UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

JACKSON GUIZZO ZANETTE

RECONHECIMENTO AUTOMÁTICO DE PLACAS VEICULARES UTILIZANDO A TECNOLOGIA OCR E A PLATAFORMA RASPBERRY PI APLICADA NA FISCALIZAÇÃO ELETRÔNICA DE RODOVIAS

CRICIÚMA

2016

JACKSON GUIZZO ZANETTE

RECONHECIMENTO AUTOMÁTICO DE PLACAS VEICULARES UTILIZANDO A TECNOLOGIA OCR E A PLATAFORMA RASPBERRY PI APLICADA NA FISCALIZAÇÃO ELETRÔNICA DE RODOVIAS

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. Sergio Coral

CRICIÚMA

2016

JACKSON GUIZZO ZANETTE

RECONHECIMENTO AUTOMÁTICO DE PLACAS VEICULARES UTILIZANDO A TECNOLOGIA OCR E A PLATAFORMA RASPBERRY PI APLICADA NA FISCALIZAÇÃO ELETRÔNICA DE RODOVIAS

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 Automação.

Criciúma, 02 de dezembro de 2016.

BANCA EXAMINADORA

Prof. Sérgio Coral - Especialista - UNESC - Orientador

Prof. Giácomo Antônio Althoff Bolan - Especialista - UNESC

Prof. Luciano Antunes - Mestre - UNESC

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, que sempre me incentivaram e possibilitaram que eu concluísse este curso.

Agradeço também à minha namorada Júlia, que sempre insistiu para que eu me dedicasse a este projeto e me ajudou a superar todas as dificuldades encontradas, sempre sendo compreensiva nas horas em que tudo parecia perdido.

Estendo meus agradecimentos à todos os professores da Ciência da Computação da UNESC, em especial ao meu orientador Sérgio Coral que acreditou em mim e forneceu todo o seu conhecimento para que este projeto se concluísse.

RESUMO

Os acidentes de trânsito causam anualmente 1 milhão e 200 mil mortes no mundo e um dos principais motivos é o excesso de velocidade. O problema da violência no trânsito é um problema complexo que vêm sido discutido por décadas e é largamente estudado pela Organização Mundial da Saúde, que define ações eficientes para prevenção de acidentes. Uma destas ações é a redução e fiscalização eletrônica da velocidade. O modelo de fiscalização eletrônica utiliza largamente os radares fixos, porém os mesmos medem apenas a velocidade no ponto instalado. Um novo modelo proposto seria utilizar radares que estivessem posicionados a distâncias conhecidas entre si interligados em uma rede e reconhecessem cada veículo que passassem pelos mesmos e medir a velocidade dos veículos pela diferença entre tempo e distância dos registros da passagem dos veículos. A versatilidade e o baixo preço do Raspberry Pi em conjunto com uma câmera possibilitam o desenvolvimento de tal modelo utilizando o processamento de imagens para reconhecer automaticamente cada veículo pela sua placa. Para desenvolver um protótipo deste modelo de fiscalização, foi utilizado um circuito fechado de autorama com um conjunto de Raspberry Pi e câmera em associação com as bibliotecas OpenCV e Tesseract OCR. Foi possível desenvolver um protótipo que capturasse fotos e reconhecesse as placas, porém percebeu-se que os algoritmos de processamento de imagens são complexos e sensíveis à mudança de ambientação da imagem, necessitando de um maior refinamento para aplicação na vida real.

Palavras-chave: Acidentes de trânsito. Fiscalização eletrônica. Processamento de imagens. OCR. Raspberry Pi.

ABSTRACT

Traffic accidents cause 1.2 millions deaths annually in the world and one of the main causes is the speeding. The traffic violence is a complex problem which have been discussed for decades and is widely studied by the World Health Organization, which defines efficient actions to prevent accidents. One of this actions is the reduction and electronic suverillance of speed. The electronic surveillance model uses widely fixed radars, homever the speed measure occurs only in the point which the radar ins installed. One new model proposed uses radars positioned at known distances connected in a network identifying the vehicles passing by and measuring the speed by the difference between time and distance of the photograph registers. The versatility and low cost of the Raspberry Pi associated with a camera make the development of that model possible by the digital image processing to identify the vehicle by your license plate. To develop a prototype of this model of electronic surveillance, was used a slot cars track with a Rasperry Pi and a camera associated with the libraries OpenCV and Tesseract OCR. Was possible to develop a prototype which made photos and identified the license plates, homever was observed that the processing algorithms are complex and sensible by the change of the enviroment of the image, needing a bigger refinement to apply in the real life.

Keywords: Traffic Accidents. Electronic Surveillance. Digital Image Processing. OCR. Raspberry Pi.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANTP	Associação Nacional de Transportes Públicos
CID-10	Prefeitura Municipal de Criciúma
CISC	*Complex Instruction Set Computer*
CLI	*Command Line Interface*
CSI	*Camera Serial Interface*
CTB	Código de Trânsito Brasileiro
Contran	Conselho Nacional de Trânsito
DRAM	*Dynamic Random Access Memory*
DSI	*Display Serial Interface*
FEVAT	Federação Européia de Vítimas de Acidentes de Trânsito
GB	*GigaBytes*
GPIO	*General-Purpose Input/Output*
GPU	*Graphic Processing Unit*
HDMI	*High-Definition Multimedia Interface*
IPEA	Instituto de Pesquisa Econômica Aplicada
Km	Quilômetros
Km/h	Quilômetros por hora
LAP	Leitor Automático de Placas
LED	*Light Emitting Diode*
mAh	miliAmpère-hora
MHz	*MegaHertz*
MP	*MegaPixels*
OCR	*Optical Character Recognition*
OEM	Órgãos Executivos Municipais de Trânsito
OMS	Organização Mundial da Saúde
ONU	Organização das Nações Unidas
PIB	Produto Interno Bruto
PID	Processamento de Imagens Digitais
RISC	*Reduced Instruction Set Computer*
RPI	Raspberry Pi
SACAVA	Sistema Automático de Controle de Acesso de Veículos Automotivos
SBC	*Single Board Computer*

SD	*Secure Digital*
STJ	Superior Tribunal de Justiça
SUS	Sistema Único de Saúde
SoC	*System on a Chip*
TJ/SC	Tribunal de Justiça de Santa Catarina
USB	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos a frota brasileira de veículos tem apresentado crescimento expressivo. Entre 2005 e 2011 por exemplo, o aumento foi de 160% (JORGE, 2013) enquanto a população cresceu 6% no mesmo período (THE WORLD BANK, 2016, tradução nossa), demonstrando o crescimento da acessibilidade do automóvel.

A boa notícia é que o número de acidentes de trânsito não demonstrou linearidade durante os anos apurados, apresentando períodos de acréscimo e decréscimo e provando não estar diretamente relacionado com o número de veículos nas vias (JORGE, 2013).

Esta não linearidade no índice de acidentes de trânsito demonstra que muitos outros fatores influenciam na ocorrência de um acidente. Segundo World Health Organization (2015, tradução nossa) o acidente de trânsito é previsível e evitável.

Segundo Peden et al (2004, tradução nossa) a velocidade dos veículos está no centro do problema dos acidentes de trânsito, influenciando nos riscos e nas consequências do mesmo. Estima-se que a cada 3 acidentes fatais em países com grande quantidade de veículos, 1 tem como principal motivo o excesso de velocidade.

A gestão da velocidade tem como principal método a fiscalização eletrônica, utilizada de forma fixa ou móvel, sinalizada ou oculta. Um dos principais problemas da fiscalização sinalizada é a redução da velocidade dos condutores trafegam em excesso e avistam a mesma, tornando a acelerar após a fiscalização (WORLD HEALTH ORGANIZATION, 2012, tradução nossa).

Os radares fixos de fiscalização eletrônica possuem uma câmera para registrar o veículo infrator, porém poucos são do tipo Leitor Automático de Placas (LAP). Este tipo de radar permite identificar automaticamente o veículo pela placa através da tecnologia de Reconhecimento Ótico de Caracteres, do inglês *Optical Character Recognition* (OCR) e abre possibilidades para a fiscalização de mais de uma infração em um único aparelho (MING, 2006). Segundo a ASTC (2011), o custo de implantação e manutenção deste tipo de radar chega a ser quase 6 vezes maior

se comparado com o radar sem esta tecnologia.

O OCR permite o reconhecimento de caracteres presente em uma imagem digital adquirida por escâneres e câmeras. Imagens obtidas de câmeras costumam apresentar um desafio para os softwares de OCR, por apresentar distorções e variações de luz (MITHE et al, 2013, tradução nossa).

1.1 OBJETIVO GERAL

Criar um radar de fiscalização eletrônica de rodovias utilizando a tecnologia OCR para reconhecimento de placas apoiando-se na plataforma Rapsberry Pi para medição da velocidade média dos veículos.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste projeto são:

a) levantar estatísticas sobre acidentes de trânsito no Brasil;
b) levantar dados sobre a eficiência da fiscalização eletrônica;
c) desenvolver uma aplicação capaz de reconhecer placas veiculares através de imagens aplicando as técnicas de processamento de imagens.
d) Integrar as aplicações com o Raspberry Pi (RPI) e com a câmera *RaspiCam*

1.3 JUSTIFICATIVA

Estima-se que 1 milhão e 200 mil pessoas morrem anualmente por causa dos acidentes de trânsito. Os acidentes de trânsito são a principal causa de morte entre pessoas com idade entre 15 e 29 anos (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

A Organização Mundial da Saúde (OMS) define os acidentes de trânsito como um grave problema de saúde pública que vem sendo discutido por décadas e requer esforços para a prevenção do mesmo (PEDEN et al, 2004, tradução nossa).

O setor que sofre mais impacto com este problema é o de saúde, pois é sobre ele que recairão todos os gastos para tratamento das vítimas. Dados levantados pelo Sistema Único de Saúde (SUS) apurou que as lesões decorrentes dos acidentes de trânsito são mais onerosas que lesões decorrentes de outros tipos de acidentes, violência e causas naturais em conjunto (JORGE, 2013).

O impacto na economia também é grande, tanto em nível pessoal quanto nacional. A maioria dos acidentes ocorrem entre a população economicamente ativa e isto gera perca renda nas famílias e gastos com tratamentos. Estima-se que os acidentes de trânsito reduzem o Produto Interno Bruto (PIB) de países em desenvolvimento em cerca de 5%, enquanto a média mundial fica em 3% (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

Estudos apontam que a fiscalização eletrônica aplicada corretamente reduz o número de acidentes em cerca de 30% e o número de vítimas fatais de aproximadamente 60% (CANNEL, 2001).

A relação distância sobre tempo aplicada à dispositivos de medição de velocidade fornece um método eficaz e indiscutível de avaliar a velocidade equitativa do condutor, eliminando possíveis justificativas para o excesso de velocidade, como a realização de uma ultrapassagem (WORLD HEALTH ORGANIZATION, 2012, tradução nossa).

A tecnologia OCR lida com o problema de reconhecer caracteres escritos por máquina ou a mão, porém enquanto o processamento de texto escrito à mão com pouca padronização é natural para humanos, pode ser um problema para o OCR. No caso de textos padronizados escritos por máquinas, caso das placas veiculares, o computador consegue realizar o processamento mais rápido que os humanos (ADHVARYU, 2013, tradução nossa).

O avanço da fotografia digital possibilitou a aplicação do OCR em dispositivos móveis, porém tais aplicações devem ser otimizadas para a arquitetura ARM visto o menor poder de processamento em frente aos desktops comuns. (MOLLAH et al, 2011, tradução nossa).

O OCR tem avançado de forma a alcançar taxas de reconhecimento acima de 99% para textos impressos e acima de 90% em textos escritos à mão em documentos (CHOPRA et al, 2014, tradução nossa).

O RPI é um computador desenvolvido no Reino Unido voltado para o ensino da ciência da computação. Trata-se de um computador de placa única (SBC, do inglês *Single Board Computer*) com tamanho próximo à um cartão de crédito com preços entre 25 e 35 dólares. O propósito do RPI era ser um computador de baixo custo destinado a experiências educacionais com o fim de despertar o interesse de adultos e crianças pela Ciência da Computação. (GOLDEN, 2013, tradução nossa).

1.4 ESTRUTURA DO TRABALHO

No capítulo 2 deste trabalho, são apresentados dados sobre os acidentes de trânsito em nível global e a nível Brasil, além de discorrer sobre a influência do excesso de velocidade nos acidentes, comportamentos que influenciam no excesso de velocidade e as consequências geradas pelos acidentes de trânsito.

O capítulo 3 aborda os conceitos da fiscalização da velocidade e os cenários da mesma no mundo e no Brasil. Este capítulo também apresenta os métodos para a fiscalização da velocidade.

O capítulo 4 discorre sobre a plataforma RPI, apresentando também alguns acessórios e uma introdução sobre a arquitetura ARM, utilizada no processador do mesmo. Neste mesmo capítulo também são abordadas brevemente as plataformas de prototipagem Arduino, Galileo e BeagleBone.

No capítulo 5 aborda o OCR apresentando sua história, etapas, técnicas. Neste mesmo capítulo são abordados brevemente os conceitos de processamento digital de imagens e visão computacional, além das bibliotecas OpenCv e Tesseract.

Por fim, o capítulo 6 apresenta os trabalhos correlatos a este projeto.

2 ACIDENTES DE TRÂNSITO

Os acidentes de trânsito são comuns no dia-a-dia, causando mortes e ferimentos de milhares de pessoas. Milhões passam por hospitais anualmente após acidentes graves, sendo que tais acidentes irão mudar a vida de algumas pessoas deixando sequelas. Tais acidentes constituem um problema de saúde pública que afeta principalmente alguns grupos de motoristas. Em países e regiões menos desenvolvidos como África, Ásia, Caribe e América Latina, a maioria das mortes em acidentes de trânsito atingem pedestres, ciclistas, usuários de veículos motorizados de duas rodas e ocupantes de ônibus. Por outro lado, em países desenvolvidos as mortes ocorrem mais frequentemente em usuários de carros. A boa notícia é que este problema pode ser prevenido: Em países desenvolvidos, um conjunto de ações tem trazido redução nos índices de acidentes e na severidade dos mesmos. Estas ações incluem alterações na legislação para controlar o excesso de velocidade, ingestão de álcool, uso obrigatório do cinto de segurança e capacetes no caso de ciclistas e motociclistas, o desenvolvimento de veículos e estradas mais seguros e claro, o uso mais consciente por parte dos motoristas desses veículos e estradas (PEDEN et al 2004, tradução nossa).

Segundo Jorge (2013, p. 15), acidente de trânsito é conceituado como:

> Acidente de trânsito é o acidente com veículo, ocorrido na via pública, sendo esta entendida como a largura total entre dois limites de propriedade e todo terreno ou caminho aberto ao público para circulação de pessoas ou bens de um lugar para outro.

O problema dos acidentes de trânsito possui pouco investimento diante de outros problemas de saúde. Uma das razões para esta negligência é a visão tradicional de que os acidentes e lesões são eventos aleatórios que acontecem aos outros, além de serem vistos como um custo inevitável dos transportes. O trânsito é um sistema complexo e perigoso que milhares de pessoas tem que lidar todos os dias, e por mais que a chance de um acidente seja relativamente baixo considerando-se um deslocamento individual, a soma dessa pequena chance multiplicado por todos esses dias no trânsito a transforma em um risco considerável de ser envolver em um acidente. O próprio termo “acidente” dá a impressão de que é um evento inevitável e imprevisível, porém é necessário desfazer este conceito e assumir que se trata de um evento causado por uma série de fatores. A visão mais

comum é que, quando ocorre um acidente, o mesmo é de inteira responsabilidade dos usuários da estrada, ignorando o fato de que as estradas e os carros podem não estar em condições ideais ou não serem seguros. Erros na direção podem contribuir com um acidente, porém podem não ser a causa principal do mesmo (PEDEN et al 2004, tradução nossa).

A maioria dos acidentes de trânsito é previsível e evitável, pois há evidências de que intervenções para tornar as estradas mais seguras foram implantadas com sucesso em alguns países resultando em uma redução nas mortes no trânsito. Levar tais intervenções para um nível global traduz em um futuro com menos danos e mortes no trânsito (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

O comportamento humano no trânsito não é influenciado apenas no conhecimento e habilidade do mesmo, mas também por todo o ambiente em volta, como o desenho da estrada, o tipo do veículo, as leis de trânsito e a presença ou ausência de fiscalização. Portanto, ações que visam a redução de acidentes devem focar-se tanto na alteração de comportamento do motorista quanto alterando o ambiente no qual este comportamento é inserido. A incerteza do comportamento humano inserido em um ambiente complexo de trânsito significa que é impossível que todos os acidentes sejam evitados, porém pode-se desenvolver um sistema de trânsito que leve em conta os limites do corpo para que quando houver um acidente, seus efeitos sejam atenuados. A maioria dos acidentes fatais ocorrem pelo fato do corpo humano sofrer uma carga e/ou uma aceleração além do tolerado. Para efeito de conhecimento, um pedestre atropelado por um carro a uma velocidade de 50 quilômetros por hora (km/h) representa um risco de 80% de morte. Este risco baixa para 10% se a velocidade for de 30 km/h (PEDEN et al, 2004, tradução nossa).

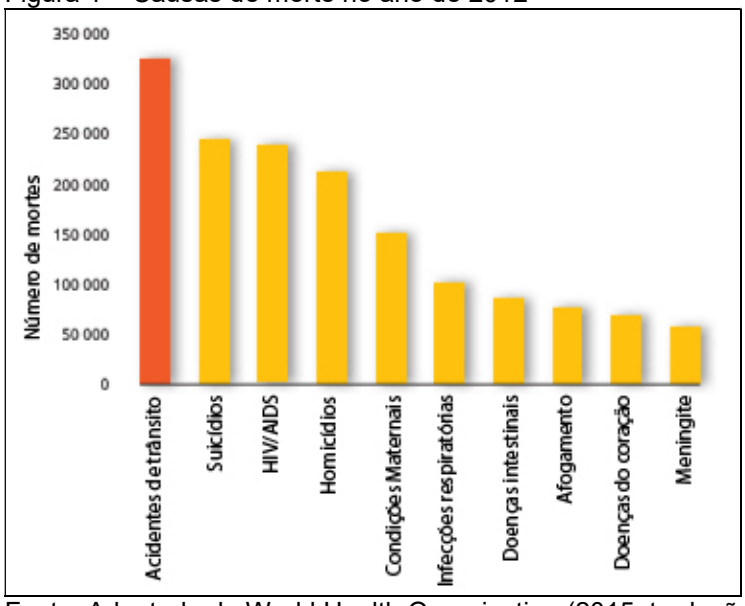
Diante do tamanho deste problema de saúde e desenvolvimento, a assembleia da Organização das Nações Unidas (ONU) adotou uma resolução estabelecendo a década das ações para a segurança no trânsito entre 2011 e 2020. Esta resolução estimula os países membros à tomarem medidas necessárias para tornar as estradas mais seguras, no qual a efetividade de tais ações é medida pela OMS anualmente pelo relatório *Global Status Report on Road Safety* (WORLD HEALTH ORGANIZATION, 2015, tradução nossa)

2.1 ESTATÍSTICAS SOBRE ACIDENTES DE TRÂNSITO NO MUNDO

Acidentes de trânsito causam mais de 1 milhão e 200 mil mortes no mundo todos os anos e cerca de 50 milhões sofrem lesões sérias podendo ficar com sequelas pelo resto da vida. Ações para combater este problema global têm sido insuficientes principalmente nos países menos desenvolvidos, que têm em média a taxa de morte duas vezes maior em comparação aos países desenvolvidos e concentram 90% do total das mortes. (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

Os acidentes de trânsito representam a causa principal de morte entre pessoas de 15 e 29 anos. Estima-se que até 2030 o trânsito será a sétima causa de morte no mundo. Este crescimento estimado é principalmente apontado pela rápida urbanização e pelo aumento da frota de veículos em países em desenvolvimento, dos quais costumam ter problemas de infraestrutura ou necessitam aplicar os investimentos em outros setores. Alguns países desenvolvidos já conseguiram quebrar a relação entre o crescimento da frota e o crescimento dos acidentes tomando ações como tornar as estradas e veículos mais seguros (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

Figura 1 – Causas de morte no ano de 2012

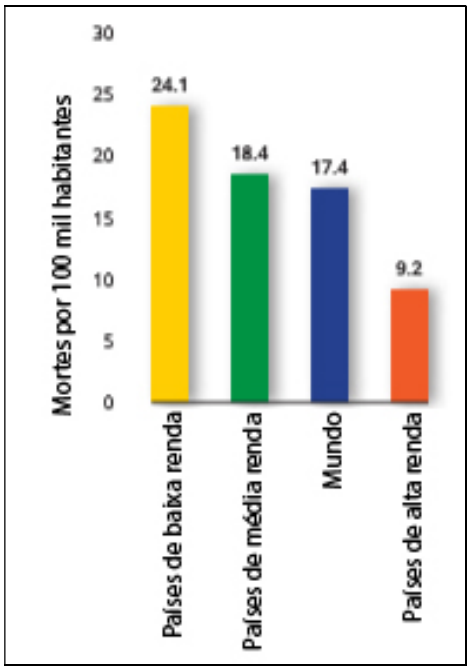


Fonte: Adaptado de World Health Organization (2015, tradução nossa).

Apesar do alto número de mortes anuais, atingindo 1.25 milhão por no ano de 2013, observou-se que este número está estabilizado desde 2007 mesmo com o aumento em 4% da população e em 16% dos veículos registrados globalmente entre 2010 e 2013, o que mostra reflexos de que os países estão dando mais atenção a este problema tornando as estradas mais seguras. Ainda que seja um bom sinal, este estancamento do crescimento no número de mortes ainda é insuficiente e não há sinais de declínio do índice. Isto significa que será necessário muito esforço político e muitos recursos para atingir o índice estabelecido pela década das ações para segurança no trânsito estabelecida pela ONU, que determina a redução em 50% das mortes até 2020 (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

Além no número absoluto de mortes, um índice importante à ser observado é o risco de morte no trânsito em cada país. O índice global é de 17,4 mortes por 100 mil habitantes, porém há uma grande diferença neste índice comparando-se países desenvolvidos com países em desenvolvimentos e subdesenvolvidos, que pode ser observado na figura 2 (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

Figura 2 – Número de mortes por 100 mil habitantes separados por índice de desenvolvimento em 2013



Fonte: Adaptada World Health Organization (2015, tradução nossa).

2.2 ESTATÍSTICAS SOBRE ACIDENTES DE TRÂNSITO NO BRASIL

Assim como em outros países, o problema no trânsito no Brasil também envolve diversos setores como segurança, engenharia automotiva e de transportes, educação, legislação, medicina, dentre outras. O setor de saúde é o mais afetado por estes problemas, pois o mesmo será envolvido desde o atendimento aos acidentes até o tratamento dos pacientes, que podem apresentar sequelas reversíveis ou não. O problema dos acidentes de trânsito gera uma sobrecarga no SUS causando altos gastos ao governo. Segundo o próprio SUS, as lesões decorrentes de acidentes de trânsito causam mais gastos que lesões consequentes de outros acidentes, violência e causas naturais somados. A maioria das mortes no trânsito ocorre na população economicamente ativa, o que agrava os efeitos dos acidentes, trazendo problemas à economia e diminuindo a taxa de esperança de vida do país (JORGE, 2013).

A frota de veículos do Brasil teve um crescimento expressivo nas últimas duas décadas. De cerca de 27 milhões de veículos em 1996 este número saltou para cerca de 70 milhões de veículos em 2011, representando um aumento de 160% no número de veículos registrados no país. No mesmo período, a quantidade de veículos por 1000 habitantes aumentou 106%, mostrando a rápida urbanização do país. Estes números podem ser observados na figura 3 (JORGE, 2013).

Figura 3 – Aumento do número de veículos entre 1996 e 2011 no Brasil

1.T.1 – Frota de veículos (número e taxa por mil habitantes), Brasil, 1996 a 2011

Ano	N	Frota por 1.000 habitantes
1996	27.747.815	176,7
1997	28.886.466	181,0
1998	30.939.466	191,2
1999	32.318.646	197,1
2000	29.722.950	175,0
2001	31.913.003	185,8
2002	34.284.967	197,3
2003	36.658.501	208,5
2004	39.240.875	220,6
2005	42.071.961	233,8
2006	45.372.640	249,2
2007	49.644.025	269,5
2008	54.506.661	292,5
2009	59.361.642	314,9
2010	64.817.974	339,8
2011	70.543.535	365,6

Fonte: Jorge (2013).

Apesar do número sempre crescente do número de veículos, é importante ressaltar que o número de acidentes não acompanhou em mesmo ritmo o crescimento de veículos. Enquanto o número de veículos nas ruas aumentou em 40% no período entre 1998 e 2005, o número de acidentes com vítimas subiu 46% no mesmo período. À primeira vista, pode-se dizer que os números estão diretamente relacionados, porém o gráfico do crescimento número de acidentes no período é irregular, podendo ser observado na figura 4 (JORGE, 2013).

Figura 4 – Taxa de acidentes de trânsito por cem mil habitantes

Fonte: Jorge (2013).

Outro dado que revela que estes índices não estão conectados aparece quando se analisa o estado de Santa Catarina. Enquanto a o número de veículos por mil habitantes cresceu em torno de 42% entre 1998 e 2005, o número de acidentes com vítimas teve uma queda expressiva de 386% no mesmo período (JORGE, 2013).

A número de mortes causadas por acidentes de trânsito cresceram de cerca de 35 mil em 1996 para cerca de 43 mil em 2011, o que significa um aumento de 22%. A taxa de mortes era de 22,4 por cem mil habitantes, sendo um número virtualmente igual ao número de 1996, que era de 22,5. Durante o período, houve um decréscimo no número entre 1996 e 2000, ano que apresentou o índice mais baixo do período, 17,1 mortes por 100 mil habitantes. Após o ano de 2000, o índice voltou a crescer até atingir o pico novamente em 2010, 22,5. Estes dados podem ser verificados na figura 5 (JORGE, 2013).

Figura 5 – Mortes por acidentes de trânsito no Brasil entre 1996 e 2011

3.1.T.1 – Mortes por acidentes de transporte terrestre – ATT (N e taxa em relação à população), Brasil, 1996 a 2011

Ano	N	Taxa (por cem mil habitantes)
1996	35.281	22,5
1997	35.620	22,3
1998	30.890	19,1
1999	29.569	18,0
2000	28.995	17,1
2001	30.524	17,8
2002	32.753	18,8
2003	33.139	18,8
2004	35.105	19,7
2005	35.994	20,0
2006	36.367	20,0
2007	37.407	20,3
2008	38.273	20,5
2009	37.594	19,9
2010	42.844	22,5
2011	43.256	22,4

Fonte: Jorge (2013).

Estas taxas são consideradas bastante elevadas, representando cerca uma vez e meia a taxa encontrada nos Estados Unidos e duas vezes a taxa encontrada no Canadá, fazendo com que o Brasil fique acima da média entre os

países das Américas.

A distribuição entre sexos nas estatísticas mostra que as taxas de mortes entre os homens estão sempre maiores do que entre as mulheres. Em 2011, houveram 37,7 mortes de homens por 100 mil habitantes, enquanto este número nas mulheres foi de 7,8.

As faixas etárias que concentram o maior número de vítimas são entre 20 e 29 anos e nas idades acima de 70 anos. Pode-se explicar as taxas altas em idades avançadas pelo fato de que os idosos estão sujeitos à mais riscos no trânsito, assim como também se apresenta uma população menor nestas faixas etárias, o que faz com que o índice fique maior. Como comparação, o número absoluto de mortes na população de 20 a 29 anos é cerca de 11 mil, enquanto este número atinge pouco mais de 3 mil na população acima de 70 anos. Tais dados podem ser verificados na figura 6.

Figura 6 – Mortes em acidentes de trânsito segundo a idade em 2011

Faixa etária	N	%	População	Taxa
0 a 4 anos	518	1,2	13.528.994	3,8
5 a 9	527	1,2	14.790.090	3,6
10 a 14	748	1,7	17.116.049	4,4
15 a 19	3.575	8,3	16.866.105	21,2
20 a 29	11.049	25,5	34.755.067	31,8
30 a 39	8.456	19,5	30.054.790	28,1
40 a 49	6.744	15,6	25.436.309	26,5
50 a 59	5.024	11,6	19.111.481	26,3
60 a 69	3.191	7,4	11.707.402	27,3
70 a 79	2.042	4,7	6.506.655	31,4
80 e +	1.015	2,3	3.070.373	33,1
Idade ignorada	367	0,8	–	*
Total	**43.256**	**100,0**	**192.943.315**	**22,4**

* Não calculada.

Fonte: Jorge (2013).

2.3 INFLUÊNCIA DO EXCESSO DE VELOCIDADE NOS ACIDENTES DE TRÂNSITO

A velocidade dos veículos está no centro do problema dos acidentes de trânsito. É preciso determinar que “excesso de velocidade” se refere à um veículo

excedendo o limite de velocidade previsto pela lei e “velocidade inapropriada’ se refere a um veículo viajando em uma velocidade inadequada para as condições da estrada e do trânsito. A lei determina o limite superior da velocidade trafegável, enquanto a velocidade adequada à via cabe ao usuário determinar (PEDEN et al, 2004, tradução nossa).

Segundo Peden et al (2004, tradução nossa) diversos fatores influenciam na velocidade que os motoristas trafegam, fatores estes relacionados com:

a) estrada: largura, alinhamento, elementos ao redor, desenho, marcações e qualidade da superfície;
b) veículo: tipo, relação peso/potência, velocidade e conforto;
c) trânsito: densidade, composição e velocidade média;
d) ambiente: condições do tempo, da superfície, luz natural e artificial, sinalização, limite de velocidade e fiscalização;
e) motorista: idade, sexo, tempo de reação, atitude, aceitação dos riscos, nível de álcool no sangue, propriedade do veículo, razão a viagem e ocupantes do veículo.

O risco de se envolver em um acidente assim como a gravidade do mesmo aumenta conforme a velocidade média do trânsito aumenta. A chance de morte e de lesões sérias aumenta em velocidades mais altas, especialmente para os chamados usuários vulneráveis, que são pedestres, ciclistas e motociclistas (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

Segundo Pedent et al (2004, tradução nossa) há muitos estudos de que a velocidade está diretamente ligada com o risco de acidentes e com a gravidade dos mesmos, dos quais pode-se destacar alguns abaixo:

a) a probabilidade de ferimentos em um acidente é proporcional ao quadrado da velocidade da batida, enquanto a probabilidade de mortes é proporcional à quarta potência da velocidade;
b) em estradas rurais com o limite de velocidade estabelecido em 60 km/h comprovou-se que a cada 5 km/h a mais na velocidade dobra a chance de se envolver em um acidente dobra.
c) a probabilidade de lesões fatais cresce de 0 até quase 100% conforme a velocidade aumenta de 20 até 100 km/h;

d) a chance de morte em um acidente à 80 km/h é 20 vezes maior do que em um acidente à 32 km/h;
e) em um atropelamento, um pedestre tem 90% de chance de sobreviver se o carro estiver à 30 km/h ou menos. Esta chance cai para 50% se a velocidade for igual ou maior que 45 km/h.

Segundo World Health Organization (2013, tradução nossa) intervenções para reduzir a velocidade podem trazer uma significante redução no número de acidentes de trânsito. Nas áreas urbanas, que concentram grande quantidade destes usuários vulneráveis, medidas para reduzir a velocidade são críticas para aumentar a segurança dos mesmos. O excesso de velocidade é um problema global que atinge toda a malha viária, desde estradas urbanas até rodovias e estradas rurais. Geralmente em países desenvolvidos, define-se alguns limites de velocidade de acordo com o tipo de estrada. São elas:

a) rodovias de alta velocidade: com mais de 1 pista, mãos separadas por barreiras e divisão entre tráfego motorizado e não motorizado. Tem limites de velocidade definidos entre 90 e 130 km/h;
b) estradas rurais: com 1 pista apenas em áreas rurais, incluindo tipos diferentes de superfície. Tem limites de velocidades definidos entre 70 e 100 km/h;
c) estradas urbanas: estradas nas cidades compartilhadas por automóveis, ciclistas, pedestres e transporte público. Estas estradas costumam ter limites de 50 km/h, porém evidências mostram que limites de 30 km/h são mais seguras para áreas com grande concentração de pedestres.

Um transporte fácil, rápido e de relativo baixo custo é importante para as pessoas tanto para uso profissional quanto para uso pessoal, assim como para a economia de um país. A mobilidade deste transporte deve ser equilibrada com a segurança, e o gerenciamento dos limites de velocidade deve prover este equilíbrio. O Gerenciamento da velocidade é um dos fatores do conceito de sistema seguro para o trânsito, sistema este que tem mostrado resultados em países como a Suécia. Este sistema denota que todos os elementos envolvidos no trânsito devem ser melhorados à fim de suprimir o erro humano. Criar um trânsito onde há limites de

velocidade segura, estradas seguras com elementos laterais à fim de proteger acidentes e carros mais seguros auxiliará na proteção dos usuários do mesmo (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

2.4 COMPORTAMENTOS RELACIONADOS COM O EXCESSO DE VELOCIDADE

O automóvel é um fenômeno mundial que ganhou força após a segunda guerra mundial. O mesmo torna-se símbolo de status social, prosperidade e independência, influenciado fortemente pela propaganda das economias capitalistas. O ato de conduzir um veículo automotor envolve muito mais que um processo mecânico no qual o motorista se desloca de um ponto a outro. As grandes velocidades que os veículos modernos atingem acabam por dar ao motorista uma sensação de poder e independência, isolando-o dos outros motoristas e demais usuários do trânsito. Adolescentes e adultos imaturos passam a compensar seu ego utilizando o veículo de forma arriscada, também influenciados pela publicidade que costuma associar altas velocidades à virilidade (MARÍN; QUEIROZ, 2000).

