

VISÃO COMPUTACIONAL: SISTEMA PARA DETECÇÃO DE VAGAS DISPONÍVEIS EM UM ESTACIONAMENTO COM VAGAS PRÉ-DETERMINADAS

COMPUTER VISION: SYSTEM FOR DETECTION OF VACANCIES AVAILABLE IN A PARKING WITH PRE-DETERMINED PLACES

Gustavo Henrique Silva Amorim¹
Marcos Vinicius Mariano da Silva¹
Dheyson Wildny Cruz Souza²

RESUMO

Hoje em dia, com o grande crescimento da frota de veículos no país, de alguns anos para cá, consequentemente, as ruas tem se tornado pequenas para tantos veículos, ficando cada vez mais difícil de encontrar vagas em determinados locais, tais como um shopping center. Visando aplicar a visão computacional, demonstrar seus conceitos em problemas do dia a dia e melhorar o conforto dos condutores de veículos, este trabalho tem como propósito desenvolver um sistema que detecta se determinada vaga de um estacionamento está disponível ou não, mostrando seu status através de um aplicativo Android. O sistema conta com o uso de visão computacional, uma tecnologia bastante avançada, que usa uma câmera para fazer o monitoramento da vaga, junto a uma aplicação Java que utiliza a biblioteca OpenCV. Ele reconhece um automóvel e muda o status do banco de dados através de um webservice que está localizado na Amazon EC2. A aplicação mobile lê as informações do banco de dados e mostra para o usuário se a vaga está disponível ou ocupada.

Palavras-chave: Visão Computacional, OpenCV, Android

ABSTRACT

Nowadays, with the great growth of the vehicle fleet in the country, a few years here, therefore, the streets are becoming small for so many vehicles, becoming increasingly difficult to find vacancies in certain places, such as a shopping mall. Aiming to apply computer vision, demonstrate their concepts in problems of everyday life and improve the comfort of vehicle drivers, this work aims to develop a system that detects whether a vacancy of a parking lot is available or not, showing their status through an Android application. The system includes the use of computer vision, a very advanced technology, which uses a camera to monitor the vacancy, along with a Java application that uses the OpenCV library. It recognizes a car and change the status of the database via a webservice which is located in the Amazon EC2. The mobile application reads the database information and shows to the user if the vacancy is available or busy.

Keywords: Computer Vision, OpenCV, Android

¹ Graduandos de 2016 do curso de Ciência da Computação da Universidade de Franca.

² Professor orientador da Universidade de Franca.

INTRODUÇÃO

O tema visão computacional vem crescendo devido ao aumento da disponibilidade de recursos que possibilitam sua aplicação para resolução de problemas atuais (RIOS, 2014).

Utilizamos a biblioteca OpenCV que conta com uma grande extensão de códigos abertos e que podem ser utilizados para a captura e processamento de imagens. Esta biblioteca possui vários algoritmos que possibilitam a criação de aplicações, que podem ser utilizados para pegar informações de uma imagem ou vídeo. Contudo, com o carregamento de imagens e vídeos é possível conseguir uma melhoria de resultado com os tratamentos realizados. Por estes motivos e também por conta de sua acessibilidade, foi escolhido esta biblioteca para se trabalhar no projeto desenvolvido.

Segundo o site de notícias da rede globo G1, atualmente o crescimento da frota de veículos vem aumentando nos últimos anos, segundo dados e estatísticas do mesmo site, a estimativa é de 1 automóvel a cada 4 habitantes, e, junto com o crescimento vem a dificuldade para encontrar vagas em estacionamentos fechados como os de shoppings, lojas e até mesmo estacionamentos particulares por exemplo. E nestes casos, muito das vezes o motorista perde muito do seu tempo procurando por vagas (G1, 2014).

Com base nisto, o projeto desenvolvido tem em vista mostrar alguns conceitos de visão computacional e auxiliar os motoristas, fazendo com que o problema citado acima seja amenizado através de um protótipo de um sistema de visão computacional, que faz a identificação de quais vagas estão disponíveis ou não em determinado estacionamento.

Vários fatores, como: condições ambientais e geométricas utilizadas na captura de medidas, são fatores importantes para um melhor aproveitamento do sistema.

1 TECNOLOGIAS UTILIZADAS

Neste tópico, é apresentado um estudo sobre as tecnologias utilizadas para o desenvolvimento do projeto proposto.

Foram realizadas pesquisas para definir as melhores condições para a detecção dos automóveis, processamento das informações, localização e o envio do status da vaga, que pode ser de “vago” ou “ocupado”, para o banco de dados, onde a aplicação mobile faz a consulta destes dados.

O QUE É VISÃO COMPUTACIONAL

A visão computacional procura imitar a visão humana, portanto também possui como entrada uma imagem, porém, a saída é uma interpretação desta imagem.

