Existe uma série de estudos que analisam trajetórias de objetos móveis. Dentre eles a identificação de similaridade de forma entre trajetórias é uma das áreas onde este tipo pesquisa se desenvolve. A similaridade de trajetórias pode indicar comportamentos comuns dentro de grupos de indivíduos e podem ser úteis em várias áreas de aplicação. No entanto, como medir a similaridade de forma de trajetórias quando estas estão em direções diferentes e distantes uma das outras? Tentando resolver este problema, este trabalho propõem um método para identificação de similaridade de forma entre trajetórias aplicando correlação estatística em vetores de deflexão angulares dos segmentos das trajetórias analisadas.

Orientador: Fabiano Baldo

Coorientador: Fernando José Braz

Joinville, 2016

ANO
2016

ALEXANDRE ALTAIR DE MELO | UM MÉTODO PARA CÁLCULO DE SIMILARIDADE DE FORMA ENTRE TRAJETÓRIAS DE OBJETOS MÓVEIS BASEADO NA CORRELAÇÃO ESTATÍSTICA DE VETORES DE DEFLEXÕES ANGULARES

UDESC

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
CURSO DE MESTRADO EM COMPUTAÇÃO APLICADA

DISSERTAÇÃO DE MESTRADO

UM MÉTODO PARA CÁLCULO DE SIMILARIDADE DE FORMA ENTRE TRAJETÓRIAS DE OBJETOS MÓVEIS BASEADO NA CORRELAÇÃO ESTATÍSTICA DE VETORES DE DEFLEXÕES ANGULARES

ALEXANDRE ALTAIR DE MELO

JOINVILLE, 2016

ALEXANDRE ALTAIR DE MELO

UM MÉTODO PARA CÁLCULO DE SIMILARIDADE DE FORMA ENTRE TRAJETÓRIAS DE OBJETOS MÓVEIS BASEADO NA CORRELAÇÃO ESTATÍSTICA DE VETORES DE DEFLEXÕES ANGULARES

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada da Universidade do Estado de Santa Catarina, como requisito parcial para obtenção do grau de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Fabiano Baldo
Coorientador: Prof. Dr. Fernando José Braz

JOINVILLE, SC
2016

FICHA CATALOGRÁFICA

M527u Melo, Alexandre Altair de

Um método para cálculo de similaridade de forma entre trajetórias de objetos móveis baseado na correlação estatística de vetores de deflexões angulares / Alexandre Altair de Melo. – 2016.

141 p. : il. ; 21 cm

Orientador: Fabiano Baldo

Coorientador: Fernando José Braz

Bibliografia: p. 115-122

Dissertação (mestrado) – Universidade do Estado Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2016.

1. Computação. 2. Trajetórias. 3. Estatística. 4. Ângulos. I. Baldo, Fabiano. II. Braz, Fernando José. III. Universidade do Estado Santa Catarina Programa de Pós-Graduação em Computação Aplicada. IV. Título.

CDD 004 - 23.ed.

ALEXANDRE ALTAIR DE MELO

UM MÉTODO PARA CÁLCULO DE SIMILARIDADE DE FORMA ENTRE TRAJETÓRIAS DE OBJETOS MÓVEIS BASEADO NA CORRELAÇÃO ESTATÍSTICA DE VETORES DE DEFLEXÕES ANGULARES

Dissertação apresentada ao Curso de Mestrado Acadêmico Computação Aplicada como requisito parcial para obtenção do título de Mestre em Computação Aplicada na área de concentração "Ciência da Computação".

Banca Examinadora

Orientador:

Prof. Dr. Fabiano Baldo
CCT/UDESC

Coorientador:

Prof. Dr. Fernando José Braz
IFC

Membros

Profa. Dra. Elisa Henning
CCT/UDESC

por video conferência

Profa. Dra. Vania Bogorny
INE/UFSC

Joinville,SC, 12 de agosto de 2016.

Este trabalho é dedicado a minha família.

Agradecimentos

Agradeço a Deus, por guiar os passos até aqui.

Ao meu orientador, professor Fabiano Baldo, que muito apoiou, cobrou e colaborou na construção deste trabalho. Igual agradecimento faço ao meu coorientador, professor Fernando José Braz, pelas contribuições e discussões sobre esta pesquisa.

A todos os professores do programa que contribuíram para esta jornada, durante a estada de aprendizado no CCT/UDESC. Em especial à professora Elisa Henning, pelos valiosos comentários sobre a abordagem estatística.

Aos novos amigos Glaucio Scheibel, Mauro Hinz, Daniel Maniglia, Ademar Alves Júnior e Sandro R. L. de Menezes por inúmeras horas de trabalho e divertidas discussões. E a todos os outros colegas do programa, com os quais tive contato durante os trabalhos das disciplinas.

Por último, mas não menos importante, e sem a qual não chegaria até aqui, a minha querida Helena Ignowski.

"Para saber que nós sabemos
o que nós sabemos,
e para saber que nós não sabemos
o que nós não sabemos,
isso é o conhecimento verdadeiro."
(Nicolau Copérnico)

RESUMO

MELO, Alexandre Altair de. **Um Método para Cálculo de Similaridade de Forma entre Trajetórias de Objetos Móveis Baseado na Correlação Estatística de Vetores de Deflexões Angulares**. 2016. 141 p. Dissertação (Mestrado em Computação Aplicada - Área: Engenharia de Software. Universidade do Estado de Santa Catarina. Programa de Pós-Graduação em Computação Aplicada, Joinville, Santa Catarina, Brasil, 2016.

Os dispositivos eletrônicos móveis vêm ganhando cada dia mais espaço, principalmente pela quantidade de funcionalidades que eles oferecem. Muitas dessas funcionalidades utilizam sensores, sendo que um dos que merecem destaque é o de localização. Esse sensor permite a coleta de informações que se ordenadas cronologicamente representam a trajetória de movimentação do objeto que o carrega. Existe uma série de estudos que analisam trajetórias de objetos móveis. Dentre eles a identificação de similaridade de forma entre trajetórias é uma das áreas onde esse tipo pesquisa se desenvolve. A similaridade de trajetórias pode indicar comportamentos comuns dentro de grupos de indivíduos e pode ser útil em várias áreas de aplicação. No entanto, como medir a similaridade de forma de trajetórias quando estas estão em direções diferentes e distantes umas das outras? Tentando resolver esse problema, este trabalho propõe um método para identificação de similaridade de forma entre trajetórias aplicando correlação estatística em vetores de deflexão angulares dos segmentos das trajetórias analisadas. Os resultados indicam que o método consegue identificar satisfatoriamente a similaridade de forma entre trajetórias, com valores de correlação de Pearson $(r) \geq 0.70$. O método também propõe uma etapa de pré-processamento de dados e compactação de trajetórias para lidar com a questão de trajetórias com tamanhos diferentes.

Palavras-chaves: Trajetórias. Estatística. Ângulos.

ABSTRACT

MELO, Alexandre Altair de. **A Method for Calculating Shape Similarity among Trajectory of Moving Object Based on Statistical Correlation of Angular Deflection Vectors**. 2016. 141 p. Master Thesis (Master Degree in Applied Computing - Area: Software Engineering. Santa Catarina State University. Graduate Program in Applied Computing, Joinville, Santa Catarina, Brazil, 2016.

In recent years, mobile electronic devices are gaining more attention due to the diversity of functionalities that they offer. Many of these functionalities use embeded sensors where the localization one brings attention. This sensor allows to collect location information that when chronologically ordered represents the trajectory of the moving object that carries it. There is a considerable number of researches that analyse objects' trajectories. Among them, the identification of trajectories with similar shape is one where researches are currently being developed. The similarity of trajectories shape may indicate common behaviors inside groups of individuals that can be useful in various application areas. However, how to measure the trajectories' shape similarity even when they are in different directions and far from each other? Trying to solve this problem, this paper proposes a method to identify shape similarity between trajectories applying statistical correlation over their vectors of angular deflections. The results indicate that the method can suitably identify shape similarity among trajectories, with Pearson (r) correlation values ≥ 0.70. The method also propose a pre-processing step and data compression trajectories to deal with the issue of trajectories with different sizes.

Key-words: Trajectory. Statistics. Angles.

Lista de ilustrações

Lista de quadros

Lista de tabelas

Lista de algoritmos

Lista de abreviaturas e siglas

DBRMS	Sistema Gerenciador de Banco de Dados Relacional (*DataBase Relational Management System*)
DDL	Linguagem de Definição de Dados (*Data Definition Language*)
DML	Linguagem de Manipulação de Dados (*Data Manipulation Language*)
DTW	Tempo Flexionado Dinamicamente (*Dynamic Time Warping)*)
ED	Distância Editada (*Edit Distance*)
EDR	Distância Editada em Sequências Reais (*Edit Distance on Real Sequences*)
EDS	Distância Editada em Segmentos (*Edit Distance on Segment*)
ETL	Extração, Transformação e Carga (*Extract, Transform and Load*)
GIS	Sistema de Informação Geográfica (*Geographic Information System*)

GMT	Hora Média de Greenwich (*Greenwich Mean Time*)
GPS	Sistemas de Posicionamento Global (*Global Positioning System*)
LCSS	Mais Longo Subsequente em Comum (*Longest Common Subsequence*)
ORM	Mapeamento Objeto-Relacional (*Object-Relational Mapping*)
QSR	Representações Espaciais Qualitativas (*Qualitative Spatial Representations*)
SQL	Linguagem Estruturada de Consulta (*Structured Query Language*)
UML	Linguagem de Modelagem Unificada (*Unified Modeling Language*)

Lista de símbolos

AZ	Azimute.
c	Coordenada.
DF	Deflexão.
p	Ponto.
r	Coeficiente de Pearson.
ρ	Coeficiente de Spearman.
S	Segmento de trajetória.
T	Trajetória.
VDF	Vetor de Deflexão.

Sumário

1

INTRODUÇÃO

Os dispositivos eletrônicos móveis vêm ganhando cada dia mais espaço no cotidiano das pessoas, principalmente pela quantidade de funcionalidades que eles oferecem. Muitas das funcionalidades por eles oferecidas se utilizam de sensores embutidos no próprio equipamento para oferecer uma experiência adequada ao usuário. Como exemplo de sensores amplamente utilizados pelos aplicativos se destaca o receptor de posicionamento global GPS (*Global Positioning System*) (GIANNOTTI et al., 2011). Esse sensor permite a coleta de informações de localização que se ordenadas cronologicamente representam a trajetória de movimentação do objeto que o carrega (MEHTA; MACHIRAJU; PARTHASARATHY, 2006). Trajetórias de objetos móveis são fonte de estudos das mais diversas áreas do conhecimento, pois fornecem uma valiosa fonte de informação para análise e compreensão do comportamento por meio da identificação de padrões nos deslocamentos de objetos de forma individual ou em grupo. Áreas como biomonitoramento, logística, sistemas de navegação, tráfego de automóveis e pedestres são exemplos de aplicações da análise de trajetórias.

De forma geral, uma trajetória pode ser representada por um identificador (T_{id}) e um conjunto de pontos formado por valores x, y e t, onde x e y são coordenadas geográficas, e t o instante

de tempo em que as coordenadas foram coletadas (ANDRIENKO et al., 2010; FRENTZOS; THEODORIDIS; PAPADOPOULOS, 2009; BRAZ; BOGORNY, 2012). Adicionalmente, uma trajetória e seus pontos podem ser enriquecidos com informações como: velocidade, direção, aceleração, pressão etc. (PARENT et al., 2013; BOGORNY et al., 2014). A fim de se obter essas informações é necessário combinar o uso do GPS com outros sensores como, por exemplo, o acelerômetro, o magnetrômetro, o barômetro, o giroscópio, entre outros. Com isso é possível analisar trajetórias também do ponto de vista semântico (FURTADO et al., 2015).

Os dados gerados por trajetórias de objetos móveis são uma vasta fonte para análises onde os resultados podem trazer contribuições para o cotidiano das pessoas. Existe um número crescente de pesquisas que analisam trajetórias de objetos móveis a fim de encontrar respostas a fenômenos relacionados aos deslocamentos de indivíduos ou grupos. Entre essas pesquisas, a identificação de similaridade entre trajetórias é uma das áreas de estudo que vem sendo explorada nos últimos anos.

Vários trabalhos mencionados na literatura apresentam abordagens para identificar similaridade entre trajetórias. Vlachos et al. (2002), por exemplo, apresentam um método que tem por objetivo identificar a maior sequência de pontos similares entre duas trajetórias. Já Chen et al. (2005) propõem a função EDR que calcula a similaridade de forma entre as trajetórias com base na distância entre elas. O método utiliza distância euclidiana para identificar similaridade entre trajetórias próximas, e utiliza a função ED para verificar quantas transformações são necessárias para achar a similaridade.

Cada proposta apresentada para lidar com o problema de identificação de similaridade define um conjunto particular de métricas a serem aplicadas sobre as trajetórias analisadas. Entretanto, segundo Pelekis et al. (2012), para serem consideradas similares as trajetórias precisam satisfazer pelo menos um dos seguintes aspectos:(i) Estarem totalmente ou parcialmente sobrepostas no espaço; (ii) Terem formas similares em lugares diferentes; (iii) Terem a mesma posição inicial e ou final; (iv) Te-

rem comportamento de movimento totalmente ou parcialmente sincronizado; (v) Ou estiverem totalmente separadas no tempo, mas similares em algum comportamento dinâmico como velocidade, aceleração etc.

Como mencionado, existe um relevante número de trabalhos que apresentam contribuições relacionadas à identificação de similaridade quanto à forma em trajetórias de objetos móveis, entretanto, não foram observados trabalhos que apresentassem soluções para o problema de identificação de similaridade de forma entre trajetórias que estão longe uma das outras ou que tenham direção de movimento diferente. Por exemplo, trajetórias que estão em diferentes cidades ou mesmo em diferentes países, e que se movam de oeste para leste ou de sul para norte.

Portanto, este trabalho aborda o seguinte problema: Como mensurar a similaridade quanto à forma de trajetórias que estão longe umas das outras e se deslocam em direções diferentes? A fim de contribuir para a solução deste problema, este trabalho assume que a similaridade de forma entre trajetórias pode ser calculada por meio da aplicação de correlação estatística sobre vetores de deflexões angulares extraídas dos segmentos das trajetórias analisadas. Para tanto o trabalho também assume que os dados precisam estar devidamente tratados, sem a presença de ruídos que possam interferir na análise das informações. Também assume-se a necessidade de normalização quanto à questão de distância para evitar problemas de comparação com escalas muito amplas, por exemplo, comparação de trajetórias com distância de um metro com outra de cem metros.

A medição de ângulos de geometrias é estudada dentro da disciplina de goniologia. Essa disciplina pertence à área da topografia e tem como objetivo propor processos e instrumentos para medir e avaliar ângulos geométricos (KAVANAGH; BIRD, 2000). Entre as medidas apresentadas na goniologia, o azimute (AZ) e a deflexão (DF) são algumas dessas medidas utilizadas para calcular a orientação de segmentos de geometrias dentro de um sistema de coordenadas. Assim, elas podem ser usadas para calcular o ângulo de movimento de cada um dos segmentos de

uma trajetória georreferenciada.

Considerando que uma trajetória é formada por n pontos, após o tratamento para eliminação de ruídos, e da normalização da escala de distância presentes nesses pontos, o cálculo da deflexão de cada um dos seus segmentos terá como resultado um vetor $n-1$ valores de deflexões angulares. Usando essa abordagem, o problema de identificação de similaridade de forma se reduz ao cálculo de similaridade entre os vetores de deflexão das trajetórias comparadas. Neste trabalho, assume-se que este problema pode ser enquadrado na análise de correlação estatística. Esta análise tem como objetivo medir o coeficiente de correlação ou associação entre os dois vetores de deflexões angulares das trajetórias comparadas (CHEN; POPOVICH, 2002). Dessa forma, a abordagem de identificação de similaridade de forma entre trajetórias se torna independente da orientação e distância entre elas.

A identificação de similaridade de forma entre trajetórias é uma área que tem forte impacto na descoberta de padrões e na análise semântica de trajetórias. Portanto, seus benefícios podem ser amplamente aplicados a soluções que efetuem comparações individuais ou coletivas de trajetórias. Como este trabalho tem por premissa a hipótese de que é possível identificar a similaridade de forma entre duas trajetórias calculando a correlação estatística entre os seus vetores de deflexão ($\overrightarrow{DF}$), espera-se com isso alcançar um conjunto de trajetórias que até então não podiam ser comparadas e terem sua similaridade de forma avaliada.

1.1 Objetivos

Nas subseções a seguir são apresentados o objetivo geral e os objetivos específicos deste trabalho.

1.1.1 Geral

Este trabalho tem por objetivo propor um método para medir a similaridade de forma de trajetórias de objetos móveis,

utilizando para tal uma abordagem baseada na aplicação de correlação estatística entre seus vetores de deflexão angulares.

1.1.2 Específicos

Com base no objetivo principal, tem-se os seguintes objetivos específicos:

- Desenvolver um método para calcular a similaridade de forma entre trajetórias;
- Implementar a etapa de pré-processamento dos dados visando melhorar a qualidade das informações analisadas;
- Tratar a questão de trajetórias com tamanhos diferentes via compactação de dados;
- Implementar o método proposto, visando demonstrar seu funcionamento e analisar suas capacidades;
- Aplicar o método em cenários variados de diferentes fontes de dados;
- Verificar a aplicabilidade de correlação estatística à proposta de pesquisa apresentada.

1.2 Resultados Esperados

Como resultado, espera-se obter um método que consiga identificar a similaridade de trajetórias quanto à forma. Além disso, o método deve suportar a identificação de similaridade independente da direção, distância e tamanhos diferentes quanto às trajetórias analisadas.

1.3 Metodologia

A seção 1.3.1 apresenta a caracterização metodológica deste trabalho. Já a seção 1.3.2 apresenta a metodologia de pesquisa utilizada.

1.3.1 Caracterização metodológica

A caracterização metodológica de um trabalho científico é importante para enquadrar seu objetivo e forma de abordagem. Além disso, ela auxilia o pesquisador na identificação de instrumentos científicos adequados e que o ajudem a alcançar o resultado pretendido. Nesse sentido, a seguir é apresentada a caracterização metodológica do presente trabalho.

Em relação ao tipo de ciência (WAZLAWICK, 2010), o presente trabalho pode ser classificado como ***ciência exata e hard***. Ele é classificado como *ciência exata*, pois ao se aplicar o método sob o mesmo conjunto de trajetórias o resultado não deve apresentar variações. Isso ocorre porque o método proposto não utiliza técnicas estocásticas ou qualquer outro tipo de abordagem não determinística. Ele também pode ser considerado por ciência *hard*. Por ciência *hard* esta pode ser entendida como a abordagem científica que utiliza fortemente a lógica e a matemática como ferramentas de construção teórica (WAZLAWICK, 2010). Normalmente entende-se a computação como uma ciência *hard* (WAZLAWICK, 2014).

Em relação ao paradigma científico dominante (EDEN, 2007), o presente trabalho pode ser classificado predominantemente dentro do ***paradigma tecnocrático***, pois a eficiência do método proposto só é conhecida após a realização de testes.

Em relação ao objetivo geral da pesquisa (GIL, 2010), o presente trabalho pode ser classificado como ***pesquisa explicativa***, pois visa identificar os fatores que determinam a ocorrência dos fenômenos, ou seja, verificar se o método proposto é capaz de identificar similaridade de forma entre trajetórias, ao contrário da pesquisa exploratória, que visa proporcionar maior familiaridade com o problema, e da pesquisa descritiva, que visa descrever um fenômeno ou estabelecer relações entre variáveis.

Em relação ao raciocínio lógico (LAKATOS; MARCONI, 2010), o presente trabalho utiliza o ***método dedutivo***, pois a partir de conhecimentos gerais se inferem conhecimentos específicos. Isso significa que o desenvolvimento do método é baseado nos testes conduzidos em cenários considerados complexos, e deduz-

se que resultados com a mesma qualidade podem ser obtidos em cenários mais simples.

Finalmente, em relação ao nível de maturidade da pesquisa (WAZLAWICK, 2014), o presente trabalho pode ser classificado no nível ***apresentação de algo diferente***, pois ele propõe um novo método e justifica os resultados encontrados e as diferenças em relação a outros métodos existentes usando argumentação teórica e pesquisas bibliográficas.

1.3.2 Metodologia de pesquisa

A metodologia de pesquisa deste trabalho se apoia nos seguintes procedimentos técnicos: pesquisa bibliográfica e pesquisa exploratória. O primeiro implica no estudo de artigos, teses, livros e outras publicações disponibilizadas por editoras e indexadas em bases de dados qualificadas. O segundo, conforme Wazlawick (2014), indica que o trabalho científico deve utilizar rigorosas técnicas de amostragem e testes de hipóteses para que os seus resultados sejam estatisticamente aceitáveis e generalizáveis. Portanto, para que o avanço ocorra na pesquisa é necessário que o pesquisador planeje e execute determinados passos.

Sendo assim, a metodologia de pesquisa para o trabalho em questão inicia com uma revisão bibliográfica de trabalhos relacionados e conceitos das áreas de trajetórias de objetos móveis, goniologia e estatística. Paralelamente, é construído um suporte computacional para implementar o método proposto. A avaliação do método é feita buscando-se verificar em quais cenários ele se mostra mais promissor ou menos para identificação de similaridade. Para esta avaliação, buscou-se *datasets* públicos de trajetórias que pudessem ser utilizados nesta tarefa. Foram selecionados preferencialmente *datasets* usados em pesquisas científicas, a fim de validar o método em cenários com dados usados pela comunidade acadêmica.

Para o desenvolvimento desta solução foi necessário:

1. Projetar o esquema conceitual de objetos para os dados das trajetórias, com base nos conceitos descritos na literatura;

2. Criar o esquema no banco de dados referente ao modelo conceitual descrito no passo 1;

3. Importar os dados dos *datasets* para a base de dados criada no passo 2;

4. Realizar a limpeza dos dados importados no passo 3;

5. Implementar algoritmo de compactação para auxiliar na normalização de trajetórias com tamanhos diferentes;

6. Segmentar as trajetórias comparando seus tamanhos quanto ao comprimento (distância) e direção para que então possam ser analisadas;

7. Implementar os algoritmos para cálculo do azimute e da deflexão;

8. Implementar o algoritmo para cálculo da correlação estatística de Pearson.

1.4 Estrutura do Trabalho

O texto está organizado da seguinte maneira. O capítulo 1 tratou do aspecto introdutório, dando um panorama acerca do problema de identificação de similaridade de forma entre trajetórias, de modo a demonstrar a relevância deste trabalho. O capítulo 2 descreve os conceitos relacionados ao trabalho onde os mais relevantes são trajetórias de objetos móveis, goniologia (azimute e deflexão) e correlação estatística. O capítulo 3 apresenta a revisão dos principais trabalhos relacionados. O capítulo 4 explica o método proposto e descreve em detalhes cada uma de suas etapas. O capítulo 5 descreve os experimentos realizados e discute os resultados obtidos. Por fim, no capítulo 6 são apresentadas as conclusões, contribuições e as propostas de trabalhos futuros.

