Suserano: sistema de controle de tráfego urbano

João Batista Sausen[1] | Leonardo Ribeiro Machado[2]

Resumo

Este artigo apresenta o projeto e desenvolvimento de um sistema para controle de tráfego em meios urbanos, usando câmeras e controladores de semáforos. Cita os problemas enfrentados pelos usuários de vias urbanas e os efeitos causados pelos congestionamentos. Apresenta as ferramentas e tecnologias utilizadas e demonstra, em detalhes, os passos percorridos na construção da ferramenta, a qual tenta solucionar alguns dos problemas enfrentados atualmente e também os benefícios de se utilizar um controlador de tráfego.

Palavras-chave: Tráfego. Visão computacional. *OpenCV*. Simulador.

Abstract

This paper presents the design and development of a urban areas traffic control system, using cameras and traffic light controllers. It demonstrates the problems faced by urban road users and the effects caused by traffic jam. It also presents the tools and technologies used and demonstrates, in details, the steps took for building the system, which tries to solve some of the problems currently faced, and also the benefits of using a traffic controller.

Keywords: *Traffic. Computer vision. OpenCV. Simulator.*

1 Introdução

Nos dias de hoje, com a crescente quantidade de carros nas ruas, o trânsito começou a tornar-se um problema para o cidadão. A perda de tempo em engarrafamentos é crescente. Mesmo dentro da própria cidade, as viagens da casa ao trabalho ou em horários de pico tomam bastante tempo.

As alternativas para o trânsito como ônibus em vias próprias, por exemplo, amenizam o problema, porém nem sempre é possível que um usuário utilize o transporte coletivo. Do mesmo modo, nem sempre uma cidade possui vias próprias para esse tipo de transporte.

Alguns sistemas atuais procuram gerenciar o trânsito com o controle dos semáforos. Em cidades como Porto Alegre, por exemplo, existem centrais de trânsito com câmeras em vários pontos da cidade, trabalhando com o controle remoto de semáforos. Isso permite respostas a casos de acidentes, quando os usuários podem utilizar-se de vias paralelas com o objetivo de enfrentar menos trânsito.

[1] Acadêmico do curso de Sistemas para Internet das Faculdades Integradas de Taquara - Faccat/RS. joaosausen@gmail.com
[2] Professor orientador. Faculdades Integradas de Taquara - Faccat/RS. leonardomachado@faccat.br

Com os problemas expostos acima, torna-se importante o desenvolvimento de uma ferramenta que auxilie os responsáveis pelo trânsito em seu trabalho de melhorar o fluxo nas grandes cidades. Para tanto, foi desenvolvida a ferramenta Suserano. Este artigo tem por objetivo detalhar a construção dessa ferramenta. O objetivo da ferramenta é identificar os semáforos com maior fluxo de carros e otimizar os tempos dos mesmos, identificando assim casos em que o fluxo de carros aumenta devido aos horários de pico, acidentes ou outras causas, como em dias de Natal ou Páscoa, por exemplo.

A seção 2 traz um referencial teórico relacionado aos assuntos pesquisados e abordados na solução. A seção 3 demonstra, passo a passo, a construção da ferramenta a partir da metodologia de desenvolvimento selecionada. A seção 4 demonstra os resultados obtidos, e a seção 5 faz uma conclusão acerca da pesquisa que foi realizada.

2 Referencial teórico

Esta seção apresenta um embasamento literário a respeito dos temas envolvidos neste projeto de pesquisa.

2.1 Trânsito e problemas relacionados

De acordo com Portal do Trânsito Brasileiro (2015), a presença dos veículos nos dias atuais torna o trânsito algo cada vez mais marcante na vida do homem moderno. É possível afirmar que, hoje em dia, não se consegue viver sem esses veículos, face o nosso grau de dependência em relação a eles.

2.1.1 Excesso de veículos

O excesso de veículos nas vias ocorre por diversos fatores. O crescente consumo de carros devido ao crescimento do poder de aquisição pela classe média e facilidades impostas pelo governo, como a diminuição de juros com montadoras e o parcelamento para a compra de veículos, trouxe para as vias 46 milhões de veículos entre 2004 e 2014, observando-se que os ônibus correspondem a menos de 1% desse número (OPPERMANN, 2014).

Outro fator responsável pelo excesso de veículos é o fato de o brasileiro acreditar que o transporte individual é muito mais vantajoso que o transporte coletivo. Segundo o IPEA, "A redução dos gastos com transporte público entre 2003 e 2009 ocorreu em praticamente todos os intervalos de renda per capita, efeito das facilidades de aquisição e uso de veículos privados pelas famílias brasileiras" (CARVALHO; PEREIRA, 2012, p. 16).

O problema é que o congestionamento é causado principalmente pelo transporte individual, e os congestionamentos trazem custos indiretos para toda a sociedade. Sabe-se que as regiões metropolitanas de São Paulo e do Rio de Janeiro tiveram prejuízos de 98 bilhões em 2013 devido aos gastos extras com combustível e trabalhos não realizados em razão dos congestionamentos.

2.1.2 Acidentes e desrespeito

Segundo Shamsher e Abdullah (2013), em um estudo feito em Bangladesh, na cidade de Chittagong, há indicativos de que os acidentes são parcialmente causados por motoristas sem instrução, que, devido à corrupção no país, adquiriram licença para dirigir sem fazer aulas ou prestar exames. Além disso, é estimado que 80 mil pessoas dirijam sem carteira em Bangladesh. O estudo também indica que um dos problemas da causa de congestionamento é o estacionamento em local proibido e ruas muito estreitas devido à posse ilegal de espaço destinado ao trânsito.

