UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA
ESPECIALIZAÇÃO EM TECNOLOGIA JAVA E DISPOSITIVOS MÓVEIS

FABIO AUGUSTO CABRAL
VICTOR HUGO PEREIRA MACHADO

IDENTIFICADOR DE PLACA VEICULAR:
alternativa para segurança em escolas

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2014

FABIO AUGUSTO CABRAL
VICTOR HUGO PEREIRA MACHADO

IDENTIFICADOR DE PLACA VEICULAR:
alternativa para segurança em escolas

Monografia de especialização apresentada ao Programa de Pós-Graduação em Tecnologia da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de especialista.

Orientador (a): Prof.ª Dr.ª Marília A. Amaral.

CURITIBA
2014

AGRADECIMENTOS

Agradeço a Prof.ª Dr.ª Marília A. Amaral pela orientação desta pesquisa e pelos momentos de aprendizado. Agradeço aos pesquisadores e professores da banca examinadora pela atenção e contribuição dedicadas a este estudo. Gostaríamos de deixar registrado também, o nosso reconhecimento à família, pois acreditamos que sem o apoio deles seria muito difícil vencer esse desafio.

A mente que se abre a uma nova ideia
jamais volta ao seu tamanho original.

(Albert Einstein)

RESUMO

Este trabalho foi desenvolvido com o propósito de reforçar a vigilância e aumentar a segurança dos alunos nas escolas, que é uma preocupação constante para gestores de instituições particulares e públicas. Nos últimos anos, este assunto tem sido tema de diversos fóruns e discussões de especialistas em segurança e também de pais e responsáveis por alunos, tornando-se um item bastante relevante no momento da escolha pela escola que irão matricular seus filhos. Neste trabalho é apresentado o desenvolvimento de um aplicativo que tem o objetivo de auxiliar as instituições de ensino básico no controle de acesso. Foi escolhido o dispositivo móvel para utilizar o aplicativo devido a sua mobilidade e capacidade de capturar a imagem, processar os dados e identificar em um único aparelho. O aplicativo tem como finalidade manter informadas as portarias das escolas sobre as permissões e restrições de entrada de veículos no interior das instituições, aumentando a segurança para os pais e alunos.

Palavras-chaves: Dispositivos Móveis, Reconhecimento de Imagens, Segurança.

ABSTRACT

This work was developed with the purpose of strengthening surveillance and increase the safety of students in schools, which is a constant concern for managers of private and public institutions. In recent years, this subject has been the topic of many forums and discussions of security experts as well as parents and guardians of students, making it a very important item when choosing the school that will enroll their children. This paper presents the development of an application that aims to help institutions of basic education in access control. The mobile device was chosen to use the application due to their mobility and ability to capture the image, process the data and identify in a single device. The application aims to keep the ordinances of schools informed about the permissions and restrictions on entry of vehicles within the institutions, increasing safety for parents and students.

Keyword: Mobile Devices, Image Recognition, Security.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API - Interface de Programação de Aplicações *(Application Programming Interface)*
APP – Aplicativo
CDMA - Acesso Múltiplo por Divisão de Código (*Code Division Multiple Access*)
DBA - Administrador de Banco de Dados (*Database administrator*)
GPS - Sistema de posicionamento global (G*lobal Positioning System)*
GSM - Sistema Global para Comunicações Móveis (*Global System for Mobile Communications)*
HTML - Linguagem de Marcação de Hipertexto (*HyperText Markup Language)*
JDK - Kit de Desenvolvimento Java (Java *Development* Kit)
JSON - *JavaScript Object Notation*
NDK - *Native Development Kit*
OCR - Reconhecimento Óptico de Caracteres *(Optical Character Recognition)*
OHA - *Open Handset Alliance*
PDA - Assistente pessoal digital (*Personal digital assistants*)
SDK - Kit de Desenvolvimento de Software (*Software Development Kit*)
SVA - Sistema de Visão Artificial
UML - *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

De acordo com Ibope Inteligência (GRSA, 2010), na hora de decidir onde os filhos vão estudar, o que vem em primeiro lugar é a segurança oferecida pela instituição. A qualidade da educação vem em segundo plano e, atrás dela, as amizades das crianças.

Nas escolas durante o horário de entrada e saída dos alunos, existe um fluxo grande de pessoas que vão buscar seus filhos e, essa grande movimentação exige atenção dos porteiros para garantir a segurança dos alunos. Esta entrada e saída de pessoas na escola possibilita uma falha de segurança fazendo com que os alunos fiquem expostos ao contato de pessoas não autorizadas.

No estudo de caso proposto, para controlar o fluxo de entrada dos veículos o porteiro identifica o motorista consultando uma agenda ou fazendo uma breve entrevista, porém mesmo tomando esses cuidados não garante que pessoas não autorizadas acabem entrando na escola e também esse tipo manual de controle acaba formando grandes filas de carros na frente das escolas.

Diante desse cenário, este trabalho pretende propor uma solução, utilizando artefatos móveis, para auxiliar na garantia de segurança em estabelecimentos de ensino.

1.1 OBJETIVOS

Esta seção apresenta o objetivo geral da pesquisa, bem como seus objetivos específicos.

1.1.1 Objetivos Gerais

Desenvolver um aplicativo para dispositivos móveis para verificar se é permitida a entrada de um veículo em uma escola, fazendo a leitura da placa do veículo.

1.1.2 Objetivos Específicos

Os objetivos específicos do trabalho são:

- Desenvolver a entrada dos dados das placas via teclado do dispositivo móvel.
- Desenvolver um módulo para captura de imagem através da câmera do dispositivo.
- Analisar e comparar mecanismos para processamento de imagem e conversão em texto.
- Definir a melhor forma de armazenamento das imagens capturadas no dispositivo.
- Desenvolver um módulo para sincronizar informação do dispositivo com sistema externo.
- Auxiliar o controle de acesso nas escolas, registrar o acesso de veículos, fornecer informações ao porteiro sobre restrições de acesso e também dados básicos de alunos, pais ou responsáveis.

1.2 JUSTIFICATIVA

O desenvolvimento do aplicativo para controle de acesso em colégios tem o objetivo de aumentar a segurança para os alunos, pais ou responsáveis. Com uma interface objetiva e de fácil operação pretende-se diminuir as grandes fila de carros nos portões dos colégios e diminuir o tempo de espera dos alunos.

1.3 METODOLOGIA EMPREGADA

Para o desenvolvimento do aplicativo foi feito um levantamento junto com a equipe de segurança da escola para verificar a melhor forma de utilização do aplicativo, as informações que o aplicativo deve apresentar como dados cadastrais de alunos foram definidos pela gestora e assessora do colégio. Para o módulo de sincronização do dispositivo foi utilizado um *WebService* que foi configurado pela equipe de TI do colégio. O aplicativo foi desenvolvido para um *smartphome* com plataforma *Android,* devendo possuir câmera e acesso a rede de dados.

Para identificação dos veículos foi utilizado uma biblioteca OCR, *Optical Character Recognition*, que faz a interpretação da imagem da placa do veículo capturada pela câmera do dispositivo. O resultado do processamento do OCR retorna um texto, e este é pesquisado na base de dados do dispositivo para localizar o registro relacionado ao veículo.

2 REFERENCIAL TEÓRICO

Neste capitulo são apresentados algumas características de sistema operacional *Android* e reconhecimento de imagem.

Dispositivos móveis são artefatos que podem facilmente ser movidos fisicamente ou cujas capacidades podem ser utilizadas enquanto eles estão sendo movidos (Reza B'Far, 2005). Como estes sistemas preveem tal mobilidade, eles normalmente oferecem recursos e características que não são encontradas em sistemas comuns, como por exemplo, monitoramento do nível de energia e prevenção de perda de dados em caso de pane de energia, armazenamento de dados local e/ou remoto, através de conexão com ou sem fio e sincronização de dados com outros sistemas. (ROBINSON, M., KALAKOTA, R., 2002).

Atualmente, são considerados dispositivos móveis *smartphones*, *palmtops*, celulares, *tablet* e similares e uma de suas vantagens mais triviais seria a possibilidade de acessar dados em qualquer lugar e qualquer momento. Como os *smartphones* são o foco deste trabalho, a seção 2.1 apresenta algumas de suas características.

O processo de transcrição no OCR retorna o texto da placa do veículo, a pesquisa descrita em Mello, C. A. B (2002) mostrou que os OCR's são sensíveis às alterações mínimas na imagem. Desta forma qualquer mudança gera um arquivo com texto distinto. Os OCR's também são sensíveis à tonalidade do papel, textura, sensibilidade à rotação do documento, sombra e ruído, os quais são capturados e passados à imagem digitalizada.

O Processamento de Imagens digitalizadas divide-se em três etapas, pré-processamento, realce e classificação. Pré-processamento refere-se ao processamento inicial de dados brutos para calibração radiométrica da imagem, correção de distorções geométricas e remoção de ruído. Realce visa melhorar a qualidade da imagem, permitindo uma melhor discriminação dos objetos presentes na imagem. Na classificação são atribuídas classes aos objetos presentes na imagem.

O elemento principal é a imagem ou um conjunto delas, captadas através de uma câmera, scanner, sensores, ou qualquer outro meio. No processamento de imagens tanto a imagem original quanto a imagem resultado apresenta-se sob uma representação visual. As técnicas de transformação da imagem visam melhorar as

características estruturais da imagem (contraste, foco, diminuir ruídos e/ou distorções) fornecendo subsídio para a sua interpretação e para outros processamentos, onde algoritmos realizam operação com o objetivo de segmentá-las. Esta é uma área que dá suporte a diversas outras como Visão Computacional e Reconhecimento de Padrões.

De acordo com Marques Filho e Vieira Neto (1999), os elementos de um sistema de processamento de imagens de uso genérico são mostrados na Figura 1. Este diagrama permite representar desde sistemas de baixo custo até sofisticadas estações de trabalho utilizadas em aplicações que envolvem intenso uso de imagens.

