# Gerenciamento automático de vagas em estacionamentos baseado em redes de sensores sem fio para ITS

**Marcelo R. O. Castro**[1], **Marcio A. Teixeira**[1], **Luis H. V. Nakamura**[1], **Rodolfo I. Meneguette**[1]

[1] Instituto Federal de São Paulo - IFSP
{marcio.andrey, nakamura, meneguette}@ifsp.edu.br

***Abstract.*** *In this paper we developed an intelligent parking service based on wireless sensors for managing public or private spaces. The proposal makes use of a low cost sensor coupled in a node to monitor the parking field. Other nodes capture images of car park field and periodically send them to the intelligent transport system. The system provides a mobile access in which you can perform various management functions, such as finding vacant parking lots and a statistical report. We implemented a prototype of this service using the OpenCV library. The service evaluation demonstrates the effectiveness of the proposed solution, showing that the service can manage the parking places without impair other services that are performed in the intelligent transport system.*

***Resumo.*** *Neste artigo foi desenvolvido um serviço inteligente baseado em sensores sem fio para gerenciamento de vagas em estacionamentos de espaços públicos ou privados. A proposta faz uso de um sensor de baixo custo acoplado em um nó para monitorar o campo de estacionamento. Outros nós capturam imagens do campo e enviam, periodicamente ao sistema de transporte inteligente. O sistema disponibiliza um acesso para dispositivos móveis no qual é possível executar várias funções de gerenciamento, tais como encontrar lotes de estacionamento vagos e gerar um relatório estatístico. Para tanto, foi implementado um protótipo desse serviço utilizando a biblioteca OpenCV. A avaliação desse serviço demonstra a eficácia da solução proposta, mostrando que o serviço pode gerenciar o estacionamento sem prejudicar outros serviços que são realizados no sistema de transporte inteligente.*

## 1. Introdução

As redes de sensores sem fio (WSN - *Wireless Sensor Network*) tem recebido considerável atenção por parte dos pesquisadores, governos e empresas devido ao seu potencial de fornecer soluções fáceis e econômicas em muitas áreas diferentes [Suryady 2014]. Portanto, uma WSN pode ser integrada com outras tecnologias para auxiliar os serviços do sistema de transporte inteligente (*Intelligent Transportation System*-ITS) [R. E. Barone 2014]. ITS é a junção de várias tecnologias com o foco na prestação de serviços e aplicações que irão monitorar e gerenciar o sistema de transporte tornando-o mais confortável e seguro [Meneguette 2016a].

De acordo com a previsão de McKinsey [Mohr 2013], o número de veículos deve aumentar para 1.32 bilhão unidades em 2020. Este aumento impactará no congestionamento do tráfego devido à maior demanda por espaço nas estradas em períodos de pico

[Meneguette 2016b], bem como, gerará uma escassez de vagas em estacionamentos nas grandes cidades à medida que a demanda excede a oferta.

Atualmente, o método mais utilizado de encontrar uma vaga de estacionamento é pelo método de "força bruta", onde o motorista fica procurando uma vaga pelas vias perto do seu destino. Este método considera o conhecimento do lugar (experiências) e conta com a sorte dos motoristas. Além disso, isso pode aumentar o congestionamento do tráfego devido à baixa velocidade que os motoristas frequentemente trafegam quando estão procurando um espaço de estacionamento [Jung 2016]. Uma alternativa é encontrar um parque de estacionamento pré definido com alta capacidade. No entanto, esta nem sempre é uma solução ideal porque o parque de estacionamento poderia estar longe do destino desejado pelo usuário [T. N. Pham 2015]. Dessa forma, sistemas computacionais poderiam gerenciar o estacionamento através de uma combinação de diferentes tecnologias. O ITS poderia prover um serviço que detecta, gerencia e informa aos usuários sobre vagas disponíveis em um estacionamento mais próximo.

Na literatura, há algumas obras que abordam o problema do estacionamento gratuito [Suryady 2014, T. N. Pham 2015, R. E. Barone 2014, J. Li 2016, Yeh 2016, Huang 2016, P. Sheelarani 2016, Fulya Yuksel Ersoy 2016]. No entanto, alguns destes trabalhos utilizam *hardware* externo para suportar a sua aplicação. Além disso, esses trabalhos incluem um RFID ou outros dispositivos para identificar vagas em estacionamento de veículo. Assim, geralmente é preciso diversos dispositivos ( $n$ sensores por vaga) e/ou equipamentos embutidos ao veículo para suportar a gestão do estacionamento.

Neste artigo é proposto um novo serviço que não necessita de vários dispositivos no veículo ou vaga para garantir o gerenciamento de vagas disponíveis nos estacionamentos das cidades. Para detectar uma vaga disponível, o serviço utiliza imagens que podem ser obtidas pelo sistema de câmara da cidade ou por um nó de rede sem fio que faz capturas instantâneas do campo de estacionamento e envia os dados ao centro de dados (*data center*). Esse centro de dados processará a imagem e indicará o estacionamento disponível para o usuário através da aplicação móvel.