Os mesmos problemas são citados em Kabul. Conforme Safi (2011), Kabul possui estradas estreitas, excesso de pedestres, falta de transporte público, estacionamento em local proibido e falta de sinaleiras na cidade. Safi cita também que uma das causas do tráfego em Kabul são os vendedores ambulantes e pedestres sem noção de trânsito. Além disso, os oficiais de trânsito são mal treinados e comumente aceitam subornos. Safi (2011) ainda conclui que a população em geral acredita que os oficiais de trânsito não só não respeitam as leis de trânsito como também não tem total conhecimento sobre as mesmas.

De acordo com Portal do Trânsito Brasileiro (2015), os mesmos problemas são encontrados no Brasil. Os acidentes de trânsito, inclusive, anualmente fazem com que milhares de pessoas percam suas vidas.

2.1.3 Motoristas lentos

Um estudo (WANG *et al*., 2012) conduzido pelo MIT e pela Universidade da Califórnia, Berkeley, utilizou dados de celulares para identificar viagens de usuários entre regiões da Califórnia. Identificando os celulares que entravam e saíam da cobertura de antenas de celular, foram capazes de identificar que, mesmo em horário de pico, 98% das vias da Califórnia estavam com o limite de carros abaixo do que foram projetadas para suportar. Mesmo com muitos carros, as vias devem fluir normalmente desde que os usuários da via respeitem as regras de trânsito, lembrando que uma via é considerada congestionada somente quando a quantidade de carros na via for maior que a projetada.

A partir disso, o estudo então identificou um pequeno grupo de motoristas que tinha o maior tempo de viagem pela distância percorrida. Ao verificar esse grupo observando viagens aleatórias dos motoristas pertencentes a ele, foram capazes de identificar que os congestionamentos são causados por esses motoristas, que obrigam os outros motoristas a efetuar manobras para trocar de pista ou a andarem na mesma velocidade que eles, aumentando assim o tempo de viagem para todos os afetados.

2.2 Sistemas para controle de tráfego

Segundo Dunn Engineering Associates (2005), os controladores de tráfego têm a função de diminuir tempos de viagem, paradas, consumo de combustível,

emissão de poluentes e ainda reduzir o número de acidentes. Controladores de tráfego não servem apenas para aliviar congestionamentos. Em situações cotidianas, qualquer pessoa precisa respeitar os semáforos, e sistemas ineficientes, por mais que passem despercebidos por motoristas, causam perdas de combustível e aumentam a emissão de poluentes. Somando todos os usuários de trânsito, o desperdício, a longo prazo, fica na casa de milhões de reais. Alguns exemplos de controladores de tráfego são explicados a seguir.

2.2.1 Full Adaptive Traffic Management System

De acordo com Isbak (2015), *Full Adaptive Traffic Management System* é um sistema de controle de tráfego que identifica e altera os tempos de semáforos em tempo real. O sistema é desenvolvido com o intuito de calcular o tempo ideal de cada semáforo para diminuir o tempo de viagem dos veículos, diminuindo assim o consumo de combustível e a emissão de poluentes. O sistema garante aumentar a eficiência dos semáforos entre 15% e 30%, comparado a semáforos sem sistemas. O sistema utiliza sensores eletromagnéticos nas vias para identificar a quantidade de carros nas mesmas.

2.2.2 Insync

De acordo com Rhythm Engineering (2015), Insync é um sistema de controle de tráfego que utiliza câmeras em semáforos para identificar o fluxo de carros. O sistema garante otimização local e global, otimizando tanto os tempos em uma única interseção sem outras ligações, assim como em regiões com muitos semáforos, buscando gerar efeitos como a onda verde. Onda verde é o efeito causado quando um motorista andando no ritmo da via passa por vários semáforos abertos em série. O Insync foi projetado imaginando-se como um humano controlaria o tráfego em cada ponto de semáforos.

2.3 Simuladores de trânsito

Esta seção discute a respeito dos simuladores de trânsito mais conhecidos e utilizados atualmente.

2.3.1 SUMO: Simulation of Urban Mobility

De acordo com *Institute of Transportation Systems* (2015), o SUMO é um microsimulador de código aberto e uso livre. Desenvolvido na Alemanha, SUMO permite a simulação de ruas e veículos, com semáforos, sistema para encontrar rotas para carros, calculadora de emissão de gases, entre outros.

SUMO é escrito na linguagem C++ e pode ser controlado por sistemas externos usando uma interface chamada TraCI *(Traffic Control Interface)*. Com o uso dessa interface, é possível buscar dados em tempo real de uma simulação, controlar os

semáforos e simulações, alterando rotas de carros, entre outras funcionalidades do simulador.

2.3.2 MATSim Multi Agent Transport Simulation

De acordo com MATSim (2015), MATSim é um simulador multiagente de código aberto e modular. Consiste de vários módulos que podem ser usados em conjunto, ou um único módulo por vez. Os módulos podem ser reescritos a fim de personalizar o simulador conforme a necessidade.

MATSim fornece ferramentas para modelagem de demanda, simulação de mobilidade baseada em agentes, replanejamento, simulações interativas e métodos para analisar os resultados das simulações que podem representar dias de trânsito em poucos minutos de simulação.

O simulador promete simular milhões de agentes ao mesmo tempo e fornece ao usuário a capacidade de seguir um agente durante a simulação.

2.3.3 MAINSIM - Multimodal Innercity Simulation

Conforme Mainsim (2015), MAINSIM é um simulador que usa dados cartográficos do OpenStreetMaps para gerar seus mapas. É capaz de gerar simulações entre pedestres, ciclistas e motoristas. Além disso, é capaz de simular vias próprias para ônibus, carros ou ciclovias e toda interação.

MAINSIM é um macro-simulador desenvolvido para a simulação de cidades inteiras ou regiões. Fornece dados como consumo de combustível e emissão de gás carbono, por exemplo. O simulador é baseado em agentes e é capaz de gerar características próprias para certos agentes, tais como agentes que não respeitam às leis de trânsito.