2

CONCEITOS

Este capítulo apresenta os aspectos mais relevantes relacionados aos principais conceitos abordados neste trabalho. Desta forma, a seção 2.1 apresenta a terminologia utilizada na conceituação da representação do movimento e de trajetórias. A seção 2.2 apresenta as principais medições de ângulos presentes na goniologia, área da topografia dedicada ao estudo de angulações. Por fim, a seção 2.3 descreve as técnicas de correlação estatísticas e conceitos correlatos.

2.1 Representação do Movimento e Trajetórias

A representação do movimento realizado por um objeto tem características espaço-temporais que são passiveis de estudo. Dentre as características do movimento, Peuquet (2002) destaca as seguintes: (i) o espaço (aonde); (ii) o tempo (quando); e os objetos em si (o quê). Ainda de acordo com Peuquet (2002), estas três características do movimento podem ser combinadas para responder às perguntas apresentadas no Quadro 1.

Já Andrienko (2003) define características espaço-temporais a serem analisadas com base nos tipos de alterações que ocorrem com o movimento ao longo do tempo, sendo elas: (i)Alterações existenciais, onde ora o movimento se apresenta de uma forma,

Quadro 1 – Perguntas aplicadas ao estudo do movimento

Pergunta	Resulta
Quando + Aonde → O quê?	Descreve o objeto ou um conjunto de objetos, presente(s) em uma dada localização num dado momento
Quando + O quê → Aonde?	Descreve a localização ou um conjunto de localizações, ocupado por dado objeto ou um conjunto destes em um determinado momento
Aonde + O quê → Quando?	Descreve quando, dado um conjunto de objetos, ocupou dada localização ou conjunto destas

Fonte: Peuquet, 2002

ora não; (ii) Alterações de propriedades espaciais: localização, forma, tamanho ou orientação; (iii) Alterações de propriedades expressas em atributos qualitativos ou quantitativos expressas através de seu aumento ou decremento.

Os conceitos de espaço, tempo e objeto (movimento) são considerados os elementos fundamentais para o estudo e análise de trajetórias. Dentre estes estudos, a identificação de similaridade de forma entre trajetórias se apresenta como uma área de destaque, dada sua relevância na identificação de padrões de deslocamentos entre indivíduos ou grupos. Renso (2013) confirma a importância desses três conceitos, onde, para ela, as informações sobre o movimento são representadas pelos seguintes conjuntos: espaço E (conjunto de localizações), tempo T (conjunto de instantes ou intervalos de tempo) e objetos O (conjunto de objetos). As próximas subseções detalham cada um desses conceitos.

2.1.1 Espaço

Espaço é um conjunto de localizações ou lugares por onde o objeto se desloca. Uma propriedade importante do conceito de espaço se refere à distância entre os seus elementos. Segundo Andrienko (2013) ao mesmo tempo, o espaço não tem origem natural e nenhuma ordenação natural entre os seus elementos. Portanto, para distinguir as posições no espaço é preciso introduzir nele algum sistema de referência, por exemplo, um sistema de coordenadas. Embora isso possa ser feito, em princípio, e de forma

arbitrária, existem alguns sistemas de referências já estabelecidos para tal fim, tais como o de coordenadas geográficas. Andrienko (2013) ainda comenta que, dependendo do tipo de necessidade, pode-se convencionar o espaço como tendo duas dimensões, onde cada posição pode ser definida como um par de coordenadas, ou três dimensões, onde cada posição pode ser definida como uma tripla de coordenadas.

2.1.2 Tempo

Matematicamente, o tempo é um conjunto contínuo que tem uma ordem linear entre seus elementos, onde esses elementos são momentos ou posições ao longo do tempo (AIGNER et al., 2011). De forma análoga ao espaço, onde as posições são definidas com base em um sistema de referência, a mesma situação é necessária para especificar o tempo. Em muitos casos, a referência temporal é feita como base no padrão definido pelo calendário Gregoriano, em que se tem a divisão de dias em horas, horas em minutos, e assim por diante. A hora do dia também pode ser especificada de acordo com o fuso horário local onde os dados são coletados ou segundo o padrão do meridiano de *Greenwich (GMT)* (ALLEN, 1983).

2.1.3 Objeto

Um conjunto de objetos inclui qualquer tipo de entidade física ou abstrata que possa ser identificada e individualizada. Objetos podem ser classificados de acordo com suas propriedades espaciais e temporais. Um objeto espacial é um objeto que tem uma determinada posição no espaço em qualquer momento de sua existência (LAUBE et al., 2007). Por sua vez, um objeto temporal, também chamado de evento, é um objeto que tem um tempo limitado de existência, ou seja, representa um objeto que tem uma posição em particular no tempo (DODGE; WEIBEL; LAUTENSCHÜTZ, 2008). Eventos espaciais são objetos que têm posições específicas no espaço e tempo. Objetos móveis são um tipo de objeto espacial capaz de mudar sua posição ao longo do tempo.

Eventos móveis, por sua vez, são eventos que mudam sua posição no espaço ao longo do tempo. Eventos espaciais e objetos móveis podem ser chamados como objetos espaço-temporais (DODGE; WEIBEL; LAUTENSCHÜTZ, 2008). O Quadro 2 apresenta uma síntese sobre os conceitos da representação do movimento.

Quadro 2 – Representação geográfica do movimento

Conceito	**Conceito Amplo**	**Propriedades**	**Exemplo**
Objeto espacial	Objeto	Tem uma certa posição no espaço (uma localização ou um conjunto de localizações, não necessariamente de forma contínua)	Prédios, residências, estradas, centro de uma cidade, carro, pedestre, iceberg, animais, chuva, trajetória, curva, parada, um animal perseguindo outro, um carro excedendo o limite de velocidade
Evento (objeto temporal)	Objeto	Aparece e ou desaparece durante o período de tempo analisado, tem uma certa posição no tempo (em um unidade de tempo ou unidades de tempo)	Chuva, trajetória, curva, parada, um animal perseguindo outro, um carro excedendo o limite de velocidade, o pôr do sol, uma estação (inverno)
Evento espacial (objeto espaço temporal)	Objeto espacial, evento	Tem certas posições no espaço e tempo	Chuva, trajetória, parada, curva, um animal perseguindo outro, um carro excedendo o limite de velocidade
Objeto espacial estático	Objeto espacial	Sua posição espacial é constante. Existe durante todo o tempo que está sob análise	Prédio, residência, estrada, centro de uma cidade
Movimento (objeto móvel)	Objeto espacial	Sua posição espacial muda ao longo do tempo	Carro, pedestres, iceberg, animais, um animal perseguindo outro, um carro excedendo o limite de velocidade
Evento de movimento	Movimento, evento	Existe durante uma sequência de tempo (não é instantâneo). A posição espacial muda ao longo do tempo	Um animal perseguindo o outro, um carro excedendo o limite de velocidade

Fonte: Andrienko, 2013

2.1.4 Definições relacionadas a Trajetórias de Objetos Móveis

Esta seção apresenta as definições referentes a trajetórias de objetos móveis e conceitos correlatos aplicados a este trabalho.

Definição 1. ***Coordenada****: Uma coordenada* (c) *é uma tupla* (x, y)*, tal que* x *é a latitude e* y *é a longitude. Dependendo do sistema de referência utilizado, uma coordenada pode definir uma posição georreferenciada sob a superfície da terra* (ZHENG et al., 2009).

Definição 2. ***Ponto****: Um ponto* (p) *é uma tupla* (c, t)*, tal que* c *é um coordenada e* t *representa o instante de tempo em que a coordenada* c *foi capturada* (BRAZ; ORLANDO, 2007).

Definição 3. ***Trajetória****: Uma trajetória* (T) *é composta por um vetor de pontos e pode ser definida da seguinte maneira* $T = [p_1, p_2, ..., p_n]$*, tal que* p_1 *é o ponto inicial,* p_n *é o ponto final e* n *é o seu número de pontos* (BRAZ; ORLANDO, 2007).

Definição 4. ***Segmento****: Um segmento* (S) *é um vetor* $S = [p_i, p_{i+1}, p_{i+2}, ..., p_f]$*, tal que* p_i *é o ponto inicial do segmento e* p_f *é o ponto final do segmento, e* $0 < p_i < p_f$ *e* $p_i < p_f < p_n$*,* n *é o número de pontos da trajetória, ou seja,* $S \subset T$ (ZHENG et al., 2009). *Portanto, um segmento pode ser considerado um subconjunto da trajetória. Segmento é o conceito aplicado quando se deseja delimitar parte da trajetória que se deseja analisar.*

2.1.5 Pré-processamento de Trajetórias

Para analisar um conjunto de trajetórias reais de forma a prover resultados significativos é necessário realizar um efetivo pré-processamento a fim de eliminar dados incorretos e/ou imprecisos, reduzir a quantidade de informações irrelevantes ou desnecessárias, assim como identificar os segmentos utilizados na análise a ser realizada. Dentre as atividades comumente realizadas no pré-processamento de trajetórias, as que mais se destacam são a limpeza, compressão e segmentação. As próximas subseções

apresentam exemplos de técnicas de limpeza, compressão e segmentação usualmente utilizadas.

2.1.5.1 Limpeza de Trajetórias

Trajetórias não são perfeitamente precisas devido ao ruído do sensor de recepção ou mesmo a outros fatores, tais como, a qualidade do sinal de GPS em área urbanas verticalizadas (ZHENG, 2015). Portanto, o processo de limpeza tem por objetivo retirar da trajetória pontos que contenham informações incorretas ou imprecisas e que, se mantidas, prejudicam o resultado da análise. O principal erro encontrado em pontos de trajetórias está relacionado à imprecisão no sinal de GPS no momento da coleta. Este ruído é justificado por diversas situações que podem impactar na qualidade do sinal de recepção do sensor de GPS. Por exemplo, lugares cercados por árvores ou prédios altos têm uma piora significativa na qualidade do sinal de satélite se comparados a lugares abertos. Além disso, é fato que receptores GPS civis têm uma piora proposital na qualidade de sinal recebido, fazendo com que a precisão máxima alcançada esteja entre 2 e 5 metros. Essa imprecisão na qualidade da identificação do ponto da trajetórias sobre a superfície terrestre faz com que a posição coletada não represente adequadamente a posição real do objeto. Essa situação introduz erro à trajetória coletada, assim como em outras informações derivadas, tais como velocidade, direção, comprimento etc.

Existem várias técnicas para realizar a limpeza de trajetórias. Zheng (2015) menciona a utilização de filtros de média (ou mediana) e de Kalman para melhorar a estimativa de um ponto com ruído. O primeiro filtro estima o valor de um ponto z_i pela média ou mediana dos valores dos seus $n-1$ predecessores. Já o segundo estima o valor de um ponto ponderando algumas medidas, tais como a velocidade e o modo de deslocamento, obedecendo às leis da física.

Entretanto, existem outros métodos que implementam heurísticas que ao invés de melhorar a estimativa do ponto, removem eles diretamente da trajetória utilizando algoritmos de

detecção de *outlier*. Costa (2014) implementa a remoção de *outlier* analisando o raio de precisão de cada ponto provido pelo próprio receptor GPS. Esse raio simboliza o erro máximo das coordenadas do ponto aferido pelo receptor. Se o raio for maior que um limite considerado aceitável o ponto é descartado. Entretanto, ainda segundo o próprio autor, em algumas situações, mesmo sendo informado pelo receptor GPS que o ponto tem uma alta precisão, esta informação pode estar equivocada e na verdade o ponto pode conter um ruído elevado. Para se identificar esse tipo de situação é necessário calcular a precisão média de um conjunto consecutivo de pontos. Se essa média de precisão for maior que um limite esperado, todos os pontos do conjunto são descartados. Além do raio de precisão, também é possível utilizar outras abordagens para detectar e descartar pontos com ruído. Uma dessas abordagens utiliza o ângulo de movimento de um ponto em relação aos adjacentes. Se a diferença ultrapassar um limite ele é descartado, pois representa um movimento irreal para o tipo de movimentação analisada, por exemplo, um carro, um ser humano, um avião etc. A velocidade de deslocamento entre um ponto e outro, assim como a proximidade entre um conjunto de pontos também podem ser utilizados como critérios para eliminar pontos que representem ruído. A proximidade entre vários pontos consecutivos pode indicar locais onde o objeto ficou parado ou se deslocando lentamente e, em alguns tipos de análises, a eliminação desses pontos não interfere no resultado.

2.1.5.2 Compressão de Trajetórias

Receptores GPS comumente são programados para coletar a localização do objeto a cada segundo. Isso gera uma enorme quantidade de dados que, por consequência, torna o processo de armazenamento e análise deles custoso e demorado. Além disso, para uma parcela considerável das análises sobre trajetórias esta quantidade de pontos por trajetória é desnecessária. Portanto, para diminuir a quantidade de pontos sem comprometer a qualidade da análise sobre a trajetória são propostos algoritmos de compressão. Existem duas categorias de algoritmos de compres-

são quanto ao modo de operação. Essas categorias são chamadas *online* e *offline* (ZHENG, 2015). A offline só pode ser aplicada após toda a trajetória a ser compactada ter sido coletada. Já a online pode compactar a trajetória ao mesmo tempo em que novos pontos são coletados.

Outra classificação dos algoritmos de compressão é baseada na métrica de distância utilizada. As métricas calculam o erro introduzido na trajetória após a aplicação do método de compressão. Existem três métricas de distâncias mencionadas pela literatura, são elas: *Perpendicular Euclidean Distance*, *Time-Ratio Distance* (MERATNIA; ROLF, 2004) e *Time Synchronized Euclidean Distance* (POTAMIAS; PATROUMPAS; SELLIS, 2006). A primeira mede a distância perpendicular entre um ponto Pi e uma linha entre um ponto antes e outro depois de Pi. A segunda métrica calcula a distância entre um ponto P_i e sua projeção P_i' sobre a linha S que conecta um ponto antes e outro depois de P_i. Esta projeção leva em consideração proporcionalmente o tempo decorrido entre os dois pontos que deram origem a S e o ponto P_i. A terceira métrica calcula a distância entre um ponto P_i e sua projeção P_i' sobre a linha S dividida em $n + 1$ intervalos de mesmo tamanho, onde P_i' se encontra na i-ésima posição de S, sendo n o número de pontos da trajetória.

Além das classificações apresentadas acima, existe uma classe particular de algoritmos que implementam uma memória fixa e pré-definida que representa o número máximo de pontos que a trajetória compactada pode ter. Esse tipo de algoritmo tem aplicação para casos particulares, pois não limita a compressão da trajetória baseado no erro máximo permitido, mas sim limita o tamanho da trajetória minimizando o erro aplicado pela compressão por meio da eliminação dos pontos que introduzem o menor erro possível.

Os principais algoritmos *offline* apresentados na literatura são *Douglas-Peucker* (DOUGLAS; PEUCKER, 1973) e *Top-Down Time-Ratio* (MERATNIA; ROLF, 2004). O primeiro utiliza a métrica *Perpendicular Euclidean Distance*, já o segundo utiliza a métrica *Time-Ratio Distance*. Em contrapartida, os prin-

cipais algoritmos *online* são *Opening-Window* e *Opening-Window Time-Ratio* (MERATNIA; ROLF, 2004), e *Spatiotemporal Trace* (STTrace) (POTAMIAS; PATROUMPAS; SELLIS, 2006) e *Spatio Quality Simplification Heuristic* (SQUISH) (MUCKELL et al., 2011). O primeiro utiliza a métrica *Perpendicular Euclidean Distance* e o segundo utiliza *Time-Ratio Distance*. O terceiro e quarto algoritmos utilizam a métrica *Time Synchronized Euclidean Distance* e implementam a memória fixa que define o número máximo de pontos da trajetória compactada.

A escolha do algoritmo de compressão mais adequado para o propósito desejado depende da análise de características tanto dos dados a serem compactados, quanto do processo de coleta e tipo de análise a ser feita. Essas características vão indicar qual métrica deve ser utilizada, bem como o tipo de processamento a ser feito (online ou offline), a necessidade de uma memória fixa ou não, assim como o erro máximo permitido. A equalização de todas essas variáveis é uma tarefa complexa, que deve ser realizada por meio de experimentação prática sobre uma amostra significativa de dados.

2.1.5.3 Segmentação

Em muitos cenários de aplicação a divisão das trajetórias em segmentos é necessária antes de dar sequência ao processamento. A segmentação não apenas reduz a complexidade computacional da trajetória, mas também possibilita a mineração de conhecimento mais rico, tais como padrões em subtrajetórias etc. (ZHENG, 2015). A literatura apresenta três tipos de métodos de segmentação, sendo eles: i) Baseado em intervalo de tempo; ii) Baseado na forma da trajetória; iii) Baseado no significado semântico dos pontos da trajetória.

Para trabalhos que envolvam visualização do resultado, a segmentação aplicada é baseada na forma da trajetória. O método tem por objetivo dividir a trajetória em segmentos classificando-os como retas, curvas à esquerda ou curvas à direita. Como pode ser visto no Algoritmo 1, um segmento s é um par ordenado (P, d), sendo P um subconjunto de pontos da trajetória T e d a

direção realizada por este segmento s. O resultado da segmentação é um vetor de segmentos $\overrightarrow{S} = [s_0, s_1, s_2, \ldots, s_n]$, onde n é o número de segmentos gerados.

O vetor de segmentos classificados é gerado pela função $segmenta(T) \rightarrow \overrightarrow{S}$. A função *segmenta* utiliza a função $adiciona(s, p) \rightarrow s$ para determinar se um ponto p pertence ao segmento analisado ou a um segmento com direção diferente, por exemplo, o segmento sendo analisado é uma reta, mas o ponto p tem direção de curva à esquerda. Nesse caso, esse ponto pertence a um segmento com outra direção e, portanto, um novo segmento s é criado. O pseudo-código da função *segmenta* é apresentado no Algoritmo 1.

Algoritmo 1 Algoritmo de Segmentação

```
função segmenta(T : Trajetoria)
    S ← {};
    P ← {};
    d ← ∅;
    s ← cria(P, d);
    para cada p ∈ T faça
        s ← adiciona(s, p);
        se p ∉ s.P então
            Adicionar s em S;
            P ← {};
            d ← ∅;
            s ← cria(P, d);
            s ← adiciona(s, p);
        fim se
    fim para
    Adicionar s em S;
devolve S;
fim função
```

O método *adiciona* tem por finalidade incluir um ponto p em um segmento s, conforme apresentado no Algoritmo 2. Entretanto, para que o ponto p seja adicionado ao segmento s é

necessário que ele tenha a mesma direção que o segmento analisado. Caso contrário, a função *adiciona* retorna s sem a inclusão de p, fazendo com que a função *segmenta* crie um novo segmento para conter p.

Algoritmo 2 Algoritmo para adicionar ponto ao segmento

```
função adiciona((P, d) : segmento, p : ponto)
    se P = {} então
        Adicionar p em P;
    senão
        se d = ∅ então
            p_i ← primeiro ponto de P;
            Adicionar p em P;
            d ← dir(p_i, p);
        senão
            p_i ← primeiro ponto de P;
            p_f ← último ponto de P;
            se dir(p_f, p) = dir(p_i, p) e dir(p_f, p) = d então
                Adicionar p em P;
            fim se
        fim se
    fim se
devolve (P, d);
fim função
```

Para se ter uma melhor estimativa sobre a direção do segmento, especialmente quando ele tem a forma de uma curva suave, é comparada a direção tanto do primeiro ponto p_i quanto do último ponto p_f do segmento em relação ao ponto p a ser adicionado. Caso o primeiro ponto do segmento não seja levado em consideração, curvas cuja angulação entre o último ponto contido no segmento e o ponto analisado seja menor que o limite de tolerância ϵ serão consideradas retas. Porém, se comparado com o primeiro ponto do segmento, é possível que o ângulo de direção entre eles seja maior que ϵ e, com isso, o segmento seja considerado uma curva para a esquerda ou direita. O Algoritmo

2 apresentado no pseudocódigo utilizado para realizar a inclusão de pontos no segmento.

Para saber se p tem mesma direção de s é utilizada a função $dir(p_1, p_2) \Rightarrow d$ que tem por objetivo classificar a direção d entre os pontos p_1 e p_2 recebidos como parâmetro, conforme apresentado no Algoritmo 3. A direção é calculada pela diferença da direção de movimento entre dois pontos (Δd), dada pela fórmula de cálculo de azimute apresentada na Equação 2.1. Tal diferença é comparada a um limite pré-estabelecido (ϵ), sendo que caso $\Delta d > \epsilon$ então d será uma curva para a "esquerda", senão, se $\Delta d < -\epsilon$ então d será uma curva para a "direita", caso contrário, d será uma "reta". O pseudocódigo dessa função é apresentado no Algoritmo 3.

Algoritmo 3 Algoritmo para definir a direção de dois pontos

```
função dir(p1 : ponto, p2 : ponto, ε : inteiro)
    direção ← reta;
    Δd ← azimute(p1) − azimute(p2);
    se Δd > ε então
        direção ← esquerda;
    senão
        se Δd < −ε então
            direção ← direita;
        fim se
    fim se
devolve direção;
fim função
```

2.2 Goniologia

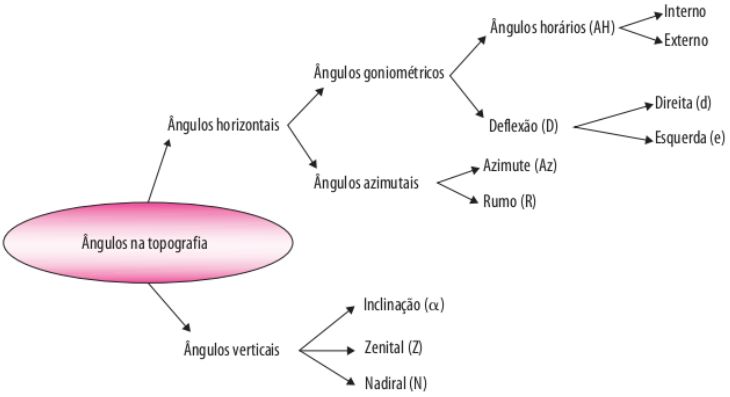
A Goniologia é a área da topografia dedicada a estudar os processos e instrumentos utilizados para avaliar e medir os ângulos das geometrias (TULER; SARAIVA, 2014). Ela é dividida em:

- Goniografia: estuda os processos de representação gráfica dos ângulos;

- Goniometria: estuda os processos e instrumentos necessários para a medição dos ângulos em campo.

A Figura 1 lista os vários tipos de ângulos usados na topografia.

Figura 1 – Ângulos da topografia



Fonte: Tuler e Saraiva, 2014

Dentre os ângulos analisados na goniologia e apresentados na Figura 1, para este trabalho os que se destacam com maior relevância são o azimute e a deflexão. Portanto, as subseções 2.2.1 e 2.2.2 apresentam em detalhes a definição e a forma de cálculo do azimute e da deflexão, respectivamente.

