UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

RENATO MAFIOLETTI MACARINI

PROTÓTIPO ANDROID PARA CONTROLE DE VAGAS DE ESTACIONAMENTO A PARTIR DE TRATAMENTO DE IMAGENS EM TEMPO REAL COM A BIBLIOTECA OPENCV

CRICIÚMA

2015

RENATO MAFIOLETTI MACARINI

PROTÓTIPO ANDROID PARA CONTROLE DE VAGAS DE ESTACIONAMENTO A PARTIR DE TRATAMENTO DE IMAGENS EM TEMPO REAL COM A BIBLIOTECA OPENCV

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. Esp. Sérgio Coral

CRICIÚMA

2015

RENATO MAFIOLETTI MACARINI

PROTÓTIPO ANDROID PARA CONTROLE DE VAGAS DE ESTACIONAMENTO A PARTIR DE TRATAMENTO DE IMAGENS EM TEMPO REAL COM A BIBLIOTECA OPENCV

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do Grau de Bacharel, no Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Computação Gráfica e Desenvolvimento Mobile.

Criciúma, 25 de Junho de 2015.

BANCA EXAMINADORA

Prof. Sérgio Coral - Esp - (UNESC) - Orientador

Prof. Fabrício Giordani - Esp. - (UNESC)

Prof. Paracelso de Oliveira Caldas - MSc - (UNESC)

Aos meus pais e irmãs que sempre me apoiaram, possibilitando a realização deste sonho.

AGRADECIMENTOS

Em primeiro lugar agradeço aos meus pais, Marlene Mafioletti Macarini e Nelson Macarini, que me acompanharam durante esta caminhada, me dando forças para a realização deste sonho.

Às minhas irmãs Maiara Macarini Rabelo e Samira Macarini Frizon, por todo o carinho e por sempre me apoiaram em todos os momentos.

Agradeço ao meu orientador Sérgio Coral, por aceitar esta tarefa de ser meu orientador e também por ter acreditado na minha capacidade para o desenvolvimento deste projeto.

Ao meu chefe e colega de trabalho Jorge Baggio Del Castanhel, por ser compreensivo e ter disponibilizado tempo durante meu trabalho para o desenvolvimento deste projeto.

Aos meus colegas acadêmicos Bruno de Mattia Amaral e Diogo Anphiloquio, que sempre me incentivaram, acreditaram em minha capacidade para o desenvolvimento deste projeto e me acompanharam durante todo o curso de Ciência da Computação.

Agradeço também a todos os professores do curso de Ciência da Computação, que compartilharam seus conhecimentos durante esta jornada.

“Não enlouqueça buscando a perfeição. Alguns defeitos são importantes.”

Paulo Coelho

RESUMO

O grande número de veículos nas ruas é um problema na atualidade, e está aumentando significativamente nos últimos anos, gerando congestionamentos não só nas estradas, mas também nos estacionamentos em função da procura por locais para estacionar o veículo. Já existem tecnologias capazes de amenizar este problema, porém não são muitas as opções, e possuem um alto custo de implantação. O presente trabalho propõe o desenvolvimento de um software capaz de identificar as vagas de estacionamento utilizando conceitos de visão computacional, realizando tratamento de imagens com auxílio da biblioteca OpenCV, que é específica para este ramo de processamento de imagens e possui licença de código aberto. O software utiliza a linguagem Java e através das imagens capturadas por uma *webcam*, o processamento de imagens é realizado com ajuda de filtros e métodos que a própria biblioteca disponibiliza, sendo possível identificar as vagas de estacionamento através de marcadores, conseguindo distinguir quais estão ocupadas e quais estão livres. Além disso, tendo em vista que grande parte da população possui um *smartphone*, e que este está se tornando algo indispensável no dia-a-dia das pessoas devido aos recursos que ele oferece, também é proposto o desenvolvimento de um aplicativo destinado a Android para que os usuários possam visualizar as vagas. A solução desenvolvida é um protótipo, sendo que para aplicação em um cenário real é necessário refinamentos nos códigos e métodos utilizados.

Palavras-chave: Visão Computacional, OpenCV, Estacionamento, Android.

ABSTRACT

The large number of vehicles on the streets is a problem today, and is increasing significantly in recent years, generating congestion not only on the roads but also in parking lots due to the demand for places to park the vehicle. There are already capable technologies to mitigate this problem, but there are not many options, and have a high cost of implementation. This paper proposes the development of software capable of identifying parking spaces using computer vision concepts, performing imaging with the help of OpenCV library, which is specific for this imaging industry and has open source license. The software uses the Java language and through the images captured by a webcam, image processing is performed with filters and methods that the library itself offers, and identifying parking spaces through markers, managing to distinguish which are busy and which are free. Moreover, given that much of the population has a smartphone, and that this is becoming something essential in nowadays live due to the features it offers, it is also proposed to develop an application for the Android to users can view the vacancies. The solution developed is a prototype, and to use it in a real parking it is necessary refinements in the used codes and methods.

Palavras-chave: Computer Vision, OpenCV, Parking, Android.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CTB Código de Trânsito Brasileiro

CMYK Padrão de cores formado por Ciano (Cyan), Magenta (Magenta), Amarelo (Yellow) e Preto (Black (Key)).

IDE Ambiente de Desenvolvimento Integrado

IOS Iphone Operations System

JAR Java Archive, formato de arquivo em java

LED Diodo emissor de luz (Light Emitting Diode).

OpenCV Open-source Computer Vision Library

PDA Personal Digital Assistant

RGB Padrão de cores formado por Vermelho (Red), Verde (Green) e Azul (Blue).

SDK Software Development Kit

USB *Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, a introdução de processos eletrônicos acrescentou velocidade e dinamismo ao dia-a-dia das pessoas, principalmente na comunicação, seja por meio de mensagens via videoconferências, rádio, internet, entre outros.

Para que seja possível acompanhar este ritmo de vida cada vez mais acelerado, onde são necessários deslocamentos até o trabalho, faculdade, banco, academia, entre outros locais que fazem parte da rotina diária, é necessário que estas locomoções se iniciem, muitas vezes, horas antes do planejado. Paralelamente, há o trânsito caótico, que frequentemente acarreta em engarrafamentos, panorama que resulta em outro problema que está cada vez mais comum, o de encontrar vagas em estacionamentos (ALHAK, 2011).

Uma grande parte da população opta por utilizar o serviço de estacionamentos privados mediante a realização de pagamento, onde deixam seu automóvel enquanto trabalham, realizam compras ou fazem outro tipo de tarefa. Porém, não raras vezes os estacionamentos já estão praticamente lotados, gerando dificuldade para a alocação de vagas para os veículos dos usuários devido ao fluxo existente dentro de grandes estacionamentos, tais como os de instituições de ensino, supermercados, shoppings, entre outros (PUERARI et al, 2011).

Grande parte das pessoas tem um celular do tipo smartphone, os quais podem ser resumidos a aparelhos com grande poder de processamento e que integram vários recursos, geralmente com telas sensíveis a toque. Com celulares smartphones, as pessoas podem, além de realizar ligações, ter alta conectividade com a Internet, falar com seus amigos através de mensagens, utilizar aplicativos, tais como GPS, sensores, entre as muitas funções que o aparelho pode proporcionar (OGLIARI; BRITO, 2014).

Cada marca de smartphone conta com um sistema operacional, sendo os mais conhecidos o Windows Phone, Android, IOS, dentre outros. O Android, objeto desse estudo, é uma plataforma para desenvolvimento e execução de programas para dispositivos móveis, robusta e de fácil utilização/aprendizagem. Ele está sendo largamente utilizado em muitos dos novos aparelhos celulares, conhecidos como smartphones (OGLIARI; BRITO, 2014).

Para a realização de aplicativos na área de visão computacional, é necessário algum tipo de biblioteca, a qual pode ser caracterizada como um

componente chave baseado na visão de aplicações. Embora existam várias bibliotecas, a maioria delas é grande e complicada, ou limitada a uma determinada combinação de hardware/plataforma. Esses fatores tendem a impedir o desenvolvimento de aplicações de pesquisa (GRANGE; FONG; BAUR, 2003, tradução nossa). O OpenCV, selecionado para o presente trabalho, é uma biblioteca multiplataforma, totalmente livre, tanto para uso acadêmico quanto para qualquer outra pessoa que tenha interesse em utilizá-la, mesmo que comercialmente (BRADSKI; KAEHLER, 2008, tradução nossa).

A visão computacional é a transformação ou nova representação de dados de uma câmera de vídeo. Todas essas transformações são feitas para alcançar algum determinado objetivo (BRADSKI; KAEHLER, 2008, tradução nossa).

O processamento de imagens consiste em procedimentos computacionais para extrair o alto nível de informações de uma imagem digital (SETCHELL, 1997, tradução nossa).

A detecção de objetos e o rastreamento (*tracking*) são alguns dos tópicos mais estudados na área de visão computacional, porque são as principais etapas de qualquer sistema de monitoramento de imagens (CUNHA, 2013).

Sendo assim, este projeto propõe o desenvolvimento de um software para tratamento de imagem nas captações feitas por uma *webcam*, a fim de identificar vagas disponíveis em um determinado estacionamento. E, por fim, o desenvolvimento de um aplicativo destinado ao Android, para que os usuários possam visualizar onde há vagas disponíveis.

1.1 JUSTIFICATIVA

Atualmente há um constante aumento da quantidade de veículos em circulação nas áreas urbanas, causando conflitos entre usuários de veículos automotores e a disponibilidade de espaços destinados para estacionar. Segundo dados estatísticos do DETRAN-SC, em 2003 a frota de veículos no estado de Santa Catarina era de 1.933.785 veículos, sendo que até abril de 2014, este número saltou para 4.257.286 (SANTA CATARINA, 2014).

Este grande crescimento no número de veículos está associado a mudanças na realidade urbana, como a implantação de Pólos Geradores de Tráfego (PGTs), que provocam impactos no tráfego e no trânsito (NUNES; JACQUES, 2004).

O presente trabalho busca amenizar os impactos dessa realidade, no tocante a necessidade de estacionamento e, em especial, de localização de vagas, para esses veículos, quando da realização de tarefas diárias. Um bom exemplo acontece na própria UNESC, onde muitos dos estacionamentos nem mesmo tem as vagas demarcadas, resultando em uma má utilização dos estacionamentos. Não esquecendo também do congestionamento gerado pela procura de vagas.

O Android, criado pelo Google e pela Open Handset Alliance, é um conjunto de ferramentas de software de código aberto desenvolvido para celulares utilizando a biblioteca *SDK*. Em poucos anos, é esperado que seja encontrado em milhões de telefones celulares e outros dispositivos móveis Android, tais como tablets, tornando-se uma importante plataforma para desenvolvedores de aplicativos (BURNETTE, 2008, tradução nossa).

Optou-se por utilizar o Android, já que a escolha de hardware é mais aberta, há mais opções no mercado de dispositivos móveis que o utilizam, abrangendo também várias faixas de preço. Para a escolha de um aparelho, basta saber o quanto o usuário está disposto a gastar e escolher o que mais lhe agrada. Alguns dos grandes nomes que utilizam esse sistema operacional em seus dispositivos são a Samsung, Motorola, Sony Ericsson, entre outros (GONÇALVES, 2011).

Por fim, o OpenCv tem vários módulos que podem ajudar, de diversas maneiras, problemas de visão computacional. Uma das partes mais úteis é sua arquitetura e gerenciamento de memória. Ela pode fornecer uma estrutura para trabalhar com imagens e vídeos em tempo real da maneira que bem entender. Além disso, vários algoritmos prontos são disponibilizados pela biblioteca ou, ainda, pode-se criar seu próprio, tudo isso sem ter de se preocupar com a alocação de memória para suas imagens (BRAHMBHATT, 2013, tradução nossa).

No desenvolvimento do presente projeto, será analisada a melhor maneira para tratar as imagens a fim de detectar as vagas de um determinado estacionamento e um aplicativo Android será desenvolvido para visualização das vagas. Para a realização dos testes será produzido um protótipo em escala reduzida.

Assim, espera-se que a elaboração deste projeto, tratando-se de um estudo inicial, possa contribuir para a implementação de futuros aplicativos que auxiliem na amenização dos impactos causados pelo trânsito e pela difícil tarefa de encontrar uma vaga em um estacionamento.

1.2 OBJETIVO GERAL

Desenvolver um software utilizando a biblioteca OpenCV a fim de identificar vagas disponíveis em determinado estacionamento através de imagens capturadas por uma webcam, e um aplicativo Android para visualização da localização das vagas disponíveis.

1.3 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desta pesquisa consistem em:

a) desenvolver um aplicativo destinado ao sistema operacional Android para visualização das vagas de estacionamento;
b) descrever o conceito de visão computacional e funcionamento de tratamento de imagens em tempo real;
c) verificar a melhor maneira para identificação das vagas em um determinado estacionamento;
d) desenvolver um software com auxílio da biblioteca OpenCV para o reconhecimento de vagas.

1.4 ESTRUTURA DA PESQUISA

A estrutura do trabalho é dividida em sete capítulos.

Primeiramente a introdução, onde se apresenta o objetivo geral e os objetivos específicos do trabalho, e a justificativa para a realização deste estudo.

Os estudos relacionados à visão computacional compõem o capítulo dois, o qual aborda o conceito, imagens digitais, espaço de cor, formatos de arquivos e por fim o processamento de imagens, mostrando o funcionamento para processar uma imagem a fim de extrair informações dela.

A mobilidade urbana e seu conceito se encontram no capítulo três, mostrando alguns dos problemas existes na atualidade e também falando um pouco sobre estacionamentos e seu funcionamento.

O conceito de dispositivos móveis é apresentado no capítulo quatro. Nele são mostrados diversos sistemas operacionais, enfatizando o Android, que domina a maior parte do mercado de *smartphones*.

A biblioteca OpenCV, no qual será utilizada para desenvolvimento do

software para reconhecimento de vagas de estacionamento é abordada no quinto capítulo.

No capítulo sexto são apresentados trabalhos correlatos ao estudo desenvolvido.

A apresentação do trabalho proposto, metodologia do projeto, os recursos necessários para a realização deste e também a discussão dos resultados obtidos é apresentada no sétimo capítulo.

Por fim, o oitavo capítulo que se compõe pela conclusão obtida após a realização do trabalho.

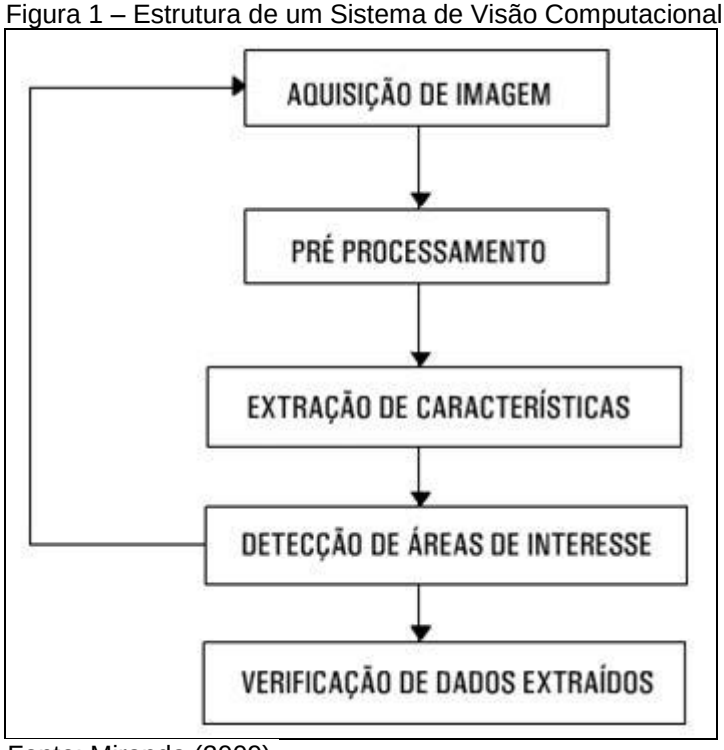
2 VISÃO COMPUTACIONAL

O conceito de Processamento Digital de Imagens, ou também chamado de PDI, refere-se ao conjunto de métodos e técnicas para interpretação de informações de imagens digitais que tem como objetivo facilitar a extração das mesmas (IBGE, 2001). Segundo Miranda (2008), a estrutura de um Sistema de Visão computacional é composta por pelo menos cinco etapas como mostrado na figura 1.

O interesse pelos vários métodos de processamento de imagens surge pelo fato de possuir duas principais categorias de aplicação que despertam um crescente interesse, uma delas é o aprimoramento da informação visual para a interpretação humana, e a outra é a extração de dados de cenas (MARQUES FILHO; VIEIRA NETO, 1999).

A visão computacional abrange diversas áreas além das duas principais já citadas, podendo ser aplicada também nas áreas de conhecimento de inteligência artificial (inteligência computacional), matemáticas, robótica (visão dos robôs), processamento de sinais, dentre outras (AUGUSTO, 2007).

Figura 1 – Estrutura de um Sistema de Visão Computacional



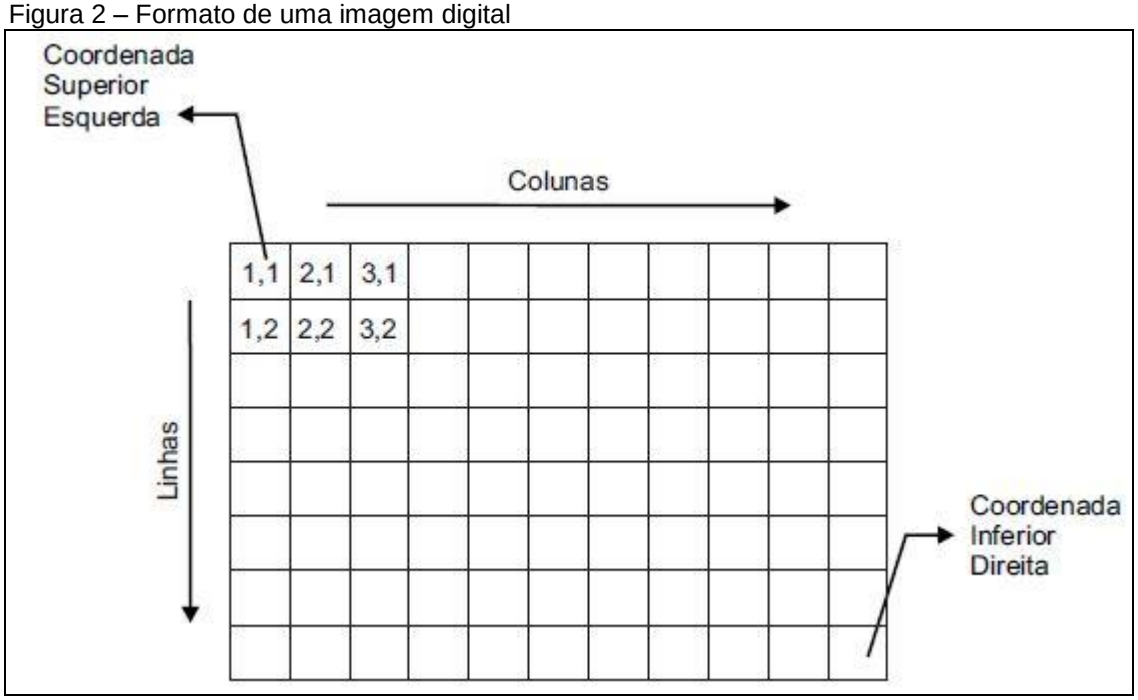
Fonte: Miranda (2009).