As principais contribuições deste trabalho são: (i) Desenvolver e analisar o serviço de processamento de imagens para detectar espaços de estacionamento disponíveis em lugares internos e externos; (ii) Desenvolver um centro de dados do sistema de transporte inteligente para gerenciar e controlar desse serviço; (iii) Desenvolver um dispositivo móvel para notificar aos usuários sobre uma vaga de estacionamento disponível; (iv) Desenvolver um dispositivo barato (nó) que capture as imagens sobre os campos de estacionamento.

O restante deste artigo está estruturado da seguinte forma. Na próxima seção há uma visão geral das principais abordagens existentes para a detecção de congestionamento de veículos em VANETs (*Vehicular Ad Hoc Network*). A solução proposta neste artigo é descrita na Seção 3, enquanto uma avaliação detalhada do desempenho e resultados são apresentados na Seção 4. Finalmente, a Seção 5 é destinada a conclusão deste trabalho e faz sugestões para trabalhos futuros.

## 2. Trabalhos Relacionados

Na literatura existem alguns trabalhos que abordaram o sistema de gestão de estacionamento para espaços públicos ou privados.

Li et. al. [J. Li 2016] propôs um método por busca de carros baseado em um sistema para *smartphone*. Esse método faz uso de algumas função de localização internas e caminhos registrados por navegação para detectar e pesquisar um veículo estacionado em um grande parque de estacionamento. O autor desenvolveu um aplicativo móvel baseado em QR-Code que é usado para formatar a informação que auxilia a pesquisa e detecção da vaga de estacionamento disponível. Os dados codificados no QR-Code são constituídos pelo estacionamento, andar ou piso e localização da vaga. As aplicações móveis criadas são: mapa *off-line*, leitura do QR-Code para gravar o local de estacionamento, planejamento do melhor caminho para buscar o carro, navegação em tempo real. Assim, os motoristas podem digitalizar e decodificar códigos QR usando um *smartphone* e usar o aplicativo de localização para encontrar o seu veículo em um grande parque de estacionamento.

Pham et.al. [T. N. Pham 2015] propuseram um sistema de estacionamento que ajuda os motoristas a encontrar um espaço gratuito. Este sistema é baseado na Internet de Coisa (IoT- *Internet of Things*) e usa um centro de dados que serve como um servidor na nuvem para calcular os custos de um estacionamento. O autor também usa uma unidade local que armazena a informação de cada vaga de estacionamento e a sua localização no mesmo. Esta unidade local inclui uma unidade de controle que consiste num módulo Arduino que é ligado em um leitor RFID para verificar e autenticar as informações do condutor e veículo calculando as vagas livres em cada parque de estacionamento. Além disso, essa proposta inclui também uma tela que mostra informações sobre a capacidade do parque de estacionamento, o total de espaços livres, o status da verificação de *tag* RFID e um mini mapa do estacionamento local.

Yeh et. al [Yeh 2016] propôs um sistema de integração para estacionamentos de uma cidade. O sistema combina dispositivos móveis inteligentes, tecnologias de computação em nuvem e o sistema de posicionamento global (GPS). Além disso, o autor desenvolveu uma aplicação que faz a reserva da vaga e prover serviços de navegação para o estacionamento. Quando o motorista ativa o aplicativo, ele envia as informações de localização do usuário para o sistema de computação em nuvem que por sua vez, também recebe as informações de estacionamentos através da uma varredura do sistema RFID no estacionamento. Assim, depois de uma análise pelo o sistema, a aplicação do contudor recebe os dados dos cinco estacionamentos mais próximos com as vagas de estacionamento disponíveis, permitindo-lhe escolher a vaga mais adequada.

Barone et.al [R. E. Barone 2014] apontou um modelo de estacionamento chamado de assistente de estacionamento inteligente (IPA - *Intelligent Parking Assistant*). Esta abordagem permite que os condutores façam alguma reserva no estacionamento de destino antes de sua partida. Além disso, ele irá fornecer informações sobre a disponibilidade de vagas em vias públicas . O IPA baseia-se no sistema SPARK que utiliza a RFID para identificar o veículo. O RFID permite que o IPA consiga entrar nas vagas de estacionamento após estas terem sido reservadas sem a necessidade de uma conexão com a Internet. IPA permitem que outras unidades possam fazer uma reserva para outras pessoas,

simplesmente entregando uma pequena identificação de rádio frequência (RFID *tag*).