2.4 Detecção de objetos através de câmeras

O método utilizado para detecção de objetos em uma imagem é o *Haar Cascades*, proposto por Viola e Jones (2001). O método consiste em ensinar um *software* classificador a identificar imagens, alimentando-o com imagens positivas, que são as que contêm objetos a serem identificados, e negativas, que são imagens que não possuem o objeto a ser identificado. Para esse método, é importante que todas as imagens utilizadas no treinamento tenham a mesma resolução. O classificador gera uma série de identificadores em uma imagem, que consistem em duas áreas próximas de uma imagem.

Cada área tem então os valores de cores de seus *pixels* somados, e é feita uma subtração entre o resultado da soma de cada área, resultando em um contraste entre elas. Esses valores são então usados pelo classificador para a detecção do objeto.

Em um carro, um identificador poderia ser criado entre o para-brisa e o teto do carro, identificando a diferença de contraste (tendo a soma das cores) entre os dois.

Esse processo gera uma grande quantidade de identificadores, entretanto

a maioria deles é irrelevante para identificar o objeto necessário. Mesmo entre os identificadores selecionados, a maioria deles só pode ser utilizada em um único ponto da imagem, já que, como no exemplo anterior, talvez não faça sentido usar o identificador entre para-brisa e teto em outra área do carro.

Após ter todos esses identificadores gerados, o classificador os utiliza em imagens positivas e negativas, encaixando-os em posições na imagem. O classificador é capaz de determinar quais identificadores serão utilizados para encontrar objetos semelhantes aos objetos dados em imagens positivas.

Na tentativa de identificar um objeto em uma imagem, os identificadores são agrupados e aplicados às imagens em cascata. Assim, se o primeiro grupo de identificadores rejeitar a imagem, a imagem é descartada.

Em um exemplo dado por Viola e Jones (2001), um identificador de faces possui 38 estágios. Os 5 primeiros possuem 1, 10, 25, 25 e 50 identificadores, respectivamente. 4916 imagens com faces foram usadas para treinar o classificador, todas tendo 24x24 *pixels*, e outras 9544 imagens sem faces foram usadas como imagens negativas. Nos 38 estágios, 6061 identificadores foram gerados.

2.5 Arquitetura, tecnologias e metodologias

Este capítulo traz um embasamento literário atual acerca da arquitetura, metodologia e tecnologias utilizadas neste projeto.

2.5.1 Kanban

Kanban é um arcabouço usado para implementar métodos ágeis. Oferece um método livre para que o responsável por organizar as tarefas consiga manter o time operando em sua máxima capacidade e entregando as tarefas de maior importância primeiro (PETERSON, 2015).

Os principais objetivos do Kanban são: visualizar o trabalho; limitar as tarefas em progresso; focar no fluxo de trabalho e na melhoria contínua.

Segundo Mariotti (2012), Kanban é uma metodologia que consiste em uma lista de tarefas organizadas por uma ordem de importância definida pelo dono do produto a ser desenvolvido. Assim que uma tarefa for completada, o desenvolvedor inicia a próxima tarefa que está no topo da fila, desde que ele seja capaz de realizar a tarefa naquele momento. Múltiplas tarefas podem estar em desenvolvimento ao mesmo tempo, porém um desenvolvedor não deve trabalhar em outra tarefa até entregar sua tarefa atual. As tarefas em andamento devem ser limitadas, e o time deve estar disposto ao crescimento constante.

Como boa prática do Kanban, é recomendado que os integrantes do time possuam conhecimento em múltiplos ciclos de uma tarefa. No caso de desenvolvimento de *software*, desenvolvedores podem realizar testes simples se necessário, pois a especialização dos integrantes do time pode causar gargalo no fluxo de trabalho quando um único funcionário é responsável por uma tarefa. Além disso, se necessário, o time todo pode focar em uma mesma tarefa, a fim de a concluir em menos tempo.

2.5.2 Drupal

Drupal é um *CMS* originalmente criado para desenvolvimento de sites. Com uma base de mais de um milhão de usuários e desenvolvedores, Drupal é distribuído sobre a licença GPL *(GNU General Public License)*. Drupal é um sistema modular, com uma grande quantidade de módulos desenvolvidos pela comunidade e igualmente distribuídos sob a licença GPL.

Drupal foi iniciado em 1999 por Dryes Buytaert, que retém os direitos sobre o sistema. Entre as empresas que hoje usam Drupal, estão o governo dos Estados Unidos, com o site da Casa Branca, empresas como a Tesla e a SpaceX e Faculdades e Universidades como Stanford, Harvard, Oxford e Faccat (DRUPAL, 2015).

2.5.3 Websockets

Conforme Lubbers e Greco (2015), *Websockets* fornecem uma API para criar uma conexão *TCP* cliente-servidor bidirecional, em que o cliente é o *browser*, usando JavaScript para iniciar a conexão, e o servidor é qualquer serviço que tenha a capacidade de implementar *sockets*.

A *Web* tradicional funciona com o navegador fazendo uma solicitação e o servidor respondendo à solicitação. Algumas tecnologias foram desenvolvidas para tentar simular o funcionamento de *sockets*, como *polling*, em que o navegador envia requisições constantes ao servidor e recebe respostas imediatas, mesmo que respostas vazias, e *long polling*, no qual o navegador faz uma requisição ao servidor e ele somente responde à requisição quando tem alguma coisa à enviar.

Com W*ebsockets*, a mesma funcionalidade de *sockets* é implementada para o navegador, observando-se que a conexão fica aberta entre o navegador e o servidor, e o navegador não precisa fazer solicitações ao servidor para receber dados do mesmo.