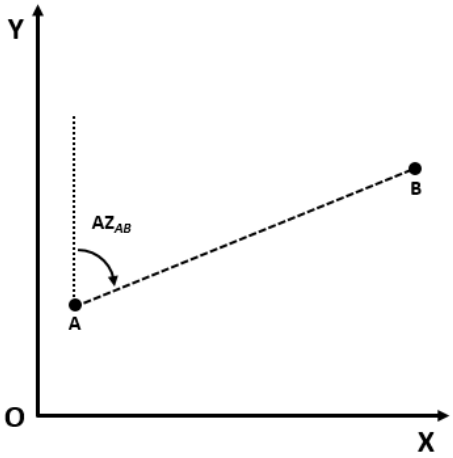
2.2.1 Azimute

Azimute (AZ) é o nome dado ao ângulo horizontal medido de forma horária entre a direção norte-sul do meridiano que passa pelo ponto em análise e um alinhamento (segmento) entre esse

ponto e o ponto seguinte. Este ângulo tem variação entre 0° e 360° (SILVA; SEGANTINE, 2014).

A Figura 2 mostra de forma gráfica como é representado o ângulo de azimute. Nela é possível perceber o ângulo horizontal horário compreendido entre o alinhamento de referência $\overline{OY}$ e o segmento $\overline{AB}$.

Figura 2 – Representação azimute



Fonte: Adaptado de Silva e Segantine, 2014

Dependendo do tipo de alinhamento adotado como referência, um azimute pode ser denominado *azimute magnético* ou *azimute adotado*. Um azimute é dito *magnético* quando o alinhamento de referência é a direção do norte magnético indicada pela agulha de uma bússola magnética. Um azimute é dito *adotado* quando o alinhamento de referência é uma direção qualquer adotada como referência (SILVA; SEGANTINE, 2014).

O azimute pode ser obtido pela Equação 2.1.

$$AZ_i = \arctan 2 \left(\frac{\sin \Delta\lambda . \cos \phi_{i+1}}{\cos \phi_i . \sin \phi_{i+1} - \sin \phi_i . \cos \phi_{i+1} . \cos \Delta\lambda} \right). \quad (2.1)$$

$\Delta\lambda = (\lambda_i - \lambda_{i+1})$, onde λ e ϕ indicam a longitude e latitude do ponto p_i, respectivamente.

Um vetor de azimutes é um conjunto de azimutes definido como $\overrightarrow{AZ} = [AZ_i, AZ_{i+1}, AZ_{i+2}, ..., AZ_j]$, tal que $0 < i < j$ e $i < j \leq n - 1$, onde n é o número de pontos da trajetória.

2.2.2 Deflexão

Deflexão é o nome dado ao ângulo formado entre o prolongamento do alinhamento (segmento) anterior e o alinhamento em análise, contado para a direita ou para a esquerda, com sua grandeza limitada entre 0° e 180° (BORGES, 2013). Neste trabalho, a fórmula de cálculo da deflexão adotada é dada pela diferença entre dois azimutes consecutivos. Essa diferença é positiva se o primeiro azimute da sequência for maior que o segundo, caso contrário, é negativa. Dessa forma, a deflexão pode ser obtida pela Equação 2.2.

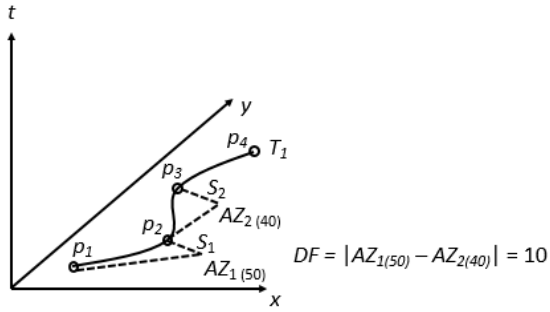
$$DF_i = |\,(AZ_i - AZ_{i+1})\,|. \quad (2.2)$$

AZ_i representa o azimute entre dois pontos dentro de uma trajetória e AZ_{i+1} o azimute subsequente entre dois pontos, também dentro da mesma trajetória. O módulo foi utilizado para desconsiderar o sinal da deflexão calculada, dado que para este trabalho o valor positivo é suficiente. Detalhes sobre o uso da deflexão no método proposto podem ser vistos no capítulo 4.

Um vetor de deflexão é um vetor definido como $\overrightarrow{DF} = [DF_i, DF_{i+1}, DF_{i+2}, ..., DF_k]$, onde $0 < i < k$ e $i < k \leq j - 1$, j é o número de azimutes da trajetória.

A Figura 3 demonstra o cálculo para obtenção de uma deflexão com base na diferença modularizada de dois azimutes consecutivos, conforme definido na Equação 2.2.

Figura 3 – Representação do cálculo da deflexão



Fonte: produção do próprio autor

2.3 Métodos Estatísticos

A análise estatística dos resultados obtidos em um determinado estudo é uma ferramenta importante na validação ou refutação desses resultados. De acordo com Normando (2010), a escolha do método estatístico apropriado requer do pesquisador conhecimentos básicos para:

- Classificar o tipo de dado estudado (contínuo, categórico: ordinal ou nominal);
- Identificar como esses dados estão distribuídos após o término da sua coleta (Distribuição Normal ou Distribuição Não Normal);
- Determinar os tipos de amostras examinadas (Independentes ou Dependentes).

Tendo conhecimento do tipo de dados que se está trabalhando, pode-se verificar como eles estão distribuídos e se as amostras são independentes ou não. Então, com base nessas informações, o pesquisador pode definir o ferramental estatístico

mais adequado para o estudo dos dados analisados (SHEATS; PANKRATZ, 2002).

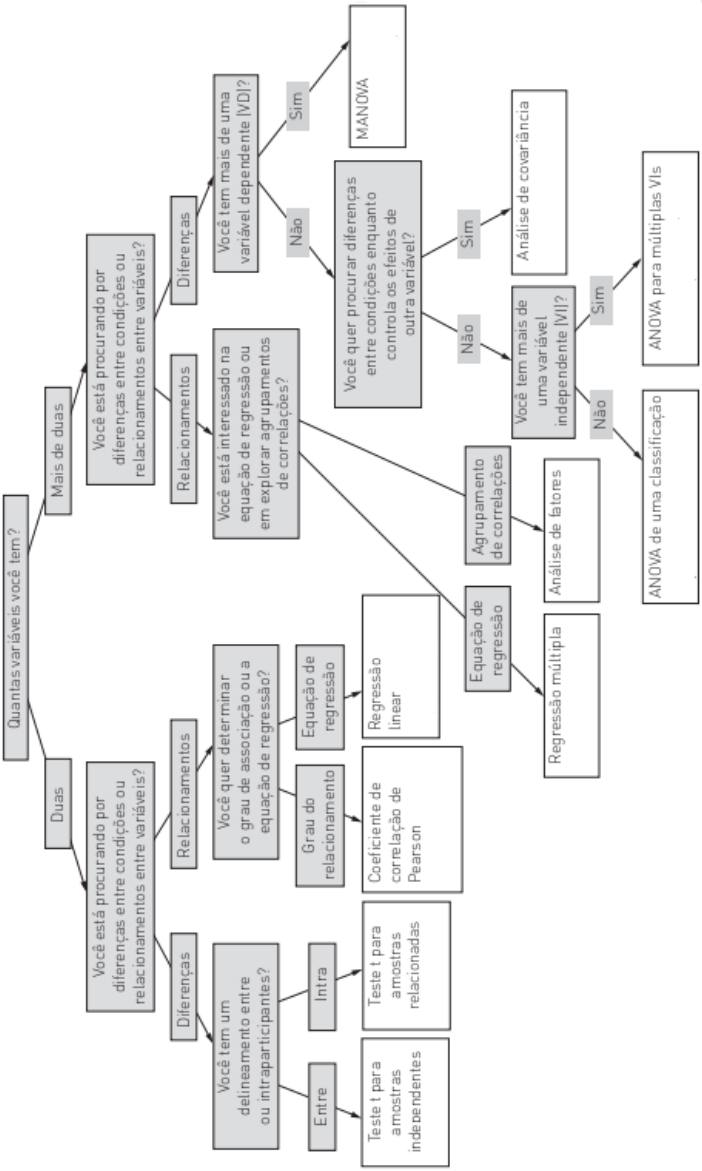
Por exemplo, para o coeficiente de correlação de Pearson (ver seção 2.3.1.1) são necessários dados que tenham uma distribuição normal.

Quando os dados estão distribuídos normalmente, distribuição esta também conhecida como distribuição Gaussiana (relativo à Carl Gauss), eles apresentam uma forma semelhante à curvatura de um sino (NORMANDO; TJÄDERHANE; QUINTÃO, 2010). Nesse tipo de distribuição os dados se concentram em torno de uma média e se dispersam simetricamente a partir desse ponto central. Porém, quando a curva de distribuição dos dados não apresenta uma forma de sino é chamada de assimétrica, não normal ou de livre distribuição (NORMANDO; TJÄDERHANE; QUINTÃO, 2010).

Muitos métodos estatísticos são baseados na estimativa de certos parâmetros relacionados à população analisada. Esses tipos de métodos são chamados de métodos paramétricos (DANCEY; REIDY, 2013). Eles fazem suposições de que as amostras são similares às distribuições de probabilidade subjacentes como a distribuição normal padrão (SHEATS; PANKRATZ, 2002). Já os métodos estatísticos que não fazem suposições sobre as distribuições subjacentes ou estimam os parâmetros de uma população em particular, são denominados de métodos não paramétricos ou métodos de livre distribuição (DANCEY; REIDY, 2013). Segundo a literatura, se a amostra analisada apresenta uma distribuição normal dos dados, então deve-se buscar a aplicação de um método paramétrico, caso contrário, se a amostra não apresenta distribuição com essa normalidade, a aplicação de um método não paramétrico é a mais indicada.

A Figura 4 apresenta uma síntese do tipo de método estatístico que pode ser aplicado conforme o tipo de análise estatística realizada.

Figura 4 – Escolha de método estatístico



Fonte: Dancey e Reidy, 2013

2.3.1 Análise de Correlação entre Variáveis

Esta seção apresenta técnicas estatísticas para mensurar a correlação entre variáveis. A correlação de variáveis é aplicada quando se deseja mensurar o grau de semelhança do comportamento de duas ou mais variáveis multivaloradas. Quando elas apresentam comportamentos semelhantes entre seus valores é possível assumir que elas apresentam correlação. Portanto, o princípio básico dessas técnicas é avaliar quão semelhantes são os comportamentos das variáveis analisadas. Quando se considera apenas o relacionamento entre duas variáveis, a isso dá-se o nome de correlação bivariada (DANCEY; REIDY, 2013). As próximas subseções apresentam duas técnicas para cálculo de correlação, que são o coeficiente de Pearson (seção 2.3.1.1) e o coeficiente de Spearman (seção 2.3.1.2).

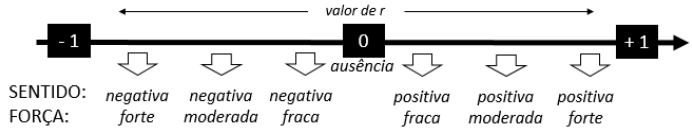
2.3.1.1 Pearson

O coeficiente de Pearson (r) é um tratamento estatístico paramétrico para verificar a tendência entre os valores de conjuntos de dados, sendo que o valor fornecido por esta correlação é adimensional (STIGLER, 1989). Outra forma de entender o coeficiente de correlação de Pearson é que este é utilizado para medir a direção e a força da relação linear entre duas variáveis, x e y (BARBETTA; REIS; BORNIA, 2010). Stigler (1989) menciona que o valor do coeficiente de correlação de Pearson está compreendido no intervalo de -1 a $+1$, sendo que os valores -1 e $+1$ indicam uma correlação perfeita entre duas variáveis, e o valor 0 indica ausência de correlação (CHEN; POPOVICH, 2002). A Figura 5 mostra os possíveis valores de r e suas interpretações em termos de sentido (positivo ou negativo) e de força (fraca, moderada ou forte) da correlação.

A literatura define unanimemente que os valores -1 e $+1$ representam forte correlação, já o valor 0 representa ausência de correlação. Entretanto, para os valores contidos nos intervalos de -1 a 0 e de 0 a $+1$ não há consenso sobre o grau de correlação entre as variáveis analisadas. Porém, alguns autores apresentam

propostas para interpretar o grau de correlação entre as variáveis dentro desses intervalos de r. Por exemplo, Cohen (2013) afirma que valores entre ± 0.10 a ± 0.29 podem ser considerados como pequena correlação, escores entre ± 0.30 a ± 0.49 podem ser considerados como média correção, e valores entre ± 0.50 e ± 1 podem ser interpretados como grande correlação. Dancey e Reidy (2013) apontam para uma classificação ligeiramente diferente. Para eles, r entre ± 0.10 e ± 0.39 significa fraca correlação, entre ± 0.40 e ± 0.69 é considerada correlação moderada, e entre ± 0.70 e ± 1 representa uma correlação forte. Figueiredo (2010) comenta que independente dos valores alcançados, quanto mais perto de ± 1 maior será o grau de dependência estatística linear entre as variáveis. Ainda segundo Figueiredo (2010), caso contrário, quanto mais próximo de zero, menor é a força de relação entre elas.

Figura 5 – Representação da força da correlação de Pearson



Fonte: Barbetta, 2010

Conforme Dancey e Reidy (2013), para utilizar o coeficiente de Pearson é necessário que as variáveis analisadas atendam às seguintes regras:

1. As variáveis x e y devem ser numéricas (ou quantitativas). Elas devem representar medições sem nenhuma restrição em seu nível de precisão. Por exemplo, os números com muitas casas decimais (por exemplo, 12,322 ou 0,219) devem ser preservados (e não arredondados);

2. As variáveis x e y devem ter uma relação linear;

3. Os valores de y devem ter uma distribuição normal para cada x, com a mesma variância em cada x, de forma a ter uma tendência linear dos dados.

O cálculo da correlação de Pearson (r) é dado pela razão entre a covariância e o desvio-padrão das variáveis x e y, conforme a Equação 2.3 descrita por (BARBETTA; REIS; BORNIA, 2010):

$$r = \frac{n \sum (x_i.y_i) - (\sum x_i)(\sum y_i)}{\sqrt{n \sum x_i^2 - (\sum x_i)^2}.\sqrt{n \sum y_i^2 - (\sum y_i)^2}}. \quad (2.3)$$

A variável n indica a quantidade de elementos observados. Já x_i representa o valor de cada um dos elementos do vetor x onde i varia de 1 a n. Da mesma forma, y_i representa o valor de cada um dos elementos do vetor y onde i varia de 1 a n.

Entretanto, o coeficiente de Pearson apresenta algumas exigências quanto aos dados utilizados conforme listado nesta seção, sendo sua utilização recomendada quando os dados satisfazem essas condições (COHEN, 2013). Portanto, para que o resultado de sua aplicação seja relevante, as variáveis analisadas devem apresentar uma distribuição normal. Uma alternativa quando não são atendidas as suposições de r Pearson, é utilizar o método de Spearman (LEVIN; FOX, 2004).

2.3.1.2 Spearman

O coeficiente de Spearman (ρ) é muito semelhante ao de Pearson. Ambos são coeficientes de correlação e são interpretados da mesma maneira (DANCEY; REIDY, 2013). Entretanto, diferentemente do coeficiente de Pearson, o coeficiente de Spearman é indicado quando os dados não satisfazem as condições dos métodos paramétricos (COHEN, 2013). Isso significa que a correlação de Spearman não exige que a relação entre as variáveis x e y seja linear, nem que as variáveis sejam numéricas. Em vez de examinar uma relação linear entre x e y, o método de correlação de Spearman verifica se duas variáveis ordinais ou quantitativas são

independentes (isto é, não se relacionam entre si), mas apresentam relacionamento monotônico entre elas (BARBETTA; REIS; BORNIA, 2010).

Da mesma forma que o coeficiente de Pearson, o coeficiente de Spearman também apresenta a correlação como um valor compreendido no intervalo de -1 a $+1$, sendo que os valores -1 e $+1$ indicam uma correlação perfeita e o valor 0 indica ausência de correlação. Tal como em Pearson, as propostas de Cohen (2013) e Dancey e Reidy (2013) também podem ser aplicadas para interpretar o grau de correlação entre as variáveis dentro dos intervalos de -1 a 0 e de 0 a $+1$. A fórmula é dada pela Equação 2.4 descrita por (BARBETTA; REIS; BORNIA, 2010):

$$r_s = 1 - \frac{6 \sum d_i^2}{n\left(n^2 - 1\right)}. \tag{2.4}$$

2.3.1.3 Seleção do Coeficiente de Correlação

Escolher um coeficiente de correlação passa pela análise dos dados. Se estes dados atendem aos pressupostos para aplicação de um método paramétrico, como é o caso do coeficiente de correlação de Pearson, esta abordagem pode ser aplicada (COHEN, 2013). Entretanto se os dados não satisfazem essas condições, uma abordagem não paramétrica como Spearman pode ser uma alternativa de pesquisa e estudo dos dados (DANCEY; REIDY, 2013).

2.3.2 Teste de Significância do r de Pearson

Também chamado de teste de hipóteses, um teste de significância é basicamente um procedimento estatístico aplicado sobre os dados para comprovar uma alegação feita sobre uma população (BARBETTA; REIS; BORNIA, 2010). O teste de significância também pode ser entendido como o teste de hipóteses realizado para confirmar ou negar uma afirmação feita sobre uma população. A alegação que está sendo testada é, em essência,

denominada hipótese nula (H_0)(BARBETTA; REIS; BORNIA, 2010).

Assim, com base no r de Pearson, este nos dá a medida de intensidade e direção da correlação na amostra sendo estudada. Mesmo tomando uma amostra aleatória de uma população específica, talvez queiramos determinar se a associação obtida entre X e Y existe na *população*, e se não se deve meramente a um erro amostral (LEVIN; FOX, 2004).

2.3.2.1 Realização do Teste e Estabelecimento das Hipóteses

Para testar a significância de uma medida de correlação, normalmente é estabelecida a hipótese de que nenhuma correlação existe na população. Em relação ao coeficiente de correlação de Pearson, a hipótese nula afirma que a correlação populacional ρ (rho) é zero (LEVIN; FOX, 2004). Isto é, $\rho = 0$, enquanto a hipótese de pesquisa diz que $\rho \neq 0$.

2.3.2.2 Nível de Significância e P-Valor

Na realização de uma análise estatística, quando se deseja confirmar ou refutar alguma hipótese, é comum estabelecer, ainda na fase de planejamento da análise, a probabilidade tolerável de incorrer no erro de rejeição do H_0, quando H_0 é verdadeira (BARBETTA; REIS; BORNIA, 2010). Em outras palavras, significa dizer que ao se estabelecer o α de 0.05 existe a probabilidade ainda de 5% da hipótese nula (H_0) ser verdadeira. Ainda conforme Barbetta (2010), esse valor é conhecido como **nível de significância do teste** e é designado pela letra grega α (alfa).

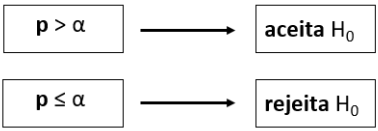
Uma vez definido o nível de significância da pesquisa, este será comparado com o *p-valor*. Conforme Barbetta (2010), o *p-valor* é definido como a probabilidade de a estatística do teste acusar um resultado tão (ou mais distante) do valor esperado quanto o ocorrido na amostra observada, supondo a hipótese nula como verdadeira.

Isso é confirmado por Dancey e Reidy (2013) que reiteram que α é o critério de significância fixado para uma análise

estatística. O *p-valor* pode ser utilizado como um ponto de decisão para os valores apurados nos experimentos estatísticos, de forma que os valores de *p-valor* menores que α levam à rejeição da hipótese nula.

Uma abstração desse conceito pode ser vista na Figura 6.

Figura 6 – Aceitação da hipótese nula



Fonte: Adaptado de Barbetta, 2010

2.3.2.3 Erro Tipo Alfa e Tipo Beta

Os erros denominados como α (alfa) e β (beta), conhecidos também como erros do tipo I e do tipo II, respectivamente, referem-se aos erros que podem acontecer em uma pesquisa estatística ao se descartar uma das hipóteses (H_0 ou H_1) de análise sobre uma população. Para Barbetta (2010), a hipótese descartada pode ou não ser verdadeira, já que não há como saber a realidade da população, uma vez que só uma amostra dela é conhecida. Desta maneira, um erro do tipo I ocorre quando o pesquisador decide rejeitar a hipótese nula (H_0) e, na verdade, ela é verdadeira na população subjacente. Isto é, conclui-se que existe um efeito na população quando tal efeito não existe (COHEN, 2013).

Ainda segundo Barbetta (2010), o erro do tipo II é a probabilidade condicional de não rejeitar a (H_0) quando ela é realmente falsa. Como α é fixado *a priori*, se durante a pesquisa opta-se por rejeitar H_0 em favor de H_1 ($p \leq \alpha$), o risco do pesquisador tomar a decisão errada, erro do tipo I, fica limitado pelo nível de significância α adotado (DANCEY; REIDY, 2013). Entretanto, se durante a pesquisa se aceita H_0 o pesquisador corre o risco de tomar uma decisão errada, erro do tipo II (β beta), pois a

probabilidade β, em geral, não é conhecida (DANCEY; REIDY, 2013). Conforme Cohen (2013), ao aceitar a hipótese nula H_0, entende-se que os dados estão em conformidade com essa hipótese, todavia isso não implica que H_0 seja realmente a hipótese verdadeira, mas que os dados não mostram evidência suficiente para rejeitá-la.

3

TRABALHOS RELACIONADOS

A literatura apresenta vários trabalhos que propõem métricas e métodos para identificar similaridades entre trajetórias. Dentre eles, os mais relevantes para o trabalho em questão são detalhados a seguir.

3.1 *Using Dynamic Time Warping to Find Patterns in Time Series*

Berndt e Clifford (1994) propõem o uso do algoritmo DTW para encontrar padrões em séries temporais. O algoritmo DTW é usado para encontrar o melhor alinhamento entre elementos de duas sequências. O algoritmo cria uma matriz com todas as possíveis combinações de dois em dois elementos fornecidos pelas séries tendo, então, como resultado a distância total entre as duas sequências com base na soma do menor caminho mínimo construído por essa matriz. Uma limitação dessa abordagem se dá quando os dados têm algum ruído, por exemplo, com a existência de valores muito altos que fogem dos padrões das séries, fazendo com que o menor caminho encontrado saia distorcido devido às combinações com este valor discrepante.

3.2 *Discovering similar multidimensional trajectories*

Vlachos et al. (2002) propõem um método que considera o tempo e o deslocamento para identificar a maior sequência de pontos que represente similaridade de forma no espaço e tempo entre duas trajetórias. A esse método foi dado o nome de *Longest Common Subsequence* (LCSS). Conforme mencionado pelos autores, o método é eficiente na presença de ruído (pontos fora do padrão), pois o método foca nas partes similares da trajetória, desconsiderando as não similares, que indicam a presença de pontos fora de contexto (ruídos). Entretanto, o método tem um problema, quando as trajetórias analisadas têm tamanhos diferentes, por exemplo, dadas três trajetórias, onde T_1 tem quatro pontos, T_2 tem três pontos e T_3 tem dois pontos e considerando que dois pontos de T_1 e T_2 casem com os pontos de T_3. A distância subsequente mais comum será igual a dois pontos em todas as situações, ignorando que existem ainda mais dois pontos em T_1 e um em T_2.