2.1 IMAGENS DIGITAIS

As imagens digitais podem ser criadas através de diversos aparelhos eletrônicos, tais como máquinas fotográficas, câmeras de vídeo, scanners, microscópios, dentre vários outros. O termo “fotografia digital” é largamente utilizado hoje, porém este é apenas um tipo de imagem digital, que pode ser adquirido através de câmeras fotográficas, celulares e até mesmo tablets.

Em sua definição uma imagem digital é vista por um computador como um arranjo de elementos (dígitos) em forma de uma grade regular, ou seja, uma função f(x,y) (MARQUES FILHO; VIEIRA NETO, 1999). Podendo também ser representado também por linha (x) e colunas (y) como mostra na figura 2.

Figura 2 – Formato de uma imagem digital



Fonte: IBGE (2001).

Segundo IBGE (2001), o nome do menor elemento na grade regular, ou seja, de cada coordenada de uma imagem digital, recebe o nome de pixel. Considerando isso, pode-se dizer que a quantidade de pixels de uma imagem está diretamente relacionada com a qualidade da mesma, ou seja, quanto maior for o número de pontos na imagem (*pixel*), maior a qualidade da imagem.

Segundo Augusto (2007), existe três tipos principais de imagens, que são:

a) imagens binárias: nesse tipo de imagem cada pixel pode assumir o valor de 0 ou 1 e ocupam 1 bit;

b) imagens em escala de cinza: também chamada de imagens monocromáticas, em que cada pixel é representado por um valor único de acordo com a intensidade do brilho;
c) imagens coloridas: nesse tipo de imagem cada pixel é representado por ao menos três valores, em outras palavras, um valor para cada canal de cor (por exemplo, RGB).

2.1.1 ESPAÇO DE COR

O espaço de cor tem o objetivo de especificar, criar e visualizar as cores. Para que a interpretação de cores de uma imagem específica seja feita, é necessário que haja pelo menos três informações sobre cada ponto da mesma. Dentro da computação, a representação das cores pode ser feita através da emissão de luz vermelha, verde e azul, do inglês Red, Green e Blue (RGB), que emitidas em diferentes tons obtém-se as cores desejadas (AUGUSTO, 2007).

Segundo Marques Filho e Vieira Neto (1999), o modelo RGB é o mais comumente utilizado na computação por câmeras e monitores de vídeo. É baseado em um sistema de coordenadas cartesianas, podendo ser comparado para análise como um cubo, onde três de seus vértices são as cores primárias, o restante são as secundárias, o vértice junto à origem é o preto e o mais afastado é o branco assim como mostra na figura 3.

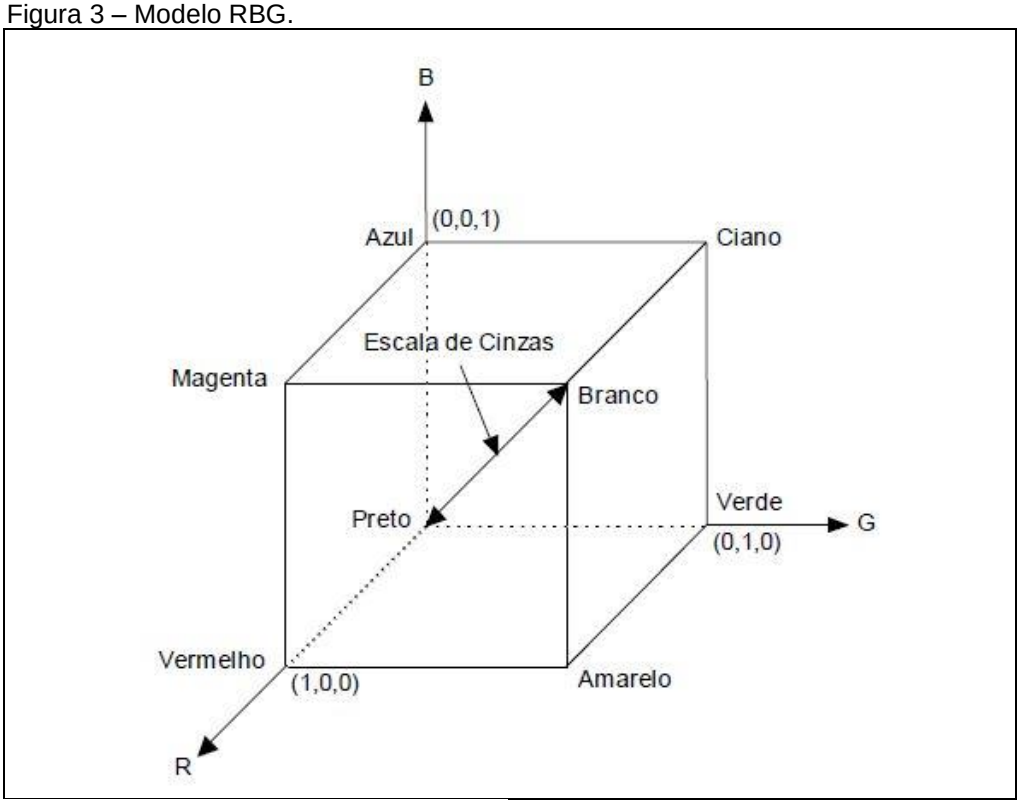
Há também o modelo de emissão de luz ciano, magenta, amarelo e preto, do inglês Cyan, Magenta, Yellow e Black (Key) (CMYK), baseado nas cores primárias ciano, magenta e amarelo. Este é principalmente utilizado por impressoras, fotocopiadoras coloridas e na maioria dos dispositivos que tem como principio a deposição de pigmentos coloridos em folhas de papeis. Porém já que o computador utiliza o formato RGB, é necessário que haja uma conversão para o CMYK (MARQUES FILHO; VIEIRA NETO, 1999). Esta conversão pode ser feita da seguinte equação:

$$Ciano = 1 - vermelho$$

$$Magenta = 1 - verde$$

$$Amarelo = 1 - azul$$

Figura 3 – Modelo RBG.



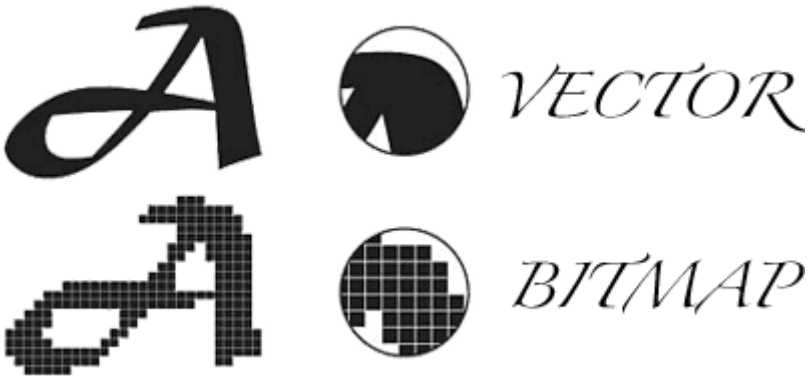
Fonte: Marques Filho e Vieira Neto (1999).

2.2 FORMATOS DE ARQUIVOS DE IMAGENS

No que diz respeito a formatos de arquivos de imagens, pode-se dizer que há dois modos básicos de representação utilizados para sua composição, sendo uma delas através de bitmaps (mapa de bits), podendo ser chamado também como *pixel maps* (mapa de pixels) ou raster (varredura), e o outro formato, por vetores (MARQUES FILHO; VIEIRA NETO, 1999).

A representação por Bitmaps é mais adequada onde imagens possuem variações complexas em suas formas e cores, como por exemplo, nos *frames* (quadros) de um vídeo. Já por vetores, onde é descrita por parâmetros de suas formas geométricas, é mais adequada para as quais há uma predominância de linhas e preenchimento simples, como por exemplo, em desenhos simples, diagramas ou gráficos (MARQUES FILHO; VIEIRA NETO, 1999).

Figura 4 – Diferença entre imagem por Vetores e Bitmaps

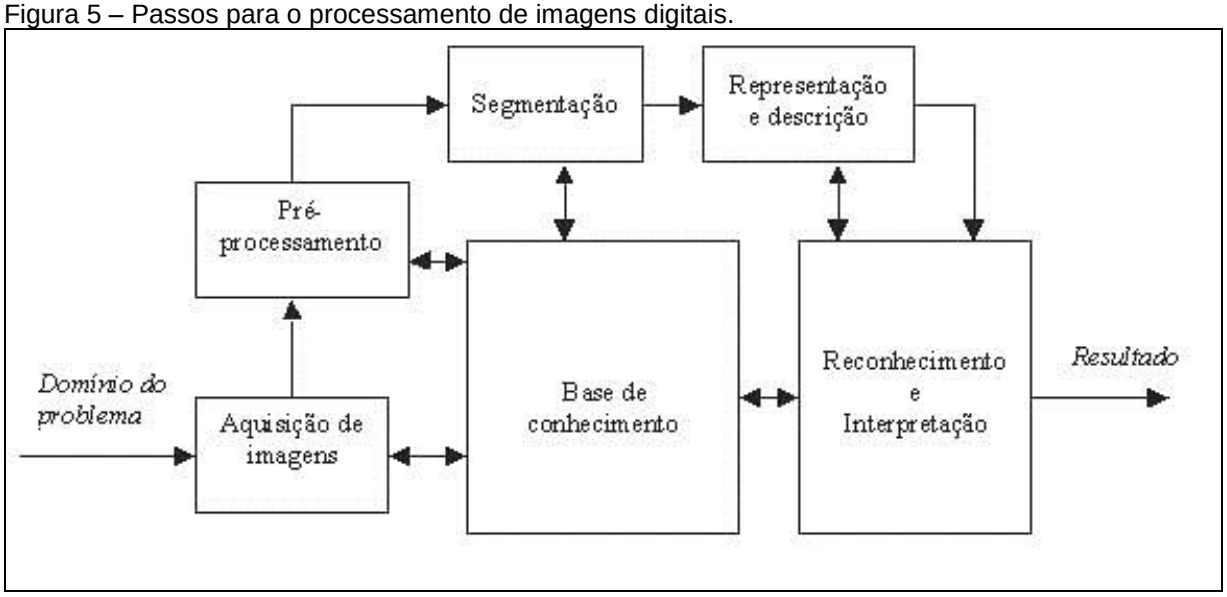


Fonte: Adobe (2015).

2.3 PROCESSAMENTO DE IMAGENS

O processamento digital de imagens é composto de processos, tendo as imagens como as entradas e saídas. Além disso, também há etapas de extração de atributos e informações de imagens ou até mesmo o reconhecimento de objetos específicos (GONZALEZ; WOODS, 2000).

Figura 5 – Passos para o processamento de imagens digitais.



Fonte: Gonzalez e Woods (2000).

Conforme mostra a figura 5, a primeira etapa para o processamento de imagens é adquirir uma imagem digital. Isto pode ser feito através de uma câmera, ou outro aparelho eletrônico capaz de capturar imagens a cada 1/30 s. Após isso ocorre o pré-processamento, sendo que o principal objetivo nesta etapa é melhorar a

imagem de acordo com o domínio do problema para poder obter um melhor resultado nas etapas seguintes, por exemplo, um aumento de contraste ou isolamento de regiões (GONZALEZ; WOODS, 2000).

A etapa de Segmentação tem como objetivo dividir a imagem em diferentes regiões de acordo com o domínio do problema, que serão posteriormente analisadas por algoritmos específicos para a extração de informações. Em uma imagem monocromática (preto e branco), por exemplo, poderia ser realizada uma segmentação considerando os seus níveis de cinza ou suas diferenças (MARQUES FILHO; VIEIRA NETO, 1999).

O processo de representação e descrição é uma parte da transformação dos resultantes da segmentação em uma forma adequada para o processamento computacional que será realizado. Além disso, também ocorre uma extração de informações ou características de interesse (GONZALEZ; WOODS, 2000).

Na próxima etapa há dois estágios, sendo o primeiro chamado de Reconhecimento e o segundo de Interpretação. Nesta parte ocorre uma classificação dos objetos a partir das informações extraídas da imagem juntamente com a base de conhecimento já estabelecida anteriormente e também a atribuição de um significado a eles (MARQUES FILHO; VIEIRA NETO, 1999).

A base de conhecimento é muito importante para o processamento de imagens, podendo ser bastante complexa, sua função é realizar a iteração entre os módulos durante os processos e também de guia-los (GONZALEZ; WOODS, 2000).

O resultado final do processamento de imagens pode sair em qualquer etapa da figura 5, pois isto depende da complexidade de iterações da aplicação. Em alguns casos nem mesmo todas as etapas são necessárias. E por fim a saída pode ser um conjunto de informações ou até mesmo uma imagem (GONZALEZ; WOODS, 2000).

2.3.1 APLICAÇÃO DE FILTROS

Segundo Gonzales e Woods (2000), a aplicação de filtros no domínio espacial da imagem, são operações aplicadas para a redução de ruídos com o auxílio de máscaras, chamadas de filtros espaciais. Sendo a imagem uma matriz, a execução parte de um ponto específico e o cálculo é realizado juntamente com os valores vizinhos.

Este processo de aplicação de filtros utilizando matrizes chamadas de máscaras, consiste na aplicação destas em toda imagem ou uma parte específica, substituindo o valor do pixel central por um novo de acordo com os valores dos pixels vizinhos (MARQUES FILHO; VIEIRA NETO, 1999).

Os filtros espaciais possuem três classificações possíveis, passa-baixa, passa-alta ou passa-faixa. Quando componentes da imagem são atenuados ou eliminados, e que forem de alta frequência na imagem, estes são denominados passa-baixas. Estes resultam em um efeito de borrado, pois os componentes de alta frequência, que são as bordas e pequenos detalhes são eliminados. Já os filtros de passa-alta atuam de forma contrária, estes realçam as bordas e os pequenos detalhes. Por fim os de passa-faixa são os mais complexos, podendo remover ou atenuar partes da imagem (MARQUES FILHO; VIEIRA NETO, 1999).

Através da figura 6 pode ser exemplificado, onde o filtro da média é executado a um ponto central, e alterado a partir da média dos valores de seus pixels vizinhos. Este filtro tem como função a remoção de ruídos, mas preservando bordas e detalhes finos na imagem.

Figura 6 – Efeito do filtro da média em uma imagem.

10	10	0	0	0	0	0	0	20	20	10
90	80	80	70	65	65	40	40	30	30	20
35	35	35	35	50	50	42	42	30	30	20
90	180	180	180	180	180	120	100	90	70	50
200	200	190	190	**180**	**180**	**120**	120	90	72	55
200	200	190	190	**150**	150	**120**	120	90	72	50
205	210	190	190	**150**	**150**	**120**	120	90	72	50
204	210	205	200	150	150	120	120	90	73	50
210	210	210	200	150	150	120	120	80	73	40
190	190	190	185	150	150	120	120	80	69	23
190	190	190	185	145	145	115	115	80	69	23

Fonte: Mexas (2014).

3 MOBILIDADE URBANA

O termo mobilidade urbana possui vários conceitos e definições, mas o mais comum deles é a facilidade de conseguir se locomover, ou seja, o deslocamento de pessoas e bens nas cidades, sendo este, feito através de transportes públicos individuais ou privado, podendo chegar ao local de destino no tempo marcado, porém com exceções de situações imprevistas, tais como acidentes de transito e congestionamentos (KNEIB, 2014).

Kneib (2014) afirma ainda que a mobilidade sofre muita influência da articulação e união de política de transportes, acessibilidade, circulação, trânsito, desenvolvimento urbano, uso e ocupação do espaço, dentre outras. Todas essas políticas afetam e são afetadas também pela mobilidade das pessoas, pois além das já citadas, ainda há as culturais, educacionais, de segurança e ambientais que refletem tanto diretamente como indiretamente na mobilidade nas pessoas.

No início do século XX, os automóveis eram a melhor maneira de deslocamento na mobilidade diária. Mas com o passar do tempo, o número de automóveis veio aumentando de forma significativa, e, sem um aumento proporcional no número de estradas, surgiram os enormes engarrafamentos no século XXI, aumentando também a poluição nas cidades (REBOUÇAS, [entre 2006 e 2014]).

3.1 PROBLEMAS NA ATUALIDADE

Na atualidade, com o crescimento da população, cresce também a demanda de carros e do inchaço urbano, sendo que ter um carro próprio para deslocamento não é mais um sinônimo de autonomia, velocidade e conforto. Ficar parado no trânsito, em semáforos e congestionamentos tem como consequências a perda de tempo e de qualidade de vida (REBOUÇAS, [entre 2006 e 2014]).

Na figura 7 podem ser observadas as consequências da quantidade excessiva de carros nas ruas.

Figura 7 - Excesso de veículos

Fonte: Elefanteverde (2013).

Segundo Kneib (2014) a sociedade na atualidade tem como foco, obter o seu próprio meio de transporte individual. Isso afeta diretamente a mobilidade urbana nas cidades, pois quanto maior for o volume de automóveis, mais congestionamentos acabam ocorrendo na cidade. E quanto mais vias são abertas, ou até mesmo estendidas de tamanho para a melhor circulação dos carros e motos, mais veículos são adquiridos e começam a rodar pela cidade, criando assim uma demanda por espaço.

