UNIFACS UNIVERSIDADE SALVADOR
MESTRADO EM SISTEMAS E COMPUTAÇÃO

DUÍLIO SANTA BÁRBARA DAS VIRGENS ANDRADE

IB-TRAJECTORY: UM MODELO SEMÂNTICO PARA REPRESENTAÇÃO DE TRAJETÓRIAS BASEADAS EM ITINERÁRIOS

Salvador
2015

DUÍLIO SANTA BÁRBARA DAS VIRGENS ANDRADE

IB-TRAJECTORY: UM MODELO SEMÂNTICO PARA REPRESENTAÇÃO DE TRAJETÓRIAS BASEADAS EM ITINERÁRIOS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Sistemas e Computação, Universidade Salvador UNIFACS, Laureate International Universities, como requisito parcial para a obtenção do título de Mestre.

Orientador: Prof. Dr. Jorge Alberto Prado de Campos

Salvador
2015

FICHA CATALOGRÁFICA
Elaborada pelo Sistema de Bibliotecas da UNIFACS Universidade Salvador, Laureate International Universities)

Andrade, Duílio Santa Bárbara das Virgens

IB-Trajectory: Um Modelo Semântico para Representação de Trajetórias Baseadas em Itinerários./ Duílio Santa Bárbara das Virgens.- Slavador: UNIFACS, 2015.

116 f. : il.

Dissertação Programa de Pós-Graduação em Sistemas e Computação de UNIFACS Universidade Salvador, Laureate International Universities como requisito parcial à obtenção do título de Mestre.

Orientador: Prof. Dr. Jorge Alberto Prado de Campos.

1. Objetos movéis. I. Campos, Jorge Alberto Prado de, orient. II.Título.

CDD 004.6

TERMO DE APROVAÇÃO

DUÍLIO SANTA BÁRBARA DAS VIRGENS ANDRADE

IB-TRAJECTORY: UM MODELO SEMÂNTICO PARA REPRESENTAÇÃO DE TRAJETÓRIAS BASEADAS EM ITINERÁRIOS

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Sistemas e Computação, Universidade Salvador - UNIFACS, pela seguinte banca examinadora:

Jorge Alberto Prado de Campos - Orientador _____
Doutor em Spatial Information Science and Engineering,University of Maine at Orono
UNIFACS Universidade Salvador

Paulo Nazareno Maia Sampaio _____
Doutor em Informatique et Télécommucations
UPS Université Toulouse III Paul Sabatier

Josemar Rodrigues de Souza _____
Doutor em Informática
UAB Universitat Autònoma de Barcelona

Salvador de de 2015

AGRADECIMENTOS

A Deus, a quem busquei força para superar todo o cansaço e coragem para transpor barreiras.

Aos meus familiares, meus pais, irmã e companheira pelo apoio constante e incondicional em cada passo dado nesta jornada.

Aos meus colegas do Mestrado pelos momentos compartilhados. Aos colegas do grupo de pesquisa GANGES da UNIFACS, principalmente a Helder Aragão e Ana Amorim pelas sugestões, ideias e incentivo.

Ao meu orientador, Prof. Jorge Campos, por ter-me apresentado uma área de pesquisa tão desafiadora e desconhecida por mim, pela paciência, ensinamentos e dedicação. Foi uma longa jornada repleta de desafios e o maior de todos foi conseguir dados reais para validar a proposta do trabalho e muitas pessoas contribuíram de forma direta ou indireta para este objetivo, desde a motivação nos momentos de desamino, até o contato ou fornecimento de dados relevantes para a pesquisa.

Enfim, a todos que contribuíram de alguma forma para a conclusão deste trabalho.

Tudo que um sonho precisa
Para ser realizado é alguém
que acredite que ele possa
ser realizado!

Roberto Shinyashiki

Ele não sabia que era impossível.
Foi lá e fez!

Jean Cocteu

RESUMO

Todos os grandes centros urbanos utilizam GPS, sistema de posicionamento global, nos ônibus e usam esta informação para disponibilizar a posição instantânea do veículo. A despeito do armazenamento de todas as posições coletadas dos veículos serem registradas na base de dados das empresas, as aplicações que utilizam informações sobre o posicionamento dos veículos só estão interessadas na informação em tempo real, ou seja, oferecem serviços apenas no nível operacional. Para o oferecimento de serviços e análises no nível estratégico ou tático, faz-se necessário o processamento e análise dos registros históricos das trajetórias. Este trabalho apresentou um modelo conceitual para representar trajetórias de veículos que se deslocam ao longo da rede viária e possuem um itinerário e uma escala horária a ser seguida. O objetivo é facilitar a análise e visualização das trajetórias dos objetos que realizam este tipo de movimento, buscando identificar padrões, descobrir anomalias, evidenciar desvios de comportamento, estabelecer relacionamentos com outros fenômenos e melhorar o conhecimento em geral sobre este tipo de movimento. O modelo proposto foi aplicado, como prova de conceito, na análise de trajetórias dos ônibus do sistema público de transporte de massa. Este modelo, entretanto, pode ser utilizado para analisar o comportamento dos veículos de empresas de logística, de entrega de mercadorias, de manutenção de infraestrutura e equipamentos ou de qualquer veículo condicionado a realização de trajetórias programadas.

Palavras Chaves: Objetos Móveis. Análise Semântica de Trajetórias. Weka-STPM. Sistemas Inteligentes de Transportes Públicos.

ABSTRACT

With the decreasing cost of positioning technology and data communication, various technological solutions have been deployed in the public transport system. Virtually all major urban centers use GPS in buses and use this information to provide the instantaneous vehicle position. Despite the storage of all positions collected from vehicles are registered in the corporate database, applications that use information about the positioning of the vehicles are only interested in the real-time information, that is, offer services only at the operational level. For offering services and analyzes the strategic or tactical level, it is necessary processing and analysis of historical records of trajectories.This paper presented a conceptual model to represent trajectories vehicles moving along the road network and have an itinerary and a time scale to be followed. The aim is to facilitate the analysis and visualization of the trajectories of objects that perform this type of movement in order to identify patterns, discover anomalies, show deviant behavior, establish relationships with other phenomena and improve the general knowledge about this type of movement. The proposed model was applied as proof of concept, the analysis of trajectories of the public bus system mass transit. This model, however, can be used to analyze the behavior of vehicle logistics companies, delivery of goods, maintenance of infrastructure and equipment or any conditioned vehicle to carry out programmed paths.

Keywords: Moving Objects. Semantic Trajectory Annotation. Itinerary. Weka-STPM. Intelligent Transport System.

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

CB-SMoT	Clustering Based Stops and Moves of Trajectory
DBSCAN	Density-Based Spatial Clustering of Application With Noise
GPS	Global Positioning System
GTFS	General Transit Feed Specification
SMoT	Stops and Moves of Trajectory
Weka	Waikato Environment for Knowledge Analysis

SUMÁRIO

1 INTRODUÇÃO

A popularização de mecanismos de posicionamento, como o GPS (*Global Positioning System*) que é um sistema de navegação por satélite que envia informações sobre posição de um veículo em qualquer horário, tornou possível acompanhar o movimento de virtualmente qualquer objeto que se mova. Atualmente, a circulação de animais, veículos, bens e pessoas são constantemente monitoradas e analisadas para todo tipo de finalidade. Os dados coletados a partir desses objetos, chamados de dados de trajetória bruta, são normalmente armazenados como uma sequência de pontos espaço-temporais que descrevem a evolução da posição dos objetos ao longo do tempo. Este tipo de dado, no entanto, necessita de uma estrutura mais abstrata para sua representação, pois possui muitos registros redundantes ou inconsistentes, que carregam pouca ou nenhuma informação semântica sobre o domínio da aplicação. A fim de superar essas deficiências, é importante converter estes dados brutos de trajetórias em entidades mais abstratas que capturam a essência do movimento de acordo com o domínio da aplicação.

A concepção de um modelo de dados que capture informações relevantes sobre o movimento dos objetos depende de muitas variáveis, tais como: o domínio da aplicação; a granularidade espacial e temporal dos dados coletados; o espaço onde o objeto realiza o movimento (restrito ou sem restrição); e o tipo de movimento.

No que concerne ao tipo de movimento, as trajetórias podem ser classificadas como casual ou programada. A trajetória casual não segue um roteiro ou cronograma conhecido ou fixo. Os felinos, por exemplo, têm um comportamento territorial e um hábito de comer bem conhecido, mas é impossível prever o horário, o local ou a vítima da próxima refeição. Outro exemplo de movimentos casuais é o sistema de transporte de táxi. Táxis se movem em um ambiente restrito (ou seja, a rede viária), mas a menos que o táxi esteja sempre vazio, a sua posição é altamente imprevisível e depende unicamente do desejo dos passageiros.

Uma trajetória programada, por outro lado, segue um caminho ou itinerário definido ou conhecido. Aves de migração, por exemplo, mudam de um lugar para outro a cada ano. Apesar do fato de que as aves se movem em um espaço irrestrito, eles obedecem a uma certa janela de tempo e seguem uma rota conhecida. Além disso, as aves migratórias possuem também lugares preferidos para parar, para se alimentar e descansar. Os ônibus urbanos e trens também realizam trajetórias

predefinidas, mas eles se movem em um espaço limitado (ou seja, rodovias e ferrovias), tem uma escala temporal bem definida, um itinerário a seguir e lugares específicos para parar, permitindo o embarque e desembarque de passageiros. Apesar destes meios de transporte estarem sujeitos a acontecimentos aleatórios, a posição dos veículos em um dado instante é previsível na maioria das vezes.

Este trabalho propõe um modelo conceitual para representar trajetórias de veículos que se deslocam ao longo da rede viária e possuem um itinerário e uma escala horária a ser seguida. O objetivo é facilitar a análise e visualização das trajetórias dos objetos que realizam este tipo de movimento, buscando identificar padrões, descobrir anomalias, evidenciar desvios de comportamento, estabelecer relacionamentos com outros fenômenos e melhorar o conhecimento em geral sobre este tipo de movimento.

O modelo proposto foi aplicado, como prova de conceito, na análise de trajetórias dos ônibus do sistema público de transporte de massa. Este modelo, entretanto, pode ser utilizado para analisar o comportamento dos veículos de empresas de logística, de entrega de mercadorias, de manutenção de infraestrutura e equipamentos ou de qualquer veículo condicionado a realização de trajetórias programadas.

1.1 MOTIVAÇÃO E OBJETIVOS

Com a utilização cada vez maior da tecnologia de posicionamento e de comunicação de dados, diversas soluções tecnológicas têm sido implantadas no sistema de transporte público. Virtualmente, todos os grandes centros urbanos utilizam GPS nos ônibus e usam esta informação para disponibilizar a posição instantânea do veículo ou em aplicações desenvolvidas para auxiliar a identificação de linhas, rotas e pontos de ônibus próximos que atendam a necessidade do usuário.

A despeito do armazenamento de todas as posições coletadas dos veículos serem registradas na base de dados das empresas, as aplicações que utilizam informações sobre o posicionamento dos veículos só estão interessadas na informação em tempo real, ou seja, oferecem serviços apenas no nível operacional. Para o oferecimento de serviços e análises no nível estratégico ou tático, faz-se necessário o processamento e análise dos registros históricos das trajetórias.

O objetivo geral deste trabalho é o desenvolvimento de um modelo conceitual e de dados para a representação do movimento de objetos móveis com movimentos condicionados a uma escala de serviço e o cumprimento de um determinado itinerário. O referido modelo visa facilitar as consultas, extrações de conhecimento e desenvolvimento de novos algoritmos para aplicação que lidam com este tipo de movimento.

Como objetivos específicos tem-se o desenvolvimento de algoritmos para conversão de dados brutos das trajetórias em entidades de primeira classe enriquecidas semanticamente com informações sobre as características do movimento e o desenvolvimento de uma infraestrutura para facilitar as consultas, extrações de conhecimento e o desenvolvimento de novos algoritmos para análise dos deslocamentos de objetos condicionados a movimentos pré-definidos.

1.2 METODOLOGIA

Para o desenvolvimento do modelo de representação de trajetórias baseado em itinerário foi necessário uma revisão e investigação de soluções aplicadas na literatura relacionadas a anotação semântica de trajetórias e à similaridade de trajetórias.

A partir do entendimento do estado da arte e da análise das principais abordagens e trabalhos relacionados, foi concebido o modelo conceitual de especialização dos *Stops* e *Moves* sob o domínio de objetos móveis que possuem seu deslocamento baseado em um itinerário.

Com o modelo conceitual definido, os algoritmos para construção das estruturas semânticas foram desenvolvidos e incorporados como um módulo adicional na ferramenta *open-source* Weka-STPM.

Finalmente, para validar o modelo proposto, foram utilizados dados extraídos de trajetórias brutas realizadas por ônibus em diversas capitais do Brasil.

1.3 CONTRIBUIÇÃO

A principal contribuição deste trabalho é a concepção de um modelo conceitual e algoritmos para incorporar anotação semântica de trajetórias baseadas em itinerários. Os algoritmos desenvolvidos apresentam contribuições tanto na

geração dos *Stops* quanto na determinação dos *Moves*. Na identificação dos *Stops*, por exemplo, foi introduzida a análise do fator de forma do cluster de pontos que define o *Stop* para eliminação de falsos positivos, isto é, paradas causadas por engarrafamentos ou outros eventos alheios a vontade do motorista. O algoritmo de anotação semântica também apresenta uma abordagem diferenciada sob a similaridade de trajetórias para identificação dos *Moves* dentro do domínio de trajetórias baseadas em itinerários.

Todos os algoritmos do modelo proposto estão implementados em um módulo adicional dentro do Weka-STPM para análises posteriores e comparativos com outros métodos de anotação semântica existentes.

Os resultados obtidos com a anotação semântica podem ser úteis para órgãos fiscalizadores e pessoas que utilizam o modal para verificar a eficácia de atendimento do transporte, seja identificando as paradas realizadas ou não e a identificação de desvios de rotas.

1.4 AUDIÊNCIA

A audiência deste trabalho constitui-se de um público diversificado e multidisciplinar. Neste trabalho estão reunidas discussões e contribuições na área de análise de trajetórias de objetos móveis, em geral, e de trajetórias de objetos móveis condicionados ao cumprimento de uma escala de horário e um itinerário, em particular. Assim, o público alvo deste trabalho envolve todas as pessoas e pesquisadores interessados em desenvolvimento de algoritmos e ferramentas voltadas à análise da trajetória de objetos móveis, principalmente de ônibus do sistema de transporte público. Incluem-se no rol de potenciais usuários do modelo as agências reguladoras dos transportes públicos, secretarias de transportes e seus órgãos gestores, empresas de transporte de logística, de entrega de encomendas, de serviços de manutenção, entre outras.

1.5 ESTRUTURA DO TRABALHO

O restante desse documento está estruturado em cinco capítulos. O Capítulo 2 trata dos estudos relacionados aos objetos móveis, semântica e similaridade de trajetórias, isto é, como estruturar os dados brutos da trajetória dos objetos móveis e adicionar informações semânticas que sejam importantes ao domínio da aplicação. Além disso, este capítulo apresenta trabalhos relacionados à similaridade de rotas.

O Capítulo 3 apresenta o modelo *IB-Trajectory*, as definições básicas utilizadas no desenvolvimento do modelo, os detalhes do modelo conceitual, os algoritmos para anotação semântica das entidades do modelo, os detalhes do módulo desenvolvido para a ferramenta Weka-STPM encarregado do mapeamento dos dados brutos da trajetória para entidades do modelo e a carga dos dados em um banco de dados relacional com extensão espacial.

O Capítulo 4 faz uma avaliação do modelo e da sua utilização. Neste capítulo são apresentadas algumas análises e consultas realizadas em uma base de dados construída com entidades do modelo proposto.

Por fim, o Capítulo 5 apresenta a conclusão do trabalho, identifica os pontos fortes e limitações do modelo e indica sugestões de trabalhos futuros.

2 ANÁLISE DE TRAJETÓRIAS URBANAS

Os avanços tecnológicos tornaram a captura de dados de localização dos objetos móveis técnica e economicamente viável, resultando em grandes volumes de dados sobre a movimentação desses objetos. De acordo com Bogorny (2012), qualquer objeto que possua um dispositivo acoplado capaz de informar a sua localização deixa registradas suas “pegadas”. Com a popularização destes dispositivos, abriu-se a possibilidade de registrar, armazenar e analisar a movimentação dos diversos objetos que se deslocam, tais como: pessoas, animais e veículos. Estes registros de movimentações têm aumentado seu volume na mesma proporção que a demanda por novas aplicações e métodos para analisar o comportamento que envolve os seus deslocamentos.

A análise de trajetórias de objetos móveis tem motivado pesquisas nas mais diversas áreas, destacando-se: modelos de dados com enriquecimento semântico (ALVARES et al., 2007; BOGORNY et al., 2013; PALMA et al., 2008; YAN et al., 2011), mineração de dados (VANIA BOGORNY; BART KUIJPERS, 2011), descoberta de conhecimento, padrões de comportamento (ZHENG et al., 2013) e Sistemas Inteligentes de Transporte (ou *ITS do inglês Intelligent Transportation Systems*) (SILVA et al., 2005).

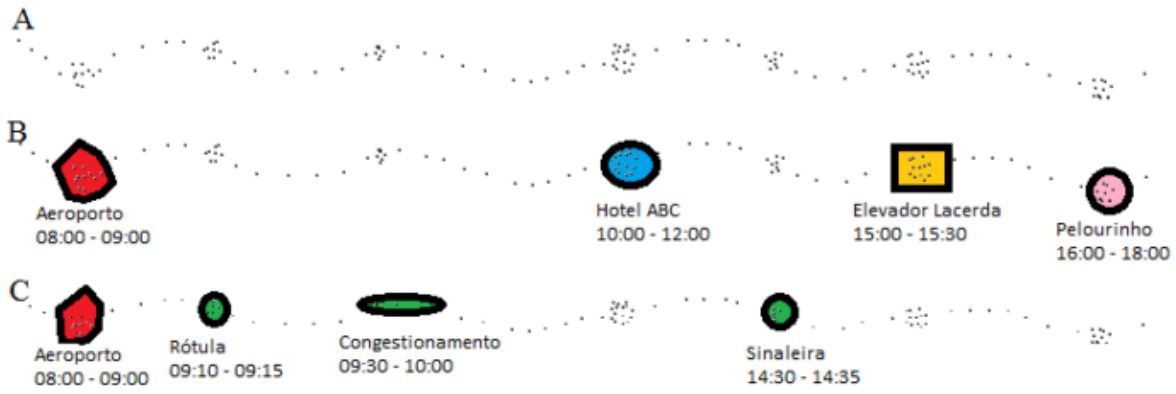
Um dos maiores problemas das pesquisas envolvendo estudos de trajetórias está no tratamento e estruturação da informação bruta. Os dados do movimento dos objetos são geralmente coletados na forma de tuplas (id,x,y,t), em que o “id” identifica o objeto a ser analisado, “x” e “y” são coordenadas geográficas do local e o “t” que o instante em que o objeto ocupou o local referenciado. Estes dados são de pouca utilidade na forma como são guardados. Desta forma, diversas iniciativas se utilizam de algoritmos ou técnicas para realizar o enriquecimento semântico dos dados e facilitar as consultas e extração de conhecimento.

De forma a situar este trabalho em relação às pesquisas envolvendo trajetórias de objetos móveis, em geral, e modelos para representação e análise dos deslocamentos de veículos, em particular, este capítulo foi dividido em duas seções. Primeiramente serão apresentados os trabalhos mais relevantes relacionados a semântica de trajetórias e em seguida serão discutidos trabalhos de análise de movimentação de veículos.

2.1 SEMÂNTICA DE TRAJETÓRIAS

O primeiro modelo reconhecido com o objetivo de adicionar semântica aos dados brutos de trajetórias foi proposto pelo grupo de pesquisa do professor Stefano Spaccapietra (SPACCAPIETRA et al., 2008). Este modelo é genericamente denominado de modelo de *Stops* e *Moves*. Spaccapietra conceituou uma trajetória como sendo a evolução espaço-temporal de um objeto móvel para atingir um determinado objetivo. Este modelo trata as trajetórias de objetos móveis como movimentos que correspondem a deslocamentos semanticamente significativos e, por definição, um conceito espaço-temporal. A Figura ilustra o processo de enriquecimento semântico de trajetórias para diferentes domínios. Na Figura .a observa-se os dados brutos do deslocamento de um objeto móvel, que pode ser uma pessoa ou um veículo. A concentração de alguns pontos ao longo da trajetória bruta, formando clusters, indica que o objeto permaneceu no local por alguns instantes, sugerindo uma parada (*Stop*). Os pontos entre as paradas são considerados deslocamentos (*Move*). Para uma aplicação no domínio de turismo, por exemplo, algumas paradas têm um propósito associado ao contexto da aplicação (Figura .b). Neste contexto, estamos interessados em saber de paradas em locais relacionados ao domínio da aplicação, tais como aeroporto, hotéis e pontos turísticos. Caso o domínio da aplicação fosse à área de transportes, as paradas de interesse envolveriam eventos relacionados a esse domínio, tais como congestionamentos e paradas em sinaleiras (Figura .b).

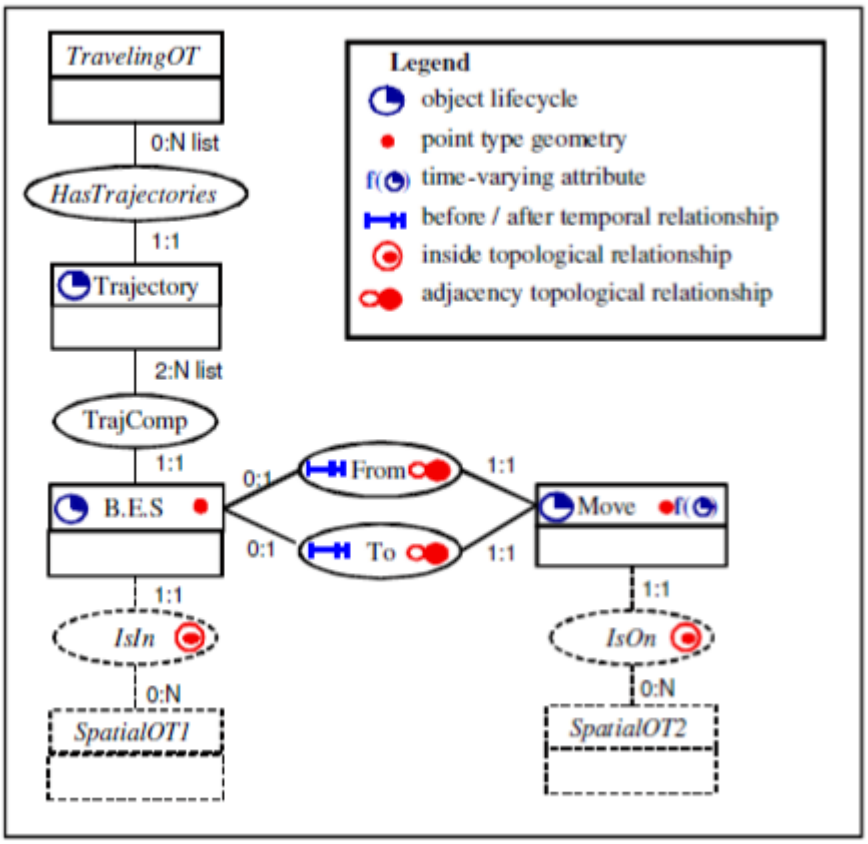
Figura 1 - Dados de uma trajetória bruta e da trajetória com enriquecimento semântico para a área de turismo e gerenciamento de transporte



Fonte: Bogorny e Braz (2012).

O trabalho de Spaccapietra apresenta também um padrão de projeto para representação de trajetórias (Figura). O referido padrão indica que um objeto móvel (*TravelingOT*) possui zero ou mais trajetórias associadas (*Trajectory*). Cada trajetória, por sua vez, é composta por uma lista de dois ou mais componentes. *TrajComp, que* modela uma relação de composição entre a trajetória e seus componentes. Os componentes de uma trajetória são os elementos *Begin*, *End*, *Stop* e *Move*. Na situação em que a trajetória pode ser composta apenas por dois componentes, um deles será *Begin* e o outro *End*.

Figura 2 - Padrão de projeto para representação de trajetórias



Fonte: Spaccapietra et al. (2008).

Na estruturação dos dados de trajetórias brutas o enriquecimento semântico é importante para identificar características dos movimentos (e.g., se o veículo está parado ou em movimento), acrescentando informações relevantes ao contexto do domínio analisado (e.g., quanto tempo a parada demorou, se a parada foi realizada ou não, se o trecho de deslocamento foi realizado ou não). A motivação para conversão de dados brutos de trajetórias em entidades mais abstratas é devido ao

fato de que dificilmente é possível identificar um padrão semântico utilizando apenas os dados brutos armazenados. Naturalmente os dados brutos podem revelar algum padrão geométrico do deslocamento ou interseções dos objetos com locais conhecidos. Qualquer outro tipo de padrão, no entanto, é difícil de obter analisando somente estes tipos de dados. A identificação de pontos de aglomeração de veículos, trechos de velocidade reduzida que podem caracterizar congestionamentos ou eventos na região, dentre outras questões, não são facilmente extraídas se os dados não sofrem algum tipo de pré-processamento.

Na perspectiva de trabalhos que abordam a análise semântica de trajetórias foram estudadas propostas que utilizam diferentes técnicas para anotar semanticamente a trajetória de objetos móveis. Dentre esses trabalhos destacam-se os métodos para identificação de *Stops* e *Moves*, apresentados em (ALVARES et al., 2007; ARBOLEDA et al., 2014; MORENO et al., 2010; PALMA et al., 2008) e os *frameworks* genéricos para anotar semanticamente trajetórias de objetos móveis de variados domínios, desenvolvidos em (BOGORNY et al., 2013; YAN et al., 2011).

Os trabalhos de (ALVARES et al., 2007; ARBOLEDA et al., 2014; MORENO et al., 2010; PALMA et al., 2008), apresentam em sua essência, algoritmos para geração das entidades *Stops* e *Moves*.

O trabalho de (ALVARES et al., 2007) apresenta um modelo semântico de anotação de trajetórias coletadas através de GPS. O modelo é extensível e tem a intenção de poder ser adaptado para diferentes domínios de aplicações. A funcionalidade básica desse modelo é identificar os elementos básicos de uma trajetória, isto é, *Stops* e *Moves*. O objetivo é facilitar as consultas aos dados anotados semanticamente, minimizando o custo computacional envolvido no processamento de dados brutos. A proposta consiste em adicionar um pré-processamento com enriquecimento semântico das trajetórias com informações geográficas para facilitar as consultas, análises e mineração de dados dos objetos móveis. O trabalho produz resultados que evidenciam que ao realizar consultas utilizando as entidades enriquecidas semanticamente se atinge resultados mais rápidos e contextualizados com o domínio da aplicação quando comparado às consultas realizadas com o cruzamento de dados de trajetórias na sua forma bruta.

