$. According to the group labels and the possible time index increases in Table 3.2, 7 candidate cells for the third element of the warp path are listed:

$$\begin{aligned}
&\text{candidate 1} \quad (1,1,2,2)+(0,0,1,0)=(1,1,3,2)\\
&\text{candidate 2} \quad (1,1,2,2)+(0,1,0,1)=(1,2,2,3)\\
&\text{candidate 3} \quad (1,1,2,2)+(0,1,1,1)=(1,2,3,3)\\
&\text{candidate 4} \quad (1,1,2,2)+(1,0,0,0)=(2,1,2,2)\\
&\text{candidate 5} \quad (1,1,2,2)+(1,0,1,0)=(2,1,3,2)\\
&\text{candidate 6} \quad (1,1,2,2)+(1,1,0,1)=(2,2,2,3)\\
&\text{candidate 7} \quad (1,1,2,2)+(1,1,1,1)=(2,2,3,3)
\end{aligned}$$

The fourth candidate cell is chosen as the third path element, i.e. $(w_1(3), w_2(3), w_3(3), w_4(3)) = (2,1,2,2)$, because its corresponding points have the lowest dissimilarity. Its associated points are connected using green dashed lines and its *successors* are indicated using green markers, as shown in Fig. 3.10c.

Fig. 3.11 shows the point associations along the full warp path. The data points associated by each element of the warp path are connected using dashed lines with the same color. Table 3.3 shows the time indexes of the points of the warped trajectories in the original trajectories. The points at the same index $k$ of the warped trajectories are associated to each other. Along the warp path, each trajectory is warped to 19 points respectively.

Although the warped trajectories are longer than the original trajectories, the trajectory alignment does not create any new point, but establishes *many-to-one* correspondences among the points of the original trajectories. Therefore, there is no positional difference between the points of the warped trajectories and the original trajectories.

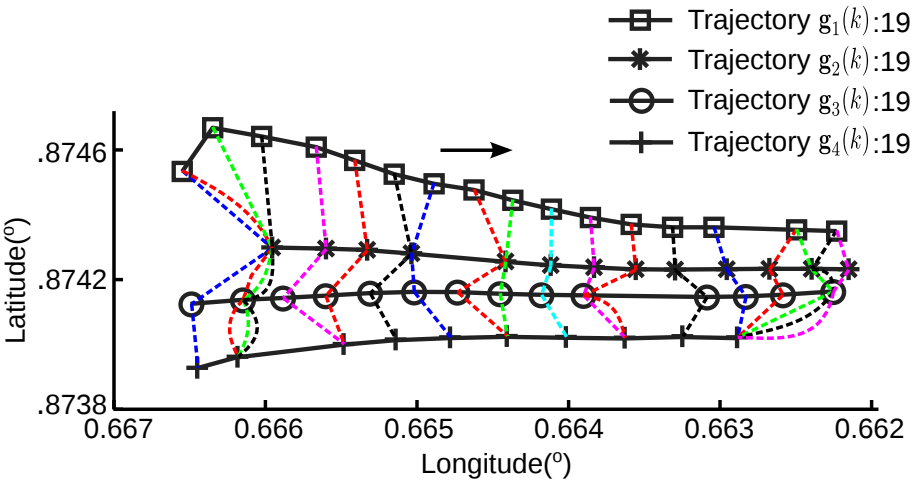



**Figure 3.11:** *Point associations.* The full warp path contains 19 elements. Each element of the warp path associates 4 points, each of which is from an individual trajectory. The associated points are connected using dashed lines of the same color. The coordinates are relative to (-87,41).

**Table 3.3:** *Time indexes* $w_n(k)$

| $k$ | $w_1(k)$ | $w_2(k)$ | $w_3(k)$ | $w_4(k)$ |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 2 | 2 |
| 3 | 2 | 1 | 2 | 2 |
| 4 | 3 | 1 | 2 | 2 |
| 5 | 4 | 2 | 3 | 3 |
| 6 | 5 | 3 | 4 | 3 |
| 7 | 6 | 4 | 5 | 4 |
| 8 | 7 | 4 | 6 | 5 |
| 9 | 8 | 5 | 7 | 6 |
| 10 | 9 | 5 | 8 | 6 |
| 11 | 10 | 6 | 9 | 7 |
| 12 | 11 | 7 | 10 | 8 |
| 13 | 12 | 8 | 10 | 8 |
| 14 | 13 | 9 | 11 | 9 |
| 15 | 14 | 10 | 12 | 10 |
| 16 | 15 | 11 | 13 | 10 |
| 17 | 15 | 12 | 14 | 10 |
| 18 | 16 | 12 | 14 | 10 |
| 19 | 16 | 13 | 14 | 10 |

## 3.5 Difference between the Proposed Methods

In the following, to distinguish the warped trajectories produced by each method, we add superscripts to their notations, $\mathbf{g}_n^{(1)}(k)$, $k = 1, \dots K^{(1)}$, $n = 1, \dots N$, where the superscript (1) refers to the stretch-and-then-compress method, $\mathbf{g}_n^{(2)}(k)$, $k = 1, \dots K^{(2)}$, $n = 1, \dots N$, where the superscript (2) refers to the greedy method based on successor classification. The differences between the proposed methods are summarized as:

- **Point correspondence.** The stretch-and-then-compress method builds *one-to-one* correspondences among the points of the trajectories. The greedy method based on successor classification establishes *many-to-one* point correspondences during the alignment.

  Fig. 3.8 shows the point associations of four GPS trajectories built using the stretch-and-then-compress method. Given any point in one warped trajectory, it can only be associated to a single point in each of other warped trajectories by following the connecting lines. Fig. 3.11 shows the *many-to-one* point correspondences of the same four GPS trajectories established using the greedy method based on successor classification. For instance, the first three points of the first trajectory $\mathbf{r}_1(t_1)$, $t_1 = 1, 2, 3$ are associated to the same point of the second trajectory $\mathbf{r}_2(1)$.

- **The length of the warped trajectories.** Another difference between the proposed methods lies in the number of points on the warped trajectories. The stretch-and-then-compress method compresses the trajectories by keeping pairs of points which are more close to each other positionally. The warped trajectories $\mathbf{g}_n^{(1)}(k)$, $k = 1, \dots K^{(1)}$, $n = 1, \dots N$, are shorter than the original trajectories $\mathbf{r}_n(t_n)$, $t_n = 1, \dots T_n$, $n = 1, \dots N$, i.e. $K^{(1)} \leq T_n$, $n = 1, \dots N$.

  In contrast, the greedy method based on successor classification aligns the trajectories by allowing many points of one trajectory matching to the same point of the other trajectories. Therefore the warped trajectories produced, $\mathbf{g}_n^{(2)}(k)$, $k = 1, \dots K^{(2)}$, $n = 1, \dots N$, are longer than the original trajectories, i.e. $K^{(2)} \geq T_n$, $n = 1, \dots N$.

  As shown in Fig. 3.3, there are 16, 13, 14 and 10 points in each GPS trajectory, respectively. Fig. 3.8 shows that each warped trajectory produced using the stretch-and-then-compress method contains 9 points, which is shorter than any of the original trajectories. However there are 19 points in the warped trajectories produced by the greedy method based on successor classification. as shown in Fig. 3.11. They are longer than any of the original trajectories.

- **The influence of aligning order.** The stretch-and-then-compress method aligns the trajectories in an ascending order of average dissimilarity. The stretch-and-then-compress module only processes two trajectories at a time, including a once-compressed trajectory and an unaligned

trajectory. The last once-compressed trajectory decides the length of all warped trajectories. The points in the other trajectories, which are associated to the points of the last once-compressed trajectory, are located through the outputs of the stretch-and-then-compress modules in a reverse order from the second last trajectory to the first trajectory. Therefore the trajectory input to the module later has more influence on the alignment. The greedy method based on successor classification processes all of the trajectories at once. The aligning order is irrelevant in this method.

As shown in Fig. 3.8, the last points of the first two trajectories, $\mathbf{r}_1(16)$ and $\mathbf{r}_2(13)$, are associated to each other, which are connected using dashed lines. However, both of them are removed because there are no points in the last trajectory matched to them, indicating more influence of the last trajectory on the result.

As shown in Fig. 3.8, the seventh point of the first trajectory, $\mathbf{r}_1(7)$, the fourth point of the second trajectory, $\mathbf{r}_2(4)$, the sixth point of the third trajectory, $\mathbf{r}_3(6)$, and the fifth point of the fourth trajectory, $\mathbf{r}_4(5)$, are associated to each other using the stretch-and-then-compress method. This association is obtained by following the arrows from the last to the first trajectory in Fig. 3.7. As shown in Table 3.3 for the greedy method based on successor classification, the same association is found at the eighth element of the warp path, $(w_1(8), w_2(8), w_3(8), w_4(8)) = (7, 4, 6, 5)$, This association is established by calculating the dissimilarity of the corresponding points of the 7 candidate cells for the eighth path element, which are selected from the adjacent cells of the seventh path element by classifying its *successors* $\mathbf{r}_n(w_n(7)+1)$, $n = 1, \ldots 4$. There is no aligning order involved in determining the eighth path element from its 7 candidate cells.

## 3.6 Average Trajectory Extraction

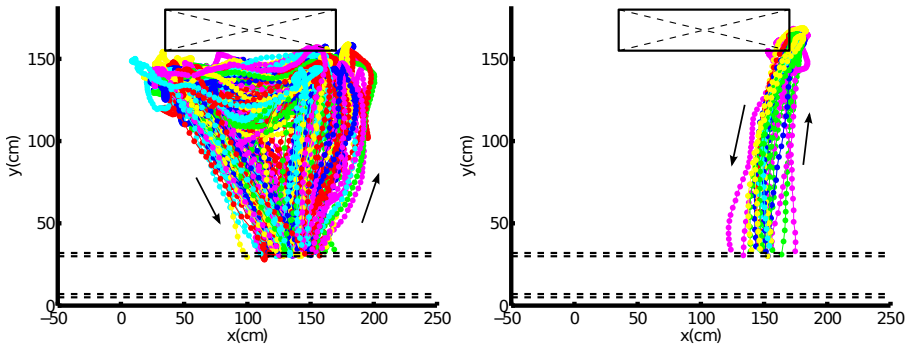
Using the point associations built during the trajectory alignment, the average trajectory can be easily extracted. Given the warped trajectories, $\mathbf{g}_n(k)$, $k = 1, \ldots K$, $n = 1, \ldots N$, the average trajectory with $K$ points is calculated as:

$$\mathbf{g}_0(k) = \frac{1}{N} \sum_{n=1}^{N} \mathbf{g}_n(k), \tag{3.4}$$

where $k = 1, \ldots K$.

In the application of work cycle optimization, the average trajectory acts as a prototypical route for the factory workers to execute the work cycle. In the other application of road network inference, the average trajectory is the geometric representation for each road segment.

The variance of the physical locations of trajectory points along each point of the average trajectory is calculated as:




**(a)** Trajectories for work cycle 1. **(b)** Trajectories for work cycle 2.

**Figure 3.12:** *Factory worker trajectories.* There are 82 and 12 trajectories in each type of executed work cycle, respectively.

$$\sigma_g(k) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{g}_n(k) - \mathbf{g}_0(k))^2, \tag{3.5}$$

where $k = 1, \ldots K$.

The mean variance of the physical locations of trajectory points along the average trajectory is calculated as:

$$\sigma = \frac{1}{K}\sum_{k=1}^{K}\sigma_g(k), \tag{3.6}$$

where $\sigma_g(k)$ is the variance of the physical locations of trajectory points along the $k^{th}$ point of the average trajectory. $\sigma$ is used to measure the performance of the alignment algorithm. The smaller $\sigma$ is, the more similar the associated points are. Higher similarity of the points associated by warp path elements better satisfy the properties of good trajectory alignment.

## 3.7 Results

We test both of presented methods using two data sets: factory worker trajectories which are obtained from a multi-camera tracking system (Factory data set), and vehicle trajectories which are recorded using GPS devices (Chicago data set and Berlin data set).

For the Factory data set, there are 124 trajectories in total, as shown in Fig. 2.5b. In this data set, the worker repeats the procedure of going to the storage rack, picking up tools and parts, coming back to the assembly platform and doing his operation. In Chapter 2, we have identified two types of executed work cycles with *many* trajectories. As show in Fig. 3.12a, there are 82 trajectories recorded for one type of work cycle, in which the workers walk

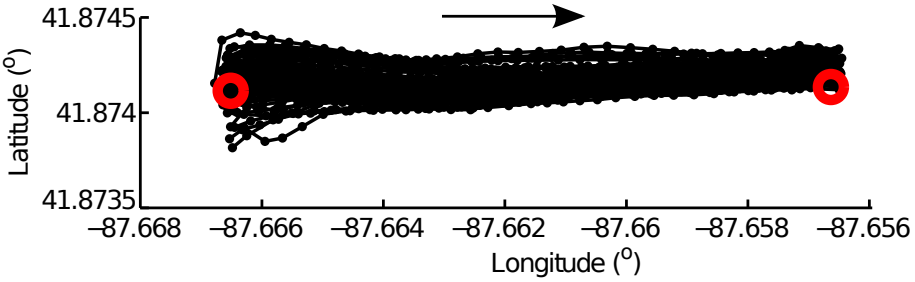



**Figure 3.13:** *GPS trajectories on one road segment.* GPS trajectories are shown in black lines with points, and the intersections in red circles. 77 GPS trajectories are recorded for this road segment directed from the left intersection to the right one.

through the whole area in front of the storage rack to pick up tools and parts. In the other type of work cycle, the workers only visit the right corner of the storage rack and come back to the assembly platform directly. Fig. 3.12b shows 12 trajectories recorded for it. We apply both proposed methods to align the trajectories, and show the results of the trajectory alignment for each type of executed work cycle separately. We also compare our extracted average trajectories with the results of other DTW-related methods for trajectory averaging.

For vehicle trajectories, we show the results of GPS trajectory alignment for one road segment in Chicago data set as an example. The alignment for the whole data set will be discussed in Section 6.6 after the intersection detection. As shown in Fig. 3.13, the road segment directly connects two intersections, which are indicated using red circles. There are 77 trajectories associated to this road segment, shown in black lines with points indicating the locations of GPS recordings. The arrow indicates the direction of these trajectories, which is from the left intersection to the right one.

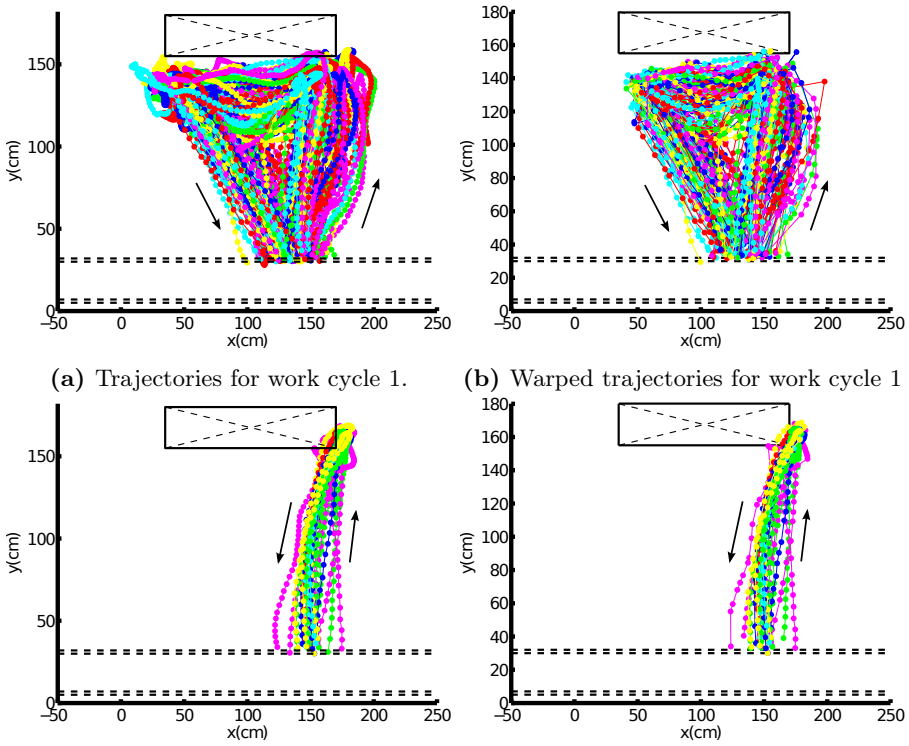
In this section, we will show the results of alignment for factory worker trajectories and vehicle trajectories, established using each of the proposed methods separately. Experimental results show that our method successfully aligned the trajectories by associating the points together, which are spatially close to each other.

## 3.7.1 Results of Factory Data Set

In this section, we show the trajectory alignment results for the Factory data set. We will first show the results using the stretch-and-then-compress method in Section 3.7.1.1, then the results using the greedy method based on successor classification in Section 3.7.1.2.

### 3.7.1.1 Stretch-and-then-Compress Method

Fig. 3.14 depicts the warped trajectories for each type of executed work cycle, respectively. The location of the storage rack is shown as a black rectangle



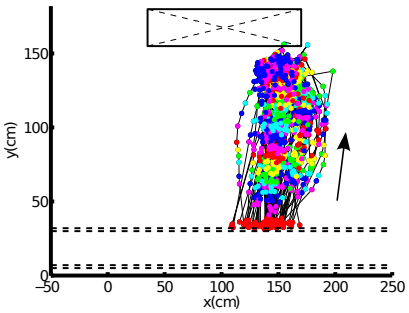

**(a)** Trajectories for work cycle 1. **(b)** Warped trajectories for work cycle 1

**(c)** Trajectories for work cycle 2. **(d)** Warped trajectories for work cycle 2

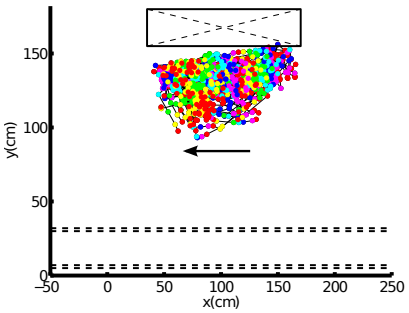
**Figure 3.14:** *Warped trajectories of factory workers using the stretch-and-then-compress Method.* (a) and (b) show the original trajectories and the warped trajectories separately for the first type of executed work cycle. (c) and (d) are for the second type of executed work cycle. Because the *compress* operation keeps pairs of points which are more close to each other positionally, the warped trajectories are shorter than the original trajectories.

near the top of the figure, and the dashed lines at the bottom show the location of the conveyor belt. The stretch-and-then-compress method produces warped trajectories of the same length. Compared to the original trajectories in Fig. 3.14a and 3.14c, the warped trajectories produced do not contain all points of the original trajectories. The number of the points in the original trajectories varies from 163 to 616 (average: 259) for work cycle 1, and from 80 to 150 (average: 104) for work cycle 2. Each of the warped trajectories in Fig. 3.14b contains 39 points, which is shorter than any of the original trajectory in Fig. 3.14a. Fig. 3.14d shows the warped trajectories of the same length with 52 points for work cycle 2. They are also shorter than any trajectory in Fig. 3.14c.

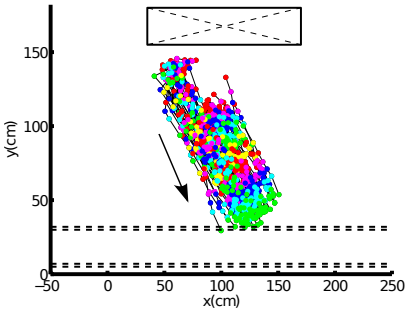
As shown in Fig. 3.14, the trajectory points recorded on the way to the storage rack overlap with points on the way back to the assembly platform. Thus




**(a)** Part 1

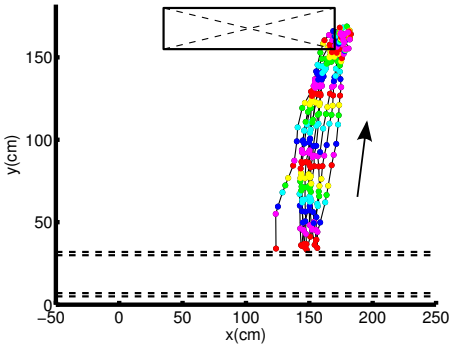



**(b)** Part 2

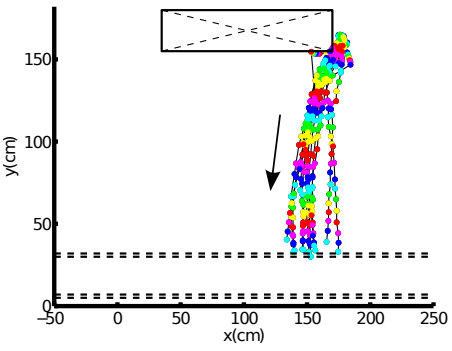



**(c)** Part 3

**Figure 3.15:** *Point associations of the trajectories for work cycle* 1 *using the stretch-and-then-compress method.* The associated points at the same index $k$ of the warped trajectories, $\mathbf{g}_n(k)$, $n = 1, \ldots N$, are indicated using dots of the same color.




**(a)** Part 1




**(b)** Part 2

**Figure 3.16:** *Point associations of the trajectories for work cycle* 2 *using the stretch-and-then-compress Method.* The points of the same color are associated to each other.

it is difficult to depict the association of the points in one single picture. To show the point associations more clearly, we partition the warped trajectories

into several parts, and picture the associations for each part separately.

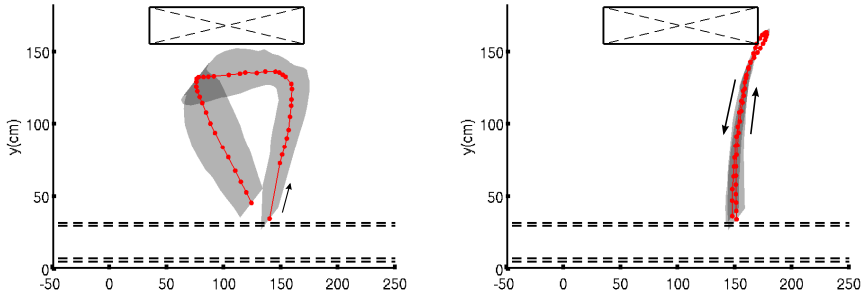
Fig. 3.15 shows the point associations for the trajectories for work cycle 1 in three parts: Part 1 in Fig. 3.15a for the journey that the workers walk from the assembly platform to the right corner of the storage rack; Part 2 in Fig. 3.15b for that the workers walk from the right to the left corner of the storage rack; Part 3 in Fig. 3.15c for that the workers walk from the left corner of the storage rack back to the assembly line. In Fig. 3.15, the points at the same time index of the warped trajectories, $\mathbf{g}_n(k)$, $n = 1, \dots N$, are indicated using dots of the same color. Red, blue, magenta, cyan, green and yellow are used iteratively to show the point associations.

Fig. 3.16 shows point associations for the trajectories for work cycle 2 in two parts: Part 1 in Fig. 3.16a the workers walk from the assembly platform to the right corner of the storage rack; Part 2 in Fig. 3.16b the workers walk from the right corner back to the assembly platform. The points at the same time index of the warped trajectories, are also indicated using dots of the same color.

For a good trajectory alignment, the associated points, i.e. the points at the same time index of the warped trajectories, should be spatially close to each other. In Fig. 3.15 and Fig. 3.16, the associated points are indicated using dots of the same color. If the physical locations of the points are close to other points of the same color, but non-overlapped with other points of different colors, the point associations will lead to a good trajectory alignment.

In Fig. 3.15, the points in different colors mix together, and the associated points can not be identified from their colors easily. In Fig. 3.16, the associated points can be identified easily by their colors. We can clearly see the iterative uses of the seven colors in indicating associated points. It means that the stretch-and-then-compress method works better in aligning the 12 trajectories for work cycle 2 with high similarity. The 82 trajectories for work cycle 1 are more diverse. The stretch-and-then-compress method is based on pairwise trajectory alignment. Although the associated points in non-adjacent trajectories can be located through the alignment between adjacent trajectories, the points may be not optimally associated, and the distance between associated points will be amplified as the associations are established through more adjacent trajectory alignments. In the future, we will try to associate the points in non-adjacent trajectories in a better way, through utilizing the *many-to-one* correspondences established by the *stretch* operation, instead of purely using *one-to-one* point correspondences built by the *compress* operation.

Fig. 3.17 depicts the average trajectory $\mathbf{g}_0(k)$, $k = 1, \dots K$ and the variance of the physical locations along the average trajectory, $\sigma_g(k)$, $k = 1, \dots K$, for each type of executed work cycle respectively. The average trajectory is shown in red lines with dots. The location variance of the trajectory points is indicated using a gray band along the average trajectory. The wider the band is, the more dissimilar the associated points are and vice versa. From Fig. 3.17, we can see the factory workers execute the first type of work cycle more variously than the second one. The mean variance of the physical locations of the trajectory points




**(a)** Work cycle 1 **(b)** Work cycle 2

**Figure 3.17:** *Physical location variance along the average trajectory extracted using the stretch-and-then-compress method.* (a) depicts the average trajectory in red lines with dots indicating the physical locations of the points on the average trajectory for work cycle 1. The width of the gray band along the average trajectory indicates the variance of the physical locations of the trajectory points. (b) is for work cycle 2.

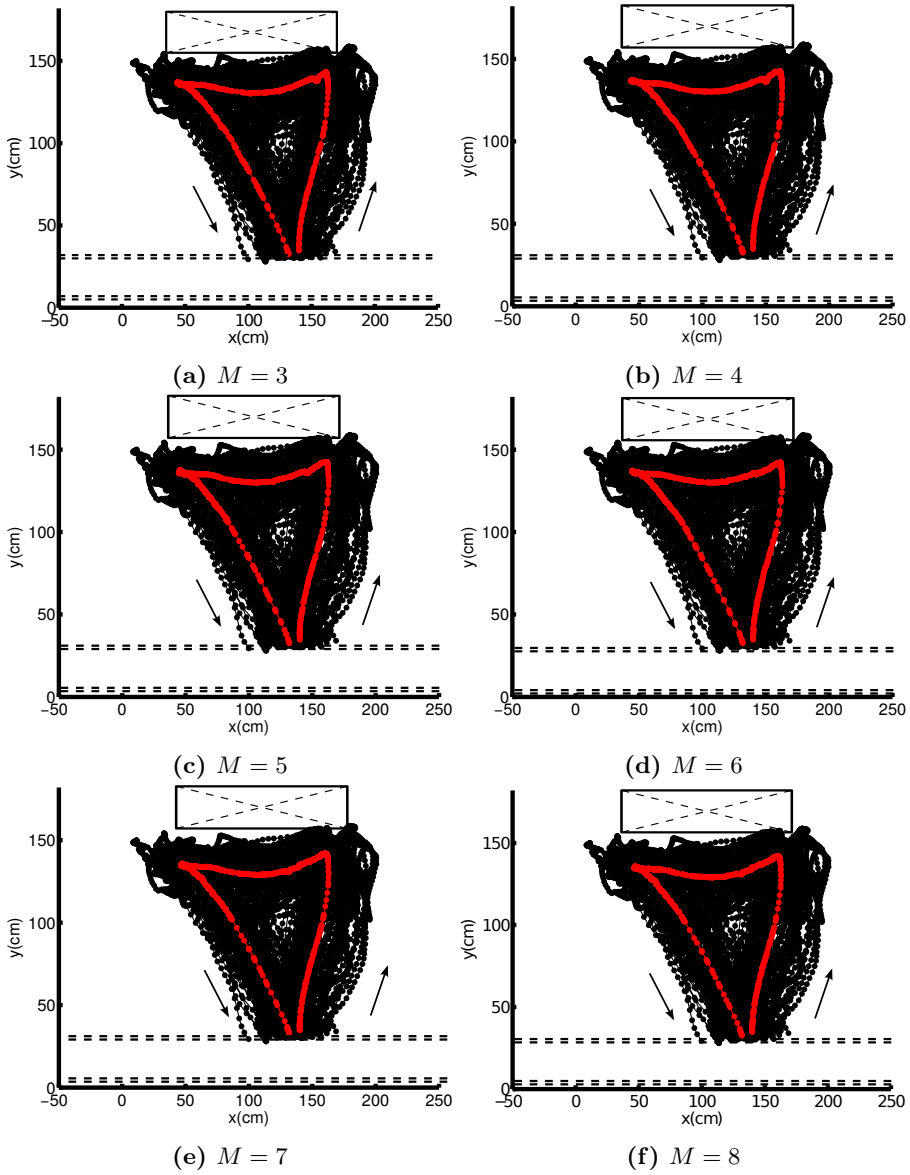
is $21.93^2$ cm for work cycle 1, and $7.60^2$ cm for work cycle 2. It also proves the better performance of the stretch-and-then-compress method in aligning trajectories with high similarity for the second type of executed work cycle.

#### 3.7.1.2 Greedy Method based on Successor Classification

Using this greedy method, the successors of the current path element are classified into $M$ groups at each step of the warp path, resulting in $2^M - 1$ candidate cells for the next path element. To analyze the influence of the category number of the successors on the average trajectory extraction, we test our greedy method on both factory worker trajectories and vehicles trajectories.

We first test our greedy method on the 82 trajectories of factory workers for the first type of executed work cycle, with $M$ from 3 to 8. Factory worker trajectories are recorded using a multi-camera system at the same frame rate of 20 fps. Because the workers do not execute the work cycle in exactly the same way, the recorded trajectories contain different number of data points, from 163 to 616 (average: 260). These trajectories are aligned using the greedy method, and the points associations established are used to extract the average trajectory. Fig. 3.18 shows the average trajectories using red lines with dots. We can see that the physical locations of the points of the average trajectory do not change much as $M$ increases.

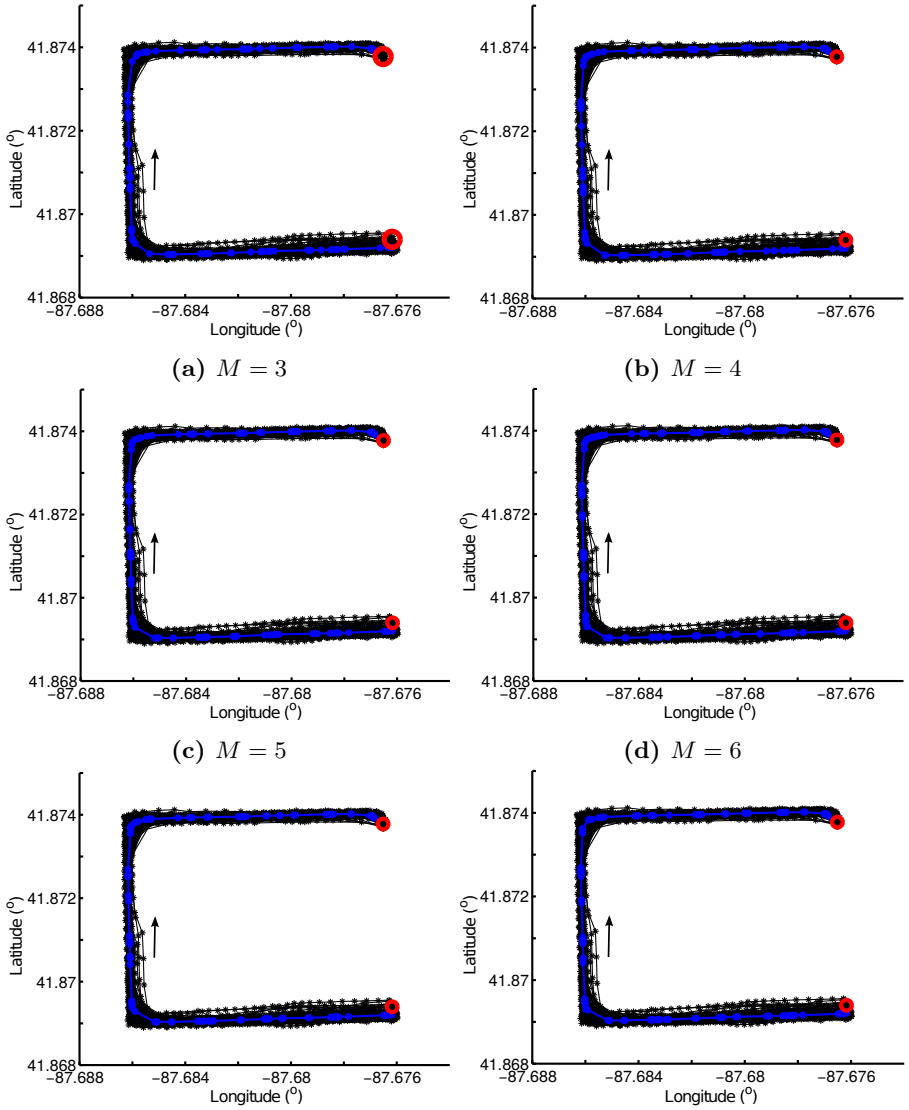
We also test our method on 113 GPS trajectories with different sampling rates from every 1 second to every 10 seconds. The number of data points in the GPS trajectories ranges from 11 to 92 (average: 33). The trajectories are averaged through jointly trajectory alignment using our greedy method as well. As shown in Fig. 3.19, the average trajectory does not change much spatially as $M$ increases from 3 to 8.




**(a)** $M = 3$ **(b)** $M = 4$ **(c)** $M = 5$ **(d)** $M = 6$ **(e)** $M = 7$ **(f)** $M = 8$

**Figure 3.18:** *Average factory worker trajectories with different $M$.* The physical locations of the points of the average trajectory do not change much as the number of successor classes increases from 3 to 8.

Although the physical locations do not change significantly, the computational cost gets higher, and more points are produced in the warped trajectories, as the category number of the successors gets bigger. As shown in Fig. 3.20, it takes 16.22 seconds to align the worker trajectories in Matlab 8.3 on a 2.0 Ghz



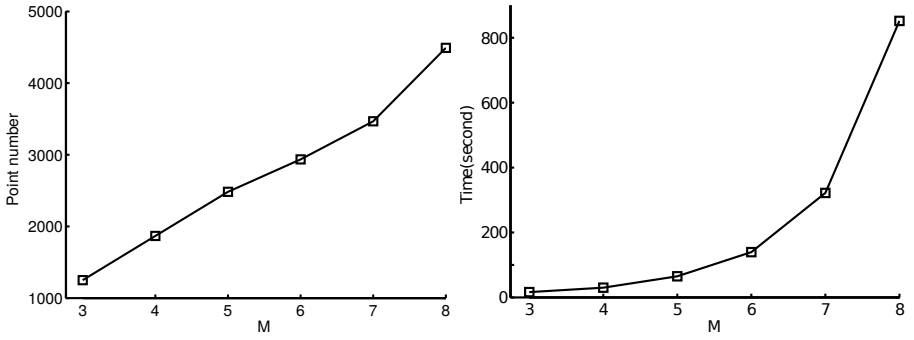

**(a)** $M = 3$ **(b)** $M = 4$

**(c)** $M = 5$ **(d)** $M = 6$

**(e)** $M = 7$ **(f)** $M = 8$

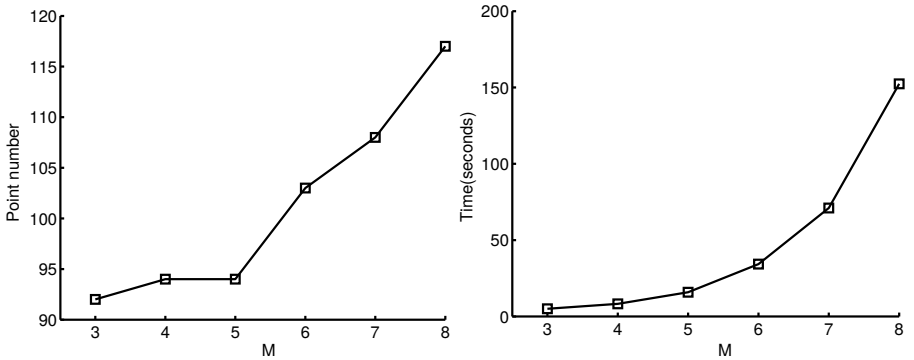
**Figure 3.19:** *Average GPS trajectories with different $M$.* 113 GPS trajectories are aligned using the greedy method with $M$ from 3 to 8, producing the average trajectories as indicated using red lines with markers.

machine with 4 GB RAM, producing an average trajectory of 1250 points, if the successors are classified into 3 groups. However, the computational time rapidly increases to 852.01 seconds, and the length of the average trajectory increases to 4492 points, for classifying the successors to 8 groups. Fig. 3.21




**(a)** The length of the warped trajectories. **(b)** Computational time.

**Figure 3.20:** *Length of the warped worker trajectories and computation time with different $M$.* As classifying the successors of the current path element into more groups, more points are produced in the warped trajectories, and the computational time increases enormously.



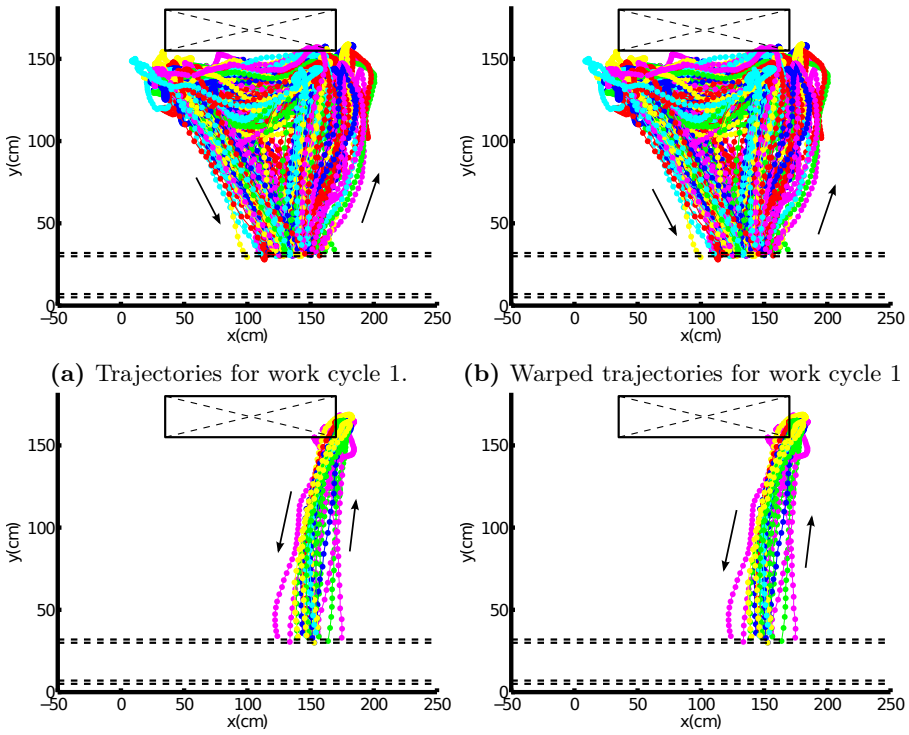

**(a)** The length of the warped trajectories. **(b)** Computational time.

**Figure 3.21:** *Length of the warped GPS trajectories and computation time with different $M$.* The computational time and length of the average trajectory increase as $M$ gets larger.

shows the results for the GPS trajectories. Both the number of the points in the average trajectory and the time spent on computing the average trajectory increases as $M$ gets larger.

Since increasing the category number of the successors $M$ does not change the physical locations in the average trajectory but only causes a rise in the computational time, we choose $M$ as 3 in this work, which forms a good alignment with low computational cost.

Fig. 3.22b and Fig. 3.22d depict the warped trajectories produced using the greedy method based on successor classification, for each type of executed work cycle separately. Compared to the original trajectories in Fig. 3.22a and



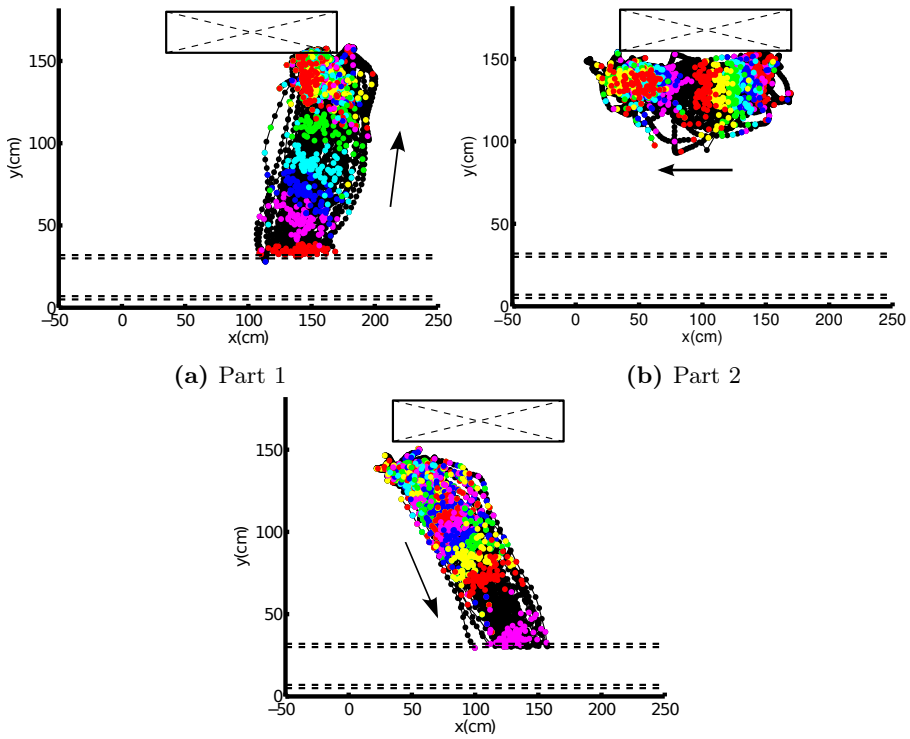

**(a)** Trajectories for work cycle 1. **(b)** Warped trajectories for work cycle 1

**(c)** Trajectories for work cycle 2. **(d)** Warped trajectories for work cycle 2

**Figure 3.22:** *Warped trajectories of factory workers using the greedy method based on successor classification.* (a) and (b) show the original trajectories and the warped trajectories separately for the first type of executed work cycle. (c) and (d) are for for the second type of executed work cycle. Because the greedy method based on successor classification only builds *many-to-one* point correspondences without creating any new points, there are no positional difference between the physical locations of the warped trajectories and those of the original trajectories.

3.22c, there are more points in the warped trajectories due to the *many-to-one* point correspondences. Each of the warped trajectories for work cycle 1, as shown in Fig. 3.22b, contains 1250 points. There are 312 points in each of warped trajectories for work cycle 2, as shown in Fig. 3.22d. Although the warped trajectories are longer than the original trajectories, the positional appearance does not change because this method does not create any new point, but duplicates points at the same position along the warp path.

Fig. 3.23 depicts the point associations of the trajectories for work cycle 1 in three parts, as in did in Fig. 3.23 using the stretch-and-then-compress method. The points at the same time index of the warped trajectory, $\mathbf{g}_n(k)$, $n = 1, \ldots N$, are associated to each other. Fig. 3.23 shows the associated points using dots of the same color. Red, blue, magenta, cyan, green and yellow are




**(a)** Part 1 **(b)** Part 2

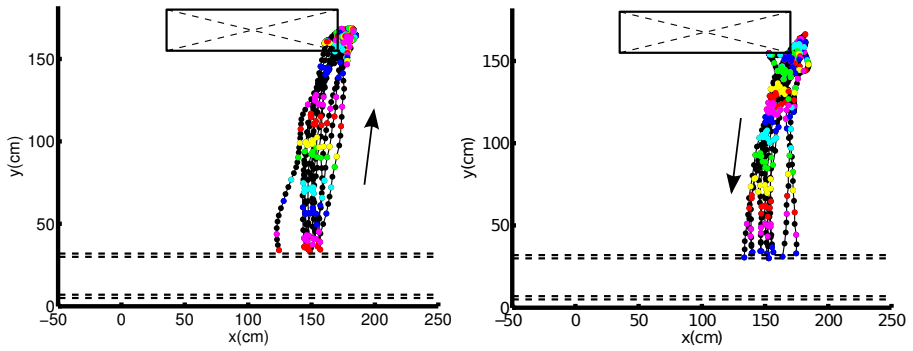
**(c)** Part 3

**Figure 3.23:** *Point associations of the trajectories for work cycle* 1 *using the greedy method based on successor classification.* The associated points at the same index $k$ of the warped trajectories, $\mathbf{g}_n(k)$, $n = 1, \ldots N$ are indicated using dots of the same color. Only part of the point associations are shown for clarity.

used iteratively to show the point associations along the warp path.

Because of the *many-to-one* correspondences, the adjacent points in the warped trajectories may share the same physical location. It is impossible to depict the points associated by every element of the warp path clearly. Fig. 3.23 shows the associated points at some of the time indexes $k$ of the warped trajectories. In Fig. 3.23a for Part 1 of work cycle 1, we show one out of every 15 points in each warped trajectory, $\mathbf{g}_n(k)$, $n = 1, \ldots N$, $k = 1, 16, \ldots \lfloor \frac{K}{15} \rfloor$. We show the associated points of the warped trajectories at time indexes $k = 1, 31, \ldots \lfloor \frac{K}{30} \rfloor$ in Fig. 3.23b for Part 2 of work cycle 1, and the associated points at time indexes $k = 1, 16, \ldots \lfloor \frac{K}{15} \rfloor$ in Fig. 3.23b for Part 3 of work cycle 1.