Kneib (2014) tem uma opinião formada de que é necessário convencer a sociedade que está acostumada a adquirir e valorizar seus automóveis e usá-los para ir aonde bem entendem, estacionar aonde querem e na hora que lhes convêm, que em um numero excessivo, eles são prejudiciais à cidade. É preciso fazer com que o número de veículos produzidos e vendidos pare de aumentar.

Segundo Lindau (2014) não é possível as redes viárias acompanharem o crescimento desenfreado da frota de automóveis. Por muito tempo priorizou-se aumentar o número de veículos da forma mais rápida possível, porém junto a isso vieram consequências que hoje são claramente visíveis nas grandes metrópoles e até mesmo em cidades pequenas.

3.2 ESTACIONAMENTOS

Além das dificuldades no trânsito citadas, outra problemática atualmente encontrada é a busca de um local adequado para estacionar o veículo.

O Anexo 1 do Código de Transito Brasileiro (CTB) (1997, p.55) diz que “estacionamento é a imobilização de veículos por tempo superior ao necessário para embarque e desembarque de passageiros”. Estacionamentos podem ser classificados de duas formas: nas vias públicas e foras das vias públicas (CET, 1979). Os estacionamentos ditos fora das vias públicas podem ser entendidos como lotes e as garagens, podendo estes serem públicos ou privados, gratuitos ou pagos. Já nas vias públicas, estes são os que estão ao longo do meio-fio, podendo ser livres ou pagos, e, oferecem um acesso mais fácil e também mais econômico na maioria dos casos.

Na atualidade foram criadas tecnologias para o problema da organização de estacionamentos, algumas delas podem ser chamadas de estacionamentos inteligentes. Existem hoje tecnologias capazes de mostrar ao motorista onde há vagas livres através de lâmpadas de diferentes cores, e ainda indicam o caminho para a vaga disponível mais próxima. Mas também há tecnologias muito mais avançadas mundo afora, onde existem estacionamentos em edificações que são completamente autônomos (figura 8), basta que o motorista deixe o carro em uma plataforma e um sistema automatizado estaciona o veículo, e na hora de ir embora, o motorista só precisa fazer a requisição de seu automóvel que ele será trazido de volta (PUERARI et al, 2011).

Figura 8 – Estacionamento vertical automatizado.

Fonte: Metalica (2011).

Segundo Metalica (2011), estacionamentos verticais são na verdade edifícios que funcionam como uma garagem, onde a condução dos veículos é totalmente automatizada e feita através de plataformas. Nestes, não é possível o acesso ao seu interior, pois os veículos postos em vagas por meio de equipamentos totalmente robotizados.

No Brasil, o primeiro Shopping a adotar um sistema de estacionamento inteligente foi o Morumbi, em São Paulo, conforme é mostrado na figura 9.

Figura 9 – Estacionamento Inteligente do Shopping Morumbi.

Fonte: Alhak (2011).

O estacionamento mostrado na figura 9 utiliza sensores e *Light Emitting Diode* (LED) verde - os quais indicam que a vaga está livre - e vermelho - que significa que a vaga está ocupada. Tais dispositivos têm como principal objetivo o de acabar com o tempo perdido na procura por vagas livres no estacionamento do shopping. O sistema ainda ajuda a administração do estabelecimento no controle do fluxo de veículos e da ocupação das vagas.

Os painéis servem para indicar a localização de vagas disponíveis e podem ser vistos a uma longa distância, havendo um destes logo na entrada do estacionamento e também em outros setores do mesmo. Além disso, também fazem a contagem das vagas disponíveis por setor e indicam ao motorista onde elas estão localizadas.

Contudo, estacionamentos no geral não têm um espaço satisfatório, com uma quantidade de vagas razoável, não tendo fiscalização nem mesmo segurança. O foco do presente trabalho visa à identificação prévia de vagas livres onde

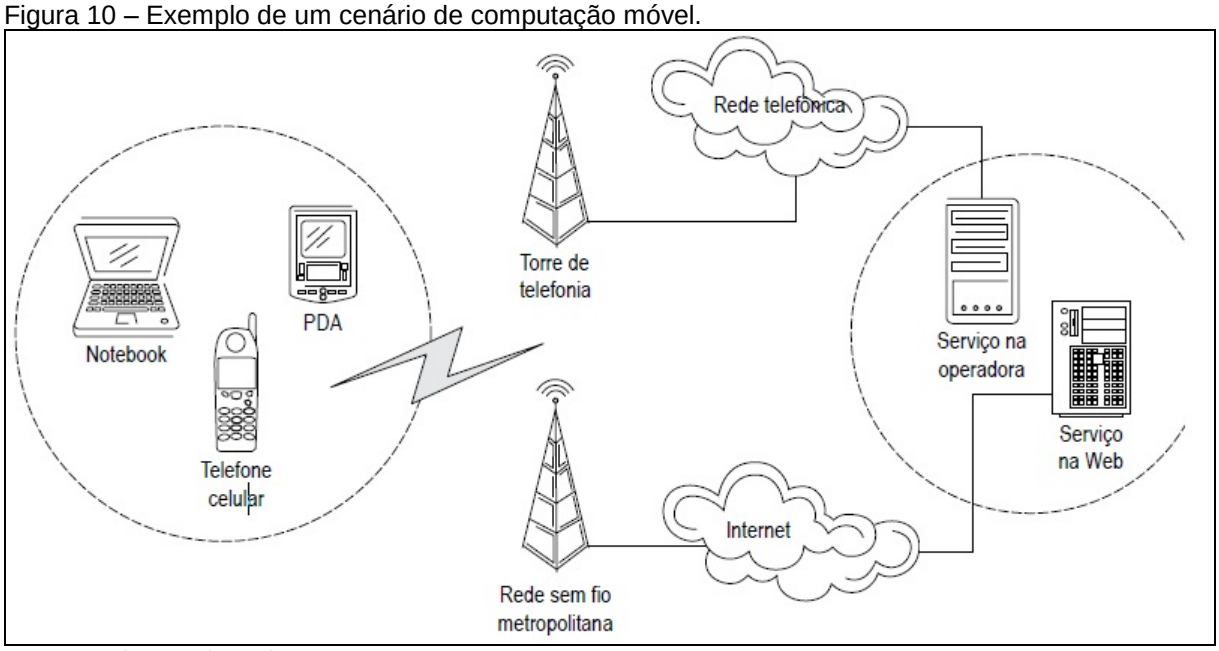
motoristas possam estacionar seus veículos sem precisar ficar dando voltas pelo estacionamento procurando. Além disso, o foco está na diminuição de custos para a implantação e utilização do sistema. Outras vantagens seriam as diminuições do congestionamento do trânsito e do gasto de combustíveis.

4 DISPOSITIVOS MÓVEIS

O número de dispositivos móveis hoje em dia está cada vez maior e está sendo usando nas mais diversas áreas. Telefones celulares, smartphones, máquinas digitais, notebooks, entre outros, estão cada vez mais se tornando parte do cotidiano das pessoas. Em alguns casos são até mesmo indispensáveis, visto que podem modificar as rotinas e até mesmo as decisões individuais, sendo estas tanto no trabalho quanto no lazer. A mobilidade está deixando de ser algo que facilita o cotidiano, e se tornando algo realmente necessário, pois é capaz de acessar dados e informações em qualquer lugar e a qualquer momento.

Com a computação móvel, usuários possuem acesso a serviços independentes de sua localização. Para isso é necessário que haja suporte a esta mobilidade e também de uma boa infraestrutura de comunicações sem fio, como pode ser visto na figura 10, onde mostra que as redes de telefonia celular e as redes sem fio são a infraestrutura necessária para o acesso aos serviços móveis (JOHNSON, 2007).

Figura 10 – Exemplo de um cenário de computação móvel.



Fonte : Johnson (2007).

Segundo Johnson (2007) os dispositivos móveis podem ser classificados em *laptops* (ou *notebooks*), *Personal Digital Assistant* (PDA) e telefones celulares. Já para Lee, Schneider e Schell (2005), os *tablet* PCs e híbridos também entram nessa classificação:

a) *laptops*: computadores móveis com poder de processamento e hardware próximos ou equivalentes a um computador desktop;

b) PDA: computadores de dimensões reduzidas que surgiram primeiramente com funções de assistente pessoal. Suas novas versões possuem a possibilidade de enviar e receber e-mails, acesso à internet, jogos, além também de aplicações personalizadas;

c) telefones celulares: suas funções primárias são realizar ligações e atendê-las. Com o avanço da tecnologia, hoje oferecem funções como: envio de mensagens de texto, câmera, acesso à internet, jogos e aplicações personalizadas;

d) *tablet* PCs: computadores móveis com o formato de uma prancheta, neste destaca-se uma tela interativa sensível ao toque, também chamada de *touch screen*;

e) híbridos: pode ser citado os smartphones como exemplos desta categoria. Estes possuem as funções primárias de PDAs e também de telefones celulares.

Os dispositivos móveis, em particular os *smartphones* e os *tablets*, estão com um grande destaque devido ao poder de conectividade, além de poderem ser utilizados tanto para uso pessoal como profissional. Com a grande explosão destes dispositivos no mercado, vários novos aplicativos exclusivos vêm sendo lançados, oferecendo aos usuários as facilidades de um mundo interligado por redes sem fio, possibilitando a qualquer hora e em qualquer lugar, o acesso a informações, bastando somente uma conexão de rede sem fio.

4.1 SISTEMA OPERACIONAIS

Segundo Jipping (2007), sistema operacional pode ser definido de várias formas, mas todas elas com algo em comum:

a) sistema operacional é um software: o local onde ele está armazenado não tem importância, pois em algum momento ele será carregado na memória e suas instruções serão executadas como qualquer outro programa;

b) sistema operacional é um modelo de recursos: projetado com a finalidade de apresentar os recursos de hardware para os softwares e

usuários, construindo um modelo, em outras palavras, um sistema que possibilita que os usuários visualizem e utilizem os recursos;

c) sistema operacional vincula o hardware ao software: O software faz um reconhecimento do hardware e consegue acesso através do modelo do sistema operacional;

d) sistema operacional é essencial: sem sistema operacional não é possível que nenhum software seja executado e nem mesmo que o hardware utilizado.

Hoje existe uma concorrência entre os dispositivos móveis, são várias marcas e vários modelos que estão no mercado. *Smarthphones* têm chamado a atenção de muitos consumidores devido as suas plataformas de desenvolvimento. No mercado atual os sistemas operacionais que mais se destacam são o *Android* da *Google*; *iOS* (*iPhone*) da *Apple;* o *Windows Phone 8* da Microsoft; e o *BlackBerry* (COSTA; DUARTE FILHO, 2013).

Segundo Costa e Duarte Filho (2013) grande parte dos sistemas operacionais para *smartphones* são "abertos" (não confundir com código-fonte aberto), ou seja, que é possível a qualquer pessoa que tenha conhecimentos desenvolver programas por meio de um *Software Development Kit (SDK)*, ou no português, Kit de Desenvolvimento de Aplicativos ou *Framework* (conjunto de classes que ajudam no desenvolvimento de um subsistema ou da aplicação em si) que possibilita ser executado por estes telefones.

Costa e Duarte Filho (2013) afirmam também que um dos grandes problemas atuais dos sistemas operacionais de dispositivos móveis é a falta de padronização entre eles, resultando em problemas tanto para empresas como para usuários. Com isso muitas organizações escolhem sistemas desconhecendo as principais funcionalidades e a arquitetura do *software*, que por fim após o desenvolvimento do aplicativo, concluem que não era o sistema ideal para seus negócios.

A tabela 1 mostra os sistemas operacionais que estiveram nos *smartphones* vendidos nos anos de 2012 e 2013 (GARTNER, 2014).

Tabela 1 - Vendas mundiais de smartphones para usuários finais por Sistema Operacional em 2012 e 2013 (milhares de unidades).

Sistema Operacional	Unidades (2013)	% (2013)	Unidade (2012)	% (2012)
Android	758.719.9	78.4	451.621.0	66.4
IOS	150.785.9	15.6	130.133.2	19.1
Windows Phone	30.842.9	3.2	16.940.7	2.5
BlackBerry	18.605.9	1.9	34.210.3	5.0
Outros OS	8.821.2	0.9	47.203.0	6.9
Total	**967.775.8**	**100.0**	**680.108.2**	**100.0**

Fonte: Adaptado de Gartner (2014).

Conforme demonstrado pela tabela 1, o Android é o sistema operacional que esteve presente em mais da metade dos *smartphones* vendidos no ano de 2012 e também de 2013, onde teve um aumento bastante significativo, seguido do iOS que caiu um pouco as vendas , *Windows Phone*, *BlackBerry*, e outros.

As vendas de celulares para usuários finais no mundo todo totalizaram 1,8 bilhão de unidades em 2013, um aumento de 3,5 por cento a partir de 2012. Foram comprados 490,3 milhões de telefones móveis no quarto trimestre de 2013, um aumento de 3,9 por cento em comparação com o mesmo trimestre de 2012 conforme mostra a tabela 2.

Tabela 2 – Vendas mundiais de telefones móveis por Fabricantes em 2013 (milhares de unidades).

Fabricantes	Unidades (2013)	% (2013)	Unidade (2012)	% (2012)
Samsung	444.444.2	24.6	384.631.2	66.4
Nokia	250.793.1	13.9	333.938.0	19.1
Apple	150.785.9	8.3	130.133.2	2.5
LG	69.024.5	3.8	58.015.9	5.0
ZTE	59.898.8	3.3	67.344.4	6.9
Huawei	53.295.1	2.9	47.288.3	
TCL Communication	49.531.3	2.7	37.176.6	
Lenovo	45.284.7	2.5	28.151.4	
Sony	37.595.7	2.1	31.394.2	
Yulong	32.601.4	1.8	18.557.5	
Outros	613.710.0	34.0	609.544.9	
Total	**1.806.964.7**	**100.0**	**1.746.175.6**	**100.0**

Fonte: Adaptado de Gartner (2014).

4.2 ANDROID

O sistema operacional Android surgiu realmente em 2005, quando a Google comprou uma pequena empresa de Palo Alto chamada de Android Inc, situada na Califórnia, como uma parte de uma estratégia para entrar no mercado de dispositivos móveis. A empresa mantinha em segredo todos os seus projetos, pois ela era capaz de desenvolver sua própria tecnologia, não precisando de auxílio de outras empresas. Hoje a plataforma é desenvolvida por um consórcio de empresas chamado *Open Handset Alliance* (OGLIARI; BRITO, 2014).

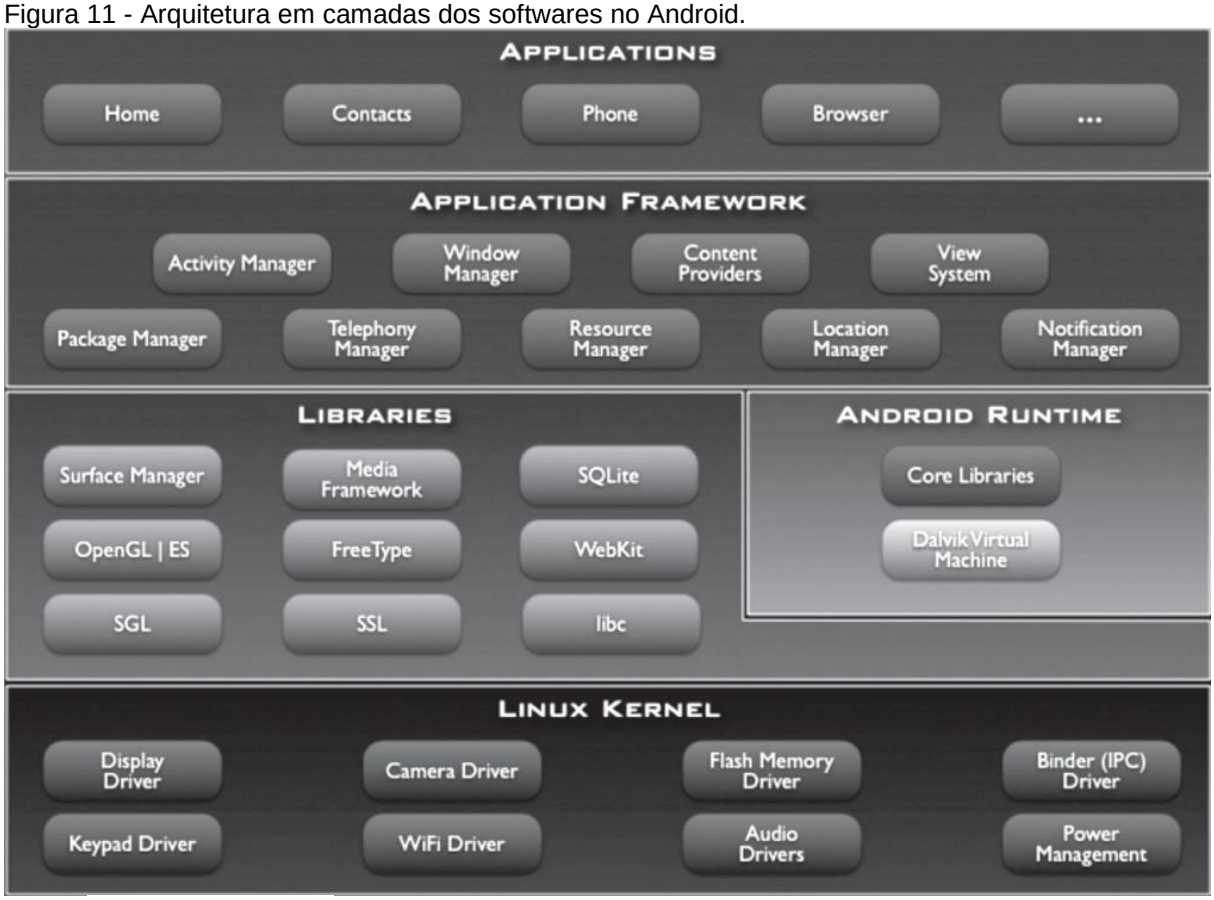
A Android Inc. tinha como principal objetivo o desenvolvimento de um sistema operacional para câmeras digitais. Ela havia percebido que não havia muita concorrência no mercado para dispositivos desse ramo, com isso focou seus esforços em um sistema operacional para smartphones, concorrendo diretamente com a Nokia e com a Microsoft. Para isso, começaram o desenvolvimento do novo Sistema Operacional chamado Android, baseado no núcleo do Linux (GUIMARÃES, 2013).

