

**CENTRO PAULA SOUZA  
FACULDADE DE TECNOLOGIA  
FATEC SANTO ANDRÉ  
Tecnologia em Eletrônica Automotiva**

**Fênix Samuel Tassinari  
Filêmon SimeiTassinari**

**SISTEMA DE IDENTIFICAÇÃO ASSISTIDA DO  
ESTADO DE SEMÁFOROS UTILIZANDO REDES  
NEURAIS**

Santo André  
2018

**CENTRO PAULA SOUZA  
FACULDADE DE TECNOLOGIA  
FATEC SANTO ANDRÉ  
Tecnologia em Eletrônica Automotiva**

**Fênix Samuel Tassinari  
Filêmon SimeiTassinari**

**SISTEMA DE IDENTIFICAÇÃO ASSISTIDA DO  
ESTADO DE SEMÁFOROS UTILIZANDO REDES  
NEURAIS**

Monografia apresentada ao Curso de Tecnologia em Eletrônica Automotiva da FATEC Santo André, como requisito parcial para obtenção do título de Tecnólogo em Eletrônica Automotiva.

Orientador: Prof. Me. Murilo Zanini de Carvalho.

Santo André  
2018

T213s

Tassinari, Fênix Samuel

Sistema de identificação assistida do estado de semáforos utilizando redes neurais / Fênix Samuel Tassinari, Filêmon Simei Tassinari. - Santo André, 2018. – 55f: il.

Trabalho de Conclusão de Curso – FATEC Santo André.  
Curso de Tecnologia em Eletrônica Automotiva, 2018.

Orientador: Prof. Me. Murilo Zanini de Carvalho

1. Eletrônica. 2. Sistema de identificação de cores. 3. Semáforos. 4. Pessoas. 5. Daltonismo. 6. Controle de tráfego urbano. 7. Deep Learning. 8. YOLO. I. Tassinari, Filêmon Simei. II. Sistema de identificação assistida do estado de semáforos utilizando redes neurais.

621.389

**LISTA DE PRESENÇA**

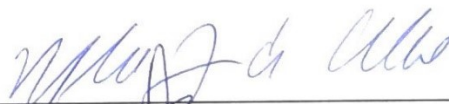
SANTO ANDRÉ, 19 DE DEZEMBRO DE 2018

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA "SISTEMA DE IDENTIFICAÇÃO ASSISTIDA DO ESTADO DE SEMAFOROS UTILIZANDO REDES NEURAIIS" DOS ALUNOS DO 6º SEMESTRE DESTA U.E.

**BANCA**

PRESIDENTE:

PROF. MURILO ZANINI DE CARVALHO

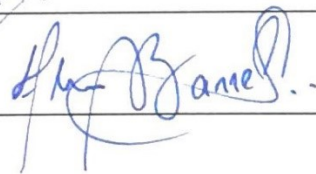


MEMBROS:

PROF. FERNANDO GARUP DALBO



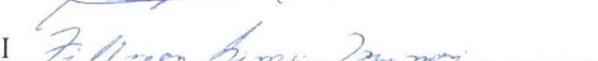
PROF. FLÁVIO AUGUSTO BARRELLA

**ALUNOS:**

FÊNIX SAMUEL TASSINARI



FILÊMOM SIMEI TASSINARI



Dedicamos esse trabalho em memória de Tabajara João Tassinari.

## **AGRADECIMENTOS**

Agradecemos ao nosso pai Tabajara João Tassinari que dedicou em nos acompanhar deste o início de nossas vidas, nos acompanhou nesse trabalho, mas por complicações de saúde não sobreviveu para ver a conclusão.

Ao Professor Murilo Zanini de Carvalho que sem a sua orientação, motivação e seu planejamento em cada fase do projeto, não teríamos conseguido concluir esse trabalho.

Ao Professor Carlos Alberto Morioka que nos ajudou e nos incentivou a não desistir do projeto e continuar até o final.