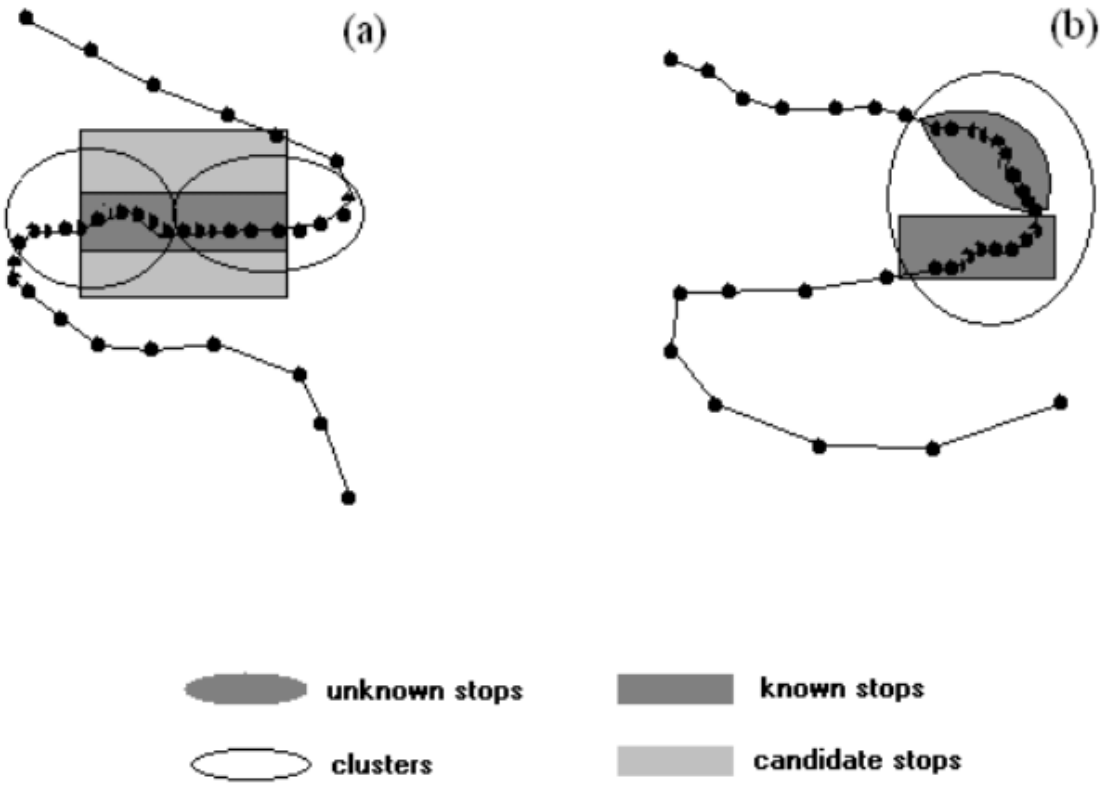
A fase de pré-processamento do trabalho de (ALVARES et al., 2007) se baseia em um algoritmo chamado SMoT (*Stops and Moves of Trajectories*). O referido algoritmo é baseado na atribuição semântica considerando a geografia por

onde passa a trajetória. Ele usa a geografia a fim de identificar os aspectos geográficos que estão relacionados com os *Stops* e, dessa forma, atribuir uma semântica aos mesmos. O SMoT introduz o conceito de *candidate Stop* que consiste em uma tupla composta da geometria de uma área geográfica de interesse e do tempo mínimo definido para identificar uma parada na região.

No trabalho de (PALMA et al., 2008) foi proposto um modelo de anotação baseado na velocidade dos objetos móveis em suas trajetórias individuais. Palma (2008) apresenta uma variação do algoritmo SMoT chamada de CB-SMoT (*Clustering-based SMoT*). O CB-SMoT é um algoritmo que identifica trechos de movimentos lentos em uma trajetória e tenta associá-los a lugares conhecidos. O algoritmo CB-SMoT é composto de duas etapas, clusterização e atribuição semântica. A etapa de clusterização é baseada em uma modificação do algoritmo DBSCAN (ESTER et al., 1996), pois utiliza conceitos espaço-temporais ao invés de apenas a abordagem clássica da densidade para a criação dos clusters. O fato de adicionar o tempo ao processo de clusterização é determinante para identificação de clusters que representam trechos onde a velocidade é lenta, que é uma situação bastante usual no contexto da movimentação de veículos em grandes centros urbanos.

Na etapa de enriquecimento semântico as trajetórias são analisadas com base na localização de pontos de interesse e na velocidade do objeto. Neste trabalho são introduzidos dois conceitos importantes relacionados ao conceito de *Candidate Stop* que são os *Known Stops* e *Unknown Stops*. Os *Known Stops* estão relacionados a pontos de interesse do usuário que existem na base de dados de *Candidate Stops*, discutido anteriormente. Os *Unknown Stops*, por outro lado, são pontos que foram identificados na etapa de clusterização como trechos com alto potencial semântico (devido a sua baixa velocidade), mas que pela seleção de interesse do usuário ou falta de *Candidate Stops* na base de dados não está relacionada a nenhuma entidade/evento geográfico de interesse.

Figura 3 - Alguns possíveis casos de *Known Stops* e *Unknown* Stops a) *Known Stops* podem estar em mais de um cluster b) um cluster que gerou um *Known Stop* e um *Unknown Stop*



Fonte: Palma (2008).

Os algoritmos SMoT e CB-SMot estão implementados na ferramenta de código aberto para a mineração de dados Weka. A ferramenta Weka possui um modulo chamado STPM (*Semantic Trajectory Process Module*) que implementa os algoritmos SMoT e CB-SMot e permite a análise semântica e visualização de trajetórias (BOGORNY; AVANCINI; PAULA, 2011)

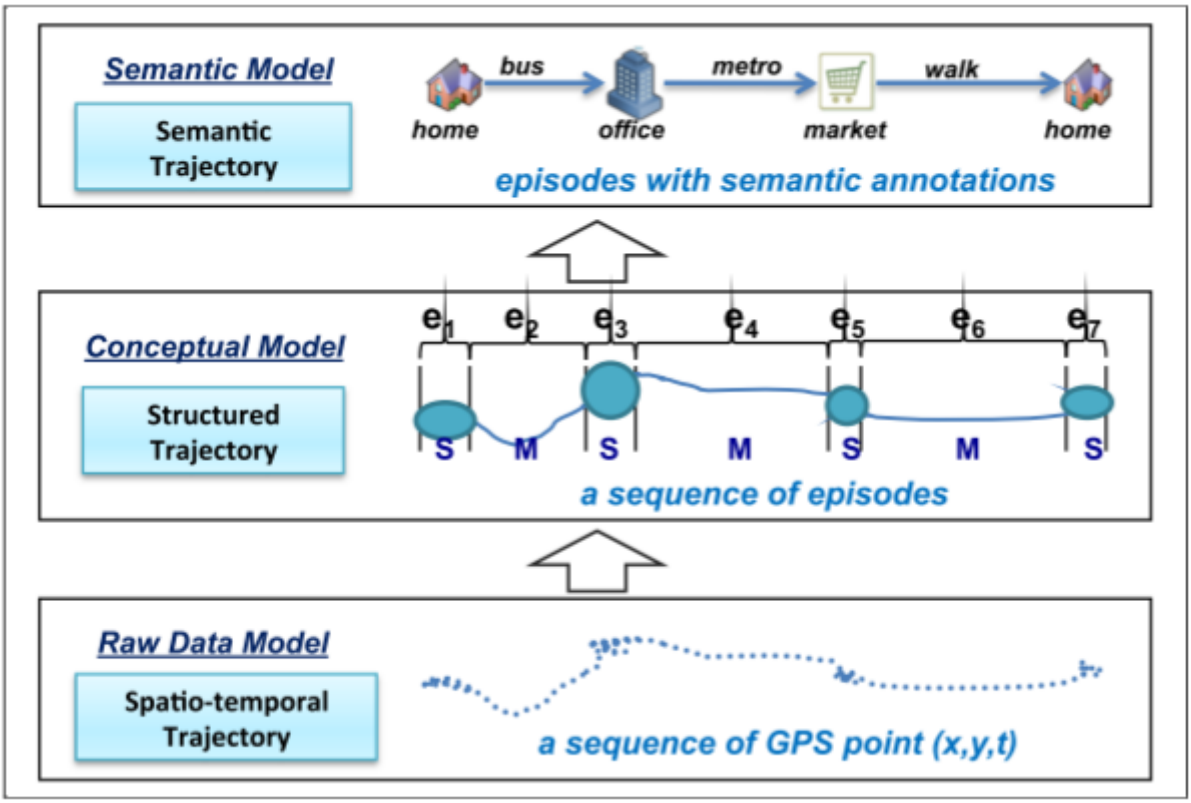
Como evolução natural dos trabalhos já mencionados, o trabalho de Rocha (2010) propôs um algoritmo chamado de DB-SMoT (*Direction-based SMoT*), que aprimorando as ideias do SMoT e do CB-SMoT, adicionou a informação de direção do deslocamento na análise. Arboleda et al. (2014) propuseram uma evolução da proposta original do SMoT chamada de SMoT+, que além de analisar as regiões geográficas isoladamente busca as regiões aninhadas conseguindo descobrir uma hierarquia de *Stops* aninhados. Em 2010, (MORENO et al., 2010) realizou uma análise dos *Stops* da trajetória considerando não apenas medidas como velocidade, direção e tempo, mas também informações semânticas armazenadas em uma base de conhecimento para inferir o objetivo de cada parada. Em 2011 (MORENO;

ARANGO, 2011) incorporou o conceito de trajetória apresentado por Spaccapietra a um *Data Warehouse*, materializando um novo tipo de dado.

Os trabalhos até aqui analisados se basearam fortemente no modelo seminal de *Stops* e *Moves* e no desenvolvimento de algoritmos que permitissem identificar estas entidades e de alguma forma anotá-las com semântica do domínio da aplicação. Outras abordagens evoluíram para a concepção de *frameworks* genéricos capazes de realizar o enriquecimento semântico de diversos tipos de trajetórias.

O *SeMiTri - Semantic Middleware for Trajectories*, é um *framework* para anotação semântica de objetos móveis apresentado em (YAN et al., 2011, 2013). Neste trabalho é demonstrado que o modelo semântico de anotação do *SeMiTri* pode descrever e gerenciar os movimentos de um objeto móvel de forma genérica. Neste trabalho, foi proposta uma especialização dos conceitos básicos de *Stops* e *Moves* com a introdução de entidades com crescentes níveis de abstração, culminando com a entidade *Episode* anotada semanticamente (Figura).

Figura 4 - Modelo híbrido semântico-espacial de trajetórias



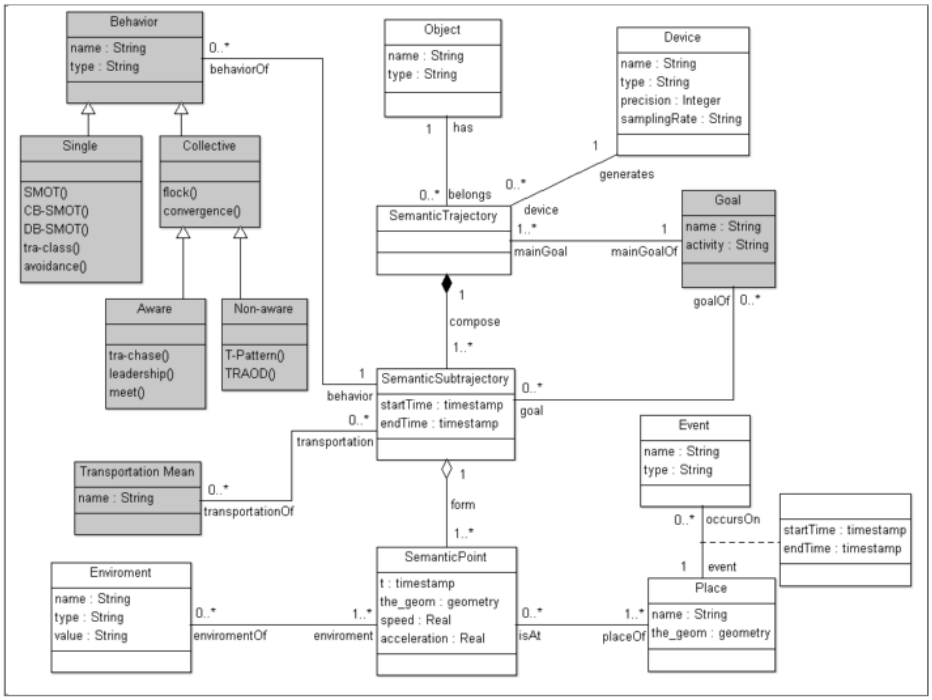
Fonte: Yan et al. (2013).

Em 2013 (BOGORNY et al., 2013), apresentou o CONSTAnT (*CONceptual model of Semantic TrAjecTories*), um modelo conceitual e de dados para a

representação das trajetórias semânticas dos objetos móveis (Figura). O modelo, CONSTAnT surgiu com o objetivo de ser um modelo mais flexível, onde qualquer tipo de informação possa ser acrescentada, enriquecendo semanticamente as trajetórias e suas subtrajetórias.

O modelo CONSTAnT considera que uma trajetória pode ter diferentes comportamentos durante o seu percurso. O modelo utiliza a entidade *Behavior* para identificar o comportamento referente a cada subtrajetória.

Figura 5 - Modelo CONSTAnT



Fonte: Bogorny et al. (2013).

No modelo CONSTAnT, o comportamento do objeto móvel pode ser especializado em comportamentos simples ou coletivo. O comportamento simples analisa o deslocamento de um objeto sem considerar o deslocamento de outros objetos próximos a ele. São informações típicas deste tipo de comportamento as motivações dos diversos *Stops* e *Moves* realizados ou se o objeto evita passar por uma determinada localidade (*avoidance*) (ALVARES et al., 2011). Os comportamentos coletivos visam identificar padrões de deslocamento de um grupo de objetos. Neste caso, é possível identificar se os objetos desenvolvem, por exemplo, um comportamento de perseguição (*chasing*), isto é, um ou mais objetos perseguem outro determinado objeto (SIQUEIRA; BOGORNY, 2011) ou se um

objeto assume a liderança (*leadership*) e coordena a movimentação de outros objetos (LAUBE; IMFELD; WEIBEL, 2005).

Todos os trabalhos discutidos nessa seção apresentam contribuições para a análise de objetos móveis em geral, sem explicitar nenhum domínio específico. A seção seguinte focará nas análises voltadas ao comportamento de movimentação dos veículos no contexto dos Sistemas de Transportes Inteligentes, que é também o foco dessa dissertação.

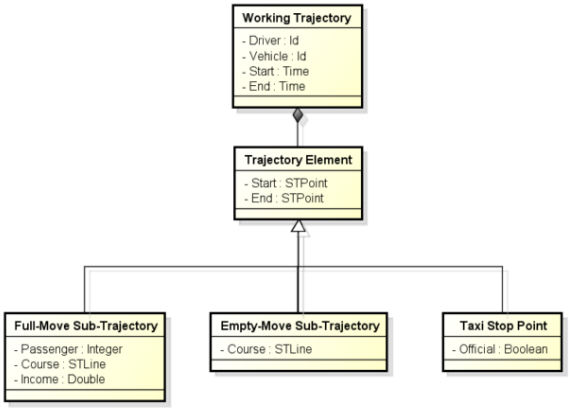
2.2 ANÁLISE DE MOVIMENTAÇÃO DE VEÍCULOS

No contexto de Sistemas Inteligentes de Transportes (ITS, do inglês *Intelligent Transportation Systems*) existem diversos estudos que envolvem a análise e representação da movimentação dos veículos. Estes trabalhos contribuem efetivamente para o entendimento e melhoraria da mobilidade urbana. Dentre os trabalhos que merecem destaque, estão os trabalhos voltados para análise do comportamento dos veículos do sistema de ônibus e táxis.

Os principais problemas abordados nos trabalhos que tratam a movimentação de táxis são: a identificação do padrão de deslocamento dos usuários do sistema em diversos horários do dia e diferentes dias da semana; identificação dos principais pontos de origem e destino e a relação entre essas localidades; recomendação para motoristas e passageiros e o desenvolvimento de ferramentas para medição de eficiência dos motoristas de táxi (AMORIM, 2013). Na revisão detalhada dos trabalhos que tratam da análise da movimentação de taxis realizada por Amorim (2013) foi detectado que a grande maioria dos trabalhos utilizava como fonte de informação dados brutos das trajetórias desses veículos. Dessa forma, foi proposto um modelo conceitual e de dados para a representação da trajetória de táxis, denominado de *TX-Trajectory Model*. O objetivo do referido modelo é a criação de uma infraestrutura de dados com entidades semanticamente relevantes para o domínio da aplicação; a criação de uma estrutura que facilite a consulta e extração de conhecimento sobre a dinâmica deste modal de transporte nos grandes centros urbanos e que também facilite o desenvolvimento de algoritmos eficientes para a descoberta de padrões e conhecimento necessários ao entendimento da movimentação deste tipo de objeto móvel.

O modelo *TX-Trajectory* é baseado na entidade *Working Trajectory* (Figura 6), que representa a trajetória da jornada de trabalho de um motorista de táxi. Nesse modelo, a trajetória é considerada como uma sequência alternada de *Stops* e *Moves*. Diferente da conceituação de Spaccapietra, entretanto, uma *Working Trajectory* é uma combinação de *Full-Move Sub-Trajectory*, *Empty-Move Sub-Trajectory* e *Taxi Stop Point*. As entidades *Full-Move Sub-Trajectory* e *Empty-Move Sub-Trajectory* correspondem aos *Moves* e a entidade *Taxi Stop Point* corresponde ao *Stop*.

Figura 6 - TX-Trajectory – Modelo Conceitual

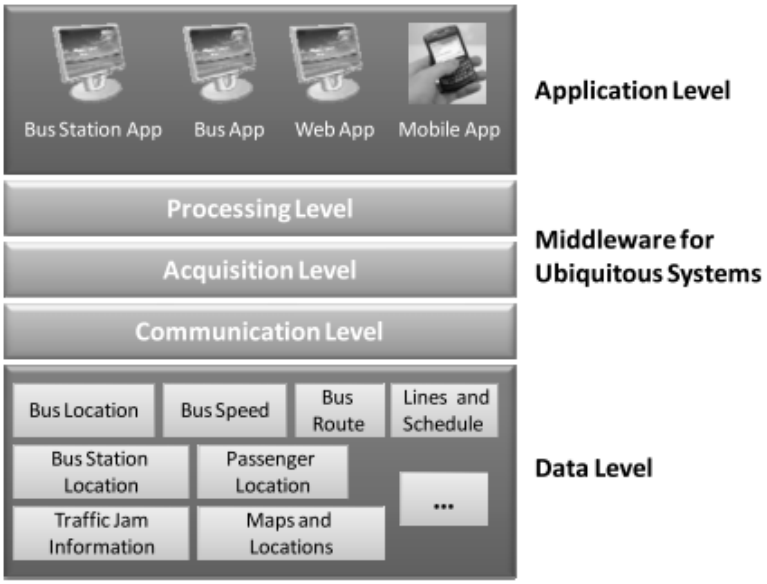


Fonte: Amorim (2013).

O modelo *TX-Trajectory* se mostrou adequado e eficiente para a realização de diversas análises, tais como a taxa de eficiência de cada veículo ou do sistema como um todo; a identificação de zonas e horários de maior demanda e os destinos preferidos; rotas usuais entre dois destinos; entre outras.

Na abordagem de sistemas sensíveis ao contexto aplicados ao conceito de Sistemas Inteligentes de Transportes, em (VIEIRA et al., 2012) é apresentado o projeto UbiBus (UBIBUS, 2011) onde são abordados sistemas embarcados em ônibus para aprimorar a qualidade de atendimento focado no transporte baseado em ônibus. O UbiBus (Figura) foca nos chamados Sistemas de Transporte Público Avançados que são voltados ao transporte público e nas aplicações de Sistemas de Informação aos Usuários (SIU), que têm por objetivo prover informações aos passageiros como, por exemplo, o tempo de espera na parada, previsão de chegada e as rotas dos ônibus, ou seja, informações baseadas em dados obtidos em tempo real.

Figura 7 - Visão geral da arquitetura UbiBus



Fonte: Vieira, Caldas e Salgado (2011).

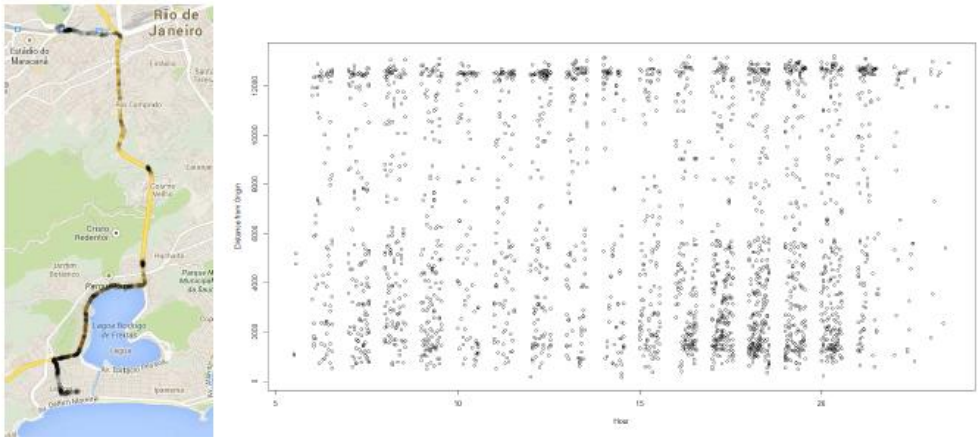
A exemplo do UbiBus, a maioria dos sistemas voltados para a área de transporte público está preocupada em resolver questões operacionais e prover suporte e funcionalidades para os usuários dos sistemas. Desta forma, os dados utilizados por estes sistemas são dados obtidos em tempo real e nenhum mecanismo para análise tática e estratégica para o sistema é suportada.

Um dos poucos trabalhos que se propõe analisar dados históricos dos sistemas de transporte público de ônibus é o Vistradas (BARBOSA et al., 2014). Vistradas é uma ferramenta visual para analisar aspectos comportamentais relacionados às trajetórias de ônibus públicos, tais como: análise de uniformidade de ônibus, a verificação da rota de ônibus, o impacto de eventos no tráfego de ônibus, entre outras.

Um dos maiores diferenciais do Vistradas é a utilização de formas de visualização não tradicionais e que contemplam igualmente aspectos espaciais e temporais dos dados. Um dos exemplos de análise suportada pelo Vistradas é a identificação de aglomeração de veículos da mesma linha. É um fato comum no funcionamento do sistema de ônibus que dois veículos da mesma linha estejam mais próximos do que é esperado para a distância entre eles. Analisando a distribuição espaço-temporal dos veículos de uma determinada linha ao longo do dia e utilizando um sistema de georeferenciamento linear, é possível identificar no mapa

de agrupamento da linha 460 do Rio de Janeiro que existe uma aglomeração de veículos desta linha entre 4 e 8 horas da manhã em um trecho situado a aproximadamente 4 quilômetros do início do itinerário (Figura 9).

Figura 8 - Distribuição espaço-temporal do ônibus da linha 460 ao longo do seu itinerário



Fonte: Barbosa et al. (2014) .

2.3 CONSIDERAÇÕES FINAIS

Este capítulo discutiu as principais iniciativas relativas a representação das trajetórias de objetos móveis. De um lado, não foram identificados trabalhos na área de semântica das trajetórias que tratem do domínio de trajetória de objetos com movimentos condicionados ao cumprimento de um itinerário. A maioria dos modelos trata da representação de trajetórias para movimentos casuais e não programadas. Por outro lado, não foram encontrados trabalhos na área de Sistemas Inteligentes de Transportes, mobilidade urbana ou análise de ônibus que tenham formalizado um modelo de dados ou que tratem da semântica das trajetórias para este domínio. Desta forma, acreditamos que a união destas linhas de pesquisa, até então dissociadas, possui o potencial de facilitar a análise e o entendimento dos dados históricos deste modal de transporte. O próximo capítulo apresenta nossa proposta para integração das áreas de semântica de trajetórias e análise de trajetórias de veículos com movimentos programados.

3 IB -TRAJECTORY – ITINERARY BASED TRAJECTORY MODEL

Atualmente, existe uma grande demanda por aplicações voltadas para o estudo da mobilidade urbana. A área de Sistemas Inteligentes de Transporte (ITS) vem desenvolvendo metodologias de monitoramento do fluxo de veículos e usuários do sistema de transporte para melhorar e facilitar as análises dos diversos modais de transporte, a exemplo dos sistemas de táxis, metrô e ônibus. A concepção de um modelo de dados com entidades mais abstratas e que associe algum tipo de semântica ao movimento dos veículos é indispensável para subsidiar os diversos tipos de análises envolvendo este tipo de objetos móveis. Esta concepção depende fortemente do domínio da aplicação, da granularidade espaço-temporal dos dados coletados, do tipo de movimento e do tipo de espaço onde o movimento é desenvolvido (restrito ou sem restrições).

No contexto da análise da movimentação de veículos em um ambiente urbano, em geral, os dados coletados são oriundos de um mecanismo de posicionamento por satélites (GPS), o que impõe certa precisão na localização e uma frequência de registro bastante elevada, usualmente um registro a cada 30 segundos. Além disso, o espaço geográfico onde o objeto desenvolve o seu movimento é restrito, isto é, os veículos somente podem se deslocar ao longo da rede de transporte, observando o sentido permitido para o fluxo e respeitando ligações e interseções. No contexto da análise da movimentação dos veículos do sistema público de transporte de ónibus, em particular, todos os veículos devem seguir um itinerário pré-estabelecido e obedecer uma escala de horário rígida.

De forma a monitorar e gerenciar a frota de veículos do sistema de transporte público de ônibus quase todas as empresas do setor e as agências regulatórias utilizam a tecnologia de GPS para monitorar a frota de veículos em tempo real. Estas informações instantâneas subsidiam boa parte das decisões e análises realizadas por estes órgãos. Contudo, identificou-se uma carência grande por análise dos dados históricos gerados por estes sistemas de monitoramento. Com o objetivo de melhorar a qualidade deste modal de transporte, surge a necessidade de analisar os dados históricos dos deslocamentos destes veículos. Esta análise tem com o objetivo principal entender os comportamentos dos veículos e identificar pontos de melhoria no sistema de transporte, como por exemplo, identificar

incidência de pontos de paradas não realizadas, desvios de itinerários, trechos e horários de maior congestionamento do itinerário, entre outras.

A massa de dados históricos pode ser aproveitada de diversas formas. Para isso, os dados não podem ser armazenados ou consultados na sua forma bruta, pois exigiria um grande esforço computacional toda vez que fosse necessária a realização de uma determinada análise. Além disso, nem todas as análises podem ser realizadas através do processamento de dados brutos. Para o desenvolvimento de algoritmos para descoberta de padrões e de conhecimento necessários ao entendimento do deste domínio é necessário adicionar algum tipo semântica a estes dados.

Este capítulo apresenta a definição formal do modelo de para a representação da trajetória de veículos condicionadas a cumprimento de um itinerário, detalha o processo de conversão dos dados brutos em entidades do modelo e a persistência das entidades do modelo em um banco de dados relacional.

3.1 DEFINIÇÕES BÁSICAS

Antes da apresentação do modelo conceitual para a representação das trajetórias de ônibus faz-se necessário a definição de alguns conceitos básicos para caracterização do domínio da aplicação. Estas informações não serão necessariamente parte do modelo, mas representam informações de base utilizadas na caracterização de algumas entidades do modelo. São elas:

Transportation Network (TN): Corresponde ao conjunto de todas as vias de uma cidade onde os veículos realizam seus deslocamentos.