O Android possui uma grande vantagem, uma vez que sua plataforma também é livre e de código aberto. Em outras palavras, sua licença permite que cada fabricante ou empresa realize alterações no código-fonte de seus aparelhos móveis, e não são obrigadas a compartilhar estas mudanças com ninguém. Ele também é “*free*”, ou seja, não é necessário que os fabricantes paguem para poder utilizá-lo, assim podem usufruir de todos os recursos oferecidos por ele, além também de fazer suas próprias alterações (COSTA; DUARTE FILHO, 2013).

O Android apesar de ser livre e de código aberto, possui uma característica própria, que é a prioridade igualitária tanto para aplicativos nativos quanto para de terceiros. As aplicações desenvolvidas pelos fabricantes utilizam todos os recursos oferecidos pelo aparelho, ou seja, toda a infraestrutura de hardware e software, da mesma maneira que os aplicativos nativos, como, por exemplo, acesso ao Google Maps, Calendário, Agenda, entre outros (OGLIARI; BRITO, 2014).

A infraestrutura de software do Android é constituída de 5 camadas conforme mostra a figura 11. Dentre as camadas estão um sistema operacional baseado em Linux, um conjunto de bibliotecas, uma API chamada de Runtime, as aplicações nativas do próprio sistema e outras diversas.

Figura 11 - Arquitetura em camadas dos softwares no Android.



Fonte: Ogliari e Brito (2014).

A responsabilidade pelas tarefas de gerenciamento de memória, acesso a rede, gerenciamento de processos, dentre outras funções é toda do kernel Linux. O sistema operacional é responsável por fazer a comunicação entre o software desenvolvido e o hardware (OGLIARI; BRITO, 2014).

As *libraries* (bibliotecas) são um conjunto de bibliotecas de linguagem C ou C++, que são usadas por componentes do sistema operacional. Para que o desenvolvedor possa realizar o acesso a estas, é utilizado o Android Application FrameWork (OGLIARI; BRITO, 2014). Segundo Ogliari e Brito (2014) as principais bibliotecas são:

a) SQLite: banco de dados relacional disponível em todas as versões;
b) 3D Libraries: baseada no OpenGL ES 1.0 APIs. Se o aparelho possuir aceleração 3D em seu hardware, ela o utiliza para realizar o processamento de imagens gráficas;

c) media libraries: baseada no PacketVideo's OpenCORE, possui suporte a gravações de vários formatos de mídia, além disso, também trabalha com imagens estáticas;

d) LibWebCore: um motor de navegador Web usado nos browsers do sistema para simplificar o carregamento da página;

e) Surface Manager: controla o acesso ao subsistema de display e compõem as camadas gráficas 2D e 3D;

f) FreeType: renderização de fontes bitmap e vetorial.

A parte de Android Runtime é responsável pela execução dos aplicativos. É compreendida pelas bibliotecas básicas do sistema operacional e são necessárias para as bibliotecas citadas anteriormente (OGLIARI; BRITO, 2014).

A parte de Application Framework são as bibliotecas mais acessadas pelos desenvolvedores, pois são elas as responsáveis pela interação com o dispositivo móvel. São ditas também como bibliotecas de alto nível, algumas de suas funções, por exemplo, é o controle de janelas, notificações, recursos do aparelho, dentre outras (OGLIARI; BRITO, 2014).

A última parte são os aplicativos, estes são as ferramentas e estão disponíveis para todos os usuários de celulares e smartphones. Outra característica do Android é que qualquer programa instalado no dispositivo possui permissão para compartilhar informações com outros, e, para concluir, um software de terceiros pode acessar aplicativos nativos (OGLIARI; BRITO, 2014).

5 OPENCV

Open Source Computer Vision Library (OpenCV) é uma biblioteca de código aberto desenvolvida pela Intel com mais de 500 funções para visão computacional, disponível no site oficial http://opencv.org. Ela foi desenvolvida com o objetivo de tornar a visão computacional mais acessível a usuários e também desenvolvedores nas área de interação entre humano e computador em tempo real, e também, a robótica (CUNHA, 2013).

A biblioteca foi projetada visando eficiência computacional e tendo como foco principal as aplicações de processamento em tempo real. Ela foi escrita na linguagem C e C++, podendo tirar proveito de múltiplos núcleos processadores, e pode ser utilizada através de Linux, Windows, e também por Mac OS X (BRADSKI; KAEHLER, 2008, tradução nossa).

Há muitas empresas que utilizam essa biblioteca, e muitas delas conhecidas, tais como a Google, Yahoo, Microsoft, Intel, IBM, Honda, Toyota, Sony, e muitas outras (BRADSKI; KAEHLER, 2008, tradução nossa).

Segundo Cunha (2013) podemos citar como exemplo de aplicações que utilizam o OpenCV: a união de imagens das ruas na detecção de intrusos em vídeos de monitoramento em Israel, equipamentos de monitoramento de minas na China, checagem de buracos nas rodovias da Turquia, detecção de afogamentos em piscinas na Europa, inspeção de rótulos de produtos nas fábricas, detecção da face humana, dentre vários outros.

O OpenCV foi desenvolvido com um objetivo principal que é fornecer uma infraestrutura com uma utilização simplificada para a visão computacional, com isso as pessoas podem construir aplicações sofisticadas e robustas com rapidez. Dentre as mais de 500 funções citadas anteriormente, está a inspeção de produtos, imagem médica, segurança, interface de usuário, calibração de câmera e várias outras (BRADSKI; KAEHLER, 2008, tradução nossa).

Sua construção foi feita em módulos integrados, sendo assim bastante versátil e eficiente para resolver problemas onde seja necessária a utilização de visão computacional. É possível cortar imagens, modificá-las, detectar formas, segmentação de imagens, detecção de objetos em movimento em vídeos, reconhecimento de objetos, dentre várias outras possibilidades (BRAHMBHATT, 2013). Seus módulos são os seguintes:

a) core: gerenciamento de memória, tipos de dados, estrutura de dados do núcleo;
b) imgproc: filtragem de imagens, transformação de imagens geométricas, estruturas, análise de formas;
c) highgui: leitura e escrita de imagens e vídeos;
d) vídeo: análise de objetos em movimento e monitoramento através de vídeos;
e) calib3d: calibragem de câmera e reconstrução 3D por meios de múltiplos ponto de vista;
f) features2d: extração de características e descrições;
g) objdetect: detecção de objetos usando cascata e histograma de gradiente;
h) ML: modelos estatísticos e classificação de algoritmos usados em aplicações de visão computacional;
i) flann: busca rápida de objetos próximos;
j) GPU: paralização de algoritmos selecionados para uma execução mais rápida;
k) stitching: ajustes de imagens;
l) nonfree: implementação de algoritmos que são patenteados em alguns países;

Segundo Cunha (2013) a biblioteca em si possui um ponto negativo, que é a necessidade de que o desenvolvedor tenha um conhecimento robusto sobre programação em C ou C++ para que a aplicação tenha um melhor desempenho. A linguagem C++ é derivada do C e possui maiores vantagens como a redução de linhas de código como mostra na figura 12 que compara um simples programa para abrir e exibir uma imagem, melhor aproveitamento de código e uma menor preocupação com a memória utilizada. Porém o OpenCV pode ser utilizado também em outras linguagens, tais como Python e Java.

Figura 12 – Comparativo entre os código em C e C++.

Algoritmo 1. Código escrito em C

```
int main
{
   IplImage* img = cvLoadImage("img.jpg");
   cvNamedWindow("Imagem", 1);
   cvShowImage("Imagem", img );
   cvWaitKey(0);
   cvReleaseImage( img );
   cvDestroyWindow("Imagem");
}
```

Algoritmo 2. Código escrito em C++

```
int main
{
   cv::Mat img = cv::imread("img.jpg");
   cv::imshow("Imagem", img );
   cv::waitkey(0);
}
```

Fonte: Cunha (2013).

5.1 SEGMENTAÇÃO DE IMAGENS

A segmentação de imagens pode ser definida como uma maneira de isolar objetos ou partes de objetos do resto da imagem, focando somente nas regiões de interesse e desfocando no que não é relevante. Em câmeras de segurança, por exemplo, onde ela está sempre olhando para um mesmo fundo, o qual muitas vezes é irrelevante, e o que realmente interessa é o fluxo de pessoas e veículos que entram em cena, ou algum objeto deixado na cena que anteriormente não estava lá **(**BRADSKI; KAEHLER, 2008, tradução nossa).

Além de separar os objetos em primeiro plano a partir do resto da imagem, há muitas situações onde é necessário separar partes de objetos, tais como isolar apenas o rosto ou mãos de uma pessoa. É possível também pré-processar uma imagem em segmentos de uma imagem que contêm coisas como membros, cabelo, rosto, tronco, folhas de árvore, lago, caminho, gramado e assim por diante **(**BRADSKI; KAEHLER, 2008, tradução nossa).

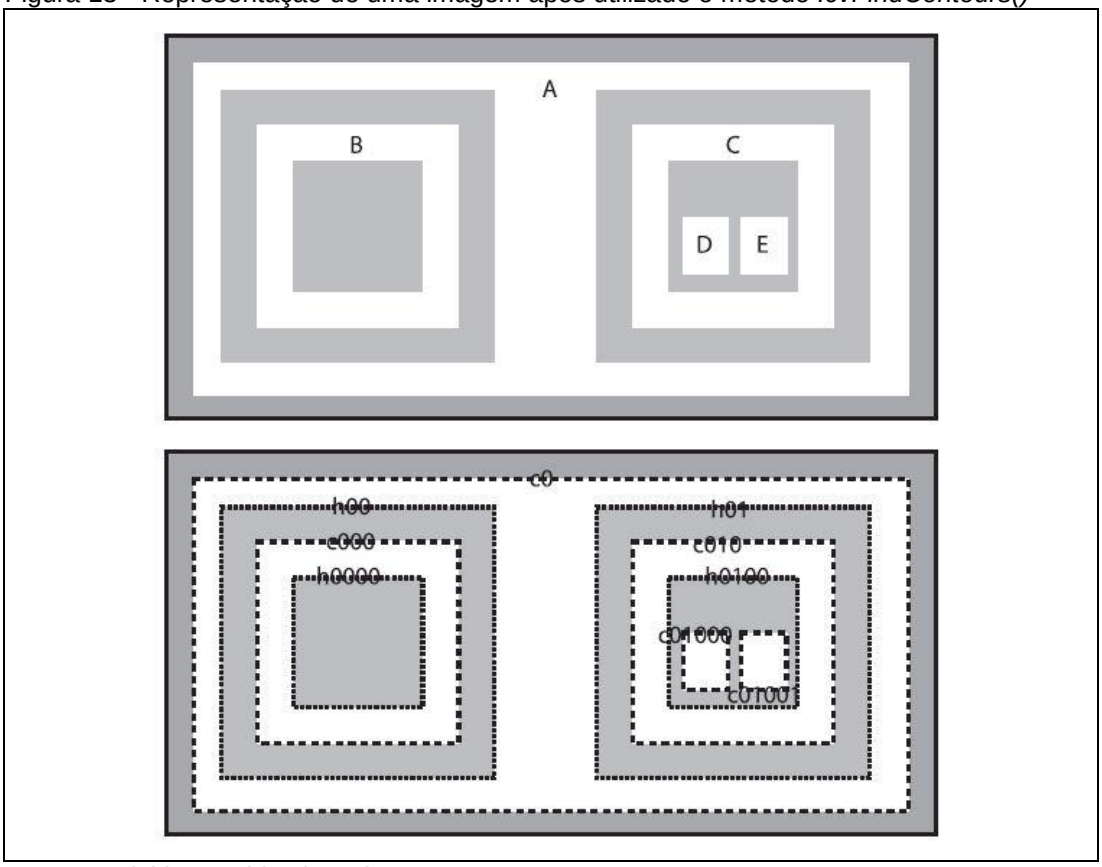
O nível do detalhamento durante um processo de segmentação depende de vários fatores, sendo alguns deles o tamanho e resolução da imagem e também da tarefa a ser executada. Um exemplo é quando se precisa procurar por uma casa em uma imagem obtida através de uma câmera estando no nível da rua ou de uma imagem obtida por um satélite. No primeiro caso, a região de interesse (casa) ocupa bastante espaço na imagem, enquanto que no segundo ela será muito menor. Mesmo que a tarefa tenha o mesmo objetivo sobre as duas imagens, as resoluções delas são distintas e precisam de um tratamento de imagens diferenciado na hora da segmentação (MARENGONI; STRINGHINI, 2009).

5.2 CONTORNOS

A biblioteca OpenCV é bastante robusta, e dentre dos vários métodos que ela disponibiliza, há um específico para encontrar os contornos da imagem, ou seja, é capaz de detectar os diferentes segmentos da imagem através de suas bordas e contornos. O método em questão é o *:cvFindCountours()* (BRADSKI; KAEHLER, 2008, tradução nossa).

Dentro do contexto de processamento de imagens a definição de um contorno é uma lista de pontos que representam de uma forma ou de outra, uma curva da imagem. No OpenCV estes contornos são representados por sequências onde cada entrada da sequência aponta para o próximo ponto na curva.

Figura 13 - Representação de uma imagem após utilizado o método *:cvFindContours()*



Fonte: Bradski e Kaehler (2008).

A figura 13 descreve como o método de achar contornos funciona. A parte superior da figura mostra uma imagem de teste que contém algumas regiões brancas em um fundo escuro. E a parte inferior retrata a mesma imagem, juntamente com os contornos que o *cvFindContours ()* irá encontrar. Os contornos podem ser

rotulados cX ou Hx, onde *c* significa *contour*, ou seja, contorno, *h* significa *hole*, ou seja, buraco, e *X* é um número. Dentre os contornos encontrados, alguns desses são linhas tracejadas, os quais representam o limite das regiões brancas (BRADSKI; KAEHLER, 2008, tradução nossa).

6 TRABALHOS CORRELATOS

Neste capítulo serão apresentados alguns trabalhos correlatos, pois é de suma importância que se tenha contato com metodologias utilizadas em outras experiências dentro ou próxima da mesma área de pesquisada, além da fundamentação teórica e metodologia.

6.1 SISTEMA AUTOMÁTICO PARA OBTENÇÃO DE PARÂMETROS DO TRÁFEGO VEICULAR A PARTIR DE IMAGENS DE VÍDEO USANDO OPENCV

A tese de doutorado desenvolvida por Cunha (2013) teve como objetivo o desenvolvimento de um sistema automático a fim de extrair dados de tráfego veicular a partir de um pós-processamento de vídeos.

A pesquisa se baseia no conceito de Visão Computacional, programação na linguagem C++ e a biblioteca OpenCV para o desenvolvimento da aplicação. Ela propõe duas etapas para a detecção de veículos, a primeira foi a modelagem do plano de fundo da imagem e a segunda a segmentação dos veículos.

Foi feita uma análise de desempenho dos métodos de segmentação e de subtração de fundo, e a partir do tempo de processamento e as taxas de acertos foi concluído que segundo método era mais adequado.

Após a definição do método de segmentação, foi desenvolvido outro método a fim de definir as trajetórias dos veículos. Comparando os parâmetros de tráfego obtidos do suposto sistema, obteve-se uma estimativa de acerto da velocidade média dos veículos de 92,7% comparando com as médias realizadas por um radar, mas por outro lado, a estimativa da taxa de fluxo de tráfego não foi bem sucedida devido a falhas na identificação da trajetória do veículo (CUNHA, 2013).

6.2 PLANEJAMENTO DE TRAJETÓRIA PARA ESTACIONAMENTO DE VEÍCULOS AUTÔNOMOS

A dissertação de mestrado de Prado (2013) abordou o desenvolvimento de um sistema inteligente que tem como objetivo gerar e executar uma trajetória planejada de caminho para o estacionamento de veículos autônomos em um ambiente semiestruturado.

O sistema foi dividido em três etapas, a localização da vaga de estacionamento livre, o planejamento da trajetória até a vaga escolhida, e por fim, o controle do veículo. Para detecção da vaga de estacionamento foi utilizado um laser. O sistema é responsável pela parte de planejamento até a vaga selecionada e também pelo controle do veículo, guiando o mesmo a até a vaga através do caminho planejado.

Segundo Prado (2013) o sistema desenvolvido é capaz de reconhecer as vagas de estacionamento por meio de sensores que estão instalados no veículo, gerando assim um caminho para uma vaga livre e também conduzindo o veículo controlando sua aceleração e seus comandos de esterçamento até a vaga.

6.3 UM PROTÓTIPO MÓVEL PARA DETECÇÃO AUTOMÁTICA DE PLACAS VEICULARES BRASILEIRAS

O trabalho de conclusão de curso de Weber (2013) teve como objetivo o desenvolvimento de um protótipo de baixo custo utilizando um *Raspberry Pi Model B* juntamente com uma câmera e mais um software escrito em C++ utilizando a biblioteca OpenCV, a fim de detectar as placas dos automóveis durante a locomoção de um veículo ao qual ele estiver acoplado.

A técnica para identificação das placas consiste na identificação de uma região específica da imagem. Após a região de interesse for localizada na imagem, é realizada uma etapa para reconhecimento dos caracteres. Tal identificação das placas pode ser utilizada para vários fins, tais como patrulhamento das ruas e até mesmo a localização de veículos roubados.

Com o protótipo acoplado no para-brisa do veículo, ele é capaz de captar a imagem onde está a placa. Após isso esta é enviada a um servidor externo para que seja realizado o processamento da imagem pelo software desenvolvido (WEBER, 2013).

Os resultados obtidos alcançaram uma média de 35% de acerto das placas visíveis, com um tempo médio de processamento de 0.09 segundos (WEBER, 2013).

6.4 USO DE VISÃO COMPUTACIONAL EM DISPOSITIVOS MÓVEIS PARA AUXÍLIO À TRAVESSIA DE PEDESTRES COM DEFICIÊNCIA VISUAL

A presente dissertação de mestrado, elaborada por Sousa (2013), teve como objetivo o desenvolvimento de um aplicativo utilizando a biblioteca OpenCV para o tratamento de imagens e a plataforma Android para dispositivos móveis que auxilie deficientes visuais na travessia de ruas.

