CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Proposta de solução para rotas de veículos de carga

ANDERSON MIYADA

Marília, 2016

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Proposta de solução para rotas de veículos de carga

Monografia apresentada ao Centro Universitário Eurípedes de Marília como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Rodolfo Barros Chiaramonte.

Marília, 2016

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
MANTIDO PELA FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Anderson Miyada

PROPOSTA DE SOLUÇÃO PARA ROTAS DE VEÍCULOS DE CARGA.

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 10 (Dez)

Orientador:Rodolfo Barros Chiaramonte

1°.Examinador:Fabio Piola Navarro

2°.Examinador:Allan Mariano de Souza

Marília, 06 de dezembro de 2016.

DEDICATÓRIA

Dedico este trabalho à minha esposa Michele e aos meus filhos Rafaela e Gustavo, pois sem o apoio e compreensão este trabalho não seria concluído.

Aos meus pais que em toda a minha vida estiveram sempre ao meu lado e principalmente dando todo apoio necessário para minha formação.

AGRADECIMENTOS

Primeiramente a Deus pela conquista de mais uma etapa em minha vida, me permitindo evoluir como ser humano e em conhecimento.

Agradeço ao meu orientador professor Me. Rodolfo pelos momentos de muita paciência para comigo, pela orientação em todos os meus questionamentos e a preocupação em me mostrar o melhor caminho a seguir para o desenvolvimento do meu trabalho.

A minha esposa Michele e meus filhos Rafaela e Gustavo, pela paciência e compreensão da privação de muitos momentos em família que tive que me ausentar.

Aos meus pais, que foram fundamentais no meu desenvolvimento como pessoa, me mostrando sempre o melhor caminho.

Ao professor Me. Bruno Marques por sua atenção, que sempre me atendeu prontamente me auxiliando na medida do possível.

Não poderia deixar também de agradecer ao meu amigo Eng. Rogério, pelos momentos de compartilhamento de seu conhecimento e também pelo seu apoio e motivação.

E a todos que de alguma forma me incentivaram por esta caminhada, explicita ou implicitamente.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

LISTA DE SIGLAS

A*:	A estrela
AASHTO:	*Americam Association of State Highway and Transportation Officials*
AC:	Arco Central
API:	*Application Programming Interface*
APP:	*Application* (Applicativo para smartphone)
CH:	*Contraction Hierarchies*
CNT:	Confederação Nacional de Transporte
CONTRAM:	Conselho Nacional de Trânsito
DEINFRA:	Departamento Estadual de Infraestrutura.
DLR:	Centro Aeroespacial da Alemanha
DLR:	*Deutschen Zentrums für Luft- und Raumfahrt (German Aerospace Center)*
DNIT:	Departamento Nacional de Infraestrutura de Transportes
FENABRAVE:	Federação Nacional da Distribuição de Veículos Automotores
GPL:	*General Public License*
GPS :	*Global Positioning System*
INMETRO:	Instituto Nacional de Metrologia
IoT:	*Internet of Things*
IPR:	Instituto de Pesquisas Rodoviárias
NASA:	*National Aeronautics and Space Administration*
NPE:	*New Popular Edition*
O/D:	Matriz Origem/destino
ODbL:	*Open Databases License*
OSM:	OpenStreetMap
POI:	*Point of Interest*
QFV:	Quadro de Fabricantes de Veículos
SIG:	Sistema de Informação Geográfica
SUMO:	*Simulation of Urban MObility*
TDA:	Taxa de Dificuldade de Acesso
TDE:	Taxa de Dificuldade de Entrega
TRT:	Taxa de Restrição ao Trânsito

VANET:	*Veicular Ad hoc NETworks*
XML:	*Extensible Markup Language*
CEFET-MG	Centro Federal de Educação Tecnológica de Minas Gerais
ITT	Impedância de Transposição de Trecho
RTIGIS	*Real Time Intelligent Geografic Information System*

MIYADA, Anderson. **Proposta de solução para rotas de veículos de carga**. 2016. 63f. Trabalho de curso. (Bacharelado em Ciência da Computação)-Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2016.

RESUMO

Devido a crescente demanda de entregas em domicílio e o fato de que as empresas tendem a diminuir a quantidade de produtos em estoques, gera uma necessidade de entrega mais eficaz. Veículos de transportes de cargas levam mais tempo do que veículos de passageiros para chegar ao seu destino devido a suas dimensões, muitas vezes ao seguir uma rota definida por um GPS não é possível realizar a curva, causando congestionamentos e até acidentes. Este trabalho visa verificar e restringir ruas pertencentes às rotas definidas pelo algoritmo de Dijkstra utilizado pelo SUMO, em conjunto com o mapa do OpenStreetMap, analisando a largura das ruas e as dimensões do veículo de carga. Auxiliando na busca por ruas que possam ser trafegadas por um veículo de carga minimizando o uso de manobras, mesmo que para isso não seja a rota de menor distância.

Palavra-Chave: Veículo de Carga, SUMO, OpenStreetMap, Dijkstra.

MIYADA, Anderson. **Proposta de solução para rotas de veículos de carga**. 2016. 63f. Trabalho de curso. (Bacharelado em Ciência da Computação)-Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2016.

ABSTRACT

Due to the increasing demand for home deliveries and the fact that companies tend to decrease the quantity of products in inventory, it generates a need for more efficient delivery. Cargo transport vehicles take longer than passenger vehicles to reach their destination due to their dimensions, often following a route defined by a GPS cannot make the curve, causing congestion and even accidents. This work aims to verify and restrict streets belonging to the routes defined by the Dijkstra algorithm used by SUMO, together with the map of the OpenStreetMap, analyzing the width of the streets and the dimensions of the load vehicle. Helping in the search for streets that can be trafficked by a vehicle of load minimizing the use of maneuvers, even if for that it is not the route of smaller distance.

Keyword: Cargo Vehicle, SUMO, OpenStreetMap, Dijkstra.

INTRODUÇÃO

A movimentação de cargas é de suma importância para a economia de qualquer país ou cidade, devido ao aumento da competitividade das empresas e as vendas pela internet o volume de entrega de mercadorias aumentou significativamente.

O Boletim Estatístico divulgado pela Confederação Nacional de Transportes (CNT) comprova a importância do transporte de cargas pelas rodovias do país. Cerca de 61% das cargas transportadas no Brasil, foram feitas através do modal rodoviário (CNT, 2016).

De acordo com os dados divulgados no Anuário FENABRAVE (2015, p. 49), o Brasil possuía uma frota de 3.134.441 caminhões no ano de 2015. Devido à quantidade de veículos de carga e atrelado a construção e conservação das ruas, os veículos de transporte de cargas no meio urbano passam por várias dificuldades no que se refere à mobilidade dos caminhões.

Estes fatores geram congestionamentos e dificuldades de acesso em grandes centros, desta forma seu desempenho é reduzido. É fato comum se deparar com dificuldades de tráfego onde existem várias restrições ao acesso às vias públicas, sendo horários de acesso restrito, capacidade total de peso suportado pelo pavimento, altura de viadutos e principalmente a largura das vias.

Para os veículos de carga o simples fato de realizar uma curva, pode se tornar um transtorno. O veículo de carga devido ao seu comprimento necessita de um espaço maior para realizar uma curva do que um carro de passeio. Na maioria das vezes é preciso realizar várias manobras para conseguir realizar uma curva, causando congestionamentos e até acidentes.

Este procedimento causa transtornos, pois outros motoristas despercebidos não dão espaço suficiente para estas manobras, dificultando o trânsito e impedindo o caminhão de se locomover. De outro lado, os aparelhos de Sistema de Posicionamento Global (GPS) da atualidade definem rotas para os veículos não levando em consideração suas dimensões, que no caso se um caminhão for impossibilitado de trafegar por uma rua, ele recalcula a rota, mas na maioria das vezes passa pelo mesmo local, fazendo com que o condutor perca tempo e combustível e tendo que seguir por sua própria experiência ou com informações de terceiros.

A proposta deste trabalho elabora um algoritmo de possível solução para suprir uma necessidade do dia-a-dia do condutor, de optar por uma rota em que seu veículo possa trafegar sem maiores problemas através das vias públicas com largura suficiente de acordo com as dimensões do seu veículo.

OBJETIVOS

Este trabalho propõe a criação de um Algoritmo de Restrição de Giro, que tem por objetivo ser executado independente, mas atuando em conjunto com o módulo de roteamento do SUMO chamado de DUAROUTER, na definição de rotas urbanas auxiliado pelo mapa OpenStreetMap (OSM).

Onde será analisada a largura das vias e as dimensões do veículo de carga, auxiliando na definição do trajeto por ruas que tenha largura suficiente para realizar uma curva minimizando o uso de manobras. Permitindo que chegue até o destino sem maiores problemas de locomoção.

JUSTIFICATIVA

Devido a crescente demanda de entregas em domicílio e o fato de que as empresas estão diminuindo a quantidade de produtos em estoques, gera uma necessidade de entrega mais eficaz.

Veículos de transportes de cargas perdem muito tempo ao chegar a determinadas localidades, pois muitas vezes ao seguir uma rota em que não é possível fazer uma curva, causando congestionamentos e até acidentes.

Isto reflete diretamente no custo do frete, as transportadoras no intuito de compensar estas dificuldades, criaram várias taxas que são incluídas no frete, sendo elas: Taxa de Dificuldade de Entrega (TDE), Taxa de Restrição ao Trânsito (TRT) e a Taxa de Dificuldade de Acesso (TDA).

TRABALHOS CORRELATOS

Um estudo realizado por Carnielle (2009) na cidade de São Carlos, interior do Estado de São Paulo, avalia a locomoção de veículos de carga que realizam entregas em um determinado Supermercado. Utiliza como ferramenta o *software* TransCAD para criação da rede viária de simulação. As informações da geometria das vias foram adquiridas através de órgãos públicos regulamentadores da malha viária urbana.

Para analisar a compatibilidade entre as dimensões disponíveis de uma interseção, com o espaço necessário do veículo para realizar a curva, foi utilizado o *software* AutoTURN. Através do AutoTURN é gerado o gabarito de giro do veículo e comparado com as dimensões das interseções que fazem parte da rota do veículo em análise.

Identificadas as restrições, foi elaborada uma tabela de restrições de interseção, e posteriormente foi restringido estas interseções diretamente no *software* TransCAD.

Portanto, não foi utilizado nenhum algoritmo específico para a análise da compatibilidade entre o veículo de carga com a rota definida.

O trabalho realizado por Silva (2006) é a segunda etapa do projeto Sistemas Inteligentes e Geoprocessamento (GEOPROC), desenvolvido no Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG).

O autor realiza a implementação de uma ferramenta de código aberto chamada de *Real Time Intelligent Geografic Information System* (RTIGIS) que, dado uma base de dados sobre o sistema viário e de um atributo denominado de Impedância de Transposição de Trecho (ITT) que identifica a dificuldade de transposição de um determinado trecho de via. Tendo como base o comprimento, números de faixas de trânsito e também a quantidade de veículos em determinado trecho.

O ITT é quantificado entre 0 e 10, sendo quanto mais próximo de zero é um trecho de fácil transposição, no entanto, quanto mais próximo de 10 maior dificuldade de transposição do trecho.

Tendo este atributo definido o RITGIS gera um melhor caminho entre a origem e o destino. O melhor caminho neste caso é resultado da avaliação do ITT de cada segmento da rota definida, evitando os trechos com mais dificuldade de transposição, consequentemente criando rotas alternativas até o destino.

Este trabalho não trata classes de veículos e, portanto, não considera se veículos de maiores dimensões podem realizar o trajeto.

ESTRUTURA DO TRABALHO

O presente trabalho está divido em 7 capítulos incluindo o presente, que apresenta uma introdução ao tema abordado, justificativa e sua estrutura.

No capítulo 1 são apresentados tipos de mapa de dados geográficos abertos e proprietário. Aborda também o Simulador de Tráfego Urbano SUMO e seus módulos de geração de rotas. Por fim, são apresentados os tipos de veículos de projetos definidos para projetos de rodovias, efeito arraste e o cálculo de giro do veiculo de projeto.

O capítulo 2 aborda os algoritmos de definição de rotas que são utilizados pelo SUMO.

Em seguida, no capítulo 3 são apresentados os métodos utilizados para o desenvolvimento da proposta de solução, sendo alteração do mapa do OpenStreetMap e a estrutura do algoritmo criado para presente proposta.

No capítulo 4 é apresentado o desenvolvimento da proposta de solução.

O capítulo 5 demonstra a aplicação no ambiente de simulação do SUMO.

No capítulo 6 são apresentados às conclusões da proposta implementada e também considerações sobre a simulação obtida, também são relatados possíveis trabalhos futuros que podem gerar resultados interessantes para solução de vários problemas envolvendo mobilidade dos veículos de carga ou veículos de grande porte.

1. COMPONENTES DE SIMULAÇÃO DE MOBILIDADE URBANA

Este capítulo apresenta os recursos computacionais necessários para o desenvolvimento deste trabalho.

1.1. Mapas de dados abertos

Atualmente existem vários mapas disponíveis na Internet, alguns possuem restrições específicas para o uso comercial de aplicativos e sistemas que utilizam seus dados.

A seguir serão abordados alguns projetos de mapeamento e suas principais características.

1.1.1. Tracksource

De acordo com o site do projeto (Projeto TrackSource, 2016), se trata de um projeto colaborativo que cria mapas do Brasil e distribui de forma gratuita para uso em aparelhos de GPS da marca Garmin e similares, e também para *Smartphones*, *Tablets*, e computadores compatíveis com os *softwares* de navegação como o *SevenWays* e *Navitel.*

É possível também utilizá-lo com os *softwares* editores de mapas no computador como o *MapSource* e *TrackMaker*.

Seu idealizador foi Alex Rodrigues em 1996, fundando o grupo GPSBrasil com objetivo de estimular a troca de informações. Mas foi em 1998 que Odilon Ferreira Júnior desenvolveu o programa *TrackMaker*, onde possibilitava que trilhas gravadas em aparelhos de GPS pudessem ser editadas e salvas em um banco de dados e posteriormente transferi-los em forma de pistas e rotas.

O *TrackSource* possui licença *Creative Commons* 2.5 e não permite que seus dados possam ser utilizados para fins comerciais.

1.1.2. Google maps

O Google Maps é um mapa global proprietário que pertence à empresa Google. A ferramenta é uma API disponibilizada de forma gratuita. Por se tratar se uma ferramenta proprietária, seu código fonte não é acessível (Termos de Utilização do Google, 2016), para aplicações que não sejam de distribuição gratuita, o acesso a API é pago (API do Google Maps, 2016).

De acordo com uma pesquisa realizada pela empresa *GlobalWebIndex*, especializada em pesquisas on-line voltadas para empresas e marketing, em agosto de 2013 divulgou que o *Google Maps* com foi o APP para Smartphone mais utilizado no mês anterior da divulgação da pesquisa, a frente até mesmo do *Facebook* e *Youtube* (GLOBALWEBINDEX, 2013).

1.2. OpenStreetMap (OSM)

Segundo a página do OpenStreetMap, é um mapa global de dados abertos sob licença ODbL (Licença de Banco de Dados Aberto), onde seus desenvolvedores fazem parte de uma comunidade voluntária que dá ênfase nos conhecimentos locais (CONTRIBUTORS, 2015a).

O OpenStreetMap é uma fundação sem fins lucrativos que está sediada no Reino Unido e possui aproximadamente dois milhões de usuários do projeto (CONTRIBUTORS, 2016).

Seus voluntários utilizam diversos meios de contribuições seja por fotografias aéreas, localização GPS e mapas locais para garantir de que as informações do OpenStreetMap sejam confiáveis. Os voluntários são dos mais variados, desde entusiastas, profissionais de sistemas geográficos e até engenheiros.

Os mapas atuais pertencem a grandes empresas privadas, que liberam seus dados a um determinado custo e direcionam a atualização dos mapas em determinadas regiões que julgam mais importante financeiramente, daí surgiu à iniciativa de um banco de dados aberto.