2.5.4 OpenCV

Segundo Eric (2011) OpenCV é uma biblioteca com uma coleção de algoritmos para ser usada pela indústria e desenvolvimento em aplicações de visão computacional. Em OpenCV, Open remete a *open source* e CV a *Computer Vision*.

Essa biblioteca foi criada na década de 1990 pela Intel e liberada como uma ferramenta de código aberto em 2000 com uma versão beta. Sua versão 1.0 foi lançada somente em 2006. Inicialmente criada como uma biblioteca para ser usada com a linguagem C, OpenCV passou a dar suporte para C++, Java e Python em sua versão 2.0. A biblioteca pode ser usada em Linux, Windows, Android, Mac OS e iOS.

Segundo Milano e Honorato (2009), visão computacional é a maneira como um computador enxerga as coisas a sua volta, seja por meio de câmeras, sensores, scanners, entre outros. Basicamente todos os sistemas de visão computacional consistem em adquirir uma imagem, extrair características como curvas, linhas, etc. e utilizar um sistema especialista para detectar o objeto que se deseja obter.

2.5.5 OpenLayers e OpenStreetMaps

Segundo OpenLayers (2015), OpenLayers é uma biblioteca de código aberto escrita em JavaScript e sob a licença BSD, criada para trabalhar com mapas de diversas fontes. Com OpenLayers, é possível mostrar informações geográficas de uma região usando uma de diversas fontes de dados, como Google Maps, OpenStreetMaps, Bing Maps, MapQuest, entre outros.

O OpenLayers divide o mapa em camadas. A camada com as imagens do mapa é fornecida por um dos serviços listados acima. Com o uso em conjunto com o OpenStreetMaps, são mantidas informações dos mapas em formato vetorial em XML e geradas imagens em formato png de regiões do mapa. Essas imagens possuem sempre o mesmo tamanho e, portanto, cobrem diferentes áreas conforme o zoom. Cada vez que o usuário move o mapa, novas imagens são carregadas a fim de preencher todo o mapa que o usuário está visualizando.

Com OpenLayers, é possível interagir com o mapa em uma camada de vetores, adicionando pontos, camadas e textos. Esses pontos são adicionados pelo sistema que implementa o mapa e não são fornecidos pelo OpenStreetMaps.

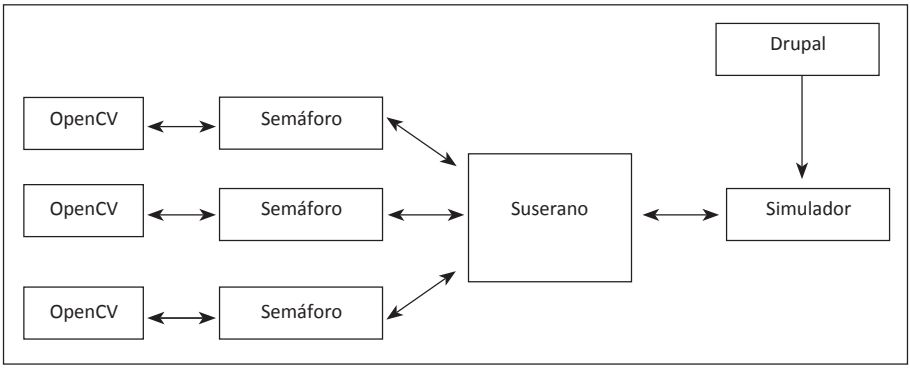
3 Desenvolvimento da ferramenta

A partir dos problemas discutidos no referencial teórico a respeito do cenário atual do trânsito, e baseando-se nas tecnologias discutidas na seção 2.6, partiu-se para o desenvolvimento da ferramenta para apoio ao controle do tráfego urbano.

A ferramenta desenvolvida é composta por alguns módulos, que possuem funções bem definidas, e que trabalham em conjunto para o funcionamento global da mesma, produzindo ao final o resultado esperado de uma ferramenta que auxilie o responsável pelo controle de tráfego em seu trabalho de planejar os aspectos do trânsito.

A Figura 1 ilustra o diagrama de blocos do sistema.

Figura 1 - Módulos do Sistema



Fonte: Autor.

Esta seção descreve a construção da ferramenta, detalhando como os aspectos de arquitetura, metodologia e tecnologia, retratados na seção 2.6, foram utilizados na construção.

3.1 Simulador

Junto ao sistema, um simulador foi construído para que o mesmo pudesse ser testado. Ao começo da construção do simulador, um escopo foi criado e as funcionalidades necessárias foram definidas. No escopo, foi decidido que o simulador seria simples e que teria apenas as funcionalidades necessárias para testar o sistema. Seguindo o escopo, as funcionalidades foram listadas na ferramenta Trello (2015) e organizadas por ordem de prioridade, seguindo os princípios do Kanban. Para o projeto do simulador, assim como para as outras funcionalidades da ferramenta, diagramas foram construídos, com a utilização da linguagem UML (*Unified Modelling Language*). Esses diagramas tiveram a finalidade de permitir que fossem avaliadas, ainda antes da implementação das funcionalidades, como elas seriam estruturadas e como deveriam se comportar. Dessa forma, os detalhes da construção da ferramenta puderam ser avaliados antes mesmo de a implementação de cada tarefa ser iniciada.

