



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA - PATOS DE MINAS
ENGENHARIA DE ELETRÔNICA E DE TELECOMUNICAÇÕES

MARCONI AMORIM MARTINS

**IDENTIFICAÇÃO DE PLACAS DE TRÂNSITO ATRAVÉS DA CLASSIFICAÇÃO
DE IMAGENS USANDO REDES NEURAIS ARTIFICIAIS**

PATOS DE MINAS
2017

MARCONI AMORIM MARTINS

**IDENTIFICAÇÃO DE PLACAS DE TRÂNSITO ATRAVÉS DA CLASSIFICAÇÃO
DE IMAGENS USANDO REDES NEURAIS ARTIFICIAIS**

Trabalho de conclusão de curso de graduação
apresentado à Faculdade De Engenharia Elétrica da
Universidade Federal de Uberlândia, como requisito
parcial para a obtenção do título de bacharel em
Engenharia Eletrônica e Telecomunicações.

Orientador: Prof.^a Dr.^a Eliana Pantaleão

Co-orientador: Prof. Ms. Júlio Cezar Coelho

PATOS DE MINAS
2017

MARCONI AMORIM MARTINS

**IDENTIFICAÇÃO DE PLACAS DE TRÂNSITO ATRAVÉS DA CLASSIFICAÇÃO
DE IMAGENS USANDO REDES NEURAIS ARTIFICIAIS**

Trabalho de conclusão de curso de graduação apresentado à Faculdade De Engenharia Elétrica da Universidade Federal de Uberlândia como requisito parcial para a obtenção do título de bacharel em Engenharia Eletrônica e Telecomunicações.

Aprovado em: ____ de _____ de ____.

BANCA EXAMINADORA:

Prof. Dr. André Luiz Aguiar da Costa (Membro 1) – FEELT/UFU

Prof.^a Dr.^a Eliana Pantaleão (Orientadora) – FACOM/UFU

Prof. Ms. Júlio Cezar Coelho (Co-Orientador) – FEELT/UFU

Prof. Dr. Laurence Rodrigues do Amaral (Membro 2) – FACOM/UFU

PATOS DE MINAS
2017

Dedico este trabalho em especial a minha mãe, Maria José de Amorim Martins, meu pai, Vicente Abner Martins, minha namorada e a toda minha família e amigos.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me dar inteligência e capacidade em desenvolver esse trabalho.

Aos meus pais, Vicente Abner Martins e Maria José de Amorim Martins, por me apoiarem e me ajudarem durante toda a minha vida.

A minha namorada, por me ajudar a superar os problemas e me mostrar que sou capaz de coisas grandiosas.

A minha avó materna, Maria de Lourdes Amorim, minha tia materna Márcia dos Reis Amorim, meu tio materno Celi Gaspar de Amorim e sua esposa Letícia Nascimento Amorim, minha afilhada Fernanda Amorim Ribeiro, meu tio materno José Gaspar de Amorim e sua esposa Divina Amorim, meu tio paterno José Donizete Martins e sua esposa Adélia Alves Martins, meu tio paterno Jorge Martins e sua esposa Lesina Ferreira Martins, minha tia Maria José Martins Caixeta, minha madrinha Marieta de Fátima Martins Costa, meu padrinho Francisco das Chagas Costa, todos meus primos e tios não citados, meu amigo e vizinho Valterson Ferreira e sua esposa Rosilene Dorneles Ferreira, a me ajudarem a apoiarem sempre que eu preciso.

A Prof.^a Dr.^a Eliana Pantaleão e Prof. Me. Júlio Cezar Coelho por me auxiliarem durante o desenvolvimento desse trabalho.

A todos os professores, técnicos e funcionários da Universidade Federal de Uberlândia – Campus Patos de Minas, que me ajudaram a chegar até aqui.

Ao Centro Universitário de Patos de Minas – UNIPAM, por ceder o espaço para a Universidade Federal de Uberlândia ministrar suas aulas.

A Pró-Reitoria de Assistência Estudantil – PROAE, por intermédio da Divisão de Assistência e Orientação Social – DIASE/DIRES, que me ajudam por meio da bolsa permanência desde o primeiro período.

A todos meus amigos (as) e colegas que me ajudaram a superar fases ruins e a comemorar fases boas durante todo o curso.

A todos, meus sinceros agradecimentos,

Marconi.

— *"Se não tivermos a chave, não poderemos abrir aquilo que não temos com que abrir, então do que adiantaria encontrar aquilo que precisa ser aberto, e que não temos, sem primeiro encontrar a chave que o abra?"*

Jack Sparrow

RESUMO

Recentemente, veículos automotores têm incluído sistemas de auxílio ao motorista, que analisam o ambiente através da aquisição e processamento digital de imagens, incluindo módulos para identificação e reconhecimento de placas de trânsito. Para este processo, o passo inicial é identificar, na imagem, as placas de trânsito presentes. Este trabalho mostra os passos desenvolvidos, como a conversão para escala de cinza, limiarização e segmentação da imagem, para isolar placas de trânsito em imagens adquiridas por uma câmera acoplada a um Raspberry Pi B+. Logo após, foi realizada a aprendizagem da rede neural artificial, e esta mostrar resultados promissores com o reconhecimento de uma determinada placa. Assim, o Raspberry Pi B+ envia comandos para um Arduino Uno, para que este faça o controle de um carro autônomo seguidor de linha, executando ações nos motores para realizar manobras de acordo com a placa reconhecida.

Palavras-chaves: Redes Neurais Artificiais, Aprendizagem, Raspberry Pi B+, Arduino Uno, Processamento Digital de Imagens, OpenCV, Carro Seguidor de Linha, Carro Autônomo.

ABSTRACT

Recently, automotive vehicles have included driver's aid systems, which analyze the environment through the acquisition and digital processing of images, including modules for identification and recognition of transit boards. For this process, the initial step is to identify the transit boards present in the image. This work shows the steps developed, such as converting to grayscale, thresholding and segmentation of the image, to isolate transit boards in images acquired by a camera coupled to a Raspberry Pi B +. Shortly after, it was accomplished the artificial neural network learning, and this showed promising results with the recognition of a particular plaque. Thus, the Raspberry Pi B + sends commands to an Arduino Uno, which controls a self-follower line, performing actions on the engines to perform maneuvers according to the recognized plaque.

Keywords: Artificial Neural Networks, Learning, Raspberry Pi B +, Arduino Uno, Digital Image Processing, OpenCV, Line Follower, Autonomous Car.

LISTA DE FIGURAS E QUADROS

Figura 1 – Raspberry Pi 2 B+.....	4
Figura 2 – Arduino Uno	6
Figura 3 – Exemplo de Carro Seguidor de Linha e Arduino Uno.....	7
Figura 4 – RNA com Três Camadas	8
Figura 5 – Exemplo da Primeira Camada da RNA	9
Figura 6 – Visão Esquemática de uma Câmera CCD.	11
Figura 7 – Matriz de uma Imagem em Nível de Cinza ou com um Canal (R, G ou B).....	11
Figura 8 – Matriz de uma Imagem RGB.....	12
Figura 9 – Exemplo Placa de 60 km/h	13
Figura 10 – Esquema de Ligação dos Componentes	16
Figura 11 – Fluxograma de Etapas do Desenvolvimento do Trabalho	16
Figura 12 – Placas Escolhidas	17
Figura 13 – Fluxograma da Implementação do Trabalho.....	18
Figura 14 – Exemplo de Organização das Imagens	20
Figura 15 – Exemplo de Conversão de Matriz em Vetor.....	20
Figura 16 – Exemplo de Conversão de uma Matriz de Imagens em uma Matriz de Vetor.....	21
Figura 17 – Desenho Gráfico da Pista	29
Quadro 1 – Mapa dos pinos do Raspberry Pi 2 B+	5
Quadro 2 – Mapa dos pinos do Arduino Uno	6
Quadro 3 – Teste 1: Aprendizagem com 100 imagens, 100x50 neurônios.....	23
Quadro 4 – Teste 2: Aprendizagem com 100 imagens, 150x100x50 neurônios.....	24
Quadro 5 – Teste 3: Aprendizagem com 100 imagens, 350x250x150 neurônios.....	24
Quadro 6 – Teste 4: Aprendizagem com 200 imagens, 100x50 neurônios.....	25
Quadro 7 – Teste 5: Aprendizagem com 200 imagens, 150x100x50 neurônios.....	25
Quadro 8 – Teste 6: Aprendizagem com 200 imagens, 350x250x150 neurônios.....	26
Quadro 9 – Teste 7: Aprendizagem com 250 imagens, 100x50 neurônios.....	26
Quadro 10 – Teste 8: Aprendizagem com 250 imagens, 150x100x50 neurônios.....	27
Quadro 11 – Teste 9: Aprendizagem com 250 imagens, 350x250x150 neurônios.....	27
Quadro 12 – Resumo dos Nove Testes Realizados.....	28

LISTA DE ABREVIATURAS E SIGLAS

2-D	2 Dimensões
3-D	3 Dimensões
APIs	Application Programming Interface
BACKPROP	Back-Propagation
EEPROM	Electrically-Erasable Programmable Read-Only Memory
GPIO	General Purpose Input/Output
HDMI	High-Definition Multimedia Interface
HSB	Hue, Saturation and Brightness (Matiz, Saturação e Brilho)
HSV	Hue, Saturation and Value (Matiz, Saturação e Valor)
I2C	Circuito Inter-Integrado
JAI	Java Advanced Imaging
JMF	Java Media Framework
k-NN	k-Nearest Neighbors
km/h	quilômetro por hora
LDR	Light Dependent Resistor
LED	Light Emitting Diode
MiB	MebiByte
Min	Minuto
MLP	Multi-Layer Perceptrons
OpenCV	Open Source Computer Vision Library
PIP	Python Package Index
PNG	Portable Network Graphics
PWM	Pulse Width Modulation
RAM	Random Access Memory
RGB	Red, Green and Blue (Vermelho, Verde e Azul)
RNA	Rede Neural Artificial
RPROP	Resilient-Backpropagation
RS-232	Recommended Standard - 232
s	Segundo
SD	Secure Digital
SIGMOID_SYM	Sigmoid Symmetric (Sigmóide simétrica)
TSR	Traffic Sign Recognition
USB	Universal Serial Bus

LISTA DE SÍMBOLOS

b	Número de bits
$\{g(\cdot)\}$	Função de ativação de um neurônio
M	Linhas
N	Colunas
$\{\Theta\}$	Limiar
$\{u\}$	Axônio de um neurônio
$\{W_1, W_2, \dots, W_n\}$	Pesos
$\{X_1, X_2, \dots, X_n\}$	Entrada
$\{y\}$	Saída

SUMÁRIO

1. INTRODUÇÃO	1
1.1. Introdução Teórica	1
1.2. Estado da Arte	1
1.3. Resumo.....	3
2. REVISÃO DE LITERATURA	4
2.1. Raspberry Pi 2 B+.....	4
2.2. Arduino UNO	5
2.3. Carro Seguidor de Linha	7
2.4. Redes Neurais Artificiais (RNA)	7
2.5. Técnicas de Processamento Digital de Imagens	10
2.5.1. Aquisição	10
2.5.2. Digitalização	11
2.5.3. Conversão Para Escala de Cinza	12
2.5.4. Limiarização.....	12
2.5.5. Segmentação.....	13
2.6. OpenCV.....	14
2.7. Resumo.....	15
3. METODOLOGIA.....	16
4. RESULTADOS E DISCUSSÕES.....	23
5. CONCLUSÃO	30
REFERÊNCIAS.....	31

1. INTRODUÇÃO

1.1. INTRODUÇÃO TEÓRICA

Nos últimos anos, sistemas de assistência à navegação automobilística têm aderido tecnologias cada vez mais sofisticadas, e têm recebido atenção de empresas especializadas. Com base nas informações do Centro de Informações Galileo (2005, p. 2), é esperado que, até 2020, metade dos veículos que estarão nas ruas devem ter um sistema avançado de assistência ao motorista.