Sheelarani et al. [P. Sheelarani 2016] têm uma proposta de aplicação que auxilia os motoristas ao estacionar seus veículos em um estacionamento vazio. Esta aplicação baseia-se na plataforma Android e utiliza *hardware* embutido para auxiliar a aplicação. Nesta abordagem, os autores utilizam um infravermelho Sensores (IRS) para encontrar a disponibilidade de uma vaga para estacionar o veículo. Além disso, o veículo possui uma etiqueta RFID que é utilizada para identificar o condutor/proprietário. Assim, os condutores podem reservar uma vaga no parque de estacionamento por meio da sua identificação (RFID *tag*).

Nossa proposta baseia-se nos métodos de processamento de imagem para detectar uma vaga de estacionamento disponível em ambientes internos ou externos. Ao contrário de alguns outros métodos, no método de proposta não é necessário em um *hardware* ou outro dispositivo incorporado no veículo, ele usa imagens capturadas pelo nó de sensores wireless para indicar uma vaga disponível para o motorista. Além disso, o serviço será acessado em qualquer lugar e em qualquer momento, pois o sistema só precisa da conexão com o centro de dados (nuvem).

## 3. Gerenciamento de vagas em estacionamento baseado em redes de sensores sem fio para ITS

Esta seção descreve uma solução baseada em um serviço para a detecção e gerenciamento de estacionamento disponível em uma cidade, nomeada como SPANS - *Smart Parking Service*. SPANS é um serviço baseado no processo de imagem que detecta vagas disponíveis em um estacionamento. Além disso, este serviço está integrado com uma arquitetura de transporte inteligente que irá fornecer a estrutura necessária para detectar, gerenciar e notificar as unidades sobre essas vagas. Estas infraestruturas visam evitar o tráfego pesado em áreas que têm estacionamento, reduzindo assim o tempo e o consumo de combustível, bem como a poluição causada por veículos que procuram por uma vaga disponível.

A infraestrutura utiliza sensores para realizar a detecção de um lugar de estacionamento e câmeras para registrar as atividades. Sensores e câmeras são usados para detectar informações significativas do estacionamento para que seja permitido inferir se há algumas vagas disponíveis. Além disso, esta infraestrutura consiste em um centro de dados (*cloud*) que fornece um bom mecanismo para a abstração de dados e processamento de imagem, bem como um bom mecanismo de comunicação entre os motoristas e sensores. Além disso, o centro de dados oferecerá um mecanismo de segurança que dificilmente outras pessoas têm acesso à informação. As unidades irão ter acesso ao centro de dados por meio de um aplicativo móvel que pode ser executado em um *smartphone* ou *tablet*. A Figura 1 exibe uma abstração da infra-estrutura proposta que é utilizada para detectar e notificar o usuário sobre o estacionamento gratuito.

Portanto, o centro de dados de tempos em tempos ou quando o sensor detecta uma mudança no ambiente, recebe informações sobre o lugar de estacionamento. Esta informação é processada pelo SPANS que irá detectar se algum estacionamento está disponível. Após esse processamento, o sistema fornecerá essas informações sobre os locais disponíveis para o aplicativo do condutor. Uma vez que o condutor acessa seu aplicativo móvel, as informações lhe mostrarão o local disponível próximo ao seu destino.

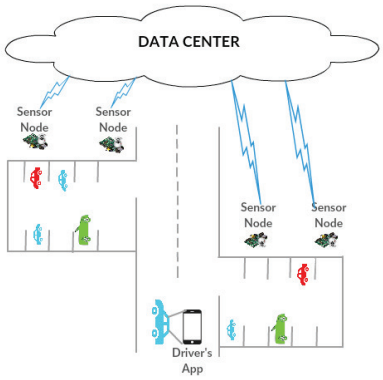


**Figura 1. Uma abstração da infra-estrutura da proposta**

Nas subsecções seguintes, é descrita a infraestrutura da proposta desenvolvida neste trabalho, bem como quais foram os serviços e equipamentos desenvolvidos e utilizados. Além disso, também são descritos para um nó sensor, no caso do local a ser monitorado não possuir uma estrutura de câmeras que capture as imagens dos estacionamentos.

### 3.1. Protótipo do Centro de Dados

Para o desenvolvimento deste trabalho, o centro de dados (*data center*) é implementado como serviços web RESTFul (*Representational state transfer*) que tem a conexão à Internet para receber solicitações dos aplicativos do motorista, bem como receber as imagens coletadas pelos sensores. O centro de dados consiste em um módulo de abstração e processamento de dados que executa as tarefas no computador central, obtendo dados dos sensores e depois analisando e salvando-os no banco de dados. Os dados são analisados pelo SPANS que irá detectar e indicar as vagas disponíveis no estacionamento. Depois disso, essas informações ficam acessessíveis para o aplicativo do condutor.

Um computador ficou dedicado para o centro de dados, nele foram instalados um banco de dados MySQL, o *framework* Apache CXF para criar serviços web, o Jetty como servidor web e *container servlet* em conjunto com nossos algoritmos Java.