Ao Professor Fernando Garup Dalbo pelas suas aulas ministradas nessa etapa final e o seu encorajamento que nos deu a força que precisávamos para continuar seguindo em frente.

A todos os professores e funcionários da Fatec Santo André que sempre nos ajudaram de diversas formas possíveis durante todos esses anos de curso.

## RESUMO

Segundo o código de trânsito brasileiro (lei nº9.503/1997), os candidatos a direção de veículos para obterem a carteira nacional de trânsito devem ter a capacidade de identificar as cores verde, amarelo, e vermelho que são as cores de sinalização dos semáforos de controle de tráfego urbano, mesmo com projeto de lei apresentado no senado em favor dos portadores de discromatopsia, o PLS 9/2013, que altera o Anexo II da Lei nº 9.503, de 23 de setembro de 1997, que institui o Código de Trânsito Brasileiro, para dispor sobre o formato da sinalização semafórica, as novas especificações do semáforo teria a luz vermelha no formato quadrado para que o foco da luz vermelha seja quadrado, a luz amarela no formato triangular e a luz verde no formato circular, mas para isso seria necessário uma padronização de todo sistema de controle de tráfego urbano brasileiro para que essa lei seja alterada. O objetivo deste trabalho é apresentar um sistema de detecção de cores para semáforos para auxiliar pessoas com daltonismo. O algoritmo utilizado para a detecção dos semáforos em imagens estáticas, previamente carregadas no sistema, foi a aplicação da rede YOLO (You Only Look Once) utilizando o conjunto de ferramentas da OpenCV em Python. Os resultados obtidos foram positivos, e as cores dos semáforos puderam ser detectados nas imagens apresentadas ao sistema.

Palavras chaves: Sistemas de Cores, Visão Computacional, *Deep Learning*, YOLO.

## ABSTRACT

According to the Brazilian traffic law (Law n ° 9.503 / 1997), candidates for vehicle management to obtain the national transit portfolio must be able to identify the green, yellow, and red colors that are the signaling colors of the traffic lights of urban traffic control, even with a bill presented in the Senate in favor of those with discromatopsia, PLS 9/2013, which amends Annex II of Law 9,503, of September 23, 1997, which establishes the Traffic Code Brazilian, in order to dispose on the format of traffic light, the new specifications of the traffic light would have the red light in the square format so that the focus of the red light is square, the yellow light in the triangular format and the green light in the circular format, but for that it would be necessary to standardize every Brazilian urban traffic control system so that this law can be changed. The objective of this project is to present a color detection system for traffic lights to help people with color blindness. The algorithm used to detect static images, preloaded in the system, was an application of the YOLO (You Only Look Once) network using the OpenCV Python toolkit. The results were positive, and the colors of the traffic lights could be detected in the images appropriate to the system.

Key words: Color Systems, Computational Vision, *Deep Learning*, YOLO.



## LISTA DE ILUSTRAÇÕES

Figura 1 – Guarda civil controlando o semáforo, foto de como era realizado o controle do tráfego urbano antes de ser implementado o controle eletromecânico de semáforo. ....	13
Figura 2 – Feixes de luz, representação gráfica da trajetória dos raios de luz.....	16
Figura 3 – Representação de uma matriz do CCD .....	20
Figura 4 – Processamento de imagem, ilustra o condicionamento do sinal obtido do CCD em imagem digital.....	21
Figura 5 – Representação dos valores de cada elemento que forma um pixel de imagem .....	21
Figura 6 – Aquisição de imagens coloridas, ilustração de como o filtro RGB decompõe as cores de uma fonte luminosa.....	22
Figura 7 – Representação da detecção de objetos com a arquitetura YOLO. ....	24
Figura 8 – Arquitetura de uma Rede Neural Convolucional.....	25
Figura 9 – Representação da previsão de quais objetos estão imagem. ....	25
Figura 10 – Celular Huawei Y6 li.....	27
Figura 11 – Fotografia com alguns elementos do trânsito que interferem na detecção dos semáforos .....	29
Figura 12 – Fotografia retirada quando o semáforo estava na mudança de estado ....	30
Figura 13 – Foto com o semáforo e seu estado identificado e classificado... ..	36
Figura 14 – Foto onde o algoritmo detecta o semáforo sem a cor .....	37
Figura 15 – Foto com falta de foco e reflexo do painel .....	38
Figura 16 – Foto com a identificação parcial do semáforo .....	38
Figura 17 – Fluxograma do projeto .....	44
Figura 18 – Fluxograma do funcionamento da função principal. ....	45
Figura 19 – Fluxograma da função que identificação das camadas.....	46
Figura 20 – Função de validação do objeto detectado da imagem... ..	47