Neste sentido o OSM é atualizado com muito mais agilidade, pois seus voluntários de determinadas regiões podem se reunir e concentrar suas contribuições para um bem comum, como no caso do furacão do Haiti onde a região foi toda mapeada em apenas dois dias (CONTRIBUTORS, 2015b).

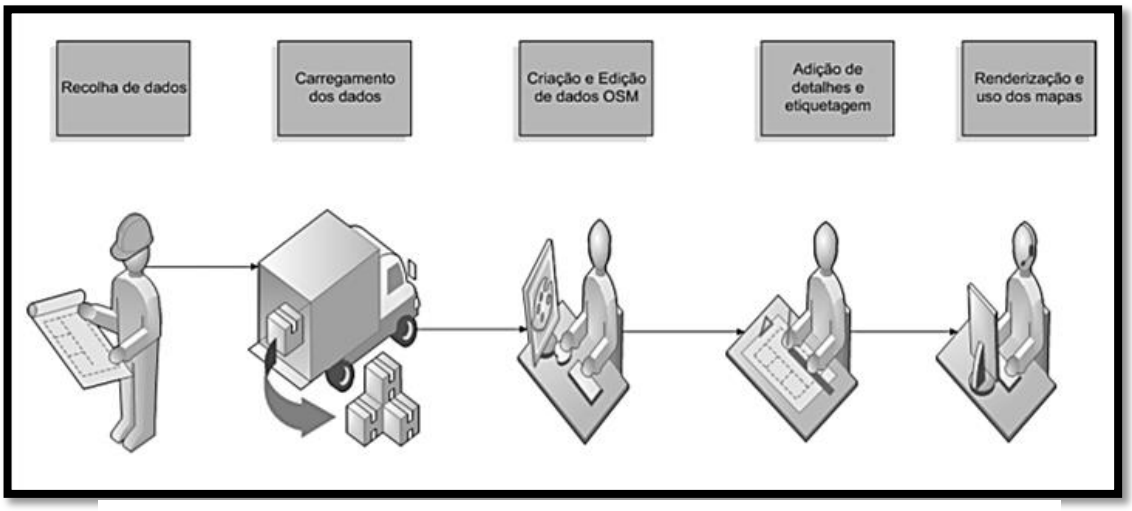
Por se tratar de uma comunidade aberta de desenvolvedores seu código fonte é aberto e disponibilizado a qualquer pessoa. Este benefício facilita a extração dos dados pertinentes à pesquisa de uma proposta de solução para o traçado de rotas para veículos de cargas.

1.2.1. Componentes do mapa OSM

De acordo com Machado (2009), a coleta de dados pode ser feita por dispositivos GPS, extração de mapa de dados do *Yahoo,* imagens fornecidas por satélites livres como o *Landsat*, pertencente a *National Aeronautics and Space administration* (NASA) e por fim acesso aos dados de mapa da *New Popular Edition* (NPE). Há também a possibilidade de utilização de mapas pessoais, desde não tenha nenhuma restrição de direitos autorais.

A seguir, a Figura 1 mostra de forma sucinta o processo de criação dos mapas.

Figura 1- Processo de criação do mapa OSM



Fonte: (MACHADO, 2009, p. 12)

O mapa OSM é composto por alguns elementos.

O **Nó** corresponde a uma coordenada geográfica que contém a latitude e longitude de um ponto específico no globo terrestre. **Nós** são necessários para que sejam definidos os caminhos, mas também podem ser usados de forma desconexa, como por exemplo para identificar um posto de combustível.

Um **Caminho** é uma interconexão entre dois e no máximo 2000 nós, que descreve uma rua ou linha férrea, por exemplo. Um mesmo caminho pode ser subdividido se este possuir propriedades diferentes. Por exemplo, um trajeto de rua que passa de mão única para mão dupla.

As **Áreas** são caminhos fechados onde o Nó inicial é também o Nó final, delimita uma região, mas não necessariamente todo caminho fechado é uma área. Uma área é definida pela interpretação das etiquetas "**tags**", descritas adiante.

Relações são constituídas por dois ou mais elementos que possuem alguma relação entre eles. As relações podem ser de nós, caminhos, área e até mesmo um conjunto de outras relações. Por exemplo, a área de uma cidade relacionada com a área de outras cidades formando um grupo de cidades que se torna um Estado.

Tag são utilizadas para relatar as informações de um determinado nó, caminho ou relação. Cada **tag** possui uma chave e um valor que especifica um elemento nó, caminho ou relação (CONTRIBUTORS, 2015b).

A partir destas informações preliminares é possível ter um conhecimento mínimo necessário para identificar os componentes e manipular e editar o mapa OSM. O mapa possui um recurso de exportar uma localidade de interesse do mapa para um arquivo de extensão XML (***EX****tensible* ***M****arkup* ***L****anguage*), que posteriormente será utilizado pelo SUMO.

Também são oferecidas várias opções de backup dos dados para desenvolvedores, podendo criar softwares específicos utilizando os dados do OpenStreetMap (CONTRIBUTORS, 2013).

1.3. Simulation of Urban MObility

Simulation of Urban Mobility (SUMO) - é um simulador de tráfego rodoviário urbano microscópico, que auxilia no desenvolvimento de projetos de melhorias nas vias públicas, ele foi projetado para suportar simulações de grandes redes rodoviárias, para que sejam testadas antes de serem implantadas.

Através do SUMO é possível realizar também simulações com semáforos, regras de trânsito, alternâncias de faixas pelos veículos, transportes públicos e pedestres. O SUMO possui licença livre (GPL), e é desenvolvido principalmente pelo *Institute of Transportation Systems at the German Aerospace Center* (Instituto de Sistemas de Transporte no Centro Aeroespacial da Alemanha), é desenvolvido na linguagem C++ (DLR, 2001).

Estas características foram fundamentais para a escolha desta ferramenta de simulação de tráfego urbano para o desenvolvimento deste trabalho.

1.4. Módulos para rotas do SUMO

Na versão 0.27.1 do SUMO são cinco módulos de roteamento que estão disponíveis para gerar rotas de viagem, sendo eles: DFROUTER, DUAROUTER, JTRROUTER, MAROUTER e OD2TRIPS.

O DFROUTER tem origem na concepção de que toda rodovia possui um *Loop de Indução*[1] que monitora a entrada ou a saída de determinada rodovia, portanto é possível conhecer a quantidade de veículos que trafegam em determinado tempo. Por meio dessas informações o DFROUTER coleta as informações dos *Loops de Indução* e reconstrói o montante de veículos e rotas (KRAJZEWICZ *et al*., 2016a).

[1]Loop de Indução - Sensor de contagem de veículos instalados no pavimento das rodovias.

DUAROUTER adota a definição de rotas baseada no algoritmo de Dijkstra por padrão, que busca sempre pelo menor caminho entre a origem e o destino. Entretanto é possível definir outros tipos de algoritmos de definição de rotas, como o A Estrela (A*) e o algoritmo *Contraction Hierarchies* (CH).

Também é possível a opção de atribuição dinâmica de rotas para os veículos a cada iteração, onde é verificado o tempo de viagem dos veículos na simulação anterior, alterando as rotas definidas inicialmente para trajetos que tiveram menor tempo de viagem na próxima simulação (MACHADO, 2009, p. 29).

JTRROUTER se baseia no volume de tráfego de veículos que passam por uma interseção. Utiliza a porcentagem de veículos que realizam uma curva em uma interseção como parâmetro de entrada, estas informações são inseridas diretamente no arquivo *turns.xml* onde deve ser definido a probabilidade de um veículo realizar uma curva, o caminho e o intervalo de tempo entre 0 e 3600s (KRAJZEWICZ *et al*., 2012b).

MAROUTER admite uma matriz Origem/Destino[2] (OD) para atribuição de tráfego de veículos por zonas ou distritos, gerando rotas de viagem para uma determinada rede de tráfego (KURCZVEIL; LÓPEZ, 2015). É possível inclusive determinar para a definição de rotas dos veículos, os algoritmos de Dijkstra, A* e CH, que são também utilizados pelo DUAROUTER.

OD2TRIPS importa uma matriz OD proveniente de outros simuladores como o VISUM, VISION e VISSIM que é o mapeamento de padrão no formato “V” e no formato “O” para gerar rotas de viagens.

O formato “V” contém as informações do tipo de veículo, número de distritos, a quantidade de veículos que saem do distrito de origem para o distrito de destino e o intervalo de tempo de início para viagem dos veículos. O formato “O” possui as mesmas informações, mas de uma forma mais objetiva, lista cada origem e destino com seu respectivo valor em cada linha (KRAJZEWICZ, *et al.*, 2016b).

1.5. Veículo de projeto

O Conselho Nacional de Trânsito (CONTRAN) regulamenta as dimensões máximas permitidas para veículos com ou sem carga.

Conforme a resolução do CONTRAN nº 210,

---

[2] Matriz Origem/Destino – Uma matriz que contém a quantidade de veículos que se deslocam de um determinado ponto de origem para seu destino (DNIT, 2005).

> [...] Art. 1º As dimensões autorizadas para veículos, com ou sem carga, são as seguintes:
> I – largura máxima: 2,60m;
> II – altura máxima: 4,40m;
> III – comprimento total:
> [...]
> d) veículos articulados com duas unidades, do tipo caminhão-trator e semi-reboque: máximo de 18,60 metros; [...] (CONTRAN, 2006).

A partir dos limites de dimensões para veículos de carga, cada modelo de veículo possui características técnicas de acordo com estudo e tecnologias incorporadas pelo seu fabricante. Isso gera uma diversidade de veículos de carga, o que inviabiliza a construção de vias com dimensões que satisfaçam todos estes modelos e há também as limitações físicas e geográficas de determinada região.

No Brasil o Instituto de Pesquisas Rodoviárias (IPR) produz normas, manuais entre outras publicações técnicas que são adotados como referência em todas as esferas de governo municipal, estadual e federal (IPR, 2016).

Sendo assim, de acordo com o IPR em seu Manual de Projetos de Interseções (2005), são definidos os tipos de veículos de projeto, que nada mais é que um veículo com dimensões máximas para tráfego na via ou interseção, para elaboração dos cálculos de projeto.

Mas isso não quer dizer que veículos maiores não poderão trafegar por este caminho, dependendo do caso é possível, mas podem ocorrer manobras para correção de sua trajetória, principalmente nas interseções.

Neste manual são especificados cinco tipos básicos de veículos de projeto para cada caso de acordo com a predominância do tráfego. Cada categoria de veículo de projeto é identificada por uma sigla, sendo:

> [...] **VP** - Representa os veículos leves, física e operacionalmente assimiláveis ao automóvel, incluindo minivans, vans, utilitários, pick-ups e similares.
> **CO** - Representa os veículos comerciais rígidos, não articulados. Abrangem os caminhões e ônibus convencionais, normalmente de dois eixos e quatro a seis rodas.
> **O** - Representa os veículos comerciais rígidos de maiores dimensões. Entre estes incluem-se os ônibus urbanos longos, ônibus de longo percurso e de turismo, bem como caminhões longos, freqüentemente com três eixos (trucão), de maiores dimensões que o veículo CO básico. Seu comprimento aproxima-se do limite máximo legal admissível para veículos rígidos.
> **SR** - Representa os veículos comerciais articulados, compostos de uma unidade tratora simples (cavalo mecânico) e um semi-reboque. Seu comprimento aproxima-se do limite máximo legal para veículos dessa categoria.

> **RE** - Representa os veículos comerciais com reboque. É composto de um caminhão trator trucado, um semi-reboque e um reboque, e que mais se aproxima do veículo conhecido como bitrem. Seu comprimento é o máximo permitido pela legislação. [...]. (DNIT, 2005, p. 79).

Na Tabela 1 é apresentado um resumo das dimensões de veículos de projeto para elaboração de projetos de rodovias, interseções e instalações similares.

Tabela 1- Dimensões dos veículos de projeto (metros).

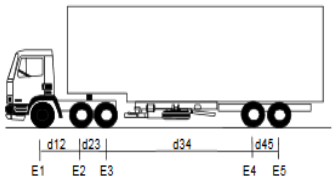
Designação do veículo / Características	Veículos leves (VP)	Caminhões e ônibus convencio-nais (CO)	Caminhões e ônibus longos (O)	Semi-reboques (SR)	Reboques (RE)
Largura total	2,1	2,6	2,6	2,6	2,6
Comprimento total	5,8	9,1	12,2	16,8	19,8
Raio min. da roda externa dianteira	7,3	12,8	12,8	13,7	13,7
Raio min. da roda interna traseira	4,7	8,7	7,1	6,0	6,9

Fonte: Manual de projetos e interseções (2005, p. 80).

Este trabalho adota a categoria SR como veículo de para simulação, pois se trata de um veículo de carga que possui grandes dificuldades de locomoção nas cidades brasileiras.

Na Tabela 2 é apresentada a silhueta do veículo de projeto SR, ou seja, uma unidade tratora e um semirreboque.

Tabela 2- Silhueta unidade tratora e um semirreboque

SILHUETA	GRUPO/ Nº EIXOS	PBT ou PBTC / (5%)	CARACTERIZAÇÃO	CLASSE	CÓDIGO
	3 / 5	**40** / (42)	**CAMINHÃO TRATOR TRUCADO + SEMI-REBOQUE** E1 = eixo simples; carga máxima 6,0 ton. E2E3 = conjunto de eixos em tandem duplo; carga máxima 17 ton. E4E5 = conjunto de eixos em tandem duplo; carga máxima 17 ton. d12, d34 > 2,40 m 1,20 m < d23, d45 < 2,40 m	**3S2**	75

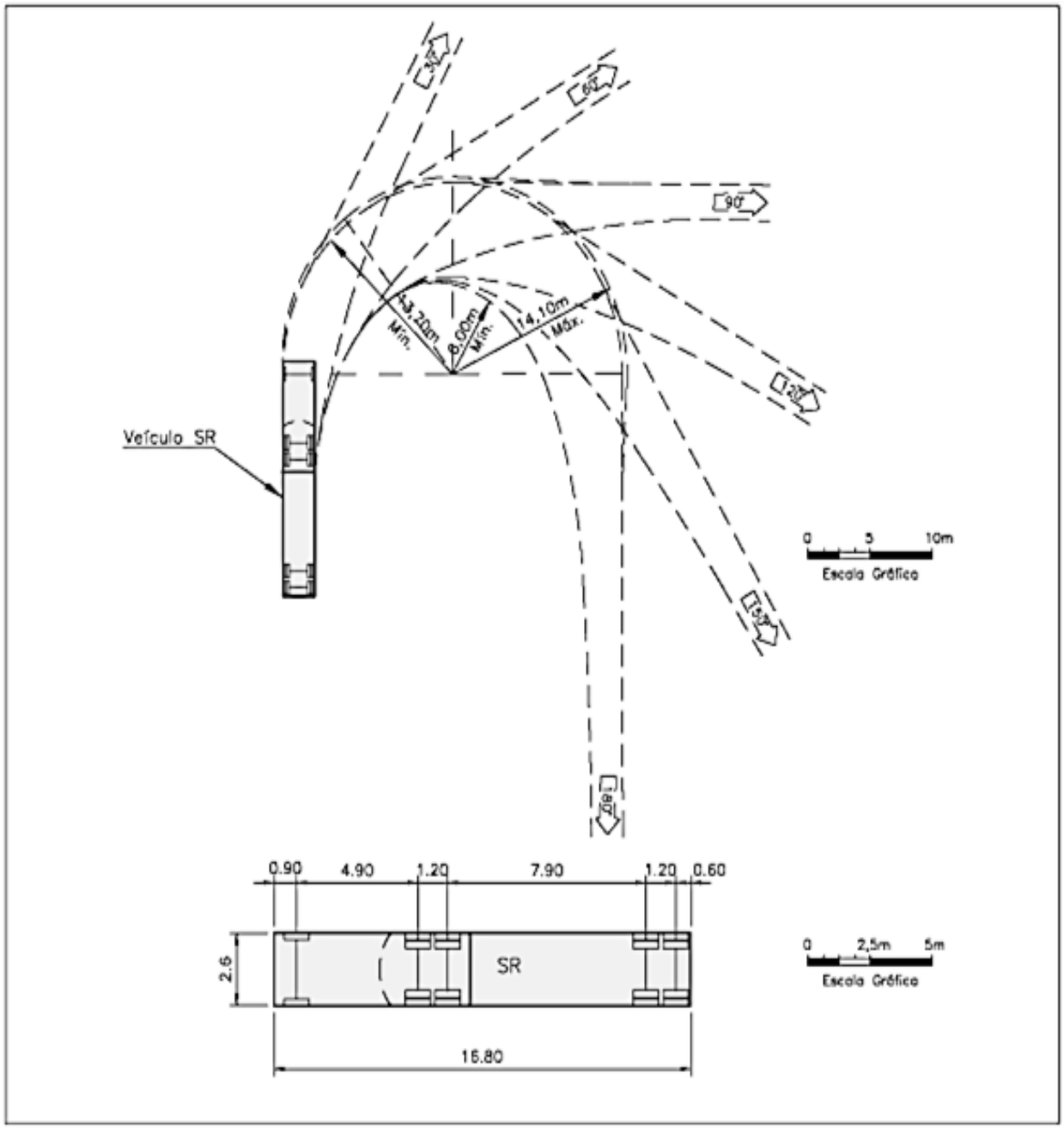
Fonte: DNIT - Quadro de Fabricantes de Veículos – QFV *(2012, on-line)*.