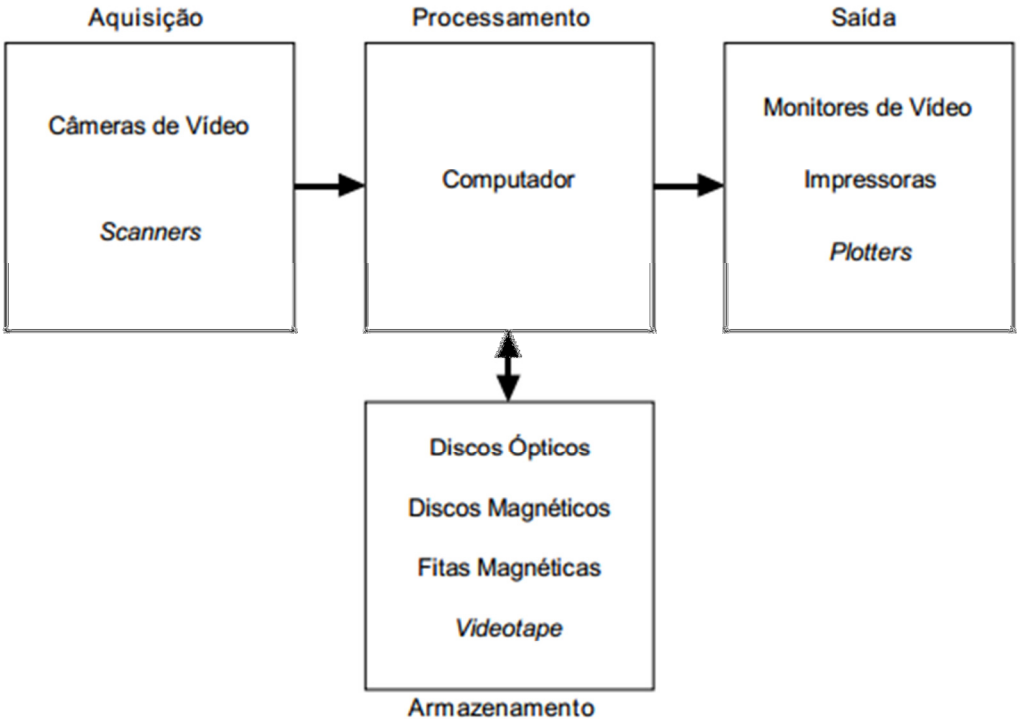

Figura 1 - Elementos de um sistema de processamento de imagens.
Fonte: (MARQUES FILHO, & VIEIRA NETO, 1999).

Conforme apresentado o diagrama da Figura 1, será utilizado o mesmo conceito de etapas nesse trabalho, mas todas executadas pelo dispositivo móvel. As imagens devem ser captadas pela câmera, processada, reconhecida pelo sistema por médio de comparação e posteriormente exibida o resultado se a imagem foi devidamente identificada.

2.1 SMARTPHONE

Os *smartphones* são a combinação de duas classes de dispositivos: os celulares e os assistentes pessoais (como os *Palm*s e os PDAs) (ROBINSON, M., KALAKOTA, R., 2002).

A primeira geração de celulares utilizava um sistema muito simples, com sinal analógico e serviam apenas como um sistema de telefonia móvel, baseado na transmissão por rádio (MORIMOTO, 2009). Com a tecnologia analógica o sinal podia ser facilmente copiado e reproduzido possibilitando a clonagem de aparelhos, esse problema acelerou a utilização do CDMA (Acesso Múltiplo por Divisão de Código) e GSM (Sistema Global para Comunicações Móveis) e com a tecnologia digital, utilizada atualmente, possibilitou a transmissão de dados baseada em pacotes. Com a evolução da tecnologia, os celulares passaram a incorporar as funções de cada vez mais dispositivos, tornando-se progressivamente mais importantes até se tornarem *smartphones*.

2.2 ANDROID

Android é um sistema operacional *open source* para dispositivos móveis, baseado em Linux. Foi criado pela *Android* Inc. em Outubro de 2003 na cidade de Palo Alto, Califórnia, Estados Unidos, e tem como seus fundadores: Andy Rubin, Rich Miner, Nick Sears e Chris White (*ANDROID*, 2014).

Em Agosto de 2005, o Google comprou a *Android* Inc., dois anos depois a OHA foi fundada com o intuito de evoluir a plataforma *Android* e investir na inovação das tecnologias de telefonia móvel (*ANDROID*, 2014). Em 2007, mesmo ano que surgiu a OHA, foi lançada a primeira versão beta da plataforma *Android*, em setembro de 2008, a operadora americana T-Mobile anuncia o T-Mobile G1, o primeiro *smartphone* baseado no *Android*. Neste mesmo ano o Google publica o código-fonte do *Android* 1.0 sob a licença Apache.

A seguir é apresentado o histórico das versões do Sistema *Android* e suas melhorias (*ANDROID*, 2014):

Android 1.5 - *Cupcake* (30 de abril de 2009)

- Melhoria da câmera;
- Aumento da velocidade de localização do GPS;
- Teclado virtual;
- Carregamento automático dos vídeos no YouTube e Picasa.

Android 1.6 – *Donut* (15 de setembro de 2009)

- Box de busca veloz e busca por voz;
- Indicador do uso da bateria;
- Reagrupamento da câmera e da galeria, além de novos modos de foto;
- Função text-to-speech multilíngue.

Android 2.0 – *Eclair* (26 de outubro de 2009)

- Contas múltiplas e-mail e sincronização dos contatos;
- Suporte para *Bluetooth* 2.1;
- Nova interface de usuário do browser e suporte para HTML5;
- Novas funções para o calendário.

Android 2.2 – *Froyo* (20 de maio de 2010)

- Suporte para criar *hotspot* (compartilhar a conexão via WiFi);
- Adobe Flash 10.1;
- Teclado multilíngue;
- Integração de um “widget guia” que ajuda a conhecer as funções do Android.

Android 2.3 – *Gingerbread* (6 de dezembro de 2010)

- Interface reformulada para maior simplicidade e velocidade;
- Novo teclado para digitação rápida;
- Seleção de texto e funções copia/cola;
- Integração de chamadas pela internet.

Android 3.0 – *Honeycomb* (23 de fevereiro de 2011)

- Versão para *tablet*, interface otimizada para telas maiores;
- Melhoria do *multitasking*, do gerenciamento das notificações, da personalização da *homescreen* e dos *widgets*;

- Acrescentado o *tethering* através do *Bluetooth*;
- Suporte integrado para transferir facilmente arquivos multimídia para o PC.

Android 4.0 – *Ice Cream Sandwich* (19 de outubro de 2011)

- Nova fonte (*Roboto*);
- Possibilidade de desbloqueio com o sorriso;
- Acrescentada funções como gerenciamento dos cartões, dos favoritos e da captura de tela;
- Acrescentado o *swipe* para esconder notificações, fechar páginas da web, etc.;
- Suporte a Wi-Fi Direct, Bluetooth HDP e Android Beam.

Android 4.1 – *Jelly Bean* (27 de junho de 2012)

- Mais veloz mais fluido e mais reativo aos inputs;
- *Widgets* redimensionáveis;
- Google Now, ditado vocal offline;
- Melhorado o Android *Beam*;
- Melhorias nas atualizações de *apps*.

Android 4.4 – *KitKat* (31 de outubro de 2013)

- Suporte para Bluetooth MAP;
- Novo framework para as transições na interface de usuário;
- Suporte para a impressão sem fio;
- Otimização da memória e da *touchscreen* para um *multitasking* mais veloz.

Android 5.0 – *Lollipop* (sem previsão até o momento)

O gráfico da Figura 2 fornece dados sobre o número relativo de dispositivos que executam uma determinada versão da plataforma Android (ANDROID, 2014).

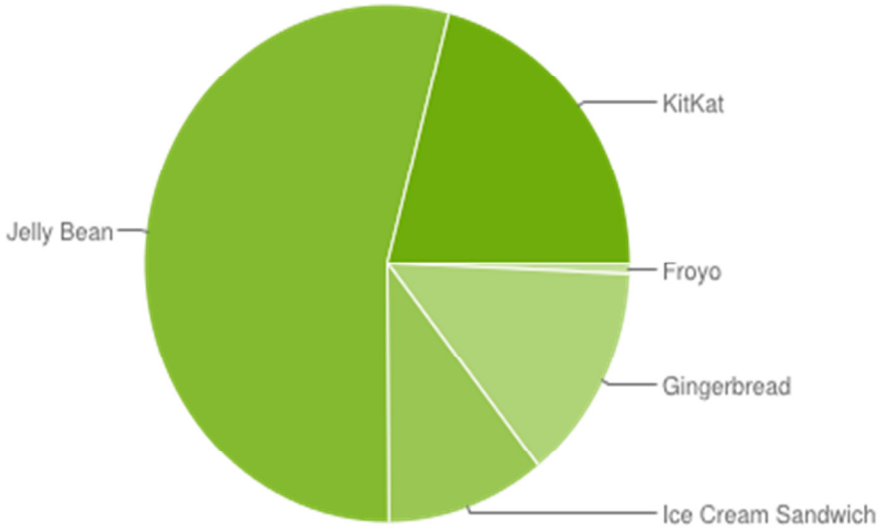

Figura 2 – Utilização das versões do Android
Fonte: (ANDROID, 2014)

Os dados apresentados na Quadro 1 foram coletados dentro do período de 7 dias encerrado em 12 de agosto de 2014. Para a coleta dos dados foi utilizado o novo aplicativo *Google Play Store* que suporta as versões do Android a partir da 2.2, dispositivos que executam versões anteriores não estão incluídos. Todas as versões com distribuição inferior a 0,1% não são apresentadas (ANDROID, 2014).