Figura 21 – Fluxograma da função que identificação das camadas.....	48
Figura 22 – Função de classificação de múltiplas detecções .....	49

## **LISTA DE QUADROS**

Quadro 1 – Temperatura das cores .....	18
Quadro 2 – Proporção das cores obtidas no sistema RGB .....	19
Quadro 3 – Proporção das cores obtidas no sistema CMY .....	20

## **LISTA DE TABELA**

Tabela 1 – Analise das amostras em bancada .....	36
--	----

## LISTA DE ABREVIATURAS E SIGLAS

ABRAMET	Associação Brasileira de Medicina do Tráfego
CONTRAN	Conselho Nacional de Trânsito
DER	Departamento de Estrada e Rodagem
RGB	Sistema de cores primário - <i>Red, Green, Blue</i>
CMY	Sistema de cores secundário - <i>Cyan, Magenta, Yellow</i>
CCD	<i>Charge-coupled device</i>
ADC	<i>Analog-to-Digital Converter</i>
YOLO	<i>You Only Look Once</i>
OpenCV	<i>Open Source Computer Vision Library</i>

# SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO.....</b>	<b>10</b>
1.1.	Motivação .....	12
1.2.	Objetivos .....	12
1.3.	Contribuições Esperadas.....	12
1.4.	Organização do Trabalho.....	13
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA.....</b>	<b>14</b>
2.1	Histórico do Semáforo .....	14
2.2	Fenômenos Ópticos .....	15
2.3	Sistemas de Cores RGB e CMY .....	17
2.4	Aquisição de Imagens .....	20
2.4.1	Aquisição de Imagens Coloridas.....	22
2.5	Redes Neurais Artificiais.....	23
2.5.1	Aprendizado Profundo .....	23
2.5.2	Biblioteca OpenCV .....	23
2.5.3	Arquitetura de Rede Neural.....	24
2.5.4	Rede Neural Convolutacional .....	24
2.5.5	Precisão da Rede Neural .....	25
<b>3</b>	<b>METODOLOGIA.....</b>	<b>27</b>
3.1	Ambiente de desenvolvimento .....	27
3.2	Sistema de detecção do semáforo de tráfego urbano.....	28
<b>4</b>	<b>DESENVOLVIMENTO .....</b>	<b>31</b>

4.1	Descrição do Algoritmo .....	31
5	RESULTADOS E DISCUSÃO .....	35
5.1	Testes Realizado.....	35
5.2	Resultados Obtidos .....	36
5.3	Análise dos Resultados .....	37
6	CONSIDERAÇÕES FINAIS .....	39
6.1	Propostas Futuras .....	39
	REFERÊNCIAS BIBLIOGRÁFICAS .....	40
	APÊNDICE .....	42

# 1 INTRODUÇÃO

Os portadores de Discromatopsia, mais conhecida como daltonismo, apresentam dificuldade na percepção de determinadas cores primárias, como o verde e o vermelho, o que se repercute na percepção das restantes cores do espectro visível (BRUNI; CRUZ, 2006).