3.3 *Robust and fast similarity search for moving object trajectories*

Chen et al. (2005) propõem uma função denominada *Edit Distance on Real Sequences* (EDR) que tem como base outra função denominada *Edit Distance* (ED). Esta última é utilizada para mensurar a similaridade entre duas sequências de caracteres (*strings*) por meio do cálculo do número mínimo de inserções, exclusões e trocas necessárias para que ambas se tornem idênticas. Já a função EDR calcula a similaridade de forma entre as trajetórias com base na distância entre elas. O método utiliza distância euclidiana para identificar similaridade entre trajetórias próximas, e utiliza a função ED para verificar quantas transformações são necessárias para achar a similaridade. O método proposto não é capaz de identificar similaridade entre trajetórias distantes umas das outras.

3.4 *The double-cross and the generalization concept as a basis for representing and comparing shapes of polylines*

Van de Weghe et al. (2005) propõem um método denominado *Qualitative Trajectory Calculus* (QTC_c) para identificar similaridade de forma das trajetórias. Esse método usa a direção para calcular essa similaridade. Para isso cria-se dois vetores de formas geométricas que por meio de 4-tuplas representam a orientação de ambos os vetores. Com base nos valores apurados nos vetores é construída uma matriz que os autores denominam matriz de forma. Por meio dessa matriz são aplicadas funções que calculam a similaridade das trajetórias analisadas. Esse método apresenta como limitação o tempo exponencial necessário para o cálculo da matriz de forma quando as trajetórias analisadas têm muitos pontos. Essa limitação inviabiliza sua utilização na comparação da maioria das trajetórias coletadas por GPS. Além disso, o método tem dificuldade em lidar com trajetórias que representam polígonos fechados.

3.5 *Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects*

Dodge et al. (2009) desenvolveram um *framework* metodológico focado na análise de similaridade com base no comportamento dinâmico de objetos móveis e suas trajetórias. A abordagem proposta envolve três etapas: preparação dos dados (filtros, suavização etc.), descritores globais (velocidade, aceleração etc.) e informações locais do objeto (ângulos de conversão, aceleração etc.). O método calcula similaridade com base no comportamento, o que faz com que os dados analisados sejam remodelados em intervalos regulares de tempo, usando interpolação linear com intervalos fixos. Isso significa que o método foca em trechos das trajetórias, em vez dos pontos das amostras, o que o torna limi-

tado a períodos fixos de amostragens.

3.6 *Similarity measurement of moving object trajectories*

Liu e Schneider (2012) propõem uma abordagem para calcular a similaridade de trajetórias que considera não só aspectos geográficos, mas também aspectos semânticos do movimento realizado. Para o cálculo do aspecto geográfico são considerados: rumo, distância entre as trajetórias, centro de massa da trajetória, menor distância entre o ponto final e inicial de uma trajetória e ângulo cosseno para encontrar subtrajetórias. Para o aspecto semântico são considerados locais (hotel, museu etc.) por onde a trajetória se desenvolveu. Entretanto, por vezes dependendo dos dados analisados, pode haver limitações quanto às informações constantes nos dados, o que tenderia a prejudicar a execução do método.

3.7 *EDS: a segment-based distance measure for subtrajectory similarity search*

Xie (2014) propõe uma métrica para calcular a distância chamada de *Edit Distance on Segment* (EDS). Essa métrica é utilizada para verificar a similaridade quanto à forma em subtrajetórias usando seus segmentos. Para calcular essa similaridade, o autor define um custo referente à transformação de um segmento em outro. A ideia é que dados dois segmentos é possível transformar ambos por meio de prologamentos, rotações ou mesmo deslocamentos destes, de maneira a identificar a similaridade entre subtrajetórias. O método não leva em conta a trajetória em sua totalidade, além disso o método não considera trajetórias distantes umas das outras.

3.8 *Multidimensional Similarity Measuring for Semantic Trajectories*

Furtado et al. (2015), propõe uma medida de similaridade multidimensional quanto a questão semântica. Os autores, realizaram um extenso estudo experimental, com base em trajetórias geradas de forma controlada, aonde é introduzindo nesse conjunto diferentes tipos e níveis de dissimilaridade. Com isso para cada conjunto gerado de trajetórias, esse é comparado com as trajetórias originais, que também foram transformadas para lidar com essas métricas. Entre as métricas propostas para essa avaliação estão: distância, tempo e abordagens semânticas como *stops* (paradas). A exemplo do trabalho de Liu e Schneider (2012), por vezes, os dados analisados podem ter limitações quanto as informações constantes em seu conteúdo. O que tenderia a prejudicar a execução do método.

3.9 *Shape Similarity Based on the Qualitative Spatial Reasoning Calculus eOPRAm*

Dorr, Latecki e Moratz (2015) investigam e propõem o método *Qualitative Spatial Representations* (QSR). Esse método investiga a relação de direção e distância para representação da forma. Para tanto são usados cálculos quanto à direção e distância dos pontos. Entretanto, uma das limitações do método se refere a que este deve ter tamanhos iguais seja em relação à direção ou distância para comparar as formas das trajetórias.

3.10 Considerações sobre os trabalhos relacionados

Pela análise da literatura resumida neste capítulo, pode-se observar que a maioria dos trabalhos apresenta abordagens para identificação de similaridade entre trajetórias próximas umas das outras. Também é possível perceber que elas consideram tanto

aspectos físicos, como a forma, quanto comportamentais, como a velocidade e a aceleração, na identificação de similaridade.

Um dos trabalhos iniciais na área de similaridade foi proposto por Vlachos et al. (2002), via (LCSS), que tenta identificar a maior sequência de similaridade para duas ou mais trajetórias comparadas entre si. Entretanto, o método não considera a direção. Já Chen et al. (2005), por sua vez, propõem a abordagem EDR. Porém, o método apresenta problemas na identificação de similaridade entre trajetórias distantes umas das outras. Essa limitação também aparece nos trabalhos de Xie (2014).

Já Berndt e Clifford (1994), Dodge et al. (2009), Liu e Schneider (2012) e Furtado et al. (2015) realizam várias transformações para obter os resultados necessários. E Van de Weghe et al. (2005) e Dorr, Latecki e Moratz (2015) têm no primeiro a apresentação de limitações quanto à manipulação de formas poligonais fechadas, e o segundo a questão de tamanhos iguais quanto à questão de direção ou distância para comparar as formas das trajetórias.

O trabalho aqui proposto tenta solucionar o problema de trajetórias quando estas estão distantes e em direções diferentes, de maneira ainda a identificar similaridade de forma quando isto acontece. Assim, a comparação entre as trajetórias é feita pelos valores de angulação entre seus segmentos, fazendo com que seja feita uma comparação relativa entre elas, o que significa que não há necessidade de elas estarem próximas e nem necessariamente na mesma direção. Dependendo de informações mínimas constantes nos dados, como latitude, longitude e tempo em que ponto da trajetória foi coletado.

4

DESENVOLVIMENTO DA SOLUÇÃO

Com a ampliação do uso de dispositivos móveis equipados com sensores de georreferenciamento, a quantidade e variedade de trajetórias coletadas têm aumentado exponencialmente, o que propiciou a criação de um vasto campo de pesquisa. A geração de conhecimento sobre a análise de trajetórias ganha cada dia mais importância, pois, a partir da interpretação dos seus dados, é possível, por exemplo, encontrar padrões nos deslocamentos de um objeto ou grupo de objetos. Entretanto, analisar e gerar conhecimento sobre esse tipo de informação requer metodologias e processos computacionais adequados.

Conforme mencionado na revisão de trabalhos relacionados apresentada no capítulo 3, é possível perceber que as contribuições para entender as trajetórias, principalmente no que tange à identificação de similaridade de forma entre elas, ainda é um campo aberto. Dentre as demandas pertinentes nessa área, destacam-se a necessidade de uma abordagem que funcione independente da proximidade das trajetórias, e que proveja uma medida quantitativa e/ou qualitativa capaz de inferir o grau de similaridade entre as trajetórias analisadas.

Com base nessas necessidades, este trabalho propõe a utilização dos ângulos de deflexão dos segmentos das trajetórias

como informação aplicada a uma técnica de correlação estatística. Como resultado, é produzido um valor de correlação que neste trabalho é considerado a medida para indicar o grau de similaridade de forma entre as trajetórias analisadas. As próximas subseções estão estruturadas da seguinte maneira.

A seção 4.1 apresenta a sequência de etapas do método. A seção 4.2 apresenta a implementação da solução proposta. Por fim, a seção 4.3 mostra o esquema de dados proposto para armazenar os dados utilizados na pesquisa.

4.1 Método para Cálculo de Similaridade de Forma

Conforme mencionado no capítulo 1, este trabalho assume que é possível identificar a similaridade de forma entre duas trajetórias por meio da aplicação de uma técnica de correlação estatística sobre os vetores de deflexão obtidos delas. Essa suposição é baseada na ideia de que se as trajetórias apresentam formas visualmente similares, então terão deflexões angulares (DF) similares entre seus segmentos.

O detalhamento das etapas deste podem ser assim descritas:

1. **Seleção da trajetória de referência:** O primeiro passo é selecionar a trajetória (ou segmento desta) T_{ref} usada como referência na comparação com outras a fim de identificar se elas têm formato parecido;

2. **Seleção do conjunto de trajetórias para comparação:** Após a seleção das trajetórias de referência, é necessário selecionar um conjunto de trajetórias (ou segmentos destas) $T_{conj} = (T_1, T_2, ..., T_m)$, onde m é o número de trajetórias selecionadas para comparação com a de referência;

3. **Limpeza das trajetórias:** Como primeiro passo do pré-processamento, tanto a trajetória de referência T_{ref} quanto o conjunto de comparação T_{conj} são limpos por meio da aplicação de técnicas discutidas na seção 2.1.5.1. Como as

técnicas de correlação (Pearson) aplicadas em experimentos preliminares se mostraram muito sensíveis às diferenças entre as deflexões das trajetórias comparadas, optou-se por aplicar duas técnicas de limpeza para tentar minimizar o ruído presente em trajetórias reais. A primeira técnica aplicada realiza a eliminação de pontos com distância inferior a 5 metros do anterior e a segunda faz a eliminação de pontos com diferença de angulação maior que 45° do anterior. A adoção da limpeza por 5 metros e 45° nessa etapa do método se deu empiricamente com base na realização de testes sobre os dados disponíveis para os experimentos. Nesses testes, foi possível verificar que o uso de valores menores que 5 metros ainda apresentavam muitos pontos próximos, e angulações acima de 45° caracterizavam pontos fora das vias nos casos pré-analisados. Por fim, também foi percebido que os valores selecionados não comprometeram a forma das trajetórias. A aplicação destas técnicas têm como principal objetivo a eliminação de pontos que indicam paradas dos objetos móveis;

4. **Segmentação das trajetórias:** Para o cálculo dos azimutes e deflexões cada trajetória é dividida em segmentos compostos por dois pontos. Então, para cada trajetória T é gerado um vetor $\overrightarrow{S} = [s_i, s_{i+1}, s_{i+2}, ..., s_f]$ de segmentos, onde $0 < i < f$ e $i < f < n$, n é o número de pontos da trajetória, e $s_i = (p_i, p_{i+1})$, onde $p_i \in T$;

5. **Cálculo do azimute para os segmentos das trajetórias:** Para cada segmento s_i de uma trajetória T é calculado o azimute com base na Equação 2.1. O valor do azimute é usado como requisito para computar a deflexão que é usada então para identificar a similaridade de forma das trajetórias. Essa informação é armazenada em um vetor $\overrightarrow{AZ} = [AZ_i, AZ_{i+1}, AZ_{i+2}, ..., AZ_j]$, onde $0 < i < j$ e $i < j < n$, n é o número de pontos da trajetória;

6. **Cálculo da deflexão entre dois segmentos consecutivos:** Após o cálculo do azimutes referentes aos segmen-

tos $s_{i's}$ é possível calcular a deflexão com base na Equação 2.2. Os valores obtidos são guardados em um vetor $\overrightarrow{DF} = [DF_i, DF_{i+1}, DF_{i+2}, ..., DF_k]$, onde $0 < i < k$ e $i < k < j$, j é o número de azimutes da trajetória;

7. **Segmentação por direção e comparação das direções dos segmentos das trajetórias analisadas:** Como primeira etapa da identificação de similaridade, é necessário classificar os segmentos de uma trajetória T como reta, curva à esquerda ou curva à direita. Para classificar um segmento é aplicado o Algoritmo 1 apresentado na seção 2.1.5.3. Pare esse trabalho o parâmetro de tolerância de algulação dentro de um segmento é de 25°. Esse valor foi definido empiricamente como adequado após uma abrangente bateria de teste. Como resultado da segmentação por direção dos segmentos, tem-se um vetor de segmentos $\overrightarrow{S_{dir}} = [s_0, s_1, s_2, \ldots, s_n]$, onde s_i é um par ordenado (P, d), sendo P o conjunto de pontos do segmento e d a direção, n é o número de segmentos com direções distintas. Após a segmentação é realizada a comparação das direções dos segmentos, sendo que cada par de segmentos da trajetória de referência e da trajetória comparada, respectivamente, devem ter a mesma direção. Para que as trajetórias possam ter as direções dos seus segmentos comparadas, elas devem ter a mesma quantidade de segmentos;

8. **Cálculo e comparação do comprimento dos segmentos das trajetórias analisadas:** Outra restrição imposta ao cálculo de similaridade entre duas trajetórias é dada pelo comprimento dos segmentos de cada uma delas. Para que duas trajetórias possam ser comparadas elas devem ter segmentos que, além de terem mesma direção (etapa 7), tenham diferença máxima de até 25% entre seus comprimentos. A adoção da tolerância de 25% nessa etapa do método se deu com base em análise prévia dos dados disponíveis nas bases e que seriam utilizados nos cenários de testes. O cálculo do comprimento de cada segmento é feito utilizando

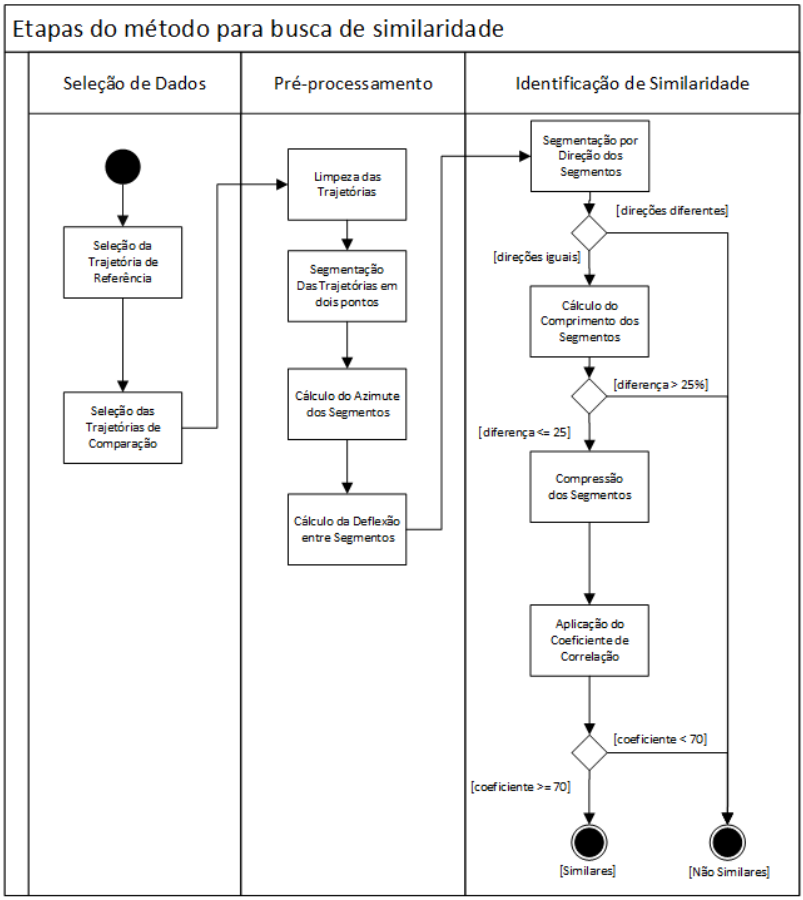
a fórmula de Haversine, conforme apresentado por Feng e Timmermans (2013);

9. **Equiparação na quantidade de pontos por segmento das trajetórias analisadas:** Caso as trajetórias analisadas tenham passado pelas verificações dos passos 7 e 8, então, para que o coeficiente de correlação possa ser calculado, é aplicado um método de compressão que utiliza memória para equiparar a quantidade de pontos em cada par de segmentos da trajetória de referência e da trajetória comparada, respectivamente. Essa memória é o número de pontos que será analisado em cada segmento, dessa maneira o tamanho da memória é definido pelo tamanho (em pontos) do menor dos dois segmentos correspondentes na trajetória de referência e na comparada. O método de compressão utilizado foi o SQUISH, apresentado na seção 2.1.5.2;

10. **Aplicação do coeficiente de correlação estatística:** Passadas as verificações feitas nos passos anteriores e de posse dos vetores de deflexão, é possível calcular o coeficiente de correlação estatística aplicando os vetores $\overrightarrow{DF_{ref}}$ e $\overrightarrow{DF_{conj_i}}$ da trajetória de referência T_{ref} e da *i-ésima* trajetória do conjunto T_{conj}. Neste trabalho, utilizou-se o coeficiente de correlação de Pearson (r), conforme descrito na seção 2.3.1.1. Para serem consideradas similares as trajetórias comparadas devem ter um coeficiente de correlação $\geq$ a 0.70, com um nível de significância $\alpha \leq$ a 0.05.

As etapas mencionadas podem ser assim listadas conforme a Figura 7.

Figura 7 – Etapas do método de identificação de similaridade de forma



Fonte: produção do próprio autor

4.2 Implementação do Método

Para a implementação do método foram utilizadas as linguagens Java e R. A escolha da primeira foi motivada pelo fato de que ela possui as bibliotecas de programação necessárias para

realizar a conexão com o banco de dados, assim como realizar as segmentações, e cálculos dos azimutes, deflexões e comprimentos dos segmentos das trajetórias. Já a segunda foi escolhida para o cálculo das correlações e análises estatísticas por ser uma das ferramentas mais utilizadas neste campo.

Java é uma linguagem especificada para lidar com os desafios no desenvolvimento de aplicações em contextos diversos, além de possibilitar a execução de aplicações em ambientes distribuídos (GOSLING; MCGILTON, 1995). Ela também é largamente utilizada para conexão com gerenciadores de banco de dados e contém uma ampla gama de bibliotecas para lidar com processamento de arquivos texto e outros formatos (GRISS et al., 2012; REESE, 2000).

A linguagem R, por sua vez, é uma linguagem para análise estatística de dados que possui um vasta quantidade de bibliotecas que possibilitam desde a abertura e manipulação de arquivos, até a plotagem de gráficos. Suas principais vantagens estão relacionadas à geração de código portátil que pode ser utilizado em vários sistemas operacionais, e a eficiência no gerenciamento de memória e escopo por seção de trabalho (IHAKA; GENTLEMAN, 1996). Tanto Java quanto R estão classificadas entre as mais utilizadas pelo índice que mede a utilização de linguagens de programação chamado TIOBE (TIOBE, 2016).

As próximas subseções descrevem a implementação das etapas do método visto na seção 4.1 utilizando as tecnologias aqui apresentadas.

4.2.1 Implementação das Segmentações, Cálculos de Azimutes, Deflexões e Comprimentos dos Segmentos

A implementação dos segmentadores das trajetórias foi feita com base nos pseudocódigos apresentados na seção 2.1.5.3, já os cálculos dos azimutes e das deflexões foram implementados conforme as equações apresentadas nas seções 2.2.1 e 2.2.2, respectivamente. Esses pseudocódigos e equações foram transformados em algoritmos da linguagem Java, assim como todos os outros processos e verificações contidos nas etapas de 1 a 9 do método

proposto na Figura 7. Por fim, se todas as condições forem satisfeitas, então é gerado um arquivo CSV (*comma-separated values*) com as deflexões das trajetórias analisadas. Esse arquivo é disponibilizado para a execução da etapa 10 com uso na linguagem R.

4.2.2 Implementação do Método Estatístico de Cálculo de Correlação

A implementação das correlações foi feita com base no coeficiente de Pearson conforme a seção 2.3.1.1. Esses coeficientes foram calculados por meio da utilização de bibliotecas estatísticas presentes na linguagem R aplicadas aos dados disponibilizados em arquivo CSV. Os resultados foram arquivados em formato de texto para posterior análise e comparação com a classificação do grau de correlação apresentado na seção 2.3.1.1.

4.3 Projeto do Esquema de Dados

Essa seção apresenta o detalhamento do projeto do esquema de dados usado na pesquisa. Na seção seguinte é descrita a forma da carga das informações para este modelo (4.3.1).

O esquema de dados relacional utilizado para dar suporte aos testes do método foi concebido para permitir o armazenamento de trajetórias e de seus segmentos, assim como dos azimutes e deflexões dos segmentos. Conforme Hoberman (2014), projeto de dados é o processo de aprendizagem sobre os dados e o esquema de dados é o resultado final do processo de modelagem. O esquema de dados concebido neste trabalho é composto pelas seguintes entidades:

1. **Entidade Trajetória:** Trajetórias têm relacionamento com os pontos que as constituem, sendo que a trajetória tem também relação com o usuário que a gerou;

2. **Entidade Ponto:** Os pontos, por sua vez, têm relação com os segmentos e trajetórias com os quais estão atrelados;

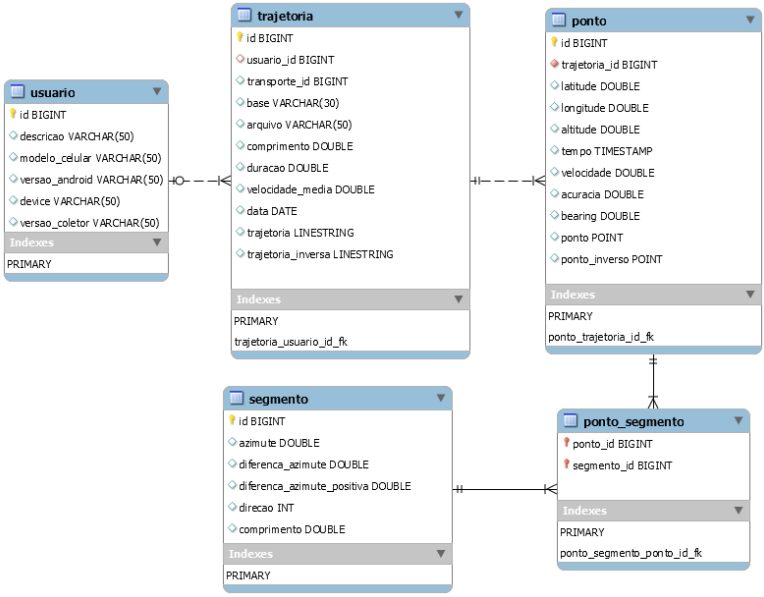
3. **Entidade Segmento:** Já os segmentos têm relação com os pontos dos quais são compostos;

4. **Entidade Ponto e Segmento:** Entidade que representa o relacionamento N para N entre pontos e segmentos;

5. **Entidade Usuário:** Entidade geradora das informações da trajetória.