Alguns desses sistemas analisam o ambiente em que haverá a locomoção do veículo através da aquisição e processamento digital de imagens, com o intuito de auxiliar ou automatizar a navegação deste veículo. Este ambiente, por onde o veículo se locomoverá, apresenta sinalização vertical em forma de placas, que podem ser de placas indicativas de velocidade máxima, sinalização de trânsito e legibilidade das placas de trânsito (CUNHA, JACQUES e CYBIS, 2006, P. 12). O reconhecimento dessas placas irá ajudar o motorista, pois ele poderá ser avisado sobre situações perigosas, e também sobre condições sobre características, como por exemplo uma curva, um cruzamento e outros.

A área que aborda esse tipo de reconhecimento foi denominada de TSR (*Traffic Sign Recognition*) e pode possuir várias dificuldades como variações de luminosidade e ambiente, desgaste da pintura da placa e captura de imagens em movimento (GREENHALGH e MIRMEHDI, 2012).

1.2. ESTADO DA ARTE

Diversos pesquisadores brasileiros vêm tentando desenvolver um veículo autônomo, como o CARINA (Carro Robótico Inteligente para Navegação Autônoma), projeto da USP São Carlos (LRM, 2015), o Carro Autônomo da UFES (LCAD, 2015) e o CADU (Carro Autônomo Desenvolvido na UFMG) (COROS, 2015), (ARRUDA, 2012), (FREITAS et al., 2009). Embora com diferentes abordagens para o problema de controle e navegação dos veículos, é comum a todos os projetos o uso de visão computacional para identificação de objetos nas vias urbanas (LIMA e PEREIRA, 2010), (FACCHINETTI, 2010). Outro ponto em comum é o uso de versões em miniatura ou simuladores para o desenvolvimento e teste dos algoritmos.

Embora possuam inúmeras possíveis aplicações, os sistemas TSR desenvolvidos ainda não apresentam desempenho totalmente satisfatório, exigindo que mais pesquisas sejam realizadas nessa área. Uma das principais dificuldades é o estado de conservação das placas nas vias urbanas brasileiras. Por esse motivo, é comum o uso de algoritmos de aprendizagem de máquina para realizar a tarefa.

Rodrigues (2008) propõe um modelo para detecção de placas baseado na percepção humana, usando informações de cor e forma para a classificação. Após utilizar um filtro de suavização, a segmentação foi realizada com base nas cores vermelha e amarela, utilizando os canais de matiz e saturação do modelo de cores HSB (*Hue, Saturation and Brightness*). A partir das imagens limiarizadas e segmentadas, foram extraídos os atributos de forma: área, perímetro, ângulo, fator de forma, circularidade, compacticidade, proporção (*aspect ratio*), maior e menor eixo. O software para realizar estas etapas foi desenvolvido na linguagem Java, usando o pacote ImageJ. A seguir, o software WEKA (HALL et al., 2009) foi utilizado para a etapa de classificação, e foram testados os métodos k-NN (*k-Nearest Neighbors*) e Redes Bayesianas. Os acertos obtidos foram acima de 80%, quando o sistema foi testado em imagens previamente adquiridas e selecionadas. Não foram feitos testes em tempo real de aquisição.

Poffo (2010) usa a quantização de cores, o filtro de Sobel e a transformada de Hough para a extração de características e uma RNA (Rede Neural Artificial) MLP (*multi-layer perceptron*) treinada com o método *backpropagation* para a classificação. A aplicação permite capturar uma imagem contendo um cenário da via à frente do veículo, detectar a região da imagem onde cada placa de trânsito aparece, reconhecer a placa de trânsito de cada região detectada, informar o usuário caso não seja possível reconhecer a placa e informar o usuário sobre cada placa de trânsito reconhecida.

Para a implementação da ferramenta foi utilizada a linguagem de programação Java, as APIs (*Application Programming Interface*): JMF (*Java Media Framework*) e JAI (*Java Advanced Imaging*), e o *framework* de RNA para Java *Neuroph*. Por usar o modelo de cor RGB (*Red, Green and Blue*), a solução sofre grande influência de variações de luminosidade nas cenas, sendo os melhores resultados obtidos nas imagens mais escuras. O autor aponta como desvantagem da implementação o alto consumo de memória da ferramenta usada na aprendizagem. Além disso, cabe ressaltar a grande necessidade de interação do usuário com a ferramenta durante sua utilização.

Ribeiro e Salomão (2014) incluem imagens de vídeo em sua análise, além das imagens fotográficas. As imagens capturadas foram segmentadas com base no modelo de cores HSV

(*Hue, Saturation and Value*) e um detector de circunferência implementado através de uma matriz de acumuladores, cujo valor máximo corresponde ao centro da circunferência. A ferramenta SIFT (*Scale Invariant Feature Transform*), que extrai características invariantes à escala, rotação e iluminação, foi utilizada para gerar o conjunto de dados para o classificador. O método de aprendizagem escolhido foi o k-NN, com uma métrica de distância baseada no produto escalar entre os vetores de características. De modo geral, o trabalho alcançou acerto acima de 85% no reconhecimento, porém apresentou problemas nas placas que regulamentam a velocidade máxima dos veículos. Este caso foi tratado separadamente, com o uso de redes neurais artificiais.

O próximo tópico irá abordar os fundamentos de cada componente e sistema que foi usado para que o trabalho fosse implementado.

1.3. RESUMO

Neste tópico, foi feita uma introdução a carros autônomos, e também foi descrito simplificadaamente alguns trabalhos publicados relacionados ao tema deste trabalho.

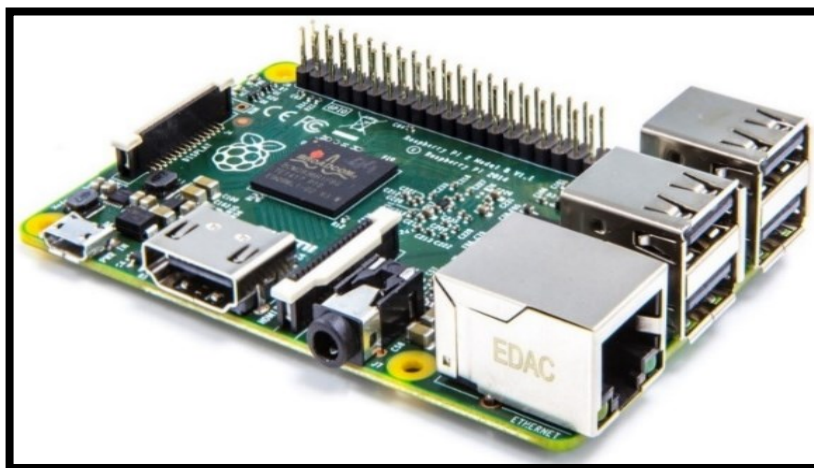
2. REVISÃO DE LITERATURA

Neste tópico, será feita uma introdução com alguns detalhes de cada componente e sistema necessário para o desenvolvimento do carro autônomo capaz de reconhecer uma placa, e realizar uma manobra consequente a essa placa. Será necessário o Raspberry Pi 2 B+, Arduino Uno, carro seguidor de linha, Redes Neurais Artificiais, Processamento Digital de Imagens e OpenCV (*Open Source Computer Vision Library*).

2.1. RASPBERRY PI 2 B+

Raspberry Pi 2 B+ (Figura 1) é um microcomputador pouco maior que um cartão de crédito, com baixo custo para adquiri-lo, sendo criado pela Fundação Raspberry Pi no Reino Unido. Contém processador, processador gráfico, *slot* para micro cartão SD (*Secure Digital*), quatro entradas para dispositivos USB (*Universal Serial Bus*), saída de vídeo e áudio via HDMI (*High-Definition Multimedia Interface*), memória RAM (*Random Access Memory*), entrada para a Câmera Pi, e barramentos de expansão (CANALTCH, 2014).

Figura 1 – Raspberry Pi 2 B+.



Fonte: https://www.raspberrypi.org/wp-content/uploads/2015/01/Pi2ModB1GB_-comp.jpeg

Esses barramentos são chamados de GPIO (*General Purpose Input/Output*), e são pinos que têm como função realizar a comunicação de entrada e saída de sinais digitais. No total, são 40 pinos. Com esse barramento, é possível fazer leitura de sensores, acionar motores, e enviar sinais para outro controlador. No Quadro 1, encontra-se o mapa dos pinos (FAZEDORES, 2014).

Quadro 1 – Mapa dos pinos do Raspberry Pi 2 B+

NOME	PINO	PINO	NOME
3.3V	01	02	5V
GPIO 2	03	04	5V
GPIO 3	05	06	GND
GPIO 4	07	08	GPIO 14
GND	09	10	GPIO 15
GPIO 17	11	12	GPIO 18
GPIO 27	13	14	GND
GPIO 22	15	16	GPIO 23
3.3V	17	18	GPIO 24
GPIO 10	19	20	GND

NOME	PINO	PINO	NOME
GPIO 09	21	22	GPIO 25
GPIO 11	23	24	GPIO 8
GND	25	26	GPIO 7
ID_SD	27	28	ID_SC
GPIO 5	29	30	GND
GPIO 6	31	32	GPIO 12
GPIO 13	33	34	GND
GPIO 19	35	36	GPIO 16
GPIO 26	37	38	GPIO 20
GND	39	40	GPIO 21

Fonte: https://www.raspberrypi.org/wp-content/uploads/2015/01/Pi2ModB1GB_-comp.jpeg

Os pinos 02 e 04 são saídas de alimentação com 5V. O pino 01 é uma saída para alimentação com 3.3V. Os pinos 06, 09, 14, 20, 25, 30, 34 e 39 são os “terra” (*GROUND*). Os pinos 02 e 03 são portas programáveis para a interface I2C (Circuito Inter-Integrado), para uma comunicação entre periféricos com baixa velocidade. Os pinos 14 e 15 são portas seriais usando o protocolo RS-232 (*Recommended Standard - 232*), com uso para envio e recebimento de sinais. Os pinos 07, 11, 12, 13, 15, 16, 18, 22, 29, 31, 32, 33, 35, 36, 37, 38 e 40 são os GPIO e têm como função enviar ou receber sinais analógicos e digitais. Os pinos 19, 21, 23, 24 e 26 são GPIO avançados, contendo a possibilidade de realizar no mesmo pino a comunicação serial *Full Duplex* síncrono, ou seja, de forma bidirecional. Os pinos 27 e 28 são portas de EEPROM (*Electrically-Erasable Programmable Read-Only Memory*), que é um tipo de memória que possui características de ser programadas e apagadas múltiplas vezes (FAZEDORES, 2014).

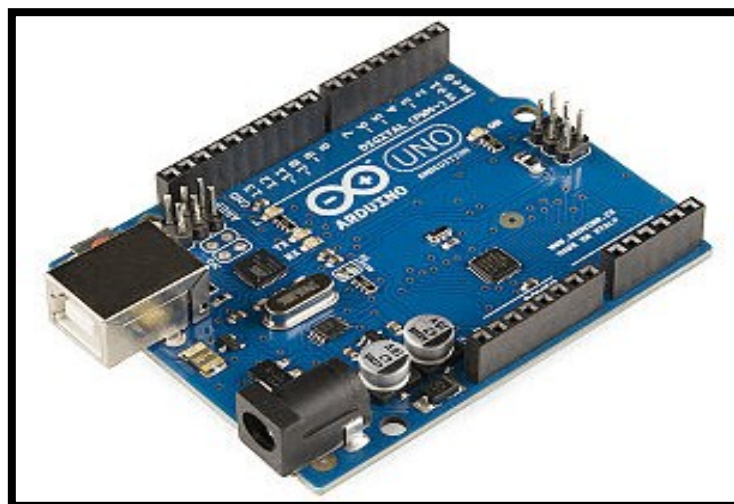
2.2. ARDUINO UNO

O Arduino é uma plataforma de prototipagem eletrônica com código aberto (*open-source*), que tem fundamentos em hardware e software flexíveis e fáceis de usar. Através dos conectores de saídas e entradas digitais e analógicas existentes no hardware, o Arduino pode perceber o ambiente em que está por meio de recepção de sinais de sensores, e pode comunicar

com seus arredores podendo acionar motores, atuadores, luzes e outros. A programação via software é implementada usando linguagem C (ARDUINO, 2016).

O Arduino Uno (Figura 2) é um hardware com microcontrolador produzido pela Atmel Corporation. Possui 13 pinos digitais que podem ser usados como entrada ou saída de sinais, e 6 destes podem ser utilizados como saídas PWM (*Pulse Width Modulation*) com função de controlar motores de corrente contínua; possui 5 pinos analógicos, um cristal oscilador de 16MHz e outros conectores para alimentação e comunicação com um computador (ARDUINO, 2016).