No olho humano, a retina normalmente possui três tipos de cones sensíveis às cores. Cada um desses tipos capta o comprimento de onda de uma luz de cor primária, com o azul, verde ou vermelho. Dependendo do comprimento de onda da luz, um determinado tipo de cone é acionado. Esse por sua vez emite um sinal ao cérebro, permitindo que a pessoa identifique a cor, no entanto pessoas daltônicas interpretam as cores de maneira diferente, visto que a sensibilidade dos cones a uma ou mais cores é fraca ou capta o comprimento de onda errado (BRUNI; CRUZ, 2006).

A maioria dos daltônicos tem dificuldade em distinguir entre amarelo, verde, vermelho e marrom. Por causa dessa deficiência, pode ser difícil identificar as cores do semáforo de trânsito urbano. Se os cones sensíveis ao vermelho são muito fracos, o sinal vermelho pode parecer preto (SOARES, 2009).

Neto (2010) aponta que 15 milhões de brasileiros, na proporção de 20 homens para cada mulher, têm alteração congênita nos cones, as células da retina que permitem distinguir as cores.

Os daltônicos normalmente nascem com essa deficiência, e as crianças com esse problema aprendem a compensá-lo sem perceber. Por exemplo, mesmo que não consigam ver a diferença entre certos tons, elas podem notar diferenças no contraste e no brilho, associando essas variações aos nomes das cores. Também podem aprender a identificar objetos por meio do aspecto da superfície e da textura. Dessa forma os portadores garantem que conseguem decifrar as cores dos semáforos pela ordem padronizada das luzes, assim como a mudança de intensidade das emissões (SOARES, 2009).



Pietrantonio (2005) destaca que com relação aos fatores humanos, o comportamento do motorista, principalmente quanto à sua capacidade de identificação e reação aos estímulos externos, proveniente do sistema viário e seu entorno e as informações necessárias para o desempenho da tarefa de dirigir um veículo são fornecidas pelos elementos de sinalização.

A Resolução 267/2008 do Conselho Nacional de Trânsito (CONTRAN), publicada em 25 de fevereiro de 2008, consta o teste de visão cromática, no parágrafo 3 do anexo II da avaliação oftalmológica, que prevê que os candidatos à direção de veículos devem ser capazes de identificar as cores verde, amarela e vermelha, que são as do semáforo de trânsito.

No (G1.COM, 2010) o Departamento de Estrada e Rodagem (DER) do Rio de Janeiro, informa que a população na sua maioria conhece a disposição das luzes do semáforo, e por isso dificilmente são reprovadas no exame médico, mas o que realmente prejudica essas pessoas é a falta de padronização de semáforos. Para que isso seja possível apresenta-se neste trabalho uma ideia de um sistema de identificação, no qual é possível identificar qual é a cor do sinal que o semáforo se encontra.

A metodologia desse trabalho é baseada em um sistema de detecção de objetos, composto de um elemento óptico que realizara a aquisição dos feixes de luz emitida pelo semáforo e depois convertê-los em sinais elétricos, esses sinais corresponde a forma do semáforo e a mudança dos estados verde, amarelo e o vermelho.

Primeiramente foi realizado um estudo dos elementos óticos normalmente encontrados nas câmeras digitais, depois um estudo sobre a utilização da visão computacional para o reconhecimento e classificação de objetos usando modelos pré-treinado, por fim o desenvolvimento de um *software* que identificará o semáforo e informará o estado do mesmo.

## 1.1 MOTIVAÇÃO

Uma das motivações deste trabalho está baseada no fato de que existe um aumento significativo na discussão sobre a inclusão de pessoas portadoras de alguma deficiência na sociedade moderna, porém muito pouco é feito para que essa inclusão seja realizada.

Como um dos problemas enfrentados pelas autoridades responsáveis é o elevado custo das propostas apresentadas, soluções de baixo custo tornam essa acessibilidade mais abrangente a todos que tem necessidades especiais.