A Figura 8 apresenta o esquema de dados descrito acima.

Figura 8 – Esquema de dados utilizado pelo método proposto



Fonte: produção do próprio autor

4.3.1 Implementação da Carga de Dados

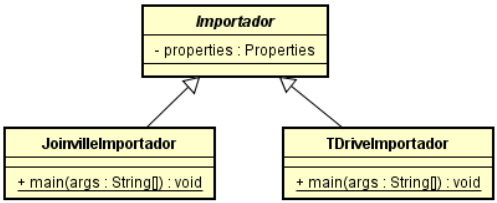
O processo de carga de dados é a etapa responsável pela leitura dos dados das bases de teste, tradução para o formato do esquema de dados do trabalho e armazenamento deles na banco

de dados utilizado nos experimentos. Esse processo de carga é equivalente ao processo de ETL (do inglês, *Extract-Transform-Load*) utilizado nos processos de descoberta de conhecimento em bancos de dados.

Para armazenar os dados das trajetórias utilizadas neste trabalho optou-se pelo banco de dados relacional PostgreSQL com a extensão PostGIS. PostgreSQL é um banco de dados relacional que não possui nativamente suporte ao armazenamento de dados geoespaciais. Entretanto, isto é compensado pela extensão PostGIS (PIÓRKOWSKI, 2011). As classes Java que representam as tabelas do esquema de dados foram mapeadas via uso da ferramenta ORM Hibernate (HIBERNATE, 2016). Dessa maneira, funções de criação de esquema (DDL) e de manipulações dos dados (DML) se tornaram transparentes para o desenvolvimento.

A Figura 9 mostra o diagrama de classes UML que representa a organização das classes implementadas para importar as trajetórias.

Figura 9 – Importadores de dados



Fonte: produção do próprio autor

Para facilitar a importação de outras bases de dados, se necessário, o programa foi projetado utilizando uma classe abstrata e especializações para cada uma das bases do trabalho. Este projeto foi implementado com o uso do padrão de projeto *Abstract Factory*.

5

EXPERIMENTOS, RESULTADOS E DISCUSSÕES

Este Capítulo apresenta os experimentos realizados para verificar a aplicabilidade do método proposto e, por consequência, a hipótese de pesquisa. Os testes foram realizados sobre um subconjunto de dados das bases dos projetos Joinville e *T-Drive*, que serão descritos nas seções 5.1.1 e 5.1.2, respectivamente. As informações fornecidas por essas bases foram armazenadas no esquema de dados apresentado na seção 4.3. Os dados contidos nessas bases compreendem uma diversificada gama de trajetórias que abrangem registros de deslocamento de carro ou táxi, tanto em perímetro urbano quanto em autoestradas. Grande parte das trajetórias tem mais de 1 km de comprimento e é composta por algumas centenas de pontos. Todavia, independentemente da quantidade de pontos, todas as trajetórias contêm as informações necessárias para realização dos experimentos desta pesquisa.

Os cenários selecionados para os experimentos foram escolhidos de forma a poder demonstrar como o método proposto se comporta na identificação de similaridade entre trajetórias com diferentes formatos. Portanto, foram construídos os seguintes cenários: 1) cenário composto por trajetórias próximas com formas predominantemente retas; 2) cenário composto por trajetórias próximas contendo retas e curvas; 3) cenário composto por traje-

tórias longas e com diferentes direções; 4) cenário composto por trajetórias distantes e de diferentes *Datasets*. As próximas subseções apresentam detalhadamente os experimentos realizados em cada um destes três cenários.

O coeficiente de correlação de Pearson foi adotado para os testes, uma vez que os dados apresentaram as características descritas na seção 2.3.1.1, que se aplicam ao coeficiente mencionado. Este, então, foi adotado como método estatístico para verificar a hipótese de pesquisa sobre similaridade de forma das trajetórias ao longo dos testes. Outro fator utilizado para realização dos testes foi a adoção da trajetória com a menor quantidade de pontos como de referência, visando assim a diminuição do tempo de processamento entre as comparações.

5.1 Bases de Dados Utilizadas nos Testes

Como base no objetivo do trabalho e após uma ampla pesquisa na literatura relacionada, os dois *datasets* selecionados para a realização dos experimentos são: Projeto Geração Automática de Mapas Rodoviários Digitais e T-Drive. As próximas subseções detalham cada um destes *datasets*.

5.1.1 Projeto Geração Automática de Mapas Rodoviários Digitais

Este projeto teve por objetivo apresentar um método capaz de gerar automaticamente mapas rodoviários digitais por meio de uma abordagem baseada na utilização de algoritmos genéticos (COSTA; BALDO, 2015). Para a realização deste trabalho foi feita a coleta, por meio de aparelhos de *smartphone* munidos de GPS, de uma base de dados própria que continha trajetórias de deslocamentos realizados na cidade de Joinville (SC) e regiões adjacentes. Essa base foi coletada no período entre os anos de 2012 e 2013 e envolveu cerca de 17 usuários. Ao total, foram coletadas 4.715 trajetórias com informações de latitude,

longitude, altura, velocidade, data e hora de cada ponto coletado.

5.1.2 Projeto T-Drive

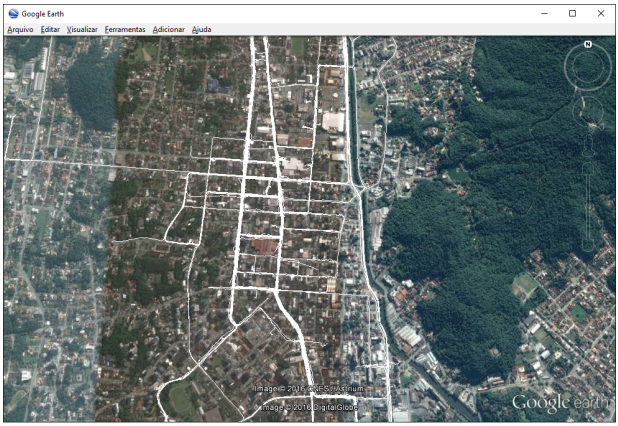
O projeto *T-Drive* é uma iniciativa da empresa Microsoft realizado em Pequim (China) que teve como propósito estudar o comportamento das trajetórias de taxistas. O objetivo era assimilar o conhecimento dos taxistas a fim de ajudar os motoristas comuns a escolher o melhor caminho baseado no horário de saída e no local de destino (YUAN et al., 2011). Esse projeto disponibilizou uma base com trajetórias de 10.357 táxis, coletadas durante o período de 2 a 8 de fevereiro de 2008, com 15 milhões de pontos e um total de 9 milhões de quilômetros percorridos (YUAN et al., 2010). Os dados contidos nessas trajetórias incluem: latitude, longitude, tempo (formato *timestamp*) e identificador do táxi. Esse projeto é parte do projeto Geolife (ZHENG et al., 2012).

5.2 Cenário 1 – Trajetórias Próximas com Formas Predominantemente Retas

O primeiro cenário é composto por trajetórias registradas em vias do centro da cidade de Joinville - Santa Catarina. Esse cenário conta conta com 162 trajetórias que passam por esta área, contendo um total de 161.280 pontos. A Figura 10 mostra a visão geral das trajetórias desse cenário.

O tamanho médio das trajetórias analisadas nesse cenário foi de 965 pontos, onde a menor trajetória tem 619 pontos e a maior 1.263 pontos, com uma média de extensão de 2.500 metros. A trajetória de referência selecionada foi a menor presente nesse conjunto, ou seja, a trajetória com 619 pontos. O conjunto de trajetórias a serem comparadas é composto por 10 trajetórias selecionadas dentre as 162 contidas no cenário, sendo que elas também se desenvolvem no sentido norte-sul da cidade, e se encontravam próximas da trajetória de referência.

Figura 10 – Trajetórias do Cenário 1 - Centro de Joinville



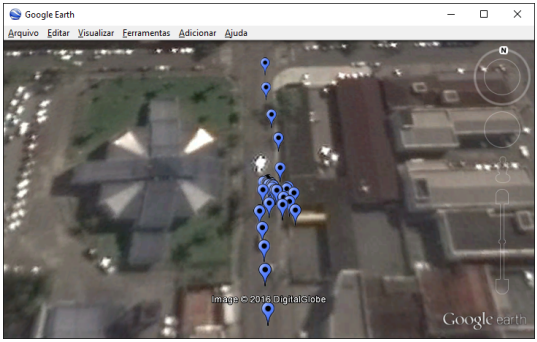
Fonte: produção do próprio autor

A etapa de limpeza envolveu a trajetória de referência e as 10 trajetórias selecionadas para a comparação, sendo que esse processo teve como resultado a eliminação de, em média, 66,39% dos pontos das trajetórias. A elevada eliminação de pontos nesse processo pode ser explicada por dois fatores. O primeiro é a quantidade de pontos muito próximos, ocorrendo possivelmente devido a paradas em semáforos, e do deslocamento lento do veículo dado que nesta região existe um tráfego pesado por ela estar situada na região central da cidade, onde acontecem frequentemente engarrafamentos. Outro fator possivelmente se deve à existência de pontos coletados com baixa precisão, dado a grande presença de prédios na região. As Figuras 11 e 12 mostram parte da trajetória de referência antes e depois da limpeza.

As etapas de segmentação, cálculo de azimutes e deflexões tiveram como intuito dividir as trajetórias em segmentos que representassem uma orientação/direção distinta, sendo elas: reta, curva à direita ou curva à esquerda. Nesse cenário, por apresentarem formas retas, após o processo de segmentação as trajetórias não foram divididas, sendo, portanto, representadas por um único

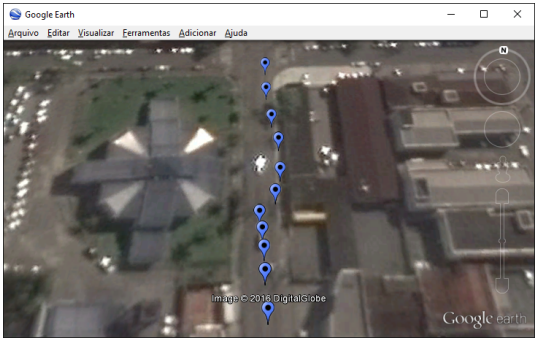
segmento classificado como reta.

Figura 11 – Parte da trajetória de referência antes da limpeza



Fonte: produção do próprio autor

Figura 12 – Parte da trajetória de referência depois da limpeza



Fonte: produção do próprio autor

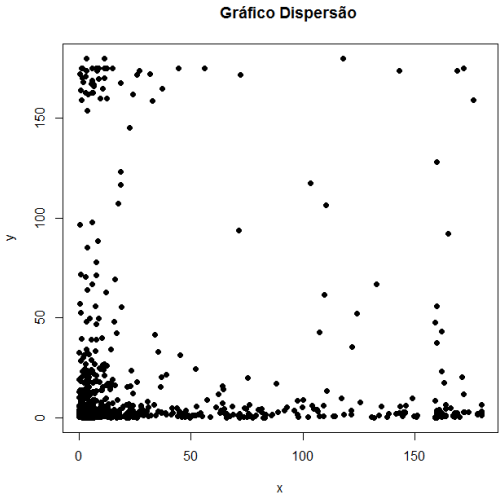
Finalizada a segmentação, sabendo que as trajetórias contêm apenas um segmento classificado como reta, observa-se que elas satisfazem a condição do passo 8 do método que trata da quantidade e direção dos segmentos. Após a conclusão da segmentação, efetuou-se a análise do comprimento de cada segmento. Nesse cenário, por apresentarem a característica de um segmento

classificado como reta, todas as trajetórias apresentavam diferença de comprimento inferior a 25% em relação à trajetória de referência.

Seguindo a execução do método, na etapa de compressão a percentagem de pontos eliminados ocorreu devido ao ajuste de normalização da quantidade de pontos por segmento, entre a trajetória de referência e as outras analisadas. Com isso, essa etapa do método eliminou em média 23,81% dos pontos das trajetórias para este cenário. Após as etapas de processamento do método restaram aproximadamente 9,80% dos pontos de cada trajetória, estes pontos então foram utilizados para aplicação do coeficiente de correlação e aferição da similaridade de forma.

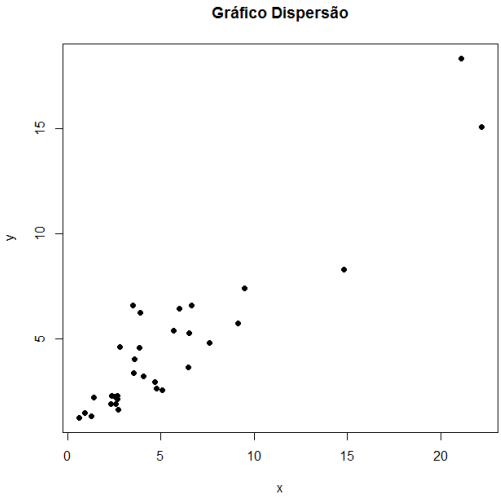
As Figuras 13 e 14 mostram, respectivamente, o diagrama de dispersão dos dados antes e depois dos processos de limpeza, segmentação e compactação. Esses dados representam o teste de número 6 constante na Tabela 1. Esse teste foi escolhido por apresentar o maior valor de correlação para ambos os coeficientes.

Figura 13 – Gráfico de dispersão dos dados antes da limpeza



Fonte: produção do próprio autor

Figura 14 – Gráfico de dispersão dos dados depois da limpeza



Fonte: produção do próprio autor

A próxima etapa do método envolve o cálculo dos coeficientes de correlação. A Tabela 1 mostra os valores aferidos para cada um dos 10 testes realizados. Para cada teste realizado as colunas da tabela apresentam: o número do teste, o identificador da trajetória de referência e da trajetória comparada, e os valores aferidos para os coeficientes de Pearson(p) e de p-valor.

Conforme descrito a Tabela 1 lista os resultados aferidos neste cenário para os coeficientes de Pearson (p) e o p-valor correspondente. Os resultados obtidos neste cenário alcançaram valores para o coeficiente de correlação de Pearson entre 0.70 e 0.92. Os coeficientes aferidos são significativos e diferentes de zero. Após a limpeza e compressão as trajetórias ficaram em média com tamanho de 73 pontos e este foi o tamanho médio das trajetórias utilizadas para a aplicação dos coeficientes de correlação. Isso explica por que o gráfico de dispersão da Figura 14 tem muito menos pontos que o gráfico de dispersão da Figura 13.

Tabela 1 – Resultados de correlação do cenário 1

Teste 1 Traj. 89 Traj. 17		Teste 2 Traj. 89 Traj. 35		Teste 3 Traj. 89 Traj. 42		Teste 4 Traj. 89 Traj. 55		Teste 5 Traj. 89 Traj. 59		Teste 6 Traj. 89 Traj. 64		Teste 7 Traj. 89 Traj. 68		Teste 8 Traj. 89 Traj. 69		Teste 9 Traj. 89 Traj. 87		Teste 10 Traj. 89 Traj. 116	
r	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*
0.71	1.327e-14	0.80	2.2e-16	0.72	2.2e-16	0.78	9.561e-13	0.71	5.934e-12	0.92	2.282e-14	0.71	1.47e-15	0.70	3.578e-09	0.79	1.734e-10	0.72	1.543e-15

Fonte: produção do próprio autor

Ainda sobre os resultados, sabendo que as trajetórias presentes neste experimento representam deslocamentos de veículos em uma mesma rua, esperava-se que os coeficientes de correlação alcançassem valores muito próximos a 1. Esses resultados podem ser explicados usando como referência a fórmula de cálculo da correlação de Pearson, que estabelece que a correlação é a razão entre a covariância e o desvio-padrão das variáveis comparadas. Portanto, baseado nessa fórmula, pode-se perceber que quanto maior for a covariância e menor for o desvio-padrão, maior será a correlação entre as variáveis. Entretanto, como pode ser visto na Figura 14, a covariância entre as variáveis é pequena, dado que as deflexões alcançam valores de 0° a 25°, dentro de um gama de 0° a 180° valores possíveis. Assim, qualquer desvio-padrão acima de 5° diminui significativamente o valor do coeficiente de correlação.

5.3 Cenário 2 – Trajetórias Próximas Contendo Retas e Curvas

O segundo cenário de teste é composto por trajetórias registradas em um trecho da BR-101 que dá acesso ao distrito industrial da cidade de Joinville (Santa Catarina). Este cenário conta com 95 trajetórias que passam por essa área, contendo um total de 91.943 pontos. A Figura 15 mostra a visão geral das trajetórias deste cenário.

O tamanho médio das trajetórias analisadas neste cenário foi de 1.060 pontos, onde a menor trajetória tem 725 pontos e a maior 1.196 pontos, com uma média de extensão de 4.483 metros. A trajetória de referência selecionada foi a menor presente nesse conjunto, ou seja, a trajetória de 725 pontos. O conjunto de trajetórias a serem comparadas é composto por 10 trajetórias selecionadas dentre as 95 contidas no cenário, sendo que elas se encontravam próximas da trajetória de referência.

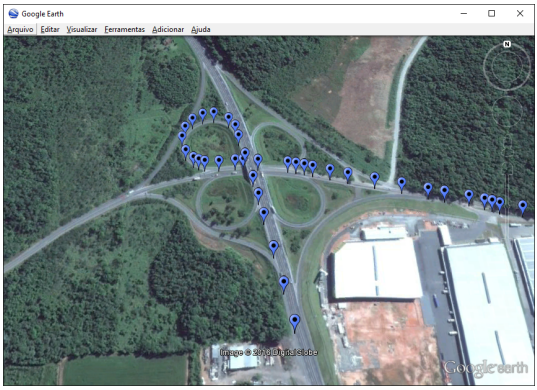
Figura 15 – Trajetórias do Cenário 2 - Distrito Industrial de Joinville

Fonte: produção do próprio autor

A etapa de limpeza envolveu a trajetória de referência e as 10 trajetórias selecionadas para a comparação, sendo que esse processo teve como resultado a eliminação de, em média, 84% dos pontos das trajetórias. A alta porcentagem de pontos eliminados se deve principalmente à grande quantidade de pontos com baixa precisão ou com pouca relevância na identificação da forma, dado que parte significativa das trajetórias representam uma linha reta. As Figuras 16 e 17 mostram parte da trajetória de referência antes e depois da limpeza.

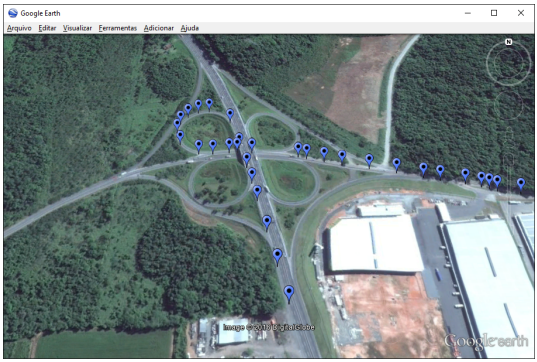
As etapas de segmentação, cálculo de azimutes e deflexões tiveram como intuito dividir as trajetórias em segmentos que representassem uma orientação/direção distinta. Nesse cenário, em específico, as trajetórias apresentavam curvas intercaladas por segmentos de reta, fazendo com que fosse possível identificar vários segmentos com orientação/direção diferentes. A Figura 18 mostra parte da segmentação da trajetória de referência, onde os segmentos em azul representam retas e o em vermelho representa uma curva à direita.

Figura 16 – Parte da trajetória de referência antes da limpeza



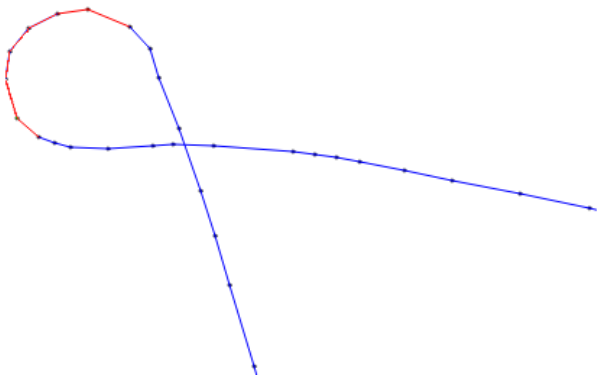
Fonte: produção do próprio autor

Figura 17 – Parte da trajetória de referência depois da limpeza



Fonte: produção do próprio autor

Figura 18 – Segmentação da trajetória de referência



Fonte: produção do próprio autor

Após a conclusão da segmentação e comparação da quantidade e direção dos segmentos, conforme definido no passo 8 do método, efetuou-se a análise do comprimento de cada segmento. Todas as 10 trajetórias analisadas apresentavam segmentos com diferença de comprimento inferior a 25% em relação à trajetória de referência.

Durante a etapa de compressão foram eliminados pontos a fim de ajustar a quantidade de pontos por segmento entre a trajetória de referência e as outras trajetórias comparadas. Essa etapa do método eliminou em média 5,37% dos pontos das trajetórias contidas no cenário. Após as etapas de processamento do método, restaram aproximadamente 10,63% dos pontos de cada trajetória, estes pontos então foram utilizados para aplicação do coeficiente de correlação e aferição da similaridade de forma.

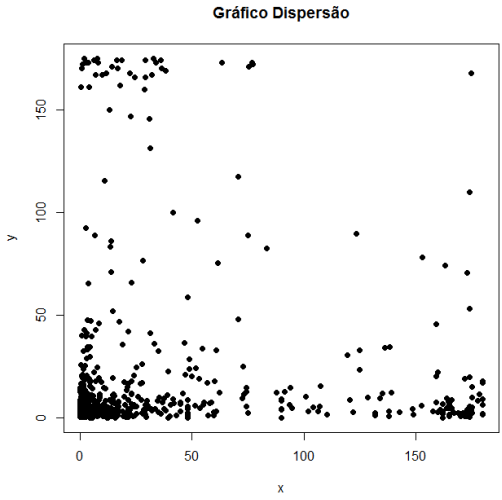
As Figuras 19 e 20 mostram, respectivamente, o diagrama de dispersão dos dados antes e depois dos processos de limpeza, segmentação e compressão. Esses dados representam o teste de número 9 constante na Tabela 2. Esse teste foi escolhido por apresentar o maior valor de correlação para ambos os coeficientes.

A próxima etapa do método envolve o cálculo dos coeficientes de correlação. A Tabela 2 mostra os valores aferidos para cada um dos 10 testes realizados. Para cada teste realizado as colunas da tabela apresentam: o número do teste, o identificador da trajetória de referência e da trajetória comparada, e os valores aferidos para os coeficientes de Pearson (p) e de p-valor.