Segundo Rios, Visão computacional é o estudo da extração de informação de uma imagem; mais especificamente, é a construção de descrições explícitas e claras dos objetos em uma imagem. É diferente do processamento de imagens pois, enquanto ele se trata somente da transformação de imagens em outras imagens, a visão computacional tem como foco obter informações das imagens processadas e a partir deste ponto tomar alguma ação (RIOS, 2014).

A visão computacional utiliza um processo de segmentação; este processo constitui-se em dividir uma imagem em regiões, ou objetos distintos. Existem vários modos que podemos utilizar para segmentar, lembrando que o artigo não tem como objetivo de esgotar estas técnicas, mas sim, mostrar algumas através do OpenCV.

A segmentação se classifica em três grupos: Segmentação por detecção de borda; Segmentação por corte; Segmentação por crescimento de região (MARENGONI e STRINGHINI, 2009).

OPENCV

Segundo o próprio site da OpenCV, é uma biblioteca open source relacionada a visão computacional e ao aprendizado de máquina. Atualmente possui mais de 2500 algoritmos. Esses algoritmos podem ser usados para reconhecimento de faces, reconhecimento de objetos e classificar ações humanas em vídeos. Foi desenvolvida pela Intel com o intuito de tornar a visão computacional aberta aos usuários e programadores em diversas áreas. A biblioteca é de código livre, com o código fonte e os executáveis otimizados para os processadores Intel. A biblioteca OpenCV tem interfaces que funcionam nas plataformas Java, C++, C, Python e MATLAB. Atualmente é suportado pelos sistemas operacionais Windows, Linux, Android e Mac OS (OPENCV, 2016).

Juntamente com o pacote OpenCV, é disponibilizado a biblioteca IPL (Image Processing Library), cujo o OpenCV necessita parcialmente, além de documentação e alguns exemplos de códigos fonte de como utilizar determinadas funções. A biblioteca é dividida em cinco grupos de funções: Processamento de imagens; Análise estrutural; Análise de movimento e rastreamento de objetos; Reconhecimento de padrões e Calibração de câmera (BRADISK, 2000).

LINGUAGEM JAVA

Segundo LUCKOW e MELO (2010), é a linguagem de programação padrão utilizada pela plataforma Java, e começou a surgir em 1991 na Sun Microsystem. Era parte de outro projeto, chamado Green Project, que tinha como objetivo possibilitar a convergência entre computador, equipamentos eletrônicos e eletrodomésticos. E anteriormente a linguagem era chamada de Oak (carvalho, em português).

Depois do Green Project, James Gosling ficou responsável por adaptar a linguagem para internet em 1995, surgindo assim a plataforma Java.

Um dos principais diferenciais da plataforma, é de que Java é executado sobre uma JVM, uma máquina virtual. E qualquer plataforma de hardware ou equipamento eletrônico que possa executar uma máquina virtual conseguira executar Java. O que justifica o slogan “write once, run anywhere”, em português “escreva uma vez, execute em qualquer lugar”.

E como destaca o site de notícias tecnológicas CANALTECH (2016), Java é a linguagem de programação mais utilizada no mundo atualmente.

BANCO DE DADOS

Atualmente, tem um impacto importante sobre o uso crescente dos computadores. Sua definição básica é uma coleção de dados relacionados.

Um banco de dados pode ser criado manualmente, como por exemplo um catálogo, ou, gerado por um computador. Seu propósito é armazenar e recuperar informações relacionadas.

Os SGBD's são sistemas gerenciadores de banco de dados. São softwares que permitem criar e gerenciar um banco de dados, podendo manipular e compartilhar estes dados para diversos usuários e aplicações (ELMASRI; NAVATHE, 2011).

MYSQL

Segundo a ORACLE, detentora do MySQL atualmente, é um banco de dados relacional de código aberto mais utilizado do mundo. Esse banco de dados foi desenvolvido em 1995, pela empresa MySQL AB, mas atualmente ela pertence a Oracle Corporation.

Como já está composta no próprio nome, esse banco de dados utiliza a linguagem SQL para fazer as funções do banco, como, inserir, alterar, consultar e listar; além de gerenciar o conteúdo armazenado no mesmo (CABRAL et al, 2012).

ANDROID

A empresa Google afirma que a primeira versão do Android™ foi lançado em setembro de 2008 e desde então está sempre atualizando suas versões. Em 2013, a marca já representava aproximadamente, 79,3% do mercado de telefonia móvel no mundo. Que demonstra que o Sistema é predominante no mercado (CORREIO DO ESTADO, 2013).

Segundo o autor LECHETA (2010), Android™ é um sistema operacional (SO) voltado para dispositivos moveis baseado no Kernel do Linux. Foi desenvolvido pela empresa de tecnologias Google em parceria com a empresa OHA, que é composta por várias empresas de tecnologia móvel.

