UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LUAN ALANO FORMENTIN

SMART COP: APLICATIVO PARA LEITURA E VALIDAÇÃO DE PLACAS VEICULARES UTILIZANDO A TECNOLOGIA ALPR APLICADA NA FISCALIZAÇÃO EM TEMPO REAL DE VEÍCULOS DURANTE UMA ABORDAGEM POLICIAL

CRICIÚMA
2020

LUAN ALANO FORMENTIN

SMART COP: APLICATIVO PARA LEITURA E VALIDAÇÃO DE PLACAS VEICULARES UTILIZANDO A TECNOLOGIA ALPR APLICADA NA FISCALIZAÇÃO EM TEMPO REAL DE VEÍCULOS DURANTE UMA ABORDAGEM POLICIAL

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. Matheus Leandro Ferreira

CRICIÚMA

2020

SMART COP: APLICATIVO PARA LEITURA E VALIDAÇÃO DE PLACAS VEICULARES UTILIZANDO A TECNOLOGIA ALPR APLICADA NA FISCALIZAÇÃO EM TEMPO REAL DE VEÍCULOS DURANTE UMA ABORDAGEM POLICIAL

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 programação para dispositivos móveis.

Criciúma, 07 de agosto de 2020.

BANCA EXAMINADORA

DocuSigned by:
Matheus Leandro Ferreira
FC5CBA7482B14BA...
Prof. Matheus Leandro Ferreira - Esp. - (UNESC) - Orientador

DocuSigned by:
87E742A7495A41A...
Prof. Evânio Ramos Nicoleit - Me. - (UNESC)

DocuSigned by:
B300826000E1448...
Prof. Giácomo Antônio Althoff Bolan - Esp. - (UNESC)

AGRADECIMENTOS

Agradeço primeiramente a Deus, em seguida minha mãe Rosalba da Silva Alano que sempre me incentivou muito nos meus estudos e que me ajudou sempre, agradecer aos meus amigos que também me apoiaram bastante nessa caminhada. Agradecer o professor Matheus, que foi um ótimo orientador e me ajudou muito durante o desenvolvimento do trabalho, sempre atencioso.

A minha madrinha Ivone que me ajudou muito nos momentos mais difíceis, ao Derion Redivo, pessoa que fez de tudo para que eu alcançasse o meu objetivo.

E a todos aqueles que de alguma forma, direta ou indiretamente contribuíram para que eu pudesse concluir esse curso.

RESUMO

Na busca constante por algo perfeito, um dos grandes princípios do ser humano é criar máquinas automatizadas que possam fazer, cada vez mais, o trabalho braçal e repetitivo. O alfabeto romano em conjunto com a escrita surgiu por volta do século VII a. C. Com o passar do tempo e com a melhoria dos sistemas de *hardware*, houve o interesse comercial para pesquisas e desenvolvimento voltadas para o reconhecimento de caracteres (OCR). No entanto, com a rápida expansão da internet nos últimos anos, novos paradigmas de interação humano-computador vêm ganhando força, como por exemplo a *Application Programming Interface* (API) *Vision*. Por este motivo o presente trabalho apresenta as técnicas de processamento digital de imagens (PDI) e a aplicação da API *Vision* para identificação automática de placas veiculares em tempo real. São expostos os princípios da morfologia matemática e algumas técnicas para tratamento de imagens. O objetivo é realizar o estudo da API e explorar suas diversas funcionalidades na identificação de placas em conjunto com o reconhecimento óptico de caracteres (OCR). Sendo assim, pode-se automatizar o processo de abordagem de trânsito por meio de um protótipo de aplicativo para leitura de placas. Os resultados obtidos foram satisfatórios tendo como base diversas placas testadas.

Palavras-chave: Reconhecimento de placas veiculares. Fiscalização em tempo real. Processamento digital de imagens. *Vision* API. Reconhecimento ótico de caracteres.

ABSTRACT

In the constant search for something perfect, one of the great principles of the human being is to create automated machines that can do, every time more, manual and repetitive work. The Roman alphabet together with writing appeared around the VII century b. c. During time and with the improvement of hardware systems, there was a commercial interest for research and development focused on character recognition (OCR). However, with the rapid expansion of the internet in recent years, new paradigms of human-computer interaction have been gaining strength, such as the Application Programming Interface (API) Vision. For this reason, this paper presents the digital image processing techniques (PDI) and the application of API Vision for automatic identification of vehicle plates in real time. They are exposed to the principles of mathematical morphology and some techniques for imaging. The objective is to create a research of the API and explore its many functionalities in the identification of vehicle plates together with optical character recognition (OCR). Therefore, it is possible to automate the transit approach process through a prototype application for identification of vehicle plates. The results obtained were satisfactory based on several vehicle plates tested.

Keywords: Vehicle plate recognition. Real-time inspection. Digital image processing. Vision API. Optical character recognition.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ALPR	*Automatic License Plate Recognition*
API	Application Programming Interface
DETRAN	Departamento Estadual de Trânsito
FURB	Fundação Universidade Regional de Blumenau
HSV	*Hue, Saturation, Value*
IBM	International Business Machines
IDE	*Integrated Development Environment*
IPVA	Imposto sobre a propriedade de veículos automotores
OCR	*Optical Character Recognition*
PDA	Personal Digital Assistants
RAPA	Reconhecimento Automático de Placas Automobilísticas
REST	*Representational State Transfer*
RGB	*Red, Green, Blue*
RPC	*Remote Procedure Call*
UFRJ	Universidade Federal do Rio de Janeiro
UFSC	Universidade Federal de Santa Catarina
UNESC	Universidade do Extremo Sul Catarinense
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

A evolução constante da tecnologia vem trazendo aos órgãos públicos novas formas de gerenciar e organizar informações. Para manter-se atualizados, os setores públicos, estão investindo em inovações de aplicativos que os auxiliam neste processo. A otimização e a facilidade gerada por uma solução mobile vem ganhando força no mercado e se destaca cada vez mais no ambiente de trabalho. Para a polícia civil, por exemplo, a importância de consultas em tempo real torna-se cada dia mais indispensável (REVISTA CIENTÍFICA ELETRÔNICA DE SISTEMAS DE INFORMAÇÃO, 2006).

Para destacar a importância de obter-se acesso a dados em tempo real, Stoner (1999) comenta que somente com informações de acesso imediato e no momento certo, o usuário consegue monitorar o processo na direção correta para posteriormente tomar uma decisão.

A evolução tecnológica combinada com o grande crescimento da necessidade e agilidade nos processos, força as organizações dependerem de aplicativos com eficácia para administrar sua gestão, reduzindo o tempo nos processos de consulta a informação (BEAL, 2004). Para Natale (2002) a automatização é um conjunto de técnicas que nos permite processar informações que anteriormente o homem quem realizava. Os processos não se limitam apenas às indústrias. Inúmeras organizações estão se adequando e utilizando de aplicativos e técnicas processuais visando maximizar ganhos, como por exemplo o tempo, e para as empresas de fiscalização, não é diferente, também estão se adequando neste novo contexto (NATALE, 2002).

Os agentes ou guardas de trânsito, como são conhecidos, por meio de um convênio com a secretaria de segurança pública, fazem a fiscalização e atuam as infrações cometidas nas vias públicas da cidade. Porém, a forma como a infração é aplicada encontra-se ultrapassada: a dificuldade no preenchimento do auto de infração, uma vez que o mesmo é preenchido de forma manual, e a comunicação com as centrais de operação da polícia, torna o processo dificultoso e lento. O tempo, no qual poderia ser usufruído para novas abordagens, se torna prolongado devido à busca das informações e respostas das centrais, que nem sempre estão disponíveis para auxiliar o agente de trânsito. Por conta disso, pode-se criar a

seguinte questão problema: Como reduzir o tempo do agente de trânsito ou policial durante a sua abordagem?

O presente trabalho consiste em aplicar a tecnologia *Automatic License Plate Recognition* (ALPR) por meio de um aplicativo Android visando automatizar o processo de fiscalização durante uma abordagem fiscal, garantindo uma melhor qualidade na prestação dos serviços. O sistema especialista denominado ***SMART COP***, por ser desenvolvido em Java, pode ser utilizado em *smartphone*, *tablets* e até mesmo em óculos inteligentes, visando facilitar o trabalho feito pelos agentes de trânsito, desonerando o tempo gasto com as centrais de operações da polícia, no momento em que estão aplicando as medidas cabíveis ao perceber uma infração ou um veículo irregular.

1.1 OBJETIVO GERAL

Aplicar a tecnologia ALPR por meio de um aplicativo Android para automatizar o processo de fiscalização durante uma abordagem policial.

1.2 OBJETIVOS ESPECÍFICOS

Para atingir o objetivo geral da pesquisa, foram delimitados os objetivos específicos:

a) levantar estatísticas sobre furtos de carros no Brasil;

b) compreender o funcionamento do método ALPR;

c) desenvolver um aplicativo em Android capaz de reconhecer placas veiculares por Meio de imagens aplicando técnicas de processamento de imagem;

d) Integrar o aplicativo com a API disponibilizada pelo Departamento Estadual de Trânsito (DETRAN) para validar os Dados capturados.

1.3 JUSTIFICATIVA

O crescimento socioeconômico junto ao crescimento desorganizado das cidades teve grandes impactos no trânsito. A saturação e o aumento no número de

veículos são alarmantes, onde acarreta o controle desses veículos em circulação aumentando a quantidade de infrações e furtos (QUEIROZ, 2015).

Conforme a Folha de São Paulo, o Brasil, em fevereiro de 2018, a cada um minuto um carro era furtado, totalizando 1440 carros roubados diariamente, o que significa aproximadamente meio milhão por ano (FOLHA, 2017).

Além do furto, existe um outro agravante direcionado ao aumento do número de veículos em circulação no Brasil. Dados divulgados pelo DETRAN em um período de 13 anos (de janeiro de 2000 até janeiro de 2013), afirmam que o crescimento de veículos chegou a 160%, tendo em vista que o crescimento da tecnologia não obteve o desenvolvimento necessário para acompanhar este aumento da frota veicular (SCARPATO et al., 2013).

Nos dias atuais, o reconhecimento automático do conteúdo de placas veiculares permite uma imensa quantidade de aplicações. Por exemplo, para os agentes de trânsito nas cidades possibilita identificar furtos e infrações em um tempo muito mais ágil, controlando de forma mais precisa condutores irregulares (Corneto et al., 2017).

No entendimento de Bretãs Pereira (1997) um sistema eficiente é aquele que facilita a interação com o usuário e traz vantagens para vários setores e segmentos, melhorando eficácia do atendimento e auxiliando na tomada de decisão. É notável que a mudança de procedimentos se faz necessário e se torna frequente nas organizações. A necessidade do acesso à informação em um tempo menor, se torna um fator decisivo para o futuro (ULRICH, 1998).

Uma solução para reconhecimento automático de placas se torna primordial para fiscalização, possibilitando ao agente de trânsito ou policial na identificação automática dos veículos, auxiliando na busca por irregularidades, furtos, controle de tráfego nas estradas e na tomada de decisão.

1.4 ESTRUTURA DO TRABALHO

No segundo capítulo é apresentado os conceitos básicos do processamento de imagem. Este capítulo apresenta como é possível a reprodução e a simulação de todo o espectro visível, com apenas as três cores primárias: vermelho, verde e azul.

O terceiro capítulo apresenta um estudo sobre o processamento de imagem, sistemas de cores primárias, modelo de cores e seu histograma. Este capítulo demonstra quais são as partes fundamentais da coloração que influenciam na captura e reconhecimento de caracteres.

No quarto capítulo são analisados os possíveis defeitos e imperfeições que devem ser tratadas durante a captura da imagem. Explica o estudo da morfologia matemática e seu objetivo.

O quinto capítulo descreve o método ALPR no qual incluí várias técnicas de processamento de imagem e visão computacional para localizar uma determinada placa de carro.

O sexto capítulo demonstra a API VISION, uma biblioteca de extrema importância para o desenvolvimento deste trabalho.

O sétimo capítulo descreve os dispositivos móveis, que atualmente são muito presentes não só no dia a dia das pessoas, mas também em diversas outras áreas auxiliando e melhorando a vida da sociedade.

No oitavo capítulo são apresentados alguns trabalhos correlatos com pesquisas semelhantes.

O capítulo nove descreve a metodologia empregada, a definição das amostras e dos testes, os materiais utilizados e o protótipo desenvolvido. Descreve como o protótipo foi desenvolvido, além de mostrar o aplicativo e quais os resultados que foram apresentados.

No capítulo dez é apresentam-se as conclusões finais.

2 CONCEITOS BÁSICOS DO PROCESSAMENTO DE IMAGENS

A representação visual de um objeto recebe o nome de imagem. Esta ação pode ser representada por meio de diversas técnicas, como por exemplo, uma fotografia, pintura ou até mesmo um vídeo. Uma imagem digital, por sua vez, é a representação da informação de um modo binário. De acordo com Ozbay, Ercebeli (2005), a partir da imagem original é possível criar uma imagem totalmente binarizada, conforme apresentado na figura 1:

Figura 1 – Representação de imagem binarizada

Fonte: Ozbay e Ercebeli (2005).

No entendimento de Gonzalez e Woods (2008) uma imagem pode ser definida como uma função (x, y), onde x e y são coordenadas num plano e f(x, y) é a intensidade (ou nível de cinza, em imagens monocromáticas) naquelas coordenadas. Quando os valores de x, y e da amplitude de f são valores discretos finitos, a imagem é dita como imagem digital.

Deste modo, uma imagem digital pode ser considerada uma matriz I com M linhas e N colunas, sendo x e y, as coordenadas da matriz, convencionalmente números inteiros variando de 0 ao tamanho do eixo da coordenada. Os elementos de uma imagem, nesse caso I(x, y) com $x < N$ e $y < M$, são denominados pixel e variam de 0, ausência total de cor (ou intensidade), e 255, nível máximo de intensidade (GONZALEZ; WOODS, 2008).

O ser humano visualiza e percebe comprimentos de ondas entre 380 nm cores violetas e 700 nm de cor vermelha, a soma das radiações desses dois valores tem um resultado da luz branca. Inúmeros comprimentos de ondas, exemplo, ondas

de rádios cósmicos, comprimento de ondas, assim como o infravermelho e o ultravioleta, na presença das bordas de comprimento de ondas que é possível enxergarem, não é possível visualizar pelo sistema visual humano (FRASER, 2005).

Conforme Tasi (2004), as cores existem na forma de energia, e um determinado comprimento da onda. No entanto, apenas em nossa mente a cor existe e identificamos somente depois de recebermos respostas no nosso sistema visual por meio do comprimento da onda. O sistema visual humano é repleto de sensores sensíveis a luz e a diferentes comprimentos de ondas que quando recebem um sinal elétrico no cérebro, os sinais são processados e o resultado é a sensação de luz e cores.

De acordo com Ogê, Ugo (1999) o mais importante a ser compreendido que o sistema visual humano é sensível aos comprimentos de ondas das cores, verde, azul e vermelho proveniente da luz. Ocasionando assim diferentes intensidades de luz verde, azul e vermelho possibilitando a visualização de várias cores. Este conjunto todo se faz a reprodução das cores, tornando possível a reprodução e a simulação de todo o espectro visível, com apenas as três cores primárias: vermelho, verde e azul, no processo aditivo, ou nas cores complementares: magenta, ciano e amarelo, no processo subtrativo.