Quadro 1 - Utilização das versões do Android

Versão	Nome	API	Distribuição
2.2	Froyo	8	0,7%
2.3.3	Gingerbread	10	13,6%
2.3.7			
4.0.3	Ice Cream Sandwich	15	10,6%
4.0.4			
4.1.x	Jelly Bean	16	26,5%
4.2.x		17	19,8%
4.3		18	7,9%
4.4	KitKat	19	20,9%

Fonte: (ANDROID, 2014)

Como foi mencionado no item 1.3, utilizou-se o sistema *Android* para desenvolvimento e uma biblioteca de OCR para reconhecimento do texto na imagem capturado pelo dispositivo. Para a utilização da biblioteca de OCR existe a dependência da versão 2.2 ou superior da plataforma *Android*.

2.2.1 Visão Geral da Arquitetura

O Android é uma plataforma composta pelo sistema operacional baseado em Linux e um conjunto de bibliotecas e aplicativos que o acompanham (LECHETA, 2010). Na Figura 3 é possível visualizar as principais camadas da plataforma *Android*.

Figura 3 - Arquitetura da plataforma Android.
Fonte: MEIER, 2009

A base do Android é uma versão modificada do Linux kernel 2.6, que provê vários serviços essenciais, como segurança, rede e gerenciamento de memória e processos, além de uma camada de abstração de hardware para as outras camadas

de softwares (Android *Developers*, 2012). A seguir, serão descritas cada uma das camadas da plataforma *Android*.

Aplicações *(Applications)*: camada que executa os aplicativos do *Android*, tanto as nativas como Navegador, Contatos, Calculadora etc., quanto aplicativo de terceiros como o protótipo aqui proposto.

Quadro de Aplicações *(Application Framework)*: camada que contêm os serviços utilizados pelos aplicativos Android, como por exemplo, o *Activity Manager* (ciclo de vida da aplicação), *Location Manager* (localização por GPS ou Rede), *Content Providers* (provedores de conteúdos, ex: Contatos).

Bibliotecas *(Libraries)*: camada onde estão algumas API's desenvolvidas em C/C++ e que oferecem recursos como renderização 3D (OpenGL ES), banco de dados (SQLite), *WebKit* (renderização de conteúdo Web).

Android Tempo/Execução (*Runtime)*: camada que possui um conjunto de core *libraries* que provém à maioria das funcionalidades disponíveis nas core *libraries* do Java. Também contem a Dalvik VM, que provê todo o ambiente necessário para a execução de uma aplicação *Android*.

Núcleo Linux (*Kernel)*: camada responsável por armazenar os serviços de baixo nível do Sistema Operacional *Android*. Contêm serviços essenciais como segurança, gerenciamento de memória, gerenciamento de processos e drivers de dispositivos.

2.2.2 SQLite

O Android traz embutido o SQLite, que é uma biblioteca que implementa um mecanismo de banco de dados, sem servidor e com código de domínio público (LECHETA, 2010). O SQLite dá suporte à maioria das funções da SQL92, a terceira revisão do padrão SQL. Armazena as informações em um arquivo único e o tamanho do seu código é pequeno; por isso, pode ser utilizado em uma ampla gama de aparelhos, que disponham de recursos limitados (SQLITE, 2014).

Com a utilização desse banco de dados, o armazenamento, consulta e manutenção dos dados se torna mais rápido e prático em relação aos celulares de gerações tecnológicas anteriores que proviam o armazenamento em arquivos simples. O limite desse banco de dados será dado pelo tamanho da memória do

dispositivo, podendo chegar a *terabytes* de armazenamento, e oferece diversos recursos, dentre eles os principais são (SQLITE, 2014):

- Zero *Configuration*: projetado para que não necessite de um DBA, o próprio programador pode configurar e administrar.
- *Atomic Transactions*: transações consistentes, isoladas e duráveis mesmo depois de falhas de sistema e falhas de energia.
- *Self-contained*: sem dependências externas.
- Cross-plataforma: Unix (Linux, Mac OS-X, Android, iOS) e Windows (Win32, *WinCE*, *WinRT*) e também funciona em plataformas embarcadas como QNX, *VxWorks*, *Symbian*, *Palm* OS, Windows CE.
- *Small code footprint*: menos de 500KiB totalmente configurado ou muito menos com os opcionais omitidos.
- *Limits*: Suporta bancos de dados de *terabytes* e *string* e *blobs* de gigabytes.

2.2.3 Json

O Json é uma formatação leve de troca de dados e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C, incluindo C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades fazem com que JSON seja um formato ideal de troca de dados (JSON, 2014).

2.3 RECONHECIMENTO DE IMAGEM

Segundo Marques Filho e Vieira Neto (1999) de 1964 aos dias atuais, as aplicações de processamento de imagens permeiam quase todos os ramos da atividade humana. Por exemplo, na Medicina o uso de imagens no diagnóstico médico. Em Biologia, a capacidade de processar automaticamente imagens obtidas de microscópios.

A evolução da tecnologia de computação digital, bem como o desenvolvimento de novos algoritmos para lidar com sinais bidimensionais está

permitindo uma gama de aplicações para reconhecimento de imagens cada vez maior, principalmente na área de Medicina (MARQUES FILHO, & VIEIRA NETO, 1999).

Conforme o autor supracitado, a tecnologia de processamento digital de imagens vem ampliando seus domínios, que incluem as mais diversas áreas, como por exemplo: análise de recursos naturais e meteorologia por meio de imagens de satélites; transmissão digital de sinais de televisão ou fax; análise de imagens biomédicas, incluindo a contagem automática de células e exame de cromossomos; análise de imagens metalógrafas e de fibras vegetais; obtenção de imagens médicas por ultrassom, radiação nuclear ou técnicas de tomografia computadorizada; aplicações em automação industrial envolvendo o uso de sensores visuais em robôs, etc.

Em Medicina, o uso de imagens no diagnóstico médico facilita a interpretação de imagens produzidas por equipamentos antigos bem como o desenvolvimento de novos equipamentos. Em Biologia, o processamento de imagens de microscópios, facilita as tarefas laboratoriais (MARQUES FILHO, & VIEIRA NETO, 1999).

De acordo com Marques Filho e Vieira Neto (1999) nas áreas de Geografia, Sensoriamento Remoto, Geoprocessamento e Meteorologia, o processamento e a interpretação de imagens capturadas por satélites auxiliam os seus trabalhos. Além do uso de processamento de imagens nas áreas de Astronomia, Publicidade, Direito, Arqueologia e Segurança.

Para a transcrição de imagem para texto, utilizam-se imagens com qualidade suficiente para extrair caractere a caractere com a maior fidelidade possível. Qualquer ruído ou rotação na captura da imagem pode prejudicar o processamento da imagem e acarretar em erros para todo o processo de reconhecimento. (MARQUES FILHO, & VIEIRA NETO, 1999).

A Figura 4 mostra esquematicamente um diagrama de blocos de um Sistema de Visão Artificial (SVA).

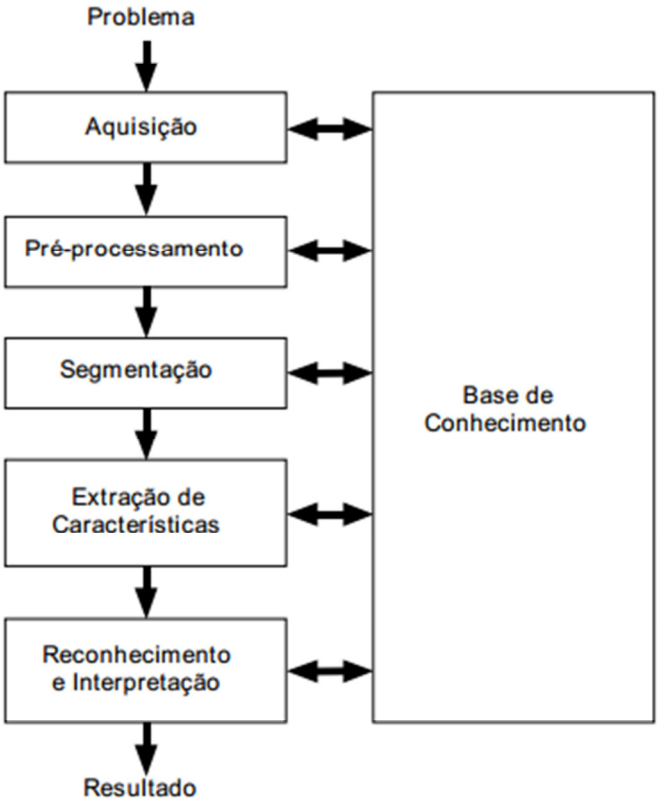

Figura 4 - Um Sistema de Visão Artificial (SVA) e suas principais etapas.
Fonte: (MARQUES FILHO, & VIEIRA NETO, 1999).

As próximas subseções explicitam as características de cada uma das etapas de um sistema de visão artificial apresentado na Figura 4.

2.1.1 Aquisição da imagem

O primeiro passo no processo de reconhecimento de imagem é a aquisição de imagens. Para tanto são necessários um sensor e um digitalizador. O sensor converterá a informação óptica em sinal elétrico e o digitalizador transformará a imagem analógica em imagem digital (KHOSHAFIAN & BAKER, 1996).

Dentre os aspectos de projeto envolvidos nesta etapa, pode-se mencionar: a escolha do tipo de sensor, o conjunto de lentes a utilizar, as condições de iluminação da cena, os requisitos de velocidade de aquisição, a resolução e o número de níveis de cinza da imagem digitalizada, dentre outros. Esta etapa produz à saída de uma imagem digitalizada onde o realce e restauração de Imagem melhoram as

propriedades visuais das imagens (KHOSHAFIAN & BAKER, 1996). Realce livra a imagem de deformações e distorções; várias técnicas de filtragem removem ruídos e borrões em uma imagem. Uma técnica popular de filtragem é a *filtragem mediana*, a qual troca o nível de cinza de cada pixel em uma imagem pelo nível de cinza mediana dos pixels circundantes.