De acordo com o DNIT em seu Quadro de Fabricantes de Veículos – QFV (2012), as dimensões especificadas na imagem foram estabelecidas pelos fabricantes para cada veículo,

e homologados pelo Instituto Nacional de Metrologia, Normalização e Qualidade Industrial (INMETRO).

A Figura 2 mostra a largura necessária para giro de um veículo de carga semirreboque (SR) em vários ângulos diferentes, sendo: 30º, 60º, 90º, 120º, 150º e 180º. O raio máximo representa a trajetória em relação ao canto esquerdo do parachoque dianteiro e o de raio mínimo corresponde o deslocamento do ponto de tangência da média dos eixos traseiro. Logo abaixo do gabarito de giro, são apresentadas as dimensões de largura e comprimento e as distâncias entre os eixos do veículo semirreboque.

Figura 2- Veículo de Projeto SR.



Fonte: IPR - Manual de Projetos e Interseções (2005, p.83).

1.6.Arraste

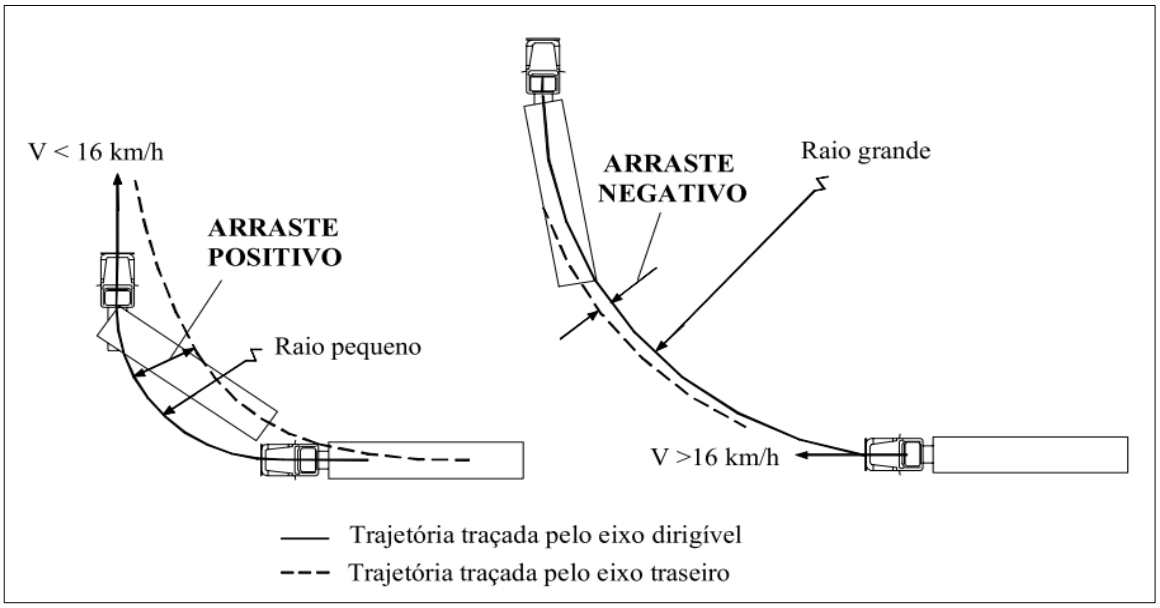
Além da distância entre eixos, outro fator que influencia no espaçamento necessário para realizar um trecho de curva é a velocidade.

Segundo Pereira Neto (2007, p.68),

> [...] Quando a manobra é realizada a baixa velocidade, as rodas do eixo traseiro seguem uma trajetória mais adentro da trajetória seguida pelo eixo dirigível, ocorrendo o chamado arraste positivo, [...]. Quando o veículo executa uma conversão a alta velocidade, é observada a tendência da traseira do veículo se mover para fora da trajetória descrita pelo eixo dirigível em virtude da influência da aceleração lateral, configurando o chamado arraste negativo. [...]

De acordo com o estudo feito por Russo (1995) citado por Pereira Neto (2007, p.69), o arraste positivo ocorre quando um veículo de carga possui velocidade menor que 16 Km/h, produzindo uma trajetória com raio pequeno, conforme demonstrado na Figura 3.

Figura 3- Tipos de Arraste em Conversões.



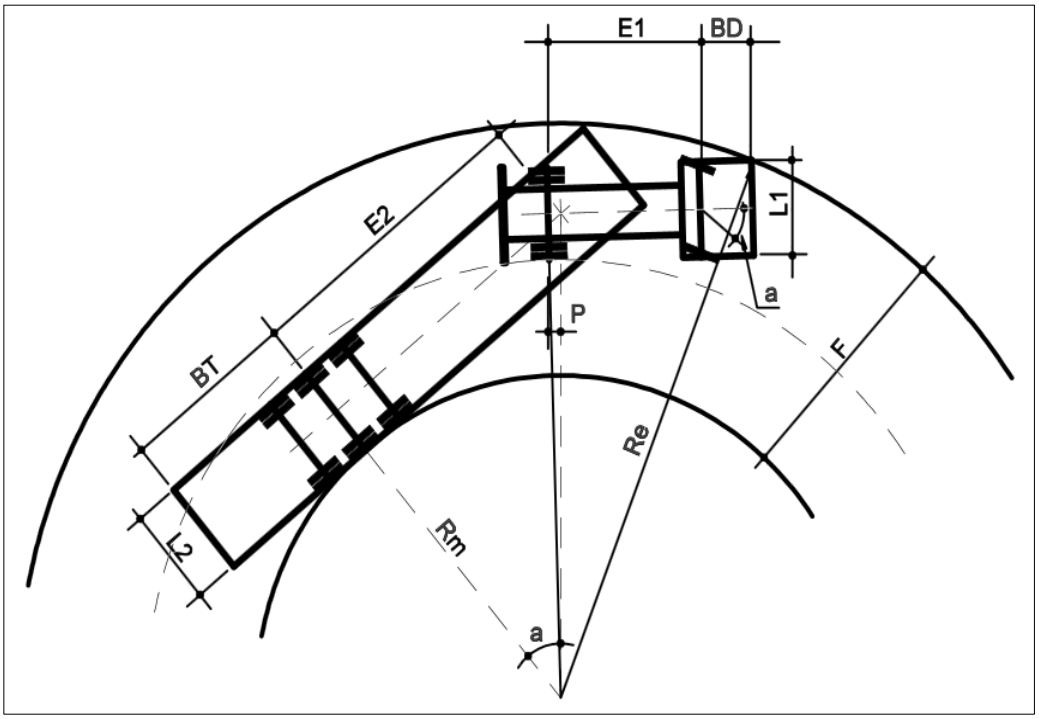
Fonte: Adaptado de Pereira Neto(2007, p.69).

O conceito do arraste é importante, pois nas interseções urbanas a velocidade é reduzida, principalmente em se tratando de veículos longos como o semirreboque, portanto nos cálculos que serão descritos no decorrer deste trabalho é sempre considerado um valor de arraste positivo, ou seja, velocidade abaixo de 16 Km/h, variável que não será encontrada nos cálculos de giro.

1.7. Cálculo de giro para veículo de projeto Semirreboque

O cálculo de giro do veículo de projeto SR foi definido conforme descrito no documento “Gabarito de Giro dos Veículos Representativos da Frota” (DEINFRA, 1998, pág 9). A fórmula apresentada no documento define o raio de giro externo e interno do veículo de projeto. A Figura 4 mostra as variáveis que devem ser consideradas para cálculo de giro.

Figura 4- Informações necessárias para o cálculo de giro.



Fonte: (DEINFRA, 1998, pág 9)

Onde:

$L1$ = Largura do veículo.

$L2$ = Largura do reboque.

BT = Balanço traseiro.

BD = Balanço dianteiro.

$E1$ = Distância entre eixos.

$E2$ = Distância eixo reboque até o pino rei.

P = Distância eixo até o pino rei.

a = Ângulo de giro médio das rodas[3].

Rm = Raio mínimo interno.

[3] Média dos ângulos das rodas dianteiras, direita e esquerda.

Re = Raio externo (Raio de Giro).

F = Re – Rm.

O cálculo de raio mínimo interno e raio externo são definidos conforme a Equação 1 e 2 abaixo.

$$Rm = \frac{E2}{\tan a} - \frac{L2}{2} \quad (1)$$

$$\boldsymbol{Re = \sqrt{\left(\sqrt{\left(\frac{E2}{\tan a}\right)^2 + E2^2 + P^2} + \frac{L1}{2}\right)^2 + (E1 + BD)^2}} \quad (2)$$

1.7. Conclusões do capítulo

O conteúdo apresentado é necessário para que se tenha um conhecimento básico sobre as ferramentas que serão utilizadas nos próximos capítulos.

No SUMO será importado o mapa da região extraída do mapa OpenStreetMap para simulação, auxiliando o algoritmo de definição de rotas para que possa identificar o melhor caminho a ser percorrido pelo veículo de projeto.

2. ALGORITMOS DE ROTAS

Este capítulo é dedicado à introdução aos algoritmos de busca e a apresentação dos principais algoritmos adotados pelo SUMO, sendo o algoritmo de Dijkstra, A Star (A*) e *Contraction Hierarchies* (CH).

2.1. Tipos de algoritmo de busca

A área da Inteligência Artificial de uma forma geral estuda e aprimora algoritmos "inteligentes", onde são encontrados vários algoritmos para solução de problemas utilizando buscas.

Basicamente existem duas classes de algoritmos de busca: *Sem Informação* e *Com Informação.*

Algoritmos de busca *Sem Informação* recebem este nome, pois não são contemplados com nenhuma informação adicional do que é dado em sua definição.

A definição do problema é formada por cinco componentes: estado inicial, função sucessor, teste de objetivo e custo de caminho (RUSSEL, STUART; NORVIG, 2004).

Estado inicial é o nó de início da busca.

Função Sucessor retorna as opções possíveis para que a busca possa ser feita em direção ao objetivo.

Teste de Objetivo verifica se o nó atual é o nó objetivo, ou seja, o destino aonde se quer chegar.

Custo de Caminho define o custo numérico para um caminho.

Algoritmo de busca *Com Informação* possuem informações específicas sobre o problema além daqueles apresentados em sua definição, podendo assim encontrar soluções mais eficientes se comparados aos algoritmos de busca sem informação.

Basicamente a diferença em relação aos algoritmos de busca sem informação é o fato de incluir uma *Função Heurística*.

A *Função Heurística* é uma estimativa do caminho de menor custo da origem até o destino, ou seja, do nó inicial até o nó objetivo(RUSSEL, STUART; NORVIG, 2004).

2.2. Algoritmo de Dijkstra

O algoritmo de Dijkstra foi criado por *Edsger Wybe Dijkstra* em 1959, para resolver a busca pelo menor caminho, ou seja, de custo mínimo nas arestas de um grafo.

Este algoritmo utiliza a chamada Busca Gulosa, escolhendo o vértice de menor custo para que seja adicionado ao conjunto solução.

Dado um vértice inicial, este algoritmo calcula o custo de todas as arestas aos vértices adjacentes, o algoritmo escolhe o caminho para o vértice seguinte de menor custo e memoriza o custo total para chegar até o vértice atual. A partir do novo vértice é verificado novamente os vértices adjacentes e seus custos de caminho, repetindo até que seja atingido o vértice objetivo (SOUZA, 2008, p. 17).

2.3. Algoritmo A Star (A*)

O algoritmo A* (lê-se A estrela) foi criado em 1968 pelos membros, *Peter Hart, Nils Nilsson e Bertram Raphael* do grupo de Inteligência Artificial do Laboratório de Física Aplicada da Universidade de Stanford (HART; NILSSON; BERTRAM, 1968).

Utiliza uma estratégia que é conhecida como *busca com informação*, ou seja, há informação específica do problema que garante uma eficiência maior se comparada com algoritmos de *busca sem informação* (RUSSEL, STUART; NORVIG, 2004, p. 96)**.**

Este algoritmo concentra-se na busca pela melhor escolha em que avalia o custo mais baixo a partir do nó de origem até o nó destino através de uma *Função Heurística*.

O algoritmo A* possui várias vantagens em relação ao algoritmo de Dijkstra, que devido à função heurística, somente expande nós de menor heurística que diminui consideravelmente o uso de alocação de memória e por consequência agiliza o processamento.

2.4. Algoritmo Contraction Hierarchies (ch)

O algoritmo *Contraction Hierarchies* foi desenvolvido por *Robert Geisberger, Peter Sanders, Dominik Schultes* e *Christian Vetter* de acordo com a publicação do artigo no *Journal Transportation Science – informs* no ano de 2012.

A ideia geral do algoritmo é retirar nós de menor importância de uma rede de transporte e criar novas arestas (ruas), preservando as vias de menor distância entre nós vizinhos, conseguindo assim uma contração de nós. Um dos problemas apresentados neste algoritmo, é em relação à quantidade excessiva de arestas criadas, se forem demasiadamente grandes, pode torná-lo lento (GEISBERGER *et al*., 2008, p. 388).

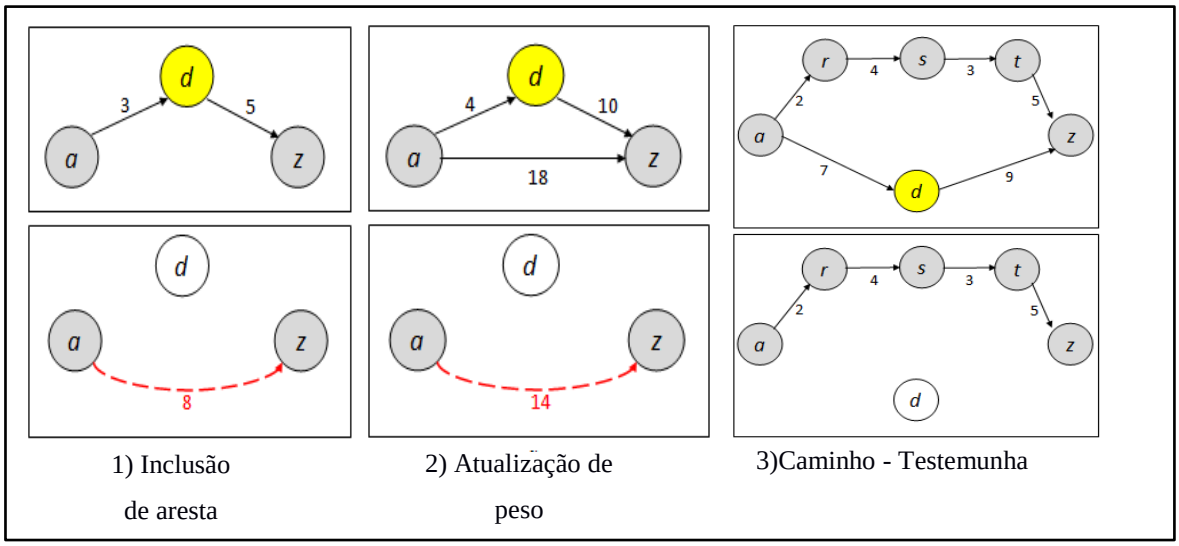
O algoritmo *Contraction Hierachies* é muito utilizado quando se necessita definir rotas de veículos, por se tratar de um algoritmo que consegue retornar uma busca muito

eficaz, reduzindo drasticamente o tempo de execução e também quanto ao uso de memória (PAIM, 2015, pág 13).