Conforme descrito a Tabela 2 lista os resultados aferidos neste cenário para os coeficientes de Pearson (p) e o p-valor correspondente. Os resultados obtidos neste cenário alcançaram valores para o coeficiente de correlação de Pearson entre 0.78 e 0.93. Os coeficientes aferidos são significativos e diferentes de zero.

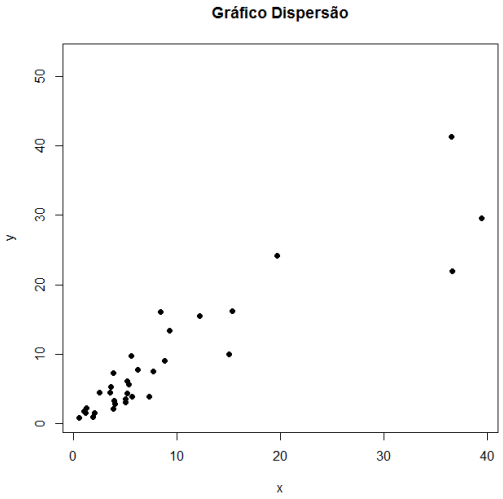
Após a limpeza e compressão as trajetórias ficaram em média com tamanho de 40 pontos e este foi o tamanho médio das trajetórias utilizadas para a aplicação dos coeficientes de correlação. Isso explica por que o gráfico de dispersão da Figura 20 tem muito menos pontos que o gráfico de dispersão da Figura 19.

Figura 19 – Gráfico de dispersão dos dados antes da limpeza



Fonte: produção do próprio autor

Figura 20 – Gráfico de dispersão dos dados depois da limpeza



Fonte: produção do próprio autor

Sobre os resultados, é possível observar que este cenário obteve valores de correlação maiores que os apresentados no cenário 1 (5.2). Esses resultados podem ser explicados fazendo uma analogia com a explicação apresentada no cenário 1. Como a correlação é dada pela razão entre a covariância e o desvio-padrão das variáveis comparadas, percebe-se pelo gráfico de dispersão da Figura 20 que a covariância entre as variáveis do cenário 2 é maior que a apresentada no cenário 1, porque nele há uma curva que faz com que as deflexões alcancem valores entre 0° e 50°. Assim, um desvio-padrão moderado não influencia significativamente no valor do coeficiente de correlação.

Tabela 2 – Resultados de correlação do cenário 2

Teste 1 Traj. 8 Traj. 1		Teste 2 Traj. 8 Traj. 4		Teste 3 Traj. 8 Traj. 6		Teste 4 Traj. 8 Traj. 9		Teste 5 Traj. 8 Traj. 13		Teste 6 Traj. 8 Traj. 17		Teste 7 Traj. 8 Traj. 19		Teste 8 Traj. 8 Traj. 22		Teste 9 Traj. 8 Traj. 24		Teste 10 Traj. 8 Traj. 26	
r	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*
0.88	6.977e-16	0.90	2.2e-16	0.92	1.304e-13	0.86	5.438e-15	0.84	2.297e-09	0.93	1.044e-13	0.86	4.052e-11	0.89	2.2e-16	0.88	1.437e-11	0.78	1.296e-05

Fonte: produção do próprio autor

5.4 Cenário 3 – Cenário Composto por Trajetórias Longas e em Diferentes Direções

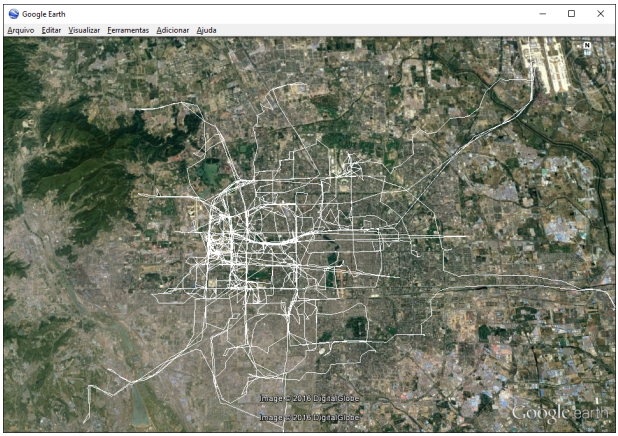
O terceiro cenário é composto por trajetórias registradas em vias da cidade Pequim (China) disponibilizadas pelo projeto *T-Drive*. Esse projeto é uma iniciativa da empresa Microsoft para analisar o comportamento das trajetórias dos taxistas que operam na referida cidade. Esse cenário contém 423 trajetórias com um total de 1.318.360 pontos. A Figura 21 mostra a visão geral das trajetórias desse cenário.

O tamanho médio das trajetórias analisadas nesse cenário foi de 2.870 pontos, onde a menor trajetória tem 2.405 pontos e a maior 3.011 pontos, com uma média de extensão de 18.564 metros. A trajetória de referência selecionada foi a menor presente nesse conjunto, ou seja, a trajetória com 2.405 pontos. O conjunto de trajetórias a serem comparadas é composto por 10 trajetórias selecionadas dentre as 423 contidas no cenário, sendo que elas se encontravam na mesma região, entretanto a uma distância da trajetória de referências entre 200 m e 500 m.

A etapa de limpeza envolveu a trajetória de referência e as 10 trajetórias selecionadas para a comparação, sendo que esse processo teve como resultado a eliminação de, em média, 83,72% dos pontos. Assim como no cenário 1, a alta eliminação de pontos nesse cenário se deve principalmente à grande quantidade de pontos muito próximos e com baixa precisão. As Figuras 22 e 23 mostram parte da trajetória de referência antes e depois da limpeza.

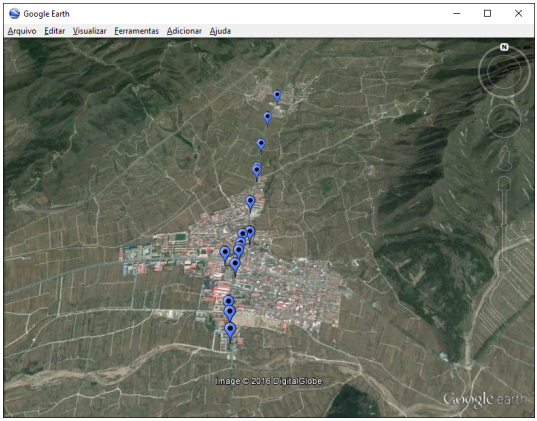
As etapas de segmentação, cálculo de azimutes e deflexões realizaram a divisão das trajetórias em segmentos de acordo com suas direções. A Figura 24 mostra parte da segmentação da trajetória de referência, onde é possível notar 2 segmentos de reta em azul e 1 de curva à esquerda em verde.

Figura 21 – Trajetórias do Cenário 3 - Pequim (China)



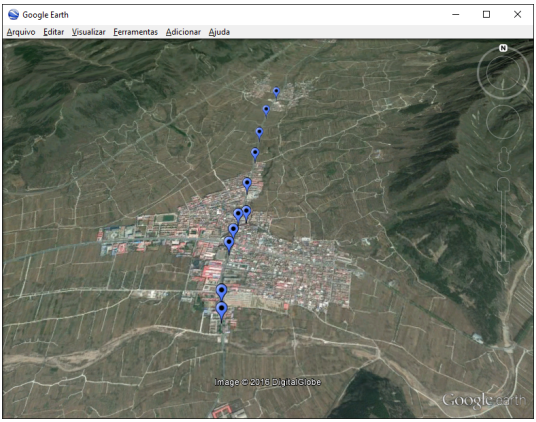
Fonte: produção do próprio autor

Figura 22 – Parte da trajetória de referência antes da limpeza



Fonte: produção do próprio autor

Figura 23 – Parte da trajetória de referência depois da limpeza



Fonte: produção do próprio autor

Figura 24 – Segmentação da trajetória de referência

Fonte: produção do próprio autor

Após a conclusão da segmentação e comparação da quantidade e direção dos segmentos, conforme definido no passo 8 do método, efetuou-se a análise do comprimento de cada segmento. Todas as 10 trajetórias analisadas apresentavam segmentos com diferença de comprimento inferior a 25% em relação à trajetória de referência.

Na etapa de compreensão eliminou em média 12,09% dos pontos das trajetórias. Após as etapas de processamento do método, restaram aproximadamente 4,19% dos pontos de cada trajetória, esses pontos então foram utilizados para aplicação do coeficiente de correlação e aferição da similaridade de forma.

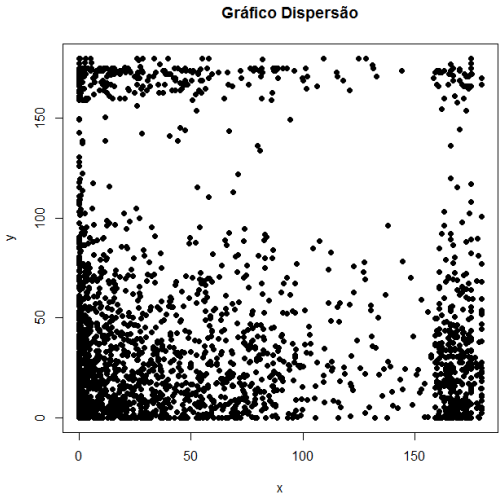
As Figuras 25 e 26 mostram o diagrama de dispersão dos dados antes e depois dos processos de limpeza, segmentação e compressão. Esses dados representam o teste de número 10 constante na Tabela 3. Esse teste foi escolhido por apresentar o maior valor de correlação para ambos os coeficientes.

A próxima etapa do método envolve o cálculo dos coeficientes de correlação. A Tabela 3 mostra os valores aferidos para cada um dos 10 testes realizados. Para cada teste as colunas da tabela apresentam: o número do teste, o identificador da trajetória de referência e de comparação, e os valores aferidos para os coeficientes de Pearson (p) e de p-valor.

Conforme descrito a Tabela 3 lista os resultados aferidos nesse cenário para os coeficientes de Pearson (p) e o p-valor correspondente. Os resultados obtidos nesse cenário alcançaram valores para o coeficiente de correlação de Pearson entre 0.77 e 0.91. Os coeficientes aferidos são significativos e diferentes de zero.

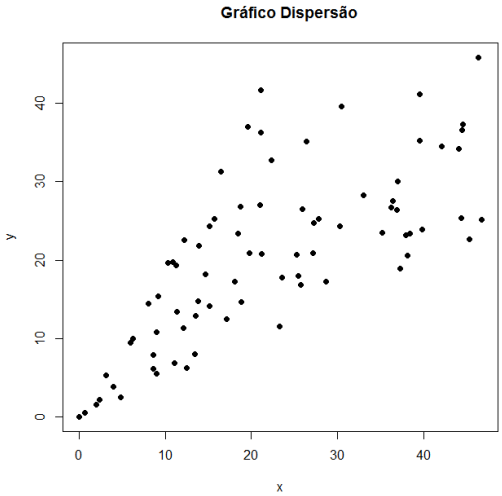
Após a limpeza e compressão as trajetórias ficaram em média com tamanho de 120 pontos e este foi o tamanho médio das trajetórias para a aplicação dos coeficientes de correlação. Isso explica por que o gráfico de dispersão da Figura 25 tem muito menos pontos que o gráfico de dispersão da Figura 26.

Figura 25 – Gráfico de dispersão dos dados antes da limpeza



Fonte: produção do próprio autor

Figura 26 – Gráfico de dispersão dos dados depois da limpeza



Fonte: produção do próprio autor

Tabela 3 – Resultados de correlação do cenário 3

Teste 1 Traj. 3 Traj. 6		Teste 2 Traj. 3 Traj. 8		Teste 3 Traj. 3 Traj. 10		Teste 4 Traj. 3 Traj. 12		Teste 5 Traj. 3 Traj. 15		Teste 6 Traj. 3 Traj. 16		Teste 7 Traj. 3 Traj. 17		Teste 8 Traj. 3 Traj. 18		Teste 9 Traj. 3 Traj. 19		Teste 10 Traj. 3 Traj. 20	
r	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*	*r*	*p-valor*
0.81	2.2e-16	0.85	2.2e-16	0.78	2.2e-16	0.80	2.019e-15	0.80	2.2e-16	0.77	2.2e-16	0.82	2.2e-16	0.81	2.2e-16	0.90	2.2e-16	0.91	2.2e-16

Fonte: produção do próprio autor

Sobre os valores alcançados nesse cenário, estes se mostraram melhores que os obtidos no cenário 1 (seção 5.2), pois, assim como o cenário 2 (seção 5.3), ele contém trajetórias com segmentos de retas e curvas. Isso fez com que as deflexões alcancem valores maiores (como visto na Figura 26), o que permite que pequenos desvios não interfiram significativamente no valor da correlação.

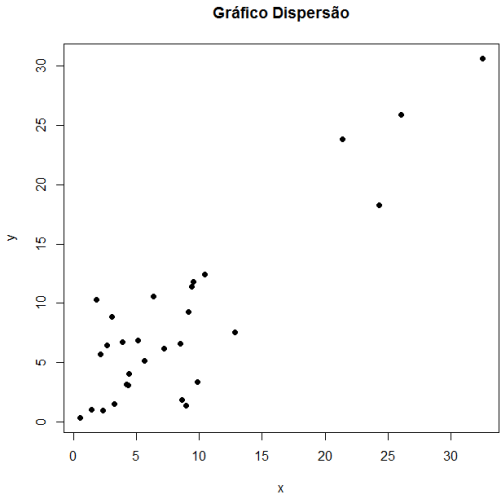
5.5 Cenário 4 - Trajetórias Distantes e de *Datasets* Diferentes

Esse cenário envolveu a identificação de similaridade entre trajetórias distantes umas das outras. Portanto, para este experimento foram selecionadas três trajetórias do cenário 1 (5.2) e três trajetórias do cenário 3 (5.4) já devidamente limpas e segmentadas. Para esse conjunto de três trajetórias de cada cenário foram realizados três testes conforme resultados apresentados na Tabela 4.

As três trajetórias selecionadas do cenário 1 representam deslocamentos no sentido geográfico norte-sul, já as três selecionadas do cenário 3 representam deslocamentos do sentido leste-oeste. Esse critério de seleção foi utilizado no intuito de demonstrar se o método é aplicável a trajetórias distantes e em sentidos diferentes. Os trechos selecionados dessas trajetórias tinham em média 500 metros, com uma amostragem média de 29 pontos após a etapa de compactação. A Figura 27 mostra o diagrama de dispersão dos dados do teste de número 1 da Tabela 4. Esse teste foi escolhido por apresentar o melhor resultado para a correlação de Pearson dentre os três realizados.

A Tabela 4 mostra os valores de correlação aferidos nos testes após todas as etapas do método. Para cada teste realizado as colunas da tabela apresentam: o número do teste, o identificador da trajetória de referência e de comparação, e os valores aferidos no coeficiente de Pearson (p) e de p-valor.

Figura 27 – Gráfico de dispersão do Cenário 4



Fonte: produção do próprio autor

Tabela 4 – Resultados de correlação do cenário 4

Teste 1 Traj. 89 Traj. 3		Teste 2 Traj. 17 Traj. 10		Teste 3 Traj. 12 Traj. 35	
r	*p-valor*	*r*	*p-valor*	*r*	*p-valor*
0.80	1.928e-08	0.73	0.0003808	0.74	0.0007609

Fonte: produção do próprio autor

Conforme descrito a Tabela 4 lista os resultados aferidos nesse cenário para os coeficientes de Pearson (p) e o p-valor correspondente. Os resultados obtidos nesse cenário alcançaram valores para o coeficiente de correlação de Pearson entre 0.73 e 0.80. Os coeficientes aferidos são significativos e diferentes de zero.

As trajetórias ao final do processo tinham um tamanho de médio de 29 pontos. Os níveis de correlação observados podem ser justificados com base na Figura 27, onde a dispersão apresenta uma forma de tendência linear.

5.6 Avaliação Comparativa

Esta seção apresenta uma comparação entre o método proposto neste trabalho e outra proposta semelhante de cálculo de similaridade. Para a comparação aqui realizada foi escolhido um dos métodos da revisão de trabalhos relacionados apresentada no Capítulo 3. No caso o método *Dynamic Time Warping* (DTW), apresentado na seção 3.1 foi o escolhido. A escolha desse método de comparação se deu para verificar, quando o método proposto nesta pesquisa encontra similaridade de forma, se isto é confirmado por outro método. No caso do DTW este usa a distância para indicar similaridade. Para realizar essa comparação foram escolhidos o cenário 1 (seção 5.2) e o cenário 2 (seção 5.3). A escolha desses cenários se deu pela razão de que eles apresentam trajetórias com formas consideravelmente distintas, onde o primeiro contém trajetórias predominantemente retas e o segundo contém trajetórias com curvas acentuadas. Entende-se que com esses dois cenários é possível cobrir os tipos mais comuns de trajetórias encontradas nos *datasets* utilizados nos experimentos.

Assim, para o cálculo de similaridade pela aplicação de coeficientes de correlação as deflexões angulares foram utilizadas. Para o cálculo de similaridade por meio do DTW foram utilizadas a posições de latitude e longitude dos pontos das trajetórias, com os pontos restantes após o processamento de limpeza e compactação. As Tabelas 5 e 6 mostram os valores aferidos via DTW aplicado sobre as deflexões angulares das trajetórias utilizadas nos cenários 1 e 2, respectivamente. Em cada tabela as colunas representam: o número do teste, o identificador da trajetória de referência e de comparação, e os valores aferido via DTW.

Tabela 5 – Resultados DTW aplicados ao cenário 1

Teste 1 Traj. 89 Traj. 17	Teste 2 Traj. 89 Traj. 35	Teste 3 Traj. 89 Traj. 42	Teste 4 Traj. 89 Traj. 55	Teste 5 Traj. 89 Traj. 59	Teste 6 Traj. 89 Traj. 64	Teste 7 Traj. 89 Traj. 68	Teste 8 Traj. 89 Traj. 69	Teste 9 Traj. 89 Traj. 87	Teste 10 Traj. 89 Traj. 116
DTW	*DTW*	*DTW*	*DTW*	*DTW*	*DTW*	*DTW*	*DTW*	*DTW*	*DTW*
0.06	0.06	0.02	0.10	0.05	0.10	0.03	0.05	0.10	0.04

Fonte: produção do próprio autor

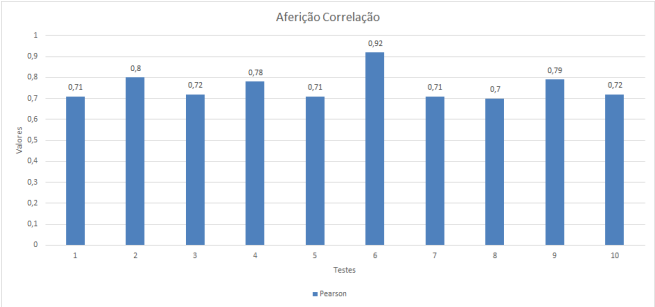
Tabela 6 – Resultados DTW aplicados ao cenário 2

Teste 1 Traj. 8 Traj. 1	Teste 2 Traj. 8 Traj. 4	Teste 3 Traj. 8 Traj. 6	Teste 4 Traj. 8 Traj. 9	Teste 5 Traj. 8 Traj. 13	Teste 6 Traj. 8 Traj. 17	Teste 7 Traj. 8 Traj. 19	Teste 8 Traj. 8 Traj. 22	Teste 9 Traj. 8 Traj. 24	Teste 10 Traj. 8 Traj. 26
DTW	*DTW*	*DTW*	*DTW*	*DTW*	*DTW*	*DTW*	*DTW*	*DTW*	*DTW*
0.16	0.15	0.13	0.13	0.16	0.16	0.16	0.15	0.11	0.15

Fonte: produção do próprio autor

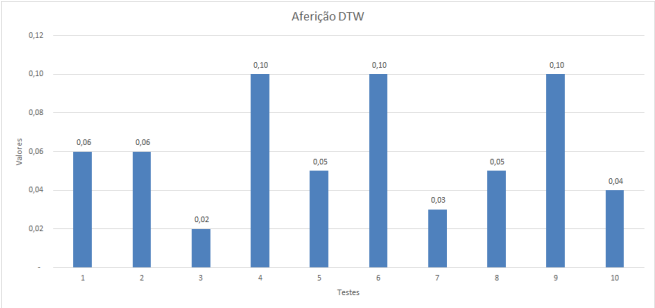
As Figuras 28 e 29 mostram os valores apurados pelo método proposto e pelo DTW para o cenário 1, respectivamente.

Figura 28 – Gráfico com valores de correlação para o Cenário 1



Fonte: produção do próprio autor

Figura 29 – Gráfico com valores do DTW para o Cenário 1

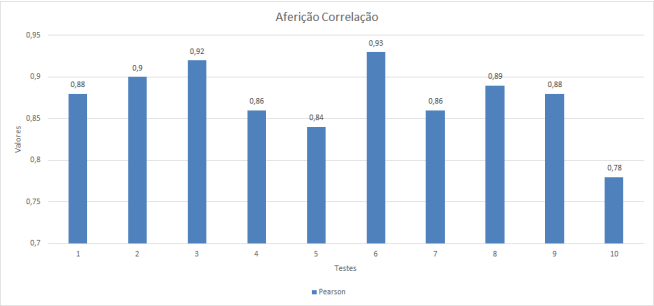


Fonte: produção do próprio autor

As Figuras 30 e 31 mostram os valores apurados pelo método proposto e pelo DTW para o cenário 2, respectivamente.

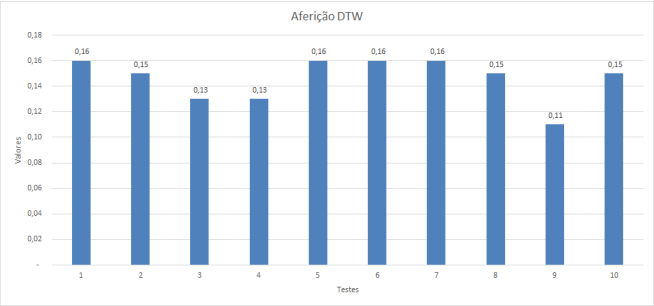
Antes de discutir os resultados é importante destacar que o método proposto e o DTW são inversamente proporcionais. Ou seja, para o método proposto quanto maior for o valor se aproximando de +1, melhor é o resultado. Já para o DTW, quanto

Figura 30 – Gráfico com valores de correlação para o Cenário 2



Fonte: produção do próprio autor

Figura 31 – Gráfico com valores do DTW para o Cenário 2



Fonte: produção do próprio autor

menor for o valor se aproximando de 0, melhor é o resultado. Tendo em mente essa relação inversamente proporcional entre os métodos comparados é possível realizar as análises.

Sobre o cenário 1, com base nos gráficos das Figuras 28 e 29 pode-se perceber que o método proposto obteve resultados significativos por apresentar valores de correlação acima de 0.70. Essa mesma situação é observada na aplicação do método DTW, pois ele teve como resultado valores considerados baixos, o que caracteriza uma alta semelhança entre as trajetórias.