O Android™ em sua estrutura é composta por um conjunto de programas, que trabalham em conjunto para produzir um resultado ou alcançar um objetivo comum, também chamado de software stack, que é subdividida em cinco camadas (ANDROID DEVELOPERS, 2013).



Figura 1 – Arquitetura do sistema Android.

Fonte: Adaptado do Google (2016).

WEBSERVICE

Web Services é uma tecnologia que tem como principal objetivo integrar sistemas distintos utilizando de seus protocolos padronizados que garantem a independência de plataforma e também de linguagem de programação que determinados sistemas foram escritos. Utilizando um webservice, é possível disponibilizar métodos em um servidor remoto e permitir que eles sejam acessados pela aplicação cliente (NETO, 2004).

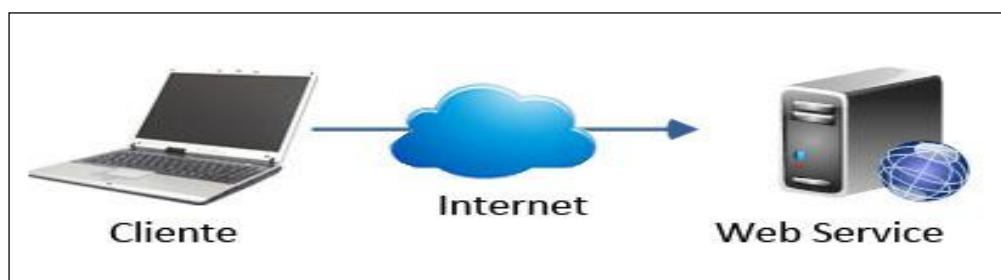


Figura 2 – Cliente chamando um serviço web através da internet.

Fonte: BORGES (2013).

REST

Representational State Transfer, ou ainda Transferencia de Estado Representacional, pode ser considerado uma abstração da arquitetura Web, uma arquitetura que se constitui de um conjunto ordenado de restrições arquiteturais aplicadas a componentes, conectores e elementos dentro de um sistema distribuído. Os princípios de REST se baseiam em um protocolo cliente/servidor, onde cada mensagem, utilizando o protocolo HTTP, contém toda a informação necessária para atender a aplicação que a chamou. A principal vantagem, é de que nem o cliente e nem o servidor necessitam de gravar os estados das comunicações entre as mensagens (FIELDING, 2000).

As principais operações do REST são:

GET	<ul style="list-style-type: none">• Buscar recursos• Cache
POST	<ul style="list-style-type: none">• Criar um novo recurso
PUT	<ul style="list-style-type: none">• Atualizar um recurso existente
DELETE	<ul style="list-style-type: none">• Remover um recurso

Figura 3 - Operações de REST.

Fonte: Adaptado de MSDN (2009).

JSON

JavaScript Object Notation é um dos modelos utilizados na metodologia REST para armazenamento e transmissão de dados no formato de texto. Apesar de ser muito simples, é utilizada frequentemente devido a capacidade de compactar as informações nela passadas, melhor do que o XML, tornando mais rápida a conversão destas informações. A representação do JSON é extremamente simples (CORREIA).

No projeto, uma consulta no banco de dados convertidos para JSON, através de um webservice hospedado em nuvem, que será acessado pelo aplicativo Android fica neste formato:

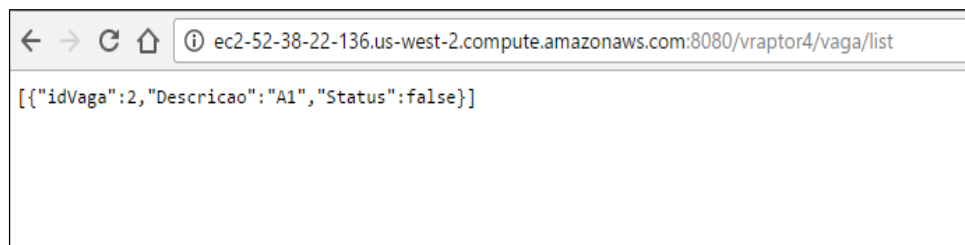


Figura 4 - Utilizando operação GET e obtendo como resposta um JSON.

Fonte: O autor.

APACHE TOMCAT

Segundo LUCKOW e MELO (2010), a Apache foi quem criou e desenvolveu o produto, sendo esta ferramenta, uma ferramenta open source. O Tomcat é um servidor web e contêiner Java ao mesmo tempo, e ele suporta as execuções de tecnologias voltadas para a web, como Servlets, JSP e Webservices, o que permite que o Java funcione em um ambiente web.

Embora o Tomcat seja bem robusto, ele permite a integração com servidores como Apache HTTP e IIS da Microsoft, o que pode fornecer uma capacidade ainda maior de trabalho.