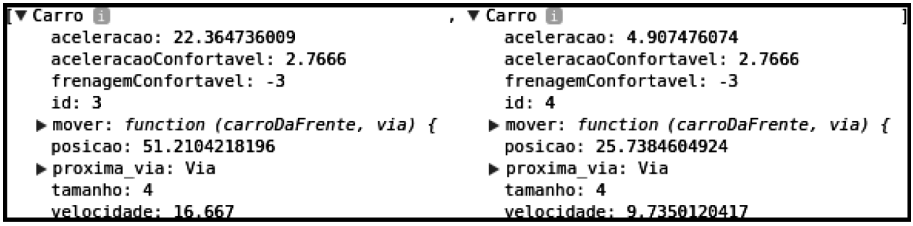
O resultado é um macro-simulador capaz de simular uma grande rede de semáforos em uma cidade ou região. Esse macro-simulador é limitado a simular vias únicas de mão dupla e considera que todos os veículos possuem mesmo tamanho e aceleração e que todos os motoristas respeitam às regras de trânsito. Ele também não leva em consideração o tempo de reação dos motoristas, ou seja, considera que todos reagem imediatamente à necessidade de frenagem ou aceleração. As funcionalidades e características listadas acima configuram, juntas, um conjunto básico de necessidades para se efetuar a simulação do tráfego urbano. Apesar de ser um conjunto limitado, elemostrou-se suficiente para o propósito deste projeto.

Para o simulador, as vias são sempre definidas com semáforos e as únicas informações no simulador são os carros nas vias e as próximas vias que os carros podem seguir a partir de um semáforo. Apesar de o simulador usar o OpenStreetMaps, nenhuma informação é extraída desse. De fato, o simulador é capaz de funcionar apenas com as informações fornecidas pelo Drupal. No simulador, a movimentação dos veículos é feita usando fórmulas de Movimento Uniformemente Variável, que são baseadas na equação de Torriceli ($V^2=Vo^2+2*a*\Delta S$). Dessa maneira, é possível que sejam calculadas a aceleração ou frenagem necessária para, em certo tempo, um veículo ir de tal velocidade até tal velocidade. Também é possível que seja calculada a velocidade alcançada, dado certo tempo e aceleração, entre outros.

Esses cálculos permitem, por exemplo, que os motoristas saibam quando precisam desacelerar o veículo sabendo apenas sua distância para o veículo ou obstáculo a frente e a sua velocidade. No simulador desenvolvido, os veículos sabem exatamente a velocidade e a distância do carro à sua frente. Assim, no momento que o carro da frente frena, o carro de trás sabe o quanto precisa desacelerar para não colidir. Isso tudo ocorre imediatamente. Todos os cálculos feitos para movimentar os veículos utilizam a fórmula demonstrada.

Os veículos no simulador são representados como objetos que possuem velocidade, tamanho, posição na via, aceleração e informações sobre as próximas vias. Os carros possuem poder para acelerar, frear e decidir por qual via vão seguir. A Figura 2 demonstra dois carros, representados no simulador construído, cada um com suas propriedades e métodos.

Figura 2 - Dois carros com suas propriedades e métodos no simulador.

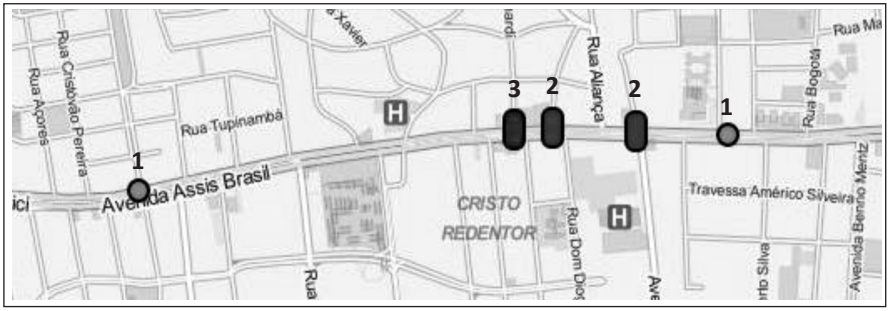


Fonte: Autor.

Quando um carro entra em uma via, ele é inserido na posição 0, com os mesmos valores de velocidade e aceleração que saíram da outra via, ou com velocidade zero e aceleração de 2.76m/s (em torno de 10k/h) caso sejam inseridos a partir de um ponto de entrada na simulação. No momento em que o carro é inserido, ele já toma a decisão de qual será a próxima via para a qual ele deve seguir. Assim, ao chegar próximo de um semáforo que está aberto, ele sabe se pode atravessar o semáforo a fim de chegar à próxima via, o que pode ser impedido caso essa via esteja congestionada.

As vias no simulador sempre acabam em semáforos. Por esse motivo, uma via para o simulador é sempre um ponto inicial (que pode ser o fim de outra via) até chegar a um semáforo. É comum que essa definição seja chamada de tamanho de caixa, porém, no simulador, a via pode ter quilômetros. Não há limite para o tamanho de caixa no simulador. A Figura 3 demonstra uma simulação simples.

Figura 3 - Simulação simples com dois pontos de entrada/saída (em azul), dois semáforos abertos (em verde) e um semáforo fechado (em vermelho)[3]



Fonte: Autor.

[3] O número 1 indica os pontos em azul; o número 2, semáforos verdes e o número 3, semáforo vermelho.

Caso os semáforos estejam muito distantes uns dos outros, são usados pontos especiais no simulador que indicam onde os carros estão saindo ou entrando na simulação, pois semáforos muito distantes são considerados desconectados pelo sistema, já que um não interfere no outro. Como exemplo, se fôssemos simular a zona norte e a zona sul de Porto Alegre sem considerar os semáforos intermediários, teríamos que adicionar pontos de entrada e saída no simulador em cada uma das zonas, para abastecer o simulador e remover os carros que sairiam de cada uma. Não existiria uma conexão entre as duas zonas, a não ser que informássemos explicitamente para o simulador que, por exemplo, um semáforo da zona norte e um da zona sul devem abrir ao mesmo tempo, o que não faz sentido em um ambiente real.

O simulador, de uma forma geral, não possui poder de decisão. Apesar de poder trocar o estado dos semáforos, ele não possui capacidade de alterar os tempos dos mesmos, simulando o funcionamento de semáforos comuns, em que um temporizador com tempo fixo regula quando os semáforos devem abrir ou fechar. No entanto, os tempos de sinal aberto ou fechado devem ser regulados por um sistema externo.