Segundo Thielen, Rtmann e Res (2008), um estudo realizado em 2008 em Curitiba abordou 36 motoristas à fim de aplicar questionamentos sobre percepção de excesso de velocidade. Destes, 16 não tinham multas e 20 deles tinha mais de 9 multas por excesso de velocidade. Salienta-se que os 16 motoristas não multados foram selecionados utilizando as mesmas variáveis de sexo e idade daqueles selecionados que tiveram multas. Ao questionar os entrevistados sobre o que é excesso de velocidade, as respostas dividiram-se em três categorias:

a) andar acima da velocidade estabelecida pela via;
b) andar em uma velocidade incompatível com a via, apresentando certas distorções pessoais;
c) andar em uma velocidade incompatível baseada totalmente em percepções pessoais.

Os motoristas da segunda categoria consideram as velocidades estabelecidas pela via compatíveis, porém têm definições pessoais de que a velocidade que trafegam não deveria ser considerado excesso ou que a via poderia tolerar uma velocidade maior. A terceira categoria contraria o texto da lei e considera

excesso de velocidade o que está além do que o motorista considera uma velocidade compatível. Estes motoristas consideram a confiança em si e no carro como fatores para determinar o que é excesso de velocidade. Destaca-se que a predominância dos discursos dos motoristas se classificaram na segunda e terceira categoria, sendo apenas 2 motoristas do total que consideraram excesso de velocidade estritamente o que a lei diz sobre a via. A convivência diária no trânsito faz com que alguns motoristas se acostumem com os riscos que ele oferece e passem a negar tais riscos. A relação entre riscos percebidos pelo motorista e riscos reais fica distorcida, pois o motorista assume que os riscos são os mesmos independente da velocidade (THIELEN; RTMANN; RES, 2008).

Segundo World Health Organization (2008, tradução nossa), existem várias razões que fazem um motorista trafegar à uma velocidade mais alta. Um dos motivos é a percepção de que o tempo de viagem será reduzido e que assumir tais riscos trará recompensa no fim da mesma, caso o motorista ande acima da velocidade limite sem consequências. Motoristas também tendem a viajar a uma velocidade maior em casos de estar sob pressão, fator frequente em motoristas profissionais. Outros excedem a velocidade simplesmente pelo prazer e pela sensação de emoção ou realização. A grande maioria dos motoristas superestima suas habilidades na direção, o que causa a impressão de que podem exceder os limites de velocidade sem se expor à riscos maiores.

2.5 IMPACTOS GERADOS PELOS ACIDENTES DE TRÂNSITO

A economia de um país em geral e da família das vítimas é fortemente afetada pelos acidentes de trânsito. A população economicamente ativa, com idade jovem, é a mais afetada em países de baixa e média renda. Muitas famílias das quais um indivíduo foi vítima de acidente de trânsito tem sua renda prejudicada, seja por morte de uma pessoa que contribuía com a renda, com gastos em tratamento médico da vítima ou com os gastos vitalícios com a vítima que ficou com sequelas do acidente. Os impactos são estendidos à economia por meio de gastos em sistemas de saúde e sistemas de seguro, assim como o possível decréscimo em uma população economicamente ativa, como já citado. Os países em

desenvolvimento são mais afetados pelo problema, visto que o investimento em outras áreas deixa a segurança nas estradas em segundo plano. Estudos afirmam que em países de baixa e média renda, acidentes de trânsito causam uma perda de 5% no PIB. A média global é de 3%. O Instituto de Pesquisa Econômica Aplicada (IPEA) em 2005 apontou que no Brasil a perda no PIB devido aos acidentes de trânsito era de 1.2% (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

No Brasil, em 2006, o IPEA e a Associação Nacional de Transportes Públicos (ANTP) realizaram um estudo sobre os custos dos acidentes de trânsito. O resultado mostrou que em 2005 o número de acidentes ocorridos em rodovias federais superou os 100 mil e totalizou um custo de 6,5 bilhões de reais. 68% deste custo se referem às pessoas, incluindo fatores como cuidados em saúde e perda de produção, enquanto 31% dos custos foram relacionados aos veículos. Os custos em rodovias estaduais e municipais foram estimados em 14,1 bilhões de reais e 1,4 bilhão de reais, totalizando a espantosa cifra de 22 bilhões de reais gastos anualmente com acidentes de trânsito (BACCHIERII; BARROSI, 2011).

Avaliar os custos dos acidentes de trânsito para a sociedade é essencial para se estimar o real retorno de políticas e intervenções aplicadas com o intuito de reduzir o número e a gravidade dos acidentes de trânsito. Enquanto é possível estimar estes custos como perda de produtividade em termos econômicos, avaliar o total impacto do sofrimento e da perda de vidas por causa dos acidentes torna-se mais complicado. Métodos para mensurar em termos econômicos a relação entre o capital e as consequências diretas de acidentes de trânsito são utilizados neste caso. Alguns estudos medem o quanto as pessoas pagariam para reduzir o risco de lesões no trânsito, enquanto outros comparam a perda de possíveis ganhos econômicos com a perda de vidas no trânsito. (PEDEN et al, 2004, tradução nossa).

Os custos relacionados diretamente à lesões e mortes são os mais difíceis de se estimar, pois os custos de tratamentos e reabilitações costumam ser altíssimos e podem se estender por tempo indeterminado (PEDEN et al, 2004, tradução nossa).

As lesões sofridas pelas vítimas do trânsito variam em natureza e gravidade. Estudos apontam que os acidentes de trânsito são a causa principal de traumas no cérebro. Isto se aplica tanto em países desenvolvidos quanto em países

em desenvolvimento. Em países de baixa e média renda apurou-se que os acidentes de trânsito são os grandes responsáveis por internação por traumas. Os índices variam de 30 a 86%, dependendo do país. Examinando o tempo de permanência média nos hospitais foi de 20 dias. Como exemplo, nos Estados Unidos em 2000 5 milhões 270 mil pessoas sofreram lesões não fatais no trânsito, dos quais 87% se enquadram em lesões leves. Estas lesões tiveram um custo total de 31,7 bilhões de dólares. (PEDEN et al, 2004, tradução nossa).

Estimar ou levantar exatamente todos os custos gerados pelos acidentes de trânsito é uma tarefa extremamente complexa, visto que as análises existentes sobre impactos socioeconômicos e na saúde não consideram totalmente os problemas psicossociais como a dor e o sofrimento gerados. As vítimas dos acidentes sofrem dores físicas e emocionais que não podem ser revertidas por compensações monetárias. Sequelas permanentes como paraplegia, tetraplegia, perda de visão ou danos cerebrais afetam a habilidade de realizar atividades simples e geram dependência física e econômica (PEDEN et al, 2004, tradução nossa).

Um estudo na Suécia mostrou que complicações psicossociais são frequentes em vítimas de acidentes de trânsito, mesmo em caso de lesões leves. Cerca da metade dos entrevistados relatou ansiedade ao viajar novamente mesmo dois anos após o acidente. Medo, dor e fatiga também foram comuns enquanto 16% dos que estavam empregados não retornaram aos seus empregos. Além das vítimas, as suas famílias também sofrem mudanças. A mudança mais relatada foi a realocação de um membro da família para ajudar a vítima a chegar ao trabalho, o que pode causar indiretamente uma perda de renda. Em alguns casos onde um membro da família foi vítima de acidente de trânsito, as crianças até se afastaram da escola. (PEDEN et al, 2004, tradução nossa).

3 FISCALIZAÇÃO DOS LIMITES DE VELOCIDADE

Segundo Peden et al (2004, tradução nossa), uma fiscalização forte é essencial para melhorar a segurança no trânsito. Os principais meios de fiscalização são feitos por meio de tecnologia de radares com câmeras. Para que se tenha uma fiscalização efetiva, alguns pontos devem ser observados:

a) os efeitos da fiscalização devem ser mantidos por um período, assegurando de que o risco de levar uma multa permaneça alto;
b) as punições daqueles que são flagrados pela fiscalização devem ser rápidas e eficientes;
c) a utilização de diferentes estratégias de fiscalização em diferentes locais selecionados aumenta a efetividade da mesma;
d) a fiscalização eletrônica é a que traz maior custo benefício.
e) a publicidade deve ser utilizada orientando e frisando a presença de fiscalização.

Segundo Cannell e Gold (2001), a fiscalização eletrônica reduz o número de acidentes em aproximadamente 30% e o número de mortes em aproximadamente 60%, tanto entre ocupantes de veículos e pedestres na área urbana. Nas rodovias, a instalação de radares fixos em pontos críticos reduz as mortes em aproximadamente 40%.

Gerir a velocidade nas vias visa reduzir o número de acidentes de trânsito incluindo medidas de engenharia, educação e principalmente fiscalização. A relevância da fiscalização está diretamente ligada com o seu alcance e com a severidade das sanções aplicadas aos condutores que excedem a velocidade (WORLD HEALTH ORGANIZATION, 2008).

Uma análise da fiscalização da velocidade em estradas rurais mostrou que a fiscalização estacionária composta por policiais em um determinado ponto parando os veículos reduziu o índice de acidentes em 6%, tanto no caso de acidentes fatais quanto não fatais. Quando combinado a estratégia da fiscalização estacionária com o uso de radares ou instrumentos que medem a velocidade média entre dois pontos, o índice de acidentes fatais teve uma redução de 14%. Nos

acidentes não fatais, a redução se manteve em 6% (PEDEN et al, 2004, tradução nossa).

A percepção do nível de fiscalização presente em uma via irá impactar no comportamento do usuário infrator. A presença da fiscalização deve ser divulgada à fim de educar os motoristas, ao invés de criar o aspecto de que a mesma foi feita para flagrar o usuário e gerar receita para os órgãos públicos, porém muitos usuários rapidamente percebem se a divulgação da fiscalização é exagerada e passam a não respeitar os limites estabelecidos. Uma desvantagem percebida se mostra quando há realmente uma fiscalização presente e ela é divulgada, fazendo com que muitos condutores reduzam a velocidade apenas até passarem pela fiscalização, acelerando novamente depois (WORLD HEALTH ORGANIZATION, 2008).

A utilização da fiscalização divulgada ou percebida produz um efeito *Halo* em relação ao tempo e à distância ao redor do local onde é instalado o radar fixo ou onde existe um posto da polícia ou uma viatura fiscalizando a velocidade. Este efeito é definido como a extensão do efeito da redução da velocidade por uma determinada distância anterior e posterior no caso do radar fixo ou posto de polícia assim como em relação ao tempo no caso de uma viatura fiscalizadora em um ponto aleatório da via. Estudos apontam que este efeito pode se estender por uma distância de até 22 quilômetros (km) em torno de um ponto de fiscalização fixo e por um período de até 10 semanas em torno de um ponto de fiscalização móvel (ELVIK et al, 2009, tradução nossa).

3.1 FISCALIZAÇÃO DOS LIMITES DE VELOCIDADE NO MUNDO

Um estudo realizado no Canadá mostrou que a fiscalização de trânsito reduz o número de acidentes fatais em países com um número grande de veículos. Por outro lado, uma fiscalização ineficiente pode contribuir para o efeito contrário. Estima-se que se as estratégias de fiscalização fossem aplicadas em toda união europeia, poderia se reduzir pela metade o número de mortes e lesões sérias causadas por acidentes de trânsito (PEDEN et al, 2004, tradução nossa).

Segundo o relatório *Global Status Report on Safety* feito em 2013, dos 195 países do qual se levantou dados relacionados à acidentes de trânsito, apenas

26 deles classificaram a fiscalização dos limites de velocidade com uma nota acima de 8 em uma escala de 0 a 10. Mesmo em países desenvolvidos que se destacam por aplicarem mais recursos na segurança no trânsito reportaram um índice baixo: Apenas 20% destes classificaram a fiscalização com nota 8 ou mais. Estes dados mostram que a atenção dada à fiscalização está aquém do problema real (WORLD HEALTH ORGANIZATION, 2013, tradução nossa).

A efetividade da limitação e fiscalização dos limites de velocidades já foi comprovada e pode ser observada na tabela 1 (PEDEN et al, 2004, tradução nossa).

Tabela 1 – Efeitos da redução dos limites de velocidade nos países

Exemplos dos efeitos da mudança dos limites de velocidade					
Data	Países	Tipo de estrada	Mudança na velocidade	Efeito na mudança da velocidade	Efeito da mudança no número de mortes
1985	Suiça	Auto-estradas	130 km/h para 120 km/h	velocidade média caiu 5 km/h	12% de redução
1985	Suiça	Estradas rurais	100 km/h para 80 km/h	velocidade média caiu 10 km/h	6% de redução
1985	Dinamarca	Estradas urbanas	60 km/h para 50 km/h	velocidade médica caiu 3-4 km/h	24% de redução
1987	Estados Unid	Interestaduais	55 milhas para 65 milhas	aumento de 2 a 4 milhas por hora na velocidade média	de 19 a 34% de aumento
1989	Suécia	Auto-estradas	110 km/h para 90 km/h	redução de 14.4 km/h nas velocidades médias	21% de redução

Fonte: Adaptado de Peden et al (2004).

O efeito de redução também ocorre nas estradas rurais. Uma experiência realizada na Tasmânia e Austrália utilizou um único veículo policial em estradas rurais de alto risco, consistindo em uma fiscalização estacionária. Esta estratégia resultou em uma redução de 3.6 km/h na velocidade média da via e uma redução de 58% dos acidentes graves, incluindo acidentes fatais e não fatais com lesões graves, onde houve necessidade de internação. O uso de fiscalização eletrônica dos limites de velocidade traz índices de redução expressivos no número de acidentes. Estes benefícios podem ser observados na tabela 2 (PEDEN et al, 2004, tradução nossa).

Tabela 2 – Efeitos da fiscalização eletrônica dos limites de velocidade nos países

Benefícios estimados de câmeras de velocidade		
País ou área	Benefícios de redução de acidentes em um nível de sistema	Benefícios de em locais indivduais do acidente
Austrália	22% de redução em todos os acidentes em New South Wales 30% de redução em todos os acidentes em estradas arteriais em Victoria 34% de redução de acidentes fatais em Queensland	
Nova Zelândia		11% de redução em acidentes e 20% de redução de vítimas durante os testes de câmeras ocultas de velocidade
República da Coréia		28% de redução em acidentes e 60% de redução em mortes em locais de alto risco
Reino Unido		35% de redução de mortes e lesões graves em acidentes de trânsito e 56% de redução de pedestres mortos ou com lesões graves no local da câmera
Europa Países Variados	50% de redução em todos os acidentes 17% de redução em acidentes com vítimas 28% de redução em todos os acidentes em áreas urbanas 4% na redução de todos os acidentes em áreas rurais	

Fonte: Adaptado de Peden et al (2004).

3.2 FISCALIZAÇÃO DOS LIMITES DE VELOCIDADE NO BRASIL

Segundo Cannell e Gold (2001), o Brasil é destaque pela violência no trânsito e desrespeito às normas de trânsito e a adoção da fiscalização eletrônica está se mostrando essencial para modificar o comportamento dos motoristas, à fim de reduzir o número e a severidade dos acidentes. Nos últimos anos, a disseminação da fiscalização eletrônica no Brasil tornou-se uma das melhores experiências de implantação da fiscalização eletrônica no mundo. O atual Código de Trânsito Brasileiro (CTB) data de 1998 e determina que o Conselho Nacional de Trânsito (Contran) deve normatizar os equipamentos de fiscalização eletrônica. O mesmo também previu que os munícipios poderiam criar seus próprios Órgãos Executivos Municipais de Trânsito (OEM). O CTB determinou os seguintes limites de velocidade:

a) 80 km/h nas vias de trânsito rápido;
b) 60 km/h nas vias arteriais;
c) 40 km/h nas vias coletoras;
d) 30 km/h nas vias locais;
e) 110 km/h para automóveis e camionetas em rodovias;
f) 90 km/h para ônibus e micro-ônibus em rodovias;
g) 80 km/h para os demais veículos em rodovias.

Nos relatórios anuais da ONU que medem a eficácia das medias

estabelecidas para a década de segurança no trânsito, o Brasil classificou a sua fiscalização do excesso de velocidade como nota 6 em 2013 (WORLD HEALTH ORGANIZATION, 2013, tradução nossa) e nota 7 em 2015 (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

Segundo Cannell e Gold (2001), grande parte do sucesso da implantação da fiscalização eletrônica se deu nos municípios. Em São Paulo, a implantação da fiscalização eletrônica iniciou em 1997 utilizando 40 radares fixos e 31 barreiras eletrônicas. O impacto da fiscalização eletrônica foi medido comparando-se os dados de 1996 e 1998, no qual foi possível observar alguns pontos:

a) o desrespeito aos limites de velocidade caiu drasticamente nos locais onde o sistema de fiscalização foi implantado em conjunto com o novo CTB;
b) os números de vítimas fatais e não fatais diminuíram, respectivamente, 31% e 22%;
c) a economia de leitos hospitalares traduziu-se em 150 milhões de dólares;
d) nas três principais rodovias de São Paulo (Marginal Tietê, Marginal Pinheiros e Av. 23 de Maio) as mortes diminuíram em média 59%.

Há casos em que a implantação da fiscalização eletrônica em alguns munícipios não obteve tanto êxito quanto em outros. Em Cascavel, no interior do Paraná, foram implantadas lombadas eletrônicas a partir de 2000, porém não houve divulgação da redução dos acidentes e a implantação coincidiu com uma fase pré-eleitoral. Rapidamente, a opinião pública considerou a fiscalização eletrônica como uma indústria de multas. Verificou-se que o sucesso da implantação da fiscalização eletrônica depende de alguns fatores, como a divulgação dos benefícios da mesma e a realização de campanhas educativas. (CANNELL; GOLD, 2001).

Nas rodovias, a fiscalização eletrônica também trouxe benefícios. Em 1994, em Santa Catarina foi contratada a operação de radares em modo de contrato de experiência, no qual todos os custos relativos à manutenção e instalação era de responsabilidade a empresa contratada. Os radares eletrônicos foram ativados em 1995. Houve grande pressão política e por parte dos usuários contra o sistema,

contando com ações individuais na justiça, o que levou o Tribunal de Justiça de Santa Catarina (TJ/SC) a suspender o contrato em julho de 1995. O Estado de Santa Catarina recorreu ao Superior Tribunal de Justiça em Brasília (STJ) e o STJ cassou a liminar do TJ/SC, destacando no seu manifesto: “Nesses dez meses que se passaram, tivessem os radares, ou seus efeitos, poupado a vida de apenas um catarinense, já se justificaria a sua manutenção, uma vez que a vida não tem preço.”. Os radares foram reativados até o fim de 1995 e removidos no início de 1996, quando se abriu o processo de licitação para contratação definitiva da fiscalização eletrônica. Os radares ficaram ativos por 6 meses em 1995 e em comparação com os mesmos 6 meses de 1994, a redução nos números de vítimas não fatais e fatais foi, respectivamente, de 6% e 42%. Nos 6 meses posteriores de 1995, quando os radares foram desativados por conta da liminar do TJ/SC ficou claro a eficiência dos mesmos: Houve aumento de 10% no número de feridos e de 54% no número de mortos se comparado com os mesmos 6 meses do ano de 1994. Com a instalação dos radares novamente em 1996 por meio do contrato licitatórios, os índices de feridos e mortos voltou a cair em relação à 1995 na ordem de 1 e 31%, respectivamente (CANNELL, 2000).

3.3 MÉTODOS DE FISCALIZAÇÃO DOS LIMITES DE VELOCIDADE

A fiscalização estacionária de velocidade feita por agentes de trânsito acabou se tornando um método difícil e de baixo custo benefício, o que abriu espaço para o desenvolvimento da fiscalização eletrônica da velocidade, à fim de medir a velocidade de forma automática e confiável. A partir de então, desenvolveu-se os radares. O termo radar significa *RAdio Detection And Ranging* e foi utilizado pela primeira vez para fiscalizar a velocidade dos veículos em 1947 nos Estados Unidos. Os primeiros radares utilizavam ondas eletromagnéticas e 1990 surgiram os radares a laser, que utilizam a banda superior do infravermelho, que eram conhecidos como Laser Detection and Ranging (ladar) ou Light Detection And Ranging (lidar). O termo radar acabou se generalizando para todos os tipos de equipamento (MING, 2006).

A fiscalização eletrônica dos limites de velocidade como os radares com câmeras é utilizada em muitos países e tem eficácia comprovada, pois a mesma

produz provas do excesso de velocidade cometido que podem serem utilizados em um possível processo. Este tipo de fiscalização é posicionado em lugares estratégicos onde há constante desobediência dos limites de velocidades estabelecidos pela via e altos riscos de acidentes (PEDEN et al, 2004, tradução nossa).

A fiscalização eletrônica de trânsito se divide em duas categorias: Aplicações metrológicas que se refere à fiscalização da velocidade e Aplicações não metrológicas que se refere à fiscalização de eventos, como avanço de sinal e invasão da faixa de pedestre. Classifica-se os equipamentos de fiscalização eletrônica de velocidade como: Radar fixo, radar estático e lombada ou barreira eletrônica (MING, 2006).

O método de detecção de velocidade é o mesmo no caso do radar fixo e da lombada eletrônica: dois ou três laços indutivos são instalados no pavimento e assim que o veículo passa por eles, os mesmos são acionados. Independentemente da quantidade de laços, a velocidade do veículo é sempre obtida pela relação distância sobre tempo, no qual um cronômetro é iniciado no laço inicial e travado no laço final. Para identificação dos infratores, uma câmera instalada faz a captura das imagens. A diferença entre a lombada eletrônica e o radar consiste na instalação de um painel com um display que exibe a velocidade registrada no caso da lombada eletrônica, sendo que a mesma é geralmente instalada em vias de velocidade mais baixa. Estes equipamentos possuem comunicação ADSL ou via rádio comunicação para transmissão dos dados e configuração. O radar fixo e a lombada eletrônica podem ser observados nas figuras 7 e 8 (MING, 2006).

Figura 7 – Radar Fixo

Fonte: ViaEPTV.com.

Figura 8 – Lombada Eletrônica

Fonte: AUTOentusiastas.

O radar estático difere dos anteriores por ser um equipamento portátil e fixado em um tripé. Este tipo de radar classifica-se como estático por manter-se em uma posição enquanto está em operação, ao contrário dos equipamentos móveis do tipo pistola que podem ser utilizados em um veículo em movimento. Estes equipamentos utilizam as duas tecnologias já mencionadas para radares: Emissão de ondas de rádio e de luz. No caso da emissão de ondas de rádio, a medição da velocidade é feita pela diferença da frequência emitida e da frequência refletida quando direcionada ao veículo, diferença esta causada pelo efeito Doppler. No caso da emissão de luz, a velocidade do veículo é determinada pela diferença do tempo em que um pulso de luz leva para ir do radar ao veículo e retornar. O radar estático e o radar do tipo pistola podem ser observados nas figuras 9 e 10 (MING, 2006).

Figura 9 – Radar Estático

Fonte: Canal RioClaro.

Figura 10 – Radar Móvel

Fonte: Jornal Ponto Inicial.

Além de utilizar câmeras para fotografar os veículos infratores, há ainda radares do tipo LAP que utiliza a tecnologia OCR, utilizados também para identificação de outras infrações identificáveis pela placa do veículo, como licenciamento atrasado e restrição de circulação, por exemplo. São Paulo, cidade conhecida por congestionamentos, implantou o rodízio de veículos para restringir veículos em determinados dias da semana considerando-se a placa do mesmo. Entre julho de 2005 e julho de 2006 foi realizado um experimento na cidade com um equipamento do tipo LAP para identificação de infrações relacionadas com o rodízio, no qual resultou em um total de 8542 notificações. A introdução deste sistema de identificação de placas abriu possibilidade para otimização na fiscalização eletrônica,

pois um mesmo equipamento pode registrar diferentes infrações com uma mesma infraestrutura (MING, 2006).

A fiscalização efetiva deve combinar métodos para cobrir vários pontos em uma ou mais vias. A implantação de uma fiscalização em um ponto da via apenas pode levar o condutor a reduzir a velocidade apenas naquele ponto. Abordagens em lugares e horários diferentes traz a sensação de que o condutor está sendo monitorado em qualquer lugar que seja, o que resulta na redução da velocidade em geral. O método definitivo de fiscalização deve considerar a relação distância sobre tempo, o qual fornece a velocidade média do condutor. Este método é empregado atualmente com o acompanhamento do veículo em alta velocidade por uma viatura, no qual é observado o odômetro da viatura e o tempo de acompanhamento para determinar a velocidade média que o condutor teve no trecho. Desta forma, é possível avaliar de forma ideal a velocidade do condutor, eliminando possíveis justificativas como o excesso de velocidade para ultrapassagem ou por pouco tempo e o comportamento de determinados motoristas de reduzir a velocidade apenas onde há a fiscalização. Em níveis básicos, pode-se adotar até mesmo cronômetros em dois pontos de distância conhecida de uma via para medir a velocidade média (WORLD HEALTH ORGANIZATION, 2008).

4 PLATAFORMA RASPBERRY PI

Em 2006 na Universidade de Cambridge na Inglaterra, percebeu-se uma deficiência no entendimento sobre o que era um computador e como o mesmo funcionava. Desenvolveu-se um projeto de um computador de baixo custo destinado a experiências inclusive feitas por crianças nas suas casas, desenvolvendo seus interesses pela Ciência da Computação. Em fevereiro de 2012, o primeiro lote de 10 mil computadores RPI foi vendido em poucos minutos. Ao fim do ano, mais de 500 mil unidades foram vendidas (GOLDEN, 2013, tradução nossa).

O RPI é um computador como qualquer outro, com as principais diferenças de que o mesmo utiliza um processador de arquitetura ARM e não aceita a instalação do Microsoft Windows, porém várias distribuições do Linux podem ser usadas nele. De fácil utilização, poderoso e de baixo custo, o RPI é um computador ideal para cientistas da computação. O sistema operacional padrão do RPI é uma versão do Linux com uma interface gráfica, porém a maneira mais utilizada de interação com o mesmo é através da interface de linha de comando do inglês *command line interface* (CLI) (FOUNDATION, 2012, tradução nossa).

Uma estratégia adotada no RPI que diminui o custo e o consumo de energia do mesmo é a adoção da tecnologia sistema-em-um-chip (SoC, do inglês *System on a Chip*) que fisicamente coloca o processador, a memória e a central de processamento gráfico no mesmo lugar, um sobre o outro, economizando também espaço na placa (NORRIS, 2014, tradução nossa).

Os principais atributos do RPI são o tamanho semelhante à um cartão de crédito e o preço de 35 dólares. O mesmo classifica-se como um SBC (GOLDEN, 2013, tradução nossa).

O chip utilizado é o Broadcom BCM2835, este incluindo o processador ARM1176JZF-S com frequência de clock de 700 megahertz (MHz) e também a central de processamento de gráficos (GPU, do inglês *Graphic Processing Unit*) Broadcom VideoCore® IV GPU. Este chip é projetado principalmente para sistemas móveis que necessitam de baixo consumo e aquecimento, o que explica a baixa frequência de clock operante. A GPU suporta vídeos em 1080p em 30 quadros por segundo e gráficos tridimensionais. O modelo B tem 512 megabytes (MB) de

memória dinâmica de acesso randômico (DRAM, do inglês *Dynamic Random Access Memory*), sendo que esta é a camada superior do SoC presente no centro da placa. O RPI também utiliza um cartão *Secure Digital* (SD) de memória flash como dispositivo de armazenamento, sendo que é recomendável utilizar um cartão de classe 4, no mínimo. A classe do cartão define a taxa máxima de transferência, no qual um cartão de classe 4 possui uma taxa de transferência máxima de 4 MB por segundo. Há cartões até de classe 10, que possui uma performance ainda maior, porém à maiores custos (NORRIS, 2014, tradução nossa).

É possível pensar no RPI como uma placa de desenvolvimento para microcontrolador como o Arduíno, porém considerando os componentes do mesmo, ele é mais parecido com um celular com vários conectores acessíveis. Até mesmo o processador utilizado é o mesmo do iPhone 3G e do Kindle 2. Além disso, o mesmo possui as interfaces descritas abaixo (RICHARDSON; WALLACE, 2013):

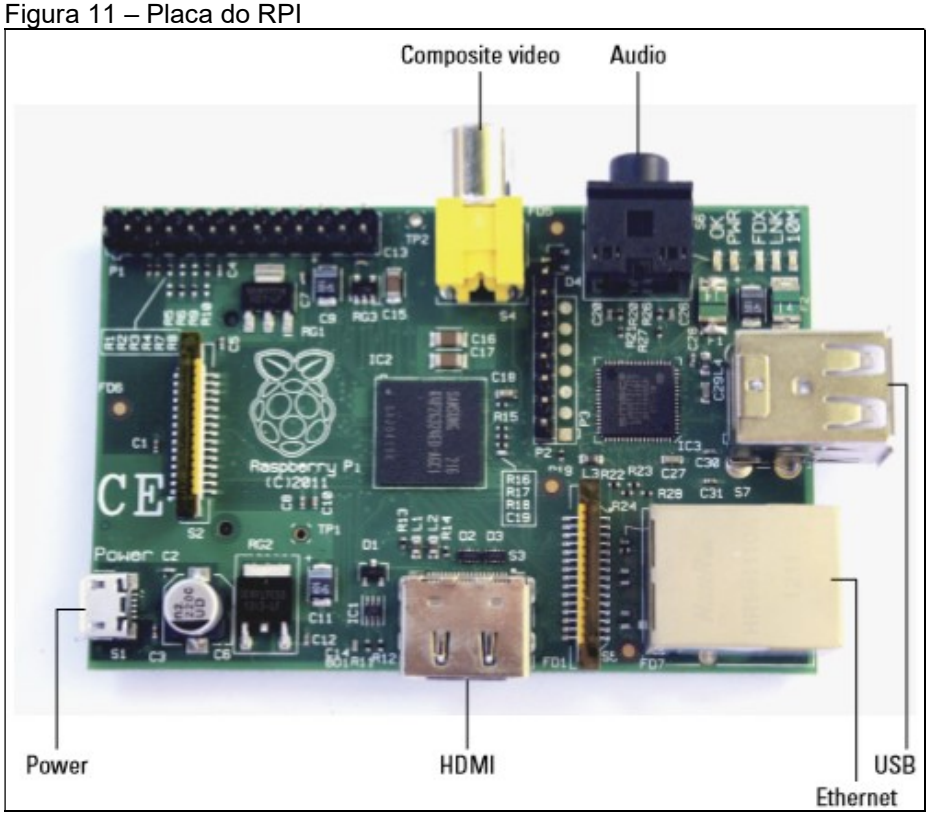
a) slot para cartão de memória SD: o RPI não possui disco rígido, sendo assim o sistema operacional deve ser instalado em um cartão de memória;
b) porta *Universal Serial Bus* (USB): os primeiros RPI tiveram a corrente limitada nas portas USB. Enquanto alguns dispositivos USB podem atingir 500 miliámperes (mA), a corrente era limitada à 100 mA;
c) porta Ethernet: há uma porta Ethernet padrão RJ45, que na verdade é um adaptador Ethernet USB embutido. Ela está presente apenas no modelo B;
d) porta *High-Definition Multimedia Interface* (HDMI): a porta HDMI fornece saída de áudio e vídeo digital em 14 resoluções diferentes;
e) saída de áudio analógico: conector de áudio padrão de 3.5 milímetros;
f) saída de vídeo composto: conector padrão do tipo RCA que fornece sinais de vídeo composto NTSC ou PAL;
g) entrada de energia: não há interruptor de alimentação no RPI, apenas uma porta micro USB para alimentação de 5V;
h) pinos de entrada e saída de uso geral (GPIO, do inglês *General-Purpose Input/Output)*: pinos para leitura de valores de circuitos eletrônicos externos;

i) conector de Interface Serial do Display (DSI, do inglês *Display Serial Interface*): conector que recebe um cabo fita de 15 pinos para comunicar-se com uma tela de cristal líquido;

j) conector de Interface Serial da Câmera (CSI, do inglês *Camera Serial Interface*): este conector permite a conexão de um módulo de câmera.

A disposição dos componentes e interfaces na placa do RPI modelo B pode ser observado na figura 11.

Figura 11 – Placa do RPI



Fonte: Kelly (2014).

Há dois modelos do RPI, com poucas diferenças entre eles. Além das diferenças já descritas nas interfaces, o modelo A, mais básico, contém 256 MB de memória, metade do modelo B. Enquanto o preço do modelo B é de 35 dólares, o modelo A custa 25 dólares. Apesar da incrível relação custo-performance do RPI, o mesmo possui algumas limitações. O mesmo se assemelha mais à um dispositivo móvel do que um computador de mesa moderno. A Raspberry Pi Foundation compara a performance do RPI à um computador com um processador Pentium 2 de 300 MHz, exceto pela parte gráfica, que é compatível com o Xbox de

primeira geração. O RPI é ideal como segunda máquina, destinada à experimentação (MCMANUS; COOK, 2013, tradução nossa).

O baixo preço do RPI foi um dos principais pilares para o desenvolvimento do mesmo, e para garantir isto a Raspberry Pi Foundation firmou acordos com alguns fabricantes para oferecer o RPI à preços fixos de 25 dólares para o modelo A e 35 dólares para o modelo B nos distribuidores para venda para o público e para revendedores (RICHARDSON; WALLACE, 2013).

O RPI 3 Modelo B é a versão mais nova do minicomputador que trouxe alguns avanços. O mesmo recebeu um processador Broadcom BCM2387 Quad-Core de 1.2 gigahertz (GHz) 10 vezes mais rápido que a primeira geração do RPI e 1 GB de memória RAM. Também foram adicionadas interfaces integradas de conectividade Wi-Fi e Bluetooth e mais duas portas USB, além da possibilidade de utilizar o sistema operacional Windows 10 (FOUNDATION, 2016B, tradução nossa).