AMAZON EC2

Segundo a própria AMAZON (2016), criadora da Amazon Elastic Compute Cloud (Amazon EC2), é um serviço web que fornece capacidade de computação redimensionável em nuvem. Foi projetado para facilitar a computação em nuvem na escala web para os desenvolvedores.

Tem uma interface de serviço simples, onde permite que se obtenha e possa ser configurado uma instância e suas capacidades com o mínimo de atrito. Ela oferece um controle completo dos recursos utilizados. Na Amazon EC2, é permitido que se faça o escalonamento de determinadas capacidades das instancias de acordo com a necessidade de forma rápida e segura. Ao criar uma instancia, você só será cobrado de acordo com o uso. E fornece ferramentas para que possam ser construídos aplicativos resistentes a falhas.

2 O PROJETO

O projeto desenvolvido tem como objetivo demonstrar conceitos de visão computacional, fazendo um protótipo de um sistema de estacionamento que exemplifique a utilização da mesma utilizando a biblioteca OpenCV. Para isso, foi desenvolvido um sistema em Java que faz o reconhecimento de um veículo através de uma simples câmera utilizando a biblioteca citada. A aplicação que faz o reconhecimento dos veículos, assim que um automóvel é identificado, manda a atualização para um servidor que está hospedado na Amazon EC2 atualizando o status da vaga que foi cadastrada previamente no banco de dados, em seguida o aplicativo Android faz busca dos dados atualizados da vaga, indicando se a vaga está disponível ou não.

Para ser feito isto, foi utilizado a metodologia a baixo descrita neste tópico.

DESENVOLVIMENTO DA APLICAÇÃO DETECTORA DE VEÍCULOS

Segundo pesquisa do site de notícias tecnológicas CANALTECH (2016) a linguagem Java é a mais utilizada mundialmente. Este status do Java se dá por conta de que ela não é somente uma linguagem de programação, mas também é uma plataforma de desenvolvimento. Com a necessidade de um melhor desempenho e por conta de rodar em qualquer máquina que suporte máquinas virtuais (LUCKOW; MELO, 2010), e olhando para o lado prático/rápido, levando em conta que esta linguagem é a de maior domínio do autor, chegou-se à conclusão de que Java seria a melhor opção para o desenvolvimento da aplicação.

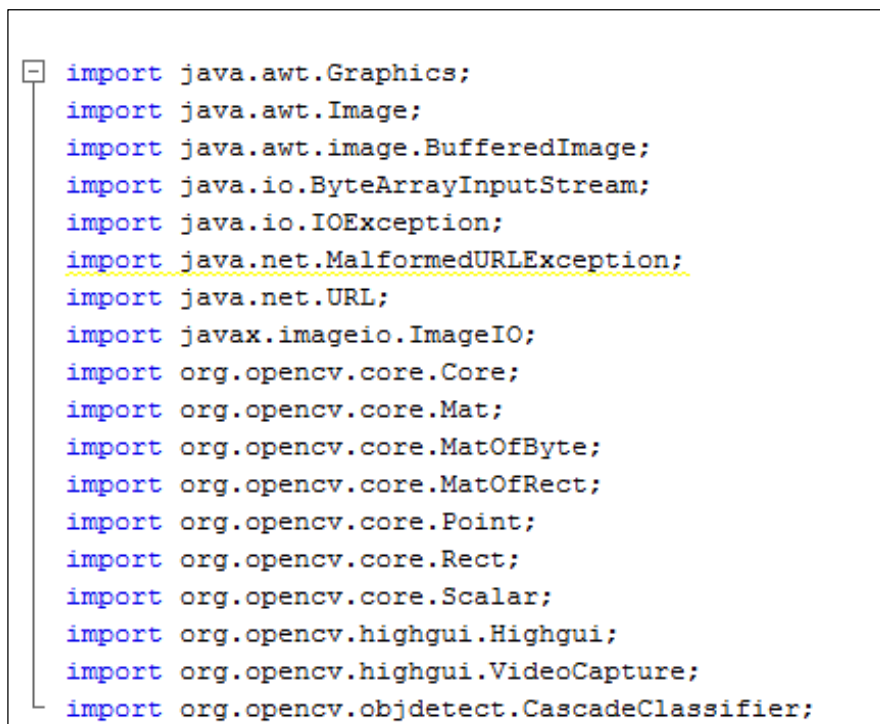


Figura 5 - A aplicação identificando um carro.

Fonte: O autor.