2.1.2 Pré-processamento

A imagem resultante do passo anterior (Aquisição de Imagem) pode apresentar diversas imperfeições, tais como: presença de pixels ruidosos, contraste e/ou brilho inadequado, caracteres (especialmente os dígitos da placa do veículo) interrompidos ou indevidamente conectados etc (KHOSHAFIAN & BAKER, 1996).

A função da etapa de pré-processamento é aprimorar a qualidade da imagem para as etapas subsequentes. As operações efetuadas nesta etapa são ditas de baixo nível porque trabalham diretamente com os valores de intensidade dos pixels, sem nenhum conhecimento sobre quais deles pertencem aos valores da placa do veículo, a outras informações ao fundo. A imagem resultante desta etapa é uma imagem digitalizada de melhor qualidade que a original (KHOSHAFIAN & BAKER, 1996).

2.1.3 Segmentação

A tarefa básica da etapa de segmentação é a de dividir uma imagem em suas unidades significativas, ou seja, nos objetos de interesse que a compõem. Esta tarefa, apesar de simples de descrever, é das mais difíceis de implementar (KHOSHAFIAN & BAKER, 1996).

No caso específico da placa do veículo, é possível que o problema seja dividido em duas etapas: em um primeiro momento os algoritmos de segmentação tentarão localizar o valor da placa do restante das informações para posteriormente, trabalhando sobre esta sub imagem, segmentar cada dígito individualmente. Segundo esta linha de raciocínio, este bloco produzirá à saída de sub imagens, cada qual correspondendo a um dígito da placa. Nesta etapa consta também a análise de

Imagem (KHOSHAFIAN & BAKER, 1996) que tem o objetivo de entender as características da imagem e as características dos objetos que ela contém. Exemplos de características de imagem incluem brilho, textura, e cor. Características dos objetos da imagem incluem forma, tamanho e contorno.

2.1.4 Extração de Características

Esta etapa procura extrair características das imagens resultantes da segmentação através de descritores que permitam caracterizar com precisão cada dígito e que apresentem bom poder de discriminação entre dígitos parecidos, como o '5' e o '6'. Estes descritores devem ser representados por uma estrutura de dados adequada ao algoritmo de reconhecimento. É importante observar que nesta etapa a entrada ainda é uma imagem, mas a saída é um conjunto de dados correspondentes àquela imagem (KHOSHAFIAN & BAKER, 1996).

Para maior clareza, supõe-se que os descritores utilizados para descrever um caractere sejam as coordenadas normalizadas x e y de seu centro de gravidade e a razão entre sua altura e largura. Neste caso, um vetor de três elementos é uma estrutura de dados adequada para armazenar estas informações sobre cada dígito processado por esta etapa (KHOSHAFIAN & BAKER, 1996).

2.1.5 Reconhecimento e Interpretação

Nesta última etapa do sistema, (KHOSHAFIAN & BAKER, 1996) denomina-se reconhecimento o processo de atribuição de um rótulo a um objeto baseado em suas características, traduzidas por seus descritores. A tarefa de interpretação, por outro lado, consiste em atribuir significado a um conjunto de objetos reconhecidos.

2.1.6 Base de Conhecimento

Todas as tarefas das etapas descritas acima pressupõem a existência de um conhecimento sobre o problema a ser resolvido, armazenado em uma base de conhecimento, cujo tamanho e complexidade podem variar devido a qualidade e luminosidade da imagem (KHOSHAFIAN & BAKER, 1996).

Idealmente, esta base de conhecimento deveria não somente guiar o funcionamento de cada etapa, mas também permitir a realimentação entre elas. Por exemplo, se a etapa de representação e descrição recebesse 7 caracteres ao invés de 8, ela deveria ser capaz de realimentar a etapa de segmentação (provável responsável pela falha) para que esta procurasse segmentar novamente a sub imagem 'suspeita' (aquela de maior largura), buscando dividi-la em duas (KHOSHAFIAN & BAKER, 1996).

Esta integração entre as várias etapas da base de conhecimento ainda é um objetivo difícil de alcançar e não está presente na maioria dos SVAs existentes atualmente (KHOSHAFIAN & BAKER, 1996).

2.4 OCR (Reconhecimento Ótico de Caracteres)

OCR significa *Optical Character Recognition* ou Reconhecimento Ótico de Caracteres, é o processo pelo qual o computador consegue “ler” o texto contido em uma imagem. O mecanismo ótico é uma interpretação de imagens e compreensão de sinais em forma textual (AIM Global, 2000). O reconhecimento ótico é feito pelo homem pelos olhos. Enquanto este reconhecimento é feito pelos olhos (entrada), a interpretação (processamento) varia de pessoa para pessoa de acordo com muitos fatores como qualidade da câmera do dispositivo, porcentagem de ruído das imagens, formato dos caracteres de entrada, entre outros.

As primeiras versões de OCR eram simples e requeriam calibragem do sistema. Esta calibragem constituía-se da prévia programação de imagens associadas a cada caractere e utilizando-se de apenas um tipo de fonte. Implementações atuais abordando OCR abrangem tanto caracteres do alfabeto latino quanto caracteres orientais como o chinês, japonês, etc. Algumas dessas implementações são de código fonte aberto, mas ainda encontram problemas na integração com os sistemas operacionais (SHUNJI MORI, 1999).

O OCR é utilizado para a entrada automática de dados em um computador, armazenamento, compartilhamento ou processamento. Um sistema de OCR é composto basicamente de: Escaneamento Ótico, Localização e Segmentação, Pré-processamento, Extração de características, Classificação e Pós-processamento, e sua funcionalidade segue desta forma; após a digitalização da imagem ela passa por um processo de localizações das regiões que contém o texto, dentro de cada região é então segmentada e são extraídos os símbolos. Cada símbolo extraído pode passar por um pré-processamento que envolve a eliminação de todo o ruído possível a fim de facilitar a extração das características deste símbolo. O símbolo é então identificado ao comparar as características pertencentes a ele com as descrições das classes de símbolos que são obtidas através de uma fase de aprendizado, ocorrida antes do início do processo de OCR. Por fim a informação contextual é utilizada para reconstruir as palavras e números do texto original (SHUNJI MORI, 1999).

2.5 Tesseract OCR

Biblioteca responsável pelo reconhecimento ótico dos caracteres de entrada, originalmente desenvolvido em C pela *Hewlett Packard* (HP) entre 1985 e 1995. Atualmente, o projeto é continuado pelo Google que fornece também uma interface Java para utilização nativa das funções (TESSERACT-OCR, 2014). Para que a biblioteca seja suportada pela plataforma *Android*, ela foi compilada em uma biblioteca que fornece uma API Java provida de acesso às funções *Tesseract* compiladas nativamente. A API Java resume-se à classe *TessBaseAPI*, esta classe é uma simples interface Java para o mecanismo de OCR da *Tesseract*.

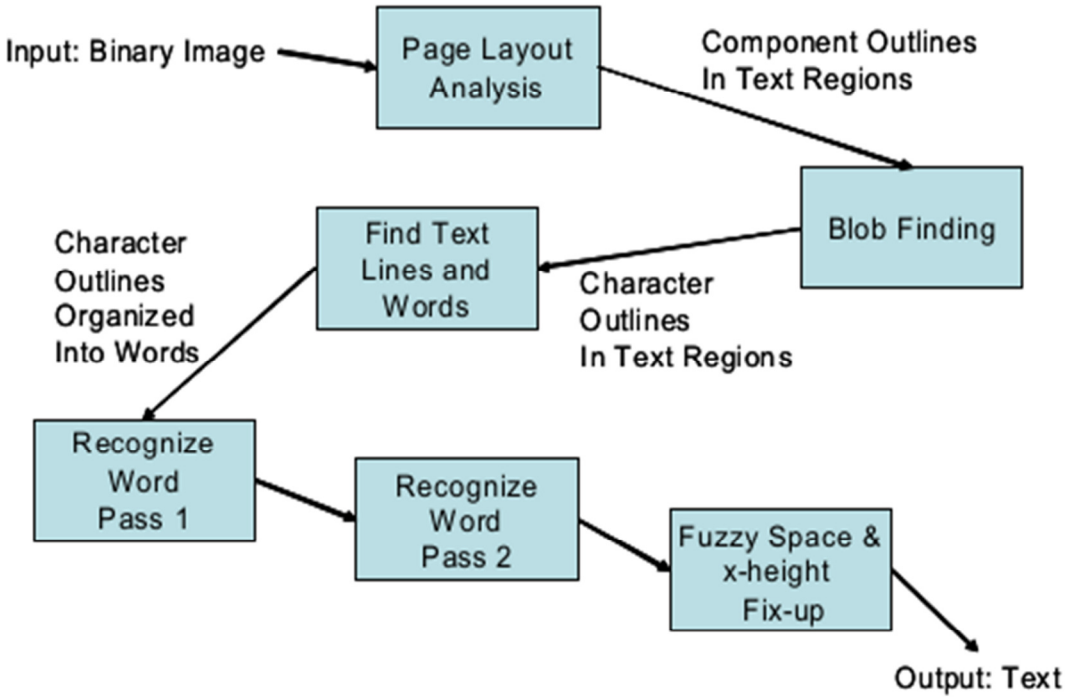

Figura 5 - Componentes básicos da biblioteca Tesseract.
Fonte: Tesseract-OCR, 2014

A Figura 5 apresenta o funcionamento do *Tesseract*, na qual utiliza-se como entrada de dados a imagem que é *binarizada* separando objetos como o fundo da imagem, blocos de texto, linhas e figuras. Na próxima etapa, os objetos são analisados delimitando linhas em torno dos objetos referentes ao texto.