Fig. 3.24 shows the point associations of the trajectories for work cycle 2 in two parts, as did in the Fig. 3.16 using the Stretch-and-then-compress method. For visual clarity of the point associations, we show one out of every 5 points in each warped trajectory, $\mathbf{g}_n(k)$, $n = 1, \ldots N$, $k = 1, 6, \ldots \lfloor \frac{K}{5} \rfloor$. The points at




**(a)** Part 1 **(b)** Part 2

**Figure 3.24:** *Point associations of the trajectories for work cycle* 2 *using the greedy method based on successor classification.* The points of the same color are associated to each other. Only part of the associations are shown for clarity.

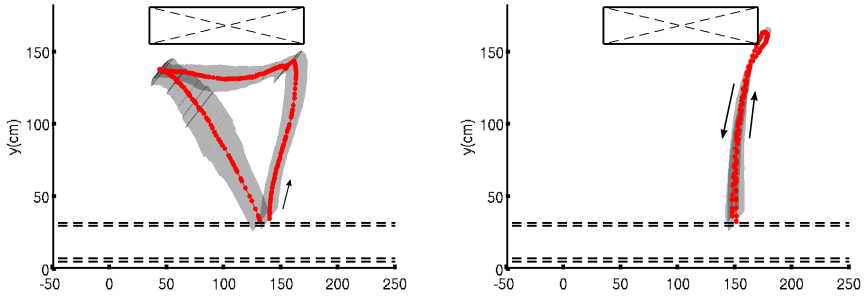
the same time index of the warped trajectories, are also indicated using dots of the same color.

Adjacent points in one warped trajectory are depicted as dots in different colors. However the adjacent points in the warped trajectory may be duplicated from the same point in the original trajectory because of the *many-to-one* correspondences. In Fig. 3.23 and 3.24, the color of the dot is renewed by the point of the same location at higher time index in the warp trajectory. Therefore, the dots in different colors may mix together, especially at the left and right corners of the storage rack where the workers stay statistic to pick up tools and parts.

Compared to the point associations in Fig. 3.15, which is established using the Stretch-and-then-compress method, the associated points can be identified by their colors easier in Fig. 3.16, which indicates better point associations built by the greedy method based on successor classification.

In Fig. 3.24, we can clearly see the iterative uses of the seven colors in indicating associated points, as we do in Fig. 3.16. It indicates that both proposed methods have good performance of aligning the trajectories with high similarity for work cycle 2.

Fig. 3.25 depicts the average trajectory $\mathbf{g}_0(k)$, $k = 1, \ldots K$ and the physical location variance $\sigma_g(k)$, $k = 1, \ldots K$ along the average trajectory for each type of executed work cycle, respectively. The width of the band indicates the intensity of the physical location variance. The wider the band is, the less similarity the associated points and vice versa. The mean variance of the physical locations $\sigma_g$ is $13.22^2$ cm for work cycle 1 using the greedy method based on successor classification, which is one thirds lower than mean variance $21.93^2$ cm using the stretch-and-then-compress method. The band in Fig. 3.25a is thinner than that in Fig. 3.17a for the same trajectories, which means that the associated points are more similar to each other. It proves the robust-



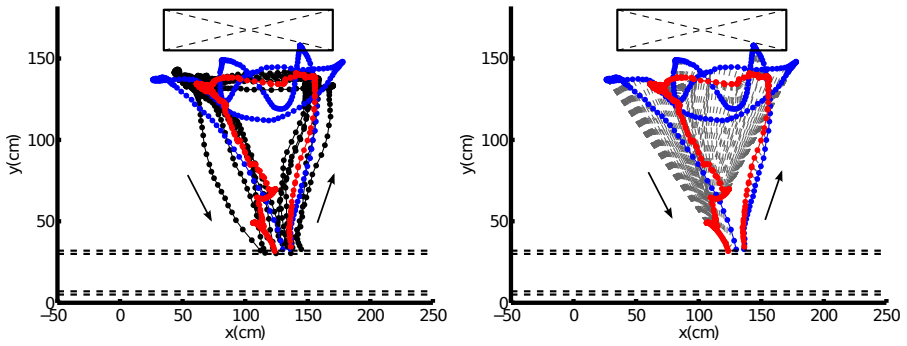

**(a)** Work cycle 1 **(b)** Work cycle 2

**Figure 3.25:** *Physical location variance along the average trajectory extracted using the greedy method based on successor classification.* (a) and (b) depict the average trajectory in red lines with dots, and the physical location variance using a gray band along the average trajectory for each type of executed work cycle, respectively.

ness of the greedy method based on successor classification in aligning highly dissimilar trajectories. For work cycle 1, the mean variance is $6.79^2$ cm in Fig. 3.25b, which is also lower than the mean variance $7.60^2$ cm obtained from the stretch-and-then-compress alignment. It proves that the greedy method based on successor classification establishes better point associations than the stretch-and-then-compress method.

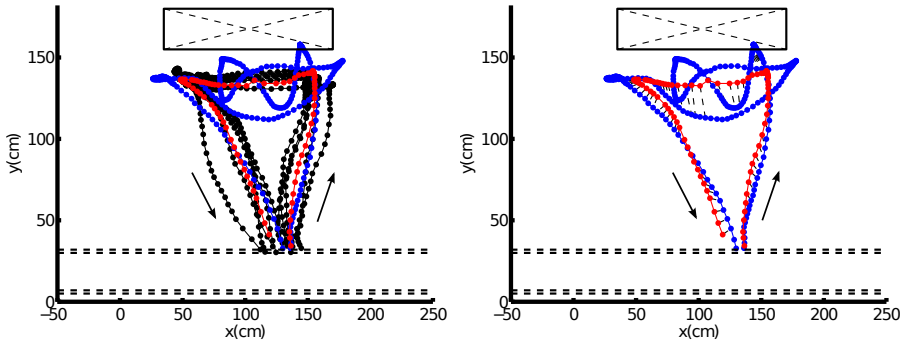
However, the greedy method based on successor classification produces warped trajectories longer than necessary. There are 1250 points in each of the warped trajectories for work cycle 1, and 312 points for work cycle 2. Along some adjacent elements of the warp path $(w_1(k), \ldots w_N(k))$ and $(w_1(k+1), \ldots w_N(k+1))$, only a few trajectories increase time index $j_n(k) = 1$, leading to a lot of points duplicated at the $k$ and $k+1$ step of the warp path. As a result, two adjacent points in the average trajectory may locate at very similar locations. In the future, we will try to increase time index in more trajectories if the dissimilarity measure does not change much, when the warp path advances to the next cell.

This greedy method proceeds the warp path at each step purely based on the current cell and its successors. One abnormal step with very dissimilar corresponding points could result in big errors at all of the following steps. However, there is no error found during the trajectory alignment for any type of work cycles, because the abnormal trajectories have been already removed through trajectory clustering, and are not aligned with the normal trajectories. To analyze the shortcomings of the greedy method, we manually select five normal trajectories and one abnormal trajectory for work cycle 1, as show in Fig. 3.26. This abnormal trajectory is generated by the users walking from the right corner to the left corner of the storage rack, revisiting the right corner, and walking to the left corner again. These six trajectories are aligned using




**(a)** Average track **(b)** Point correspondences

**Figure 3.26:** *Dissimilar trajectory alignment using the greedy method based on successor classification.* (a) shows 5 of the normal trajectories for work cycle 1 in Fig. 2.7a using black lines with dots, 1 of abnormal trajectories in Fig. 2.7d using blue lines with dots, and the average trajectory extracted from these six trajectories using red lines with dots. (b) shows the point correspondences between the abnormal trajectory and the average trajectory.




**(a)** Average track **(b)** Point correspondences

**Figure 3.27:** *Dissimilar trajectory alignment using the stretch-and-then-compress method.* (a) shows the extracted average trajectory using red lines with dots. (b) shows the point correspondences between the abnormal trajectory and the average trajectory.

the greedy method, and the warped trajectories produced are averaged directly.

Fig. 3.26a shows the average trajectory using red lines with dots. We can see that the average trajectory is noisy on the worker's way back to the assembly platform. Because it is difficult to visually show the point correspondences of all 6 trajectories at the part with big errors, we only illustrate the point correspondences between th abnormal trajectory and the average trajectory in Fig. 3.26b. The matched points are linked using dashed lines. In the normal trajectories, the worker only walks through the area in front of the storage

rack once from the right corner to the left corner. However, the worker walks through this area three times from right to left, left to right, and right back to left again in the abnormal trajectory. The points generated by the worker walking in front of the storage rack in the normal trajectories are matched to the points produced by the worker walking from the right corner to the left corner for the first time in the abnormal trajectory. Because the greedy method keeps the points of each warped trajectories in the same relative order within the original trajectory, the points generated by the worker revisiting the right corner and walking back to the left corner again in the abnormal trajectory are matched to the points generated on the worker's way back to the assembly platform in the normal trajectories, producing a windy average trajectory.

We also test our stretch-and-then-compress method on the same six trajectories, and show the results in Fig. 3.27. Because the *compress* operation keeps pairs of points with the smallest dissimilarity, the points generated by the worker revisiting the right corner and walking back to the left corner again are removed, and not used to produce the average trajectory. Therefore the extracted average trajectory is more smooth than the one in Fig. 3.26. It proves that the stretch-and-then-compress method is more robust to the special case that the trajectory is generated by the user going back to the location he or she visits earlier during the journey.

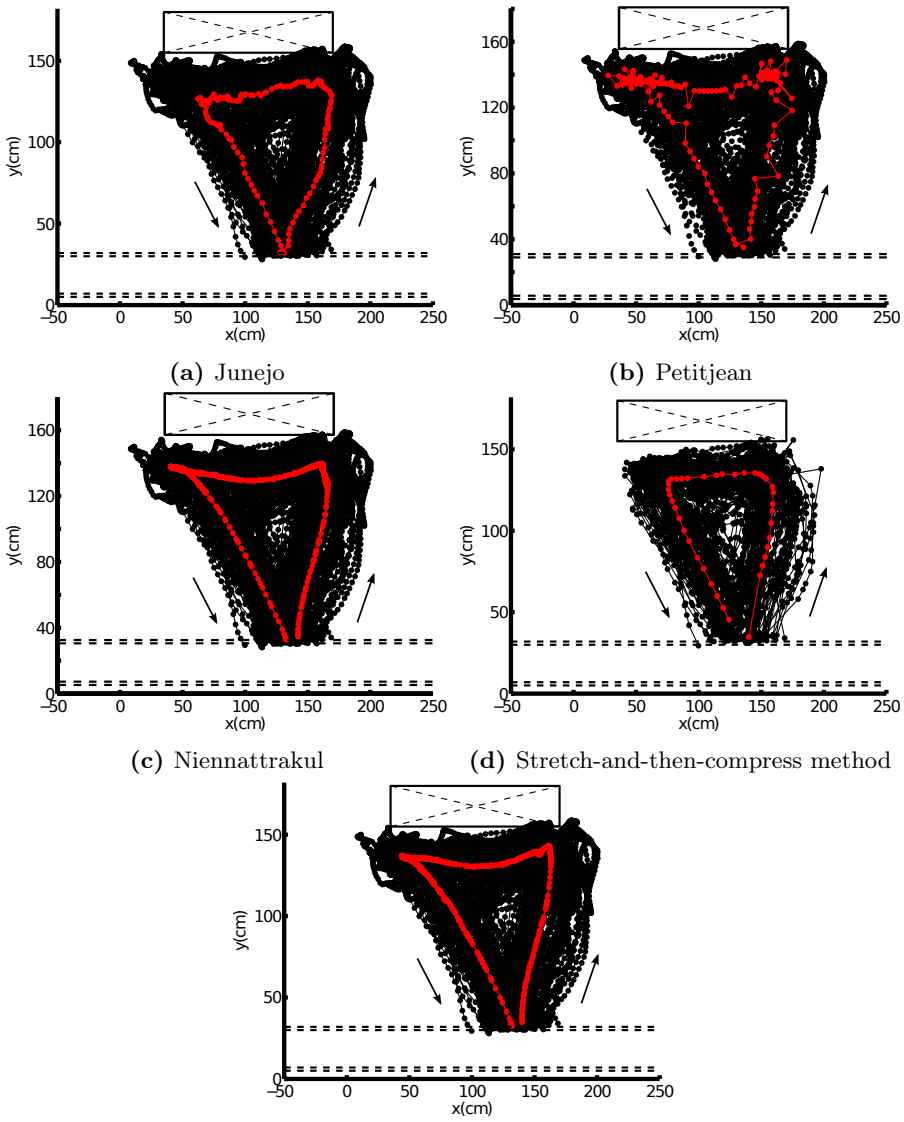
#### 3.7.1.3 Average Trajectory Comparison

Considering other DTW-related methods mentioned in Section 3.1 are designed to average the trajectories without building such point associations, we will compare our proposed methods with them on this point on both factory worker trajectories and vehicle trajectories.

Fig. 3.28 shows average trajectories for the first type of executed work cycle using the methods mentioned in Section 3.1, and both of our proposed methods. Fig. 3.29 is for the second type of executed work cycle. The original trajectories are depicted using black lines with dots The average trajectories are indicated using red lines with dots. The dots indicate the locations of the points on the trajectories.

In our application of work cycle optimization, the average trajectories are extracted as prototypical routes, which can be used to guide the workers to execute their work cycles efficiently. A prototypical route for each type of work cycle is defined as a route representing how the workers spatially travel to the storage rack, pick up tools and parts and travel back to the assembly platform, with considering all of their trajectories.

Junejo extracts the average trajectories from the inner and outer boundaries of the trajectories [Junejo 07], as shown in Fig. 3.28a and 3.29a. The executed average trajectory is unrelated to the spatial distribution of the points of the trajectories. Ignoring the trajectory points between these two boundaries leads to the average trajectories not representative. Therefore they are not suitable to be used as prototypical routes for the workers.




**(a)** Junejo

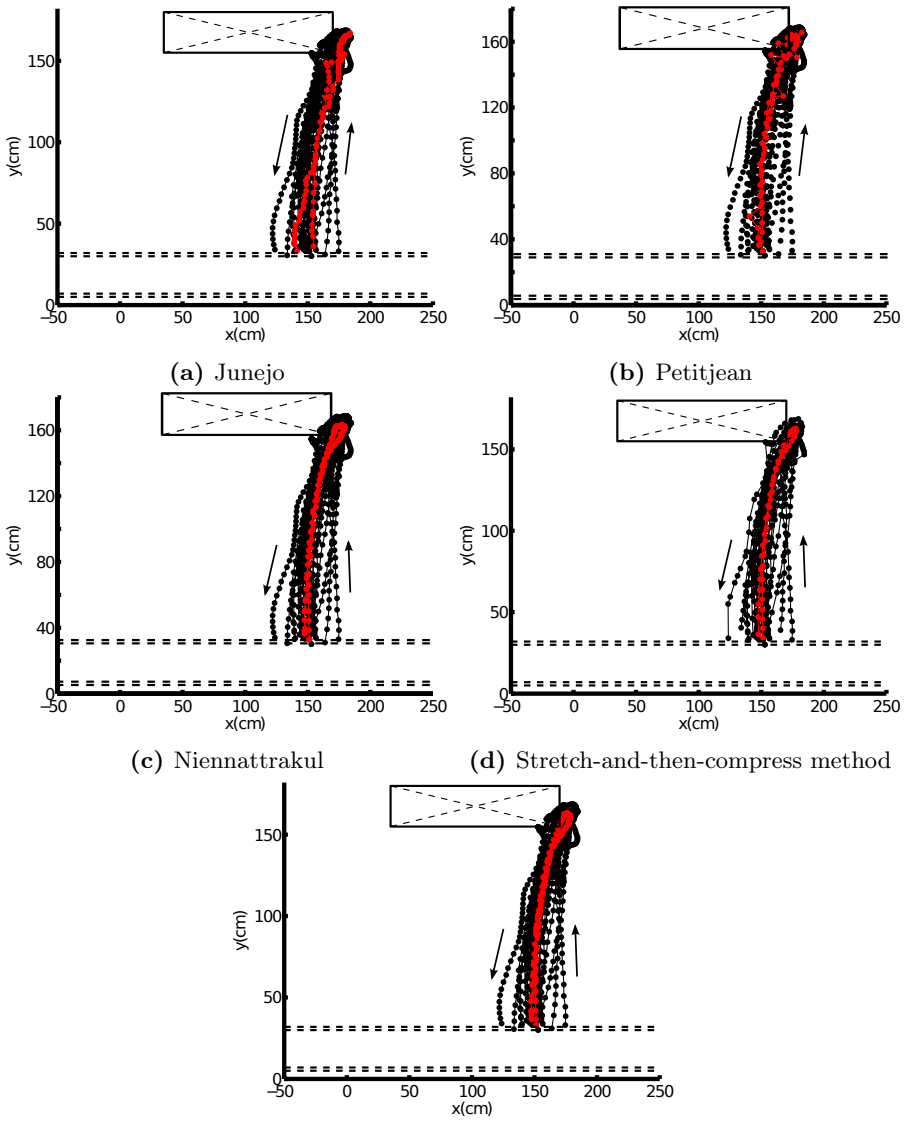
**(b)** Petitjean

**(c)** Niennattrakul

**(d)** Stretch-and-then-compress method

**(e)** Greedy method based on successor classification

**Figure 3.28:** *Results of average trajectory extraction for work cycle* 1.




**(a)** Junejo

**(b)** Petitjean

**(c)** Niennattrakul

**(d)** Stretch-and-then-compress method

**(e)** Greedy method based on successor classification

**Figure 3.29:** *Results of average trajectory extraction for work cycle* 2.

The average trajectories extracted using Petitjean's DBA method [Petitjean 11] are very noisy and jumpy, as shown in Fig. 3.28b and 3.29b. In reality, they cannot be used as prototypical routes because the workers will spend more time on the windy routes, and miss the next part to be assembled on the assembly line. The smoothness of route is important in path planning [Kanayama 89] and the windy routes will make the worker less efficient.

As shown in Fig. 3.28c and 3.29c, Niennattrakul's method based on shape averaging extracts more representative and smooth average trajectories, which can be used as prototypical routes to guide the workers for their journey of picking up tools and parts. The main downside of this method is that computational cost gets higher very quickly as aligning more trajectories.

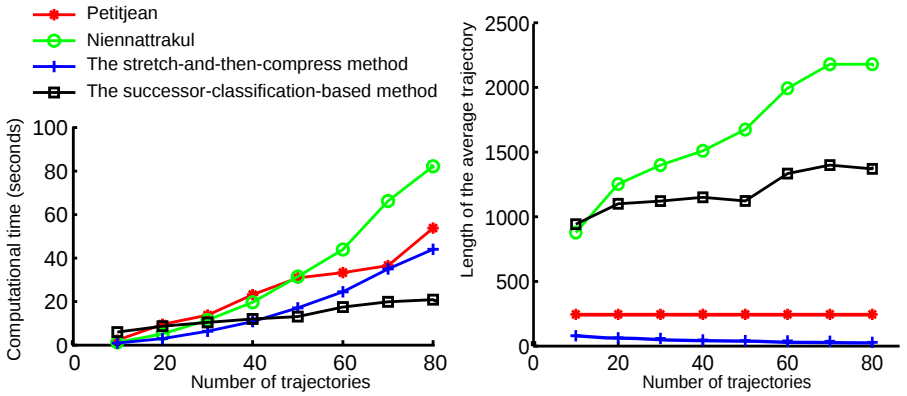
Fig. 3.28d and 3.29d show the average trajectories extracted by aligning the trajectories using our stretch-and-then-compress method, for each type of work cycle respectively. The *compress* operation keeps only pairs of points which are more similar to each other, therefore some information is lost during the trajectory alignment, especially at the area where the physical locations of the points are more diverse, for instance, the left and right corners of the storage rack in Fig. 3.28d. The workers will not reach the left and right corners of the storage rack if the average trajectory in Fig. 3.28d is used as the prototypical route. The trajectories for work cycle 2 are more similar to each other, thus the *compress* operation does not remove many points. The average trajectory extracted for work cycle 2 is more representative, as shown in Fig. 3.29d.

Fig. 3.28e and 3.29e depict the average trajectories produced by joint aligning the trajectories using our greedy method based on successor classification. The average trajectories extracted for both types of work cycles are smooth and representative. Different from the stretch-and-then-compress method, the greedy method establishes *many-to-one* point correspondences without removing any points. There is no information lost in the warped trajectories. The average trajectories are calculated using full information of the original trajectories, they are therefore more representative than those extracted using the stretch-and-then-compress method.

Establishing the time correspondences among the points of the trajectories using our proposed methods not only benefits calculating the average trajectory with a better spatial representation of the trajectories, but also help to calculate the average speed, speed variance and location variance along the average trajectory. The location variances for each method have been shown in Fig. 3.17 and 3.25. The average speed and speed variance will be discussed in Chapter 5 for work cycle optimization. The other methods mentioned in Section 3.1 are only designed to extract the average trajectory, providing no support for statistical analysis.

#### 3.7.1.4 Computation time analysis

We implemented and tested our algorithm in MATLAB 8.3 on a 2.0 Ghz machine with 4 GB RAM. Fig. 3.30 shows the computation time in seconds and the length of the average trajectory (the number of the data points in the av-




**(a)** Computational time **(b)** Length of the average trajectory

**Figure 3.30:** *Computational time with different number of trajectories.* As aligning more trajectories, the computational time increases for all methods, and the length of the average trajectory increases using Petitjean's method and our greedy method.

erage trajectory), as aligning more trajectories for work cycle 1 as shown in Fig. 3.14a.

The results using Petitjean's method [Petitjean 11], Niennattrakul's method [Niennattrakul 09], and each of our proposed methods are depicted in red star, green circle, blue cross and black square separately. As shown in Fig. 3.30a, the computational time increases for all of the aforementioned methods as aligning more trajectories. But the computation time using our greedy method based on successor classification does not increase extraordinarily as other methods do. It takes 20.61 seconds using the greedy method based on successor classification, 43.55 seconds using the stretch-and-then-compress method, 56.33 seconds using Petitjean's method and 82.66 seconds using Niennattrakul's method to align 80 trajectories. This shows that our greedy method is more scalable on larger dataset than other methods.

Fig. 3.30b shows the length of the average trajectory. Using Petitjean's method, the number of data points in the average trajectory stays the same as 245, no matter how many trajectories are aligned. As aligning more trajectories, the number of the data points in the average trajectory increases gradually using both Niennattrakul's method and our greedy method based on successor classification, but decreases using our stretch-and-then-compress method because of the *compression* operation.

Factory worker trajectories in our work are very different from those used in Junejo's work. Junejo applied image processing technique to get the boundaries from all of the tracks, so as to extract the road path from the boundaries. However our trajectories are recorded during the work cycles of the factory workers going forth and back between the assembly platform and the storage rack. As aligning more trajectories, it is more difficult to find the inner bound-

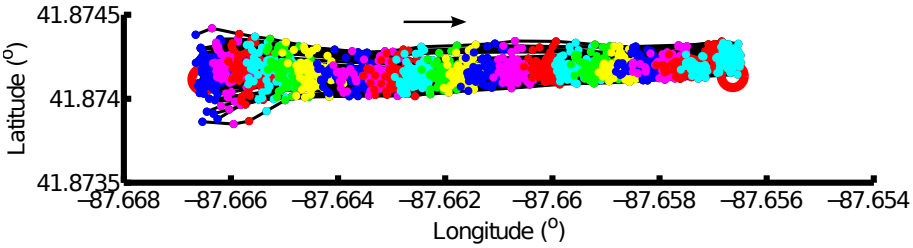



**Figure 3.31:** *Point associations of the warped GPS trajectories on one road segment using the stretch-and-then-compress method.* The associated points at the same index $k$ of the warped trajectories, $\mathbf{g}_n(k)$, $n = 1, \ldots N$, are indicated using dots of the same color.

ary automatically using image processing because the trajectories of going to the storage rack and coming back from there are overlapped. In order to apply their algorithm on factory worker trajectories, the inner boundary obtained needs to be corrected manually. Therefore, their computation time in terms of the number of trajectories is not shown in Fig. 3.30.

## 3.7.2 Results of Chicago Data Set

In this section, we show the trajectory alignment results for one road segment of the Chicago data set. We will first show the results using the stretch-and-then-compress method in Section 3.7.2.1, then the results using the greedy method based on successor classification in Section 3.7.2.2.

### 3.7.2.1 Stretch-and-then-Compress Method

Fig. 3.31 depicts the point associations of the warped GPS trajectories on one road segment, which are produced using the stretch-and-then-compress method. The road segment is directed from the left to the right intersection, as shown in red circles. The associated points, i.e. the points at the same time index $k$ of the warped trajectories, $\mathbf{g}_n(k)$, $n = 1, \ldots N$, $k = 1, \ldots K$, are indicated using dots of the same color. Red, blue, magenta, cyan, green and yellow are used iteratively to show the point associations at time indexes $k = 1, \ldots K$.

From Fig. 3.31, we can identify the associated points easily by their colors, although a few points in different colors mix together. We can clearly see iterative use of the seven colors in indicating associated points. This indicates the stretch-and-then-compress method has a a good performance in aligning *many* GPS trajectories.

The original trajectories shown in Fig. 3.13 contain between 27 to 32 points (30 on average). The warped trajectories produced using the stretch-and-then-compress method contain 22 points, respectively. The points at the same time index of the warped trajectories are averaged to create a geometric represen-

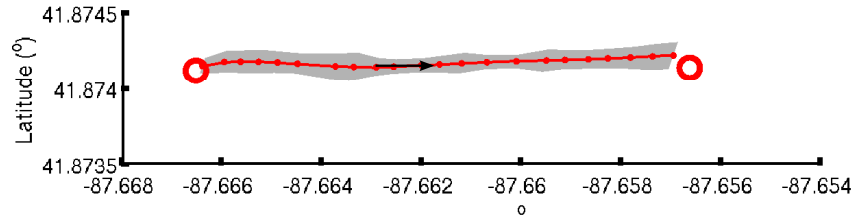



**Figure 3.32:** *Location variance along the average trajectory extracted using the stretch-and-then-compress method.* The average trajectory is shown in red lines with dots indicating the locations of the points of the average trajectory. The width of the gray brand along the average trajectory indicates the location variance.

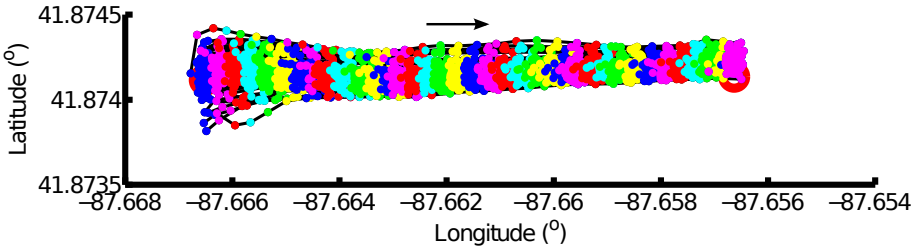



**Figure 3.33:** *Point associations of the warped GPS trajectories on one road segment using the greedy method based on successor classification.* The associated points at the same index $k$ of the warped trajectories, $\mathbf{g}_n(k)$, $n = 1, \ldots N$, are indicated using dots of the same color.

tation of the road segment, $\mathbf{g}_0(k)$, $k = 1, \ldots 22$, as shown in Fig. 3.32. The average trajectory extracted using this alignment method locates in the middle of the physical locations of the warped trajectories, along the road direction from the left to the right intersection. The location variance of the GPS points of the trajectories is shown using a gray band along the average trajectory. The width of the band indicates the intensity of the location variance. The mean variance of the physical locations of the points in the warped trajectories along the average trajectory is $(2.35 \times 10^{-4})^2$ degrees in longitude, and $(6.29 \times 10^{-5})^2$ degrees in latitude. Compared to the longitude range of 0.01 degrees and the latitude range of 0.001 degrees, the mean variances are very small, which shows a good geometric representation for the road segment.

#### 3.7.2.2 Greedy Method based on Successor Classification

Fig. 3.33 depicts the point assignations for the same GPS trajectories as shown in Fig. 3.13, which is produced using the greedy method based on successor classification. The points at the same time index $k$ of the warped trajectories $\mathbf{g}_n(k)$, $n = 1, \ldots N$, $k = 1, \ldots K$ are associated to each other, i.e. spatially close to each other. The associated points are indicated using dots of the same

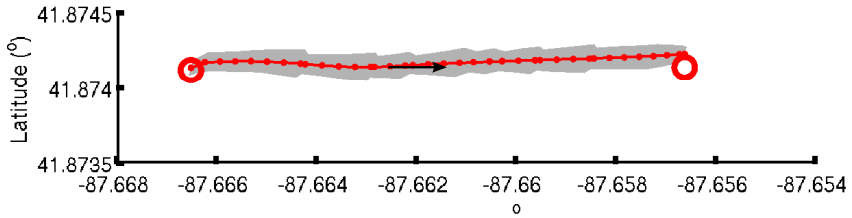



**Figure 3.34:** *Location variance along the average trajectory extracted using the greedy method based on successor classification.* The average trajectory is shown in red lines with dots, and the location variance in a gray brand along the average trajectory.

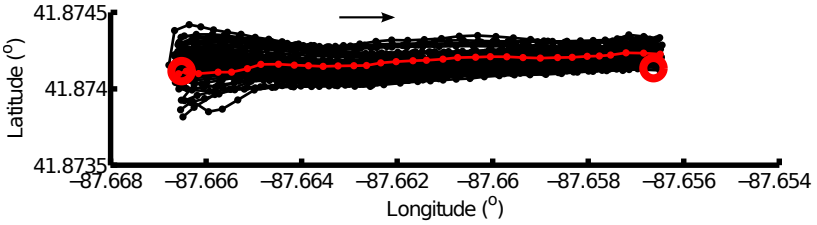
color. Red, blue, magenta, cyan, green and yellow are used iteratively to show the point associations along the warp path. The associated points can be easily identified by their colors in Fig. 3.33. Points of the same color gather at similar physical locations, which proves good point associations.

Along the warp path, all of the trajectories are warped to 38 points. Although the warped trajectories are longer than any of the original trajectories, there is no positional difference between the warped trajectories and the original trajectories, since the greedy method only duplicates points of the original trajectories rather than creating new points. A geometric of representation of the directed road segment $\mathbf{g}_0(k)$, $k = 1, \ldots 38$ is extracted by averaging the points of the warped trajectories at the same time index, as shown in Fig. 3.34. The width of the gray band along the average trajectory indicates the location variance of the GPS points of the trajectories. The mean variance of the point locations of the warped trajectories along the average trajectory is $(1.69 \times 10^{-4})^2$ degrees in longitude, and $(6.02 \times 10^{-5})^2$ degrees in latitude, which are slightly smaller than the mean variance of the point locations of the warped trajectories produced by the stretch-and-then-compress method. It shows that our greedy method also builds a good alignment for *many* GPS trajectories.

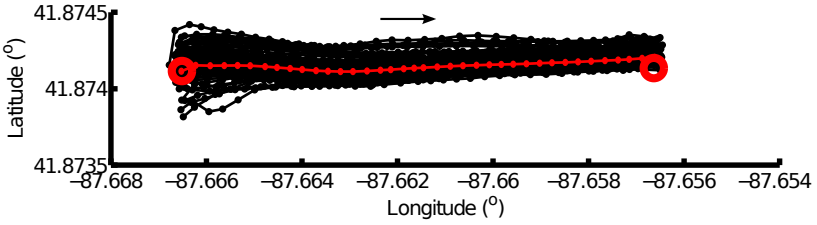
#### 3.7.2.3 Average Trajectory Comparison

We extract the average trajectory from 79 GPS trajectories using other DTW-related methods mentioned in Section 3.1, and both of our proposed methods. Fig. 3.35 shows the average trajectories in red lines with dots. The dots indicates the locations of the points of the average trajectories. The original trajectories are shown in black lines with dots. In all of 79 GPS trajectories, the road users keep moving at a speed of around 35 km/h. None of the adjacent points share the same location.

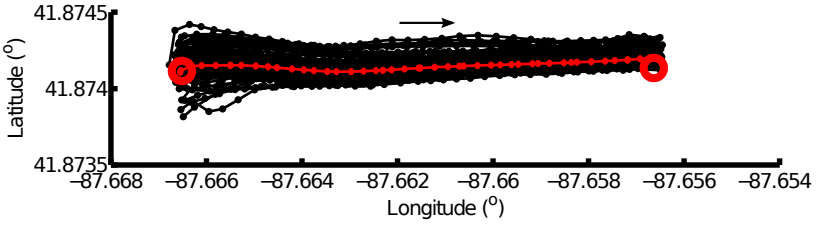
The original trajectories contain between 27 to 32 points (30 on average). The average trajectories extracted using Junejo's, Petitjean's and Niennattrakul's methods contain 34, 38 and 47 points, respectively. There are 22



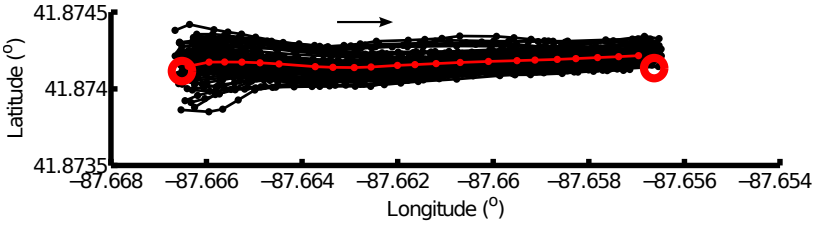

**(a)** Junejo



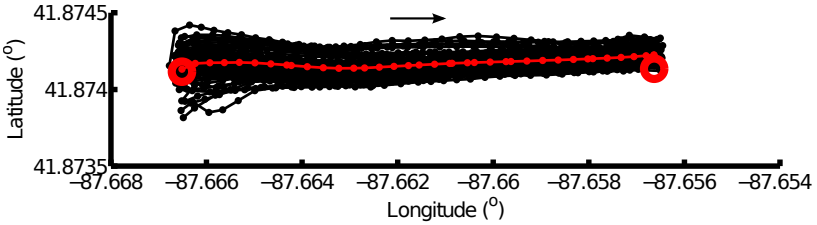

**(b)** Petitjean




**(c)** Niennattrakul




**(d)** Stretch-and-then-compress method




**(e)** Greedy method based on successor classification

**Figure 3.35:** *Results of average trajectory extraction for* 79 *GPS trajectories.*

and 38 points in the average trajectories produced using our stretch-and-then-compress method and our greedy method based on successor classification, respectively. All of the methods show good performance on averaging the highly similar GPS trajectories.

## 3.8 Conclusions

In this chapter, we presented two novel approaches to align *many* trajectories. We apply DTW with a "stretch and then compress" strategy to align the trajectories pairs in the first method. The trajectories are aligned once together using the greedy method based on successor classification. The stretch-and-then-compress method produces *one-to-one* correspondences among the points of the warped trajectories, which are shorter than any of the original trajectories. However, the warped trajectories produced by the greedy method are longer than any of the original trajectories, because of the *many-to-one* point correspondences. By comparing the variance of the physical locations of the associated points in the warped trajectories, the greedy method based on successor classification established better point associations than the stretch-and-then-compress method did.

We extracted the average trajectory using the point associations built by our proposed methods, and compared with other DTW-related methods trajectory averaging on both factory worker and vehicle trajectories. Experimental results show good performance of our alignment methods on averaging trajectories, and the robustness of our greedy method on aligning diverse trajectories.

The joint trajectory alignment not only allows route estimation by averaging the warped trajectories, but also a deeper statistical analysis, such as location variance and speed variance. For Factory data set, the prototypical route and the prototypical speed and dwell time along the route will be inferred using the trajectory alignment in Chapter 5. For Chicago data set, the road network will be generated using the trajectory alignment in Chapter 6.

This research resulted in one publication in International Journal of Geo-Information [Xie 15b]. Furthermore, several papers have been published in the proceedings of the international conferences [Xie 14a], [Xie 14c].

# 4

# Room Layout Exploration from Trajectories

Object recognition has always been one of the important topics in computer vision. However it has not been widely used in ambient intelligence which deals with human behavior analysis and analysis of human interactions with the environment [Peursum 05a]. Typically, researchers encode useful information about the smart environment, such as the size, type and and location of objects of interest and the ground plan of the room manually. Automatically detecting the objects will not only relieve them from this tedious job, but is also an important step to develop real applications: time consuming calibration and configuration is an important cost factor in those applications.

Researchers traditionally employ image features such as color, shape, and texture to classify objects. However, methods based on such features are sensitive to illumination changes, as well as to the view point from which an object is seen and occlusions [Mel 97]. Moreover, these methods need cameras with a high resolution to extract detailed features of objects, also making them computationally expensive [Mel 97, Torralba 10]. This is uneconomic for home and office-based applications.

In this chapter we propose a new method to recognize the presence of objects in a meeting room, such as chairs, table and walking areas, indirectly from low-precision trajectories of people. Rather than relying on detailed image information, our method relies on the long-term integration of trajectory data to learn about the room layout and also tolerates imprecise data. Thus, the presence of objects is inferred from contextual knowledge about the relationship between the objects, room areas and human activities. The resulting information about the objects is useful as statistical prior information for other computer vision tasks (e.g. as feedback about possible walking and sitting areas to a tracking system). It can also enhance the existing direct recognition methods by providing indirect evidence of a very different nature.

A trajectory includes the ground location of all meeting participants and their head height as a function of time. This head height changes as people perform different activities, specifically it gets lower when sitting down. Es-

timating the trajectories is not the topic of this thesis, but such trajectories can be created from low resolution images [Gruenwedel 11a] and moreover in many applications they are computed anyway [Wang 06]. The trajectories used in this chapter are obtained from a multi-camera system developed or the project 'iCocoon" (Immersive COmmunication by means of COmputer visiON) by Image Processing and Interpretation (IPI) research group, in cooperation with the Vision Systems (VIS) research group. The project aims to dramatically change the way people communicate remotely by creating third-generation video conferencing applications. The multi-camera system focuses on *real-time*, *low-latency* and *scalable* tracking of multiple people. More information about the setup, the tracking algorithm and other applications of our system can be found in our earlier papers [Gruenwedel 11b, Jelaca 11, Gruenwedel 12].

We propose to categorize people's instantaneous activities using the speed and height information extracted from the trajectories, and merge these instantaneous activities into higher-level activities spanning longer periods of time. From this higher level activity information we build two occupancy maps, one for sitting space and the other for walking space. Positions of chairs are estimated using the sitting space map. The table is inferred based on the walking space map and chairs positions.

The remainder of this chapter is structured as follows: In the next section, we briefly introduce related work on object recognition. Section 4.2 gives an overview of our approach, and Section 4.3 discusses activity classification. Section 4.4 describes how to build and update the occupancy maps and recognize the objects from the maps. In Section 4.5, we present experiments Section 4.6 concludes the chapter.

## 4.1 Related Work

In the past decades, there has been much progress on functional object recognition through analyzing human activities in the environment. Several researchers have exploited context knowledge and human interaction with the environment as important means to classify objects [Peursum 05a, Veloso 05, Veloso 06, Wu 09, Wu 11, Kjellström 11].

Peursum *et al.* employ a Bayesian network to classify image region patches with object labels by segmenting and recognizing human actions in an office room [Peursum 05b, Peursum 05a]. They firstly generate the skeleton of the single person in the office room by extracting the person's silhouette from video using background segmentation. They then extract ten features from the skeleton, such as leg lenth, torso length and torso angle, and use these features to train Hidden Markov Models (HMMs). Actions are recognized using HMMs. The recognized actions are then used to label specific image patches using the tags "floor," "chair," "keyboard," "printer" and "paper." Although this object recognition method is independent on the object shape, a fine skeleton is needed to get the features of the moving person for action classification.

Veloso *et al.* develop an object recognition algorithm to classify chairs

through both their function and visual features [Veloso 05, Veloso 06]. The color of the chairs is simply chosen as the visual feature. The function of the chairs is defined by how people use them: sitting down on them. They first detect people in the video stream by face recognition, and input the face positions to HMMs to identify people's activities, such as sit still, stand still and fidget right. Chairs are recognized from people's activities and the color feature. Because the activity recognition in this work is highly dependent on the face recognition, which is limited by the face orientation, they are not able to detect all of the chairs in the room.

Wu *et al.* define the relationships between human activities and objects in a knowledge base constructed by Markov Logic Network (MLN) [Wu 09, Wu 11]. They first detect some features related to the user, such as position, height, gaze direction and hand motion. A Conditional Random Field (CRF) model is trained and then used to recognize the following three activities from image features: lying, seated and walking. Finally objects in both living room and kitchen, such as chair, sofa and TV, are recognized from these activities. This work requires high-resolution images to extract some of the features, for instance, gaze direction and gaze area.

In the above mentioned research, image features are needed to recognize activities. However, our approach is purely based on trajectory data, which describes a person's position over time and the persons' current head height. Trajectory data of all people in the room is processed rather than the video itself. Therefore, by "piggy-backing" on person tracking, which is needed for other purposes in smart meeting rooms, our approach is computationally very efficient.

## 4.2 Overview of the proposed approach

Fig. 4.1 shows the flowchart of our algorithm. First, instantaneous speeds are computed from people's trajectories. Instantaneous speed and head height are the features input to the SVMs to categorize instantaneous activity, resulting in the instantaneous activity state of each person $z_k(t)$, $t = 1 \dots T$, $k = 1, \dots K$ as a function of time $t$. The instantaneous sitting activities of all people are used as observations to estimate the probability of each cell on the ground plane being sit using Bayesian theory, so as to create an occupancy map for the sitting space, and the instantaneous walking activities are used to create an occupancy map for the walking space .

Regions with high probability in the sitting space map are recognized as candidate chairs, and false detections of chairs due to a very short period of *sitting* activity will be removed. We are not able to distinguish between real chairs and other objects on which people can sit (like tables), because a chair (in the wide sense, i.e. including couches) is defined rather by its function than its color or shape. We aim to detect areas where people perform "sitting" activities, no matter whether there are real chairs at these areas or not.

Our approach to detect tables is based on three key assumptions:

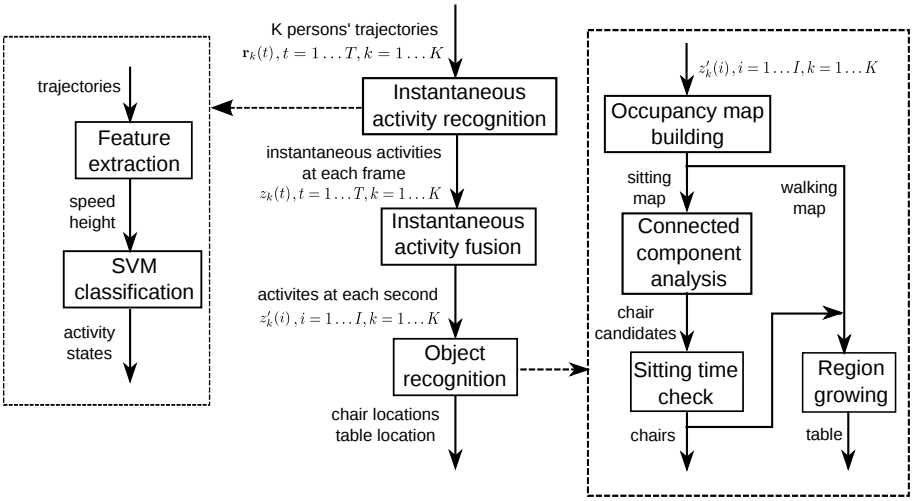



**Figure 4.1:** *Flowchart of the proposed method.*

- A table is always considered as a rectangular in shape.
- A table is located in the interior of a walking band and bounded by the surrounding chairs.
- A table cannot be walked through or over by people.

With these assumptions, the largest rectangle surrounded by walking space and chairs is considered as the table, and it is adjusted according to the locations of the chairs. But these assumptions lead to some possible limitations. The shape of a table is not within consideration, and empty space surrounded by chairs will be recognized as a table.

## 4.3 Activity Classification

We define three kinds of human activities: *sitting*, *standing*, and *walking*. Detecting sitting allows us to detect the location of chairs in the room, assuming that the chairs are the objects people usually sit on. Next, based on walking activity and the detected chairs, a table is indirectly recognized. Although standing is not directly used to identify the location of chairs, tables or other objects, it still needs to be detected in order to distinguish this from walking. For instance, standing is typically the last "activity" just before sitting.

These three activities can be distinguished from each other by the person's height and moving speed. When a person sits, his or her speed and height tends to be very low. High height and speed indicates *walking* activity, and high height and low speed are for *standing* activity. We employ a SVM classifier

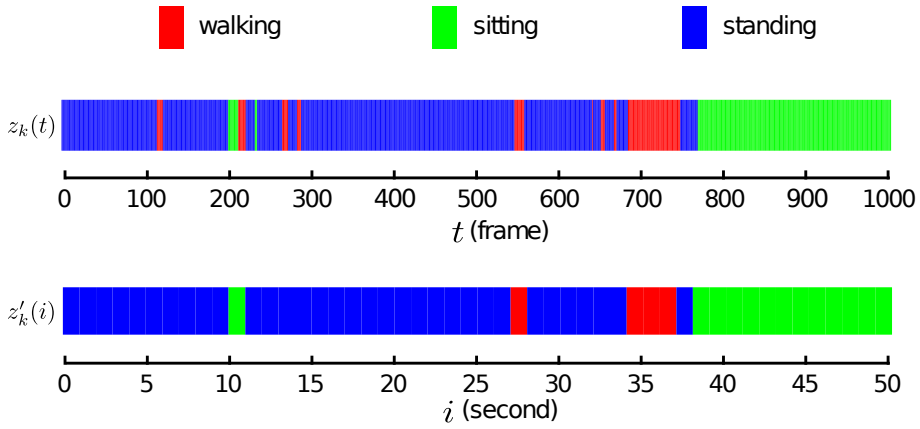



**Figure 4.2:** *An example of aggregating the instantaneous activities.* Instantaneous activities over 20 frames (frame rate: 20 fps) are merged to form one higher-level activity at each second, so as to reduce the errors in the instantaneous activity detection.

to classify instantaneous human activities using person's height and speed as features.