Para o cenário 2, com base nos gráficos das Figuras 30

e 31, o método proposto se mostra melhor que o DTW, pois o primeiro alcançou valores de correlação acima de 0.78 o que significa alta similaridade, enquanto o segundo obteve valores acima de 0.11, o que caracteriza uma similaridade menor que a alcançada pelo primeiro.

5.7 Considerações sobre os Resultados

Com base nos experimentos apresentados nas seções 5.2, 5.3, 5.4 e 5.5, é possível perceber que o método proposto possibilita a identificação de similaridade entre trajetórias distantes e em direções diferentes. Além disso, ao se comparar o método proposto com um dos métodos apresentados na literatura (seção 5.6), é possível verificar que ele tem um resultado igual ou, em alguns casos, melhor que o alcançado pelo DTW, método utilizado na comparação. Isso reforça a ideia de que o método é capaz de alcançar resultados compatíveis aos obtidos por técnicas amplamente aplicadas na identificação de similaridade.

Entretanto, o método apresenta algumas limitações. Dentre essas limitações, duas merecem destaque. A primeira está relacionada à sensibilidade do método à presença de ruídos nos dados-fonte. A segunda diz respeito à performance na identificação de similaridade quando as trajetórias apresentam formato de retas, se comparado com a performance de identificação de similaridade quando elas apresentam retas e curvas. A primeira limitação pode ser contornada com uma abrangente etapa de pré-processamento que compreenda a limpeza e compressão das trajetórias, conforme aplicado no método proposto. Entretanto, mesmo depois da aplicação da etapa de limpeza que compreendeu a eliminação dos pontos de parada e dos pontos com baixa precisão, assim como a compressão e identificação das direções, ainda assim o método não alcançou resultados próximos ao máximo esperado, que é o valor de correlação 1. Isso significa que ainda há espaço para melhorar / aprimorar o pré-processamento das trajetórias a fim de alcançar valores mais próximos do ideal para trajetórias que são consideradas similares, *a priori*, conforme as

utilizadas nos experimentos realizados nos cenários 1 e 2.

Como visto, a primeira limitação apresenta algumas técnicas e abordagens para mitigação, entretanto, a segunda é uma situação mais complexa de ser resolvida. Ela está diretamente relacionada ao espectro de variação dos dados utilizados na aplicação da correlação. Conforme mencionado nas considerações sobre os experimentos do cenário 1, quanto maior for o espectro de variação dos dados da variável analisada, ou seja, a sua covariância, menor será a influência do desvio-padrão dessas variáveis no cálculo da correlação. Portanto, no cenário composto por retas, a variação dos valores de deflexão é pequena, na ordem de 2° a 10°, dentro de uma gama de 180° possíveis, o que potencializa o prejuízo causado por um ruído em uma deflexão de 5° que faz com que o desvio-padrão da variável aumente consideravelmente. Esse tipo de situação é minimizado em trajetórias que contêm curvas, pois elas apresentam uma covariância entre suas deflexões que podem chegar a valores entre 0° e 50°, conforme apresentado no cenário 2 (seção 5.3), o que diminui a influência de deflexões com ruídos, pois o aumento do desvio-padrão da variável não influencia significativamente no cálculo da correlação.

Apesar dessas limitações, o método conseguiu alcançar resultados que confirmam que ele é capaz de identificar similaridade de forma entre trajetórias.

6

CONSIDERAÇÕES FINAIS

Neste capítulo são apresentadas as conclusões, contribuições e propostas para trabalhos futuros.

6.1 Conclusões

Trajetórias são consideradas uma fonte valiosa de informações para análise e entendimento do comportamento de objetos móveis. Entretanto, apesar dos avanços referentes à identificação de similaridade de forma entre trajetórias encontrados na literatura, eles não apresentam métodos capazes de medir a similaridade de forma de trajetórias em diferentes direções distantes umas das outras. A identificação de similaridade de forma entre trajetórias distantes tem a mesma relevância que a identificação de similaridade para trajetórias próximas, com o diferencial de que se pode assim encontrar comportamentos, baseados na forma, de objetos que estão em países ou até mesmo continentes diferentes. Tratam-se de situações que não podem ser identificadas utilizando as abordagens fornecidas pelos trabalhos até então propostos.

Portanto, o objetivo deste trabalho foi desenvolver um método capaz de calcular a similaridade de forma entre trajetórias independentemente da distância e da direção do movi-

mento entre elas. Para se alcançar este objetivo foi utilizada uma abordagem que leva em consideração os ângulos de deflexão dos segmentos formados por dois pontos e sobre eles é aplicada uma técnica estatística de correlação entre variáveis. A solução é composta por várias etapas que abrangeram desde o pré-processamento para a eliminação de ruído, até o enriquecimento da trajetória com a identificação das direções dos segmentos.

Com base nos experimentos realizados, é possível observar que o método atingiu resultados satisfatórios quando comparado com outras técnicas de identificação de similaridade, tais como o DTW. Os experimentos apontam para valores significativos para correlação de Pearson com valores acima de 0.70. Segundo a literatura de referência, esses valores podem ser considerados como alta correlação entre as variáveis analisadas. Portanto, para este trabalho, é possível dizer que as médias de correlação alcançadas reforçam a ideia de que o método é capaz de identificar similaridade de forma entre trajetórias de objetos móveis.

Entretanto, conforme apontado nas considerações sobre os experimentos, é possível perceber que o método apresenta algumas limitações. Dentre as limitações identificadas, pode-se destacar a sua sensibilidade quanto a ruídos nos valores dos dados-fonte, assim como a necessidade de utilização de um volume considerável de dados para se obter um resultado significativo. Além disso, é possível constatar que o método alcança melhores resultados em trajetórias com curvas do que em trajetórias predominantemente retas. Isso é explicado pelo fato de que em trajetórias retilíneas os valores das deflexões dos segmentos é baixo, o que representa um baixa covariância entre os valores das variáveis analisadas. Assim, qualquer deflexão que apresente um valor elevado de ruído não eliminado na etapa de pré-processamento prejudica consideravelmente o valor da correlação.

Como contribuição, o método proposto apresenta uma forma para identificar similaridade entre trajetórias distantes e em direções distintas, fato não observado nos trabalhos relacionados. Por fim, é possível perceber que o método ainda precisa ser melhorado, principalmente no que diz respeito ao pré-

processamento devido à sensibilidade que os ruídos causam ao resultado da aplicação dos coeficientes de correlação.

6.1.1 Trabalhos Futuros

Como trabalhos futuros, sugere-se a aplicação de filtros, tais como o filtro de Kalman, para melhorar a qualidade dos dados antes da aplicação das correlações de Pearson. Outro caminho a ser explorado pelo método é a possibilidade de identificar similaridade em subtrajetórias contidas dentro de trajetórias. Dessa forma, seria possível delimitar qual trecho das trajetórias que estão sendo comparadas tem maior grau de similaridade. Além disso, a abordagem proposta não considera outras características para calcular similaridade, como o tempo de duração da trajetória. Portanto, uma outra proposta de trabalho futuro seria a expansão do método proposto neste trabalho para que ele pudesse considerar na identificação de similaridade questões temporais e relacionadas a trajetórias, entre outros.

Referências

AIGNER, W.; MIKSCH, S.; SCHUMANN, H.; TOMINSKI, C. *Visualization of time-oriented data*. [S.l.]: Springer Science & Business Media, 2011.

ALLEN, J. F. Maintaining knowledge about temporal intervals. *Communications of the ACM*, ACM, v. 26, n. 11, p. 832–843, 1983.

ANDRIENKO, G.; ANDRIENKO, N.; BAK, P.; KEIM, D.; WROBEL, S. *Visual analytics of movement*. [S.l.]: Springer Science & Business Media, 2013.

ANDRIENKO, G.; ANDRIENKO, N.; DEMSAR, U.; DRANSCH, D.; DYKES, J.; FABRIKANT, S. I.; JERN, M.; KRAAK, M.-J.; SCHUMANN, H.; TOMINSKI, C. Space, time and visual analytics. *International Journal of Geographical Information Science*, Taylor & Francis, v. 24, n. 10, p. 1577–1600, 2010.

ANDRIENKO, N.; ANDRIENKO, G.; GATALSKY, P. Exploratory spatio-temporal visualization: an analytical review. *Journal of Visual Languages & Computing*, Elsevier, v. 14, n. 6, p. 503–541, 2003.

BARBETTA, P. A.; REIS, M. M.; BORNIA, A. C. *Estatística: para cursos de engenharia e informática*. [S.l.]: Atlas São Paulo, 2010.

BERNDT, D. J.; CLIFFORD, J. Using dynamic time warping to find patterns in time series. In: SEATTLE, WA. *KDD workshop*. [S.l.], 1994. v. 10, n. 16, p. 359–370.

BOGORNY, V.; RENSO, C.; AQUINO, A. R.; SIQUEIRA, F. L.; ALVARES, L. O. Constant–a conceptual data model for semantic trajectories of moving objects. *Transactions in GIS*, Wiley Online Library, v. 18, n. 1, p. 66–88, 2014.

BORGES, A. *Topografia Aplicada à Engenharia Civil–Volume I*. [S.l.]: Editora Edgard Blücher Ltda, 2013.

BRAZ, F. J.; BOGORNY, V. *Introdução a trajetórias de objetos móveis*. [S.l.]: Editora Univille, 2012. ISBN 9788582090022.

BRAZ, F. J.; ORLANDO, S. Trajectory data warehouses: Proposal of design and application to exploit data. In: CITESEER. *GeoInfo*. [S.l.], 2007. p. 61–72.

CHEN, L.; ÖZSU, M. T.; ORIA, V. Robust and fast similarity search for moving object trajectories. In: ACM. *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. [S.l.], 2005. p. 491–502.

CHEN, P. Y.; POPOVICH, P. M. *Correlation: Parametric and nonparametric measures*. [S.l.]: Sage, 2002.

COHEN, J. *Statistical power analysis for the behavioral sciences*. [S.l.]: Academic press, 2013.

COSTA, G. H.; BALDO, F. Generation of road maps from trajectories collected with smartphone–a method based on genetic algorithm. *Applied Soft Computing*, Elsevier, v. 37, p. 799–808, 2015.

DANCEY, C. P.; REIDY, J. *Estatística sem Matemática para Psicologia*. [S.l.]: Penso, 2013.

DODGE, S.; WEIBEL, R.; FOROOTAN, E. Revealing the physics of movement: Comparing the similarity of movement characteristics of different types of moving objects. *Computers, Environment and Urban Systems*, Elsevier, v. 33, n. 6, p. 419–434, 2009.

DODGE, S.; WEIBEL, R.; LAUTENSCHÜTZ, A.-K. Towards a taxonomy of movement patterns. *Information visualization*, SAGE Publications, v. 7, n. 3-4, p. 240–252, 2008.

DORR, C. H.; LATECKI, L. J.; MORATZ, R. Shape similarity based on the qualitative spatial reasoning calculus eopram. In: SPRINGER. *International Workshop on Spatial Information Theory*. [S.l.], 2015. p. 130–150.

DOUGLAS, D. H.; PEUCKER, T. K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, University of Toronto Press, v. 10, n. 2, p. 112–122, 1973.

EDEN, A. H. Three paradigms of computer science. *Minds and machines*, Springer, v. 17, n. 2, p. 135–167, 2007.

FENG, T.; TIMMERMANS, H. J. Transportation mode recognition using gps and accelerometer data. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 37, p. 118–130, 2013.

FILHO, D. B. F.; JUNIOR, J. A. S. Desvendando os mistérios do coeficiente de correlação de pearson (r). *Revista Política Hoje*, v. 18, n. 1, 2010.

FRENTZOS, E.; THEODORIDIS, Y.; PAPADOPOULOS, A. N. Spatio-temporal trajectories. In: *Encyclopedia of Database Systems*. [S.l.]: Springer, 2009. p. 2742–2746.

FURTADO, A. S.; KOPANAKI, D.; ALVARES, L. O.; BOGORNY, V. Multidimensional similarity measuring for

semantic trajectories. *Transactions in GIS*, Wiley Online Library, 2015.

GIANNOTTI, F.; NANNI, M.; PEDRESCHI, D.; PINELLI, F.; RENSO, C.; RINZIVILLO, S.; TRASARTI, R. Unveiling the complexity of human mobility by querying and mining massive trajectory data. *The VLDB Journal—The International Journal on Very Large Data Bases*, Springer-Verlag New York, Inc., v. 20, n. 5, p. 695–719, 2011.

GIL, A. C. *Como Elaborar Projetos de Pesquisa.* [S.l.]: Atlas, 2010.

GOSLING, J.; MCGILTON, H. The java language environment. *Sun Microsystems Computer Company*, v. 2550, 1995.

GRISS, J.; REISINGER, F.; HERMJAKOB, H.; VIZCAÍNO, J. A. jmzreader: A java parser library to process and visualize multiple text and xml-based mass spectrometry data formats. *Proteomics*, Wiley Online Library, v. 12, n. 6, p. 795–798, 2012.

HIBERNATE. *Hibernate. Everything data.* 2016. Acesso em: 02 fev. 2016. Disponível em: <http://hibernate.org>.

HOBERMAN, S. *Data Modeling for MongoDB: Building Well-Designed and Supportable MongoDB Databases.* [S.l.]: Technics Publications, 2014. ISBN 9781935504702.

IHAKA, R.; GENTLEMAN, R. R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, Taylor & Francis, v. 5, n. 3, p. 299–314, 1996.

KAVANAGH, B. F.; BIRD, S. G. *Surveying: Principles and applications.* [S.l.]: Prentice Hall Upper Saddle River (NJ), 2000.

LAKATOS, E. M.; MARCONI, M. d. A. Fundamentos da metodologia científica. In: *Fundamentos da metodologia científica.* [S.l.]: Altas, 2010.

LAUBE, P.; DENNIS, T.; FORER, P.; WALKER, M. Movement beyond the snapshot–dynamic analysis of geospatial lifelines. *Computers, Environment and Urban Systems*, Elsevier, v. 31, n. 5, p. 481–501, 2007.

LEVIN, J.; FOX, J. A. Estatística para ciências humanas. In: *Estatística para ciências humanas*. [S.l.]: Pearson, 2004.

LIU, H.; SCHNEIDER, M. Similarity measurement of moving object trajectories. In: ACM. *Proceedings of the Third ACM SIGSPATIAL International Workshop on GeoStreaming*. [S.l.], 2012. p. 19–22.

MEHTA, S.; MACHIRAJU, R.; PARTHASARATHY, S. *Towards Object based Trajectory Representation and Analysis*. [S.l.], 2006.

MERATNIA, N.; ROLF, A. Spatiotemporal compression techniques for moving point objects. In: SPRINGER. *International Conference on Extending Database Technology*. [S.l.], 2004. p. 765–782.

MUCKELL, J.; HWANG, J.-H.; PATIL, V.; LAWSON, C. T.; PING, F.; RAVI, S. Squish: an online approach for gps trajectory compression. In: ACM. *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*. [S.l.], 2011. p. 13.

NORMANDO, D.; TJÄDERHANE, L.; QUINTÃO, C. C. A. A escolha do teste estatístico–um tutorial em forma de apresentação em powerpoint. *Revista Dental Press de Ortodontia e Ortopedia Facial*, SciELO Brasil, v. 15, p. 107–112, 2010.

PARENT, C.; SPACCAPIETRA, S.; RENSO, C.; ANDRIENKO, G.; ANDRIENKO, N.; BOGORNY, V.; DAMIANI, M. L.; GKOULALAS-DIVANIS, A.; MACEDO, J.; PELEKIS, N. et al. Semantic trajectories modeling and analysis. *ACM Computing Surveys (CSUR)*, ACM, v. 45, n. 4, p. 42, 2013.

PELEKIS, N.; ANDRIENKO, G.; ANDRIENKO, N.; KOPANAKIS, I.; MARKETOS, G.; THEODORIDIS, Y. Visually exploring movement data via similarity-based analysis. *Journal of Intelligent Information Systems*, Springer, v. 38, n. 2, p. 343–391, 2012.

PEUQUET, D. J. *Representations of space and time*. [S.l.]: Guilford Press, 2002.

PIÓRKOWSKI, A. Mysql spatial and postgis–implementations of spatial data standards. *EJPAU*, v. 14, n. 1, p. 03, 2011.

POTAMIAS, M.; PATROUMPAS, K.; SELLIS, T. Sampling trajectory streams with spatiotemporal criteria. In: IEEE. *18th International Conference on Scientific and Statistical Database Management (SSDBM'06)*. [S.l.], 2006. p. 275–284.

REESE, G. *Database Programming with JDBC and JAVA*. [S.l.]: "O'Reilly Media, Inc.", 2000.

RENSO, C.; SPACCAPIETRA, S.; ZIMÁNYI, E. *Mobility Data*. [S.l.]: Cambridge University Press, 2013.

SHEATS, R. D.; PANKRATZ, V. S. Common statistical tests. In: ELSEVIER. *Seminars in Orthodontics*. [S.l.], 2002. v. 8, n. 2, p. 77–86.

SILVA, I.; SEGANTINE, P. C. L. *Topografia para Engenharia*. [S.l.]: Elsevier Brasil, 2014.

STIGLER, S. M. Francis galton's account of the invention of correlation. *Statistical Science*, JSTOR, p. 73–79, 1989.

TIOBE. *TIOBE Index*. 2016. Acesso em: 31 jan. 2016. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.

TULER, M. O.; SARAIVA, S. *Fundamentos da topografia*. [S.l.]: Editora Bookman, 2014.

VLACHOS, M.; KOLLIOS, G.; GUNOPULOS, D. Discovering similar multidimensional trajectories. In: IEEE. *Data Engineering, 2002. Proceedings. 18th International Conference on*. [S.l.], 2002. p. 673–684.

WAZLAWICK, R. *Metodologia de Pesquisa para Ciência da Computação, 2ª Edição*. [S.l.]: Elsevier Brasil, 2014.

WAZLAWICK, R. S. Uma reflexão sobre a pesquisa em ciência da computação à luz da classificação das ciências e do método científico. *Revista de Sistemas de Informação da FSMA*, n. 6, p. 3–10, 2010.

WEGHE, N. Van de; TRÉ, G. D.; KUIJPERS, B.; MAEYER, P. D. The double-cross and the generalization concept as a basis for representing and comparing shapes of polylines. In: SPRINGER. *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*. [S.l.], 2005. p. 1087–1096.

XIE, M. Eds: a segment-based distance measure for sub-trajectory similarity search. In: ACM. *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. [S.l.], 2014. p. 1609–1610.

YUAN, J.; ZHENG, Y.; XIE, X.; SUN, G. Driving with knowledge from the physical world. In: ACM. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2011. p. 316–324.

YUAN, J.; ZHENG, Y.; ZHANG, C.; XIE, W.; XIE, X.; SUN, G.; HUANG, Y. T-drive: driving directions based on taxi trajectories. In: ACM. *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*. [S.l.], 2010. p. 99–108.

ZHENG, V. W.; ZHENG, Y.; XIE, X.; YANG, Q. Towards mobile intelligence: Learning from gps history data for collaborative recommendation. *Artificial Intelligence*, Elsevier, v. 184, p. 17–37, 2012.

ZHENG, Y. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, ACM, v. 6, n. 3, p. 29, 2015.

ZHENG, Y.; ZHANG, L.; XIE, X.; MA, W.-Y. Mining interesting locations and travel sequences from gps trajectories. In: ACM. *Proceedings of the 18th international conference on World wide web*. [S.l.], 2009. p. 791–800.

A

Bibliotecas utilizadas nos experimentos

Afim de acelerar o desenvolvimento dos experimentos, as seguintes bibliotecas *open-source* foram utilizadas:

Hibernate

Hibernate é um projeto que tem como objetivo fornecer uma solução completa para o problema de gerenciamento de dados persistentes em Java. Seu ORM consiste em um núcleo, um serviço de base para a persistência com bancos de dados SQL, e uma API proprietária nativa.

<http://hibernate.org>

PSPEARMAN

O pacote PSPEARMAN da linguagem R, é uma pacote complementar da correlação de Spearman ρ (rho). Que possibilita três aproximações de distribuição que são distruição do tipo t, AS89 e a preprocessando do tipo nula para um distribuição com um total de elementos ≤ 22.

<https://cran.r-project.org/web/packages/pspearman/>

PWR

O pacote PWR da linguagem R, é um pacote para calcular o poder de um teste, baseado no trabalho de Cohen (1988).