Figura 2 – Arduino Uno



Fonte: <http://static.arduino.org/media/k2/galleries/90/A000066-Arduino-Uno-TH-2tri.jpg>

No Quadro 2, encontra-se o mapa dos pinos do Arduino Uno (ARDUINO, 2016).

Quadro 2 – Mapa dos pinos do Arduino Uno

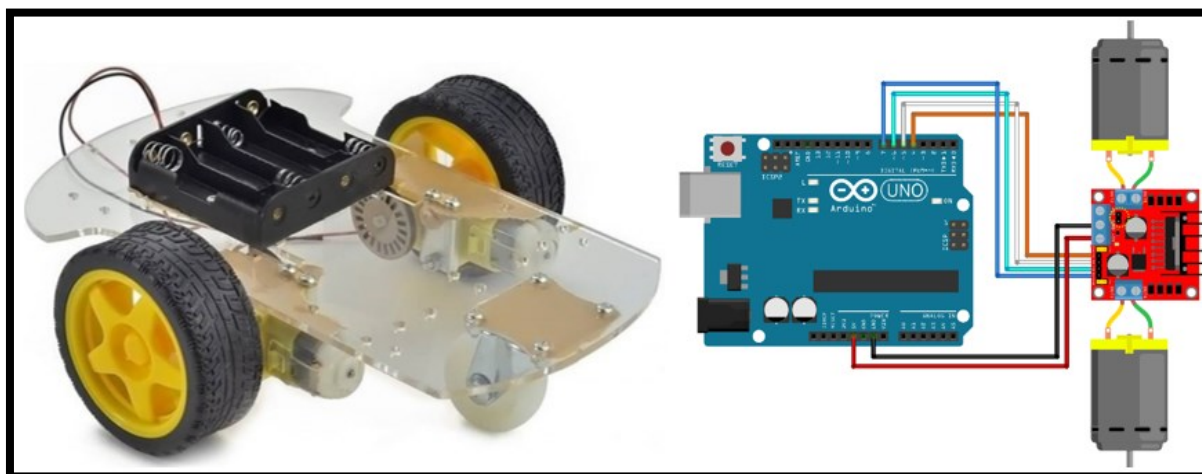
Pino	Função
0	Comunicação Digital TX0
1	Comunicação Digital RX0
1 a 13	Digital I/O
3,5,6,9,10 e 11	Digital PWM
A0 a A5	Analógicas I/O
3.3V, 3.5V, GND, Vin e VREF	Power

Fonte: <https://www.embarcados.com.br/arduino-uno/>

2.3. CARRO SEGUIDOR DE LINHA

Um carro seguidor de linha (ou faixa) é um veículo autônomo que não necessita de um condutor humano, e que consegue se locomover através de uma linha colocado no chão. Por meio de sensores ópticos, ele consegue identificar uma faixa de cor diferente em relação às outras cores do chão. Normalmente, coloca-se uma faixa preta em cima do chão de cores mais claras. Este sistema de sensoriamento pode ser realizado por sensores fotoelétricos utilizando LEDs (*Light Emitting Diode*) ou LDRs (*Light Dependent Resistor*). Cada sensor é constituído por um LED de baixo brilho, que emite um feixe de luz sobre a superfície, e um LDR para captação da energia refletida (ADEL e KENNET, 2007). Um exemplo de carro seguidor de linha está mostrado na Figura 3, o qual possui duas rodas presas a motores de corrente contínua, e uma roda “guia” na frente. Os motores são alimentados por uma bateria ou conjunto de pilhas em série, ligada a uma Ponte H (NEWTON, 2015) controlada pelo Arduino Uno ou outro microcontrolador.

Figura 3 – Exemplo de Carro Seguidor de Linha e Arduino Uno



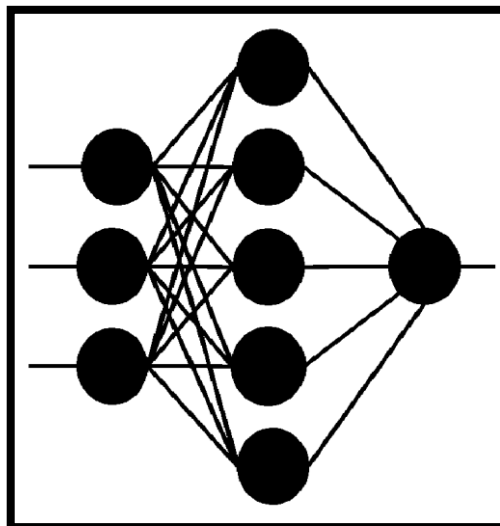
Fonte: <http://blog.filipeflop.com/wp-content/uploads/2014/02/>

2.4. REDES NEURAIS ARTIFICIAIS (RNA)

Problemas complexos relacionados a computação e tecnologia são solucionados muitas das vezes usando os conceitos das redes neurais artificiais (Figura 4). De modo análogo ao cérebro humano, as redes neurais artificiais trabalham com processamentos complexos ao mesmo tempo graças à divisão do processamento para os neurônios. Assim, a rede neural

artificial deve ser capaz de aprender e qualificar decisões a partir da aprendizagem (INFOWESTER, 2016).

Figura 4 – RNA com Três Camadas



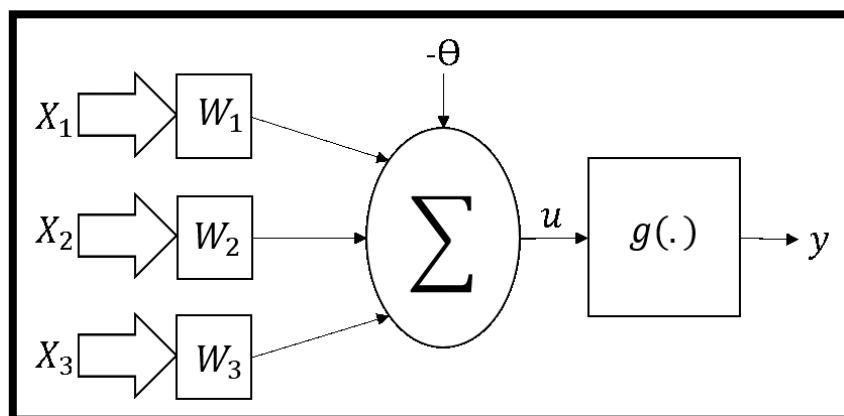
Fonte:Adptada de (PETCIVIL, 2013)

Na Figura 4, é mostrado um exemplo de RNA do tipo MLP com três camadas, onde cada círculo preenchido de preto simboliza um neurônio artificial. Cada linha é chamada de sinapse, representando a ligação entre dois neurônios, e são atribuídas grandezas chamadas pesos sinápticos. Esses pesos são empregados para armazenar o conhecimento. De acordo com o problema a ser resolvido, deve ser determinado o número de camadas, a quantidade de neurônio em cada camada, os pesos de cada sinapse e a saída desejada. A saída de cada neurônio é chamada de axônio e as entradas de cada neurônio são chamadas de dendritos. Assim, as pontas de cada sinapse são formadas por axônio e dendritos (INFOWESTER, 2016).

O número de nós na primeira camada é determinado pelo tamanho do vetor de atributos que será utilizado na aplicação e o número de neurônios na camada de saída, pela quantidade de classes existentes no problema. Já a quantidade de nós nas camadas intermediárias, bem como a quantidade dessas camadas, deve ser determinada de forma empírica com base na complexidade do problema de classificação (THEODORIDIS E KOUTROUMBAS, 2006).

Na primeira camada da rede (Figura 5), os neurônios estão submetidos a receber entradas $\{X_1, X_2, \dots, X_n\}$ e multiplicar pelo pesos $\{W_1, W_2, \dots, W_n\}$ das sinapses. Se a soma ponderada dos valores multiplicados ultrapassar o limiar definido $\{-\Theta\}$, um sinal é enviado para o axônio deste neurônio $\{u\}$. Este sinal passa por uma função de ativação $\{g(\cdot)\}$, chegando até os dendritos de cada neurônio da próxima camada $\{y\}$. Essa etapa é repetida para todos os neurônios contidos na RNA (PETCIVIL, 2013).

Figura 5 – Exemplo da Primeira Camada da RNA



Fonte:Adptada de (PETCIVIL, 2013)

Quando ocorrem mudanças nas sinapses, significa que está ocorrendo o processo de aprendizagem, e está relacionada com a ativação dos neurônios. Se uma sinapse está sendo utilizada, esta será fortificada, enquanto sinapses que têm pouco uso serão debilitadas. Assim, para a rede ser implantada, necessita-se de um tempo para ser treinada/aprendida. Existem três tipos de aprendizagem: supervisionado, não-supervisionada e híbrido (INFOWESTER, 2013).

Na supervisionada, as entradas e saída desejadas são padronizadas, ocorrendo o ajuste nos pesos até que exista um erro mínimo entre os padrões de saída e os desejados. Na não-supervisionada, a rede busca encontrar e determina algumas características contidas na entrada, e assim, os pesos são definidos e ocorre o aprendizado. Na híbrida, ocorre a mistura entre supervisionada e não-supervisionada, e cada camada pode usar um método de aprendizagem diferente (INFOWESTER, 2013). Neste trabalho, o método de aprendizagem é o supervisionado.

Existem várias aplicações de uma RNA com finalidade de resolver problemas. Atualmente, é empregada para reconhecimento de voz humana, robôs com inteligência artificial, scanners com tradutor de texto, mercado financeiro, reconhecimento de objetos em imagens digitais, entre outros. Uma rede neural artificial implementada não significa que terá 100% de acertos em relação com a saída desejada, pois nenhuma máquina ainda criada é tão poderosa como o cérebro humano (INFOWESTER, 2013).

Todo condutor precisa ao máximo estar atento às informações contidas na pista durante o trajeto, e assim, o processamento visual é rápido e intenso. Este condutor, possuindo em seu veículo um sistema automático, que identificasse e reconhecesse a sinalização contida no trajeto, poderia ajudar na navegação, deixando o motorista concentrado apenas na direção do veículo, podendo diminuir o número acidentes por falta de informação (KUS ET AL., 2008).

2.5. TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS

Para que o carro autônomo consiga reconhecer uma placa, devem ser aplicadas técnicas de processamento digital de imagens, como: aquisição, digitalização, conversão dos pixels de uma imagem em escala de cinza, limiarização (binarização) da imagem em escala cinza, e segmentação da imagem limiarizada. Esses processamentos devem ser feitos para que o reduza o número de informações a ser inserida na RNA, afim de que possa ter melhores e mais eficientes resultados (GREENHALGH e MIRMEHDI, 2012). Os códigos serão implementados usando as linguagens de programação C++, usando a biblioteca do OpenCV instalada no Raspberry Pi 2 B+.

Nos próximos subitens, será feita uma introdução à aquisição, digitalização, conversão em escala de cinza, limiarização e segmentação.

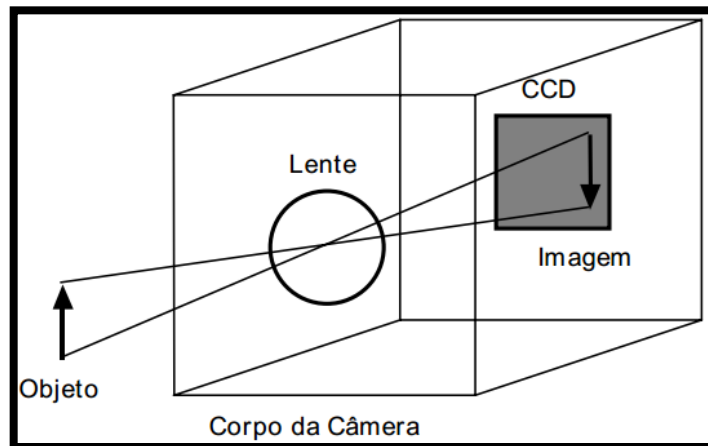
2.5.1. AQUISIÇÃO

A aquisição de uma imagem é o processo de conversão de uma cena real tridimensional em uma imagem analógica, ou seja, o processo de transdução optoeletrônica. Para converter uma cena real tridimensional em uma imagem analógica é preciso uma redução de dimensionalidade. Uma câmera fotográfica, *WebCam* ou outro dispositivo converterá a cena 3-D (3 Dimensões) em uma representação 2-D (2 Dimensões) adequada.