A Figura 5 mostra três cenários sendo que o primeiro é a inserção de um atalho entre o nó "*a*" e "*z*" Figura 5(1), em seguida na Figura 5(2), representa a mudança de peso de uma aresta que já existia, e por fim, na Figura 5(3) não é adicionado nenhuma aresta pois já existe um caminho de menor custo, sendo este chamado de *Caminho – Testemunha*.

Para que não se crie arestas desnecessárias, o *Contraction Hierarchies* utiliza outros algoritmos de menor caminho como o Dijkstra, por exemplo, para encontrar um menor caminho entre os nós, sendo executado somente uma vez em cada rede, após concluída, a consulta é feita na rede resultante.(PAIM, 2015, pág 14).

Figura 5- Antes e depois da remoção do nó ***d*** em três cenários.



Fonte:(PAIM, 2015, pág 15), adaptado pelo autor.

Os algoritmos *Higway Hierarchies* e *Highway – Node Routing* foram a base deste algoritmo (GEISBERGER *et al.*, 2008, p. 389).

2.5. Conclusões do capítulo

O SUMO possui o módulo de definição de rotas chamado DUAROUTER, como descrito na seção 1.4, que pode utilizar quatro algoritmos, o algoritmo de Dijkstra, A estrela (A*), *Contraction Hierarchies* e o *Contraction Hierarchies Wrapper*..

Dentre as opções disponibilizadas pelo SUMO, o *Contraction Hierarchies* (CH), segundo Paim (2015, pág 13), está entre os melhores algoritmos para solução de rotas urbanas.

.

3. INTEGRAÇÃO DO MAPA OSM COM O SUMO

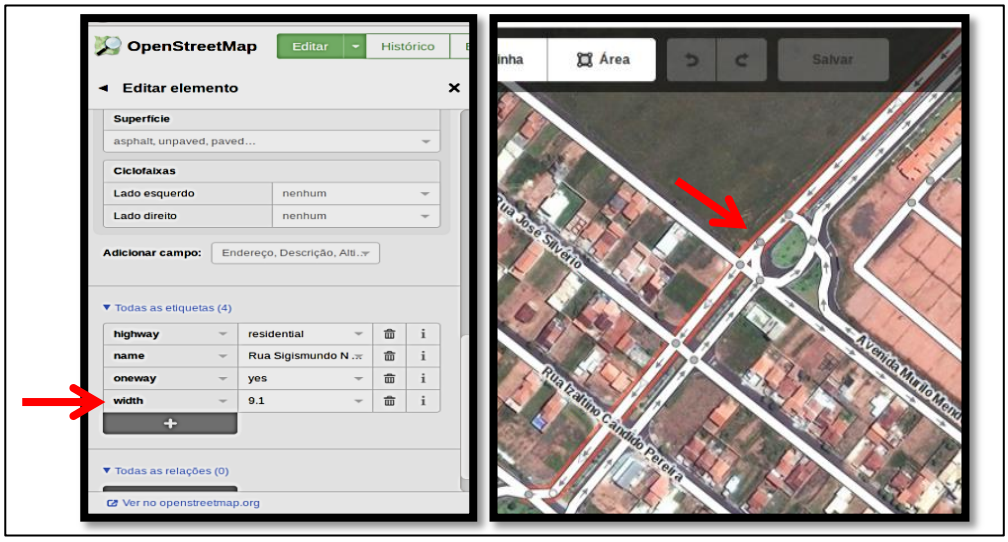
Para que seja possível a simulação no ambiente do SUMO utilizando o mapa do OSM é preciso que o próprio mapa ofereça as informações de largura de ruas. Este dado atualmente não é normalmente inserido pelos colaboradores da comunidade, apesar de possuir TAG para que estes dados sejam disponibilizados nos mapas.

Neste caso, para efeito de simulação foi criado um perfil colaborador no OSM para que fossem efetuados testes para verificar se estes dados são transferidos durante a exportação da região do mapa OSM, que será convertido para o padrão do SUMO. A seguir será descrito os detalhes de como foram inseridos os dados de largura de ruas no mapa OSM.

3.1. Editando o mapa OSM

O mapa OSM possui uma interface onde é possível inserir dados de ruas, descrições de pontos de interesses (POIs) entre outros. A Figura 6 mostra a inserção de uma **tag** *Width* contendo o valor da largura da rua destacada em vermelho.

Figura 6- Destaque da rua com largura inserida.



Fonte: www.openstreetmap.org

.

Com as alterações salvas as informações se tornam disponíveis para todos os usuários, a partir dos dados inseridos no mapa é possível exportá-lo em um arquivo .xml que poderá ser convertido para os padrões do SUMO para geração de rotas urbanas para simulação.

3.2. Importação do mapa OSM no SUMO

Depois de efetuado o *download* da área especificada no mapa OSM é preciso convertê-lo para que o SUMO possa utilizá-lo.

Para realizar a conversão é utilizado o comando em *prompt* de comando chamado de NETCONVERT. Este comando é responsável em identificar os nós e vias do arquivo de extensão .osm gerado pelo OSM e transformá-los em nós e arestas para o arquivo de extensão .net.xml. Este arquivo possui nós, arestas, junções, conexões entre as arestas, incluindo sentido das vias, paradas obrigatórias, ou seja, as restrições necessárias para criação de uma rede rodoviária urbana básica. É possível também serem inseridas outras informações que o usuário deseje como, por exemplo, inserção de semáforos e tempo de transição, gerando assim uma rede de tráfego que possa ser interpretado pelo SUMO.

Em seguida é necessária a criação de um arquivo que identifica polígonos constantes no mapa OSM, utilizando o comando POLYCONVERT juntamente com um arquivo contendo todos os polígonos suportados pelo SUMO disponível na página do próprio simulador.

Na sequência é adicionado um ponto de partida e de chegada, que na realidade se trata de uma aresta (rua) de origem e de destino para que seja possível gerar a viagem (rota) a ser percorrida pelo veículo de projeto. Também é disponibilizado na página do SUMO o script para criação do veículo e definição da viagem, bastando somente serem alterados os dados específicos para cada mapa.

Por fim, é chamado o comando para geração da viagem entre os dois pontos através do DUAROUTER especificando os arquivos necessários de rede rodoviária (net), viagem (*trips*) e o tipo de algoritmo a ser utilizado para geração da rota, se caso não for definido nenhum dos algoritmos mencionados anteriormente (Dijkstra, A*, CH e CH*Wrapper*), o SUMO por padrão utiliza o algoritmo de Dijkstra.

A proposta deste trabalho sugere o auxílio de um algoritmo que analisa se as arestas escolhidas pelo algoritmo de Dijkstra possuem largura necessária para que um veículo de carga possa realizar ou não uma curva.

4. DESENVOLVIMENTO DO ALGORITMO

O algoritmo da proposta de solução foi desenvolvido utilizando o método do Teorema de Pitágoras e de Geometria Plana Euclidiana, tendo um cenário bem definido em relação a suas restrições, buscando uma forma de analisar a possibilidade de conversão para veículos de cargas baseados somente na largura das ruas.

4.1. Definição do escopo do algoritmo

O algoritmo foi desenvolvido levando em consideração interseções com quatro vias de acesso, tendo entre elas ângulos de 90º.

Com relação ao sentido do fluxo da via, no caso de ruas de mão dupla, projetos de ruas com baixo tráfego de veículos, são utilizados veículos de projeto de menores dimensões, diminuindo, portanto a largura total da via.

Entretanto este fato não restringe a circulação de veículos maiores. Toda via veículos de carga como ditos anteriormente, necessitam de maior espaço para conversão utilizando, portanto o espaço reservado à pista de sentido contrário (contramão), próximo às interseções é utilizado para realizar a curva, sendo esta a realidade nas ruas das cidades brasileiras.

Para a simulação não foi considerado um destino inatingível, pois neste caso se a restrição for o próprio trecho do destino ou que não tenha outra rua alternativa, não será possível ao algoritmo de definição de rota do SUMO encontrar uma rota, pois todos os acessos ao destino estarão restritos.

Neste caso o algoritmo não leva em consideração a habilidade do condutor ou a possibilidade de realização de manobras para correção da trajetória em curva, que pode fazer com que uma interseção seja transposta mesmo havendo restrição avaliada pelo algoritmo. Para a análise de cada interseção havendo largura disponível suficiente, é possível que o veículo de carga possa transpor a interseção sem correção de sua trajetória em curva.

Outro fator importante é com relação ao recuo de cada esquina, considerando a largura da calçada de 2,5 metros foi definido um recuo de 1,03 metros para cada lado utilizando o cálculo de recuo descrito na seção 4.3 na Equação 7.

Na Figura 7 são destacadas as restrições de uma interseção aqui descritas.

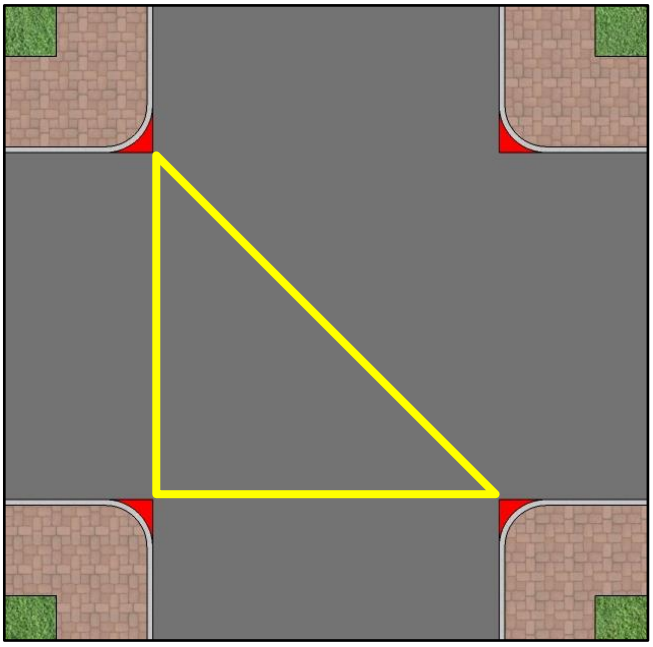
Figura 7- Destaque das restrições

Fonte: Elaborado pelo autor.

4.2. O ALGORITMO DE RESTRIÇÃO DE GIRO

O algoritmo de restrição de giro trata uma interseção como sendo um triângulo isóscele, como mostra a Figura 8 a seguir.

Figura 8- Triângulo na interseção



Fonte: Elaborado pelo autor.

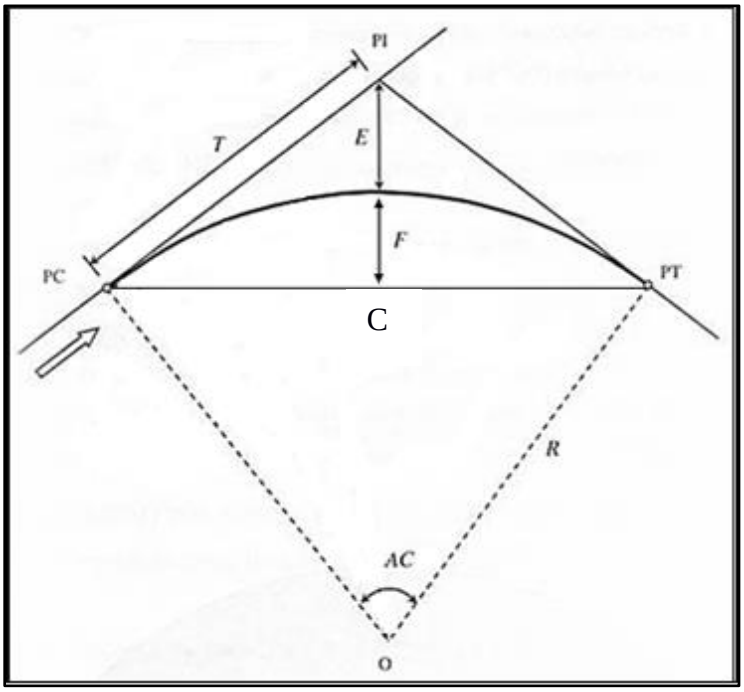
A largura das ruas com interseção de 90º são os catetos, então de acordo com o teorema de Pitágoras o valor da hipotenusa do triângulo é dado pela Equação 3:

$$\boldsymbol{h = \sqrt{a^2 + b^2}} \quad (3)$$

O resultado traz a medida da diagonal da interseção, ou seja, do ponto de interseção de duas calçadas com ângulo de 90º até a outra junção correspondente. Somente com a medida da diagonal da rua não é possível definir se um determinado veículo de carga pode realizar a curva.

Para isso é necessário utilizar conceitos de geometria plana. A Figura 9 a seguir mostra os principais elementos de uma curva circular.

Figura 9- Elementos da curva.



Fonte: (PONTES FILHO, 1998, pag 73). Adaptado pelo autor.

Onde:

PC = Ponto de Curva

PT = Ponto de Tangente

PI = Ponto de Interseção das Tangentes

AC = Ângulo Central da Curva

R = Raio da Curva Circular

O = Centro da Curva

T = Tangente Externa

E = Afastamento

F = Afastamento da Curva à Corda PC à PT

C = Corda

Dentre os elementos apresentados os que são de interesse ao desenvolvimento do algoritmo são a corda (**C**) e o afastamento (**E**).

O tamanho da corda é o valor encontrado pelo cálculo da hipotenusa.

Para que o algoritmo consiga tratar interseções com larguras diferentes de ruas, é preciso comparar a largura disponível para giro dentro da interseção, com a largura de giro do veículo de carga.

Para definir a largura de giro da interseção é necessário ter conhecimento da medida da flecha da curva, que é obtido a partir do valor do recuo (***E***).

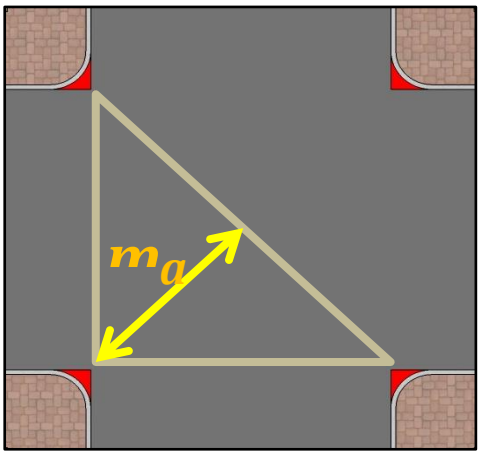
Para isso são necessários alguns cálculos que serão descritos na seção 4.3 a seguir.

4.3. Definindo o recuo da interseção (E)

Primeiramente é necessário conhecer a distância do lado que contém o ângulo reto, ou seja, de 90° até seu lado oposto na qual é a hipotenusa. Esta distância é obtida pela fórmula de *Stewart* para o cálculo das medianas, conforme demonstração de aplicação feita por Macedo (2014, pag 40).

Na Figura 10, são demonstradas as relações para compreensão da fórmula em uma interseção.

Figura 10- Mediana do triângulo (ma).



Fonte: Elaborado pelo autor.