O texto delimitado é analisado e quebrado em palavras de acordo com o espaçamento entre as letras. No passo seguinte as palavras são analisadas pelo classificador do idioma, uma de cada vez, caso seja reconhecida, é classificada como um dado de treinamento. Caso o classificador tenha utilizado esta palavra para o reconhecimento, um segundo passo é realizado a partir do início do texto (TESSERACT-OCR, 2014).

Já na Figura 6, as palavras de uma imagem são processadas por duas vezes. No primeiro processamento, palavras bem sucedidas, sendo aquelas que estão num dicionário e não são perigosamente ambíguas, são passados para um classificador adaptativo de treinamento. Assim que o classificador adaptativo tem amostras suficientes, ele pode fornecer resultados da classificação, mesmo no primeiro processamento. Para palavras que são classificadas como perigosamente ambíguas serão processadas por uma segunda vez (TESSERACT-OCR, 2014).

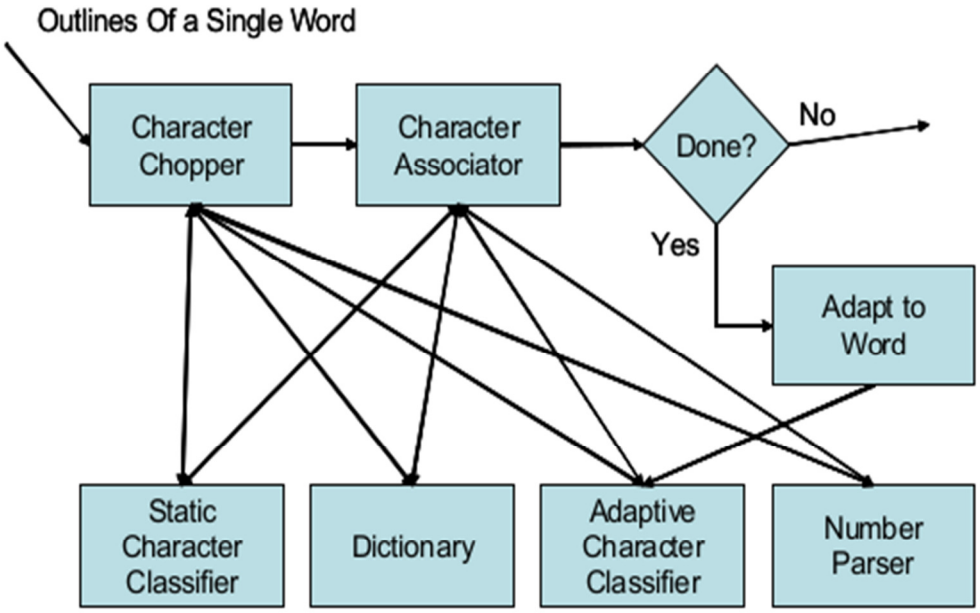

Figura 6 - Diagrama de reconhecimento de palavra Tesseract.
Fonte: Tesseract-OCR, 2014

2.6. Reconhecimento de imagem em dispositivos móveis

O reconhecimento de imagem em dispositivos móveis pode ser utilizado para área de Medicina conforme Marques Filho e Vieira Neto (1999), mas também pode ser utilizado para diminuir a dificuldade de deficientes visuais para se locomover (SOUSA, 2013) utilizando a visão computacional, que tem como objetivo emular a visão humana, tendo assim uma imagem como entrada e uma interpretação desse conteúdo é então fornecida como saída.

Para executar esse processo com êxito é necessário primeiramente realizar a aplicação de filtros que permitam remover ruído que, por definição, representa qualquer informação não desejada na imagem e, após a realização dessa etapa, torna-se necessária à execução dos processos de segmentação (agrupamento de pixels de forma a definir regiões) e reconhecimento de objetos através de algoritmos apropriados. Por último utilizam-se os dados recebidos para geração de informações que auxiliem na resolução do problema proposto (STRINGHINI, 2011).

De acordo com Rodrigues Ivo Almeida (2014), os dispositivos móveis têm hoje em dia muitos subsistemas, entre os quais: câmara de filmar/fotografar, flash, GPS, giroscópios, wireless, sensores de luz, entre outros, e com esses recursos é possível rastrear objetos pelo meio de marcadores. Um Marcador consiste em um

padrão conhecido, tipicamente num quadrado preto e branco, que podem ser visualizados, nas Figuras 7, 8, e 9. Desta forma é simples efetuar o reconhecimento permitindo um custo muito baixo de complexidade computacional, tempo e elevada taxa de sucesso (TEIXEIRA, 2013).

Figura 7- Exemplo de Marcador
Fonte: (TEIXEIRA, 2013).

Figura 8 - Exemplo de Código de Barra 1D
Fonte: (TEIXEIRA, 2013).

Figura 9 - Exemplo de Código de Barra 2D
Fonte: (TEIXEIRA, 2013).

3 DESENVOLVIMENTO

Nesta seção serão apresentados especificações e características do sistema assim como explicações sobre a utilização da biblioteca *Tesseract* -OCR.

3.1 SISTEMA PROPOSTO

O protótipo proposto neste trabalho consiste em um aplicativo que utiliza uma biblioteca de OCR, *Optical Character Recognition*, para tradução de imagens em texto, e um serviço web para atualização de dados no dispositivo. Todas as tecnologias utilizadas no desenvolvimento do aplicativo podem ser utilizadas em qualquer dispositivo com o sistema operacional *Android* 2.2 ou superior.

3.2 DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

O protótipo identificador de placa veicular foi desenvolvido para Android com integração com sistema externo via Web Service, armazenando os dados localmente no dispositivo móvel em um banco de dados SQLite para melhor resposta na procura da placa do veículo e não gerar dependência de conexão de externa na identificação da placa do veículo.

O banco de dados SQLite foi escolhido devido a compatibilidade com o Android e sua facilidade na implementação, também por um software livre e suportar bases de até 2 *terabytes* o mesmo armazena todas as informações dos alunos, responsáveis e seus respetivos veículos para ser consultado localmente.

O diagrama da Figura 10, apresenta o funcionamento do aplicativo de forma geral, é possível observar na figura o veículo como elemento fornecedor da entrada de dados, que deve ser realizada por meio do teclado ou da câmera do dispositivo com captura da imagem da placa do veículo. Posteriormente esses dados serão processados pelo aplicativo. A comunicação com o Webservice ocorre no momento que o usuário executa a funcionalidade de sincronização, que é responsável por manter os dados do dispositivo atualizado.

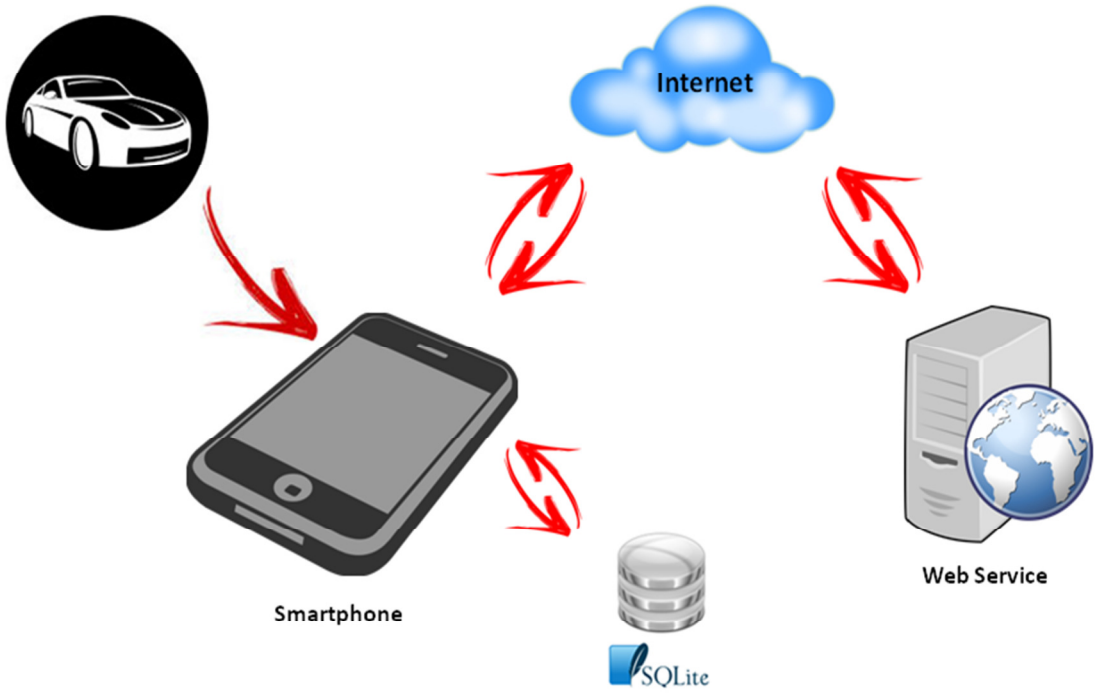

Figura 10 – Fluxo do sistema
Fonte: Dos Autores

3.3 CARACTERÍSTICAS DA IMAGEM ALVO

As placas de veículos seguem um padrão e devem atender as especificações determinadas pelo Conselho Nacional de Trânsito - CONTRAN. Todos os veículos devem possuir placas de identificação contendo 7 (sete) caracteres alfanuméricos individualizados sendo o primeiro grupo composto por 3 (três), resultante do arranjo, com repetição de 26 (vinte e seis) letras, tomadas três a três, e o segundo grupo composto por 4 (quatro), resultante do arranjo, com repetição, de 10 (dez) algarismos, tomados quatro a quatro (DENATRAN, 2014).

De acordo com a resolução 231 de 15 de março de 2007 do CONTRAN as placas de identificação dos veículos devem ter dimensões de 400 milímetros de largura e 130 milímetros de altura como mostra a Figura 11. Em motocicletas, monaretas, ciclomotores e triciclos motorizados as dimensões são de 187 milímetros de largura e 136 milímetros de altura como mostra a Figura 12 (DENATRAN, 2014).