Foi feita uma análise em cima dos desafios e uma busca por soluções para o desenvolvimento de interfaces móveis acessíveis e possíveis de processar as imagens no intuito de reconhecer as pessoas e as faixas de pedestres (SOUSA, 2013).

Sobre a usabilidade de uma aplicação deste tipo, foi realizada uma pesquisa que mostrou a aceitação de aparelhos celulares e foi constatado que já há alguns aplicativos voltados para cegos, tais como presença de luz para cegos totais, notas de dinheiro, cores e também de reconhecimento de texto impresso.

O funcionamento do sistema é ativado pelo usuário através de um toque, o software então faz um reconhecimento do ambiente para a detecção da faixa de pedestres. Ao reconhecer a faixa o sistema informa ao usuário através da vibração do aparelho ou por sintetizador de voz se existe uma faixa de pedestre e se existem pessoas neste caminho (SOUSA, 2013).

6.5 ESTACIONAMENTO INTELIGENTE

No trabalho de conclusão de curso de Alhak (2011) foi apresentado um protótipo de estacionamento inteligente a fim de mostrar aos motoristas onde há vagas livres com um display na entrada do mesmo mostrando quantas vagas estão disponíveis e em quais setores se encontram. Chegando no setor, outro display mostra a localização das vagas.

Cada vaga possui dois LEDs indicativos, sendo um vermelho indicando que a vaga está ocupada, e outro verde que indica que está disponível. A detecção da vaga é feita por um sensor infravermelho (ALHAK, 2011).

O objetivo deste trabalho foi desenvolver um protótipo de baixo custo de implementação, tecnologia acessível, facilitando os motoristas na hora de encontrar uma vaga livre, evitando o desperdício de tempo e também de gasolina.

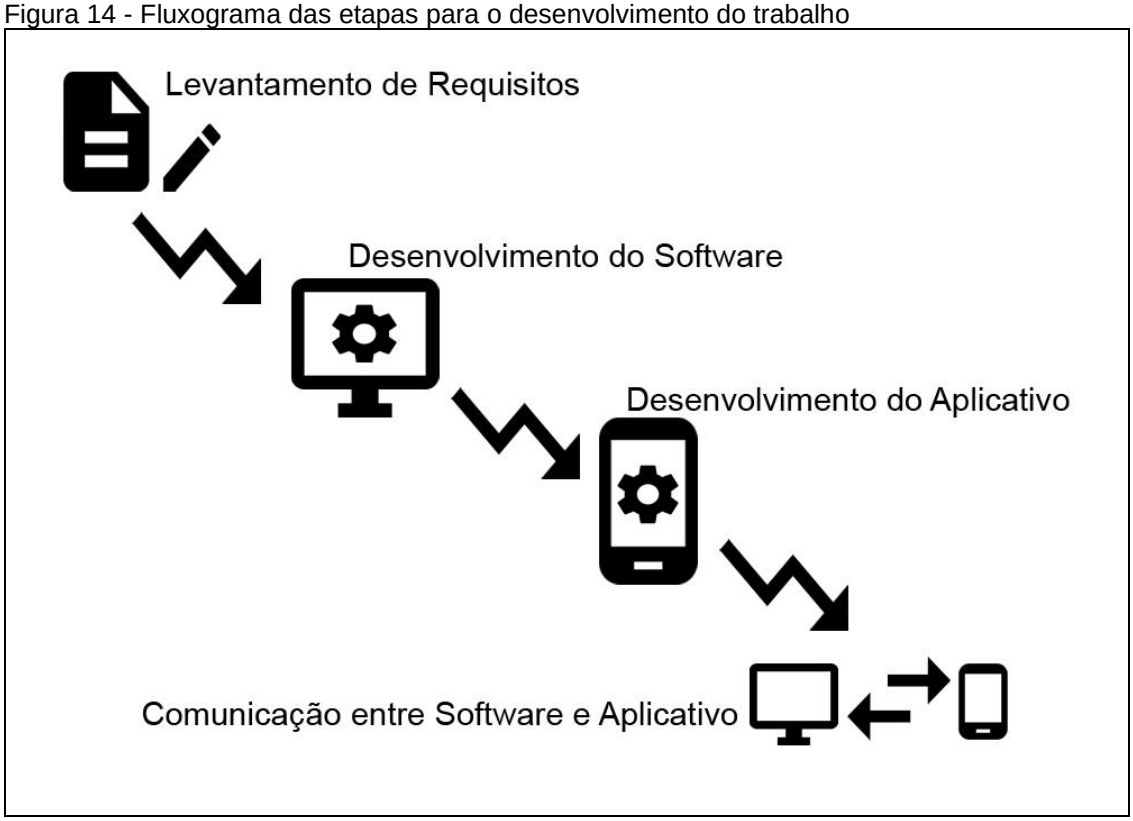
7 SOFTWARE DE DETECÇÃO DE VAGAS DE ESTACIONAMENTO E APLICATIVO ANDROID PARA VISUALIZAÇÃO

Por meio dos conhecimentos adquiridos no estudo de visão computacional, da biblioteca OpenCV e também sobre a plataforma Android, no presente trabalho de conclusão de curso foi possível o desenvolvimento de um protótipo constituído por um software desenvolvido na linguagem Java para a identificação das vagas de estacionamento e um aplicativo onde é possível a visualização do local das vagas livres e ocupadas. Tendo em vista que a mobilidade urbana está cada vez mais complicada, o intuito deste projeto é a criação de algo de fácil implantação e útil para o dia-a-dia das pessoas.

7.1 METODOLOGIA

A metodologia para o desenvolvimento do trabalho foi constituída a partir das etapas mostradas na figura 14, ou seja, quatro etapas, onde a primeira delas foi o levantamento de requisitos com base em outros trabalhos com objetivos semelhantes ao apresentado e no estudo realizado neste, visando encontrar uma maneira para a identificação de vagas de estacionamento. A segunda etapa se constitui do entendimento da biblioteca OpenCV e do conceito de visão computacional para assim ser elaborado o protótipo do software para detecção das vagas. A terceira etapa se caracteriza pelo desenvolvimento do aplicativo destinado a dispositivos com sistema operacional Android. E, por fim, na quarta etapa foi feita a integração entre software e aplicativo, tornando possível a visualização das vagas livres e ocupadas.

Figura 14 - Fluxograma das etapas para o desenvolvimento do trabalho



Fonte: Do autor.

7.1.1 LEVANTAMENTO DE REQUISITOS

A etapa de levantamento de requisitos compreendeu o estudo sobre o conceito de visão computacional, a fim de compreender a teoria para, então, aplicá-la no desenvolvimento do software para a detecção de vagas de estacionamento. Além disso, também foi realizada uma pesquisa sobre projetos desenvolvidos a partir da mesma tecnologia que seria utilizada para este trabalho, no caso a biblioteca de tratamento de imagens OpenCV. A partir dessa pesquisa foi possível o levantamento dos requisitos iniciais para o desenvolvimento do protótipo.

Ainda na etapa do levantamento de requisitos, realizou-se uma pesquisa acerca da biblioteca de processamento de imagens OpenCV, que é uma das bibliotecas mais utilizadas nessa área (como já abordado no item 5 do presente trabalho). Além disso, ela é *Open Source*, ou seja, possui seu código disponível para que qualquer pessoa possa utilizá-la, fornecendo também uma ampla documentação onde foi possível estudar e compreendê-la.

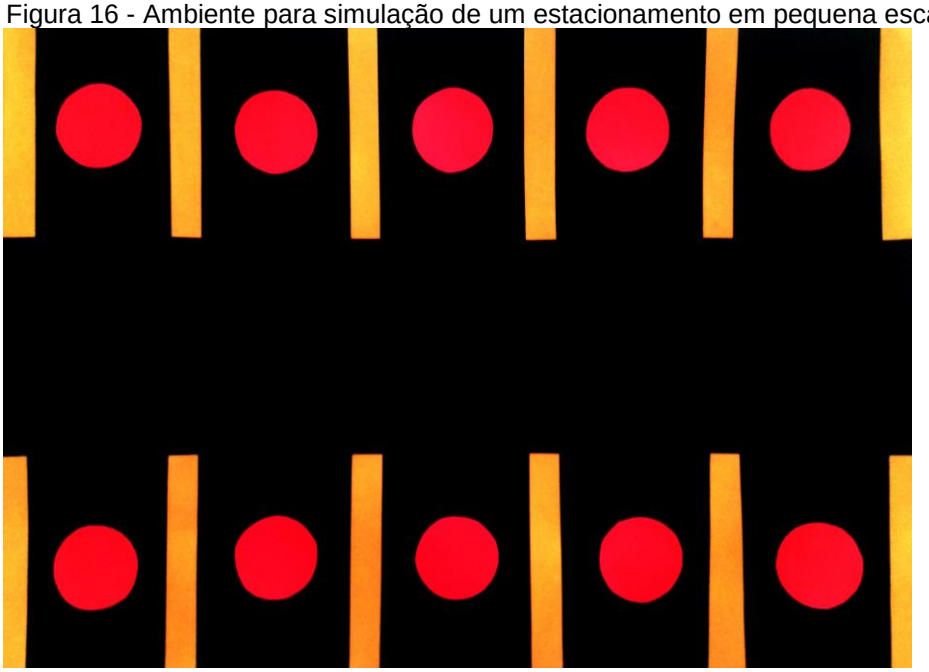
Para que fosse possível o desenvolvimento do protótipo, foi adquirida uma câmera *webcam* com interface USB, que pode ser visualizada na figura 15. Este dispositivo foi utilizado para a captura do vídeo, pois o intuito do projeto é a realização de tratamento de imagens em tempo real para a detecção das vagas de estacionamento, tornando possível que o usuário possa visualizar a localização de vagas livres onde possa estacionar seu carro, sem ter que ficar dando voltas dentro de um determinado estacionamento em função da procura.

Figura 15 – Webcam

Fonte: Do autor.

Com isso, após a aquisição da *webcam* se fez necessário a criação de um ambiente para a simulação de um estacionamento em pequena escala, mostrado na figura 16, para assim ser desenvolvido o software do protótipo.

Figura 16 - Ambiente para simulação de um estacionamento em pequena escala



Fonte: Do autor.

A partir da pesquisa realizada, a maneira encontrada para a identificação das vagas de estacionamento se fez através de marcadores e identificação de cores. Como mostra a figura 16, no centro de cada vaga há um círculo vermelho, sendo que o objetivo é a detecção desse círculo a partir de sua cor, ou seja, se a câmera detectá-la significa que a vaga está livre. Caso contrário, será considerado que há um carro naquela vaga, pois este estaria por cima do marcador, impossibilitando a câmera de visualiza-lo.

Com a ideia formada de como seria realizada a identificação das vagas, o próximo passo foi escolher uma linguagem. Apesar de a biblioteca OpenCV ter sido escrita em C/C++, a linguagem escolhida para este trabalho foi Java, por ser orientada a objetos, possuir sintaxes parecidas com a linguagem nativa do OpenCV e também pelos recursos de rede pensando na comunicação com o aplicativo.

7.1.2 DESENVOLVIMENTO DO SOFTWARE

Para o início do desenvolvimento do software, foi preciso encontrar um ambiente de desenvolvimento integrado (IDE), o qual seria utilizado em conjunto com a biblioteca OpenCV. Tendo em vista que a linguagem escolhida foi Java, a plataforma escolhida foi o *NetBeans*, por ser gratuita e também multiplataforma,

além de oferecer as ferramentas necessários para o desenvolvimento de um software voltado a desktop.

Escolhida a IDE para desenvolvimento do software, foi necessário baixar a biblioteca OpenCV para poder utilizá-la no projeto, bastando então adicionar ao projeto o arquivo de formato *Java Archive* (JAR), que é um arquivo compactado utilizado para armazenar classes já compiladas e metadados, fornecendo assim acesso aos métodos disponibilizados pela biblioteca.

Com o projeto criado e pronto para o desenvolvimento, o próximo passo foi pesquisar e realizar testes para se familiarizar com a biblioteca OpenCV, criando uma interface onde seria mostrado o vídeo capturado pela câmera em tempo real, e então o código para identificação das vagas. Esta parte tornou-se o maior desafio do trabalho, pois o tratamento de imagens é bastante complexo tanto na teoria quanto na prática, mesmo tendo uma vasta documentação, e por esse mesmo motivo tornou-se mais difícil encontrar um meio de desenvolver o software cumprindo os objetivos deste trabalho.

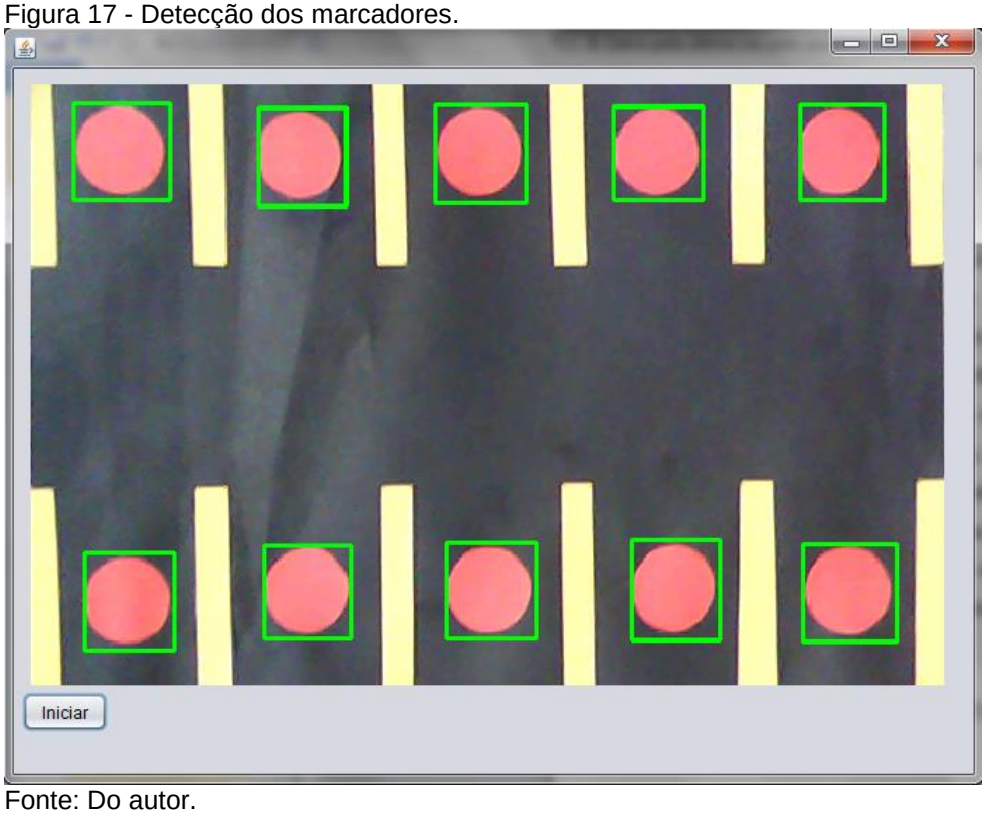
Tendo em vista que a biblioteca oferece mais de 500 métodos, e após ter compreendido os fundamentos de visão computacional, o próximo passo foi encontrar uma maneira para tornar possível a identificação das vagas de estacionamento, e também detectar se estão livres ou ocupadas.

A aplicação foi desenvolvida no *NetBeans IDE 8.0.2* na linguagem Java, e foi dividida em três classes:

a) view: esta classe é responsável pela interface do trabalho;
b) cameraView: responsável por renderizar a imagem capturada pela câmera e ficar atualizando constantemente na interface;
c) processaCaptura: esta classe é a mais importante, pois ela quem processa as imagens e identifica as vagas.

Pensando prioritariamente nas vagas onde carros seriam estacionados, a maneira encontrada para a identificação das vagas se fez através de marcadores no centro de cada uma delas. Para o desenvolvimento do protótipo em pequena escala, foram utilizados marcadores vermelhos para atuarem como os marcadores, assim o software teria a função de localizá-los. Caso sim significaria que a vaga estaria livre, caso contrário um carro estaria em cima do marcador, ocupando a vaga e impedindo que o software detecte o círculo vermelho.

Figura 17 - Detecção dos marcadores.



Fonte: Do autor.

Conforme mostra a figura 17 o funcionamento do software se baseia na busca pelos marcadores. O método utilizado para mostrar os retângulos diretamente no vídeo capturado pela câmera é o *Core.rectangle*. Nele é preciso referenciar quatro parâmetros, o primeiro deles é a matriz da imagem capturada, o segundo são as coordenadas do ponto superior esquerdo do contorno encontrado, o terceiro são as coordenadas do ponto inferior direito do contorno encontrado, e o quarto se compõe da escala de cor desejada para a criação do retângulo, no caso foi utilizada a cor verde, e também a espessura do mesmo.

Mas antes de poder utilizar este método foi preciso aplicar filtros e métodos a fim de poder identificar esses marcadores. Após isso foi possível capturar os contornos dos marcadores e enfim aplicar a função para a criação do retângulo.