O dispositivo de aquisição de imagens mais usado atualmente é a câmera CCD (*Charge Coupled Device*), que consiste de uma matriz de células semicondutoras fotossensíveis, que operam como capacitores, armazenando carga elétrica proporcional à energia luminosa incidente na cena. O sinal elétrico produzido é conservado por circuitos eletrônicos especializados, produzindo à saída um SCV (Sinal Composto de Vídeo) analógico e monocromático.

Para a aquisição de imagens coloridas usando CCDs é obrigatório um conjunto de prismas e filtros de cor encarregados de decompor a imagem colorida em suas componentes de cores RGB, cada uma capturada por um CCD independente. Os sinais elétricos correspondentes a cada componente RGB são combinados subsequentemente de acordo com o padrão de cor utilizado, por exemplo NTSC (*National Television Standards Committee*) ou PAL (*Phase Alternating Line*). A Figura 6 mostra uma visão simplificada da aquisição de imagens com câmera CCD.

Figura 6 – Visão Esquemática de uma Câmera CCD.



Fonte: Adaptada de (MARQUES FILHO e VIEIRA NETO, 1999).

2.5.2. DIGITALIZAÇÃO

O sinal analógico, obtido na saída do dispositivo de aquisição, deve ser submetido a uma discretização espacial e em amplitude para se tornar o formato pretendido ao processamento computacional. Intitula-se amostragem como o processo de discretização espacial e o nome de quantização ao processo de discretização em amplitude. Simplesmente, a amostragem converte a imagem analógica em uma matriz de M (linhas) por N (colunas) pontos em nível de cinza ou em um único canal (R, G ou B), estes também denominados pixels, como visto na Figura 7.

Depois da quantização, cada um destes pixels assume um valor inteiro, no intervalo de 0 a $(2^b - 1)$. Quanto maior o valor de b , maior o número de níveis de cinza presentes na imagem.

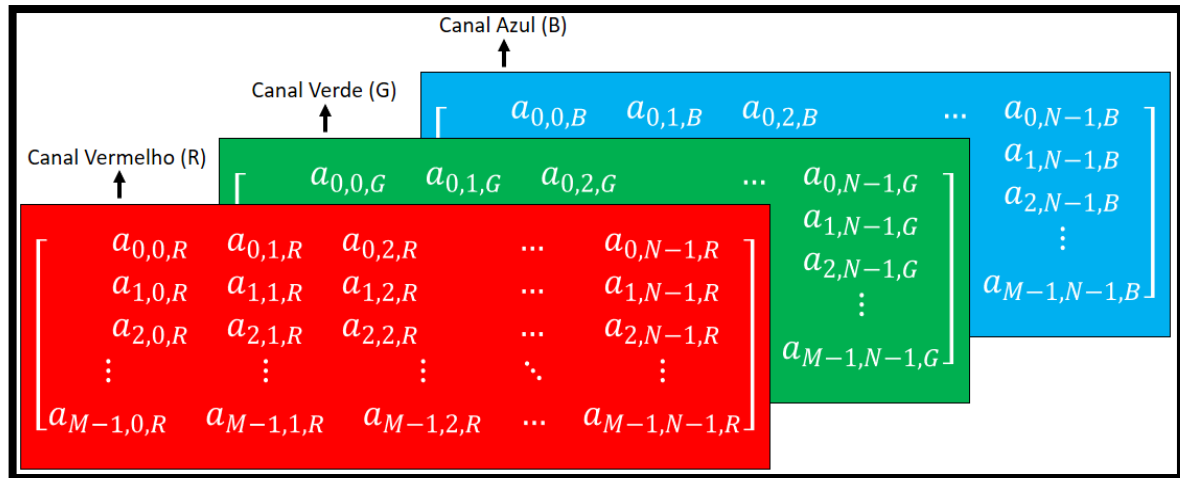
Figura 7 – Matriz de uma Imagem em Nível de Cinza ou com um Canal (R, G ou B)

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & \dots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & a_{1,2} & \dots & a_{1,N-1} \\ a_{2,0} & a_{2,1} & a_{2,2} & \dots & a_{2,N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{M-1,0} & a_{M-1,1} & a_{M-1,2} & \dots & a_{M-1,N-1} \end{bmatrix}$$

Fonte: Adaptada de (MARQUES FILHO e VIEIRA NETO, 1999).

Já para a imagem RGB, a amostragem converte a imagem analógica em três matrizes de M por N pontos, cada uma com um dos canais (R, G ou B), como visto na Figura 8:

Figura 8 – Matriz de uma Imagem RGB



Fonte: Adaptada de (MARQUES FILHO e VIEIRA NETO, 1999).

Portando, a imagem digital é uma imagem analógica discretizada tanto espacialmente quanto em amplitude. Ela pode ser vista como uma matriz, onde as linhas e colunas identificam um ponto na imagem (pixel), cujo valor corresponde ao nível de cinza ou de cores RGB da imagem naquele ponto.

2.5.3. CONVERSÃO PARA ESCALA DE CINZA

Imagens convertidas em tons de cinza têm características de cores variando do preto como a menor intensidade luminosa, até o branco com maior intensidade luminosa. Assim, partes de uma imagem que contem figuras escuras são convertidas em tons com intensidade com valor próximo à intensidade luminosa do preto, e figuras claras são convertidas em tons com intensidade com valor próximo à intensidade luminosa do branco (MARQUES FILHO e VIEIRA NETO, 1999). A Figura 9.A mostra um exemplo de placa de trânsito de 60 km/h e a Figura 9.B, sua conversão para tons de cinza.

2.5.4. LIMIAZIZAÇÃO

A limiarização de uma imagem baseia-se na diferença dos níveis de cinza que compõem diferentes objetos de uma imagem. A partir de um limiar estabelecido de acordo com as características dos objetos que se quer isolar, a imagem pode ser segmentada em dois grupos: o grupo de pixels com níveis de cinza abaixo do limiar e o grupo de pixels com níveis de cinza

acima do limiar. Logo, uma imagem limiarizada será binária, ou seja, terá somente dois valores para cada pixel, sendo estes o branco ou o preto (MARQUES FILHO e VIEIRA NETO, 1999). A Figura 9.C mostra um exemplo de limiarização da imagem convertida em escala de cinza.

2.5.5. SEGMENTAÇÃO

Segmentação de imagens é o processo de particionar uma imagem, ou seja, dividi-la em regiões de modo que a interseção entre duas regiões diferentes seja vazia e a união de todas as regiões resulte na imagem original (HARALICK e SHAPIRO, 1992). O processo de segmentação pode ser aplicado com vários objetivos, sendo comum o propósito de separar objetos em uma imagem de seu fundo (*background*) (GONZALEZ, R.; WOODS, 2000). A Figura 9.D mostra um exemplo de segmentação da imagem original, retirando somente a região em interesse: a placa.

Figura 9 – Exemplo Placa de 60 km/h

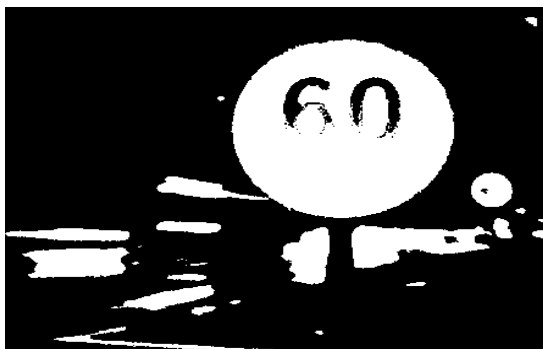
A) Exemplo Imagem Placa Original



B) Imagem em Escala de Cinza



C) Exemplo Imagem Placa Limiarizada



D) Exemplo Placa Segmentada



Fonte Figura 9.A): <http://www.cirielli.com.br/figuras/inicio-novo.jpg>

Fonte Figura 9.B), C) e D): Modificação da Figura 9.A) em 15/11/2016 pelo Autor.

2.6. OPENCV

OpenCV (*Open Source Computer Vision Library*) é uma biblioteca que funciona em várias plataformas e equipamentos diferentes, e tem como finalidade ajudar no desenvolvimento de softwares ou aplicativos. Foi criada pela Intel em meados do ano 2000 usando os códigos da linguagem de programação C/C++, e também transferida para códigos em *Java*, *Python* e *Visual Basic*. A versão mais atual é a OpenCV 3.2 lançada em dezembro de 2016. Essa biblioteca pode ser instalada e importada para o Raspberry Pi 2 B+, podendo realizar processamento de filtros em imagens, detecção de objetos, aplicação na aprendizagem de uma RNA, reconhecimento facial, entre outros (CONHECIMENTOGERAL, 2016).

A seguir, será mostrado informações das funções da biblioteca OpenCV 3.2 utilizadas na implementação do trabalho (INTEL, 2016):

- *fswebcam* – Captura uma imagem pela *WebCam* e salva em um arquivo.
- *imread* – Carrega uma imagem de um arquivo.
- *split* – Divide a imagem multicanal em vários *arrays* de canal único.
- *cvtColor* – converte uma imagem de um espaço de cor para outro. Usado para transformar de RGB para HSB, HSV, escala de cinza, entre outros.
- *threshold* – Limiariza a imagem em escala de cinza conforme os valores dos limiares utilizados. Se o valor de pixel for maior do que um valor de limiar, ele será atribuído o valor 1 (branco), senão ele será atribuído o valor 0 (preto).
- *findContours* – encontra contornos em uma imagem limiarizada.
- *approxPolyDP* – aproxima uma(s) curva(s) poligonal com a precisão especificada, ou seja criando caixas nos contornos encontrados.
- *boundingRect* – procura informações das caixas dos contornos e cria um retângulo quando encontrar um.
- *minEnclosingCircle* – procura informações das caixas dos contornos e cria um círculo quando encontrar um.
- *drawContours* – desenha contornos encontrados nas funções *boundingRect* e/ou *minEnclosingCircle*.
- *rectangle/circle* – salva informações dos contornos desenhados, como coordenadas das arestas de um retângulo ou do centro e raio de um círculo.
- *resize* – redimensiona uma imagem.
- *imshow* – mostra a imagem numa janela.

- *InRange* – Verifica se os elementos de matriz estão entre os elementos de dois outros *arrays*.
- *erode* – Corrói uma imagem usando um elemento estruturador específico.
- *dilate* – Dilata uma imagem usando um elemento de estruturação específica.
- *create* – cria uma RNA especificando as entradas, saídas desejadas.
- *setLayerSizes* – configura o número de neurônios em cada camada, incluindo o número de camadas intermediárias.
- *setActivationFunction* – configura o parâmetro especificando a função de ativação da RNA. Pode ser *IDENTITY*, *SIGMOID_SYM* e *GAUSSIAN*. Na literatura pesquisada, a função mais usada é *SIGMOID_SYM*.
- *setTrainMethod* – configura o parâmetro especificando o método de treinamento da RNA. Pode ser o algoritmo *BACKPROP* (*back-propagation*) e *RPROP* (*resilient-backpropagation*). Na literatura pesquisada, o método mais usado é o *BACKPROP*.
- *setTermCriteria* – configura os critérios de término do algoritmo de treinamento. Pode configurar o máximo de iterações, e/ou o quanto o erro pode alterar entre as iterações para fazer o algoritmo continuar.
- *train* – realiza o treinamento/aprendizagem da RNA.
- *save* – salva o arquivo de treinamento/aprendizagem da RNA.
- *predict* – realiza a predição de um dado com o arquivo de treinamento/aprendizagem da RNA.

No próximo tópico, será abordada a explicação da metodologia e os materiais utilizados para a implementação do trabalho.