Em um contexto mais técnico, os dados da simulação são fornecidos pelo Drupal para o servidor, que é escrito em JavaScript, porém usando o Drupal como *framework*. Ao usuário final do simulador é fornecido um sistema com a capacidade de adicionar semáforos de simulação, adicionar pontos de entrada e saída em uma simulação, criar uma simulação e rodar uma simulação. Ao rodar a simulação, ao usuário é apresentado um mapa do OpenStreetMaps (2015), com os semáforos do simulador sobre o mapa, adicionados usando a biblioteca OpenLayers (2015). Cada semáforo fornece informações de tempo de semáforo aberto, tempo de semáforo fechado, quantidade de carros na via e estado atual do semáforo.

Assim como os semáforos reais, os semáforos do simulador possuem capacidade de se conectar ao sistema e simular um semáforo real. Cada semáforo abre uma conexão separada, mesmo que todos sejam controlados por um mesmo sistema simulador. Os semáforos no simulador agem como os semáforos reais, enviando a quantidade de carros em certo ponto do tráfego de tempos em tempos. O sistema não diferencia um semáforo do simulador para um semáforo real.

3.2 Suserano

Usando também de Kanban, o principal sistema desse projeto é chamado de Suserano. O nome suserano não é um acrônimo: o suserano, na época feudal, era alguém que entregava um feudo a um vassalo em troca de lealdade.

O Suserano é o sistema principal que controla o tráfego. Foi construído em C++ e funciona como um servidor, aceitando conexões via *websockets* dos semáforos. Seu principal objetivo é analisar os dados enviados pelos semáforos e decidir os tempos ideais de semáforo aberto e fechado, a fim de otimizar todo o tráfego controlado por ele.

O Suserano desconhece o que é real e o que é simulação, assim semáforos de simulação e semáforos reais podem estar conectadas ao suserano ao mesmo tem-

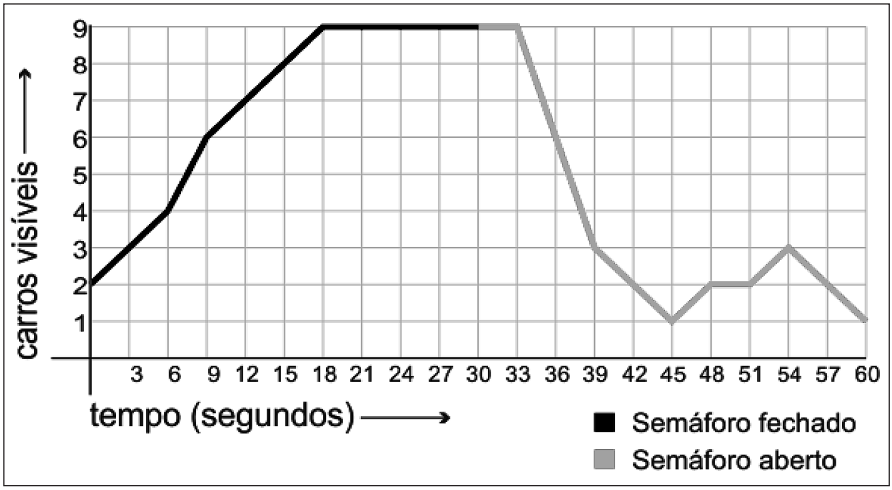
po e ele pode gerir ambos, porém o Suserano não foi projetado pensando nisso. É importante deixar claro que os semáforos usam sistemas próprios que se conectam ao Suserano, usam OpenCV para identificar a quantidade de carros na via e alimentar o Suserano, assim como os semáforos do simulador alimentam o Suserano com informações da simulação. Tanto os semáforos reais quanto os do simulador usam o mesmo protocolo para se comunicar com o Suserano.

Para o Suserano, cada semáforo possui um identificador único que é recebido no momento que o semáforo se conecta ao mesmo. Outros dados também são enviados, entre eles, o tempo atual de verde e vermelho, identificadores de semáforos que estão paralelos ou díspares, identificadores de semáforos que podem receber fluxo desse semáforo, estado atual do semáforo, quantidade de vias e tamanho de caixa. Além disso, em tempos definidos, os semáforos enviam a quantidade de carros no ponto visível da câmera, assim o Suserano possui informações de aumento ou redução de tráfego naquele momento.

A partir dos dados enviados de tempos em tempos, é possível gerar um gráfico de fluxo de carros no semáforo. A partir disso, o sistema consegue manter um histórico de tráfego. Dessa forma, é possível que esses dados sejam utilizados para se calcular os tempos ideais de um semáforo.

As alterações de tempo nos semáforos são feitas gradualmente, sem comprometer as vias de maneira instantânea. Assim, é possível alterar os tempos de semáforo e ir medindo os resultados sem afetar o fluxo bruscamente.

Figura 4 - Gráfico demonstrando o fluxo de carros



Fonte: Autor.

A conexão do Suserano com um semáforo foi implementada manualmente, uma vez que C++ não implementa *websockets* por padrão. A troca de mensagens para o início da conexão foi implementada usando os padrões definidos pela IETF e a biblioteca de sockets Asio, fornecida junto ao Boost. De acordo com IETF (2015), a

IETF é um grupo de voluntário que tem por missão produzir material de alta qualidade a fim de influenciar as pessoas que usam e controlam a internet. Já a biblioteca de *sockets* Asio é uma biblioteca multiplataforma para programação baixo nível de rede (BOOST, 2015). Para a conexão entre o Suserano e os semáforos, não existe a necessidade de se usar *Websockets*, já que ambos são feitos em C++ e usando a Asio, porém, como o simulador não possui a capacidade de usar sockets, foi decidido usar *Websockets* em todos os sistemas construídos, a fim de normalizar a comunicação.