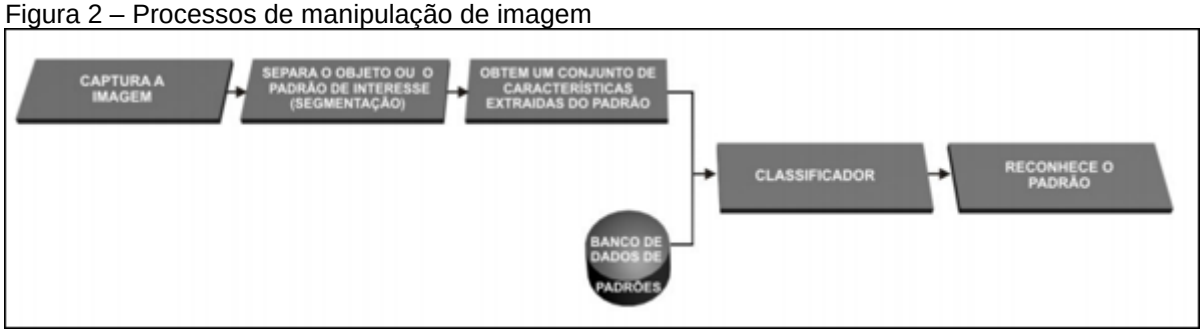
2.1 PROCESSAMENTO DE IMAGENS

A etapa de pré-processamento tem como objetivo a melhoria da qualidade da imagem vinda da aquisição, para este processo se utiliza das técnicas de atenuação de ruídos, correção de contraste e brilho, a citar a equalização de histograma. Ela é seguida pela etapa de segmentação a qual consiste no processo de separar a imagem em regiões de pixels similares (RUSSELL;NORVIG, 2010).

No processo de transformar a imagem para texto, é utilizado imagens com uma qualidade aceitável para que o processo de extração de caractere seja realizado com a maior precisão na identificação. Qualquer detalhe que interfira na captura da imagem pode influenciar no processo de reconhecimento (GONZALEZ; WOODS, 2009).

Para Gonzalez e Woods (2009) tendo uma imagem com qualidade, é possível realizar diversas operações, tais como, manipular os pixels das imagens, coletar características, segmentar objetos, entre outros objetivos.

Figura 2 – Processos de manipulação de imagem



Fonte: Do autor (2019).

Deste modo, o processamento digital de imagens engloba processos que tem como entrada e saída imagens e processos que extraem atributos de imagens. As subseções seguintes abordam alguns desses processos.

2.1.1 Aquisição e Captura da imagem

O processo de aquisição de imagem se inicia por meio de um sensor e um digitalizador. Pedrini, Schwartz (2008) comenta que o sensor transformará a informação óptica em um sinal elétrico e o digitalizador irá converter a imagem analógica em uma imagem digital.

De acordo com Gonzalez e Woods (2009), nos tipos de projetos que passam por esta etapa, podemos indicar: a escolha em qual tipo de sensor que será utilizado, as lentes, a iluminação, a velocidade da aquisição, a resolução e o número de tons de cinza que irão influenciar na imagem digitalizada. Nesta fase é reproduzida uma imagem digitalizada, onde a restauração e o realce da imagem condicionam uma melhora nas propriedades visuais das imagens.

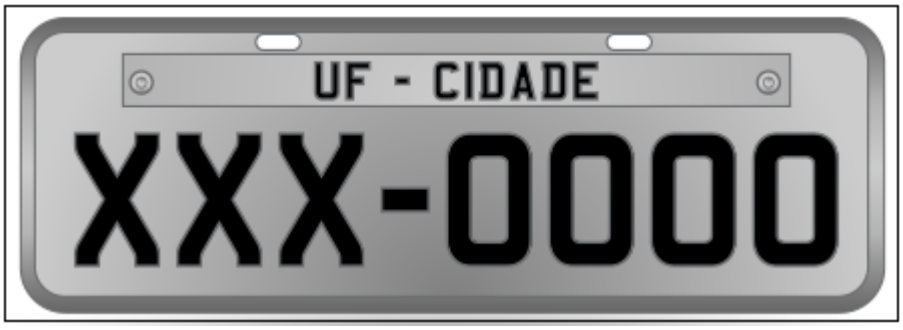
2.1.2 Segmentação da imagem

O processo de “segmentação de imagem” é considerado como uma das etapas mais importantes durante a manipulação de uma imagem digital. É neste momento que se segmenta uma imagem em objetos interessantes e perceptíveis.

Para Khoshafian e Baker (1996) o objetivo da etapa de segmentação é de separar as imagens em componentes individuais importantes, ou seja, os objetos de interesse que serão utilizados. Khoshafian e Baker (1996) complementa ainda que é uma etapa muito crucial e a mais difícil de se implementar.

Grande parte dos algoritmos de segmentação baseia-se nas propriedades fundamentais de valores de níveis de cinza (intensidade), que visam à detecção de contornos de objetos e similaridades na imagem (FORESTI, 2006; PEDRINI; SCHWARTZ, 2008), conforme pode ser visualizado na figura 2:

Figura 3 – Placa veicular representada em tons de cinza



Fonte: Do autor (2019).

Especificamente nas placas veiculares a segmentação pode ser dividida em duas fases, na primeira os algoritmos buscarão encontrar somente o valor da placa, e na segunda o algoritmo buscará trabalhar com a sub imagem, cujo objetivo é segmentar os dígitos separadamente.

2.1.3 Extração de Características

Conceito de extração de características, segundo Jain et al (2000), pode ser definido como um método que determina um subespaço apropriado de dimensionalidade m (conjunto contendo m características) a partir de um espaço de dimensionalidade d (a cena), sendo (m <= d). Para Jain et al (2000) a definição deste método “é um dos fatores mais importantes para a construção de um sistema de visão ou reconhecimento de padrões”.

Conforme Du (2013), as alternativas mais utilizadas para o reconhecimento de caracteres é a extração de características em conjunto com o uso de classificadores. O processo se fundamenta na extração de algumas características para auxiliar na identificação do caractere que está sendo tratado. Essas características são gravadas em um vetor chamado (vetor de características), com a finalidade de ser comparado com vetores de característica anteriormente armazenados para medir a sua equivalência.

2.1.4 Reconhecimento e Interpretação

No último processo é realizado o reconhecimento, processo de atribuição de um rótulo a um objeto fundamentado nas características encontradas a etapa de interpretação por sua vez resume-se em conceder significado para o conjunto de características reconhecido (DU et al., 2013).

A interpretação de imagens computadorizadas pode ser considerado um processo extremamente complexo. A grande quantidade de dados a serem processados e a falta de ferramentas de processamento torna a interpretação custosa para a máquina.

Para Gonzalez e Woods (2009), interpretação envolve a fixação de significado a um grupo de objetos reconhecidos, em outras palavras, dar significados a imagem.

3 SISTEMAS DE CORES

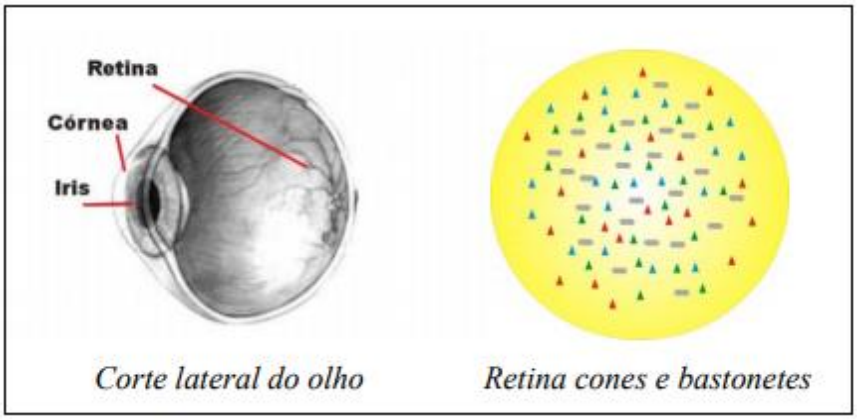
As cores estão tão presentes em nossa vida, em nosso dia a dia, que mal lhes damos importância. Na verdade, a cor, bem como sua percepção, faz parte de um conjunto quase indissociável que permeia várias esferas que vão da ciência à arte, do técnico ao estético.

Ao definirmos cor, é necessário primeiramente que seja lembrada sua relação direta com a luz. A cor, na verdade, é uma sensação ocasionada pela interação do olho e a luz (GUIMARÃES, 2000).

Nossos olhos possuem na retina dois tipos de sensores: os cones e os bastonetes; os cones permitem a percepção das cores e os bastonetes a percepção dos tons de cinza.

Nós herdamos de nossos antepassados a visão tricrômica, ou seja, vemos todas as cores baseadas em apenas três: o vermelho, o azul e o verde. Os bastonetes nos permitem “ver” à noite, ou seja, podemos perceber silhuetas com algum grau de precisão, sem, no entanto, notarmos os detalhes, conforme ilustra a figura 4.

Figura 4 – Sensores presentes no olho humano



Fonte: Do Autor (2019).

A luz visível faz parte de um conjunto de vibrações eletromagnéticas, das quais só uma porção é percebida por nós. Conforme aponta Goethe (1993) o olho deve sua existência à luz. De órgãos animais a ela indiferentes, a luz produz um órgão que se torna seu semelhante. Assim o olho se forma na luz e para a luz, a fim de que a luz interna venha ao encontro da luz externa.

Para Goethe (1993), a cor é natureza na forma de lei para o sentido da visão, e trata-se de um fenômeno elementar que nos permite comunicar com o universo que nos rodeia.

No próximo tópico abordaremos o sistema de coloração primária com seus respectivos modelos de cores RGB e HSV.

3.1 COLORAÇÃO PRIMÁRIA

A identificação de cores é um processo complexo, em uma abordagem mais simples a cor está relacionada nas características espectrais de radiação que chega até a retina (MOTA, 2017).

Segundo Mota (2017) nossa retina possui milhares de células que são sensíveis à radiação visível, denominada "cones". Geralmente estão juntos em três tipos, dependendo do tipo de radicação à qual são mais sensíveis. Os três tipos de cones são sensíveis á faixa de comprimentos de onda do espetro luminoso, sendo mais intenso aos picos situados a 445 nm (azul-violeta), 535 nm (verde) e 575 nm (laranja/vermelho).

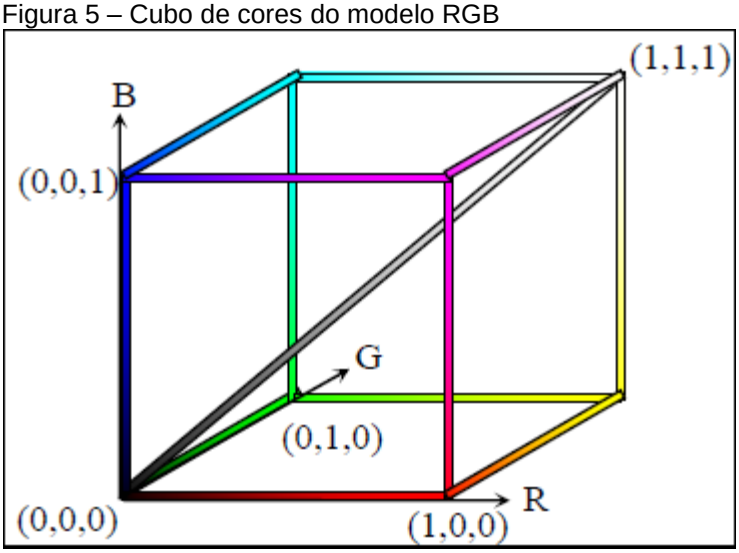
3.2 MODELOS DE CORES RGB

Segundo Gonzalez e Woods (2000), no modelo RGB cada cor é determinada na sua componente espectral primária vermelha (*Red*), verde (*Green*) e azul (*Blue*). Cada eixo varia entre 0 e 1 e é baseado no sistema cartesiano de coordenadas.

Para Lopes (2013) o modelo RGB “teve sua origem com base nos dispositivos gráficos visuais, tais como monitores e televisores que funcionam com o princípio de variação da intensidade das três cores primárias”

Na figura 5 o vértice demonstrado na diagonal realiza a ligação do preto ao branco e se refere aos tons de cinza demonstrados neste modelo. Outras cores são criadas a partir das cores primárias que são combinadas de diferentes maneiras.

Figura 5 – Cubo de cores do modelo RGB



Fonte: Castleman (1996).

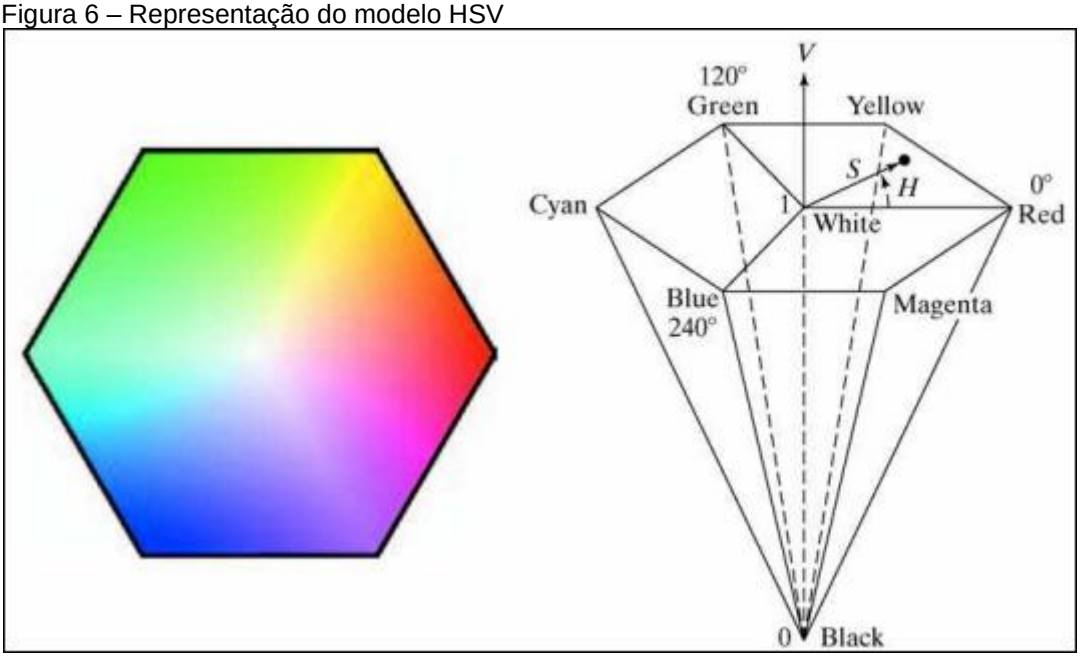
Os pontos ao longo da diagonal principal têm valores de cinza normalizados, a partir do preto na origem (0, 0, 0) na direção do branco (1, 1, 1).

3.3 MODELOS DE CORES HSV

O modelo *Hue, Saturation, Value* (HSV) é representado por componentes de tonalidade, brilho e saturação. O HSV é um hexágono, sua saturação é tratada a

distância, partindo do eixo central (0-100). A tonalidade vai da distância do ângulo tendo início o vermelho (360º) e o brilho irá variar de 0 a 1 iniciando na parte de baixo. A figura 6 representa detalhes do modelo HSV.

Figura 6 – Representação do modelo HSV



Fonte: Cámara (2016).