4.1 ARQUITETURA ARM

O desenvolvimento dos processadores nos últimos 50 anos permitiu um crescimento incrível na performance acompanhado de uma incrível redução de preço. Grande parte do avanço se deve à tecnologia empregada na construção dos processadores, que passou por válvulas, transistores, circuitos até chegar à atual tecnologia de integração de larga escala, combinando milhões de transistores em um único chip (FURBER, 2000, tradução nossa).

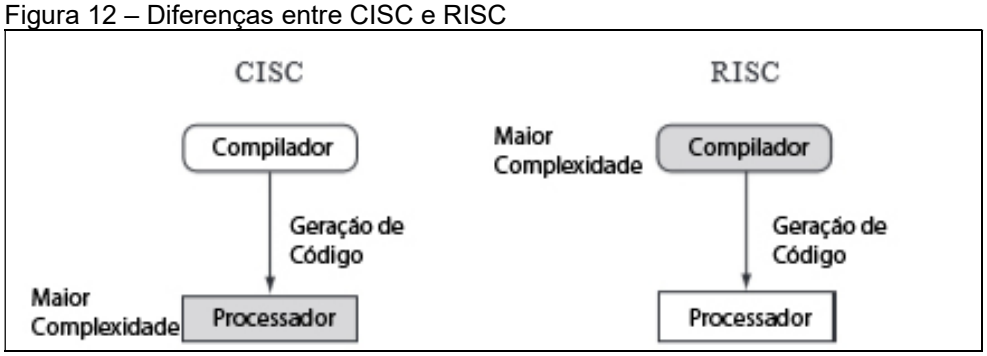
A explosão dos sistemas embarcados e móveis nos últimos anos tem como principal pilar a arquitetura ARM, que se tornou a arquitetura 32 bits com a maior participação no mercado. Processadores baseados na arquitetura ARM estão presentes desde smartphones até sistemas automotivos e sofre constantes inovações técnicas desde o primeiro protótipo do ARM1 de 1985. A arquitetura ARM compõe uma família de processadores com características semelhantes e um conjunto de instruções de computador reduzida (RISC, do inglês *Reduced Instruction Set Computer*) adaptada à fim de criar um processador flexível. As principais características destes chips são o baixo consumo de energia e tamanho reduzido, tornando-os ideais para sistemas embarcados e móveis (SLOSS; SYMES; WRIGHT,

2004, tradução nossa).

O desenvolvimento de um processador deve definir um conjunto de instruções para serem executadas pelas funções implementadas pelo programador, e esta ponte entre a programação de alto nível e as instruções de máquinas é feito pelo compilador. Até 1980, o método para oferecer mais possibilidades entre as operações da máquina e a programação de alto nível consistia em inserir mais instruções suportadas pelo processador, aumentando a complexidade. Este método era utilizado em minicomputadores e definiu o conjunto de instruções complexo de computadores mono chip (CISC, do inglês *Complex Instruction Set Computers*), que continha instruções executadas e muitos ciclos de processamento. Diante disto, foi desenvolvido o RISC que teve grande influência no desenho dos processadores ARM. Este conjunto de instruções teve como objetivo reduzir a complexidade entre as instruções de máquina e a linguagem de alto nível (FURBER, 2000, tradução nossa).

A arquitetura RISC foi desenvolvida para fornecer instruções simples e poderosas que sejam executadas em um ciclo de processamento. Ela objetiva também trazer a complexidade dos comandos do hardware para o software, visto que a flexibilidade é muito maior no software. Por outro lado, a arquitetura RISC demanda maior complexidade no compilador, que comunica a o hardware e o software. A figura 12 demonstra a diferença básica entre a arquitetura CISC e RISC (SLOSS; SYMES; WRIGHT, 2004, tradução nossa).

Figura 12 – Diferenças entre CISC e RISC



Fonte: Adaptado de Sloss, Symes e Wright (2004, tradução nossa).

O primeiro processador ARM foi desenvolvido em Cambridge, na

Inglaterra, entre 1983 e 1985 na *Acorn Computers Limited*. A empresa já era consagrada pelo sucesso de um microcomputador baseado em um microprocessador 6502 de 8 bits, utilizado largamente em escolas. A busca por sucessores para este produto foi falha, visto que os processadores de 16 bits baseados na arquitetura CISC em 1983 eram lentos e os engenheiros da empresa tinham instruções que levariam mais de um ciclo de processamento. O desenvolvimento de um microprocessador proprietário foi considerado, porém limitações da própria empresa impediram isto. Enquanto isto, estudantes de pós-graduação desenvolveram em um ano o Berkeley RISC I, um processador competitivo e simples sem utilizar instruções complexas. A arquitetura ARM se estabeleceu então como componente principal da *Acorn*. O significado da sigla ARM foi definido em 1990 como *Acorn Risc Machine* (FURBER, 2000, tradução nossa).

Alguns conceitos definiram o desenho dos processadores ARM. Um deles é que processadores ARM são desenhados para serem pequenos e terem baixo consumo de bateria. Um código otimizado também é extremamente importante, visto que sistemas embarcados possuem memória limitada por custos ou tamanho, sendo muitas vezes limitadas na capacidade e/ou velocidade. A arquitetura ARM incorporou também a tecnologia de depuração de hardware, permitindo aos engenheiros uma mais rápida resolução de problemas (SLOSS; SYMES; WRIGHT, 2004, tradução nossa):

4.2 MÓDULOS E ACESSÓRIOS

Como o RPI foi criado com o intuito de desenvolver projetos, há muitos módulos e acessórios que podem ser acoplados ao mesmo, alguns inclusive certificados pela Raspberry Pi Foundation (RICHARDSON; WALLACE, 2013).

Apesar de ser dotado de um processador projetado para aplicações móveis onde não é necessário um sistema de resfriamento, há casos em que o processamento do RPI é alto e ocasiona geração de calor, tornando necessário o uso de um dissipador de calor. O dissipador é um pequeno objeto de metal com aletas projetado para criar a maior área de superfície possível para dissipar o calor. Há casos também em que o chip de rede pode aquecer, onde também é possível

aplicar um dissipador (RICHARDSON; WALLACE, 2013).

A AdaFruit é uma das mais famosas na distribuição e fabricação de módulos para o RPI. A mesma possui telas LCD acopláveis ao RPI sensíveis ao toque. Estas telas têm resolução baixa para os padrões atuais, de 480 pixels de largura por 320 pixels de altura na versão de 3.5 polegadas, mas são suficientes para projetos de aplicações móveis. As mesmas utilizam os pinos GPIO. O modelo de 3.5 polegadas pode ser observado na figura 13 (ADA, 2016, tradução nossa).

Figura 13 – Display Adafruit 3,5 polegadas

Fonte: Ada (2016).

Exceto no RPI 3, não há conectividade Wi-Fi nativa nos outros modelos. Isto pode ser resolvido com adaptadores externos USB (RICHARDSON; WALLACE, 2013).

A Adafruit tem um adaptador Wi-Fi em miniatura que é ideal para projetos com o RPI, porém o mesmo deve ser utilizado apenas em casos em que o roteador está perto. Em casos em que o RPI fica em definitivo em lugares fechados, é recomendável utilizar o adaptador com antena. O mesmo pode ser visto na figura 14 (FARNELL, 2016, tradução nossa).

Figura 14 – Módulo de Wi-Fi

Fonte: Farnell (2016).

Um dos acessórios mais recomendados para o RPI é a adoção de um gabinete, pois há risco de dano em algum componente caso o mesmo fique exposto. Diferentemente de muitas placas de circuito impresso em que há trilhas apenas em um lado ou no máximo, nos dois lados da placa, o RPI contém 6 camadas de trilhas conectando os componentes, dessas 1 em cada lado da placa e 4 camadas prensadas entre a parte superior e inferior. Caso a placa seja flexionada, estas trilhas mais sensíveis podem ser danificadas. Há diversos modelos de gabinetes pré-fabricados e é possível até projetos disponíveis para download para fabricação em impressora 3D ou cortadora a laser. Alguns modelos de gabinete podem ser observados na figura 15 (RICHARDSON; WALLACE, 2013).

Figura 15 – Cases para RPI

Fonte: (GOLDEN, 2013).

4.2.1 Câmera

Um dos módulos mais interessantes do RPI é a placa com câmera projetado para o mesmo. A mesma conecta-se ao RPI pela porta CSI, não ocupando uma porta USB. Este módulo possui várias funções, como fotografias em *time-lapse*,

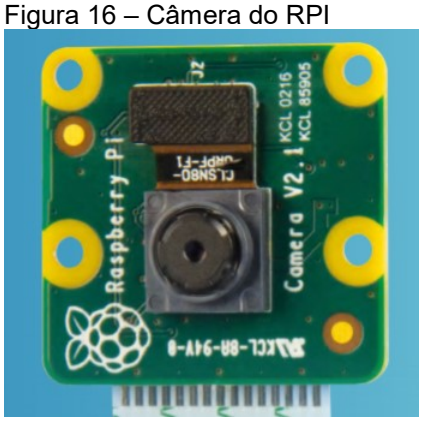
em que fotos são tiradas à uma frequência menor do que o vídeo que será reproduzido, trazendo a sensação de tempo acelerado. Da mesma forma, é possível capturar vídeos em *slow-motion*, processo inverso do descrito. (KELLY, 2014, tradução nossa).

O sensor da câmera é de 5 megapixels (MP) e faz vídeos em 30 quadros por segundo na resolução 1080p e em 60 quadros por segundo na resolução 720p. A mesma possui alguns controles de imagem como controle automático de exposição, balanço de branco e de preto (FOUNDATION, 2016, tradução nossa).

Desde o primeiro modelo, o RPI tinha o conector CSI para conectar a câmera à GPU do mesmo. Esta conexão utiliza o protocolo elétrico CSI-2 e é utilizada na maioria dos celulares e é extremamente rápida, capaz de transmitir imagens com resolução de até 1080p e 30 quadros por segundos. Desde a concepção, já havia a intenção de desenvolver um módulo de câmera para utilizar esta conexão de alta velocidade para interagir com a GPU sem ter que passar pelo processador, tornando a câmera muito mais eficiente do que qualquer câmera USB. Este módulo então foi lançado em 2013 (THE MAGPI, 2013, tradução nossa).

O módulo de câmera do RPI ganhou uma segunda versão e como tradição, o preço se manterá em 25 dólares. Após 3 anos do lançamento da primeira versão, a segunda versão foi desenvolvida para substituir a original da forma mais natural possível. A troca pelo sensor Sony IMX219 trouxe como principal melhoria a resolução máxima de 8 MP. O tamanho do pixel também diminui, tornando as fotos mais detalhadas. Outra tradição que foi mantida é a compatibilidade do módulo com as versões anteriores do RPI. O desenho do módulo continua similar à primeira versão, e pode ser visto na figura 16 (THE MAGPI, 2016, tradução nossa).

Figura 16 – Câmera do RPI



Fonte: The MagPi (2016).

4.3 SISTEMA OPERACIONAL

O sistema operacional utilizado pelo RPI é o Linux, porém apenas o núcleo (popularmente conhecido como *kernel*) pode ser considerado como Linux, pois toda a parte de coleção de drivers, serviços e aplicações divergem de acordo com a distribuição do Linux. Algumas das mais comuns são Ubuntu, Debian, Fedora e Arch. Como o RPI é mais parecido com um dispositivo móvel do que com um computador de mesa, os requisitos de software são diferentes principalmente devido ao tipo do processador e às limitações de armazenamento e memória do mesmo (RICHARDSON; WALLACE, 2013).

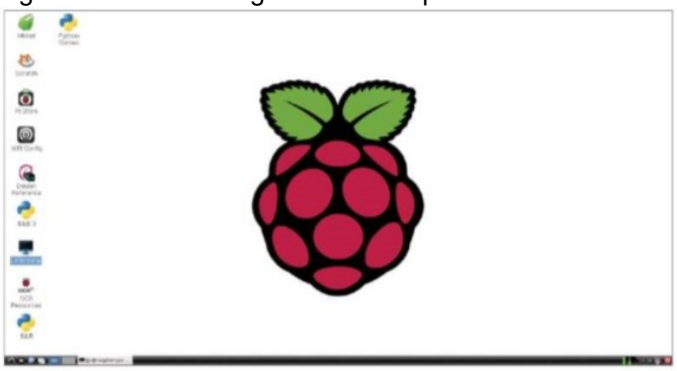
A principal diferença do Linux é que o mesmo grátis e de código aberto, requisitos básicos para o RPI, tornando o sistema operacional poderoso e personalizável. Isto é possível pelas grandes comunidades de usuários e desenvolvedores por trás das distribuições existentes. O RPI pode ser a porta de entrada de futuros programadores do Linux por causa do seu cunho educacional (KELLY, 2014, tradução nossa).

Enquanto o RPI pode ser o primeiro contato com o Linux que as pessoas terão, a história deste sistema operacional já é antiga. Em 1984 Richard Stallman criou o projeto GNU com o objetivo de construir um sistema operacional livre em que se pudesse copiar, estudar e modificar o mesmo sem custo algum. Em 1991 Linus Torvalds criou o *kernel*, componente principal do Linux responsável por integrar o software e os recursos de hardware como memória e processador (GOLDEN, 2013, tradução nossa).

Pelo fato do RPI ser baseado em um processador ARM, o número de distribuições do Linux é mais limitado, porém número está em constante mudança (KELLY, 2014, tradução nossa). Com comunidades fortes em torno do RPI e do Linux, há várias distribuições do Linux em variados estágios de desenvolvimento disponíveis (MCMANUS; COOK, 2013, tradução nossa).

A distribuição oficial e recomendada para o RPI é o Raspian, distribuição baseada no Debian e otimizado especialmente para o RPI. O mesmo foi desenvolvido para ser de fácil utilização e instalação, tendo uma comunidade muito forte. A interface gráfica do Raspian pode ser visto na figura 17 (KELLY, 2014, tradução nossa).

Figura 17 – Interface gráfica do Raspian



Fonte: Kelly (2014).

Muitas vezes, é possível encontrar cartões SD à venda já com o sistema operacional instalado, sendo esta a opção recomendada para iniciantes. Se não for este o caso, o Raspian possui um instalador em que é possível executar do próprio RPI apenas passando os arquivos para o cartão SD ou fazer o download da imagem do Raspian e utilizar um utilitário de imagem de disco para transferi-la para o cartão SD (RICHARDSON; WALLACE, 2013).

4.4 OUTRAS PLATAFORMAS DE PROTOTIPAGEM

O grande problema do desenvolvimento de novas aplicações envolvendo hardware sempre foram os custos e a complexidade da montagem de um laboratório. Diante deste problema surgiram as plataformas de prototipagem

eletrônica, também conhecidas como placas ou kits de desenvolvimento (FONSECA; LA VEGA, 2011).

Uma placa de desenvolvimento é uma placa única de hardware com um microcontrolador. Esta placa permite aos usuários utilizar dispositivos de entrada e saída conectadas para desenvolver tarefas programadas no microcontrolador. Algumas placas incluem microprocessadores tornando-as pequenos computadores (BOYD et al, 2015, tradução nossa).

4.4.1 Arduino

O projeto Arduino iniciou em 2005 e desde lá, o número de placas vendidas superou 150 mil unidades considerando apenas a placa oficial do Arduino. Existem também as placas-clone baseadas no Arduino e estima-se que a venda destas tenha passado das 500 mil unidades. A facilidade de aprendizagem do Arduino é um dos seus principais atributos, assim como o tamanho da sua comunidade, que disponibiliza códigos e projetos para todos (MCROBERTS, 2011).

O Arduino é uma plataforma eletrônica com software e hardware livres para criação de protótipos. O mesmo possui pinos de entrada dos quais ele pode obter dados do ambiente por meio de vários sensores disponíveis e também controlar dispositivos por meio dos seus pinos de saída. O mesmo possui linguagem e ambiente de desenvolvimento próprios (SHOP, 2014).

A mais versátil e popular versão do mesmo é o Arduino Uno, podendo ser visualizada na figura 18 (MCROBERTS, 2011).

Figura 18 – Arduino Uno

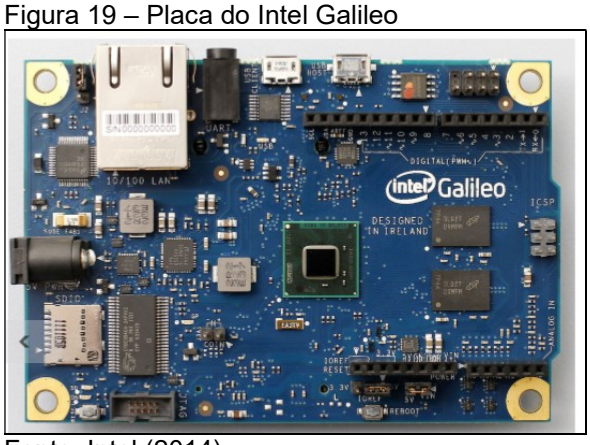
Fonte: McRoberts (2011).

4.4.2 Galileo

O Galileo é uma placa de desenvolvimento criada pela Intel que possui o hardware e o software compatíveis com o a plataforma Arduino. O Galileo utiliza o Linux como sistema operacional e também pode ser utilizada como um computador (BOYD et al, 2015, tradução nossa).

Trata-se da primeira placa de desenvolvimento com arquitetura Intel projetada para ser compatível com os *shields* do Arduino Uno. O processador utilizado é o Intel Quark SoC X1000 com arquitetura 32 bits baseado no Intel Pentium e a memória é de 256 MB. A placa do Galileo pode ser visualizada na figura 19 (INTEL, 2014, tradução nossa).

Figura 19 – Placa do Intel Galileo



Fonte: Intel (2014).

Podendo o Galileo ser utilizado como computador, ele tem várias interfaces similares ao RPI utilizadas para conectar periféricos como slot para cartão SD, portas USB e uma porta Ethernet para conexões de rede. A diferença é que ele não possui portas de vídeo de deve ser acessado remotamente por meio de outro computador (BOYD et al, 2015, tradução nossa).

4.4.3 BeagleBone

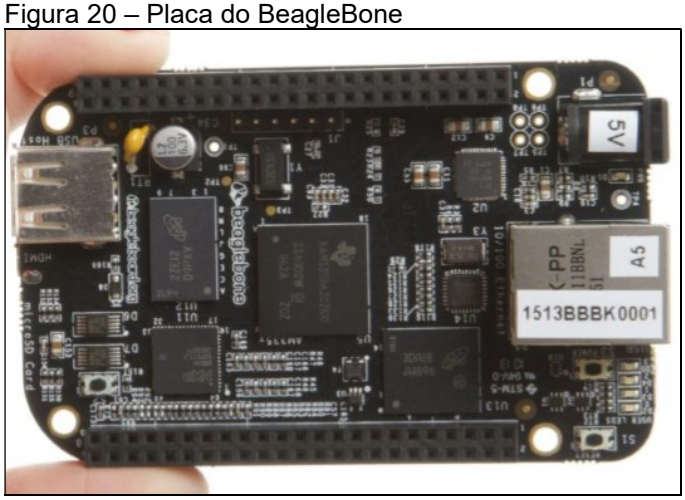
O projeto BeagleBoard.org nasceu com o objetivo de fornecer hardwares

livres para estudantes e entusiastas montarem sistemas com baixo custo e facilidade. A mais recente plataforma de desenvolvimento é a BeagleBone Black, uma placa com o preço de 45 dólares com um processador ARM Cortex-A8 de 1 GHz, 512 MB de memória e um diferencial visando a performance: um dispositivo de armazenamento integrado na placa do tipo eMMC, um tipo de memória flash parecida com o cartão SD (KRIDNER, 2013, tradução nossa).

O eMMC embutido na placa possui 4 GB de armazenamento na última versão da placa e é o dispositivo padrão de boot do computador, porém ainda há o slot MicroSD para utilização de um cartão com maior capacidade, onde também será possível configurar o boot principal (COLEY, 2014, tradução nossa).

O BeagleBone Black pode trabalhar conectado à um computador via USB agindo como um dispositivo de armazenamento ou como um computador completo quando conectado à um display, um mouse e um teclado. Assim como seus concorrentes, o BeagleBone também utiliza circuitos de expansão chamados *capes* (capas) com vários recursos diferentes. A placa do BeagleBone Black pode ser visualizada na figura 20 (COLEY, 2014, tradução nossa).

Figura 20 – Placa do BeagleBone



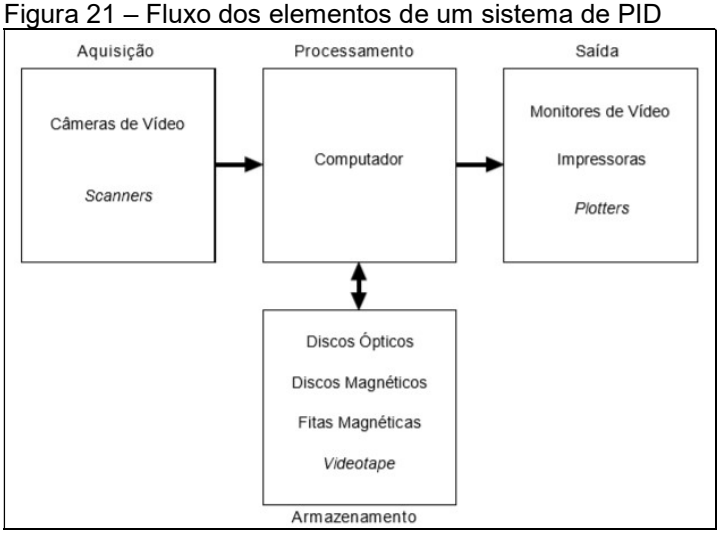
Fonte: Coley (2014).

5 PROCESSAMENTO DE IMAGENS DIGITAIS

O Processamento de Imagens Digitais (PID) é a tecnologia de tratar imagens por meio de algoritmos computacionais e este processo é comumente encontrado em sistemas robóticos ou inteligentes, sistemas médicos, dentre outros. Resulta deste processo uma ou mais imagens ou características da imagem original. As imagens digitais são compostas de *pixels*, milhares de pontos que combinam as cores vermelho, verde e azul para formar as cores do mundo real. O principal propósito do PID é auxiliar humanos na obtenção de imagens de alta qualidade ou características da imagem original. (ZHOU; WU; ZHANG, 2010, tradução nossa).

A área de PID possibilitou a viabilização de aplicações de análise automática por computador de informações de uma cena, que pode ser definida como “análise de imagens”, “visão computacional” ou “reconhecimento de padrões”. Um sistema de PID é composto por alguns elementos que se aplicam desde sistemas de baixo custo até estações de trabalho sofisticadas. Estes elementos são: aquisição, armazenamento, processamento e exibição. O Fluxo destes elementos pode ser observado na figura 21 (MARQUES FILHO; VIEIRA NETO, 1999).

Figura 21 – Fluxo dos elementos de um sistema de PID



Fonte: Marques Filho, Vieira Neto (1999).

A aquisição converte uma imagem em uma representação numérica compreensível para o computador processar. Esta etapa contém dois elementos, onde um é um dispositivo físico sensível a uma faixa de energia no espectro

eletromagnético (visível ou invisível aos humanos) que produz um sinal elétrico analógico e um digitalizador que converte este sinal em uma informação binária (MARQUES FILHO; VIEIRA NETO, 1999).

A etapa de processamento envolve algoritmos e dependem da grande maioria das vezes apenas do software do sistema, diferente das outras etapas que envolvem algum tipo de hardware (MARQUES FILHO; VIEIRA NETO, 1999).

5.1 VISÃO COMPUTACIONAL

Visão computacional é transformação de dados de uma imagem em uma nova forma de representação ou em uma decisão. Todas as transformações têm um propósito que levará em conta a entrada de dados para estabelecer um resultado ou tomar uma decisão, como identificar um objeto bloqueando o caminho de um carro autônomo pela câmera e instruir o carro a desviar ou definir quantos tumores há em uma imagem do corpo humano. O que é natural para o humano, como identificar se há ou não um carro em uma imagem, pode ser complexo para uma máquina, pois o cérebro humano consegue separar o sinal de visão em várias camadas abstraindo partes da imagem menos importantes e focando-se no mais importante, associando a imagem rapidamente com outras semelhantes observadas durante toda a vida. A máquina, por outro lado, recebe apenas uma matriz de dados que forma a imagem, porém esta matriz deve ser analisada por algoritmos para que sejam identificados padrões (BRADSKI; KAEHLER, 2008, tradução nossa).

Segundo Marques Filho e Vieira Neto (1999), dentro da área de PID, os sistemas de visão artificial têm como objetivo imitar o sistema visual humano aplicando-o em máquinas. Nas palavras de Robert J. Schalkoff (MARQUES FILHO; VIEIRA NETO, 1999 aput Schalkoff 1989) no seu livro *Digital Image Processing and Computer Vision*, “Estamos tentando ensinar robôs a enxergar”. O sistema visual humano possui três características desafiadoras a se copiar:

a) uma base de dados muito rica;

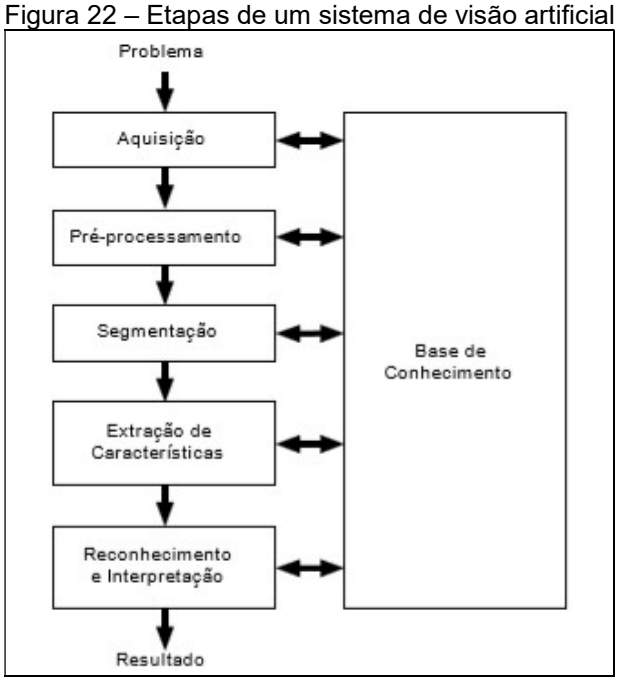
b) altíssima velocidade de processamento;

c) a capacidade de trabalhar sob condições muito variadas.

O avanço da tecnologia de armazenamento e processamento dos

computadores mostram que as duas primeiras características podem ser aplicadas, porém o principal desafio continua sendo fazer com que os sistemas trabalhem em condições variadas de luminosidade, contraste e o posicionamento dos objetos na cena para interpretação da mesma. Os sistemas de visão artificial são definidos como sistemas capazes de adquirir, processar e interpretar imagens de cenas reais e tem etapas definidas expostas na figura 22 (MARQUES FILHO; VIEIRA NETO, 1999).

Figura 22 – Etapas de um sistema de visão artificial



Fonte: Marques Filho, Vieira Neto (1999).

5.2 OCR

A possibilidade de fazer uma máquina ler textos com a mesma facilidade que o ser humano sempre foi cobiçada, e nos últimos 50 anos o OCR têm sido a mais bem-sucedida tecnologia nas áreas de reconhecimento de padrões e inteligência artificial, apesar de não se comparar ainda com as capacidades humanas de leitura (EIKVIL, 1993, tradução nossa).

Ainda segundo Eikvil (1993, tradução nossa), O OCR pertence ao grupo de técnicas de identificação automáticas, que consistem em técnicas alternativas de inserir dados à um computador. Esta tecnologia é necessária quando a informação

deve ser lida tanto por humanos quanto por máquinas. Em comparação às outras técnicas, o OCR é a única em que não é necessário controlar o processo que produz a informação.

O reconhecimento de caracteres pode ser feito tanto quando o caractere é escrito por máquinas quanto escritos à mão, porém a performance e a precisão do reconhecimento serão dependentes da qualidade da fonte do texto (EIKVIL, 1993, tradução nossa).

O OCR é dividido em dois tipos: reconhecimento *Offline* e *Online*. No reconhecimento *Offline* a fonte de informações é uma imagem obtida por uma fotografia ou escâner enquanto o reconhecimento *Online* a fonte de informações é representada por pontos em função do tempo. Em suma, o reconhecimento *Online* reconhece o texto enquanto o mesmo é escrito até mesmo à mão, enquanto o reconhecimento *Offline* reconhece o texto após o mesmo estar pronto (CHARLES et al, 2012, tradução nossa).

As áreas de reconhecimento de padrões e análise de imagens são hoje largamente estudadas graças ao OCR, que é uma subárea das primeiras, mas é a mais antiga e a que incentivou o desenvolvimento de outros métodos. O primeiro registro data de 1870, quando foi inventado um escâner de retina baseado em fotocélulas, porém o OCR moderno apareceu na década de 1950 junto da revolução tecnológica e do computador digital. A primeira máquina com essa capacidade foi desenvolvida em 1954 com o propósito de converter relatórios escritos em cartões perfurados (EIKVIL, 1993, tradução nossa).

Em 1929 a primeira patente sobre OCR foi registrada na Alemanha por Gustav Taushcek. A máquina de Tauschek era um dispositivo mecânico com um detector de imagens e é considerado o primeiro computador OCR, tendo seu desenvolvimento iniciado em 1949. O objetivo era auxiliar pessoas cegas na Administração dos Veteranos dos Estados Unidos. (CHARLES et al, 2012, tradução nossa).

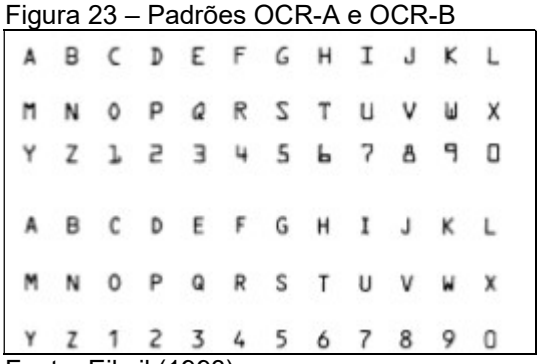
É possível separar o OCR em três gerações. A primeira geração é compreendida entre 1960 e 1965, quando as máquinas de leitura de OCR eram capazes de ler uma determinada forma de caractere especialmente desenhados para a identificação, sendo que os primeiros não pareciam nem mesmo naturais

para os humanos. Com o surgimento de mais fontes, algumas máquinas podiam ler até 10 tipos diferentes de fonte (EIKVIL, 1993, tradução nossa).

Os primeiros sistemas eram limitados a reconhecer apenas um tipo de fonte de um único tamanho de caracteres impressos. O espaçamento entre os caracteres era fixo para que cada caractere segmentado fosse reconhecido. Era essencial que não houvesse rotação, diferença de escala e qualquer outro tipo de distorção nos caracteres. Estes sistemas eram usados em aplicações muito específicas e foram usados por pouco tempo. Rapidamente, percebeu-se a necessidade de desenvolver métodos mais eficientes para reconhecer diferentes tipos de fontes com tamanhos e características variadas impressas por diferentes tipos de impressoras (O'GORMAN, 1997, tradução nossa).

A segunda geração aparece na segunda metade de 1960 até início de 1970, quando os sistemas eram capazes de ler caracteres impressos regularmente e escritos à mão. A partir de então, a padronização passou a ter importância nos sistemas de OCR. Neste período, desenvolveu-se dois padrões de caracteres para reconhecimento, o OCR-A (Americano) e OCR-B (Europeu). Os padrões podem ser observados na figura 23 (EIKVIL, 1993, tradução nossa).

Figura 23 – Padrões OCR-A e OCR-B



Fonte: Eikvil (1993).

Houveram tentativas de tornar os dois padrões em um único, porém o que aconteceu foi o surgimento de máquinas capazes de ler os dois padrões. A terceira geração apareceu na metade de 1970, quando o objetivo era trabalhar com documentos de pouca qualidade e escritos à mão. Nesta geração também apareceram as preocupações com baixo custo e velocidade de leitura. A partir de 1986 quando os custos de hardware começaram a cair e sistemas de OCR

começaram a ser vendidos como pacotes de software, popularizando a tecnologia (EIKVIL, 1993, tradução nossa).

Muitos métodos de reconhecimento de caracteres escritos à mão são similares aos métodos daqueles impressos, porém há algumas diferenças. No reconhecimento em tempo real, os dados reconhecidos são uma representação de uma sequência de tempo da posição da caneta, guardando o movimento da mesma. Nos métodos de reconhecimento pós-escrita, os dados são capturados pela digitalização semelhante aos caracteres impressos. A principal desafio fica por conta da segmentação dos caracteres conectados (O'GORMAN, 1997, tradução nossa).

Na última década o desenvolvimento da fotografia digital e dos dispositivos móveis possibilitou a aplicação do OCR nestes dispositivos com o objetivo de obter resultados em tempo real. Enquanto o OCR era aplicado em escâneres nas décadas anteriores que limitava a portabilidade e a aplicação do OCR exclusivamente em documentos, agora o OCR poderia ser utilizado em muitos outros casos. No entanto, a aquisição das imagens feita de forma diferente por fotografia ao invés de escaneamento de documentos trouxe o desafio de tratar a inclinação, distorção e outros possíveis problemas na imagem adquirida (MOLLAH et al 2011, tradução nossa).

Apesar do desenvolvimento crescente dos dispositivos móveis arquitetados em ARM, o processamento exigido pelas aplicações OCR é alto e ainda não se compara a performance das aplicações desktop com as aplicações com móveis. Aplicações voltadas para estes dispositivos devem ser computacionalmente eficientes e com menor consumo de memória (MOLLAH et al, 2011, tradução nossa).