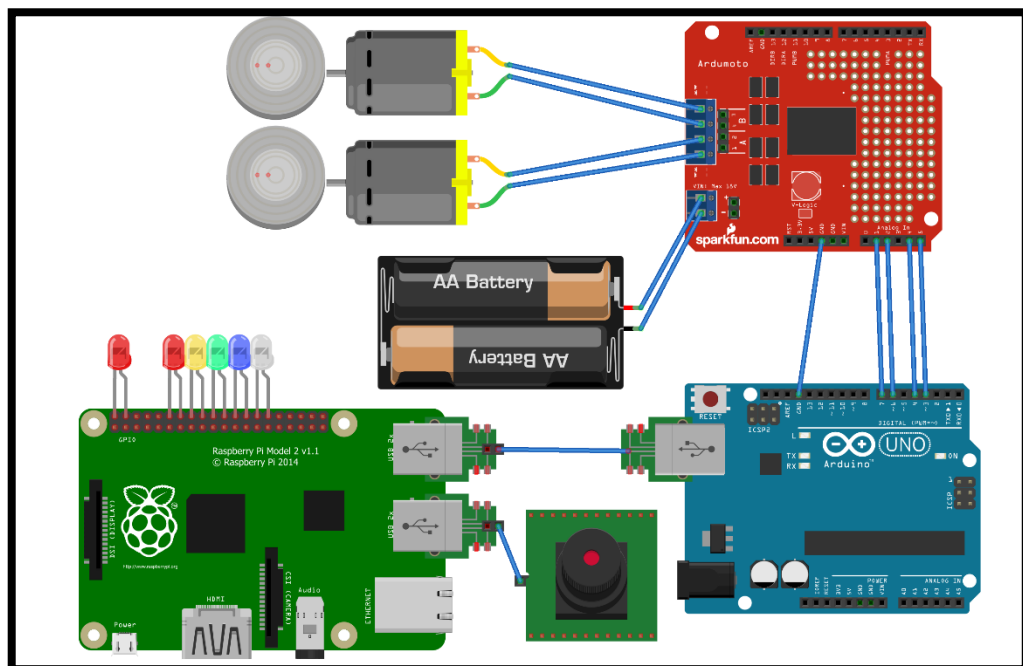
2.7. RESUMO

Nesse tópico foi abordado um resumo dos componentes e sistemas a ser utilizado no desenvolvimento do trabalho, como: Raspberry Pi 2 B+, Arduino Uno, carro seguidor de linha, Redes Neurais Artificiais, Processamento Digital de Imagens e OpenCV; exibindo suas funcionalidades e métodos de operação.

3. METODOLOGIA

O objetivo desse trabalho de conclusão de curso é criar um sistema através do Raspberry Pi 2 B+ para identificar e reconhecer placas de sinalização vertical de trânsito durante o percurso feito por um veículo, este podendo fazer manobras conforme a função de uma placa reconhecida. Será utilizado um Raspberry Pi 2 B+, Arduino Uno, Ponte H, Motores, Pilhas, *WebCam*, LEDs, chassi de um carrinho seguidor de linha, como visto na Figura 10.

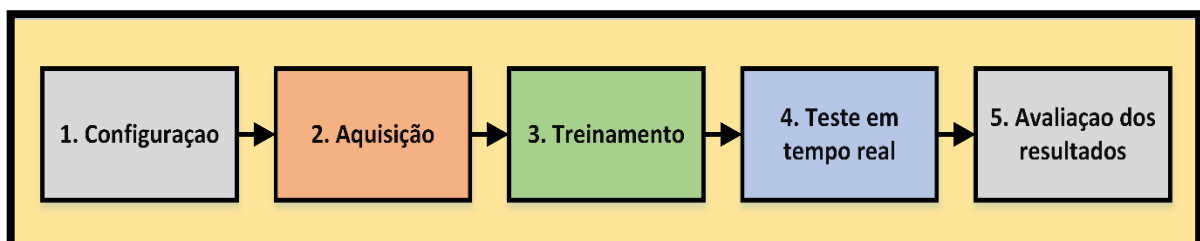
Figura 10 – Esquema de Ligação dos Componentes



Fonte: O Autor (2017) usando o software livre Fritzing .

Neste tópico, será visto uma breve explicação da metodologia e materiais a serem utilizados para a implementação do trabalho. A Figura 11 mostra as etapas do desenvolvimento do trabalho.

Figura 11 – Fluxograma de Etapas do Desenvolvimento do Trabalho



Fonte: O Autor (2017).

As imagens de placas escolhidas para o processo de aprendizagem da rede neural artificial e para teste em tempo real, foram: 40 km/h, 80 km/h, PARE, Vire à Direita e Vire à Esquerda, respectivamente, como visto na Figura 12, e que foram impressas e colocadas a distâncias e ângulos distintos para o processo de captura.

Para começar todo o processo, primeiramente configurou o Raspberry Pi 2 B+ usando o sistema operacional *Raspbian Jessie*, que é derivado do *Debian GNU/Linux*. Este foi escolhido após pesquisas de um sistema operacional para o Raspberry Pi 2 B+ que proporcionaria uma maior satisfação para a produção dos resultados.

Figura 12 – Placas Escolhidas



Fonte: Editada de http://www.detran.se.gov.br/educ_sinal.asp#ui-tabs-1 em 18/06/2017 pelo Autor.

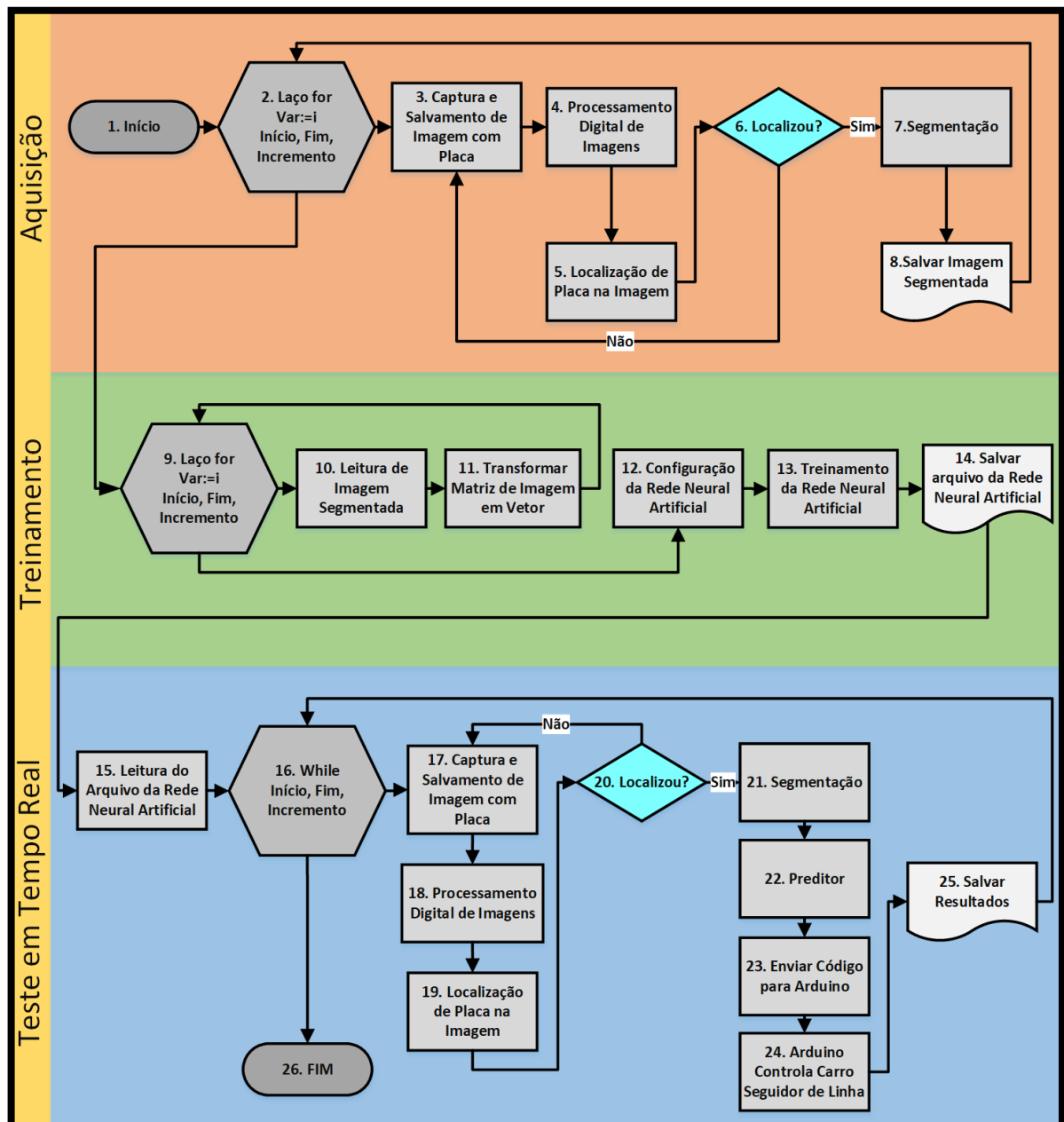
Logo após a instalação do sistema operacional, instalou todas as dependências e atualizações do Linux, com uso do Terminal. O Raspberry Pi 2 B+ estava conectado à internet para todas as instalações. Também foi instalado os pacotes de recursos para imagens e vídeo, *drivers* para acesso da *WebCam* conectada ao Raspberry Pi 2 B+, o software para compilar os códigos criados: *CodeBlocks*, juntamente com os pacotes de compilação do OpenCV e o software para compilar os códigos do Arduino Uno: *IDE Arduino*.

Na aquisição das imagens, foi utilizada uma *WebCam* tradicional conectada no Raspberry Pi 2 B+, e através dessa imagem adquirida, criou-se algoritmos para a extração de características contidas na imagem e segmentação (MARQUES FILHO e VIEIRA NETO, 1999). Para o reconhecimento da placa, foi utilizado redes neurais artificiais (THEODORIDIS E KOUTROUMBAS, 2006), através de um algoritmo de aprendizado.

Após o reconhecimento, o Raspberry Pi 2 B+ enviou comandos para o Arduino Uno, que realizou os movimentos no carrinho, como virar para a esquerda caso a placa seja de obrigatoriedade para virar à esquerda. Este carrinho andou por uma trilha, e ao longo dessa trilha tinham placas de sinalização vertical de trânsito. Os algoritmos para a realização de todos os processos, foram elaborados em linguagem C++ com ajuda de funções do OpenCV.

Este resumo, pode ser visto na Figura 13, onde os blocos que estão sobre a cor laranja, representam o processo de aquisição das imagens para aprendizagem, bem como o processamento digital de imagens e segmentação. Os blocos sobre a cor verde representam o processo de aprendizagem da rede neural artificial, onde as imagens de placas foram colocadas em forma de vetor, criando uma matriz de vetores de imagens para a entrada da rede. Já os blocos sobre a cor azul, representam o teste em tempo real usando a função preditora do OpenCV.

Figura 13 – Fluxograma da Implementação do Trabalho



Fonte: O Autor (2017).

A organização nessas três camadas se deve ao fato de que quando for realizar o teste em tempo real, por exemplo, não ter que capturar e nem realizar a aprendizagem da rede neural artificial, uma vez que este demora um tempo muito grande para ser executado. Cada bloco será explicado detalhadamente a seguir.

No bloco 2, o programa entrou em um laço *for*, onde a *WebCam*, no bloco 3, capturou 250 imagens com placas, ou seja, 50 imagens de cada placa da Figura 12, e as salvou no arquivo de imagem com formato PNG (*Portable Network Graphics*). Já, no bloco 4, cada placa sofreu um processamento digital de imagens, sendo esses a separação de canais de cores RGB, e usando somente o canal R (Vermelho) da imagem para os processos posteriores; a transformação em escala de cinza e limiarização.