A fórmula de *Stewart* é descrita pela Equação 4.

$$m_a = \frac{1}{2} \cdot \sqrt{2(b^2 + c^2) - a^2} \tag{4}$$

Depois de encontrado o valor da mediana é possível definir a medida do Ângulo Central (AC), de acordo com a razão trigonométrica do *Seno* definida pela Equação 5.

$$\mathbf{AC} = \mathbf{sen}\, \boldsymbol{x} = \frac{\boldsymbol{CO}}{\boldsymbol{H}} \tag{5}$$

E depois convertendo para graus calculando a inversa do *Seno*, ou seja, $\mathrm{sen}^{-1}\, x.$

O passo seguinte é encontrar a distância até o ponto de interseção das tangentes (T) Equação 6, e o afastamento do arco (E) Equação 7, conforme Pontes Filho (1998, pag 77).

$$\boldsymbol{T} = \boldsymbol{R}\ .\ \mathbf{tan}\left(\frac{\Delta}{\mathbf{2}}\right) \tag{6}$$

Onde:

T = Comprimento do ponto de tangência até o ponto de interseção das tangentes.

R = Raio da circunferência.

Δ = Ângulo Central (AC).

O afastamento do arco é dado por:

$$\boldsymbol{E} = \boldsymbol{T}\ .\mathbf{tan}\left(\frac{\Delta}{\mathbf{4}}\right) \tag{7}$$

Onde:

E = Afastamento do arco.

T = Comprimento do ponto de tangência até o ponto de interseção das tangentes.

Δ = Ângulo Central (AC).

Por fim, a medida da flecha é definida pela Equação 8 abaixo.

$$\boldsymbol{F} = \ (\boldsymbol{R} + \boldsymbol{E}) - \ (\boldsymbol{m_a} + \boldsymbol{E}) \tag{8}$$

Onde:

F = Comprimento da Flecha.

R = Raio da circunferência.

E = Afastamento do arco.

m_a = Mediana.

4.4. Definindo a largura de giro da interseção

A partir dos raios externo e interno no momento do giro do veículo de carga é possível quantificar do espaço necessário para realizar a curva. Esta informação deverá ser comparada com a largura disponível na interseção para giro do veículo.

A obtenção da largura de giro da interseção é apresentada na Equação 9 logo abaixo.

$$\boldsymbol{LT_{giro} = Rc + m_a + F} \tag{9}$$

Onde:

LT_{giro} = Largura Total de giro disponível para o veículo de carga.

Rc = Recuo da calçada.

m_a = Mediana da interseção (triângulo retângulo).

F = Flecha.

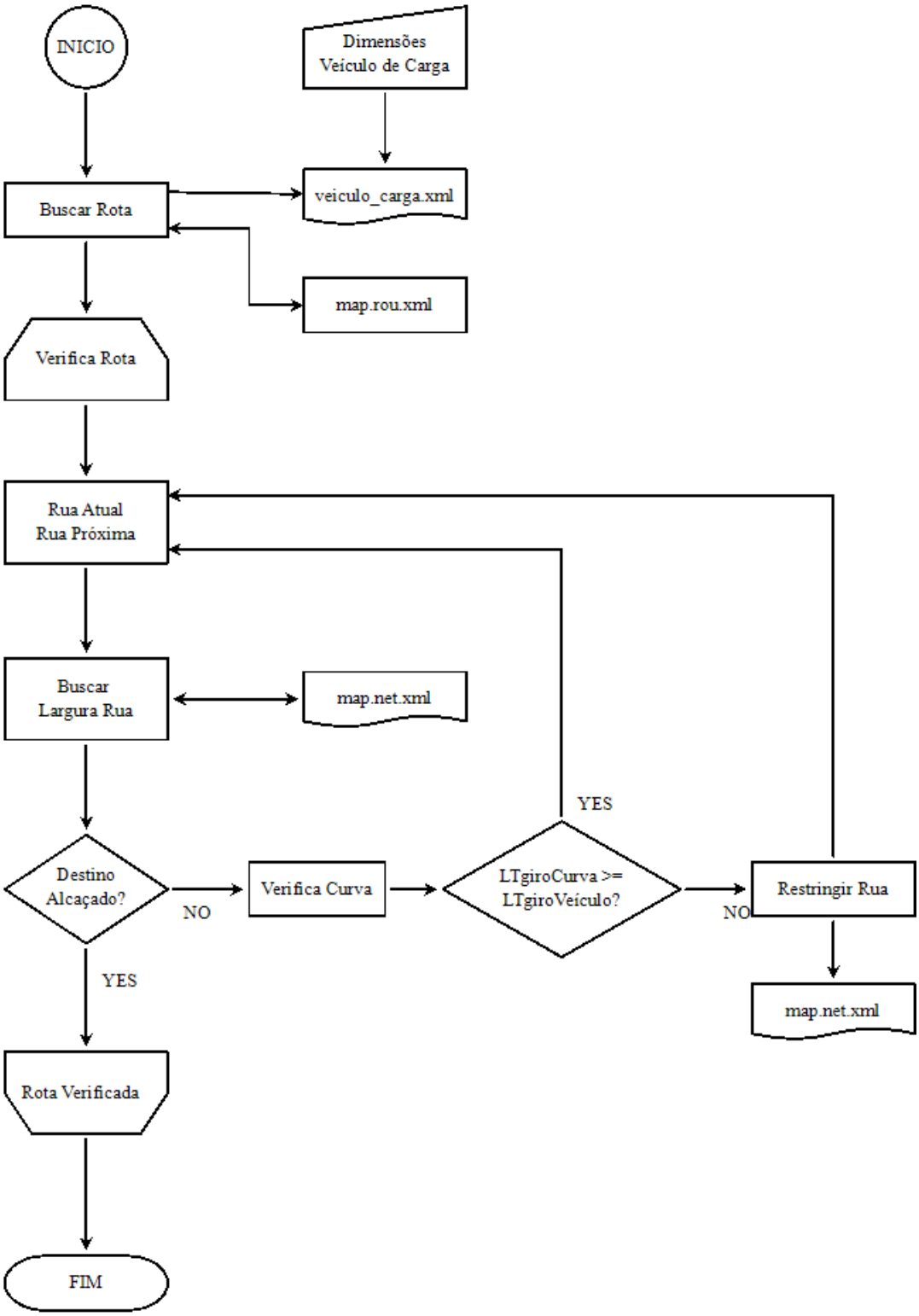
O algoritmo verifica se a largura necessária para o giro do veículo de carga é atendida pela largura disponível na interseção, se sim, ele permite que o algoritmo de roteamento utilize determinada rua, caso contrário, o algoritmo de restrição de giro bloqueia o tráfego de veículos de carga nos trechos onde o giro é restrito. Na próxima execução do DUAROUTER o algoritmo de roteamento descarta este caminho e procura pelo próximo caminho alternativo e sem restrição gerando uma nova rota.

Então é executado novamente o algoritmo de restrição de giro, e assim sucessivamente até que encontre um caminho sem restrições até o destino.

O resultado do algoritmo de roteamento do SUMO retornará as ruas que o veículo de carga poderá trafegar, sem maiores problemas.

A Figura 11 a seguir mostra o fluxograma de execução do algoritmo de restrição de giro.

Figura 11- Fluxograma do algoritmo de restrição de giro.



Fonte: Elaborado pelo autor.

Na Figura 12 mostra um pseudocódigo do método verificaRota, que recebe o ângulo entre a rual atual em que o veículo de carga se encontra em determinado momento e a próxima rua da rota.

O método “FazCurva” na linha 15, é responsável em realizar os cálculos para avaliar a compatibilidade entre as dimnesões do veículo de carga com a largura disponível para giro da interseção. Não será descrito em pseudocódigo o método FazCurva, pois sua lógica já foi detalhado no capítulo 4.

Figura 12 – Pseudocódigo do algoritmo de Restrição de Giro

```
1  verificaRota(anguloDaCurva, recuoDaCalçada)
2  {
3      Veiculo v;
4
5      V.raioGiroVeiculo ;
6
7      origem, destino <- ler do arquivo de rotas;
8
9      vetorDeRota <- ler do arquivo de rota;
10
11     PARA cada passo do vetor de rotas FAÇA
12
13         largRuaAtua, largRuaProx <- ler do arquivo de mapa;
14
15         FazCurva(largRuaAtual, largRuaProx, angDaRua,
16                  v.raioMax, v.raioMin, v.largGiroMin)
17
18         SE não faz curva
19             Restringir rua arquivo mapa;
20
21     FIM_PARA
22 }
23
```

Fonte: Elaborado pelo autor.

5. APLICAÇÃO DO ALGORITMO DE RESTRIÇÃO DE GIRO

Neste capítulo é apresentado o resultado obtido dentro de um cenário que simula um veículo de projeto SR (Semirreboque), utilizando o mapa OpenStreetMap e o simulador SUMO.

Para a simulação, foi definido o Sistema Operacional Ubuntu 14.04 e a versão 0.27.1 do SUMO. Toda implementação do algoritmo de restrição de giro foi desenvolvida na linguagem C++.

5.1. Veículo de projeto utilizado

O veículo utilizado na simulação segue a especificação utilizada na documentação elaborada pela AASHTO (2001, pág 32), devido ao conteúdo de informações que satisfazem as variáveis utilizadas no cálculo de giro, onde se encontra principalmente o ângulo de esterção das rodas dianteiras.

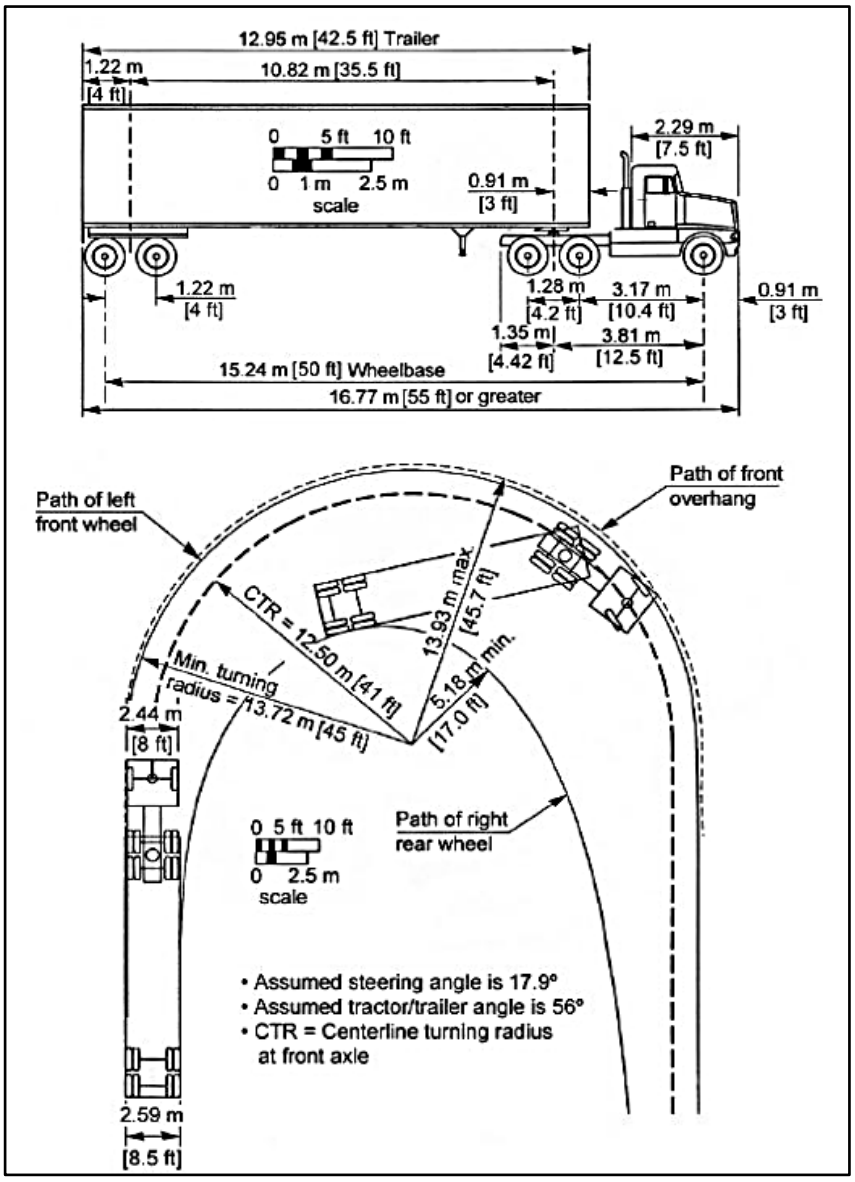
Portanto as dimensões do veículo que serão úteis para o cálculo de giro são descritas na Tabela 3, e posteriormente na Figura 13 contém as dimensões do veículo utilizado.

Tabela 3- Dimensões do veículo de carga Semirreboque.

VARIÁVEIS	DESCRIÇÃO	DIMENSÕES (m)	GRAU
L1	Largura do veículo	2,44	*
L2	Largura do reboque.	2,59	*
BT	Balanço traseiro.	0.61	*
BD	Balanço dianteiro.	0,91	*
E1	Distância entre eixos veículo.	1,28	*
E2	Distância eixo reboque / pino rei.	10,82	*
P	Distância eixo até o pino rei veículo	0,64	*
a	Ângulo de giro médio das rodas	*	56

Fonte: Elaborado pelo autor.

Figura 13- Dimensões do veículo da simulação.



Fonte: *American Association of State Highway and Transportation Officials* (AASHTO, 2001, pág 32).

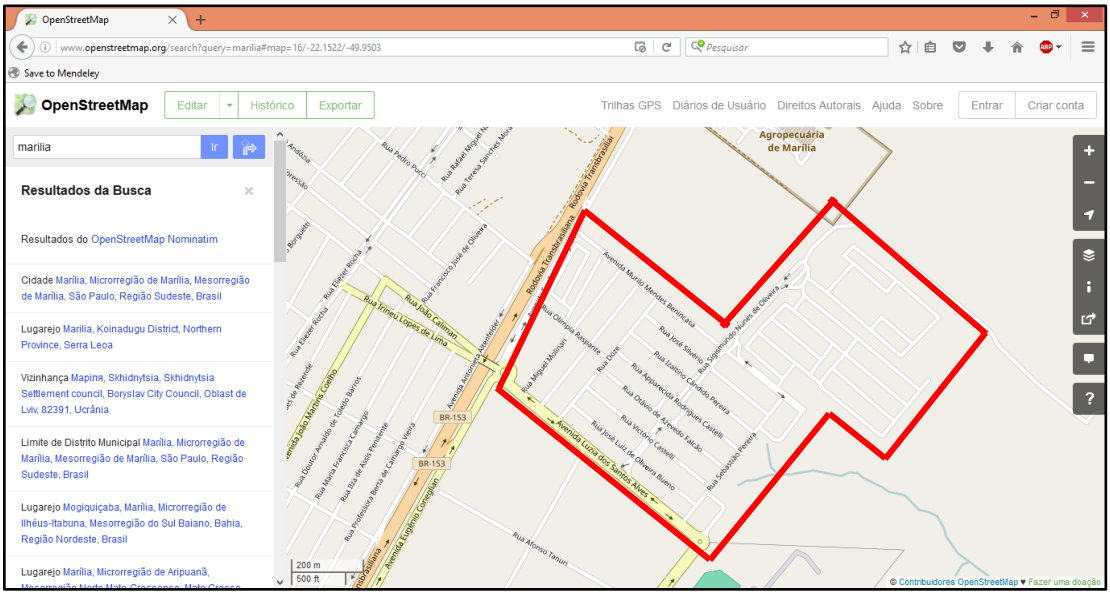
Na seção 5.2 será demonstrada a aplicação do algoritmo de restrição de giro utilizando as dimensões do veículo descrito neste capítulo.

5.2. Simulação

Primeiramente foi definido o mapa da região que será utilizada na simulação, sendo o bairro Jardim Nazareth localizado no município de Marília, Estado de São Paulo. Esta região facilitou a coleta das larguras de ruas do bairro devido ao baixo tráfego de veículos, as medições foram colhidas em campo utilizando uma trena.

Posteriormente, os dados coletados foram inseridos no mapa OSM, podendo ser visualizado abrindo o arquivo de download do mapa que um arquivo .xml, onde se encontra a tag *width* com a respectiva largura da rua. A Figura 14 apresenta o mapa da região retirado do mapa OpenStreetMap e o bairro em destaque.