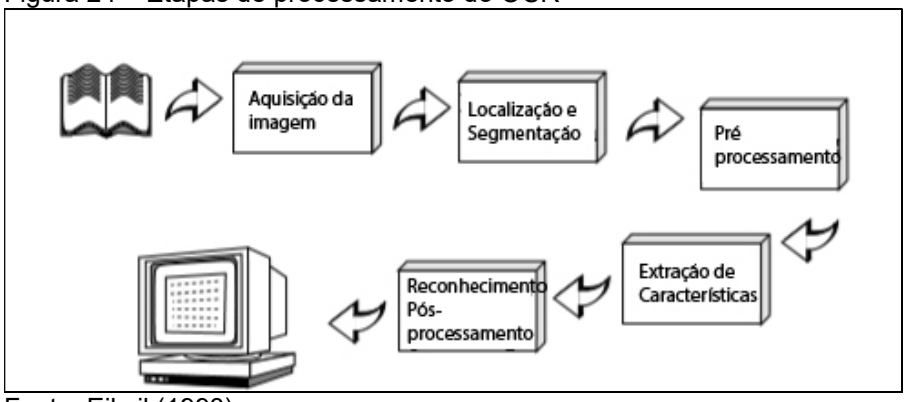
A área de estudo do OCR desde a metade do século passado tem avançado de forma a alcançar taxas de reconhecimento acima de 99% para textos impressos e acima de 90% em textos escritos à mão em documentos (CHOPRA et al, 2014, tradução nossa).

5.2.1 Etapas do processamento de OCR

O reconhecimento de padrões tem um princípio que é ensinar a máquina

os padrões. Os padrões no caso do OCR são letras, números e caracteres especiais. O treinamento da máquina ocorre ao inserir os caracteres para que cada um seja classificado e no processo de reconhecimento os caracteres desconhecidos são comparados a aqueles já classificados, procurando pelo melhor correspondente. Um sistema OCR é consistido em algumas etapas que são: A aquisição da imagem, localização e segmentação do texto, pré-processamento, extração de características e reconhecimento. A figura 24 ilustra estes processos (EIKVIL, 1993, tradução nossa).

Figura 24 – Etapas de processamento do OCR



Fonte: Eikvil (1993).

Até o desenvolvimento da fotografia digital, o escaneamento óptico era a única forma de aquisição de imagem para o processamento de OCR. No processo de escaneamento, uma imagem digital é gerada através de um mecanismo de transporte com um dispositivo sensível que converte a luz em escalas de cinza (EIKVIL, 1993, tradução nossa).

A imagem em escalas de cinza é transformada então em uma imagem apenas em preto e branco. Este processo chamado de binarização (CHOPRA et al, 2014, tradução nossa).

A aquisição de uma imagem de um escâner é diferente de uma câmera digital. Enquanto o escâner captura um documento idealmente sem distorções de rotação e luz, uma fotografia digital pode trazer este problema tornando-a mais difícil de ser tratada por uma aplicação OCR (MITHE; INDALKAR; DIVEKAR, 2013,

tradução nossa).

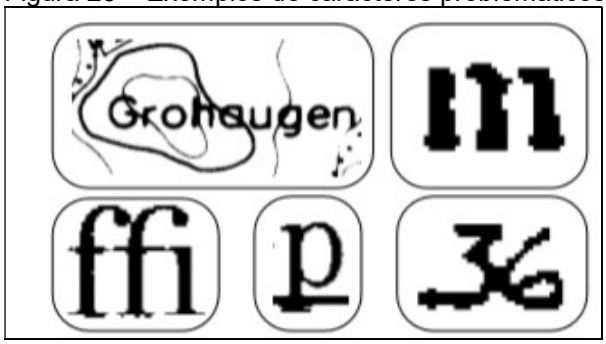
A obtenção de uma imagem digital é dividida em duas etapas: a aquisição da imagem e a digitalização da mesma. A primeira etapa pode ser definida como transdução optoeletrônica. Esta etapa consiste na redução da dimensionalidade da imagem, transformando uma cena real tridimensional em uma representação bidimensional e isto é realizado por uma câmera fotográfica com um sensor CCD que gera um sinal analógico. A etapa de digitalização é a conversão do sinal analógico em uma matriz de pontos em que cada ponto corresponde a um *pixel* (MARQUES FILHO; VIEIRA NETO, 1999).

A etapa de localização e segmentação consiste na definição das partes que constituem a imagem, separando texto de figuras e gráficos. Localizados os textos, a segmentação é aplicada para separar os caracteres com o intuito de reconhecer cada um separadamente. Sendo os caracteres bem separados e limpos, esta etapa costuma ser simples, porém alguns problemas podem aparecer:

a) junções entre os caracteres podem levar o sistema a considerar dois ou mais caracteres como um só, assim como a fragmentação de um caractere levará a situação contrária;
b) um ponto ou vírgula pode ser interpretado como ruído pelo pré-processamento e pode ser eliminado, o contrário também podendo ocorrer;
c) gráficos podem ser confundidos com texto e enviados para reconhecimento, assim como um texto pode ser confundido com um gráfico e não reconhecido.

Alguns exemplos de caracteres com problemas podem ser vistos na figura 25 (EIKVIL, 1993, tradução nossa).

Figura 25 – Exemplos de caracteres problemáticos



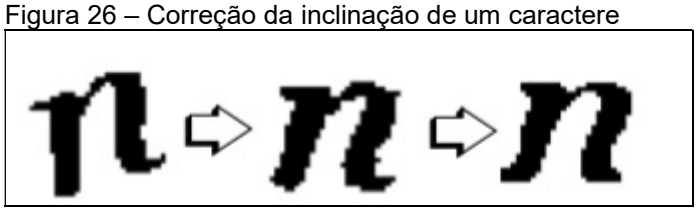
Fonte: Eikvil (1993).

Defeitos nos caracteres podem ser inseridos tanto no processo de impressão dos documentos ou no escaneamento do mesmo. Poucos *pixels* sobrando ou faltando em caracteres pode representar pouco para a visão humana, mas faz grande diferença para o OCR (NAGY; NARTKER; RICE, 2000, tradução nossa).

Os caracteres são reconhecidos primeiramente pela forma pelos sistemas de OCR e alguns caracteres têm formatos parecidos, requerendo que estes estejam com uma qualidade boa para que sejam reconhecidos precisamente. O caso mais clássico é dos caracteres “1”, “I” e “l”. A pontuação também pode ser um desafio, principalmente com pontos e vírgulas (NAGY; NARTKER; RICE, 2000, tradução nossa).

Problemas como ruídos, inclinação e muitos outros podem prejudicar o reconhecimento correto dos caracteres. A etapa de pré-processamento é responsável por corrigir tais problemas e facilitar o processo de OCR. Várias técnicas podem ser necessárias para obter o melhor resultado possível. A figura 26 mostra um caractere que passou por um processo de correção de inclinação e suavização (EIKVIL, 1993, tradução nossa).

Figura 26 – Correção da inclinação de um caractere



Fonte: Eikvil (1993).

Nesta etapa de pré-processamento os procedimentos aplicados são ditos de baixo nível, pois trabalha-se com os pixels da imagem ainda sem saber de que caracteres são estes *pixels* (MARQUES FILHO; VIEIRA NETO, 1999).

A etapa de extração de características dos símbolos é uma das partes mais difíceis do reconhecimento de padrões. Existem algumas técnicas diferentes para reconhecimento de caracteres que se dividem em três classes principais: distribuição de pontos, transformações e expansões em séries e análise estrutural. A classificação dos caracteres é etapa de assimilar cada caractere reconhecido com a classe correta de caractere já conhecida (EIKVIL, 1993, tradução nossa).

Os sistemas mais comuns dividem-se entre os que utilizam os métodos de análise estrutural e entre os que utilizam a comparação de modelos (que é basicamente o método de distribuição de pontos). Existem sistemas chamados híbridos que utilizam os dois métodos associados (IVANSKI; SILVA; BELLON, 2008).

A extração de características utiliza descritores para identificar com precisão símbolos parecidos como “5” e “6”, “8” e “B”, dentre outros. A entrada nesta etapa é uma imagem e a saída é um conjunto de dados. No reconhecimento os descritores utilizados podem ser as coordenadas *x* e *y* do centro de gravidade do caractere e razão entre sua largura e altura (MARQUES FILHO; VIEIRA NETO, 1999).

A classificação divide-se em duas abordagens que são os métodos de decisão teórica que são utilizados quando os caracteres são representados numericamente em um vetor e a abordagem baseada na estrutura física do caractere e a relação entre os elementos desta estrutura (EIKVIL, 1993, tradução nossa).

No reconhecimento, os descritores são utilizados para rotular o caractere baseado em suas características com a classe conhecida de caractere. A interpretação define um significado para o conjunto de caracteres (MARQUES FILHO; VIEIRA NETO, 1999).

Após o reconhecimento, o pós-processamento é responsável por algumas tarefas. Uma delas é o agrupamento dos símbolos tornando-os palavras e números. Este processo é realizado considerando-se os espaçamentos entre os caracteres,

que são maiores na separação de palavras. Em textos impressos, este processo costuma ser simples, porém em textos escritos à mão este processo pode ser problemático com a falta de constância dos espaçamentos (EIKVIL, 1993, tradução nossa).

Mesmo após a transformação dos caracteres em palavras completas, alguns caracteres podem ter sido reconhecidos erroneamente. Até mesmo o melhor do sistema de reconhecimento não irá terá uma taxa de erro nula, e estes erros podem ser detectados e corrigidos com o uso de contexto. Para esta correção, há duas abordagens. Uma delas baseia-se na possibilidade dos caracteres que aparecem em sequência utilizando regras de sintaxe. Exemplificando, após um ponto final deve aparecer uma letra maiúscula e na língua inglesa a probabilidade da letra “k” aparecer após a letra “h” é nula, e caso isto aconteça, detecta-se um erro. Outra abordagem é o uso de dicionários, que se prova a mais eficiente. Ao identificar uma palavra, a presença da mesma é verificada no dicionário. Caso a mesma não seja encontrada, pode-se alterar a mesma para a palavra com maior similaridade (EIKVIL, 1993, tradução nossa).

5.2.2 Técnicas de identificação de caracteres

A identificação dos caracteres é a parte mais difícil do processo de OCR e existem algumas abordagens diferentes. Uma delas é a comparação de modelos ou distribuição de pontos em que o caractere é comparado à um conjunto de caracteres conhecidos para que seja encontrada a classe correspondente. Esta técnica é de simples implementação e está presente em muitos sistemas de ÓCR comerciais, porém é sensível a ruídos e variações de estilo, além de não lidar com caracteres com rotação (EIKVIL, 1993, tradução nossa).

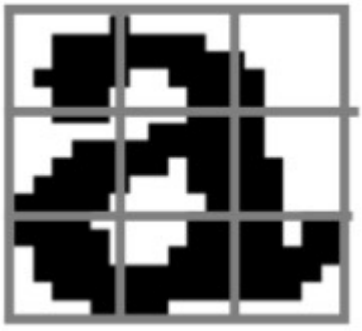
Outra abordagem utilizada é a baseada em características. Técnicas que utilizam esta abordagem utilizam medidas e descritores de um caractere para comparar com outro já existente obtido na fase de treinamento. As características são representadas por um vetor de números (EIKVIL, 1993, tradução nossa). Algumas destas técnicas são:

a) zoneamento: o retângulo ao redor do caractere é divido em várias

regiões e a intensidade de pontos pretos nestas regiões são é utilizada como característica. A ilustração desta técnica pode ser vista na figura 27;

b) momentos: a frequência de pontos pretos em relação ao centro de gravidade é utilizada como característica;
c) cruzamentos e distâncias: esta técnica verifica a quantidade de cruzamentos no formato de caractere em diferentes direções e também os tamanhos no vetor do mesmo;
d) n-tuplas: a ocorrência de junção de pontos pretos e brancos em uma ordem específica é utilizada como característica.

Figura 27 – Técnica de zoneamento



Fonte: Eikvil (1993).

A densidade de cada zona no método de zoneamento é calculada dividindo-se o número de *pixels* preenchidos pelo número total de *pixels* de cada zona (CHARLES et al, 2012, tradução nossa).

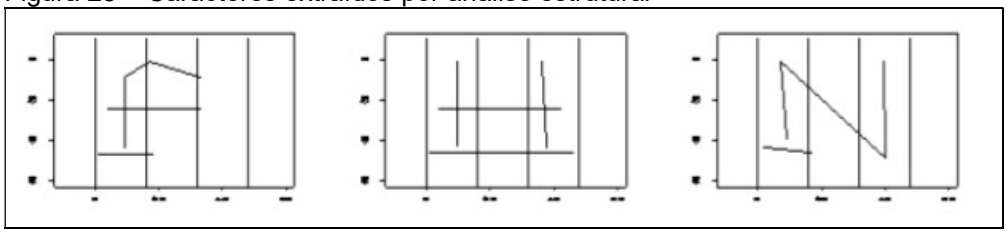
O histograma do caractere também pode ser usado no reconhecimento. Para isto, são contados os *pixels* na vertical, horizontal e nas diagonais (CHARLES et al, 2012, tradução nossa).

As técnicas baseadas em transformações e expansões em séries utiliza transformações como Fourier e são invariantes a deformações como translação e rotação. Transformações como estas são baseadas em curvas que descrevem o contorno do caractere. Isto significa que este método é muito sensível a ruído nas bordas do caractere (EIKVIL, 1993, tradução nossa).

A análise estrutural baseia-se nas estruturas geométricas e topológicas de um símbolo. Características fim de linha e intersecções entre linhas e círculos são

utilizadas na definição. Esta técnica é altamente tolerante a ruídos e mudança de estilos, porém tem tolerância moderada à rotação e translação. A figura 28 exibe linhas extraídas dos caracteres “F”, “H” e “N” (EIKVIL, 1993, tradução nossa).

Figura 28 – Caracteres extraídos por análise estrutural



Fonte: Eikvil (1993).

5.2.3 Técnicas de classificação de caracteres

O processo de classificação consiste em assimilar cada símbolo reconhecido com a classe correspondente de caractere. Os métodos de classificação de dividem em métodos de decisão teórica de métodos estruturais (EIKVIL, 1993, tradução nossa).

As principais abordagens das técnicas de decisão teórica são os classificadores de distância, classificadores estatísticos as redes neurais (EIKVIL, 1993, tradução nossa).

As técnicas de correspondência utilizam classificadores de distância e baseiam-se no cálculo da similaridade das medidas do vetor de características dos caracteres identificados para cada classe de caractere conhecida. A medida mais comum utilizada nestes métodos é a distância Euclidiana (EIKVIL, 1993, tradução nossa).

Um destes métodos é algoritmo de vizinho mais próximo (*K-Nearest Neighbour*). Este método classifica o caractere baseado no exemplo de treinamento mais próximo e está entre os algoritmos de aprendizado de máquina mais simples (SINGH; BUDHIRAJA, 201?, tradução nossa).

Os classificadores estatísticos utilizam a probabilidade para diminuir a chance de erros de classificação. O classificador Bayes, por exemplo, compara o vetor de características de um símbolo desconhecido com o vetor de todas as classes de caracteres conhecidas, selecionando aquela com a máxima probabilidade

(EIKVIL, 1993, tradução nossa).

As redes neurais são os mais novos atores no uso de reconhecimento de caracteres e outros padrões. Tais redes são compostas de várias camadas de elementos conectados. Um vetor de características desconhecido é utilizado na camada de entrada e cada elemento na camada calcula o peso desta entrada e transformada em uma função não linear. Durante a fase de treinamento, o peso calculado de cada conexão é ajustado até que a saída desejada seja obtida. As redes neurais têm como grande vantagem a natureza adaptativa (EIKVIL, 1993, tradução nossa).

A abordagem mais utilizada nos métodos estruturais são os métodos sintáticos. Nestes métodos utiliza conceitos gramáticos para definir a composição de cada classe de um caractere (EIKVIL, 1993, tradução nossa).

5.2.4 Métodos de pré-processamento de imagens

Com o avanço das câmeras digitais na aquisição de imagens para o OCR, o pré-processamento de imagens ganhou ainda mais importância. A redução de ruídos, a correção da orientação e a binarização adaptativa são exemplos de técnicas aplicáveis. A maioria das aplicações de OCR sempre trabalhou com escâneres, produzindo uma imagem quase perfeita e livre de rotação. Testes realizados com um documento mostram que uma rotação de 10 graus na imagem já torna a mesma irreconhecível sem pré-processamento (BIENIECKI; GRABOWSKI; ROZENBERG, 2007, tradução nossa).

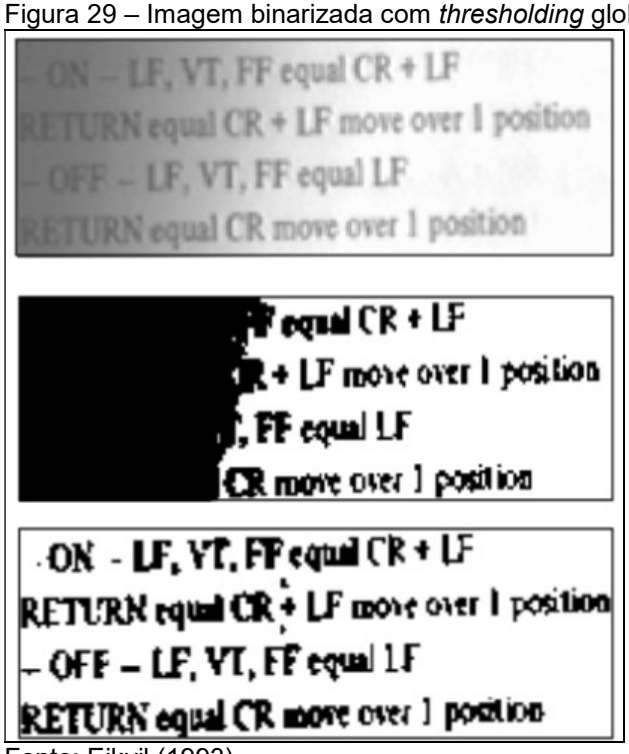
Pode ser dizer que a digitalização das bibliotecas só foi possível graças ao OCR. Esta digitalização enfrenta desafios como o uso de fontes diferentes, ruído de impressão, depreciação do documento, espaços não padronizados, dentre outros. Algoritmos OCR são preparados para trabalhar com imagens apenas com as cores preto e branco. A transformação da imagem apresentadas em escalas de cinza para uma imagem preta e branca é chamada de binarização (GUPTA; JACOBSON; GARCIA, 2007, tradução nossa).

A binarização geralmente utiliza um processo chamado de *thresholding* para particionar a imagem em dois conjuntos baseados no histograma. Há dois tipos

principais: um global, que considera a imagem inteira na comparação e a adaptativa, que separa a imagem em pequenas regiões para efetuar as comparações (ELMORE; MARTONOSI, 2008, tradução nossa).

O processo de binarização é importante pois afeta diretamente nos resultados do reconhecimento dos caracteres. O processo de *thresholding* global é simples: considerando a imagem, é definido um limite na escala de cinza onde todo *pixel* acima deste limite será branco e todo *pixel* abaixo desta escala será preto. Em alguns casos em que o contraste do documento varia muito, o *thresholding* adaptativo é mais aconselhado, utilizando pequenas regiões da imagem para realizar a comparação. A figura 29 ilustra uma mesma imagem binarizada com *thresholding* global e adaptativo (EIKVIL, 1993, tradução nossa).

Figura 29 – Imagem binarizada com *thresholding* global e adaptativo



Fonte: Eikvil (1993).

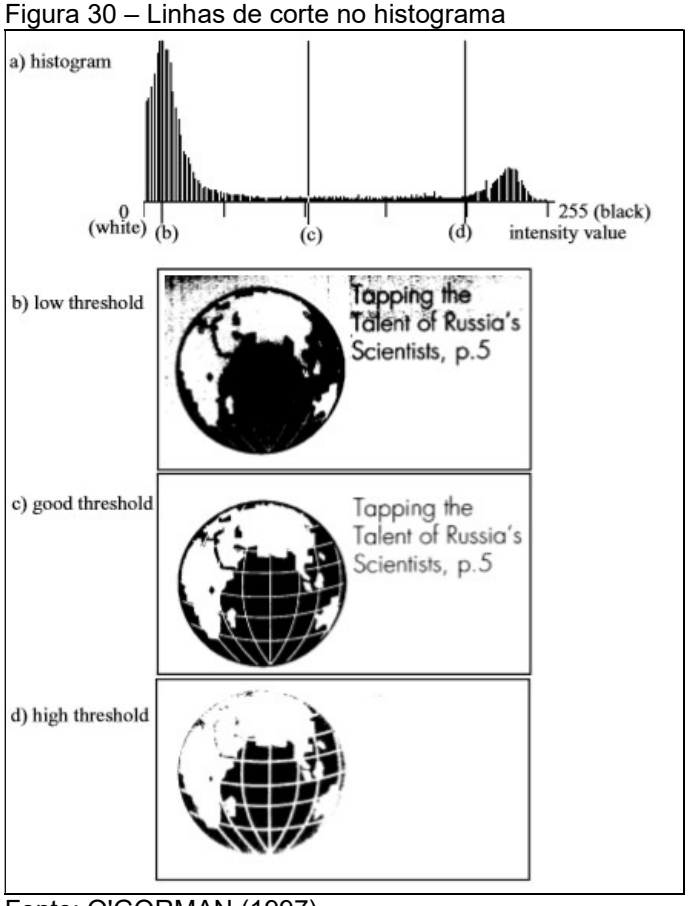
O histograma de uma imagem em escalas de cinza apresenta a quantidade de *pixels* separadas pela intensidade que vai de 0 (branco) a 255 (preto). Em imagens com os planos frontal e traseiro bem definidos, haverá basicamente dois picos no histograma nas extremidades. Neste caso, o *thresholding* global pode ser aplicado satisfatoriamente, definindo um limite entre a média destes dois picos

(O'GORMAN, 1997, tradução nossa).

Uma imagem em escalas de cinza com o histograma bem distribuído pode sofrer de classificação incorreta dos *pixels* ao utilizar o método de *thresholding* global. Neste caso, o *threshold* global ou adaptativo é aplicado para separar a imagem em pequenas regiões, cada uma com seu *threshold* particular. O problema neste caso é definir o tamanho destas subdivisões da imagem, de modo que o resultado seja satisfatório (O'GORMAN, 1997, tradução nossa).

O limite estabelecido no *thresholding* é um valor entre 0 e 255 na escala de cinza que significa o corte que definirá se o *pixel* será preto ou branco. A figura 30 mostra as linhas de corte no histograma da imagem de três níveis de *threshold* e a imagem resultante correspondente. Nota-se que nesta imagem o histograma tem um grande vale entre dois picos, o que caracteriza que o método de *thresholding* global pode ser aplicado (O'GORMAN, 1997, tradução nossa).

Figura 30 – Linhas de corte no histograma



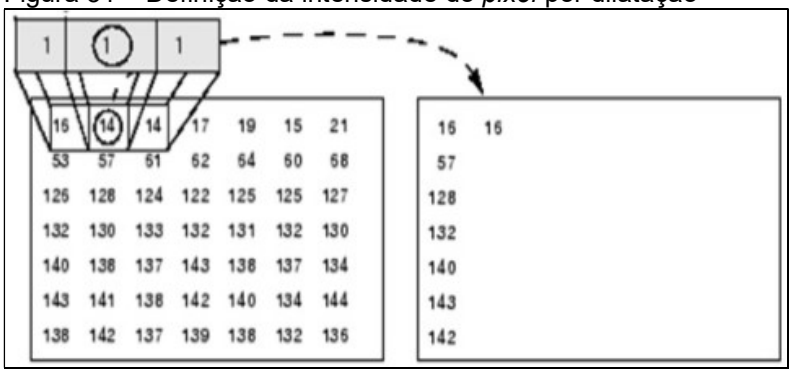
Fonte: O'GORMAN (1997).

Os caracteres podem sofrer de buracos ou ruídos durante o processo de binarização. Estes buracos podem causar a quebra dos caracteres em um ou mais dois objetos, assim como ruídos podem juntar caracteres. O processamento morfológico utiliza métodos para tratar tais problemas, sendo alguns os principais a erosão e a dilatação. (ALGINAHI, 2010, tradução nossa).

A dilatação e a erosão, que são métodos removem e adicionam *pixels* nos objetos resultados. A dilatação define o *pixel* de saída com o valor máximo dos vizinhos, enquanto a erosão define o *pixel* de valor mínimo dos vizinhos. As figuras 31 e 32 ilustram a dilatação e a erosão de uma imagem em escala de cinza (GUPTA; NAIR, 2013, tradução nossa).

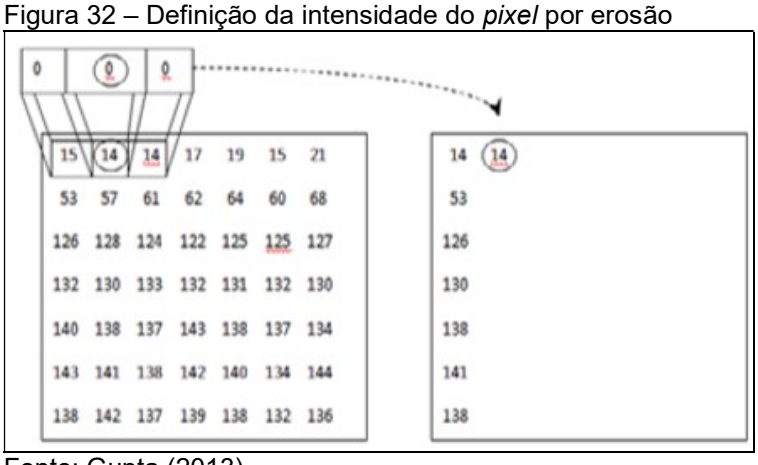
.

Figura 31 – Definição da intensidade do *pixel* por dilatação



Fonte: Gupta (2013).

Figura 32 – Definição da intensidade do *pixel* por erosão

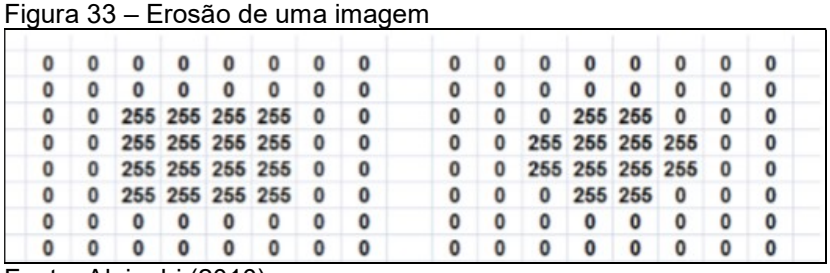


Fonte: Gupta (2013).

Em uma imagem binarizada, isto significa adicionar ou remover *pixels* pretos. O resultado será o aumento ou a diminuição do tamanho do objeto. A erosão irá verificar os *pixels* vizinhos das bordas e caso o número de *pixels* apagados seja

maior que o limite definido, o *pixel* também é apagado. Isto resulta na redução do objeto conforme observado na figura 33 (ALGINAHI, 2010, tradução nossa).

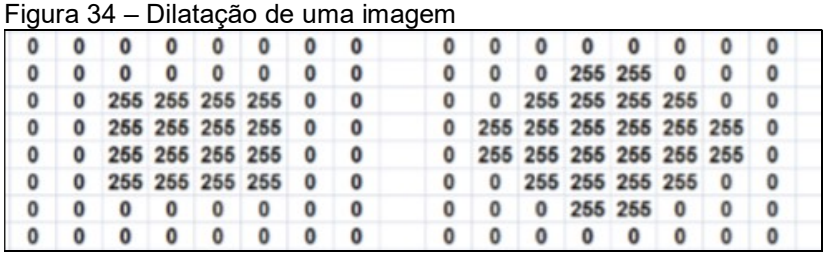
Figura 33 – Erosão de uma imagem



Fonte: Alginahi (2010).

O processo de dilatação irá realizar o processo contrário, acendendo o *pixel* caso a quantidade de *pixels* vizinhos acesos seja maior que o limite definido. O resultado é o aumento do objeto, observado na figura 34 (ALGINAHI, 2010, tradução nossa).

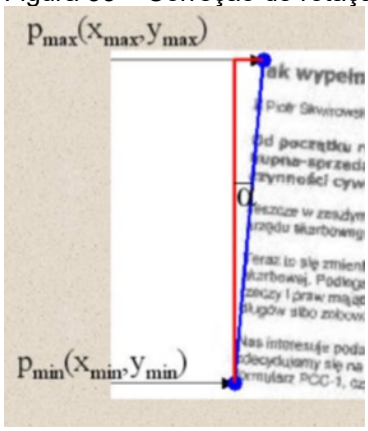
Figura 34 – Dilatação de uma imagem



Fonte: Alginahi (2010).

A rotação das áreas de texto de uma imagem consiste em um algoritmo que busca o primeiro *pixel* preto da esquerda para a direita na área de texto em intervalos entre o início e o fim desta área. Este intervalo menor aumenta a precisão, porém também o tempo de processamento. A diferença na distância máxima e mínima das coordenadas da área de texto resultam no ângulo de rotação. A figura 35 este processo (BIENIECKI; GRABOWSKI; ROZENBERG, 2007, tradução nossa).

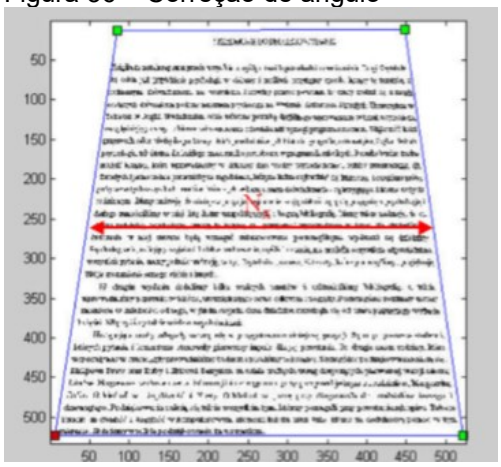
Figura 35 – Correção de rotação



Fonte: Bieniecki, Grabowski e Rozenberg (2007).

O ângulo da imagem capturada por uma câmera também pode estar prejudicado. Baseando-se também na definição da área de texto, buscando os *pixels* pretos, é possível definir as coordenadas x e y iniciais e finais da área de texto. A figura 36 ilustra este processo (BIENIECKI; GRABOWSKI; ROZENBERG, 2007, tradução nossa).

Figura 36 – Correção de ângulo



Fonte: Bieniecki, Grabowski e Rozenberg (2007).

Estes tratamentos são suficientes para imagens de documentos obtidos por um escâner ou câmera, porém ao tratar uma cena real com OCR o problema de rotação e/ou inclinação exige um pouco mais de trabalho, visto que a imagem irá apresentar maiores variações na detecção das bordas dos objetos (ELMORE; MARTONOSI, 2008, tradução nossa).

5.2.5 Bibliotecas para processamento de imagens

Na ciência da computação as bibliotecas são coleções de programas utilizados no desenvolvimento de softwares. Há muitas bibliotecas destinadas ao processamento de imagens, porém serão abordadas apenas o Tesseract, específica para OCR e o OpenCV, software para processamento de imagens em geral.

5.2.5.1 Tesseract

O Tesseract é uma biblioteca OCR de código aberto que foi desenvolvida pela HP entre 1984 e 1994 secretamente e apareceu em 1995 com resultados surpreendentes. A HP projetou o mesmo com o objetivo de oferecer como uma extensão de hardware ou software para escâneres. O desenvolvimento do Tesseract foi motivado principalmente pelo fato de que os motores OCR comerciais estavam em desenvolvimento (SMITH, 2016, tradução nossa).

Em 1995 o Tesseract foi enviado para a Universidade de Nevada para Teste Anual de Precisão de OCR, onde provou-se valioso contra os motores OCR da época. Em 2005, a HP tornou o Tesseract de código aberto (SMITH, 2016, tradução nossa).

O processamento do Tesseract tem alguns passos definidos. O primeiro passo do processo é um componente que analisa as bordas externas dos objetos. Este processo possibilita a detecção de texto branco em um fundo preto. O Tesseract foi um dos primeiros sistemas OCR de lidar com este tipo de texto (SMITH, 2016, tradução nossa).

O algoritmo que define as linhas é projetado para reconhecer uma página inclinada sem que a mesma tenha que passar por uma correção de inclinação causando em uma perda de qualidade da imagem. As bordas são unidas organizando os objetos em blocos de texto, e estes textos são analisados de acordo com os espaçamentos, quebrando o texto em palavras (SMITH, 2016, tradução nossa).

O reconhecimento ocorre em duas etapas: Na primeira, há uma tentativa de reconhecimento de cada palavra. As palavras identificadas são armazenadas

como dado treinamento para um classificador adaptativo, tornando a classificação mais fácil nas próximas palavras. Ao chegar no fim da página, com o classificador treinado, um segundo reconhecimento é feito (SMITH, 2016, tradução nossa).

5.2.5.2 OpenCV

O OpenCV é uma biblioteca de visão computacional de código aberto escrita em C e C++ que pode ser utilizada no Linux, Windows e Mac OS X e aproveita-se de processadores de vários núcleos. O mesmo foi desenvolvido com foco em aplicações em tempo real. Um dos objetivos do OpenCV é oferecer uma estrutura de visão computacional simples para a construção de aplicações para inspeção industrial de produtos, imagens médicas, segurança, robótica, dentre outras (BRADSKI; KAEHLER, 2008, tradução nossa).

O surgimento do OpenCV ocorreu em uma pesquisa da Intel para aplicações de intenso uso de CPU. Um dos autores que trabalhava da Intel percebeu que muitas universidades como o MIT tinham grupos que tinham estruturas de visão computacional bem desenvolvidas pelos alunos, e os alunos sempre desenvolviam a partir do que já existia, aperfeiçoando os códigos. A partir deste conceito, o OpenCV foi desenvolvido para tornar esta estrutura de visão computacional universalmente disponível (BRADSKI; KAEHLER, 2008, tradução nossa).