O modelo HSV é muito utilizado quando se faz necessário identificar cores que são muito parecidas com outras cores, pois o que determina a cor do modelo HSV é a tonalidade, diferente do modelo RGB onde o determinante das cores é (vermelho, verde e azul). No cubo RGB mesmo que utilize a distância

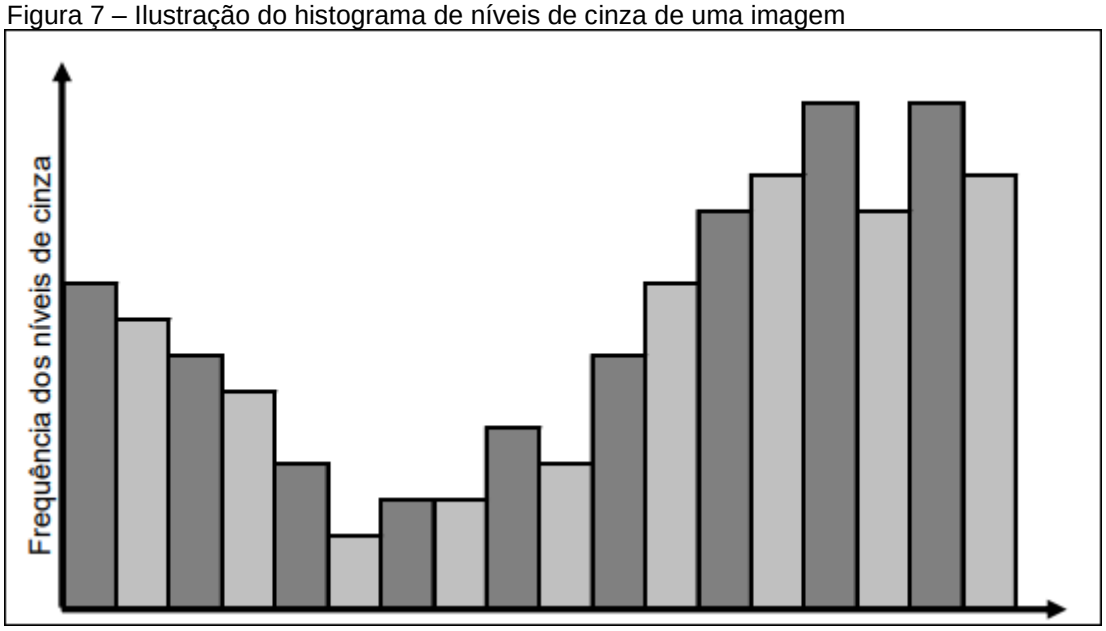
euclidiana, entre duas cores, o resultado final provável que não seja de cores parecidas (CARDANI, 2001).

3.4 HISTOGRAMA

Conforme Dalal e Triggs (2005), o histograma é a forma mais comum para se representar a distribuição de pixel a uma imagem. Quando avaliada pode-se observar de maneira instantânea as características da imagem, sendo que pela sua forma conseguimos inferir informações relevantes, por exemplo, a distribuição dos valores do pixel e a intensidade média.

No ano de 1994, Gome e Velho (1994) definia o histograma sendo uma forma de analisar e realçar a imagem, revelando a distribuição dos níveis de cinza. Conforme ilustrado na figura 7.

Figura 7 – Ilustração do histograma de níveis de cinza de uma imagem



Fonte: Gomes e Velho (1994).

Os elementos do histograma são calculados com a seguinte equação:

$$p_k(r_k) = \frac{n_k}{n} \quad (1.1)$$

Onde:

a) rk = nível de cinza;
b) nk = número de pixels na imagem cujo nível de cinza corresponde a rk
n = número total de pixels na imagem;
c) k = 0, 1, 2, ..., L-1, onde L é o número de valores de pixel da imagem digitalizada;
d) pk = resultado da função discreta representada por rk.

O conceito de histograma pode ser aplicado a imagens coloridas. Neste caso, a imagem precisa ter as cores RGB (já mencionadas anteriormente) e calcular o histograma de cada componente (MARQUES FILHO, 1999).

4 OPERAÇÕES MORFOLÓGICAS

Segundo Facon (2001), a base dessa matemática está formulada sobre o elemento estruturante e sobre as operações dilatação e erosão. O elemento estruturante é a matriz que interage com os objetos da imagem modificando sua aparência e sua forma. As características mais importantes desse elemento são a sua forma e seu tamanho. A morfologia matemática em sua busca verifica se o elemento estruturante está ou não está contido no objeto da imagem.

De acordo com Gonzalez e Woods (2000), para tratar os principais defeitos e imperfeições da segmentação de caracteres são utilizadas técnicas de morfologia matemática para que possa ser corrigido por meio da utilização de sequência de filtros morfológicos que analisam deforma quantitativos os pixels da imagem.

A morfologia matemática é o estudo da quantificação da forma e estrutura de pixels no caso das imagens. Onde o objetivo principal é mostrar de forma clara a estrutura dos objetos que se formar a partir dos pixels por meio dos conjuntos de transformação que os modelam (GONZALEZ; WOODS, 2000).

Esta etapa é realizada a partir dos operadores morfológicos. Os conjuntos das operações morfológicas são: Erosão, Dilatação, Abertura e Fechamento. Nas utilizações mais simples, os operadores morfológicos usam apenas a imagem s ser analisada e um elemento estruturantes. O elemento estruturante é criado por um determinado conjunto de pixels utilizado para procurar a imagem em no momento da aplicação do operador morfológico. Caso uma parte estruturante parecer com parte da estrutura da imagem, uma ação é realizada ou não (GONZALEZ; WOODS, 2000).

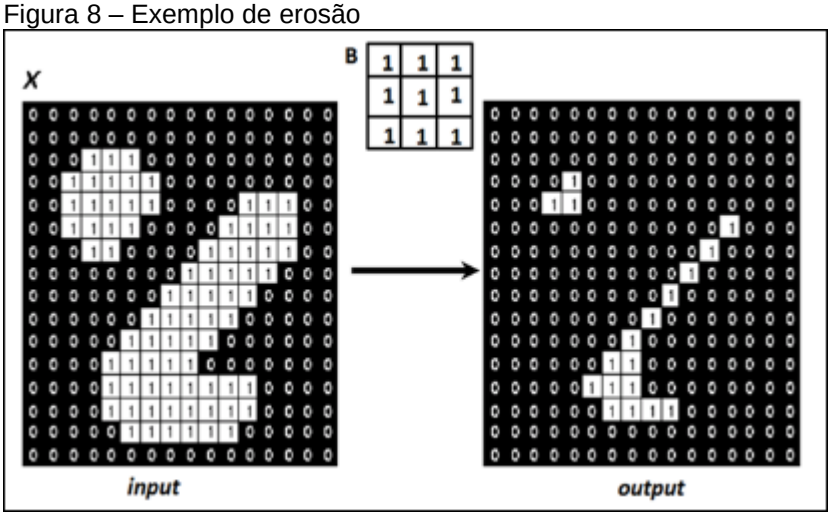
4.1 EROSÃO

A erosão morfológica é a transformação que combina dois conjuntos utilizando vetores de subtração. Nesta etapa, podemos interpretar como a intersecção nas translações do vetor original - e do elemento estruturante. As causas das erosões podem ser: Apagar elementos menores que o componente estruturante; aumentar espaços vazios (buracos) da imagem; diminuir partículas; dar permissão

para que se separem componentes que estão próximos (GONZALEZ; WOODS, 2000).

A figura 8 demonstra os resultados da erosão aplicada na imagem de exemplo com o mesmo elemento estruturador.

Figura 8 – Exemplo de erosão



Fonte: Gonzalez e Woods (2002).

4.2 BINARIZAÇÃO OU LIMIARIZAÇÃO

A binarização é um dos processos mais comuns para analisar imagens digitais, sendo um dos métodos mais simples na segmentação de imagens.

O processo inicia-se na aplicação de uma função de transformação de modo que a imagem tenha somente dois níveis de intensidade, melhor dizendo é uma operação não linear que irá converter a imagem para uma escala de cinza para uma imagem binaria, neste processo os dois níveis serão atribuídos a pixels que se encontram acima ou abaixo do valor limite estipulado nomeado de limiar (GONZALEZ; WOODS, 2006).

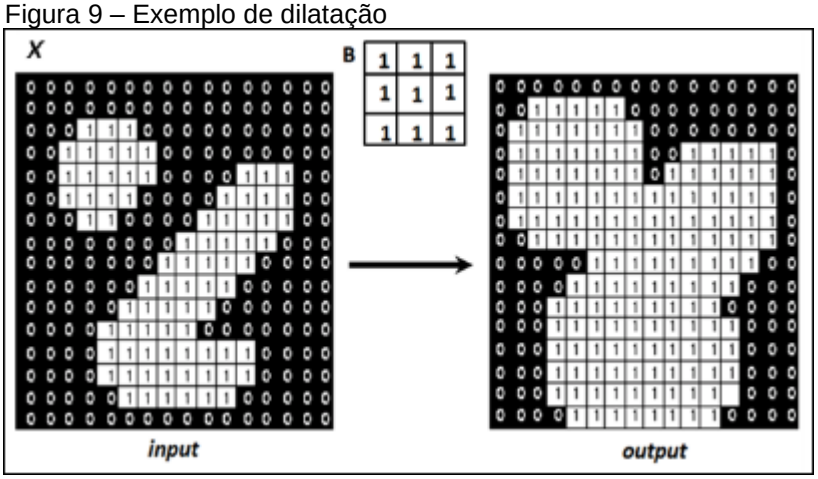
Conforme Jain e Lala (2013), o modelo de binarização mais conhecido tem o nome de binarização global. Caso o pixel da imagem em escala de cinza possuir o valor maior que o limiar, assume a esse pixel o valor de 255 (branco). Caso não possuir um valor maior, o pixel obtém o valor de 0 (preto). Enfim, os pixels que tiverem o valor 0 serão destinados para o fundo da imagem (*background*),

durante o tempo em que os objetos segmentados estarão no *foreground* (frente) da imagem.

4.3 DILATAÇÃO

Dilatação é a transformação morfológica que realiza a junção de dois conjuntos utilizando a adição vetorial. Esta etapa pode ser visualizada como a união de todas as translações da imagem original por meio do vetor do elemento estruturante. Sendo assim, os resultados da dilatação podem ser: aumentar (engordar) objetos; completar "buracos" da imagem; interligar pixels que estejam próximos. A dilatação é associativa e comutativa (JAIN; LALA, 2013).

Figura 9 – Exemplo de dilatação



Fonte: Gonzalez e Woods (2002).

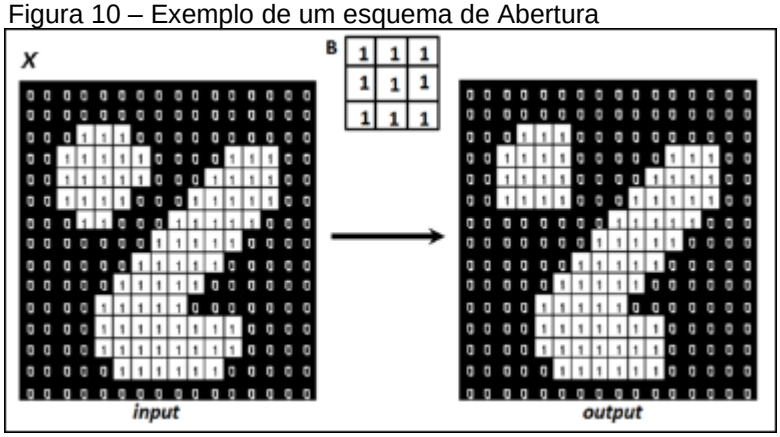
4.4 ABERTURA

Para Gonzalez e Woods (2006), abertura trata-se de uma operação de erosão seguida de uma dilatação fazendo o uso do mesmo elemento estruturante. A abertura cria uma suavidade no contorno de uma imagem, quebra estreitos e faz a eliminação de proeminências finas. Também utilizada para remover ruídos da imagem. Sendo assim, a abertura causa o efeito de: eliminar partículas, separar componentes.

A figura 10 demonstra um exemplo da operação de abertura.

Aqui, o elemento estruturante é também retangular com tamanho 3x3 e origem no meio.

Figura 10 – Exemplo de um esquema de Abertura



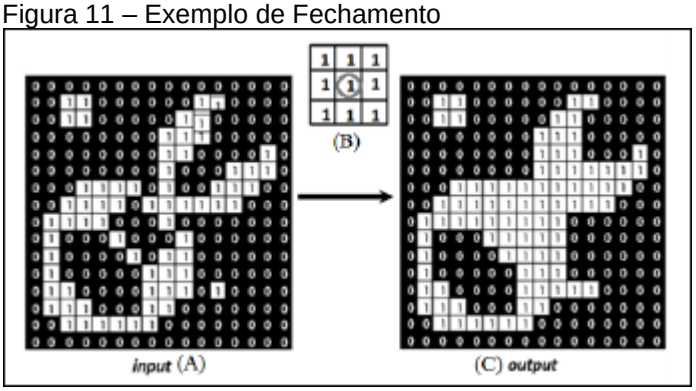
Fonte: Gonzalez e Woods (2002).

4.5 FECHAMENTO

Conforme Gonzalez e Woods (2006), fechamento fecha pequenos buracos e conecta espaços de um componente e é definido com uma operação morfológica onde na abertura é seguida de uma erosão.

A figura 11 ilustra um exemplo da operação de fechamento com um elemento estruturante de tamanho 3x3.

Figura 11 – Exemplo de Fechamento



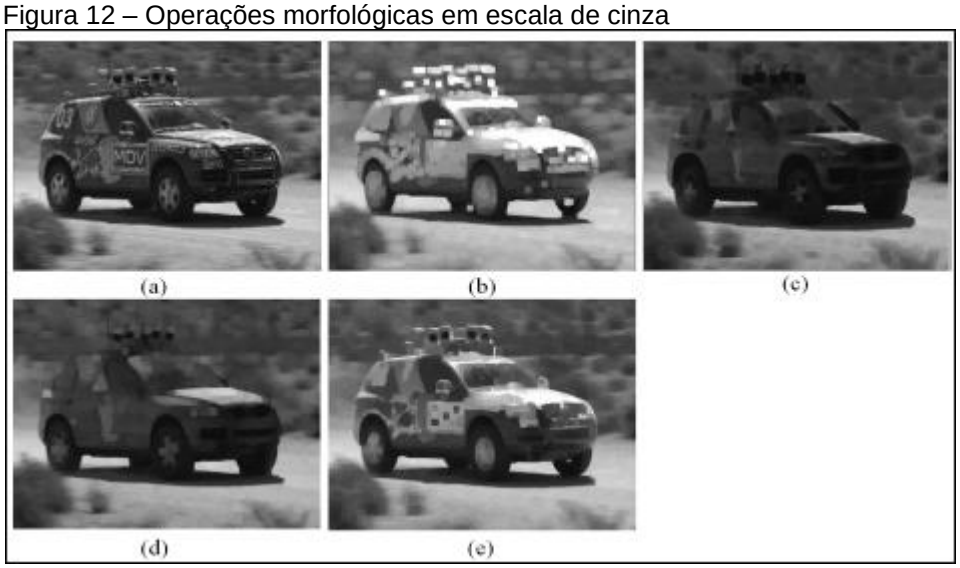
Fonte: Gonzalez e Woods (2002).

4.6 SEGMENTAÇÃO

Segundo Britto (2001), o objetivo da segmentação é identificar os segmentos significativos na imagem, identificados por meio de forma, geometria, textura, topologia, cor ou brilho. Dentro de diversas técnicas que existem para este fim, podemos citar a segmentação por região binarização e multi-binarização, textura, contorno, cor, redes neurais artificiais, modelos de contorno ativos, algoritmos genéticos, modelos de Markov, morfologia matemática, baseando-se na teoria dos conjuntos nebulosos e até mesmo híbridas.