### 3.2. Protótipo de um nó sensor sem fio

Muitas vezes os lugares que oferecem estacionamento não têm um sistema de vigilância que monitora a entrada e saída de veículos, bem como não monitoram o estacionamento em sí. Em outras palavras, esse estabelecimento não dispõe de um sistema de câmeras capaz de verificar a entrada e saída de veículos em um espaço de estacionamento particular. Assim, desenvolvemos uma solução para poder monitorar esses ambientes. Este dispositivo, além de enviar de tempos em tempos imagens do lugar monitorado, também detecta a saída e entrada de um veículo em vagas de estacionamento. Em outras palavras, o sensor captura imagens que são utilizadas para detectar mudanças no cenário monitorado, assim, possíbilitando a detectação dos movimentos em vagas de um estacionamento particular.

O nó sensor sem fio é baseado em um Raspberry Pi [Pi 2017] modelo B. Raspberry Pi possui uma interface de rede sem fio USB e uma Webcam tradicional. Além disso, o

Raspberry usou um sistema operacional Raspbian, um programa Python para ler os dados dos pinos GPIO (*General Purpose Input / Output*) da Raspberry e o programa Motion para monitorar sinais de vídeo da Webcam.

Foi escolhido o movimento porque este *software* pode ser usado como uma detecção de movimento que grava um arquivo a cada vez que há uma mudança nos quadros de imagem. Para conseguir isso, foi definido um intervalo de tempo em segundos sem nenhuma detecção de movimento que aciona o final de um evento. Assim, uma vez detectado um movimento no ambiente, o programa iria esperar por 10 segundos e então tiraria uma foto. Este intervalo de tempo foi escolhido para que fosse possível ao motorista do veículo sair ou estacionar na vaga.

### 3.3. As informações do estacionamento

A área do estacionamento está dividida em vários lugares (vagas), como exibe a Figura 2. Cada vaga de estacionamento tem as seguintes informações:

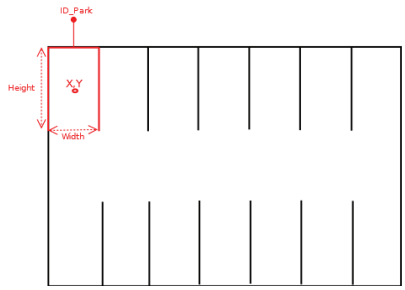


**Figura 2. Informação do estacionamento**

- ID_Park: Identificação da vaga do estacionamento;
- Desc: Descrição da vaga de estacionamento;
- X: Coordenada "X"da vaga na área do estacionamento;
- Y: Coordenada "Y"da vaga na área do estacionamento;
- Largura: Largura da vaga do estacionamento;
- Altura: Altura da vaga de estacionamento;
- Status: Armazena o *status* atual do espaço do estacionamento. 0 está disponível; 1 está ocupado;

As informações do estacionamento são armazenadas em um banco de dados. Uma vez definidas as vagas de estacionamento, utilizando uma câmera, o sistema receberá uma imagem denominada "Imagem de Base"de cada vaga/espaço do estacionamento tendo em conta as coordenadas X e Y e os valores de largura e altura. A "Imagem Base"será armazenada no banco de dados e será usada para verificar se o lugar do estacionamento está disponível ou ocupado. Inicialmente, todos os lugares do estacionamento serão considerados como mostrado na Figura 2.

### 3.4. Processamento de Imagem

Uma vez que o centro de dados receber as fotos da vaga de estacionamento, ele inicia o serviço SPANS. O SPANS realizará o processamento da imagem com a finalidade de verificar a vaga disponível que o condutor pode estacionar. Para conseguir isso, SPANS

precisa seguir alguns passos para permitir uma detecção eficiente em ambientes internos e externos. A Figura 3 descreve as etapas que o SPANS precisa realizar para poder ter a percepção de vaga de estacionamento disponível independentemente de ser em uma área fechada ou ao ar livre.

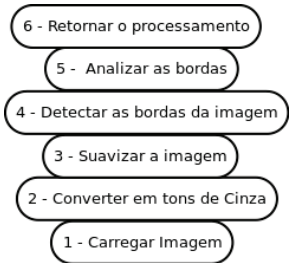


**Figura 3. Etapas que o processamento de imagem do SPANS precisa seguir.**

Assim, o primeiro passo do SPANS é carregar a imagem que está no centro de dados e que foi recebida a partir do nó sensor. Depois disso, o SPANS converterá a imagem em uma matriz MxN, onde a linha e a coluna indicam como coordenadas espaciais na imagem, a posição do pixel.