Os principais objetivos definidos pela equipe de desenvolvimento foram:

a) prover um código aberto e otimizado para uma infraestrutura de visão, sem reinventar a roda;
b) disseminar o conhecimento provendo uma infraestrutura em que os desenvolvedores poderiam trabalhar, tornando o código mais legível e transferível;
c) uma licença que não obriga as aplicações comerciais utilizarem a serem grátis ou abertas.

O crescimento de aplicações de visão computacional aumenta a necessidade de processadores mais rápidos e com isso, a receita da Intel aumentaria com a venda de processadores cada vez mais rápidos. O projeto do

OpenCV iniciou-se em 1999 e a versão oficial foi lançada em 2007 (BRADSKI; KAEHLER, 2008, tradução nossa).

5.2.6 Aplicações do OCR

Historicamente, é possível definir três áreas diferentes na utilização do OCR: entrada de dados, entrada de texto e automação de processos.

As primeiras aplicações na área de entrada de dados foram utilizadas em aplicações bancárias, onde era necessário inserir uma grande quantidade de dados restritos. As primeiras aplicações eram limitadas à conjuntos específicos de caracteres e possuíam um fluxo de até 150 mil documentos por hora, tendo taxas de erro e rejeição muito abaixo de 1% (EIKVIL, 1993, tradução nossa).

O OCR é utilizado para processar cheques sem a intervenção humana, fazendo a leitura dos dados automaticamente. Assim como em todas as outras aplicações, a performance destas aplicações com cheques com os dados impressos é quase perfeita, porém com cheques escritos à mão podem apresentar falhas (MITHE; INDALKAR; DIVEKAR, 2013, tradução nossa).

A área de entrada de texto compreende máquinas de leitura automáticas principalmente utilizadas em escritórios. As primeiras máquinas para este fim tinham um conjunto de caracteres maior que as máquinas de entrada de dados e consequentemente, tinham taxa de erro e rejeição pouco maior (EIKVIL, 1993, tradução nossa).

As áreas jurídicas foram muito beneficiadas com o OCR, já que em muitos casos é necessário armazenar documentos em arquivos. O OCR permite substituir caixas de arquivos por arquivos digitais com possibilidade de pesquisa rápida (MITHE; INDALKAR; DIVEKAR, 2013, tradução nossa).

A área de automação de processos controla algum processo baseado nos reconhecimentos feitos, como a ordenação automática de cartas (EIKVIL, 1993, tradução nossa).

Além destas áreas, outras aplicações já foram descritas utilizando o OCR como a leitura de documentos combinado com um sistema de fala com o fim de ler documentos para pessoas cegas. O reconhecimento de caracteres escritos à mão

também pode ser utilizado na verificação de assinaturas. Um caso clássico de aplicação do OCR são os sistemas de leitura automática de placas, onde a imagem é adquirida por uma câmera rápida e geralmente precisa de um tratamento grande antes do reconhecimento (EIKVIL, 1993, tradução nossa)

6 TRABALHOS CORRELATOS

Para o desenvolvimento deste projeto de pesquisa, foram pesquisados alguns trabalhos correlatos para validar a viabilidade do mesmo.

6.1 PROTÓTIPO DE APLICATIVO ANDROID PARA EXTRAÇÃO DE TEXTO EM IMAGENS E CONVERSÃO EM VOZ, ORIENTADO AO APOIO A LEITURA PARA DEFICIENTES VISUAIS

A leitura é uma das funções mais importantes do ser humano, porém a mesma é prejudicada no caso de deficientes visuais pela falta de materiais digitalizados. A popularização e evolução dos dispositivos móveis tem aberto possibilidades para a execução de muitas tarefas, inclusive na Tecnologia Assistiva (MARCÍLIO, 2013).

O objetivo deste trabalho foi utilizar o OCR associado com a conversão de texto em áudio em um protótipo de aplicativo Android voltado à leitura de documentos impressos para auxílio à usuários portadores de deficiências visuais (MARCÍLIO, 2013).

A inclusão de pessoas portadoras de necessidades ainda é deficiente na sociedade atual. Muitos acervos de bibliotecas contêm problemas como a escassez de exemplares em Braille ou digitalizados, além do que o espaço para armazenagem de livros em Braille é muito maior. Outras situações também apresentam problemas no acesso à deficientes visuais como cartazes, rótulos de embalagens, manuais de instrução, dentre muito outros (MARCÍLIO, 2013).

O desenvolvimento de uma aplicação para digitalização de textos e conversão em voz elimina problemas como espaço de armazenagem, portabilidade, dentre outros (MARCÍLIO, 2013).

Os objetivos deste trabalho foram atingidos utilizando-se o Tesseract como motor de OCR e a biblioteca *TextToSpeech* aplicados na plataforma Android. Testes apontaram um bom desempenho do protótipo apresentando tem médio de 30 segundos para reconhecimento e taxa média de erro em 2,77% (MARCÍLIO, 2013).

Verificou-se que em algumas variações de luz e foco interferiram um

pouco nos resultados, problemas estes que podem ser tratados com técnicas de pré-processamento de imagens (MARCÍLIO, 2013).

6.2 EM RUMO A UM SISTEMA AUTOMÁTICO DE CONTROLE DE ACESSO DE VEÍCULOS AUTOMOTIVOS: RECONHECIMENTO DE CARACTERES EM PLACAS DE VEÍCULOS

Este trabalho aborda os conceitos de um Sistema Automático de Controle de Acesso de Veículos Automotivos (SACAVA) (GUALBERTO, 2010).

O primeiro protótipo de sistema deste tipo foi criado em 1976 no Reino Unido com o objetivo de identificar veículos roubados, tendo a primeira identificação realizada em 1981 (GUALBERTO, 2010).

O trabalho propôs a realização das etapas de pré-processamento da placa de um veículo e o reconhecimento dos caracteres por meio da tecnologia OCR (GUALBERTO, 2010).

O objetivo geral do trabalho foi reconhecer os caracteres de placas de licença de veículos a partir de imagens digitais (GUALBERTO, 2010).

O desenvolvimento do trabalho embasou-se em diferentes técnicas de localização da placa, pré-processamento, segmentação e reconhecimento dos caracteres. Os testes envolveram o uso de 343 fotos de placas veiculares em diferentes estados como placas sujas, amassadas e com reboques na frente, apresentando uma taxa de reconhecimento de 75% (GUALBERTO, 2010).

6.3 *AUTOMATIC LICENSE PLATE RECOGNITION SYSTEM*

O objetivo deste trabalho foi desenvolver um sistema automático de reconhecimento de placas veiculares (ALPR, do inglês *Automatic License Plate Recognition System*) aplicado na automatização de um estacionamento (DALAL; D'SOUZA, 2011, tradução nossa).

Um dos desafios levantados no desenvolvimento este tipo de sistema foi a diferença de padronização das placas nos países. A maioria dos países desenvolvidos possuem um padrão restrito de placas, porém em países como a

Índia possuem diferentes tipos de placas em cada estado (DALAL; D'SOUZA, 2011, tradução nossa).

Por motivos como este, a maioria dos sistemas ALPR são específicos para cada país (DALAL; D'SOUZA, 2011, tradução nossa).

Os resultados apontaram um maior sucesso no reconhecimento de placas com bordas pretas ao redor, facilitando a localização das bordas e de placas com condições de iluminação adequada, cuja taxa de sucesso ficou entre 80 e 90% (DALAL; D'SOUZA, 2011, tradução nossa).

O autor concluiu que a localização da placa depende fortemente da borda bem definida da placa e que a câmera deve estar posicionada à uma distância em que a quantidade de *pixels* identificados na parte interior da placa permaneça constante. O mesmo concluiu também o processo de segmentação pode falhar caso não haja um pico claro no histograma entre cada caractere (DALAL; D'SOUZA, 2011, tradução nossa).

6.4 SISTEMA DE CONTROLE DE ACESSO UTILIZANDO RECONHECIMENTO DE CARACTERES EM PLACA DE AUTOMÓVEL

Este trabalho teve como objetivo criar um sistema de identificação de caracteres presentes em uma imagem de placa de automóvel integrado à um computador de pequeno porte, com o propósito de controlar o acesso à ambientes reservados (NARDI et al., 2015).

O projeto tinha como objetivo inicial o desenvolvimento de um software próprio utilizando as bibliotecas OpenCV e Tesseract-OCR, porém os autores encontraram dificuldades e passaram a utilizar a biblioteca ALPR, que é uma solução pronta para o reconhecimento de placas veiculares (NARDI et al., 2015).

Nos resultados obtidos foi possível observar que durante os testes de 25 placas, 11 tiveram 0 caracteres reconhecidos enquanto 14 placas tiveram 4 ou mais caracteres reconhecidos. Foi observado também a confusão entre caracteres semelhantes como o número "8" e a letra "B" e o número "1" e a letra "I", efetuando-se um tratamento baseado na posição do caractere reconhecido na placa (NARDI et al., 2015).

Os autores concluíram que mesmo utilizando uma solução pronta de software, houveram grandes desafios na integração do software com o hardware. Os mesmos também concluem que mesmo a biblioteca ALPR não é capaz de fazer 100% do reconhecimento correto das placas (NARDI et al., 2015).

6.5 PROTÓTIPO DE APLICATIVO ANDROID PARA EXTRAÇÃO DE TEXTO EM IMAGENS PARA BUSCA SEMÂNTICA SOBRE RÓTULOS DE CERVEJAS

O objetivo deste trabalho foi desenvolver um protótipo no sistema Android para busca semântica de informações através de extração de texto em rótulos de cervejas com o uso do reconhecimento óptico de caracteres (SILVA, 2015).

A biblioteca Tesseract foi utilizada no projeto através da interface Tess4J para o reconhecimento dos caracteres. Primeiramente a autora efetuou testes com as imagens dos rótulos sem nenhum pré-processamento, onde foi observada uma baixa taxa de reconhecimento. Após isto, foi aplicado o *Threshold* para binarizar a imagem antes de efetuar o reconhecimento (SILVA, 2015).

Um dos problemas encontrado no reconhecimento foi o fato da imagem possuir outros gráficos, atrapalhando o reconhecimento. Uma das soluções aplicadas foi configurar um dicionário de palavras que poderiam ser encontradas na imagem. Caso a palavra encontrada seja similar à alguma palavra configurada no dicionário, assume-se este resultado (SILVA, 2015).

A autora apurou que o reconhecimento foi bem-sucedido em rótulos de cerveja que não tinham muitos símbolos ou fontes estilizadas. Nos testes observou-se uma taxa de reconhecimento de 60% em um ambiente claro (SILVA, 2015).

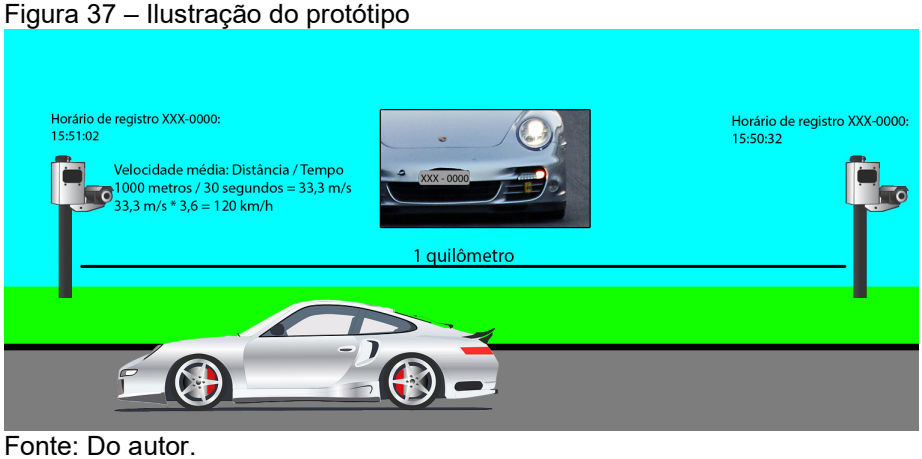
Concluiu-se que o protótipo apresentou sensibilidade quanto à resposta do reconhecimento via OCR, visto que fatores como iluminação, alinhamento e rotação do texto influenciam na taxa de reconhecimento (SILVA, 2015).

7 RECONHECIMENTO AUTOMÁTICO DE PLACAS VEICULARES UTILIZANDO A TECNOLOGIA OCR E A PLATAFORMA RASPBERRY PI APLICADA NA FISCALIZAÇÃO ELETRÔNICA DE RODOVIAS

Com o embasamento adquirido sobre os assuntos levantados, se deu o desenvolvimento de um protótipo capaz de capturar e processar fotos de veículos à fim de reconhecer a placa do mesmo.

O protótipo visa trabalhar de forma integrada com mais de um conjunto de hardware (RPI e câmera) ligados em rede distribuídos em uma rodovia. A placa veicular é única e oferece uma identidade à cada veículo e isto permitirá que o mesmo veículo seja identificado por mais de um conjunto posicionado em distâncias conhecidas. Comparando-se os registros feitos por cada módulo de um mesmo veículo, é possível verificar o horário em que o registro ocorreu e obter a velocidade média do veículo relacionando a diferença do horário com a distância entre os módulos. A figura 37 ilustra este processo.

Figura 37 – Ilustração do protótipo



Fonte: Do autor.

Visto a inviabilidade técnica e financeira de implementar o protótipo com estradas e veículos em tamanho real, foram buscadas outras soluções alternativas. Definiu-se então a construção do protótipo utilizando um autorama. A escala das placas utilizadas no protótipo foi 1:10.

7.1 METODOLOGIA

Durante o desenvolvimento do projeto de pesquisa foram realizados os levantamentos bibliográficos sobre os assuntos necessários para o desenvolvimento deste trabalho de conclusão do curso.

Primeiramente, como embasamento social ao projeto, foram levantados os problemas relacionados aos acidentes de trânsito e como o excesso de velocidade influencia nestes problemas.

Logo após, fez-se pesquisas sobre como a fiscalização eletrônica está ajudando na redução dos acidentes de trânsito e quais pontos podem ser melhorados na mesma.

Realizou-se então pesquisas sobre as particularidades da plataforma RPI escolhida para o protótipo.

Levantado a questão sobre a fiscalização eletrônica com reconhecimento de placas, pesquisou-se sobre o OCR para embasamento do assunto e como é realizada a aplicação desta tecnologia com os recursos de software existentes.

Após a finalização da fase de embasamento teórico, iniciou-se a fase de implementação do protótipo.

7.2 DESCRIÇÃO DOS RECURSOS UTILIZADOS

Durante o levantamento bibliográfico, alguns recursos necessários foram levantados, porém durante a implementação verificou-se a necessidade de alguns recursos adicionais. Os mesmos são descritos separadamente entre software e hardware.

7.2.1 Hardware

O primeiro recurso de hardware necessário para desenvolvimento do trabalho é um computador notebook HP com 8GB de memória RAM, HD de 500GB e processador Intel Core i5 de 2.6 GHz e sistema operacional Windows 8.1, este já em posse do autor antes do início do projeto. Este computador já continha um leitor

de cartões de memória, necessário para configuração do RPI.

Foi necessário adquirir com recursos do autor um computador RPI para o desenvolvimento. O modelo escolhido foi o RPI 3 por possuir um hardware mais potente, contando com um processador quad-core de 1.2 GHz e 1GB de memória RAM. Priorizou o modelo mais potente levando-se em conta o alto custo de processamento de imagens.

Em conjunto com RPI, foram adquiridos alguns acessórios necessários. O mais importante deles é uma câmera Raspicam V2. Esta câmera é nativa do RPI e optou-se pelo modelo mais novo por possuir um sensor de 8MP. Alguns outros acessórios também foram adquiridos, sendo eles:

a) Um case em acrílico;
b) Um case plástico para a câmera;
c) Três dissipadores de calor;
d) Um cooler de 4 centímetros;
e) Uma fonte de energia com capacidade de 2.2 ampères;
f) Um cartão de memória SD classe 10 de 32GB;
g) Um combo de teclado e mouse sem fio;
h) Um sensor de presença infravermelho;
i) Fios de conexão macho-macho e macho-fêmea.

Alguns acessórios não são estritamente essenciais para o desenvolvimento do projeto, porém auxiliaram no mesmo ou foram adquiridos visando a proteção e melhor funcionamento do mesmo. Os cases do RPI e da câmera foram adquiridos visto a sensibilidade dos circuitos eletrônicos. Os dissipadores de calor e o cooler foram adquiridos visto a preocupação com o aquecimento dos circuitos devido ao alto processamento necessário para trabalhar com algoritmos de visão computacional.

A fonte de energia foi escolhida baseando-se no consumo total dos acessórios tem uma capacidade maior que a fonte mais comum recomendada para o RPI. O combo de teclado e mouse sem fio foi escolhido justamente por drenar menos corrente do RPI do que um combo com fio e também pela praticidade.

O sensor infravermelho e os cabos para conectar o mesmo foram adquiridos em uma fase posterior da implementação que será descrita mais à frente.

O autorama utilizado no protótipo foi obtido de doação de conhecido do autor, necessitando apenas de uma manutenção.

No início do projeto foi necessário também um monitor com entrada HDMI para configuração inicial do RPI.

7.2.1 Software

Os recursos de software utilizados no projeto consistiram em ambientes de programação, bibliotecas de programação e programas auxiliares para o desenvolvimento.

O RPI vem de fábrica sem sistema operacional, sendo necessário a configuração do mesmo. Para isto, foi necessário utilizar um arquivo de imagem de disco do sistema *Raspian* e o software *Win32DiskImager* para gravar a imagem do sistema operacional no cartão de memória.

Para eliminar a necessidade de utilizar um monitor junto ao RPI, foi utilizado o software *TightVNC* para acessar remotamente o RPI. Isto foi essencial no desenvolvimento e teste do protótipo. Também foi utilizado o pacote Samba do Linux para permitir o compartilhamento de arquivos entre Linux e Windows pela rede.

O desenvolvimento das aplicações necessárias se deu em duas linguagens de programação diferentes: Foi utilizado a linguagem Java no ambiente NetBeans 8.1 com o JDK 1.8 e a linguagem Python na versão 2.7. A aplicação em Python pode ser escrita em qualquer editor, sendo esta desenvolvida no Notepad++.

A aplicação desenvolvida em Java utilizou as bibliotecas OpenCV na versão 3.1 e a interface Tess4J na versão 3.2.1, que utiliza a biblioteca Tesseract OCR.

A imagem *Raspian* utilizada já continha configurada o suporte para aplicações em Python na versão 2.7 e Java versão 1.7.

7.3 DESENVOLVIMENTO

O desenvolvimento do protótipo foi separado em diferentes fases para que o mesmo fosse concluído. As principais dificuldades concentraram-se na

integração das aplicações em diferentes plataformas, sendo elas Windows e Linux.

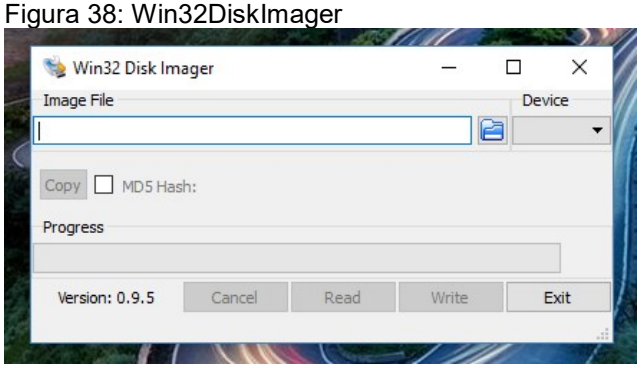
7.3.1 Configuração Raspberry Pi

A primeira fase do protótipo foi a configuração do RPI, que começou com a escolha, obtenção e instalação do sistema operacional.

O sistema operacional escolhido foi o *Raspian*, por ser o mais comum do RPI e por ser o recomendado pela *Raspberry Pi Foundation*. Como é baseado em Linux, o mesmo é *open-source* e pode ser obtido no site oficial do RPI no link https://www.raspberrypi.org/downloads/. A versão utilizada foi a *Raspian Jessie With Pixel*, que apresenta uma interface gráfica ao contrário da *Raspian Jessie Lite*. Após efetuar o download, é possível verificar um arquivo de imagem do tipo ".IMG".

Para gravar a imagem do sistema operacional no cartão pelo Windows, foi necessário obter o software *Win32DiskImager*. No mesmo, é necessário selecionar a imagem baixada anteriormente do sistema operacional e a letra correspondente da unidade leitora de cartões de memória do computador. A figura 38 mostra a interface do programa.

Figura 38: Win32DiskImager



Fonte: Do autor.

Com o sistema operacional gravado no cartão, foi possível realizar a primeira inicialização do RPI. Foi utilizado um monitor HDMI externo e um conjunto de mouse e teclado sem fio e não foi necessário realizar nenhuma configuração para que os periféricos funcionassem. O ingresso em uma rede sem fio também não requer nenhuma configuração extra.

Visto que o protótipo necessitava de portabilidade, buscou-se uma solução de acesso remoto total da plataforma que pudesse ser acessada pelo Windows e que permitisse exibir a interface gráfica do *Raspian*.

A solução encontrada foi o *TightVNC*. No RPI, a instalação do *TightVNC Server* é feita pelo terminal executando o comando: ***sudo apt-get install tightvncserver***

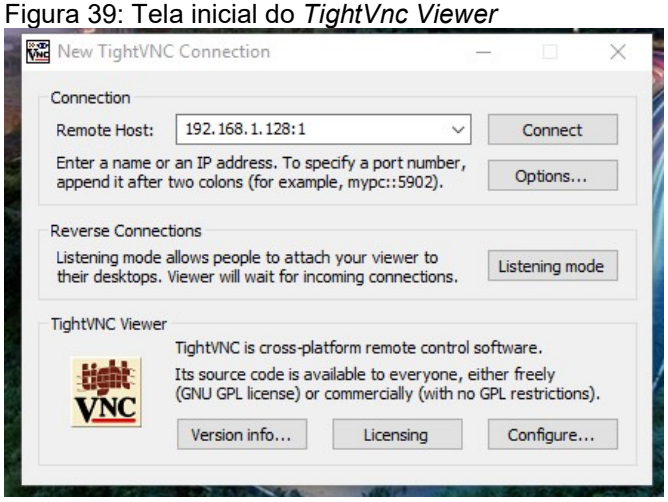
É importante ressaltar que a maioria das instruções de instalação de pacotes no Linux recomendam que antes seja executada uma atualização nos pacotes já presentes através do comando: ***sudo apt-get update***

Após a instalação do programa, é necessário rodar o servidor com o comando: ***tightvncserver***

Em seguida o comando "***vncserver :0 -geometry 1920x1080 -depth 24***" irá abrir o servidor. É possível configurar a inicialização automática do servidor no *boot* do sistema operacional, porém esta configuração não será abordada.

Após a inicialização do servidor, é possível acessar o mesmo pelo Windows através do *TightVnc Viewer*, obtido através do site http://www.tightvnc.com/download.php. A figura 39 mostra a tela inicial do *TightVnc Viewer*.

Figura 39: Tela inicial do *TightVnc Viewer*

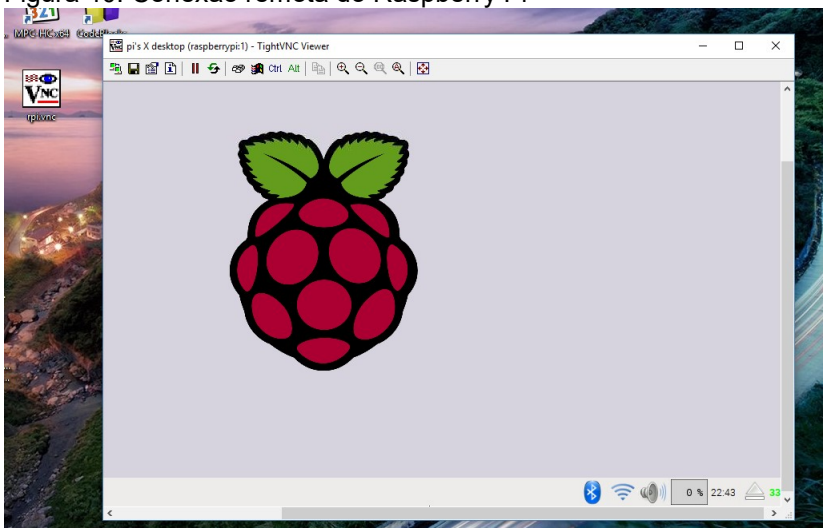


Fonte: Do autor.

Foi necessário configurar um IP fixo no RPI, visto que é o mesmo é utilizado para efetuar o acesso remoto. Após configuração do IP e acesso remoto, é

possível observar na figura 40 a interface gráfica do RPI sendo exibida no ambiente Windows.

Figura 40: Conexão remota do Raspberry Pi



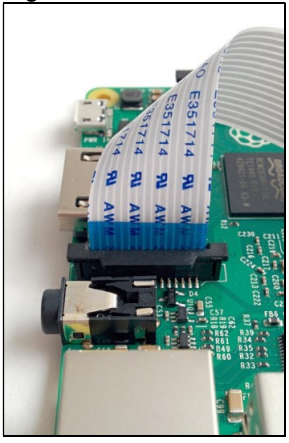
Fonte: Do Autor.

7.3.2 Captura de foto e detecção de movimento

Após efetuado a configuração do RPI, iniciou-se os testes com a *Raspicam*. Sendo um acessório desenvolvido especialmente para o RPI, a câmera possui fácil configuração e operação.

Inicialmente, com o RPI desligado, deve-se conectar a câmera à porta própria CSI do mesmo como mostra a figura 41.

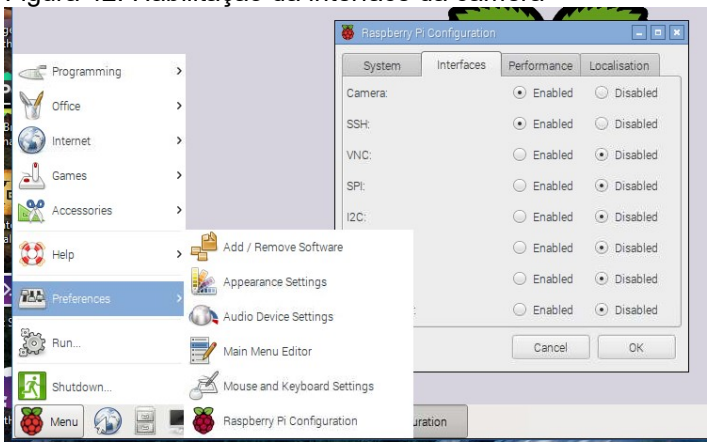
Figura 41: Conexão da câmera



Fonte: RaspberryPi.org

Com o RPI ligado, é necessário habilitar a interface da câmera através do menu *Preferences* > *Raspberry Pi Configuration* e marcar a opção "*Camera*" como "*Enabled*" na aba "*Interfaces*" como mostra a figura 42. Após efetuada a configuração, deve-se reiniciar o RPI.

Figura 42: Habilitação da interface da câmera



Fonte: Do autor.

Com isso, é possível fazer um teste básico da câmera no próprio terminal do Linux, executando o comando: ***raspistill -o /home/pi/teste.jpeg -w 1920 -h 1080 -p -f***

Caso a configuração e a montagem esteja correta, será salvo uma imagem no caminho "*/home/pi*" definido pelo parâmetro "*-o*". Os parâmetros "*-w*" e "*-h*" indicam a largura e altura da foto salva, respectivamente. Os parâmetros "*-p*" e "*-f*" indicam que antes de salvar a foto, será exibida na tela do RPI uma prévia da foto em tela cheia. Uma observação é que essa prévia só será exibida caso o RPI esteja conectado à um monitor. Ao acessar remotamente sem um monitor conectado a foto é salva, mas sem exibir a prévia.

Um dos requisitos do protótipo foi que a câmera fosse disparada assim que o veículo passasse pela mesma. Sendo assim, iniciou-se um estudo de como utilizar a própria câmera como um detector de movimento.

Por meio de pesquisa bibliográfica, encontrou-se uma solução pronta para detecção de movimento desenvolvida em Python por um usuário denominado "brainkflakes" no fórum oficial do RPI (BRAINFLAKES, 2013). Utilizou-se o *Notepad++* para salvar o código na extensão ".py" e transferir para o RPI, onde é

possível compilar e executar o programa no terminal através do comando: ***python detector.py***

Esta solução baseia-se basicamente na comparação de imagens. Enquanto estiver rodando, o programa captura imagens com resolução 100x75 seguidas e compara os *pixels* da imagem atual com a anterior. Caso seja detectada uma mudança de 20 pixels, o programa detecta que houve uma mudança no cenário e captura uma imagem em resolução 1292x972. Apenas a imagem em maior resolução é salva no disco, enquanto as imagens que são comparadas permanecem apenas em memória. Todos os valores de resolução e sensibilidade de mudança podem ser facilmente alterados.

Efetuando os testes, foi possível observar que a solução funcionou, capturando uma foto quando fosse detectado o movimento, porém, a velocidade de detecção de movimento e captura não foi satisfatória. Visto que o veículo do autorama passa em alta velocidade, a foto era capturada apenas depois que o mesmo já estava fora do enquadramento da câmera. Avaliou-se que a eficiência do algoritmo era baixa pois todas as fotos capturadas eram executadas como um outro processo utilizando o comando *raspistill*. Toda vez que o comando é executado, há um ciclo na câmera que leva no mínimo 1 segundo. Esse ciclo é constituído pela ativação da câmera, captura da foto e desativação da câmera.

Diante do problema apresentado, iniciou-se a busca por uma solução para otimizar o tempo de ciclo da câmera. Foi encontrada uma solução alternativa desenvolvida por um usuário chamado “nrother” no fórum do RPI denominada *RaspiFastCamD* (NROTHER, 2013). O mesmo disponibilizou o projeto no site *BitBucket* para ser compilado no link https://bitbucket.org/niklas_rother/rasperry-pi-userland/raw/master/host_applications/linux/apps/raspicam.

Este projeto trata de uma alternativa ao comando *raspistill* e já possui três arquivos diferentes executáveis para a interface de outros programas com a câmera. São eles:

a) *Start_camd.sh*: O mesmo serve para inicializar um processo em segundo plano que deixa a câmera no estado sempre ativo, pronta para capturar uma foto;
b) *Stop_camd.sh*: O mesmo serve para encerrar o processo iniciado

pelo comando “*start_camd.sh”* e desativar a câmera.;

c) *Do_caputure.sh:* Este comando é executado para efetuar a captura de uma foto. O mesmo envia um sinal ao processo sinalizando que uma foto deve ser capturada.

Os três comandos podem ser executados de contextos diferentes, visto que todos baseiam-se no identificador do processo para trabalhar. Após testes básicos, foi dado o início da adaptação da solução de detecção de movimento para substituir o comando “*raspistill*” pela utilização do *RaspiFastCamD*.

Durante a adaptação, foi encontrado um problema com a imagem que é capturada em sequência para realizar a comparação e definir se houve ou não movimento na câmera. A figura 43 exibe a função que captura esta imagem.

Pode-se observar que o comando “*raspistill*” é utilizado juntamente com o parâmetro “*-o*”. Este parâmetro tem dois comportamentos: Caso seja especificado um caminho após o mesmo, a imagem será salva neste caminho. Caso não seja especificado o caminho, o resultado da imagem é escrito na saída do processo. Na linha 26 é possível observar que a variável “*imageData”* receberá o conteúdo retornado pelo comando “*subprocess.check_output(command, shell=true)*”. Esta função que possibilita que as imagens para comparação sejam mantidas apenas em memória, evitando a necessidade de escrever e ler do disco.

Figura 43: Função para realizar a captura da foto para detecção de movimento

```
22  # Capture a small test image (for motion detection)
23  def captureTestImage():
24      command = "raspistill -w %s -h %s -t 0 -e bmp -o -" % (100, 75)
25      imageData = StringIO.StringIO()
26      imageData.write(subprocess.check_output(command, shell=True))
27      imageData.seek(0)
28      im = Image.open(imageData)
29      buffer = im.load()
30      imageData.close()
31      return im, buffer
32
```

Fonte: Do autor.

A figura 44 exibe uma parte do algoritmo do arquivo “*start_camd.sh*”. Nele é possível observar que o processo será iniciado com o parâmetro “-o” especificando um caminho temporário para cada foto capturada.

Figura 44: Arquivo “start_camd.sh”

```
13  output_file=${1-~/tmp_%d.jpg}
14
15  echo "Output will be written to $output_file"
16
17  #This will make pictures of 200x200px feel free to change.
18  ./raspifastcamd -w 200 -h 200 -o $output_file &
19  pid=$!
20
21  echo "Pid of raspifastcamd is $pid"
22
23  echo $pid > $pid_file
24
25  exit 0
```

Fonte: Do autor.

O primeiro problema na utilização deste algoritmo está no fato de que, uma vez executado o arquivo “*start_camd.sh*” para preparar a câmera para captura de fotos, não é possível modificar o parâmetro “-o” para escrever a imagem na saída do processo sob demanda, requisito necessário para comparar as imagens e detectar e movimento. No algoritmo de detecção de movimento era necessário salvar a imagem na memória e também no disco, quando a imagem fosse capturada pela função “*saveImage*” mostrada na figura 45.

Figura 45: Função para salvar a imagem capturada em alta resolução

```
# Save a full size image to disk
def saveImage(width, height, diskSpaceToReserve):
    keepDiskSpaceFree(diskSpaceToReserve)
    time = datetime.now()
    filename = "capture-%04d%02d%02d-%02d%02d%02d.jpg" % (time.year, time.month, time.day, time.hour, time.minute, time.second)
    subprocess.call("raspistill -w 1296 -h 972 -t 0 -e jpg -q 15 -o %s" % filename, shell=True)
    print "Captured %s" % filename
```

Fonte: Do autor.