IMPORTANDO AS BIBLIOTECAS

O aplicativo, tem como objetivo identificar um carro através de uma câmera conectada ao computador onde ele está sendo executado. Porém antes de começar a ser desenvolvido, é necessário que seja importado no projeto a biblioteca OpenCV, para que suas classes sejam utilizadas. É necessário também a importação da biblioteca que faz manipulação de URL's, pois através desta biblioteca que o webservice é acessado, fazendo com que o banco de dados seja atualizado com o status atual da vaga, se é de “true” para quando está ocupado e “false” para quando está disponível. A biblioteca awt, se refere a construção da parte gráfica do Java, ou seja, as janelas e botões utilizados pelo sistema (OPENCV e LUCKOW; MELO).

A screenshot of a code editor showing a list of Java import statements. The code is as follows:

```
import java.awt.Graphics;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import javax.imageio.ImageIO;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.MatOfByte;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.highgui.Highgui;
import org.opencv.highgui.VideoCapture;
import org.opencv.objdetect.CascadeClassifier;
```

The code is displayed in a monospaced font with syntax highlighting: keywords like 'import' are in blue, and package/class names are in black. A small icon of a document with a minus sign is visible in the top left corner of the code block.

Figura 6 - Bibliotecas importadas na aplicação.

Fonte: O autor.

ACIONANDO A CAMERA PELO JAVA E COMPARANDO OS FRAMES

Para o acompanhamento da movimentação que ocorre na vaga, para fazer o reconhecimento do veículo, foi utilizado uma câmera. A câmera utilizada é a própria câmera padrão do computador, uma webcam. A câmera utilizada deve ser definida como padrão no sistema operacional. Juntamente do recurso de visão computacional utilizando a biblioteca OpenCV, a câmera faz o reconhecimento do automóvel através do software que desenvolvido na linguagem Java, juntamente com a ferramenta de desenvolvimento Netbeans.

Segundo ALBUQUERQUE (2012), do Centro Brasileiro de Pesquisas Físicas, a qualidade da câmera influencia na análise da imagem, ou seja, quanto melhor a câmera, melhor será a precisão na identificação de um veículo.

Para que a câmera seja iniciada pela aplicação, é necessário que seja instanciado a classe Java VideoCapture, responsável por captura de imagens e vídeos. Classe esta derivada do pacote OpenCV importado pela aplicação (OPENCV, 2016).

```
int count = 0;  
VideoCapture webSource = null;  
Mat frame = new Mat();  
MatOfByte mem = new MatOfByte();
```

Figura 7 - Instanciando a captura de vídeo no Java.

Fonte: O autor.

Após ser instanciado, o sistema fara uma comparação, para verificar se existem um próximo frame de imagem. Caso existe, ele fará outra comparação, para verificar se o frame corresponde a um carro. Os parâmetros que correspondem ao carro, que serão analisados, estão contidos dentro de um arquivo XML chamado de Hogcascade.

```
Mat frame = new Mat();  
MatOfByte mem = new MatOfByte();  
// pega o arquivo xml que contem os parametros para identificacão de um carro  
CascadeClassifier carDetector = new CascadeClassifier(CarDetection.class.getResource("hogcascade_cars_sideview.xml").getPath().substring(1));  
MatOfRect carDetections = new MatOfRect();  
///  
// Este Runnable implementa Runnable /
```

Figura 8 - Instancia do Hogcascade.

Fonte: O autor.

```
if (webSource.grab()) { //Verifica se há um próximo Frame  
    try {  
        webSource.retrieve(frame);  
        Graphics g = jPanel1.getGraphics();  
        carDetector.detectMultiScale(frame, carDetections); // Compara o frame com os parametro necessarios para identificar um veiculo
```

Figura 9 - Comparação de Frames.

Fonte: O autor.

RECONHECENDO UM VEÍCULO

A aplicação, no momento em que um veículo é reconhecido, após realizar a comparação dos frames, é criado um retângulo em volta do objeto localizado. Após ser criado o retângulo entorno do objeto, é feita a atualização no banco de dados através de uma URL, que foi criada pelo webservice, que contém a função de update, alterando o estado da vaga de vazio para ocupado.

```
for (Rect rect : carDetections.toArray()) {  
    Core.rectangle(frame, new Point(rect.x, rect.y), new Point(rect.x + rect.width, rect.y + rect.height),  
        new Scalar(0,255,0));  
}
```

Figura 10 - Criando retângulo entorno do objeto

Fonte: O autor.

Nas imagens a seguir, pode-se ver o banco de dados quando não há um veículo presente, onde contém o valor de 0 para o status e quando há um veículo presente é alterado para 1.