Figura 14- Mapa com destaque do bairro Jd Nazareth – Marília/SP



Fonte: www.openstreetmap.org

A seguir na Figura 15 mostra o arquivo xml da região do mapa escolhido com a inclusão da largura de rua.

Figura 15- Rua com a largura inserida.

```
<way id="129809787" visible="true" version="3"
   changeset="41788010" timestamp="2016-08-29T20:52:55Z"
   user="Anderson bcc" uid="3749943">
  <nd ref="1431447092"/>
  <nd ref="1431447015"/>
  <nd ref="1431447002"/>
  <nd ref="1431447126"/>
  <nd ref="1431446992"/>
  <tag k="highway" v="residential"/>
  <tag k="name" v="Rua Sigismundo Nunes de Oliveira"/>
  <tag k="oneway" v="yes"/>
  <tag k="width" v="9.1"/>
</way>
```

Fonte: www.openstreetmap.org. Imagem elaborada pelo autor.

O SUMO não trata como relevante largura de ruas, apesar de conter atributo para isso na classe *MSLane.h*, esta informação não influencia para os tipos de simulações a que se propõe o simulador. Entretanto, a informação de largura da rua é essencial para os testes do algoritmo de Restrição de Giro.

Todavia, é possível definir um valor de largura de rua que seja aplicado a todo o mapa no momento da conversão do arquivo OSM para o SUMO.

Para definir a medida de largura de rua foi utilizado o seguinte comando no SUMO:

```
$ Netconvert --osm-files map.osm --output.street-names true --default.lanewidth 9.0 -
    o map.net.xml
```

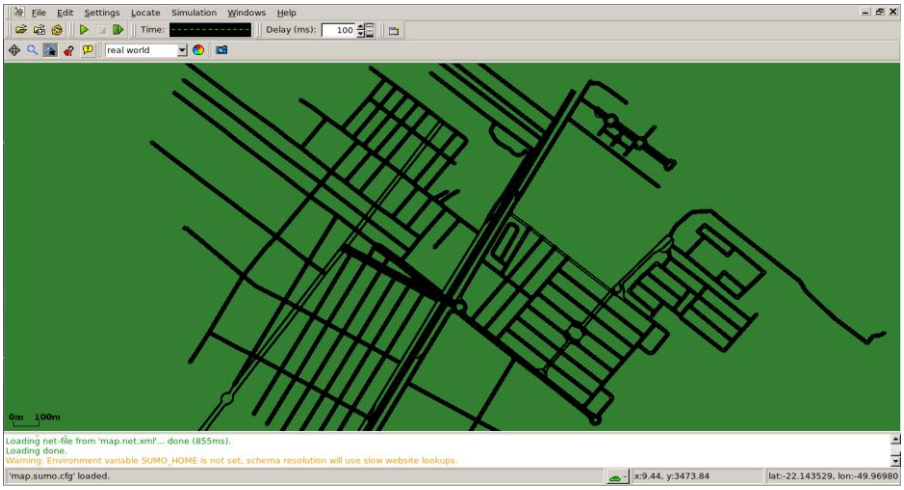
Onde a opção *--default.lanewidth 9.0* define a largura padrão para todas as ruas no momento da importação do mapa.

Após a conversão, foi necessário alterar manualmente a largura da rua de acordo com o mapa do OSM da Avenida Murilo Mendes Benincasa para 6 metros.

O arquivo alterado foi o *map.net.xml* que é criado após a execução do NETCONVERT.

A Figura 16 mostra o mapa OSM convertido para o SUMO visualizado pela sua interface visual SUMO-GUI.

Figura 16- Mapa OSM importado pelo SUMO.



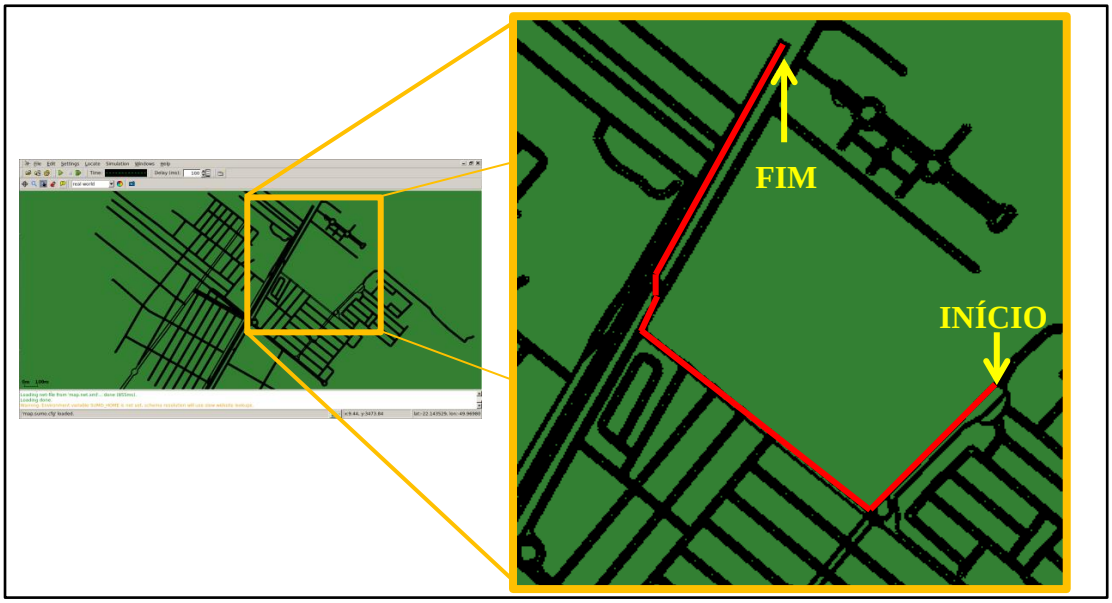
Fonte: Imagem elaborada pelo autor.

Tendo realizado as configurações básicas necessárias, foi criado um script de nome *trips.trips.xml* com a informação da rua de origem e destino. A partir deste arquivo foi utilizado o módulo DUAROUTER para definição da rota do veículo de carga, por padrão é utilizado o algoritmo de Dijkstra.

Executando o módulo DUAROUTER é criado em outro arquivo .xml denominado *map.rou.xml*, onde é descrita as identificações das ruas pelas quais o veículo irá percorrer.

A primeira rota definida pelo módulo DUAROUTER padrão é apresentada na Figura 17, o percurso é destacado em vermelho.

Figura 17- Rota algoritmo de Dijkstra do SUMO



Fonte: Elaborado pelo autor.

É importante ressaltar que tanto o mapa OSM quanto a mapa convertido pelo SUMO, uma rua pode ser segmentada em várias partes, portanto uma rua pode ser descrita por vários identificadores que juntos formam a extensão total da rua.

O arquivo gerado pelo DUAROUTER chamado de *map.rou.xml* contém todos os identificadores por onde o veículo deverá passar para chegar até seu destino.

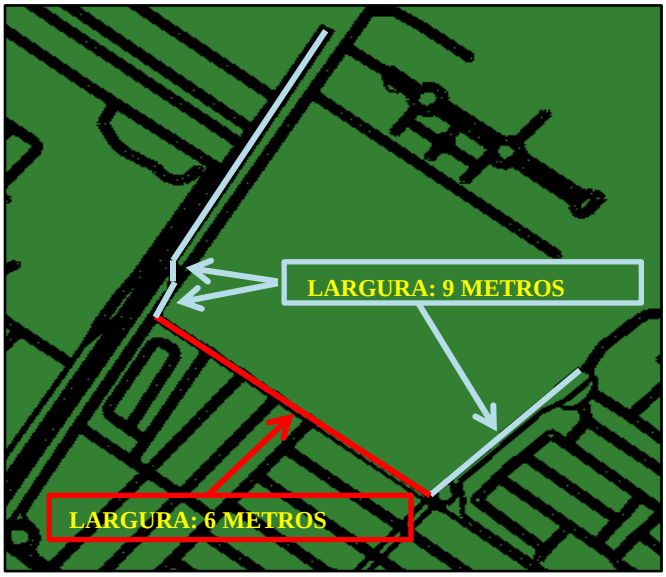
Para que o mapa possa ser visualizado pelo SUMO-GUI, a interface visual do SUMO, é necessário criar o arquivo *map.sumo.cfg*, um arquivo xml que contém o nome do arquivo de informações da rede SUMO, arquivo de configuração e rota.

Para gerar a simulação é utilizado o comando:

```
$ sumo-gui map.sumo.cfg
```

Esta rota é o percurso mais curto entre a rua de origem e destino. Entretanto nesta rota se encontra a “Avenida Murilo Mendes Benincasa” destacada em vermelho que possui uma largura de 6 metros, a Figura 18 mostra a largura de cada rua.

Figura 18- Largura das ruas.

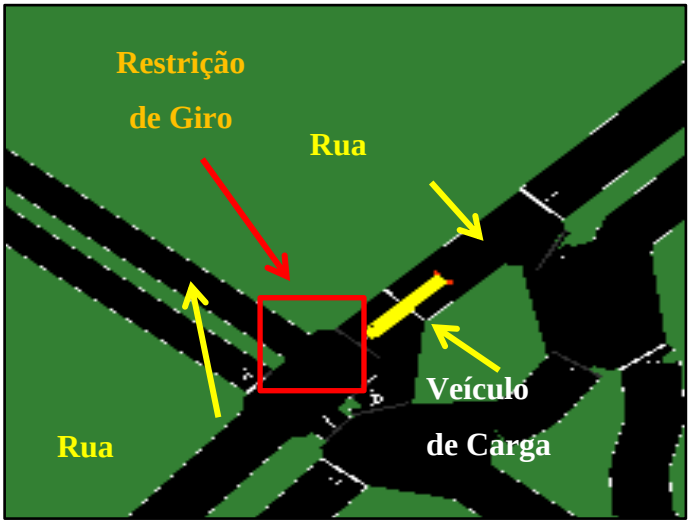


Fonte: Elaborado pelo autor.

O algoritmo considera que todo identificador de rua que seja diferente é uma interseção de 90°, pois atualmente o algoritmo não identifica o ângulo entre as ruas.

As informações da largura das ruas *Atual* e *Seguinte* representado na Figura 19, são obtidas diretamente do arquivo *map.net.xml*, realiza os cálculos e verifica se o veículo de carga pode ou não realizar a curva.

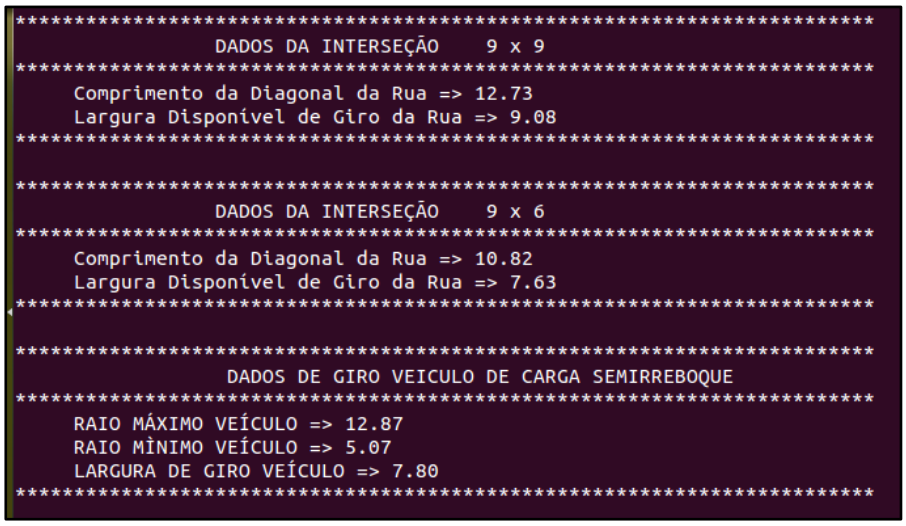
Figura 19- Rua atual e Rua Seguinte



Fonte: Elaborado pelo autor.

Na Figura 20 apresenta a tela com as informações da interseção e do veículo de carga utilizado na simulação.

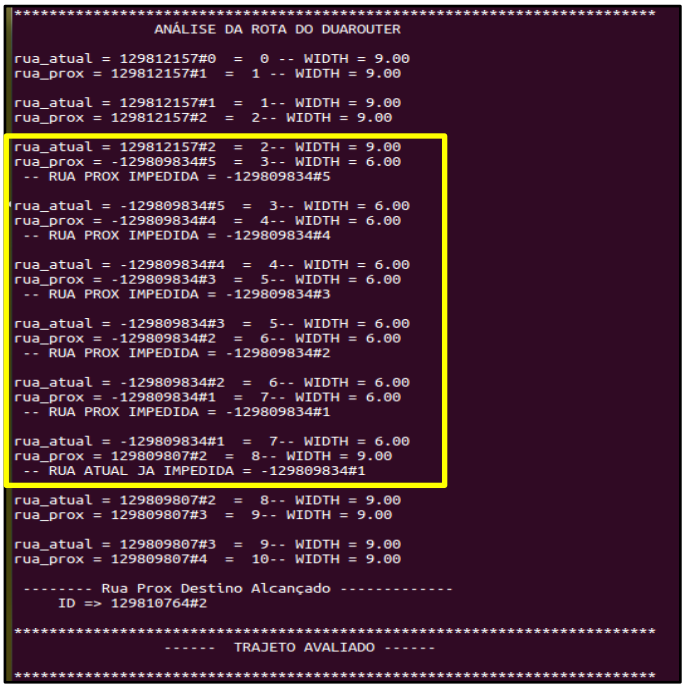
Figura 20- Resultado obtido pelo algoritmo de Restrição de giro.



Fonte: Elaborado pelo autor.

Ao executar o algoritmo de restrição de giro, ele detecta que existe uma rua em que o veículo de carga não consegue realizar a curva, fazendo com que a rua seja restringida para tráfego de caminhões no arquivo *map.net.xml*, todas os identificadores que tiverem a largura da diagonal incompatível serão impedidas para tráfego, como mostra a Figura 21.

Figura 21- Ruas com restrição de giro.

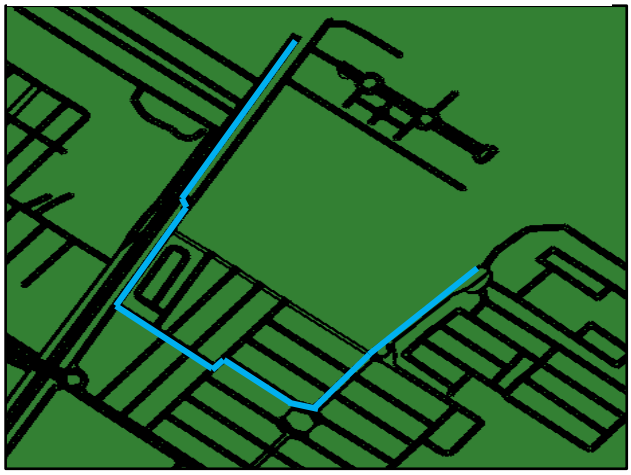


Fonte: Elaborado pelo autor.

Após a conclusão da análise da rota, é executado novamente o DUAROUTER para que seja gerado uma nova rota alternativa.

A nova rota gerada é um percurso maior do que a primeira como é domonstrado na próxima Figura 22.

Figura 22- Rota alternativa.

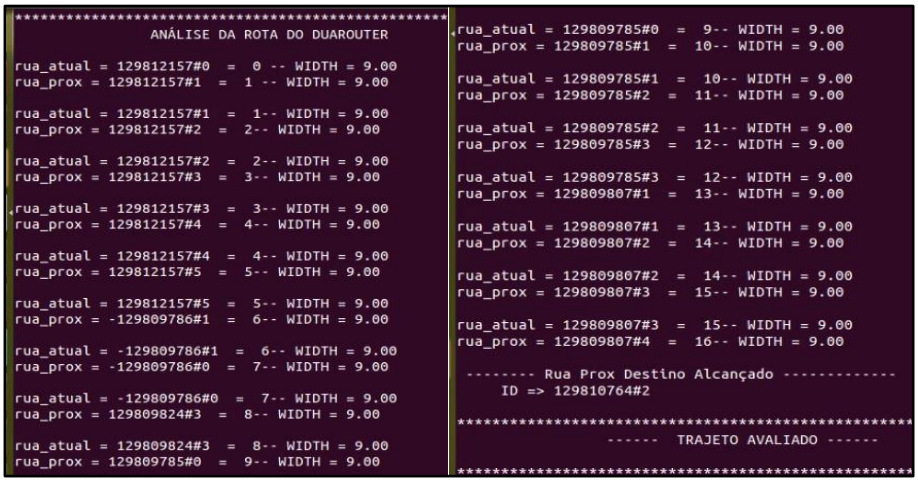


Fonte: Elaborado pelo autor.