Itinerary (I): corresponde a um percurso pré-estabelecido que deve ser realizado pelo veículo ao longo das vias de uma cidade. Todo itinerário começa e termina em pontos de parada, chamados de ponto de parada inicial e final, respectivamente. Ao longo do percurso existem outros pontos de parada (*Known Stop*).

Known Stop (KS): Um ponto de parada do itinerário, é representado por KS = (x, y) em que x e y representam coordenadas geográficas, KS ⊂ I.

Stretch (ST): É um trecho da via ST = { (P_0, P_1, P_2, ...P_n) } em que ST ⊂ I compreendido entre dois pontos de paradas, isto é, P_0 é um KS e P_N é um KS. O somatório do comprimento de todos os trechos é igual ao comprimento do itinerário.

Sub-stretch (SST): Os trechos (*stretch*) podem ser divididos em sub-trechos (*sub*-stretch). Desta forma, um *sub-stretch* é representado como SST = {(P_0, P_1, P_2, ...P_n)} em que SST ⊂ ST. Um SST é um segmento de um ST, que se caracteriza como o trecho entre um ponto de parada conhecida (KS) e uma interseção na rede de transporte (TN), entre duas interseções, ou entre uma interseção e um ponto de parada. Um SST corresponde ao único caminho possível ao longo da rede de transporte para se deslocar entre os pontos ao longo da via, denominados de ponto inicial e final do SST. A soma de todos os *sub*-stretches é equivalente a um *stretch*.

Trajectory (T): É uma sequência de pontos espaço temporais representados por tuplas (identificador, latitude, longitude, tempo), em que a latitude e longitude são coordenadas geográficas. Uma Trajectory é representada por uma lista de pontos espaço-temporal chamada de T { $p_0 = (x_0,y_0,t_0)$, $p_1 = (x_1,y_1,t_1)$, ..., $p_N = (x_n, y_n, t_n)$ }, onde $x_i, y_i \in R$, $t_i \in R+$ para i = 0,1, ..., N, e $t_0 < t_1 < t_2 < ... < t_n$.

3.2 MODELO CONCEITUAL

O primeiro modelo que trata trajetórias de objetos móveis como movimentos que correspondem a deslocamentos semanticamente significativos e, por definição, um conceito espaço-temporal, foi proposto por Spaccapietra et al. (2008). Semanticamente, Spaccapietra considera uma trajetória como uma lista ordenada de *Stops* e *Moves*. Um *Stop* é uma parte da trajetória que é relevante para a aplicação e onde é considerado que o objeto não se movimentou efetivamente, ou seja, permaneceu parado por um determinado período de tempo. Um *Move* é uma parte da trajetória onde o objeto encontra-se em movimento. A representação espacial de um *Stop* é um único ponto, enquanto um *Move* é representado por uma função de deslocamento ou uma polilinha construída com base nos pontos da trajetória.

Baseado em Palma *et al.* (2008), definimos *Stop* e *Move* como:

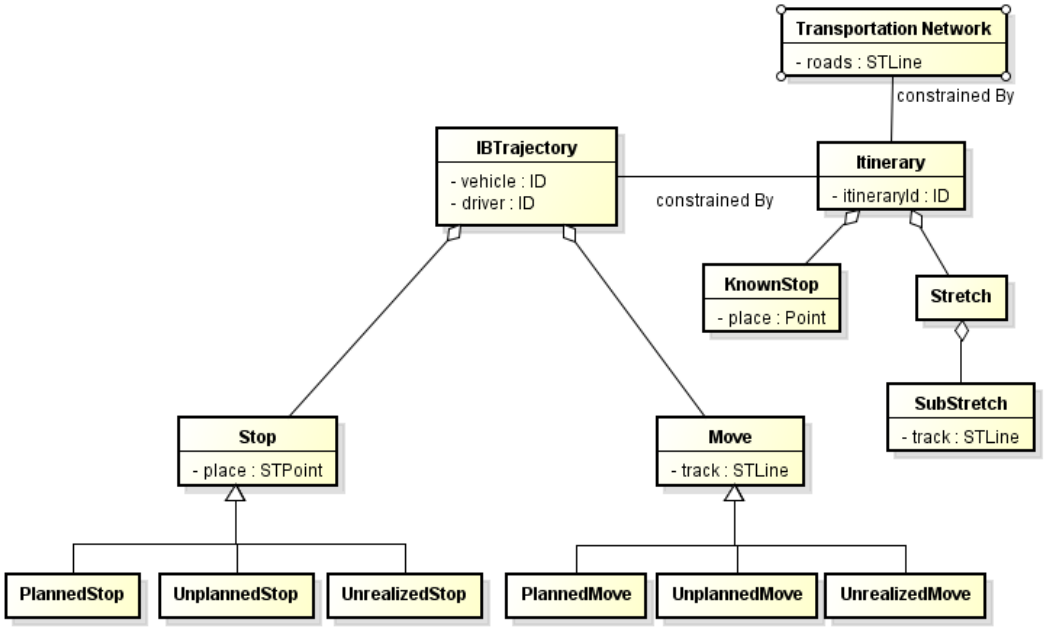
Stop: Um *Stop* de uma trajetória T de uma tupla (R_k, t_j, t_j+n) como uma subtrajetória máxima de T { (x_i, y_i, t_i) | (x_i,y_i) $intercepta$ R_k } = { (x_j, y_j, t_j), (x_j+1, y_j+1, t_j+1), ..., (x_j+n, y_j+n, t_j+n) } , onde R_k é uma geometria de C_k e | $t_j+n – t_j$ | ≥ Δ_k.

Move: Um *Move* de uma trajetória T é: (i) uma contínua subtrajetória de T entre duas paradas consecutivas de T; ou (ii) uma continua subtrajetória de T entre o

ponto de partida em T e o primeiro ponto de parada em T; ou (iii) uma continua subtrajetória de T entre o último *Stop* de T e o último ponto de parada em T; ou (iv) a própria trajetória T, se T não tiver *Stops*.

O modelo seminal de *Stops* e *Moves*, entretanto, é muito abstrato para subsidiar a análise de trajetórias de objetos móveis baseadas em itinerários, para esta finalidade, propomos uma extensão do referido modelo denominada de *Itinerary-Based Trajectory* (*IB-Trajectory*). O modelo *IB-Trajectory* utiliza as informações do banco de dados da rede de transporte, dos itinerários e dos dados brutos coletados dos veículos durante a operação para compor entidades que capturem a semântica do movimento deste tipo de objeto (Figura).

Figura 9 - Modelo *IB-Trajectory*



A entidade básica do modelo é *IBTrajectory*. Esta entidade representa uma única viagem de um veículo com o objetivo de cumprir com um determinado itinerário. *IBTrajectory* é uma composição de tipos de dados abstratos (ou seja, *Stops* e *Moves*). Os tipos de dados abstratos *Stop* e *Move* representam partes distintas da trajetória. A entidade *Stop* representa um ponto de parada do veículo para algum propósito. A entidade *Move* representa segmentos da viagem no qual o veículo está em movimento. Estas entidades, entretanto, não carregam nenhuma informação semântica sobre os aspectos do movimento. Assim, é impossível distinguir, por exemplo, se um veículo parou em um local programado ou em algum

outro lugar por algum motivo desconhecido; ou ainda, se o veículo viajou ao longo do itinerário programado ou seguiu uma rota alternativa. Com o objetivo de capturar a essência de uma trajetória do veículo no contexto de movimentos pré-definidos, as entidades *Stops* e *Moves* são especializadas por três subtipos semanticamente relevantes. Subtipos da entidade *Stops* são *PlannedStop(PS)*, *UnplannedStop(UPS)* e *UnrealizedStop(URS)*. Da mesma forma, a entidade *Move* é especializada através dos subtipos *PlannedMove(PM)*, *UnplannedMove(UPM)* e *UnrealizedMove(URM)*.

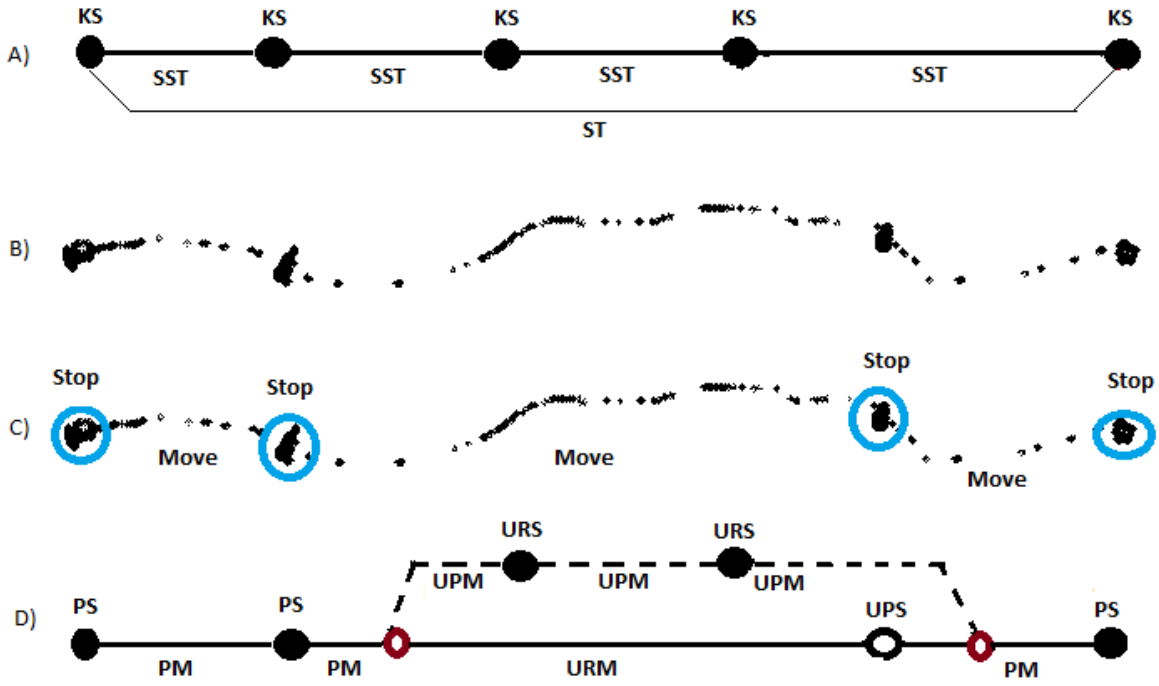
As entidades *PlannedStop (PS)* representam todas as paradas detectadas dos veículos que coincidem com os pontos de parada (*Known Stop*) descritos na base de itinerário. Já as entidades *UnplannedStop (UPS)* representam paradas detectadas que não fazem parte dos pontos de parada definidos na base de itinerário. Finalmente, as entidades *UnrealizedStop (URS)* representam pontos de parada da base de itinerário que não foram detectadas na trajetória coletada.

Quando os veículos estão trafegando, as características do movimento são representadas por subtipos de *Moves* (ou seja, *PlannedMove*, *UnplannedMove* e *UnrealizedMove*). Estas entidades são semanticamente equivalentes às partes estacionárias da trajetória. Uma *PlannedMove (PM)*, por exemplo, representa trechos da trajetória definida na base de itinerário que efetivamente foram percorridos pelo veículo. Entidades *UnplannedMove (UPM)* representam trechos da trajetória realizada pelo veículo que não correspondem a seguimentos de trechos mapeados na base de itinerários, ou seja, são trechos que não estão previstos no itinerário, mas que o veículo realizou. As entidades *UnrealizedMove (URM)* representam trechos que não foram percorridos na trajetória, mas que fazem parte dos segmentos de trecho mapeados na base de itinerários, ou seja, são trechos do itinerário que não foram percorridos.

A Figura representa uma visão geral do processamento de transformação dos dados brutos em entidades semanticamente enriquecidas. A Figura .a representa os dados da rede viária e do itinerário de forma esquemática. A base de itinerários é composta por *Known Stops* (KS), *Stretch* (ST) e *Substretch* (SST) que representam os pontos de parada definidos no itinerário, os trechos e subtrechos do itinerário, respectivamente. A Figura .a mostra a representação do mesmo segmento do itinerário em um mapa. Observe que no itinerário existem cinco pontos de parada (*known stop*) rotulados com as letras A, B, C, D e E.

A Figura .b ilustra os dados de trajetória bruta coletadas por algum mecanismo de posicionamento. A concentração de pontos com as coordenadas dos veículos sugere que houve quatro paradas. Observando o caminho percorrido pelo veículo no mapa (Figura .b), vê-se que o ônibus não cumpriu o itinerário, passando direto e evitando o contorno determinado no itinerário. O processamento dos dados brutos (Figura .c) identificou as quatro paradas realizadas (*Stop*) e os três deslocamentos entre essas paradas (*Move*). Os *Stops* e *Moves* identificados passam por um processo de anotação semântica, gerando as entidades especializadas do modelo *IB-Trajectory*.

Figura 10 - Etapas do processamento: (a) Base de Itinerário (b) Trajetória Coletada (c) Anotação de *Stops* e *Moves* (d) Anotação semântica com Stop e Moves especializados

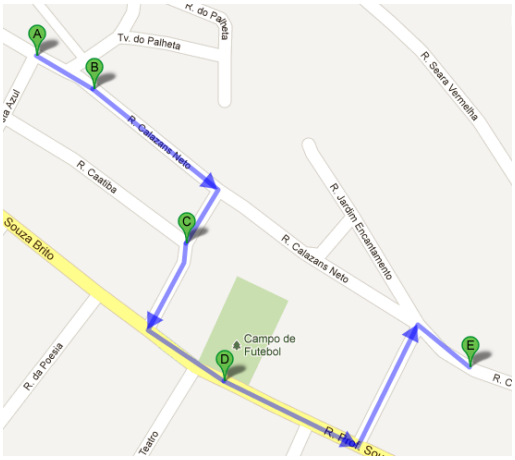


A Figura .d ilustra de forma esquemática a ordem em que estas entidades ocorrem. Das quatro paradas realizadas, três eram previstas no itinerário (PS) e uma parada ocorreu em um local onde não há ponto de ônibus (UPS). Como eram previstas cinco paradas e somente três paradas previstas foram realizadas, foram geradas duas entidades representando as paradas não realizadas (URS). No que se refere aos Moves, o trecho do itinerário por onde o veículo não passou gera instâncias da entidade URM. Se o veículo não cumpriu parte do itinerário, provavelmente ele percorreu trechos da rede viária não previstos no itinerário. Nesse

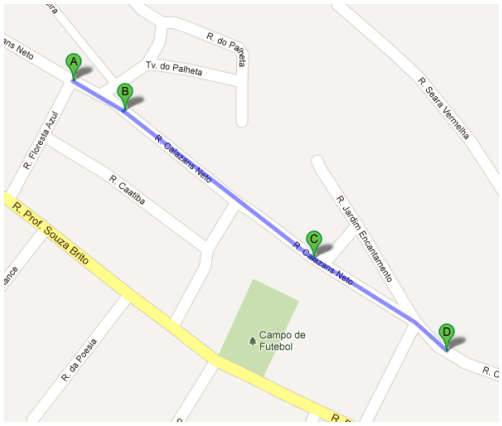
caso, são geradas instâncias de UPM. Finalmente, todos os deslocamentos ao longo das vias do itinerário são representados através de PM (Figura .d).

A Figura .c apresenta as entidades do modelo *IB-Trajectory* no mapa. Os três círculos azuis indicam as paradas previstas e realizadas pelo veículo (PS). O quadrado amarelo a parada realizada fora do local previsto (UPS) e os quadrados vermelhos as paradas não realizadas (URS) devido ao desvio do itinerário realizado pelo ônibus. De forma similar, os deslocamentos realizados dentro do itinerário (PM) foram desenhados com linhas azuis, o trecho não cumprido (URM) foi desenhado com uma linha pontilhada vermelha e o trecho que o veículo percorreu fora do itinerário (UPM) como uma linha amarela.

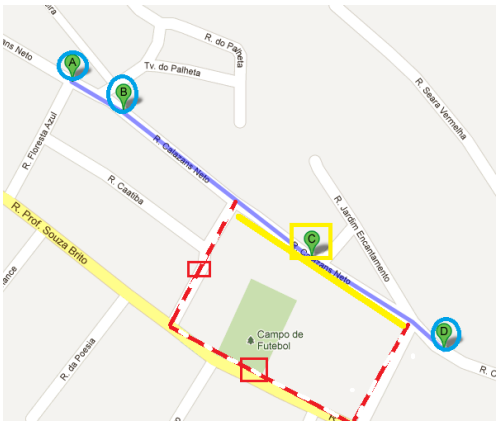
Figura 11 –Apresentação das entidades do modelo *IB-Trajectory* em mapa: (a) Itinerário e paradas previstas (b) percurso realizado pelo veículo e c) instâncias das especializações de *Stops* e *Moves* identificados



a)



b)

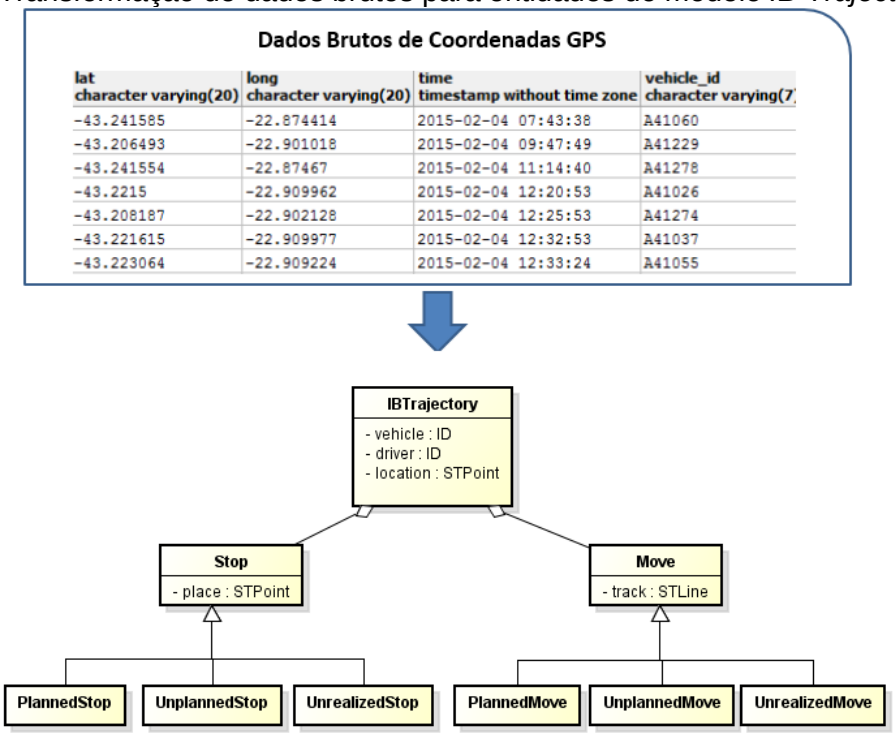


c)

3.3 ANOTAÇÃO SEMÂNTICA DO *IB-TRAJECTORY*

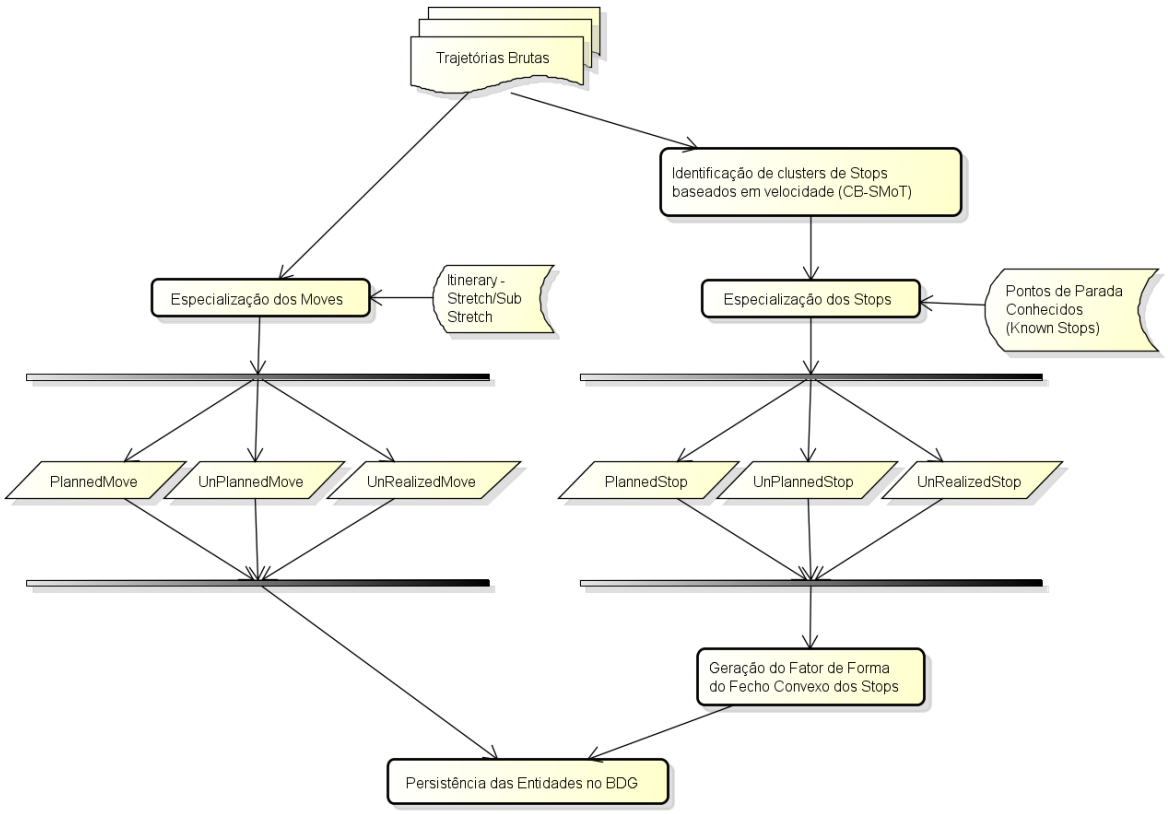
Além do modelo conceitual para representação da movimentação de veículos com trajetórias programadas, este trabalho tem como objetivo desenvolver algoritmos que processem os dados brutos de trajetória e gerem de forma automática as entidades semanticamente enriquecidas do modelo proposto. Esta seção tem como objetivo explicar os detalhes técnicos do processo de conversão de dados brutos em *Stops* e *Moves* e depois a geração das versões especializadas, isto é, *PlannedStop*, *UnplannedStop*, *UnrealizedStop*, *PlannedMove*, *UnplannedMove*, *UnrealizedMove* (Figura).

Figura 12 - Transformação de dados brutos para entidades do modelo *IB-Trajectory*



O processo de transformação das trajetórias brutas em entidades enriquecidas semanticamente é dividido em cinco etapas principais: identificação dos *Stops* e *Moves*, especialização dos *Stops*, especialização dos *Moves*, atribuição semântica e persistência das entidades (Figura). Para um melhor entendimento do processo iremos dividir o fluxo de conversão em dois grandes grupos: especialização dos *Stops* e especialização dos *Moves*. Nas próximas subseções serão discutidos os detalhes de cada uma dessas etapas.

Figura 13 - Fluxo de transformação dos dados brutos para entidades do modelo *IB-Trajectory*



A próxima seção irá discutir o processo de identificação dos *Stops* e *Moves* e a geração das versões especializadas do modelo *IB-Trajectory*.

3.3.1 Algoritmos de Especialização dos *Stops*

Para identificação dos *Stops* foi utilizado o algoritmo de identificação de clusters CB-SMoT (PALMA, 2008). O CB-SMoT utiliza critérios espaço-temporais ao invés de apenas a abordagem clássica de densidade de pontos para a criação de clusters. Ao considerar o tempo no processo de clusterização, o algoritmo consegue identificar agrupamentos de pontos onde a velocidade é baixa, o que é uma característica desejável para aplicações na área de transporte.

Conforme discutido anteriormente, o algoritmo CB-SMoT gera dois tipos de *Stops*: *Known* e *Unknown*. Configuramos o algoritmo para produzir *knownstops* correspondentes a ocorrência de cluster na vizinhança das paradas de ônibus e *UnknownStop* para os demais clusters. Com esta modificação o algoritmo para geração das especializações dos *Stops* se torna trivial, conforme mostrado no pseudocódigo a seguir (Algoritmo).

Algoritmo 1 - Algoritmo de especialização de *Stops*

```
1.  INPUT:
2.
3.  CandidateStops   // Collection of candidate stops from the itinerary
    //database knownstop of IB-Trajectory Model
4.
5.  TrajectoryStops  // Collection of knownstops and unknownstops from CB-SMOT
6.
7.  OUTPUT:
8.
9.  PlannedSTops     // Collection of Planned Stops
10. UnplannedStops   // Collection of Unplanned Stops
11. UnrealizedStops  // Collection of Unrealized Stops
12.
13. METHOD:
14. FOR each stop s in TrajectoryStops DO
15.   IF (s is knownstop)
16.      PlannedSTops <- PlannedStops + new PlannedStop(s);
17.      CandidateStops <- CandidateStops - s;
18.    ELSE IF ( s is unknownstop )
19.      UnplannedSTops <- UnplannedStops + new UnplannedStop(s);
20.    ENDIF
21. ENDFOR
22.
23. FOR each CandidateStops cs in CandidateStops DO
24.   UnrealizedStops <- UnrealizedStops + new UnrealizedStops(cs);
25. ENDFOR
26. END METHOD
27.
```

Como atributos de entrada o algoritmo recebe a lista de todas as paradas registra na base de itinerários (*Candidaters Stops*) *e* a coleção de *Knownstops* e *Unknownstops* geradas pelo CB-SMoT. Todos os *Knownstops* se transformam em *Plannedstops* (linha 14 a 17). Cada *PlannedStop* gerado, elimina o respectivo *Candidate Stop* da coleção de entrada. Todos os *Unknownstops* são transformados em *Unplannedstops* (linha 18 a 20). Finalmente, Os *Unrealizedstops* são gerados baseado na lista de remanescente de *Candidate Stops*, isto é, os pontos de parada que não foram identificados como *Knownstop* pelo CB-SMoT (linha 23 a 25).

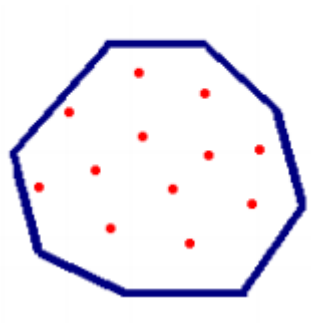
Após o Algoritmo gerar as entidades *Plannedstops*, *Unplannedstops* e *Unrealizedstops*, é calculado o fator de forma para os *Stops* do tipo *Planned* e *Unplanned* através do Algoritmo , e após esta etapa as entidades são persistidas em banco de dados.

O fator de forma foi inserido na semântica do modelo por se mostrar interessante para a caracterização da geometria do cluster de pontos que formam estes tipos de *Stops*.

Para a discussão do fator de forma, faz-se necessário a revisão de alguns conceitos da geometria computacional. De acordo com (CASTRO, 2009), dado um conjunto S com n pontos no plano a união dos pontos mais externos forma um único polígono convexo que contém, junto com a sua fronteira, todos os n pontos do conjunto (Figura). Essa região é a menor área que limita os n pontos e é chamado de fecho convexo (*convex hull*).