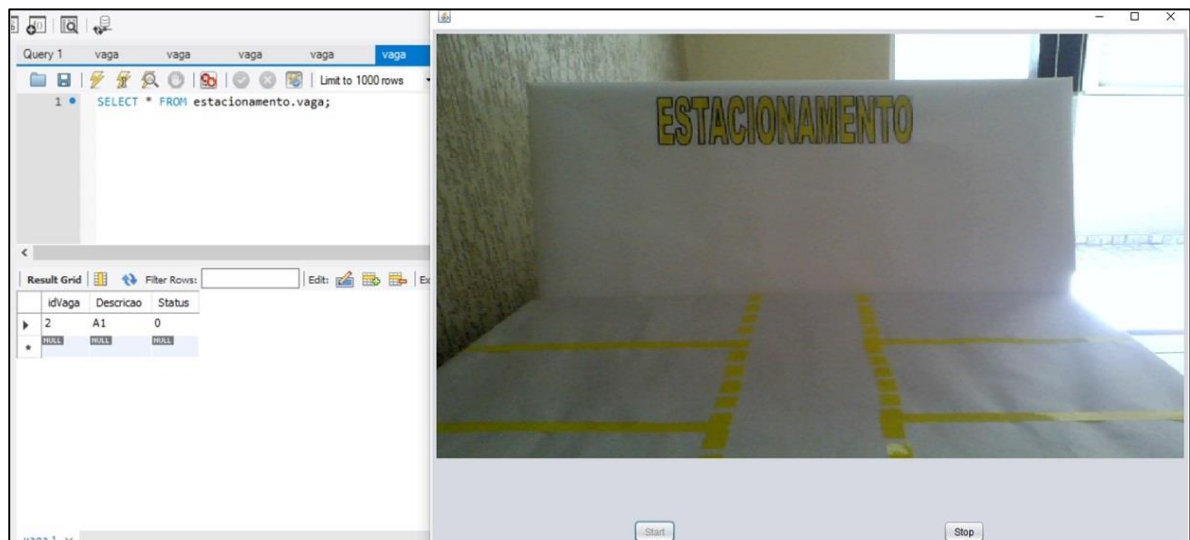


Figura 11 - Banco de dados sem veículo identificado.

Fonte: O autor.

INTEGRAÇÃO ENTRE APLICAÇÕES UTILIZANDO UM WEBSERVICE

O webservice é utilizado para fazer a integração entre as aplicações mobile e aplicação Java que faz a detecção de carros (NETO, 2004). O webservice faz conexão direta com o banco de dados, e ambos estão hospedados em uma instancia de uma máquina em nuvem, na Amazon EC2. A função deste webservice é fornecer uma URL, que será acessada pelo aplicativo Android, que será retornado um JSON com todos os dados da tabela “vaga”, utilizando a operação de GET do REST, que é utilizada quando é necessário realizar uma consulta no banco de dados. Já na aplicação que faz a detecção de carros, para fazer a atualização desta tabela alterando o atributo “status”, é utilizado a operação de POST do REST, que é utilizado quando há a necessidade de realizar uma operação de “update” no banco de dados.

ARMAZENAMENTO E COMPOSIÇÃO DOS DADOS

O banco de dados também hospedado em nuvem, na Amazon EC2, é um MySQL, conforme já citado. Ele é composto por somente uma tabela chamada “vaga”, que está dentro de um schema chamado “estacionamento”, onde nela será armazenada as vagas do estacionamento. A tabela contém os campos “Id”, como chave primaria pois não pode se repetir e é assim que se diferencia uma vaga da outra, um campo “Descrição”, para indicar o nome da vaga, onde este nome já identifica a localização da vaga dentro do estacionamento como por exemplo “A1”, mostrando que a vaga pertence a fileira A na posição 1, e por fim o campo “Status”, que será alterado conforme o sistema identifique a presença de um carro no local da vaga.

APLICAÇÃO MOBILE

Para o desenvolvimento foi utilizado a linguagem Java, como citado anteriormente. Junto com a linguagem, foi utilizado a ferramenta Android Studio. Ferramenta específica para desenvolvimento de aplicativos para a plataforma Android LECHETA (2013).

O aplicativo desenvolvido faz a busca do status da vaga do estacionamento utilizando o webservice, assim o usuário que utiliza o aplicativo conseguirá identificar qual vaga está disponível. A aplicação mostra somente as vagas do estacionamento, deixando a vaga da cor verde se estiver disponível e vermelho para indisponível.

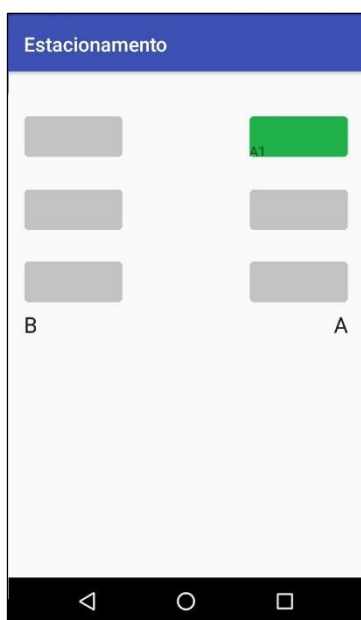


Figura 12: Vaga disponível.
Fonte: O autor.