Given $K$ trajectories for $K$ persons, $\mathbf{r}_k(t)$, $t = 1, \ldots T$, $k = 1, \ldots K$, where $\mathbf{r}_k(t) = (x_k(t), y_k(t))$ is the physical location of the person $k$ at time $t$, their instantaneous speed is defined as $\frac{d\mathbf{r}_k}{dt}$, and simply estimated as $\frac{d\mathbf{r}_k}{dt} = \approx (\mathbf{r}_k(t+1) - \mathbf{r}_k(t-1)) \times F/2$, where $F$ is the frame rate. A recursive smoothing over time technique is used to remove the noise in the speed data. Together with the height information as inputs to a SVM classifier, we obtain the instantaneous activity states for each person $z_k(t)$, $t = 1, \ldots T$, $k = 1, \ldots K$. There are three possible states for each person at each time, $z_k(t) \in [1, 2, 3]$, where 1 is for *sitting* activity, 2 is for *walking* and 3 is for *standing*.

Because of the occlusion by the furniture and other people in the meeting room the height information from our multi-camera system is not very accurate. This may cause the instantaneous activities classified wrongly. This is evident from experimental results, which show that some activities only last for a split second, which is not realistic because of physiological limits. Activities typically last for at least short period of time, e.g., one second. Therefore, instead of updating the occupancy maps using the instantaneous activities at each time $t$, we first aggregate the instantaneous activities over short time periods into a higher-level activity state for each person at each time period $i$, $z'_k(i)$; specifically this data is aggregated over a fixed number of frames, $z_k(t)$, $t = (i-1) \cdot L + 1, \ldots i \cdot L$, where $L$ is the number of frames in this time period. The instantaneous activity state, which is detected most frequently during the time period, is taken as the state of this time period. The average of $L$ instantaneous locations during this period is taken as the location of the person at time period $i$, $\mathbf{r}'_k(i) = \frac{1}{L} \sum_{t=(i-1)\cdot L+1}^{i \cdot L} \mathbf{r}_k(t)$.

An example is shown in Fig. 4.2. There are 1000 points in one 50-seconds trajectory. Using the SVM classifier, we get one instantaneous activity at each frame, from the person's height and velocity at this frame. In total, there are 1000 instantaneous activities. Every 20 consecutive instantaneous activities are merged to one higher-level activity at each second, which are used to update the occupancy maps. Because some body movements can still be detected when the person is standing, his instantaneous activity can be recognized as walking mistakenly at some frames, as shown in the upper figure. These errors are reduced in the higher-level activities at each second, as shown in the lower figure.

## 4.4 Object Recognition

Occupancy grid map is a popular tool for representing the surrounding environments of the moving objects. They represent environments by fine-grained grids of variables that reflect the occupancy of the environment. Occupancy map updated using a specific type of activities shows the probability of the environment occupied by the moving objects with such activities. Objects corresponding to this specific type of activities can be inferred from the areas with high probability on the map. For instance, chairs tends to appear at areas with high probability on the occupancy map for sitting space.

In this section, we build an occupancy map for the sitting space, and use the observations of sitting to update the map. The presence and location of chairs will then be inferred from the sitting map. Similarly, an occupancy map for the walking space is built and updated according to observations of walking. Finally a table is recognized by the walking space in the neighborhood of the chair locations.

### 4.4.1 Occupancy map computation

In this section, we build 2D occupancy map for sitting space and walking space using sitting and walking activities, respectively. The ground plane is discretized into a fine grid of cells. Each cell in the sitting map stores the probability of people sitting in that cell, while the value of a cell in the walking map is the probability of people walking in that cell. The area of each cell, $c \times c$ $cm^2$, is chosen smaller than the size of objects in the environment. Otherwise, we can not differentiate spatially close objects that occupy the same cell.

Considering the sitting map with $M \times N$ cells as an example, the initial probability of all cells is 0.5, i.e. $p(s_{m,n}(1)) = 0.5$, $m = 1, \dots M$, $n = 1, \dots N$, which means that we have no knowledge about whether this cell is occupied or not. Given the higher-level activity state for each person at time period $i$, $z'_k(i)$, the probability of one cell being sit depends on depends on both the likelihood of this new observations and the occupancy probability of this cell at the prior time period $i$. We use Bayesian theory [Elfes 89], as shown in Eq. (4.1), to update the sitting map if a sitting activity is detected, i.e. $z'_k(i) = 1$. If no

sitting activity is detected from any person, we do not update the sitting map. Otherwise, if a person leaves his or her chair in the middle of the meeting, the probability of the cell being sit will get low as the meeting goes on. After updating the occupancy map using all of the trajectories, the probability of the cell will be too low that no chair can be detected.

$$p(s_{m,n}(i))|z'_k(i)) = \frac{p(z'_k(i)|s_{m,n}(i))p(s_{m,n}(i-1))}{p(z'_k(i))} \tag{4.1}$$

Because the size of a chair is bigger than one cell, we update not only one single cell occupied by person $k$, $(u_k, v_k) = (\lceil \frac{x'_k(i)}{c} \rceil, \lceil \frac{y'_k(i)}{c} \rceil)$, whose activity state is *sitting*, $z'_k(i) = 1$, but also its surrounding cells within 25 cm (The size of a chair is normally 50 × 50 cm). The probability of the other cells at time period $i$ stays the same as the previous time period $i-1$. In the region around the cell $(u_k, v_k)$, the closer other cells are to this cell, the more likely they are occupied by a chair. So we use a 2D Gaussian distribution to calculate the likelihood of these surrounding cells in this region:

$$p(z'_k(i)|s_{m,n}(i)) = 0.5 + 0.5e^{-(\frac{(m-u_k)^2}{2\sigma_m^2} + \frac{(n-v_k)^2}{2\sigma_n^2})}, \tag{4.2}$$

where $u_0 - \frac{25}{c} < m < u_0 + \frac{25}{c}$, and $v_0 - \frac{25}{c} < n < v_0 + \frac{25}{c}$.

The occupancy map for the walking space is initialized and updated in a similar way as the sitting map, but only using the higher-level walking activity at each time period.

### 4.4.2 Object recognition by analyzing occupancy maps

Chairs are areas with high probability on the occupancy map for sitting space. We binarize the sitting map and apply connected-component analysis on it to extract the sitting areas [Di Stefano 99]. In the binary sitting map, the connected cells with value 1 are labeled for the same chair candidate. In total we get $J$ chair candidates whose covering areas are indicated using a $M \times N$ label map. If the label value at cell $(m, n)$ is $j$, this cell belongs to the $j^{th}$ chair. Because of the imprecise trajectory data due to the camera's viewing direction or illumination problems, activities could be misclassified. Therefore not all of the connected components are real chairs.

Whether one detected sitting area is a real chair or not depends on how long people have been sitting on this area. During the meeting, people may change chairs. It is possible that people take the same chair at different time. Therefore, instead of accumulating the sitting time duration for each person, we first count the sitting time duration for each person at each sitting area, and accumulate the sitting time duration of different people at the same sitting area. Given the sitting activity for each person at each time period $i$, we count the time period of each cell being sit during the whole meeting, and accumulate the time duration of the cells with the same label, i.e. the same chair candidate. Finally we find all of all of candidate chairs in the meeting

room with durations of being sitting, and remove the chairs on which people perform sitting behaviors for a very short time.

To detect the table, we proceed as follows. First we calculate the central position of the locations of people performing sitting activity on all cells belonging to each chair, as the chair center. Because a table is surrounded by the detects chairs. The mean of the locations of all chair is on the table. We choose the chairs' mean location as the seed, and employ a region-growing algorithm to segment the area which is inside the closed banding of walking space. The largest rectangle in this area is segmented as the table.

When a person walks around the table, his or her position may not be tracked accurately because of the occlusion by the table. Instead of besides the table, the person could be detected at one location on the table area. This leads to walking area expanding to the table. A smaller table is eventually detected by region growing within the walking band. During the meeting, more sitting activities are detected than walking activities, so that the chairs detected from the sitting activities are more accurate. We scale and stretch the rectangle detected so that the border of the table is next to the chairs.

## 4.5 Experiments

Our meeting room lab is 880 $\times$ 500 cm, observed by two top-view and four side-view cameras at 20 FPS. To evaluate the strength of our algorithm, we recorded 9 meeting sequences over 75 minutes in total. There is only one table in all of these sequences, but the numbers of chairs and people are not fixed: three to eight chairs and three to five people. The ground plane is divided into cells of $5 \times 5$ $cm^2$ to make sure the objects will not fall within one cell.

### 4.5.1 Activity classification

Activity classification results are shown in Fig. 4.3. We define the ground truth by segmenting and labeling sequence 1 manually. Although it is not very precise, ground truth is still able to evaluate the classification results. Sometimes our algorithm cannot distinguish standing from walking activity because there are many body and hand movements even when people stand and present. For instance, person 2 stands in front of a white board and gives a presentation during the meeting, but sometimes his activities are mistakenly recognized as walking. When a person gets up from a chair and moves to another location close to his chair to shake hands with others, it is hard to accurately label the standing and walking activities even when defining the ground truth in this case. However, the SVM classifier successfully detects the standing activity before and after the user sits on a chair successfully (the green sitting period is between two blue standing periods).

Compared to the ground truth, all sitting activities are successfully detected for both person 1 and 2. There are four sitting periods for each of them. Person 3 only takes chairs to sit twice during the meeting, but we detect it four times.

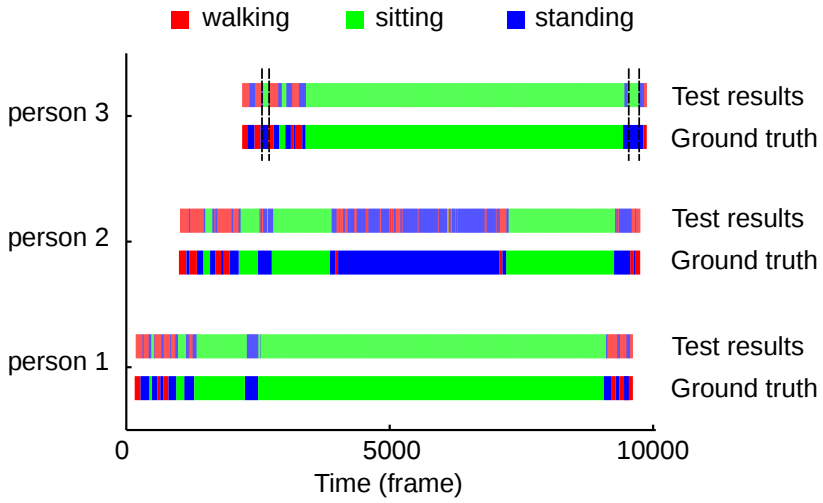


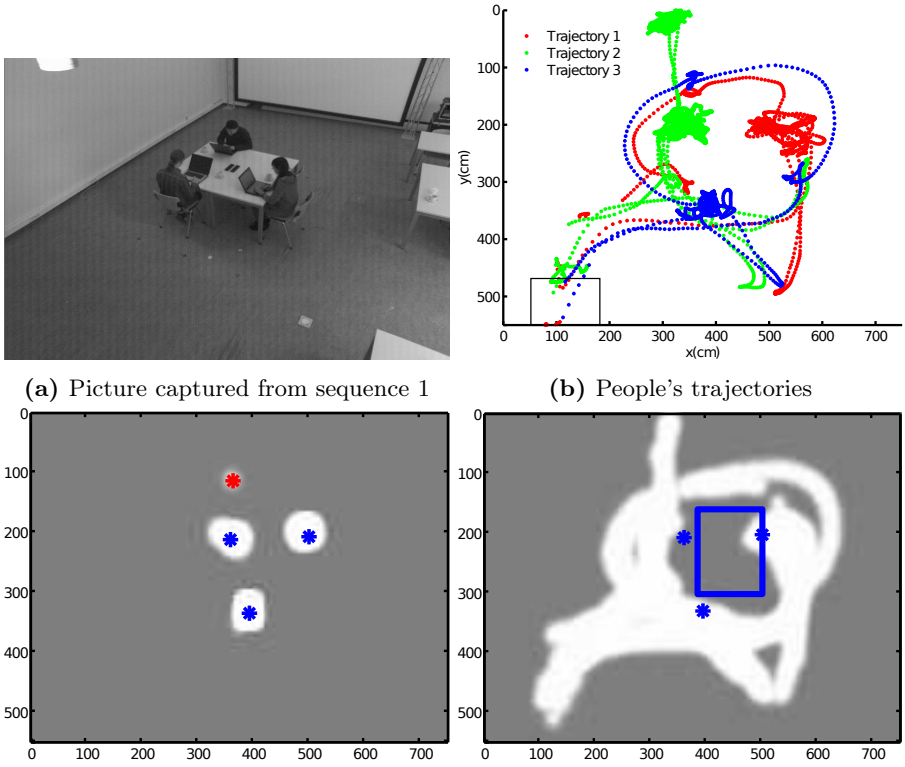

**Figure 4.3:** *Instantaneous activity classification results.* Person 1's activities are classified more accurately than the other people. For person 2, his or her standing activities during his presentation are mistakenly recognized as walking because of the body and hand movements. For person 3, his or her activities are mistakenly recognized as *sitting* twice because of the inaccurate height information.

These two mis-detected sitting periods are indicated using black dashed lines. Both of these mis-detected periods happen when person 3 stands up and shakes hands with others. Our tracker detects his height much lower than the actual value. Therefore his activity is categorized mistakenly as sitting, resulting in wrong sitting area detection. We will remove these false detections in the next step.

### 4.5.2 Objects recognition results

Fig. 4.4 shows the chairs and table recognition results for sequence 1. In this sequence, 3 people are participating in a meeting. There are 3 chairs and 1 table in the meeting room. People never change chairs during the meeting. The range of the probability in Fig. 4.4c and Fig. 4.4d is from 0 (black) to 1 (white). The table in Fig. 4.4d is represented by a blue rectangle, and the chairs are blue stars near the blue rectangle (the table). Fig. 4.5 shows the results for sequence 2 in the same way, while there are 3 participants and 8 chairs. People sit at different places to ensure that all of chairs have been used.

For sequence 1, these sitting areas are categorized into 4 candidate chairs. Table 4.1 shows their estimated positions, the time length of the chair being sit during the meeting (in seconds), and the distance between the chair center and the table edge. The third chair candidate, which is indicated using a red star in Fig. 4.4, is a false detection, because the sitting activities there have only lasted for 2 seconds. Although some standing activities is mistakenly detected



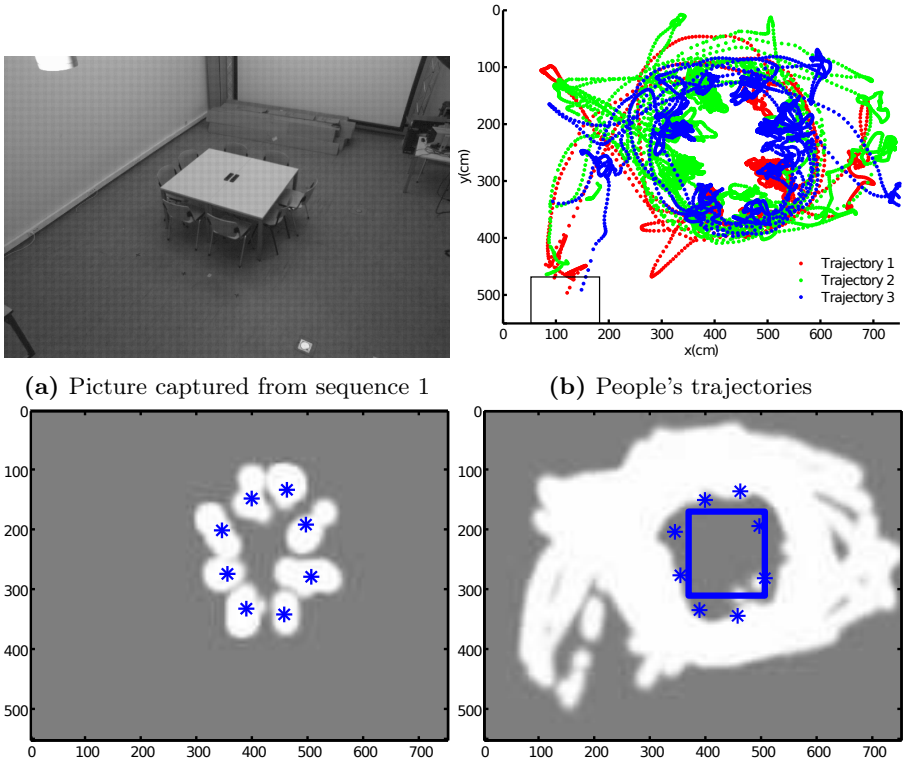

**(a)** Picture captured from sequence 1 **(b)** People's trajectories

**(c)** The probability map of sitting space **(d)** The probability map of walking space

**Figure 4.4:** *Detected chairs and table in sequence* 1. 4 chair candidates are detected using the sitting activities, and one of them is removed because of the short sitting time period. The table is represented by a blue rectangle inside the enclosed banding of walking space. (b) shows the area where people enter and exit the meeting room using a rectangle at the left-bottom corner.

as sitting activities for person 3, as shown in the dashed lines in Fig. 4.3, no other chairs are falsely detected from the occupancy map for the sitting space, because person 3 stands at the location of his or her chair when his or her standing activities are mistakenly detected as sitting activities.

Although chairs may not stay in a fixed position in the meeting room as they are used, they do not move too far from the table and usually also not from their original position. The size of the chairs in this paper is $50 \times 50$ cm. We set the range of the distance between chairs' center and table as $(-25cm, +50cm)$, where $-25$ cm means that the chair is totally under the table as shown in Fig. 4.5a, and $+$ means that user pulls his chair far away from the table and the largest distance between chairs' center and table is 50 cm. As shown in Table 4.1, the distance for every chair is within the predefined range, illustrating the good performance of chair detection.




**(a)** Picture captured from sequence 1 **(b)** People's trajectories

**(c)** The probability map of sitting space **(d)** The probability map of walking space

**Figure 4.5:** *Detected chairs and table in sequence* 2. 8 Chairs are detected, as indicated using blue stars. The table is represented by the blue rectangle.

**Table 4.1:** Candidate chairs in sequence 1.

| Candidate chair | Estimated position(cm) | Time duration of being sit (second) | Distance to the table (cm) |
|---|---|---|---|
| 1 | (497.9, 199.4) | 188 | -4.1 |
| 2 | (354.7, 204.9) | 85 | 10.3 |
| 3 | (356.9, 112.2) | 2 | |
| 4 | (391.2, 335.4) | 155 | 30.4 |

Table 4.2 shows the parameters of the top-left and bottom-right corners of the rectangle which represents the table, and their localization error. The ground truth is obtained through calibrating the environment. All of the errors are below 20 cm. The results of the second sequence with more tracks are more accurate than that of the first sequence.

**Table 4.2:** The difference between the position of detected table and ground truth.

| | Top-left corner(cm) | Bottom-right corner(cm) | error(cm) |
|---|---|---|---|
| Seq. 1 | (375.0, 150.0) | (505.0, 305.0) | 15.96 |
| Seq. 2 | (365.0, 160.0) | (510.0, 310.0) | 9.47 |
| Ground truth | (362.0, 160.4) | (502.0, 320.4) | |

## 4.6 Conclusion

In this chapter, we presented a method to recognize objects in a meeting environment using people's trajectories. We extract the speed and height of people as inputs into SVM classifier to categorize people's instantaneous activities, and merge the instantaneous activities to higher-level activities at every time period, We build occupancy maps for sitting and walking spaces separately, and utilize the higher-level activities to update the maps using Bayesian theory. Finally, we infer the chairs from the sitting map, and the table from the walking map.

Compared to the other methods presented in the other chapters of the PhD, the work in this chapter is only a proof of principle. In the future we would like to extend this work by improving the probability models and classification techniques. The proposed method to detect tables may not work when there are more than one table present in the room, or the table is close to the wall or locates at one corner of the room. In the future, future work could also focus on solving these issues.

This research resulted in one paper in the proceeding of 2012 International Conference on Distributed Smart Cameras (ICDSC) [Xie 12].

# 5

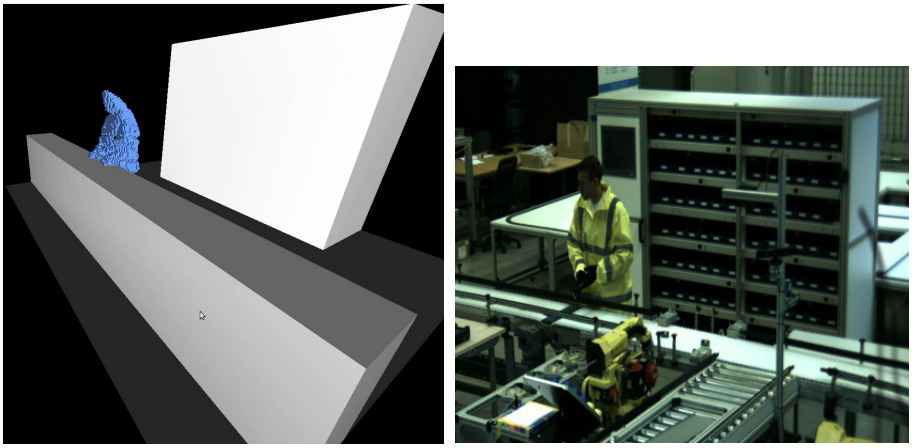
# Work Cycle Analysis

Assembly lines form the core of flow-line production systems in the industrial production of high quantity standardized commodities [Gebus 09, Becker 06], such as automobiles and electronic goods. Each assembly line consists of several work stations arranged along a mechanical transportation system, e.g. a conveyor belt. The speed of this belt, i.e., the production rate, is mainly determined by the speed of factory workers assembling various components [Hartmann 09]. If the speed is too high, workers may experience physical and psychological stress and may start making more mistakes, reducing the quality of the produced goods, or even leading to injuries.

Researchers have made a lot efforts to optimize the efficiency of the work cycle performed by each worker by carefully planning and scheduling the elementary operations of the work cycle and by designing the work stations. The former is called the Assembly Line Balancing Problem (ALBP) [McCormick 89, Boysen 07]. These optimizations are crucial to improve work efficiency for long-term productions.

However, it is practical to design new assembly line and allocate the operations among the work stations for each new product since the production just runs for a short time. In this case, the work cycle needs to be changed frequently to cope with different and new products. It is more economic to apply the same assembly line for all of the short-term productions, but try to optimize the work cycle itself. In this chapter, we address the particularly challenging use case of work cycle optimization for manufacturing strategies with short production runs.

Specifically, we focus on analyzing the tracks of factory workers around the work station of an assembly line, as shown in Fig. 5.1. The factory worker at each work station performs his or her task repeatedly, which consists of certain operations, regarding the desired cycle time. Although this work cycle is scheduled for the workers, how they execute it is different, which may lead to work inefficiencies. Examples of inefficiencies which can be detected this way are workers having to walk too much back and forth between the conveyor belt and part storage racks, or workers being idle at the part storage racks because they suddenly forget what parts to pick up, or workers walking back



**(a)** Simulated work station in 3D **(b)** Real work station

**Figure 5.1:** *Work station and storage rack.* (a) shows the storage rack in white, the work station in grey and the worker in blue. (b), the factory worker stands at the work station assembling the parts. There is a storage rack behind him, where the tools and parts are kept.

and forth from one side to the other side of the storage racks too much, or having to hurry during certain executed work cycle phases. Fast and low-cost work cycle optimization depends on efficiently measuring the time spent on individual operations of scheduled work cycle and on analyzing variances between executed work cycles statistically to detect inefficiencies or to discover factors which may induce errors. On the other hand, this analysis also uncovers irregularities in executed work cycles e.g. due to a temporarily exhausted supply of components. The results of this analysis can also be used as feedback to the workers to help them improve their efficiency.

In indoor environments, worker trajectories can be recorded using various sensing systems. Radio-Frequency Identification (RFID) works well but samples the trajectories only at specific positions. As such it provides limited spatial resolutions. Measuring the Received Signal Strength (RSS) of RF signals and B1uetooth devices suffers from the same problems [Al Nuaimi 11, Ijaz 13]. In our own work "Multi-camera human behavior monitoring and unusual event detection" [Bo 14], we have experimented with Ultra-wide band (UWB) based positioning systems, which can yield an accuracy of about 3 cm. However, this equipment is too bulky for practical use.

In our work the trajectories of the workers are measured using a multi-camera vision system. We opted for such a system, rather than for other measuring devices, because video analysis provides very accurate data and is quite flexible and avoids the need to wear portable devices. Moreover, it allows the analysis of body posture and gestures as well as trajectories, which will allow us to refine the analysis in future work. The video analysis uses sophisticated

methods based on the visual hull concept and occlusion modeling to extract three-dimension (3D) silhouettes of factory workers [Slembrouck 14]. In this chapter, we do not focus on the video analysis but on the trajectory analysis.

In the factory environment, when factory workers move to the storage rack to pick parts, they may stay at different locations in front of the storage rack if they need pick up more tools and objects, or they may head to a specific location in front of the storage rack, pick the required parts and then head back to the assembly platform directly. While the executed work cycles will display small variances, the tracks will tend to be similar, both spatially and temporally. However, sometimes the tracks will differ significantly from the typical executed track. Temporal deviations may occur if the worker is waiting for the conveyor belt, or if the worker needs to hurry or slows down. Spatial deviations can also occur, e.g. when the worker leaves the work station for some reason, needs to replenish a storage rack or takes a shorter or less regular route due to increasing or decreasing time pressure. Track analysis relies on breaking down the scheduled work cycle in its components and on comparing individual instances of executed work cycles to each other and to the scheduled work cycle, both spatially (which areas does the trajectory move through?) and temporally (how fast does the worker move at each point of the cycle?).

Trajectories analysis of moving objects provide crucial clues for behavior analysis especially in surveillance applications. To a certain extent, the trajectories of the objects tend to be similar when they perform the same or similar behavior. Behavioral pattern analysis in factories therefore focuses on this repetitive behavior, and deviations from it. Typical parameters of interest relate to speed and space coverage. In literature, spatial-temporal trajectory data have been used to understand and characterize the behaviors of moving objects and also to detect their abnormal behaviors [Morris 08]. Nguyen *et al.* modeled inherent hierarchy and shared structures of human behaviors through defining primitive behaviors using sub-trajectories between the furnitures [Nguyen 05]. Behavioral pattern recognition through clustering the trajectories has been studied throughly (Piciarelli *et al.* [Piciarelli 06], Naftel *et al.* [Naftel 06], Kalnis *et al.* [Kalnis 05], Yang *et al.* [Yang 12] etc.).

In this chapter we propose to segment the trajectories obtained from our multi-camera system into tracks regarding work cycle, and analyze the spatial and temporal difference of the executed work cycles. We first cluster executed work cycle tracks according to spatial similarity based on a spatial similarity measure between tracks. The result is a set of clusters, each containing tracks with high spatial similarity, but possibly low temporal similarity. In the second step of the analysis, we temporally align all tracks of the same spatial cluster, to establish the point correspondences among all tracks. The third step of the analysis creates a *prototypical route* for each cluster, and an associated (time-varying) speed along the prototypical route.

The prototypical route created using the tracks of the executed work cycle provides information on the average work cycle. The prototypical velocity and dwell time are the most frequency velocity and dwell time in which the workers

travel along the prototypical route. This is useful for training new workers, and for analyzing and optimizing the work cycle both spatially and temporally. For instance, an analysis of the speed in each part of the work cycle may indicate bottle necks and idle points. The results show that it is more accurate to build the prototypical route using the proposed track alignment method, than by averaging non-aligned tracks in other DTW-based methods.

The remainder of the paper is structured as follows. Related work is listed in the next section. Section 5.2 describes how to extract the spatially prototypical route with prototypical velocity and dwell time along the route using the track alignment. Section 5.3 shows our experimental results. Section 5.5 concludes this chapter.

## 5.1 Related work

In literature engineers have proposed to document the work in the factory using films and evaluate the workers' performance. Stork *et al.* use a video camera to record the worker's hands movements, and study three different instruction modes with their influence on the hands movements during manual assembly [Stork 08]. Stoessel *et al.* depict a workbench with motion sensors and cameras as the implementation platform for the worker assistance system [Stoessel 08]. It will be helpful to understand the entirety of human mental functions dedicated to information processing, such as perception, attention and action, leading to a more efficient manual assembly performance.

Different from their work, we focus on studying how the workers execute the work cycle. The time of the worker spending on assembling tasks in each cycle relates directly to the overall productivity of the worker in the work cycle. Therefore, analyzing the work cycle can help redesigning the work cells based on the actual productivity and to improve workers' performance by analyzing the factors leading to inefficiencies in the work cycle.

Elnekave and Gilad use a single video camera and by visually inspecting the video they measure the distance between points of interest and – using a stopwatch – the time interval between events of interest. This way they estimate real work time, idle time and the time for tasks unnecessarily productive [Elnekave* 06].

In earlier work [Bauters 14], we use a multi-camera system to locate the factory worker's position at second intervals. The estimated locations are then clustered using the $K$-Means algorithm and the dwell time at each location is then estimated. Cycle time analysis is not included in this paper.

In this chaptor, we aim to analyze the work cycle using the workers' trajectories around the work station. Since the workers take various routes to pick up the parts to be assembled, we need to cluster the tracks regarding to the work cycles. For each type of executed work cycle, the tracks will be aligned together using the methods described in Chapter 3. The point associations produced during the alignment will be used to average the tracks, so as to extract a prototypical route and the prototypical velocity along it.

Researchers have proposed some methods related to Dynamic Time Warping (DTW) to average the tracks by aligning them, as introduced in Section 3.1. Junejo and Foroosh propose to average the boundary trajectories of the "spatial envelope", which are created from all the pedestrian trajectories [Junejo 07, Junejo 08]. Niennattrakul and Ratanamahatana propose a hierarchical clustering approach called Prioritized Shape Averaging (PSA) based on DTW to average all of the trajectories [Niennattrakul 09] . Petitjean *et al.* propose a global averaging method for DTW, called DTW Barycenter Averaging (DBA) [Petitjean 11].

The DTW-related methods mentioned above can average multiple trajectories using different strategies but not establish the time correspondences among the trajectories, which are used to analyze the location and speed fluctuations in our work. We will compare our methods of track alignment on extracting the prototypical route, i.e. the average track, with other methods on factory worker trajectories.

## 5.2 Work cycle optimization

In our case, depending on how the factory workers traverse the area in front of the storage rack so as to pick up tools and parts, the created tracks can be grouped into different work cycles. Therefore, we first cluster the tracks for different work cycles depending on their dissimilarity, which is described in Chapter 2. We optimize the work cycle by analyzing the corresponding tracks, creating a prototypical route and prototypical velocity and dwell time along the route. Prototypical route provides information for the workers about how they should execute the work cycle spatially. Prototypical velocity provides guiding speed for the workers to follow. Prototypical dwell time gives time information of how long the workers need to pick up tools and parts in front of the storage rack. We utilize the track alignment, elaborated in Chapter 3, to calculate the three key information for each type of work cycle.

During the work cycle, sometimes workers spend more time than necessary on their way to pick up the tools and objects, leaving too little time left to later assemble the parts at the assembly platform. At other times workers may hurrying needlessly leading to idle time at the assembly platform. Both spatial and temporal issues can lead to the worker arriving at the assembly platform later or earlier than he should do. These include unnecessary detours, unusual routes, abnormally fast or slow movement, unnecessary stops, needless dwelling.

In this chapter, we compute a prototypical route for each type of work cycle, represented using a series of data points. This route is accompanied with specifics velocity at each point on the route, and with dwell times thus forming a prototypical work cycle. The prototypical route is computed by averaging the warped tracks along the warping paths. And the prototypical velocity is computed as the mode velocity which maximize its probability function, so as the prototypical dwell time.

### 5.2.1 Prototypical Route

Given $N$ tracks for one work cycle, $\mathbf{r}_n(t_n)$, $t_n = 1, \dots T_n$, we utilize the methods of trajectory alignment, which are elaborated in Chapter 3, to establish the point associations among the tracks, resulting in warped tracks $\mathbf{g}_n(k)$, $k = 1, \dots K$, $n = 1, \dots N$, and the time indexes of the points of the warped tracks in the original tracks $w_n(k)$, $k = 1, \dots K$, $n = 1, \dots N$. The warped tracks all have the same length and for specific values of $k$, the track points $\mathbf{g}_n(k)$, $n = 1, \dots N$ are spatially close. This allows them to be averaged directly, whereas averaging of non-warped tracks would fail in most cases.

As shown in Equation 3.4, the warped tracks are averaged at the same time index to form a prototypical route for the work cycle, $\mathbf{g}_0(k)$, $k = 1, \dots K$. The variance of the physical locations of track points along the prototypical route, $\sigma_g(k)$, $k = 1, \dots K$, is calculated as shown in Equation 3.5:

### 5.2.2 Prototypical Instantaneous Velocity

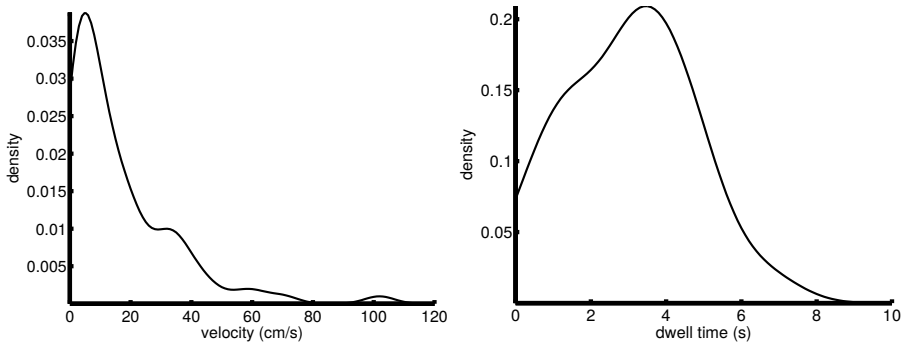
The instantaneous velocity of trajectory $\mathbf{r}_n(t_n)$ is defined as $\frac{d\,\mathbf{r}_n}{dt_n}$ and can be estimated using various numerical techniques, the simplest one being $\frac{d\,\mathbf{r}_n}{dt_n} \approx (\mathbf{r}_n(t_n + 1) - \mathbf{r}_n(t_n - 1))/2$. However, some temporal smoothing can provide temporal robustness. In the following we will focus primarily on the magnitude $|\frac{d\,\mathbf{r}_n}{dt_n}|$ of the velocity.

We could similarly define the instantaneous velocity of the prototypical route as $\frac{d\,\mathbf{g}_0}{dk}$, but the problem with this approach is that $k$ does not have a direct relationship with time. Therefore, we proceed differently: with each $k$ there corresponds a specific spatial location $\mathbf{g}_0(k)$, and for each track a specific time $w_n(k)$ at which this point is reached. Given the velocity magnitudes of the original tracks $s_n(t_n)$, $t_n = 1, \dots T_n$, $n = 1, \dots N$, and the time indexes of the points of warped tracks in the original tracks $w_n(k)$, $k = 1, \dots K$, $n = 1, \dots N$, the instantaneous velocity along each warped track is expressed as $v_n(k) = s_n(w_n(k))$, $k = 1, \dots K$, $n = 1, \dots N$. We then define the *prototypical instantaneous velocity*, $v_0(k)$, $k = 1, \dots K$, as the average velocity along the prototypical route when it reaches the location $\mathbf{g}_0(k)$.

There are three types of *average* in statistics: arithmetic mean, median and mode. In this paper, we calculate the mode, instead of arithmetic mean to indicate prototypical instantaneous velocity. Given the velocity of each track at the location $\mathbf{g}_0(k)$, $v_n(k)$, $n = 1, \dots N$, the value which maximizes its probability function is taken as the mode velocity, i.e. the prototypical instantaneous velocity $v_0(k)$.

Kernel density estimation (KDE) is used to estimate the probability density function (pdf) of the velocities along the prototypical route, the prototypical instantaneous velocity $v_0(k)$ is calculated as $v_0(k) = arg \max_v \hat{f}(v; h)$.

$$\hat{f}(v; h) = \frac{1}{Nh} \sum_{n=0}^{N-1} K\left(\frac{v - v_n(k)}{h}\right), \tag{5.1}$$




**(a)** Velocity **(b)** Dwell time

**Figure 5.2:** *velocity and dwell time density.* (a) is the density of velocity at one step of the warp path, and (b) is for the dwell time.

where $K$ is the kernel function, $h$ is its bandwidth. In this work we use the normal kernel $K(x) = e^{-x^2/2}/\sqrt{2\pi}$ [Minnotte 93] and choose the bandwidth $h$ equal to 10 centimeter per second (cm/s).

The velocity variance of the track points along the prototypical route, $\sigma_v(k)$, $k = 1, \ldots K$, is calculated as:

$$\sigma_v(k) = \frac{1}{N} \sum_{n=1}^{N} (v_n(k) - v_0(k))^2 \tag{5.2}$$

where $k = 1, \ldots K$.

### 5.2.3 Prototypical Dwell Time

In some locations, the instantaneous velocity will be very low, e.g. near picking places, or other places where the worker remains stationary for a while. In those locations, *dwell time* is a much more useful measure of the work cycle. For a given track $\mathbf{r}_n(t_n)$, $t_n = 1, \ldots T_n$, and time $t$ when the worker keeps stationary, let $t_n^{(1)}(t)$ and $t_n^{(2)}(t)$ be the first time before and after $t$ separately when the worker's velocity is lower than then given threshold, e.g. $20cm/s$. Then we define the dwell time $D_n(t)$ as $D_n(t) \stackrel{\triangle}{=} t_n^{(2)}(t) - t_n^{(1)}(t)$. The prototypical dwell time, $D_0(k)$, $k = 1, \ldots K$ and probability density functions of dwell time can then be defined in terms of the track dwell times, in a similar way as for velocities.

Fig. 5.2 shows the velocity and dwell time density at a specific location of the prototypical route. There the factory workers stay stationary to pick up tools and parts. According to the density of the velocity and dwell time, 7 cm/s is chosen as the prototypical instantaneous velocity, 3.5 s is chosen as the prototypical dwell time at that location. The mean velocity and dwell time are

20 cm/s and 2.0 s respectively. 7 cm/s fits to the factory workers' behavior of being stationary better.

We proposed two methods to align many trajectories in Chapter 3. Based on the point assignations established by different methods, the prototypical work cycles deduced are also different. Therefore we use the superscripts to distinguish them. $\mathbf{g}_0^{(1)}(k)$, $v_0^{(1)}(k)$ and $D_0^{(1)}(k)$, $k = 1, \dots K^{(1)}$ are prototypical route, prototypical instantaneous velocity and prototypical dwell time extracted using the stretch-and-then-compress method, and $\mathbf{g}_0^{(2)}(k)$, $v_0^{(2)}(k)$ and $D_0^{(2)}(k)$, $k = 1, \dots K^{(2)}$ are extracted using the greedy method based on successor classification.

## 5.3 Results

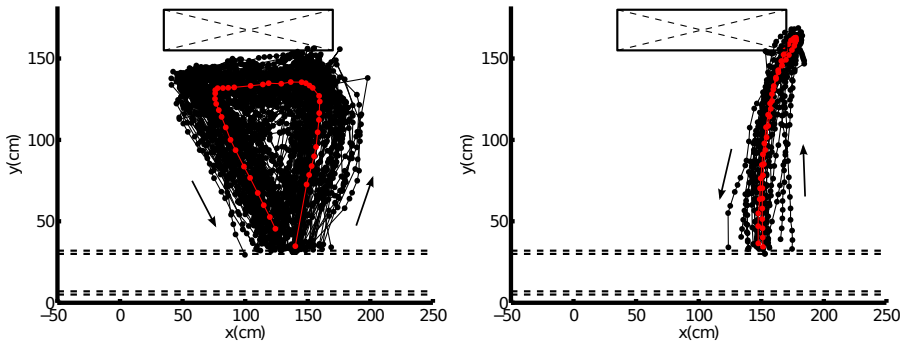
Using different track alignments, the work cycle can be optimized in different ways. We first show the results obtained using the stretch-and-then-compress method presented in Chapter 3.3. In Section 5.3.2, we will show the results using the greedy method based on successor classification, which is elaborated in Section 3.4. The results include the prototypical route, the prototypical velocity and dwell time along the prototypical route, and location variance and velocity variance along the prototypical route.

### 5.3.1 Results using the stretch-and-then-compress method

In this section, we show the results of work cycle optimization based on the point associations produced using the stretch-and-then-compress method, including the prototypical routes and prototypical velocity along the routes. We also show the statistical analysis results of the spatial variance and velocity variance.

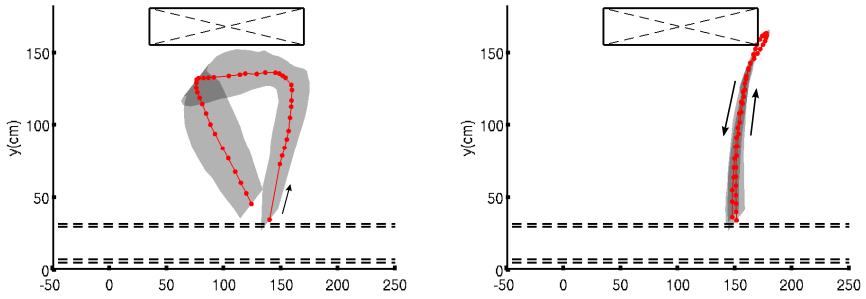
The average tracks in Fig. 5.3a and 5.3b act as prototypical routes for each work cycle, respectively. There are 82 tracks for the first type of executed work cycle, which are more dissimilar to each other than the 12 tracks for work cycle 2. The inferred prototypical route with 52 points for work cycle 2 represents how the workers execute it better, as shown in Fig. 3.29d. The prototypical route with 39 points for work cycle 1 is less representative, because the *compress* operation removes too many points. For instance, the workers will not be able to visit the left corner of the storage rack by following this route.

Fig. 5.4 shows the variance of physical locations of the track points, $\sigma^{(1)}(k)$, $k = 1, \dots K^{(1)}$, along the prototypical route for each type of executed work cycle. The prototypical routes are indicated using red lines with dots. The width of the band along the prototypical route indicates the intensity of the location variance. The wider the band is, the more dissimilar the associated points are and vice versa. From Fig. 5.4, we can see the factory workers execute the first type of work cycle more variously than the second one. And for the first type of work cycle shown in Fig. 5.4a, the variance of the locations where




**(a)** Work cycle 1 **(b)** Work cycle 2

**Figure 5.3:** *Prototypical routes calculated using the stretch-and-then-compress method.* The arrows indicate the moving directions of workers in the trajectories.

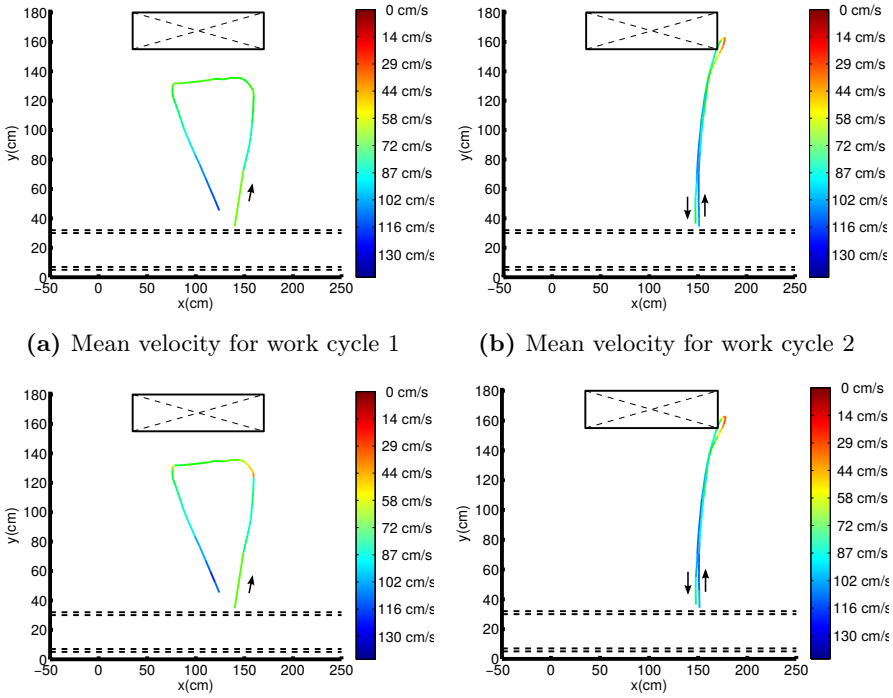



**(a)** Work cycle 1 **(b)** Work cycle 2

**Figure 5.4:** *Location variance.* (a) shows the variance of physical locations of the track points along the prototypical route for work cycle 1. (b) is for work cycle 2. The arrows indicate the moving directions of workers in the average trajectories.

the factory workers go to the storage rack is smaller than that on their way back to the assembly line, indicating that the factory workers go to the storage rack in a more similar way than back from there.