Figura 14 - Exemplo de um fecho convexo

$$conv(S) = \bigcap_{K \supseteq S} K$$



Onde:

O conv(S) é um polígono convexo;

O conjunto formado pelos n pontos ⊆ conv(S);

Se existe um conjunto K convexo, S ⊆ K ⇒K ⊇ conv(S)

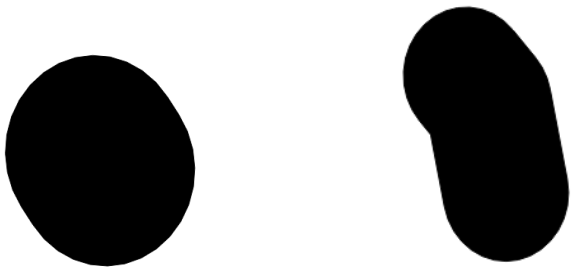
Fonte: Castro (2009).

Uma vez determinada a geometria do fecho convexo, podemos calcular o fator de forma do cluster utilizando a comparação da área do fecho convexo com a mais compacta das formas geométricas, isto é, o círculo (LONGLEY et al., 2006).

Para o cálculo do fator de forma, utilizamos a seguinte fórmula $S = \frac{P}{2\sqrt{\pi}\ \sqrt{A}}$, onde P e A são o perímetro e a área do fecho convexo, respectivamente. Como não existe forma mais compacta que um círculo, o fator de forma varia de 1, para um fecho convexo circular até valores muito maiores que 1, para fechos convexos que se alongam e se afastam da forma do círculo (Figura).

O fator de forma do fecho convexo dos clusters de pontos que formam os *PlannedStops* e *UnplannedStops* é utilizado na realização de algumas análises e será discutido no próximo capítulo.

Figura 15 - Exemplos de fatores de forma distintos a) fator próximo de um círculo b) fator de forma para um polígono alongado.



a) b)

O algoritmo para geração do fator de forma (Algoritmo) é executado logo após a criação dos *Stops* especializados e é igualmente trivial. Neste algoritmo utilizamos operadores existentes no banco de dados espaciais para o cálculo do fecho convexo e sua área e perímetro.

Algoritmo 2 - Algoritmo de fator de forma dos *Stops*

```
1.  INPUT:
2.
3.  PlannedStops // Collection of Planned Stops of IB-Trajectory
4.  UnPlannedStops // Collection of UnPlanned Stops of IB-Trajectory
5.
6.
7.  OUTPUT:
8.  CollectionStops  // Collection of Planned, UnPlanned and UnRealized Stops
9.
10.
11. METHOD:
12.
13. FOR each stop in ( PlannedStops, UnPlannedStops, UnRealizedStops)  DO
14.
15.     area <- area(convexhull(stop.the_geom));
16.     perimeter <- perimeter(convexhull(stop.the_geom));
17.     stop.s  <- (perimeter / (2 * sqrt(Pi) * sqrt(area) );
18.     collectionStops  <- collectionstops + stop;
19.
20. ENDFOR
21.
22. END METHOD
23.
```

3.3.2 Algoritmos de Especialização dos *Moves*

Para a identificação dos *Moves* foi concebido um algoritmo (Algoritmo) que se baseia na comparação geométrica dos pontos da trajetória bruta com os pontos do itinerário sendo utilizado como referência para efeito de comparação.

O Algoritmo tem como entrada os *TrajectoryMoves* que são os conjuntos de dados brutos do deslocamento dos objetos móveis a serem analisados, e o *ItineraryStretch* que são os trechos ou subtrechos que representam o deslocamento esperado para a realização do itinerário.

Os *Moves* são especializados percorrendo todos os dados brutos (linha 11 a 19) e verificando se intercepta ou não a região mapeada pelo itinerário (linha 13), representado pelo *ItineraryStretch*. Se o ponto intercepta ele é agrupado no conjunto de *PlannedMoves* (linha 13 e 14), caso contrário, ele é adicionado ao conjunto dos *UnplannedMoves* (linha 15 e 16). Os trechos do *ItineraryStrech* que não forem identificados como *PlannedMoves* (linha 21 a 24) são agrupados como *UnrealizedMoves.*

Algoritmo 3 - Algoritmo de especialização dos *Moves*

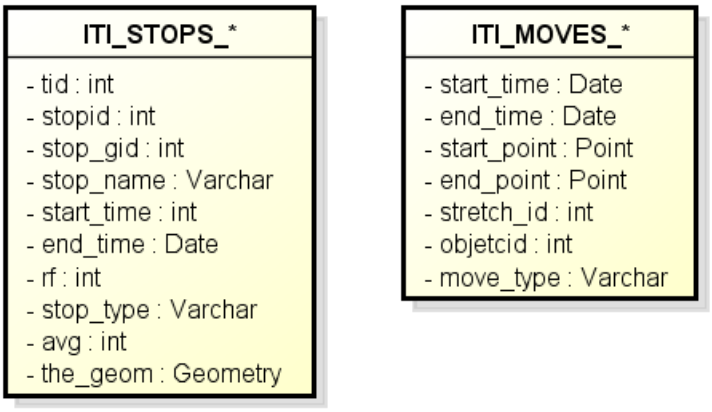
```
1.  INPUT:
2.  TrajectoryMoves  // Collection of Moves obtained from raw trajectories
3.  ItineraryStretch // Collection of stretches of itinerary
4.
5.  OUTPUT:
6.  PlannedMoves     // Collection of Planned Moves
7.  UnplannedMoves   // Collection of Unplanned Moves
8.  UnrealizedMoves  // Collection of Unrealized Moves
9.
10. METHOD:
11. FOR each Point p in TrajectoryMoves DO
12.
13.       IF (p intersects ItineraryStretch)
14.          PlannedMoves <- PlannedMoves + new PlannedMove(p);
15.       ELSE
16.          UnPlannedMoves <- UnPlannedMoves + new UnPlannedMove(p);
17.       ENDIF
18.
19. ENDFOR
20.
21. FOR each Point p in ItineraryStretch DO
22.    IF ( p not intersects (PlannedMoves) )
23.          UnRealizedMoves-> UnRealizedMoves + new UnRealizedMove(p);
24.    ENDIF
25. ENDFOR
26. END METHOD
```

O algoritmo para geração das entidades *PlannedMove*, *UnplannedMove* e *UnrealizedMove* utiliza uma estratégia de força bruta. Por restrições de tempo não foi possível implementar nenhuma técnica de comparação de trajetórias mais

sofisticada. As interseções dos *Moves* com os pontos do itinerário são verificadas utilizando um buffer parametrizado similar ao utilizado para a verificação dos *Stops* no CB-SMoT. Desta forma, o algoritmo possui um alto custo computacional, pois realiza as comparações percorrendo todos os pontos, tanto da trajetória bruta quanto do itinerário base.

Após a etapa de identificação dos *PlannedMoves*, *UnplannedMoves* e *UnrealizedMoves*, as entidades são persistidas em um banco de dados espacial. Para armazenar os *Stops* e os *Moves* especializados são geradas tabelas chamadas respectivamente de ITI_STOPS e ITI_MOVES (Figura).

Figura 16 – Modelo Entidade Relacionamento dos resultados gerados pelo *IB-Trajectory*



Todo o processo de conversão de dados brutos em entidades do modelo *IB-Trajectory* e a consequente persistência em banco de dados está completamente automatizado e incorporado como um módulo da ferramenta Weka-STPM. A próxima seção discute essa implementação.

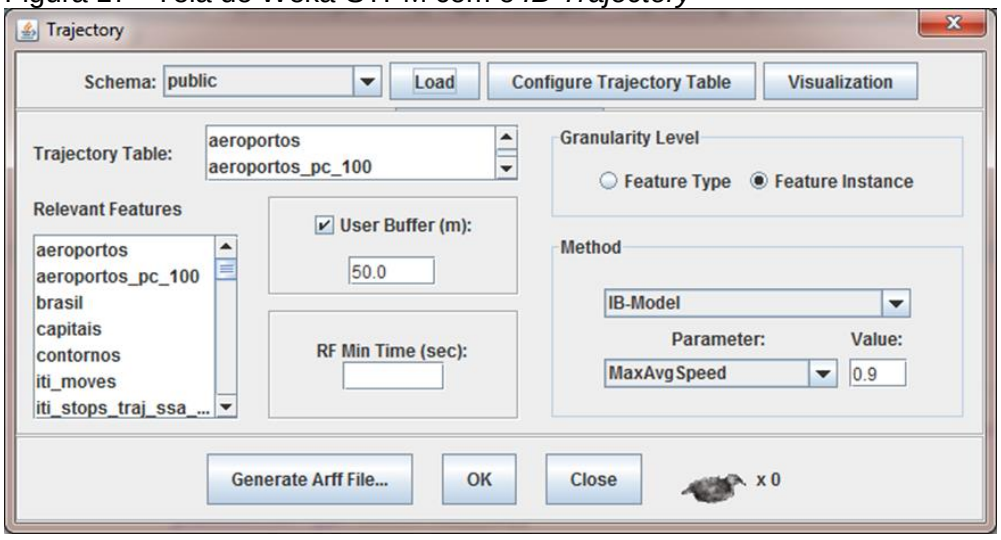
3.4 IMPLEMENTAÇÃO NO WEKA-STPM

Weka é uma ferramenta gratuita e de código aberto, voltada para mineração de dados em bancos de dados tradicionais. É muito utilizada em instituições acadêmicas (FRANK et al., 2005). Uma extensão da ferramenta Weka, com suporte a dados espaciais, denominada Weka-GDPM (*Geographic Data Prepocessing Module*) foi proposta em 2000 (BOGORNY et al., 2000). Em 2011 foi proposta uma nova extensão da ferramenta Weka, denominada Weka-STPM (*Semantic Trajectory Preprocessing Module*) com suporte a dados de trajetória, com enriquecimento

semântico dos dados, data mining em trajetórias semânticas e visualização de padrões de dados de objetos móveis (BOGORNY; AVANCINI; PAULA, 2011).

Neste trabalho foi desenvolvido e incorporado à ferramenta Weka-STPM (BOGORNY; AVANCINI; PAULA, 2011) o módulo IBTGM (*Itinerary-Based Trajectory Generation Module*), mais detalhes da implementação estão apresentados no Apêndice A desta dissertação. O IBTGM é carregado com os dados brutos da trajetória, a base viária da região e com a base de itinerários e paradas conhecidas e produz como saída as tabelas com as entidades *IBTrajetory* e suas versões especializadas dos *Stops* e *Moves* (Figura). Uma vez carregada as informações de base, todo o processo de conversão é feito de forma automática selecionando-se o método *IB-Model* na interface do módulo IBGTM (Figura). Ao selecionar este modelo e definidos os parâmetros de configuração, mais detalhes no Apêndice B desta dissertação, é executado o algoritmo CB-SMoT para identificação de *Stops* e *Moves* e os algoritmos de especialização dos *Stops* e *Moves* apresentados anteriormente.

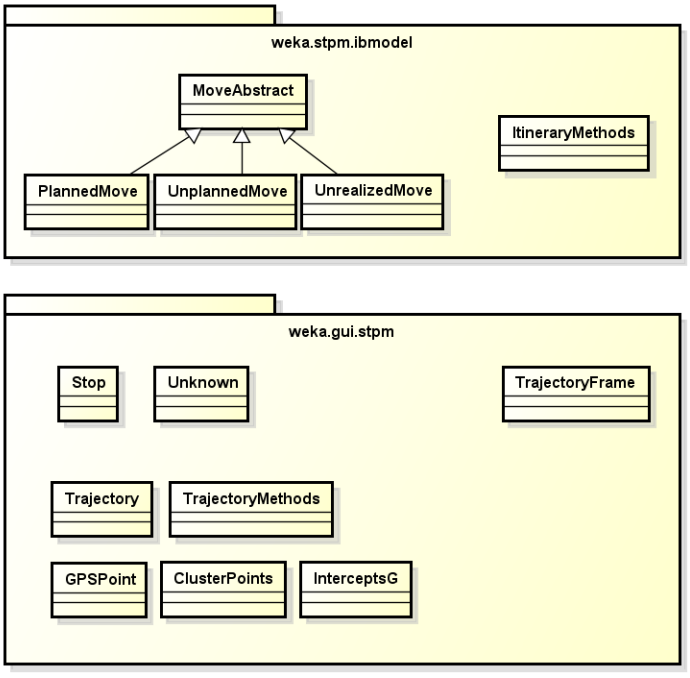
Figura 17 - Tela do Weka-STPM com o *IB-Trajectory*



As modificações realizadas para construção do módulo IBGTM no pacote de classes da ferramenta Weka-STPM estão ilustradas na Figura 1. No agrupamento do pacote weka.gui.stpm são apresentadas as principais classes implementadas para o SMoT e o CB-SMoT. As classes *Stop* e *Unknown* representam respectivamente os *knownstop* e *unknownstop* descobertos pelo algoritmo. As classes *Trajectory*, *GPSPoint*, *ClusterPoints*, *InterceptG* são classes de apoio aos

algoritmos SMoT e CB-SMoT. Estes algoritmos são métodos da classe *TrajectoryMethods*. Finalmente, a classe *TrajectoryFrame* é a responsável pela interface gráfica apresentada na Figura .

Figura 1 - Diagrama das principais classes implementadas no Weka-STPM



.

Para a implementação dos algoritmos de especialização dos *Stops* e *Moves* do *IB-Trajectory*, foram criadas classes no pacote weka.stpm.ibmodel. Neste pacote a especialização dos *Moves* é presentada pelas classes *MoveAbstract*, *PlannedMove*, *UnplannedMove* e *UnrealizedMove* e a especialização dos *Stops* utiliza as mesmas classes já implementadas para o CB-SMoT. Maiores detalhes da implementação no Weka-STPM estão documentados no Apêndice A.

3.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou um modelo conceitual semântico chamado de *IB-Trajectory*. O modelo representa trajetórias de objetos móveis sujeitos ao cumprimento de itinerário. Além do modelo conceitual, foi discutido o algoritmo para conversão de dados brutos de trajetórias em entidades do modelo *IB-Trajectory*. Por fim, foram apresentados detalhes da implementação do modulo IBGTM, responsável pela automatização de todo o processo de geração de entidades do modelo.

Embora o modelo *IB-Trajectory* possa ser utilizado para análise de qualquer movimento de veículos condicionados a movimentos programados, utilizamos como exemplo e prova de conceito algumas análises dos ônibus do sistema dos ônibus do sistema público de transporte de massa. Estas análises são detalhadas no próximo capítulo.

4 ANÁLISES BASEADAS NO MODELO *IB-TRAJECTORY*

O processo conversão de dados brutos de trajetórias em entidades do modelo *IB-Trajectory* requer um conjunto mínimo de informações, a saber: as rotas dos itinerários os pontos de paradas do veículo ao longo do itinerário georeferenciados e os dados brutos da trajetória na forma de tuplas com o identificador do veículo, latitude, longitude e tempo.

Todos os grandes centros urbanos possuem atualmente as informações dos itinerários e dos pontos ônibus do sistema de transporte público e guardam os registros da movimentação desses veículos. Desta forma, adotamos nesse trabalho a estratégia de realizar estudos de casos do modelo *IB-Trajectory* com dados provenientes de diferentes centros urbanos no Brasil. O objetivo desses estudos de caso é: 1) demonstrar que os algoritmos desenvolvidos são compatíveis com diferentes fontes de dados; 2) que as entidades do modelo evidenciam padrões não facilmente detectados nos dados brutos e que as entidades semanticamente enriquecidas do modelo *IB-Trajectory* podem ser incorporadas aos sistemas de análise de transporte, agregando valor e informações ainda não totalmente incorporadas nesses sistemas. Assim, a próxima seção apresenta as análises realizadas com dados reais de quatro grandes centros urbanos (Recife, São Paulo, Rio de Janeiro e Salvador). Para cada estudo de caso, realizamos algumas análises para salientar características relevantes e algumas vezes únicas do modelo.

4.1 ESTUDOS DE CASOS

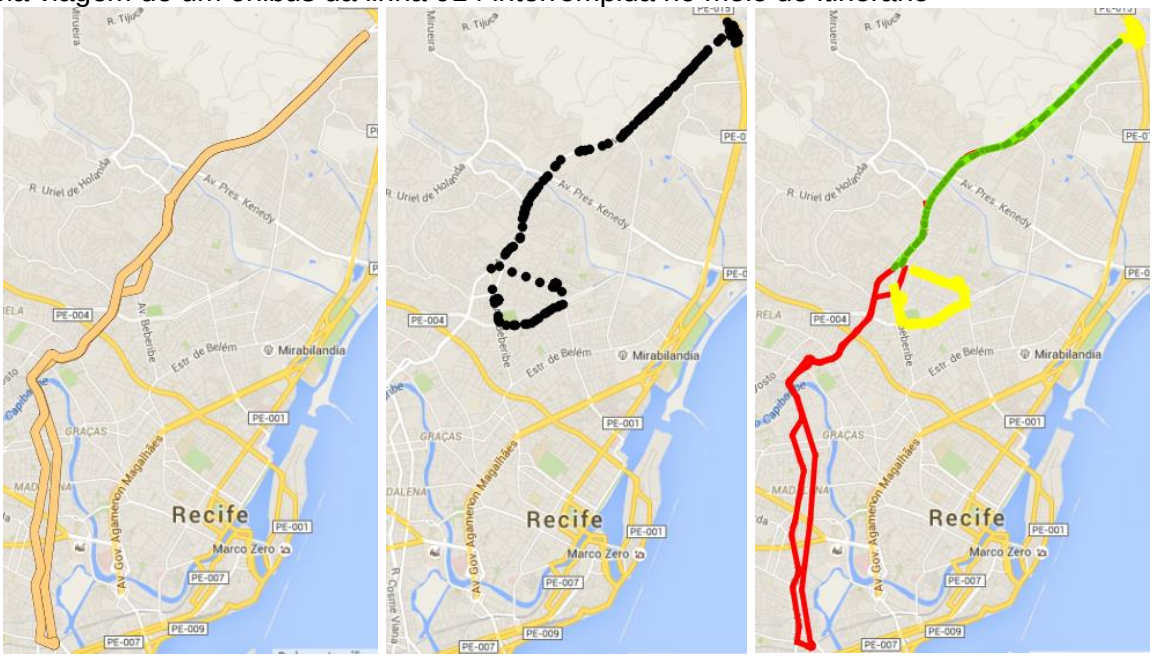
Nesta seção serão apresentados os estudos de casos aplicados a partir do modelo proposto. O modelo foi aplicado em quatro realidades diferentes com o objetivo de validar aspectos como robustez, escalabilidade e flexibilidade para a sua utilização em qualquer centro urbano que possua um sistema de ônibus urbanos rastreados via GPS. Em cada estudo de caso são discutidos diferentes aspectos do modelo.

4.1.1 Recife

O primeiro estudo de caso foi realizado com dados da cidade do Recife. Estes dados foram obtidos de uma amostragem do projeto Ubibus. Este projeto explora a utilização de contexto e computação ubíqua para construir aplicações que auxiliem o passageiro do sistema público de transporte (VELOSO; PHITHAKKITNUKOON; BENTO, 2011; VIEIRA et al., 2012). A base de dados disponibilizada pelo projeto UbiBus corresponde a aproximadamente dez dias de trajetórias brutas de 384 linhas de ônibus com aproximadamente 40 milhões de registros.

A nossa primeira análise envolve uma viagem de um ônibus da linha 914 (Terminal Integrador PE-15/ Terminal Integrador Afogados) realizada no dia 21/08/2012. A Figura .a apresenta uma visão geral do itinerário. Consideramos itinerário de uma linha de ônibus o caminho de ida e volta entre dois pontos extremos. Em nossa base de dados, itinerários são armazenados como uma polilinha, desta forma, a apresentação de um itinerário no mapa é feita com uma linha contínua. Em certos trechos da Figura .a, o itinerário parece uma única linha. Este efeito é causado pelo nível de zoom e pelo fato de que em alguns trechos os itinerários passam pela mesma via, mas em mãos opostas.

Figura 19 - Dados da movimentação incompleta de um ônibus da linha 914: a) apresentação do itinerário da linha 914; b) dados da trajetória bruta; e c) Moves do modelo *IB-Trajectory* uma viagem de um ônibus da linha 914 interrompida no meio do itinerário



a) b) c)

No processo de conversão de dados brutos de trajetória em entidades do modelo *IB-Trajectory*, são identificados, em um primeiro passo, os pontos da trajetória bruta que pertencem a uma entidade *IBTrajectory*. No contexto dos ônibus do sistema público de transporte, uma *IBTrajectory* é uma viagem realizada (ida e volta) com o objetivo de cumprir um itinerário. Para efeito de ilustração, escolhemos duas *viagens* do ônibus da linha 914: a primeira trajetória com um problema no cumprimento do itinerário e outra que respeitou integralmente o itinerário programado.

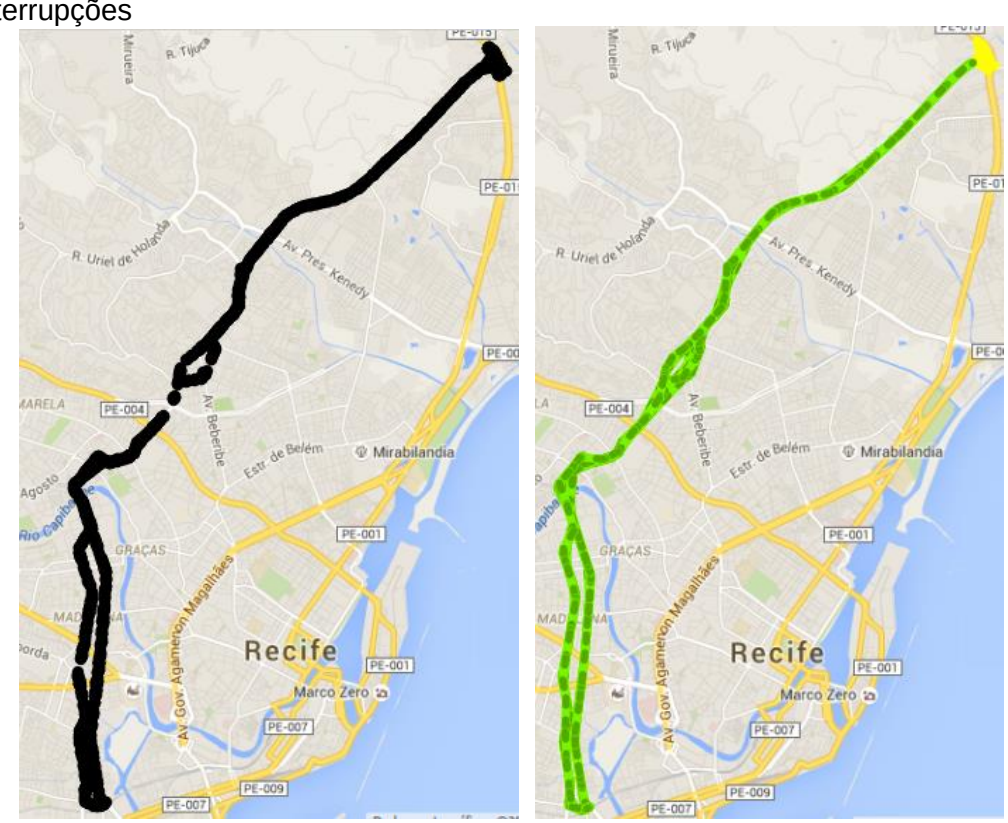
Os dados brutos da primeira trajetória mostram que o ônibus, por algum motivo desconhecido, não cumpriu todo o itinerário (Figura .b). Esta é uma situação bastante comum no sistema de transporte público brasileiro. Os passageiros denunciam esta prática e alegam que, em certos horários, o número de passageiros em alguns trechos é deficitário, por isso, os motoristas desviam ou deixam de cumprir parte do itinerário. A situação mais frequente é a de retornar sem ter atingido um dos pontos extremos do itinerário. As empresas, por sua vez, alegam que todas as viagens são cumpridas integralmente, independentes do número de passageiros no veículo.

Para os dados brutos de trajetória apresentados na Figura .b, o módulo IBTGM adicionado ao Weka-STPM gera entidades do tipo *PlannedMove*, *UnplannedMove* e *UnrealizedMove*, as linhas verde, amarela e vermelhas, respectivamente (Figura .c). Analisando as representações dos *Moves* pode-se inferir que o ônibus iniciou a viagem no canto superior direito e cumpriu o itinerário até certo ponto. No meio do itinerário, aproximadamente, o ônibus realiza um pequeno contorno (linha amarela no centro da imagem), e retorna ao início do itinerário, deixando assim de cumprir a última metade do seu itinerário (linha vermelha no canto inferior direito).

Observa-se também uma ocorrência estranha de uma *UnplannedMove* no início da trajetória (linha amarela no canto superior direito). Verificamos que este evento ocorre em virtualmente todas as viagens da linha 914. Neste caso, foi identificado um problema na base de dados de itinerários. O itinerário da linha 914 não contempla uma pequena volta que o veículo tem que fazer para se reposicionar no início do itinerário. Este fato gera uma ocorrência de um *UnplannedMove* no final de cada trajetória.

Os dados brutos da segunda viagem do ônibus da linha 914 representa uma trajetória sem maiores problemas (Figura .a). O ônibus nesta trajetória cumpriu rigorosamente o itinerário. Desta forma, o processamento dos dados brutos gerou somente instâncias de *PlannedMove*, ou seja, linhas verdes na Figura .b. A exceção fica por conta dos *UnplannedMove* no início da trajetória já discutido anteriormente.