Dentre os métodos utilizados, o principal deles foi o *Core.inRange* , o qual é responsável por detectar a escala de cor informada, no caso o vermelho. Esta parte foi onde surgiu outro grande desafio, pois estaria detectando pequenos pontos indesejados na imagem. A fim de resolver este problema, foram aplicados então outros métodos e filtros para remover os maiores detalhes da imagem, tais como o *Imgproc.erode* e o *Imgproc.dilate.* Estes dois utilizados em sequência atuam como um filtro onde é possível remover pequenos detalhes da imagem, possibilitando a

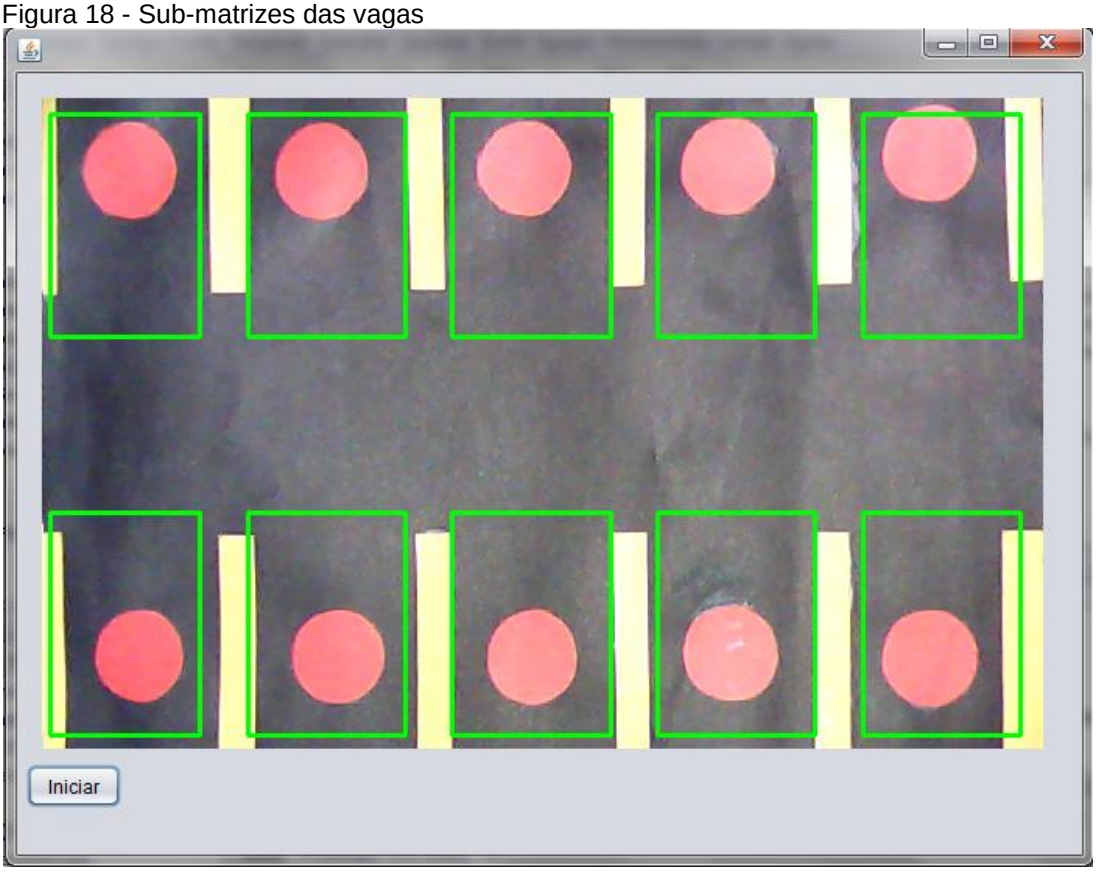
continuação do desenvolvimento do trabalho, aplicando então a função *findCountors* para identificação dos contornos encontrados pelo método *Core.inRange*.

Ambas as funções para erodir e dilatar a imagem tomam como parâmetros uma matriz de imagem origem e uma de destino. O terceiro parâmetro é o kernel, o padrão deste é nulo, mas o utilizado neste trabalho foi o retangular *CV_SHAPE_RECT*. E o quarto parâmetro é o número de iterações, que caso não seja definida, o padrão é um. Assim torna-se possível a remoção de ruídos indesejados na imagem, pois as regiões que contêm um conteúdo maior não são afetadas.

A próxima etapa do projeto foi encontrar uma maneira de saber a localização das vagas, ou seja, saber qual a vaga número um, qual a vaga número dois, e assim por diante. Esta etapa fez-se necessária porque até então só estavam sendo detectados os marcadores em vermelho, não sendo possível ainda saber a posição das vagas livres e das que estavam ocupadas.

Desta forma a partir da pesquisa realizada para o desenvolvimento do trabalho e dos objetivos propostos, foi necessário a criação de sub-matrizes a partir da matriz principal que é formada pela imagem capturada pela *webcam*, onde cada uma delas representaria uma vaga.

Figura 18 - Sub-matrizes das vagas

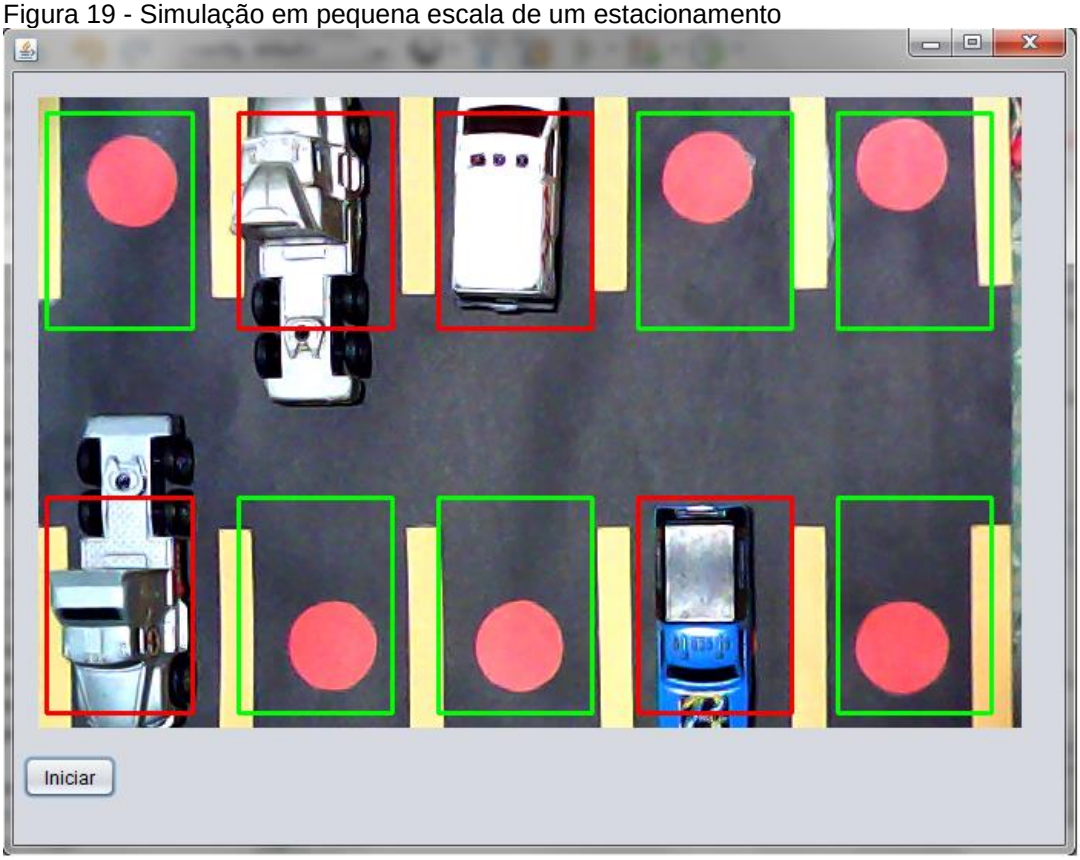


Fonte: Do autor.

Conforme mostra a figura 18, cada vaga corresponde a uma sub-matriz, e somente nelas é realizada a verificação para saber se a vaga está livre ou ocupada. Como não há nenhum carro e o software está detectando todos os marcadores, a sub-matriz está recebendo uma marcação verde, caso contrário receberia uma marcação vermelha, que é a indicação de a vaga está ocupada.

As sub-matrizes estão fixas no software para poder saber a localização de cada vaga, portanto o posicionamento da câmera é muito importante para um bom funcionamento. Na figura 19 isto pode ser observado com mais clareza, onde há carros nas vagas simulando em pequena escala um estacionamento real. Os carros estão impedindo o software de detectar os marcadores, sendo que quando isso ocorre entende-se que a vaga está ocupada.

Figura 19 - Simulação em pequena escala de um estacionamento



Fonte: Do autor.

7.1.3 DESENVOLVIMENTO DO APLICATIVO

Esta etapa do trabalho é muito importante, pois é através do aplicativo que o usuário poderá visualizar a localização das vagas livre e ocupadas. O software mesmo possuindo uma interface, seu principal e único objetivo é o tratamento da imagem para identificação das vagas.

Por meio das pesquisas realizadas sobre dispositivos móveis, foi visto que *smartphones* são muito utilizados no dia-a-dia das pessoas, e que grande parte deles possui o sistema operacional Android, esse foi o principal motivo pela escolha dele para o desenvolvimento do aplicativo deste trabalho.

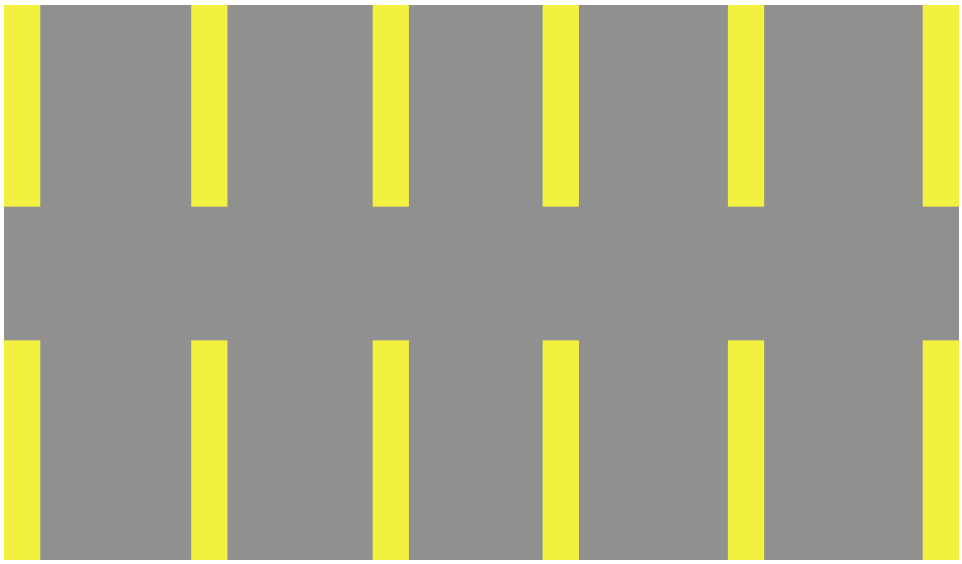
Primeiramente para o início do desenvolvimento do aplicativo, assim como para o software, foi necessário a escolha de uma plataforma para a elaboração do mesmo. Duas delas são muito conhecidas e utilizadas por desenvolvedores, o *Eclispe*, e o *Android Studio,* ambos são de licença gratuita.

O *Eclipse* é uma plataforma para desenvolvimento em *Java,* mas que suporta várias outras linguagens a partir de plugins. Já o *Android Studio* é

especializado para a criação de aplicativos Android, com várias ferramentas específicas e úteis, isto fez com que fosse escolhido para o desenvolvimento deste trabalho.

Com a plataforma de desenvolvimento baixada e instalada, o próximo passo foi pensar em uma maneira em que o usuário consiga visualizar as vagas e identifica-las com clareza, pensando nisso foi criado uma imagem, sendo semelhante a do protótipo do estacionamento em pequena escala, como mostra na figura 20.

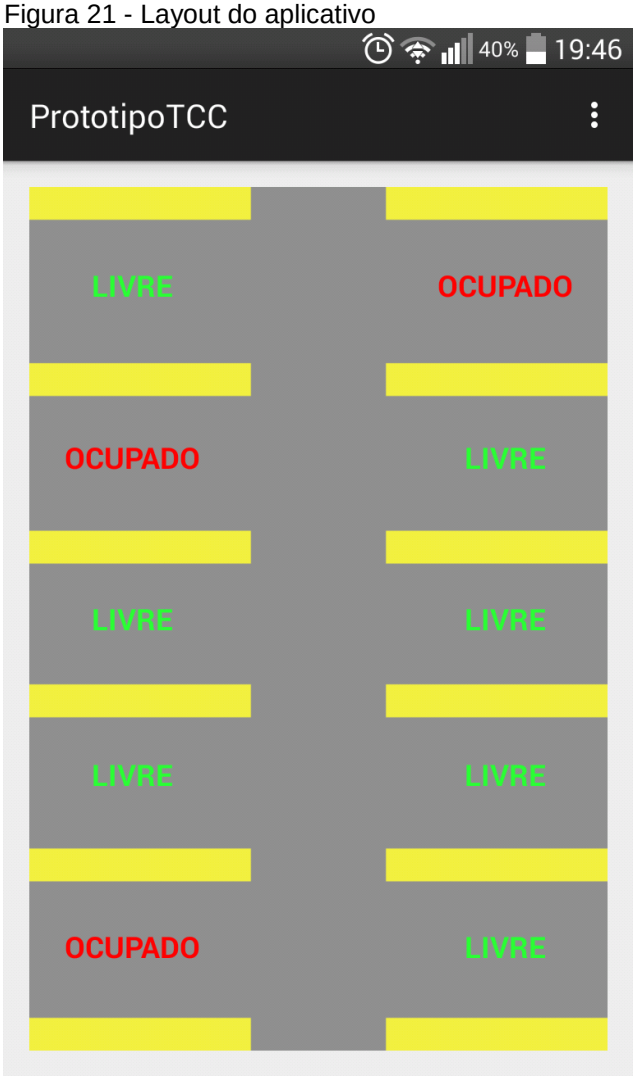
Figura 20 - Imagem semelhante ao do estacionamento para o aplicativo



Fonte: Do autor.

A figura 20 foi aplicada como uma imagem de fundo do aplicativo, e então *labels* foram adicionados em cada uma das vagas, indicando se estão livres ou ocupadas.

Figura 21 - Layout do aplicativo



Fonte: Do autor.

A figura 21 mostra o layout do aplicativo, que possui uma imagem semelhante ao estacionamento em escala reduzida utilizado para este trabalho, tornando simples a visualização das vagas, pois se pode identificar a localização das mesmas e seu status, ou seja, se estão livres ou ocupadas.

Com o layout do aplicativo pronto, a próxima foi encontrar uma maneira para que o aplicativo se comunicasse com o software, obtendo as informações das vagas, e atualizando as mesmas em tempo real.

7.1.4 INTEGRAÇÃO ENTRE SOFTWARE E APLICATIVO

Existem algumas maneiras para se realizar a comunicação entre softwares e aplicativos, como utilização de banco de dados ou *Web Services*, isso depende muito da situação e o objetivo de ambos. No caso deste trabalho, por se tratar de um protótipo, decidiu-se utilizar *Socket* para realizar esta comunicação.

A utilização de *Socket* para um cenário real não é viável, pois é necessário que tanto a máquina que esteja com o software rodando, quanto o aplicativo, esteja conectada na mesma rede. Este tipo de comunicação utiliza endereço de IP e um número de porta, com base nesse endereço é feito a entrega, ou troca de pacotes de dados.

De forma geral, uma aplicação faz o papel de servidor abrindo um *Socket* na rede, ou seja, um número de porta, e outra aplicação faz o papel de cliente, que acessa esta porta juntamente com o endereço de IP do servidor. Com isto, é possível o envio de pacotes de dados, tanto do servidor para o cliente, quanto do cliente para o servidor sem a necessidade de um local para armazenamento de dados.

No caso deste trabalho o software para detecção das vagas de estacionamento fará o papel de servidor, pois ele que possui as informações das vagas e fará o envio delas para o aplicativo, onde estas serão abstraídas e transformadas de forma que possam ser visualizadas.

Dentro do software foi criado um novo pacote, chamado de *serverSocket*, e dentro dele possui duas classes, a *Server*, que possui o método que abre um número de porta e espera a conexão do cliente, e a *Sender*, que envia as informações para o cliente conectado.

Figura 22 - Código para abrir um número de porta na rede

```
1  try {
2      serverSocket = new ServerSocket(4444);
3      System.out.println("Server Iniciado. Escutando a porta 4444. Esperando conexão do cliente.");
4      clientSocket = serverSocket.accept();
5      System.out.println("Cliente conectado na porta 4444.");
6  } catch (IOException e) {
7      System.out.println("Não foi possível abrir a porta : 4444");
8      e.printStackTrace();
9      return;
10 }
```

Fonte: Do autor.

A figura 22 mostra o código utilizado para abrir um número de porta na rede, no caso a porta 4444, e então espera o cliente se conectar a porta, capturando o endereço de IP do cliente para então enviar as informações diretamente para ele.

No lado do aplicativo, que fará o papel do cliente, basta informar o endereço de IP do servidor e o número da porta, após isso será identificado e receberá as informações das vagas, e os dados abstraídos para que o usuário visualize em tempo real as vagas de estacionamento.

Como neste trabalho o protótipo do estacionamento possui apenas dez vagas, as informações destas foram concatenadas de modo que o software envia apenas uma mensagem contendo os dados sobre todas as vagas, mantendo sempre o aplicativo atualizado acerca de todas elas.

7.2 RESULTADOS OBTIDOS

A partir da etapa de levantamento de requisitos realizada neste trabalho de conclusão, foi possível conhecer os principais conceitos sobre visão computacional e as possibilidades que a biblioteca OpenCV oferece para tratamento de imagens. De um entendimento sobre mobilidade urbana, os problemas e soluções já existentes quando o assunto é estacionamento e suas respectivas vagas. E também sobre dispositivos móveis, que na atualidade está se tornando algo indispensável devido aos recursos que ele oferece.

O estudo realizado neste trabalho possibilitou o desenvolvimento de um protótipo para detecção de vagas de estacionamento através de câmeras e tratamento de imagens, identificando quais estão livres ou ocupadas, e também um aplicativo destinado a Android, onde os usuários possam visualizar o estacionamento e as vagas livres, evitando gastos com combustível em função da procura e também um possível congestionamento.

Os trabalhos existentes na área de visão computacional utilizando a biblioteca OpenCV são inúmeros, alguns destes envolvem o tema mobilidade urbana, tais como controle de fluxo de tráfego, identificação de placas de veículos, entre outros. É uma biblioteca largamente utilizada devido as funcionalidades que ela oferece e também por ser de licença livre.

Na atualidade as tecnologias utilizadas para o mesmo propósito deste trabalho possuem um custo altíssimo de implantação, envolvendo circuitos com

sensores em todo o estacionamento, máquinas pesadas que posicionam automaticamente os automóveis nas vagas, dentre outras.

É importante ressaltar que, todos os objetivos propostos por este trabalho de conclusão foram atingidos por meio do desenvolvimento do protótipo, composto pelo software que com auxílio de imagens capturadas por uma *webcam* e também da biblioteca OpenCV, que possibilitou a identificação das vagas de estacionamento, e de um aplicativo para visualização. Sendo que para a utilização em um ambiente real, são necessárias modificações, mas é importante salientar que é possível manter a mesma maneira em que as vagas são identificadas, através de marcadores, mesmo que o veículo seja da mesma cor do marcador, pois é possível capturar o tamanho do conteúdo utilizando os recursos que a biblioteca oferece, e em um ambiente real, o veículo possuirá um conteúdo muito maior do que o respectivo marcador.