<https://cran.r-project.org/web/packages/pwr/>

B

Códigos fonte dos experimentos

B.1 Calculo de Azimute

Essa classe foi construída, para auxiliar os cálculos durante os experimentos do método. Contém os cálculos necessários para realizar as experimentações, por exemplo, cálculo do Azimute.

```
1  package br.udesc.mca.matematica;
2
3  import br.udesc.mca.modelo.ponto.Ponto;
4
5  /**
6   * Classe que trabalha com a questão do cálculo de Azimute e
7   * Distância em KM.
8   */
9  public final class Azimute {
10
11   /**
12    * Valor do raio de curvatura de terra em KM
13    */
14   public static double RAIO_TERRA_KM = 6371030.0;
15
16   /**
17    * Retorna a distância entre dois pontos usando a fórmula de
18    * haversine.
19    *
20    * @param lat1
```

```
21      *       latitude do primeiro ponto
22      * @param lon2
23      *       logintude do primeiro ponto
24      * @param lat2
25      *       latitude do segundo ponto
26      * @param lon2
27      *       logintude do segundo ponto
28      *
29      * @returns Distância em KM entre os dois pontos.
30      */
31     public static double calculaDistanciaKM(double lat1, double lon1,
32         double lat2, double lon2) {
33
34       double latDistance = Math.toRadians(lat2 - lat1);
35       double lonDistance = Math.toRadians(lon2 - lon1);
36
37       double a = Math.sin(latDistance / 2) * Math.sin(latDistance / 2)
38         + Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2))
39         * Math.sin(lonDistance / 2) * Math.sin(lonDistance / 2);
40
41       double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
42       double distance = Azimute.RAIO_TERRA_KM * c;
43
44       return distance / 1000;
45     }
46
47     /**
48      * Retorna a distância entre dois pontos usando a fórmula de haversine.
49      *
50      * @param lat1
51      *       latitude do primeiro ponto
52      * @param lon2
53      *       logintude do primeiro ponto
54      * @param lat2
55      *       latitude do segundo ponto
56      * @param lon2
57      *       logintude do segundo ponto
58      *
59      * @param arredondar
60      *       indicativo se deseja arredondar o valor retornado
61      *
62      * @returns Distância em metros entre os dois pontos.
63      */
64     public static double calculaDistanciaMetros(double lat1, double lon1,
65         double lat2, double lon2, boolean arredondar) {
```

```
66
67      double latDistance = Math.toRadians(lat2 - lat1);
68      double lonDistance = Math.toRadians(lon2 - lon1);
69
70      double a = Math.sin(latDistance / 2) * Math.sin(latDistance / 2)
71          + Math.cos(Math.toRadians(lat1))
72          * Math.cos(Math.toRadians(lat2))
73          * Math.sin(lonDistance / 2) * Math.sin(lonDistance / 2);
74
75      double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
76      double distance = Azimute.RAIO_TERRA_KM * c;
77
78      if (arredondar) {
79        distance = Math.round(distance / 1000);
80      }
81      return distance;
82    }
83
84    /**
85     * Calcula a distância a partir de um ponto P passando pelo
86     * grande círculo que passa por outros dois pontos A e B.
87     *
88     * @param p
89     *      ponto
90     * @param a
91     *      primeiro ponto
92     * @param b
93     *      segundo ponto
94     * @return distância, em metros
95     */
96    public static double distanciaPerpendicular(Ponto p, Ponto a,
97        Ponto b) {
98      double lata = Math.toRadians(a.getLatitude());
99      double lnga = Math.toRadians(a.getLongitude());
100     double latb = Math.toRadians(b.getLatitude());
101     double lngb = Math.toRadians(b.getLongitude());
102     double latp = Math.toRadians(p.getLatitude());
103     double lngp = Math.toRadians(p.getLongitude());
104     double sinlata = Math.sin(lata);
105     double coslata = Math.cos(lata);
106     double sinlnga = Math.sin(lnga);
107     double coslnga = Math.cos(lnga);
108     double sinlatb = Math.sin(latb);
109     double coslatb = Math.cos(latb);
110     double sinlngb = Math.sin(lngb);
```

```
111     double coslngb = Math.cos(lngb);
112     double sinlatp = Math.sin(latp);
113     double coslatp = Math.cos(latp);
114     double sinlngp = Math.sin(lngp);
115     double coslngp = Math.cos(lngp);
116     double costh = sinlata * sinlatb + coslata * coslatb * (coslnga
117         * coslngb + sinlnga * sinlngb);
118     double sin2th = 1 - costh * costh;
119     if (sin2th < 1.0E-20) {
120       // a e b são muito próximos; return a distância de a para p
121       double costhp = sinlata * sinlatp + coslata * coslatp * (coslnga
122           * coslngp + sinlnga * sinlngp);
123       // se não tem altitude no ponto atribui-se 0
124       if (p.getAltitude() == null) {
125         return Math.acos(costhp) * (RAIO_TERRA_KM + 0);
126       }
127       return Math.acos(costhp) * (RAIO_TERRA_KM + p.getAltitude());
128     }
129     double num = sinlata * (coslatb * coslatp * coslngb * sinlngp
130         - coslatb * coslatp * sinlngb * coslngp) + coslata * coslnga
131         * (coslatb * sinlatp * sinlngb - sinlatb * coslatp * sinlngp)
132         + coslata * sinlnga * (sinlatb * coslatp * coslngp - coslatb
133         * sinlatp * coslngb);
134     double sinr = Math.abs(num) / Math.sqrt(sin2th);
135
136     // se não tem altitude no ponto atribui-se 0
137     if (p.getAltitude() == null) {
138       return (RAIO_TERRA_KM + 0) * Math.asin(sinr);
139     }
140     return (RAIO_TERRA_KM + p.getAltitude()) * Math.asin(sinr);
141   }
142
143   /**
144    * Retorna o azimute a frente entre 2 pontos.
145    *
146    * @param lat1
147    *      latitude do primeiro ponto
148    * @param lon2
149    *      logintude do primeiro ponto
150    * @param lat2
151    *      latitude do segundo ponto
152    * @param lon2
153    *      logintude do segundo ponto
154    * @returns azimute em graus.
155    */
```

```
156   public static double azimute(double lat1, double lon1, double lat2,
157       double lon2) {
158     double phi1 = Math.toRadians(lat1);
159     double phi2 = Math.toRadians(lat2);
160     double delta = Math.toRadians(lon2 - lon1);
161     double y = Math.sin(delta) * Math.cos(phi2);
162     double x = Math.cos(phi1) * Math.sin(phi2) - Math.sin(phi1)
163         * Math.cos(phi2) * Math.cos(delta);
164     double teta = Math.atan2(y, x);
165     return (Math.toDegrees(teta) + 360D) % 360D;
166   }
167
168   /**
169    * Retorna a diferença de azimute em graus decimais do primeiro
170    * subtraindo do segundo.
171    *
172    * @param azimute1
173    *       recebe o azimute em graus decimais.
174    * @param azimute2
175    *       recebe o azimute em graus decimais.
176    * @returns direferença de azimute em graus decimais.
177    */
178   public static double diferencaAzimute(double azimute1,
179       double azimute2) {
180     return azimute1 - azimute2;
181   }
182
183   /**
184    * Retorna a diferença de azimute em graus decimais do primeiro
185    * subtraindo do segundo, retornando o valor positivo.
186    *
187    * @param azimute1
188    *       recebe o azimute em graus decimais.
189    * @param azimute2
190    *       recebe o azimute em graus decimais.
191    * @returns direferença de azimute em graus decimais.
192    */
193   public static double diferencaAzimutePositivo(double azimute1,
194       double azimute2) {
195     return Math.abs(azimute1 - azimute2);
196   }
197
198   public static void computarDistanciaEAzimute(double lat1,
199       double lon1, double lat2, double lon2, float[] results) {
200
```

```
201       int MAXITERS = 20;
202       lat1 *= Math.PI / 180.0;
203       lat2 *= Math.PI / 180.0;
204       lon1 *= Math.PI / 180.0;
205       lon2 *= Math.PI / 180.0;
206
207       double a = 6378137.0;
208       double b = 6356752.3142;
209       double f = (a - b) / a;
210       double aSqMinusBSqOverBSq = (a * a - b * b) / (b * b);
211
212       double L = lon2 - lon1;
213       double A = 0.0;
214       double U1 = Math.atan((1.0 - f) * Math.tan(lat1));
215       double U2 = Math.atan((1.0 - f) * Math.tan(lat2));
216
217       double cosU1 = Math.cos(U1);
218       double cosU2 = Math.cos(U2);
219       double sinU1 = Math.sin(U1);
220       double sinU2 = Math.sin(U2);
221       double cosU1cosU2 = cosU1 * cosU2;
222       double sinU1sinU2 = sinU1 * sinU2;
223
224       double sigma = 0.0;
225       double deltaSigma = 0.0;
226       double cosSqAlpha = 0.0;
227       double cos2SM = 0.0;
228       double cosSigma = 0.0;
229       double sinSigma = 0.0;
230       double cosLambda = 0.0;
231       double sinLambda = 0.0;
232
233       double lambda = L;
234       for (int iter = 0; iter < MAXITERS; iter++) {
235         double lambdaOrig = lambda;
236         cosLambda = Math.cos(lambda);
237         sinLambda = Math.sin(lambda);
238         double t1 = cosU2 * sinLambda;
239         double t2 = cosU1 * sinU2 - sinU1 * cosU2 * cosLambda;
240         double sinSqSigma = t1 * t1 + t2 * t2; // (14)
241         sinSigma = Math.sqrt(sinSqSigma);
242         cosSigma = sinU1sinU2 + cosU1cosU2 * cosLambda; // (15)
243         sigma = Math.atan2(sinSigma, cosSigma); // (16)
244         double sinAlpha = (sinSigma == 0) ? 0.0 :
245             cosU1cosU2 * sinLambda / sinSigma; // (17)
```

```
246         cosSqAlpha = 1.0 - sinAlpha * sinAlpha;
247         cos2SM = (cosSqAlpha == 0) ? 0.0 :
248             cosSigma - 2.0 * sinU1sinU2 / cosSqAlpha; // (18)
249
250         double uSquared = cosSqAlpha * aSqMinusBSqOverBSq; // defn
251         A = 1 + (uSquared / 16384.0) * // (3)
252             (4096.0 + uSquared * (-768 + uSquared *
253             (320.0 - 175.0 * uSquared)));
254         double B = (uSquared / 1024.0) * // (4)
255             (256.0 + uSquared * (-128.0 + uSquared *
256             (74.0 - 47.0 * uSquared)));
257         double C = (f / 16.0) * cosSqAlpha * (4.0 + f *
258             (4.0 - 3.0 * cosSqAlpha)); // (10)
259         double cos2SMSq = cos2SM * cos2SM;
260         deltaSigma = B * sinSigma * // (6)
261             (cos2SM + (B / 4.0) * (cosSigma * (-1.0 + 2.0 * cos2SMSq)
262             - (B / 6.0) * cos2SM * (-3.0 + 4.0 * sinSigma * sinSigma)
263             * (-3.0 + 4.0 * cos2SMSq)));
264
265         lambda = L + (1.0 - C) * f * sinAlpha
266             * (sigma + C * sinSigma * (cos2SM + C * cosSigma *
267             (-1.0 + 2.0 * cos2SM * cos2SM))); // (11)
268
269         double delta = (lambda - lambdaOrig) / lambda;
270         if (Math.abs(delta) < 1.0e-12) {
271           break;
272         }
273       }
274
275       float distance = (float) (b * A * (sigma - deltaSigma));
276       results[0] = distance;
277       if (results.length > 1) {
278         float initialBearing = (float) Math.atan2(cosU2 * sinLambda,
279             cosU1 * sinU2 - sinU1 * cosU2 * cosLambda);
280         initialBearing *= 180.0 / Math.PI;
281         results[1] = initialBearing;
282         if (results.length > 2) {
283           float finalBearing = (float) Math.atan2(cosU1 * sinLambda,
284               -sinU1 * cosU2 + cosU1 * sinU2 * cosLambda);
285           finalBearing *= 180.0 / Math.PI;
286           results[2] = finalBearing;
287         }
288       }
289     }
290 }
```

B.2 Classe do Segmento

Esta classe representa o segmento da trajetória, no mapeamento do modelo de dados construído para suportar o armazenamento dos pontos utilizados nos experimentos. As outras classes para representar o mapeamento objeto-relacional seguem o mesmo estilo.

```
1  package br.udesc.mca.modelo.segmento;
2
3  import java.io.Serializable;
4  import java.util.List;
5
6  import javax.persistence.*;
7
8  import br.udesc.mca.modelo.ponto.Ponto;
9
10 @Entity
11 @Table(name = "segmento")
12 @NamedQuery(name = "consultaSegmentoTrajetoria",
13     query = "select distinct(s) from Segmento s "
14     + "inner join s.ponto p where p.trajetoria.id = "
15     + " :trajetoriaId order by s.id")
16 public class Segmento implements Serializable {
17
18   public static final int DIREITA = 1;
19   public static final int ESQUERDA = 2;
20   public static final int RETA = 3;
21
22   private static final long serialVersionUID = 1L;
23
24   @Id
25   @SequenceGenerator(name = "gen_segmento", sequenceName = "seq_segmentoid")
26   @GeneratedValue(generator = "gen_segmento")
27   @Column(name = "id")
28   private Long id;
29
30   private Double azimute;
31
32   @Column(name = "diferenca_azimute")
33   private Double diferencaAzimute;
34
35   @Column(name = "diferenca_azimute_positiva")
36   private Double diferencaAzimutePositiva;
```

```
37
38      @ManyToMany
39      @JoinTable(name = "ponto_segmento",
40          foreignKey = @ForeignKey(name = "ponto_segmento_segmento_id_fk"),
41          inverseForeignKey = @ForeignKey(name =
42              "ponto_segmento_ponto_id_fk"),
43          joinColumns = {
44          @JoinColumn(name = "segmento_id", referencedColumnName = "id")},
45          inverseJoinColumns = {
46              @JoinColumn(name = "ponto_id", referencedColumnName = "id")})
47      private List<Ponto> ponto;
48
49      @Transient
50      private int direcao;
51
52      @Transient
53      private double comprimento;
54
55      public Long getId() {
56        return id;
57      }
58
59      public void setId(Long id) {
60        this.id = id;
61      }
62
63      public Double getAzimute() {
64        return azimute;
65      }
66
67      public void setAzimute(Double azimute) {
68        this.azimute = azimute;
69      }
70
71      public Double getDiferencaAzimute() {
72        return diferencaAzimute;
73      }
74
75      public void setDiferencaAzimute(Double diferencaAzimute) {
76        this.diferencaAzimute = diferencaAzimute;
77      }
78
79      public Double getDiferencaAzimutePositiva() {
80        return diferencaAzimutePositiva;
81      }
```

```
82
83    public void setDiferencaAzimutePositiva(Double
84        diferencaAzimutePositiva) {
85      this.diferencaAzimutePositiva = diferencaAzimutePositiva;
86    }
87
88    public List<Ponto> getPonto() {
89      return ponto;
90    }
91
92    public void setPonto(List<Ponto> ponto) {
93      this.ponto = ponto;
94    }
95
96    public int getDirecao() {
97      return direcao;
98    }
99
100   public void setDirecao(int direcao) {
101     this.direcao = direcao;
102   }
103
104   public double getComprimento() {
105     return comprimento;
106   }
107
108   public void setComprimento(double comprimento) {
109     this.comprimento = comprimento;
110   }
111
112   @Override
113   public int hashCode() {
114     final int prime = 31;
115     int result = 1;
116     result = prime * result + ((azimute == null) ? 0 :
117         azimute.hashCode());
118     long temp;
119     temp = Double.doubleToLongBits(comprimento);
120     result = prime * result + (int) (temp ^ (temp >>> 32));
121     result = prime * result + ((diferencaAzimute == null) ? 0 :
122     diferencaAzimute.hashCode());
123     result = prime * result + ((diferencaAzimutePositiva == null) ? 0 :
124     diferencaAzimutePositiva.hashCode());
125     result = prime * result + direcao;
126     result = prime * result + ((id == null) ? 0 : id.hashCode());
```

```
127       result = prime * result + ((ponto == null) ? 0 : ponto.hashCode());
128       return result;
129     }
130
131     @Override
132     public boolean equals(Object obj) {
133       if (this == obj)
134         return true;
135       if (obj == null)
136         return false;
137       if (getClass() != obj.getClass())
138         return false;
139       Segmento other = (Segmento) obj;
140       if (azimute == null) {
141         if (other.azimute != null)
142           return false;
143       } else if (!azimute.equals(other.azimute))
144         return false;
145       if (Double.doubleToLongBits(comprimento) != Double.doubleToLongBits(
146       other.comprimento))
147         return false;
148       if (diferencaAzimute == null) {
149         if (other.diferencaAzimute != null)
150           return false;
151       } else if (!diferencaAzimute.equals(other.diferencaAzimute))
152         return false;
153       if (diferencaAzimutePositiva == null) {
154         if (other.diferencaAzimutePositiva != null)
155           return false;
156       } else if (!diferencaAzimutePositiva.equals(other.
157           diferencaAzimutePositiva))
158         return false;
159       if (direcao != other.direcao)
160         return false;
161       if (id == null) {
162         if (other.id != null)
163           return false;
164       } else if (!id.equals(other.id))
165         return false;
166       if (ponto == null) {
167         if (other.ponto != null)
168           return false;
169       } else if (!ponto.equals(other.ponto))
170         return false;
171       return true;
```

```
172   }
173 }
```

B.3 Segmentação por cor

Este código foi construído com o objetivo de facilitar a visualização da segmentação por direção através do uso de diferentes cores. Este algoritmo utiliza-se do formato vetorial SVG para a exportação dos dados.

```
1  package br.udesc.mca.segmentador.cores;
2
3  import java.awt.Color;
4  import java.io.FileWriter;
5  import java.io.IOException;
6  import java.text.ParseException;
7  import java.util.List;
8
9  import org.hibernate.Session;
10 import org.jfree.graphics2d.svg.SVGGraphics2D;
11
12 import br.udesc.mca.conexao.HibernateUtil;
13 import br.udesc.mca.modelo.ponto.Ponto;
14 import br.udesc.mca.modelo.trajetoria.Trajetoria;
15 import br.udesc.mca.modelo.trajetoria.TrajetoriaDAOPostgreSQL;
16 import br.udesc.mca.segmentador.compactacao.SegmentadorCompactador;
17
18 public class TesteSegmentoCor {
19   private static final int ZOOM = 100000;
20   private static final int ERROR = 25;
21
22   public static void main(String[] args) throws IOException,
23       ParseException {
24     Session sessao = HibernateUtil.getSessionFactory().openSession();
25     TrajetoriaDAOPostgreSQL trajetoriaDAOPostgreSQL = new
26         TrajetoriaDAOPostgreSQL(sessao);
27
28     Trajetoria trajetoria = trajetoriaDAOPostgreSQL.
29         selecionarTrajetoria(1);;
30
31     draw(trajetoria.getPontos());
32   }
33
```

```
34    public static void draw(List<Ponto> pontosGPS) {
35      int menorx = 0;
36      int menory = 0;
37      for (Ponto ponto : pontosGPS) {
38        if (ponto.getLatitude() * ZOOM < menory) {
39          menory = (int) (ponto.getLatitude() * ZOOM);
40        }
41        if (ponto.getLongitude() * ZOOM < menorx) {
42          menorx = (int) (ponto.getLongitude() * ZOOM);
43        }
44      }
45      menorx = menorx * -1;
46      menory = menory * -1;
47      int eixoy = 0;
48      for (Ponto ponto : pontosGPS) {
49        int y = (int) (ponto.getLatitude() * ZOOM + menory);
50        if (y > eixoy) {
51          eixoy = y;
52        }
53      }
54      SVGGraphics2D g2 = new SVGGraphics2D(menorx, menory);
55
56      Ponto ant = null;
57      for (Ponto ponto : pontosGPS) {
58        g2.setPaint(Color.BLACK);
59        g2.drawOval((int) (ponto.getLongitude() * ZOOM + menorx - 1),
60            (int) (ponto.getLatitude() * ZOOM + menory) * -1
61            + eixoy - 1, 2, 2);
62        if (ant != null) {
63          if (ponto.getBearing() - ant.getBearing() > ERROR) {
64            g2.setPaint(Color.RED);
65          } else if (ponto.getBearing() - ant.getBearing() < -ERROR) {
66            g2.setPaint(Color.GREEN);
67          } else {
68            g2.setPaint(Color.BLUE);
69          }
70          g2.drawLine((int) (ant.getLongitude() * ZOOM + menorx),
71              (int) (ant.getLatitude() * ZOOM + menory) * -1 + eixoy,
72              (int) (ponto.getLongitude() * ZOOM + menorx),
73              (int) (ponto.getLatitude() * ZOOM + menory) * -1 + eixoy);
74        }
75        ant = ponto;
76      }
77      String svgElement = g2.getSVGElement();
78
```

```
79      try {
80        FileWriter arquivoSVG = new FileWriter("imagem.svg");
81        arquivoSVG.append(g2.getSVGElement());
82        arquivoSVG.flush();
83        arquivoSVG.close();
84      } catch (IOException e) {
85        e.printStackTrace();
86      }
87    }
88  }
```

B.4 Compactação de trajetória

Este algoritmo foi feito para auxiliar a compactação das trajetórias, através da redução do número de pontos para a quantidade definida por parâmetro.

```
1   package br.udesc.mca.compactacao;
2
3   import java.util.ArrayList;
4
5   import br.udesc.mca.modelo.ponto.Ponto;
6
7   public class SQUISH {
8     private SQUISH() {
9       throw new UnsupportedOperationException("Instanciation not allowed");
10    }
11
12    public static Ponto[] compress(Ponto[] stream, int bufferSize){
13
14      if(bufferSize <= 3) return stream;
15
16      ArrayList<Ponto> buffer = new ArrayList<Ponto>();
17
18      for (int i = 0; i < stream.length; i++) {
19
20        buffer.add(stream[i]);
21
22        if(buffer.size() > 2){
23          Ponto item = buffer.get(buffer.size() - 2);
24          item.setSed(ErrorMetrics
25              .getCartesianSynchronizedEuclideanDistance(item,
26              buffer.get(buffer.size() - 3), buffer.get(buffer.size()
```

```
27              - 1)));
28        }
29
30        if(buffer.size() > bufferSize){
31
32          //Find the minor sed in the buffer; first point is not
33          //included; last point is not include because its the intent
34          //point to get in the buffer;
35          int minorSedPosition = 1;
36          for (int j = 1; j < buffer.size() -1; j++) {
37            if(buffer.get(j).getSed() < buffer.get(minorSedPosition)
38                .getSed()) minorSedPosition = j;
39          }
40
41          //Remove the minor sed and update the neighbors;
42          buffer.get(minorSedPosition - 1).setSed(
43              buffer.get(minorSedPosition - 1).getSed()
44                  + buffer.get(minorSedPosition).getSed());
45
46          buffer.get(minorSedPosition + 1).setSed(
47              buffer.get(minorSedPosition + 1).getSed()
48                  + buffer.get(minorSedPosition).getSed());
49
50          buffer.remove(minorSedPosition);
51        }
52      }
53
54      return buffer.toArray(new Ponto[buffer.size()]);
55    }
56  }
```

B.5 Executor de rotinas na linguagem R

Este algoritmo foi feito para auxiliar a execução das funções na linguagem *R*. A integração ocorre através da exportação das duas trajetórias em arquivos CSV, o que são lidos pelas funções em *R*.

```
1  package br.udesc.mca.benchmarking.algoritmos;
2
3  /**
4   * Processo de execução do scripts em R:
5   *
```

```
6   * - Criar a classe java equivalente a chamada do algoritmo em R;
7   * - Criar o script em R equivalente ao algoritmo chamador da classe Java;
8   */
9  import java.io.BufferedReader;
10 import java.io.File;
11 import java.io.IOException;
12 import java.io.InputStreamReader;
13 import java.io.PrintWriter;
14 import java.util.StringTokenizer;
15 import br.udesc.mca.modelo.ponto.Ponto;
16 import br.udesc.mca.modelo.trajetoria.Trajetoria;
17
18 public class AlgoritmoR {
19   public static double distance(File scriptFile, Trajetoria traj1,
20       Trajetoria traj2) {
21     double ret = 0;
22     try {
23       File f1 = File.createTempFile("Traj1", "csv");
24       f1.deleteOnExit();
25       File f2 = File.createTempFile("Traj2", "csv");
26       f2.deleteOnExit();
27       PrintWriter pw = new PrintWriter(f1);
28       for (Ponto p : traj1.getPontos()) {
29         pw.println(p.getLatitude() + "," + p.getLongitude());
30       }
31       pw.flush();
32       pw.close();
33       pw = new PrintWriter(f2);
34       for (Ponto p : traj2.getPontos()) {
35         pw.println(p.getLatitude() + "," + p.getLongitude());
36       }
37       pw.flush();
38       pw.close();
39       ProcessBuilder pb = new ProcessBuilder("Rscript.exe",
40       scriptFile.getAbsolutePath(), f1.getAbsolutePath(), f2
41           .getAbsolutePath());
42       Process p = pb.start();
43       InputStreamReader isr = new InputStreamReader(p
44           .getInputStream());
45       BufferedReader br = new BufferedReader(isr);
46       String saida = br.readLine();
47       System.out.println(saida);
48       StringTokenizer st = new StringTokenizer(saida);
49       st.nextToken();
50       saida = st.nextToken();
```

```
51        ret = Double.parseDouble(saida);
52        br.close();
53        isr.close();
54      } catch (IOException e) {
55        e.printStackTrace();
56      }
57      return ret;
58    }
59  }
```