O protocolo utilizado é um protocolo próprio e simples, em que são enviadas mensagens compostas de números inteiros. O primeiro inteiro identifica o tipo de mensagem, e os próximos inteiros variam conforme a mensagem. Em uma mensagem de envio da quantidade de carros no semáforo, por exemplo, é enviado um inteiro (3) junto à quantidade de carros no semáforo (16 por exemplo). Assim sendo, a mensagem no exemplo anterior seria **3 16**. O Suserano identifica o semáforo que enviou os dados pela própria conexão, que cria um identificador único para cada semáforo no momento de conexão.

3.3 Semáforos

O sistema dos semáforos também foi construído em C++, usando a biblioteca Asio para a comunicação com o Suserano. Em momento algum os semáforos se comunicam com o simulador, já que eles "concorrem" com o mesmo pelo Suserano.

A identificação de carros em um semáforo é feita usando o openCV, que, por sua vez, usa *Haar Cascades* para identificar carros em uma imagem. De tempos em tempos, o sistema do semáforo identifica a quantidade de carros no *frame* atual do vídeo e o envia para o Suserano por meio de uma conexão previamente aberta.

O sistema dos semáforos recebe dados do Suserano, que informam quanto tempo elas devem permanecer abertas e fechadas. Esse sistema é responsável por controlar o estado do semáforo e enviar os dados para o simulador. Os tempos de semáforo sempre são os últimos tempos recebidos do Suserano. Assim, se o semáforo não receber dados do simulador, assume-se que os tempos não precisam ser alterados. Caso o semáforo desconecte, os últimos tempos recebidos serão usados.

Para o pleno funcionamento dos semáforos, é necessário que o sistema tenha capacidade de controlar o hardware para alterar os estados do semáforo, ligando e desligando lâmpadas ou *leds*, por exemplo. Porém a parte do *hardware* não foi desenvolvida nesse sistema. Cada semáforo deve ter uma câmera apontada para o local em que os carros ficam parados em frente a ela. Várias câmeras podem estar conectadas a um único computador e cada semáforo deve rodar seu próprio *software*, que vai usar uma das câmeras. Esse computador então deve estar conectado a Internet ou a alguma rede, pois a finalidade é conectar os semáforos ao Suserano. A quantidade de semáforos usando de um mesmo computador é limitada pelos recursos da máquina, que incluem conexões USB para as câmeras, e capacidade de processamento da mesma, já que a identificação de carros em uma imagem é um processo que exige uma boa quantidade de recursos.

Porém, como indicado acima, os semáforos só precisam enviar dados de

quantidade de carros em certos tempos definidos. Em uma simulação, identificando carros em um vídeo 320x240 usando de um computador com processador i5, 4 núcleos de 2.3Ghz e 4gb de memória *ram*, o *software* foi capaz de identificar 38 vezes por segundo a quantidade de carros no vídeo. Por padrão os semáforos enviam a quantidade de carros a cada 3 segundos, sendo assim é necessário 1 processamento a cada 3 segundos por semáforo. Basicamente é seguro afirmar que 4 semáforos em um cruzamento poderiam usar de um único computador até mesmo inferior ao usado para os testes para o processamento de imagens.

Quanto à internet necessária, conforme a descrição do protocolo acima, em que uma mensagem era de apenas dois inteiros, estima-se que a maior das mensagens possui em torno de 20 inteiros, o que resulta em 160 *bytes*. Essa é a mensagem de conexão que envia todos os dados para criar o semáforo no Suserano. O número de inteiros enviados não é fixo, pois um dos dados enviados, por exemplo, é a relação desse semáforo com os semáforos ao seu redor, em paralelo ou em série com essa e esse valor é variável.

Estima-se que mesmo uma conexão discada com *download* e *upload* de 5kbps é capaz de suportar 4 semáforos de um cruzamento. O interessante então é que não haja perda de pacotes ou alta latência na conexão.

4 Resultados obtidos

O principal benefício do uso da ferramenta é o de diminuir os tempos de viagem das pessoas, seja pelo uso convencional, regulando os semáforos para que seja possível a onda verde, seja em casos não convencionais, como, por exemplo, regulando-se os tempos de semáforos em vias paralelas às vias bloqueadas.

O Quadro 1 demonstra os resultados obtidos em simulações.

Quadro 1 - Resultados obtidos

	Com o Suserano		**Sem o Suserano**	
Tempo de Simulação	Quantidade total de carros	Carros que concluíram a viagem	Quantidade total de carros	Carros que concluíram a viagem
10 min	57	42	69	54
10 min	181	165	130	112
20 min	789	771	479	461

Fonte: Autor.

A diminuição de tempo de viagem traz consigo vários outros benefícios, além da comodidade dos motoristas: a redução de combustível utilizado e de CO2 emitido. Esses fatores afetam diretamente a economia e o bem-estar das pessoas

envolvidas nessa mudança.

Além disso, como o sistema procura aumentar os tempos de verde em semáforos muito ocupados, em caso de bloqueios em algumas pistas, as vias escolhidas pelos motoristas automaticamente teriam seus semáforos reguladas após algumas iterações, diminuindo o atraso causado pelos bloqueios. Esses acidentes podem e costumam acontecer em vias muito congestionadas devido ao horário de pico, que normalmente acontece nos horários de ida ou volta ao trabalho. Esses tempos perdidos no trânsito, atualmente, geram transtornos a toda a sociedade. Os benefícios para o trânsito podem ser ainda maiores, caso, em conjunto com o uso desta ferramenta, os motoristas usem de sistemas de *GPS,* que consideram o tráfego para gerar as melhores rotas.