The mean and prototypical velocity along the prototypical route for each work cycle are shown in Fig. 5.5. Fig. 5.5a and 5.5c are mean and prototypical velocity for first type of work cycle respectively, and Fig. 5.5b and 5.5d are for work cycle 2. The velocity is color coded. At the right and left corners of the storage racks, the factory workers stay stationary to pick up tools and parts. Their velocity there should be very low. Both Fig. 5.5b and 5.5d show low velocity in red at the right corner of the storage rack for the second type of work cycle. In Fig. 5.5a, the mean velocity at both of the corners are indicated in green (around 60 cm/s). The prototypical velocity at the right corner of




**(a)** Mean velocity for work cycle 1 **(b)** Mean velocity for work cycle 2

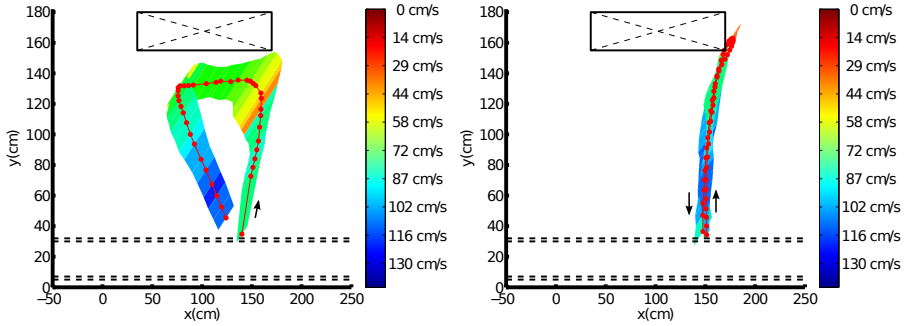
**(c)** Prototypical velocity for work cycle 1 **(d)** Prototypical velocity for work cycle 2

**Figure 5.5:** *Mean and prototypical velocities along the prototypical routes.* (a) and (b) show the mean velocity along the prototypical routes for each type of work cycle respectively. (c) and (d) are for the prototypical velocity.

the storage rack, shown in Fig. 5.5a, is around 40 cm/s. It proves that the prototypical velocity (mode of the velocity distribution) can guide the factory workers better. However, the stretch-and-then-compress method does not align the tracks for work cycle 1 well because the *compress* operation removes too many points. Therefore the prototypical velocity along the prototypical route for the first type of work cycle is not as accurate as the second one.

Fig. 5.6 shows the velocity variance along the prototypical routes. Wider band means higher variance. The prototypical routes are indicated in red lines with dots, and the color indicates the prototypical velocity. Using the point associations established by the stretch-and-then-compress method, the velocities of the associated points for work cycle 1 at the right corner of the storage rack are very different from each other, producing wider band there.

The mean and prototypical dwell time along the prototypical route for each work cycle are shown in Fig. 5.7. The instantaneous dwell time is color coded. At the right and left corners of the storage racks, the factory workers stay stationary to pick up tools and parts. Their dwell time there should be longer




**(a)** Work cycle 1 **(b)** Work cycle 2

**Figure 5.6:** *Velocity variance along the prototypical routes.* The width of the band indicates the variance in local velocity, whereas the color indicates the prototypical velocity itself.

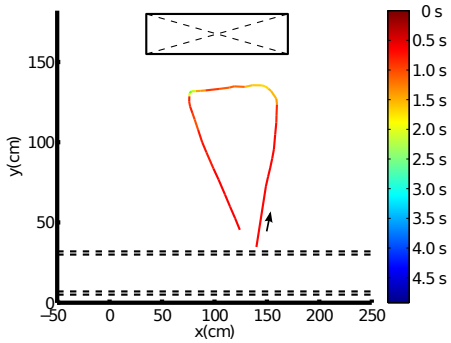
than that at other places. Both Fig. 5.7b and 5.7d show that the dwell time at the right corner of the storage rack is around 2.5 seconds for work cycle 2. Both Fig. 5.7a and 5.7c show that the workers dwell at the right corner of the storage rack for around 1.5 seconds when they execute Work cycle 1. Fig. 5.7a shows they dwell at the left corner of the storage rack for around 2.5 seconds averagely, and the mod of the dwell time distribution there is around 1.5 seconds. Actually by checking the video footage, we find that the workers mostly stay at the left corner for 3 to 6 seconds. Using the point associations established by the first alignment method, the *compress* operation removes some points with longer dwell time at the left corner of the storage rack. Therefore, neither the average dwell time nor prototypical dwell time is correct for work cycle 1.

### 5.3.2 Results using the greedy method based on successor classification

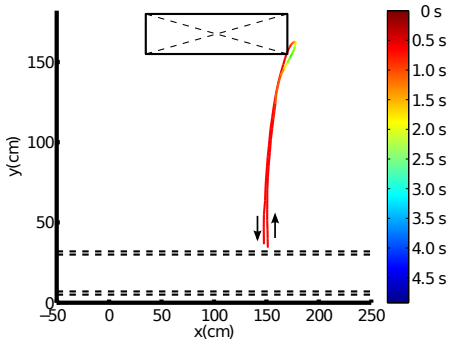
In this section, we will show the results of the work cycle optimization using the point associations established by the the greedy method based on successor classification, as elaborated in Section 3.4.

The average tracks in Fig. 3.28e and 3.29e act as prototypical routes for each work cycle, respectively. For work cycle 1, the point associations shown in Fig. 3.23 are used to calculate the prototypical route shown in Fig. 3.28e. There are 1250 points in the prototypical route representing this work cycle. Fig. 3.29e shows the prototypical route with 312 points for work cycle 2, through point associations shown in Fig. 3.24.

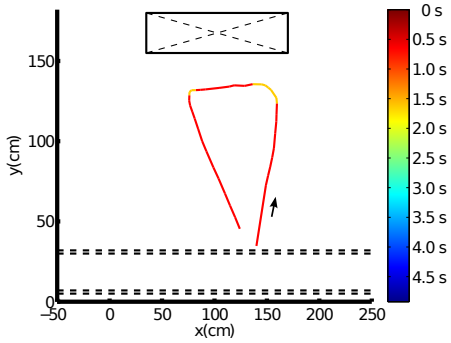
For work cycle 2, the stretch-and-then-compress method described in Section 3.3 creates a prototypical route with 52 data points, whereas there are 312 data points representing the prototypical route created by the greedy method based on successor classification in Section 3.4. These two prototypical routes, as shown in Fig. 3.29d and 3.29e, are spatially similar to each other. Both of



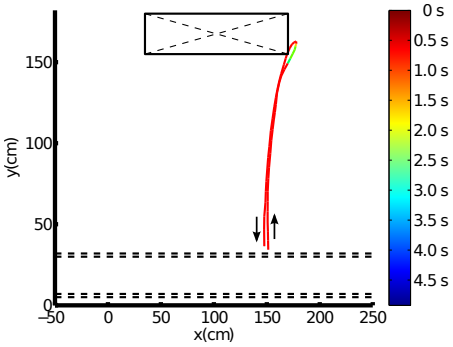

**(a)** Mean dwell time for work cycle 1




**(b)** Mean dwell time for work cycle 2

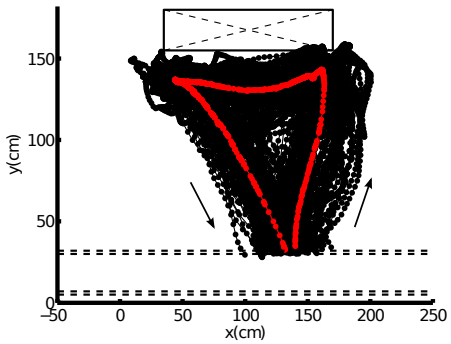



**(c)** Prototypical dwell time for work cycle 1

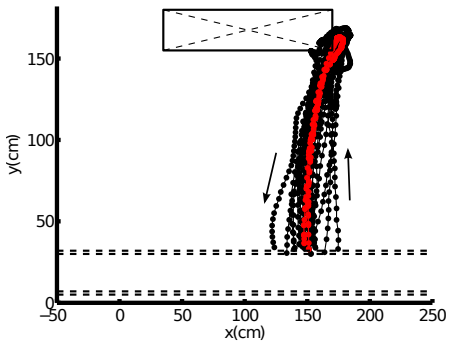



**(d)** Prototypical dwell time for work cycle 2

**Figure 5.7:** *Mean and prototypical dwell time along routes.* (a) and (b) show the mean dwell time along the prototypical routes for each type of work cycle respectively. (c) and (d) are for the prototypical dwell time.

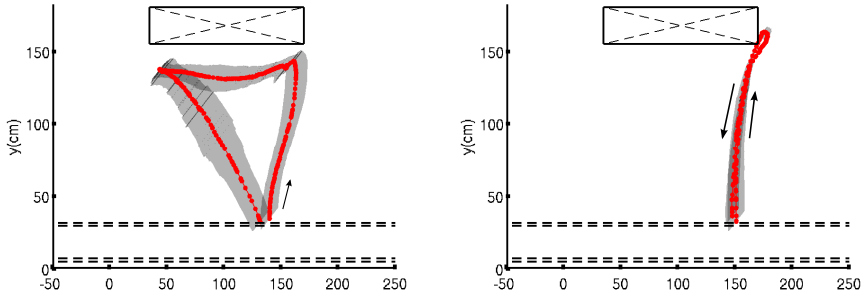



**(a)** Work cycle 1




**(b)** Work cycle 2

**Figure 5.8:** *Prototypical routes calculated using the greedy method based on successor classification.*




**(a)** Work cycle 1 **(b)** Work cycle 2

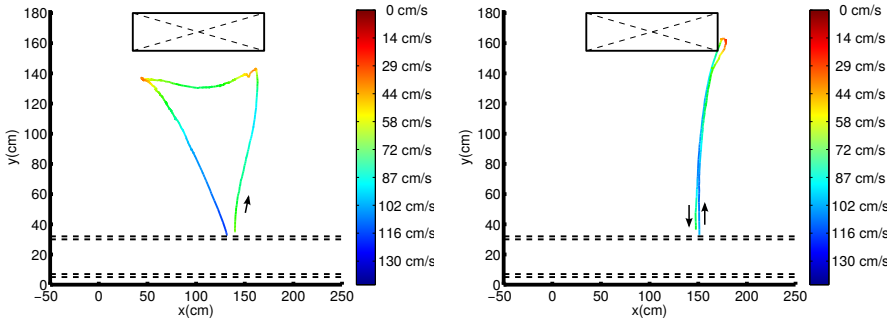
**Figure 5.9:** *Location variance.* (a) shows the variance of the physical locations of the track points along the prototypical route for work cycle 1. (b) is for work cycle 2.

them can be used to guide the factory workers to execute the second type of work cycle, in which they go to the right corners of the storage rack to pick up tools and parts.

For work cycle 1, there are only 39 data points representing the prototypical route created using the stretch-and-then-compress method, but as many as 1250 data points using the greedy method based on successor classification. Although it is redundant to represent the prototypical route using 1250 data points because some of them are very spatially close to each other, all detailed information is kept in this route. As shown in Fig. 3.28e, the factory workers will visit both of the corners of the storage rack to pick up tools and parts by following this prototypical route. But the prototypical route created using the stretch-and-then-compress method, as shown in 3.28d, does not reach the left corner of the storage rack because the data points there are removed by the *compress* operation during the track alignment. Therefore, this prototypical route in Fig. 3.28e can guide the factory workers better to finish pickup job.

Fig. 5.9 shows the variance of the physical locations of the points at the same indexes of the warped tracks along the prototypical route for each work cycle. The same as from Fig. 5.4, we conclude from Fig. 5.9 that the factory workers execute the first type of work cycle more variously than the second one because the band along the prototypical route in Fig. 5.9b, i.e. location variance is wider generally than that in Fig. 5.9a. We also conclude about work cycle 1 from Fig. 5.9a that the factory workers go to go to the storage rack in a more similar way than back from there.

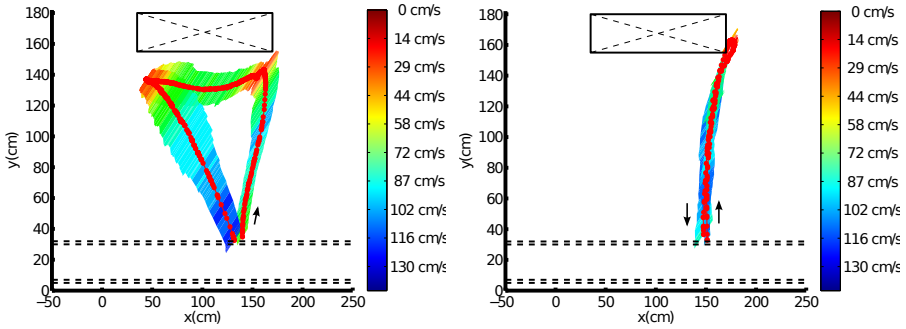
Fig. 5.10 shows the mean and prototypical velocity along the prototypical route for each type of work cycle. Fig. 5.5a and 5.5c show mean and prototypical velocity for first type of work cycle respectively, and Fig. 5.5b and 5.5d are for work cycle 2. The instantaneous velocity is color coded. Both Fig. 5.10b and 5.10d show low velocity in red at the right corner of the storage rack. Both Fig. 5.10a and 5.10c show low velocity at both of the picking up places (the




**(a)** Mean velocity for Work cycle 1 **(b)** Mean velocity for work cycle 2

**(c)** Prototypical velocity for work cycle 1 **(d)** Prototypical velocity for work cycle 2

**Figure 5.10:** *Mean and prototypical velocity along routes.* (a) and (b) show the mean velocity along the prototypical routes for each type of work cycle respectively. (c) and (d) are for the prototypical velocity.




**(a)** Work cycle 1 **(b)** Work cycle 2

**Figure 5.11:** *Velocity variance along the prototypical routes.* The width of the band indicates the variance in local velocity, whereas the color indicates the prototypical velocity itself.

left and right corner of the storage rack). In Fig. 5.5c the prototypical velocity, which is calculated using the stretch-and-then-compress method, is as high as around 60 cm/s. This does not agree to the behavior of the factory workers there: standing still and picking up tools and parts. It proves that the greedy method based on successor classification aligns the tracks in a better way, creating more representative prototypical route and velocity.

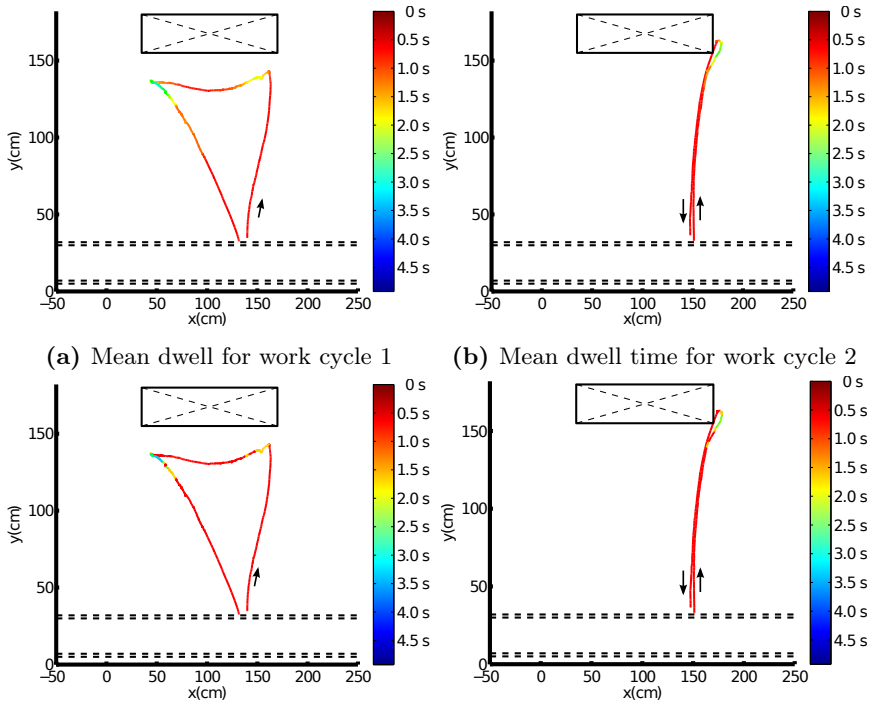
Fig. 5.10a shows the value of the mean velocity at the right corner of the storage rack reaching 35 cm/s. But from Fig. 5.10c, we can see the prototypical velocity there dropping to 20 cm/s. Obviously, the prototypical velocity provides the clearest picture there and its value is lower, which better agrees with the observations that workers actually stop near these locations so as to pick up tools and objects.

Fig. 5.11 shows the velocity variance along the prototypical routes. For work cycle 1, the band along part of the prototypical route that factory workers go to the storage rack is wider than the other part of the band that they come back from the storage rack to the assembly line. It indicates that the factory workers keep their velocities more stably on their way to the storage rack than back to the assembly line.

Fig. 5.12 shows the mean and prototypical dwell time along the prototypical route for each type of work cycle. The instantaneous dwell time is color coded. Both Fig. 5.12b and 5.12d show that the dwell time at the right corner of the storage rack is around 2.5 seconds for work cycle 2, the same as Fig. 5.7b and 5.7d. Fig. 5.12a shows that the average dwell time is around 2.0 seconds at the right corner of the storage rack, and around 2.5 seconds at the left corner. From Fig. 5.12c, we can see that the factory workers stays for 1.5 seconds at the right corner and for 3.5 seconds at the left corner prototypically. The prototypical dwell time at the left corner of the storage rack agrees with the actual dwell time there. Therefore, the prototypical dwell time, which is obtained by aligning the tracks using the greedy method based on successor classification, can guide the workers better to pick up the tools and parts than that calculated using the stretch-and-then-compress method.

## 5.4 Comparison

Researchers have proposed some methods to average many tracks, as elaborated in Section 3.1. We apply these methods on the factory worker trajectory to exact the average track as the prototypical route, as shown in Fig. 3.28 and 3.28. Junejo only considers the boundaries of the points of the tracks along the route, but ignores the spatial distribution of the points between the boundaries. Therefore, the average track extracted is representative for work cycle execution. Petitjean produces a very flexuous track, which is not suitable to guide the workers to execute their work cycles, and the average track extracted using Niennattrakul's method is more representative and smooth than the other two methods, but the computational cost is quite high. Compared to these methods, both of our methods are not only able to average many tracks, but also




**(a)** Mean dwell for work cycle 1

**(b)** Mean dwell time for work cycle 2

**(c)** Prototypical dwell time for work cycle 1

**(d)** Prototypical dwell time for work cycle 2

**Figure 5.12:** *Mean and prototypical dwell time along routes.* (a) and (b) show the mean dwell time along the prototypical routes for each type of work cycle respectively. (c) and (d) are for the prototypical dwell time.

benefit the statistical analysis. Using the point associations established during the track alignment, we calculate the prototypical velocity and dwell time along the prototypical route, while other methods mentioned are designed to average the tracks without considering the point correspondences among all tracks.

## 5.5 Conclusions

In this chapter, we proposed to extract the prototypical route and prototypical velocity and dwell time along the route using the track alignment as elaborated in Chapter 3. The track alignment enables us to average the warped tracks to form a prototypical route from the tracks, which describes how the workers execute the work cycle spatially. It is also beneficial to analyze the tracks statistically, such as location and velocity variance along the prototypical routes.

This research resulted in on several papers in the proceedings of the international conferences [Xie 14b, Xie 15a, Xie 15b].

# 6

# Road Network Inference from GPS Traces

Automatic road map inference is an important tool in the field of Intelligent Transportation Systems (ITS): it allows unexplored geographic regions (e.g., in developing countries) to be mapped quickly [Biagioni 12b, Schroedl 04, Davics 06, Cao 09], it can update the existing maps [Shi 09, Sato 06], and it also provides information on the density of traffic [Vlahogianni 05], which can be used in navigation [Syberfeldt 09] and for urban planning [Schultes 08, Niehoefer 09, Morris 04].

Road maps were previously constructed through labor intensive geographic surveying using telescopes, sextants and other devices. Today, these methods have been replaced by mobile mapping vehicles, which can map whole cities with high accuracy [Kukko 07]. However, mobile mapping campaigns are expensive, and as a result, the data is updated relatively infrequently. Moreover, mobile mapping does not provide traffic related info, such as the traffic density as a function of time, the preferred routes of travelers, and the delays due to traffic jams.

Thanks to the ubiquitous use of Global Positioning System (GPS) devices, digital road maps can now be derived from GPS traces of various road users [Bellens 11, Lee 13]. As hand-held GPS devices have become increasingly popular in the last decade, geographical data is more easily obtainable from not only cars, taxis and trucks, but also from cyclists and pedestrians. This abundance of GPS derived geospatial data has stimulated the development of both crowd-sourced mapping projects [Haklay 08, Haklay 10], as well as commercial products. The research on GPS trajectory analysis has focused on building road maps [Biagioni 12a] and learning people's mobility modes [Zheng 08a]. Other research focuses on calculating the best path between two locations [Edelkamp 03, Schultes 08, You 14] or finding the most efficient route for a garbage truck [Syberfeldt 09].

Road networks are a critical aspect of both path optimization and route planning. In literature, a variety of techniques are used to generate the road network from GPS traces, from image processing to network topology

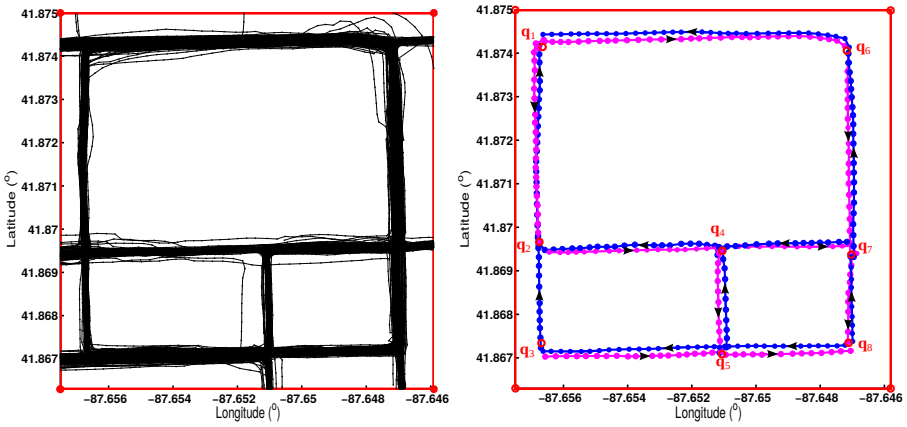
[Davics 06, Biagioni 12b, Shi 09, Chen 08, Edelkamp 03, Schroedl 04]. Most existing algorithms generate the digital map as a whole directly from GPS traces, but most omit one important component of the road network: intersections. A few researchers address the problem of intersection detection after the map generation [Edelkamp 03,Schroedl 04,Cao 09]. The accuracy of the extracted roads depends on quality of the GPS traces. It is difficult to remove the abnormal GPS traces during the road network generation because different GPS traces may cover different road segments. If the intersections are detected in advance, the road network structure can be then inferred conveniently by discovering the intersection connectivity. By segmenting the GPS traces into track pieces for individual road segments, the abnormal tracks can be easily detected for each road segment using similarity measure. This will improve the accuracy of the generated road representation. Therefore in this chapter we will first detect the intersections from GPS traces, then calculate the geometric representation for each road segment using the GPS track pieces belonging to this road segment.

The main elements of a more general transportation network include intersections, roads, railways, highways, motor vehicular lanes, bicycles and pedestrians lanes. However we will only explore a subset of these elements in the form of a road network, which is a system that represents the interconnecting roads located in a given area. In order to clarify the scope of our work, we present the three elements that define a road network below.

- **Intersections**: Road intersections are represented by their geographic position $\mathbf{q} = (\varphi, \lambda)^T$, where $\varphi$ and $\lambda$ are coordinates in latitude and longitude. We give two intersection definitions: 1) An intersection is defined as a location where road users can change directions in multiple ways, regardless of the number of road segments meeting at a particular intersection. 2) An intersection is defined as a junction where at least three road segments converge.

- **Connectivity graph**: Road connectivity graphs encode which intersections are directly connected by road segments containing no other intersections. These graphs are typically represented by a binary $M \times M$ connectivity matrix $D$, with $M$ the number of intersections. By definition, $D(i, j) = 1$ if intersections $i$ and $j$ are directly connected by a single road segment. In our paper, $D$ is asymmetric, i.e. $D(i, j)$ can differ from $D(j, i)$ due to one-way traffic. Moreover, we assume that the main diagonal elements of $C$ are 0, i.e., an intersection is not connected to itself.

- **Road segments**: The directed road segments $R$, connect pairs of intersections. The geometry of each road segment is represented using a sequence of geographical locations. The average velocity and velocity variance along each road segment will be analyzed using the alignment of tracks belonging to it. However, the type of the road, and the number of lanes on the road will not be analyzed.




**(a)** GPS traces **(b)** Inferred road network

**Figure 6.1:** *An example of a road network inferred from GPS traces.* 8 intersections and 19 directed road segments are inferred from GPS traces.

Fig. 6.1 shows an example of a road network. As shown in Fig. 6.1b, 8 intersections are indicated using red circles. 9 pairs of intersections are directly connected to each other in both directions. The road between $\mathbf{q}_2$ and $\mathbf{q}_3$ is one-way. In total there are $9 \times 2 + 1 = 19$ directed road segments in this road network. The geometric representation of each road segment is a series of locations indicated in dots.

In this chapter, we propose two methods to identify the intersections using their two definitions, respectively. In our first method, intersection candidates are detected by clustering turning points of traces, i.e., places where the road users change direction. It is possible to falsely detect bends as intersections on a single windy road. At the bend, the road users always change their direction in the same way. By checking the entering and exiting directions of the road users, the bends are removed from the intersection candidates. In the second method, we detect the intersections through finding the Longest Common Sub-Sequence (LCSS) between pairwise GPS traces. The starting and ending points of the sub-tracks, which are common to both traces, are collected as connecting points, and the local maximums of their Kernel Density Estimation (KDE) are recognized as intersections.

The connectivity between each pair of intersection is then deduced from the GPS traces, which are segmented into tracks corresponding to road segments between each pair of directly-connected intersections. For clarity, a trace is defined as a trajectory of a single moving vehicle over a certain time period recorded by a single GPS device. A track is defined as a small piece of trajectory for one road segment, segmented from the traces by directly-connected intersections. The tracks for each road segment are aligned using the trajectory alignment algorithms we proposed in Chapter 3. The warped tracks produced

are used to extract an an average track to act as the geometric representation of the road segment. In addition, the average velocity and velocity variance along each road segment are analyzed based on the track alignment.

Biagioni comparatively evaluated some of the existing methods for road network inference [Biagioni 12a] on a middle-size data set: Chicago dataset with 889 GPS traces. Furthermore, Ahmed *et al.* tested some methods on a large-size data set: Berlin dataset with 27189 GPS traces [Ahmed 14]. We will compare our methods with their work on both datasets. Our results show better performance with eliminating many spurious roads as well as more accurate geographical locations for our detected roads.

The rest of the chapter is structured as follows: In Section 6.1, we discuss the related work. In Section 6.3, two methods are proposed to detect the intersections from GPS traces: one is based on turning points detection; the other one on connecting points detection. In Section 6.4, we utilize the alignment methods proposed in Chapter 3 to align the tracks together for each road segment and estimate the geometric representation of the road segment using the warped tracks produced during the track alignment. In Section 6.6, we detail how to evaluate the topological accuracy and geographical accuracy of the extracted road network. In Section 6.6, we experimentally test our proposed algorithms and compare them with other existing methods. Section 6.7 concludes this chapter.

## 6.1 Related work

### 6.1.1 Approaches for Road Network Inference

The goal of road network inference is to automatically generate a directed graph from raw GPS traces, representing the topology and geometry of the road network. Depending on how the GPS traces are processed, the approaches for road network inference can be broadly classified into two groups:

**Methods operating on a binary image created from GPS traces.** These methods first divide the the geographical area covered by GPS traces into a two-dimensional grid of cells and estimate the Kernel Densities (KD) of the tracking data points for each cell [Davics 06, Biagioni 12b, Shi 09, Chen 08]. A binary representation of all tracks is then produced by applying a threshold on the KD Estimation (KDE). These methods differ in how the road centerlines are extracted from the resulting binary image. Davies *et al.* apply a contour follower to the binary image to extract a set of closed polygons which describe the road regions' outline, and then compute the road centerlines by producing a Voronoi graph of the contours describing the road edges [Davics 06]. Chen *et al.* use an image-processing approach to extract the road map from the binary image [Chen 08]. First, a morphological dilation and closing operations are used to merge the discrete data points of GPS traces. A thinning operation is then used to produce the skeleton along the road centerlines. Shi *et*

*al.* propose a very similar method to Chen's approach [Chen 08], but they try to extract the crossings of the roads from the road network skeleton [Shi 09]. Biagioni and Eriksson do not produce binary map image by applying a simple threshold to the KDE, because a single threshold cannot achieve both high accuracy and high coverage of road map. To solve this problem, they apply a gray-scale skeletonization technique to extract a threshold-free skeleton from the KDE [Biagioni 12a].

Methods based on KDE suffer from the limitations of thresholding: a too low threshold will produce spurious edges but a too high threshold will lead to interrupted tracks in sparse areas, leading to unsuccessful detection of the roads that are not traversed frequently. Although the geometry of the road network is built using the geographical locations along the road centerlines, the topology of the road network, formed by the interconnections between roads, is not analyzed thoroughly in this type of binary image based road analysis. Nevertheless, accurate road network topology information is essential to path optimization and route planning. In our work, we operate on the traces (spatial positions as a function of time) directly in order to extract road intersections and analyze their connectivity using GPS traces.

**Methods operating on the data points of GPS traces** Given a set of track pieces, a variety of approaches, ranging from curve fitting to graph segmentation, have been proposed for extracting a representative road segment from its corresponding track pieces. These approaches can be divided into three categories by their algorithmic foundations.

- *Curve fitting methods.* Edelkamp and Schroedl employ a K-means algorithm to cluster the data points of raw GPS traces based on the Euclidean distance measure. The cluster seeds are merged to road segments, and the precise centerline for each segment is generated from the GPS data points corresponding to it using a weighted least squares fitting [Edelkamp 03,Schroedl 04]. Worrall and Nebot cluster the GPS data points into regions of similar position with similar headings, and apply non-linear least squares fitting to exact arcs and lines from the cluster data [Worrall 07].

- *Topological methods* Morris et al. [Morris 04] build a topological graph to represent the physical network of the GPS traces. An initial graph is generated from the GPS points and connection lines between adjacent GPS points in all GPS traces. Those connection lines are reduced to extract a single representation for each road segment using graph algorithms such as parallel reduction, face reduction, serial reduction and edge contraction. A parallel reduction takes two parallel edges in the graph and reduces them to a single edge. Faces of a graph are regions bounded by the edges of the graph. Any face in the graph is reduced if all of its components are sufficiently close, which is called face reduction. A serial reduction eliminates a vertex with only two outgoing edges. Edge contraction deletes spurious edges under some criteria.

- *Trace merging methods.* Cao and Krumm [Cao 09] propose to reduce the effects of GPS noise using simulations of physical forces among the traces, and merge the cleaned GPS traces greedily into a graph. Edges from each raw GPS trace are added to the graph, unless an edge with similar location and bearing already exists in the graph under construction. Intersections are then detected from the generated graph. A very similar method is used by Niehoefer *et al.*, which merges each new trace to an existing map and updates the position of existing roads [Niehoefer 09]. Ahmed and Wenk develop an incremental method that employs the FrÃľchet distance to match partial trajectories to a graph [Ahmed 12].

All of the methods mentioned above suffer from producing spurious road edges at areas that contain relatively tall buildings and significant GPS error. Our proposed methods infer the topology of the road network through intersection identification, and proceed to extract the geometric representation of each road segment through track alignment. They outperform the existing methods in producing much less spurious road edges and a better geographical accuracy of the inferred road segments.

### 6.1.2 Approaches for Intersection Detection

Detecting intersections before road map generation benefits to build the topology of the road network through analyzing the intersection connectivity.

Fathi and Krumm design a localized shape descriptor to represent the distribution of GPS traces around a point [Fathi 10]. A classifier is trained over the shape descriptors on ground truth data, and used to discriminate intersection points from non-intersection points. Their method requests ground-truth training samples, and high-sampling-rate traces.

Karagiorgou and Pfoser use a speed threshold in combination with a change in direction to detect the turn samples from GPS traces [Karagiorgou 12]. The vector from the current point to its next adjacent point is used to describe the moving direction at this point. The turn samples are clustered into intersection nodes using an agglomerative hierarchical clustering method and a distance threshold. At last, GPS traces between intersection nodes are linked to create road edges.

Wu *et al.* first find turning points from coarse-gained GPS traces using the moving direction provided by the GPS devices [Wu 13]. The intersecting points, where the turning points converge, are gathered to improve the concentration of the turning points. They cluster the converging points into intersections using X-means algorithm [Pelleg 00], which is similar to K-means but does not require a pre-specified $K$. The turning points are detected simply by thresholding the change of moving direction. Fluctuations of moving direction are not distinguished from the turning points in their results.

Wang *et al.* first detect *conflict points*, which are points where two or more traces cross, converge or diverge. The conflict points are used to compute spatial position and an uncertainty bound for each intersection [Wang 15].

In our first proposed method, intersections are defined as the locations where the road users change their moving directions, as in [Karagiorgou 12, Wu 13]. Turning points are collected and clustered into intersections. However, the moving direction at each GPS point is calculated using the point at a fixed distance ahead of it, instead of the adjacent point. Mis-detected bends are removed from the intersections by checking the turn types the road users make at the turning points.

In the second method we proposed, intersections are defined as junctions which connect different road segments. The starting and ending points of the sub-tracks , which are common to each pair of traces are detected as connecting points, and used to detect intersections based on their KDE. The connecting points connect three road segments in different directions. Therefore bends which only connect two road segments, will not be recognized as intersections. The method also overcomes the limitation of the thresholding methods for turning point detection.

## 6.2 Overview of our Proposed Approach

Fig. 6.2 presents an overview of our approach. First, we detect intersections from GPS traces. Two methods are proposed to identify the intersections. In the first method, we detect turning points from the traces according to the moving direction of the road users, and cluster the turning points into intersection candidates. Bends are distinguished from the intersections by checking their turn patterns. The second method we proposed defines intersections as junctions which connect at least three road segments. LCSS is used to detect common sub-tracks between pairwise GPS traces, whose starting and ending points are gathered as connecting points, and used to detect intersections based on their KDE.

Second, we analyze the connectivity of the intersections, and segment the GPS traces into track pieces directly connecting two intersections. GPS device users often follow different routes to the same destination, creating different GPS traces with varying velocities and locations. However, the traces share common sub-tracks corresponding to individual road segments.

Third, the tracks for each road segment are clustered based on their spatial similarity. Only normal tracks, which are spatially similar to each other, will be used to infer the geometric representation of the road segment.

cluster the tracks using their spatial similarity and align the normal tracks

Third, we jointly align the normal tracks for each road segment using the trajectory alignment algorithms presented in Chapter 3. The points associations established are used to average the tracks to form a geometric representation for the road segment, and analyze the tracks statistically.

Lastly, we evaluate the topological and geographical accuracy of the inferred road network.

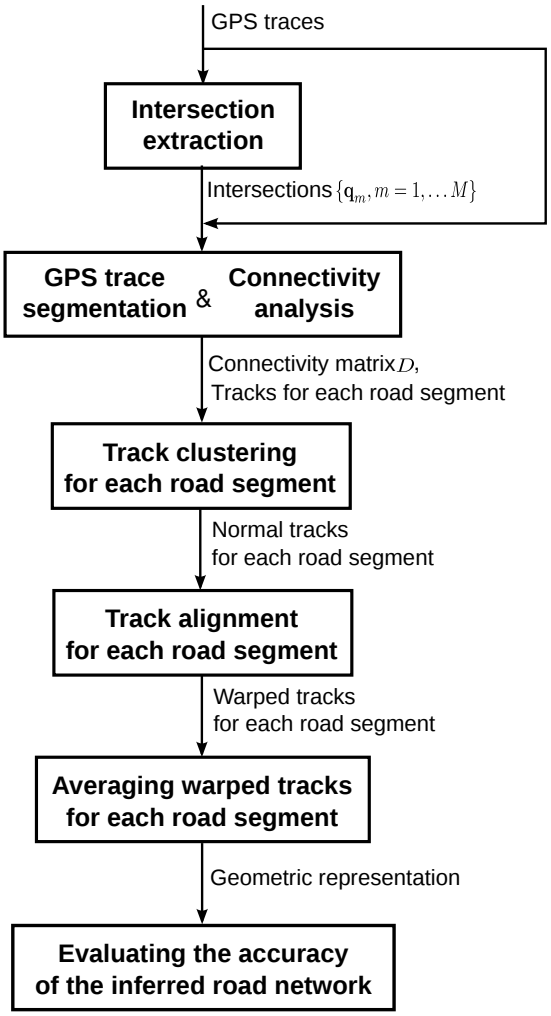



**Figure 6.2:** *Overview of the proposed approach.*

## 6.3 Intersection and Connectivity Detection

Intersection detection is crucial to build the topology of the road network. In this section, we propose two methods to identify the intersections from GPS traces: one is based on turning points detection; the other one on analyzing common sub-sequences to identify connecting points, then detect intersections from the connecting points.

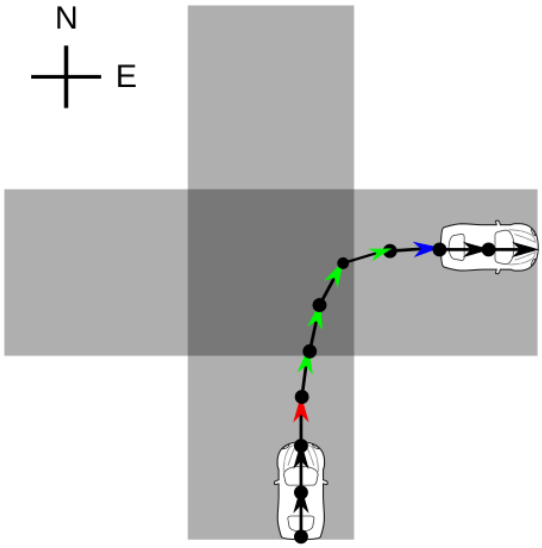



**Figure 6.3:** *An example of a turn* A part of a GPS trace around one intersection is represented by a black line with dots indicating the positions of data recordings, and arrows indicating the moving direction at each point.

### 6.3.1 Intersection Detection based on Turning Points

We first calculate the moving direction at each GPS sample by locating the first future sample which is approximately a fixed distance from it, as will be explained in sect. 6.3.1.1. Turning points are detected as locations where the road users change their moving directions. We group the turning points into intersection candidates. Bends are distinguished from real intersections by analyzing turning patterns.

#### 6.3.1.1 Turning Point Detection

The rationale of our turning point detection is that while some road users may travel straight through intersections, at least some of them will turn onto other roads at intersections. This turning behavior is not unique to intersections, as road users may also appear to make turns at bends in roads which are not intersections. Therefore, when we detect spatial locations where many tracks show turning points, we also perform additional analysis to distinguish bends in roads from true cross roads, which is described in Section 6.3.1.2.

Fig. 6.3 shows an example of a vehicle making a right turn at an intersection. The moving directions of the turning points are shown with colored arrows, where red indicates the first turning point, blue indicates the last turning point, and green for other turning points. The moving directions of other points without a direction change are shown as black arrows. The example shows that the moving direction of the road user keeps changing during the turn.

Therefore, the intersection can be detected by checking the direction change of the road users. If the road users always make the same type of turn by changing their moving direction from north to east, or following the same route but in the opposite direction, this is a bend, rather than an intersection. In this case, their entering directions will always be the same as the red arrow, or be the opposite of the blue arrow. The exiting directions will always be the same as the blue arrow, or be the opposite of the red arrow. The mis-detected bends can thus be removed by examining the entering and exiting directions.

When road users make turns at intersections, the change in their moving direction over a fixed amount of time is different depending on their speed. Thus it is difficult to detect turns by thresholding instantaneous direction change. Instead, to make the method robust against speed differences, we calculate the change in the moving direction over a fixed distance. If the fixed distance is too small, the direction change at the intersections cannot be distinguished from minor fluctuations in moving direction, e.g., due to changing lanes or to small bends in the road.

Based on our experience, direction change over a distance of at least 2 meters at the intersection can be adequately distinguished from other changes. However, depending on the sampling rate of the GPS units and the speed of the road users, it is possible that the distance between two adjacent GPS points is larger than 2 meters. Therefore, the moving direction of the user at $(\varphi_a, \lambda_a)$ is computed as follows:

Given a point on a GPS trace, $(\varphi_a, \lambda_a)$, we select a second point, $(\varphi_b, \lambda_b)$, that is 2 meters ahead on the same trace, if the next point is further than 2 meters away, we select the next point.

The moving direction of the user at $(\varphi_a, \lambda_a)$ is computed as

$$\theta_a = \begin{cases} \arctan \frac{\lambda_b - \lambda_a}{\varphi_b - \varphi_a} & \varphi_b > \varphi_a \\ \arctan \frac{\lambda_b - \lambda_a}{\varphi_b - \varphi_a} + \pi & \lambda_b \geq \lambda_a, \varphi_b < \varphi_a \\ \arctan \frac{\lambda_b - \lambda_a}{\varphi_b - \varphi_a} - \pi & \lambda_b < \lambda_a, \varphi_b < \varphi_a \\ -\frac{\pi}{2} & \lambda_b > \lambda_a, \varphi_b = \varphi_a \\ +\frac{\pi}{2} & \lambda_b < \lambda_a, \varphi_b = \varphi_a \end{cases}, \tag{6.1}$$

where $\theta_a \in [-\pi, \pi]$ ($-\pi$ is west, radians counter-clockwise).

The moving direction of the user, $\theta_b$, at the point, $(\varphi_b, \lambda_b)$, is calculated similarly using the the next point on the GPS trace that is a fixed distance away. If the direction change $\Delta\theta_b = \theta_b - \theta_a$ between the points $(\varphi_a, \lambda_a)$ and $(\varphi_b, \lambda_b)$ exceeds a predefined threshold, then it is considered as a turning point.

Since intersections are typically much larger than our 2 meter turn detection distance threshold, many GPS points with direction changes can be detected inside the same intersection. We consider all of the neighboring data points with direction changes detected from GPS traces as turning points. We keep track of the first and last point in a sequence of moving direction changes, which can represent the user entering and exiting the intersection respectively, which will be later used to distinguish bends in roads from true intersections.

**Algorithm 6** Cluster Turning Points

**Input:** $\{\mathbf{p}_k,\, k = 1 \dots K\}$
**Output:** $\{P_i : |P_i| > T_{\min} \quad i = 1 \dots I\}$
1: Initialization $P_{un} \leftarrow \{\mathbf{p}_k,\, k = 1 \dots K\} \quad i \leftarrow 1$
2: **for** each $\mathbf{p} \in P_{un}$ **do**
3: $P_i = \{\mathbf{p}\}$
4: **for** each $\mathbf{p}' \neq \mathbf{p} \in P_{un}$ **do**
5: **if** $\frac{1}{|P_i|} \sum_{\mathbf{p}'' \in P_i} d(\mathbf{p}', \mathbf{p}'') \leq d_{thre}$ **then** $d_{thre}$ **is the predefined threshold for the average Euclidean distance.**
6: $P_i \leftarrow P_i \cup \{\mathbf{p}'\}$
7: **end if**
8: **end for**
9: $P_{un} = P_{un} \setminus P_i$
10: **end for**

The output of this step is a collection of turning points $\{\mathbf{p}_k, k = 1, \dots K\}$, along with binary indicators of whether the user enters the intersection $\{e_k^{(1)}, k = 1, \dots K\}$ or exits the intersection $\{e_k^{(2)}, k = 1, \dots K\}$ at a particular turning point, where $e_k^{(1)} \in [0, 1]$ and $e_k^{(2)} \in [0, 1]$. $e_k^{(1)} = 1$ indicates that the user enters an intersection at the turning point $\mathbf{p}_k$, and 0 indicates not entering. We describe how these turning points will be clustered into intersections in next section.

#### 6.3.1.2 Intersection Extraction from Turning Points

In our work, we propose a clustering technique to group the collection of turning points $P$ into intersections based on Euclidean distance. As shown in Algorithm 6, an initial cluster is grown from a seed point by iteratively adding points to the cluster if their average distance to the current points in the cluster is small enough. This procedure is then repeated on the points which have not yet been clustered. The output of this step is a set of clusters $P_i = \{\mathbf{p}_n^{(i)}, n = 1 \dots |P_i|\}$, with $|P_i|$ as the number of turning points per cluster. As a post processing step we also discard clusters which contain too few turning points; these may be caused by GPS noise, or by an insufficient sampling of an intersection.