8 CONCLUSÃO

Os conceitos de processamento de imagens e visão computacional podem ser utilizados de diversas formas e em diversas áreas, tais como identificação de objetos, realidade virtual, reconhecimento de movimentos, reconhecimento facial, dentre outras. As possibilidades de desenvolvimento de aplicações envolvendo esta tecnologia são infinitas quando em conjunto com a criatividade.

A biblioteca OpenCV utilizada neste projeto para auxiliar no processamento de imagens, apesar de sua grande extensão, foi possível abstrair um pouco dos recursos que ela oferece, possibilitando o desenvolvimento do protótipo proposto.

Ao longo do estudo deste trabalho de conclusão, foi possível o desenvolvimento de um protótipo para a detecção de vagas de estacionamento através de imagens capturadas por uma *webcam*, realizando tratamento das imagens com auxílio da biblioteca OpenCV, identificando se estão livres ou ocupadas, e também de um aplicativo destinado a Android para que possa ser feita a visualização das vagas.

Para continuidade da pesquisa, tem-se como possibilidade de trabalhos futuros, aprimorar o protótipo para que possa ser aplicado em um cenário real, realizando melhoras no software, no aplicativo e mudar o modo de comunicação entre os dois para que vários usuários possam se conectar.

REFERÊNCIAS

ADOBE HELP RESOURCE CENTER. **Bitmaps.** Disponível em: <http://help.adobe.com/en_US/Director/11.0/help.html?content=06_bitmaps_01.html >. Acesso em: 15 maio 2015.

ALHAK, Said Hikmat Abd. **Estacionameto Inteligente**. 2011. 76 f. TCC(Graduação) - Curso de Engenharia da Computação, Centro Universitário de Brasília – Uniceub Fatecs – Faculdade de Tecnologia e Ciências Sociais Aplicadas Curso de Engenharia de Computação, Brasília, 2011.

AUGUSTO, Filipe Manzi. **Aplicação de Visão Computacional para Extração de Características de Imagens do Olho Humano.** 2007. 47 f. TCC (Graduação) - Curso de Engenharia de Computação, Escola de Engenharia de São Carlos, São Carlos, 2007.

BRADSKI, Gary; KAEHLER, Adrian. **Learning OpenCv**. Sebastopol: O'reilly Media, 2008. 571 p.

BRAHMBHATT, Samarth. **Pratical OpenCv**. New York: Technology In Action, 2013. 229 p.

BURNETTE, Ed. Hello, Android: **Introducing Google's Mobile Development Plataform**. Dallas: The Pragmatic Bookself, 2008. 247 p.

CET. Companhia de Engenharia de Tráfego (1979) *Um estudo sobre os problemas de estacionamentos de veículos.* Boletim Técnico 21, São Paulo.

COSTA, Norben P. O.; DUARTE FILHO, Nemésio F.. ANÁLISE E AVALIAÇÃO FUNCIONAL DE SISTEMAS OPERACIONAIS MÓVEIS: VANTAGENS E DESVANTAGENS.**Revista de Sistemas e Computação,** Salvador, v. 3, n. 1, p.66-67, jun. 2013.

CTB - Código de Trânsito Brasileiro. Lei Federal n° 9503/97.

CUNHA, André Luiz Barbosa Nunes da. **Sistema Automático para Obtenção de Parâmetros do Tráfego Veicular a partir de Imagens de Vídeo usando OpenCv**. 2013. 128 f. Curso de Escola de Engenharia de São Carlos, Universidade de São Paulo, São Paulo, 2013.

ELEFANTEVERDE. **A (des) importância da mobilidade urbana**. 2013. Disponível em: <http://www.megaportalcriciuma.com.br/2011/images/stories/noticias/criciuma-online/janeiro2012/recadatramento-cartao-do-estudante.jpg>. Acesso em: 23 set. 2014.

GARTNER. **Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013**. 2014. Disponível em: < http://www.gartner.com/newsroom/id/2665715> Acesso em: 10 nov. 2014.

GONÇALVES, Matheus. Android x iOS - **Finalmente uma comparação imparcial**. Disponível em: <http://toad.geek.com.br/posts/13778-android-x-ios-finalmente-uma-comparacao-imparcial>. Acesso em: 13 maio 2014.

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento De Imagens Digitais**, São Paulo: Edgard Blücher, 2000.

GRANGE, Sébastien; FONG, Terrence; BAUR, Charles. **TLIB: a Real-time Computer Vision Library for HCI**. Sydney: Swiss Federal Institute Of Technology, 2003.

GUIMARÃES, Gleyser. **A história do sistema operacional Android.** 2013. Disponível em: <http://www.dsc.ufcg.edu.br/~pet/jornal/agosto2013/materias/historia_da_computacao.html>. Acesso em: 11 nov. 2014.

IBGE. **Introdução ao processamento digital de imagens**. Rio de Janeiro: IBGE, 2001. 92p.

JIPPING, Michael J. **Smartphone Operating System Concepts with Symbian OS**. Revisão Attila Vamos et al. Chichester, Inglaterra: John Wiley & Sons Ltd, 2007. 356 p.

JOHNSON, Thienne M. **Java para Dispositivos Móveis:** Desenvolvendo Aplicações com J2ME. São Paulo: Novatec, 2007. 336 p.

KNEIB, Erika Cristine. **Mobilidade urbana e qualidade de vida: do panorama geral ao caso de Goiânia.** 2012**.** Disponível em: <http://www.proec.ufg.br/revista_ufg/julho2012/arquivos_pdf/09.pdf>. Acesso em: 06 nov. 2014.

LEE, Valentino; SCHNEIDER, Heather; SCHELL, Robbie. **Aplicações móveis**: Arquitetura, projeto e desenvolvimento. Tradução: Amaury Bentes e Deborah Rüdiger. Revisão técnica: Renato Haddad. São Paulo: Person Education do Brasil, 2005. 330 p.

LINDAU, Luis Antonio. **Mobilidade urbana**. 2014. Disponível em: <http://embarqbrasil.org/node/136>. Acesso em: 24 set. 2014.

MARENGONI, Maurício; STRINGHINI, Denise. Tutorial: Introdução à visão computacional usando opencv. **Revista de Informática Teórica e Aplicada**, Porto Alegre v. 16, n.1, p. 125-160, 2009.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento Digital de Imagens.** Rio de Janeiro: Brasport, 1999. 410 p.

METALICA. **Estacionamentos Verticais.** 2011. Disponível em: < http://www.metalica.com.br/o-que-sao-e-como-funcionam-os-estacionamentos-verticais>. Acesso em: 29 set. 2014

MEXAS, Antonio Henrique. **PROCESSO DE RECONHECIMENTO NÃO SUPERVISIONADO DE ÁREAS DE ESTACIONAMENTO.** 2014. 71 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Presbiteriana Mackenzie, São Paulo, 2014.

MIRANDA, Jeferson José. MOVE-IN: **Uma API para interatividade com a webcam**. Centro de Ciências Tecnológicas / Universidade do Estado de Santa Catarina, 2008 Disponível em: http://www2.joinville.udesc.br/~larva/portal/uploads/TCC-II. Jeferson. J.M.pdf Acesso em 15 out. 2014.

NUNES, Juliana Lopes; JACQUES, Maria Alice Prudêncio. **Definição de um Procedimento para o Dimensionamento de Estacionamentos em Instituições de Ensino Superior**. Brasília: Universidade de Brasília, 2004.

OGLIARI, Ricardo da Silva; BRITO, Robison Cris. **Android do Básico ao Avançado**. Rio de Janeiro: Editora Ciência Moderna Ltda, 2014. 397 p.

PRADO, Marcos Gomes. **Planejamento de tragetória para estacionamento de veículos autônomos.** 2013. 84 f. Dissertação (Mestrado) - Curso de Ciências da Computação e Matemática Computacional, Icmc-usp, São Carlos, 2013.

PUERARI, André Farias et al. **Estacionamentos Inteligentes como Solução ao Congestionamento de Grandes Centros Urbanos**. Blumenau: Cobenge, 2011.

REBOUÇAS, Fernando. **Mobilidade urbana.** InfoEscola: navegando e aprendendo. Transporte. [entre 2006 e 2014]. Disponível em: <http://www.infoescola.com/transporte/mobilidade-urbana/>. Acesso em: 23 set. 2014.

SANTA CATARINA. DETRAN-SC. **Frota de Veículos no Estado**. Disponível em: <http://www.detran.sc.gov.br/index.php/estatistica/veiculos>. Acesso em: 11 ma. 2014.

SETCHELL, Christopher John. **Applications of Computer Vision to Road-tra**. 1997. 170 f. Curso de Faculty Of Engineering, Department Of Electrical And Electronic Engineering, University Of Bristol, Bristol, 1997.

SOUSA, Kelly Aparecida Oliveira. **USO DE VISÃO COMPUTACIONAL EM DISPOSITIVOS MÓVEIS PARA AUXÍLIO À TRAVESSIA DE PEDESTRES COM DEFICIÊNCIA VISUAL.** 2013. 85 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Presbiteriana Mackenzie, São Paulo, 2013.

WEBER, Henrique. **Um protótipo móvel para detecção automática de placas veiculares brasileiras.** 2013. 44 f. TCC (Graduação) - Curso de Ciências da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2013.

APÊNDICE(S)

APÊNDICE A – ARTIGO CIENTÍFICO

Protótipo Android para Controle de Vagas de Estacionamento a partir de Tratamento de Imagens em Tempo Real com a Biblioteca Opencv

Renato Mafioletti Macarini[1], Sérgio Coral[2]

[1]Acadêmico do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias (UnaCET) – Universidade do Extremo Sul Catarinense (UNESC)
Av. Universitária, 1105 - Bairro Universitário – Criciúma – SC – Brasil

[2]Professor do Curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias (UnaCET) – Universidade do Extremo Sul Catarinense (UNESC)
Av. Universitária, 1105 - Bairro Universitário – Criciúma – SC – Brasil

renatomacarini@hotmail.com, sergiocoral@unesc.net

Abstract. *The large number of vehicles on the streets is a problem today, causing problems such as congestion, and also difficulties in finding parking spaces to park the car. There are technologies that seek to mitigate these problems, but have high implementation costs. This paper describes the study of computer vision, as well as OpenCV library which is a library for image processing, developing a software to detect parking spaces, and in order that a large part of the population owns a smartphone, and it is becoming essential day-to-day lives because of the features it offers, the development of a prototype for the Android operating system for visualization of vacancies.*

Resumo. *O grande número de veículos nas ruas é um problema na atualidade, causando problemas como congestionamentos, e também dificuldades na procura de vagas de estacionamento para poder estacionar o carro. Existem tecnologias que buscam amenizar estes problemas, porém possuem alto custo de implantação. Este artigo descreve o estudo sobre visão computacional, bem como a biblioteca OpenCV, que é uma biblioteca para processamento de imagens, o desenvolvimento de um software para detecção de vagas de estacionamento, e tendo em vista que grande parte da população possui um smartphone, e que ele está se tornando indispensável no dia-a-dia das pessoas devido aos recursos que oferece, o desenvolvimento de um protótipo destinado ao sistema operacional Android para a visualização das vagas.*

1. Introdução

Com a introdução de processos eletrônicos ao dia-a-dia das pessoas, para que seja possível acompanhar este ritmo de vida cada vez mais acelerado, onde são necessários deslocamentos até o trabalho, faculdade, banco, academia, entre outros locais que fazem parte da rotina diária, é necessário que estas locomoções se iniciem, muitas vezes, horas antes do planejado. Paralelamente, há o trânsito caótico, que frequentemente acarreta em engarrafamentos, panorama que resulta em outro problema que está cada vez mais comum, o de encontrar vagas em estacionamentos (ALHAK, 2011).

Uma grande parte da população opta por utilizar o serviço de estacionamentos privados mediante a realização de pagamento, onde deixam seu automóvel enquanto trabalham, realizam compras ou fazem outro tipo de tarefa. Porém, não raras vezes os estacionamentos já estão praticamente lotados, gerando dificuldade para a alocação de vagas para os veículos dos usuários devido ao fluxo existente dentro de grandes estacionamentos, tais como os de instituições de ensino, supermercados, shoppings, entre outros (PUERARI et al, 2011).

Atualmente há um constante aumento da quantidade de veículos em circulação nas áreas urbanas, causando conflitos entre usuários de veículos automotores e a disponibilidade de espaços destinados para estacionar. Segundo dados estatísticos do DETRAN-SC, em 2003 a frota de veículos no estado de Santa Catarina era de 1.933.785 veículos, sendo que até abril de 2014, este número saltou para 4.257.286 (SANTA CATARINA, 2012).

Para a realização de aplicativos na área de visão computacional, é necessário algum tipo de biblioteca, a qual pode ser caracterizada como um componente chave baseado na visão de aplicações. O OpenCV, selecionado para o presente trabalho, é uma biblioteca multiplataforma, totalmente livre, tanto para uso acadêmico quanto para qualquer outra pessoa que tenha interesse em utilizá-la, mesmo que comercialmente.

Grande parte das pessoas tem um celular do tipo smartphone, os quais podem ser resumidos a aparelhos com grande poder de processamento e que integram vários recursos, geralmente com telas sensíveis a toque. Com celulares smartphones, as pessoas podem, além de realizar ligações, ter alta conectividade com a Internet, falar com seus amigos através de mensagens, utilizar aplicativos, entre outros.

O presente artigo aborda a elaboração de um protótipo composto por um software para detecção de vagas de estacionamento utilizando os conceitos de visão computacional e a biblioteca OpenCV, bem como um protótipo Android para visualização das vagas.

2. Visão Computacional

O conceito de Processamento Digital de Imagens, ou também chamado de PDI, refere-se ao conjunto de métodos e técnicas para interpretação de informações de imagens digitais que tem como objetivo facilitar a extração das mesmas (IBGE, 2001). Segundo Miranda (2009), a estrutura de um Sistema de Visão computacional é composta por pelo menos cinco etapas , a aquisição da imagem, pré processamento, extração das características, detecção das áreas de interesse e verificação dos dados extraídos.

As imagens digitais podem ser criadas através de diversos aparelhos eletrônicos, tais como máquinas fotográficas, câmeras de vídeo, scanners, microscópios, dentre vários outros. O termo “fotografia digital” é largamente utilizado hoje, porém este é apenas um tipo de imagem digital, que pode ser adquirido através de câmeras fotográficas, celulares e até mesmo tablets.

Em sua definição uma imagem digital é vista por um computador como um arranjo de elementos (dígitos) em forma de uma matriz, ou seja, uma função f(x,y).

O nome de cada elemento da matriz, ou seja, de cada coordenada de uma imagem digital, recebe o nome de pixel. Considerando isso, pode-se dizer que a quantidade de pixels de uma imagem está diretamente relacionada com a qualidade da mesma, ou seja, quanto maior for o número de pontos na imagem (pixel), maior a qualidade da imagem.

3. OPENCV

Open Source Computer Vision Library (OpenCV) é uma biblioteca de código aberto desenvolvida pela Intel com mais de 500 funções para visão computacional, disponível no site oficial http://opencv.org. Ela foi desenvolvida com o objetivo de tornar a visão computacional

mais acessível a usuários e também desenvolvedores nas área de interação entre humano e computador em tempo real, e também, a robótica (CUNHA, 2013).

Segundo Cunha (2013) podemos citar como exemplo de aplicações que utilizam o OpenCV: a união de imagens das ruas na detecção de intrusos em vídeos de monitoramento em Israel, equipamentos de monitoramento de minas na China, checagem de buracos nas rodovias da Turquia, detecção de afogamentos em piscinas na Europa, inspeção de rótulos de produtos nas fábricas, detecção da face humana, dentre vários outros.

O OpenCV foi desenvolvido com um objetivo principal que é fornecer uma infraestrutura com uma utilização simplificada para a visão computacional, com isso as pessoas podem construir aplicações sofisticadas e robustas com rapidez. Dentre as mais de 500 funções citadas anteriormente, está a inspeção de produtos, imagem médica, segurança, interface de usuário, calibração de câmera e várias outras (BRADSKI; KAEHLER, 2008, tradução nossa).

Sua construção foi feita em módulos integrados, sendo assim bastante versátil e eficiente para resolver problemas onde seja necessária a utilização de visão computacional. É possível cortar imagens, modificá-las, detectar formas, segmentação de imagens, detecção de objetos em movimento em vídeos, reconhecimento de objetos, dentre várias outras possibilidades (BRAHMBHATT, 2013).

Segundo Cunha (2013) a biblioteca em si possui um ponto negativo, que é a necessidade de que o desenvolvedor tenha um conhecimento robusto sobre programação em C ou C++ para que a aplicação tenha um melhor desempenho. A linguagem C++ é derivada do C e possui maiores vantagens como a redução de linhas de código como mostra na figura 12 que compara um simples programa para abrir e exibir uma imagem, melhor aproveitamento de código e uma menor preocupação com a memória utilizada. Porém o OpenCV pode ser utilizado também em outras linguagens, tais como Python e Java.

4. Mobilidade Urbana

O Anexo 1 do Código de Transito Brasileiro (CTB) (1997, p.55) diz que “estacionamento é a imobilização de veículos por tempo superior ao necessário para embarque e desembarque de passageiros”. Estacionamentos podem ser classificados de duas formas: nas vias públicas e foras das vias públicas (CET, 1979). Os estacionamentos ditos fora das vias públicas podem ser entendidos como lotes e as garagens, podendo estes serem públicos ou privados, gratuitos ou pagos. Já nas vias públicas, estes são os que estão ao longo do meio-fio, podendo ser livres ou pagos, e, oferecem um acesso mais fácil e também mais econômico na maioria dos casos.

Na atualidade foram criadas tecnologias para o problema da organização de estacionamentos, algumas delas podem ser chamadas de estacionamentos inteligentes. Existem hoje tecnologias capazes de mostrar ao motorista onde há vagas livres através de lâmpadas de diferentes cores, e ainda indicam o caminho para a vaga disponível mais próxima. Mas também há tecnologias muito mais avançadas mundo afora, onde existem estacionamentos em edificações que são completamente autônomos, basta que o motorista deixe o carro em uma plataforma e um sistema automatizado estaciona o veículo, e na hora de ir embora, o motorista só precisa fazer a requisição de seu automóvel que ele será trazido de volta (PUERARI et al, 2011).