## 1.2 OBJETIVO

O objetivo principal deste trabalho foi construir uma metodologia que auxilie na detecção das cores em um semáforo, para auxiliar os portadores de [daltonismo], seguindo a lei nº9.503/1997, e a Resolução 267/2008-CONTRAN.

Este objetivo será alcançado, atendendo os seguintes objetivos secundários:

- Criação de um sistema para captura de imagens;
- Criação de um sistema para localizar os dados dentro da imagem;
- Garantir que os sistemas criados obtenham 15% de assertividade.

## 1.3 CONTRIBUIÇÕES ESPERADAS

As contribuições estão relacionadas com os objetivos descritos na subseção 1.2 e são elas:

- Uma nova proposta de acessibilidade para o trânsito brasileiro, sem a necessidade de uma padronização visto que apenas 15 milhões de brasileiros têm daltonismo;
- Adaptação de uma métrica para captura e reconhecimento de imagem aplicada no auxílio de identificação da sinalização de trânsito.

## 1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado de maneira que se possa associar os conhecimentos de óptica geométrica com as representações digitais da luminosidade das cores dos pixels no processamento de imagens, realizando o reconhecimento do estado do semáforo e gerando um sinal para o condutor.

Na Fundamentação teórica descrevem-se sobre uma revisão bibliográfica sobre o sistema de controle de tráfego, fenômenos ópticos e de propagação de ondas, também a construção e utilização de um sistema de aquisição de imagem, e o processo de obtenção das cores em sistemas digitais.

Na Metodologia descrevem-se sobre a câmera, e o ambiente de desenvolvimento utilizado.

No Desenvolvimento discorre na prática o desenvolvimento do *software* de *Deep Learning* em *Python*, utilizando a arquitetura YOLO.

Nos Resultados e Discussão discorre sobre os resultados obtidos e testes realizados em imagens estáticas de um veículo durante o tráfego urbano em algumas alterações climáticas e em horários diferenciados.

Nas Considerações Finais encontra-se as conclusões sobre o trabalho e as propostas futuras para o projeto.

## 2 FUNDAMENTAÇÃO TEÓRICA

Conceitos fundamentais necessários para compreender o funcionamento e a utilização de sinalização para o controle de tráfego urbano, e conceito utilizados para o desenvolvimento e para a elaboração de um sistema de identificação de objetos e cores.

As seções seguintes discutem o contexto histórico do desenvolvimento e utilização do controle de tráfego urbano, também descreve os sistemas digitais utilizados na formação e reprodução das cores.

### 2.1 HISTÓRICO DO SEMÁFORO

Em 1866, devido aos acidentes de trânsito que em apenas um ano mais de 1.102 pessoas foram mortas e 1.334 feridos nas estradas em Londres, John Peake Knight propôs um sistema de sinalização baseado em sinais ferroviários composto de uma lanterna a gás rotativo com um vermelho e uma luz verde (RENTERÍA, 2002).

Os primeiros semáforos de trânsito elétrico foram instalados na cidade de Cleveland, em Ohio nos Estados Unidos em 1913 com operação manual onde havia um guarda civil que controlava a mudança de estado conforme mostrado na Figura 1 (RENTERÍA, 2002).

**Figura 1** – Guarda civil controlando o semáforo, foto de como era realizado o controle do tráfego urbano antes de ser implementado o controle eletromecânico de semáforo.



**Fonte:**[Extraída de (<http://historiacomfotos.blogspot.com/2016/03/o-semaforo-e-sua-origem.html>)

Acessado em 08/12/2018].

O primeiro semáforo com controle elétrico foi em 1917 em Salt Lake City, e depois em 1928, muitas outras cidades como Nova Iorque tinham vários semáforos.

Em 1952, engenheiros de Denver utilizaram computadores analógicos para a mudança do seletor de estado, apresentando uma melhoria para o plano de tempo para os sinais (RENTERÍA, 2002).