To remove road bends mis-detected as intersections from the clusters, we need to check for the type of turns for each cluster using the entering (with $e_k^{(1)} = 1$) and exiting turning points (with $e_k^{(2)} = 1$) as described in previous step. For each intersection candidate, we select its entering and exiting turning points and cluster their directions separately, as shown in Fig. 6.4. If all entering points are grouped into one cluster and all exiting points into one cluster, this intersection candidate is a bend on a one-way road segment, At this bend, the road users can only make the same type of turn in one direction. If the entering points are grouped into two clusters, as well as the exiting points, and

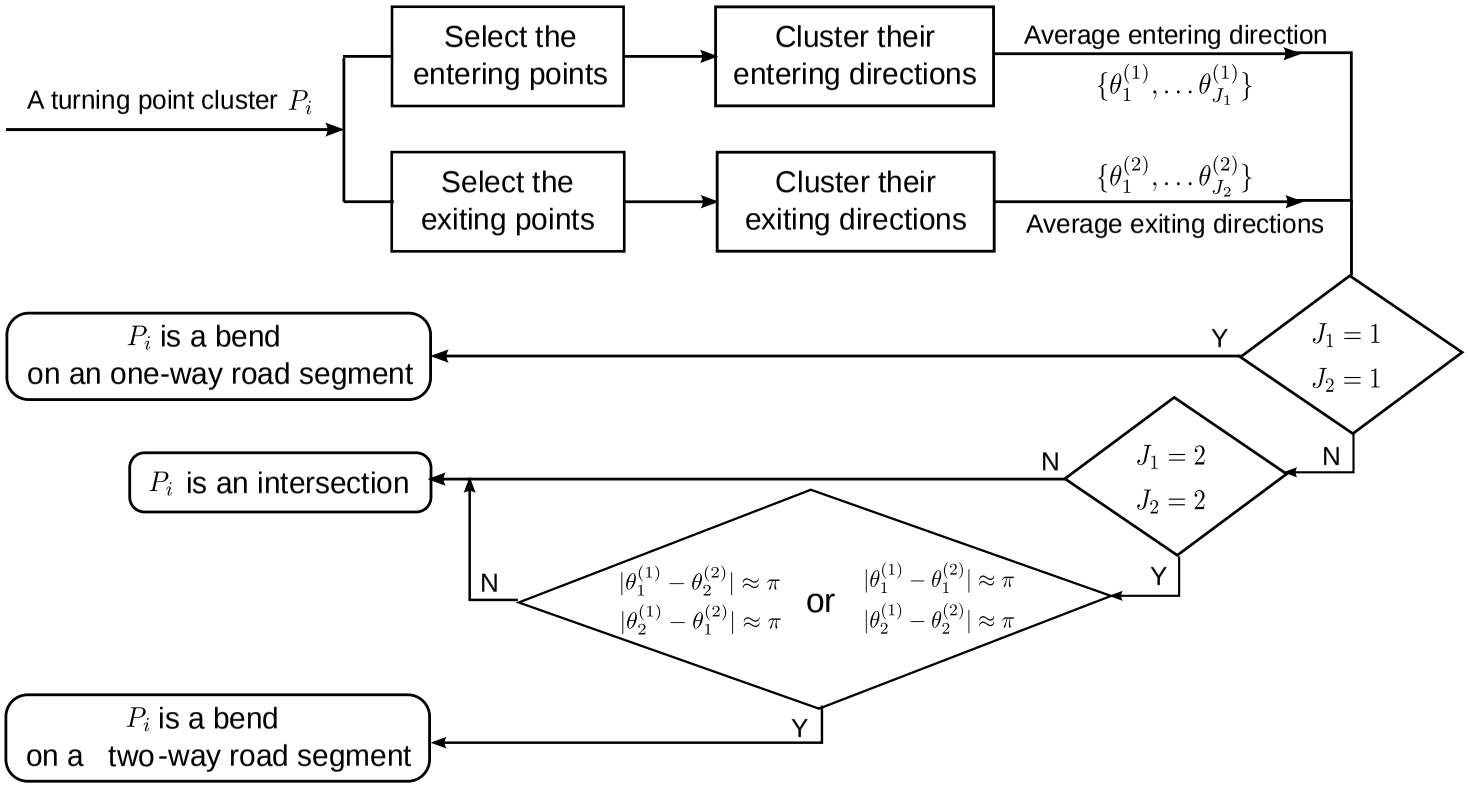



**Figure 6.4:** *Bend detection* Bends are detected by checking the entering and exiting directions.

the average entering directions are opposite to the average exiting directions, this intersection candidate is a bend on a two-way road segment. At this bend the road users can only make one type of turn in two directions. These misdetected bends will be removed from the intersections.

Finally the spatial locations of the clusters are obtained by averaging the turning points in each cluster. The result is the set of intersections $Q^{(1)} = \{\mathbf{q}_i^{(1)}, i = 1, \dots I^{(1)}\}$. The superscript (1) refers to the algorithm of intersection detection through moving direction analysis. In the next section, we introduce the second method to detect the intersections.

### 6.3.2 Intersection Detection based on Connecting Points

In this section, intersections are detected using their direct definition of connecting road segments, instead of indirectly analyzing the turning patterns of the road users. The road users often travel through a sequence of road segments to arrive at their destination. They may travel through the same road segments on their journeys. As a result, the generated GPS traces may merge from different road segments onto the same road segment, or diverge from the same road segment onto different road segments. The converging and diverging points in the GPS traces are located on the intersections, if they are not the starting and ending points of the GPS traces. Therefore, we can detect the intersections from converging and diverging points by finding the common sub-tracks of the GPS traces. A common sub-track is a sequence of points that appears in the same relative order and occupies consecutive positions in both original traces. Each common sub-track corresponds to a shared road segment. One end of the common sub-track is the diverging point, and the other end is converging point.

Fig. 6.5 shows an example of two GPS traces diverging at one intersection and then converging at another intersection. The first trace, which contains 16 points, is shown using blue lines with blue dots indicating the point locations. There are 12 points in the second trace, shown in red lines with green dots. The arrows show that the road users move from higher-latitude area to lower-latitude area in both trajectories. During these two journeys, both of the road users travel through three road segments. The first road segments they travel through are the same. They separate onto different road segments at the first intersection indicated using a black star. At the second intersection indicated using a black triangle, they come to their third road segments, which are the same on the map. The generated GPS traces diverge at the first intersection and converge at the second intersection.

In this section, we aim to find the common sub-tracks of these two GPS traces, which correspond to shared road segments, and identify intersections from the starting and ending points of these sub-tracks.

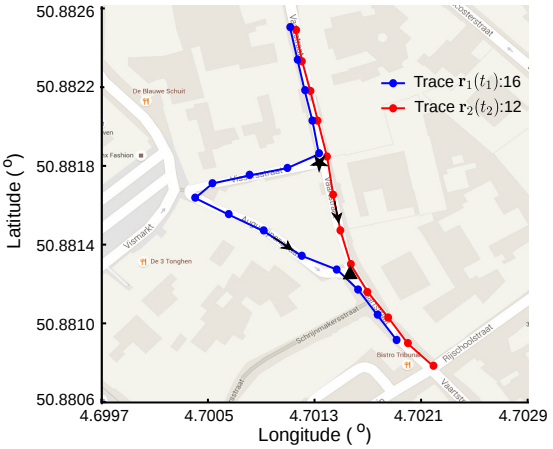



**Figure 6.5:** *An example of two GPS traces sharing roads* Two GPS traces diverge from one common road segment to different road segments at the first intersection, which is indicated using a black star, and then converge to another common road segment at the second intersection indicated using a black triangle.

#### 6.3.2.1 Longest Common Subsequence Detection

Given two GPS traces $\mathbf{r}_1(t_1)$, $t_1 = 1 \dots T_1$ and $\mathbf{r}_2(t_2)$, $t_2 = 1 \dots T_2$, we aim to find the alignment between these two traces that maximize the number of the points in common subsequence [Hirschberg 77, Smith 81] using Longest Common Subsequence (LCSS) algorithm.

A trace subsequence is a sequence of data points which appear in the same relative order within the original trace, but not necessarily occupies consecutive positions as a sub-track does. A common subsequence of two traces is a subsequence present in both of them. A longest common subsequence is a common subsequence of maximal length. The "length" in this research means the number of points of the trace, or trace subsequence. The most naive approach to find the longest subsequence would be to enumerate all subsequences common to both traces, and select the one with the most points. However, it is extraordinarily time-consuming to apply this naive approach. To overcome this challenge, Dynamic Programming is employed to find the longest subsequence efficiently, which is similar to the implementation of DTW in Section 2.2.

The first stage of implementing Dynamic Programming in the LCSS problem is to create a two-dimensional (2D) length matrix $L$, instead of a overall dissimilarity matrix in DTW. The value at each cell, $L(t_1, t_2)$, represents the length of the LCSS between the prefixes of the given GPS traces, i.e. $\mathbf{r}_1(i)$, $i = 1 \dots t_1$ and $\mathbf{r}_2(j)$, $j = 1 \dots t_2$. As shown in Equation 6.2, the value of $L(t_1, t_2)$ depends on the similarity of the current points $\mathbf{r}_1(t_1)$ and $\mathbf{r}_2(t_2)$, and the value of its adjacent cells $\{L(t_1, t_2 - 1), L(t_1 - 1, t_2), L(t_1 - 1, t_2 - 1)\}$:

**Algorithm 7** FindPath

**Input:** $L$, $B$
**Output:** $\mathbf{g}_1(k)$, $\mathbf{g}_2(k)$, $w_2(k)$, $w_2(k)$, $k = 1 \ldots K$

```
    Initialization t1 ← T1, t2 ← T2, k ← 1
 2: while t1 + t2 > 2 do
        if t1 = 1 then
 4:         t2 ← t2 − 1
        else if t2 = 1 then
 6:         t1 ← t1 − 1
        else
 8:         switch B(t1, t2)
                case "↖"
10:                 g1(k) = r1(t1), g2(k) = r2(t2)
                    w1(k) = t1,w2(k) = t2
12:                 k ← k + 1
                    t1 ← t2 − 1, t2 ← t2 − 1
14:             case "↑"
                    t1 ← t1 − 1
16:             case "←"
                    t2 ← t2 − 1
18:     end if

20: end while
    if L(1, 1) = L(t1, t2) − 1 then
22:     g1(k) = r1(1), g2(k) = r2(1)
        w1(k) = 1,w2(k) = 1
24: end if
    g1 ← reverse(g1), g2 ← reverse(g2)
26: w1 ← reverse(w1), w2 ← reverse(w2)
```

$$L(t_1,t_2) \triangleq \begin{cases} 0, & \text{if } t_1 = 0 \text{ or } t_2 = 0 \\ L(t_1-1,t_2-1)+1, & \text{if } t_1,t_2 > 0 \text{ and } d(\mathbf{r}_1(t_1),\mathbf{r}_2(t_2)) \le d_{thre} \\ \max\begin{pmatrix} L(t_1,t_2-1), \\ L(t_1-1,t_2) \end{pmatrix}, & \text{if } t_1,t_2 > 0 \text{ and } d(\mathbf{r}_1(t_1),\mathbf{r}_2(t_2)) > d_{thre}, \end{cases} \tag{6.2}$$

where $t_1 \in [0, T_1]$ and $t_2 \in [0, T_2]$, and $d(\mathbf{r}_1(t_1), \mathbf{r}_2(t_2))$ is the distance between $\mathbf{r}_1(t_1)$ and $\mathbf{r}_2(t_2)$.

We also create a two-dimensional (2D) arrow matrix $B$. The arrow $B(t_1, t_2)$ at a cell points to the adjacent cell $L(t_1, t_2)$ it is calculated from.

$$B(t_1,t_2) \triangleq \begin{cases} \uparrow, & \text{if } L(t_1,t_2) = L(t_1-1,t_2) \text{ and } L(t_1-1,t_2) \geqslant L(t_1,t_2-1) \\ \leftarrow, & \text{if } L(t_1,t_2) = L(t_1,t_2-1) \text{ and } L(t_1-1,t_2) < L(t_1,t_2-1) \\ \nwarrow, & \text{if } L(t_1,t_2) = L(t_1-1,t_2-1)+1, \end{cases} \tag{6.3}$$

where $t_1 \in [1, T_1]$ and $t_2 \in [1, T_2]$.

Once the length matrix L and direction matrix B are built, the LCSSs are deduced in a "backtrack" procedure that follows the arrows backwards through matrix. The procedure starts at the last cell at $(T_1, T_2)$, as described by Algorithm 7. If the length at the cell $(t_1, t_2)$ decreases, point $\mathbf{r}_1(t_1)$ in the first trace is similar, i.e. spatially close, to point $\mathbf{r}_2(t_2)$. These data points are added to the longest subsequence, $\mathbf{g}_1$ and $\mathbf{g}_2$, for each GPS trace respectively, and their time indexes in the original traces are added to $w_1$ and $w_2$, i.e. i.e. $\mathbf{g}_1(k) = \mathbf{r}_1(w_1(k))$ and $\mathbf{g}_2(k) = \mathbf{r}_2(w_2(k))$. The procedure ends until it reaches the first cell $(1, 1)$.

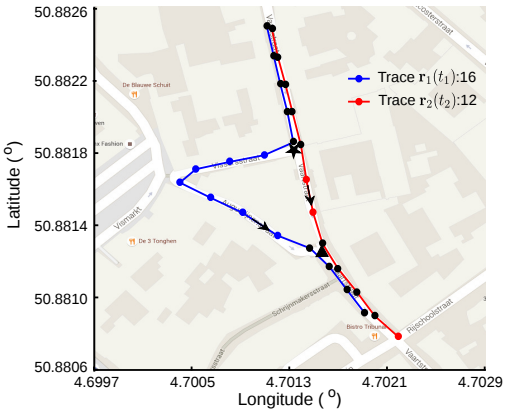
Fig. 6.6 illustrates the procedure of finding the LCSSs between two GPS traces. Fig. 6.6a depicts the LCSS for each GPS trace using blue lines with black dots, and red lines with black dots, respectively. In total, there are 9 points in the LCSS for each trace. The LCSS for the first trace is expressed as $\mathbf{r}_1(t_1)$, $t_1 = 1, 2, 3, 4, 5, 13, 14, 15, 16$, and $\mathbf{r}_2(t_2)$, $t_2 = 1, 2, 3, 4, 5, 9, 10, 11, 12$ is for the second one.

Fig. 6.6b depicts the length matrix $L$ and arrow matrix $B$ in grids. The "backtrack" procedure starts at cell $(16, 12)$, follows the direction of the arrows to first cell at $(1, 1)$. The path through the matrix is indicated using cells with gray background. Only the corresponding points to the cells with decreasing length are saved for the LCSS, indicated by green numbers.

#### 6.3.2.2 Connecting Points Collection

A subsequence is a sequence of points that appear in the same relative order and do not have to occupy consecutive positions in the original trace. The longest common subsequence may correspond to more than one road segment. However, we are more interested in the common sub-tracks to both GPS traces, whose points do occupy consecutive positions in the original traces. Each common sub-track corresponds to one road segment that the road users traverse in both traces. Therefore the starting and ending points of the sub-track corresponding to the two ends of the road segment, i.e. intersections. The common sub-tracks between two GPS traces can be obtained by partitioning the longest common subsequences.

We first check the consecutiveness of the points of the longest common subsequences. Given the $k^{th}$ points of the LCSSs for both traces, $\mathbf{g}_1(k)$ and $\mathbf{g}_2(k)$, and their time indexes in the original traces $w_1(k)$ and $w_2(k)$, they are nonconsecutive points if the next points $\mathbf{g}_1(k+1)$ and $\mathbf{g}_2(k+1)$, are recorded much later than them in the original traces, i.e. $w_1(k+1) - w_1(k) > \xi$ and $w_2(k+1) - w_2(k) > \xi$, where $\xi$ is a predefined threshold. The longest common




**(a)** GPS traces

$t_2$

| $t_1$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | **1** | ←1 | ←1 | ←1 | ←1 | ←1 | ←1 | ←1 | ←1 | ←1 | ←1 | ←1 |
| 2 | 0 | ↑1 | **2** | ←2 | ←2 | ←2 | ←2 | ←2 | ←2 | ←2 | ←2 | ←2 | ←2 |
| 3 | 0 | ↑1 | ↑2 | **3** | ←3 | ←3 | ←3 | ←3 | ←3 | ←3 | ←3 | ←3 | ←3 |
| 4 | 0 | ↑1 | ↑2 | ↑3 | **4** | ←4 | ←4 | ←4 | ←4 | ←4 | ←4 | ←4 | ←4 |
| 5 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | **5** | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 |
| 6 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | ↑5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 |
| 7 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | ↑5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 |
| 8 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | ↑5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 |
| 9 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | ↑5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 |
| 10 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | ↑5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 |
| 11 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | ↑5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 |
| 12 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | ↑5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 | ←5 |
| 13 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | ↑5 | ↑5 | ↑5 | **6** | ←6 | ←6 | ←6 | ←6 |
| 14 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | ↑5 | ↑5 | ↑5 | ↑6 | **7** | ←7 | ←7 | ←7 |
| 15 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | ↑5 | ↑5 | ↑5 | ↑6 | ↑7 | **8** | ←8 | ←8 |
| 16 | 0 | ↑1 | ↑2 | ↑3 | ↑4 | ↑5 | ↑5 | ↑5 | ↑6 | ↑7 | ↑8 | **9** | ←9 |

**(b)** Warp path

**Figure 6.6:** *LCSS problem of two GPS traces.* (a) The LCSSs between Trace $\mathbf{r}_1(t_1)$ and $\mathbf{r}_2(t_2)$ using black dots. 9 black dots connected using blue lines are for Trace $\mathbf{r}_1(t_1)$, and 9 black dots with red lines for Trace $\mathbf{r}_2(t_2)$. (b) The "backtrack" procedure of finding the LCSSs between these two traces by following the arrows. The LCSSs are indicated using cells with green numbers.

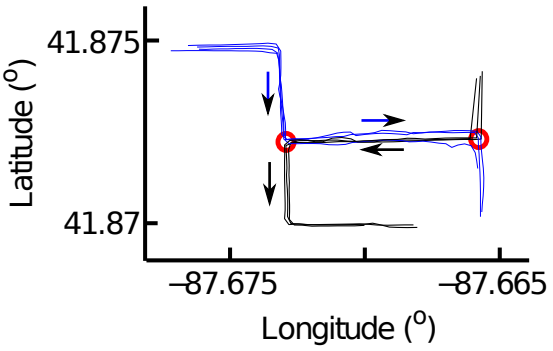



**Figure 6.7:** *GPS traces in opposite directions.* Red and blue traces are in the same direction on the road segment between the intersections indicated by red circles, and black traces in the opposite direction.

subsequence is segmented by these nonconsecutive points into common sub-tracks. If all points of the LCSSs are consecutive, we directly take the starting and ending points of the LCSSs as connecting points if they are not the starting and ending points of these two GPS traces.

As shown in Fig. 6.6, the fifth points of the LCSSs are not consecutive in the original traces. For trace $\mathbf{r}_1(t_1)$, the sixth point of its LCSS occupies the thirteenth position in $\mathbf{r}_1(t_1)$. The fifth and six points of the LCSS are separated by seven blue dots. For trace $\mathbf{r}_2(t_2)$, the sixth point of its LCSS occupies the eighth position in $\mathbf{r}_2(t_2)$. The fifth and six points of the LCSS are separated by two red dots. Therefore, each LCSS is partitioned to two sub-tracks by the fifth point: the first sub-track containing 5 points, and the second one containing 4 points. For the first trace, the detected common sub-tracks are $\mathbf{r}_1(t_1)$, $t_1 = 1, 2, 3, 4, 5$ and $\mathbf{r}_1(t_1)$, $t_1 = 13, 14, 15, 16$. The common sub-tracks for the second trace are $\mathbf{r}_2(t_2)$, $t_2 = 1, 2, 3, 4, 5$ and $\mathbf{r}_2(t_2)$, $t_2 = 8, 9, 10, 11$. The ending points of the first common sub-track for each trace, $\mathbf{r}_1(5)$ and $\mathbf{r}_2(5)$, are connecting points, and the starting points of the second sub-track for each trace, $\mathbf{r}_1(13)$ and $\mathbf{r}_2(8)$, are also connecting points. The starting points of the first common sub-track for each trace, $\mathbf{r}_1(1)$ and $\mathbf{r}_2(1)$, are not connecting points because they are also the starting points of the GPS traces. The ending points of the second sub-track for each trace, $\mathbf{r}_1(16)$ and $\mathbf{r}_2(11)$, are not connecting points because $\mathbf{r}_1(16)$ is the ending point of trace $\mathbf{r}_1(t_1)$.

The procedure of detecting the LCSSs processes two GPS traces directionally, which start from the beginning of both traces and end at the ending of both traces. If the road users traverse the same road segment in two traces, but in the opposite directions, no common sub-tracks will be found between them. As shown in Fig. 6.7, all of the GPS traces share the same road segment between two intersections, which are indicated by red circles. On this road segment, the road users travel from left to right in the blue traces, but from right to lefts in the black traces. No common subsequences will found between

the blue and black traces, although they do share the same road segment. This can be solved by reversing the data points of the black traces.

Given a data set of $N$ GPS traces, we first find the common sub-tracks between each pair of GPS traces as elaborated above, and collect the starting and ending points of the common sub-tracks as connecting points, if they are not the starting and ending points of these two GPS traces. We then reverse one trace in the each pair of the GPS traces and repeat the procedure again to find more connecting points.

#### 6.3.2.3 Intersection Extraction from Connecting Points

The same connecting point may be found in one GPS trace when it is aligned with different other GPS traces using LCSS algorithm. Therefore there are a lot more connecting points detected than turning points. At the areas with tall buildings and high-error GPS traces, the connecting points for two close intersections may mix together on the road between the two intersections. It is difficult to apply Algorithm 6 to distinguish these two intersections.

Instead of Algorithm 6, we detect the intersections by estimating the density of the connecting points in this section. The map is first discretized into a fine grid of cells. The number of connecting points in each cell is then counted, producing a 2-Dimensional (2-D) histogram [Biagioni 12a]. Because of GPS errors, connecting points may be detected on the road segment, rather than at the intersections. Therefore, the 2-D histogram is convolved with a Gaussian smoothing function $N(0, \sigma^2)$ to produce a density map. If the cell size is small enough, the density map produced is a close approximation of the Kernel Density Estimation (KDE). The choice of $\sigma$ of the Gaussian function should be made depending on the expected GPS error and the intersection size.

The local maxima in the density map, whose values are greater than those of their neighbors in a $3 \times 3$ neighborhood, are detected as intersection candidates. The outputs of this step are the geographical locations of the intersections, and the connecting points which belong to each intersection.

The connecting points are detected from pairwise GPS trace alignment using the LCSS algorithm. One single trace meeting with a lot of other similar GPS traces at the same location can produce a large number of connecting points, leading to a local maximum on the density map. However, this GPS trace can be an abnormal trace which deviates from the road because of GPS errors. An example is shown in Fig. 6.8. This will lead to an intersection falsely detected. Therefore, we need to check the patterns of the GPS traces meeting at one intersection, and remove this kind of mis-detected intersections, so as to make our algorithm more robust to GPS errors.

Given an intersection candidate $\mathbf{q}$, the connecting points belonging to this intersection are expressed as $\mathbf{p}_m^{(1)}$ and $\mathbf{p}_m^{(2)}$, $m = 1, \ldots M$. Each pair of connecting points are produced through finding the common sub-tracks of two traces $\mathbf{r}_{i(m)}$ and $\mathbf{r}_{j(m)}$, $m = 1, \ldots M$: $\mathbf{p}_m^{(1)}$ is from $\mathbf{r}_{i(m)}$ and $\mathbf{p}_m^{(2)}$ is from $\mathbf{r}_{j(m)}$, where $i(m) \in [1, N]$ and $j(m) \in [1, N]$. Let $K_n$ be the number of connecting points which are detected by aligning trace $\mathbf{r}_n$ with other traces using LCSS,

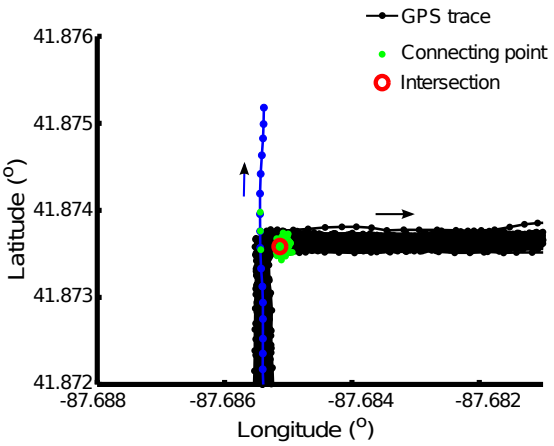



**Figure 6.8:** *A false positive intersection.* One intersection candidate in red circle is detected from the connecting points in green dots. It is false positive because all of the connecting points are produced from 152 traces interconnecting with one single trace. The black arrow indicates the moving directions of road users in the 152 traces shown in black lines with dots, and the blue arrow is for the single trace shown in blue lines with dots.

$K_n = \sum_{m=1}^{M} 1\{i(m) = n\} + \sum_{m=1}^{M} 1\{j(m) = n\}$. If all connecting points are detected from aligning trace $\mathbf{r}_n$ with every of other traces, half of the connecting points will be from trace $\mathbf{r}_n$, i.e. $K_n = M$. In this case, an intersection will be falsely detected from these connecting points if trace $\mathbf{r}_n$ is an abnormal trace. For each GPS trace, we calculate $p_n$, $n = 1, \ldots N$, and remove this intersection candidate if any of $p_n$ equals $M$.

As shown in Fig. 6.8, the intersection candidate shown in red circle is removed from the true intersections because all of its connecting points shown in green dots are detected from aligning the GPS trace, indicated using blue lines with dots, with every of other 152 GPS traces shown in black lines with dots. We admit that we will also remove true intersections connecting three road segments, in which the road users traverse two of the road segments very frequently, but through the third road segment only once. This detected intersection candidate will be removed until we have more GPS traces to confirm the existence of this third road segment.

The result is a set of intersections $Q^{(2)} = \{\mathbf{q}_i^{(2)}, i = 1, \ldots I^{(2)}\}$. The superscript (2) refers to the algorithm of intersection detection from connecting points, different from the intersections $Q^{(1)} = \{\mathbf{q}_i^{(1)}, i = 1, \ldots I^{(1)}\}$ which are detected using the turning points.

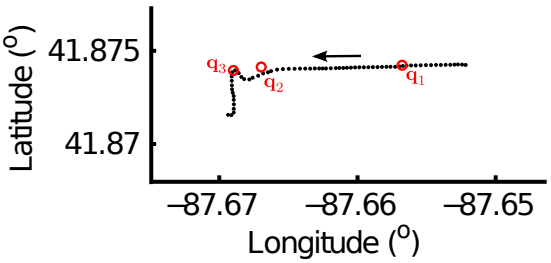



**Figure 6.9:** *Intersection connectivity.* Because of GPS error, $\mathbf{q}_2$ is missed in a trace indicated by black dots, leading to $\mathbf{q}_1$ and $\mathbf{q}_3$ falsely connected.

### 6.3.3 Connectivity Analysis and GPS Trace Segmentation

The connectivity of intersections is analyzed by determining if there are GPS traces that travel directly from one intersection to another. We first determine if a given GPS trace contains any intersections by computing the distance between each point in the trace with the location of our detected intersections. If the distance between any point in the trace and an intersection is within a predetermined threshold, we conclude that the GPS trace travels through this intersection.

Ideally, two intersections are directly connected if there is a trace that travels directly through both intersections without encountering a third. However, GPS errors can contribute to false positive intersection connections if an intersection is not detected along a trace. For example, a road user travels through three intersections sequentially in one trace, as shown in Fig. 6.9. However, only the first and the last intersection are detected using our distance thresholding, the second one $\mathbf{q}_2$ is missed because its distance to any point of the trace exceeds the threshold. This would lead to that $\mathbf{q}_1$ and $\mathbf{q}_3$ are falsely detected directly-connected. We count the number of traces in which the road users travel through two intersections continuously and use them to decide whether they are directly connected. If two intersections are consecutively traveled through only once or twice, they are considered as unconnected.

After the connectivity is determined, we partition all GPS traces into track segments between each pair of directly connected intersections. Because of GPS errors, some of the tracks may drift away from the road segment. We apply the clustering algorithm using spatial similarity in Section 2.4 to group the tracks. The cluster with the most tracks is identified as the the cluster of true tracks, and the tracks in other clusters as outliers. Only the "true" tracks are used to generate the geometric representation of the road segment.

The output of this step is the connectivity matrix $D$ and the track segments associated with each road segment. We will use the following notations: For one road segment $R$, there are $N$ normal tracks $\mathbf{r}_n(t_n)$, $t_n = 1 \dots T_n$, $n = 1, \dots N\}$. Specifically, $\mathbf{r}_n(t_n) = (x_n(t_n), y_n(t_n))$ is composed of the latitude and longitude

of the point $\mathbf{r}_n(t_n)$.

## 6.4 Aligning Tracks for a Road Segment

We proposed two methods based on DTW to align *many* trajectories in Chapter 3. In this chapter, we apply these methods to associate the data points of the tracks for each road segment, resulting in the warped tracks with the same length, and the time indexes of the points of the warped tracks in the original tracks. Given $N$ tracks for one road segment $\mathbf{r}_n(t_n)$, $t_n = 1, \dots T_n$, $n = 1, \dots N$, the outputs of the track alignment using the stretch-and-then-compress method are expressed as: the warped tracks $\mathbf{g}_n^{(1)}(k)$, $k = 1, \dots K^{(1)}$, $n = 1, \dots N$ and the time indexes of their points in the original tracks $w_n^{(1)}(k)$, $k = 1 \dots K^{(1)}$, $n = 1, \dots N$ . The outputs for the greedy method based on successor classification are the warped tracks $\mathbf{g}_n^{(2)}(k)$, $k = 1, \dots K^{(2)}$, $n = 1, \dots N$ and the time indexes of their points in the original tracks $w_n^{(2)}(k)$, $k = 1 \dots K^{(2)}$, $n = 1, \dots N$.

Using the point associations established during the track alignment, we calculate the average of the warped tracks as the geometric representation of road segment, as shown in Equation 3.4. Given the velocity magnitudes of the original tracks $s_n(t_n)$, $t_n = 1, \dots T_n$, $n = 1, \dots N$, and the time indexes of the points of warped tracks in the original tracks $w_n(k)$, $k = 1, \dots K$, $n = 1, \dots N$, the average velocity and velocity variance along the average track are calculated as below:

$$\mu_s(k) = \frac{1}{N} \sum_{n=1}^{N} s_n(w_n(k)), \tag{6.4}$$

$$\sigma_s(k) = \frac{1}{N} \sum_{n=1}^{N} (s_n(w_n(k)) - \mu_s(k))^2, \tag{6.5}$$

where $k = 1 \dots K$.

For the stretch-and-then-compress method, the average track extracted is expressed as $\mathbf{g}_0^{(1)}(k)$, $k = 1 \dots K^{(1)}$, and the average velocity and velocity variance along it are expressed as $\mu_s^{(1)}(k)$, $\sigma_s^{(1)}(k)$, $k = 1, \dots K^{(1)}$. For the greedy method based on successor classification, $\mu_s^{(2)}(k)$, $\sigma_s^{(2)}(k)$, $k = 1, \dots K^{(2)}$ are average velocity and velocity variance along the average track $\mathbf{g}_0^{(2)}(k)$, $k = 1, \dots K^{(2)}$.

Chapter 3 shows good performance of our proposed alignment methods on aligning the tracks with high sampling rate for one road segment. In this section, we will process the whole data sets which contain hundreds or even thousands of GPS traces over a large area, instead of just one single road. We will also discuss the influence of varying sampling rates on the road generation for each of the proposed method.

## 6.5 Performance Evaluation

In order to evaluate the performance of our proposed methods, we calculate the accuracy of the generated road network, which consists of intersections and road segments. The generated road network has to be accurate both topologically and geometrically, with respect to a ground truth map. The topological accuracy of the road network describes how many intersections are extracted correctly. The geometric accuracy refers to how accurate are the geographic locations contained in the road network compared to the ground truth map. Both the geographic locations of the intersections and the sequences of data points representing the roads segments need to be evaluated.

### 6.5.1 Topological accuracy calculation

Our intersections are extracted from GPS traces. Due to the limited accuracy and outliers in the GPS traces, we may detect false intersections which do not exist on the ground truth map and we may also fail to detect some true intersections (i.e., which exist in the ground truth map). The accuracy of the intersections depends on three primary aspects: the number of detected true intersections , the number of detected false intersections, and the number of the missing true intersection we fail to detect.

We use two parameters to quantify the reliability of intersection detection: $i_s$ the proportion of falsely detected intersections, and $i_m$ the proportion of missing true intersections [Biagioni 12a]. They are defined as:

$$\begin{aligned} i_s &= \frac{I_s}{I_s+I_c} \\ i_m &= \frac{I_m}{I_m+I_c}, \end{aligned} \tag{6.6}$$

where $I_s$ is the number of falsely detected intersections, $I_c$ the number of correctly detected intersections and $I_m$ the number missing true intersections.

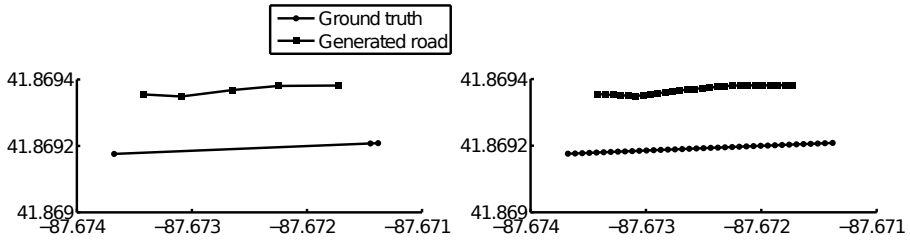
The well-known $F$-score, based on a combination of precision and recall, is used as an overall quality score:

$$F_i = 2\frac{(1-i_s)(1-i_m)}{(1-i_s)+(1-i_m)} \tag{6.7}$$

Higher values of $F_i$-score indicate that the intersection detection is more reliable [Liu 07].

### 6.5.2 Geographical accuracy evaluation

Measuring the accuracy of the generated roads compared to the ground truth map is related to the topic of map matching. As a first step, this requires matching road segments from the experimental results with those in the ground truth map. A variety of algorithms can be applied to solve this problem [Greenfeld 02], for example, Kalman filtering, fuzzy logic, and many others. In our case, the road segment can be identified easily by finding the corresponding




**(a)** Generated road segment and its ground truth

**(b)** Interpolated generated road segment and its ground truth

**Figure 6.10:** *Inferred road segment and its ground truth.* (a) shows the geometric representation of one road segment and its ground truth . In (b), both of them are interpolated so as to reduce the error of their distance caused by uneven distribution of the data points.

intersections through checking the distance between the data points on the ground truth map and the inferred intersections, since at most one directed road segment exists between each pair of intersections.

To judge the difference between an inferred road segment and the corresponding road segment in the ground truth map, we could naively compute a measure based on the distance between points in both road segments. However, this poses some complications, as the road segments extracted by our method are represented as a sequence of data points (the average of points of the warped tracks). The available ground truth road segments are also represented as a time series of data points. However, both the total number of points per road segment and its distribution over time can vary considerably between the inferred and ground truth data, and from one road segment to another. Therefore, quality measures based on distances between points in ground truth and computed road segments are affected by this variability.

A partial solution would be to apply DTW to align the two point series and warp them to new series of the same length, and then compute the quality measure on those warped track segments. However, this does not fully solve the problem, as the warped series can only contain repeated points, but can never "fill in" missing data. Therefore, large time gaps in the ground truth or inferred data could lead to large spatial distances between ground truth and inferred data, even if the track segments are otherwise very similar. For instance, the first and second points of the generated road in Fig. 6.10a are matched to the first point on the ground truth using DTW because no closer points are found on the ground truth. The distance between these paired points are actually larger than the real distance between the generated road segment and its ground truth [Calcagno 08].

We address this problem by linearly interpolating the data points of both the inferred and ground truth road segments. The interpolation limits the spatial distance between successive points on the same road segment to be a approximately fixed distance $d_1$. Through interpolation, the data points are

more dense and distributed more evenly, as shown in Fig. 6.10, reducing the error of the distance measure.

The number of the points interpolated between every two successive points on the generated road segment depends on their distance. Given the generated geometric representation for one road segment, $(\varphi(k), \lambda(k))$, $k = 1, \dots K$, there are $N(k)$ points $(\varphi'(n,k), \lambda'(n,k))$, $n = 1, \dots N(k)$, where $N(k) = \lfloor \frac{d_2}{d_1} \rfloor - 1$ interpolated between two successive points $(\varphi(k), \lambda(k))$ and $(\varphi(k+1), \lambda(k+1))$ depends on the their distance $d_2$.

$$\varphi'(n,k) = \varphi(k) + n\frac{\varphi(k+1) - \varphi(k)}{N(k)+1} \tag{6.8}$$

$$\lambda'(n,k) = \varphi(k) + n\frac{\lambda(k+1) - \lambda(k)}{N(k)+1} \tag{6.9}$$

where $n = 1, \dots N(k)$. $(\varphi'(0,k), \lambda'(0,k))$ and $(\varphi'(N(k)+1,k), \lambda'(N(k)+1,k))$ are the successive points $(\varphi(k), \lambda(k))$ and $(\varphi(k+1), \lambda(k+1))$, respectively.

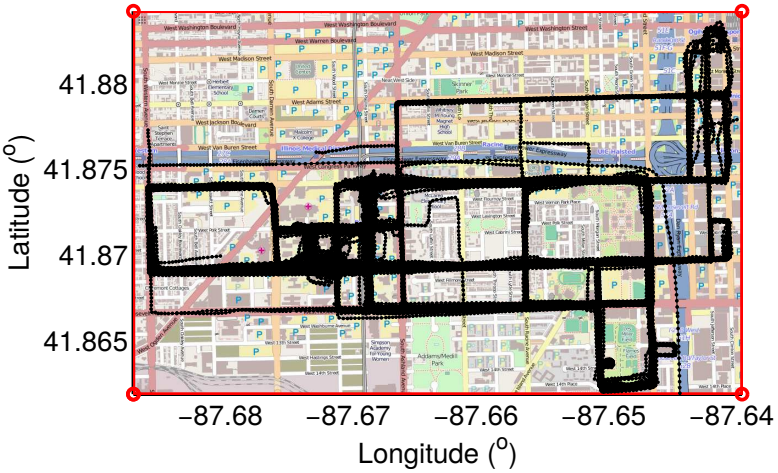
By interpolating data points between every two successive points in the generated road representation, a denser road representation is obtained: $(\varphi'(n,k), \lambda'(n,k))$, $n = 0, \dots N(k+1)$, $k = 1, \dots K-1$. The ground truth data is interpolated in the same way. We apply DTW to align the interpolated road representation and its ground truth. The average distance between the paired interpolated data points is calculated as shown in Equation 2.4, to measure the geographical accuracy of the generated road.

Given the correctly detected intersections $\mathbf{q}_i = (\varphi_i, \lambda_i)$, $i = 1, \dots I_c$, where $\varphi_i$ and $\lambda_i$ are coordinates in latitude and longitude, and their ground truths $\mathbf{p}_i = (\varphi'_i, \lambda'_i)$, $i = 1, \dots I_c$, the geographical accuracy of the detected intersections are measured using their average distance to the ground truth using the spherical law of cosines formula.

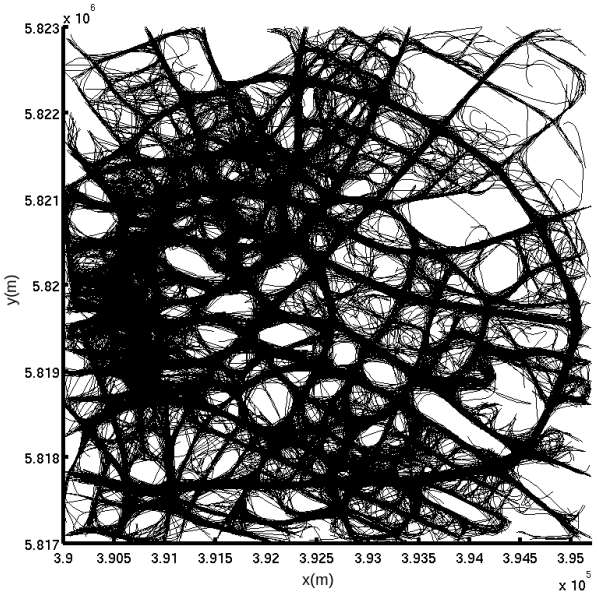
$$d = \frac{1}{I_c}\sum_{i=1}^{I_c} \arccos\left(\sin\varphi_i \sin\varphi'_i + \cos\varphi_i \cos\varphi'_i \cos\left(\lambda_i - \lambda'_i\right)\right)R \tag{6.10}$$

## 6.6 Results

We test our methods on two datasets: Chicago dataset and Berlin dataset. Chicago dataset is gathered by campus shuttle drivers. Each shuttle driver carries an iPhone running the BITS Laboratory's tracking application [Laboratory 11]. In total, this dataset contains 889 GPS traces covering an area of $7 \times 4.5\ km^2$, as shown in Fig. 6.11a. The participating shuttles travel through both the areas with low-rise buildings and other areas with significant GPS error because of the tall buildings. The traces range from 100 to 363 samples, with a sampling rate of 1 seconds to 29 seconds (average: 3.61 seconds and standard deviation: 3.67 seconds). The Chicago dataset has a geographic coordinate system, which uses latitude and longitude to describe locations on the




**(a)** Chicago dataset




**(b)** Berlin dataset

**Figure 6.11:** *Raw traces.* (a) shows 889 GPS traces of Chicago dataset in black lines with dots which indicate the positions of GPS recordings. (b) shows 27189 GPS traces in black lines for Berlin dataset.

Earth's surface. Berlin dataset consists of 27189 GPS traces obtained from a taxi fleet [Portal 14]. It covers an area of $6 \times 6\ km^2$, as shown in Fig. 6.11b. The tracks are composed of from 22 up to 58 position samples, with a sampling rate of 15 seconds to 127 seconds (average: 41.98 seconds and standard deviation: 38.70 seconds). The Berlin dataset has a projected coordinate system, which projects maps of the earth's spherical surface onto a two-dimensional Cartesian coordinate plane and uses $x$ and $y$ to describe locations.

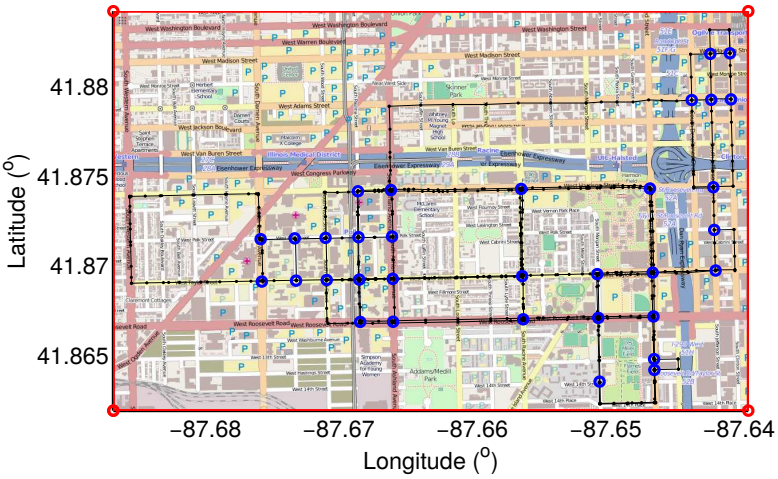
To measure the accuracy of the road network generated from the shuttle traces, we use OpenStreetMap for a ground truth map. Although the positional error of GPS traces on OpenStreetMap is approximately 10-20 meters, it is still popularly used as ground truth for research in road network generation, because of its open source access and large coverage. For Chicago dataset, we download the ground truth data from OpenStreetMap website directly. The ground truth data consists both *nodes* and *ways*. A node represents a specific point on the Earth's surface, and a way is an ordered list of nodes that define a road segment. By comparing the directly-connected pairwise intersections with the starting and ending nodes on the ways, we extract the ground truth road segments. In total there are 33 true intersections, as shown in blue circles in Fig. 6.12a. There are 57 road segments shown in black lines with dots on the ground truth map: 35 of them are one-way, and 22 are two-way. Using the ground truth data, we will evaluate our proposed methods both qualitatively and quantitatively. For Berlin data set, the dataset owner gives a collection of road edges as ground truth. Fig. 6.12b shows the ground truth map with black lines for road edges and black dots for nodes. The ground truth map covers all of the streets in the whole area, including streets not traversed by the taxi fleet. Each edge on the ground truth map is composed of two nodes. A road edge in the ground truth data is different from a road segment defined in our work. A road segment may be comprised of several road edges. It is difficult to directly utilize the ground truth data for quantitative evaluation. Therefore only we only evaluate the performance of our algorithms on Berlin data set qualitatively.

### 6.6.1 Results of Chicago Data Set

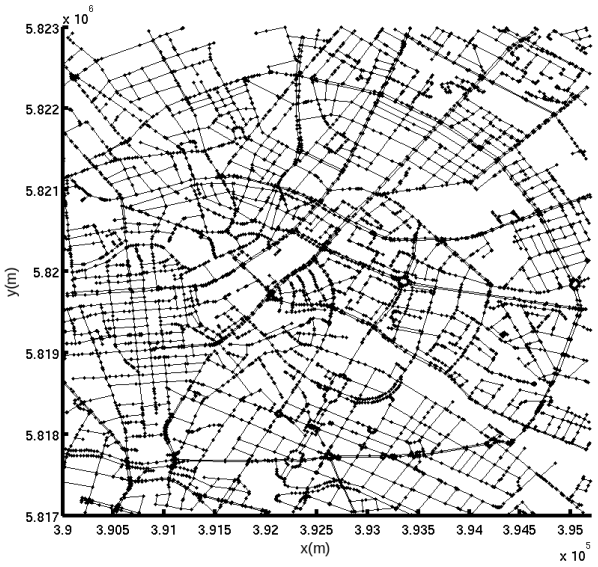
In this section, we present the results of Chicago data set. We first show the intersections detected using both of the proposed methods in Section 6.6.1.1. The tracks segmented from the GPS traces by the intersections and their clustering results are shown in Section 6.6.1.2 and 6.6.1.3. We give the results of the track alignment in Section 6.6.1.4, and lastly the track averaging results in Section 6.6.1.5.

#### 6.6.1.1 Results of Intersection Detection

In this section, we show the intersections detected using both of the proposed methods. We first show the intersections from the turning point detection




**(a)** Chicago




**(b)** Berlin

**Figure 6.12:** *Ground truth map.* (a) shows 33 intersections in blue circles and road segments in black lines with dots for Chicago dataset. There are 35 one-way road segments and 22 two-way road segments. (b) shows the road edges in black lines as ground truth for Berlin dataset.