Entretanto, uma nova rota não significa que esteja isento de restrição para o veículo em questão, necessitando novamente ser executado o algoritmo de restrição de giro para que a nova rota seja avaliada.

Após a execução do algoritmo, a avaliação é positiva, pois não se encontrou nenhum trajeto que possua restrição de giro, como mostra a Figura 23.

Figura 23- Análise do algoritmo de restrição de giro.



Fonte: Elaborado pelo autor.

Esta nova rota permite que o veículo de carga possa trafegar sem maiores problemas por ruas de acesso mais facilitado de acordo com suas dimensões.

6. CONCLUSÃO E TRABALHOS FUTUROS

O algoritmo de restrição de giro apresentado é uma proposta de abordagem para uma possível solução de um problema que é do cotidiano dos condutores de veículos de carga.

A largura de rua no SUMO foi definida como padrão de 9 metros e alterado manualmente as ruas com larguras inferiores.

Este procedimento foi necessário para a simulação no SUMO, entretanto, todo esforço da inclusão da largura de ruas no mapa OSM não é em vão. Pois se os dados de largura estiverem no mapa, eles podem ser usados futuramente em uma atualização no simulador SUMO.

A solução proposta não resolve situações mais complexas como, rotatórias, interseções de diferentes ângulos de giro, ou seja, ruas com ângulos inferiores ou superiores a 90°, problemas em relação a obstáculos próximos das interseções, como obras de manutenção da pista ou carros estacionados em condições irregulares de acordo com a legislação de trânsito brasileira.

A princípio em ambiente controlado, apresenta uma possibilidade de solução viável podendo ser aprimorada e posteriormente realização de testes em campo para validar a sua eficiência na prática.

Na simulação, a transição entre IDs diferentes são consideradas como uma interseção de 90º, pelo fato de que o algoritmo neste trabalho não verifica ângulos entre as ruas. Esta restrição não impede o desenvolvimento da simulação pelo fato de restringir somente ruas de largura insuficiente para giro do veículo.

Havendo restrições de acordo com as dimensões do veículo, é possível afirmar de que nem sempre a rota de menor custo será adequada para o veículo de carga, como foi apresentado no capítulo 5.

Para trabalhos futuros é sugerido o aprimoramento do algoritmo para avaliar interseções com ângulos diferentes de 90° e rotatórias.

Outra possibilidade também é de facilitar o acesso aos dados de largura de ruas, pois atualmente os mapas não dispõem destes dados. Podendo ser utilizado algoritmos de análises de imagens para detecção da largura de ruas e posteriormente serem adicionados aos arquivos do mapa.

Um estudo interessante também é incluir a detecção de altura, onde se pode analisar a altura disponível por trajetos sob viadutos, passarelas de pedestres e túneis.

Com o advento da Internet das coisas (IoT) integrando veículos que formam uma rede veicular denominada *Vehicular Adhoc NETworks* (VANET), onde várias informações podem ser trocadas entre os veículos, há também a possibilidade de aplicação do algoritmo aprimorado para veículos de bombeiros em situações de emergência auxiliando no traçado de rotas mais eficazes para chegar no endereço da ocorrência.

REFERÊNCIAS

AASHTO. **A Policy on Geometric Design of Highways and Streets**. Disponível em: <http://nacto.org/docs/usdg/geometric_design_highways_and_streets_aashto.pdf>. Acesso em: 16 jun. 2016.

API do Google Maps: desenvolvimento e análise de aplicativos. Disponível em: <https://www.google.com.br/intx/pt-BR/work/mapsearth/pricing/#compare-editions>. Acesso em: 5 ago. 2016.

CARNIELLE, L. G. **Proposta de um Sistema de Análise da Logística Urbana para Cidades de Pequeno e Médio Porte**. 2008. 97p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2008.

CNT. **Boletim Estatístico - 01 - 2016**. Disponível em: <http://www.cnt.org.br/Boletim/boletim-estatistico-cnt>. Acesso em: 30 maio. 2016.

CONTRAN. **Resolução nº 210**. Disponível em: <http://www.denatran.gov.br/consolidadas.htm>. Acesso em: 24 jun. 2016.

CONTRIBUTORS, O. W. **Databases and data access APIs**. Disponível em: <http://wiki.openstreetmap.org/w/index.php?title=Databases_and_data_access_APIs&oldid=1367952>. Acesso em: 18 nov. 2016.

CONTRIBUTORS, O. W. **Pt-br:Portal:Press**. Disponível em: <http://wiki.openstreetmap.org/w/index.php?title=Pt-br:Portal:Press&oldid=1168633>. Acesso em: 24 jun. 2016a.

CONTRIBUTORS, O. W. **Pt-br:Introduction**. Disponível em: <http://wiki.openstreetmap.org/w/index.php?title=Pt-br:Introduction&oldid=1169083>. Acesso em: 23 jun. 2016b.

CONTRIBUTORS, O. W. **Contributors**. Disponível em: <http://wiki.openstreetmap.org/w/index.php?title=Contributors&oldid=1316057>. Acesso em: 23 jun. 2016.

DEINFRA - DEPARTAMENTO ESTADUAL DE INFRA-ESTRUTURA. **GABARITOS_DE_GIRO_MANUAL**. Disponível em: <http://www.deinfra.sc.gov.br/jsp/relatorios_documentos/doc_tecnico/download/engenharia_r

odoviaria/GABARITOS_DE_GIRO_MANUAL.pdf>.

DNIT. **MANUAL DE PROJETO DE INTERSEÇÕES_Versão Final.doc - 718_manual_de_projeto_de_intersecoes.pdf**. Disponível em: <http://ipr.dnit.gov.br/normas-e-manuais/manuais/documentos/718_manual_de_projeto_de_intersecoes.pdf>. Acesso em: 18 maio. 2016.

DNIT. **Quadro de Fabricantes de Veículos**. Disponível em: <http://www.dnit.gov.br/download/rodovias/operacoes-rodoviarias/pesagem/qfv-2012-abril.pdf>. Acesso em: 18 maio. 2016.

FENABRAVE. **O desempenho da distribuição automotiva no Brasil**. Disponível em: <http://www3.fenabrave.org.br:8082/plus/modulos/listas/index.php?tac=indices-e-numeros&idtipo=6&layout=indices-e-numeros>. Acesso em: 21 mar. 2016.

GEISBERGER, ROBERT; SANDERS, PETER; SCHULTES, DOMINIK;DELLING, D. **Contraction Hierachies: Faster and Simpler Hierachical Routing in Road Networks**Karlsruhe, Alemanha. **Anais**...2008Disponível em: <http://algo2.iti.kit.edu/schultes/hwy/contract.pdf>. Acesso em: 25 jun. 2016

GLOBALWEBINDEX. **Top global smartphone apps, who's in the top 10**. Disponível em: <http://www.globalwebindex.net/blog/top-global-smartphone-apps>. Acesso em: 5 ago. 2016.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. **IEEE Transactions on Systems Science and Cybernetics**, v. 4, n. 2, p. 100–107, 1968.

INSTITUTE, T. S. **DLR**. Disponível em: <http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/>. Acesso em: 19 maio. 2016.

IPR. **Instituto de Pesquisas Rodoviárias – IPR**. Disponível em: <http://ipr.dnit.gov.br/ipr/servicos>. Acesso em: 19 jun. 2016.

KRAJZEWICZ, DANIEL; ERDMANN, JAKOB; BEHRISCH, MICHAEL; BIEKER, L. **Demand/Routing by Turn Probabilities**. Disponível em: <http://sumo.dlr.de/w/index.php?title=Demand/Routing_by_Turn_Probabilities&oldid=7191>. Acesso em: 23 jun. 2016.

KRAJZEWICZ, DANIEL; ERDMANN, JAKOB; BEHRISCH, MICHAEL; BIEKER, L. **Demand/Routes from Observation Points**. Disponível em: <http://sumo.dlr.de/w/index.php?title=Demand/Routes_from_Observation_Points&oldid=6844>. Acesso em: 24 jun. 2016a.

KRAJZEWICZ, DANIEL; ERDMANN, JAKOB; BEHRISCH, MICHAEL; BIEKER, L. **Demand/Importing_O/D_Matrices - SUMO - Simulation of Urban Mobility**. Disponível em: <http://www.sumo.dlr.de/userdoc/Demand/Importing_O/D_Matrices.html>. Acesso em: 8 dez. 2016b.

KURCZVEIL, T.; LÓPEZ, P. Á. eNetEditor: Rapid prototyping urban traffic scenarios for SUMO and evaluating their energy consumption. **Proceedings of the SUMO User Conference - Intermodal Simulation for Intermodal Transport**, n. MAY 2015, p. 146–169, 2015.

MACEDO, D. M. R. **Resgatando alguns teoremas clássicos da geometria plana**. 2014. 57p. Dissertação (Mestrado). Universidade Federal do Ceará, Programa de Pós-graduação em Matemática em Rede Nacional, Juazeiro do Norte, 2014.

MACHADO, R. J. F. **Adaptação de mapas geográficos e planeamento de rotas na simulação de tráfego de veículos em ambiente citadino**. 2009. 57p. Tese(Mestrado). Universidade de Coimbra, Departamento de Engenharia Electrotécnica e de Computadores, Coimbra, Portugal. 2009.

PAIM, R. R. **Estudo de algoritmos para redes de transporte**. 2015. 53p. Projeto de Graduação. UFRJ - Escola Politécnica. Rio de Janeiro. 2015.

PEREIRA NETO, W. A. **Análise de fatores intervenientes nas características dimensionais de segmentos rodoviários sob a óptica da compatibilidade veículo–via**. 2007. 190p. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo. 2007.

PONTES FILHO, G. **Estradas de Rodagem - Projeto Geométrico**. São Carlos: G. Pontes Filho. 1998. 432p.

RUSSEL, STUART; NORVIG, P. **Inteligência Artificial**. 3ª edição. Rio de Janeiro: Editora Campus, 2004. 1021p.

Projeto TrackSource. Disponível em: <http://tracksource.org.br/sobre-o-projeto/>. Acesso em: 4 ago. 2016.

SILVA, G. **Modelagem e Implementação de uma ferramenta inteligente de código aberto para inserção automática de inferência *fuzzy* em SIG convencionais**. 2006. 152p. Dissetação (mestrado) - Centro Federal de Educação Tecnológica de Minas Gerais. Belo Horizonte. 2006.

SOUZA, S. C. B. DE. **Planejamento de trajetória para um robô móvel com duas rodas utilizando um algoritmo a-estrela modificado**. 2008. 97p. Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia Elétrica, Rio de Janeiro. 2008.

Termos de Utilização do Google – Privacidade e Termos de Utilização – Google. Disponível em: <https://www.google.com/intl/pt-PT/policies/terms/index.html>. Acesso em: 5 ago. 2016.

APÊNDICE

Neste apêndice é apresentado o código fonte do algoritmo de restrição de giro. A linguagem de programação utilizada foi o C++ que utiliza o paradigma Programação Orientado a Objetos (POO).

Na Figura A.1 mostra o código fonte da classe VeiculoCarga.cpp, onde são implementados os cálculos para obtenção dos giros máximos e mínimos para um veículo de carga Semirreboque (SR).

Figura A. 1- Implementação da classe VeiculoCarga.

```
#include "VeiculoCarga.h"
#include<cmath>
#include<tgmath.h>

#define PI 3.14159265

VeiculoCarga::VeiculoCarga()
{
    //ctor
}

VeiculoCarga::~VeiculoCarga()
{
    //dtor
}

//Método que calcula os Raios Mínimo e Máximo do Veículode
// Carga Semirreboque
void VeiculoCarga::minimumRadius(int l1, int l2, int bt, int bd,
                                 int e1, int e2, int p, float a)
{
    //Calcula o raio mínimo do giro
    rm = (e2/(tan(a * PI / 180)))-(l2/2);

    //Calcula o Raio Externo do giro
    re = sqrt(pow(sqrt(pow(e2/tan(a * PI / 180),2) + pow(e2,2)) + (l1/2),2) + pow(e1+bd,2));

    //Calcula a largura total para o giro
    f = re - rm;
}
//Retorna o Raio Máximo do veículo
float VeiculoCarga::getRaioMaximo(){
    return re;
}

//Retorna o Raio Mínimo do veículo
float VeiculoCarga::getRaioMinimo(){
    return rm;
}

//Retorna a Largura disponível para Giro
float VeiculoCarga::getLarguraGiro(){
    return f;
}
```

Fonte: Elaborado pelo autor.

A seguir na Figura A.2.a mostra o código fonte da classe Curva.cpp, que é responsável de verificar se o veículo de carga semirreboque pode ou não realizar a conversão em determinada interseção.

Figura A.2.a - Implementação da classe Curva.

```
#include "Curva.h"
#include <iostream>
#include <cmath>
#include <tgmath.h>
#include <stdlib.h>
#include <stdio.h>

#define PI 3.14159265
using namespace std;

Curva::Curva()
{
    //ctor
}

Curva::~Curva()
{
    //dtor
}

//Verifica se a curva possui largura suficiente para conversão do caminhão
bool Curva::getFazCurva(float ruaLargAtual, float ruaLargProx, float angRuas, float raioMaxVeic,
                        float recuo_calcada, float lgiroveiculo)
{
        //Verifica se as duas ruas eh uma esquina
        if(angRuas == 90)
        {
            lgirorua = larguraGiroRua(ruaLargAtual, ruaLargProx, recuo_calcada, raioMaxVeic);

            //Compara se é possível realizar a curva.
            if((lgirorua) >= (lgiroveiculo))
            {
                bool fazCurva=true;
                return fazCurva;
            }

        else
        {
            bool fazCurva=false;
            return fazCurva;
        }
    }

    return 0;
}

/********* MÉTODOS DE CÁLCULO DE GIRO DA CURVA *******/

//Retorna a largura disponível para giro
float Curva::larguraGiroRua(float ruaLargAtual, float ruaLargProx, float recuo_calcada, float rveic)
{
    hip = hipRua(ruaLargAtual, ruaLargProx);

    medrua = medianaStewart(ruaLargAtual, ruaLargProx, hip);

    flecha = Flecha(ruaLargAtual, ruaLargProx, rveic);

    float lg =  recuo_calcada + medrua + flecha;

    return lg;
}

// Calcula a diagonal da rua
float Curva::hipRua(float largAtual, float largProx )
{
    // Calcula o comprimento da diagonal da interseção
    float h = sqrt(pow(largAtual,2) + pow(largProx,2));

    return h;
}

// Determina o para   para determinar o comprimento da flecha
float Curva::Flecha(float largAtual, float largProx, float raioMaxVeic)
{
    //Determina o comprimento da diagonal da rua
    hip = hipRua(largAtual,largProx);

    //Determina o comprimento da mediana do triângulo da rua
    med_veiculo = medianaStewart(raioMaxVeic, raioMaxVeic, this->hip);

    //Determinao ângulo do arco central (AC)
    AC = arcoCentral(raioMaxVeic, hip);

    //Determina o comprimento da tangente (T)
    T = compTangente(raioMaxVeic, AC);

    //Determina o afastamento do arco (E)
    E = afastArco(T,AC);

    //Determina o comprimento da flecha
    float f = compFlecha(raioMaxVeic, E, med_veiculo);

    return f;
}

float Curva::medianaStewart(float b, float c, float a)
{
    // Fórmula da mediana de Stewart
    float md = 0.5 * sqrt((2*(pow(b,2)+pow(c,2))) - pow(a,2));

    return md;
}
```

Fonte: Elaborado pelo autor.