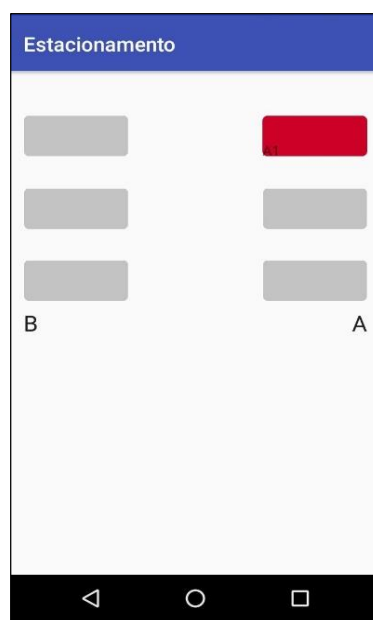


Figura 13: Vaga indisponível.
Fonte: O autor.

3 DESENVOLVIMENTOS FUTUROS

Para futuros estudos e complementação deste projeto, pretende-se fazer que com uma única câmera, seja possível fazer o monitoramento de mais de uma vaga, onde feito um estudo prévio, pode ser utilizado para diferenciar uma vaga da outra, números, que serão desenhados nestas vagas e distribuídos sequencialmente, onde cada vaga terá seu número, e a aplicação através da câmera conseguira identificar estes números e caso algum número dos cadastrados e escritos na vaga esteja encoberto por um carro, significará que a vaga estará ocupada.

Utilizando visão computacional também, pretende-se que seja identificado o momento em que o carro estiver entrando e saindo da vaga, para que seja registrado e mostrado ao usuário estas ações a fins de aumentar a segurança do local.

No webservice, que é quem faz a conexão direta com o banco de dados, também pretende-se aumentar a segurança de acesso, para que os dados não sejam acessados por estranhos.

Realizando algumas destas melhoras no sistema e abrangendo os estudos na área de visão computacional, tem-se a intenção que este sistema seja implantado primeiramente para homologação.

CONCLUSÃO

No decorrer do trabalho foram levantadas várias informações para o entendimento e construção do sistema onde foi obtido um conhecimento sobre o tema visão computacional que não era possuído anteriormente. Nas pesquisas feitas, mostram que com o aumento da disponibilidade de recursos, possibilita o uso de visão computacional na resolução de problemas atuais.

Utilizando a biblioteca OpenCV junto com a linguagem Java e uma webcam instalada no computador onde está sendo executada a aplicação, foi possível fazer a detecção de um carro, alterar o status da vaga no banco de dados quando o carro é identificado e mostrar ao usuário o status atual da vaga através de um aplicativo.

A respeito do webservice se mostrou ser uma ferramenta de grande valia no projeto, pois, tanto na aplicação que faz a detecção de carros, quanto no aplicativo para Android, não foi necessário fazer uma conexão direta com o banco de dados, fazendo com que o delay (atraso) na aplicação que faz a detecção de carros fosse diminuído quando realizado a operação de update, utilizando REST, alterando o status da vaga. Além de ter sido obtido um maior conhecimento sobre o assunto.

REFERÊNCIAS

ALBUQUERQUE, Marcio P. et al. **Análise de Imagens e Visão Computacional**, Centro Brasileiro de Pesquisas Físicas, p.31, 23 jun, 2012. Disponível em: <<http://mesonpi.cat.cbpf.br/e2012/arquivos/g06/CursoE2012-PI.pdf>>. Acesso em: 11 out.2016.

AMAZON WEB SERVICES. **Amazon ec2 – hospedagem de servidor virtual**. Disponível em: <<https://aws.amazon.com/pt/ec2/>>. Acesso em: 28 mai. 2016.

ANDROID DEVELOPERS. **Api 23 now available for android**. Disponível em: <<http://android-developers.blogspot.com.br/2015/11/api-23-sdk-now-available-for-android.html>>. Acesso em: 01 mai. 2016.

ANDROID DEVELOPERS. **The developer's guide** [2013]. Disponível em: <<http://developer.android.com/guide/index.html>>. Acesso em: 01 de mai. 2016.

BRADISK, Gary. **The opencv library**. Disponível em: <<http://www.drdobbs.com/open-source/the-opencv-library/184404319?pgno=1>>. Acesso em: 25 mar. 2016.

BORGES, Diego G. **Sistema de informação para registro de informações geográficas utilizando Android e GWT**. Universidade de Franca, Franca, p.13-70, nov. 2013.

CABRAL, Carlos A.; BRITO, Humberto R.; SOUZA, Robson Barbosa. **SGBD MySQL**. Centro Paula Souza, p.10. Disponível em:< <http://www.trabalhosfeitos.com/ensaios/Artigo-Mysql/360208.html> >. Acesso em 12 ago. 2016.