Figura 20 - Dados da movimentação completa de um ônibus da linha 914: a) dados da trajetória bruta e b) Moves do modelo *IB-Trajectory* uma viagem de um ônibus da linha 914 sem interrupções

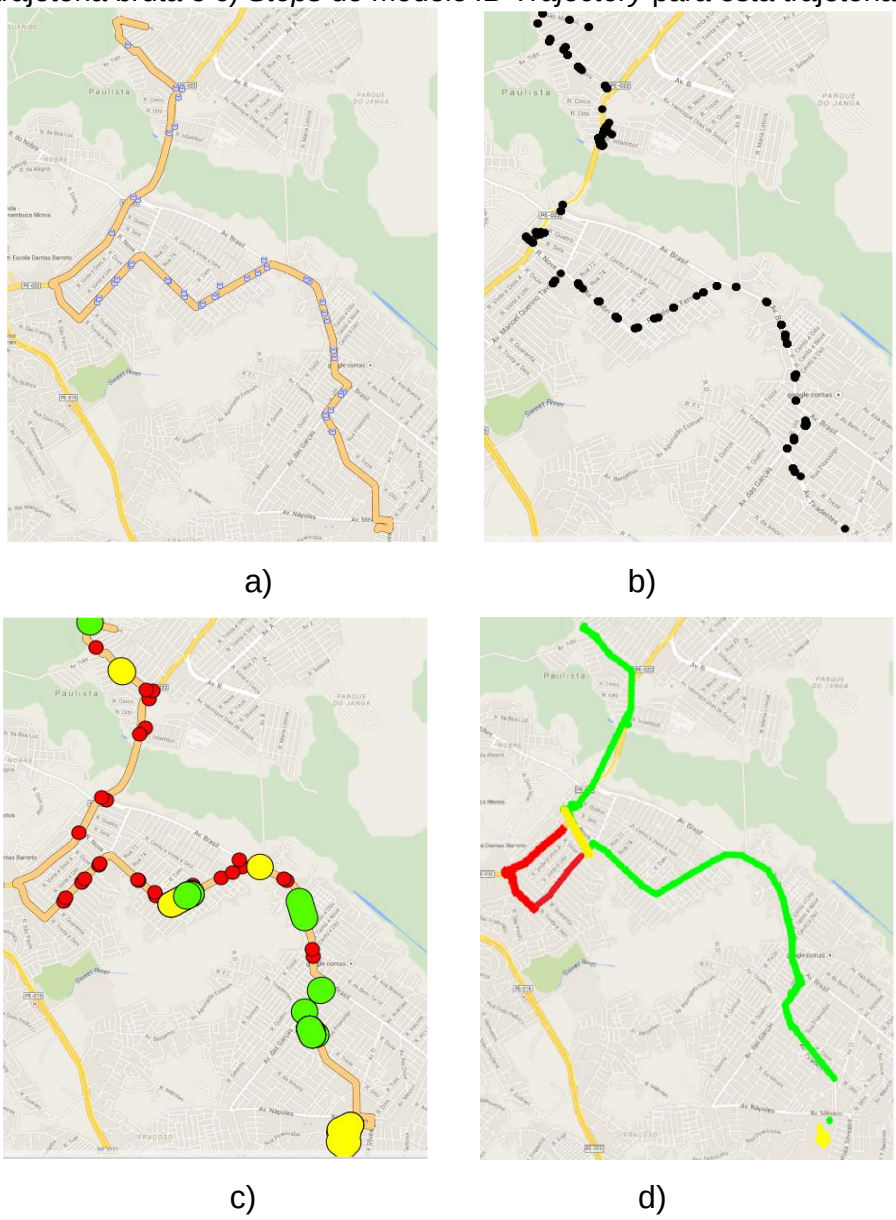


a) b)

Para efeito da discussão da parte estacionária das trajetórias, isto é, os *Stops*, escolhemos estudar uma linha mais curta e alimentadora do sistema. A linha em questão é a 929-Alameda Paulista / Terminal Integrador Rio Doce (Maranguape I).

A Figura .a apresenta o itinerário da linha 929 e a Figura .b os dados brutos da trajetória de uma viagem realizada no dia 21/08/2012. Considerando os dados brutos da trajetória e o registro da localização de todos os pontos de ônibus ao longo do itinerário, o módulo IBTGM gera as instâncias das versões especializadas dos *Stops* (i.e.,*PlannedStop*, *UnplannedStop* e *UnrealizedStop*), representados por regiões verdes, amarelas ou pontos vermelhos, respectivamente (Figura .c). As entidades *PlannedStop* e *UnplannedStop* são representados por uma região pois equivalem ao fecho convexo dos conjuntos de pontos que foram utilizados na geração do *Stop* durante o processo de conversão de dados brutos em entidades do modelo. *UnrealizedStop* por sua vez é representado por um ponto, que é a localização da parada que deveria ter sido feita, mas não o foi.

Figura 21 - Dados de um ônibus da linha 929: a) apresentação do itinerário da linha 929, b) dados da trajetória bruta e c) *Stops* do modelo *IB-Trajectory* para esta trajetória



a) b)

c) d)

.

A análise da parte dinâmica da trajetória indica que o ônibus não cumpriu parte do itinerário, linha vermelha no centro do mapa da Figura .d. Diferente do que ocorreu no exemplo anterior, o veículo chegou ao final do itinerário, mas deixou de cumprir um trecho intermediário. Não possuímos informações ou indícios que expliquem tal comportamento, mas a análise dos dados históricos nos permite descobrir se este é um comportamento usual ou esporádico, se é um comportamento habitual de um determinado motorista ou de um grupo, se existe alguma relação deste comportamento com o horário do dia ou o dia da semana, entre outras.

Consultas envolvendo as diversas especializações de *Stops* podem ser realizadas para determinar se os veículos de certa linha param regularmente em locais não previstos ou se ignora as paradas que deveriam ter sido feitas. As análises envolvendo trechos estacionários irão requerer algumas informações de contexto. Determinar se um veículo está cheio, por exemplo, pode ser uma informação relevante na análise das paradas não realizadas. As condições de trafegabilidade das vias, por outro lado, podem ajudar na eliminação de falsos positivos de paradas não planejadas. As consultas envolvendo as especializações de *Moves* podem apresentar comportamentos de desvios recorrentes e trânsito intenso. Nos resultados apresentados na próxima seção iremos discutir outros aspectos dos *Stops* e *Moves*.

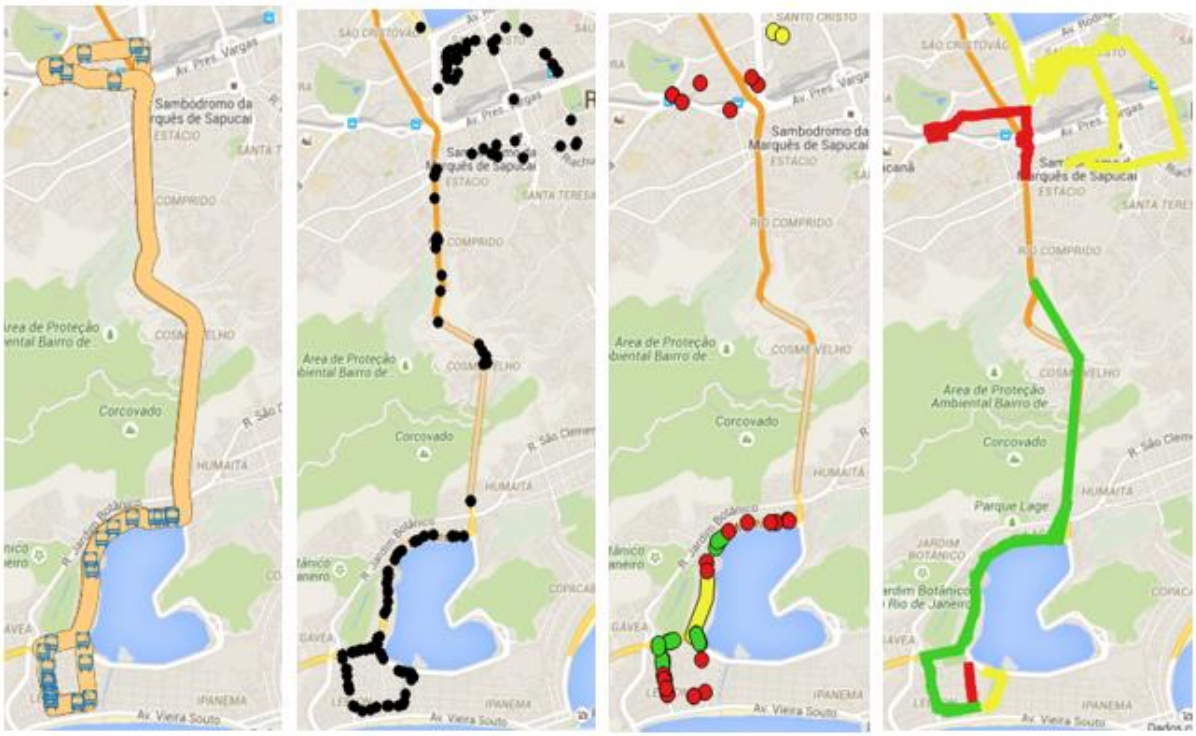
4.1.2 Rio de Janeiro

Esta seção apresenta uma instanciação do modelo *IB-Trajectory* com dados abertos do portal da cidade do Rio de Janeiro (DATARIO, 2015). A iniciativa DataRio oferece acesso público a dados de diversas áreas, tais como administração pública, educação, saúde, entretenimento, meio ambiente e transporte público. Em relação ao sistema de transporte público, é possível obter dados sobre os itinerários dos ônibus, a posição em tempo real e dados históricos de cada trajetória de ônibus. Utilizamos essas informações para instanciar o modelo *IB-Trajectory* com dados reais para realizar algumas análises. Esta análise pode ser conduzida por qualquer pessoa interessada em verificar o sistema como um todo ou o comportamento de um determinado condutor.

Como prova de conceito foi selecionado o itinerário entre dois distritos no Rio de Janeiro: São Cristóvão e Leblon. Selecionamos esta linha porque ela tem características não usuais: os ônibus que atendem esta linha passam dentro de um longo túnel (Américo Rebouças), o que significa que temos perda do sinal de GPS em longos trechos do itinerário. Nesta linha, os pontos de ônibus ou estão em São Cristovão ou no Leblon, não há pontos de ônibus ao longo do itinerário fora destes bairros. A Figura .a mostra a rota do itinerário, iniciando no Leblon (na parte inferior do mapa) e terminando em São Cristovão (no topo do mapa). Ícones azuis indicam os pontos de ônibus ao longo do itinerário.

Nossa primeira análise considera a trajetória de uma única viagem do veículo A1193. Os dados brutos da viagem são mostrados como pontos espalhados ao longo do itinerário (Figura .b). Analisando espacialmente os dados brutos é perceptível a existência de vários registos fora do itinerário. Nós não temos ideia do que motivou esse comportamento, o interessante foi a identificação e representação de um comportamento anômalo. Outra característica saliente dos dados brutos é a ausência de registos dentro do túnel, causada pela perda de sinal de GPS (Figura .b).

Figura 22 - Dados da trajetória de ônibus da linha entre Leblon e São Cristovão: a) Itinerário e paradas de ônibus; b) Dados de uma trajetória bruta; c) Resultados dos *Stops*; d) Resultados dos *Moves*



a) b) c) d)

Os dados de trajetória bruta de uma única viagem do veículo A1193 totalizam 1.484 registros. Estes dados foram processados pelo módulo IBTGM que produziu 54 paradas enriquecidas semanticamente, ou seja, 18 *UnplannedStops*, 13 *PlannedStops* e 23 *UnrealizedStops*) e 15 movimentações enriquecidas semanticamente (ou seja, 4 *UnplannedMoves*, 4 *PlannedMoves* e 7 *UnrealizedMoves*). A representação gráfica dos *Stops* e *Moves* especializados são apresentados com as cores usuais na Figura .c e 22.d, respectivamente.

Analisando as ocorrências de *UnrealizedStops*, pode-se notar que há muitas ocorrências deste tipo de parada ao longo de todo o itinerário. As ocorrências de *UnrealizedStops* na parte inferior do mapa e ao longo da lagoa Rodrigues de Freitas provavelmente foram geradas porque o motorista simplesmente não parou no ponto de ônibus. Os *UnrealizedStops* no topo do mapa, entretanto, têm uma causa diferente. Para identificar uma possível causa, é necessário analisar os *Moves*.

O comportamento do ônibus no início e no final do itinerário é estranho e inesperado. No início do itinerário existe uma ocorrência de um *UnplannedMove* e outra de uma *UnrealizedMove*, isto é, as linhas amarelas e vermelhas na parte inferior do mapa (Figura 2.d). *UnplannedMove e UnrealizedMove* são entidades do modelo que geralmente aparecem em pares. Isto fato ocorre porque quando o ônibus roda fora do itinerário (*UnplannedMove*), ele provavelmente irá ignorar outros trechos (*UnrealizedMove*). Há outra ocorrência do par de entidades *UnplannedMove* e *UnrealizedMove* no topo do mapa (Figura 2.d). No final do itinerário, o ônibus roda ao longo de ruas completamente fora do itinerário (*UnplannedMove)*, deixando de cumprir trechos significativos do roteiro previsto (*UnrealizedMove*). Assim, todas as paradas de ônibus ao longo da entidade *UnrealizedMove* produzem ocorrências de *UnrealizedStop*, o que explica a razão de tantos *UnrealizedStop* nesta região, discutidas anteriormente.

O comportamento do ônibus no meio do itinerário é coerente com o que se espera para um veículo que cumpre o seu itinerário. Neste trecho existem apenas ocorrências de *PlannedMoves*, isto é, não existem parada ou movimentos inesperados. Considerando-se a perda do sinal GPS e a consequente falta de registos nos dados da trajetória bruta, seria razoável esperar a ocorrência de algumas *UnrealizedMoves* nesta região. O algoritmo do módulo IBTGM, entretanto, se mostrou robusto o suficiente para lidar com a perda de sinal do GPS.

As análises das entidades do modelo *IB-Trajectory* por si só não são suficientes para destacar padrões ou comportamentos interessantes. Essas análises devem ser combinadas com outras informações do domínio da aplicação para obter revelar padrões e descobrir relacionamentos.

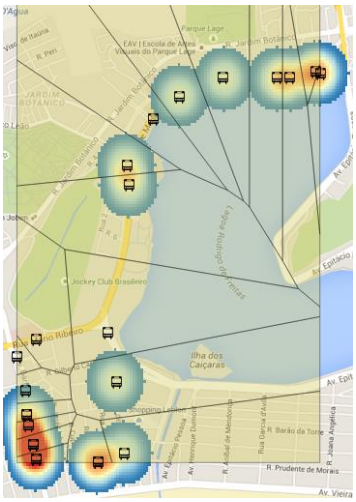
Um possível uso do modelo *IB-Trajectory* é no contexto da análise da distribuição geográfica dos pontos de ônibus. No trabalho de (SILVEIRA; BARROS FILHO, 2006), foi levantada a dificuldade para os gestores de transportes públicos para estabelecer uma distribuição espacial razoável de pontos de ônibus ao longo

de um itinerário. Assim, o artigo apresentou uma proposta para definir a localização dos pontos de ônibus levando em consideração o número de pessoas atendidas em cada parada. Para este cálculo, foram utilizadas informações sobre a densidade populacional na área de influência de cada ponto de ônibus. Os autores utilizaram o diagrama de Voronoi para estabelecer as áreas de influência dos pontos de ônibus.

De volta ao nosso exemplo e considerando que os locais de paradas de ônibus no Rio de Janeiro são estabelecidos usando a abordagem sugerida por de (SILVEIRA; BARROS FILHO, 2006), combinamos a apresentação de todos os *UnrealizedStops* com o diagrama da área de influência ou da área de cobertura todos pontos de ônibus ao longo do itinerário. Esta apresentação objetiva evidenciar padrões nos dados.

Para este estudo, ampliamos as observações da trajetória do ônibus A1193. Os dados de uma única viagem não são suficientes para sugerir um padrão de comportamento do veículo. Desta forma, estendemos nossa análise a uma amostra de 20 dias de observação. Para esta nova janela de tempo foram recuperados 590,988 registros de trajetórias bruta. O processamento desses dados pelo modulo IBTGTM deu origem a 1.336 *PlannedStops*, 3.049 *UnrealizedStops* e 8.005 *UnplannedStops*.

Figura 2 – Mapa de influência dos *UnrealizedStops* dos pontos de ônibus do itinerário da linha entre Leblon e São Cristovão



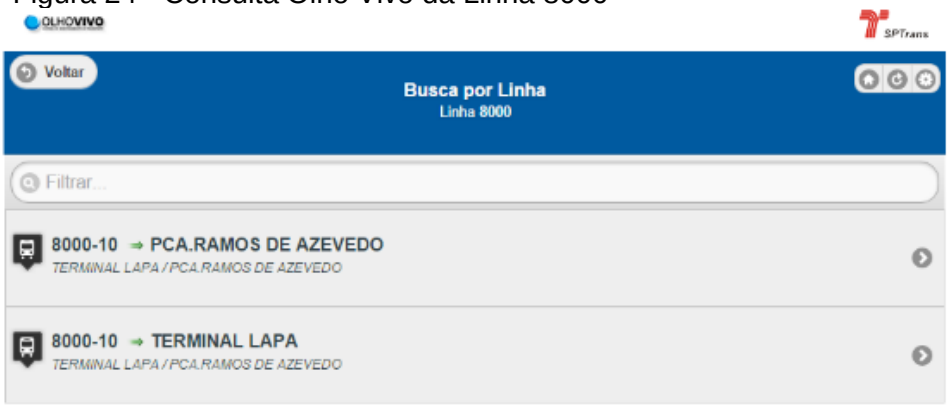
Um mapa de pontos com mais de três mil registros *UnrealizedStops* é confuso e dificilmente evidencia algum padrão nos dados. Esse número de registros é melhor visualizado utilizando uma abordagem de mapa de calor, que se torna uma ferramenta útil quando se deseja apresentar cenários com volume de dados

grandes, por apresentar em cores avermelhadas as regiões com maior incidência de pontos e as regiões com menor incidência tendem a cor azul. Combinando o mapa de calor com os polígonos da visualização da área de influência de pontos de ônibus, percebe-se uma correlação entre estes dados (Figura 2). Os *UnrealizedStops* ocorrem com maior frequência nos pontos de ônibus onde a área de influência é pequena, isto é, onde os pontos são próximos uns dos outros. Este fato pode dar origem a uma hipótese de que a elevada concentração de pontos de ônibus na região pode não ser necessária e que o número de pontos de ônibus e a sua distribuição geográfica precisam ser reavaliadas. Estes resultados seriam mais difíceis de serem identificados utilizando apenas os dados brutos, por conta do volume a ser analisado além da contextualização semântica incorporada.

4.1.3 São Paulo

Esta seção apresenta uma instanciação de modelo *IB-Trajectory* com dados da cidade São Paulo obtidos através da iniciativa Olho Vivo da SPTrans (SPTRANS, 2015). Para este estudo de caso foram analisados os dados do itinerário da linha 8000 (Figura).

Figura 24 - Consulta Olho Vivo da Linha 8000



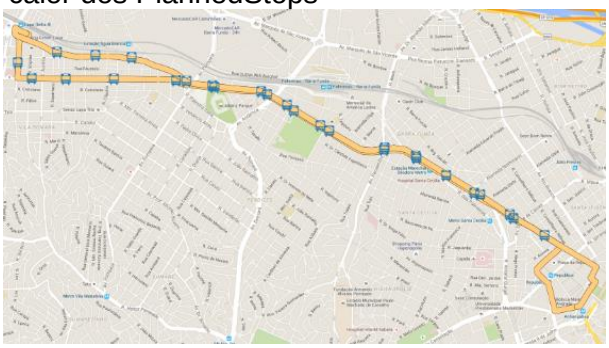
Fonte: Sptrans (2015).

O objetivo desse estudo de caso é ilustrar uma característica única do modelo *IB-Trajectory*, que é a capacidade de avaliar o fator de forma dos *clusters* de pontos que formam *UnplannedStops*. O fator de forma é uma medida importante para diferenciar paradas do veículo devido a engarrafamentos das paradas realmente

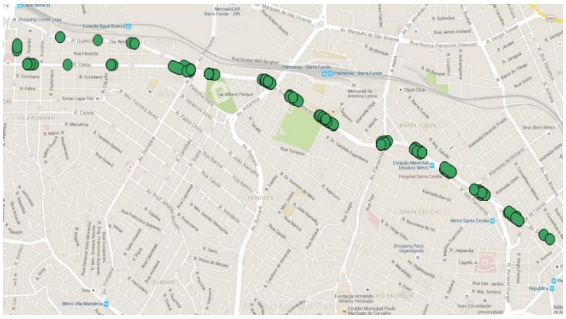
realizadas fora dos locais programados. O modelo *IB-Trajectory* persiste informações sobre o fator de forma das entidades *UnplannedMove* e *PlannedMove*. Isto permite adicionar o fator de forma nas análises realizadas com o modelo sem a necessidade de reprocessar os dados brutos da trajetória.

Para a linha 8000 foi analisada uma amostragem de 29.135 registros de trajetórias brutas produzidas por 68 veículos. Estes dados foram coletados entre os dias 04 e 14 de abril de 2014, com registros da posição do veículo gerados a cada 1 minuto (SPTRANS, 2015). A Figura .a apresenta o itinerário completo juntamente com as paradas programadas do itinerário da linha 8000 nos dois sentidos, Terminal Lapa / Praça Ramos de Azevedo e vice-versa. Os registros brutos da trajetória deram origem a 3.283 registros com semântica. Na mesma figura é possível visualizar somente as ocorrências dos *PlannedStops* através de uma visão de mapa de pontos (Figura .b) e mapa de calor (Figura .c). Na última visualização é possível identificar os pontos onde as paradas planejadas ocorrem com maior frequência. Do ponto de vista de estudos mercadológicos para implantação de uma loja, por exemplo, esta informação possui um grande valor.

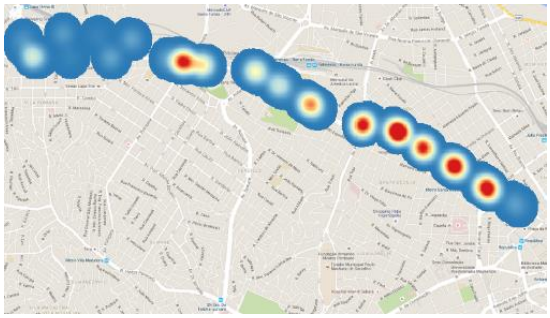
Figura 25 – Dados da linha 8000: a) Itinerário e pontos de parada; b) visualização em mapa de pontos e c) em mapa de calor dos PlannedStops



a)



b)



c)

O interesse maior da análise linha 8000, entretanto, é discutir as ocorrências dos *UnplannedStops*. Um *UnplannedStop* é qualquer parada do veículo fora de uma região em torno de um ponto de parada programada. Quando os *UnplannedStops* são identificados no processo de conversão dos dados brutos em entidades do modelo, o fator de forma do *cluster* de pontos utilizado na determinação da parada é calculado e persistido para todas as instâncias desse tipo de *Stop*. O fator de forma (s) apresenta valores maiores ou igual a 1. Para valores de s próximo a 1, a forma do cluster de pontos se aproxima da forma compacta de um círculo, que é o padrão de paradas intencionais, isto é, paradas que não são causadas pela retenção do fluxo de veículos. À medida que o fator de forma aumenta, cresce o alongamento do cluster de pontos, obtendo uma forma que se afasta gradativamente do círculo. Formas alongadas indicam paradas causadas por engarrafamentos ou por outros eventos que inibam o fluxo de veículos.

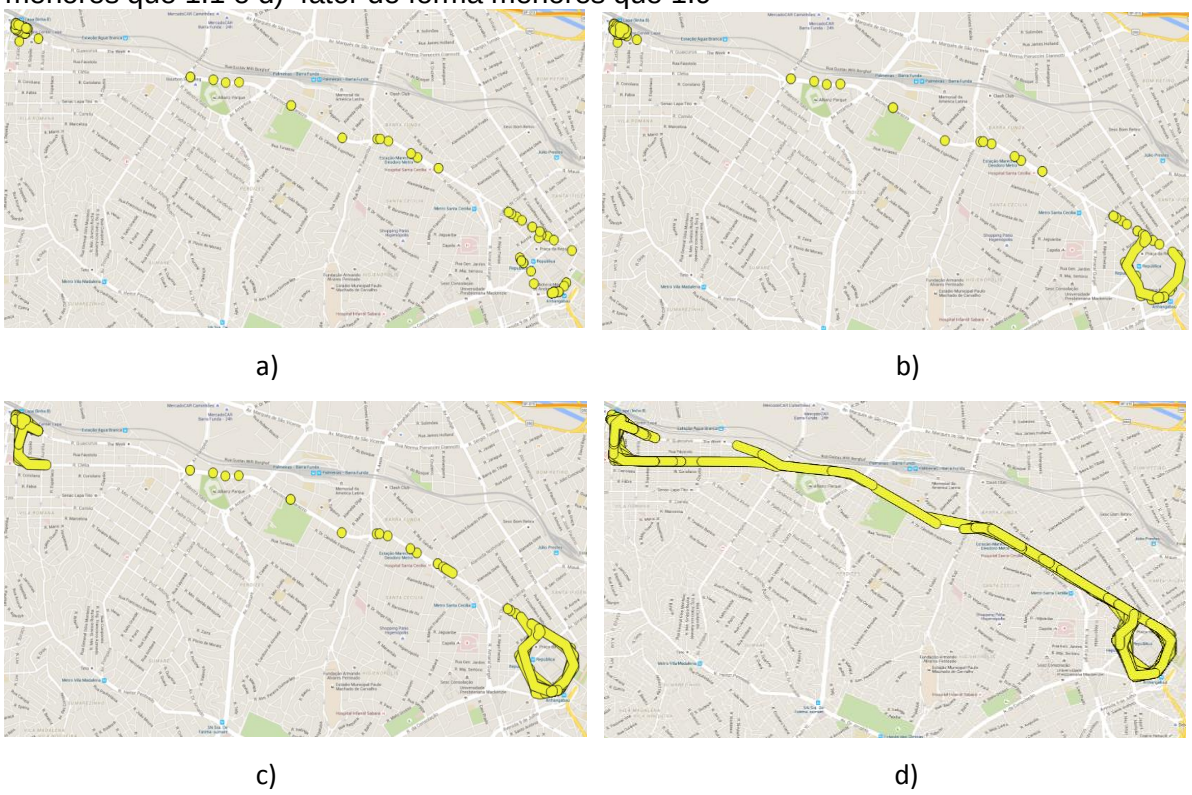
Como o fator de forma é armazenado nas estruturas semanticamente enriquecidas, é possível realizar consultas no banco de dados que envolva as entidades *UnplannedStops* e o fator de forma (s) para produzir diferentes visões dos dados e subsidiar a análise desse tipo de parada. As consultas listadas na Tabela 1, por exemplo, recuperam os *UnplannedStops* para valores crescentes do fator de forma. Cada consulta realizada produz um mapa com a geometria dos *UnplannedStops* (Figura).

Tabela 1 – *Consultas realizadas na tabela de Stops* com diferentes fatores de forma

Consulta 1	select * from iti_stops where stop_type = 'unplanned' and s <= 1.01	Figura 29.a
Consulta 2	select * from iti_stops_traj where stop_type = 'unplanned' and s <= 1.05	Figura 29.b
Consulta 3	select * from iti_stops where stop_type = 'unplanned' and s <= 1.1	Figura 29.c
Consulta 4	select * from iti where stop_type = 'unplanned' and s <= 1.9	Figura 29.d

As imagens resultantes das consultas evidenciam comportamentos extremos do veículo ao percorrer o itinerário. Se por um lado, a ocorrência de paradas não programadas que possuem uma forma compacta no cluster indica a ocorrência de paradas intencionais (Figura .a), por outro lado, formas alongadas indicam que o veículo está parando sistematicamente em curtos espaços de tempo e por longos e consecutivos trechos, este é o comportamento típico de um veículo em engarrafamentos (Figura .d).

Figura 26 – Mapas com diferentes fatores de forma de *UnPlannedStops* na linha 8000: a) fator de forma menores que 1.01, b) fator de forma menores que 1.05, c) fator de forma menores que 1.1 e d) fator de forma menores que 1.9



a) b)

c) d)

Incorporando a dimensão tempo nas consultas envolvendo o fator de forma das paradas não programadas, pode-se evidenciar padrões temporais para este tipo de ocorrência. É natural supor que a ocorrência de *UnplannedStops* com cluster alongados ocorram principalmente nos horários de pico. Já a análise das paradas não programadas e que não foram causadas por algum tipo de retenção do fluxo podem servir de base para estudos de implantação de novos pontos de ônibus ou para posicionar fiscais para coibir este tipo de prática, pois afeta o funcionamento do sistema e atenta contra a segurança do passageiro.