Na figura 12 é demonstrado o resultado dessas operações aplicadas a uma imagem em tons de cinza.

Figura 12 – Operações morfológicas em escala de cinza



Fonte: Kaehler e Bradsk (2016).

com um elemento estruturante retangular de tamanho 3x3. (a) Imagem original. (b) Dilatação. (c) Erosão. (d) Abertura. (e) Fechamento.

5 INTRODUÇÃO AO MÉTODO ALPR

Os sistemas ALPR geralmente incluem várias técnicas de processamento de imagem e visão computacional primeiramente localizar uma placa de carro em uma cena, segmentar caracteres na placa e executar o reconhecimento óptico de caracteres para determinar a sequência de caracteres (PAUL et al, 2015, tradução nossa).

Além disso, é frequentemente necessário determinar a jurisdição, como o estado e o tipo de placa. As tecnologias ALPR podem ser utilizadas, por exemplo, em pedágio eletrônico, aplicação de fotos, estacionamento e outras aplicações de transporte onde a identidade do veículo é necessária. Os requisitos de desempenho para os sistemas ALPR são muito altos e estão sempre aumentando neste espaço competitivo de mercado. Os erros de desempenho podem ser classificados em modos de falha comuns. Um modo de falha predominante é chamado de "caracteres confusos". Os caracteres são confundidos entre si devido ao ruído da imagem. Por exemplo, *B* é frequentemente confundido com *8* e *D* é frequentemente confundido com *0* na presença de imagem tremida. Em outro modo de falha comum, o quadro da placa cobre parcialmente a placa, obscurecendo os traços de caracteres, como fazer um *T* parecer um *1* e um *E* parecer um *F*. Outros modos de falha comuns e dados de aumento desejados também são conhecidos *a priori* (PAUL et al, 2015, tradução nossa).

5.1 DETECÇÃO DE PLACAS

De acordo com Du et al. (2013) no processo de detecção da placa é uma das fases mais importante dentro no método ALPR, já que influência diretamente na acurácia do sistema, são inúmeros algoritmos proposto para segmentar as placas.

Conforme trabalho realizado por (Neto et al., 2015) onde utilizam placas Brasileiras como base para encontrar qualquer tipo de retângulos presentes em uma imagem e posteriormente segmentar apenas a região de interesse que contém nas placas dos veículos. No primeiro processo são convertidas as imagens em uma escala de cinza, em seguida é utilizado o operador Canny (CANNY, 1986) para detectar todas as bordas existentes nas imagens.

Em seguida por meio da transformada de Houng (DUDA; HART, 1972) que realiza a detecção de formas geométricas em imagens digitais são identificadas linhas verticais e horizontais para representar as bordas detectadas, assim formam-se retângulos que por fim serão filtrados, levanto em conta os aspectos da placa, tais como: o rádio, o *aspect ratio* e a área, que por fim poderá segmentar a placa (NETO et al., 2015).

5.2 RECONHECIMENTO OPTICO DE CARACTERES

Os sistemas de reconhecimento automático de placas (ALPR) são geralmente compostos por câmeras, iluminadores e gatilhos de veículos que capturam imagens de veículos, sistemas de processamento de imagens e visão computacional que executam processos que localizam a placa em uma cena. Segmenta os caracteres dentro da placa do carro e realiza o *Optical Character Recognition* (OCR) para determinar a sequência de caracteres na placa do carro. Além disso, a jurisdição do estado, o tipo de placa e outras informações geralmente precisam ser determinados. Os sistemas ALPR são amplamente utilizados em sistemas de transporte, como sistemas de pedágio eletrônico e sistemas de aplicação de fotos (aplicação de velocidade ou aplicação de luz vermelha), onde a identidade do veículo é necessária (PAUL et al, 2015, tradução nossa).

5.2.1 Etapas do OCR

Estas são as etapas de pré-processamento frequentemente executadas no OCR:

a) **binarização** - geralmente apresentada com uma imagem em escala de cinza, a binarização é simplesmente uma questão de escolher um valor limite;
b) **operadores morfológicos** - remover manchas e orifícios isolados nos caracteres, pode usar o operador majoritário;
c) **segmentação** - verifique a conectividade de formas, rótulos e isolados. Dificuldades com caracteres que não estão conectados, por exemplo. a letra i, um ponto-e-vírgula ou dois pontos (; ou :).

Dentre as etapas do pré-processamento, a segmentação é a mais importante, permitindo que o reconhecedor extraia recursos de cada caractere individual. No caso mais complicado de texto manuscrito, o problema de segmentação se torna muito mais difícil, pois as letras tendem a ser conectados um ao outro. A principal função de um sistema de reconhecimento de padrões é decisões sobre a participação na classe dos padrões com os quais ela é confrontada. No contexto de um sistema de OCR, o reconhecedor é confrontado com padrões de recursos de sequência que deve determinar as classes de caracteres (PAUL et al, 2015, tradução nossa).

6 API VISION

A API Vision do Google *Cloud* oferece modelos de aprendizado de máquina pré-treinados por meio das APIs REST e RPC. É possível atribuir marcadores às imagens e classificá-los em categorias predefinidas. Identificar objetos e faces, leitura de textos impressos e manuscritos (GOOGLE, 2017, tradução nossa).

a) identificar objetos automaticamente: Facilidade para identificar e classificar vários objetos, incluindo a localização de cada objeto na imagem. A localização do objeto identifica vários objetos em uma imagem e fornece um *LocalizedObjectAnnotation* para cada objeto na imagem. Cada um *LocalizedObjectAnnotation* identifica informações sobre o objeto, a posição do objeto e limites retangulares para a região da imagem que contém o objeto. A localização de objetos identifica objetos importantes e menos importantes em uma imagem (GOOGLE, 2017, tradução nossa);
b) criar e implantar modelos: o AutoML Vision Edge é utilizado para criar e implantar modelos rápidos com boa precisão para classificar imagens ou identificar objetos na borda e acionar ações em tempo real com base em dados locais. O AutoML Vision Edge suporta uma variedade de dispositivos de borda em que os recursos são limitados e a latência é crítica (GOOGLE, 2017, tradução nossa);
c) compreender o texto e agir de acordo com ele: a API Vision usa o OCR para detectar texto em imagens em mais de 50 idiomas e vários tipos de arquivo. Também faz parte da IA de compreensão de documentos, que permite

processar documentos rapidamente e automatizar fluxos de trabalho (GOOGLE, 2017, tradução nossa);

d) identificar conteúdo explícito: a API Vision pode revisar suas imagens usando a pesquisa segura e estimar a probabilidade de qualquer imagem incluir conteúdo adulto, violência entre outros (GOOGLE, 2017, tradução nossa);
e) utilizar e serviço de rotulagem de dados: se você tem imagens para o AutoML Vision que ainda não foram identificadas, o Google tem uma equipe de pessoas que pode ajudá-lo a anotar imagens, vídeos e texto para obter dados de tratamento de imagens (GOOGLE, 2017, tradução nossa).

7. SMARTPHONE

Segundo Sarwar e Soomro (2013, tradução nossa), os smartphones são dispositivos móveis que oferecem diversos serviços integrados como facilidade de comunicação sem fio, mensagens, aplicativos de gerenciamento, além de possibilitar a exibição e gravação de fotos e vídeos, jogar videogames, navegar na internet, ouvir músicas, receber e-mail e entre diversos outros recursos que estão presente neste dispositivo.

O primeiro smartphone foi lançado de fato no ano de 1993 (o “The Simon” da IBM), a diferença dos atuais para os primeiros smartphones que surgiram, é que inicialmente o objeto de uso eram apenas empresariais. (SARWAR; SOOMRO, 2013, tradução nossa). Mesmo atualmente, após se passarem vários anos desde a popularização dos smartphones, os aplicativos continuam crescendo a um nível muito rápido em escala global (BANERJEE et al., 2018, tradução nossa).

Um Sistema Operacional (SO) para smartphone muito conhecido é o Android, lançado pelo Google em 2008, de código aberto baseado no Linux. Em 2013, relatórios afirmaram que o SO Android possuía 81.3% de todo o mercado de smartphone, enquanto 13.4% ficava com a Apple, 4.1% com a Microsoft e 1% com a BlackBerry. O Android com o passar dos anos se expandiu ainda mais e não ficou apenas nos smartphones, atualmente é possível encontrar o Android também em tablets, e-readers, robôs, geladeiras, e vários outros dispositivos (DEITEL et al., 2015).

Segundo Deitel et al. (2018), a linguagem de programação oficial do Android era o Java por tratar-se de uma das linguagens de programação mais utilizada no mundo, mas em maio de 2018, segundo Banerjee et al. (2018, tradução

nossa), a Google anunciou que a linguagem oficial do Android passou a ser o Kotlin e não mais o Java.

O SO utilizado nos smartphones da Apple é o IOS, lançado em 2007 e que tinha como objetivo ser utilizado no iPhone e o iPod Touch, mas foi estendido para outros dispositivos, como iPad e Apple TV. Enquanto o Android possui uma política aberta, o IOS possui uma política fechada e não é open-source, além de somente permitir instalação de aplicativos da própria loja de aplicativos da Apple. (APPLE, 2019, tradução nossa).

8 TRABALHOS CORRELATOS

O desenvolvimento da pesquisa relacional dos trabalhos correlatos amplifica o conhecimento na área de pesquisa que será desenvolvida, engrandece o conhecimento e auxilia para trazer novos resultados. Neste capítulo são apresentados os trabalhos correlatos que foram desenvolvidos e que contribuíram nos estudos.

8.1 APLICATIVO PARA AUXÍLIO NA EMISSÃO DOS AUTOS DE INFRAÇÕES DE TRÂNSITO NO MUNICÍPIO DE BLUMENAU

O trabalho de conclusão de curso para o grau de Bacharel em Ciências da computação, da Fundação Universidade Regional de Blumenau (FURB) apresenta uma ferramenta desenvolvida para PDA ou *hand-helds*, que auxilia o preenchimento do auto de infração. O aplicativo foi desenvolvido para ser utilizado de substituto do papel e caneta, que atualmente são utilizados pelos agentes de trânsito, quando necessário realizar um auto de infração. Foi abordado a os problemas que surgem após o auto de infração, o aplicativo desenvolvido retira esse processo da necessidade de digitalização e ajuda a reduzir quantidade de cancelamento de notificações (PASTA, 2003).

Foi levantada a grande deficiência com relação às linguagens de programação para este tipo de computadores de mão, levantando a grande limitação de memórias disponibilizada. Após os testes realizados o autor aponta certas

dificuldades na ferramenta de programação a qual o aplicativo foi desenvolvido, a Genexus e sugere uma melhoria na impressão do auto de infração (PASTA, 2003).

8.2 RECONHECIMENTO AUTOMÁTICO DE PLACAS AUTOMOBILÍSTICAS

Este trabalho foi desenvolvido como intuito de obter o título de Engenheiro Eletrônico e de Computação pela Universidade Federal do Rio de Janeiro (UFRJ), no qual descreve a forma de realizar o Reconhecimento Automático de Placas Automobilísticas (RAPA) termo é mais conhecido em inglês: *Automatic License Plate Recognition* (ALPR) em vídeos. Visando a segurança em ambientes públicos ou privados controlando a entrada e saída de veículos em prédios, estacionamentos etc. Utilizando classificadores haar em cascata na detecção das placas veiculares. O trabalho visa realizar o reconhecimento das placas automobilísticas em tempo real gravadas em vídeos produzidos por câmeras de segurança. Foi utilizado a *Open Source Computer Vision Library* (*OpenCV*) para o aprimoramento e recorte dos caracteres presentes nas placas juntamente com o *Tesseract* um programa de código aberto utilizado na aplicação de Reconhecimento Óptico de Caracteres (PAES, 2017).

Nos resultados obtidos após os testes do aplicativo o autor conclui que obteve resultados satisfatórios na detecção e extração das placas no período diurno, mas ressalta que para o período noturno sugere aplicar medidas para aumentar o percentual de acerto e mesmo com a baixa resolução técnica de classificadores Haar se mostrou muito eficiente. Indicando também que câmeras com alta qualidade poderiam aumentar nesse percentual (PAES, 2017).

8.3 RECONHECIMENTO AUTOMÁTICO DE PLACAS VEICULARES BRASILEIRAS EM AMBIENTES NÃO CONTROLADOS

Este trabalho foi desenvolvido a fim de conceber a pós-graduação no curso engenharia de automação e sistemas pela Universidade Federal de Santa Catarina (UFSC), Neste trabalho é demonstrado um método para o reconhecimento de placas veiculares Brasileiras em ambientes não controlados, neste caso rodovias da cidade de São Paulo, tendo como base um banco de imagens fornecido pela empresa Brasileira Brascontrol (SALAZAR, 2017).

De acordo com Salazar (2017), também foram investigadas diferentes propostas que sanam o problema do reconhecimento de placas veiculares tanto em ambientes controlados, como em ambientes não controlados. De modo que a pesquisa realizada apontou o desenvolvimento de estratégias para solucionar o problema do ALPR em ambientes não controlados, neste caso utilizado as rodovias da cidade de São Paulo. O autor revisou as técnicas mais utilizadas para identificar placas veiculares e partir do esquema de pré-processamento da imagem de entrada proposto no trabalho foi possível minimizar os efeitos de sombras, baixo contraste e iluminação não uniforme nas placas. Uma das limitações encontradas no trabalho desenvolvido foi o tempo de processamento total se comparado com alguns dos trabalhos encontrados na literatura (SALAZAR, 2017).

9 SMART COP: APLICATIVO PARA LEITURA E VALIDAÇÃO DE PLACAS VEICULARES UTILIZANDO A TECNOLOGIA ALPR APLICADA NA FISCALIZAÇÃO EM TEMPO REAL DE VEÍCULOS DURANTE UMA ABORDAGEM POLICIAL

Após o conhecimento adquirido durante o tempo de levantamento bibliográfico, o processo metodológico da pesquisa foi dividido em algumas etapas: levantamento de requisitos, modelagem de dados, escolha das ferramentas de desenvolvimento, implementação da aplicação *mobile*, testes, e por fim, a análise dos resultados obtidos por meio da aplicação *mobile*.

Para a criação do protótipo de aplicação dando ênfase na captura de placas veiculares, como foi apresentado nos capítulos anteriores, utilizou-se a ferramenta IDE Android Studio da empresa Google e seu conjunto de bibliotecas OCR (denominada *google cloud vision*) juntamente com a bibliotecas RETROFIT para comunicação de dados entre o dispositivo móvel e o servidor de API instalado na nuvem. O servidor de API foi desenvolvido na linguagem C# criada pela empresa Microsoft e hospedado no portal KingHost (empresa Brasileira prestadora destes serviços).

O protótipo oferece para os policiais e agentes de trânsito um meio de obter as informações relevantes, tais como PLACA e ESTADO do veículo, utilizando a câmera do próprio celular. Ao fazer o reconhecimento via dispositivo, é possível obter informações relevantes sobre o carro e diversos indicativos relacionado a segurança conforme aponta a legislação de trânsito brasileira.

9.1 METODOLOGIA

Para atingir os objetivos deste trabalho, fez-se necessário seguir algumas etapas, e determinar o que seria essencial em cada uma delas. Após a aprovação da proposta, foi iniciado o desenvolvimento do projeto de pesquisa, constituído do levantamento bibliográfico e elaboração do referencial teórico. Para isso, fez-se necessário utilizar dos recursos disponibilizados pela instituição UNESC, livros e recursos do Orientador do Projeto e/ou adquiridos pelo acadêmico.