Entre o primeiro sinal de trânsito em Cleveland e os anos 50, o sistema mais usado para controladores de tráfego urbano era o controlador eletromecânico pré-ajustado. Durante os anos 50 e 60, a maioria das cidades americanas tinham esse dispositivo (RENTERÍA, 2002).

Os avanços na eletrônica e a disponibilidade comercial de novos componentes, que ocorreu na década de 60 forneceram muitos avanços nos equipamentos para controle de tráfego, denominados de Sistemas de Controle de Tráfego Urbano (UTCS Urban Traffic Control System) que consistiam de conjuntos de semáforo eletromecânicos (RENTERÍA, 2002).

Computadores digitais foram pela primeira vez utilizados em controle de tráfego em Toronto no Canadá, em 1963. Empregando-se transistores que antigamente eram reservados para aplicações militares, produtos para controle de tráfego com maiores recursos foram lançados no mercado (RENTERÍA, 2002).

Os controladores eletromecânicos foram substituídos por componentes em estado sólido, resultando em sistemas com baixa manutenção, e nova fonte de luz, como o uso de diodos emissores de luz com alta intensidade em comparação às fontes de luz utilizadas convencionalmente (RENTERÍA, 2002).

De forma geral, o avanço no controle de sinais de trânsito vem melhorando cada vez mais, junto com a busca de novas tecnologias que diminuam os efeitos causados pelo número crescente de veículos nas ruas (RENTERÍA, 2002), porém ainda é preciso soluções para a falta de acessibilidade desses equipamentos.

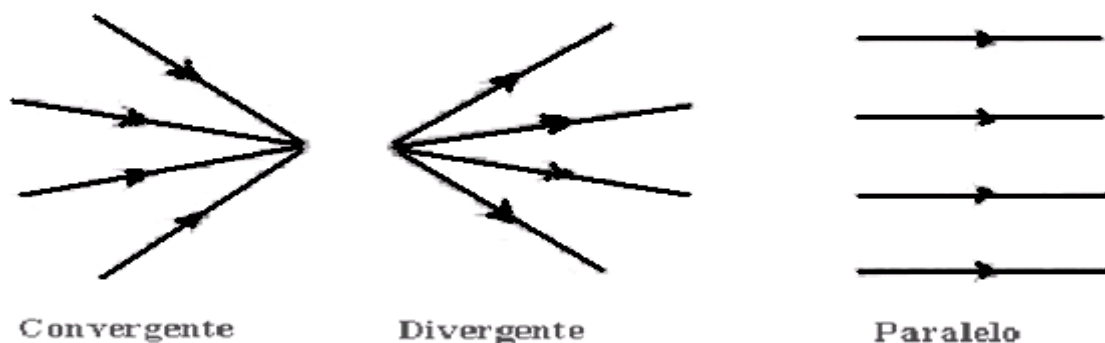
## 2.2 FENÔMENOS ÓPTICOS

Óptica é o estudo sobre a luz e os fenômenos luminosos, no meio homogêneo existem fenômenos luminosos onde é possível ser estudado e descrito o sentido, e a direção da propagação da luz graficamente apenas com fundamentos de geometria denominado óptica geométrica (TOLEDO, 2009).

Os corpos luminosos são aqueles que produzem e emitem luz, como as lâmpadas elétricas de acordo com a fonte a luz pode ser classificada em simples ou monocromática, e composta ou policromática, no caso das fontes de luz onde suas dimensões são desprezíveis em relação a distância de outros corpos são chamadas de fonte puntiforme ou pontual, as fontes de luz que possuem suas dimensões relevantes a distância de outros corpos são chamados de fonte extensa (TOLEDO, 2009).