No Brasil, o primeiro Shopping a adotar um sistema de estacionamento inteligente foi o Morumbi, em São Paulo, que utiliza sensores e Light Emitting Diode (LED) verde - os quais indicam que a vaga está livre - e vermelho - que significa que a vaga está ocupada. Tais dispositivos têm como principal objetivo o de acabar com o tempo perdido na procura por vagas livres no estacionamento do shopping. O sistema ainda ajuda a administração do estabelecimento no controle do fluxo de veículos e da ocupação das vagas.

Os painéis servem para indicar a localização de vagas disponíveis e podem ser vistos a uma longa distância, havendo um destes logo na entrada do estacionamento e também em outros setores do mesmo. Além disso, também fazem a contagem das vagas disponíveis por setor e indicam ao motorista onde elas estão localizadas.

Contudo, estacionamentos no geral não têm um espaço satisfatório, com uma quantidade de vagas razoável, não tendo fiscalização nem mesmo segurança. O foco do presente trabalho visa à identificação prévia de vagas livres onde motoristas possam estacionar seus veículos sem precisar ficar dando voltas pelo estacionamento procurando. Além disso, o foco está na diminuição de custos para a implantação e utilização do sistema. Outras vantagens seriam as diminuições do congestionamento do trânsito e do gasto de combustíveis.

5. Dispositivos Móveis

O número de dispositivos móveis hoje em dia está cada vez maior e está sendo usando nas mais diversas áreas. Telefones celulares, smartphones, máquinas digitais, notebooks, entre outros, estão cada vez mais se tornando parte do cotidiano das pessoas. Em alguns casos são até mesmo indispensáveis, visto que podem modificar as rotinas e até mesmo as decisões individuais, sendo estas tanto no trabalho quanto no lazer. A mobilidade está deixando de ser algo que facilita o cotidiano, e se tornando algo realmente necessário, pois é capaz de acessar dados e informações em qualquer lugar e a qualquer momento.

Os dispositivos móveis, em particular os smartphones e os tablets, estão com um grande destaque devido ao poder de conectividade, além de poderem ser utilizados tanto para uso pessoal como profissional. Com a grande explosão destes dispositivos no mercado, vários novos aplicativos exclusivos vêm sendo lançados, oferecendo aos usuários as facilidades de um mundo interligado por redes sem fio, possibilitando a qualquer hora e em qualquer lugar, o acesso a informações, bastando somente uma conexão de rede sem fio.

Segundo Costa e Duarte Filho (2013) grande parte dos sistemas operacionais para smartphones são "abertos" (não confundir com código-fonte aberto), ou seja, que é possível a qualquer pessoa que tenha conhecimentos desenvolver programas por meio de um Software Development Kit (SDK), ou no português, Kit de Desenvolvimento de Aplicativos ou Framework (conjunto de classes que ajudam no desenvolvimento de um subsistema ou da aplicação em si) que possibilita ser executado por estes telefones.

Costa e Duarte Filho (2013) afirmam também que um dos grandes problemas atuais dos sistemas operacionais de dispositivos móveis é a falta de padronização entre eles, resultando em problemas tanto para empresas como para usuários. Com isso muitas organizações escolhem sistemas desconhecendo as principais funcionalidades e a arquitetura do software, que por fim após o desenvolvimento do aplicativo, concluem que não era o sistema ideal para seus negócios.

O Android possui uma grande vantagem, uma vez que sua plataforma também é livre e de código aberto. Em outras palavras, sua licença permite que cada fabricante ou empresa realize alterações no código-fonte de seus aparelhos móveis, e não são obrigadas a compartilhar estas mudanças com ninguém. Ele também é “free”, ou seja, não é necessário que os fabricantes paguem para poder utilizá-lo, assim podem usufruir de todos os recursos oferecidos por ele, além também de fazer suas próprias alterações (COSTA; DUARTE FILHO, 2013).

6. Software para Detecção de Vagas e Protótipo Android

A metodologia para o desenvolvimento do protótipo foi constituída a partir de quatro etapas, onde a primeira delas foi o levantamento de requisitos com base em outros trabalhos com objetivos semelhantes ao apresentado e no estudo realizado neste, visando encontrar uma

maneira para a identificação de vagas de estacionamento. A segunda etapa se constitui do entendimento da biblioteca OpenCV e do conceito de visão computacional para assim ser elaborado o protótipo do software para detecção das vagas. A terceira etapa se caracteriza pelo desenvolvimento do protótipo destinado a dispositivos com sistema operacional Android. E, por fim, na quarta etapa foi feita a integração entre software e aplicação, tornando possível a visualização das vagas livres e ocupadas.

6.1. Levantamento de Requisitos

A etapa de levantamento de requisitos compreendeu o estudo sobre o conceito de visão computacional, a fim de compreender a teoria para, então, aplicá-la no desenvolvimento do software para a detecção de vagas de estacionamento. Além disso, também foi realizada uma pesquisa sobre projeto desenvolvidos a partir da mesma tecnologia que seria utilizada para o software de detecção de vagas, no caso a biblioteca de tratamento de imagens OpenCV. A partir dessa pesquisa foi possível o levantamento dos requisitos iniciais para o desenvolvimento do protótipo.

Ainda na etapa do levantamento de requisitos, realizou-se uma pesquisa acerca da biblioteca de processamento de imagens OpenCV, que é uma das bibliotecas mais utilizadas nessa área. Além disso, ela é Open Source, ou seja, possui seu código disponível para que qualquer pessoa possa utilizá-la, fornecendo também uma ampla documentação onde foi possível estudar e compreendê-la.

Para que fosse possível o desenvolvimento do protótipo, foi adquirida uma câmera webcam com interface USB. Este dispositivo foi utilizado para a captura do vídeo, pois o intuito do projeto é a realização de tratamento de imagens em tempo real para a detecção das vagas de estacionamento, tornando possível que o usuário possa visualizar a localização de vagas livres onde possa estacionar seu carro, sem ter que ficar dando voltas dentro de um determinado estacionamento em função da procura.

A partir da pesquisa realizada, a maneira encontrada para a identificação das vagas de estacionamento se fez através de marcadores e identificação de cores. Foi criado um ambiente de simulação onde no centro de cada vaga há um círculo vermelho, sendo que o objetivo é a detecção desse círculo a partir de sua cor, ou seja, se a câmera detectá-la significa que a vaga está livre. Caso contrário, será considerado que há um carro naquela vaga, pois este estaria por cima do marcador, impossibilitando a câmera de visualiza-lo.

6.2. Software de Detecção de Vagas de Estacionamento

Pensando prioritariamente nas vagas onde carros seriam estacionados, a maneira encontrada para a identificação das vagas se fez através de marcadores no centro de cada uma delas. Para o desenvolvimento do protótipo em pequena escala, foram utilizados marcadores vermelhos para atuarem como os marcadores, assim o software teria a função de localizá-los. Caso sim significaria que a vaga estaria livre, caso contrário um carro estaria em cima do marcador, ocupando a vaga e impedindo que o software detecte o círculo vermelho.

Dentre os métodos utilizados, o principal deles foi o Core.inRange , o qual é responsável por detectar a escala de cor informada, no caso o vermelho. Esta parte foi onde surgiu outro grande desafio, pois a estaria detectando pequenos pontos indesejados na imagem. A fim de resolver este problema, foram aplicados então outros métodos e filtros para remover os maiores detalhes da imagem, tais como o Imgproc.erode e o Imgproc.dilate. Estes dois utilizados em sequência atuam como um filtro onde é possível remover pequenos detalhes da imagem, possibilitando a continuação do desenvolvimento do trabalho, aplicando então a função findCountors para identificação dos contornos encontrados pelo método Core.inRange.

Ambas as funções para erodir e dilatar a imagem tomam como parâmetros uma matriz de imagem origem e uma de destino. O terceiro parâmetro é o kernel, o padrão deste é nulo, mas o utilizado neste trabalho foi o retangular CV_SHAPE_RECT. E o quarto parâmetro é o número de iterações, que caso não seja definida, o padrão é um. Assim torna-se possível a remoção de ruídos indesejados na imagem, pois as regiões que contêm um conteúdo maior não são afetadas.

A próxima etapa do projeto foi encontrar uma maneira de saber a localização das vagas, ou seja, saber qual a vaga número um, qual a vaga número dois, e assim por diante. Esta etapa fez-se necessária porque até então só estavam sendo detectados os marcadores em vermelho, não sendo possível ainda saber a posição das vagas livres e das que estavam ocupadas. Foi necessário a criação de sub-matrizes a partir da matriz principal que é formada pela imagem capturada pela webcam, onde cada uma delas representaria uma vaga.

Conforme mostra a figura 1, cada vaga corresponde a uma sub-matriz, e somente nelas é realizada a verificação para saber se a vaga está livre ou ocupada. Como não há nenhum carro e o software está detectando todos os marcadores, a sub-matriz está recebendo uma marcação verde, caso contrário receberia uma marcação vermelha, que é a indicação de a vaga está ocupada.

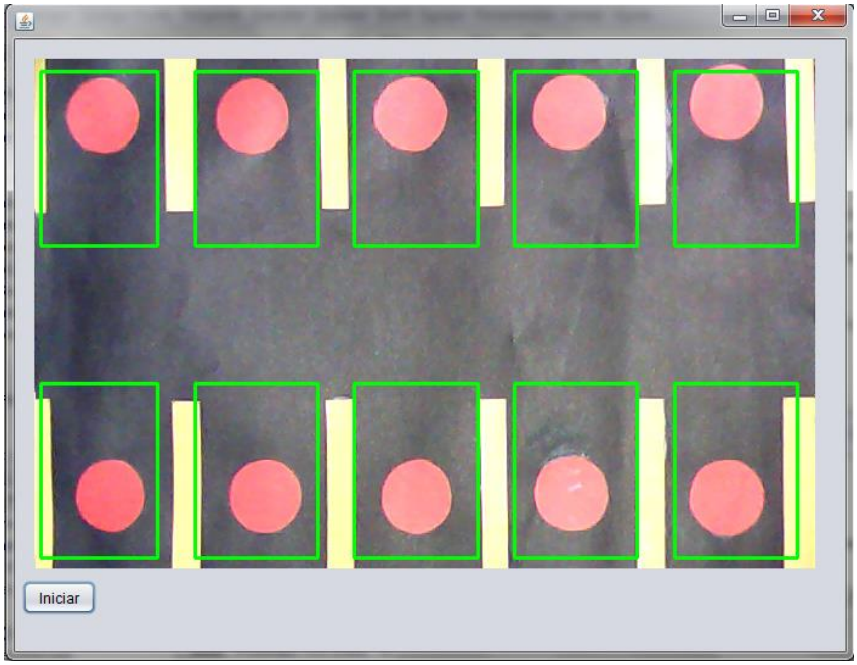


Figura 1. Sub-matrizes das vagas.

6.3. Protótipo Android

Por meio das pesquisas realizadas sobre dispositivos móveis, foi visto que smartphones são muito utilizados no dia-a-dia das pessoas, e que grande parte deles possui o sistema operacional Android, esse foi o principal motivo pela escolha dele para o desenvolvimento do protótipo deste trabalho.

Primeiramente para o início do desenvolvimento do protótipo, assim como para o software, foi necessário a escolha de uma plataforma para a elaboração do mesmo. Duas delas são muito conhecidas e utilizadas por desenvolvedores, o Eclispe, e o Android Studio, ambos são de licença gratuita.

O Eclipse é uma plataforma para desenvolvimento em Java, mas que suporta várias outras linguagens a partir de plugins. Já o Android Studio é especializado para a criação de aplicativos Android, com várias ferramentas específicas e úteis, isto fez com que fosse escolhido para o desenvolvimento deste trabalho.

Com a plataforma de desenvolvimento baixada e instalada, o próximo passo foi pensar em uma maneira em que o usuário consiga visualizar as vagas e identifica-las com clareza, pensando nisso foi criado uma imagem, sendo semelhante a do protótipo do estacionamento em pequena escala,

A figura 2 mostra o layout do protótipo, que possui uma imagem semelhante ao estacionamento em escala reduzida utilizado para este trabalho, tornando simples a visualização das vagas, pois se pode identificar a localização das mesmas e seu status, ou seja, se estão livres ou ocupadas.

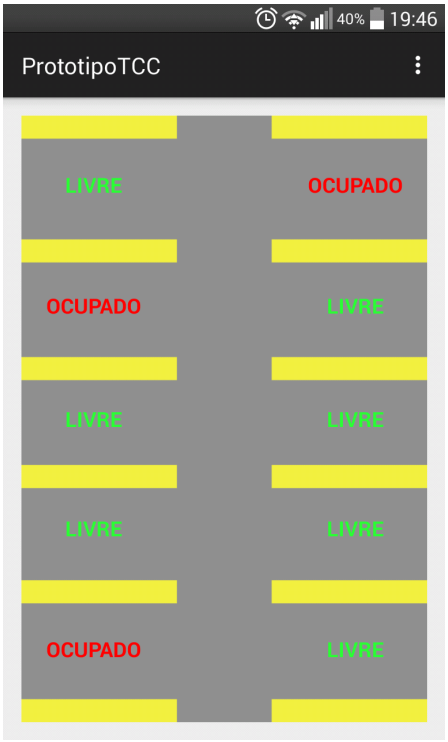


Figura 2. Layout do Protótipo Android.

6.4. Integração entre Software e Protótipo Android

Existem algumas maneiras para se realizar a comunicação entre softwares e aplicativos, como utilização de banco de dados ou Web Services, isso depende muito da situação e o objetivo de ambos.

A utilização de Socket para um cenário real não é viável, pois é necessário que tanto a máquina que esteja com o software rodando, quanto o aplicativo, esteja conectada na mesma rede. Este tipo de comunicação utiliza endereço de IP e um número de porta, com base nesse endereço é feito a entrega, ou troca de pacotes de dados.

De forma geral, uma aplicação faz o papel de servidor abrindo um Socket na rede, ou seja, um número de porta, e outra aplicação faz o papel de cliente, que acessa esta porta juntamente com o endereço de IP do servidor. Com isto, é possível o envio de pacotes de dados, tanto do servidor para o cliente, quanto do cliente para o servidor sem a necessidade de um local para armazenamento de dados.

Neste caso, o software para detecção das vagas de estacionamento fará o papel de servidor, pois ele que possui as informações das vagas e fará o envio delas para o aplicativo, onde estas serão abstraídas e transformadas de forma que possam ser visualizadas.

7. Conclusão

Os conceitos de processamento de imagens e visão computacional podem ser utilizados de diversas formas e em diversas áreas, tais como identificação de objetos, realidade virtual, reconhecimento de movimentos, reconhecimento facial, dentre outras. As possibilidades de desenvolvimento de aplicações envolvendo esta tecnologia são infinitas quando em conjunto com a criatividade.

A biblioteca OpenCV utilizada para auxiliar no processamento de imagens, apesar de sua grande extensão, foi possível abstrair um pouco dos recursos que ela oferece, possibilitando o desenvolvimento do protótipo proposto.

Os trabalhos existentes na área de visão computacional utilizando a biblioteca OpenCV são inúmeros, alguns destes envolvem o tema mobilidade urbana, tais como controle de fluxo de tráfego, identificação de placas de veículos, entre outros. É uma biblioteca largamente utilizada devido as funcionalidades que ela oferece e também por ser de licença livre.

Na atualidade as tecnologias utilizadas para o mesmo propósito possuem um custo altíssimo de implantação, envolvendo circuitos com sensores em todo o estacionamento, máquinas pesadas que posicionam automaticamente os automóveis nas vagas, dentre outras.

O estudo realizado possibilitou o desenvolvimento de um protótipo para detecção de vagas de estacionamento através de câmeras e tratamento de imagens, identificando quais estão livres ou ocupadas, e também um protótipo destinado a Android, onde os usuários possam visualizar o estacionamento e as vagas livres, evitando gastos com combustível em função da procura e também um possível congestionamento.

Referências

ALHAK, Said Hikmat Abd. Estacionameto Inteligente**.** 2011. 76 f. TCC(Graduação) - Curso de Engenharia da Computação, Centro Universitário de Brasília – Uniceub Fatecs – Faculdade de Tecnologia e Ciências Sociais Aplicadas Curso de Engenharia de Computação, Brasília, 2011.

BRADSKI, Gary; KAEHLER, Adrian. Learning OpenCv**.** Sebastopol: O'reilly Media, 2008. 571 p.

BRAHMBHATT, Samarth. Pratical OpenCv**.** New York: Technology In Action, 2013. 229 p.

COSTA, Norben P. O.; DUARTE FILHO, Nemésio F.. ANÁLISE E AVALIAÇÃO FUNCIONAL DE SISTEMAS OPERACIONAIS MÓVEIS: VANTAGENS E DESVANTAGENS.**Revista de Sistemas e Computação,** Salvador, v. 3, n. 1, p.66-67, jun. 2013.

CTB - Código de Trânsito Brasileiro. Lei Federal n° 9503/97.

CUNHA, André Luiz Barbosa Nunes da. Sistema Automático para Obtenção de Parâmetros do Tráfego Veicular a partir de Imagens de Vídeo usando OpenCv**.** 2013. 128 f. Curso de Escola de Engenharia de São Carlos, Universidade de São Paulo, São Paulo, 2013.

IBGE. Introdução ao processamento digital de imagens. Rio de Janeiro: IBGE, 2001. 92p.

MIRANDA, Jeferson José. MOVE-IN: Uma API para interatividade com a webcam. Centro de Ciências Tecnológicas / Universidade do Estado de Santa Catarina, 2008 Disponível

em: http://www2.joinville.udesc.br/~larva/portal/uploads/TCC-II. Jeferson. J.M.pdf Acesso em 15 out. 2014.

PUERARI, André Farias et al. Estacionamentos Inteligentes como Solução ao Congestionamento de Grandes Centros Urbanos**.** Blumenau: Cobenge, 2011.

SANTA CATARINA. DETRAN-SC. Frota de Veículos no Estado**.** Disponível em: <http://www.detran.sc.gov.br/index.php/estatistica/veiculos>. Acesso em: 11 ma. 2014.