Na próxima seção continuaremos a explorar o modelo em mais um cenário de um grande centro urbano: Salvador. Salvador é uma cidade extremamente dependente desse modal de transporte e qualquer contribuição na análise dos dados tem o potencial de contribuir para a melhora na qualidade de vida de um grande número pessoas.

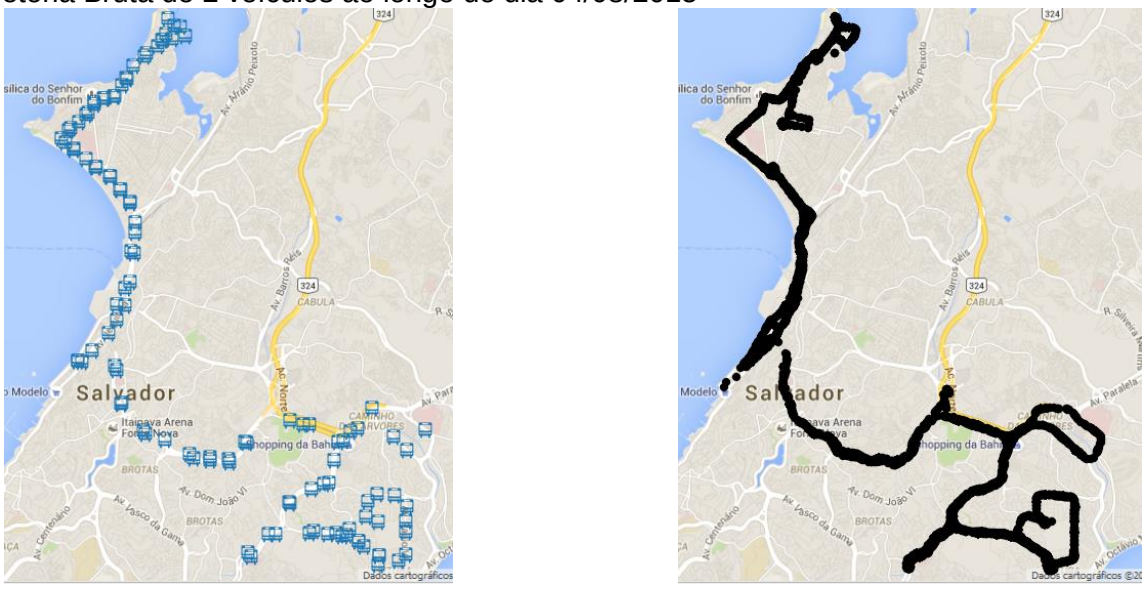
4.1.4 Salvador

No estudo caso de Salvador foram comparados os resultados obtidos com as entidades do modelo *IB-Trajectory* com os relatórios operacionais de uma empresa responsável pela gestão da frota. Esta comparação objetiva evidenciar a aderência do modelo ao domínio de aplicação, demonstrar que o *IB-Trajectory* constitui uma alternativa para mineração dos dados históricos do sistema de transporte público e validar os resultados obtidos através da confrontação da análise de dados históricos com os resultados produzidos pelo sistema de monitoramento em tempo real.

Com relação aos dados necessários ao processamento do módulo IBGTM, foram obtidos somente os dados georeferenciados dos pontos de ônibus e dados da trajetória bruta dos veículos. Os dados dos itinerários não foram fornecidos pela empresa administradora do sistema. Desta forma, só é possível realizar análises com as especializações de *Stops*.

Para efeito desse estudo foi utilizado dados da linha 0202 – Ribeira / Pituba (Circular). Os pontos de ônibus desta linha estão indicados na Figura .a. Foram obtidos também os dados brutos da trajetória dos veículos da referida linha. Os dados são do dia 04 de agosto de 2015. Neste dia somente dois veículos realizaram viagens para este itinerário (Figura .b)

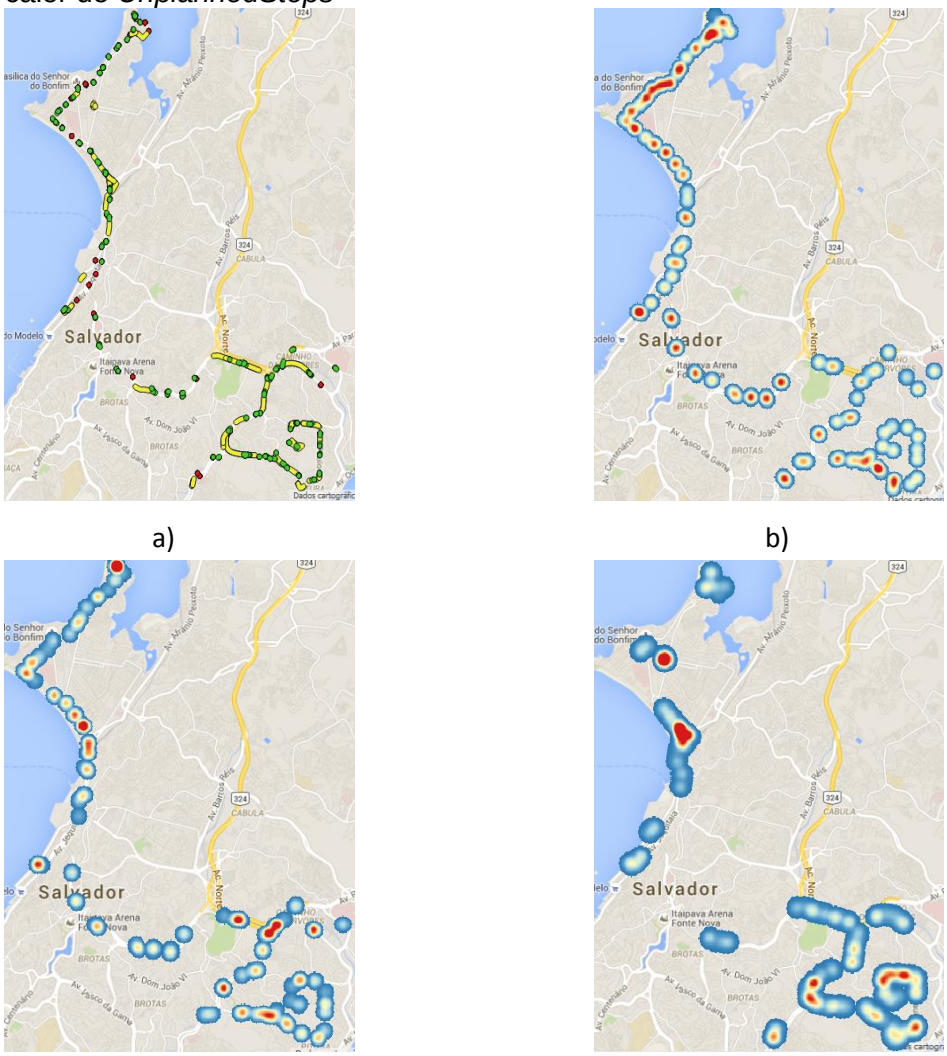
Figura 27 – Dados da linha 0202 - Ribeira/Pituba Circular: a) Pontos de ônibus e b) Trajetória Bruta de 2 veículos ao longo do dia 04/08/2015



a) b)

Devido à falta de informações sobre o itinerário, o módulo IBTGM só é capaz de gerar entidades do tipo *UnrealizedStop*, *PlannedStop* e *UnplannedStop* (Figura .a). Isolando a apresentação dessas entidades e produzindo os respectivos mapas de calor é possível identificar as paradas de ônibus mais negligenciadas pelos motoristas (Figura .b), as paradas de ônibus onde os ônibus param com maior frequência (Figura .c) e locais onde os veículos param mais não deveriam o estar fazendo (Figura .d), indicando provavelmente locais com grande retenção de fluxo.

Figura 28 – Resultado do processamento dos dados brutos dos veículos da linha 0202: a) Mapa de pontos de todos as ocorrências de *UnrealizedStops*, *PlannedStops* e *UnplannedStop*s; b) mapa de calor de *UnrealizedStops;* c) mapa de calor de *PlannedStops*; d) mapa de calor de *UnplannedStops*



a) b)

c) d)

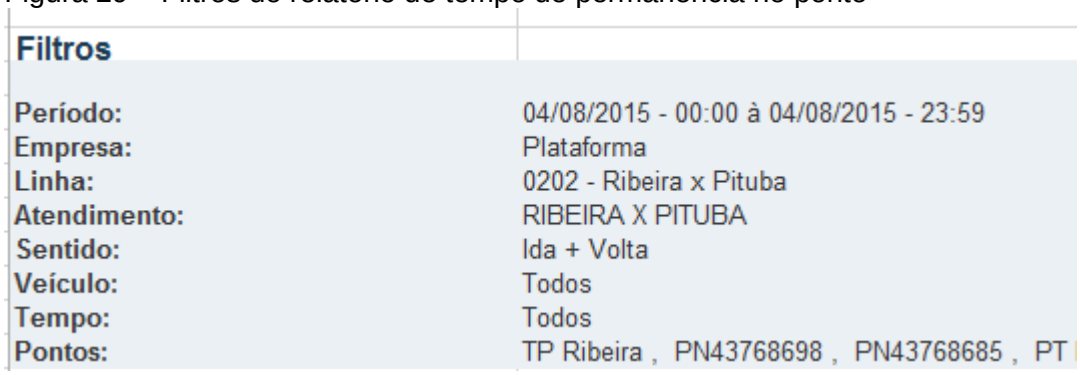
O objetivo do estudo de caso com dados de Salvador é verificar a consistência dos resultados obtidos pela análise histórica dos deslocamentos

quando confrontados com os dados obtidos pela análise de dados obtidos em tempo real.

O sistema de informação utilizado pela empresa que gerencia a frota de ônibus da cidade do Salvador não possui entidades equivalentes às especializações de *Stops* e *Moves* do modelo *IB-Trajectory*. Desta forma, as comparações são limitadas à informações que apresentam alguma similaridade. No sistema da empresa que gerencia o sistema calcula e armazena o tempo de permanência dos veículos em cada ponto. O tempo de permanência é calculado pela diferença dos horários nos quais os ônibus entram e saem da área de influência do ponto de ônibus. A área de influência de um ponto de ônibus é definida como um círculo de 60 metros de raio e centro no ponto de ônibus (isto é, um buffer de 60m).

O sistema de informação permite a emissão de relatórios do tempo de permanência em todos os pontos de ônibus de um determinado itinerário. A Figura mostra a tela de configuração do tempo de permanência para todo o dia 04/08/2015 da linha 0202.

Figura 29 – Filtros do relatório de tempo de permanência no ponto



O relatório do tempo de permanência no ponto de parada (Figura) indica os horários que o ônibus entrou e saiu da área de influência do ponto, a velocidade do veículo nesses instantes, o sentido da viagem (ida ou volta) e tempo que ele levou dentro da área de influência, denominado de “Tempo no Ponto”. O relatório parcial do tempo de permanência da linha 0202 no dia 04/08/2015 identificou 17 paradas no ponto 44265281 (Av. ACM, 3684, Itaigara).

As informações contidas no relatório de permanência no ponto podem ser adaptadas para produzir um relatório de paradas realizadas e não realizadas. De acordo com as boas práticas adotadas pela empresa administradora do sistema,

considera-se que um ônibus realizou uma parada no ponto se o tempo de permanência na área de influência do ponto for superior a 20 segundos. A empresa estabeleceu esse tempo baseado em observações de campo e considerando o tempo mínimo necessário para o ônibus atravessar a área de influência e parar completamente o veículo para o embarque ou desembarque de um passageiro. Utilizando a métrica dos 20 segundos, o trecho do relatório de tempo de permanência (Figura) apresenta 3 paradas realizadas e 14 paradas não realizadas para o ponto de ônibus 44165281. As paradas não realizadas, indicadas com uma seta na Figura , ocorreram aproximadamente às 06:50hs, 12:55hs e 16:07hs.

Figura 30 – Relatório de tempo de permanência no ponto 44165281 dos veículos servindo a linha 0202

Ponto :	44165281 - Avenida Antônio Carlos Magalhães, 3684 - Itaigara, Salvador - BA, Públic					
Chegada		Saída				
Data/Hora	Velocidade	Data/Hora	Velocidade	Sentido	Tempo no Ponto	
4/8/2015 18:56:05	38,00 Km/h	4/8/2015 18:56:08	39,00 Km/h	Volta	00:00:03	
4/8/2015 17:56:23	19,00 Km/h	4/8/2015 17:56:31	32,00 Km/h	Ida	00:00:08	
4/8/2015 14:44:07	10,00 Km/h	4/8/2015 14:44:17	32,00 Km/h	Ida	00:00:10	
4/8/2015 12:26:56	55,00 Km/h	4/8/2015 12:26:58	57,00 Km/h	Volta	00:00:02	
4/8/2015 11:33:52	48,00 Km/h	4/8/2015 11:33:57	45,00 Km/h	Ida	00:00:05	
4/8/2015 08:53:50	23,00 Km/h	4/8/2015 08:53:58	41,00 Km/h	Ida	00:00:08	
4/8/2015 06:11:00	49,00 Km/h	4/8/2015 06:11:02	43,00 Km/h	Volta	00:00:02	
4/8/2015 05:51:05	60,00 Km/h	4/8/2015 05:51:08	60,00 Km/h	Ida	00:00:03	
4/8/2015 06:49:12	13,00 Km/h	4/8/2015 06:50:20	15,00 Km/h	Ida	00:01:08	←
4/8/2015 07:34:08	51,00 Km/h	4/8/2015 07:34:09	50,00 Km/h	Volta	00:00:01	
4/8/2015 10:06:27	19,00 Km/h	4/8/2015 10:06:35	26,00 Km/h	Ida	00:00:08	
4/8/2015 10:56:01	36,00 Km/h	4/8/2015 10:56:02	36,00 Km/h	Volta	00:00:01	
4/8/2015 12:54:10	11,00 Km/h	4/8/2015 12:55:27	18,00 Km/h	Ida	00:01:17	←
4/8/2015 13:35:41	39,00 Km/h	4/8/2015 13:35:44	36,00 Km/h	Volta	00:00:03	
4/8/2015 16:06:27	24,00 Km/h	4/8/2015 16:07:41	20,00 Km/h	Ida	00:01:14	←
4/8/2015 19:38:35	14,00 Km/h	4/8/2015 19:38:44	29,00 Km/h	Ida	00:00:09	
4/8/2015 20:17:24	53,00 Km/h	4/8/2015 20:17:25	53,00 Km/h	Volta	00:00:01	

De forma a comparar os dados obtidos da empresa com entidades geradas pelo módulo IBGTM, selecionamos todas as paradas programadas (PM) no mesmo ponto de ônibus. Os resultados obtidos utilizando o *IB-Trajectory* detectaram somente duas paradas, uma às 16:07hs e outra às 12:55hs (Figura). A parada identificada no relatório gerencial da empresa ocorrida às 06:50hs não foi identificada no nosso modelo.

Figura 31 – Resultados do *IB-Trajectory* obtidos para o ponto 44165281 realizado por dois veículos

objectid character varying	stop_gid character varying	start_time timestamp without time zone	end_time timestamp without time zone	stop_type character varying
34354	272	2015-08-04 16:07:18	2015-08-04 16:07:41	planned
34354	272	2015-08-04 12:54:39	2015-08-04 12:55:27	planned

De forma a entender o motivo da discrepância entre as informações geradas pelas aplicações, realizamos uma análise mais detalhada dos dados brutos da trajetória neste horário. Assim, selecionamos os registros da trajetória bruta em torno das 06:50hs. Para este horário foram identificados 5 registros (Figura .a). Os registros brutos podem ser visualizados também no mapa (Figura .b). Nota-se que a velocidade associada a cada ponto na tabela indica uma grande oscilação, mesmo para pontos espaço-temporalmente próximos, a exemplo dos registros 11206 e 11207 e os registros 11209 e 11210. Esta variação de velocidade indica um provável erro no dispositivo de posicionamento. Para efeito da detecção de *Stops* no algoritmo CB-SMoT, a variação de velocidade é determinante na identificação de *Stops*, o que justifica a não identificação dessa parada.

Figura 32 – Recorte de dados brutos do veículo 34354 no trecho do ponto 4416528: a) visão dos registros dos dados brutos; b) visualização geospacial dos dados

gid integer	id bigint	veiculo character	x character var	y character va	time timestamp without time z	velocidade character v
11206	34928	34354	-12.9881	-38.468456	2015-08-04 06:48:58	0
11207	34928	34354	-12.988221	-38.468773	2015-08-04 06:49:12	13
11208	34928	34354	-12.988317	-38.468884	2015-08-04 06:49:28	0
11209	34928	34354	-12.988319	-38.468887	2015-08-04 06:49:58	0
11210	34928	34354	-12.988221	-38.468773	2015-08-04 06:50:20	15

a)

b)

Uma outra análise dos dados de Salvador poderia ser identificar os pontos de paradas que os ônibus mais realizaram paradas (PS). No contexto do modelo *IB-*

Trajectory, as paradas mais contempladas podem ser obtidas através da seguinte consulta:

```
select * from (select stop_gid, objectid, count(*) as qtd
from iti_stops_traj_linha_0202_1_as_0_9_mt_15_ms_1_1
where stop_type = 'planned' group by stop_gid, objectid ) as tab
order by tab.qtd desc
```

Como resultado da consulta, podemos observar que entre os 10 maiores pontos de paradas com PM estão as paradas 237 e 253 que correspondem as paradas: “TP Ribeira” e “PT Minipreço”, respectivamente. A parada 237 possui dez paradas do veículo 34224 e 9 paradas do veículo 34354. Já a parada 253 só possui 7 paradas do veículo 34224 e 5 paradas do veículo 34354 (Figura).

Figura 33 – Top 10 das paradas mais contempladas de acordo com o *IB-Trajectory*

stop_gid character varying	objectid character varying	qtd bigint
237	34224	10
237	34354	9
253	34224	7
310	34354	5
250	34224	5
274	34224	5
253	34354	5
310	34224	5
304	34224	4
311	34224	4

Confrontando os dados do relatório operacional apresentado na Figura , identificamos que o primeiro ponto se trata do terminal da Ribeira. Este ponto possui paradas com tempos superiores a 30 minutos, o que sugere que o terminal é utilizado parada para descanso ou ajuste de horário do veículo à linha. Desta forma, é natural que este seja o ponto com maior número de paradas planejadas.

Figura 34 – Relatório de paradas do Terminal da Ribeira da linha 0202

Ponto :	TP Ribeira - Largo da Ribeirão, 524-592 - Ribeira, Salvador - BA, 40421-530, Brasil					
Chegada		Saída				
Data/Hora	Velocidade	Data/Hora	Velocidade		Sentido	Tempo no Ponto
4/8/2015 16:31:02	31,00 Km/h	4/8/2015 16:54:52	25,00 Km/h		Ida	00:23:50
4/8/2015 13:38:24	31,00 Km/h	4/8/2015 13:55:12	19,00 Km/h		Ida	00:16:48
4/8/2015 10:45:59	27,00 Km/h	4/8/2015 10:55:15	31,00 Km/h		Ida	00:09:16
4/8/2015 07:09:02	32,00 Km/h	4/8/2015 07:40:45	29,00 Km/h		Ida	00:31:43
4/8/2015 05:14:07	21,00 Km/h	4/8/2015 05:19:15	25,00 Km/h		Ida	00:05:08
4/8/2015 06:03:30	26,00 Km/h	4/8/2015 06:05:58	25,00 Km/h		Ida	00:02:28
4/8/2015 08:38:24	33,00 Km/h	4/8/2015 09:10:52	26,00 Km/h		Ida	00:32:28
4/8/2015 11:54:41	31,00 Km/h	4/8/2015 12:10:58	26,00 Km/h		Ida	00:16:17
4/8/2015 14:37:05	29,00 Km/h	4/8/2015 15:12:47	14,00 Km/h		Ida	00:35:42
4/8/2015 18:14:44	23,00 Km/h	4/8/2015 18:49:43	12,00 Km/h		Ida	00:34:59

O segundo ponto de parada mais contemplado é o “PT Minipreço”. Gerando-se uma visualização no mapa dos pontos da trajetória na região, observa-se que a região do ponto do Minipreço possui realmente uma alta concentração de pontos de trajetória (Figura). Este ponto do Minipreço trata-se de um mercado de grande fluxo na região da Ribeira o que se justifica como um grande ponto de utilização dos usuários do sistema de transporte.

Figura 35 – Visão no mapa dos dados brutos próximos do Minipreço

A inspeção visual dos dados brutos é confirmada pelo relatório operacional do ponto “PT Minipreço” (Figura). No relatório é possível identificar que dos 21 registros de parada, apenas 7 não atendem ao critério de tempo mínimo, que considera uma parada os veículos que tiveram pelo menos uma permanência de 20s no ponto de ônibus.

Figura 36 – Relatório do ponto MiniPreço: a) Veículo de ID 34354; b) Veículo de ID 34224

Ponto :	PT MINIPREÇO - Avenida Fernandes da Cunha, 81-275 - Mares, Salvador - BA, 40445-2					
Chegada		Saída				
Data/Hora	Velocidade	Data/Hora	Velocidade	Sentido	Tempo no Ponto	
4/8/2015 06:21:39	17,00 Km/h	4/8/2015 06:22:18	28,00 Km/h	Ida	00:00:39	
4/8/2015 08:19:51	25,00 Km/h	4/8/2015 08:23:01	19,00 Km/h	Volta	00:03:10	
4/8/2015 09:27:24	30,00 Km/h	4/8/2015 09:27:46	17,00 Km/h	Ida	00:00:22	
4/8/2015 11:35:49	10,00 Km/h	4/8/2015 11:36:30	14,00 Km/h	Volta	00:00:41	
4/8/2015 12:25:16	31,00 Km/h	4/8/2015 12:25:22	33,00 Km/h	Ida	00:00:06	
4/8/2015 14:17:04	16,00 Km/h	4/8/2015 14:17:50	18,00 Km/h	Volta	00:00:46	
4/8/2015 15:30:02	10,00 Km/h	4/8/2015 15:31:36	12,00 Km/h	Ida	00:01:34	
4/8/2015 17:53:03	13,00 Km/h	4/8/2015 17:54:48	18,00 Km/h	Volta	00:01:45	
4/8/2015 19:05:01	18,00 Km/h	4/8/2015 19:05:43	17,00 Km/h	Ida	00:00:42	
4/8/2015 19:39:56	27,00 Km/h	4/8/2015 19:40:22	25,00 Km/h	Volta	00:00:26	
4/8/2015 17:14:35	20,00 Km/h	4/8/2015 17:15:31	20,00 Km/h	Ida	00:00:56	
4/8/2015 16:11:56	15,00 Km/h	4/8/2015 16:12:34	11,00 Km/h	Volta	00:00:38	
4/8/2015 14:12:03	13,00 Km/h	4/8/2015 14:12:50	12,00 Km/h	Ida	00:00:47	
4/8/2015 13:18:29	8,00 Km/h	4/8/2015 13:19:05	0,00 Km/h	Volta	00:00:36	
4/8/2015 11:08:32	17,00 Km/h	4/8/2015 11:08:40	34,00 Km/h	Ida	00:00:08	
4/8/2015 10:29:08	40,00 Km/h	4/8/2015 10:29:13	10,00 Km/h	Volta	00:00:05	
4/8/2015 08:06:16	16,00 Km/h	4/8/2015 08:06:25	30,00 Km/h	Ida	00:00:09	
4/8/2015 06:50:18	0,00 Km/h	4/8/2015 06:50:31	8,00 Km/h	Volta	00:00:13	
4/8/2015 06:50:14	0,00 Km/h	4/8/2015 06:50:15	0,00 Km/h	Volta	00:00:01	
4/8/2015 06:48:51	28,00 Km/h	4/8/2015 06:49:37	0,00 Km/h	Volta	00:00:46	
4/8/2015 05:31:27	37,00 Km/h	4/8/2015 05:31:33	36,00 Km/h	Ida	00:00:06	

Considerando as entidades do modelo *IB-Trajectory* para o ponto “PT Minipreço”, foram encontradas 12 paradas programadas e 8 paradas não realizadas (Figura).

Os resultados apresentados na Figura , demonstram 10 paradas do veículo ID 34224 (7 *PlannedStops* e 3 *UnrealizedStops)*. Comparando com os resultados obtidos no relatório operacional da Figura são identificadas 7 paradas realizadas, coincidindo com a quantidade de *PlannedStops* identificadas pelo modelo. Ao analisar com maior atenção os dados do relatório, observa-se uma falha para os dados do veículo de ID 34224 no horário entre 06:48:51 até 06:50:31.

Figura 37 - *PlannedStops* e *UnrealizedStops* obtidos para o ponto Minipreço

objectid character varying	stop_gid character varying	start_time timestamp without time zone	end_time timestamp without time zone	stop_type character varying
34224	253			unrealized
34224	253	2015-08-04 06:49:23	2015-08-04 06:50:37	planned
34224	253	2015-08-04 08:04:25	2015-08-04 08:06:13	planned
34224	253	2015-08-04 10:29:13	2015-08-04 10:29:47	planned
34224	253			unrealized
34224	253	2015-08-04 13:19:05	2015-08-04 13:19:25	planned
34224	253	2015-08-04 14:12:11	2015-08-04 14:12:50	planned
34224	253	2015-08-04 16:12:07	2015-08-04 16:12:41	planned
34224	253	2015-08-04 17:14:53	2015-08-04 17:15:27	planned
34224	253			unrealized
34354	253			unrealized
34354	253	2015-08-04 08:19:57	2015-08-04 08:23:01	planned
34354	253			unrealized
34354	253			unrealized
34354	253			unrealized
34354	253	2015-08-04 14:17:09	2015-08-04 14:17:54	planned
34354	253	2015-08-04 15:29:49	2015-08-04 15:31:36	planned
34354	253	2015-08-04 17:53:26	2015-08-04 17:54:48	planned
34354	253			unrealized
34354	253	2015-08-04 20:48:54	2015-08-04 20:49:12	planned

Neste intervalo de tempos foram registradas 3 paradas (Figura .a), que provavelmente caracterizariam uma única parada neste horário. Analisando os dados do modelo *IB-Trajectory* as paradas foram corretamente agrupadas como um único *PlannedStop* (Figura .b).

Figura 38 – Analise do veículo 34224: a) relatório operacional; b) modelo *IB-Trajectory*

4/8/2015 06:50:18	0,00 Km/h	4/8/2015 06:50:31	8,00 Km/h	Volta	00:00:13
4/8/2015 06:50:14	0,00 Km/h	4/8/2015 06:50:15	0,00 Km/h	Volta	00:00:01
4/8/2015 06:48:51	28,00 Km/h	4/8/2015 06:49:37	0,00 Km/h	Volta	00:00:46

a)

objectid character varying	stop_gid character varying	start_time timestamp without time zone	end_time timestamp without time zone	stop_type character varying
34224	253	2015-08-04 06:49:23	2015-08-04 06:50:37	planned

b)

Outra fonte de divergência entre os resultados do relatório operacional e os resultados obtidos através do *IB-Trajectory* ocorreu no intervalo entre 08:04:25 e 08:06:13 (Figura .a) e no intervalo entre às 10:29:13 e às 10:29:47 (Figura .b). Os mapas com os registros da trajetória apresentam uns aglomerados de pontos na região nestas faixas de horários. Assim, o modelo identificou *IB-Trajectory* identificou estes aglomerados como *PlannedStops.* Estas paradas, entretanto, não foram registradas nos relatórios operacionais, talvez pelo fato do raio de identificação da parada (60 m) não contemplar alguns pontos dos dados brutos.