Os Raios de luz são a representação gráfica da trajetória da luz indicado o sentido e a direção através de um segmento de setas orientadas, o feixe de luz é o conjunto dos raios de luz denominados convergente quando os raios de luz se concentram de um ponto em comum, divergente onde os raios de luz saem de um ponto comum, ou paralelo também denominada de fonte de luz imprópria por se encontrar no infinito conforme a Figura 2 (TOLEDO, 2009).

**Figura 2** – Feixes de luz, representação gráfica da trajetória dos raios de luz.



**Fonte:**[Extraída de (Os Fundamentos da Física Vol. 2 editora MODERNA, 2009)].

A trajetória de um feixe de luz não é afetada durante a intersecção de um outro feixe de luz esse princípio é chamado de independência dos raios de luz (TOLEDO, 2009).

## 2.3 SISTEMAS DE CORES RGB E CMY

Em suas experiências com a luz, Isaac Newton, descobriu que o fluxo luminoso tem uma frequência regular, decompondo a luz do sol e obteve as cores vermelho, verde, azul claro, azul escuro, amarelo, laranja e violeta. Depois quando tentou recompor a luz branca através de um disco onde pintou as sete cores e as girou por meio de uma manivela obteve somente as cores vermelha, verde e azul (ROCHA, 2010).

Posteriormente ele pintou num outro disco as cores vermelha, verde e azul e obteve algo próximo do branco. Desta forma, Newton descobre que para recompor a luz branca não são necessárias as setes cores originais, mas apenas o vermelho, o verde e o azul (ROCHA, 2010).

Como nesse processo é preciso que o vermelho, o verde e o azul precisam ser misturados em proporções diferentes para se obter as outras cores, um padrão técnico para a definição de cor em tecidos foi desenvolvido no século 19, basicamente uma especificação da proporção de pigmentos usada para a formação das cores (ROCHA, 2010).

Visto que todo os corpos que possuem uma temperatura maior que  $-273^{\circ}\text{C}$  podem emitir luz, o físico britânico Willian Thomson, (Sir Willian Thomson, 1° Barão de Kelvin of Largs) ou simplesmente Lord Kelvin. Ao estudar a expansão dos gases, elaborou uma escala (escala Kelvin) que, mais tarde, seria usada para medir a cor das fontes luminosas (ROCHA, 2010).

Em uma experiência para medir a cor, Thomson aqueceu uma barra de ferro até atingir a cor vermelha essa temperatura foi medida na sua própria graduação, ou seja, graus Kelvin (ou  $\text{K}^{\circ}$ ). Nesta escala, a barra de ferro aquecida até o vermelho atingiu  $1200 \text{ K}^{\circ}$  e através de cálculos, as outras cores foram obtidas (ROCHA, 2010).

Na escala Kelvin,  $0^{\circ}$  equivale a  $-273^{\circ}$  Celsius, portanto, basta subtrair 273 de qualquer medição em Kelvin para se obter o equivalente em graus Celsius, pois as duas escalas seguem o sistema métrico-decimal, desta forma as diferentes fontes

luminosas possuem várias cores, mas apenas uma cor é visível este fenômeno é chamado de temperatura da cor (ROCHA, 2010), conforme o Quadro 1:

**Quadro 1** - Temperatura das cores.

<b>Cor</b>	<b>Temperatura (Kelvin)</b>
Vermelho	1200°K
Laranja	3200°K
Amarelo	3800°K
Verde	4500°K
Azul claro	5600°K
Azul escuro	7000°K
Violeta	11000°K

**Fonte:**[Extraída e adaptada de (ROCHA, 2010)].

No entanto, o sistema de Isaac Newton que permite a obtenção de cores através da combinação do vermelho, do azul e do verde é completamente diferente daquele empregado na obtenção das cores de tintas. Desta forma notou-se que existiam duas formas principais para realizar o manuseio das cores, um sistema para cores de corpos emissores de luz, chamado de RGB e outro sistema para corpos que reflete a luz chamado de CMY (ROCHA, 2010).