Posteriormente, o sistema irá transformar a imagem colorida para uma imagem em escala de cinza. Para conseguir isso, cada pixel da imagem é lido e uma cor de escala de cinza é calculada nesse pixel. A Figura 4 descreve um algoritmo em Java que recebe uma imagem colorida e retorna uma imagem em escala de cinza.

```java
public static BufferedImage convertToCinza(BufferedImage imgRGB) {
    // Cria um novo Buffer para BYTE GRAY
    BufferedImage img = new BufferedImage(imgRGB.getWidth(),
            imgRGB.getHeight(), BufferedImage.TYPE_BYTE_GRAY);
    WritableRaster raster = img.getRaster();
    WritableRaster rasterRGB = imgRGB.getRaster();
    // Para cada Pixel realiza a transformação para tons de cinza
    //e joga na nova imagem.
    for (int h = 0; h < 256; h++) {
        for (int w = 0; w < 256; w++) {
            int[] p = new int[4];
            rasterRGB.getPixel(w, h, p);
            p[0] = (int) (0.3 * p[0]);
            p[1] = (int) (0.59 * p[1]);
            p[2] = (int) (0.11 * p[2]);
            int y = p[0] + p[1] + p[2];
            raster.setSample(w, h, 0, y);
        }
    }
    return img;
}
```

**Figura 4. Convertendo a imagem colorida em um imagem em escala de cinza com Java.**

Após este processo, a imagem é passada através de um filtro chamado Glaussian Blur [Robert A. Hummel 1987] para suavizar a imagem. Trata-se de um filtro passa-baixo que permite passar as baixas frequências, mas irá eliminar os valores relacionados com as altas frequências. Assim, o efeito deste filtro é uma suavização da imagem, uma vez que as altas frequências que correspondem às transições abruptas são atenuadas. A suavização tende a reduzir o ruído nas imagens [M. S. Arulampalam 2002]. O Glaussian Blur usa uma função de Gaussian que possa ser descrita como:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \quad (1)$$

Quando consideramos a função em duas dimensões resulta no produto de duas equações de uma dimensão:

$$G(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

Onde $x$ é a distância da origem no eixo x, y é a distância da origem no eixo y e $\sigma$ é o desvio padrão da distribuição gaussiana. Escolhendo valores para alfa é possível obtidos *arrays* que serão usados como o *kernel* para convolver a imagem. Após a convolução, é possível verificar uma suavização da imagem.

Outra técnica utilizada é a detecção de bordas que realça contornos de borda e também amplifica o ruído de cena e, em grande parte, os operadores de borda, desfrutam algum tipo de suavização de imagem antes da operação diferencial [Canny 1986]. Neste artigo, usamos o algoritmo de Canny [P. Bao 2005] que é um primeiro operador gaussiano derivativo, suavizando o ruído e localizando as arestas. Para isso, este algoritmo utiliza três regras: (i) detecção: capacidade de localizar e marcar todas as bordas existentes; (Ii) Localização: reduzir a distância entre a borda verdadeira ea fronteira detectada; (Iii) Resposta: há apenas uma resposta para cada aresta.

Para implementar todo esse processo usamos a biblioteca OpenCV [Kaehler 2016] que fornece vários métodos para serem usados no processamento da imagem. O algoritmo converte as imagens lidas em escala de cinza (linhas 4-5) através da função cvtColor (linhas 2-3 no Algoritmo 1). Depois disso, aplica filtros para remover as imperfeições e ruídos (linhas 6-7) pela função Blur. Assim, o algoritmo detecta o número de bordas nas imagens (linhas 8-9) usando Canny e comparando-as (10-12) através do *findContours* que recuperam os contornos da imagem. Os contornos são usados para detectar os objetos nas imagens. Se o número da Imagem Atual for maior que a Imagem Base, o sistema definirá o *status* da vaga do estacionamento como ocupado. Caso contrário, o sistema define o *status* da vaga do estacionamento como disponível (linhas 13-16). Essas informações são atualizadas no banco de dados (linha 17) e são usadas pelo aplicativo móvel.

Os resultados desse algoritmo podem ser vistos na Figura 5 que exibe as imagens obtidas a partir da implementação do algoritmo proposto usando as funções OpenCV.

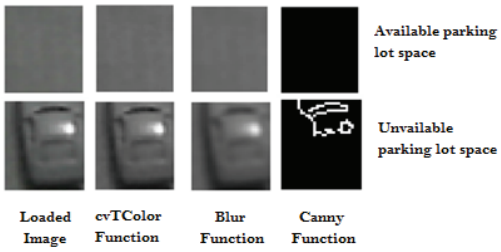


**Figura 5. Algoritmo de processamento de imagem.**

**Algorithm 1** Algoritmo de processamento de imagem.

*Read Base_Image*
*Read Current_Image*
*Transform the Base_Image to Grayscale*
*Transform the Current_Image to Grayscale*
*Apply Filter to Remove Noise (Base_Image)*
*Apply Filter to Remove Noise (Current_Image)*
*Edge Detection(Base_Image)*
*Edge Detection(Current_Image)*
*Number_Boder_base = Extract Border (Base_Image)*
*Number_Boder_Current = Extract Border (Current_Image)*
*Compare Borders (Number_Borders_base, Number_Boder_Current)*
**if** (*Number_Boder_Current* > *Number_Boder_Current* ) **then**
  *Status[ID_Park]= Busy*
**else**
  *Status[ID_Park]= Availabel*
**end if**
*Update Database*