Figura 39 – Visão no mapa de recortes de horários do veículo 34224: a) entre 06:48 e 06:50; b) 08:04; c) 10:29

a)

b)

Confrontando os dados do relatório operacional com os dados coletados do *IB-Trajectory*, podemos observar que os resultados do modelo estão coerentes com dados coletados pelo relatório operacional, mas além disso o modelo apresenta um maior número de informações, a exemplo dos *UnplannedStop.*

4.2 CONSIDERAÇÕES FINAIS

Os estudos de casos apresentados com resultados de quatro diferentes centros urbanos demonstram a flexibilidade e a aderência do *IB-Trajectory* para este domínio de aplicação. Resultados distintos que sugeriram identificação de padrões de comportamentos, congestionamentos, regiões de paradas menos contempladas dentre outras análises, demonstram o potencial de contribuição que o modelo possui.

Os dados das análises se mostraram compatíveis com os dados gerados por uma aplicação de gerência do sistema de transporte público de Salvador. Os resultados obtidos através do modelo *IB-Trajectory*, entretanto, apresentaram resultados semanticamente mais ricos, o que é o principal diferencial do modelo.

5 CONCLUSÃO

Este trabalho apresentou um modelo conceitual de dados, denominado de *IB-Trajectory*, para representar trajetórias de objetos móveis que podem ou devem seguir um determinado itinerário. *IB-Trajectory* possui entidades mais abstratas e semanticamente enriquecidas para capturar a essência do movimento deste tipo de objetos móveis. O referido modelo tem como objetivo criar uma infraestrutura de dados para suportar uma ampla gama de análises sobre as trajetórias dos veículos que devem seguir determinado itinerário.

O modelo *IB-Trajectory* fornece entidades de primeira classe para representar os lugares onde o veículo fez paradas programadas, fez paradas inesperadas e lugares onde o veículo deveria ter parado, mas não o fez. Da mesma forma, especializações dos movimentos do veículo são utilizadas na representação de segmentos percorridos ou ignorados pelo veículo; e segmentos da rede rodoviária fora do itinerário, mas percorrida pelo veículo.

Os diversos cenários de estudo de caso aplicados em diferentes capitais: Recife, Rio de Janeiro, São Paulo e Salvador, foram utilizados para demonstrar o potencial do modelo proposto para análises em um domínio de aplicação que está em plena ascensão no Brasil.

As definições do modelo com dados sintéticos produzidos foram aprovados no FOS4G 2014 (ANDRADE; CAMPOS; AMORIM, 2014), o primeiro estudo de caso com dados reais foram publicados e apresentados no Simpósio Brasileiro de Sensoriamento Remoto – SBSR 2015 (ANDRADE; CAMPOS, 2015a) , o modelo *IB-Trajectory* também foi apresentado no ERBASE 2015 (ANDRADE; CAMPOS, 2015b).

Foram identificados alguns pontos relevantes nesta pesquisa que apontam para trabalhos futuros, a saber:

1) Melhorar o algoritmo de identificação dos *Moves* utilizando técnicas de similaridade de trajetórias.
2) Avaliar as técnicas do algoritmo SMot+ (ARBOLEDA et al., 2014) que podem ser utilizadas no *IB-Trajectory* para identificação dos *Stops*.
3) Avaliar uma estratégia de limpeza dos dados brutos e identificação de viagem por trecho do itinerário para melhorar a investigação e a *anotação* semântica das trajetórias.

4) Utilizar dados de outras trajetórias realizadas no mesmo trecho como base de conhecimento para adicionar novas semânticas.
5) Utilizar métricas de avaliação indicadas no trabalho (BARBOSA et al., 2014) como comparativo de eficácia do modelo.
6) Utilizar a abordagem do CONSTAnT (BOGORNY et al., 2013), baseada em *behaviors* para o enriquecimento das *Moves* e *Stops* no domínio dos transportes públicos para verificar a flexibilidade ou não da proposta.
7) Melhorar a interface do Weka-STPM na incorporação da análise do *IB-Trajectory*.
8) Introduzir uma etapa de processamento dos dados brutos para separar o que é rota útil para análise das rotas inúteis, dividir a trajetória bruta em *payload routes*, quando o veículo está em rota de coleta de passageiros e *off-duty routes*, quando o veículo está em deslocamento para a garagem.
9) Explorar o aspecto temporal na anotação semântica.
10) Realizar estudo de caso com outros veículos que devem seguir um itinerário, como transportadoras, correios, trens, etc.

O modelo *IB-Trajectory*, constitui uma alternativa inovadora para representação, estruturação e análise de dados relacionados com a dinâmica do objeto móvel no modal de ônibus urbanos, viabilizando e simplificando o processo de modelagem das aplicações voltadas para a análise de histórico do deslocamento dos ônibus em centros urbanos. Esta abordagem abre perspectivas para ampliação dos conhecimentos intrínsecos às trajetórias produzidas por este modal de transporte, incluindo aspectos sociais relacionados aos usuários deste meio de transportes, políticos e econômicos, visto que esta ferramenta de análise do histórico auxilia a visualização de anormalidades no deslocamento dos veículos que atendem a este modal, dando suporte à gestão dos sistemas de transporte, especialmente, nas grandes cidades.

REFERÊNCIAS

ALVARES, L. et al. A model for enriching trajectories with semantic geographical information. In: ANNUAL ACM INTERNATIONAL SYMPOSIUM ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS (GIS ’07) 22:1-22:8 (2007), 15., 2007. **Proceedings…**

ALVARES, L. O. et al. An algorithm to identify avoidance behavior in moving object trajectories. **Journal of the Brazilian Computer Society**, v. 17, n. 3, p. 193–203, 2011.

AMORIM, A. **TX-Trajectory:** um modelo para representação de trajetórias de táxi. [S.l: s.n.], 2013.

ANDRADE, D.; CAMPOS, J. Análise de Trajetória de Veículos baseado em Itinerários. In: SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 17., 2015, João Pessoa-PB, Brasil. **Anais...** João Pessoa-PB: INPE, 2015a. p. 6437–6444.

ANDRADE, D.; CAMPOS, J. IB-Trajectory: Um Modelo Semântico para Representação de Trajetórias Baseadas em Itinerários. In: ERBASE, 16., 2015. **Anais...** 2015b.

ANDRADE, D.; CAMPOS, J.; AMORIM, A. Semantic Enrichment of GPS Trajectories. **Free and Open Source Software for GIS (FOSS4G)**, p. 1–12, 2014.

ARBOLEDA, F. et al. SMoT+: Extending the SMoT Algorithm for Discovering Stops in Nested Sites. **Computing and Informatics**, v. 22, p. 327–342, 2014.

BARBOSA, L. et al. Vistradas : Visual Analytics for Urban Trajectory Data. **GeoInfo 2014**, p. 11–17, 2014.

BOGORNY, V. et al. **Weka-GDPM** – Integrating Classical Data Mining Toolkit to Geographic Information Systems. [S.l: s.n.], 2000.

BOGORNY, V. et al. CONSTAnT – A Conceptual Data Model for Semantic Trajectories of Moving Objects. **Transactions in GIS**, v. 18, n. 1, p. 66–88, 2013.

BOGORNY, V.; AVANCINI, H.; PAULA, B. DE. Weka-STPM: a Software Architecture and Prototype for Semantic Trajectory Data Mining and Visualization. **Transactions in GIS 15**, p. 227–248, 2011.

BOGORNY, V.; BRAZ, F. J. **Introdução a trajetórias de objetos móveis**. 1. ed. Joinville SC: Editora Univille, 2012.

CASTRO, R. D. F. C. R. **Tesselação de Voronoi em Empilhamentos Granulares**. [s.l.] Centro Federal de Educação Tecnológica de Minas Gerais, 2009.

DATARIO. **DataRio**, 2015. Disponível em: <http://data.rio/dataset/gps-de-onibus>

Acesso em: 1 fev. 2015.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING (KDD-96), 2., 1996. **Proceedings**… 1996.

p. 226–231.

FRANK, E. et al. WEKA: A Machine Learning Workbench for Data Mining. Springer US. In: **Data Mining and Knowledge Discovery Handbook**, p. 1305–14, 2005.

LAUBE, P.; IMFELD, S.; WEIBEL, R. Discovering relative motion patterns in groups of moving point objectsInternational. **Journal of Geographical Information Science**, 2005.

LONGLEY, P. et al. Geographic Information systems and science. **Wiley**, v. 2, p. 536, 2006.

MORENO, B. et al. Looking inside the stops of trajectories of moving objects. In: BRAZILIAN SYMPOSIUM ON GEOINFORMATICS. 2010. **Anais**... 2010. Disponível em: <http://www.inf.ufsc.br/~vania/artigos/LISTMO.pdf>. Acesso em: 3 nov. 2011.

MORENO, F.; ARANGO, F. A conceptual trajectory multidimensional model: An application to public transportation . Un modelo conceptual multidimensional para trayectorias: Una aplicación en el transporte público. **Dyna,** v. 78, n. 166, p. 142–149, 2011.

PALMA, A. et al. **A Clustering-based Approach for Discovering Interesting Places in Trajectories Proceedings of the 2008**, 2008. Disponível em: <http://dl.acm.org/citation.cfm?id=1363886>. Acesso em: 15 abr. 2012

PALMA, Andrey Tietbohl et al. A clustering-based approach for discovering interesting places in trajectories. In: ACM SYMPOSIUM ON APPLIED COMPUTING. ACM, 2008. **Proceedings...** 2008. p. 863-868.

PALMA, A. L. T. **A clustering-based approach for discovering interesting places in a single trajectory**. [S.l.] Universidade Federal do Rio Grande do Sul, 2008.

ROCHA, J. et al. DB-SMoT: A direction-based spatio-temporal clustering method.In: **I**NTELLIGENT SYSTEMS (IS), 2010 IEEE INTERNATIONAL CONFERENCE, 5., 2010. **Proceedings**… 2010.

SILVA, B. C. et al. ITSUMO: an intelligent transportation system for urban mobility. In: INNOVATIVE Internet Community Systems. Springer Berlin Heidelberg, 2004. p. 224-235.

SILVEIRA, T. D. A.; BARROS FILHO, M. B. D. B. O geoprocessamento aplicado à gestão dos transportes públicos. In: SIMPÓSIO DE GEOTECNOLOGIAS NO PANTANAL, 2006. **Anais...** 2006. p. 718–725.

SIQUEIRA, F. D. L.; BOGORNY, V. Discovering Chasing Behavior in Moving Object Trajectories. **Transactions in GIS**, v. 15, n. 5, p. 667–688, 2011.

SPACCAPIETRA, S. et al. A conceptual view on trajectories. **Data and Knowledge Engineering**, v. 65, n. 1, p. 126–146, abr. 2008.

SPTRANS. **Olho Vivo**, 2015. Disponível em: <http://www.sptrans.com.br/desenvolvedores/APIOlhoVivo.aspx> Acesso em: 3 nov. 2011.

UBIBUS. **UBIBUS**. Disponível em: <http://www.cin.ufpe.br/~ubibus/>. Acesso em: 3 nov. 2011.

VANIA BOGORNY, BART KUIJPERS, L. O. A. A Spatio-temporal Data Mining Query Language for Moving Object Trajectories. **Arthritis care & research**, n. i, p. 1–22, 2011.

VELOSO, M.; PHITHAKKITNUKOON, S.; BENTO, C. Urban mobility study using taxi traces. In: INTERNATIONAL WORKSHOP ON TRAJECTORY DATA MINING AND ANALYSIS TDMA 11., 2011. **Proceedings…**. 2011. p.23–30.

VIEIRA, V. et al. The UbiBus Project: Using Context and Ubiquitous Computing to build Advanced Public Transportation Systems to Support Bus Passengers. In: SIMPÓSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO, 8., 2012. **Anais...** 2012.

VIEIRA, V.; CALDAS, L. R.; SALGADO, A. C. Towards an Ubiquitous and Context sensitive public transportation system. In: INTERNATIONAL CONFERENCE ON UBI-MEDIA COMPUTING, U-MEDIA, 4., 2011. **Proceedings…** 2011. Disponível em: <http://www.computer.org/portal/web/csdl/doi/10.1109/U-MEDIA.2011.19>. Acesso em: 5 nov. 2011

YAN, Z. et al. SeMiTri: a framework for semantic annotation of heterogeneous trajectories. INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, 14., 2011. **Proceedings…** 2011. p. 259–270.

YAN, Z. et al. Semantic trajectories: mobility data computation and annotation. **ACM Transactions on Intelligent Systems and Technology**, v. 4, n. 3, p. 49:1–49:38, 2013.

ZHENG, K. et al. On discovery of gathering patterns from trajectories. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING (ICDE), 2013. **Proceedings…** 2013. p. 242–253.

ANEXO A – APLICAÇÃO NO WEKA-STPM

A seguir o código fonte das principais classes criadas e adaptadas no WEKA-STPM é listado.

MoveAbstract.java

```
package weka.stpm.ibmodel;

import java.util.ArrayList;
import weka.gui.stpm.GPSPoint;
public class MoveAbstract {
      protected int idTrecho;
      protected String objectid;
      protected      ArrayList<GPSPoint>      points      =      new
ArrayList<GPSPoint>();
      protected String moveType;
      protected String nameFeature;

   public String toSQL() {
      // respecting the minimum of 2 points
         if (points.size() < 2) return "null";
         String ret = "'MULTILINESTRING(";
         for (int i=1;i<points.size();i++) {
             ret  +=  "  (  "+points.get(i-1).point.getX()  +  "  "  +
points.get(i-1).point.getY() + ","
                    +points.get(i).point.getX()    +    "    "    +
points.get(i).point.getY()+" )," ;
         }
         ret = ret.substring(0,ret.length()-1) + ")' ";

         return ret;
    }
      @Override
      public String toString() {
            return "Trecho: "+idTrecho+" Pontos: "+points.toString();
      }
```

```
/**
 * TODO gerar a geometria Multilinestring do move
 * @return
 */
public String theGeom() {
      return "";
}
public int getIdTrecho() {
      return idTrecho;
}
public void setIdTrecho(int idTrecho) {
      this.idTrecho = idTrecho;
}
public String getMoveType() {
      return moveType;
}
public void setMoveType(String moveType) {
      this.moveType = moveType;
}
public String getNameFeature() {
      return nameFeature;
}

public void setNameFeature(String nameFeature) {
      this.nameFeature = nameFeature;
}

public ArrayList<GPSPoint> getPoints() {
      return points;
}
public void setPoints(ArrayList<GPSPoint> points) {
      this.points = points;
}

public String getObjectid() {
      return objectid;
}
```

```
	public void setObjectid(String objectid) {
		this.objectid = objectid;
	}
}
```

MovePlanned.java

```
package weka.stpm.ibmodel;
import java.util.ArrayList;
import weka.gui.stpm.GPSPoint;
public class MovePlanned extends MoveAbstract {

	public MovePlanned(String objetcid,int
idTrecho,ArrayList<GPSPoint> points, String nameFeature) {
		this.objectid = objetcid;
		this.idTrecho = idTrecho;
		this.points.addAll(points);
		this.nameFeature = nameFeature;
		this.moveType = "planned";
	}

    @Override
    public String toString() {
	 return super.toString() + " SQL: "+this.toSQL();
    }

}
```

Move UnRealized

```
package weka.stpm.ibmodel;
import java.util.ArrayList;
import weka.gui.stpm.GPSPoint;
public class MoveUnRealized extends MoveAbstract {
```

```
     public        MoveUnRealized(String        objectid,        int
idTrecho,ArrayList<GPSPoint> points, String nameFeature) {
           this.objectid = objectid;
           this.idTrecho = idTrecho;
           this.points.addAll(points);
           this.nameFeature = nameFeature;
           this.moveType = "unrealized";
     }

    @Override
    public String toString() {
     return super.toString() + " SQL: "+this.toSQL();
    }

}
```

Move UnPlanned

```
package weka.stpm.ibmodel;
import java.util.ArrayList;
import weka.gui.stpm.GPSPoint;

public class MoveUnPlanned extends MoveAbstract {

     public            MoveUnPlanned(String            objectid,int
idTrecho,ArrayList<GPSPoint> points, String nameFeature) {
           this.objectid = objectid;
           this.idTrecho = idTrecho;
           this.points.addAll(points);
           this.nameFeature = nameFeature;
           this.moveType = "unplanned";
     }

}
```