Figura 11 - Dimensões da placa de Automóvel
Fonte: Denatran, 2014

Figura 12 - Dimensões da Placa de Moto
Fonte: DENATRAN, 2014.

3.4 REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Essa seção descreve os requisitos funcionais e não funcionais referentes ao aplicativo proposto.

3.4.1 Requisitos Não Funcionais

O aplicativo deve ter uma resposta rápida às requisições devido ao fluxo de entrada e saída de alunos em curto prazo de tempo. Deve ser de fácil interação com o usuário e uma interface objetiva. O aplicativo opera *off-line* garantido seu funcionamento independentemente de rede de dados e energia elétrica.

A base de dados utilizada na verificação dos veículos é armazenada no próprio dispositivo e importada de um sistema externo.

3.4.2 Requisitos Funcionais

O aplicativo deve permitir leitura da placa do veículo por meio da câmera do *smartphone*.

O aplicativo deve permitir leitura da placa do veículo por meio do teclado.

O aplicativo deve transformar a imagem capturada em conteúdo texto.

O aplicativo deve consultar a base de dados para emitir uma resposta ao usuário, que pode ser: placa cadastrada, placa não cadastrada.

3.5 DIAGRAMA DE CASOS DE USO GERAL

O diagrama de Casos de Uso descreve a funcionalidade proposta para o aplicativo desenvolvido no ponto de vista do usuário, faz parte de um grupo de artefatos da linguagem UML (*Unified Modeling Language*) usado para modelar e documentar sistemas orientados a objetos. Para o sistema Identificador foram desenvolvidos duas funcionalidades principais que são apresentadas no diagrama da Figura 13.

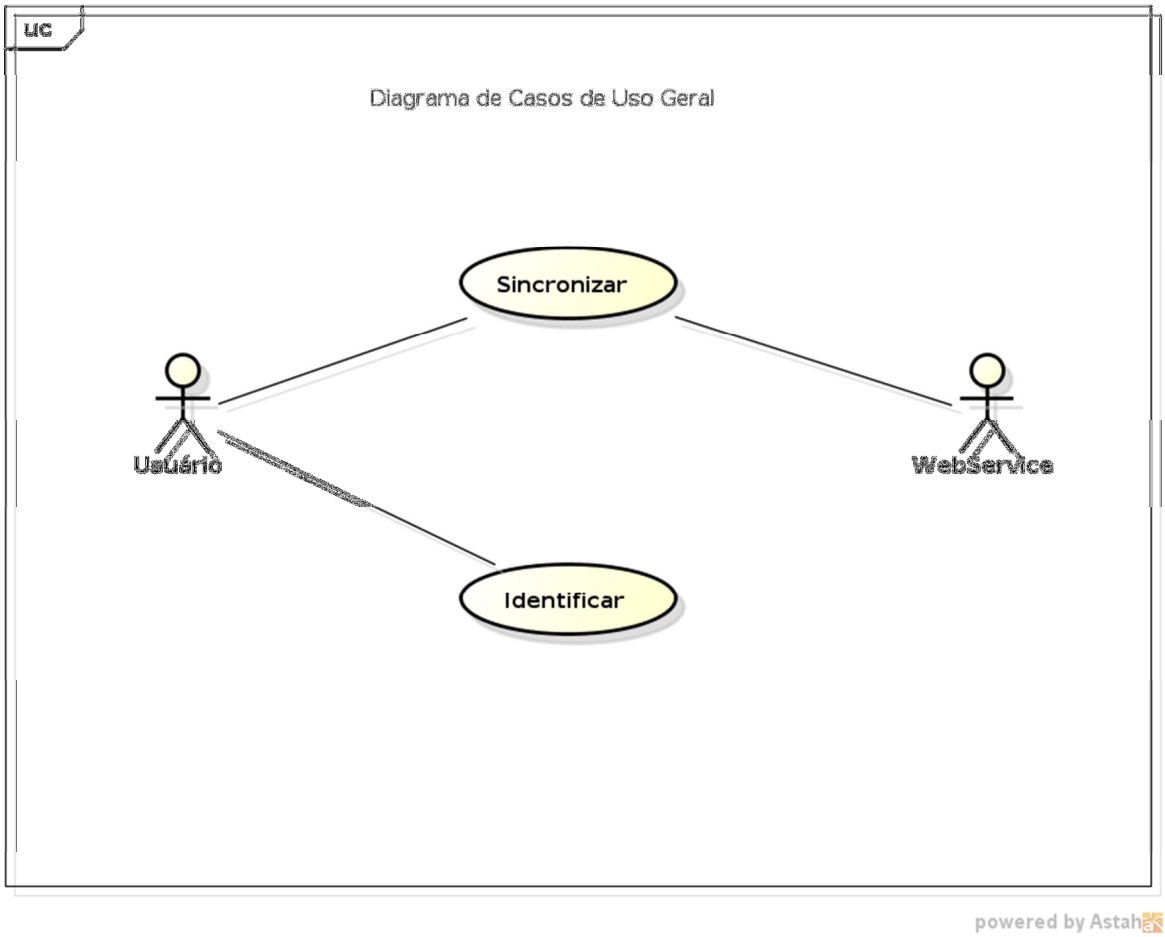

Figura 13 - Diagrama de Caso de Uso
Fonte: Dos Autores

A representação dos atores neste diagrama de Casos de Uso se dá pela figura do usuário e do WebService que interagem com a aplicação. O ator WebService receberá requisições para atualização de dados quando o ator usuário solicitar ao sistema pela funcionalidade “Sincronizar dados”. Esta funcionalidade mantém os dados do aplicativo atualizados, realizando uma chamada ao *WebService* informando o *username* e senha para que o *WebService* forneça os dados para atualização do aplicativo.

No quadro de especificação do Caso de Uso “Sincronizar” são descritas as etapas para o usuário executar a sincronização dos dados apresentados no Quadro 2.

Quadro 2 - Especificação do Caso de Uso Sincronizar

Sincronizar: possibilita o ator usuário manter os dados atualizados no dispositivo.	
Cenário Normal	1. O sistema apresenta a tela de sincronização (Figura 15); 2. O usuário pressiona o botão sincronizar; 3. O sistema apresenta tela de login; 4. O usuário digita o usuário; 5. O usuário digita a senha; 6. O sistema realiza a conexão com o *WebService*; 7. O *WebService* verifica as informações de *login*; 8. O sistema realiza a importação dos dados do *WebService*; 9. O sistema apresenta tela com resultado da sincronização.
Cenário de exceção	1. No passo 7 do Cenário Normal o *WebService* verifica que usuário e/ou senha estão incorretos; 2. O sistema apresenta mensagem de erro; 3. O fluxo retorna ao passo 3 do Cenário Normal.

Fonte: Dos Autores

O Cenário de Exceção contempla a falha no acesso ao sistema, por erro no usuário ou senha do usuário.

A funcionalidade Identificar permite ao ator usuário realizar a identificação de um veículo pré-cadastrado fornecendo os dados de registro do aluno. Esta funcionalidade tem como pré-requisito a sincronização de dados descrita no Quadro 2, sendo a escola responsável por manter os cadastros dos alunos, pais, responsáveis e veículos atualizados.

Como cenário para esta funcionalidade, tem-se a figura do porteiro da escola como usuário do aplicativo, o veículo a ser identificado é o carro do pai ou responsável pelo aluno. A identificação ocorre no momento que os pais ou responsáveis chegam à escola para buscar o aluno.

No Quadro 3 estão descritas às etapas para o usuário executar a funcionalidade Identificar.

Quadro 3 - Especificação do Caso de Uso Identificar

Identificar: possibilita o ator usuário a realizar a identificação do veículo.	
Cenário Normal	1. O sistema apresenta a tela de identificação (Figura 16); 2. O usuário pressiona o botão de captura para ativa a câmera do dispositivo; 3. O usuário realiza a captura da imagem focando a câmera do dispositivo na frente do veículo buscando a placa; 4. O usuário realiza o corte da imagem centralizando a placa; 5. O sistema realiza o processo de OCR; 6. O sistema realiza a busca da placa na base; 7. O sistema apresenta na tela os dados de identificação.
Cenário de exceção	1. No passo 5 do Cenário Normal o processo de OCR não reconhece adequadamente o texto da imagem; 2. O usuário informa a placa do veículo.

Fonte: Dos Autores

O cenário de exceção ocorre quando o processo de reconhecimento da imagem não é bem sucedido, quando isso ocorre o sistema não consegue localizar a placa do veículo no cadastro e é então necessário que o usuário forneça essa informação via teclado.

3.6 DIAGRAMA DE NAVEGAÇÃO

No diagrama da Figura 14, é apresentado o fluxo de navegação das telas do aplicativo Identificador, sua navegação foi desenvolvida com o objetivo de manter fácil a interação do usuário com uma interface objetiva. São três as principais telas do aplicativo, tela de Identificação, tela de Sincronização e tela de Observação. A tela de Identificação é apresentada primeiramente, assim que o aplicativo é executado, e nesta tela podemos navegar para tela de Sincronização ou para tela de Observação.