### 3.5. Aplicativo do Condutor

Para que os motoristas tenham acesso as informações sobre as vagas de estacionamentos disponíveis perto do seu local de destino, desenvolvemos um aplicativo para a plataforma Android que realiza uma comunicação segura com o centro de dados através do protocolo SSL. Esta aplicação também tem uma tela para mostrar o mapa de espaços de estacionamento através do processamento das fotos tiradas pelos sensores, para que o usuário possa verificar a melhor vaga disponível para ele.

## 4. Caso de Uso

Nesta seção, descrevemos mais detalhes sobre o serviço proposto. O SPANS foi implementado em linguagem C ++ para manter uma compatibilidade entre os elementos que envolve o serviço. No entanto, a implementação do nó sensor sem fio também usa Python para facilitar o gerenciador do dispositivo incorporado. Além disso, o sistema usa a biblioteca OpenCV não só para implementar o processamento de imagem, mas também para criar o mecanismo para controlar as Webcams remotamente. A Figura 6 mostra o diagrama de classes do serviço SPANS no centro de dados.

A classe *Camera* foi criada para encapsular a classe do OpenCV chamada *VideoCapture* a qual fornece uma API para capturar vídeo da Webcam embutida no nó sensor, além de permitir que o SPANS leia arquivos de vídeo e seqüências de imagens no banco de dados ou em qualquer outro sistema de monitoramento. Assim, a câmera de classe é derivada da classe *VideoCapture* que define métodos para acessar as imagens a partir do banco de dados.

A classe *CapThread* é uma das mais importantes do aplicativo, pois é ela que executa todos os pesos de processamento de comparação entre as imagens para descobrir se a vaga monitorada está livre ou não. A renderização compara todas as imagens vazias

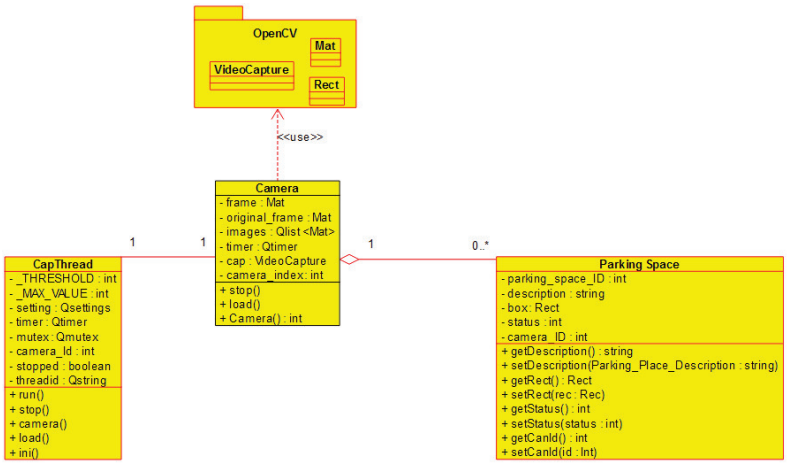


**Figura 6. Diagrama de classes usando UML do SPANS.**

que foram salvas no repositório local com a imagem atual extraída da moldura que é constantemente atualizada pela câmera. Ao instanciar um novo *CapThread* ele passa como uma instância de uma classe *Camera* em seu construtor de lá o *CapThread* terá acesso a todas as vagas que foram carregadas na classe *Camera*.

Para que a infra-estrutura tenha uma boa eficiência, inicialmente é tirada uma foto do estacionamento. Esta fotografia é solicitada apenas uma vez para registar as coordenadas de cada espaço de estacionamento, além disso esta imagem será utilizada como a imagem predefinida (Base_Image) desse estabelecimento. Assim, a infra-estrutura pode subsequentemente realizar a detecção de vagas de estacionamento disponíveis. A Figura 7 mostra este primeiro estágio do sistema, no qual há uma vaga registrada no sistema, representada pelo quadrado amarelo. O registo das vagas pode ser efetuado pelo proprietário do estabelecimento.