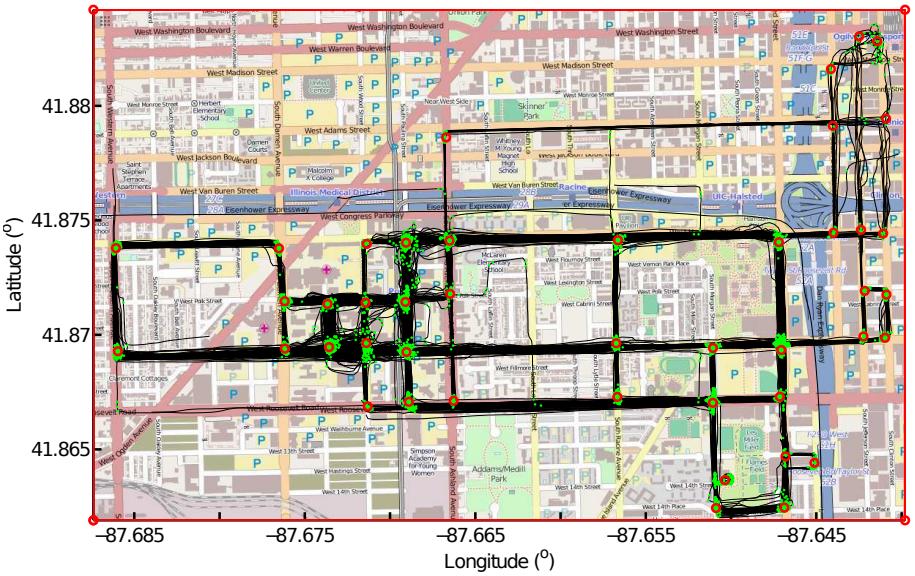



**Figure 6.13:** *Intersections before bend removal.* 44 intersection candidates in red markers are detected from the turning points depicted in green dots, including both intersections and bends.

method, and then the results for the connecting point methods. At last, we give the results of accuracy evaluation of the detected intersections.

**Results based on Turning Point Detection** Fig. 6.13 shows the turning points extracted from the 889 GPS traces as green dots. The locations where many turning points cluster are recognized as intersection candidates shown in red markers. Sometimes isolated turning points are detected because of GPS errors or insufficient coverage by shuttle tracks. These isolated turning points are not considered as intersections by our algorithm. In total, 44 intersection candidates are detected using the turning points.

Fig. 6.14 shows the intersections detected after removing the bends by checking the entering and exiting directions of the connecting points at the intersection candidates. 8 bends are removed from the intersection candidates, leaving 36 intersections, shown in red markers. For instance, the two bends at the left edge of the map are removed from the intersection candidates. But not all of the bends are removed because of GPS noise, for instance, $\mathbf{q}_{25}^{(1)}$, $\mathbf{q}_{27}^{(1)}$ and $\mathbf{q}_{34}^{(1)}$. 29 of 36 intersections are true intersections, shown as red circles, and 7 of them in red crosses are spurious intersections which do not exist on the ground truth map. Besides, there are 5 missed intersections which are undetected, depicted in blue circles.

An intersection apparently exists between $\mathbf{q}_{14}^{(1)}$ and $\mathbf{q}_{15}^{(1)}$ in Fig. 6.14 . However, no intersection is detected there using this method because only a few

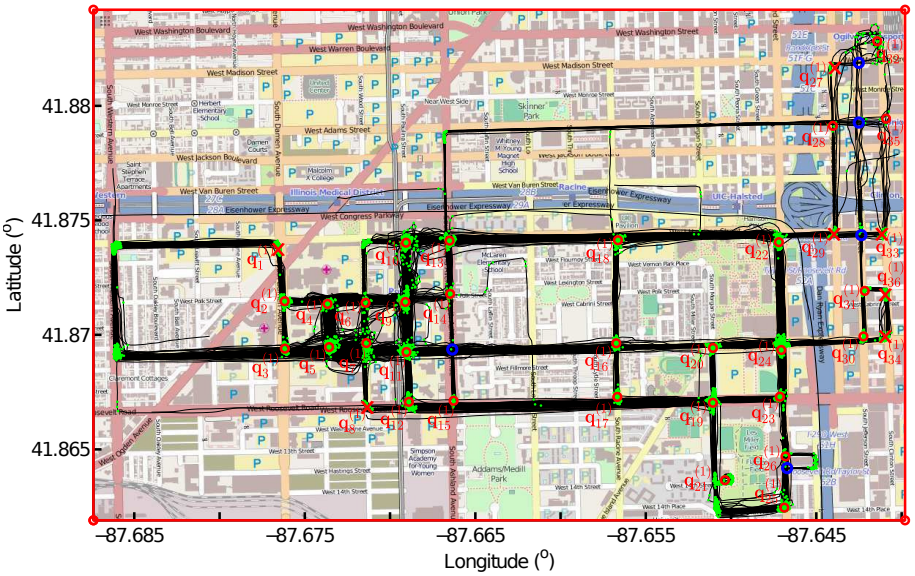



**Figure 6.14:** *Intersections after bend removal.* 36 of 44 intersection candidates are identified as intersections after removing 8 bends, including 29 true intersections in red circles and 7 spurious intersections in red crosses. 5 intersections on the ground truth map are undetected, which are indicated by blue circles.

isolated turning points exist. At this location, the shuttles always go straight without changing their moving directions, possibly because of the presence of a bridge.

Moreover, there are two intersections on the ground truth map around the location of the extracted intersection $\mathbf{q}_{26}^{(1)}$. We found that the turning points are clustered to the same group $\mathbf{q}_{26}^{(1)}$, because they are too close to each other.

On the ground truth map, each intersection is localized at the center of the junction between road segments, but the intersections detected using our method sometimes deviate from the center, depending on the type of turns the shuttles make at a particular junction. For instance, the intersection $\mathbf{q}_{22}^{(1)}$ connects 4 road segments, but the shuttle only visited 3 of them. The shuttle only made one type of turn there: coming from $\mathbf{q}_{18}^{(1)}$, turning right at $\mathbf{q}_{22}^{(1)}$ to $\mathbf{q}_{24}^{(1)}$. More turning points are detected along this type of turn, resulting in the location of $\mathbf{q}_{22}^{(1)}$ deviating from the center of the crossroad. The average distance between the correctly detected intersections in Fig. 6.14 and their ground truth in Fig. 6.12 is 22.69 meters.

**Results based on Connecting Point Detection** Fig. 6.15 shows the connecting points in green dots. Most of the connecting points gather at the intersections. At the top-right corner of the map, only a few connecting points

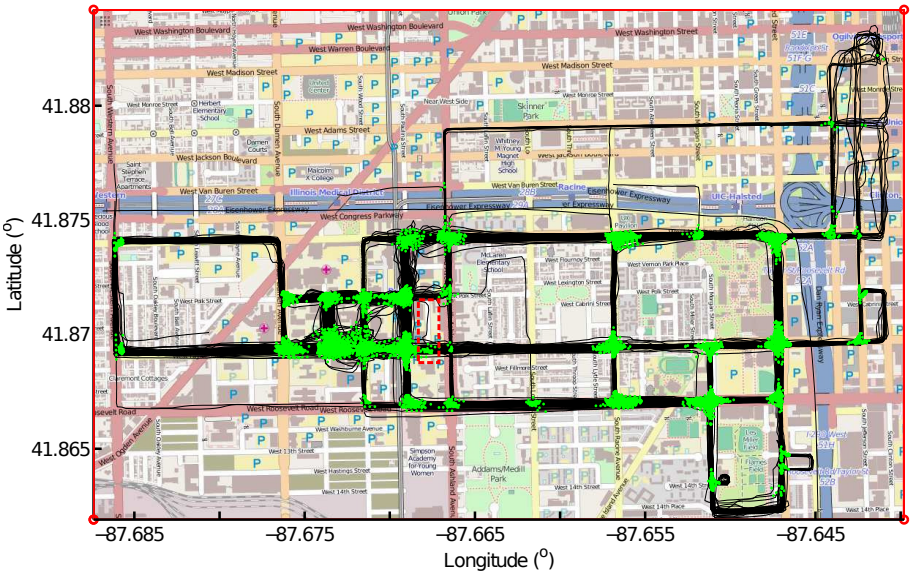



**Figure 6.15:** *Connecting points.* The connecting points are shown in green dots.

are detected because of high-error and low-coverage GPS traces. Some of connecting points are detected on the road segments instead of at the intersections because of GPS errors. An example is shown in the red rectangle. One GPS trace deviates from the main road. Common sub-tracks between this GPS trace and other traces end in the middle of the road segment, rather than at the end of the road segment, resulting in false positive connecting points.

Fig. 6.16 shows the kernel density estimation of the connecting points. The map shown is discretized into $1 \times 1$ square meters cells. 2-D histogram is produced from the connecting points, and convolved with a normal distribution function $N(0, \sigma^2)$. We chose $\sigma = 15$ meters depending on the size of the intersections and the minimum distance between two adjacent intersections. The density estimate shows clear peaks at the intersections on the map.

Fig. 6.17 shows 41 intersection candidates in red circles, which are local maximas on the density map. At some of the intersection candidates, the road user turns to a different direction in only one of the traces, for instance, the location at $(41.8768°, -87.667°)$. The method based on moving direction change only detects a few turning points from one of the GPS passing by this location; this is not enough to detect an intersection there. The method based on connecting point detection detects a common sub-track between this trace with direction change and each of the other traces, producing a lot of connecting points for intersection detection.

Fig. 6.18 shows the true intersections after checking the patterns of trace meeting at the intersection candidates. If all of the connecting points around one intersection candidate are obtained by one GPS trace intersecting with a

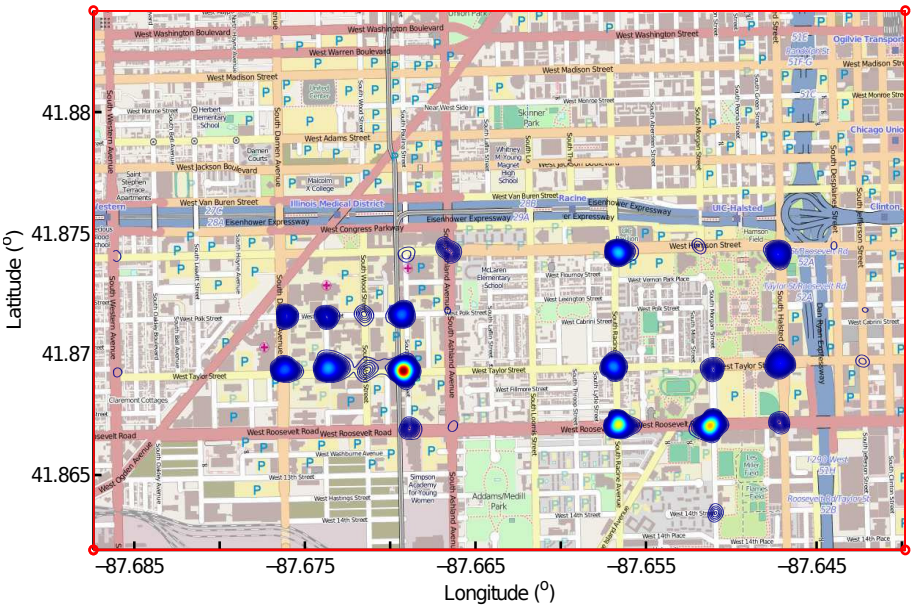



**Figure 6.16:** *KDE of the connecting points.* Peaks of the KDE locate at the intersections.

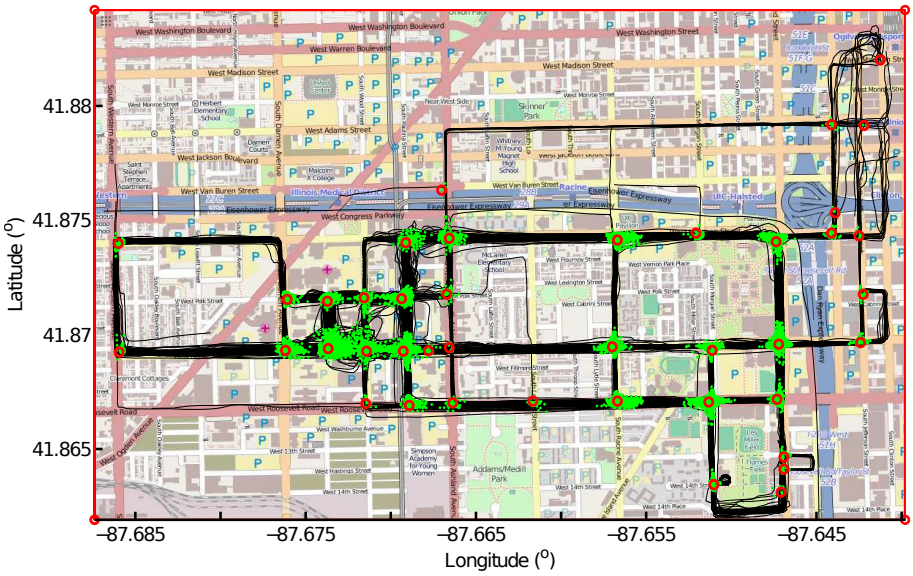



**Figure 6.17:** *Intersection candidates.* The intersection candidates are extracted from the connecting points by finding local maximums on the density map. In total, 41 candidates are detected.

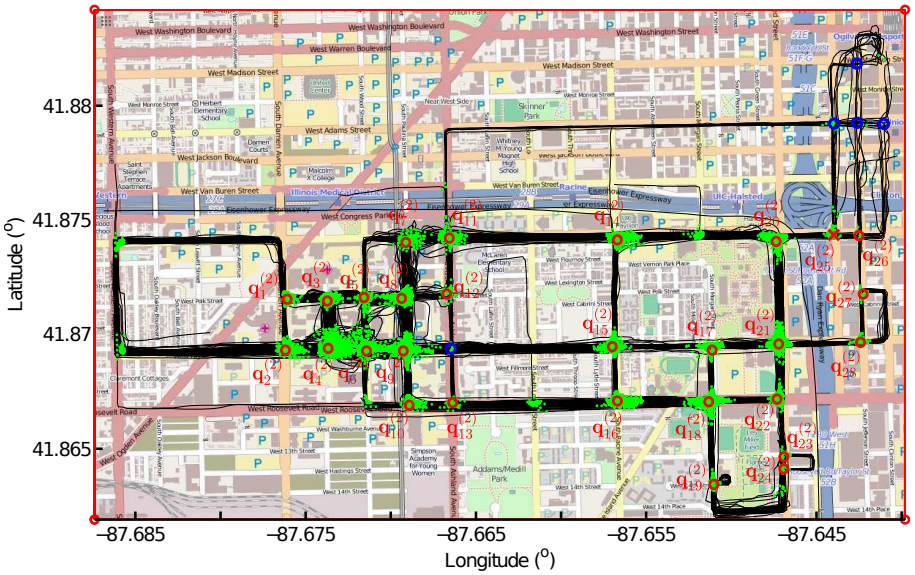



**Figure 6.18:** *Intersections.* By checking the patterns of the traces meeting at the intersections 13 intersection candidates are removed, resulting in 28 intersections in red circles.

number of other traces, it is removed because the existence of a road segment should be confirmed by more traces. In total, 28 intersections are kept after removing 13 candidates, including 27 true intersections in red circles and 1 spurious intersection $\mathbf{q}_{25}^{(2)}$, indicated by a red cross, which is not matched to any intersection on the ground truth map. The blue circles are for intersections we fail to detect. The average distance between the detected intersections in Fig. 6.18 and their ground truth in Fig. 6.12 is 14.90 meters, much smaller than that between the detected intersections using moving direction change and the ground truth, which is 22.69 meters.

As shown in Fig. 6.14, some bends are mistakenly detected as intersections by the first method even after checking the entering and exiting directions. These bends only connect two road segments. The second method based on connecting points is designed to detect the locations which connect three road segments. Therefore, no bends are mis-detected as intersections in Fig. 6.18.

**Discussion** Fig. 6.19 shows the topological accuracy of the intersections. The matching threshold is defined as the allowable distance between the intersections and their positions on the ground truth map. An intersection will be considered as falsely detected if there is no ground truth intersection within the allowable distance. We can see that with the lowest 5-meters matching threshold shown in Fig. 6.19, all of the intersections we detected are classified as false intersections, and all of the intersections on the ground truth map as

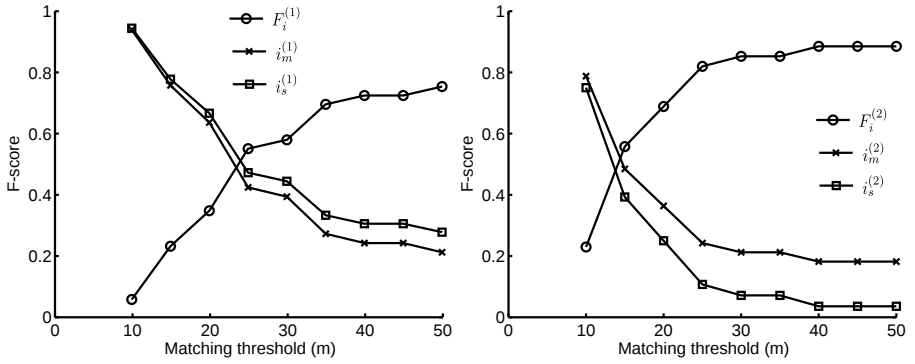



**Figure 6.19:** *Intersections accuracy analysis.* (a) shows accuracy of the intersections detected from turning points. (b) is for the other method based on connecting point detection. The bigger the F-score is, the more accurate it is.

missing intersections. As the matching threshold gets bigger, more detected intersections are correctly detected, which means that there are intersections on the ground truth map spatial close to them. Therefore, the fraction of missing intersections $i_m^{(1)}$ and $i_m^{(2)}$ gets lower and the number of spurious intersections $i_s^{(1)}$ and $i_s^{(2)}$ drop as well, resulting in higher F-score $F_i^{(1)}$ and $F_i^{(2)}$.

Because the shuttle drivers visit the top-right area on the map too little, as shown in Fig. 6.11a, there are not enough traces to detect the intersections there. Therefore, the F-score will not achieve 1 using either of our methods. We give an threshold of 0.8 to the F-score. If it is higher than 0.8, the intersection detection is successful. With a matching threshold of 30 meters, F-score of the connecting point method achieves 0.85, and F-score of the turning point method is only 0.6. F-score of the turning-point-based method reaches around 0.8 with a matching threshold of 40 meters. It demonstrates that intersections detected from the connecting points are more accurate.

As shown in Fig. 6.14 and 6.18, both methods almost detect the same number of intersections correctly: 29 for the turning point method and 28 for the connecting point method. Both methods fail to detect 5 intersections (the blue circles), so that $i_m^{(1)}$ in Fig. 6.19a is similar to $i_m^{(2)}$ in Fig. 6.19b. Some bends are detected as intersections using the turning point method, resulting in higher $i_s^{(1)}$ but lower $F_i^{(1)}$ in Fig. 6.19a compared to $i_s^{(2)}$ and $F_i^{(2)}$ in Fig. 6.19b.

For the turning point method, we need to calculate the moving directions for each GPS point and find the points with moving direction change in each of $N$ GPS traces. For the connecting point method, we need to find the common sub-tracks between each of the $\frac{(N-1)N}{2}$ pairs of GPS traces. Besides, it is more complex to calculate the length and direction matrix for each pair of GPS traces than the moving directions at each GPS point. We implemented and tested our algorithm in MATLAB 8.3 on a 2.0 Ghz machine with 4 GB RAM. The

computation time is around 36 minutes using the connecting point method, and 5 minutes using the turning point method. This means that the turning point method is more scalable onto bigger data set than the connecting point method.

We detect the intersections in two different methods at this step, and the results of one method will be used to partition the GPS traces for individual road segments at the next step. Since more intersections are detected at the top-right corner of the map using the turning point method, we will choose them to segment the GPS traces. Although some road segments are split into smaller pieces because of falsely detected intersections, more directed road segments will not affect the track alignment on each road segment.

#### 6.6.1.2 Results of GPS Trace Segmentation

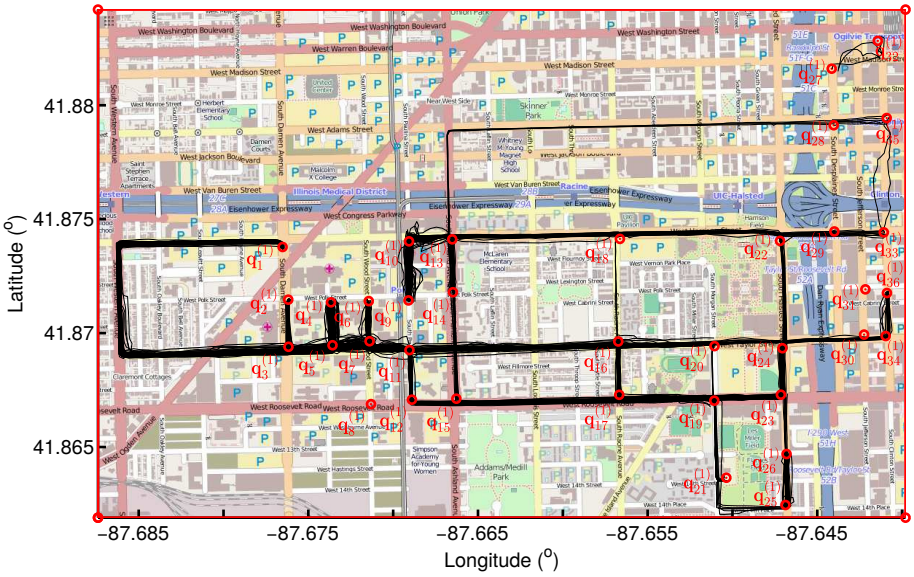
With 50 meters as the matching thresholds for the intersections, there are 77 non-zero elements in the connectivity matrix, indicating 77 pairs of intersections directly connected to each other. According to the connectivity matrix, the GPS traces shown in Fig. 6.11a are segmented, resulting in Fig. 6.20. Fig. 6.20a depicts the tracks in black lines, for 38 road segments which start from lower-index intersections and end at higher-index intersections. While the tracks shown in Fig. 6.20b are for the 39 road segments from higher-index intersections to lower-index intersections.

Our method also detects that some of the roads are one-directional, for instance, the road segment between the first and third intersection in Fig. 6.20. The shuttles always travel from $\mathbf{q}_3^{(1)}$ to $\mathbf{q}_1^{(1)}$, and never take the opposite direction. In another case, the shuttles travel between the intersection $\mathbf{q}_{22}^{(1)}$ and $\mathbf{q}_{29}^{(1)}$ in both directions according to their GPS traces. However, there are no tracks for the road segment existing between them on the ground truth map as shown in Fig. 6.12. This could be caused by temporary road construction. The Chicago data set was recorded in April, 2011, but OpenStreetMap was downloaded on August 2014. By checking this road segment under Google street view on dates closest to the access dates, we find that the road segment between intersection $\mathbf{q}_{22}^{(1)}$ and $\mathbf{q}_{29}^{(1)}$ was passable in May, 2011, but was blocked in October, 2014, because of road construction.

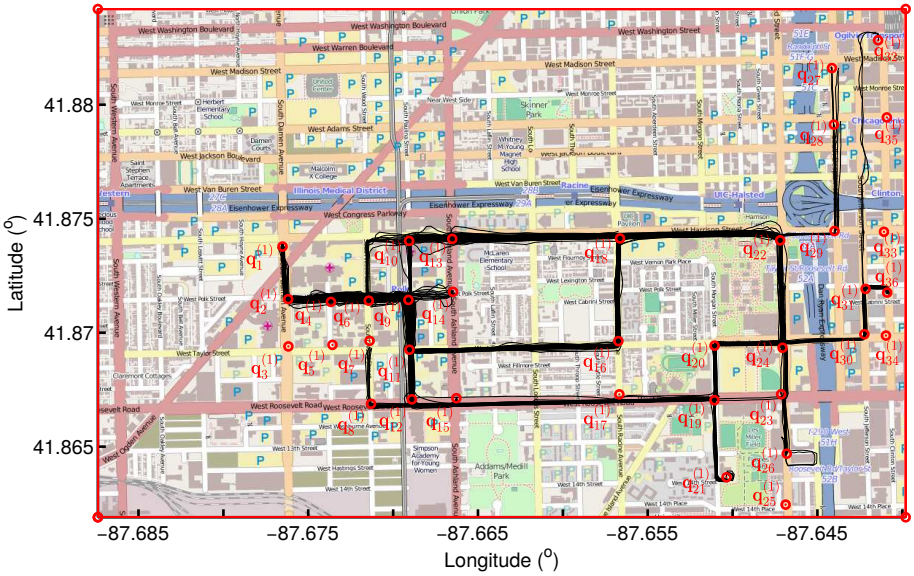
#### 6.6.1.3 Results of Track Clustering

The directed tracks in Fig. 6.20 are clustered using the algorithm based on similarity measure in Section 2.4. Only the normal tracks with high similarity to each other are aligned and used to extract the geometric representation of each road segment and analyze the velocity variance along each road segment. Fig. 6.21a and 6.21b show the normalities and abnormalities of the tracks in Fig. 6.20a. The tracks in Fig. 6.20b are grouped into normal tracks in Fig. 6.22a and abnormal tracks in Fig. 6.22b.

From Fig. 6.21b and 6.22b, we can see that most of the abnormal tracks are located at the same area, i.e., the one with tall buildings. In Fig. 6.20b,

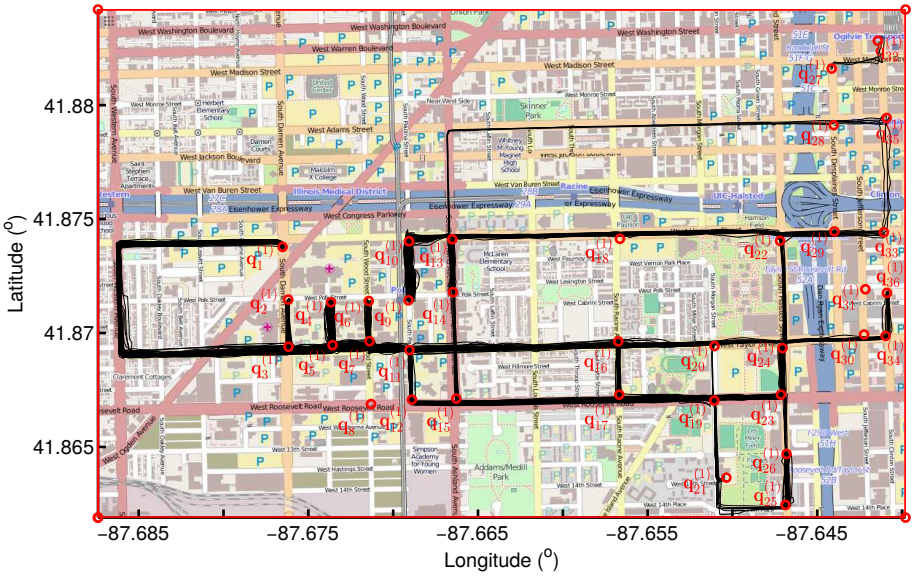



**(a)** Tracks 1

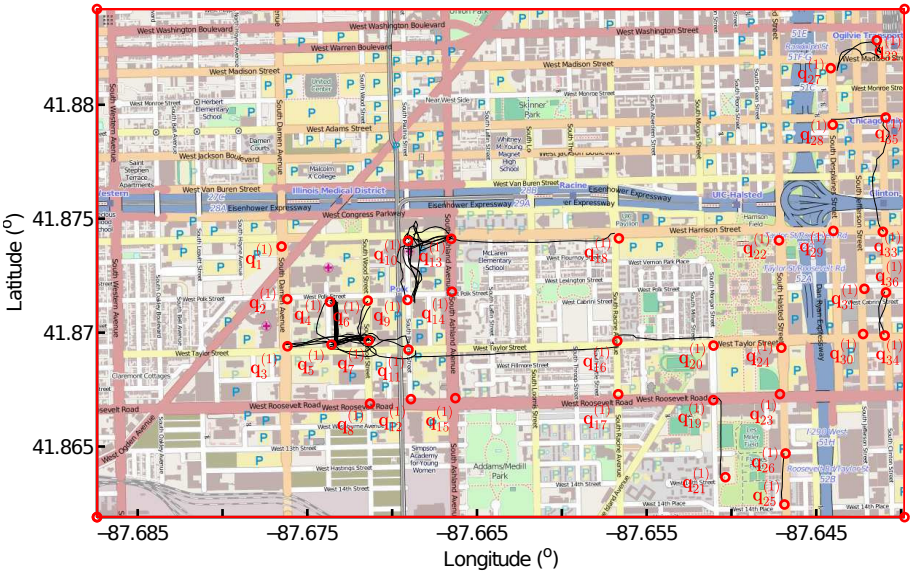



**(b)** Tracks 2

**Figure 6.20:** *Directed GPS tracks.* (a) shows the tracks for directed road segments from higher-index intersections to lower-index intersections. (b) shows the tracks for directed road segments from lower-index intersections to higher-index intersections.

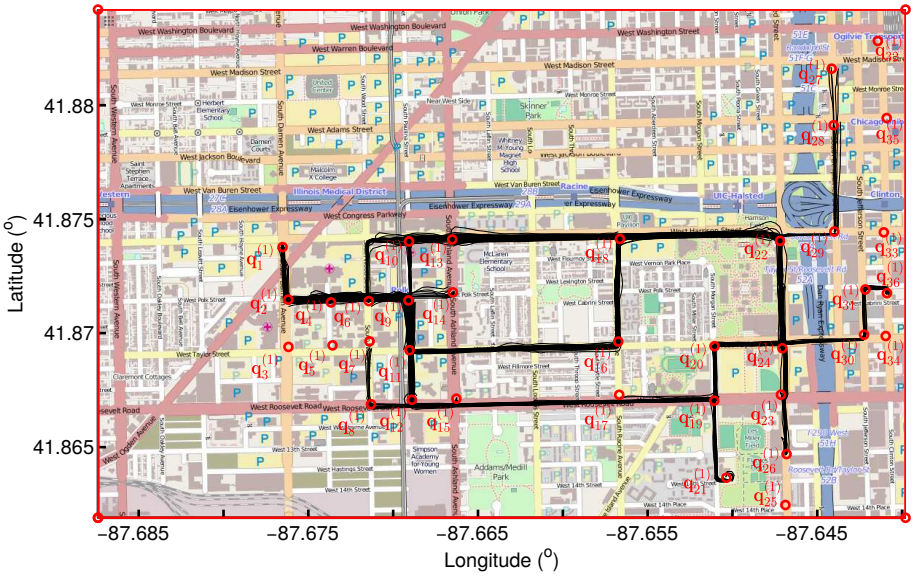



**(a)** Normal tracks 1

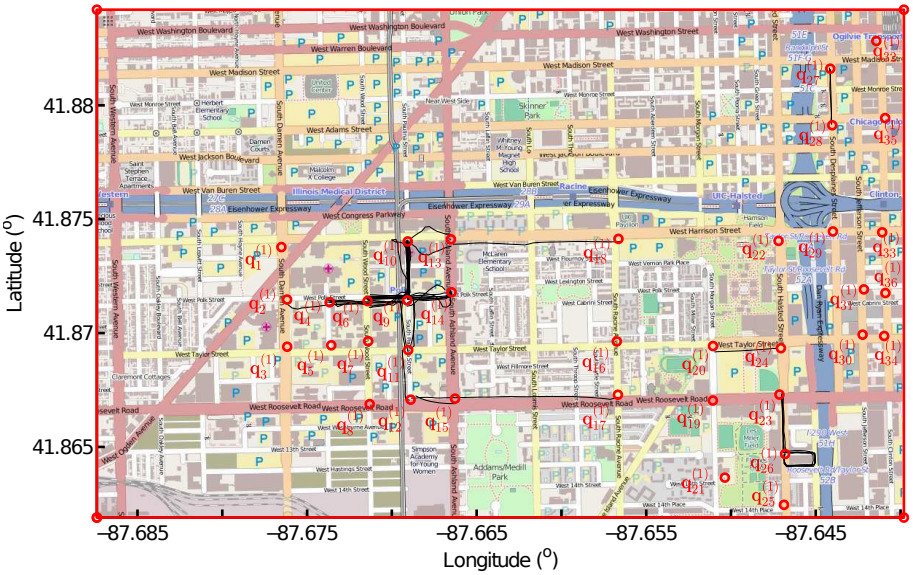



**(b)** Abnormal tracks 1

**Figure 6.21:** *Track clustering* 1. Tracks in Fig. 6.20a, which start from the higher-index intersections and end at the lower-index intersections, are clustered into normal tracks in (a) and abnormal tracks in (b).




**(a)** Normal tracks 2




**(b)** Abnormal tracks 2

**Figure 6.22:** *Track clustering* 2. (a) and (b) are the normal and abnormal tracks of Fig. 6.20b, respectively. The tracks start from the lower-index intersections and end at the higher-index intersections.

there is one track from Intersection $\mathbf{q}_{15}^{(1)}$ to $\mathbf{q}_{17}^{(1)}$, which is above other tracks between these two intersections. Our algorithm successfully detects it as abnormal track, as shown in Fig. 6.22b. Without removing this track, the geometric representation, which is averaged from the warped tracks, will be inaccurate.

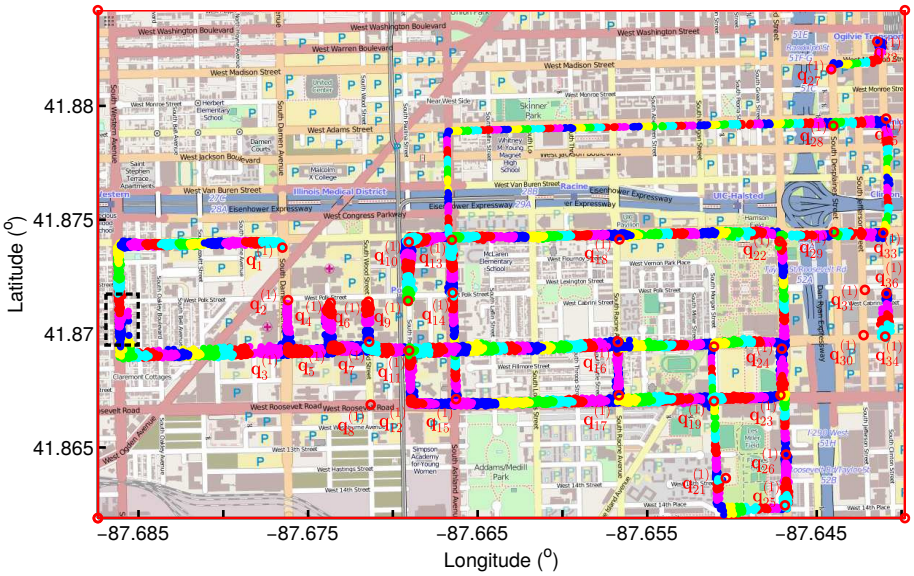
The ground truth map in Fig. 6.12 shows two intersections around the location of the extracted intersection $\mathbf{q}_{26}^{(1)}$. Our algorithm merged them together because the turning points are too close to each other. From Intersection $\mathbf{q}_{23}^{(1)}$ to $\mathbf{q}_{26}^{(1)}$, two types of tracks are detected: one is directly from Intersection $\mathbf{q}_{23}^{(1)}$ to $\mathbf{q}_{26}^{(1)}$; the other one is from Intersection $\mathbf{q}_{23}^{(1)}$ to $\mathbf{q}_{26}^{(1)}$, and to $\mathbf{q}_{26}^{(1)}$ again through a small loop. The first type of tracks are detected as the normal ones in Fig. 6.22a and the second type of tracks as the outliers in Fig. 6.22b, because one intersection is not connected to itself by our connectivity definition.. Only the normal tracks from Intersection $\mathbf{q}_{23}^{(1)}$ to $\mathbf{q}_{26}^{(1)}$ in Fig. 6.22a are aligned in the next step.

#### 6.6.1.4 Results of Track Alignment

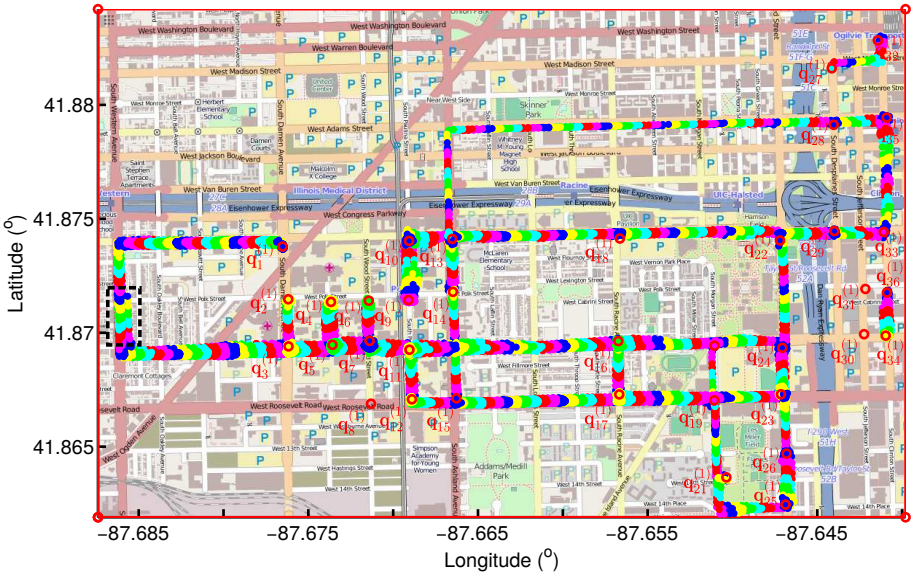
The normal tracks in Fig. 6.21 and 6.22 are aligned separately using the stretch-and-then-compress method and using the greedy method based on successor classification, resulting in Fig. 6.23 and 6.24. Fig. 6.23a shows the point associations of part of tracks, shown in Fig. 6.21a, using the stretch-and-then-compress method, and Fig. 6.23b is for point associations of the same tracks using the greedy method based on successor classification. Fig. 6.24a and 6.24b show the point associations of the other part of tracks, shown in Fig. 6.22a, separately using each alignment method. The associated points are indicated in the same color.

Because the *compress* operation keeps only one pair of points with the largest similarity among the *many-to-one* correspondence, the stretch-and-then-compress method produces the warped tracks shown in Fig. 6.23a and 6.24a, which have fewer points than the original normal tracks shown in Fig. 6.21a. Due to the *many-to-one* correspondence produced by the greedy method based on successor classification, there are more data points in the warped tracks than the original tracks, as shown in Fig. 6.23b and 6.24b. Although the warped tracks produced using the greedy method based on successor classification have more points than the original tracks, the physical locations of the points do not change because the greedy method only duplicates data points at the same position along the warp paths.

The alignment of the tracks in Fig. 6.23b and 6.24b is more smooth than that in Fig. 6.23a and 6.24a. For instance, the *compress* operation of the stretch-and-then-compress method removes some data points of the tracks on the road segment from Intersection $\mathbf{q}_{3}^{(1)}$ to $\mathbf{q}_{1}^{(1)}$, but the greedy method based on successor classification keeps all of the data points, as shown in the black boxes. Therefore the greedy method produces better and denser representation of the road segment.

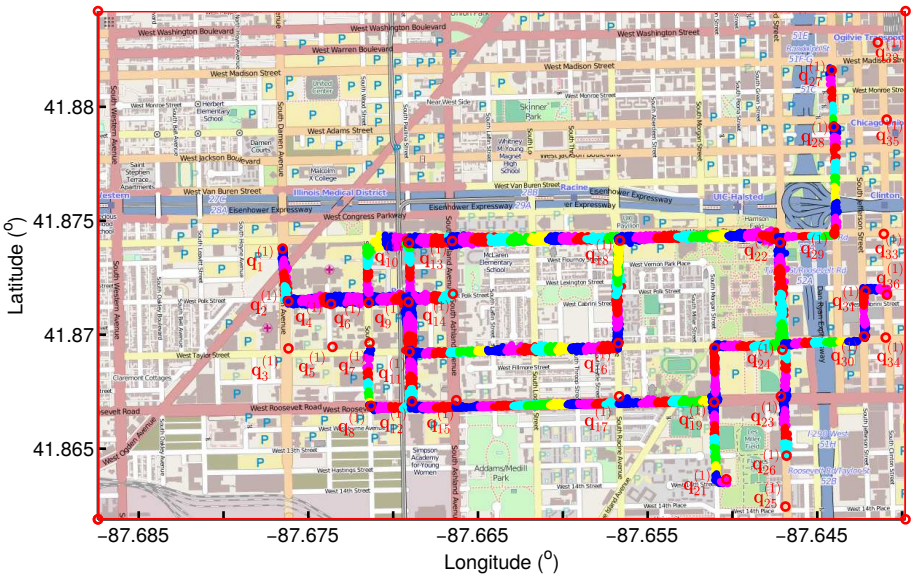



**(a)** Stretch-and-then-compress method

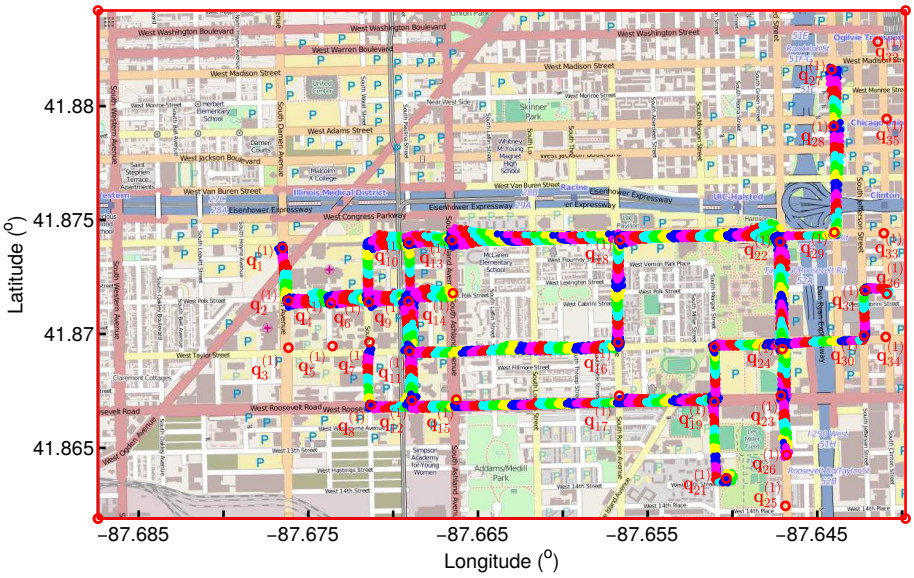



**(b)** Greedy method based on successor classification

**Figure 6.23:** *Track alignment* 1. (a) depicts the point associations of the tracks in Fig. 6.21a using the pairwise alignments with a "stretch and then compress" strategy, and (b) shows the point associations of the same tracks using the greedy method based on successor classification. The associated points are indicated in the same color.




**(a)** Stretch-and-then-compress method




**(b)** Greedy method based on successor classification

**Figure 6.24:** *Track alignment* 2. The tracks in Fig. 6.22a are aligned using two methods respectively, resulting in the point associations, as shown in (a) and in (b).

#### 6.6.1.5 Results on Track Averaging

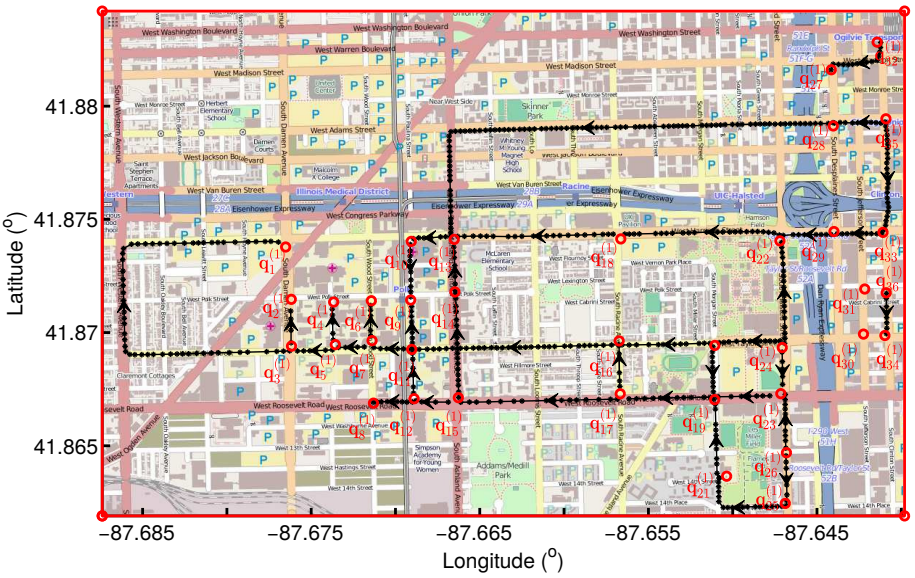
We first show the average tracks as the geometric representations of the directed road segments, then depict the relation between the aligning order of the tracks and the average distance between the average tracks inferred using the stretch-and-then-compress method and their ground truth. At the end, we present the average velocity and the velocity variance along the average track for each road segment, extracted using two proposed alignment methods respectively.

**Geometric accuracy** The tracks in Fig. 6.21a are averaged using the point associations established by the stretch-and-then-compress method and the greedy method based on successor classification separately, resulting in Fig. 6.25a and Fig. 6.25b, repetitively. Fig. 6.25a and Fig. 6.25b are the geometric average tracks for the tracks in Fig. 6.21a. The arrows indicate the direction of the roads, and the black dots are for the data points representing the road segments. Fig. 6.26a and Fig. 6.26a show the geometric representation of the directed roads which are averaged from the tracks in Fig. 6.22a using the alignments produced by two methods respectively.