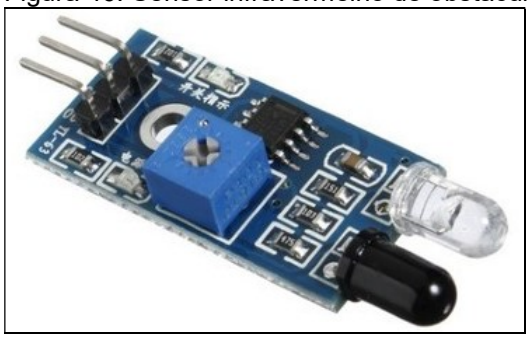
Levantou-se a possibilidade de escrever a imagem sempre em memória e na função “*saveImage*” escrever a memória em disco posteriormente, em um comando separado. Para isto, o algoritmo do arquivo “*start_camd.sh*” foi alterado para que não fosse especificado um caminho no parâmetro “-o”.

Realizada a alteração, detectou-se um segundo problema: O comando “*raspistill*” utilizado originalmente para receber o conteúdo na imagem na saída é um processo síncrono, ou seja: Quando o mesmo é executado, a câmera é preparada e captura uma foto. Em seguida a câmera é desativada é o conteúdo na foto é escrito na saída. Todo este ciclo acontece em um único processo. O arquivo “*start_camd.sh*” inicia um processo que fica rodando indefinidamente com a câmera pronta para capturar uma foto, até que o arquivo “*stop_camd.sh*” seja executado. O

arquivo “*do_caputure.sh*” que é utilizado para capturar a foto apenas envia um sinal ao processo aberto pelo arquivo “*start_camd.sh*” para que o mesmo capture uma foto. O conteúdo da foto é impresso na saída deste processo que já está rodando, e não no retorno do comando “*do_caputure.sh”*. Sendo assim, seria necessário estudar uma maneira de redirecionar a saída do processo ou capturar a mesma em outro contexto.

Diante das dificuldades apresentadas na implementação de um sensor de movimento utilizando apenas a câmera, foram iniciadas pesquisas sobre sensores de presença para serem utilizados no protótipo. O sensor escolhido para o projeto foi um sensor de obstáculo infravermelho de três pinos que pode ser observado na figura 46.

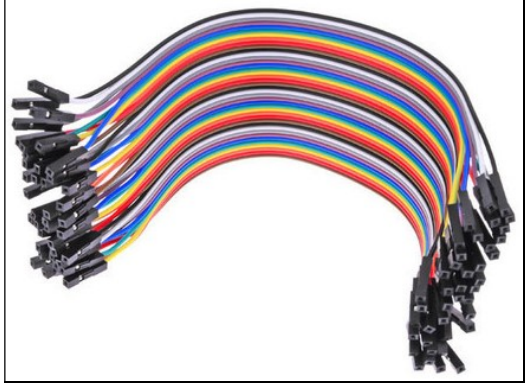
Figura 46: Sensor infravermelho de obstáculo



Fonte: FilipeFlop

Foram adquiridos juntamente ao sensor cabos para conexão do mesmo ao GPIO do RPI, mostrados na figura 47.

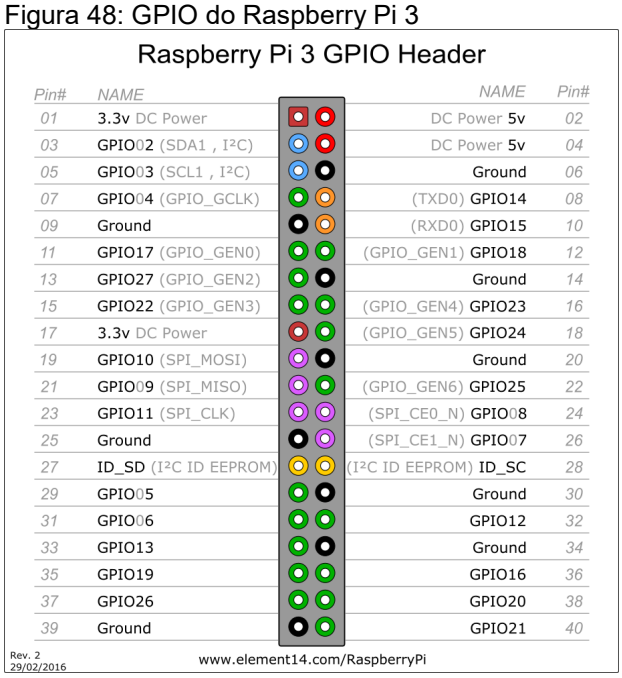
Figura 47: Cabos fêmea-fêmea para conexão do sensor



Fonte: FilipeFlop

O sensor é de simples operação, contendo pinos de alimentação (positivo e negativo) e um pino de sinal que envia o sinal 0 ao detectar um obstáculo. O mesmo foi conectado aos pinos 4 e 14 do GPIO para alimentação e ao pino 12 para sinal, que corresponde ao pino “GPIO18”. Para o protótipo não foi necessário estender o GPIO para uma outra placa ou módulo, por exemplo, pois o mesmo conta exatamente com 2 pinos de 5V que foram utilizados para alimentar o cooler e o sensor. O layout do GPIO pode ser observado na figura 48.

Figura 48: GPIO do Raspberry Pi 3



Fonte: BlogFilipeFlop

Após ligar o sensor ao RPI, foi possível observar que o mesmo possui dois LED’s SMD: Um se acende ao alimentar o sensor e o outro se acende ao detectar um obstáculo em frente aos LED’s infravermelho. Assim sendo, foi fácil configurar a sensibilidade do sensor girando o parafuso na caixa azul presente na placa do sensor.

O próximo passo foi desenvolver um algoritmo que fosse capaz de ler os pinos do GPIO à fim de detectar o sinal do sensor. Devido à facilidade e documentação vasta no RPI, a linguagem para desenvolvimento do algoritmo foi Python.

A configuração do GPIO no Python se dá pela importação de um pacote e a configuração dos pinos à serem utilizados. A figura 49 mostra um exemplo da configuração.

Figura 49: Algoritmo em Python para leitura do sinal do sensor

```
1  import RPi.GPIO as GPIO
2
3  GPIO.setmode(GPIO.BCM)
4  GPIO.setwarnings(False)
5  GPIO.setup(18, GPIO.IN)
6
7  while True:
8      if GPIO.input(18) == 0:
9          print('Obstáculo detectado')
```

Fonte: Do autor.

A linha 1 contém a importação do pacote necessário. A interação com o GPIO pode ser configurada de duas formas: Pelo número físico do pino ou pelo número do GPIO. No algoritmo acima, configurou-se a interação pelo número do GPIO na linha 3. É necessário também especificar se o pino é uma saída ou entrada de sinal. Verifica-se que o pino 18 do GPIO (que corresponde ao pino 12 no esquema físico, como pode ser observado na figura 71) foi configurado como entrada na linha 5. As linhas 7 até 9 contém um ciclo que irá imprimir uma mensagem na tela sempre que um obstáculo for detectado.

Os testes com o sensor de obstáculo foram bem-sucedidos e, sendo assim, a detecção de movimento pela câmera foi abortada.

O passo seguinte consistia em desenvolver o algoritmo completo que fosse capaz de disparar a câmera assim que o sensor capturasse um obstáculo. O algoritmo completo da aplicação de detecção de movimento e captura da foto pode ser visualizado no apêndice B.

A primeira parte do algoritmo faz as configurações iniciais do GPIO e da câmera, como pode ser visualizado na figura 50. Para a interação da câmera, foi importado o pacote “*picamera*”.

Figura 50: Configurações iniciais do algoritmo de detecção e captura da foto

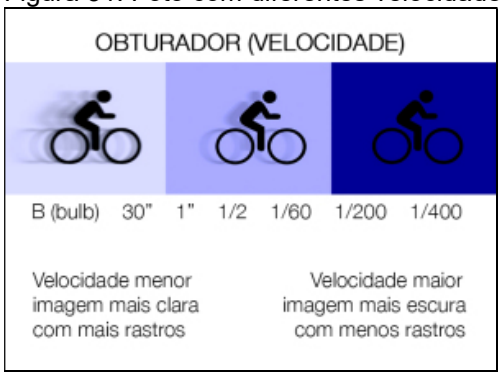
```
16  with picamera.PiCamera() as camera:
17      camera.drc_strength = 'high'
18      camera.sharpness = 100
19      camera.resolution =(1920, 1080)
20      camera.shutter_speed = 600
21      camera.iso = 800
22      time.sleep(2)
23      # Now fix the values
24      camera.exposure_mode = 'off'
25      g = camera.awb_gains
26      camera.awb_mode = 'off'
27      camera.awb_gains = g
28      GPIO.setmode(GPIO.BCM)
29      GPIO.setwarnings(False)
30      GPIO.setup(18, GPIO.IN)
31      signal.signal(signal.SIGINT, signal_handler)
32      print('Sleep de inicio...')
33      camera.start_preview()
34      time.sleep(2)
35      fotoCapturada = False
36      i = 0;
```

Fonte: Do autor.

Basicamente, são feitas configurações na câmera através da variável "*camera*" como resolução especificada em 1920x1080. Tais configurações foram ajustadas após observar as fotos capturadas. Algumas das propriedades foram configuradas após leitura da documentação da câmera disponível em http://picamera.readthedocs.io/en.

A configuração da câmera mais importante para o protótipo funcionar de maneira satisfatória foi o "*shutter_speed".* Esta propriedade trata da velocidade de disparo do obturador. Como era necessário capturar uma foto nítida do veículo que passava em alta velocidade, foi necessário alterar esta propriedade até um resultado satisfatório. A figura 51 mostra a diferença de fotos capturadas com diferentes velocidades de disparo do obturador.

Figura 51: Foto com diferentes velocidades de obturador



Fonte: foTOURgrafe

Sendo assim, a captura da foto em ambientes sem uma excelente iluminação ficou prejudicada. Na figura 52 é possível observar a diferença da foto no mesmo ambiente interno com a velocidade do obturador padrão na esquerda e com a velocidade do obturador maior na direita.

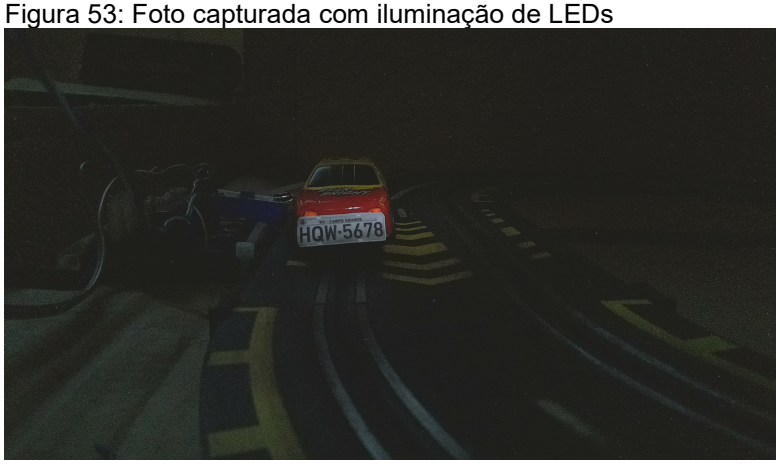
Figura 52: Fotos com diferentes velocidades de obturador

Fonte: Do Autor.

Para tentar resolver o problema da baixa iluminação da foto, foram inseridos no circuito três LEDs brancos de alto brilho. A intenção inicial seria ativar os LEDs assim que a passagem era detectada, porém seria necessário desenvolver um circuito auxiliar, visto que a saída GPIO não fornece corrente suficiente para ativar os mesmos. Outro requisito foi a tensão necessária para ativar os LEDs, visto que a tensão máxima que o RPI fornece é 5V e cada LED requer 2,5V. A solução encontrada foi ligar os LEDs na alimentação do autorama, cujos testes detectaram um pico de 8,5V a 9V.

Detectou-se uma queda de tensão nos LEDs assim que o controle do veículo é acionado, fazendo com que os mesmos não ficassem com o brilho máximo. A solução amenizou o problema em alguns ambientes, porém não resolveu totalmente o problema. Na figura 53 pode-se observar a foto capturada no mesmo ambiente da figura 52, porém com a iluminação dos LEDs.

Figura 53: Foto capturada com iluminação de LEDs



Fonte: Do Autor.

A segunda parte do algoritmo inicia o ciclo de detecção de obstáculo e captura de fotos. O mesmo pode ser observado na figura 54. Nele é possível observar que o sinal do pino 18 é lido a todo momento à fim de verificar um obstáculo no sensor. Assim que o mesmo é detectado, uma foto é capturada. Enquanto o programa está rodando, todas as fotos são salvas com o nome diferente controlada pela variável “i” que é incrementada à cada captura. Foi realizado um tratamento para que o programa não capturasse duas fotos seguidas do mesmo veículo, visto que o ciclo “*while True:*” da linha 38 é executado sem tempo de aguardo. Enquanto o pino 18 permanece com o obstáculo detectado, a foto é capturada uma vez até que o sensor esteja livre de obstáculo.

Figura 54: Ciclo de detecção de movimento e captura de fotos

```
37      print('Processo pronto!')
38      while True:
39          if GPIO.input(18) == 0:
40              if fotoCapturada == False:
41                  fotoCapturada = True
42                  i = i + 1;
43                  camera.capture('{0}.jpeg'.format(i), 'jpeg')
44                  print('Foto capturada!')
45              else:
46                  time.sleep(0.5)
47          else:
48              if fotoCapturada == True:
49                  fotoCapturada = False
50                  print('Camera estabilizada')
```

Fonte: Do autor.

Para realizar o teste, foi impresso uma placa veicular em escala e colada na dianteira do carro do autorama. Após posicionar o sensor e a câmera, foi possível

capturar fotos satisfatórias em ambientes bem iluminados, como a foto exibida na figura 55. A execução do programa é feita pelo terminal do Linux executando o comando: *python gpiopicamera.py*.

Figura 55: Foto capturada pela aplicação Python

Fonte: Do autor.

7.3.3 Localização da placa

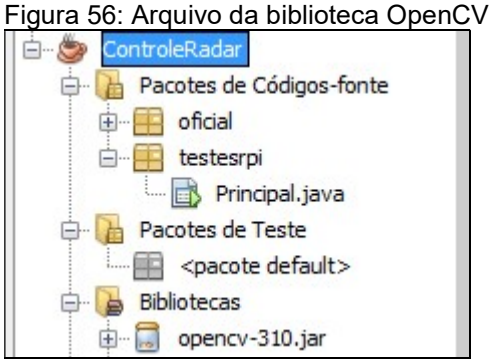
A partir de uma imagem exemplo capturada pelo RPI, começou-se o trabalho de tratamento desta imagem com a biblioteca OpenCV.

Inicialmente, foi montada uma aplicação teste que carregasse a imagem na tela para que fossem aplicados diversos filtros da biblioteca OpenCV à fim de definir a melhor estratégia de localização da placa.

Esta aplicação foi desenvolvida em Java no ambiente NetBeans 8.0.2 e contém uma classe *JForm* com um painel *JLabel* interno no qual será carregado a imagem.

De início, foi obtida a biblioteca OpenCV versão 3.1 através do link https://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.1.0/ e realizada a instalação da mesma. Após realizada a instalação, no diretório instalado há a pasta "*build*" e dentro da mesma a pasta "*java*" que conterá o arquivo necessário a ser referenciado, denominado "*opencv-310.jar*". A figura 56 exibe o arquivo configurado no projeto.

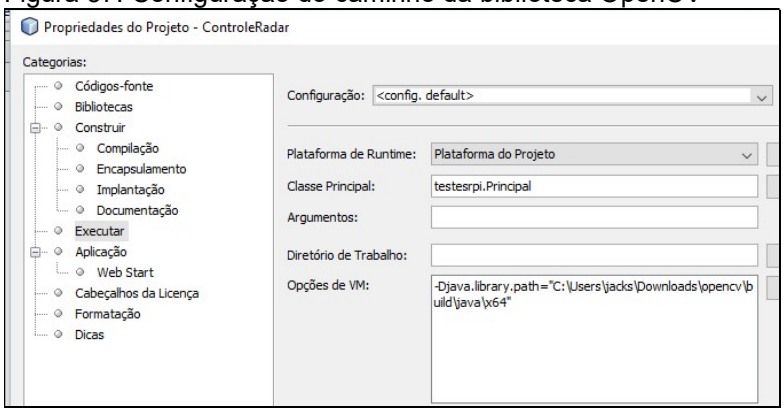
Figura 56: Arquivo da biblioteca OpenCV no projeto Java



Fonte: Do autor.

Este arquivo é uma interface Java que será o mesmo tanto no Windows, quanto no Linux, porém a biblioteca que contém as funções necessárias é específica em cada caso. No Windows, é necessário configurar o caminho da DLL "opencv_java310.dll" na inicialização do programa. No NetBeans, é possível realizar esta configuração nas propriedades do projeto. Deve-se selecionar o caminho adequado à arquitetura do sistema (32 ou 64 bits). No campo "Opções de Vm" na opção "Executar" deve-se informar: *-Djava.library.path="caminho"*. A figura 57 exibe a configuração realizada.

Figura 57: Configuração do caminho da biblioteca OpenCV



Fonte: Do Autor.

Para carregar a DLL no programa é necessário inserir o código "*System.loadLibrary(Core.NATIVE_LIBRARY_NAME)"* do qual a classe "*Core*" pertence ao pacote "*org.opencv.core*". Se tudo estiver configurado corretamente, o programará executará sem erros.

Para carregar uma imagem com a biblioteca OpenCV, é necessário

primeiro ler a imagem em um objeto do tipo "*Mat*" com o comando "*Imgcodecs.imread*". Este objeto não pode ser aplicado diretamente à tela para exibição da imagem, necessitando de uma conversão para o tipo "*BufferedImage*" para só então ser aplicado no *JLabel*. Esta conversão é feita pela função "*mat2img*". A figura 58 exibe a leitura, conversão e aplicação da imagem na tela.

Figura 58: Funções para carregamento da imagem

```
public void carregarImagemOriginal(){
    String path = isDiretorioImagens + txtArquivo.getText();
    imagemMat = Imgcodecs.imread(path, CV_LOAD_IMAGE_GRAYSCALE);
    atualizaImagem();
}
public void atualizaImagem(){
    BufferedImage image = mat2Img(imagemMat);
    ImageIcon imgIcon = new ImageIcon(image);
    jLabel1.setIcon(imgIcon);
}
public BufferedImage mat2Img(Mat in){
    BufferedImage out;
    byte[] data = new byte[in.width() * in.height() * (int)in.elemSize()];
    int type;
    in.get(0, 0, data);

    if(in.channels() == 1)
        type = BufferedImage.TYPE_BYTE_GRAY;
    else
        type = BufferedImage.TYPE_3BYTE_BGR;

    out = new BufferedImage(in.width(), in.height(), type);

    out.getRaster().setDataElements(0, 0, in.width(), in.height(), data);
    return out;
}
```

Fonte: Do autor.

Neste caso, a variável "*imagemMat*" é uma instância da classe. O primeiro parâmetro da função "*imread*" é o caminho da imagem no disco e o segundo parâmetro indica que a mesma deve ser carregada em escalas de cinza.

Após extenso estudo dos filtros disponíveis na biblioteca OpenCv e das estratégias adotadas nas bibliografias relatadas, identificou-se uma sequência de passos que permitem a localização da placa.

A imagem base para demonstração dos passos é exibida na figura 49.

Figura 59: Imagem base capturada pelo RPI

Fonte: Do autor.

Foi percebido que a região da placa contém uma grande variação de contraste, ao contrário do resto do ambiente. O primeiro filtro aplicado foi o *Median Blur* através da função “Imgproc.medianBlur”. Esta função tem três argumentos: A imagem fonte, a imagem destino e o tamanho do bloco. A imagem é sempre do tipo “*Mat”.* O tamanho do bloco especifica quantos pixels terá o lado do bloco calculado. Este valor deve ser sempre ímpar. O *Median Blur* irá calcular a intensidade dos pixels vizinhos e aplicar o valor médio no pixel central. O tamanho mínimo deve ser três e neste caso, ao calcular o valor do pixel, serão considerados os oito pixels vizinhos formando um quadrado de três pixels de altura por três de largura. Este filtro foi aplicado mais de uma vez na imagem.

Sendo a placa um polígono cinza com caracteres pretos, ao aplicar o *Median Blur*, a região da placa torna-se um polígono totalmente cinza. O resultado pode ser observado na figura 60. Em seguida, o resultado da aplicação do *Median Blur* é subtraído da imagem original em escalas de cinza através da função “*Core.subtract*” que possui três parâmetros: as duas imagens que serão subtraídas e a imagem na qual será armazenado o resultado da subtração.

Figura 60: Imagem após a aplicação do *Median Blur*

Fonte: Do Autor.

O resultado da subtração é mostrado na figura 61. É possível observar no resultado que apenas áreas de alto contraste permaneceram na imagem.

Figura 61: Subtração da imagem com *Median Blur* da imagem original

Fonte: Do autor.

Além da placa, outras áreas de alto contraste permaneceram na imagem. Com o objetivo de minimizar estas áreas, foi aplicado uma erosão através da função "*Imgproc.erode*". Esta função possui três parâmetros, sendo dois deles a imagem de entrada e imagem de saída e o terceiro um elemento que será utilizado como parâmetro da erosão. Este elemento deve ser um retângulo de 4x4 pixels e pode ser obtido pela função "*Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(4, 4))".* O resultado após a aplicação da erosão é mostrado na figura 62. É possível observar que várias áreas foram removidas da imagem, aumentando a

chance de identificar corretamente a área da placa.

Figura 62: Imagem com a erosão aplicada

Fonte: Do autor.

O próximo passo é transformar os caracteres é um único polígono. Isto pode ser obtido através de uma dilatação na imagem, operação inversa à erosão. A operação é semelhante, porém é feita através da função "*Imgproc.dilate*" e o elemento utilizado é um retângulo de 25x4 pixels. Este elemento tem a largura maior com o objetivo de expandir lateralmente os caracteres da placa. O mesmo é obtido pela função "*Imgproc.getStructuringElement(Imgproc.MORPH_RECT, new Size(25, 4))*". O resultado pode ser observado na figura 63.

Figura 63: Imagem após dilatação

Fonte: Do autor.

O próximo passo é aplicar o *Otsu Threshold* na imagem resultante, à fim

de binarizar a imagem. A função utilizada para isto é “*Imgproc.threshold”*, sendo os dois primeiros parâmetros a imagem de entrada e saída. O terceiro parâmetro é um valor que define o ponto de Threshold. No exemplo, o valor aplicado foi 55. O quarto parâmetro e o valor que será atribuído aos pixels acesos, sendo 255 neste projeto. O quinto parâmetro é o tipo de Threshold aplicado, definido como “Imgproc.THRESH_OTSU”. O resultado é mostrado na figura 64.

Figura 64: Imagem após a aplicação do *Otsu Threshold*

Fonte: Do autor.

Com o processamento aplicado anteriormente, é possível isolar grande parte da imagem mantendo apenas áreas que apresentam alto contraste. O próximo passo buscar os contornos na imagem exibida na figura 85. A busca do contorno se dá pela função “*Imgproc.findContours*”. Esta função irá fornecer um objeto do tipo “*List<MatOfPoint>*” com todos os contornos encontrados na imagem.

Para cada contorno encontrado, é possível aproximar o mesmo do polígono mais parecido com menos vértices. Isso é possível através da função “*Imgproc.approxPolyDP*”. Esta função irá aproximar o contorno de um polígono com menos vértices. A figura 65 exibe a região da placa antes de depois da aproximação.

Figura 65: Aproximação do polígono da região da placa

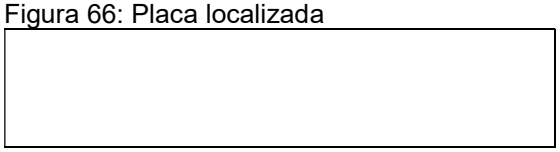
Fonte: Do autor.

É possível observar na figura 86 que o contorno identificado possui nove vértices, enquanto o ideal seria um contorno de quatro vértices. É possível considerar ou não o contorno por um número limitado de vértices, porém em alguns casos outras regiões de alto contraste podem ser identificadas incorretamente com um alto número de vértices.

Com os contornos aproximados, é realizado o cálculo da área do mesmo. Isto é possível através da função "*Imgproc.contourArea*", que tem como argumento apenas um objeto do tipo "*MatOfPoint*" com o contorno especificado. O Contorno que tiver a maior área na imagem caracteriza a região da placa.

O corte da imagem original se dá pelo desenho de um retângulo ao redor do contorno selecionado, sendo obtido pela função "*Imgproc.boundingRect*" passando o contorno do tipo "*MatOfPoint*". Este retângulo é usado então como região de interesse da imagem. Para criar uma nova imagem da placa isolada, usa-se o construtor da classe "*Mat*" passando como parâmetro a imagem base e o retângulo, retornando a imagem recortada, como exibida na figura 66.

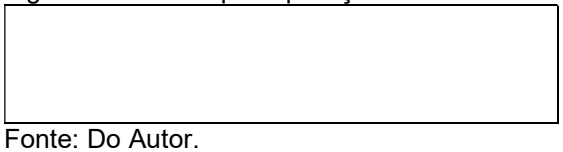
Figura 66: Placa localizada



Fonte: Do autor.

Antes de fazer o reconhecimento, é necessário binarizar a imagem resultante da placa para facilitar o mesmo. Isto é feito através da aplicação de Threshold pela função "*Imgproc.threshold*". A figura pronta para reconhecimento dos caracteres é exibida na figura 67.

Figura 67: Placa após aplicação de Threshold



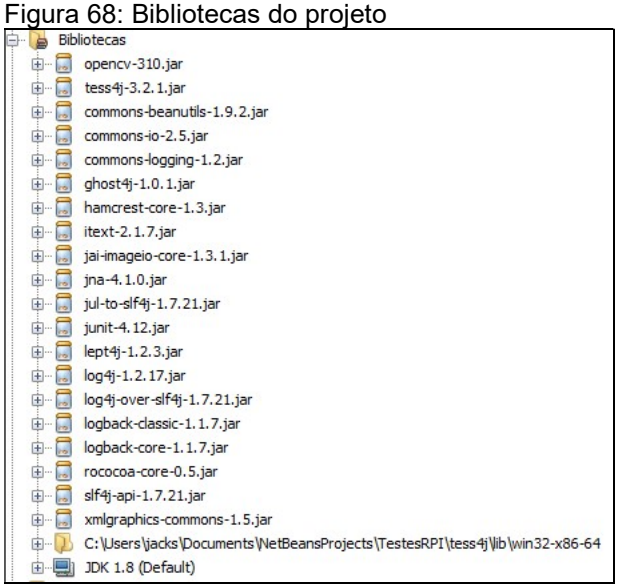
Fonte: Do Autor.

7.3.4 Reconhecimento dos caracteres

O reconhecimento dos caracteres presentes na imagem já filtrada da placa foi realizado com a biblioteca TesseractOCR através da extensão Tess4J para Java. A versão utilizada foi a 3.2.1.

A configuração da biblioteca no projeto Java se dá pela adição de referência aos arquivos necessários. O primeiro deles é o arquivo "*tess4j-3.2.1.jar*" presente na pasta "*Tess4J/dist*" que foi descompactada. Este arquivo contém funções para a interação com a biblioteca TesseractOCR. O segundo arquivo é a própria biblioteca em formato de DLL, porém referencia-se a pasta e não o arquivo. É possível fazer isto configurando o "*Djava.library.path*" já mencionado anteriormente ou adicionando a pasta nas referências do projeto. A pasta que contém estes arquivos é "*Tess4J\lib\win32-x86-64*" ou "*Tess4J\lib\win32-x86*" dependo da arquitetura do sistema operacional utilizado. Por fim, é necessário referenciar todas as bibliotecas Java de apoio presentes em "*Tess4J\lib*". A figura 68 mostra todas as referências do projeto após a configuração.

Figura 68: Bibliotecas do projeto



Fonte: Do autor.

Assim como o OpenCV, não foi possível utilizar os mesmos arquivos da biblioteca devido à diferença de sistemas operacionais e arquitetura. Foi necessário

configurar a biblioteca TesseractOCR de forma diferente no RPI, porém a mesma foi mais simples que a configuração da biblioteca OpenCV, não necessitando de compilação da mesma. A biblioteca pode ser obtida através de instalação de pacotes do Linux executando o comando: **sudo apt-get install tesseract-ocr**.

A interação com a biblioteca é relativamente simples através do Tess4J. Cria-se um objeto do tipo "*Tesseract*" e configura-se algumas propriedades antes de realizar o reconhecimento. Uma das configurações é o "*dataPath*" que indica o local que o TesseractOCR contém os arquivos "*.traineddata*". Estes arquivos contêm informações sobre os reconhecimentos efetuados, fazendo com que os próximos reconhecimentos sejam mais precisos. O reconhecimento é feito utilizando-se a função "*doOCR"* do objeto do tipo "*Tesseract"* criado. Esta função tem como parâmetro um objeto do tipo "*BufferedImage"* que pode ser obtido pela função de conversão "*mat2Img*" passando a imagem do tipo "*Mat*" correspondente a placa exibida na figura 67 e retornará uma *String* com o texto reconhecido. A figura 69 exibe a função utilizada para efetuar o reconhecimento dos caracteres.

Figura 69: Função para reconhecimento dos caracteres

```
private String OCR(Mat imagem){
    String result;
    String dataTesseract;
    if (System.getProperty("os.name").startsWith("Windows")){
        dataTesseract = "C:\\Users\\jacks\\Documents\\NetBeansProjects\\TestesRPI\\tess4j\\tessdata";
    } else {
        dataTesseract = "/usr/share/tesseract-ocr/tessdata";
    }

    try {
        Tesseract instance = new Tesseract();
        instance.setDatapath(dataTesseract);
        instance.setLanguage("eng");
        instance.setTessVariable("load_system_dawg", "F");
        instance.setTessVariable("load_freq_dawg", "F");

        result = instance.doOCR(mat2Img(imagem));
        result = result.replaceAll("[\\W]","");
    } catch (Exception e) {
        result = "Problema no OCR";
    }

    return result;
}
```

Fonte: Do Autor.

7.3.5 Cálculo da velocidade

Para calcular a velocidade do veículo, foram definidas duas classes: *RegistroPassagem* que corresponde à uma detecção e *Veiculo* que corresponde à

um veículo que pode conter várias passagens pelo radar de velocidade. O atributo que liga as duas classes é a placa reconhecida pelo programa. O programa mantém também uma lista de veículos na memória onde cada veículo contém uma lista de passagem. Sempre que é detectada uma passagem, é buscado um veículo já existente com a placa reconhecida e é adicionada uma passagem. Caso não exista um veículo correspondente, é adicionado um novo veículo. A data de modificação da foto é armazenada para cálculo da velocidade posterior. A figura 70 mostra a função que realiza este procedimento.

Figura 70: Função para a lógica de identificação dos veículos

```
RegistroPassagem registroPassagem = new RegistroPassagem();
registroPassagem.setArquivo(nomeArquivo);
BasicFileAttributes attr = Files.readAttributes(arquivo.toPath(), BasicFileAttributes.class);
registroPassagem.setDataRegistro(attr.lastModifiedTime().toMillis());
registroPassagem.setPlacaReconhecida(resultadoOCR);

Veiculo veiculo = new Veiculo();
veiculo.setPlaca(resultadoOCR);

if (veiculos.contains(veiculo)){
    veiculo = veiculos.get(veiculos.indexOf(veiculo));
    veiculo.addPassagem(registroPassagem);
} else {
    veiculo.addPassagem(registroPassagem);
    veiculos.add(veiculo);
}
```

Fonte: Do autor.

Sempre que é adicionada uma passagem para determinado veículo, uma vez que o veículo já possui duas passagens, é calculada a velocidade média do mesmo considerando-se a distância do circuito fechado do autorama e a data de registro de cada foto. A constante *“DISTANCIA_RADAR_METROS*” corresponde a 1 metro e 43 centímetros. A velocidade é calculada em metros por segundo e posteriormente transformada em km/h. A figura 71 mostra a função de cálculo da velocidade.

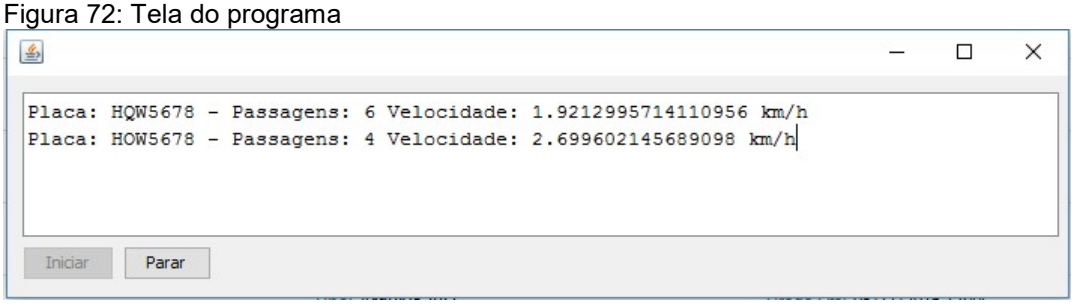
Figura 71: Função para cálculo da velocidade média do veículo

```
public void addPassagem(RegistroPassagem registroPassagem){
    long dataRegistroInicial;
    long dataRegistroFinal;
    Double diferencaSegundos;
    Double velocidadeMetrosPorSegundo;
    Double velocidadesAcumuladas = 0D;
    passagens.add(registroPassagem);

    if (passagens.size() > 1){
        for(int i = 0; i < passagens.size() -1; i++){
            dataRegistroInicial = passagens.get(i).getDataRegistro();
            dataRegistroFinal = passagens.get(i + 1).getDataRegistro();
            diferencaSegundos = ((double)(dataRegistroFinal - dataRegistroInicial)) / 1000;
            velocidadeMetrosPorSegundo = ThreadProcessamento.DISTANCIA_RADAR_METROS / diferencaSegundos;
            velocidadesAcumuladas += velocidadeMetrosPorSegundo * 3.6;
        }

        velocidadeMedia = velocidadesAcumuladas / (passagens.size() - 1);
    }
}
```

Fonte: Do autor.