CANALTECH. **Java é a linguagem de programação mais utilizada no mundo**. Disponível em: <<https://canaltech.com.br/noticia/programacao/java-e-a-linguagem-de-programacao-mais-utilizada-no-mundo-55819/>>. Acesso em: 15 abr. 2016.

CORREIA, Eduardo. Introdução ao formato JSON. **Devmedia**. Disponível em: < <http://www.devmedia.com.br/introducao-ao-formato-json/25275> >. Acesso em: 03 out. 2016.

CORREIO DO ESTADO. **Android controla 80% do mercado de smartphones**. Disponível em: <http://www.correiodoestado.com.br/noticias/android-controla-80-do-mercado-de-smartphones_190285/>. Acesso em: 23 de jun. 2016.

DEITEL, P. et al. **Título do livro: Uma abordagem baseada em aplicativos**. 1 ed. Porto Alegre: Bookman, 2013. 481 p.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de banco de dados**. 6. ed. São Paulo: Addison Wesley, 2011.

FIELDING, Roy Thomas. Architectural styles and the design of network-based software architectures. **University of california**, Irvine. 2000. Disponível em:

<http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm#tab_5_3>. Acesso em: 02 out. 2016.

FRANK, Diego R; SEIBT, Leonardo. **JavaScript**. Faculdade de Informática de Taquara, Taquara. Disponível em: < <https://fit.faccat.br/~leonardoseibt/ArtigoJavaScript.pdf>>. Acesso em: 10 out. 2016.

G1. **Com aumento da frota, país tem 1 automóvel para cada 4 habitantes**. Disponível em: <<http://g1.globo.com/brasil/noticia/2014/03/com-aumento-da-frota-pais-tem-1-automovel-para-cada-4-habitantes.html>>. Acesso em: 10 mar. 2016.

GODINHO, Rafael. Criando serviços REST com WCF, **Microsoft**, jun. 2009. Disponível em: < <https://msdn.microsoft.com/pt-br/library/dd941696.aspx>>. Acesso em: 10 out. 2016.

JAVASCRIPT. **Javascript**. Disponível em: <<https://www.javascript.com>>. Acesso em: 15 jun. 2016.

JUNIOR, Peter Jandl. **Java**: guia do programador. 2 ed. São Paulo: Novatec, 2013. 640 p.

KASPERBAUER, Marcelo. **Desenvolvendo aplicativos multiplataforma com tecnologias web**. Univale. Disponível em: < http://www.univale.com.br/unisite/mundo-j/artigos/56_multiplataforma.pdf>. Acesso em: 10 out. 2016.

LECHETA, Ricardo R. **Google android**: aprenda a criar aplicações para dispositivos móveis com Android SDK. 3 ed. São Paulo: Novatec, 2013. 824 p.

LECHETA, Ricardo Rodrigues. **Google Android**: aprenda a criar aplicações para dispositivos móveis com Android SDK. 2 ed. São Paulo: Novatec, 2010.

LUCKOW, Décio Heinzemann; MELO, Alexandre Altair. **Programação Java para a Web**. Aprenda a desenvolver uma aplicação financeira pessoal com as ferramentas mais modernas da plataforma Java 1 ed. Novatec, 2010.

MARENGONI, Maurício; STRINGHINI, Denise. Tutorial: Introdução à Visão Computacional usando OpenCV. **RITA**, São Paulo, v. 16, n. 1, p. 125, ago. 2009. Disponível em: <http://www.seer.ufrgs.br/index.php/rita/article/view/rita_v16_n1_p125/7289>. Acesso em: 11 mai. 2016.

OPENCV.**About**. Disponível em: <<http://opencv.org/about.html>>. Acesso em: 03 ago. 2016.

ORACLE. **MySQL**. Disponível em: < <https://www.oracle.com/br/products/mysql/overview/index.html> >. Acesso em 03 out. 2016.

RIOS, Luiz Romario Santana. Visão computacional. **Universidade federal da bahia**, Salvador. Disponível em: <[http://homes.dcc.ufba.br/~luizromario/apresenta%c3%a7%c3%a3o%20de%20ia/artigo%20\(final\).pdf](http://homes.dcc.ufba.br/~luizromario/apresenta%c3%a7%c3%a3o%20de%20ia/artigo%20(final).pdf)>. Acesso em: 16 jun. 2016.

TECMUNDO. **O que é e como usar mysql.** Disponível em: <<http://www.techtudo.com.br/artigos/noticia/2012/04/o-que-e-e-como-usar-o-mysql.html>>. Acesso em: 23 abr. 2016.

TOMCAT APACHE. **Tomcat apache.** Disponível em: <<http://tomcat.apache.org/>>. Acesso em: 12 mai. 2016.