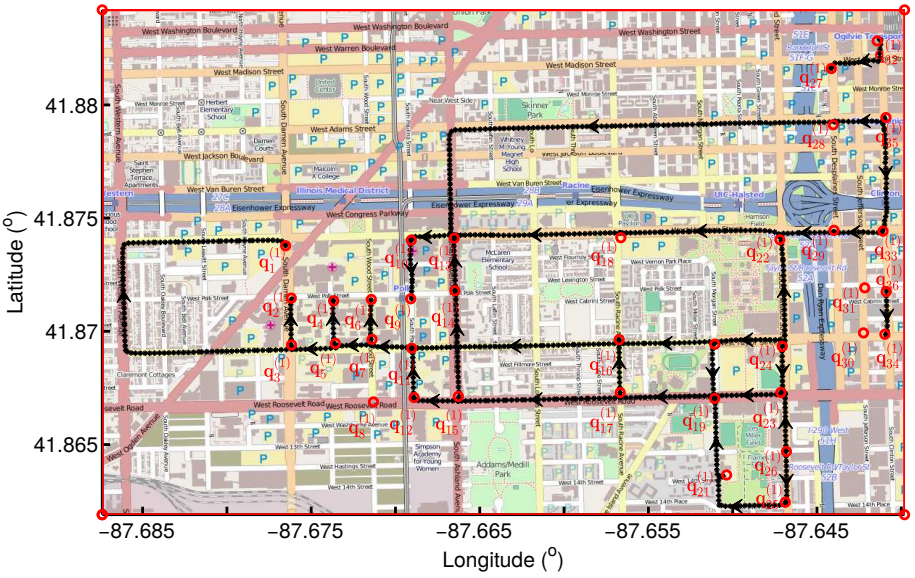
As shown in Fig. 6.25 and 6.26, the greedy method based on successor classification produces more points for the directed roads than the stretch-and-then-compress method. For instance, the stretch-and-then-compress method keeps only 4 points as the geometric representation of the road segment from Intersection $\mathbf{q}_3^{(1)}$ to $\mathbf{q}_2^{(1)}$, as shown in Fig. 6.25a. The greedy method based on successor classification produces 91 data points to represent the same road segment. This proves that the greedy method is able to produce a geometric representation of the road segment in more details than the other method, but sometimes the details are superfluous because the adjacent points in the average track are too close to each other.

To analyze the influence of the various sampling rates on the road generation for each of the proposed method, we re-sampled the tracks for road segment from $\mathbf{q}_3^{(1)}$ to $\mathbf{q}_1^{(1)}$ at from 1/2 to 1/10 times the original sampling rate. The number of data points in the original tracks ranges from 80 to 96 (average: 89). After re-sampling, it ranges from 11 to 92 (average: 33). We applied both of the alignment methods on the re-sampled tracks separately, and obtained the road representation as shown in Fig. 6.27. The re-sampled tracks are depicted in black lines with stars indicating the data points of the tracks. The generated road representation is shown in blue lines with dots. 10 points are kept using the stretch-and-then-compress method, with information lost at the two "bends." The greedy method based on successor classification, represent the road segment with 93 points. Although the distribution the points is uneven, the road shape is not changed in the generated geometric representation. It proves that the greedy method based on successor classification outperforms the stretch-and-then-compress method in producing good geometric representation for tracks with various sampling rates.

The average distance between all of the generated road segments and their ground truth is used as a measure geographical accuracy of the generated road

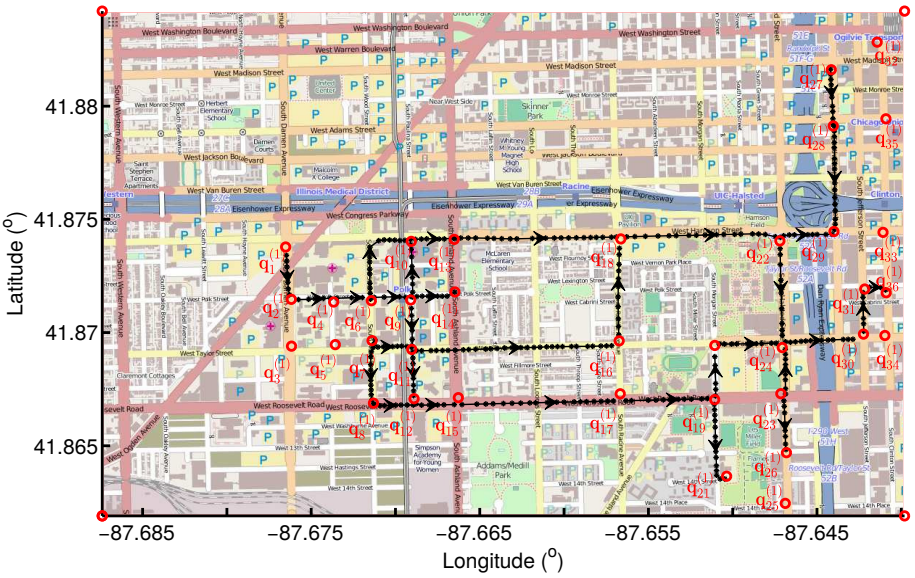



**(a)** Stretch-and-then-compress method

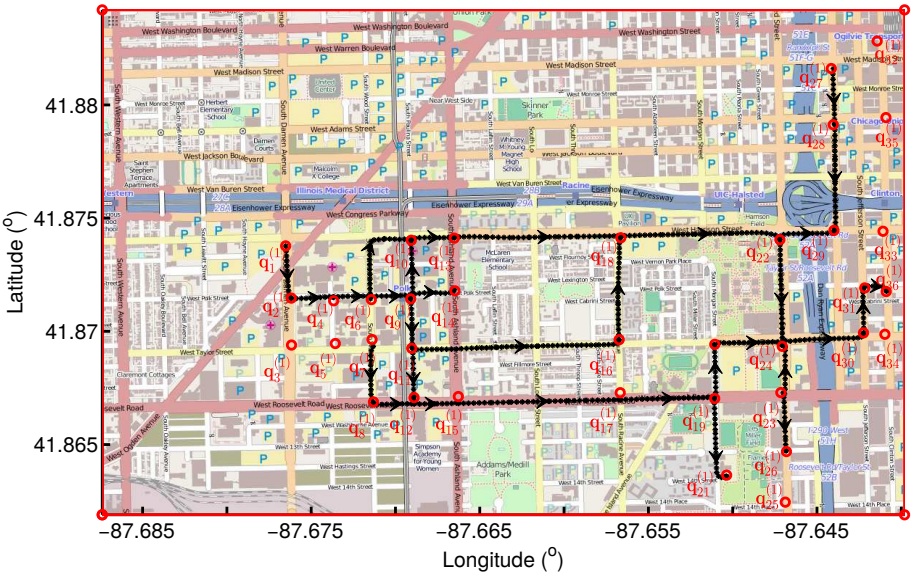



**(b)** Greedy method based on successor classification

**Figure 6.25:** *Geometric representation* 1. The tracks in Fig. 6.21a, which start from the higher-index intersections and end at the lower-index intersections, are averaged to form the geometric "average tracks" using two alignment methods, resulting in (a) and (b) respectively.

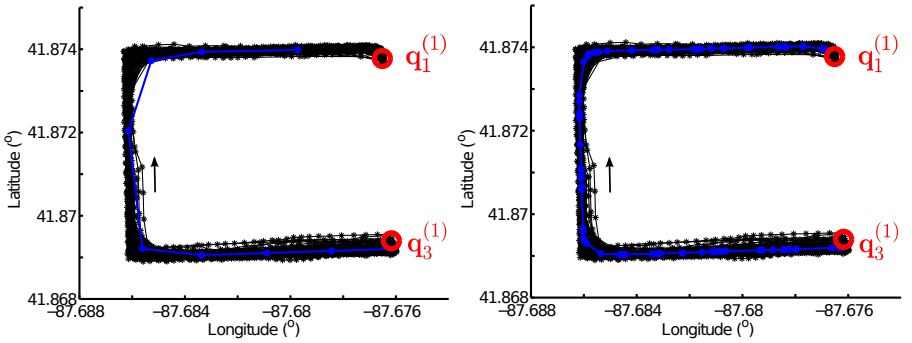



**(a)** Stretch-and-then-compress method




**(b)** Greedy method based on successor classification

**Figure 6.26:** *Geometric representation* 2. The tracks in Fig. 6.22a, which start from the lower-index intersections and end at the higher-index intersections, are averaged to form the geometric "average tracks" using two methods separately, resulting in (a) and (b).




**(a)** Stretch-and-then-compress method

**(b)** Greedy method based on successor classification

**Figure 6.27:** Generated road using tracks with various sampling rate. The geometric representation of the road segment from $\mathbf{q}_3^{(1)}$ to $\mathbf{q}_1^{(1)}$ is extracted from re-sampled tracks using our two alignment methods respectively. (a) depicts the road representation with 10 points using the old alignment method. The new method produces 93 points in (b).

network. We align all of the tracks together simultaneously using the greedy method based on successor classification, producing a fixed average distance of 6.87 meters. For the stretch-and-then-compress method, it is different if the tracks are aligned in a different order.

The results shown in Fig. 6.25a and 6.26a are extracted using a specific order based on their similarity score to other tracks, which is calculated as the sum of their similarity score to every other track. To evaluate the influence quantitatively, we also averaged the tracks by aligning them in 50 other random orders. The average distance between the generated road segment and its ground truth with 50 different orders is calculated and shown in Fig. 6.28. The horizontal and vertical axis indicate the index of aligning order and the average distance, respectively. No matter which order is selected to align the tracks, the distance between the generated road segments and the ground truth is always around 7.38 meters with maximum value 7.85 meters and minimum value 6.81 meters, which means the aligning order has no significant influence on averaging the highly similar tracks.

Considering that the precision of the OpenStreetMap is around 10-20 meters, both proposed methods show excellent performance in terms of geographical accuracy, which is 6.87 meters for the greedy method based on successor classification, and around 7.38 meters for the stretch-and-then-compress method.

**Average Velocity and Velocity variance** For each road segment, the average velocity and the velocity variance along the data points of the average track are calculated as explained in Section 6.4. Fig. 6.29 and 6.30 show the re-

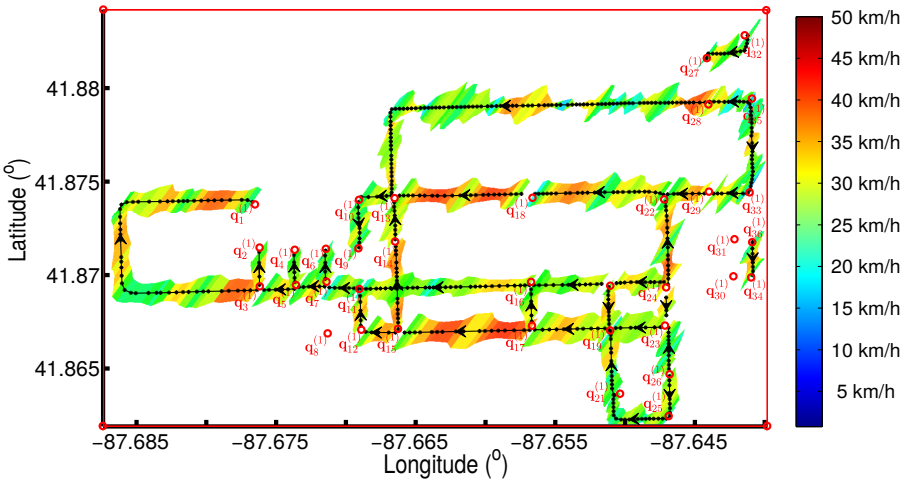

**Figure 6.28:** Influence of the aligning order using Method 1. No matter in which order the tracks are aligned, the average distance between all of the generated road segments and their ground truth is around 7.37 meters stably, which means the aligning order does not affect the track alignment significantly.

sults. Fig. 6.29 is for the tracks on the directed road segments which start from higher-index the intersections and end at lower-index ones, as shown in Fig. 6.21a. Fig. 6.29a depicts the results which are calculated using the point assignations in Fig. 6.23a produced using the stretch-and-then-compress method. Fig. 6.29b shows the results of the same tracks using the point assignations in Fig. 6.23b produced using the greedy method based on successor classification. The color of the points represents the strengths of the velocity, and the width of the band around the average track indicates the variance of the velocity. The wider the band is, the more variation in velocity.

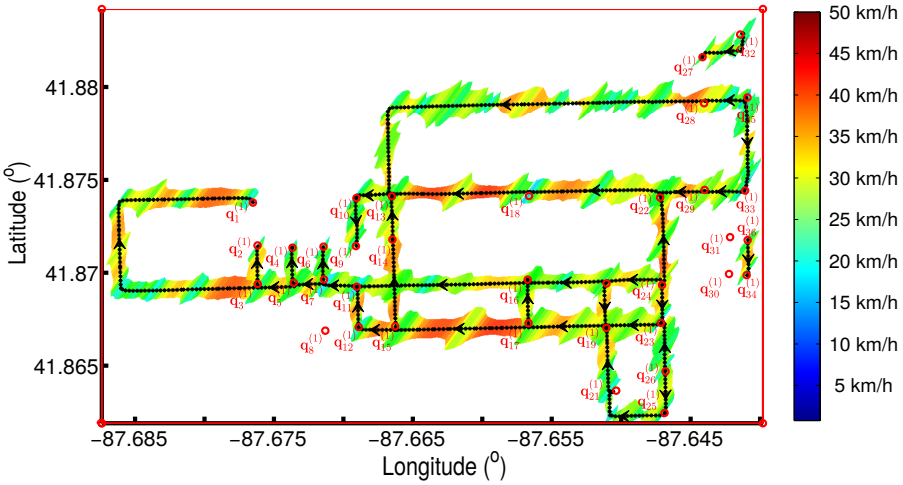
The two sub-figures in Fig. 6.30 show the average velocity and the velocity variance of the other part of the tracks on the directed road segments, which connect higher-index intersections to lower-index intersections. Fig. 6.30a and 6.30b are for different alignment methods respectively.

Because more data points are kept to represent the geometry of the road segments using the greedy method based on successor classification than the stretch-and-then-compress method, the average velocity and velocity variance are more smoothly in Fig. 6.23b and 6.24b. Because the average velocities calculated are only slightly different, and the velocity variances along the average tracks are also similar, both of the proposed methods can be used to do the statistical analysis.

From Fig. 6.29 and 6.30, we can see that the road users mostly drive at a speed between 25 to 35 kilometers per hour. Sometimes the speed is as high as 40 km/h on some roads, for instance, on the directed road segments between Intersection $\mathbf{q}_{17}^{(1)}$ and $\mathbf{q}_{15}^{(1)}$ in both directions. The velocity of the shuttles on the road segment from Intersection $\mathbf{q}_{28}^{(1)}$ to $\mathbf{q}_{13}^{(1)}$ varies widely, resulting in uneven bands around the average track.

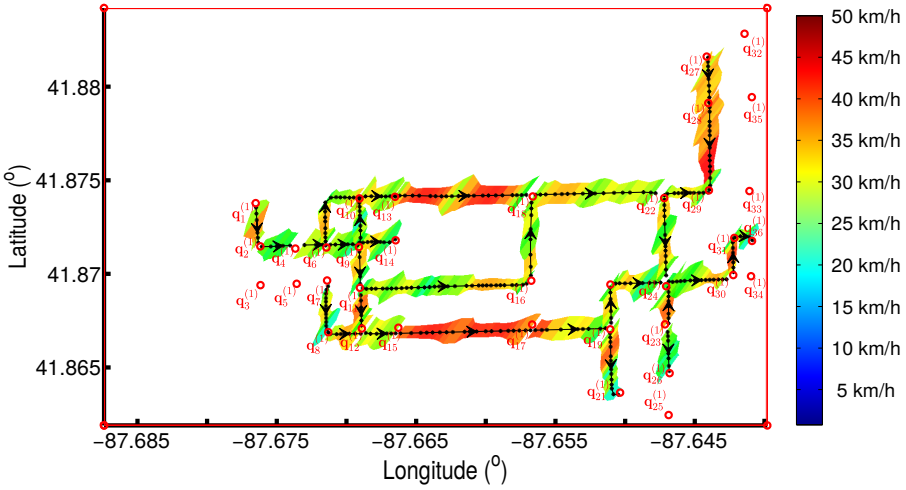



**(a)** Stretch-and-then-compress method

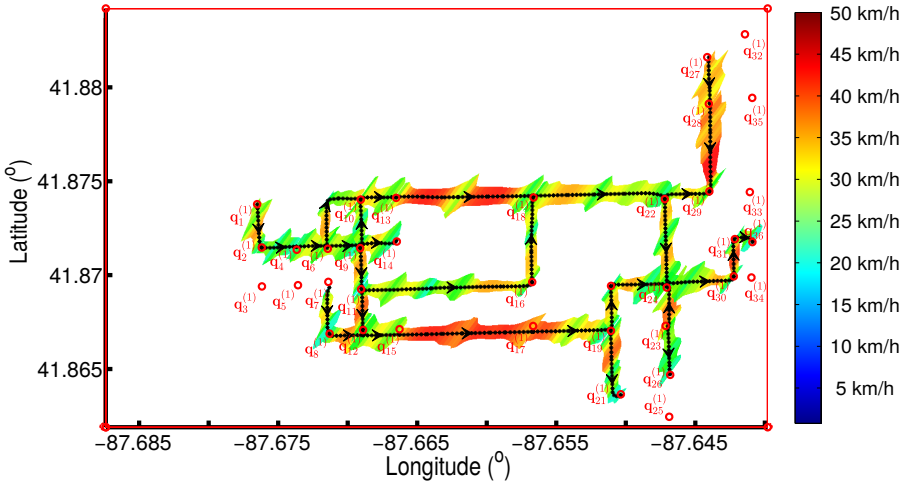



**(b)** Greedy method based on successor classification

**Figure 6.29:** *Average velocity and its variance* 1. The average velocity of the tracks in Fig. 6.21a is calculated using the point assignations in Fig. 6.23a produced through pairwise alignments using a "Stretch and then Compress" strategy, resulting in (a). (b) shows the average velocity of the same tracks using the greedy method based on successor classification, as shown in Fig. b.




**(a)** Stretch-and-then-compress method




**(b)** Greedy method based on successor classification

**Figure 6.30:** *Average velocity and its variance* 2. The average velocity of the tracks in Fig. 6.22a is calculated using the point assignations in Fig. 6.24a produced by the stretch-and-then-compress method, resulting in (a). (b) shows the average velocity of the same tracks using the point assignations in Fig. 6.24b produced by the greedy method based on successor classification.

#### 6.6.1.6 Comparison with Other Methods

Our results are compared to the following existing methods: the curve fitting method proposed by Schroedl [Schroedl 04], the KDE-based method by Davies [Davics 06] and also the trace-merging method proposed by Cao [Cao 09]. Since qualitative and quantitative evaluations of these three existing algorithms have been made by Biagioni and Eriksson in [Biagioni 12a] on the same Chicago campus shuttle dataset as us, we can compare the results of their algorithms directly with ours.

Fig. 6.31 shows our results for the same two examples that Biagioni used in his paper [Biagioni 12a], one example at the area of tall-rise buildings with high-error GPS traces, while the other one at the area of low-rise buildings with low-error GPS traces. The directed roads are depicted in different colors depending on their directions. Magenta is for the directed roads from the low-index intersections to high-index intersections, and blue is for the directed roads from the high-index intersections to low-index intersections. As shown in Fig. 6.31, both of our alignment algorithms produce clean directed roads without spurious edges. Because of the *compress* operation of the stretch-and-then-compress method, the inferred geometric representation of the road segments contain less data points. Although the detected intersection $\mathbf{q}^{(1)}$ does not locate at the center of the crossroad, the road segments it connects intersecting at the center. In the future, we will refine the intersection locations after extracting the geometric representations of the road segments.

Fig. 6.31 shows the results of three other methods with the same two examples as Biagioni and Eriksson presented in his paper [Biagioni 12a]. All of the methods work well in areas with low GPS noise, as shown in Fig. 6.32b, 6.32d and 6.32f. Although Cao and Krumm apply a clarification preprocessing to reduce the effects of GPS noise, they are not able to merge the spatially dispersed traces into one graph. Therefore a large amount of spurious edges are produced at the area of high GPS noise using their method, as shown in Fig. 6.32a. Edelkamp and Schroedl cluster the raw data points, merge the cluster seeds to road segment, and apply a curve fitting methods to generate the geometric representation for each road segment. Because this cluster-merging method is easily led astray by GPS noise, spurious roads are also produced at the area of high GPS noise, as shown in Fig. 6.32e. Our algorithms, based on intersection detection, extract the road topology successfully without adding any extraneous edges. Although Davies also produces clean roads, our methods still outperform his, because our generated roads are directed, while he only extracted undirected roads.

### 6.6.2 Results of Berlin Data Set

In this section, we present the results of Berlin data set. We first show the intersections detected using both of the proposed methods elaborated in Section 6.6.2.1. The results of GPS trace segmentation and clustering are illustrated in Section 6.6.2.2. At last, we give the track averaging results in Section 6.6.2.3.

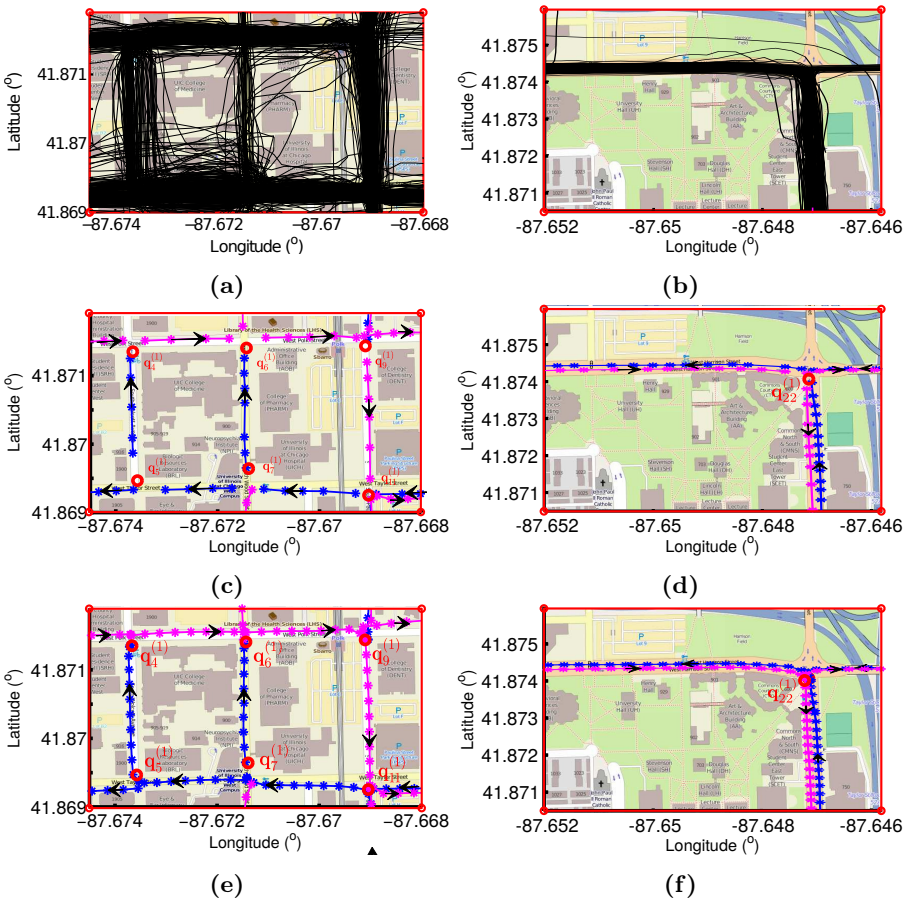



**Figure 6.31:** Results using our proposed methods in areas with high and low GPS error respectively. (a) shows an area with high-error GPS tracks, which are used to produce the directed roads in (c) using the stretch-and-then-compress method and (e) using the greedy method based on successor classification. The low-error GPS tracks shown in (b ) are used to extract the directed roads in (d) and (f) using our two proposed alignment methods respectively.

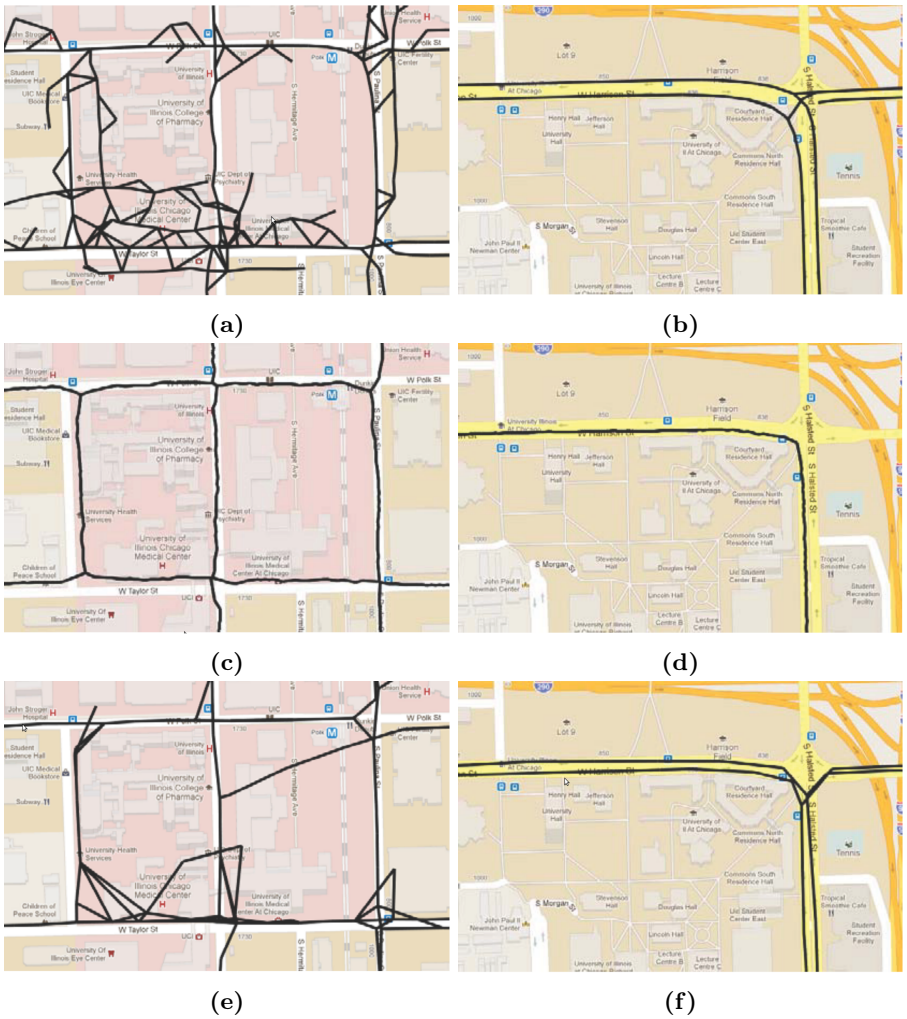



**Figure 6.32:** Results of 3 other algorithms in areas with high and low GPS error respectively. (a) and (b) show the results of high-error and low-error sample respectively using Cao's algorithm. (c) and (d) are for Davies' algorithm, and (e) and (f) for Edelkamp's algorithm.

#### 6.6.2.1 Results of Intersection Detection

Fig. 6.33 and 6.34 show the intersection detected from turning points and connecting points, respectively. By checking the moving direction on the GPS traces, 41,974 turning points are detected. Due to low sampling rate and high GPS error, we detect turning points both at the location of the intersections and also on the road segments for this large-scaled area with very dense road network, as shown in Fig. 6.33a. The clustering method in Section 6.3.1.2 is no longer suitable for identifying intersections from turning points. For Berlin dataset, we apply the KDE on the turning points and find local maximas as intersections, the same as we do for intersection detection from connecting points. In total, there are 252 intersection detected, as shown in Fig. 6.33b.

By finding the common sub-tracks between pairwise GPS traces, we detect 711,830 connecting points, as shown in Fig. 6.34a. From the connecting points, 240 intersections are identified, as shown in Fig. 6.34b. A lot of connecting points are detected at locations where abnormal traces deviate from the roads, therefore more spurious intersections are detected, compared to the turning point method. Sometimes the inaccurate GPS traces diverge on the road segment rather than at the intersection, leading to that the connecting points detected at somewhere close to the intersection. Thus several intersections could be detected around a true intersection, on the roads the true intersection connects.

At the middle-left area of the map, GPS traces on different roads mix together because of GPS error. It is difficult to separate them for different road segments, therefore neither of our methods work well at this area.

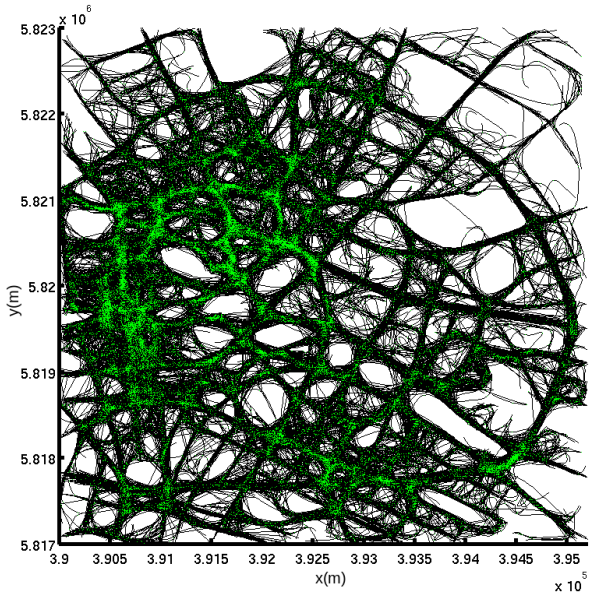
#### 6.6.2.2 Results of GPS Trace Segmentation and Clustering

Because the intersections detected from turning points show higher accuracy than that from connecting points, the intersections in Fig. 6.34b are used to segment GPS traces. With 150 meters as the matching threshold, the GPS traces shown in Fig. 6.11b are segmented into tracks shown in Fig. 6.35. Fig. 6.35a and 6.35b shows the tracks in different directions, respectively.

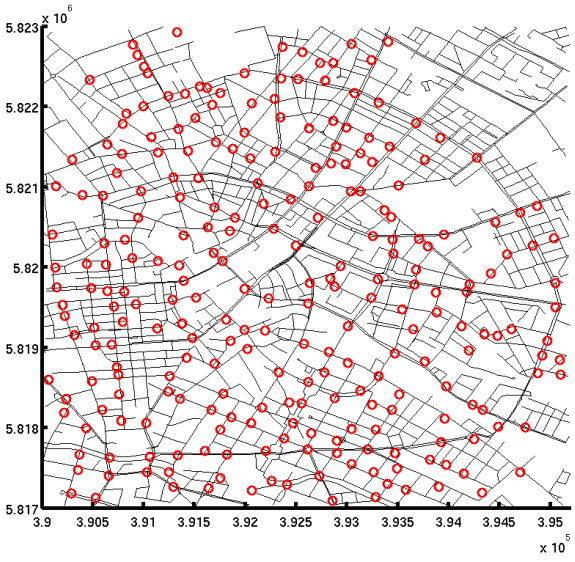
Using the algorithm based on similarity measure presented in Section 2.4, the directed tracks in Fig. 6.35a are clustered into normal tracks shown in Fig. 6.36a and abnormal tracks shown in Fig. 6.36b. The tracks in Fig. 6.35b are grouped into normalities and abnormalities as shown in Fig. 6.37a and Fig. 6.37b respectively. The normal tracks show the road network more clearly. Removing the abnormal tracks for track averaging helps to improve the accuracy of the generated geometric representation.

#### 6.6.2.3 Results on Track Alignment and Averaging

Most methods of road map generation in literature could not cope with big data set. The methods we used to process Chicago data set for comparison fail to align the tracks in Berlin data set, as well as our stretch-and-then-compress method. Therefore for Berlin data set, we will only show the results of the

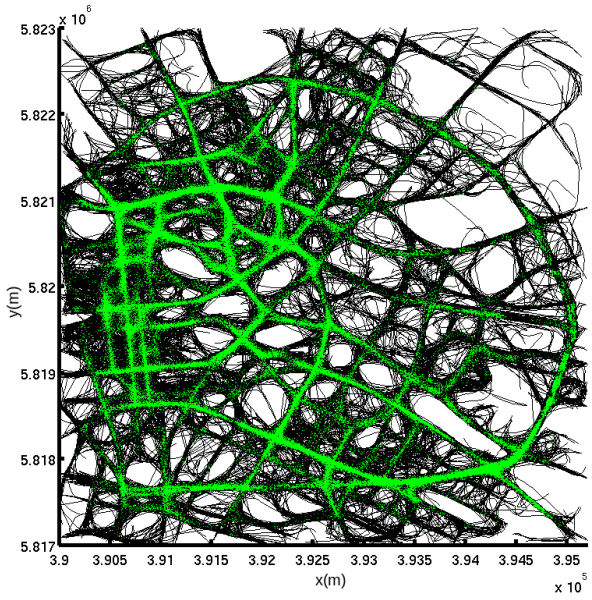



**(a)** Turning points

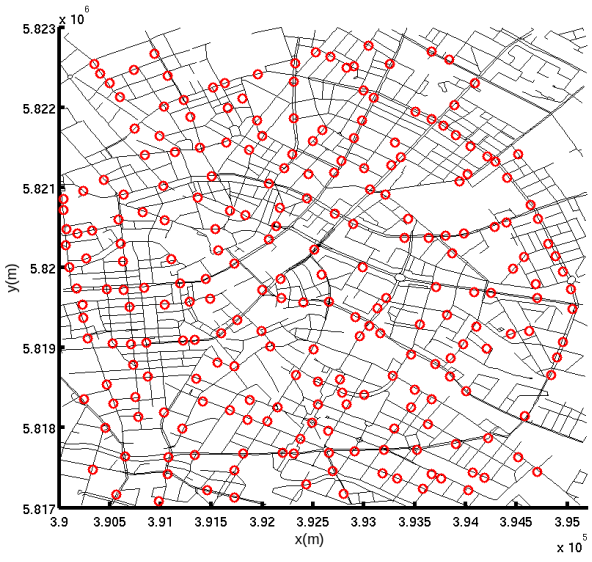



**(b)** Intersections

**Figure 6.33:** *Intersections detected from turning points.* (a) shows 41,974 turning points using green dots. (b) shows 240 intersections in red circles, which are detected from the turning points.

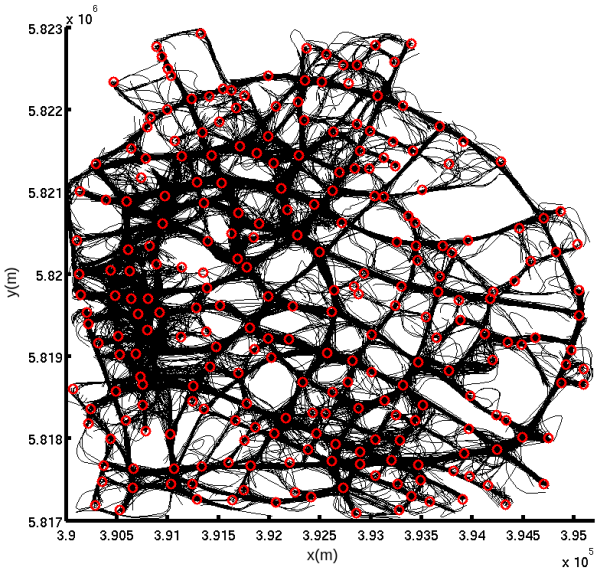



**(a)** Connecting points

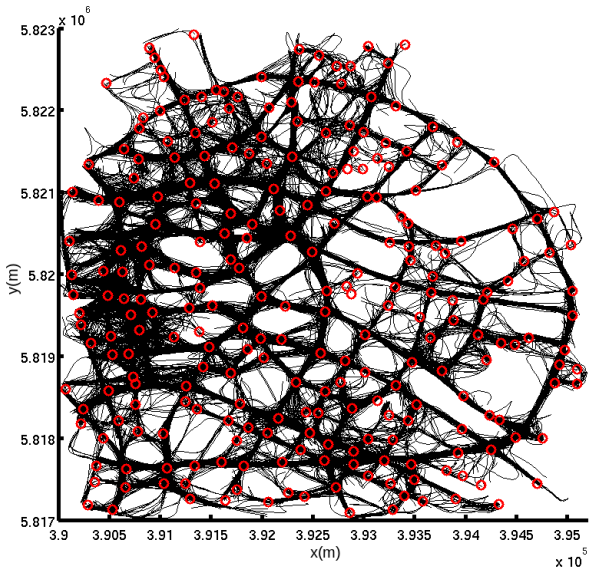



**(b)** Intersections

**Figure 6.34:** *Intersections detected from connecting points.* (a) shows 711,830 turning points using green dots. (b) shows 252 intersections in red circles, which are detected from the connecting points.

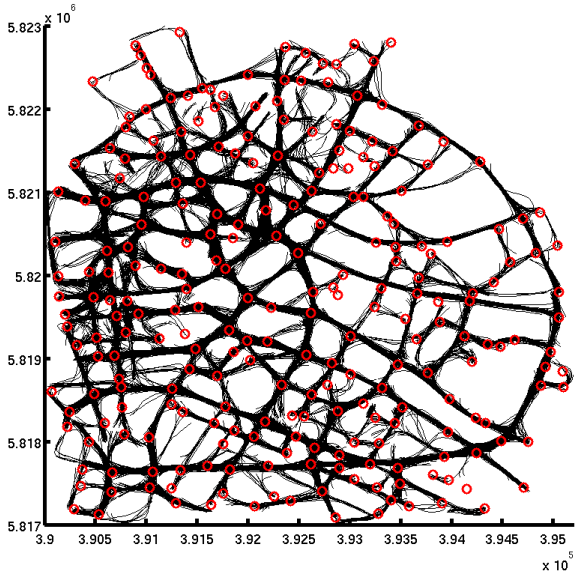



**(a)** Tracks 1

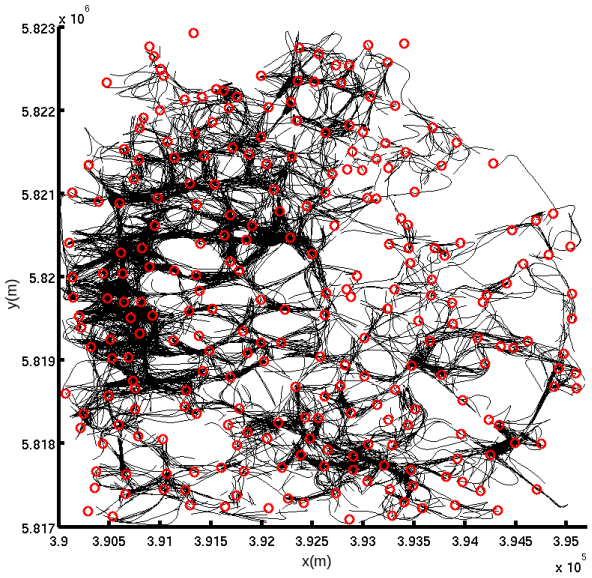



**(b)** Tracks 2

**Figure 6.35:** *Directed GPS tracks.* (a) and (b) shows the tracks segmented from the GPS traces by intersections in different directions, respectively.

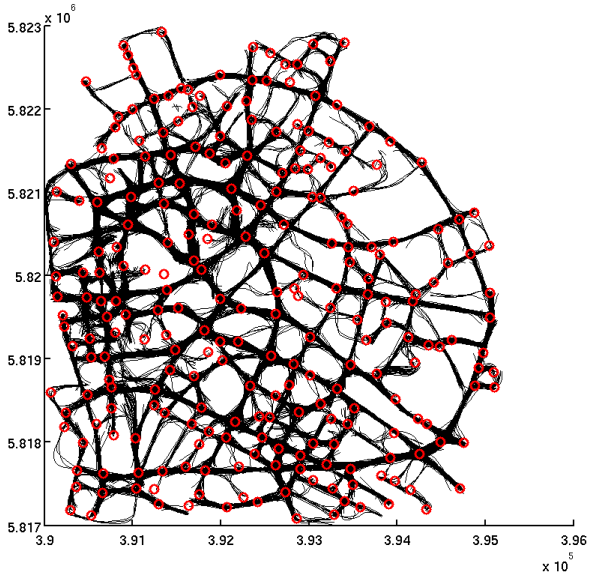



**(a)** Normal tracks 1

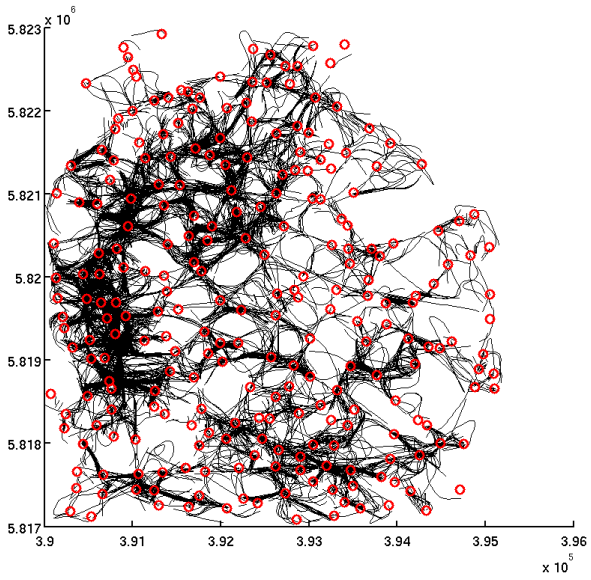



**(b)** Normal tracks 1

**Figure 6.36:** *Track clustering* 1. Tracks in Fig. 6.35a are clustered into normal tracks in (a) and abnormal tracks in (b).

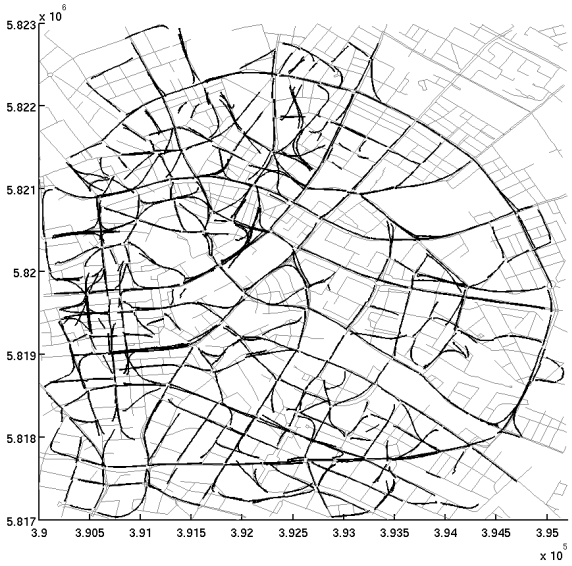



**(a)** Normal tracks 2

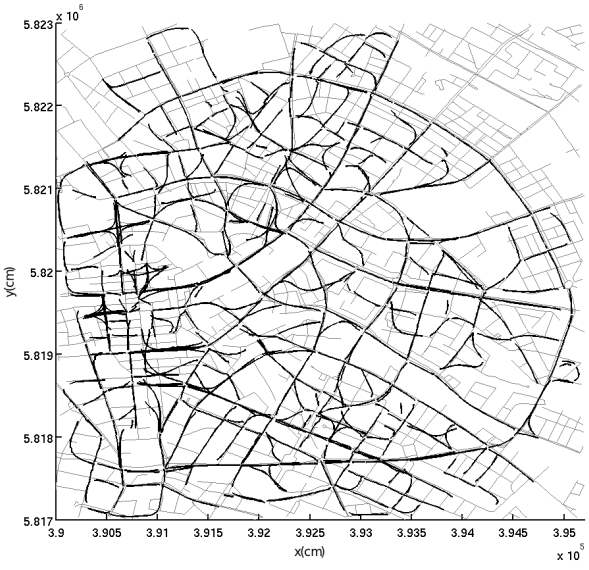



**(b)** Normal tracks 2

**Figure 6.37:** *Track clustering* 2. Tracks in Fig. 6.35b are clustered into normal tracks in (a) and abnormal tracks in (b).

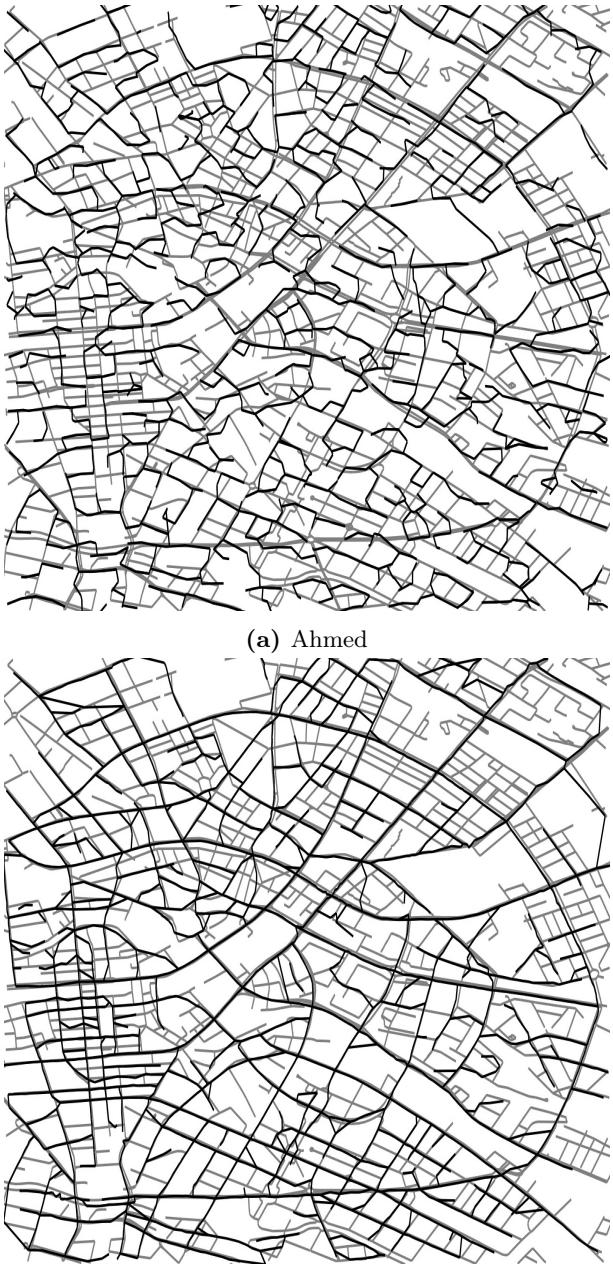



**(a)** Average tracks 1




**(b)** Average tracks 2

**Figure 6.38:** *Geometric representation.* Tracks in Fig. 6.36a are aligned to form the average tracks in (a), and (b) is from the tracks Fig. 6.37a.