Todo o processo desde o processamento da imagem até o cálculo da velocidade é realizado em uma *thread* em segundo plano definida pela classe *ThreadProcessamento*, sendo que a aplicação contém apenas uma janela com botões para parar e iniciar esta *thread* e uma caixa de texto que exibe os veículos identificados. A figura 72 mostra a tela de exibição do processamento de dez imagens.

Figura 72: Tela do programa



Fonte: Do Autor.

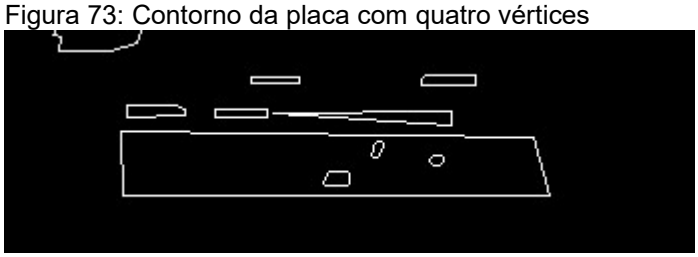
7.4 RESULTADOS OBTIDOS

Após tratamento de alguns pontos no software, foi possível identificar a placa em diferentes ambientes.

Um dos problemas que encontrou-se foi a localização incorreta da placa em algumas imagens devido a presença de outras áreas de alto contraste na imagem e pelo número de vértices do contorno da placa ser maior ou menor

dependendo do ambiente. Para amenizar este problema, implementou-se um algoritmo adaptativo para localização da placa. O valor máximo de vértices para o contorno primariamente foi definido em cinco. Após a localização da região que pode ser a placa, validou-se a proporção entre altura e largura da imagem. Considerou-se a proporção correta quando a largura da região correspondia de quatro vezes a seis vezes a altura. Em média, a razão entre largura e altura ficou entre quatro vezes e meia e cinco vezes e meia. Deste modo, foi possível localizar corretamente a região da placa que tivessem mais vértices (como a imagem exibida na figura 65) e que contivesse menos vértices, exibida na figura 73.

Figura 73: Contorno da placa com quatro vértices



Fonte: Do autor.

Outro problema encontrado foi o reconhecimento da placa em diferentes iluminações. Como estabeleceu-se um valor limite para a aplicação do *threshold* em cinquenta e cinco, nem todas as imagens tiveram a separação dos caracteres e do fundo apropriadamente. A figura 74 exibe a aplicação do *threshold* com o valor limite muito alto ou muito baixo para iluminação apresentada.

Figura 74: Aplicação incorreta de *Threshold*

Fonte: Do autor.

Esta aplicação incorreta causa o reconhecimento incorreto da placa. Foram realizados dois tratamentos para atenuar tal problema. O primeiro deles foi na saída do resultado do OCR, onde foi aplicada uma expressão regular para eliminar

todos os caracteres diferentes de números e de letras maiúsculas.

Após isto, aplicou-se uma validação na saída também com uma expressão regular, garantindo que a saída fosse no formato da placa, com três letras seguidas de quatro números. Caso a saída não esteja formatada desta maneira, é realizada uma nova aplicação de *Threshold* na imagem da placa com o valor limite ajustado. A cada tentativa, o valor limite é ajustado em uma direção diferente, ou seja, caso o reconhecimento após a aplicação do *Threshold* com o valor limite 55 seja inválido, a próxima tentativa é efetuada com o valor limite ajustado paara 56. Caso novamente o formato seja inválido, a próxima tentativa é efetuada com o valor limite ajustado 54. As tentativas posteriores terão os valores 57, 53, 58, 52 até que o resultado seja três letras seguidas de quatro números ou que o valor limite seja 0 ou 255 ou que o programa já tenha efetuado 100 tentativas de reconhecimento da placa, onde o programa trata a placa como “Não reconhecida”. A função realiza isto é exibida na figura 75.

Figura 75: Algoritmo de ajuste do valor limite de *Threshold*

```
while(!resultadoOCR.matches("[A-Z][A-Z][A-Z][0-9][0-9][0-9][0-9]")){
    System.out.println("Formato invalido. Ajustando thresh para " + ajusteThresh.toString());
    imagemPlaca = imagemPlacaOriginal.clone();
    Imgproc.threshold(imagemPlaca, imagemPlaca, THRESH + ajusteThresh, MAX_VALUE, THRESH_BINARY);
    salvarImagem(arquivo, imagemPlaca, "passo7-thresholdBinary-tentativa" + quantidadeTentativasOCR);
    resultadoOCR = OCR(imagemPlaca);
    quantidadeTentativasOCR++;

    if (quantidadeTentativasOCR > QUANTIDADE_TENTATIVAS_OCR){
        resultadoOCR = "Placa não reconhecida";
        break;
    }

    ajusteThresh *= -1;
    if ((quantidadeTentativasOCR & 1) == 1){
        ajusteThresh += AJUSTE_THRESHOLD;
    }

    if (((THRESH + ajusteThresh) > 255) || (((THRESH + ajusteThresh) < 0))) {
        resultadoOCR = "Placa não reconhecida";
        break;
    }
}
```

Fonte: Do Autor.

Com esta adaptação, foi possível identificar de maneira mais precisa imagens com diferentes níveis de iluminação.

Para realizar os testes, foram utilizadas as placas impressas exibidas na figura 76.

Figura 76: Placas utilizadas nos testes

Fonte: Do Autor.

Foram realizadas cerca de 50 passagens para todas as placas em um ambiente externo com boa iluminação. A tabela 3 exibe os resultados para a placa HQW5678.

Tabela 3 - Resultado de 50 passagens com a placa HQW5678

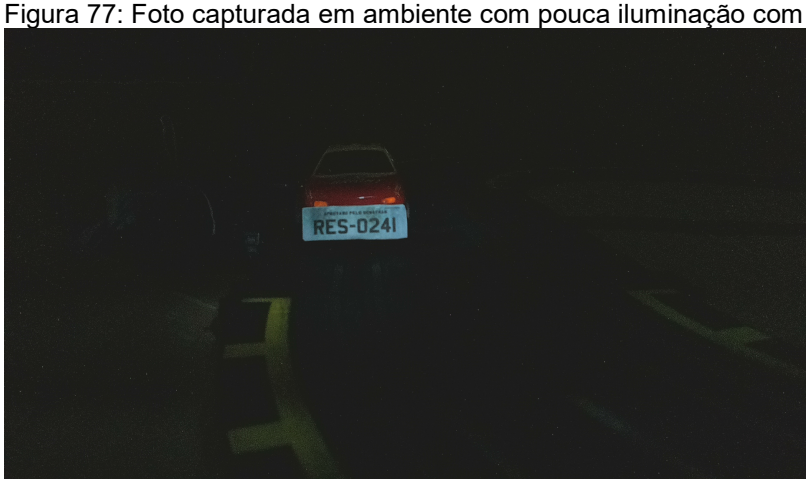
Placa reconhecida	Quantidade de vezes
HOW5678	18
HQW5678	17
HAW5678	3
HEW5678	2
HOW5578	1
HEM5678	1
Falha na localização da placa	7

Fonte: Do autor.

Resultados para as demais placas podem ser observados no apêndice A.

Foram realizados testes também em ambiente interno com iluminação menos intensa, porém com o auxílio dos LEDs. A taxa de sucesso foi bem menor neste caso, visto que percebeu-se que nem sempre o facho de luz dos LEDs focou na placa. A figura 77 mostra uma foto capturada com o auxílio dos LEDs cujo reconhecimento obteve sucesso.

Figura 77: Foto capturada em ambiente com pouca iluminação com auxílio de LEDs



Fonte: Do autor.

Foi percebido que o parâmetro de *Threshold* deve variar conforme o ambiente. O algoritmo adaptativo teve sucesso em alguns casos, porém com o custo de performance visto que o processamento é realizado mais vezes para a mesma imagem.

Em alguns casos também foi possível observar uma identificação errada dos caracteres. Alguns caracteres com semelhança como a letra “O” e o número “0” , a letra “G” e o número “6” e a letra “I” com o número “1” podem causar a identificação errada. Foi desenvolvida uma função para tratar alguns desses caracteres baseado na posição identificada. Caso fossem identificados número nas três primeiras posições, as mesmas eram substituídas por letras correspondentes. Nas quatro últimas posições, caso fosse identificada alguma letra, as mesmas eram substituídas por números. Assim, uma placa identificada incorretamente com “JP6DDDI” era substituída por “JPG0001”.

8 CONCLUSÃO

O propósito deste trabalho de conclusão foi baseado em um grave problema de saúde pública que são os acidentes de trânsito, que toma 1.2 milhões de vidas anualmente no mundo.

Diante dos levantamentos bibliográficos realizados, foi possível implementar um protótipo em escala para medição da velocidade média dos veículos.

Um dos primeiros pontos observados na implementação do protótipo foi a capacidade da câmera utilizada. Percebeu-se que a *RaspiCam* não possui uma boa capacidade para capturar fotos de objetos em alta velocidade com nitidez. A mesma possui operação simples e é interessante para vários projetos, porém foi necessário fazê-la atender os requisitos do projeto, que eram um rápido disparo e a captura da imagem sem rastros. Alterando a velocidade de obturador da mesma houve uma grande perda na iluminação da foto.

Foram implantados LEDs para compensar a falta de iluminação. Os mesmos amenizaram o problema, mas não resolveram totalmente além de estarem subalimentados.

A etapa de processamento da imagem foi aquela que reteve maior tempo da pesquisa, na qual foi necessário explorar a biblioteca OpenCV à fim de conhecer vários filtros da mesma. A localização automática da placa na imagem foi resultado de uma associação de vários filtros desta biblioteca. As primeiras tentativas da localização da placa foram realizadas na tentativa da busca dos objetos geométricos na imagem, com o objetivo de localizar um objeto retangular, porém sem sucesso. O algoritmo final utilizado detecta regiões de alto contraste, visto que a região da placa possui caracteres pretos sobre um fundo cinza. Percebeu-se uma sensibilidade deste algoritmo caso haja outras regiões de alto contraste na imagem.

Observou-se que a localização da placa e o reconhecimento de caracteres depende de vários parâmetros que podem ser influenciados pelo ambiente no qual o protótipo é instalado, principalmente em relação à iluminação e aos objetos ao redor da pista.

A utilização da biblioteca Tesseract OCR na fase de reconhecimento dos

caracteres mostrou a capacidade da mesma. Após a localização da placa, é aplicada apenas a binarização na placa resultante antes de fazer o reconhecimento. A biblioteca Tesseract OCR mostrou que é capaz de lidar com caracteres com pequenos ruídos e inclinados, fazendo o reconhecimento da placa como um todo. Esta facilidade traz consigo alguns problemas na precisão do reconhecimento.

Com o desenvolvimento do protótipo, foi possível atingir os objetivos estabelecidos e construir um modelo em escala de uma fiscalização eletrônica baseada na velocidade média dos veículos, objetivando a reeducação dos motoristas e a preservação de vidas no trânsito real.

Para trabalhos futuros, algumas melhorias no projeto são:

a) Desenvolver um tipo de inteligência artificial para que o programa pudesse se adaptar à cada local instalado e treinar a configuração dos parâmetros de localização da placa e reconhecimento dos caracteres;
b) Realizar testes com versão *NoIR* da *RaspiCam*, esta utilizada para capturar fotos em ambientes sem iluminação;
c) Implantar um circuito próprio de LEDs com alimentação adequada e ativar os mesmos apenas na detecção de passagem do veículo;
d) Melhorar o processo de localização da placa ou adotar uma outra estratégia menos sensível e mais precisa;
e) Desenvolver a capacidade de capturar a passagem de dois veículos ao mesmo tempo, adaptando o hardware inserindo mais um sensor e o software para buscar duas regiões de placas;
f) Aplicar um pré-processamento na região da placa antes de aplicar o OCR efetuando a limpeza de ruídos, correção da inclinação e segmentação dos caracteres efetuando o reconhecimento de cada caractere separado;
g) Utilizar uma câmera mais adequada aos requisitos necessários de velocidade de obturador e ISO;

REFERÊNCIAS

ADA, Lady. **Adafruit PiTFT 3.5" Touch Screen for Raspberry Pi**. 2016. Disponível em: <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-pitft-3-dot-5-touch-screen-for-raspberry-pi.pdf>. Acesso em: 26 jun. 2016.

ADHVARYU, Rachit Virendra. Optical Character Recognition Using Template Matching (Alphabet & Numbers). **International Journal Of Computer Science Engineering And Information Technology Research (ijcseitr).** [s.i], p. 227-232. out. 2013. Disponível em: <https://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwiT-rSa8b7LAhUGQpAKHYxuDf4QFggiMAA&url=http://www.tjprc.org/download.php?fname=2-14-1381407981-29.%20Optical%20character.full.pdf&usg=AFQjCNEoLzPyJv5iLA1cmC91WZNPIoxNww&sig2=ieZD3dsdSDE_jS0Vx8xV4A&bvm=bv.116636494,d.Y2I&cad=rja>. Acesso em: 12 mar. 2016.

ALGINAHI, Yasser. **Preprocessing Techniques in Character Recognition**. 2010. Disponível em: <http://cdn.intechopen.com/pdfs/11405.pdf>. Acesso em: 26 jun. 2016.

ASTC. **EDITAL DE LICITAÇÃO Nº. 064/2011.** 2011. Disponível em: <http://www.astc.sc.gov.br/web/arquivos/files/EDITAL 064-2011.pdf>. Acesso em: 12 mar. 2016.

BACCHIERII, Giancarlo; BARROSI, Aluísio J D. Acidentes de trânsito no Brasil de 1998 a 2010: muitas mudanças e poucos resultados**. Rsp - Revista de Saúde Pública da Usp**, São Paulo, v. 5, n. 45, p.949-463, 2011. Disponível em: <http://www.scielo.br/pdf/rsp/v45n5/2981.pdf>. Acesso em: 26 abr. 2016.

BIENIECKI, Wojciech; GRABOWSKI, Szymon; ROZENBERG, Wojciech. Image Preprocessing for Improving OCR Accuracy. **Memstech**. Ucraina, p. 75-80. jan. 2007.

BOYD, Sarah et al. **A Teacher's Guide to the Intel® GALILEO**. North Ryde: Macquarie Ict Innovations Centre, 2015. 80 p.

BRAINFLAKES. **Lightweight python motion detection.** 2013. Disponível em: <https://www.raspberrypi.org/forums/viewtopic.php?t=45235>. Acesso em: 17 set. 2016.

BRADSKI, Gary; KAEHLER, Adrian. **Learning OpenCV**: Computer Vision with the OpenCV Library. Sebastopol: O'reilly, 2008. 577 p.

CANNELL, Alan E. R.; GOLD, Philip Anthony. **Reduzindo acidentes**: o papel da fiscalização de trânsito e do treinamento de motoristas. Washington: Idb Bookstore, 2001. 86 p.

CANNELL, A. E. R. Inovações na fiscalização de trânsito em Argentina, Brasil, Chile e Uruguai. 2000. Disponível em: <https://www.academia.edu/1157758/Reduzindo_acidentes_o_papel_da_fiscaliza%C3%A7%C3%A3o_de_tr%C3%A2nsito_e_do_treinamento_de_motoristas?auto=download>. Acesso em: 26 jun. 2016.

CHARLES, Pranob K et al. A Review on the Various Techniques used for Optical Character Recognition. **International Journal Of Engineering Research And Applications (IJERA)**. [s. l.], p. 659-662. 1 jan. 2012.

CHOPRA, Shalin A. et al. Optical Character Recognition. **International Journal Of Advanced Research In Computer And Communication Engineering**. [S. l.], p. 4956-4958. 01 jan. 2014.

COLEY, Gerald. **BeagleBone Black System Reference Manual**. 2014. Disponível em: <https://cdn-shop.adafruit.com/datasheets/BBB_SRM.pdf>. Acesso em: 26 jun. 2016.

DALAL, Saprem; D'SOUZA, Aaron. **Automatic License Plate Recognition System**. 2011. 39 f. TCC (Graduação) - Curso de Computer Science And Engineerring, National Institute Of Technology Rourkela, Orissa, 2011.

DELANEY, Amanda; WARD, Heather; CAMERON, Max. **The History and Development of Speed Camera Use**. 2005. 56 f. Tese (Doutorado) - Curso de Accident Research Centre, University College London, Londres, 2006.

ELMORE, Megan; MARTONOSI, Margaret. **A Morphological Image Preprocessing Suite for OCR on Natural Scene Images**. 2008. Disponível em: <http://archive2.cra.org/Activities/craw_archive/dmp/awards/2008/Elmore/melmore_dmp.pdf>. Acesso em: 26 jun. 2016.

EIKVIL, Line. **Optical Character Recognition**. 1993. Disponível em: <https://www.nr.no/~eikvil/OCR.pdf>. Acesso em: 26 jun. 2016.

ELVIK, Rune et al. **The handbook of Road Safety Measures**. 2. ed. Bingley: Emerald, 2009. 1124 p.

FARNELL. **USB WiFi (802.11b/g/n) Module: For Raspberry Pi and more**. Disponível em: <http://www.farnell.com/datasheets/1848503.pdf>. Acesso em: 26 jun. 2016.

FONSECA, Erika G. P. da; LAVEGA, Alexandre S. de. Tutorial sobre introdução a projetos utilizando o kit de desenvolvimento arduino. In: Congresso Brasileiro da Educação em Engenharia, 39., 2011, Blumenau. **Artigo**. Niterói: Universidade Federal Fluminense – Escola de Engenharia, 2011. p. 1 - 7.

FOUNDATION, Raspberry Pi (Org.). **The Raspberry Pi**: Education Manual. 2012. Disponível em: <http://pi.cs.man.ac.uk/download/Raspberry_Pi_Education_Manual.pdf>. Acesso em: 30 ago. 2015.

FOUNDATION, Raspberry Pi. **Camera module**. Disponível em: <https://iprototype.nl/docs/raspberry-pi-camera-module-datasheet.pdf>. Acesso em: 26 jun. 2016.

FOUNDATION B, **Raspberry Pi 3 Model B**. Disponível em: <https://www.inet.se/files/pdf/1974044_0.pdf> Acesso em: 26 jun. 2016.

FTDICHIP (Org.). **Future Technology Devices International Ltd**: RPi HUB Module Datasheet. Disponível em: <http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_RPi_HUB_Module.pdf>. Acesso em: 26 jun. 2016.

FURBER, Steve. **ARM System-on-Chip Architecture**. Londres: Addison-wesley Professiona, 2000. 432 p.

GOLDEN, Rick. **Raspberry Pi Networking Cookbook**. Mumbai: Packt, 2013. 191 p. Disponível em: <http://daz-pi.com/ebooks/Raspberry Pi Networking Cookbook - Rick Golden.pdf>. Acesso em: 26 jun. 2016.

GUALBERTO, Daniel Rocha. **Em rumo a um sistema automático de controle de acesso de veículos automotivos: reconhecimento de caracteres em placas de veículos**. 2010. 64 f. TCC (Graduação) - Curso de Ciência da Computação, Universidade Federal de Ouro Preto, Ouro Preto, 2010.

GUPTA, Maya R.; JACOBSON, Nathaniel P.; GARCIA, Eric K.. OCR binarization and image pre-processing for searching historical documents. **The Journal Of The Pattern Recognition**. [S. l.], p. 389-397. jan. 2007.

GUPTA, Dr. Deepa; NAIR, Leema Madhu. Improving ocr by effective pre-processing and segmentation for devanagiri script:a quantified study. **Journal Of Theoretical And Applied Information Technology**. Karnataka, p. 142-153. 20 jun. 2013.

HORA, Tomas; NV, Imas. **UNIPI technical documentation**. 2015. Disponível em: <http://unipi.technology/wp-content/uploads/manual_en.pdf>. Acesso em: 26 jun. 2016.

INTEL. **Intel® Galileo**. 2014. Disponível em: <http://download.intel.com/support/galileo/sb/galileo_boarduserguide_330237_001.pdf>. Acesso em: 26 jun. 2016.

IVANSKI, William; SILVA, Luciano; BELLON, Olga R. P.. **Método Híbrido de Reconhecimento Ótico de Caracteres**. 2008. Disponível em: <http://www.gpec.ucdb.br/sibgrapi2008/wuw/47964.pdf>. Acesso em: 26 jun. 2016.

JORGE, Maria Helena Prado de Mello. Acidentes de trânsito no Brasil: um atlas de sua distribuição. In: **Acidentes de trânsito no Brasil: um atlas de sua distribuição.** Associação Brasileira de Medicina de Tráfego, 2013. Disponível em: <http://www.abramet.com.br/files/acidentes_de_transito_atlas_2013.zip>.Acesso em: 06 mar. 2016

KELLY, Hannah (Ed.). **Raspberry PI for Beginners**. Londres: Imagine Publishing, 2014.

KRIDNER, Jason. **BeagleBone Black opensource Linux™ computer unleashes innovation**. 2013. Disponível em: <http://www.ti.com/lit/wp/spry235/spry235.pdf>. Acesso em: 26 jun. 2016.

LOPES, Maria Margaret Bastos; PORTO JUNIOR, Walter. Fiscalização eletrônica da velocidade de veículos no trânsito: caso de Niterói. In: CLATPU – congresso latino-americano transporte público e urbano, 14., 2007, Rio de Janeiro. **Anais**... . Rio de Janeiro: XIV CLATPU – Congresso Latino-americano Transporte Público e Urbano – Anais em Cd-rom – Rio/rj, 2007. p. 20 - 24.

MARCILIO, Claiton de Melo. **Protótipo de aplicativo android para extração de texto em imagens e conversão em voz, orientado ao apoio a leitura para deficientes visuais**. 2013. 82 f. TCC (Graduação) - Curso de Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense - Unesc, Criciúma, 2013.

MARÍN, Letícia; QUEIROZ, Marcos S.. **A atualidade dos acidentes de trânsito na era da velocidade: uma visão geral.** 2000. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0102-311X2000000100002>. Acesso em: 20 abr. 2016.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport, 1999. 331 p.

MCMANUS, Sean; COOK, Mike. **Raspberry Pi for Dummies**. New Jersey: Hoboken, 2013. 412 p.

MCROBERTS, Michael. Arduino Básico. São Paulo: Novatec, 2011.

MING, Sun Hsien. **Fiscalização eletrônica de trânsito**. 2006. Disponível em: <http://www.sinaldetransito.com.br/artigos/fiscalizacao-eletronica-do-transito.pdf>. Acesso em: 15 maio 2016.

MITHE, Ravina; INDALKAR, Supriya; DIVEKAR, Nilam. Optical Character Recognition. **International Journal Of Recent Technology And Engineering**. [S.l.], p. 72-75. mar. 2013.

MOLLAH, Ayatullah Faruk et al. Design of an Optical Character Recognition System for Camerabased Handheld Devices. **International Journal Of Computer Science**

Issues. Kolkata, p. 283-289. 1 jun. 2011.

NAGY, George; NARTKER, Thomas A.; RICE, Stephen V.. Optical Character Recognition: An illustrated guide to the frontier. **Procs. Document Recognition And Retrieval**. [s. L.], p. 58-69. jan. 2000.

NARDI, Eduardo Amadeo de Carli et al. **Sistema de Controle de Acesso Utilizando Reconhecimento de Caracteres em Placa de Automóvel.** Curitiba, 2015. 32 p. Disponível em: <http://paginapessoal.utfpr.edu.br/msergio/portuguese/ensino-de-fisica/oficina-de-integracao-ii/Placas-trabalho_escrito_principal.pdf/view>. Acesso em: 03 dez. 2016.

NORRIS, Donald. **Raspberry Pi Projects for the Evil Genius**. Nova York: Codemantra, 2014. 288 p

NROTHER. **RaspiFastCamD: A daemonized and FAST version of raspistill.** 2013. Disponível em: <https://www.raspberrypi.org/forums/viewtopic.php?f=43&t=50075>. Acesso em: 8 out. 2016.

O'GORMAN, Lawrence; KASTURI, Rangachar. **Document image analysis**. Los Alamitos, CA: IEEE Computer Society Press, 1997.

PEDEN, **Margie et al. World report on road traffic prevention.** Genova: World Healt Organization, 2004. 244 p. Disponível em: <http://whqlibdoc.who.int/publications/2004/9241562609.pdf?ua=1>. Acesso em: 7 mar. 2016.

PICCHIONI, Iara et al. Percepção de Risco e Velocidade: A lei e os Motoristas. **Psicologia Ciência e Profissão**, Curitiba, v. 4, n. 27, p.730-745, maio 2007.

PIMODULES. **UPiS - Uninterruptible Power intelligent Supply**. Disponível em: <http://pimodules.com/_pdf/UPiS Module.pdf>. Acesso em: 26 jun. 2016.

RICHARDSON, Mat T; WALLACE, Shawn. **Primeiros Passos com o Raspberry Pi**. São Paulo: Novatec, 2013.

SMITH, Ray. **An Overview of the Tesseract OCR Engine**. Disponível em: <http://static.googleusercontent.com/media/research.google.com/pt-BR//pubs/archive/33418.pdf>. Acesso em: 26 jun. 2016.

SHOP, Multilógica (Org.). **Arduino Guia Iniciante**. Santo André: Documento Online, 2014. 149 p. Disponível em: <https://multilogica-shop.com/download_guia_arduino>. Acesso em: 26 jun. 2016.

SILVA, Luana Gomes. **Protótipo de Aplicativo para Extração de Texto em Imagens para Busca Semântica Sobre Rótulos de Cervejas.** 2015. 87 f. TCC

(Graduação) - Curso de Ciência da Computação, Unacet, Universidade do Extremo Sul de Santa Catarina, Criciúma, 2015.

SINGH, Pritpal; BUDHIRAJA, Sumit. Feature Extraction and Classification Techniques in O.C.R. Systems for Handwritten Gurmukhi Script – A Survey. **International Journal Of Engineering Research And Applications (IJERA)**. [s. L.], p. 1736-1739. jan. 201?.

SLOSS, Andrew N.; SYMES, Dominic; WRIGHT, Chris**. ARM System Developer's guide**: Designig and optimizing system software. Amsterdam: Elsevier, 2004. 689 p.

THE MAGPI: A Magazine for Raspberry Pi Users. California: The Magpi, 14 jul. 2013.

THE MAGPI: A Magazine for Raspberry Pi Users. California: The Magpi, 15 maio. 2016.

THE WORLD BANK. **Population, total.** 2016. Disponível em: <http://data.worldbank.org/indicator/SP.POP.TOTL/countries/BR?display=default>. Acesso em: 06 mar. 2016.

THIELEN, Iara Picchioni; RTMANN, Rica Rdo Carlos Ha; RES, Diogo Picchioni Soa. **Percepção de risco e excesso de velocidade**. 2008. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0102-311X2008000100013>. Acesso em: 30 jan. 2016.

WORLD HEALTH ORGANIZATION**. Global status report on road safety.** 2015. Disponível em: <http://www.who.int/violence_injury_prevention/road_safety_status/2015/GSRRS2015_Summary_EN_final2.pdf?ua=1>. Acesso em: 06 mar. 2016.

WORLD HEALTH ORGANIZATION**. Global status report on road safety.** 2013. Disponível em: <http://www.who.int/violence_injury_prevention/road_safety_status/2013/en/ >. Acesso em: 06 mar. 2016.

WORLD HEALTH ORGANIZATION,. **Gestão da velocidade**: Um manual de segurança viária para gestores e profissionais da área. Washington: Global Road Safety Partnership, 2008. 175 p.

ZHOU, Huiyu; WU, Jiahua; ZHANG, Jianguo. **Digital Image Processing**. [S.l]: Book Boon, 2010. 71 p.

APÊNDICES

APÊNDICE A – Resultados do reconhecimento de placas

PLACA BCD3456

```
Placa: BUN3455 - Passagens: 3 Velocidade: 0.11869866131707882 km/h
Placa: BCD3455 - Passagens: 2 Velocidade: 0.3856179775280899 km/h
Placa: ECU3456 - Passagens: 2 Velocidade: 0.05928826442473799 km/h
Placa: Placa não localizada - Passagens: 12 Velocidade: 1.3156455550681367 km/h
Placa: BED3456 - Passagens: 10 Velocidade: 1.0905210661932903 km/h
Placa: MUN3455 - Passagens: 2 Velocidade: 0.0981880602708373 km/h
Placa: BED3455 - Passagens: 2 Velocidade: 0.15159010600706713 km/h
Placa: BED3458 - Passagens: 3 Velocidade: 0.48915203211239333 km/h
Placa: BAN3455 - Passagens: 1 Velocidade: 0.0 km/h
Placa: BEN3455 - Passagens: 1 Velocidade: 0.0 km/h
Placa: LAM3455 - Passagens: 1 Velocidade: 0.0 km/h
Placa: BTU3456 - Passagens: 2 Velocidade: 0.20867450344548033 km/h
Placa: HAL3456 - Passagens: 1 Velocidade: 0.0 km/h
Placa: BTU3455 - Passagens: 2 Velocidade: 0.2416901408450704 km/h
Placa: ECU3455 - Passagens: 1 Velocidade: 0.0 km/h
Placa: Placa não reconhecida - Passagens: 1 Velocidade: 0.0 km/h
Placa: IEC0456 - Passagens: 1 Velocidade: 0.0 km/h
Placa: ELY1943 - Passagens: 1 Velocidade: 0.0 km/h
```

PLACA CXP0000

```
Placa: CXP0000 - Passagens: 3 Velocidade: 0.17069629559360622 km/h
Placa: EXP0000 - Passagens: 15 Velocidade: 1.0792445640493493 km/h
Placa: Placa não localizada - Passagens: 20 Velocidade: 2.487410896971397 km/h
Placa: Placa não reconhecida - Passagens: 7 Velocidade: 1.6184520855055675 km/h
Placa: EXP1000 - Passagens: 2 Velocidade: 3.386842105263158 km/h
Placa: FII1111 - Passagens: 1 Velocidade: 0.0 km/h
Placa: EXP0006 - Passagens: 1 Velocidade: 0.0 km/h
Placa: LEI9511 - Passagens: 1 Velocidade: 0.0 km/h
```

PLACA JPG0001

```
Placa: UPC8001 - Passagens: 1 Velocidade: 0.0 km/h
Placa: JPC0001 - Passagens: 13 Velocidade: 1.666703614519691 km/h
Placa: Placa não localizada - Passagens: 10 Velocidade: 1.039651649603528 km/h
Placa: JPG0001 - Passagens: 2 Velocidade: 2.475 km/h
Placa: JPC6001 - Passagens: 2 Velocidade: 0.07140182249407065 km/h
Placa: JFK8801 - Passagens: 1 Velocidade: 0.0 km/h
Placa: JPU6001 - Passagens: 1 Velocidade: 0.0 km/h
Placa: JPE0001 - Passagens: 3 Velocidade: 0.7924772907663815 km/h
Placa: JPL0001 - Passagens: 2 Velocidade: 0.24893617021276598 km/h
Placa: JPB0001 - Passagens: 1 Velocidade: 0.0 km/h
Placa: JPC6601 - Passagens: 1 Velocidade: 0.0 km/h
Placa: JFC0001 - Passagens: 1 Velocidade: 0.0 km/h
Placa: JPS6001 - Passagens: 1 Velocidade: 0.0 km/h
Placa: MIL1380 - Passagens: 1 Velocidade: 0.0 km/h
Placa: JPE6001 - Passagens: 1 Velocidade: 0.0 km/h
Placa: JFE0001 - Passagens: 1 Velocidade: 0.0 km/h
Placa: JPU0000 - Passagens: 1 Velocidade: 0.0 km/h
Placa: JRE0001 - Passagens: 1 Velocidade: 0.0 km/h
Placa: JPD6601 - Passagens: 1 Velocidade: 0.0 km/h
```

PLACA NQC0876

```
Placa: Placa não localizada - Passagens: 3 Velocidade: 0.11787086690784006 km/h
Placa: NUL3875 - Passagens: 1 Velocidade: 0.0 km/h
Placa: HUI3838 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NUI1876 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NHL3876 - Passagens: 3 Velocidade: 1.658908057641567 km/h
Placa: NHL8878 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NUT1826 - Passagens: 1 Velocidade: 0.0 km/h
Placa: MAI3876 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NUT3878 - Passagens: 3 Velocidade: 0.5423155526992287 km/h
Placa: NUT3336 - Passagens: 1 Velocidade: 0.0 km/h
Placa: INC0876 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NWO8761 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NQL3878 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NHL8838 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NHL3878 - Passagens: 5 Velocidade: 1.2363707488472007 km/h
Placa: NHL3875 - Passagens: 2 Velocidade: 0.12673559822747416 km/h
Placa: NHL3828 - Passagens: 2 Velocidade: 0.3459677419354838 km/h
Placa: NHL3816 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NBC3876 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NHL6526 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NUT6876 - Passagens: 1 Velocidade: 0.0 km/h
Placa: AND3878 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NQE3878 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NHL3376 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NQE3876 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NRC0876 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NQC8870 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NRC0878 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NUT3876 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NUN3358 - Passagens: 1 Velocidade: 0.0 km/h
Placa: MRI3878 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NUL3870 - Passagens: 1 Velocidade: 0.0 km/h
Placa: HAT3835 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NEH0876 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NUL3828 - Passagens: 1 Velocidade: 0.0 km/h
Placa: NHL3818 - Passagens: 1 Velocidade: 0.0 km/h
```