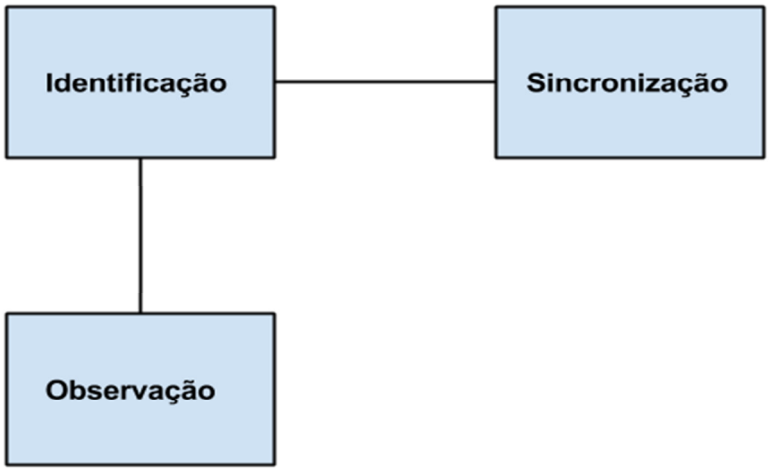

Figura 14 – Diagrama de Navegação
Fonte: Dos Autores

3.7 TELAS DO SISTEMA

Esta seção apresenta as telas do sistema desenvolvidas para um *smartphone-Android* com tela de 4” ou maior.

Para o desenvolvimento foi utilizada a ferramenta *MockFlow* (MOCKFLOW, 2014), já que esta é caracterizada por seu fácil manuseio com a organização dos componentes na tela. Com essa ferramenta foi possível trabalhar de forma compartilhada, utilizando o processo de compartilhamento do projeto para que os dois membros da equipe pudessem atuar de forma paralela.

3.7.1 Tela de importação de dados do sistema externo.

A tela de importação de dados do sistema externo, visualizada na Figura 15, sincroniza as informações de alunos, responsáveis e veículos com o WebService.

Ao clicar no botão “Sincronizar” o aplicativo abre uma tela de *Login* para informar usuário e senha, caso o usuário e senha sejam válidos, os dados devem ser importados conforme a unidade do funcionário. Com a importação dos dados, são preenchidos os campos com o número de registros encontrados na base de dados:

- Usuário Logado
- Unidade
- Alunos
- Responsáveis
- Veículos

Figura 15 - Sincronização dos dados
Fonte: Dos Autores

3.7.2 Tela de Identificação do veículo

A Tela de identificação do veículo, apresentada na Figura 16, tem a principal função de capturar a imagem da placa do veículo e verificar se o mesmo está autorizado a entrar na unidade. Clicando no botão Foto, o funcionário ou segurança captura a imagem e o sistema processa a imagem, mostrando a placa no campo Veículo.

Figura 16 - Identificação do Veículo
Fonte: Dos Autores

As Figuras 17, 18 e 19 apresentam as respostas do sistema ao identificar dependendo da situação da placa:

Figura 17 - Acesso não identificado
Fonte: Dos Autores

- Não foi identificado o valor da placa do veículo. O funcionário ou segurança pode clicar no campo “Veículo” para informar o valor da placa manualmente.

Figura 18 - Acesso Restrito
Fonte: Dos Autores

- A placa foi identificada e verificada. Há registro de restrição de acesso para o veículo.

Figura 19 - Acesso Autorizado
Fonte: Dos Autores

- A placa foi identificada e não foi encontrada nenhuma restrição para o veículo, sendo assim sua entrada está autorizada.

Com a identificação da placa, os seguintes campos serão preenchidos:

- Dados do Aluno
  - Matricula
  - Nome
  - Série
  - Curso
  - Turno

- Dados do Responsável
    - Nome
    - Tipo Parentesco

Para verificar as restrições de acesso caso o responsável esteja nessa condição, basta clicar na imagem de Acesso Restrito (Figura 18).

3.7.3 Tela de Observação adicional.

A tela de observação adicional que pode ser visualizada na Figura 20, o funcionário salva uma possível observação no momento da identificação do veiculo caso julgue necessário.

Figura 20 - Observação adicional
Fonte: Dos Autores

3.8 AMBIENTE DE DESENVOLVIMENTO

O aplicativo foi desenvolvido em um ambiente composto de: Sistema Operacional *Linux* distribuição Ubuntu 13.04, 64 bits; *IDE* Eclipse versão Indigo *Service Release* 2; *JDK* (*Java Development Kit*) versão: Java 7 *update* 9; *Android SDK* versão 22.3; *Android NDK* r10; Bibliotecas *Tesseract* 3.04 e *Leptonica* 1.71.

3.9 IMPLEMENTAÇÃO

Para o desenvolvimento da *Activity* responsável pela sincronização dos dados com o WebService foi utilizada a classe *JsonDownloadAsyncTask* que *extende AsyncTask*. Na Figura 21 segue o trecho do código responsável por transformar o *Json* recebido do WebService em um objeto da classe Retorno. A classe Retorno é um *Model* que reúne os atributos contendo dados do aluno, responsável e veículo.

```
50
51          //Transforma json em objeto do tipo Retorno;
52          JsonParse<Retorno> retornoParse = new JsonParse<Retorno>() {
53
54              @Override
55              public Retorno parse(JSONObject json) throws JSONException {
56                  Retorno retorno = new Retorno();
57
58                  retorno.setUsuario(json.getString("usuario"));
59                  retorno.setUnidade(json.getString("unidade"));
60                  retorno.setAluno(json.getString("aluno"));
61                  retorno.setNomeAluno(json.getString("nomeAluno"));
62                  retorno.setCurso(json.getString("curso"));
63                  retorno.setSerie(json.getString("serie"));
64                  retorno.setTurno(json.getString("turno"));
65                  retorno.setSit_aluno(json.getString("sit_aluno"));
66                  retorno.setTipo(json.getString("tipo"));
67                  retorno.setResp(json.getString("resp"));
68                  retorno.setNomeResp(json.getString("nomeResp"));
69                  retorno.setCpf_cnpj(json.getString("cpf_cnpj"));
70                  retorno.setRg(json.getString("rg"));
71                  retorno.setTelefone(json.getString("telefone"));
72                  retorno.setPlaca(json.getString("placa"));
73                  retorno.setRestrito(json.getBoolean("restrito"));
74                  retorno.setVeiculo(json.getString("veiculo"));
75                  retorno.setCor(json.getString("cor"));
76                  retorno.setObservacao(json.getString("observacao"));
77
78                  return retorno;
79              }
80          };
81
```

Figura 21 – Converte *Json* para Model
Fonte: Dos Autores

O trecho de código apresentado na Figura 22 é responsável por fazer o download de forma assíncrona do *Json*. Foi desenvolvido um servidor *WebService* com dados fictícios para o desenvolvimento desta parte do aplicativo.

```
82
83      JsonDownloadAsyncTask<Retorno> download = new JsonDownloadAsyncTask<Retorno>(this, retornoParse);
84
85      download.execute("http://gepoweb.com.br:8080/WebService/acesso/_"+token);
86
```

Figura 22 – Download do Json
Fonte: Dos Autores

Para a chamada do OCR foi implementado o método *initOcrEngine* que recebe como parâmetro a imagem e retorna o texto nela contido. A Figura 23 apresenta a implementação do código em que a variável *DATA_PATH*, determina onde a biblioteca esta localizada, o parâmetro “eng” é o idioma que foi utilizado. A escolha do idioma Inglês deu-se pela razão de não haver caracteres acentuados nas placas dos veículos.

```
84
85      private String initOcrEngine(Bitmap bitmap) {
86          // mecanismo de OCR da biblioteca Tesseract
87          TessBaseAPI baseApi = new TessBaseAPI();
88          baseApi.setDebug(true);
89
90          // o segundo parametro "eng" é o idioma da biblioteca que foi usado.
91          baseApi.init(Utils.DATA_PATH, "eng");
92          baseApi.setImage(bitmap);
93          String texto = baseApi.getUTF8Text();
94          baseApi.end();
95
96          texto = texto.replaceAll("[^a-zA-Z0-9]+", " ");
97          return texto.trim();
98      }
99
```

Figura 23 – Método initOcrEngine
Fonte: Dos Autores

A Figura 24 mostra o diagrama de implantação, em que o aplicativo Identificador foi desenvolvido para ser executado em um sistema operacional Android instalado no dispositivo móvel e então realizar sua busca no banco de dados SQLite.