ItineraryMethods.java

```
package weka.stpm.ibmodel;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Vector;

import org.postgis.Point;

import weka.gui.stpm.AssociatedParameter;
import weka.gui.stpm.Config;
import weka.gui.stpm.GPSPoint;
import weka.gui.stpm.Interc;
import weka.gui.stpm.InterceptsG;
import weka.gui.stpm.Stop;
import weka.gui.stpm.Trajectory;
import weka.gui.stpm.TrajectoryFrame;
import weka.gui.stpm.TrajectoryMethods;
import weka.gui.stpm.Unknown;

public class ItineraryMethods extends TrajectoryMethods {

      /**
       * Save STOPS (Planned, Unplanned, Unrealized) no contexto do
IB-Model
       * @author duilio
       * @param list
       * @param conn
       * @param featureType
```

```
       * @param buffer
       * @param stopextern
       * @return
       */
      public static int saveStopsIBModel(Vector list, Connection
conn, boolean featureType, double buffer,int stopextern){
                  Statement s,s1, s2;
                  boolean flag=false;
                  int stopId=stopextern;
                  String sql="";
                  int tIDGlobal = 0 ;
                  String tObjectIDGlobal = "0" ;

                  HashMap<String, ArrayList<Integer>> mapFeatures =
new HashMap<String, ArrayList<Integer>>();
                  ArrayList<Integer>    idCandidateStops    =    new
ArrayList<Integer>();

                  int i = 0;
                  for (;i<list.size();i++) {
                  try{
                        s = conn.createStatement();
                        s1 = conn.createStatement();
                      Object obj = list.elementAt(i);

                      // PlannedStop
                      if (obj.getClass() == Stop.class) {

                          Stop stop = (Stop) obj;
                          tIDGlobal = stop.tid;

                          System.out.println("stop.objectid         "+
stop.objectid);

                          if           (stop.objectid!=           null
&& !stop.objectid.isEmpty() ) {
                             tObjectIDGlobal = stop.objectid;
```

```
                        }else {
                           tObjectIDGlobal = "0";
                        }


                        // guarda os idStops para verificar os
candidates stops que não foram encontrados
                        idCandidateStops.add(stop.gid);

                        // se nao existir idsStops para a tabela,
cria uma nova chave
                        if
(!mapFeatures.containsKey(stop.tableName)) {

                           ArrayList<Integer>    idStops    =    new
ArrayList<Integer>();
                           idStops.add(stop.gid);
                           mapFeatures.put(stop.tableName, idStops);

                        }else { // caso já exista uma chave para a
tabela, apenas adiciona o id

     mapFeatures.get(stop.tableName).add(stop.gid);
                        }

                        String    stopName    =    featureType    ?
stop.tableName : (stop.gid + "_" + stop.tableName);
                        sql          =          "INSERT          INTO
"+TrajectoryFrame.getCurrentNameTableStop()+"
(tid,objectid,stopid,start_time,end_time,stop_gid,stop_name,the_geom
,rf,stop_type,avg) VALUES "+

"("+stop.tid+",'"+stop.objectid+"',"+stopId+",'"+stop.enterTime.toSt
ring()+"','"+stop.leaveTime.toString()+"','"+stop.gid+"','"+stopName
+"',"+stop.toSQL(buffer)+",'"+stop.tableName+"','planned',"+stop.avg
Speed()+")";
                        stopId++;
```

```
                    flag=false;

                // UnplanedStop
                }else if (obj.getClass() == Unknown.class) {

                    Unknown unk = (Unknown) obj;

                    int tid = unk.pontos.firstElement().tid;
                    String objetcid = "0";

                    System.out.println("unknow          objectId
"+unk.pontos.firstElement().objectId);

                    if     (unk.pontos.firstElement().objectId!=
null && !unk.pontos.firstElement().objectId.isEmpty() ) {
                       tObjectIDGlobal
=unk.pontos.firstElement().objectId;
                    }

                       String  query  =  "select  stop_name  from
"+TrajectoryFrame.getCurrentNameTableStop()+"  where  rf='unplanned'
AND intersects(the_geom,"+unk.toSQL(buffer)+");";
                       ResultSet rs = s1.executeQuery(query);
                       if(rs.next()){
                            //joining same unknowns...
                            //System.out.println("aqui");
                            /**@TODO  verificar  esse  trecho,
provavelmente deveria realizar um update ao inves do insert */
                            sql      =      "INSERT      INTO
"+TrajectoryFrame.getCurrentNameTableStop()+"
(tid,objectid,stopid,start_time,end_time,stop_gid,stop_name,the_geom
,rf,stop_type,avg) VALUES "+

     "("+tid+",'"+objetcid+"',"+stopId+",'"+unk.enterTime.toString()
+"','"+unk.leaveTime.toString()+"',"+-
1+",'"+rs.getString("stop_name")+"',"+unk.toSQL(buffer)+",'unknown',
'unplanned',"+unk.avgSpeed()+")";
```

```
                        }
                        else {//or creating another
                              sql       =       "INSERT       INTO
"+TrajectoryFrame.getCurrentNameTableStop()+"
(tid,objectid,stopid,start_time,end_time,stop_gid,stop_name,the_geom
,rf,stop_type,avg) VALUES "+

     "("+tid+",'"+objetcid+"',"+stopId+",'"+unk.enterTime.toString()
+"','"+unk.leaveTime.toString()+"',"+-
1+",'"+nextUnknown()+"_unplanned',"+unk.toSQL(buffer)+",'unknown','u
nplanned',"+unk.avgSpeed()+")";
                        }
                        stopId++;
                        flag=false;//poe pra executar a query...
                     //else flag=true;//ou NAO executa...

                  }


                  if(!flag){//teste pra saber se executa ou nao a
query
                   System.out.print("Stop "+i+" saving... ");
                   System.out.println("##### "+sql);
                   s.execute(sql);
                   System.out.println("Saved");
                  }
                   }catch(Exception e){
                        System.out.println(e.getMessage());
                        e.printStackTrace();
                        break;
                   }
               }

               //  ################    INICIO  SALVAMENTO  DE  STOPS
UNREALIZADED ######################
```

```
                try {

                     s = conn.createStatement();
                     s2 = conn.createStatement();

                     if (!mapFeatures.isEmpty()) {

                          for      (String      interceptName      :
mapFeatures.keySet()) {

                                    String  nameFeature  =  interceptName;

                                    String             ids             =
mapFeatures.get(nameFeature).toString().replace("[",
"").replace("]", "");

                                    // save os UnrealizedStop
                                    String query = "SELECT gid, the_geom
FROM "+ nameFeature +"_buf where gid not in ("+ids+")";
                                    ResultSet            rs            =
s2.executeQuery(query);
                                    while(rs.next()){
                                         // faz a inclusao
                                         sql     =     "INSERT     INTO
"+TrajectoryFrame.getCurrentNameTableStop()+"
(tid,objectid,stopid,stop_gid,stop_name,rf,stop_type,avg,the_geom)
VALUES "+

     "("+tIDGlobal+",'"+tObjectIDGlobal+"',"+stopId+","+rs.getString
("gid")+",'"+nextUnrealized()+"_unrealized','"+interceptName+"','unr
ealized','0',geomfromtext('"+rs.getString("the_geom")+"'))";

                                         System.out.print("Stop
"+(i++)+" saving... ");
                                    s.execute(sql);
                                    System.out.println("Saved");
                                    stopId++;
```

```
                              }


                        }
                  } else {


                        String          interceptName          =
TrajectoryFrame.getInterceptName();

                        // save os UnrealizedStop
                        String query = "SELECT gid, the_geom FROM
"+ interceptName +"_buf ";
                        ResultSet rs = s2.executeQuery(query);
                        while(rs.next()){
                              // faz a inclusao
                              sql       =       "INSERT       INTO
"+TrajectoryFrame.getCurrentNameTableStop()+"
(tid,objectid,stopid,stop_gid,stop_name,rf,stop_type,avg,the_geom)
VALUES "+

     "("+tIDGlobal+",'"+tObjectIDGlobal+"',"+stopId+","+rs.getString
("gid")+",'"+nextUnrealized()+"_unrealized','"+interceptName+"','unr
ealized','0',geomfromtext('"+rs.getString("the_geom")+"'))";

                              System.out.print("Stop      "+(i++)+"
saving... ");
                              System.out.println("##########
"+sql);
                        s.execute(sql);
                        System.out.println("Saved");
                        stopId++;
                        }

                  }

                  // calcula o fecho convexo.
```

```
                    calculateFactor(conn);

                } catch (SQLException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

                //  ################     FIM  SALVAMENTO  DE  STOPS
UNREALIZADED ######################

          return --stopId;
     }

    /**
     * Save Moves (Planned, Unplanned, Unrealized) no contexto do
IB-Model
     * @param list
     * @param conn
     * @param buffer
     * @return
     */
     public  static  int  saveMovesIBModel(ArrayList<MoveAbstract>
list, Connection conn,double buffer){
          int movesQtd = 0;
          String sql = "";
          Statement s, s2,s3;
          int i = 0;

          HashMap<String,  ArrayList<GPSPoint>>  mapFeatures  =  new
HashMap<String, ArrayList<GPSPoint>>();
          String nameFeature = "";
          String objectid = "";

          for (MoveAbstract moveAbstract : list) {
              i++;
               try {
```

```
                    objectid = moveAbstract.getObjectid();
                    s = conn.createStatement();

                    sql          =          "INSERT          INTO
"+TrajectoryFrame.getCurrentNameTableMove()+"
(objectid,move_type,stretch_id,rf,the_geom) VALUES "+

"('"+moveAbstract.getObjectid()+"','"+moveAbstract.getMoveType()+"',
'"+moveAbstract.getIdTrecho()+"',
'"+moveAbstract.getNameFeature()+"' ,"+moveAbstract.toSQL()+")";


                System.out.print("Move "+i+" saving... ");
                s.execute(sql);
                System.out.println("Saved");
                movesQtd++;

                if (moveAbstract.getMoveType().equals("planned")) {

                     nameFeature = moveAbstract.getNameFeature();

                       // se nao existir points para a tabela, cria
uma nova chave
                       if
(!mapFeatures.containsKey(moveAbstract.getIdTrecho()+"")) {

                          ArrayList<GPSPoint>    points    =    new
ArrayList<GPSPoint>();
                          points.addAll(moveAbstract.getPoints());

     mapFeatures.put(moveAbstract.getIdTrecho()+"", points);

                       }else { // caso já exista uma chave para a
tabela, apenas adiciona os pontos

     mapFeatures.get(moveAbstract.getIdTrecho()+"").addAll(moveAbstr
act.getPoints());
```

```
                    }

                }


                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

            //salvar os unrealized moves de trechos que n possuem
rotas

        boolean geraUnrealized = false;

        if (geraUnrealized) {

                try {

                    for (String idTrecho : mapFeatures.keySet()) {

                        s2 = conn.createStatement();
                        s3 = conn.createStatement();

                        //String ids =
mapFeatures.get(idTrecho).toString().replace("[",    "").replace("]",
"");


                        String query = "SELECT gid, the_geom FROM
"+ nameFeature +" WHERE gid = "+ idTrecho;
                        ResultSet rs = s2.executeQuery(query);

                        while(rs.next()){
                            // faz a inclusao

                            // aplicar a regra do difference
```

```
                                    sql       =       "INSERT       INTO
"+TrajectoryFrame.getCurrentNameTableMove()+"
(objectid,move_type,stretch_id,rf,the_geom) VALUES "+

     "('"+objectid+"´,'unrealized','"+rs.getInt("gid")+"','"+nameFea
ture+"',geomfromtext('"+difference(rs.getString("the_geom"),mapFeatu
res.get(idTrecho)) +"'))";

                                    System.out.print("Move      "+(++i)+"
saving... ");
                                    s3.execute(sql);
                                    System.out.println("Saved");
                                    movesQtd++;
                               }

                          }

                } catch (SQLException e) {
                     // TODO Auto-generated catch block
                     e.printStackTrace();
                }
        }
/*         try {

                for (String interceptName : mapFeatures.keySet()) {

                     s2 = conn.createStatement();
                     s3 = conn.createStatement();

                     String                    ids                    =
mapFeatures.get(interceptName).toString().replace("[",
"").replace("]", "");

                     String query = "SELECT gid, the_geom FROM "+
interceptName +" where gid not in ("+ids+")";
                     ResultSet rs = s2.executeQuery(query);
```

```
                    while(rs.next()){
                        // faz a inclusao
                        sql = "INSERT INTO
"+TrajectoryFrame.getCurrentNameTableMove()+"
(move_type,stretch_id,rf,the_geom) VALUES "+

     "('unrealized','"+rs.getInt("gid")+"','"+interceptName+"',geomf
romtext('"+rs.getString("the_geom")+"'))";

                        System.out.print("Move "+(++i)+"
saving... ");
                        s3.execute(sql);
                        System.out.println("Saved");
                        movesQtd++;
                    }

                }

            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }*/

            // fazer o diff dos trechos percorridos pela metade

            return movesQtd;
     }

     private static String difference(String theGeom,
ArrayList<GPSPoint> points) {


            // respecting the minimum of 2 points
               if (points.size() < 2) return "null";

               String ret = "'MULTILINESTRING(";
               for (int i=1;i<points.size();i++) {
```

```
                ret += " ( "+points.get(i-1).point.getX() + " " +
points.get(i-1).point.getY() + ","
                        +points.get(i).point.getX()   +   "   "   +
points.get(i).point.getY()+" )," ;
            }
            ret = ret.substring(0,ret.length()-1) + ")' ";
            ret = "difference(the_geom, geomfromtext(" +ret+") )";
            return ret;

     }

        /**
     * TODO implementar logica para detectação dos moves
     * ao percorrer trajetoria
     * @param t
     */
    public   static   void   moveDetect(Trajectory   t,   InterceptsG
interc,Config config) {


     System.out.println("@@@@@@@@@@ moveDetect @@@@@@@@@@@@@@");

     ArrayList movePlannedAux = new ArrayList();
     ArrayList moveUnPlannedAux = new ArrayList();

     ArrayList movePlanned = new ArrayList();
     ArrayList moveUnPlanned = new ArrayList();
     ArrayList moveUnRealized = new ArrayList();

     ArrayList<MoveAbstract> list = new ArrayList();

     //int idTrechoAnterior = 0;
     Interc trechoAnterior = null;

     Integer gidPlannedAnt = 0;
     Integer gidUnPlannedAnt = 0;
```

```
      System.out.println("Quantidade de Pontos "+t.points.size());

      int cont = 0;
      String objectid = "";

      for (GPSPoint p: t.points) {

            objectid = p.objectId;

            if ( cont++% 100 == 0 )
                  System.out.println("Move Detect qtd "+ cont + " "+
new Date());

            // compara se existe intersecao no trecho

            Interc trecho = interc.isIntersecTrecho(p.gid);
            if ( trecho != null ) {

                  // se é sequencial e foi identificado intersecao no
mesmo trecho gidPlannedAnt == (p.gid -1)
                  if ( (trechoAnterior == null || trechoAnterior.gid
== trecho.gid ))  {

                        movePlannedAux.add(p);


                  }else { // se nao for sequencial cria o trecho de
moveplanned

                        // cria o MovePlanned
                              if (!movePlannedAux.isEmpty()) {

                                    // so cria se tive mais que 1 ponto
                                    if (movePlannedAux.size() > 1) {
                                          movePlanned.add(new
MovePlanned(p.objectId,trechoAnterior.gid,
movePlannedAux,trechoAnterior.rf));
```

```
                        }
                        movePlannedAux.clear();

                        }

                        movePlannedAux.add(p);
                }

                gidPlannedAnt = p.gid;
                trechoAnterior = trecho; // guarda o ID do trecho
anterior para gerar o PlannedMove


          }else {

                // TODO verificar uma forma de identificar
proximidade entre dois pontos

                if (gidUnPlannedAnt == (p.gid-1) ) { // se é
sequencial mantem no mesmo conjunto
                      // unplanned move
                      moveUnPlannedAux.add(p);

                }else { // se n for sequencial cria o trecho de
moveUnplanned

                      // cria o MoveUnPlanned
                      if (!moveUnPlannedAux.isEmpty()) {

                            // so cria se tiver mais que 1 ponto
                            if (moveUnPlannedAux.size() > 1) {
                                  moveUnPlanned.add(new
MoveUnPlanned(p.objectId, 0, moveUnPlannedAux,"unplanned"));
                            }
                            moveUnPlannedAux.clear();
                      }
                      moveUnPlannedAux.add(p);
```

```
                }

                gidUnPlannedAnt = p.gid;

            }


        }


        // casos das extremidades
            // verifica se ainda tem MovePlanned
            if (!movePlannedAux.isEmpty()) {

                // so cria se tive mais que 1 ponto
                if (movePlannedAux.size() > 1) {
                    movePlanned.add(new
MovePlanned(objectid,trechoAnterior.gid,
movePlannedAux,trechoAnterior.rf));
                }
                movePlannedAux.clear();
            }

        // verifica se ainda tem unplannedmoves
            // cria o MoveUnPlanned
            if (!moveUnPlannedAux.isEmpty()) {

                // so cria se tiver mais que 1 ponto
                if (moveUnPlannedAux.size() > 1) {
                    moveUnPlanned.add(new MoveUnPlanned(objectid,0,
moveUnPlannedAux,"unplanned"));
                }
                moveUnPlannedAux.clear();
            }

        System.out.println("########## movePlanned "+movePlanned );
        System.out.println("########## moveUnPlanned "+moveUnPlanned );
```

```
      // salvar os moves
      list.addAll(movePlanned);
      list.addAll(moveUnPlanned);

      System.out.println(" antes de salvar ");
      saveMovesIBModel(list, config.conn, 0);
      System.out.println(" depois de salvar ");
     }

  /** Move Detect 2
  * Detectar a rota baseado apenas em points
  * ao percorrer trajetoria
  * @param t
  */
 public static void moveDetect2(Trajectory t, InterceptsG
interc,Config config) {


      System.out.println("@@@@@@@@@@ moveDetect 2 @@@@@@@@@@@@@@@");

      ArrayList movePlannedAux = new ArrayList();
      ArrayList moveUnPlannedAux = new ArrayList();

      ArrayList movePlanned = new ArrayList();
      ArrayList moveUnPlanned = new ArrayList();
      //ArrayList moveUnRealized = new ArrayList();

      ArrayList<MoveAbstract> list = new ArrayList();

      //int idTrechoAnterior = 0;
      Interc trechoAnterior = null;

      Integer gidPlannedAnt = 0;
      Integer gidUnPlannedAnt = 0;

      int tidAnterior = 0;
```

```
	int cont = 0;

	String objectid = "";

	for (GPSPoint p: t.points) {

		objectid = p.objectId;

		//TODO testar se é convergente com o inicio ou fim do itinerario

		// identifica que é um nov TID, paga limpar os planned e os unplanned
		if (tidAnterior != 0 && tidAnterior != p.tid ) {

			System.out.println("@@@@@@@@ tidAnterior != 0 && tidAnterior != p.tid ");
			movePlannedAux.clear();
			moveUnPlannedAux.clear();
		}

		if ( cont++% 100 == 0 )
		System.out.println("Move Detect qtd "+ cont + " "+ new Date());

		// compara se existe intersecao no trecho

		Interc trecho = interc.isIntersecTrecho(p.gid);
		//Interc trecho = interc.is_in(p);

		if ( trecho != null ) { // em tese é um ponto para o movePlanned

			// se tiver moveUnPlanned ele guarda os pontos para o UnPlanned
```

```
                // Inicio cria o MoveUnPlanned
                if (!moveUnPlannedAux.isEmpty()) {

                      // so cria se tiver mais que 1 ponto
                      if (moveUnPlannedAux.size() > 1) {
                            moveUnPlanned.add(new
MoveUnPlanned(p.objectId, 0, moveUnPlannedAux,"unplanned"));
                      }
                      moveUnPlannedAux.clear();
                }
                // Fim cria o MoveUnPlanned

                movePlannedAux.add(p);

                gidPlannedAnt = p.gid;
                trechoAnterior = trecho; // guarda o ID do trecho
anterior para gerar o PlannedMove


           }else { // em tese é um ponto para o moveUnPlanned

                // Inicio cria o MovePlanned
                if (!movePlannedAux.isEmpty()) {

                      // so cria se tive mais que 1 ponto
                      if (movePlannedAux.size() > 1) {
                            movePlanned.add(new
MovePlanned(p.objectId,trechoAnterior.gid,
movePlannedAux,trechoAnterior.rf));
                      }
                      movePlannedAux.clear();

                }
                // Fim cria o MovePlanned

                moveUnPlannedAux.add(p);
```

```
                gidUnPlannedAnt = p.gid;

            }

            tidAnterior = p.tid;


      }


      // casos das extremidades
            // verifica se ainda tem MovePlanned
            if (!movePlannedAux.isEmpty()) {

                  // so cria se tive mais que 1 ponto
                  if (movePlannedAux.size() > 1) {
                        movePlanned.add(new       MovePlanned(objectid,
trechoAnterior.gid, movePlannedAux,trechoAnterior.rf));
                  }
                  movePlannedAux.clear();
            }

      // verifica se ainda tem unplannedmoves
            // cria o MoveUnPlanned
            if (!moveUnPlannedAux.isEmpty()) {

                  // so cria se tiver mais que 1 ponto
                  if (moveUnPlannedAux.size() > 1) {
                        moveUnPlanned.add(new MoveUnPlanned(objectid,0,
moveUnPlannedAux,"unplanned"));
                  }
                  moveUnPlannedAux.clear();
            }

      System.out.println("########## movePlanned "+movePlanned );
      System.out.println("########## moveUnPlanned "+moveUnPlanned );
```

```
      list.addAll(movePlanned);
      list.addAll(moveUnPlanned);


 // salvar os moves
      System.out.println(" antes de salvar ");

      saveMovesIBModel(list, config.conn, 0);

      // TODO alterar para pegar dinamicamente o nome da tabela de
itinerario

      unrealizedDetect(config.conn,TrajectoryFrame.getInterceptName()
);

      System.out.println(" depois de salvar ");
 }

 private static void calculateFactor(Connection conn) {

      Statement s, s1;
      String sql = "";

      try {
                  s = conn.createStatement();

                  s1 = conn.createStatement();

                  String       query       =       "select       gid,
area(convexhull(the_geom)) as area, perimeter(convexhull(the_geom))
as perimeter from "+TrajectoryFrame.getCurrentNameTableStop() +"
where the_geom is not null"; //+" where rf='unplanned' ";
                  System.out.println("calculateFactor query "+query);
                  ResultSet rs = s.executeQuery(query);
```

```
                while(rs.next()){
                      // faz a inclusao
                      sql                  =                  "UPDATE
"+TrajectoryFrame.getCurrentNameTableStop()+"    SET    S    =    "+
formFactor(rs.getDouble("area"), rs.getDouble("perimeter")) +" WHERE
gid = "+rs.getString("gid");

                      System.out.println(sql);

                s1.execute(sql);

                }

          } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
          }

    }

    /**
     * Fator de formula para semelhança com um circulo
     * @param area
     * @param perimetro
     * @return
     */
    public static double formFactor(double area, double perimetro) {
      if (area == 0 || perimetro == 0) {
            return 0;
      }else {
            return   perimetro   /   ((   2*   Math.sqrt(Math.PI)   *
Math.sqrt(area)));
      }

    }
```

```
    public static ArrayList unrealizedDetect(Connection conn, String
itTable) {

     System.out.println("#####################      unrealizedDetect
############### table "+itTable);

     Statement s, s1;
     String sql = "";

     ArrayList moveUnRealized = null;

     try {
               s = conn.createStatement();

               String  query  =  "  select  astext(the_geom)  from
"+itTable;
               ResultSet rs = s.executeQuery(query);

               int idTraj = 1;
               ArrayList<Trajectory>     trajetorias     =     new
ArrayList<Trajectory>();

               int timeIndex = 1;

               while(rs.next()){

                    Trajectory trajectory = new Trajectory(-1);
                    trajectory.tid = idTraj++;

                    String[]               pontos               =
rs.getString(1).replaceAll("LINESTRING|\\(|\\)", "").split(",");

                    for (String string : pontos) {


                         System.out.println(string);
```

```
                        Point p = new Point("POINT("+string+")");

                        Timestamp time = new Timestamp(new
Date().getTime());

                        GPSPoint gps = new
GPSPoint(trajectory.objectId, trajectory.tid,time,p,timeIndex);
                    gps.gid = timeIndex;
                    trajectory.points.addElement(gps);

                    // guarda o ponto inicial do itinerario
                    if (timeIndex == 1)
                                trajectory.ptInicial = gps;

                    // guarda o ponto final do itineario
                    trajectory.ptInicial = gps;

                    timeIndex++;
                      }

                      trajetorias.add(trajectory);


                }

                //System.out.println("###########
"+trajetorias.toString());

                //System.out.println("###########
getCurrentNameTableMove
"+TrajectoryFrame.getCurrentNameTableMove());


     //TrajectoryFrame.setCurrentNameTableMove("iti_moves_recorte_tr
aj_944_914_1_as_0_9_mt_30_ms_1_1");
```

```
                Object[]            moves            =            new
Object[]{TrajectoryFrame.getCurrentNameTableMove()};

                System.out.println("ittable "+itTable);

                InterceptsG                    i                    =
createInterceptsItinerario(conn,trajetorias,moves,itTable);

                System.out.println("Intercepts "+i.toString());

                //  desenvolver  a  logica  do  teste  similar  ao
moveDetect
                int cont = 0;
                ArrayList moveUnRealizedAux = new ArrayList();
                moveUnRealized = new ArrayList();

                for (Trajectory traj : trajetorias) {

                     ArrayList<MoveUnRealized> movesUnrealized = new
ArrayList<MoveUnRealized>();

                     for (GPSPoint p: traj.points) {


                          if ( cont++% 100 == 0 )
                          System.out.println("Move        Unrealized
Detect qtd "+ cont + " "+ new Date());

                          // compara se existe intersecao no trecho

                          Interc              trecho              =
i.isIntersecTrecho(p.gid);

                          if (trecho == null) {

                               moveUnRealizedAux.add(p);          //
adiciona o pont no conjunto de unrealized
```

```
                    }else {

                         // movesUnrealized.add
                        // so cria se tive mais que 1 ponto
                         if (moveUnRealizedAux.size() > 1) {
                              moveUnRealized.add(new
MoveUnRealized(traj.objectId,traj.tid,
moveUnRealizedAux,"itinerario"));
                         }
                         moveUnRealizedAux.clear();
                    }

               }

          }

          System.out.println("@@@@@@@      Move      Unrealized
"+moveUnRealized.size());
          System.out.println("@@@@@@@      Move      Unrealized
"+moveUnRealized);


          System.out.println(" antes de salvar unrealized ");
          saveMovesIBModel(moveUnRealized, conn, 0);
          System.out.println(" depois de salvar unrealized");

     } catch (SQLException e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
     }

  return moveUnRealized;
 }
```

```
    private static InterceptsG createInterceptsItinerario(Connection
conn, ArrayList<Trajectory> trajetorias, Object[] moves, String
interceptName) throws SQLException{

       // definicoes
      double buffer = 0.5;

      // tabela com pontos ( itinerario )
      String table = interceptName+"_pt";

      Statement s = conn.createStatement();
      Statement s1 = conn.createStatement();

      // TODO criar tabela com os pontos do itinerario
      try {
            s1.execute("DROP TABLE "+table);
            s1.execute("DELETE    FROM    geometry_columns    WHERE
f_table_name = '"+table+"'");

        } catch (SQLException ex) {
           ex.printStackTrace();
        }finally {
           s1.execute(
                     "CREATE TABLE "+table+" ("+
                     "      gid integer NOT NULL,"+
                    // "        time timestamp without time zone,"
      +
                     "        CONSTRAINT "+table+"_pkey PRIMARY KEY
(gid)"+
                     ")WITHOUT OIDS"
                  );
        s1.execute("SELECT            AddGeometryColumn('"+table+"',
'the_geom','-1', 'POINT', 2)");
        }

        try {
```

```
            s1.execute("ALTER TABLE "+table+" DROP CONSTRAINT
enforce_geotype_the_geom");
        } catch (SQLException ex) {
            try {
                s1.execute("ALTER TABLE "+table+" DROP CONSTRAINT
\"$2\"");
            }catch (SQLException e) {
                ex.printStackTrace();
            }
        }
     // com gid e the_geom

     for (Trajectory traj : trajetorias) {

          System.out.println("@@@@ trajetoria "+traj.tid);

          for (GPSPoint ponto : traj.points) {

               String sqlInsert = "INSERT INTO "+table+"
(gid,the_geom) VALUES "+

"("+ponto.gid+",'POINT"+ponto.point.getValue()+"')";

               s1.execute(sqlInsert);
          }

          }



          // tabela dos moves planned encontrados
          Object[] objs = moves;

          AssociatedParameter[] relevantFeatures = new
AssociatedParameter[1];
         for (int i=0;i<objs.length;i++) {
```

```
            relevantFeatures[i] = new
AssociatedParameter(objs[i].toString(),"MULTILINESTRING");
        }
        //for each rf, execute the query and save, in main memory,
the results

        InterceptsG intercs = new InterceptsG();

        for(AssociatedParameter a:relevantFeatures){

         // Create a table of registers with:
         // pt -> gid of the trajectory point
         // gid -> gid from RF wich intercept it
         // rf -> rf_name
         java.util.Date  tempo2,fim2,ini2  =  new  java.util.Date();

         System.out.println("\t\t...with "+a.name);

         String sql = "";

         String sqlCount = "";

         if(a.type.contains("LINESTRING")                    ||
a.type.contains("MULTILINESTRING")) {

              // realizar o buffer para as multilinestring

              try {
                   s.execute("DROP TABLE "+a.name+"_buf;");
               } catch (SQLException ex) {
                   // do nothing
               } finally {

                   s.execute("create   table   "+a.name+"_buf   as
select  gid,   buffer(the_geom,"+(buffer/1000)+")  as  the_geom  from
"+a.name+" where move_type = 'planned' ;");
```

```
                    s.execute("alter   table   "+a.name+"_buf   add
constraint "+a.name+"_buf_pk primary key (gid);");
                 }
               sql=("select A.gid as pt, B.gid as gid, '"+a.name+"'
as rf "+
                    "from "+table+" A,"+a.name+"_buf B " +
                    "where
st_intersects(A.the_geom,B.the_geom);");

               sqlCount=("select     count(*)     from     "+table+"
A,"+a.name+"_buf           B           "           +           "where
st_intersects(A.the_geom,B.the_geom);");

          }


          System.out.println(" ##### SQL Count "+ sqlCount);

          System.out.println(" ##### SQL "+ sql);

          if (!sqlCount.equals("")) {
               ResultSet rsCount = s.executeQuery(sqlCount);
               rsCount.next();
               System.out.println("###     Interceps     size     "
+rsCount.getInt(1));
          }

             ResultSet Intercep = s.executeQuery(sql);
             fim2 = new java.util.Date();
             tempo2     =     new     java.util.Date(fim2.getTime()-
ini2.getTime());
             System.out.println("\t\t"+a.name+"       time:       "
+tempo2.getTime()+" ms");
             // then, save the registers from the query in an
adequate struct

             int cont = 0;
```

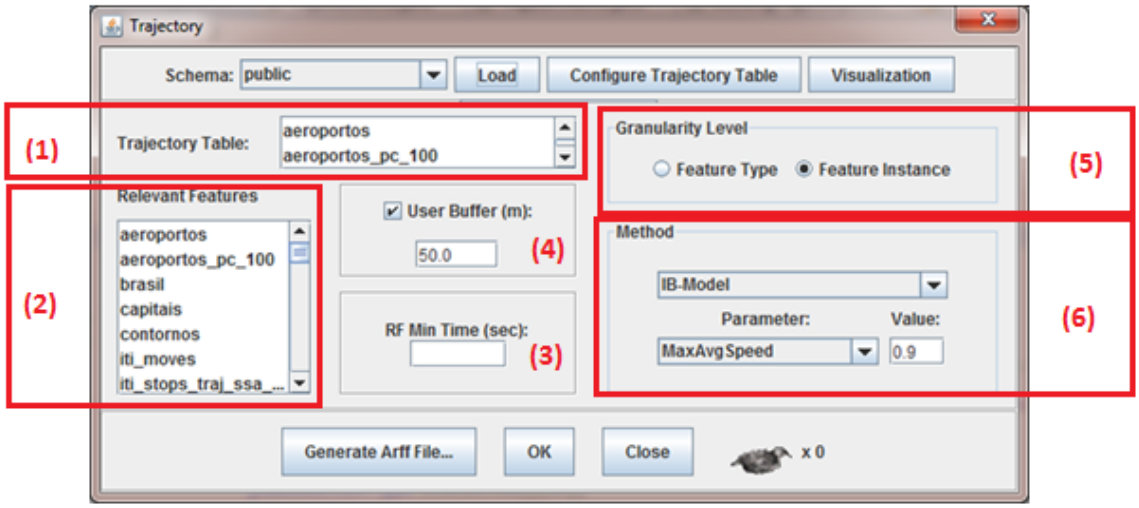
```
            //System.out.println("###         Interceps         size
"+Intercep.getsFetchSize());

            while(Intercep.next()){

               if ( cont++% 10000 == 0 )
                     System.out.println("Interceps qtd "+ cont + "
"+ new Date());

               Interc  i  =  new  Interc  (Intercep.getInt("pt"),
Intercep.getInt("gid"),
Intercep.getString("rf"),a.value.intValue());
               intercs.addTrecho(i);

               // adiciona o name feature relacionado
               intercs.addNameFeature(a.name);

            }

         }

     /* for (Interc item : intercs.trechos) {

        System.out.println("@@@@@ gid "+item.gid);
        System.out.println("@@@@@ pt "+item.pt);

      }    */

        return intercs;//intercs;
     }


    public static void main(String[] args) {


        String url = "jdbc:postgresql://localhost:5432/datario";
```

```
        String user = "postgres";
        String pass = "postgres";

           try {
                Connection  conn  =  DriverManager.getConnection(url,
user, pass);

                //stops_traj_ssa_1_as_0_9_mt_30_ms_1_0
iti_stops_traj_ssa_6_as_0_9_mt_30_ms_1_1

     TrajectoryFrame.setCurrentNameTableStop("iti_stops_traj_linha_4
60_a41193_as_0_9_mt_30_ms_1_1");

                calculateFactor(conn);

                //unrealizedDetect(conn,"itinerario_rio_460");

           } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
           }

     }
}
```

ANEXO B – GUIA DE USO DO WEKA-STPM

O STPM (*Semantic Trajectory Preprocessing Module*) é o módulo para processar trajetórias obtidas a partir de dispositivos móveis. As trajetórias utilizadas são uma sequência de pontos (objetos espaciais) no tempo. Estas trajetórias são processadas e transformadas em *Stops* e *Moves*. O primeiro equivale aos pontos de interesse da trajetória, geralmente aglomerados de pontos “lentos” em relação a velocidade da trajetória sendo explorada. Maiores detalhes podem ser vistos na Figura 3 em (6) ou (7).

Figura 3 – Weka-STPM janela principal.



(1) - Trajectory Table:

É a seleção de uma ou mais tabelas com dados espaço-temporais. Cada tabela tem de conter um atributo “time” e outro atributo “tid”. Se isso não acontecer, deve-se selecionar cada tabela individualmente e configurá-la com o botão “Configure Table”. Usa-se a tecla CTRL (pressionada) para se escolher mais de uma tabela de trajetórias.

(2) - Relevant Features:

É uma lista de seleção de uma ou mais tabelas as quais serão utilizadas pelo método escolhido pelo usuário. Pode-se pensar que a trajetória escolhida em (1) será analisada de acordo com as tabelas desta lista. Exemplificando, a trajetória

“traj” da figura será avaliada de acordo com as escolas da tabela “escola” para identificar *Stops* e *Moves*.

(3) - RF Min Time (sec):

Para cada item selecionado na lista em (2) pode-se definir um Tempo Mínimo, em segundos. Se uma trajetória permanecer interceptando aquele objeto selecionado em (2) por um tempo maior do que o Tempo Mínimo, ela é considerada um *Stop*. A cada item selecionado, pode ser digitado um valor diferente nesta caixa e, logo após, apertar a tecla ENTER (ou TAB) para que ele seja salvo. O valor padrão é de 60 segundos.

(4) - User Buffer (m):

Para corrigir possíveis imperfeições de medidas geográficas causadas por dispositivos móveis, sejam estes erros nos dados das trajetórias ou nos dos Relevant Features, é feito uma “expansão” em cada registro de cada Relevant Feature selecionado. Este valor “expande” o tamanho destes objetos num raio (em metros) definido na caixa de seleção apropriada. Deve ser escolhido apenas um valor para todos os Relevant Features selecionados. O valor padrão é de 50 metros.

(5) - Granularity Level:

Seleção para informar se as tabelas serão usadas a nível de Instância (onde cada registro de cada tabela é um elemento separado) ou a nível de Tipo (onde cada tabela é um elemento). Por exemplo, numa tabela de escolas de uma região teríamos, se fosse pré-processada a nível de Instância, que a escola São Pedro seria diferente da escola Anchieta. Por outro lado, se não quiséssemos tal diferenciação, poderíamos escolher a granularidade de Tipo e teríamos que todas as ocorrências, tanto da escola São Pedro quanto da escola Anchieta e também de todas as outras escolas da tabela, seriam tratadas como "escola".

(6) - Method:

Nesta parte da interface é escolhido o método que será operado sobre a(s) trajetória(s) escolhida(s), considerando todos os Relevant Features e outras características adequadas. Três métodos estão disponíveis: o SMoT (*Stops and Moves of Trajectories*), o CB-SMoT (*Cluster Based Stops and Moves of Trajectories*)

e o IB-SMoT (*Itinerary Based Stops and Moves Trajectories*), descritos em seções particulares adiante.

Uso Geral

Seguindo a Figura 3, devemos selecionar uma ou mais tabelas de trajetórias em (1), alguns itens a serem avaliados em (2) e seus tempos mínimos em (3), definir a "expansão" que deve ser feita em cada item em (4), avaliar se queremos granularidade por tipo ou por instância e selecionar o método que queremos aplicar a esses dados, em (5).

Independente de qual método for escolhido, ele gerará uma tabela de *Stops* e uma tabela de *Moves* com as anotações semântica inerentes ao método selecionado no item (5).