RES0241

```
Placa: RES0241 - Passagens: 13 Velocidade: 0.5502128997687733 km/h
Placa: Placa não localizada - Passagens: 31 Velocidade: 1.9785421818302384 km/h
Placa: MAS0241 - Passagens: 2 Velocidade: 0.05081031998262895 km/h
Placa: Placa não reconhecida - Passagens: 1 Velocidade: 0.0 km/h
Placa: RES0246 - Passagens: 1 Velocidade: 0.0 km/h
Placa: RES0264 - Passagens: 1 Velocidade: 0.0 km/h
Placa: WES3274 - Passagens: 1 Velocidade: 0.0 km/h
```

APÊNDICE B – ALGORITMO DE DETECÇÃO E CAPTURA DE FOTO

```
#!/usr/bin/python
import time
import subprocess
import RPi.GPIO as GPIO
import signal
import sys
import picamera;

def signal_handler(signal, frame):
        print('You pressed Ctrl+C!')
        time.sleep(2)
        camera.stop_preview()
        sys.exit(0);

with picamera.PiCamera() as camera:
    camera.drc_strength = 'high'
    #camera.saturation = -100
    camera.sharpness = 100
    camera.resolution =(1024, 768)
    camera.shutter_speed = 600
    camera.iso = 800
    time.sleep(2)
    # Now fix the values
    camera.exposure_mode = 'off'
    g = camera.awb_gains
    camera.awb_mode = 'off'
    camera.awb_gains = g
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(18, GPIO.IN)
    signal.signal(signal.SIGINT, signal_handler)
    print('Sleep de inicio...')
    camera.start_preview()
    time.sleep(2)
    fotoCapturada = False
    i = 0;
    print('Processo pronto!')
```

```
    while True:
        if GPIO.input(18) == 0:
            if fotoCapturada == False:
                fotoCapturada = True
                i = i + 1;
                camera.capture('{0}.jpeg'.format(i), 'jpeg')
                print('Foto capturada!')
            else:
                time.sleep(0.5)
        else:
            if fotoCapturada == True:
                fotoCapturada = False
                print('Camera estabilizada')
```

APÊNDICE C – ARTÍGO CIENTÍFICO

Reconhecimento Automático De Placas Veiculares Utilizando A Tecnologia OCR E A Plataforma Raspberry Pi Aplicada Na Fiscalização Eletrônica De Rodovias

Jackson G. Zanette[1], Sergio Coral[2]

[1]Acadêmico do Curso de Ciência da Computação – Departamento de Ciência da Computação - Universidade do Extremo Sul Catarinense (UNESC)
Caixa Postal 3167 – 88.806-000 – Criciúma – SC – Brasil

[2]Professor do Curso de Ciência da Computação – Departamento de Ciência da Computação - Universidade do Extremo Sul Catarinense (UNESC)
Caixa Postal 3167 – 88.806-000 – Criciúma – SC – Brasil

jacksongz@hotmail.com, sergiocoral@unesc.net

Abstract. *Traffic accidents cause 1.2 millions deaths annually in the world and one of the main causes is the speeding. The traffic violence is a complex problem which have been discussed for decades and is widely studied by the World Health Organization, which defines efficient actions to prevent accidents. One of this actions is the reduction and electronic suverillance of speed. The electronic surveillance model uses widely fixed radars, homever the speed measure occurs only in the point which the radar ins installed. One new model proposed uses radars positioned at known distances connected in a network identifying the vehicles by the plate passing by and measuring the speed by the difference between time and distance of the photograph registers.. To develop a prototype of this model of electronic surveillance, was used a slot cars track with a Rasperry Pi and a camera associated with the libraries OpenCV and Tesseract OCR. Was possible to develop a prototype which made photos and identified the license plates, homever was observed that the processing algorithms are complex and sensible by the change of the enviroment of the image, needing a bigger refinement to apply in the real life.*

Resumo. *Os acidentes de trânsito causam anualmente 1 milhão e 200 mil mortes no mundo e um dos principais motivos é o excesso de velocidade. A violência no trânsito é um problema complexo e vêm sido discutido por décadas e é largamente estudado pela Organização Mundial da Saúde, que define ações eficientes para prevenção de acidentes. Uma destas ações é a redução e fiscalização eletrônica da velocidade. O modelo de fiscalização eletrônica utiliza largamente os radares fixos, porém os mesmos medem apenas a velocidade no ponto instalado. Um novo modelo proposto seria utilizar radares que estivessem posicionados a distâncias conhecidas entre si interligados em uma rede e reconhecessem cada veículo que passassem pelos mesmos pela placa e medir a velocidade dos veículos pela diferença entre tempo e distância dos registros da passagem dos veículos. Para desenvolver um protótipo deste modelo de fiscalização, foi utilizado um circuito fechado de autorama com um conjunto de Raspberry Pi e câmera em associação com as bibliotecas OpenCV e Tesseract OCR. Foi possível desenvolver um protótipo que capturasse fotos e reconhecesse as placas, porém percebeu-se que os algoritmos de*

processamento de imagens são complexos e sensíveis à mudança de ambientação da imagem, necessitando de um maior refinamento para aplicação na vida real.

1. Introdução

Estima-se que 1 milhão e 200 mil pessoas morrem anualmente por causa dos acidentes de trânsito. Os acidentes de trânsito são a principal causa de morte entre pessoas com idade entre 15 e 29 anos (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

Segundo Peden et al (2004, tradução nossa) a velocidade dos veículos está no centro do problema dos acidentes de trânsito, influenciando nos riscos e nas consequências do mesmo. Estima-se que a cada 3 acidentes fatais em países com grande quantidade de veículos, 1 tem como principal motivo o excesso de velocidade.

A gestão da velocidade tem como principal método a fiscalização eletrônica, utilizada de forma fixa ou móvel, sinalizada ou oculta. Um dos principais problemas da fiscalização sinalizada é a redução da velocidade dos condutores trafegam em excesso e avistam a mesma, tornando a acelerar após a fiscalização (WORLD HEALTH ORGANIZATION, 2012, tradução nossa).

Os radares fixos de fiscalização eletrônica possuem uma câmera para registrar o veículo infrator, porém poucos são do tipo Leitor Automático de Placas (LAP). Este tipo de radar permite identificar automaticamente o veículo pela placa através da tecnologia de Reconhecimento Ótico de Caracteres, do inglês Optical Character Recognition (OCR) e abre possibilidades para a fiscalização de mais de uma infração em um único aparelho (MING, 2006). Segundo a ASTC (2011), o custo de implantação e manutenção deste tipo de radar chega a ser quase 6 vezes maior se comparado com o radar sem esta tecnologia.

O OCR permite o reconhecimento de caracteres presente em uma imagem digital adquirida por escâneres e câmeras. Imagens obtidas de câmeras costumam apresentar um desafio para os softwares de OCR, por apresentar distorções e variações de luz (MITHE et al, 2013, tradução nossa).

2. Acidentes de Trânsito

Os acidentes de trânsito são comuns no dia-a-dia, causando mortes e ferimentos de milhares de pessoas. Milhões passam por hospitais anualmente após acidentes graves, sendo que tais acidentes irão mudar a vida de algumas pessoas deixando sequelas. Tais acidentes constituem um problema de saúde pública que afeta principalmente alguns grupos de motoristas. Em países e regiões menos desenvolvidos como África, Ásia, Caribe e América Latina, a maioria das mortes em acidentes de trânsito atingem pedestres, ciclistas, usuários de veículos motorizados de duas rodas e ocupantes de ônibus. Por outro lado, em países desenvolvidos as mortes ocorrem mais frequentemente em usuários de carros. A boa notícia é que este problema pode ser prevenido: Em países desenvolvidos, um conjunto de ações tem trazido redução nos índices de acidentes e na severidade dos mesmos. Estas ações incluem alterações na legislação para controlar o excesso de velocidade, ingestão de álcool, uso obrigatório do cinto de segurança e capacetes no caso de ciclistas e motociclistas, o desenvolvimento de veículos e estradas mais seguros e claro, o uso mais consciente por parte dos motoristas desses veículos e estradas (PEDEN et al 2004, tradução nossa).

A maioria dos acidentes de trânsito é previsível e evitável, pois há evidências de que intervenções para tornar as estradas mais seguras foram implantadas com sucesso em alguns países resultando em uma redução nas mortes no trânsito. Levar tais intervenções para um nível global traduz em um futuro com menos danos e mortes no trânsito (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

Os acidentes de trânsito representam a causa principal de morte entre pessoas de 15 e 29 anos. Estima-se que até 2030 o trânsito será a sétima causa de morte no mundo. Este crescimento estimado é principalmente apontado pela rápida urbanização e pelo aumento da frota de veículos em países em desenvolvimento, dos quais costumam ter problemas de infraestrutura ou necessitam aplicar os investimentos em outros setores. Alguns países desenvolvidos já conseguiram quebrar a relação entre o crescimento da frota e o crescimento dos acidentes tomando ações como tornar as estradas e veículos mais seguros (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

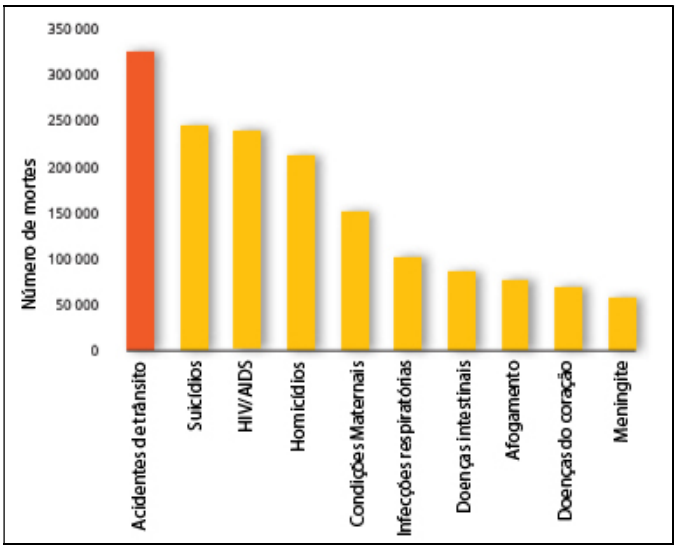


Figura 1 - Causas de morte no ano de 2012

Assim como em outros países, o problema no trânsito no Brasil também envolve diversos setores como segurança, engenharia automotiva e de transportes, educação, legislação, medicina, dentre outras. O setor de saúde é o mais afetado por estes problemas, pois o mesmo será envolvido desde o atendimento aos acidentes até o tratamento dos pacientes, que podem apresentar sequelas reversíveis ou não. O problema dos acidentes de trânsito gera uma sobrecarga no SUS causando altos gastos ao governo. Segundo o próprio SUS, as lesões decorrentes de acidentes de trânsito causam mais gastos que lesões consequentes de outros acidentes, violência e causas naturais somados. A maioria das mortes no trânsito ocorre na população economicamente ativa, o que agrava os efeitos dos acidentes, trazendo problemas à economia e diminuindo a taxa de esperança de vida do país (JORGE, 2013).

A número de mortes causadas por acidentes de trânsito cresceram de cerca de 35 mil em 1996 para cerca de 43 mil em 2011, o que significa um aumento de 22%. A taxa de mortes era de 22,4 por cem mil habitantes, sendo um número virtualmente igual ao número de 1996, que era de 22,5. Durante o período, houve um decréscimo no número entre 1996 e 2000, ano que

apresentou o índice mais baixo do período, 17,1 mortes por 100 mil habitantes. Após o ano de 2000, o índice voltou a crescer até atingir o pico novamente em 2010, 22,5 (JORGE, 2013).

3.1.T.1 – Mortes por acidentes de transporte terrestre – ATT (N e taxa em relação à população), Brasil, 1996 a 2011

Ano	N	Taxa (por cem mil habitantes)
1996	35.281	22,5
1997	35.620	22,3
1998	30.890	19,1
1999	29.569	18,0
2000	28.995	17,1
2001	30.524	17,8
2002	32.753	18,8
2003	33.139	18,8
2004	35.105	19,7
2005	35.994	20,0
2006	36.367	20,0
2007	37.407	20,3
2008	38.273	20,5
2009	37.594	19,9
2010	42.844	22,5
2011	43.256	22,4

Figura 2 - Mortes por acidentes de trânsito no Brasil entre 1996 e 2011

Estas taxas são consideradas bastante elevadas, representando cerca uma vez e meia a taxa encontrada nos Estados Unidos e duas vezes a taxa encontrada no Canadá, fazendo com que o Brasil fique acima da média entre os países das Américas.

O risco de se envolver em um acidente assim como a gravidade do mesmo aumenta conforme a velocidade média do trânsito aumenta. A chance de morte e de lesões sérias aumenta em velocidades mais altas, especialmente para os chamados usuários vulneráveis, que são pedestres, ciclistas e motociclistas (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

A economia de um país em geral e da família das vítimas é fortemente afetada pelos acidentes de trânsito. A população economicamente ativa, com idade jovem, é a mais afetada em países de baixa e média renda. Muitas famílias das quais um indivíduo foi vítima de acidente de trânsito tem sua renda prejudicada, seja por morte de uma pessoa que contribuía com a renda, com gastos em tratamento médico da vítima ou com os gastos vitalícios com a vítima que ficou com sequelas do acidente. Os impactos são estendidos à economia por meio de gastos em sistemas de saúde e sistemas de seguro, assim como o possível decréscimo em uma população economicamente ativa, como já citado. O Instituto de Pesquisa Econômica Aplicada (IPEA) em 2005 apontou que no Brasil a perda no PIB devido aos acidentes de trânsito era de 1.2% (WORLD HEALTH ORGANIZATION, 2015, tradução nossa).

No Brasil, em 2006, o IPEA e a Associação Nacional de Transportes Públicos (ANTP) realizaram um estudo sobre os custos dos acidentes de trânsito. O resultado mostrou que em 2005 o número de acidentes ocorridos em rodovias federais superou os 100 mil e totalizou um custo de 6,5 bilhões de reais. 68% deste custo se referem às pessoas, incluindo fatores como cuidados em saúde e perda de produção, enquanto 31% dos custos foram relacionados aos veículos. Os custos em rodovias estaduais e municipais foram estimados em 14,1 bilhões de

reais e 1,4 bilhão de reais, totalizando a espantosa cifra de 22 bilhões de reais gastos anualmente com acidentes de trânsito (BACCHIERII; BARROSI, 2011).

3. Fiscalização Dos Limites de Velocidade

Segundo Peden et al (2004, tradução nossa), uma fiscalização forte é essencial para melhorar a segurança no trânsito. Os principais meios de fiscalização são feitos por meio de tecnologia de radares com câmeras

Segundo Cannell e Gold (2001), a fiscalização eletrônica reduz o número de acidentes em aproximadamente 30% e o número de mortes em aproximadamente 60%, tanto entre ocupantes de veículos e pedestres na área urbana. Nas rodovias, a instalação de radares fixos em pontos críticos reduz as mortes em aproximadamente 40%.

Um estudo realizado no Canadá mostrou que a fiscalização de trânsito reduz o número de acidentes fatais em países com um número grande de veículos. Por outro lado, uma fiscalização ineficiente pode contribuir para o efeito contrário. Estima-se que se as estratégias de fiscalização fossem aplicadas em toda união europeia, poderia se reduzir pela metade o número de mortes e lesões sérias causadas por acidentes de trânsito (PEDEN et al, 2004, tradução nossa).

Segundo o relatório Global Status Report on Safety feito em 2013, dos 195 países do qual se levantou dados relacionados à acidentes de trânsito, apenas 26 deles classificaram a fiscalização dos limites de velocidade com uma nota acima de 8 em uma escala de 0 a 10. Mesmo em países desenvolvidos que se destacam por aplicarem mais recursos na segurança no trânsito reportaram um índice baixo: Apenas 20% destes classificaram a fiscalização com nota 8 ou mais. Estes dados mostram que a atenção dada à fiscalização está aquém do problema real (WORLD HEALTH ORGANIZATION, 2013, tradução nossa).

Segundo Cannell e Gold (2001), o Brasil é destaque pela violência no trânsito e desrespeito às normas de trânsito e a adoção da fiscalização eletrônica está se mostrando essencial para modificar o comportamento dos motoristas, à fim de reduzir o número e a severidade dos acidentes. Nos últimos anos, a disseminação da fiscalização eletrônica no Brasil tornou-se uma das melhores experiências de implantação da fiscalização eletrônica no mundo.

A fiscalização estacionária de velocidade feita por agentes de trânsito acabou se tornando um método difícil e de baixo custo benefício, o que abriu espaço para o desenvolvimento da fiscalização eletrônica da velocidade, à fim de medir a velocidade de forma automática e confiável. A partir de então, desenvolveu-se os radares (MING, 2006).

A fiscalização eletrônica dos limites de velocidade como os radares com câmeras é utilizada em muitos países e tem eficácia comprovada, pois a mesma produz provas do excesso de velocidade cometido que podem serem utilizados em um possível processo. Este tipo de fiscalização é posicionado em lugares estratégicos onde há constante desobediência dos limites de velocidades estabelecidos pela via e altos riscos de acidentes (PEDEN et al, 2004, tradução nossa).

4. Plataforma Raspberry Pi

O RPI é um computador como qualquer outro, com as principais diferenças de que o mesmo utiliza um processador de arquitetura ARM e não aceita a instalação do Microsoft Windows, porém várias distribuições do Linux podem ser usadas nele. De fácil utilização, poderoso e de

baixo custo, o RPI é um computador ideal para cientistas da computação. O sistema operacional padrão do RPI é uma versão do Linux com uma interface gráfica, porém a maneira mais utilizada de interação com o mesmo é através da interface de linha de comando do inglês command line interface (CLI) (FOUNDATION, 2012, tradução nossa).

Uma estratégia adotada no RPI que diminui o custo e o consumo de energia do mesmo é a adoção da tecnologia sistema-em-um-chip (SoC, do inglês System on a Chip) que fisicamente coloca o processador, a memória e a central de processamento gráfico no mesmo lugar, um sobre o outro, economizando também espaço na placa (NORRIS, 2014, tradução nossa).

Os principais atributos do RPI são o tamanho semelhante à um cartão de crédito e o preço de 35 dólares. O mesmo classifica-se como um SBC (GOLDEN, 2013, tradução nossa).

O RPI 3 Modelo B é a versão mais nova do minicomputador que trouxe alguns avanços. O mesmo recebeu um processador Broadcom BCM2387 Quad-Core de 1.2 gigahertz (GHz) 10 vezes mais rápido que a primeira geração do RPI e 1 GB de memória RAM. Também foram adicionadas interfaces integradas de conectividade Wi-Fi e Bluetooth e mais duas portas USB, além da possibilidade de utilizar o sistema operacional Windows 10 (FOUNDATION, 2016B, tradução nossa).

Um dos módulos mais interessantes do RPI é a placa com câmera projetado para o mesmo. A mesma conecta-se ao RPI pela porta CSI, não ocupando uma porta USB. Este módulo possui várias funções, como fotografias em time-lapse, em que fotos são tiradas à uma frequência menor do que o vídeo que será reproduzido, trazendo a sensação de tempo acelerado. Da mesma forma, é possível capturar vídeos em slow-motion, processo inverso do descrito. (KELLY, 2014, tradução nossa).

O sistema operacional utilizado pelo RPI é o Linux, porém apenas o núcleo (popularmente conhecido como kernel) pode ser considerado como Linux, pois toda a parte de coleção de drivers, serviços e aplicações divergem de acordo com a distribuição do Linux. Algumas das mais comuns são Ubuntu, Debian, Fedora e Arch. Como o RPI é mais parecido com um dispositivo móvel do que com um computador de mesa, os requisitos de software são diferentes principalmente devido ao tipo do processador e às limitações de armazenamento e memória do mesmo (RICHARDSON; WALLACE, 2013).

5. Processamento de Imagens Digitais

O Processamento de Imagens Digitais (PID) é a tecnologia de tratar imagens por meio de algoritmos computacionais e este processo é comumente encontrado em sistemas robóticos ou inteligentes, sistemas médicos, dentre outros. Resulta deste processo uma ou mais imagens ou características da imagem original. As imagens digitais são compostas de pixels, milhares de pontos que combinam as cores vermelho, verde e azul para formar as cores do mundo real. O principal propósito do PID é auxiliar humanos na obtenção de imagens de alta qualidade ou características da imagem original. (ZHOU; WU; ZHANG, 2010, tradução nossa).

A aquisição converte uma imagem em uma representação numérica compreensível para o computador processar. Esta etapa contém dois elementos, onde um é um dispositivo físico sensível a uma faixa de energia no espectro eletromagnético (visível ou invisível aos humanos) que produz um sinal elétrico analógico e um digitalizador que converte este sinal em uma informação binária (MARQUES FILHO; VIEIRA NETO, 1999).

A possibilidade de fazer uma máquina ler textos com a mesma facilidade que o ser humano sempre foi cobiçada, e nos últimos 50 anos o OCR têm sido a mais bem-sucedida tecnologia nas áreas de reconhecimento de padrões e inteligência artificial, apesar de não se comparar ainda com as capacidades humanas de leitura (EIKVIL, 1993, tradução nossa).

Ainda segundo Eikvil (1993, tradução nossa), O OCR pertence ao grupo de técnicas de identificação automáticas, que consistem em técnicas alternativas de inserir dados à um computador. Esta tecnologia é necessária quando a informação deve ser lida tanto por humanos quanto por máquinas. Em comparação às outras técnicas, o OCR é a única em que não é necessário controlar o processo que produz a informação.

O reconhecimento de caracteres pode ser feito tanto quando o caractere é escrito por máquinas quanto escritos à mão, porém a performance e a precisão do reconhecimento serão dependentes da qualidade da fonte do texto (EIKVIL, 1993, tradução nossa).

O reconhecimento de padrões tem um princípio que é ensinar a máquina os padrões. Os padrões no caso do OCR são letras, números e caracteres especiais. O treinamento da máquina ocorre ao inserir os caracteres para que cada um seja classificado e no processo de reconhecimento os caracteres desconhecidos são comparados a aqueles já classificados, procurando pelo melhor correspondente. Um sistema OCR é consistido em algumas etapas que são: A aquisição da imagem, localização e segmentação do texto, pré-processamento, extração de características e reconhecimento (EIKVIL, 1993, tradução nossa).

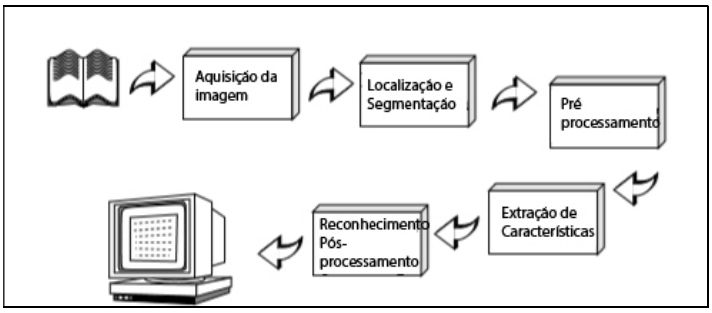


Figura 3 - Etapas de processamento do OCR

A aquisição de uma imagem de um escâner é diferente de uma câmera digital. Enquanto o escâner captura um documento idealmente sem distorções de rotação e luz, uma fotografia digital pode trazer este problema tornando-a mais difícil de ser tratada por uma aplicação OCR (MITHE; INDALKAR; DIVEKAR, 2013, tradução nossa).

6. Reconhecimento Automático De Placas Veiculares Utilizando A Tecnologia OCR E A Plataforma Raspberry Pi Aplicada Na Fiscalização Eletrônica De Rodovias

Com o embasamento adquirido sobre os assuntos levantados, se deu o desenvolvimento de um protótipo capaz de capturar e processar fotos de veículos à fim de reconhecer a placa do mesmo.

O protótipo visa trabalhar de forma integrada com mais de um conjunto de hardware (RPI e câmera) ligados em rede distribuídos em uma rodovia. A placa veicular é única e oferece uma

identidade à cada veículo e isto permitirá que o mesmo veículo seja identificado por mais de um conjunto posicionado em distâncias conhecidas. Comparando-se os registros feitos por cada módulo de um mesmo veículo, é possível verificar o horário em que o registro ocorreu e obter a velocidade média do veículo relacionando a diferença do horário com a distância entre os módulos.

A primeira parte da aplicação desenvolvida foi o software de detecção de movimento e captura da foto. O mesmo utiliza um de obstáculo infravermelho de três pinos.

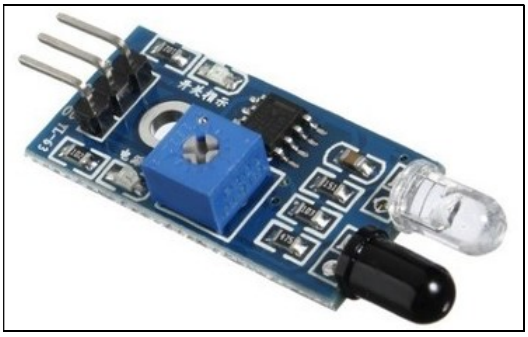


Figura 4 – Sensor de obstáculo infravermelho

O sensor é de simples operação, contendo pinos de alimentação (positivo e negativo) e um pino de sinal que envia o sinal 0 ao detectar um obstáculo. O mesmo foi conectado aos pinos 4 e 14 do GPIO para alimentação e ao pino 12 para sinal, que corresponde ao pino “GPIO18”.

O próximo passo foi desenvolver um algoritmo que fosse capaz de ler os pinos do GPIO à fim de detectar o sinal do sensor. Devido à facilidade e documentação vasta no RPI, a linguagem para desenvolvimento do algoritmo foi Python. O passo seguinte consistia em desenvolver o algoritmo completo que fosse capaz de disparar a câmera assim que o sensor capturasse um obstáculo.

A configuração da câmera mais importante para o protótipo funcionar de maneira satisfatória foi o “*shutter_speed”*. Esta propriedade trata da velocidade de disparo do obturador. Como era necessário capturar uma foto nítida do veículo que passava em alta velocidade, foi necessário alterar esta propriedade até um resultado satisfatório

Figura 5 – Fotos com diferentes velocidades de obturador

Para realizar o teste, foi impresso uma placa veicular em escala e colada na dianteira do carro do autorama. Após posicionar o sensor e a câmera, foi possível capturar fotos satisfatórias em ambientes bem iluminados.

A partir de uma imagem exemplo capturada pelo RPI, começou-se o trabalho de tratamento desta imagem com a biblioteca OpenCV.

Inicialmente, foi montada uma aplicação teste que carregasse a imagem na tela para que fossem aplicados diversos filtros da biblioteca OpenCV à fim de definir a melhor estratégia de localização da placa.

Após a aplicação de sucessivos filtros da biblioteca OpenCV, foi possível localizar a placa com sucesso em imagens com boa iluminação.

Figura 6 – Placa separada e pronta para reconhecimento

7. Resultados obtidos

Após tratamento de alguns pontos no software, foi possível identificar a placa em diferentes ambientes.

Um dos problemas que encontrou-se foi a localização incorreta da placa em algumas imagens devido a presença de outras áreas de alto contraste na imagem.

Outro problema encontrado foi o reconhecimento da placa em diferentes iluminações. Como estabeleceu-se um valor limite para a aplicação do *threshold* em cinquenta e cinco, nem todas as imagens tiveram a separação dos caracteres e do fundo apropriadamente.

Para realizar os testes, foram utilizadas as placas impressas exibidas na figura 7.

Figura 7 – Placas de amostra para os testes

Foram realizadas cerca de 50 passagens para todas as placas em um ambiente externo com boa iluminação. A tabela 1 exibe os resultados para a placa HQW5678.

Tabela 1 - Resultado de 50 passagens com a placa HQW5678

Placa reconhecida	Quantidade de vezes
HOW5678	18
HQW5678	17
HAW5678	3
HEW5678	2
HOW5578	1
HEM5678	1
Falha na localização da placa	7

Foi percebido que o parâmetro de Threshold deve variar conforme o ambiente. O algoritmo adaptativo teve sucesso em alguns casos, porém com o custo de performance visto que o processamento é realizado mais vezes para a mesma imagem.

Em alguns casos também foi possível observar uma identificação errada dos caracteres. Alguns caracteres com semelhança como a letra “O” e o número “0” , a letra “G” e o número “6” e a letra “I” com o número “1” podem causar a identificação errada. Foi desenvolvida uma função para tratar alguns desses caracteres baseado na posição identificada. Caso fossem identificados número nas três primeiras posições, as mesmas eram substituídas por letras correspondentes. Nas quatro últimas posições, caso fosse identificada alguma letra, as mesmas eram substituídas por números. Assim, uma placa identificada incorretamente com “JP6DDDI” era substituída por “JPG0001”.

8. Conclusão

Um dos primeiros pontos observados na implementação do protótipo foi a capacidade da câmera utilizada. Percebeu-se que a RaspiCam não possui uma boa capacidade para capturar fotos de objetos em alta velocidade com nitidez. A mesma possui operação simples e é interessante para vários projetos, porém foi necessário fazê-la atender os requisitos do projeto, que eram um rápido disparo e a captura da imagem sem rastros. Alterando a velocidade de obturador da mesma houve uma grande perda na iluminação da foto.

Observou-se que a localização da placa e o reconhecimento de caracteres depende de vários parâmetros que podem ser influenciados pelo ambiente no qual o protótipo é instalado, principalmente em relação à iluminação e aos objetos ao redor da pista.

A utilização da biblioteca Tesseract OCR na fase de reconhecimento dos caracteres mostrou a capacidade da mesma. Após a localização da placa, é aplicada apenas a binarização na placa resultante antes de fazer o reconhecimento. A biblioteca Tesseract OCR mostrou que é capaz de lidar com caracteres com pequenos ruídos e inclinados, fazendo o reconhecimento da placa como um todo. Esta facilidade traz consigo alguns problemas na precisão do reconhecimento.

Referências

ASTC. EDITAL DE LICITAÇÃO Nº. 064/2011. 2011. Disponível em: <http://www.astc.sc.gov.br/web/arquivos/files/EDITAL 064-2011.pdf>. Acesso em: 12 mar. 2016.

Bacchierii, Giancarlo; Barrosi, Aluísio J D. Acidentes de trânsito no Brasil de 1998 a 2010: muitas mudanças e poucos resultados. Rsp - Revista de Saúde Pública da Usp, São Paulo, v. 5, n. 45, p.949-463, 2011. Disponível em: <http://www.scielo.br/pdf/rsp/v45n5/2981.pdf>. Acesso em: 26 abr. 2016.

Cannell, Alan E. R.; Gold, Philip Anthony. Reduzindo acidentes: o papel da fiscalização de trânsito e do treinamento de motoristas. Washington: Idb Bookstore, 2001. 86 p.

Eikvil, Line. Optical Character Recognition. 1993. Disponível em: <https://www.nr.no/~eikvil/OCR.pdf>. Acesso em: 26 jun. 2016.

Foundation, Raspberry Pi (Org.). The Raspberry Pi: Education Manual. 2012. Disponível em: <http://pi.cs.man.ac.uk/download/Raspberry_Pi_Education_Manual.pdf>. Acesso em: 30 ago. 2015.

Foundation B, Raspberry Pi 3 Model B. Disponível em: <https://www.inet.se/files/pdf/1974044_0.pdf> Acesso em: 26 jun. 2016.

Golden, Rick. Raspberry Pi Networking Cookbook. Mumbai: Packt, 2013. 191 p. Disponível em: <http://daz-pi.com/ebooks/Raspberry Pi Networking Cookbook - Rick Golden.pdf>. Acesso em: 26 jun. 2016.

Jorge, Maria Helena Prado de Mello. Acidentes de trânsito no Brasil: um atlas de sua distribuição. In: Acidentes de trânsito no Brasil: um atlas de sua distribuição. Associação Brasileira de Medicina de Tráfego, 2013. Disponível em:

<http://www.abramet.com.br/files/acidentes_de_transito_atlas_2013.zip>.Acesso em: 06 mar. 2016

Kelly, Hannah (Ed.). Raspberry PI for Beginners. Londres: Imagine Publishing, 2014.

Marques Filho, Ogê; Vieira Neto, Hugo. Processamento Digital de Imagens. Rio de Janeiro: Brasport, 1999. 331 p.

Ming, Sun Hsien. Fiscalização eletrônica de trânsito. 2006. Disponível em: <http://www.sinaldetransito.com.br/artigos/fiscalizacao-eletronica-do-transito.pdf>. Acesso em: 15 maio 2016.

Mithe, Ravina; Indalkar, Supriya; Divekar, Nilam. Optical Character Recognition. International Journal Of Recent Technology And Engineering. [S.I.], p. 72-75. mar. 2013.

Norris, Donald. Raspberry Pi Projects for the Evil Genius. Nova York: Codemantra, 2014. 288 p

Peden, Margie et al. World report on road traffic prevention. Genova: World Healt Organization, 2004. 244 p. Disponível em: <http://whqlibdoc.who.int/publications/2004/9241562609.pdf?ua=1>. Acesso em: 7 mar. 2016.

Richardson, Mat T; Wallace, Shawn. Primeiros Passos com o Raspberry Pi. São Paulo: Novatec, 2013.

World Health Organization. Global status report on road safety. 2015. Disponível em: <http://www.who.int/violence_injury_prevention/road_safety_status/2015/GSRRS2015_Summary_EN_final2.pdf?ua=1>. Acesso em: 06 mar. 2016.

World Health Organization. Global status report on road safety. 2013. Disponível em: <http://www.who.int/violence_injury_prevention/road_safety_status/2013/en/ >. Acesso em: 06 mar. 2016.

Zhou, Huiyu; WU, Jiahua; Zhang, Jianguo. Digital Image Processing. [S.I]: Book Boon, 2010. 71 p.