Por meio do levantamento bibliográfico, aprimorar o tema proposto. Entre eles, pode-se citar a utilização da biblioteca Google *Cloud Vision*. Observou-se que

as bibliotecas, existentes no mercado, utilizadas para reconhecimento de placas brasileiras não funcionavam por completo e eram insuficientes. Após escolher a utilização da *API Vision*, deu-se início ao desenvolvimento do protótipo, o qual tem por objetivo capturar placas veiculares para automatizar o processo durante uma abordagem policial ou fiscal.

O projeto de desenvolvimento foi dividido em quatro etapas. A primeira etapa foi a busca pelo *Hardware* ideal. O protótipo precisaria funcionar com um *Hardware* de câmera suficiente para conseguir trabalhar com reconhecimento ótico de caracteres. Nesta etapa descobriu-se que deveríamos utilizar a versão do Android ice cream sandwich ou superior, pois nestes modelos o *Hardware* já é mais bem otimizado.

A segunda etapa foi a construção da *interface* do aplicativo, utilizando de diversos emuladores presentes no Android Studio para verificar a compatibilidade de layout em relação aos tipos de sistema operacionais. Como já é de conhecimento, o Android possuí diversas versões e tamanho de tela que podem desconfigurar o designer feito pelo programador.

A terceira etapa esteve relacionada a utilização da *API Google Cloud Vision* para reconhecimento das placas veiculares. Nesta etapa alguns problemas foram apresentados pelas bibliotecas e fui obrigado a trata-los de maneira individual, como por exemplo a identificação de algumas letras de forma equivocada pela biblioteca.

A quarta e última etapa foi a construção do portal Web, responsável por fazer as validações das placas veiculares, apontando o status de IPVA Atrasado, IPVA Pendente e Carro Roubado.

9.2 DEFINIÇÃO DAS FERRAMENTAS E RECURSOS

O primeiro passo para dar início ao desenvolvimento do projeto foi a seleção de recursos a serem utilizados. Foi necessário para o desenvolvimento do protótipo, na parte de Hardware, um aparelho smartphone com 8GB de Armazenamento, 512 de RAM e resolução da câmera traseira superior a 4MP. Em relação a recursos de software, necessitou-se usar a versão do Sistema Operacional Ice cream sandwich ou superior.

No que diz respeito a programação do dispositivo móvel, foi definido a linguagem nativa Java, linguagem já apresentada e utilizada na instituição, por meio da ferramenta IDE Android Studio 3.5.3.

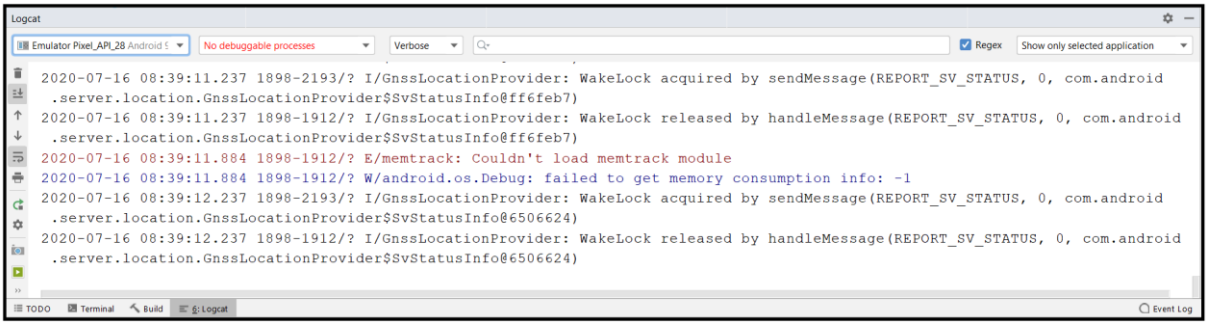
Para leitura das placas veiculares optou-se pela utilização da biblioteca *Google Cloud Vision*. A biblioteca foi selecionada, pois comparado aos demais meios de captura, este modelo conseguiu realizar diversas leituras em pontos diferentes de cena além de reconhecer o objeto “placa” de maneira mais eficiente.

A comunicação feita entre o aplicativo e o portal Web foi feita utilizando a biblioteca gratuita RETROFIT e o portal foi totalmente desenvolvido na linguagem C# da empresa Microsoft já de conhecimento do autor.

O LogCat é uma ferramenta que também vem junto ao Android Studio, que é disponibilizada pelo SDK e tem a funcionalidade de apresentar todos os logs da aplicação. Esta ferramenta nos possibilitou o entendimento de alguns processos errôneos no decorrer do desenvolvimento do aplicativo.

Para finalizar, utilizou-se o *software Astah Community*, versão temporária, sendo possível criar os diagramas utilizados no decorrer deste trabalho.

Figura 13 – Ferramenta de depuração Logcat.



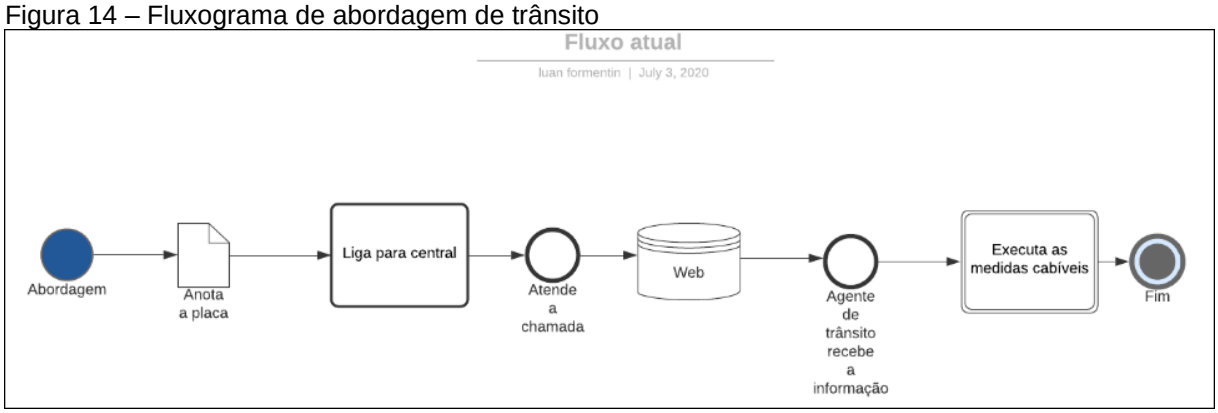
Fonte: Do Autor (2020).

9.3 FUNCIONAMENTO DA APLICAÇÃO

Após a definição dos recursos tecnológicos necessário, foi necessário realizar um estudo do atual ambiente de trabalho dos policiais. Isto é, conhecer como é praticado uma abordagem policial nos dias de hoje. Por trabalhar e servir o exército por alguns anos, foi possível realizar este tipo de estudo. Para exemplificar isso foi desenvolvido um fluxograma de processo, pois segundo Silveira (2018), o fluxograma é uma forma de representar os processos de uma empresa ou

organização por meio do uso de símbolos gráficos, cujo objetivo é descrever passo-a-passo a natureza e o fluxo de processos.

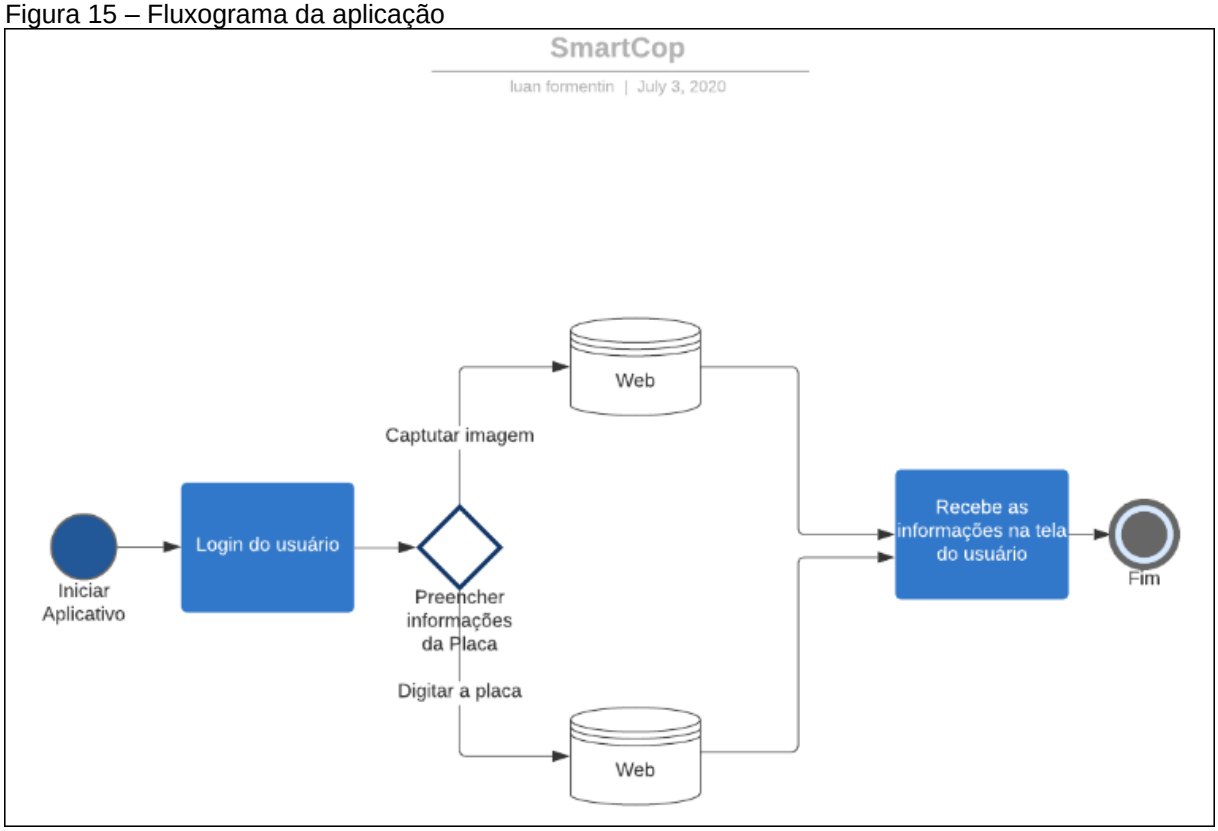
Figura 14 – Fluxograma de abordagem de trânsito



Fonte: Do Autor (2020).

Com o estudo e realização do fluxograma de processo, foi possível descobrir os possíveis problemas durante a abordagem e então definir uma nova tecnologia para suprir essa necessidade, conforme aponta a figura 15.

Figura 15 – Fluxograma da aplicação



Fonte: Do Autor (2020).

Observou-se com a figura 15, que o processo da abordagem policial passou a utilizar o dispositivo móvel na detecção da placa do veículo, o que retornou um resultado mais eficiente e rápido em relação ao método tradicional de captura.

9.4 DESENVOLVIMENTO DO APLICATIVO SMART COP

Para o desenvolvimento da aplicação, foi imaginado uma estrutura que facilitasse ao máximo o uso por qualquer policial ou agente de trânsito. O sistema possui um controle com login, para identificar o agente de transmissão dos dados. A autenticação ocorre por meio do cadastro do e-mail e uma senha do agente diretamente no portal Web. A figura 16 mostra a tela de login para realizar a validação do acesso ao aplicativo:

Figura 16 – Tela de Login

Fonte: Do Autor (2020).

O funcionamento principal do aplicativo ocorre fazendo uso da câmera. Como pode ser visualizado na tela inicial do aplicativo figura 17, a captura das placas se inicia pressionando o botão **Capturar placa**.

Figura 17 – Tela inicial

Fonte: Do Autor (2020).

Neste momento, antes de possibilitar que o aplicativo use a câmera, deve-se considerar algumas questões de segurança onde o usuário precisa consentir o uso do Hardware, conforme normas adotadas pela Google:

a) exigência de câmera: o uso de uma câmera é tão importante para o aplicativo que você não quer a instalação dele em um dispositivo que não tenha esse recurso. Nesse caso, declare a exigência de câmera no seu manifesto;

b) solicitar permissões do app: cada aplicativo Android é executado em um sandBox com acesso limitado. Se um aplicativo usar recursos ou informações fora do próprio sandBox, ele precisará a permissão apropriada.

Conforme mencionado, as duas etapas foram implementadas para evitar erros de utilização do aplicativo. As exigências de câmera foram colocadas no manifesto da aplicação, conforme aponta a figura 18.

Figura 18 – Exigências da câmera

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.google.android.gms.samples.vision.ocrreader"
    android:installLocation="auto" >

    <uses-feature android:name="android.hardware.camera" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Fonte: Do Autor (2020).

E a solicitação de permissões do app também foram implementadas, conforme demonstra a figura 19.

Figura 19 – Solicitação de permissões

```
/**
 * Método responsável por solicitar acesso a camera do dispositivo.
 */
private void requestCameraPermission() {
    Log.w(TAG,  msg: "Requisitando acesso a camera");

    final String[] permissions = new String[]{Manifest.permission.CAMERA};

    if (!ActivityCompat.shouldShowRequestPermissionRationale( activity: this,
            Manifest.permission.CAMERA)) {
        ActivityCompat.requestPermissions( activity: this, permissions, RC_HANDLE_CAMERA_PERM);
        return;
    }

    final Activity thisActivity = this;

    View.OnClickListener listener = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            ActivityCompat.requestPermissions(thisActivity, permissions,
                    RC_HANDLE_CAMERA_PERM);
        }
    };

    Snackbar.make(mGraphicOverlay, R.string.permission_camera_rationale,
            Snackbar.LENGTH_INDEFINITE)
            .setAction("OK", listener)
            .show();
}
```

Fonte: Do Autor (2020).

Além destas implementações, foi necessário criar o nosso próprio objeto câmera, isso para que fosse possível obter o máximo de desempenho na captura de imagens e especificamente das placas do veículo. Para criar este objeto, algumas etapas precisaram ser desenvolvidas:

a) **detectar e acessar a câmera:** criou-se um código para verificar a existência de câmeras no Smartphone e solicitar acima (apontado anteriormente);
b) **criar uma classe de visualização:** criou-se uma classe de visualização da câmera que estenda a *SurfaceView* e implemente a interface *SurfaceHolder*. Essa classe exibe as imagens em tempo real da câmera;
c) **criar um layout de visualização:** depois de ter a classe de visualização da câmera, criou-se o layout que incorporou a visualização e os controles de interface do usuário;
d) **configurar *listeners* para a captura:** foi necessário criar ouvintes de captura para as placas veiculares;
e) **capturar e salvar os dados:** criou-se rotinas de captura e salvamento das placas capturadas pelo aplicativo;
f) **liberar o uso da câmera:** após toda coleta de dados, foi necessário criar uma rotina para liberar o uso da câmera para não bloquear outros aplicativos.

Conforme apontado no site da Google *Developer*, acessado em 2020, o Hardware da câmera é um recurso compartilhado que precisa ser gerenciado cuidadosamente para que seu aplicativo não entre em conflito com outros aplicativos que também possam querer utilizá-lo.

Após implementar todos os recursos de câmeras, o aplicativo abrirá a tela denominada *SurfaceView* e iniciará o processo de captura, a figura 20 demonstra este processamento.

Figura 20 – Tela de captura

Fonte: Do Autor (2020).