5 Conclusão

O objetivo deste projeto foi o de apresentar um sistema capaz de otimizar o tráfego de uma região a partir do controle dos semáforos. Durante o projeto, foi construído um simulador possuindo apenas os recursos necessários para demonstrar o funcionamento do sistema, o sistema propriamente dito, usando o Drupal para manter as configurações do sistema e a linguagem C++ para a construção do sistema de controle de tráfego e para o sistema dos semáforos.

O estudo revelou que os problemas de tráfego estão muito além dos tempos de semáforos. Alguns problemas só podem ser resolvidos com planejamento, obras para melhoria das vias, melhor educação dos motoristas e até mesmo combate à corrupção. No entanto, a utilização da ferramenta aqui proposta pode fazer parte de um ajuste fino nos sistemas de trânsito atuais, já que não é necessário um investimento muito alto para a sua utilização.

Também fica claro que sistemas de controle de tráfego são uma opção viável para tentar reduzir os problemas de tráfego no Brasil. Poderiam ser implantados em todas as cidades com problemas de tráfego, e os benefícios seriam sentidos por todos os envolvidos.

Alguns trabalhos futuros foram identificados durante o desenvolvimento desta ferramenta. Tendo um sistema como esse instalado, os mesmos equipamentos podem ser utilizados no futuro para, por exemplo, identificar as placas de carros para localizar carros roubados, ou até mesmo para forçar o semáforo a abrir em caso de o veículo ser uma ambulância ou caminhão dos bombeiros, por exemplo. Carros clonados podem ser identificados registrando as últimas sinaleiras pelas quais os carros passaram, assim se uma mesma placa passou em horários parecidos em locais diferentes, é simples concluir que um dos carros é clonado.

Além disso, usando as placas dos carros, é possível traçar com precisão rotas de carros, e garantir que os carros sejam afetados pela onda verde, alterando os tempos de semáforo com base em alguns carros específicos. Com esses dados todos, outros estudos podem ser realizados fazendo análises reais dos veículos.

Referências

BOOST. Disponível em: <http://www.boost.org>. Acesso em: 15 set. 2015.

CARVALHO, C. H. R.; PEREIRA, R. H. M. *Gastos das Famílias Brasileiras com Transporte Urbano Público e Privado no Brasil*: uma análise da POF 2003 e 2009. IPEA - Texto para discussão. Brasília, p. 16, dez. 2012.

DRUPAL. Disponível em: <http://drupal.org/>. Acesso em: 15 set. 2015.

DUNN ENGINEERING ASSOCIATES. *Traffic Control Systems Handbook*. Westhampton Beach, NY, out. 2005.

ERIC, G. *Introduction To Computer Vision Using OpenCV. Oakland, Califórnia USA*, 2011. Disponível em: <http://www.embedded-vision.com/platinum-members/bdti/embedded-vision-training/documents/pages/introduction-computer-vision-using-op>. Acesso em: 24 set. 2015.

IETF. Disponível em: <http://www.ietf.org>. Acesso em: 9 set. 2015.

ISBAK. Disponível em: <http://www.isbak.com.tr>. Acesso em: 10 ago. 2015.

INSTITUTE OF TRANSPORTATION SYSTEMS. Berlin, Alemanha. Disponível em: <http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000>. Acesso em: 1 ago. 2015.

LUBBERS, P.; GRECO F. *HTML5 Web Sockets*: A Quantum Leap in Scalability for the Web. Disponível em: <https://www.websocket.org/quantum.html>. Acesso em: 26 set. 2015.
MAINSIM. Disponível em: <http://www.mainsim.de/?lang=en>. Acesso em: 6 ago. 2015.

MARIOTTI F. S. Kanban: o ágil adaptativo. *Engenharia de Software Magazine*. Edição 45. 2012. p. 6-10.

MATSim. Disponível em: <http://www.matsim.org>. Acesso em: 18 ago. 2015.

MILANO, D.; HONORATO, L. B. *Visão Computacional.* 2009. Disponível em: <http://www.ft.unicamp.br/liag/wp/monografias/monografias/2010_IA_FT_UNICAMP_visaoComputacional.pdf>. Acesso em: 18 set. 2015.

OPENLAYERS. Disponível em: <http://openlayers.org/>. Acesso em: 22 set. 2015.

OPENSTREETMAP. Disponível em: <http://www.openstreetmap.org>. Acesso em: 22 set. 2015.

OPPERMANN, N. DOTS *Cidades* - Manual de Desenvolvimento Urbano Orientado ao Transporte Sustentável. 2014. Disponível em: <http://embarqbrasil.org/research/publication/dots-cidades-manual-de-desenvolvimento-urbano-orientado-ao-transporte>. Acesso em: 10 out. 2015.

PETERSON, D. *Kanban Blog*. Disponível em: <http://kanbanblog.com/>. Acesso em: 8 set. 2015.

PORTAL DO TRÂNSITO BRASILEIRO. Disponível em: <http://www.transitobr.com.br>. Acesso em: 1 set. 2015.

RHYTHM ENGINEERING. Disponível em: <http://rhythmtraffic.com>. Acesso em: 19 ago. 2015.

SAFI, Z. U. *Survey Report of Causes of Vehicles Traffic Problem in Kabul Afghanistan*. Kabul, 2011.

SHAMSHER, R.; ABDULLAH M. N. Traffic Congestion in Bangladesh - Causes and Solutions: A study of Chittagong Metropolitan City. *Asian Business Review*. v. 2, n. 1, Bangladesh, 2013.

TRELLO. Disponível em:<https://trello.com>. Acesso em: 19 ago. 2015.

VIOLA, P.; JONES, M. *Rapid Object Detection using a Boosted Cascade of Simple Features*. Cambridge, Massachusetts, 2001.

WANG, P. *et al*. *Understanding Road Usage Patterns in Urban Areas*. 2012. Disponível em: <http://www.nature.com/articles/srep01001>. Acesso em: 21 out. 2015.