Figura A.2.b- Continuação da implementação da classe Curva.

```
103
104  float Curva::arcoCentral(float rVeic, float corda)
105  {
106      // Usa-se meia corda para que seja um triângulo retângulo
107      float meiacorda = corda/2;
108      // Calcula o seno em radianos
109      float xrad = meiacorda/rVeic;
110      //Retorna o arcoseno em graus
111      float seno = asin(xrad)* 180/PI;
112
113      float ac = seno * 2;
114
115      //Retorna o dobro pois são dois triângulo para definir o Ângulo Central
116      return ac;
117  }
118
119  // Retorna o comprimento da tangente
120  float Curva::compTangente(float raio, float AC)
121  {
122      float comp_tang = raio * tan(((AC*PI/180)/2));
123
124      return comp_tang;
125  }
126
127  //Calcula o afastamnto do arco
128  float Curva::afastArco(float t, float ac)
129  {
130      float e = t * tan(((ac*PI/180)/4));
131
132      return e;
133  }
134
135  //Determina o comprimento da flecha
136  float Curva::compFlecha(float R, float &E, float &H)
137  {
138      float fl = ( R + E ) - ( H + E );
139
140      return fl;
141  }
142
143  /***************** MÉTODOS GET *****************/
144
145 float Curva::getDiagonalCurva(){
146      return this->hip;
147  }
148
149  float Curva::getLarguraGiroRua()
150  {
151      return this->lgirorua;
152  }
153
154  float Curva::getMedRua()
155  {
156      return medrua;
157  }
158
159  /***************** MÉTODOS PRINT *****************/
160
161  void Curva::printDiagRua()
162  {
163      cout << fixed;    // limita em
164      cout.precision(2);// 2 casa decimais
165      cout << "\n     Comprimento da Diagonal da Rua => " << hip;
166  }
167
168  void Curva::printLargGiroRua()
169  {
170      cout << fixed;    // limita em
171      cout.precision(2);// 2 casa decimais
172      cout << "\n     Largura Disponível de Giro da Rua => " << lgirorua;
173  }
174
175  void Curva::printMedRua()
176  {
177      cout << fixed;    // limita em
178      cout.precision(2);// 2 casa decimais
179      cout << "\n     Mediana da Rua => " << medrua;
180  }
181
182  void Curva::printMedVeiculo()
183  {
184      cout << fixed;    // limita em
185      cout.precision(2);// 2 casa decimais
186      cout << "\n     Mediana do Veiculo => " << med_veiculo;
187  }
188
189  void Curva::printArcCentralRua()
190  {
191      cout << fixed;    // limita em
192      cout.precision(2);// 2 casa decimais
193      cout << "\n     Arco Central (AC) da Rua (Referente Raio do Veículo) => " << AC;
194  }
```

Fonte: Elaborado pelo autor.

Figura A.2.c- Continuação da implementação da classe Curva.

```
195
196 void Curva::printCompTangRua()
197 {
198     cout << fixed;    // limita em
199     cout.precision(2);// 2 casa decimais
200     cout << "\n     Comprimento da Tangente (T) da Rua (Referente Raio do Veículo) => " << T;
201 }
202
203 void Curva::printCompAfastamentoRua()
204 {
205     cout << fixed;    // limita em
206     cout.precision(2);// 2 casa decimais
207     cout << "\n     Comprimento do afastamento (E) do";
208     cout << "\n     Arco da Rua (Referente Raio do Veículo) => " << E;
209 }
210
211 void Curva::printCompflechaRua()
212 {
213     cout << fixed;    // limita em
214     cout.precision(2);// 2 casa decimais
215     cout << "\n     Comprimento da Flecha (Referente Raio do Veículo) => " << flecha;
216 }
```

Fonte: Elaborado pelo autor.

A Figura A.3 mostra a classe RestriçãoGiro que acessa os arquivos “map.rou.xml” e “map.net.xml” do SUMO

Figura A. 3.a- Classe RestricaoGiro.

```
1  #include "RestricaoGiro.h"
2  #include <iostream>
3  #include <stdlib.h>
4  #include <stdio.h>
5  #include <string.h>
6  #include <string>
7  #include <cstring>
8  #include <cstdio>
9  #include "Curva.h"
10 #include "VeiculoCarga.h"
11 #include </home/anderson/rapidxml-1.13/rapidxml.hpp>
12 #include </home/anderson/rapidxml-1.13/rapidxml_print.hpp>
13 #include </home/anderson/rapidxml-1.13/rapidxml_utils.hpp>
14
15 using namespace std;
16 using namespace rapidxml;
17
18 RestricaoGiro::RestricaoGiro()
19 {
20     //ctor
21 }
22
23 RestricaoGiro::~RestricaoGiro()
24 {
25     //dtor
26 }
27
28 bool RestricaoGiro::getRota(float angRuas, float recuo_calcada)
29     {
30
31         xml_document<> doc;
32         xml_node<> *node_principal;
33         xml_node<> *node_vehicle;
34         char *partida, *chegada;
35         bool fazcurva;
36         Curva d;
37
38         VeiculoCarga *v = new VeiculoCarga();
39
40         v->minimumRadius(v->getL1(),v->getL2(),v->getBt(),v->getBd(),v->getE1(),v->getE2(),v->getP(),v->getA());
41
42         float raioMaxVeic = v->getRaioMaximo();
43         float lgiroveiculo = v->getLarguraGiro();
44
45         cout << "\n*************************************************************************\n";
46         cout << "                  DADOS DE GIRO VEICULO DE CARGA SEMIRREBOQUE ";
47         cout << "\n*************************************************************************";
48         v->printRaioMaximo();
49         v->printRaioMinimo();
50         v->printLarguraGiro();
51         cout << "\n*************************************************************************\n";
52
53         string arquivoROU = "/home/anderson/restricao_giro/map.rou.xml";
54
55         // Carrega o arquivo xml map.rou.xml do SUMO
56         ifstream rou (arquivoROU);
57         vector<char> buffer((istreambuf_iterator<char>(rou)), istreambuf_iterator<char>());
58         buffer.push_back('\0');
```

Fonte: Elaborado pelo autor.

Figura A.3.b- Continuação Classe RestricaoGiro.

```
59
60          // Analisa o arquivo xml  no buffer com a biblioteca
61          doc.parse<0>(&buffer[0]);
62
63          // Encontra o nó raiz
64          node_principal = doc.first_node("routes");
65          node_vehicle = node_principal->first_node("vehicle");
66
67          //Armazena em routes as ids das edges da rota
68          string routes = node_vehicle->first_node("route")->first_attribute("edges")->value();
69
70          //Tamanho da String
71          int tam = routes.size();
72          char id[tam];
73          char * pch;
74          char *seqId;
75          int vtam=0;
76          char *rua_atual, *rua_prox;
77          float atual, prox;
78          string larg_rua_prox, larg_rua_atual;
79
80          //Copia String para Vetor char
81          strcpy( id, routes.c_str() );
82          //Lê seq de caracteres até o terminador "espaço"
83          pch = strtok (id," ");
84
85          cout << "\n\n*********************************************************************************\n";
86          cout << "                IDs DO CAMINHO GERADO PELO DUAROUTER\n";
87
88          //Percorre por todo o caminho e mostra a ID da RUA
89          while (pch != NULL)
90          {
91              printf ("\nID = %s --> %d",pch, vtam);
92              pch = strtok (NULL, " ");
93              vtam++;
94          }
95
96          //Reinicia para percorrer a String
97          strcpy( id, routes.c_str());
98          pch = strtok (id," ");
99
100         //Identifica a edge de origem e destino
101         for(int i=0,j=1; i<vtam; i++, j++)
102         {
103             if(i == 0)
104             {
105                 partida = pch;
106                 printf ( "\n\nID Partida %s",partida);
107             }
108
109             pch = strtok(NULL, " " );
110
111             if(j+2 == vtam)
112             {
113                 chegada = strtok(NULL, " " );
114                 printf ( "\nID Chegada %s\n\n",chegada);
115             }
116         }
117
118         //Reinicia para percorrer a String
119         strcpy( id, routes.c_str());
120 pch = strtok (id," ");
121
122         //Parâmetro para a conversão de String to Float
123         std::string::size_type sz;
124
125         cout << "\n\n*********************************************************************************\n";
126         cout << "                ANÁLISE DA ROTA DO DUAROUTER\n";
127
```

Fonte: Elaborado pelo autor

Figura A.3.c- Continuação Classe RestricaoGiro.

```
        //Percorre os IDs identificando a edge Atual e a Próxima
        for(int i=0,j=0; i<=vtam; i++,j++)
        {
            //Identifica a largura da rua de Origem e a Próxima
            if(i==0)
            {
                rua_atual = pch;

                larg_rua_atual = getLargura(rua_atual);

                rua_prox = strtok(NULL," ");

                larg_rua_prox = getLargura(rua_prox);

                atual = std::stof(larg_rua_atual,&sz);
                prox = std::stof(larg_rua_prox,&sz);

                printf("\nrua_atual = %s  =  %d",rua_atual, j);
                cout << " -- WIDTH = " << larg_rua_atual;
                printf("\nrua_prox = %s  =  %d",rua_prox, ++j);
                cout << " -- WIDTH = " << larg_rua_prox << "\n";

                //Verifica se o veiculo realiza a curva
                fazcurva = d.FazCurva(atual, prox, angRuas, raioMaxVeic, recuo_calcada, lgiroveiculo);

                if(fazcurva==false)
                {
                cout << "n----- RUA PROX IMPEDIDA = " << rua_prox << "\n";
                //cout << " -- RUA PROX = " << rua_prox << "\n\n";
                restringirRua(rua_prox);
                }

                i--;
            }//end if

                rua_atual = rua_prox;

                if(rua_atual == chegada)
            {
                printf("\n -------- Rua atual = Destino Alcançado -------------");
                break;
            }

            rua_prox = strtok(NULL," ");

            if((rua_prox==NULL) || (rua_prox==chegada))
            {
            printf("\n -------- Rua Prox Destino Alcançado -------------");
            cout << "\n     ID => " << rua_prox;
            break;
            }

            larg_rua_atual = getLargura(rua_atual);
            larg_rua_prox = getLargura(rua_prox);

            atual = stof(larg_rua_atual,&sz);
            prox = stof(larg_rua_prox,&sz);

            //Verifica se o veiculo realiza a curva
            fazcurva = d.FazCurva(atual, prox, angRuas, raioMaxVeic, recuo_calcada, lgiroveiculo);

            printf("\nrua_atual = %s  =  %d",rua_atual, j);
            cout << "-- WIDTH = " << larg_rua_atual;
            printf("\nrua_prox = %s  =  %d",rua_prox, ++j);
            cout << "-- WIDTH = " << larg_rua_prox << "\n";

            if(fazcurva==false)
            {
                if(larg_rua_atual < larg_rua_prox)
                {
                   cout << " -- RUA ATUAL JA IMPEDIDA = " << rua_atual << "\n";
                    //restringirRua(rua_atual);
                }
                else
                {
```

Fonte: Elaborado pelo autor

Figura A.3.d- Continuação Classe RestricaoGiro.

```
                    cout << " -- RUA PROX IMPEDIDA = " << rua_prox << "\n";
                    restringirRua(rua_prox);
                }

            }

            j--;
            i--;

        }//end for

        // Salvando o arquivo
        std::string xml_as_string;
        rapidxml::print(std::back_inserter(xml_as_string), doc);
        // Save to file
        std::ofstream file_stored(arquivoROU);
        file_stored << doc;
        file_stored.close();
        doc.clear();

        return fazcurva;

    }

string RestricaoGiro::getLargura(string id)
    {
        //Ponteiros e Variáveis
        xml_document<> doc;
        xml_node<> *node_principal;
        xml_node<> *node_edge;
        xml_node<> *node_lane;
        xml_node<> *alter_node;
        string largura;

        string arquivoNET = "/home/anderson/restricao_giro/map.net.xml";

        try
        {
            // Carrega o arquivo xml map.net.xml do SUMO
            ifstream mapa (arquivoNET);
            vector<char> buffer((istreambuf_iterator<char>(mapa)), istreambuf_iterator<char>());
            buffer.push_back('\0');

            // Analisa o arquivo xml  no buffer com a biblioteca
            doc.parse<0>(&buffer[0]);

            // Encontra o nó raiz
            node_principal = doc.first_node("net");
            node_lane = doc.first_node("net")->first_node("edge")->first_node("lane");

            // Percorre todo o arquivo xml e retorna o conteudo encontrado
            for (node_edge = node_principal->first_node("edge"); node_edge; node_edge = node_edge->next_sibling(0,0,true))
            {
                // Compara se o id atual é o procurado
                if(node_edge->first_attribute("id")->value() == id)
                {
                        // Altera o atributo de todas as Lanes dentro da Edge.
                        for (node_lane = node_edge->first_node("lane"); node_lane; node_lane = node_lane->next_sibling())
                        {
                            // Altera o conteúdo para "truck" contido em "text"
                            largura = node_lane->first_attribute("width")->value();
                           // printf("\n Lane ID -> %s -- Width: %s \n", node_edge->first_attribute("id")->value(),
                           // node_lane->first_attribute("width")->value());
                        }

                    break;

                }//end if

            }//end for

```

Fonte: Elaborado pelo autor

Figura A.3.e- Continuação Classe RestricaoGiro.

```
            // Fecha o arquivo
            std::string xml_as_string;
            rapidxml::print(std::back_inserter(xml_as_string), doc);
            doc.clear();

        }//fim try

        catch(int e)
        {
            std::cout << "Erro ao editar o arquivo" << e <<endl;
        }

    return largura;

    }//getLargura;

void RestricaoGiro::restringirRua(string id)
    {
        //Ponteiros e Variáveis
        xml_document<> doc;
        xml_node<> *node_principal;
        xml_node<> *node_edge;
        xml_node<> *node_lane;
        xml_node<> *alter_node;
        const string edgeID = id;
        string arquivoNET = "/home/anderson/restricao_giro/map.net.xml";

        try
        {
            // Carrega o arquivo xml
            ifstream mapa (arquivoNET);
            vector<char> buffer((istreambuf_iterator<char>(mapa)), istreambuf_iterator<char>());
            buffer.push_back('\0');

            // Analisa o arquivo xml  no buffer com a biblioteca
            doc.parse<0>(&buffer[0]);

            // Encontra o nó raiz
            node_principal = doc.first_node("net");
            node_lane = doc.first_node("net")->first_node("edge")->first_node("lane");

            // Percorre todo o arquivo xml e retorna o conteudo encontrado
            for (node_edge = node_principal->first_node("edge"); node_edge; node_edge = node_edge->next_sibling(0,0,true))
            {
                // Compara se o id atual é o procurado
                if(node_edge->first_attribute("id")->value() == edgeID)
                {
                    // Aloca a String "truck" para o atributo disallow - Restrição para caminhão -
                    std::string truck = "truck";
                    const char * restricao_veiculo = doc.allocate_string(truck.c_str(), strlen(truck.c_str()));

                        // Altera o atributo de todas as Lanes dentro da Edge.
                        for (node_lane = node_edge->first_node("lane"); node_lane; node_lane = node_lane->next_sibling(0,0,true))
                        {
                            // Altera o conteúdo para "truck" contido em "text"
                            node_lane->first_attribute("disallow")->value(restricao_veiculo);
                            //printf("\nDisallow: %s\n", node_lane->first_attribute("disallow")->value());
                        }

                    break;

                }//end if

            }//end for

            // Salvando o arquivo
            std::string xml_as_string;
            rapidxml::print(std::back_inserter(xml_as_string), doc);
            // Save to file
            std::ofstream file_stored(arquivoNET);
            file_stored << doc;
            file_stored.close();
            doc.clear();

            //cout << "\n ID Rua Restringida \n" << edgeID;
            //printf("\n\nRua restringida %s\n\n",edgeID);

        }//fim try

        catch(int e)
        {
            std::cout << "Erro ao editar o arquivo" << e <<endl;
        }

    }//end restringirRua
```

Fonte: Elaborado pelo autor