Feito o reconhecimento de uma placa, o sistema automaticamente irá voltar para a tela inicial, preenchendo os campos ***Placa*** e ***Estado*** dando a possibilidade ao agente conferir os dados reconhecidos pela câmera do celular. Esse passo foi importante, pois em algumas situações, a placa pode estar suja ou com ambiente escuro e não reconhecer conforme deveria. Dessa forma existe a

possibilidade manual de correção. Tratamentos foram necessários para que o OCR em conjunto com a API Vision capturasse as placas de uma forma mais precisa, para isso criou-se máscaras de utilizando expressão regular (regex) conforme pode ser visualizado na figura 21 apresentada abaixo.

Figura 21 – Expressão regular para tratamento das placas

```
250     /*
251     Método responsável por adicionar a placa do veículo.
252      */
253     private boolean PlacaAdicionar() {
254
255         boolean isAdd = false;
256
257          /*
258         Busca a placa e faz uso de expressão regular.
259          */
260         Pattern pattern = Pattern.compile("[a-zA-Z]{3,3}-\\d{4,4}");
261         Matcher matcher = pattern.matcher(item.getValue().replace( target: " ", replacement: "").trim());
262
263         boolean b = false;
264         while (b = matcher.find()) {
265             if (b) {
266                 isAdd = true;
267                 placa = matcher.group();
268                 listaPlacas.add(placa);
269             }
270         }
271
272         return isAdd;
273     }
```

Fonte: Do Autor (2020).

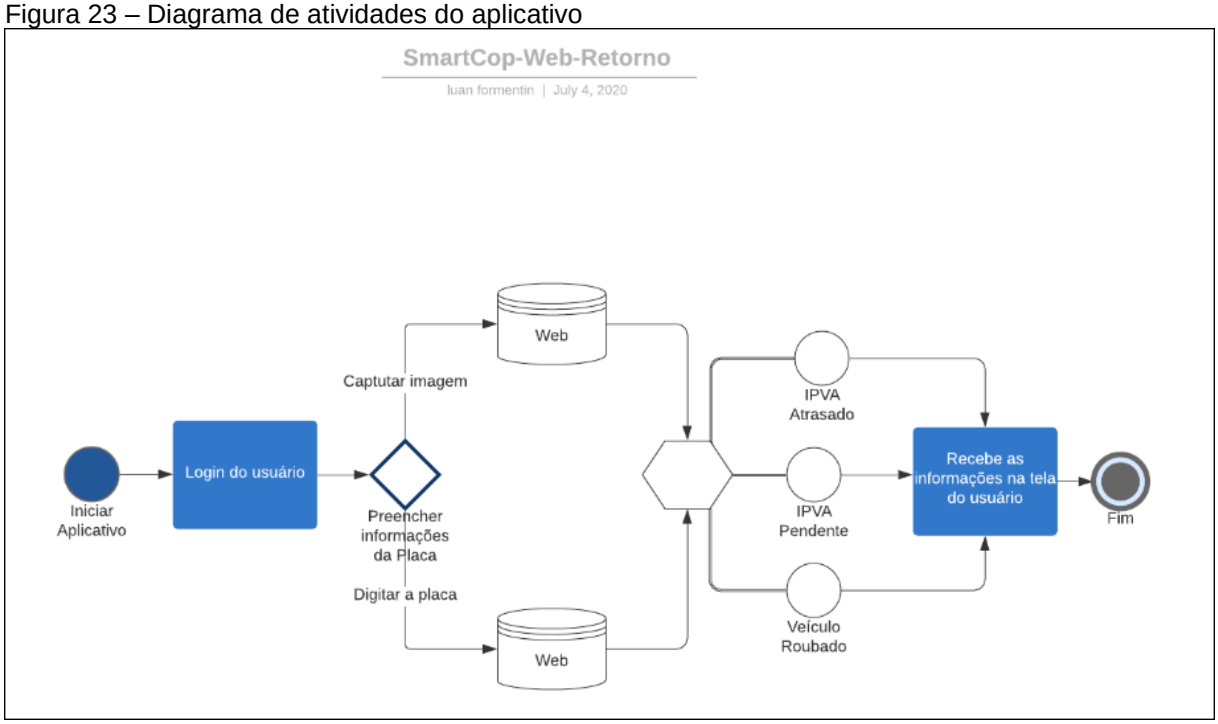
A figura 22 aponta a placa e o estado já capturados após seu devido tratamento. Sabendo que mesmo após todo o tratamento a precisão de captura pode não ser 100% precisa, o aplicativo dá a possibilidade para o agente de trânsito visualizar o que foi coletado pela câmera do *smartphone*. Feito a conferência do conjunto das informações, pode-se clicar na opção **ENVIAR** apresentada por meio da figura 23 listada a seguir.

Figura 22 – Placa capturada

Fonte: Do Autor (2020).

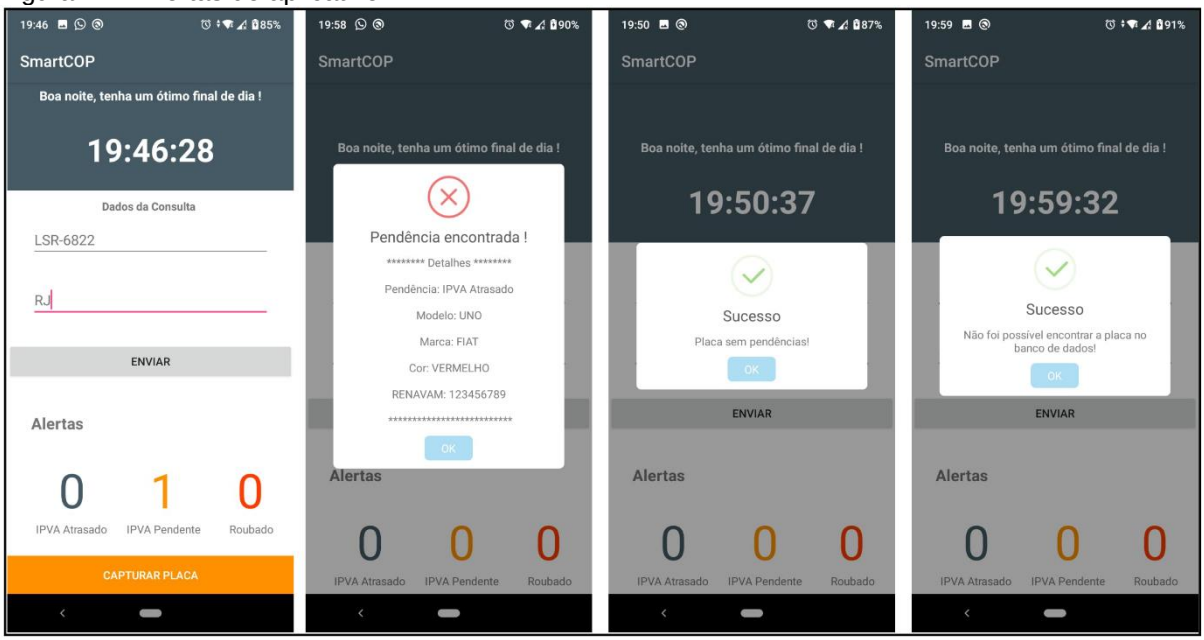
Enviando os dados para o servidor Web, o aplicativo aguardará um retorno desta placa, e como pode ser visualizado na figura 22, os indicativos de alerta receberam a informações e o número 0 será incrementado conforme resposta da API. Para um melhor entendimento da solução, desenvolvemos um Diagrama de atividades do aplicativo no qual é mostrado na figura 23 e possíveis respostas apresentadas por meio do aplicativo na figura 24**.**

Figura 23 – Diagrama de atividades do aplicativo



Fonte: Do Autor (2020).

Figura 24 – Alertas do aplicativo

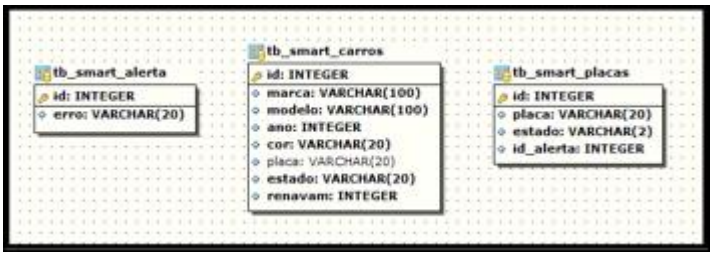


Fonte: Do Autor (2020).

9.5 DESENVOLVIMENTO DA API SMART COP

Para o desenvolvimento da API, foi elaborado uma estrutura utilizando o banco de dados MySQL com três tabelas. A figura 25 mostra a representação das tabelas criadas no banco de dados.

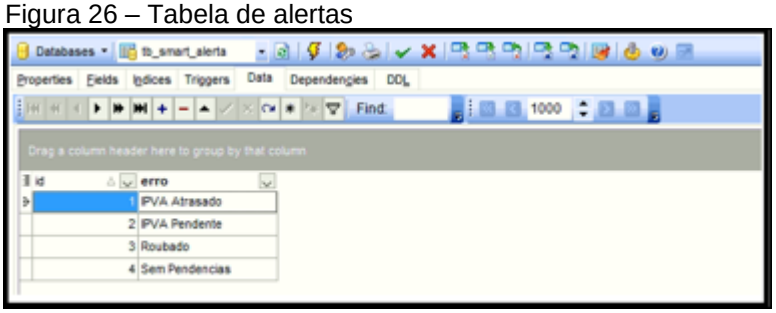
Figura 25 – Tabelas do banco de dados



Fonte: Do Autor (2020).

Conforme demonstrado na figura 26 a tabela ***tb_smart_alerta*** contém todos os códigos de alertas utilizados e tratados pelo aplicativo. Para tal foram definidos os códigos 1, 2, 3 e 4 para simbolizar os alertas de IPVA Atrasado, IPVA Pendente, Roubado e Sem Pendencias respectivamente.

Figura 26 – Tabela de alertas



Fonte: Do Autor (2020).

A tabela ***tb_smart_carros*** contém os carros cadastrados de maneira aleatória. Com isso será possível obter dados sobre determinado veículo após a consulta via aplicativo.

Por último tem-se a tabela ***tb_smart_placas***, no qual deverá ser alimentada com as placas disponibilizadas pela API do DETRAN. Vale ressaltar, que o serviço de API disponibilizado pelo DETRAN passou a ser pago e para não

inviabilizar o projeto de conclusão de curso, as placas e seus respectivos alertas foram cadastradas manualmente nessas tabelas.

A codificação da API foi totalmente feita na linguagem C# e utilizando a IDE de desenvolvimento Visual Studio Community 2017. O servidor foi hospedado na empresa ***GoDaddy*** e no domínio www.nextcodedev.com.br ambos disponibilizados pelo professor Matheus Leandro Ferreira (orientador deste projeto de conclusão de curso). Utilizou-se o conceito de API *Web RestFull* no qual foi definido uma *URL Enpoint* para consulta e validação da placa.

a) *URL* Root: http://www.nextcodedev.com.br/;
b) *URL* EndPoint: api/placa/buscar;
c) *URL* Full: http://www.nextcodedev.com.br/api/placa/buscar

O método *GET* desenvolvido para o uso do *EndPoint* faz uma busca na tabela **tb_smart_placas**, já explicada anteriormente, verificando se existe algum alerta configurado para a placa e estado recebida do aplicativo. Ao mesmo tempo, é feito via **INNER JOIN** a procura das características do veículo na tabela **tb_smart_carros**. Caso encontrado, o policial poderá ter além do código de alerta todos os dados do veículo para a sua devida conferência. A figura 27 mostra a implementação do método *GET* e sua respectiva rota.

Figura 27 – Implementação do método GET

```
17          [HttpGet]
18          [Route("api/placa/buscar")]
19          public RespostaAlerta GetInfo(string placa, string estado)
20          {
21              RespostaAlerta resposta = new RespostaAlerta();
22
23              // Busca os dados da placa.
24              resposta.placa = DAL.GetObjeto<Placa>(string.Format("placa='{0}' and estado='{1}'", placa, estado));
25
26              // Se encontrado ...
27              if (resposta.placa != null) {
28
29                  // Indicativo que a placa possuí um alerta.
30                  resposta.encontrada = true;
31
32                  // Busca o alerta.
33                  resposta.alerta = DAL.GetObjeto<Alerta>(string.Format("id={0}", resposta.placa.id_alerta));
34              }
35
36              // Captura informações sobre o carro se existirem ...
37              resposta.carro = DAL.GetObjeto<Carro>(string.Format("placa='{0}' and estado='{1}'", placa, estado));
38
39              // Faz o retorno das informações.
40              return resposta;
41          }
```

Fonte: Do Autor (2020).

9.6 RESULTADOS OBTIDOS E DISCUSSÃO DOS RESULTADOS OBTIDOS

Neste subcapítulo apresenta-se os resultados obtidos com base nos testes realizados, será demonstrado as placas capturadas pelo aplicativo, algumas falhas e sucesso na captura das placas. Os testes foram realizados com claridades diferentes, em dia mais claro com bastante luz, no final de tarde com luz moderada e a noite para verificar a eficácia do aplicativo com as condições de iluminação.

Na tabela 1 mostra-se os números de placas capturadas com uma amostra de 70 placas e foi possível obter os seguintes resultados:

Tabela 1 – Quantidade de placas testadas

TOTAL DE PLACAS TESTADAS	70
IDENTIFICADAS	63
NÃO IDENTIFICADAS	7

Fonte: Do autor (2020).

Na tabela 1 é apresentados 63 placas que foram identificadas com sucesso e 7 placas não identificas, essas 7 placas não foram identificadas devido a leitura do OCR não conseguir identificar os caracteres, devido muita iluminação do sol em cima da placa e também em alguns casos a placa não estava em bom estado de conservação nas letras e números.

Na tabela 2 demonstra-se que das 63 placas que foram identificadas 58 foi possível reconhecer com sucesso a leitura do OCR e trazer a informação correta na tela do aplicativo totalizando 92% de acerto e apenas 5 placas não tiveram sucesso na leitura do OCR sendo assim, um total de 8% de erro nos testes realizados.

Tabela 2 – Placas identificadas pelo OCR

IDENTIFICADAS	
SUCESSO NO OCR	**ERROS DE OCR**
58	5
92%	8%

Fonte: Do autor (2020).

Tabela 3 – Placas testadas

PLACAS TESTADAS			
IRX-5500	RS	MKG-9926	SC
MTJ-6699	SC	MHM-0058	SC
CIV-9743	SP	HOW-5678	RJ
GBR-7683	SP	MJL-5442	SC
MPI-7392	SC	MOJ-9584	SC
MQW-5678	SC	MIP-5167	SC
MVY-2043	SC	MKW-6582	SC
MSQ-4582	SC	MFQ-0656	SC
MTP-4541	SC	MGE-3687	SC
MYD-1277	SC	MTR-2069	SC
MMJ-4871	SC	MRE-9752	SC
MLG-8784	SC	MKK-6672	SC
MWO-0208	SC	MJP-3268	SC
MRT-0819	SC	MIZ-6297	SC
PHL-4506	AM	MVK-0036	SC
ALH-3930	PR	MMO-6577	SC
AEG-2883	PR	MEY-5498	SC
ATQ-9616	PR	MHF-1326	SC
PIA-8430	PR	MGS-6587	SC
AVB-8063	PR	MGL-6472	SC
ALE-0879	PR	MGT-9726	SC
AYE-8319	PR	MYK-3197	SC
PP0-2617	PR	MWY-3516	SC
QRB-0969	ES	MRJ-3697	SC
MRT-8871	ES	MXD-3589	SC
MKQ-5207	SC	MNQ-0324	SC
MIK-6582	SC	MYW-9751	SC
MHT-4263	SC	MTR-3069	SC
LWR-7801	SC	MHJ-4168	SC
MNT-0204	SC	MKD-5873	SC
LRS-4570	SC	MLS-9647	SC
MKP-3352	SC	MOL-1012	SC
MTT-5246	SC	MSS-6694	SC
MPU-5552	SC	MWZ-1189	SC
MYS-1120	SC	MRF-3065	SC

Fonte: Do autor (2020).