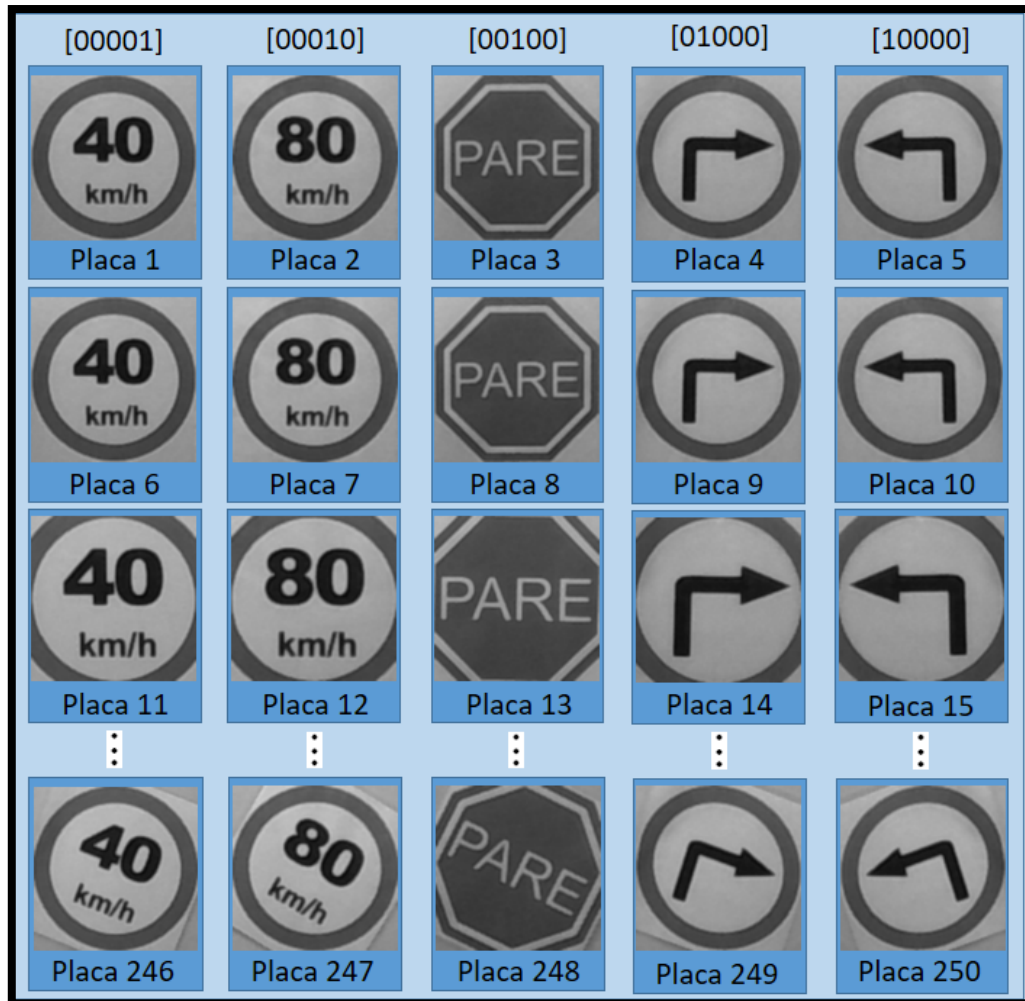
Caso a imagem não apresentasse nenhum polígono com placa, o bloco 6, que é o código de decisão, voltava para o bloco 3 para capturar uma nova imagem. Já se houve polígono com placas, o bloco 6 passou para o bloco 7, para segmentar a imagem, e assim recortou a imagem nas coordenadas que haviam o polígono com a placa e salvou em um arquivo de imagem com formato PNG no bloco 8.

Após a segmentação de 250 imagens contendo placas dos 5 tipos escolhidos, estas imagens foram renomeadas e organizadas na seguinte ordem: Placa 1 (40 km/h), Placa 2 (80 km/h), Placa 3 (Pare), Placa 4 (Vire à Direita) e Placa 5 (Vire à Esquerda), Placa 6 (40 km/h), Placa 7 (80 km/h), Placa 8 (Pare), Placa 9 (Vire à Direita) e Placa 10 (Vire à Esquerda), e assim até completar as 250 imagens de placas, conforme visto na Figura 14.

No bloco 9, o laço *for* realizou o processo do bloco 10, onde este leu todas as 250 imagens segmentadas salvas nos arquivos PNG e as colocou cada uma em uma posição de uma matriz, para que o bloco 11 fizesse a conversão de cada imagem em um vetor. Assim formando-se uma matriz de vetores contendo as 250 imagens, uma em cada linha da matriz, ou seja, cada linha dessa nova matriz é referente a uma das 250 imagens.

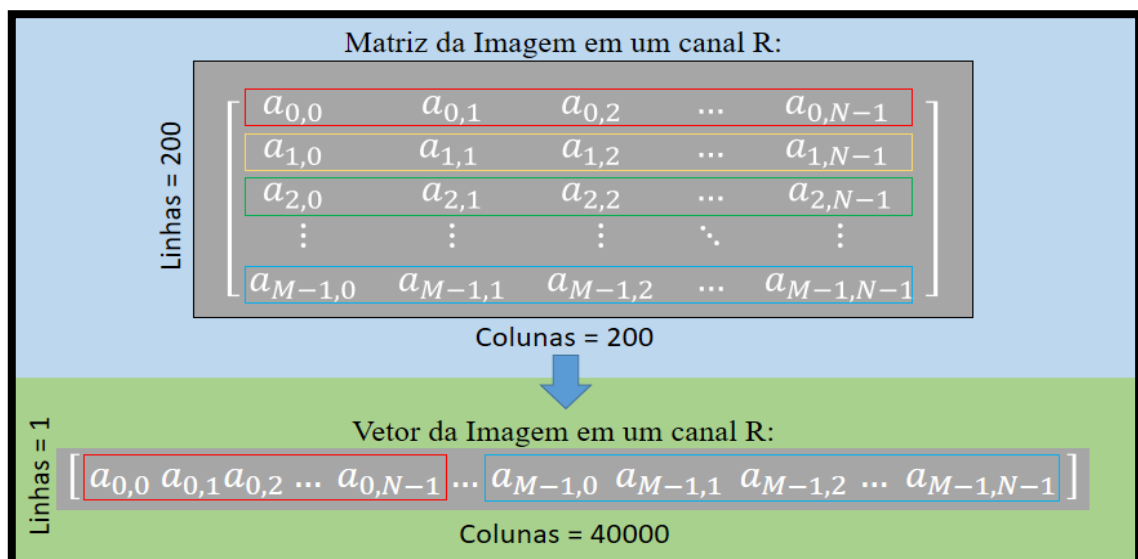
A Figura 15 mostra um exemplo de uma matriz de imagem convertida em um vetor, onde cada linha é inserida depois da última coluna da linha anterior. A matriz de imagem tem dimensões de 200 linhas por 200 colunas e um único canal R (Vermelho), resultando em um vetor de dimensão de 1 linha por 40000 colunas. Já a Figura 16 mostra um exemplo de uma matriz, contendo em cada posição uma imagem, sendo convertidas em uma matriz de vetores. Essa matriz será uma das partes da entrada da função de criação da rede neural artificial no processo de configuração no bloco 12.

Figura 14 – Exemplo de Organização das Imagens



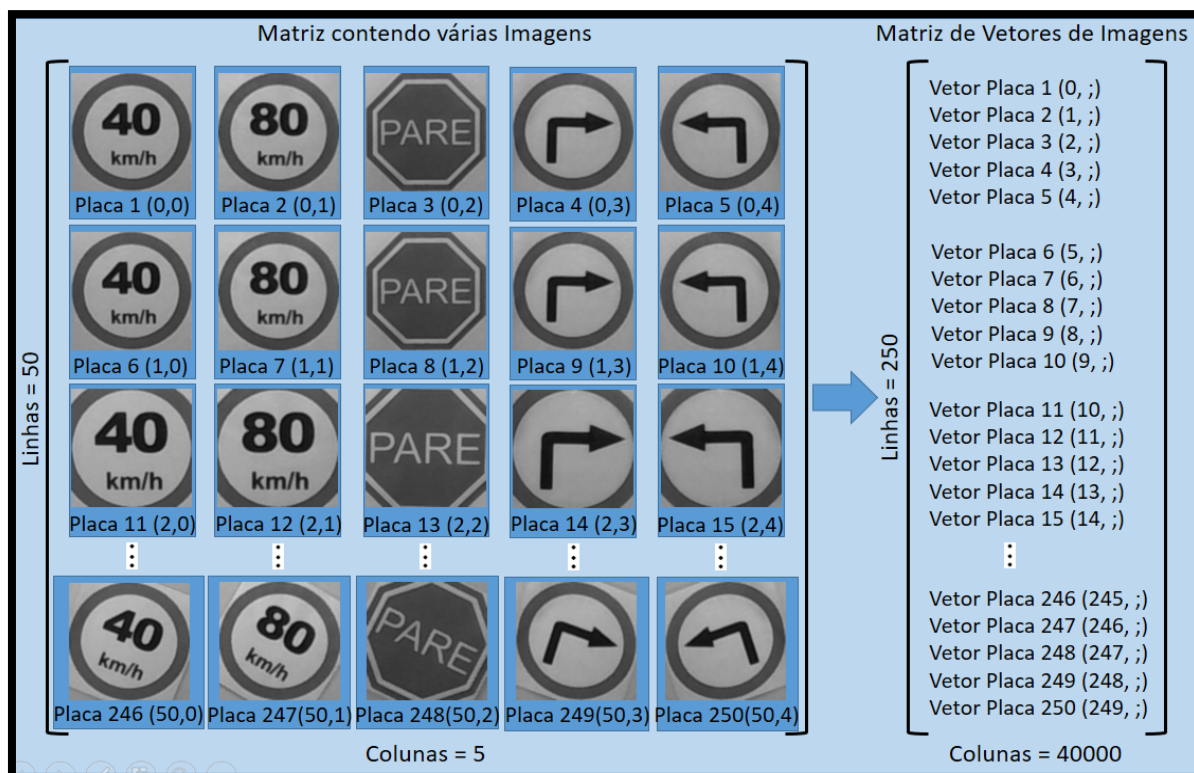
Fonte: O Autor (2017).

Figura 15 – Exemplo de Conversão de Matriz em Vetor



Fonte: Adaptada de (MARQUES FILHO e VIEIRA NETO, 1999).

Figura 16 – Exemplo de Conversão de uma Matriz de Imagens em uma Matriz de Vetor



Fonte: O Autor (2017).

Outra parte da entrada da função de criação da rede neural artificial no processo de configuração são as saídas esperadas de cada placa, ou seja, o vetor correspondente de cada uma das 250 imagens. Cada um dos 5 tipos escolhidos de placas terá um vetor código, que são: as placas de 40 km/h = [00001], 80 km/h = [00010], PARE = [00100], Vire à Direita = [01000] e Vire à Esquerda [10000], como visto na Figura 13.

Também foi definido na configuração da rede neural artificial, a quantidade de camadas intermediárias que iria conter, bem como a quantidade de neurônios artificiais contidos em cada uma dessas camadas, e a função de ativação de cada neurônio. Outra configuração importante realizada foi a definição do método de aprendizagem. Foi escolhido o método de aprendizagem *BACKPROP*, e a função de ativação pelo algoritmo da sigmóide simétrica (*SIGMOID_SYM*). Após realizar a configuração da rede neural artificial, o bloco 12 fez a aprendizagem desta rede neural artificial, e o bloco 13, o armazenamento em um arquivo.

Após a aprendizagem, realizou-se o teste em tempo real. Do bloco 16 ao 21, acontecem os mesmos processos já descritos anteriormente nos blocos 2 ao 7. No bloco 22 ocorreu a

predição da placa capturada desconhecida, onde a rede neural artificial retornou informações de qual placa foi identificada.

No bloco 23, o Raspberry Pi 2 B+ enviou códigos, via comunicação USB serial, para o Arduino Uno referente à placa reconhecida. Assim, no bloco 24, o Arduino controlou o carro seguidor de linha e realizou os movimentos na pista referentes à placa. No bloco 25, o Raspberry Pi 2 B+ salvou informações da placa localizada em um arquivo de texto, para futuras avaliações.

4. RESULTADOS E DISCUSSÕES

Neste tópico, serão vistos os testes realizados para a aprendizagem da RNA e os resultados em tempo real, bem como o teste realizado pelo carrinho seguidor de linha.

Toda a implementação e testes foram realizados no Raspberry Pi 2 B+. O primeiro passo após a configuração foi realizado capturando diferentes quantidades de imagens, contendo cada um dos tipos de placa escolhido, e assim realizando o processamento digital de imagens em cada imagem capturada, separando o canal R (Vermelho), transformando em escala de cinza e a limiarização, e então realizando a localização e o recorte da placa na imagem.

Por fim foi realizado a aprendizagem da RNA com essas imagens segmentadas de placa. Diferentes quantidades de camadas intermediárias e número de neurônios em cada uma dessas, foram escolhidos empiricamente para a aprendizagem.