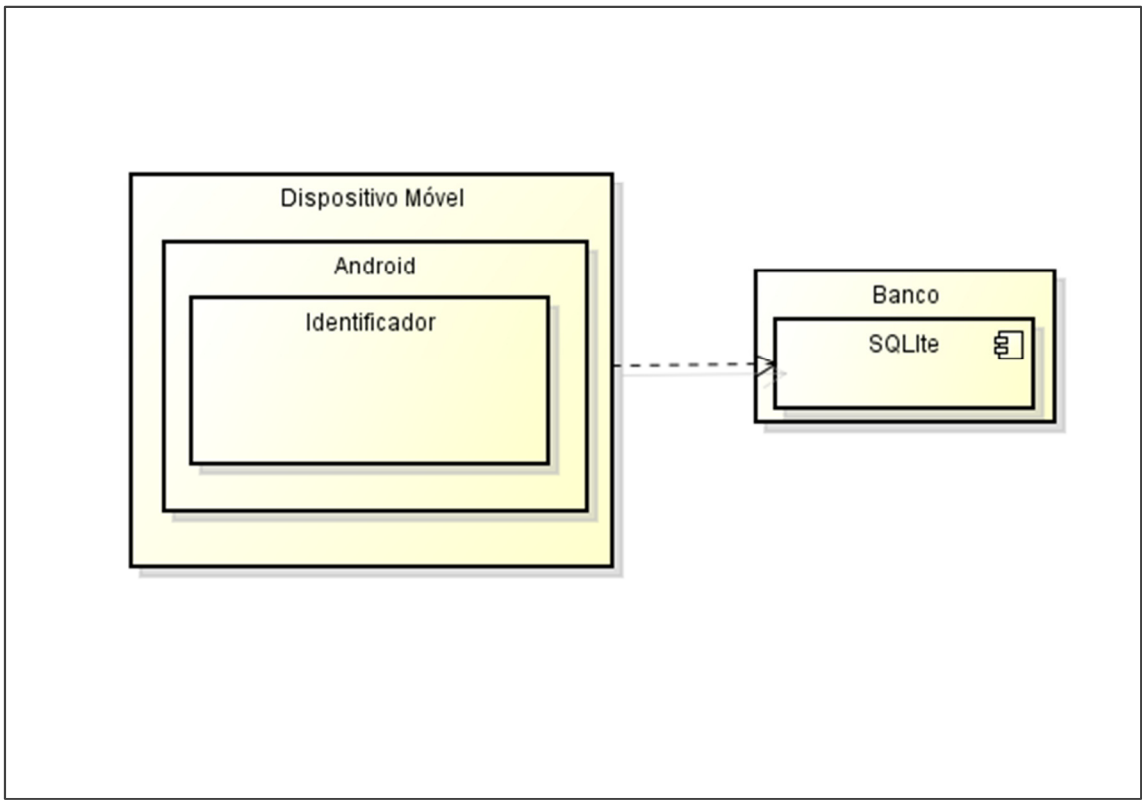

Figura 24 - Diagrama de Implantação
Fonte: Dos Autores

3.10 TESTES DO ARTEFATO

Na etapa dos testes foi verificado que a sincronização com o *Webservice* ocorreu sem problemas, utilizando tanto a conexão *Wireless* quanto o 3G. A captura da imagem pela câmera do dispositivo foi realizada com sucesso e quando esta foi enviada para a *API Tesseract* o resultado não atingiu 100% de acertos, devido ao tratamento da imagem não ser eficiente por causa das variações da iluminação, contraste e proximidade da placa do veículo. Os resultados obtidos na fase de teste do OCR estão representados no **Erro! Fonte de referência não encontrada.**.

Quadro 4 – Testes do OCR e resultados obtidos

Imagem da Placa	Texto reconhecido		Resultado
	1º	B457	Falha
	2º	6457	Sucesso
	3º	B457	Falha
	1º	234	Falha
	2º	234	Falha
	3º	234	Falha
	1º	3828	Falha
	2º	3828	Falha
	3º	3826	Sucesso
	1º	3458	Falha
	2º	3456	Sucesso
	3º	3456	Sucesso
	1º	9595	Sucesso
	2º	9595	Sucesso
	3º	9595	Sucesso
	1º	5678	Sucesso
	2º	5678	Sucesso
	3º	5678	Sucesso
	1º	4567	Sucesso
	2º	4587	Falha
	3º	4567	Sucesso
	1º	5748	Sucesso
	2º	Q48	Falha
	3º	5748	Sucesso

Fonte: Dos Autores

Nos testes com placas de veículos nem todas as tentativas foram bem-sucedidas na obtenção do texto pelo mecanismo de OCR. Para uma mesma placa foram feitas três tentativas e obtidos resultados diferentes para a maioria das capturas. O fato de utilizar o *flash* da câmera já altera o resultado do

reconhecimento. Distância e posição da câmera também afetaram os resultados. Para melhorar o resultado do OCR foi utilizado apenas o valor numérico das placas, desconsiderando as letras. Enviando apenas a parte numérica da imagem para o OCR obtemos melhorias significativas nos resultados.

O motivo para examinar apenas os números, seria para simplificar a leitura, ao invés do OCR identificar a qual das 36 classes de caracteres (10 números e 26 letras) a mesma pertence. Evitando confusões como diferenciar o caractere relativo ao numero “zero” daquele que representa a letra “O” (ANPR, 2014).

Nos testes de reconhecimento da imagem, foi obtido um resultado de 58% de acertos na identificação da placa e o mecanismo de OCR apresentou uma resposta satisfatória em relação ao desempenho. Vale ainda considerar que os testes realizados com entrada via teclado tiveram sucesso em todas as ocasiões.

.

4 CONSIDERAÇÕES FINAIS

Este trabalho consistiu em desenvolver um protótipo que auxilie o porteiro ou segurança a identificar os veículos dos responsáveis que acessam as unidades dos Colégios para buscar os alunos, indicando se é permitida a sua entrada ou não.

Durante o desenvolvimento do projeto foram realizadas pesquisas sobre dispositivos móveis, reconhecimento de imagem e como os dois citados respectivamente podem trabalhar junto com a utilização de algoritmos de leitura de imagens.

Foi desenvolvido o módulo de sincronização e captura de imagem, em que a sincronização é feita do dispositivo com o sistema externo. O módulo de captura de imagem foi desenvolvido e implementado para trabalhar com a biblioteca de OCR, porém, como visto no Quadro 4 o reconhecimento da imagem pelo processamento do OCR obteve 58% de sucesso. No momento da captura da imagem, caso a imagem não seja identificada, o usuário do aplicativo deve informar os dados da placa manualmente.

O fator motivacional importante para o desenvolvimento deste trabalho foi aumentar a segurança em Instituições de ensino, utilizando a tecnologia de reconhecimento de imagem digital para identificar e controlar o acesso de veículos.

Duas sugestões para trabalhos futuros podem ser, a primeira quando o responsável for autorizado a entrar na unidade, na sala onde os alunos esperam seus responsáveis, um painel luminoso emitirá um sinal com a identificação do aluno para o inspetor poder localizar e assim ficando mais ágil a busca do aluno. A segunda seria identificar o veículo e sua posição no estacionamento da empresa com base no *Google Maps* e verificar se o mesmo se encontra estacionado em local permitido conforme regra da empresa.

REFERÊNCIA BIBLIOGRÁFICA

ANDROID. **Introduction to Android.** 2014.
Disponível em: <http://developer.android.com/guide/index.html> Acesso em 18/08/2014.

ALMEIDA R. I., **Interface baseada em objetos visuais usando dispositivos móveis aplicação a serviço de cinema**. Dissertação de Mestrado. Faculdade de Engenharia da Universidade do Porto. Porto. 2014 Disponível em: <http://paginas.fe.up.pt/~ei11030/dissertation.pdf> Acessado em 18/08/2014.

DENATRAN. **Departamento Nacional de Trânsito**. Disponível em: <http://www.denatran.gov.br/download/Resolucoes/RESOLUCAO_231.pdf> Acesso em 15/06/2014

GOMES, J.; VELHO, L., **Computação Gráfica: Imagem**. Rio de Janeiro:IMPA/SBM, 1994.

GONZALEZ, R. C.; WOODS, R. E. **Processamento de Imagens Digitais**. São Paulo: Editora edgard lücher LTDA, 2000.

GRSA. Pais prezam mais a segurança do que qualidade em escolas. 2010.
Disponível em <http://www.grsa.com.br/imprensa/PAIS-PREZAM-MAIS-A-SEGURANCA-DO-QUE-QUALIDADE-EM-ESCOLAS.asp> Acessado em 20/07/2014.

JAIN, A. K. **Fundamentals of Digital Image Processing.** Prentice Hall, 1989

ANPR. **Automatic number plate recognition.** Disponível em http://javaanpr.sourceforge.net/anpr.pdf Acessado em 30/09/2014.

JSON. **Introdução ao JSON**.2014. Disponível em: <http://json.org/json-pt.html>. Acesso em 18/08/2014

KHOSHAFIAN, S.; BAKER, A. B. **Multimedia and Imaging Databases.** Morgan Kaufmann Publishers, 1996.

LECHETA, R. R. **Google Android: aprenda a criar aplicativos para dispositivos móveis com o Android SDK**. 2. Ed; São Paulo: Novatec Editora 2010.

ROBINSON, M., KALAKOTA, R.**M-Business Tecnologia Móvel e Estratégia de Negócios.** 1. Ed; Bookman. 2002.

MARQUES FILHO, O.; VIEIRA NETO, H.. **Processamento Digital de Imagens**, Rio de Janeiro: Brasport, 1999. Disponível em:<http://pessoal.utfpr.edu.br/hvieir/download/pdi99.pdf> Acesso em 17/06/2014.

MEIER, R. **Professional Android Application Development**. [S.l.]: Wrox, 2009.

MELLO, C. A. B. **Aspectos Computacionais do Processamento de Imagens de Documentos Históricos.** Recife: UFPE, 2002. Tese (Doutorado em Ciência da Computação), Centro de Informática, Universidade Federal de Pernambuco.

MOCKFLOW. **Super-easy Wireframing.**2014. Disponível em: <http://www.mockflow.com> Acesso em 18/08/2014

MORIMOTO, C. E. **Smartphones – Guia prático**. GDH Press e Sul Editores, Fevereiro 2009.

SHUNJI M., HIROBUMI N., HIROMITSU Y. **Optical Character Recognition 1st.** John Wiley & Sons, Inc. New York, NY, USA, 1999.

SOUSA, K. A. O. **Uso de visão computacional em dispositivos móveis para auxílio à travessia de pedestres com deficiência visual**. Dissertação de Mestrado. Universidade Presbiteriana Mackenzie, São Paulo 2013. Disponível em:< http://www.mackenzie.br/fileadmin/PUBLIC/UP_MACKENZIE/servicos_educacionais/stricto_sensu/Engenharia_Eletrica/Kelly_Aparecida_Oliveira_Sousa.pdf> Acessado em 18/08/2014

SQLITE. **Categorical Index Of SQLite Documents**.
Disponível em < http://www.sqlite.org/index.html> acessado em 15/08/2014.

STRINGHINI, *et al*. **Visão Computacional usando OpenCV.** In: PITERI, M. A.; RODRIGUES, J. C. Fundamentos da Visão Computacional. Presidente Prudente: FCT/UNESP-SP, 2011. p. 113-166.

TEIXEIRA H. **Recognition of Visual Objects Using Mobile Devices**. MSc thesis,
Faculdade de Engenharia da Universidade do Porto, 2013.

TESSERACT. **An OCR Engine that was developed at HP Labs between 1985 and 1995 and now at Google.**2014.
Disponível em: <https://code.google.com/p/tesseract-ocr> Acessos entre: 10/07/2014 e 11/08/2014.