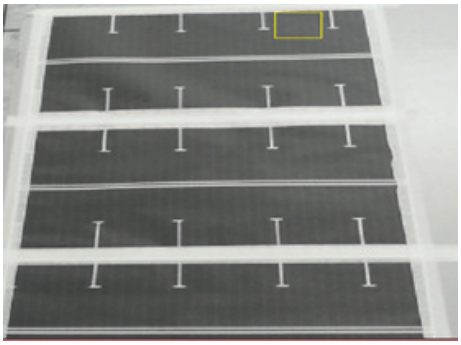

**Figura 7. Maquete da área de estacionamento do protótipo.**

Uma vez definidos as vagas de estacionamento, o sistema inicia o monitoramento delas através das novas imagens provenientes dos nós de sensores. O centro de dados usará a imagem *Base_Image* e a nova imagem obtida em tempo real do sensor para detectar se a vaga do estacionamento está disponível. O serviço desenha um retângulo verde em todos os lugares disponíveis. Se o sistema detecta que uma vaga foi ocupada, ele atualiza a informação de *status* dessa vaga no banco de dados e desenha um retângulo vermelho nesse lugar, como pode ser visto na Figura 8.

**Figura 8. Exemplo do sistema em execução.**

As informações armazenadas no banco de dados sobre as vagas de um determinado estacionamento podem ser acessadas pelo aplicativo móvel usando uma interface amigável. A Figura 9 ilustra essa interface que é exibida aos usuários. O aplicativo móvel tem uma interface de grade onde mostra o *status* atual de cada vaga em um estacionamento definido no sistema.


**Figura 9. Interface do aplicativo móvel.**

### 4.1. Resultados

Para avaliar o serviço SPANS, foi utilizada a infra-estrutura desenvolvida para fazer testes com um modelo em escala reduzida (maquete) da área de estacionamento onde foram definidos 20 lugares de estacionamento. Os lugares de vagas de estacionamento foram inseridos no banco de dados com o *status* inicial como sendo disponível. Para isso, criamos um modelo que representa o estacionamento da Instituição Federal de São Paulo, Catanduva, Brasil. A Figura 10 mostra o resultado do processamento de imagem levando em consideração o modelo em escala usado nos testes.

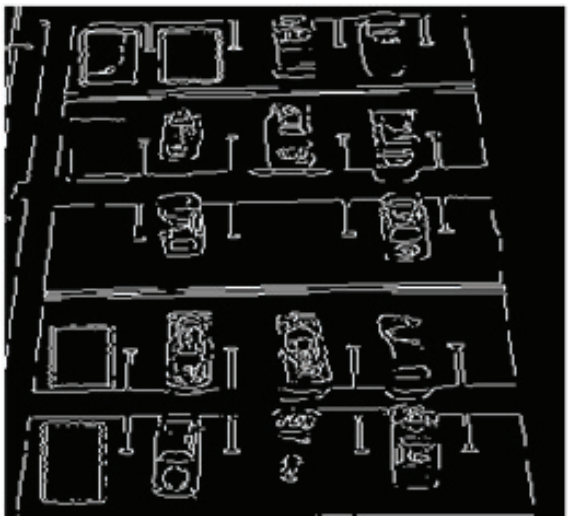

**Figura 10. Modelo em escala reduzida da área de estacionamento após o processamento da imagem.**

Este teste avalia o tempo que o serviço terá para processar as imagens com base no número de vagas existentes no estacionamento. Portanto, o objetivo é verificar o tempo gasto para encontrar as vagas de estacionamento disponíveis. A Figura 11 mostra o tempo de processamento gasto para encontrar tais vagas disponíveis.

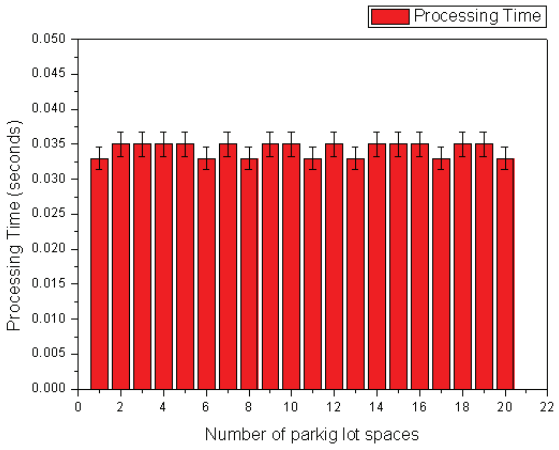


**Figura 11. Modelo de escala da área de estacionamento após o processamento da imagem**

Como pode ser visto na Figura 11, o tempo médio para encontrar um lugar disponível na área do estacionamento é de 0.036 segundos. Isto significa que o tempo gasto para encontrar as vagas disponíveis é proporcional a área do estacionamento. Assim, o SPANS pode gerenciar lugares de estacionamento sem prejudicar outros serviços que podem ser executados no sistema de transporte inteligente.