O teste em tempo real foi realizado sempre utilizando 30 novas imagens de cada placa. Cada uma dessas imagens foi passada pelo mesmo processamento digital de imagens, e quando foi encontrada uma placa, foi efetuada a segmentação e logo após a predição, salvando num arquivo contendo a matriz de confusão.

Os quadros a seguir mostram a porcentagem de acerto e a porcentagem de erro de cada tipo de placa localizada, para cada um dos testes realizados, sendo com 100, 150, 200 e 250 imagens segmentadas de placas para a aprendizagem com diferentes quantidades de camadas intermediárias e número de neurônios.

O Quadro 3 mostra o Teste 1 utilizando o arquivo de aprendizagem da RNA com 100 imagens segmentadas de placas e duas camadas intermediárias, com 100 neurônios na primeira camada e 50 neurônios na segunda. O tempo de aprendizagem no Teste 1 foi de aproximadamente 53.3 minutos. O arquivo da RNA salvo teve o tamanho de 134.1 MiB. Cada imagem em tempo real foi predita em 0.31 segundos.

Quadro 3 – Teste 1: Aprendizagem com 100 imagens, 100x50 neurônios

Placa	Acertos	% de Acertos	Erros	% de Erros
40 km/h	14	46.66	16	53.34
80 km/h	18	60.00	12	40.00
Pare	30	100.00	0	0.00
Vire à Direita	16	53.34	14	46.66
Vire à Esquerda	18	60.00	12	40.00
				Acurácia: 64%

Fonte: O Autor (2017).

O Quadro 4 mostra o Teste 2 utilizando o arquivo de aprendizagem da RNA com 100 imagens segmentadas de placas e três camadas intermediárias, com 150 neurônios na primeira camada, 100 na segunda e 50 na terceira. O tempo de aprendizagem no Teste 2 foi de aproximadamente 138.3 minutos. O arquivo da RNA salvo teve o tamanho de 200.5 MiB. Cada imagem em tempo real foi predita em 0.42 segundos.

O Quadro 5 mostra o Teste 3 utilizando o arquivo de aprendizagem da RNA com 100 imagens segmentadas de placas e três camadas intermediárias, com 350 neurônios na primeira camada, 250 na segunda e 150 na terceira.

Quadro 4 – Teste 2: Aprendizagem com 100 imagens, 150x100x50 neurônios

Placa	Acertos	% de Acertos	Erros	% de Erros
40 km/h	22	73.33	8	26.67
80 km/h	21	70.00	9	30.00
Pare	30	100.00	0	0.00
Vire à Direita	22	73.33	8	26.67
Vire à Esquerda	25	83.33	5	16.67
				Acurácia: 80%

Fonte: O Autor (2017).

Quadro 5 – Teste 3: Aprendizagem com 100 imagens, 350x250x150 neurônios

Placa	Acertos	% de Acertos	Erros	% de Erros
40 km/h	24	80.00	6	20.00
80 km/h	26	86.66	4	13.34
Pare	30	100.00	0	0.00
Vire à Direita	26	86.66	4	13.34
Vire à Esquerda	28	93.33	2	6.67
				Acurácia: 89.3%

Fonte: O Autor (2017).

O tempo de aprendizagem no Teste 3 foi de aproximadamente 250 minutos. O arquivo da RNA salvo teve o tamanho de 467.2 MiB. Cada imagem em tempo real foi predita em 0.67 segundos.

Captou-se mais 100 imagens contendo placa, sendo 20 de cada tipo escolhido, e cada uma dessas imagens foi passada pelo mesmo processamento digital de imagens. Nesse instante, a aprendizagem passa a conter 200 imagens segmentadas de placa, sendo 40 de cada tipo.

O Quadro 6 mostra o Teste 4 utilizando o arquivo de aprendizagem da RNA com 200 imagens segmentadas de placas e duas camadas intermediárias, com 100 neurônios na primeira camada e 50 neurônios na segunda.

O tempo de aprendizagem no Teste 4 foi de aproximadamente 90.3 minutos. O arquivo da RNA salvo teve o tamanho de 154.1 MiB. Cada imagem em tempo real foi predita em 0.39 segundos.

Quadro 6 – Teste 4: Aprendizagem com 200 imagens, 100x50 neurônios

Placa	Acertos	% de Acertos	Erros	% de Erros
40 km/h	25	83.33	5	16.67
80 km/h	20	66.66	10	33.34
Pare	30	100.00	0	0.00
Vire à Direita	24	80.00	6	20.00
Vire à Esquerda	27	90.00	3	10.00
Acurácia: 84%				

Fonte: O Autor (2017).

O Quadro 7 mostra o Teste 5 utilizando o arquivo de aprendizagem da RNA com 200 imagens segmentadas de placas e três camadas intermediárias, com 150 neurônios na primeira camada, 100 na segunda e 50 na terceira.

Quadro 7 – Teste 5: Aprendizagem com 200 imagens, 150x100x50 neurônios

Placa	Acertos	% de Acertos	Erros	% de Erros
40 km/h	26	86.66	4	13.34
80 km/h	26	86.66	4	13.34
Pare	30	100.00	0	0.00
Vire à Direita	27	90.00	3	10.00
Vire à	28	93.33	2	6.67
Acurácia: 91.3%				

Fonte: O Autor (2017).

O tempo de aprendizagem no Teste 5 foi de aproximadamente 163.9 minutos. O arquivo da RNA salvo teve o tamanho de 290.5 MiB. Cada imagem em tempo real foi predita em 0.51 segundos.

O Quadro 8 mostra o Teste 6 utilizando o arquivo de aprendizagem da RNA com 200 imagens segmentadas de placas e três camadas intermediárias, com 350 neurônios na primeira camada, 250 na segunda e 150 na terceira.

O tempo de aprendizagem no Teste 6 foi de aproximadamente 332.3 minutos. O arquivo da RNA salvo teve o tamanho de 529.2 MiB. Cada imagem em tempo real foi predita em 0.78 segundos.

Quadro 8 – Teste 6: Aprendizagem com 200 imagens, 350x250x150 neurônios

Placa	Acertos	% de Acertos	Erros	% de Erros
40 km/h	28	93.33	2	6.67
80 km/h	27	90.00	3	10.00
Pare	30	100.00	0	0.00
Vire à Direita	27	90.00	3	10.00
Vire à Esquerda	28	93.33	2	6.67
Acurácia: 93.3%				

Fonte: O Autor (2017).

Captou-se mais 50 imagens contendo placa, sendo 10 de cada tipo escolhido, e cada uma dessas imagens foi passada pelo mesmo processamento digital de imagens. Nesse instante, a aprendizagem passa a conter 250 imagens segmentadas de placa, sendo 50 de cada tipo. O Quadro 9 mostra o Teste 7 utilizando o arquivo de aprendizagem da RNA com 250 imagens segmentadas de placas e duas camadas intermediárias, com 100 neurônios na primeira camada e 50 neurônios na segunda.

Quadro 9 – Teste 7: Aprendizagem com 250 imagens, 100x50 neurônios

Placa	Acertos	% de Acertos	Erros	% de Erros
40 km/h	26	86.66	4	13.34
80 km/h	23	76.66	7	23.34
Pare	30	100.00	0	0.00
Vire à Direita	27	90.00	3	10.00
Vire à Esquerda	26	86.66	4	13.34
Acurácia: 88%				

Fonte: O Autor (2017).

O tempo de aprendizagem no Teste 7 foi de aproximadamente 124.6 minutos. O arquivo da RNA salvo teve o tamanho de 254.1 MiB. Cada imagem em tempo real foi predita em 0.64 segundos.

O Quadro 10 mostra o Teste 8 utilizando o arquivo de aprendizagem da RNA com 250 imagens segmentadas de placas e três camadas intermediárias, com 150 neurônios na primeira camada, 100 na segunda e 50 na terceira.

Quadro 10 – Teste 8: Aprendizagem com 250 imagens, 150x100x50 neurônios

Placa	Acertos	% de Acertos	Erros	% de Erros
40 km/h	27	90.00	3	10.00
80 km/h	28	93.33	2	6.67
Pare	30	100.00	0	0.00
Vire à Direita	28	93.33	2	6.67
Vire à	28	93.33	2	6.67
Acurácia: 94%				

Fonte: O Autor (2017).

O tempo de aprendizagem no Teste 8 foi de aproximadamente 252.8 minutos. O arquivo da RNA salvo teve o tamanho de 448.8 MiB. Cada imagem em tempo real foi predita em 0.72 segundos.

O Quadro 11 mostra o Teste 9 utilizando o arquivo de aprendizagem da RNA com 250 imagens segmentadas de placas e três camadas intermediárias, com 350 neurônios na primeira camada, 250 na segunda e 150 na terceira.

Quadro 11 – Teste 9: Aprendizagem com 250 imagens, 350x250x150 neurônios

Placa	Acertos	% de Acertos	Erros	% de Erros
40 km/h	28	93.33	2	6.67
80 km/h	29	96.66	1	3.34
Pare	30	100.00	0	0.00
Vire à Direita	29	96.66	1	3.34
Vire à Esquerda	29	96.66	1	3.34
Acurácia: 96.6%				

Fonte: O Autor (2017).

O tempo de aprendizagem no Teste 9 foi de aproximadamente 450 minutos. O arquivo da RNA salvo teve o tamanho de 634.1 MiB. Cada imagem em tempo real foi predita em 0.93 segundos.

Alguns outros testes foram realizados aumentando a quantidade de camadas intermediárias e o número de neurônios. Porém, os resultados sempre tenderam aos mesmos do Teste 9, e o tempo de aprendizagem da RNA bem como o tempo de predição de cada imagem em tempo real foram elevados para a ordem de dias e minutos, respectivamente.

O Quadro 12 mostra um resumo de todos os nove testes realizados, contendo o tempo aproximado de aprendizagem, o tamanho em MiB de cada arquivo da RNA salvo, o tempo de

predição médio de cada placa, a acurácia, a quantidade de imagens segmentadas de placas para a aprendizagem, e a quantidade de imagens para teste em tempo real.

Quadro 12 – Resumo dos Nove Testes Realizados

Teste	Tempo de Aprendizagem (min)	Tamanho (MiB)	Tempo de Predição (s)	Acurácia (%)	Quantidade de Imagens para Aprendizagem	Quantidade de Imagens para Teste em Tempo Real
1	53.3	134.1	0.31	64	100	30
2	138.3	200.5	0.42	80	100	30
3	250	467.2	0.67	89.3	100	30
4	90.3	154.1	0.39	84	200	30
5	163.9	290.5	0.51	91.3	200	30
6	332.3	539.2	0.78	93.3	200	30
7	124.6	254.1	0.64	88	250	30
8	252.8	448.8	0.72	94	250	30
9	450	634.1	0.93	96.6	250	30

Fonte: O Autor (2017).

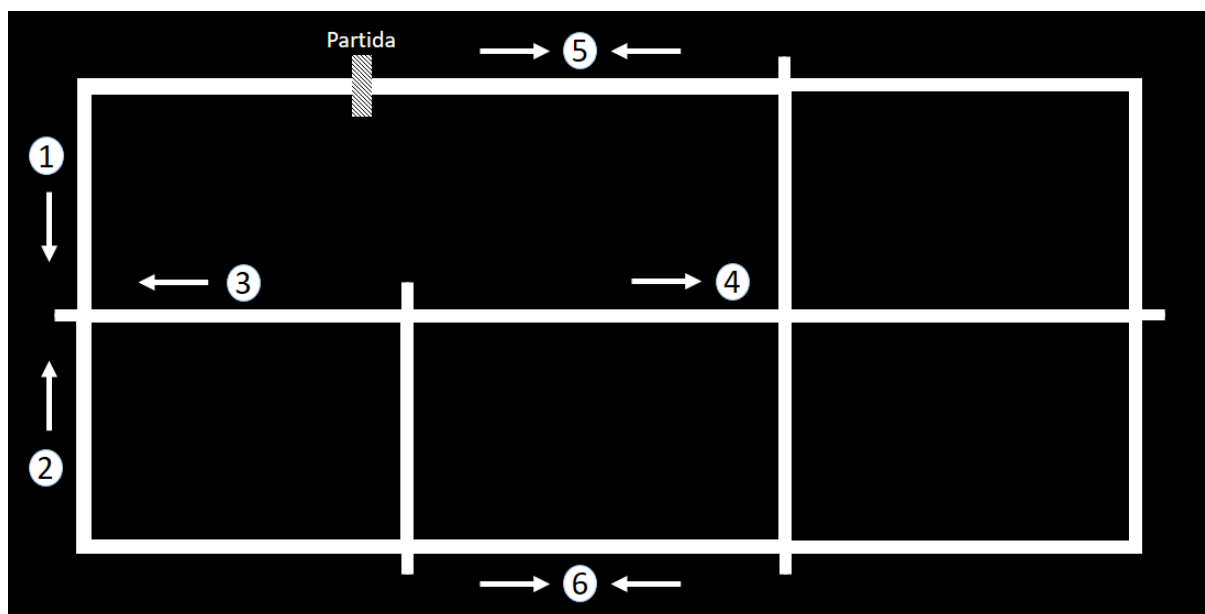
Observando todos os testes, percebe-se principalmente que quanto maior o número de imagens contidas na aprendizagem, e mais camadas intermediárias com mais neurônios, melhor é o resultado. Porém, o tempo para aprendizagem e o tamanho do arquivo da RNA salvo aumenta consideravelmente, bem como o tempo de predição. A acurácia chegou a 96.6% de acertos quando se teve 250 imagens segmentadas de placa para a aprendizagem, e três camadas intermediárias contendo 350 neurônios na primeira camada, 250 na segunda e 150 na terceira.