Na figura 28 apresenta a captura de uma placa após ser identificada pelo aplicativo SmartCop. Após esse processo, conforme demonstra na cor azul, o aplicativo trouxe para a tela as informações com sucesso.

Figura 28 – Placa capturada na câmera

Fonte: Do Autor (2020).

Após identificada a placa, na figura 29 apresenta-se as informações na tela do aplicativo, teste realizado com sucesso, placa capturada e com as informações corretas.

Figura 29 – Informações na tela com sucesso

Fonte: Do Autor (2020).

Na figura 30 é apresentada uma placa que o aplicativo não identificou o objeto como uma placa, demonstrando em vermelho que não foi possível identificar e apresentar na tela as informações da placa. Neste caso, a leitura ficou tentando capturar a placa, porém não obteve sucesso.

Figura 30 – Placa não identificada

Fonte: Do Autor (2020).

Na figura 31 é apresentado o teste que foi identificado a placa, no entanto, o OCR apresentou falhas, a placa que foi realizada o teste era a placa MJL-5442, após captura da placa, o aplicativo apresentou a placa WBL-5442.

Figura 31 – Placa com erros de OCR

Fonte: Do Autor (2020).

9.6.1 Sugestões de melhorias

Propõe-se algumas melhorias para dar continuidade ao projeto, conforme segue:

a) Implementação de um banco de dados diretamente no aplicativo, para o agente de trânsito, fiscal ou policial ter um histórico das capturas e diversas informações mostradas no processo de validação da placa;

b) Comunicação da API em conjunto com o DETRAN de cada estado;

c) Criação de um portal com interface para manipulação e relatórios do sistema;

d) Sugestão de implementação e comunicação com uma impressora *bluetooth* para imprimir recibos durante a abordagem, caso o retorno da consulta da placa atender os estados de IPVA Atrasado e IPVA Pendente.

10 CONCLUSÃO

A utilização de aplicativos para automação de processos torna-se cada vez mais necessária devido sua eficácia e o tempo ágil. Em virtude do que foi apresentado neste trabalho, foi possível analisar que os aplicativos são bastante uteis explorando suas câmeras para identificar objetos, realizar o OCR e tratamentos de imagens junto com o processamento digital de imagens.

Sendo assim, o trabalho apresentou uma solução de automação de processos utilizando a api *vision*, aplicada na área de segurança de trânsito, auxiliando nos processos de abordagem, permitindo um trabalho mais ágil além de garantir um tempo de resposta mais rápido para o abordado poder seguir o seu trajeto mediante a velocidade de resposta do aplicativo.

Diante da relevância deste projeto de conclusão de curso para as tecnologias que poderão surgir sobre esse tema, este trabalho trouxe a exploração de diversas técnicas aplicadas no processamento de imagem digital em conjunto com a api *vision* para identificar objetos e realizar leitura de caracteres.

Os resultados tiveram como base 70 placas de carros que foram testadas em diversas condições para analisar a eficácia do aplicativo e das técnicas implementadas para identificar as placas. Diversas variáveis influenciaram nos testes, luminosidade, aproximação da câmera e conservação da placa.

Conclui-se que a api *vision* é uma “ferramenta” alternativa na identificação de objetos conforme apresentado nos resultados obtidos. Em um ambiente controlado o sucesso poderia ser ainda mais satisfatório devido as placas não estarem todas um ótimo estado de conservação.

Em relação ao método ALPR utilizado no reconhecimento de caracteres pode-se dizer que é bastante assertivo quando as placas estão em bom estado conservação, após testes de leitura foi possível verificar que o método trazia informação incorreta ao tentar preencher os espaços que faltavam nos caracteres.

Diante da api relativamente nova e com poucos trabalhos utilizando para a finalidade de leitura de placas e com alguns métodos ainda para serem explorados, conclui-se que este trabalho teve o objetivo alcançado obtendo bons resultados e demonstrando-se muito eficiente.

REFERÊNCIAS

BEAL, A. (2004). **Gestão estratégica da informação**: como transformar a informação e a tecnologia da informação em fatores de crescimento e de alto desempenho nas organizações. São Paulo: Atlas.

BRETAS PEREIRA, Maria José Lara de. Faces da decisão: as mudanças de paradigmas e o poder da decisão. São Paulo: Atlas, 1997.

CÁMARA-CHÁVEZ, Guillermo. **SISTEMA DE CORES.** Disponível em: <http://www.decom.ufop.br>. Acesso em: 26 nov. 2019.

Cardani, d. **Adventures in hsv space,** 2001. Disponível em: http://visl.technion.ac.il/labs/anat/hsvspace.pdf>. Acesso em 07 de nov de 2019.

Castleman, k. R. Digital image processing. New jersey: prentice hall, 1996. 667p.

CONNEL**L**, S. D.; JAIN, A. K. Template-based online character recognition. V. 34, n. August 1999, p. 1–14, 2001.

Corneto, G., da Silva, F. A., Pereira, D. R., de Almeida, L. L., Artero, A. O., Papa, J. P.,de Albuquerque, V. H. and Sapia, H. M. (2017). A new method for automatic vehicle license plate detection, IEEE Latin America Transactions.

Crósta, a.p. **processamento digital de imagens de sdsssensoriamento remoto**. Campinas: instituto de sdssgeociências/unicamp, 1992.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. **Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005**, v. I, p. 886–893, 2005.

DELHI, Instituto de Tecnologia Indiano. **Computer Science & Engineering**. 2019. Disponível em: <http://cse19-iiith.vlabs.ac.in/index.html>. Acesso em: 26 nov. 2019.

DE QUEIROZ, L. C. **Estado da motorização individual no Brasil-Relatorio 2015.** Rio de Janeiro: [s.n.]. Disponível em:<http://www.observatoriodasmetropoles.net/download/automoveis_e_motos2015.pdf>.

DU, S. et al. Automatic license plate recognition (ALPR): A state-of-the-art review. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 23, n. 2, p. 311–325, 2013.

FACON, J. **A morfologia matemática e suas aplicações em processamento de imagens**. In: VII Workshop de Visão Computacional–WVC. [S.l.: s.n.], 2011. v. 13.

FOLHA.UOL. **Brasil tem 1 roubo ou furto de veículo a cada minuto**; Rio lidera o ranking. Disponivel em:<https://www1.folha.uol.com.br/cotidiano/2017/10/1931061-brasiltem-1-roubo-ou-furto-de-veiculo-a-cada-minuto-rio-lidera-o-ranking.shtml> Acesso em: 24 jul. 2019.

FORESTI, Renan Luís.: Sistema de Visão Robótica Para Reconhecimento de Contornos de Componentes na Aplicação de Processos Industriais. 2006. 64p. Dissertação, Mestrado em Engenharia, Programa de Pós-Graduação em Engenharia Mecânica. Universidade Federal do Rio Grande do Sul, Porto Alegre, 2006.

FUNDAÇÃO INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Normas de apresentação tabular**. 3. ed. Rio de Janeiro: IBGE, 1993. Disponível em: < http://biblioteca.ibge.gov.br/visualizacao/livros/liv23907.pdf>. Acesso em: 10 out 2019.

FRASER, Bruce; MURPHY, Chris; BUNTING, Fred. **Real World Color Management**. Second Edition. Estados Unidos: Peachpit Press, 2005. 582 p.

GOMES, Jonas; VELHO, Luiz. **Computação Gráfica: Imagem**, 1ª ed., Rio de Janeiro: IMPA/SBM, 1994.

GOETHE, Johann Wolfgang Von. Doutrina das cores. Apresentação, seleção e tradução Marco Gianotti. São Paulo : Nova Alexandria, 1993.

GOOGLE. **Vision AI**. Disponível em: <https://cloud.google.com/vision/?hl=pt-br>. Acesso em: 20 nov. 2019.

GONZALEZ, R. C.; WOODS, R. E. Processamento de imagens digitais. 3. ed. São Paulo: Edgard Blücher, 2000. 509p.

GONZALEZ, Rafael C.; WOODS , Richard E. **Digital Image Processing**. 3. ed, 2002. 976 p.

GONZALEZ, R. C.; WOODS, R. E. **Digital image processing (3rd edition).** Upper saddle river, nj, usa: prentice-hall, inc., 2006.

GONZALEZ, R. C.; WOODS, R. E. Digital image processing. Upper Saddle River, N.J.: Prentice Hall, 2008.

GONZALEZ, R. C., WOODS, R. C., **Processamento digital de imagens**. Pearson Educación, 2009.

GUIMARÃES, Luciano. A cor como Informação: a construção biofísica, linguística e cultural da simbologia das cores - São Paulo : Annablumme, 2000.

JAIN, N.; LALA, A. **Image segmentation: A short survey**. Confluence 2013: The Next Generation Information Technology Summit (4th International Conference). Anais...2013

KAEHLER, A.; BRADSK, G. . **Learning OpenCV 3 Computer Vision in C++ with the OpenCV Library**. Sebastopol, CA: O'Reilly Media, 2016.

K. Jain, R. P. W. Duin, J. Mao (2000). Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1):4-37.

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

LOPES, João Manuel Brisson. **Cor e Luz**. 2013. 47 páginas. Texto elaborado para a disciplina de Computação Gráfica no curso de Licenciatura em Engenharia Informática e de Computadores. Instituto Superior Técnico. Universidade de Lisboa. Disponível em:. http://disciplinas.ist.utl.pt/leic-cg/textos/livro/Cor.pdf. Acesso em: 29 nov. 2019.

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 06/11/2019.

MOTA, Ana. Cor. **Revista de Ciência Elementar**, [s.l.], v. 5, n. 2, p.61-66, 30 jun. 2017. ICETA. http://dx.doi.org/10.24927/rce2017.019.

NATALE, Ferdinando. **Automação industrial**. São Paulo: Érica, 2002.

NETO**, e. C. Et al. Brazilian vehicle identification using a new embedded plate recognition system.** Measurement: journal of the international measurement confederation**, v. 70, p. 36–46, 2015.**

OZBAY, S.; ERCEBELI, E. **Automatic Vehicle Identification by Plate Recognition**. Proceedings of World Academy of Sciense, Engineerring and Technology. [S.l]: [s.n.]. 2005. p. Volume 9.

PAES, Alexandre Santos Lima. **RECONHECIMENTO AUTOMÁTICO DE PLACAS AUTOMOBILÍSTICAS**. 2017. 59 f. TCC (Graduação) - Curso de Engenharia Eletrônica e de Computação,, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2017.

PASTA, Arquelau. **APLICATIVO PARA AUXÍLIO NA EMISSÃO DOS AUTOS DE INFRAÇÕES DE TRÂNSITO NO MUNICÍPIO DE BLUMENAU**. 2003. 66 f. TCC (Graduação) - Curso de CiÊncias da ComputaÇÃo, Universidade Regional de Blumenau, Blumenau, 2003.

PAUL, Et al. Privacy-preserving evidence in alpr applications. **2015. Disponível em: <https://patents.google.com/patent/ep2887333a1/en>. Acesso em: 08 out. 2019.**

SCARPATO, Christine Vieira et al. **Curso de Tecnologia em Segurança no Trânsito**. Criciúma: UNESC, 2013.

PEDRINI, Hélio; SCHWARTZ, William Robson. Análise de imagens digitais. 1ª ed. São Paulo: Thonsom, 2008.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence - A Modern Approach**. New Jersey: Pearson Education, 2010.

SALAZAR, José Alexander Dueñas. **RECONHECIMENTO AUTOMÁTICO DE PLACAS VEICULARES BRASILEIRAS EM AMBIENTES NÃO CONTROLADOS.** 2017. 135 f. Monografia (Especialização) - Curso de Engenharia de Automação e Sistemas, Universidade Federal de Santa Catarina, Florianópolis, 2017.

STONER, J.A.F.(1999). **Administração**.5.ED. Rio de Janeiro: LTC

TASI – Technical Advisory Service for Images. **An Overview of Color Management**. Disponível em: <http://www.tasi.ac.uk/advice/managing/pdf/colour_manage.pdf>. Acesso em: 02 oct. 2019.

APÊNDICE A – ARTIGO CIENTÍFICO

SMART COP: APLICATIVO PARA LEITURA E VALIDAÇÃO DE PLACAS VEICULARES UTILIZANDO A TECNOLOGIA ALPR APLICADA NA FISCALIZAÇÃO EM TEMPO REAL DE VEÍCULOS DURANTE UMA ABORDAGEM POLICIAL

Prof. Esp. Matheus Leandro Ferrera[1], Luan Alano Formentin[1]

[1]Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC)

CEP 88806-000 – Criciúma – SC – Brasil

mlf@unesc.net, luanformentin@unesc.net

Abstract. *In the constant search for something perfect, one of the great principles of the human being is to create automated machines that can do, every time more, manual and repetitive work. The Roman alphabet together with writing appeared around the VII century b. c. n the course of time and with the improvement of hardware systems, there was a commercial interest for research and development focused on character recognition (OCR). However, with the rapid expansion of the internet in recent years, new paradigms of human-computer interaction have been gaining strength, such as the Application Programming Interface (API) Vision. For this reason, this paper presents the digital image processing techniques (PDI) and the application of API Vision for automatic identification of vehicle plates in real time. They are exposed to the principles of mathematical morphology and some techniques for imaging. The objective is to create a research of the API and explore its many functionalities in the identification of vehicle plates together with optical character recognition (OCR). Therefore, it is possible to automate the transit approach process through a prototype application for identification of vehicle plates. The results obtained were satisfactory based on several vehicle plates tested.*

Keywords: *Vehicle plate recognition. Real-time inspection. Digital image processing. Vision API. Optical character recognition.*

Resumo. *Na busca constante por algo perfeito, um dos grandes princípios do ser humano é criar máquinas automatizadas que possam fazer, cada vez mais, o trabalho braçal e repetitivo. O alfabeto romano em conjunto com a escrita surgiu por volta do século VII a. C. Com o passar do tempo e com a melhoria dos sistemas de hardware, houve o interesse*

comercial para pesquisas e desenvolvimento voltadas para o reconhecimento de caracteres (OCR). No entanto, com a rápida expansão da internet nos últimos anos, novos paradigmas de interação humano-computador vêm ganhando força, como por exemplo a Application Programming Interface (API) Vision. Por este motivo o presente trabalho apresenta as técnicas de processamento digital de imagens (PDI) e a aplicação da API Vision para identificação automática de placas veiculares em tempo real. São expostos os princípios da morfologia matemática e algumas técnicas para tratamento de imagens. O objetivo é realizar o estudo da API e explorar suas diversas funcionalidades na identificação de placas em conjunto com o reconhecimento óptico de caracteres (OCR). Sendo assim, pode-se automatizar o processo de abordagem de trânsito por meio de um protótipo de aplicativo para leitura de placas. Os resultados obtidos foram satisfatórios tendo como base diversas placas testadas.