## 5. Conclusão

Neste artigo foi proposto um serviço que detecta e gerencia vagas em um estacionamento. O serviço proposto é baseado no processamento de imagens para detectar vagas disponí-

veis em um estacionamento de forma automática para uma arquitetura de transporte inteligente que visa evitar o tráfego pesado em áreas que oferecem vagas de estacionamento. Assim, torna-se possível reduzir o tempo de busca por uma vaga livre e consequentemente o consumo de combustível e a emissão de poluentes. Além disso, neste artigo também foram apresentados detalhes que mostram como desenvolver a solução proposta, desde os recursos necessários para a criação do nó sensor até um exemplo de caso de uso que teve o seu desempenho avaliado. Os resultados mostraram que o SPANS pode gerenciar lugares de estacionamento sem prejudicar outros serviços que podem ser realizados no sistema de transporte inteligente.

Como trabalhos futuros, será desenvolvido um submódulo para que os proprietários de estacionamentos possam servir com um nó *skink* no qual as informações de seus nós sensores sejam pré processadas, facilitando a gestão do seu estabelecimento. Este submódulo também permitirá uma diminuição na utilização da largura banda utilizada na comunicação entre os nós e o centro de dados, uma vez que o submódulo só precisará enviar um arquivo de texto ao invés da imagem e isso somente quando houver a indicação de vagas disponíveis. Acrescenta-se ainda a intenção de difundir esta estrutura não somente na universidade, mas também em outros estacionamentos da cidade, em especial, nos estacionamentos com grande números de vagas a fim de avaliar o desempenho do SPANS nesses cenários. Dessa forma, pretende-se ainda realizar testes que avaliem as questões de capacidade de comunicação da rede em cenários com vários estacionamentos distribuídos geograficamente.

## 6. Agradecimento

Os autores agradecem a Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP (processo número 2015/11536-4) pelo apoio financeiro.

## Referências

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698.

Fulya Yuksel Ersoy, K. H. e. E. I. (2016). Parking as a loss leader at shopping malls. *Transportation Research Part B: Methodological*, 91:98 – 112.

Huang, Shizhen e Fu, H. (2016). Design of embedded parking management system. In *Communication Problem-Solving (ICCP), 2016 International Conference On*, pages 1–2. IEEE.

J. Li, Y. An, R. F. e. H. W. (2016). Smartphone based car-searching system for large parking lot. In *2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA)*, pages 1994–1998.

Jung, J. K. S. H. G. (2016). Automatic parking space detection and tracking for underground and indoor environments. *IEEE Transactions on Industrial Electronics*, 63(9):5687–5698.

Kaehler, Adrian e Bradski, G. (2016). *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. "O'Reilly Media, Inc.".

M. S. Arulampalam, S. Maskell, N. G. e. T. C. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

Meneguette, Rodolfo I., F. G. P. R. G. D. L. P. G. V. L. A. e. U. J. (2016a). Increasing intelligence in inter-vehicle communications to reduce traffic congestions: Experiments in urban and highway environments. *PLOS ONE*, 11(8):1–25.

Meneguette, R. I. (2016b). A vehicular cloud-based framework for the intelligent transport management of big cities. *International Journal of Distributed Sensor Networks*, 12(5):8198597.

Mohr, Detlev, M. N. K. A. G. P. K. H. K. A. e. H. R. (2013). The road to 2020 and beyond: what's driving the global automotive industry? *McKinsey&-Company (Pub.), Automotive & Assembly–Latest thinking, available online at http://www. mckinsey. com/˜/media/McKinsey/dotcom/client_service/Automotive% 20and% 0 Assembly/PDFs/McK_The_road_to_2020_and_beyond. ashx, accessed*, 28(3):2014.

P. Bao, L. Z. e. X. W. (2005). Canny edge detection enhancement by scale multiplication. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1485–1490.

P. Sheelarani, S. P. Anand, S. S. e. K. S. (2016). Effective car parking reservation system based on internet of things technologies. In *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*, pages 1–4.

Pi, R. (2017). Raspberry pi. Disponível em: https://www.raspberrypi.org/. Acessado em 2017.

R. E. Barone, T. Giuffre, S. M. S. M. A. M. e. G. T. (2014). Architecture for parking management in smart cities. *IET Intelligent Transport Systems*, 8(5):445–452.

Robert A. Hummel, B. K. e. S. W. Z. (1987). Deblurring gaussian blur. *Computer Vision, Graphics, and Image Processing*, 38(1):66 – 80.

Suryady, Zeldi, S. G. R. H. S. S. M. T. e. E. M. F. M. (2014). Rapid development of smart parking system with cloud-based platforms. In *Information and Communication Technology for The Muslim World (ICT4M), 2014 The 5th International Conference on*, pages 1–6. IEEE.

T. N. Pham, M. F. Tsai, D. B. N. C. R. D. e. D. J. D. (2015). A cloud-based smart-parking system based on internet-of-things technologies. *IEEE Access*, 3:1581–1591.

Yeh, Her-Tyan, C. B.-C. e. W. B.-X. (2016). A city parking integration system combined with cloud computing technologies and smart mobile devices. *Eurasia Journal of Mathematics, Science & Technology Education*, 12(5):1231–1242.