Com base nesses resultados, o arquivo da RNA usado para o teste em tempo real utilizando o carrinho seguidor de linha foi o Teste 9. Devido a *WebCam* usada ter baixa taxa de captura, quando o carrinho seguidor de linha estava em movimento, a imagem era capturada com distorções na imagem, atrapalhando na predição da RNA. Assim, o carrinho seguidor de linha foi programado para parar de tempo em tempo para realizar a captura sem distorções.

Um desenho gráfico da pista pode ser visto na Figura 17. Após a partida, o carrinho seguidor de linha poderia seguir pelo sentido horário ou anti-horário, dependendo da posição colocada. Caso ele passasse por um sentido igual ao 1 ou 2 indicados pelas setas, ele iria passar

por uma interseção em T, e nesse caso, se houvesse uma placa de vire à direita ou à esquerda, ele efetuou, e caso não houvesse nenhuma placa, ele seguiria em frente, conforme a preferência contida no Código de Trânsito Brasileiro.

Figura 17 – Desenho Gráfico da Pista



Fonte: O Autor (2017).

Se o seguidor de linha passasse por um sentido igual ao 3 indicado pela seta, deveria obrigatoriamente conter uma placa de vire à direita, vire à esquerda ou pare, e assim efetuou a manobra indicada pela placa. Caso nesse mesmo sentido fosse reconhecido uma placa de pare, o carrinho seguidor de linha após parar, iria sempre virar à direita, pois mesmo sem a indicação de uma placa de virar, o carrinho deverá realizar um movimento de virar, pois não há trilha para frente.

Caso ele passasse por um sentido igual ao 4 indicado pela seta, poderia acontecer diversos cenários, sendo reconhecido uma placa de pare ou uma de virar à esquerda ou à direita. Se ele visse uma placa de virar à esquerda ou à direita, ele efetuou a manobra indicada. Já se fosse uma placa de pare, ele parou por um instante e logo após seguiu em frente.

Se o seguidor de linha passasse no local 5 e 6 indicado pelas setas, poderia reconhecer as placas de 80 km/h e 40 km/h. Caso fosse de 80 km/h, o carrinho seguiu em frente em velocidade máxima conseguida pelos motores; porém se fosse uma de 40 km/h, o carrinho seguidor de linha diminuiu a velocidade dos motores pela metade.

5. CONCLUSÃO

Por meio da biblioteca OpenCV instalada em um Raspberry Pi 2 B+, foi possível identificar e reconhecer placas de trânsito durante o percurso de um carro seguidor de linha, através de artifícios de processamento digital de imagens e da aprendizagem da rede neurais artificiais.

Com base em todos os testes realizados, percebe-se principalmente que quanto maior o número de imagens contidas na aprendizagem, e mais camadas intermediárias com mais neurônios, melhor é o resultado. Porém, o tempo para aprendizagem e o tamanho do arquivo da RNA salvo aumenta consideravelmente, bem como o tempo de predição. A acurácia chegou a 96.6% de acertos quando se teve 250 imagens segmentadas de placa para a aprendizagem, e três camadas intermediárias contendo 350 neurônios na primeira camada, 250 na segunda e 150 na terceira.

Durante todo o processo de implementação do trabalho, foram encontradas dificuldades relacionadas à configuração dos códigos para o uso de funções da biblioteca OpenCV. Apesar de bastante usada ao redor do mundo, a maioria das referências sobre RNA implementadas resumem-se a mostrar somente o código, sem muitas discussões sobre as alternativas possíveis e as razões para as configurações escolhidas. Outra dificuldade encontrada, é que a iluminação varia em determinado local onde as placas estão, influenciando no brilho da imagem capturada e atrapalhando a localização da placa.

Devido a *WebCam* usada ter baixa taxa de captura, quando o carrinho seguidor de linha estava em movimento, a imagem era capturada com distorções na imagem, atrapalhando na predição da RNA. Assim, o carrinho seguidor de linha foi programado para parar de tempo em tempo para realizar a captura sem distorções, e quando identificada uma placa, realizou o movimento conforme o esperado.

Como trabalhos futuros, recomenda-se:

- Aumentar o número de imagens para aprendizagem, pois como foi verificado, quanto maior o número de imagens para treinamento, melhor o resultado da RNA.
- Realizar testes com outras configurações de rede, modificando a quantidade de camadas intermediárias, bem como o número de neurônios.
- Realizar testes com localização de placas em rodovias, com intuito de ajudar motoristas no trânsito.
- Realizar a extração de características da imagem, para diminuir o número de neurônios na camada de entrada da rede.
- Realizar processamento digital para o ajuste automático de brilho na imagem capturada.

REFERÊNCIAS

ADEL S. SEDRA, KENNETH C. SMITH. **Microeletrônica**. 5. ed. São Paulo: Pearson, 2007. ISBN 8576050226.

ARDUINO. **Arduino Uno 2560 & Genuino Uno 2560**. Itália, 2016. Disponível em: <<https://www.arduino.cc/en/Main/ArduinoBoardUno2560>>. Acesso em: 15 nov. 2016.

ARRUDA, TIAGO AMADEU. **Arquitetura de hardware e software para supervisão e controle de um carro autônomo**. Dissertação (mestrado em Engenharia Elétrica). Escola de Engenharia da UFMG, 2012.

CANALTCH. **O que é Raspberry Pi?**. São Bernardo do Campo – SP, 2014 . Disponível em: <<https://canaltech.com.br/o-que-e/hardware/o-que-e-raspberry-pi/>>. Acesso em: 01 nov. 2016.

CENTRO DE INFORMAÇÕES DO GALILEO. **Aplicações GALILEO rodoviário**. São José dos Campos – SP, 2005. Disponível em: <<http://www.galileoic.org/node/98?language=pt-br>>. Acesso em: 11 abr. 2016.

CONHECIMENTOGERAL. **OpenCV**. Porto Alegre – RS, 2016. Disponível em: <<http://www.conhecimentogeral.inf.br/opencv>>. Acesso em: 01 nov. 2016.

CUNHA, KÉLITA R. M. G.; JACQUES, MARIA A. P.; CYBIS, HELENA B. B.. **Efeito da percepção dos motoristas sobre as características viário-ambientais nas velocidades praticadas em vias urbanas**. In: Congreso panamericano ingenieria tránsito y transporte, 14., 2006, Las Palmas de Gran Canaria. Não paginado. Disponível em: <redpgv.coppe.ufrj.br/index.php/pt-BR/producao-da-rede/artigos-cientificos/2006-1/228-cunha-jacques-cybis-panam2006/file>. Acesso em: 11 abr. 2016.

FACCHINETTI, LEANDRO; OSÓRIO, FERNANDO SANTOS. **Navegação visual de robôs móveis autônomos baseada em métodos de correlação de imagens**. 20th Brazilian Symposium on Artificial Intelligence (SBIA 2010). São Bernardo do Campo, 2010.

FAZEDORES. **Raspberry Pi B+: Introdução a Porta GPIO**. São Paulo – SP, 2014. Disponível em: <<http://blog.fazedores.com/raspberry-pi-b-introducao-porta-gpio>>. Acesso em: 01 nov. 2016.

FREITAS, ELIAS J. DE R.; VINTI, MATHEUS N. W.; SANTOS, MICHELLE M.; ISCOLD, PAULO; TORRES, LEONARDO A. B.; PEREIRA, GUILHERME A. S.. **Desenvolvimento de automação embarcada para um robô móvel baseado em um carro de passeio**. IX Simpósio Brasileiro de Automação Inteligente (SBAI 2009). Brasília, 2009.

GONZALEZ, R.; WOODS, R.E.. **Processamento de Imagens Digitais**. Editora Edgard Blucher, São Paulo, 2000.

GREENHALGH, J. AND MIRMEHDI, M.. **Real-Time Detection and Recognition of Road Traffic Signs**. IEEE Transactions on Intelligent Transportation System, Vol. 13, No. 4; pp. 1498-1506, 2012.

HALL, MARK; FRANK, EIBE; HOLMES, GEOFFREY; PFAHRINGER, BERNHARD; REUTEMANN, PETER; WITTEN, IAN H.. **The WEKA Data Mining Software: An Update**. SIGKDD Explorations, Volume 11, Issue 1. 2009.

HARALICK R. M.; SHAPIRO, L. G.. **Computer and Robot Vision**, v. 1, 672 p., Addison-Wesley, 1992.

INFOWESTER. **Redes Neurais Artificiais**. São Paulo – SP, 2016. Disponível em: <<http://www.infowester.com/redesneurais.php>>. Acesso em: 02 nov. 2016.

INTEL. **OpenCV 3.2.0**. 23 dez. 2016. Disponível em: <<http://docs.opencv.org/3.2.0/>>. Acessado em <28 jun. 2017>.

KUS, M.C.; GOKMEN, M.; ETANER-UYAR, S.. **Traffic Sign Recognition using Scale Invariant Feature Transform and Color Classification**. 23rd International Symposium on Computer and Information Sciences; pp. 1-6, 2008.

LIMA, DANILO ALVES DE; PEREIRA, GUILHERME A. S.. **Um sistema de visão estéreo para navegação de um carro autônomo em ambientes com obstáculos**. XVIII Congresso Brasileiro de Automática (CBA 2010). Bonito-MS, 2010.

MARQUES FILHO, OGÊ; VIEIRA NETO, HUGO. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport, 1999. ISBN 8574520098.

NEWTON, BRAGA C.. **Ponte-H com controle PWM (mec009)**. São Paulo – SP, 2015. Disponível em: <<http://www.newtoncbraga.com.br/index.php/artigosmec/1213-ponte-h-com-pwm.html>>. Acesso em: 01 jul. 2017.

PETCIVIL. **Redes Neurais Artificiais**. São Paulo – SP, 2013. Disponível em: <<https://blogdopetcivil.com/2013/07/05/redes-neurais-artificiais>>. Acesso em: 02 nov. 2016.

POFFO, FERNANDO. **Visual Autonomy – protótipo para reconhecimento de placas de trânsito**. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação). Universidade Regional de Blumenau, 05 de julho de 2010.

RIBEIRO, MATHEUS V. L.; SALOMÃO, JOÃO M.. **Desenvolvimento de um sistema para reconhecimento automático de placas de trânsito**. Anais do XX Congresso Brasileiro de Automática. Belo Horizonte, MG, 2014.

RODRIGUES, RICARDO CEZAR B.; PELLEGRINO, SERGIO ROBERTO M.; FORSTER, CARLOS HENRIQUE Q.. **Extração de Informações de Cor e Forma para Detecção de Placas de Trânsito em Imagens**. XXI Brazilian Symposium on Computer Graphics and Image Processing (SIBIGRAPI 2008). Campo Grande, MS, 2008.

THEODORIDIS, S.; KOUTROUMBAS, K.. **Pattern recognition**. 3. ed. San Diego: Academic Press, 2006.