**(a)** Ahmed

**(b)** Karagiorgou

**Figure 6.39:** *Comparison.* (a) shows the results of Ahmed's trace-merging method, and (b) is for the results of Karagiorgou's intersection-linking method.

greedy method based on successor classification, and compare with other two methods: the trace-merging method proposed by Ahmed [Ahmed 12] and the intersection-linking method by Karagiorgou [Karagiorgou 12]. Since Ahmed *et al.* have evaluated both her and Karagiorgou's method on Berlin dataset in her later work [Ahmed 14]. we can compare our greedy method with their methods directly.

The normal tracks with high similarity to each other, as shown in Fig. 6.36a and 6.37a, are aligned, producing warped tracks with associated points at the same time index. The associated points are averaged to form the geometric representation of the road segment, resulting in the average tracks shown in Fig. 6.38. The average tracks are shown in black lines, and the ground truth in grey lines. Because of the low accuracy of intersection detection at the middle-left area of the map, we detect spurious road segments from falsely connected intersections. At the top-right corner, the road users keep moving straight, resulting in few turning points detection. Thus no intersection is detected, leading to no road segment extracted.

Fig. 6.39 shows the results of Ahmed's and Karagiorgou's method on Berlin dataset [Ahmed 14]. As shown in Fig. 6.39a, Ahmed *et al.* extracted the most road segments from the dataset, but with low geographical accuracy. Especially a lot of spurious road segments can be found around the intersections. In Fig. 6.39b, Karagiorgou *et al.* produced highly accurate intersection nodes, resulting in a clean road network. However, they also suffered from producing spurious road segments around the intersections covered with deviating traces, as we do.

## 6.7 Conclusions

In this chapter, we presented two methods to detect intersections from GPS traces: one method analyzes the turning patterns of the road users; the other one utilizes connected road segments. Then we segmented the GPS traces and aligned the tracks for each road segment using our proposed alignment methods in Chapter 3. Based on the point assignations, we extracted the geometric representation of each road segment and analyzed the velocity of the road users along the road segment.

We tested our methods on two datasets: Chicago dataset with 889 GPS traces, and Berlin dataset with 26,831 GPS traces. Experimental results showed a higher accuracy of intersection detection, and a better geographical accuracy of the extracted road segments, compared to other algorithms.

The limitation of the stretch-and-then-compress method lies mainly in the sensitivity of the variety of data density. For various-sampling-rate GPS traces, this method produces average track with less data points than the GPS trace with the lowest sampling rate.

Future work will focus on inferring the existence and coverage of intersections and road segments statistically using a probabilistic method, instead of the two-valued logic in our current methods. Besides, we also intend to test

our algorithm in areas with more complicated road features, such as overpasses, non-orthogonal road segments and roundabouts.

This research resulted in three publications in international peer-reviewed journals: one published in International Journal of Geo-Information [Xie 15b], two under revision [Xie 16a, Xie 16b]. Furthermore one paper is published in the proceedings of the IEEE Intelligent Transportation Systems Conference [Xie 14c], and one submitted to 2016 IEEE International Geo-science and Remote Sensing Symposium [Xie 16c].

# 7

# Conclusions

*Ever tried. Ever failed. No matter. Try Again. Fail again. Fail better.*
—Samuel Beckett

In this thesis, we presented different techniques to analyze trajectories thoroughly, such as temporal dissimilarity measure, trajectory clustering based on dissimilarity measure, aligning many trajectories, etc. Specifically we applied these techniques in several applications: mainly work cycle optimization and road network inference.

## 7.1 Summary of Achievements

We have made contributions in similarity measure, joint trajectory alignment, and their applications.

### 7.1.1 Similarity Measure Approaches

Certain methods in machine learning and pattern recognition have been applied in trajectory clustering, such as K-means and Mean-shift algorithm [Li 06, Anjum 10]. The accuracy of the results using these clustering methods depends on the feature selection. Usually, overall distance, average speed and moving direction are selected as features to cluster the trajectories. These coarse features may lead to inaccurate trajectory clustering. For instance, two trajectories with the same average speed may be grouped into the same category even if their speed profiles are very different. Researchers then adapt other methods, such as agglomerative hierarchical clustering and spectral clustering, to categorize the trajectories using similarity measure, instead of coarse features. Using these methods, the clustering results depend on accuracy of the similarity measure between pairwise trajectories.

A variety of techniques, which are commonly used in other fields, have been applied to measure the spatial similarity between trajectories [Bashir 07, Niennattrakul 09, Vlachos 02, Zhang 06], such as DTW, LCSS, PCA and Euclidean distance. Although, the temporal similarity has not been well studied yet,

which describes whether the velocity difference between the points of the trajectories when moving objects follow the same route.

In Chapter 2, we proposed to measure trajectory (dis)similarity both spatially and temporally using DTW. The spatial dissimilarity was calculated through warping the trajectories along the time instance by finding the best match between the physical locations along the trajectories. The temporal dissimilarity was measured using the optimal warp path which minimizes the overall dissimilarity between the point velocities along the trajectories.

Using the spatial dissimilarity between pairwise trajectories, we presented a greedy clustering method to categorize the trajectories into different patterns, in which the objects followed different routes, and spatial abnormalities unfit to any of the routes. The same method was used to categorize the trajectories in each spatial cluster using temporal dissimilarity between pairwise trajectories. The cluster with the most trajectories was considered as the prototypical itinerary, and the trajectories in other clusters as temporal abnormalities, which were unfit to the speed profile of the prototypical itinerary.

We evaluated our proposed methods using a data set of 124 factory workers' trajectories, and compared our results with the results of two start-of-the-art similarity measures on the same data set. The experimental results showed a higher accuracy of trajectory clustering based on our similarity measures than on other similarity measures

### 7.1.2 Joint Trajectory Alignment Approaches

In literature, some strategies have been proposed to average the trajectories directly using pairwise trajectory alignment [Junejo 07, Junejo 08, Niennattrakul 09, Petitjean 11], such as aligning boundary trajectories, hierarchically aligning the two most similar trajectories, and globally updating a selected prototypical trajectory using other trajectories. Researchers aim to average many trajectories using these strategies. The point associations among all trajectories have not been established, which is not only helpful to trajectory averaging, but also beneficial to statistical analysis of the trajectories.

In Chapter 3, we proposed two methods to jointly align many trajectories. In our first method, we applied a "stretch and then compress" strategy based on DTW algorithm to align the trajectories one by one in ascending order of their average dissimilarity. Firstly, the first two trajectories were aligned using DTW in the *stretch* operation, producing two stretched trajectories. Secondly, the repeated points were removed from the stretched trajectories in the *compress* operation, producing two compressed trajectories with *one-to-one* correspondences between the points. A third trajectory was then aligned with the second compressed trajectory using the "stretch and then compress" strategy again. The procedure was repeated until the last trajectory is processed. The last compressed trajectory was taken as the warped trajectory, and the points in other trajectories, which were associated the points of last compressed trajectory, were induced through the intermediate warp paths between pairwise

adjacent trajectories, establishing *one-to-one* point correspondences among the warped trajectories of the same length.

In the second proposed method, we applied a greedy procedure to locate a good warp path through the local dissimilarity tensor in a way which keeps the points associated by each path element spatially close to each other. The successors of the current path element were classified to limit the candidate cells for the next path element. This reduced the prohibitively expensive computational cost of back-tracking the minimal overall dissimilarity for DTW. In this method, all of the trajectories were aligned simultaneously along the time dimension, instead of in a pairwise fashion. *Many-to-one* point correspondences among all trajectories were established.

We evaluated the proposed methods on both factory worker trajectories and vehicle trajectories. Results showed that the point correspondences were built successfully. The second method was proved to build better point correspondences. Compared to exiting methods, our proposed methods using the trajectory alignment showed better performance on averaging trajectories.

### 7.1.3 Intersection Detection Approaches

In literature, most researchers utilize the road users' moving direction change to detect intersections from the GPS trajectories [Karagiorgou 12, Wu 13, Wang 15]. In Chapter 6, we proposed two methods to identify intersections by defining them in two different ways.

In the first method, intersections were defined as the locations where the road users change their moving directions very often, as other researchers did in their work. Turning points with moving direction change were collected and clustered into intersection candidates. By checking the turn types the road users make at the intersection candidates, mis-detected bends were removed from the true intersections.

In the second method, intersections were defined as junctions which connect at least three road segments. The starting and ending points of the common sub-tracks between pairwise traces were detected as connecting points. The local maximas of the KDE of the connecting points were detected as intersections. Because the connecting points connect at least three road segments in different directions, the bends, which only connect two road segments, were not recognized as intersections.

We evaluated our proposed methods two datasets. Experimental results showed a high accuracy of intersection detection compared to the ground truth.

### 7.1.4 Applications

We applied the proposed techniques mentioned above in two of our applications. In the application of work cycle optimization in Chapter 5, a data set of 124 trajectories is used to analyze the work cycle of the factory workers. We clustered the trajectories into different types of executed work cycles and detected the abnormal trajectories which are unfit to any of the prototypical

itineraries. Through joint trajectory alignment, we averaged the trajectories as a prototypical route for each type of executed work cycle, and built the prototypical velocity and dwell time along each route. These would be used to guide the factory workers to execute their work cycles more efficiently. By analyzing the spatial and temporal abnormalities, we concluded the factors leading to work efficiency as: visiting other area of the work station unnecessarily, dwelling on his or her way to the storage rack or back to the assembly line, strolling in front of the storage rack too slowly, walking back and forth too many times in front of the storage rack, etc.

In the second application of the road network inference in Chapter 6, a data set of 889 GPS trajectories was used to extract the elements of the road network: intersections, intersection connectivity and geometric representation of each road segment. Intersections were detected from turning points and connecting points separately as mentioned above. Whether one intersection was directionally connected to another intersection was determined by checking whether the road users ever traveled through them consecutively without passing by any other intersections. GPS traces were segmented to track pieces for each road segment by the directly connected intersections. For each road segment, spatially abnormal tracks were detected by clustering the tracks using the spatial dissimilarity between pairwise tracks. Normal tracks were aligned using the proposed joint alignment methods, producing warped tracks. The warped tracks were averaged as the geometric representation of the road segment. Experimental results showed a higher geographical accuracy of road segments, compared to other algorithms.

Chapter 4 is another application of the trajectory analysis: room layout exploration. Inferring the objects in the environment, such as chairs, tables and walking areas, is beneficial to analyze the people's interaction with the environment. Instead of detecting the objects directly using their image features, such as color, shape and texture, we proposed to recognize the presence of chairs, tables and walking areas in a smart meeting room indirectly from people's trajectories. We first categorized people's instantaneous activities using their speed and height information, then fused the instantaneous activities into higher-level activities at each time period to improve their accuracy. We built the occupancy maps for the sitting space and walking space separately. The sitting activities were used to update the sitting map, and walking activities were used to update the walking map. Finally, we inferred the chairs from the sitting map, and the table from the walking map. We provided qualitative and quantitative results on two experiments. Experimental results showed that the table and chairs were successfully detected.

## 7.2 Future Research

In this thesis, we focused on the joint alignment of many trajectories. The first proposed method with a "stretch and then compress" strategy removes too many data points to form the *one-to-one* point correspondences, especially if

the trajectories are very different from each other. This may lead to too few points left in the warped trajectories, so the average of the warped trajectories may not be representative enough. The second proposed method, which uses a greedy procedure to find a good warp path through the local dissimilarity tensor, warps the trajectories to too many points. Therefore we need to design new alignment methods, or improve the current methods. For instance, we can calculate the dissimilarity of the corresponding points to each of these $2^M - 1$ cells, and choose the cell with a smaller dissimilarity and more successors, instead of the cell with the smallest dissimilarity. In this way, the warp path can travel through the N-dimensional local cost matrix faster, producing less redundant warped trajectories.

In our application of road network inference, the road map is generated from historical GPS traces. In the future, we will develop algorithms to update the current map and perform real time traffic analysis and collision prediction. Researchers have been using OpenStreetMap as ground truth map to measure the accuracy of inferred road segments. However, the positional error of GPS traces on OpenStreetMap data is as high as 10 to 20 meters. In the future, we plan to extract the ground truth map from high-resolution orthophotos using image processing techniques, and reevaluate all of the algorithms in the field of road network inference.

We apply LCSS algorithm to find the common sub-tracks between pairwise GPS traces, and detect the intersections from the connecting points which are the ends of the common sub-tracks. Although the warp path produced by LCSS optimally associates the points of the traces, the associated points may be not locally optimal. The poor local association may lead to larger local distance, and thus producing unmatched points skipped by the path-finding procedure. In this case, one common sub-tracks may be segmented into two or more sub-tracks, therefore some connecting points may be detected on the road segments, instead of at the intersections. In the future, local dissimilarity matrix will be used directly to detect the common sub-tracks, so as to remove these falsely detected connecting points.

In this thesis, we have focused on three applications of trajectory analysis: work cycle optimization, road network inference and room layout exploration. In the future, we will work on other applications, such as sport analysis, urban mobility analysis, etc. We will detect movement patterns of the players on the field from their trajectories, so as to reveal tactical patterns in the game and thus to provide technical support to the coaches. We will analyze people's moving patterns in the city, which is beneficial to build and improve the bike sharing systems and public transportation system. We will also analyze the impact of the weather and road construction on the moving patterns, providing better guidance services for road users.

Furthermore, the algorithms will be tested on other moving objects, except human beings and vehicles. For instance, multiple trajectory alignment can be used to analyze the location and dwelling time variation of the animals at the intermediate stations during their migration.

# Bibliography


[Ahmed 12] Mahmuda Ahmed & Carola Wenk. *Constructing street networks from GPS trajectories.* In Algorithms–ESA 2012, pages 60–71. Springer, 2012.

[Ahmed 14] Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser & Carola Wenk. *A comparison and evaluation of map construction algorithms using vehicle tracking data.* GeoInformatica, pages 1–32, 2014.

[Al Nuaimi 11] Klaithem Al Nuaimi & Hesham Kamel. *A survey of indoor positioning systems and algorithms.* In Innovations in Information Technology (IIT), 2011 International Conference on, pages 185–190. IEEE, 2011.

[Anjum 10] Nadeem Anjum & Andrea Cavallaro. *Trajectory clustering for scene context learning and outlier detection.* In Video Search and Mining, pages 33–51. Springer, 2010.

[Antonini 06] Gianluca Antonini & Jean Philippe Thiran. *Counting pedestrians in video sequences using trajectory clustering.* Circuits and Systems for Video Technology, IEEE Transactions on, vol. 16, no. 8, pages 1008–1020, 2006.

[Atev 06] Stefan Atev, Osama Masoud & Nikos Papanikolopoulos. *Learning Traffic Patterns at Intersections by Spectral Clustering of Motion Trajectories.* In Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pages 4851–4856. IEEE, 2006.

[Atev 10] Stefan Atev, Grant Miller & Nikolaos P Papanikolopoulos. *Clustering of vehicle trajectories.* Intelligent Transportation Systems, IEEE Transactions on, vol. 11, no. 3, pages 647–657, 2010.

[Basharat 08] Arslan Basharat, Alexei Gritai & Mubarak Shah. *Learning object motion patterns for anomaly detection and improved object detection.* In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008.

[Bashir 07] Faisal I Bashir, Ashfaq A Khokhar & Dan Schonfeld. *Real-time motion trajectory-based indexing and retrieval of video sequences.* Multimedia, IEEE Transactions on, vol. 9, no. 1, pages 58–65, 2007.

[Bauters 14] Karel Bauters, Hendrik Van Landeghem, Maarten Slembrouck, Dimitri Van Cauwelaert & Dirk Van Haerenborgh. *An automated work cycle classification and disturbance detection tool for assembly line work stations.* In International Conference on Informatics in Control, Automation and Robotics (ICINCO-2014), volume 2, 2014.

[Becker 06] Christian Becker & Armin Scholl. *A survey on problems and methods in generalized assembly line balancing.* European journal of operational research, vol. 168, no. 3, pages 694–715, 2006.

[Bellens 11] Rik Bellens, Sven Vlassenroot & Sidharta Gautama. *Collection and analyses of crowd travel behaviour data by using smartphones.* In 5th Transport Research Day, pages 536–544. BIVEC-GIBET, 2011.

[Biagioni 12a] James Biagioni & Jakob Eriksson. *Inferring Road Maps from Global Positioning System Traces.* Transportation Research Record: Journal of the Transportation Research Board, vol. 2291, no. 1, pages 61–71, 2012.

[Biagioni 12b] James Biagioni & Jakob Eriksson. *Map inference in the face of noise and disparity.* In Proceedings of the 20th International Conference on Advances in Geographic Information Systems, pages 79–88. ACM, 2012.

[Biliotti 05] David Biliotti, Gianluca Antonini & Jean Philippe Thiran. *Multi-layer hierarchical clustering of pedestrian trajectories for automatic counting of people in video sequences.* In Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on, volume 2, pages 50–57. IEEE, 2005.

[Bo 14] Nyan Bo Bo, Francis Deboeverie, Mohamed Eldib, Junzhi Guan, Xingzhe Xie, Jorge Niño, Dirk Van Haerenborgh, Maarten Slembrouck, Samuel Van de Velde, Heidi Steendam*et al.* *Human mobility monitoring in very low resolution visual sensor network.* Sensors, vol. 14, no. 11, pages 20800–20824, 2014.

[Boysen 07] Nils Boysen, Malte Fliedner & Armin Scholl. *A classification of assembly line balancing problems.* European

Journal of Operational Research, vol. 183, no. 2, pages 674–693, 2007.

[Calcagno 08] Philippe Calcagno, Jean-Paul Chilès, Gabriel Courrioux & Antonio Guillen. *Geological modelling from field data and geological knowledge: Part I. Modelling method coupling 3D potential-field interpolation and geological rules.* Physics of the Earth and Planetary Interiors, vol. 171, no. 1, pages 147–157, 2008.

[Cao 09] Lili Cao & John Krumm. *From GPS traces to a routable road map.* In Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pages 3–12. ACM, 2009.

[Castro 12] Pablo Samuel Castro, Daqing Zhang & Shijian Li. *Urban traffic modelling and prediction using large scale taxi GPS traces.* In Pervasive Computing, pages 57–72. Springer, 2012.

[Chang 10] Ming-Ching Chang, Nils Krahnstoever, Sernam Lim & Ting Yu. *Group level activity recognition in crowded environments across multiple cameras.* In Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on, pages 56–63. IEEE, 2010.

[Chapman 10] Jason W Chapman, Rebecca L Nesbit, Laura E Burgin, Don R Reynolds, Alan D Smith, Douglas R Middleton & Jane K Hill. *Flight orientation behaviors promote optimal migration trajectories in high-flying insects.* Science, vol. 327, no. 5966, pages 682–685, 2010.

[Chavoshi 15] Seyed Hossein Chavoshi, Bernard De Baets, Tijs Neutens, Guy De Tré & Nico Van de Weghe. *Exploring dance movement data using sequence alignment methods.* PLOS ONE, vol. 10, no. 7, page 25, 2015.

[Chen 08] Chen Chen & Yinhang Cheng. *Roads digital map generation with multi-track gps data.* In Education Technology and Training, 2008. and 2008 International Workshop on Geoscience and Remote Sensing. ETT and GRS 2008. International Workshop on, volume 1, pages 508–511. IEEE, 2008.

[Davics 06] JJ Davics, Alastair R Beresford & Andy Hopper. *Scalable, distributed, real-time map generation.* Pervasive Computing, IEEE, vol. 5, no. 4, pages 47–54, 2006.

[Di Stefano 99] Luigi Di Stefano & Andrea Bulgarelli. *A simple and efficient connected components labeling algorithm.* In Image Analysis and Processing, 1999. Proceedings. International Conference on, pages 322–327. IEEE, 1999.

[Edelkamp 03] Stefan Edelkamp & Stefan Schrödl. *Route planning and map inference with global positioning traces.* In Computer Science in Perspective, pages 128–151. Springer, 2003.

[Eldib 14] Mohamed Eldib, Bo Bo Nyan, Francis Deboeverie, Xingzhe Xie, Hamid Aghajan & Wilfried Philips. *Behavior analysis for aging-in-place using similarity heatmaps.* In 8th ACM/IEEE international conference on Distributed Smart Cameras, Proceedings, page 6. ACM/IEEE, 2014.

[Eldib 15] Mohamed Eldib, Francis Deboeverie, Bo Bo Nyan, Junzhi Guan, Xingzhe Xie, Dirk Van Haerenborgh, Hamid Aghajan & Wilfried Philips. *A low-cost visual sensor network for elderly care.* In FallRisk en Little Sister Slotsymposium, Abstracts, page 1, 2015.

[Elfes 89] A. Elfes. *Occupancy grids: A probabilistic framework for robot perception and navigation.* 1989.

[Elnekave* 06] M Elnekave* & I Gilad. *Rapid video-based analysis system for advanced work measurement.* International journal of production research, vol. 44, no. 2, pages 271–290, 2006.

[Fashandi 05] H Fashandi*et al.* *A new rotation invariant similarity measure for trajectories.* In Computational Intelligence in Robotics and Automation, 2005. CIRA 2005. Proceedings. 2005 IEEE International Symposium on, pages 631–634. IEEE, 2005.

[Fathi 10] Alireza Fathi & John Krumm. *Detecting road intersections from gps traces.* In Geographic Information Science, pages 56–69. Springer, 2010.

[Faude 12] Oliver Faude, Thorsten Koch & Tim Meyer. *Straight sprinting is the most frequent action in goal situations in professional football.* Journal of sports sciences, vol. 30, no. 7, pages 625–631, 2012.

[Fu 05] Zhouyu Fu, Weiming Hu & Tieniu Tan. *Similarity based vehicle trajectory clustering and anomaly detection.* In Image Processing, 2005. ICIP 2005. IEEE International Conference on, volume 2, pages II–602. IEEE, 2005.

[Gebus 09] Sébastien Gebus & Kauko Leiviskä. *Knowledge acquisition for decision support systems on an electronic assembly line.* Expert Systems with Applications, vol. 36, no. 1, pages 93–101, 2009.

[Greenfeld 02] Joshua S Greenfeld. *Matching GPS observations to locations on a digital map.* In Transportation Research Board 81st Annual Meeting, 2002.

[Gruenwedel 11a] S. Gruenwedel, V. Jelaca, P. Van Hese, R. Kleihorst & W. Philips. *PhD forum: Multi-view occupancy maps using a network of low resolution visual sensors.* In Distributed Smart Cameras (ICDSC), 2011 Fifth ACM/IEEE International Conference on, pages 1–2. IEEE, 2011.

[Gruenwedel 11b] S. Gruenwedel, P. Van Hese & W. Philips. *An edge-based approach for robust foreground detection.* Advances Concepts for Intelligent Vision Systems, pages 554–565, 2011.

[Gruenwedel 12] S. Gruenwedel, V. Jelaca, J.O. Niño-Castañeda, P. Van Hese, D. Van Cauwelaert, P. Veelaert & W. Philips. *Decentralized tracking of humans using a camera network.* In Proceedings of SPIE, volume 8301, page 83010D, 2012.

[Grünwedel 12] Sebastian Grünwedel, Xingzhe Xie, Wilfried Philips, Chih-Wei Chen & Hamid Aghajan. *A best view selection in meetings through attention analysis using a multi-camera network.* In 2012 Sixth international conference on distributed smart cameras (ICDSC), page 6. IEEE, 2012.

[Haklay 08] Mordechai Haklay & Patrick Weber. *Openstreetmap: User-generated street maps.* Pervasive Computing, IEEE, vol. 7, no. 4, pages 12–18, 2008.

[Haklay 10] Mordechai Haklay. *How good is volunteered geographical information? A comparative study of OpenStreetMap and Ordnance Survey datasets.* Environment and planning. B, Planning & design, vol. 37, no. 4, page 682, 2010.

[Hartmann 09] Bastian Hartmann, Christoph Schauer & Norbert Link. *Worker behavior interpretation for flexible production.* World Academy of Science, Engineering and Technology, pages 494–502, 2009.

[Herrera 10] Juan C Herrera, Daniel B Work, Ryan Herring, Xuegang Jeff Ban, Quinn Jacobson & Alexandre M Bayen. *Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment.* Transportation Research Part C: Emerging Technologies, vol. 18, no. 4, pages 568–583, 2010.

[Hirschberg 77] Daniel S Hirschberg. *Algorithms for the longest common subsequence problem.* Journal of the ACM (JACM), vol. 24, no. 4, pages 664–675, 1977.

[Hodgson 06] Sayre Hodgson, Thomas P Quinn, Ray Hilborn, Robert C Francis & Donald E Rogers. *Marine and freshwater climatic factors affecting interannual variation in the timing of return migration to fresh water of sockeye salmon (Oncorhynchus nerka).* Fisheries Oceanography, vol. 15, no. 1, pages 1–24, 2006.

[Ijaz 13] Faheem Ijaz, Hee Kwon Yang, Arbab Waheed Ahmad & Chankil Lee. *Indoor positioning: A review of indoor ultrasonic positioning systems.* In Advanced Communication Technology (ICACT), 2013 15th International Conference on, pages 1146–1150. IEEE, 2013.

[Jelaca 11] V. Jelaca, S. Griinwedel, J.O. Nino-Castaneda, P. Van Hese, D. Van Cauwelaert, P. Veelaert & W. Philips. *Demo: Real-time indoors people tracking in scalable camera networks.* In Distributed Smart Cameras (ICDSC), 2011 Fifth ACM/IEEE International Conference on, pages 1–2. IEEE, 2011.

[Jeong 11] Young-Seon Jeong, Myong K Jeong & Olufemi A Omitaomu. *Weighted dynamic time warping for time series classification.* Pattern Recognition, vol. 44, no. 9, pages 2231–2240, 2011.

[Junejo 07] Imran N Junejo & Hassan Foroosh. *Trajectory rectification and path modeling for video surveillance.* In Computer Vision (ICCV), 2007 IEEE 11th International Conference on, pages 1–7. IEEE, 2007.

[Junejo 08] Imran N Junejo & Hassan Foroosh. *Euclidean path modeling for video surveillance.* Image and Vision computing, vol. 26, no. 4, pages 512–528, 2008.

[Kalnis 05] Panos Kalnis, Nikos Mamoulis & Spiridon Bakiras. *On discovering moving clusters in spatio-temporal data.* In Advances in spatial and temporal databases, pages 364–381. Springer, 2005.

[Kanayama 89] Yutaka Kanayama & Bruce I Hartman. *Smooth local path planning for autonomous vehicles.* In Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on, pages 1265–1270. IEEE, 1989.

[Karagiorgou 12] Sophia Karagiorgou & Dieter Pfoser. *On vehicle tracking data-based road network generation.* In Proceedings of the 20th International Conference on Advances in Geographic Information Systems, pages 89–98. ACM, 2012.

[Keogh 05] Eamonn Keogh & Chotirat Ann Ratanamahatana. *Exact indexing of dynamic time warping.* Knowledge and information systems, vol. 7, no. 3, pages 358–386, 2005.

[Kjellström 11] H. Kjellström, J. Romero & D. Kragic. *Visual object-action recognition: Inferring object affordances from human demonstration.* Computer Vision and Image Understanding, vol. 115, no. 1, pages 81–90, 2011.

[Kukko 07] Antero Kukko, Constantin-Octavian Andrei, Veli-Matti Salminen, Harri Kaartinen, Yuwei Chen, Petri Rönnholm, Hannu Hyyppä, Juha Hyyppä, Ruizhi Chen, Henrik Haggrén*et al. Road environment mapping system of the Finnish Geodetic InstituteâĂŤFGI Roamer.* Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci, vol. 36, pages 241–247, 2007.

[Laboratory 11] BITS Networked Systems Laboratory. http://bits.cs.uic.edu/, 2011.

[Lee 13] Dongwook Lee & Minsoo Hahn. *Bicycle Safety Map System Based on Smartphone Aided Sensor Network.* Advanced Science and Technology Letters, vol. 42(Mobile and Wireless 2013), pages 38–43, 2013.

[Li 06] Xi Li, Weiming Hu & Wei Hu. *A coarse-to-fine strategy for vehicle motion trajectory clustering.* In Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, volume 1, pages 591–594. IEEE, 2006.

[Liu 07] Bing Liu. Web data mining. Springer, 2007.

[Lopez Aguirre 15] Angel Javier Lopez Aguirre, Daniel Ochoa & Sidharta Gautama. *Detecting changes of transportation-mode by using classification data.* In 18th International Conference on Information Fusion, Proceedings, page 6. International Society of Information Fusion (ISIF), 2015.

[McCormick 89] S Thomas McCormick, Michael L Pinedo, Scott Shenker & Barry Wolf. *Sequencing in an assembly line with blocking to minimize cycle time.* Operations Research, vol. 37, no. 6, pages 925–935, 1989.

[Mel 97] B.W. Mel. *SEEMORE: combining color, shape, and texture histogramming in a neurally inspired approach to visual object recognition.* Neural computation, vol. 9, no. 4, pages 777–804, 1997.

[Minnotte 93] Michael C Minnotte & David W Scott. *The mode tree: A tool for visualization of nonparametric density features.* Journal of Computational and Graphical Statistics, vol. 2, no. 1, pages 51–68, 1993.

[Morris 04] Scott Morris, Alan Morris & Kobus Barnard. *Digital trail libraries.* In Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries, pages 63–71. ACM, 2004.

[Morris 08] Brendan Tran Morris & Mohan Manubhai Trivedi. *A survey of vision-based trajectory learning and analysis for surveillance.* Circuits and Systems for Video Technology, IEEE Transactions on, vol. 18, no. 8, pages 1114–1127, 2008.

[Morris 09] Brendan Morris & Mohan Trivedi. *Learning trajectory patterns by clustering: Experimental studies and comparative evaluation.* In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 312–319. IEEE, 2009.

[Morris 11] Brendan Tran Morris & Mohan Manubhai Trivedi. *Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 33, no. 11, pages 2287–2301, 2011.

[Naftel 06] Andrew Naftel & Shehzad Khalid. *Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space.* Multimedia Systems, vol. 12, no. 3, pages 227–238, 2006.

[Nguyen 05] Nam Thanh Nguyen, Dinh Q Phung, Svetha Venkatesh & Hung Bui. *Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model.* In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 2, pages 955–960. IEEE, 2005.

[Niehoefer 09] Brian Niehoefer, Ralf Burda, Christian Wietfeld, Franziskus Bauer & Oliver Lueert. *GPS community map generation for enhanced routing methods based on trace-collection by mobile phones.* In Advances in Satellite and Space Communications, 2009. SPACOMM 2009. First International Conference on, pages 156–161. IEEE, 2009.

[Niennattrakul 09] Vit Niennattrakul & Chotirat Ann Ratanamahatana. *Shape averaging under time warping.* In Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on, volume 2, pages 626–629. IEEE, 2009.

[Patel 09] Dhaval Patel, Chidansh Bhatt, Wynne Hsu, Mong Li Lee & Mohan Kankanhalli. *Analyzing abnormal events from spatio-temporal trajectories.* In Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on, pages 616–621. IEEE, 2009.

[Pelekis 07] Nikos Pelekis, Ioannis Kopanakis, Gerasimos Marketos, Irene Ntoutsi, Gennady Andrienko & Yannis Theodoridis. *Similarity search in trajectory databases.* In Temporal Representation and Reasoning, 14th International Symposium on, pages 129–140. IEEE, 2007.

[Pelleg 00] Dan Pelleg, Andrew W Moore*et al. X-means: Extending K-means with Efficient Estimation of the Number of Clusters.* In ICML, pages 727–734, 2000.

[Petitjean 11] François Petitjean, Alain Ketterlin & Pierre Gançarski. *A global averaging method for dynamic time warping, with applications to clustering.* Pattern Recognition, vol. 44, no. 3, pages 678–693, 2011.

[Peursum 05a] P. Peursum, G. West & S. Venkatesh. *Combining image regions and human activity for indirect object recognition in indoor wide-angle views.* In Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on, volume 1, pages 82–89. IEEE, 2005.

[Peursum 05b] Patrick Peursum, Hung H Bui, Svetha Venkatesh & Geoff West. *Robust recognition and segmentation of human actions using HMMs with missing observations.* EURASIP Journal on Applied Signal Processing, vol. 2005, pages 2110–2126, 2005.

[Piciarelli 06] Claudio Piciarelli & Gian Luca Foresti. *On-line trajectory clustering for anomalous events detection.* Pattern Recognition Letters, vol. 27, no. 15, pages 1835–1842, 2006.

[Portal 14] Map Construction Portal. http://www.mapconstruction.org/, 2014.

[Sakoe 78] Hiroaki Sakoe & Seibi Chiba. *Dynamic programming algorithm optimization for spoken word recognition.* Acoustics, Speech and Signal Processing, IEEE Transactions on, vol. 26, no. 1, pages 43–49, 1978.

[Sato 06] Junji Sato, Tomokazu Takahashi, Ichiro Ide & Hiroshi Murase. *Change detection in streetscapes from GPS coordinated omni-directional image sequences.* In Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, volume 4, pages 935–938. IEEE, 2006.

[Schroedl 04] Stefan Schroedl, Kiri Wagstaff, Seth Rogers, Pat Langley & Christopher Wilson. *Mining GPS traces for map refinement.* Data mining and knowledge Discovery, vol. 9, no. 1, pages 59–87, 2004.

[Schultes 08] Dominik Schultes. *Route Planning in Road Networks.* In Ausgezeichnete Informatikdissertationen, pages 271–280, 2008.

[Semanjski 15] Ivana Semanjski & Sidharta Gautama. *Smart city mobility application: gradient boosting trees for mobility prediction and analysis based on crowdsourced data.* SENSORS, vol. 15, no. 7, pages 15974–15987, 2015.

[Sester 12] Monika Sester, Udo Feuerhake, Colin Kuntzsch & Lijuan Zhang. *Revealing underlying structure and behaviour from movement data.* KI-Künstliche Intelligenz, vol. 26, no. 3, pages 223–231, 2012.

[Shamoun-Baranes 10] Judy Shamoun-Baranes, Willem Bouten & E Emiel van Loon. *Integrating meteorology into research on migration.* Integrative and Comparative Biology, page icq011, 2010.

[Shi 09] Wenhuan Shi, Shuhan Shen & Yuncai Liu. *Automatic generation of road network map from massive GPS, vehicle trajectories.* In Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on, pages 1–6. IEEE, 2009.

[Slembrouck 14] Maarten Slembrouck, Dimitri Van Cauwelaert, David Van Hamme, Dirk Van Haerenborgh, Peter Van Hese, Peter Veelaert & Wilfried Philips. *Self-learning voxel-based multi-camera occlusion maps for 3D reconstruction.* In 9th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP-2014). SCITEPRESS, 2014.

[Smith 81] Temple F Smith & Michael S Waterman. *Identification of common molecular subsequences.* Journal of molecular biology, vol. 147, no. 1, pages 195–197, 1981.

[Stoessel 08] Christian Stoessel, Mathey Wiesbeck, Sonja Stork, Michael F Zaeh & Anna Schuboe. *Towards optimal worker assistance: investigating cognitive processes in manual assembly.* In Manufacturing Systems and Technologies for the New Frontier, pages 245–250. Springer, 2008.

[Stork 08] Sonja Stork, Christian Stößel & Anna Schubö. The influence of instruction mode on reaching movements during manual assembly. Springer, 2008.

[Suzuki 07] Naohiko Suzuki, Kosuke Hirasawa, Kenichi Tanaka, Yoshinori Kobayashi, Yoichi Sato & Yozo Fujino. *Learning motion patterns and anomaly detection by human trajectory analysis.* In Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on, pages 498–503. IEEE, 2007.

[Syberfeldt 09] Anna Syberfeldt & Lars Persson. *Using Heuristic Search for Initiating the Genetic Population in Simulation-Based Optimization of Vehicle Routing Problems.* In Proceedings of Industrial Simulation Conference. EUROSIS-ETI, 2009.

[Torralba 10] A. Torralba, K.P. Murphy & WT Freeman. *Using the forest to see the trees: exploiting context for visual object detection and localization.* Communications of the ACM, vol. 53, no. 3, pages 107–114, 2010.

[UETA 00] MUTSUYUKI UETA, FUMIO SATO, HAJIME NAKAGAWA & NAGAHISA MITA. *Migration routes and differences of migration schedule between adult and young Steller's Sea Eagles Haliaeetus pelagicus.* Ibis, vol. 142, no. 1, pages 35–39, 2000.

[Veloso 05] M. Veloso, F. Von Hundelshausen & P.E. Rybski. *Learning visual object definitions by observing human activities*. In Humanoid Robots, 2005 5th IEEE-RAS International Conference on, pages 148–153. IEEE, 2005.

[Veloso 06] M.M. Veloso, P.E. Rybski & F. Von Hundelshausen. *Focus: a generalized method for object discovery for robots that observe and interact with humans*. In Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, pages 102–109. ACM, 2006.

[Vlachos 02] Michail Vlachos, George Kollios & Dimitrios Gunopulos. *Discovering similar multidimensional trajectories*. In Data Engineering, 2002. Proceedings. 18th International Conference on, pages 673–684. IEEE, 2002.

[Vlahogianni 05] Eleni I Vlahogianni, Matthew G Karlaftis & John C Golias. *Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach*. Transportation Research Part C: Emerging Technologies, vol. 13, no. 3, pages 211–234, 2005.

[Wang 06] X. Wang, K. Tieu & E. Grimson. *Learning semantic scene models by trajectory analysis*. Computer Vision–ECCV 2006, pages 110–123, 2006.

[Wang 13a] Xiaogang Wang. *Intelligent multi-camera video surveillance: A review*. Pattern recognition letters, vol. 34, no. 1, pages 3–19, 2013.

[Wang 13b] Zuchao Wang, Min Lu, Xiaoru Yuan, Junping Zhang & Huub Van De Wetering. *Visual traffic jam analysis based on trajectory data*. Visualization and Computer Graphics, IEEE Transactions on, vol. 19, no. 12, pages 2159–2168, 2013.

[Wang 15] Jing Wang, Xiaoping Rui, Xianfeng Song, Xiangshuang Tan, Chaoliang Wang & Venkatesh Raghavan. *A novel approach for generating routable road maps from vehicle GPS traces*. International Journal of Geographical Information Science, vol. 29, no. 1, pages 69–91, 2015.

[Worrall 07] Stewart Worrall & Eduardo Nebot. *Automated process for generating digitised maps through GPS data compression*. In Australasian Conference on Robotics and Automation, 2007.

[Wu 09] C. Wu & H. Aghajan. *Using context with statistical relational models: object recognition from observing user activity in home environment.* In Proceedings of the Workshop on Use of Context in Vision Processing, page 5. ACM, 2009.

[Wu 11] C. Wu & H. Aghajan. *User-centric environment discovery with camera networks in smart homes.* Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, vol. 41, no. 2, pages 375–383, 2011.

[Wu 13] Junwei Wu, Yunlong Zhu, Tao Ku & Liang Wang. *Detecting Road Intersections from Coarse-gained GPS Traces Based on Clustering.* Journal of Computers, vol. 8, no. 11, pages 2959–2965, 2013.

[Xie 12] Xingzhe Xie, Sebastian Gruenwedel, Vedran Jelaca, Jorge Oswaldo Nino Castaneda, Dirk Van Haerenborgh, Dimitri Van Cauwelaert, Peter Van Hese, Peter Veelaert, Wilfried Philips & Hamid Aghajan. *Learning about objects in the meeting rooms from people trajectories.* In Distributed Smart Cameras (ICDSC), 2012 Sixth International Conference on, pages 1–6. IEEE, 2012.

[Xie 13a] Xingzhe Xie, Kevin Wong, Hamid Aghajan & Wilfried Philips. *Analyzing cyclists' behaviors and exploring the environments from cycling tracks.* In Cycling Data Challenge (CDC) workshop (AGILE-2013), 2013.

[Xie 13b] Xingzhe Xie, Kevin Wong, Hamid Aghajan & Wilfried Philips. *Smart route recommendations based on historical GPS trajectories and weather information.* In Mobile Ghent 2013, 2013.

[Xie 14a] Xingzhe Xie, Jonas De Vylder, Dimitri Van Cauwelaert, Peter Veelaert, Wilfried Philips & Hamid Aghajan. *Average Track Estimation of Moving Objects Using RANSAC and DTW.* In Proceedings of the 8th ACM/IEEE International Conference on Distributed Smart Cameras, pages 215–220. ACM/IEEE, 2014.

[Xie 14b] Xingzhe Xie, Francis Deboeverie, Mohamed Eldib, Wilfried Philips & Hamid Aghajan. *PhD Forum: Analyzing behaviors patterns of the elderly from low-precision trajectories.* In Proceedings of the International Conference on Distributed Smart Cameras, page 47. ACM, 2014.

[Xie 14c] Xingzhe Xie, Wilfried Philips, Peter Veelaert & Hamid Aghajan. *Road network inference from GPS traces using DTW algorithm.* In Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on, pages 906–911. IEEE, 2014.

[Xie 15a] Xingzhe Xie, Dimitri Van Cauwelaert, Maarten Slembrouck, Karel Bauters, Johannes Cottyn, Dirk Van Haerenborgh, Hamid Aghajan, Peter Veelaert & Wilfried Philips. *Abnormal work cycle detection based on dissimilarity measurement of trajectories.* In Proceedings of the 9th International Conference on Distributed Smart Camera, pages 68–73. ACM, 2015.

[Xie 15b] Xingzhe Xie, Kevin Wong, Hamid Aghajan, Peter Veelaert & Wilfried Philips. *Inferring Directed Road Networks from GPS Traces by Track Alignment.* International Journal of Geo-Information, vol. 4, pages 2446–2471, 2015.

[Xie 16a] Xingzhe Xie, Kevin Bing-Yung Wong, Hamid Aghajan, Peter Veelaert & Wilfried Philips. *Road network inference through multiple track alignment.* Transportation Research Part C: Emerging Technologies, 2016. (under review).

[Xie 16b] Xingzhe Xie, Wenzhi Liao, Hamid Aghajan, Peter Veelaert & Wilfried Philips. *Detecting Road Intersections from GPS Traces using Longest Common Subsequence Algorithm.* IEEE Transactions on Intelligent Transportation Systems, 2016. (under review).

[Xie 16c] Xingzhe Xie, Wenzhi Liao, Hamid Aghajan, Peter Veelaert & Wilfried Philips. *A NOVEL APPROACH FOR DETECTING INTERSECTIONS FROM GPS TRACES.* In Proceedings of the 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2016). IEEE, 2016.

[Xie 16d] Xingzhe Xie, Dimitri Van Cauwelaert, Maarten Slembrouck, Dirk Van Haerenborgh, Mohamed Eldib, Karel Bauters, Johannes Cottyn, Hamid Aghajan, Peter Veelaert & Wilfried Philips. *Analysis of work cycle in assembly lines through aligning the worker's tracks.* Journal of Ambient Intelligence and Humanized Computing, 2016. (under review).

[Yang 12] Yuanfeng Yang, Zhiming Cui, Jian Wu, Guangming Zhang & Xuefeng Xian. *Trajectory analysis using spectral clustering and sequence pattern mining.* Journal of Computational Information Systems, vol. 8, no. 6, pages 2637–2645, 2012.

[You 14] Quanzeng You & John Krumm. *Transit tomography using probabilistic time geography: planning routes without a road map.* Journal of Location Based Services, vol. 8, no. 4, 2014.

[Zhang 06] Zhang Zhang, Kaiqi Huang & Tieniu Tan. *Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes.* In Pattern Recognition, 2006. ICPR 2006. 18th International Conference on, volume 3, pages 1135–1138. IEEE, 2006.

[Zheng 08a] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie & Wei-Ying Ma. *Understanding mobility based on GPS data.* In Proceedings of the 10th international conference on Ubiquitous computing, pages 312–321. ACM, 2008.

[Zheng 08b] Yu Zheng, Like Liu, Longhao Wang & Xing Xie. *Learning transportation mode from raw gps data for geographic applications on the web.* In Proceedings of the 17th international conference on World Wide Web, pages 247–256. ACM, 2008.

[Zheng 10] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie & Wei-Ying Ma. *Understanding transportation modes based on GPS data for web applications.* ACM Transactions on the Web (TWEB), vol. 4, no. 1, page 1, 2010.

[Zhou 07] Yue Zhou, Shuicheng Yan & Thomas S Huang. *Detecting anomaly in videos from trajectory similarity analysis.* In Multimedia and Expo, 2007 IEEE International Conference on, pages 1087–1090. IEEE, 2007.