Palavras-chave: *Reconhecimento de placas veiculares. Fiscalização em tempo real. Processamento digital de imagens. Vision API. Reconhecimento ótico de caracteres.*

1. Introdução

A evolução constante da tecnologia vem trazendo aos órgãos públicos novas formas de gerenciar e organizar informações. Para manter-se atualizados, os setores públicos, estão investindo em inovações de aplicativos que os auxiliam neste processo. A otimização e a facilidade gerada por uma solução mobile vem ganhando força no mercado e se destaca cada vez mais no ambiente de trabalho. Para a polícia civil, por exemplo, a importância de consultas em tempo real torna-se cada dia mais indispensável (REVISTA CIENTÍFICA ELETRÔNICA DE SISTEMAS DE INFORMAÇÃO, 2006).

Para destacar a importância de obter-se acesso a dados em tempo real, Stoner (1999) comenta que somente com informações de acesso imediato e no momento certo, o usuário consegue monitorar o processo na direção correta para posteriormente tomar uma decisão.

A evolução tecnológica combinada com o grande crescimento da necessidade e agilidade nos processos, força as organizações dependerem de aplicativos com eficácia para administrar sua gestão, reduzindo o tempo nos processos de consulta a informação (BEAL, 2004). Para Natale (2002) a automatização é um conjunto de técnicas que nos permite processar informações que anteriormente o homem quem realizava. Os processos não se limitam apenas às indústrias. Inúmeras organizações estão se adequando e utilizando de aplicativos e técnicas processuais

visando maximizar ganhos, como por exemplo o tempo, e para as empresas de fiscalização, não é diferente, também estão se adequando neste novo contexto (NATALE, 2002).

Os agentes ou guardas de trânsito, como são conhecidos, por meio de um convênio com a secretaria de segurança pública, fazem a fiscalização e atuam as infrações cometidas nas vias públicas da cidade. Porém, a forma como a infração é aplicada encontra-se ultrapassada: a dificuldade no preenchimento do auto de infração, uma vez que o mesmo é preenchido de forma manual, e a comunicação com as centrais de operação da polícia, torna o processo dificultoso e lento. O tempo, no qual poderia ser usufruído para novas abordagens, se torna prolongado devido à busca das informações e respostas das centrais, que nem sempre estão disponíveis para auxiliar o agente de trânsito. Por conta disso, pode-se criar a seguinte questão problema: Como reduzir o tempo do agente de trânsito ou policial durante a sua abordagem?

O presente trabalho consiste em aplicar a tecnologia Automatic License Plate Recognition (ALPR) por meio de um aplicativo Android visando automatizar o processo de fiscalização durante uma abordagem fiscal, garantindo uma melhor qualidade na prestação dos serviços. O sistema especialista denominado SMART COP, por ser desenvolvido em Java, pode ser utilizado em smartphone, tablets e até mesmo em óculos inteligentes, visando facilitar o trabalho feito pelos agentes de trânsito, desonerando o tempo gasto com as centrais de operações da polícia, no momento em que estão aplicando as medidas cabíveis ao perceber uma infração ou um veículo irregular.

2. Justificativa

O crescimento socioeconômico junto ao crescimento desorganizado das cidades teve grandes impactos no trânsito. A saturação e o aumento no número de veículos são alarmantes, onde acarreta o controle desses veículos em circulação aumentando a quantidade de infrações e furtos (QUEIROZ, 2015).

Conforme a Folha de São Paulo, o Brasil, em fevereiro de 2018, a cada um minuto um carro era furtado, totalizando 1440 carros roubados diariamente, o que significa aproximadamente meio milhão por ano (FOLHA, 2017).

Além do furto, existe um outro agravante direcionado ao aumento do número de veículos em circulação no Brasil. Dados divulgados pelo DETRAN em um período de 13 anos (de janeiro de 2000 até janeiro de 2013), afirmam que o crescimento de veículos chegou a 160%, tendo

em vista que o crescimento da tecnologia não obteve o desenvolvimento necessário para acompanhar este aumento da frota veicular (SCARPATO et al., 2013).

Nos dias atuais, o reconhecimento automático do conteúdo de placas veiculares permite uma imensa quantidade de aplicações. Por exemplo, para os agentes de trânsito nas cidades possibilita identificar furtos e infrações em um tempo muito mais ágil, controlando de forma mais precisa condutores irregulares (Corneto et al., 2017).

No entendimento de Bretãs Pereira (1997) um sistema eficiente é aquele que facilita a interação com o usuário e traz vantagens para vários setores e segmentos, melhorando eficácia do atendimento e auxiliando na tomada de decisão. É notável que a mudança de procedimentos se faz necessário e se torna frequente nas organizações. A necessidade do acesso à informação em um tempo menor, se torna um fator decisivo para o futuro (ULRICH, 1998).

Uma solução para reconhecimento automático de placas se torna primordial para fiscalização, possibilitando ao agente de trânsito ou policial na identificação automática dos veículos, auxiliando na busca por irregularidades, furtos, controle de tráfego nas estradas e na tomada de decisão.

3. Etapas do desenvolvimento

O projeto de desenvolvimento foi dividido em quatro etapas. A primeira etapa foi a busca pelo Hardware ideal. O protótipo precisaria funcionar com um Hardware de câmera suficiente para conseguir trabalhar com reconhecimento ótico de caracteres. Nesta etapa descobriu-se que deveríamos utilizar a versão do Android ice cream sandwich ou superior, pois nestes modelos o Hardware já é mais bem otimizado.

A segunda etapa foi a construção da interface do aplicativo, utilizando de diversos emuladores presentes no Android Studio para verificar a compatibilidade de layout em relação aos tipos de sistema operacionais. Como já é de conhecimento, o Android possuí diversas versões e tamanho de tela que podem desconfigurar o designer feito pelo programador.

A terceira etapa esteve relacionada a utilização da API Google Cloud Vision para reconhecimento das placas veiculares. Nesta etapa alguns problemas foram apresentados pelas bibliotecas e fui obrigado a tratá-los de maneira individual, como por exemplo a identificação de algumas letras de forma equivocada pela biblioteca.

A quarta e última etapa foi a construção do portal Web, responsável por fazer as validações das placas veiculares, apontando o status de IPVA Atrasado, IPVA Pendente e Carro Roubado.

3.1. Ferramentas e recursos

O primeiro passo para dar início ao desenvolvimento do projeto foi a seleção de recursos a serem utilizados. Foi necessário para o desenvolvimento do protótipo, na parte de Hardware, um aparelho smartphone com 8GB de Armazenamento, 512 de RAM e resolução da câmera traseira superior a 4MP. Em relação a recursos de software, necessitou-se usar a versão do Sistema Operacional Ice cream sandwich ou superior.

No que diz respeito a programação do dispositivo móvel, foi definido a linguagem nativa Java, linguagem já apresentada e utilizada na instituição, por meio da ferramenta IDE Android Studio 3.5.3.

Para leitura das placas veiculares optou-se pela utilização da biblioteca *Google Cloud Vision*. A biblioteca foi selecionada, pois comparado aos demais meios de captura, este modelo conseguiu realizar diversas leituras em pontos diferentes de cena além de reconhecer o objeto "placa" de maneira mais eficiente.

A comunicação feita entre o aplicativo e o portal Web foi feita utilizando a biblioteca gratuita RETROFIT e o portal foi totalmente desenvolvido na linguagem C# da empresa Microsoft já de conhecimento do autor.

O LogCat é uma ferramenta que também vem junto ao Android Studio, que é disponibilizada pelo SDK e tem a funcionalidade de apresentar todos os logs da aplicação. Esta ferramenta nos possibilitou o entendimento de alguns processos errôneos no decorrer do desenvolvimento do aplicativo.

Para finalizar, utilizou-se o *software Astah Community*, versão temporária, sendo possível criar os diagramas utilizados no decorrer deste trabalho.

3.2. Api Smart Cop

Para o desenvolvimento da API, foi elaborado uma estrutura utilizando o banco de dados MySQL com três tabelas**.**

tabela ***tb_smart_alerta*** contém todos os códigos de alertas utilizados e tratados pelo aplicativo. Para tal foram definidos os códigos 1, 2, 3 e 4 para simbolizar os alertas de IPVA Atrasado, IPVA Pendente, Roubado e Sem Pendencias respectivamente.

A tabela ***tb_smart_carros*** contém os carros cadastrados de maneira aleatória. Com isso será possível obter dados sobre determinado veículo após a consulta via aplicativo.

A tabela ***tb_smart_placas***, no qual deverá ser alimentada com as placas disponibilizadas pela API do DETRAN. Vale ressaltar, que o serviço de API disponibilizado pelo DETRAN passou a ser pago e para não inviabilizar o projeto de conclusão de curso, as placas e seus respectivos alertas foram cadastradas manualmente nessas tabelas.
A codificação da API foi totalmente feita na linguagem C# e utilizando a IDE de desenvolvimento Visual Studio Community 2017. O servidor foi hospedado na empresa GoDaddy e no domínio www.nextcodedev.com.br ambos disponibilizados pelo professor Matheus Leandro Ferreira (orientador deste projeto de conclusão de curso). Utilizou-se o conceito de API Web RestFull no qual foi definido uma URL Enpoint para consulta e validação da placa.

a) URL Root: http://www.nextcodedev.com.br/;
b) URL EndPoint: api/placa/buscar;
c) URL Full: http://www.nextcodedev.com.br/api/placa/buscar

O método GET desenvolvido para o uso do EndPoint faz uma busca na tabela tb_smart_placas, já explicada anteriormente, verificando se existe algum alerta configurado para a placa e estado recebida do aplicativo. Ao mesmo tempo, é feito via INNER JOIN a procura das características do veículo na tabela tb_smart_carros. Caso encontrado, o policial poderá ter além do código de alerta todos os dados do veículo para a sua devida conferência. A figura 27 mostra a implementação do método GET e sua respectiva rota.

3.3. Desenvolvimento do aplicativo

O funcionamento principal do aplicativo ocorre fazendo uso da câmera.

a) exigência de câmera: o uso de uma câmera é tão importante para o aplicativo que você não quer a instalação dele em um dispositivo que não tenha esse recurso. Nesse caso, declare a exigência de câmera no seu manifesto;

b) solicitar permissões do app: cada aplicativo Android é executado em um sandBox com acesso limitado. Se um aplicativo usar recursos ou informações fora do próprio sandBox, ele precisará a permissão apropriada.

Além destas implementações, foi necessário criar o nosso próprio objeto câmera, isso para que fosse possível obter o máximo de desempenho na captura de imagens e especificamente das placas do veículo. Para criar este objeto, algumas etapas precisaram ser desenvolvidas:

a) **detectar e acessar a câmera:** criou-se um código para verificar a existência de câmeras no Smartphone e solicitar acima (apontado anteriormente);

b) **criar uma classe de visualização:** criou-se uma classe de visualização da câmera que estenda a *SurfaceView* e implemente a interface *SurfaceHolder*. Essa classe exibe as imagens em tempo real da câmera;

c) **criar um layout de visualização:** depois de ter a classe de visualização da câmera, criou-se o layout que incorporou a visualização e os controles de interface do usuário;

d) **configurar listeners para a captura:** foi necessário criar ouvintes de captura para as placas veiculares;

e) **capturar e salvar os dados:** criou-se rotinas de captura e salvamento das placas capturadas pelo aplicativo;

f) **liberar o uso da câmera:** após toda coleta de dados, foi necessário criar uma rotina para liberar o uso da câmera para não bloquear outros aplicativos.

Após implementar todos os recursos de câmeras, o aplicativo abrirá a tela denominada *SurfaceView* e iniciará o processo de captura.

4. Resultados

Os testes foram realizados com claridades diferentes, em dia mais claro com bastante luz, no final de tarde com luz moderada e a noite para verificar a eficácia do aplicativo com as condições de iluminação.

Foram testadas 70 placas, 63 placas que foram identificadas com sucesso e 7 placas não identificas, essas 7 placas não foram identificadas devido a leitura do OCR não conseguir identificar os caracteres, devido muita iluminação do sol em cima da placa e também em alguns casos a placa não estava em bom estado de conservação nas letras e números.

Das 63 placas que foram identificadas 58 foi possível reconhecer com sucesso a leitura do OCR e trazer a informação correta na tela do aplicativo totalizando 92% de acerto e apenas 5 placas não tiveram sucesso na leitura do OCR sendo assim, um total de 8% de erro nos testes realizados.

5. Conclusão

O trabalho apresentou uma solução de automação de processos utilizando a api *vision*, aplicada na área de segurança de trânsito, auxiliando nos processos de abordagem, permitindo um trabalho mais ágil além de garantir um tempo de resposta mais rápido para o abordado poder seguir o seu trajeto mediante a velocidade de resposta do aplicativo.

Diante da relevância deste projeto de conclusão de curso para as tecnologias que poderão surgir sobre esse tema, este trabalho trouxe a exploração de diversas técnicas aplicadas no processamento de imagem digital em conjunto com a api *vision* para identificar objetos e realizar leitura de caracteres.

Os resultados tiveram como base 70 placas de carros que foram testadas em diversas condições para analisar a eficácia do aplicativo e das técnicas implementadas para identificar

as placas. Diversas variáveis influenciaram nos testes, luminosidade, aproximação da câmera e conservação da placa.

Conclui-se que a api *vision* é uma "ferramenta" alternativa na identificação de objetos conforme apresentado nos resultados obtidos. Em um ambiente controlado o sucesso poderia ser ainda mais satisfatório devido as placas não estarem todas um ótimo estado de conservação.

Em relação ao método ALPR utilizado no reconhecimento de caracteres pode-se dizer que é bastante assertivo quando as placas estão em bom estado conservação, após testes de leitura foi possível verificar que o método trazia informação incorreta ao tentar preencher os espaços que faltavam nos caracteres.

Diante da api relativamente nova e com poucos trabalhos utilizando para a finalidade de leitura de placas e com alguns métodos ainda para serem explorados, conclui-se que este trabalho teve o objetivo alcançado obtendo bons resultados e demonstrando-se muito eficiente.

Referências

BEAL, A. (2004). **Gestão estratégica da informação:** como transformar a informação e a tecnologia da informação em fatores de crescimento e de alto desempenho nas organizações. São Paulo: Atlas.

Corneto, G., da Silva, F. A., Pereira, D. R., de Almeida, L. L., Artero, A. O., Papa, J. P.,de Albuquerque, V. H. and Sapia, H. M. (2017). A new method for automatic vehicle license plate detection, IEEE Latin America Transactions.

DE QUEIROZ, L. C. Estado da motorização individual no Brasil-Relatorio 2015. Rio de Janeiro: [s.n.]. Disponível em:<http://www.observatoriodasmetropoles.net/download/automoveis_e_motos2015.pdf>.

FOLHA.UOL. **Brasil tem 1 roubo ou furto de veículo a cada minuto**; Rio lidera o ranking. Disponivel em:<https://www1.folha.uol.com.br/cotidiano/2017/10/1931061-brasiltem-1-roubo-ou-furto-de-veiculo-a-cada-minuto-rio-lidera-o-ranking.shtml> Acesso em: 24 jul. 2019.

NATALE, Ferdinando. **Automação industrial**. São Paulo: Érica, 2002.

PAUL, Et al. Privacy-preserving evidence in alpr applications. **2015. Disponível em: <https://patents.google.com/patent/ep2887333a1/en>. Acesso em: 08 out. 2019.**

SCARPATO, Christine Vieira et al. **Curso de Tecnologia em Segurança no Trânsito**. Criciúma: UNESC, 2013.

STONER, J.A.F.(1999). **Administração**.5.ED. Rio de Janeiro: LTC