O RGB é o sistema mais conhecido e usado para a geração e propagação dos feixes de luz das cores, utilizados na produção de fotografia, cinema, vídeo, televisão, câmeras e etc (ROCHA, 2010).

Esse sistema trabalha por adição, ou seja, se somarmos as três cores básicas, nas proporções corretas, obteremos a cor branca (ROCHA, 2010). As cores no sistema RGB são obtidas na seguinte proporção conforme o Quadro 2:



**Quadro 2** - Proporção das cores obtidas no sistema RGB.

<b>Cor</b>	<b>Proporção RGB Porcentual</b>	<b>RGB Porcentual</b>
Branco	30% Vermelho + 59% Verde + 11% Azul	100%
Amarelo	30% Vermelho + 59% Verde	89%
Ciano	59% Verde + 11% Azul	70%
Verde	59% Verde	59%
Magenta	41% Cor secundaria	41%
Vermelho	30% Vermelho	30%
Azul	11% Azul	11%
Preto	Considerado ausência de informação no RGB	0

**Fonte:**[Extraída e adaptada de (ROCHA, 2010)].

A complexidade da percepção de cores do olho humano torna o processo de obtenção de cores do sistema RGB impreciso, mas entre todos os sistemas existentes como CMYK, HLS, HSB e HSV é o que permite melhor correlação entre o real e o virtual (ROCHA, 2010).

Já o CMY é conhecido o sistema de Cor Pigmento, e trabalha por subtração, ou seja, se somarmos as três cores nas proporções corretas obteremos preto, isso torna necessário que o papel seja branco para que possa ter a reflexão da luz, caso contrário haveria uma distorção nas cores. Na indústria é utilizado o CMYK, no qual o Preto é adicionado e não obtido por meio de mistura (ROCHA, 2010).

Assim, o CMYK é baseado em quatro cores e foi criado como uma opção mais barata, pois não necessita de pigmentos puros e mais caros, sendo usado para impressões em larga escala. A letra K, do CMYK, tanto significa preto (Black), como chave (Key), pois a cor preta é usada para interferir nos detalhes na impressão (ROCHA, 2010). As cores no sistema CMY seguem a seguinte proporção em relação ao RGB conforme o Quadro 3:

**Quadro 3** - Proporção das cores obtidas no sistema CMY.

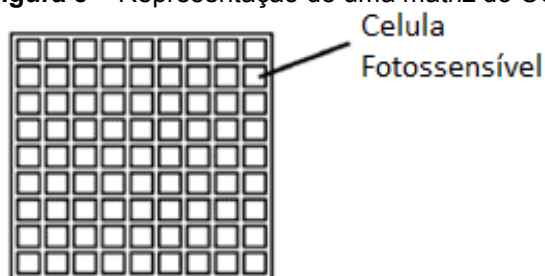
Cor	Proporção RGB Porcentual	RGB Porcentual
Preto	30% Ciano + 59% Magenta + 11% Amarelo	100%
Azul	30% Ciano + 59% Magenta	89%
Vermelho	59% Magenta + 11% Amarelo	70%
Magenta	59% Magenta	59%
Verde	30% Ciano + 11% Amarelo	41%
Ciano	30% Ciano	30%
Amarelo	11% Amarelo	11%
Branco	Considerado ausência de informação no CMY	0

Fonte:[Extraída e adaptada de (ROCHA, 2010)].

## 2.4 AQUISIÇÃO DE IMAGENS

A aquisição de imagem em uma câmera digital consiste em um sensor de imagem que converte a luz refletida através da lente em cargas elétricas. O sensor de imagem utilizado pela maioria das câmeras digitais é um elemento constituído de uma matriz de células fotossensível formado pelo dispositivo de CCD que convertem os feixes luz em elétrons, gerando um sinal elétrico que representa a imagem capturada pela lente. A tensão elétrica gerada em cada célula representa um ponto da imagem capturada pela lente (SCURI, 2002), demonstrada pela Figura 3.

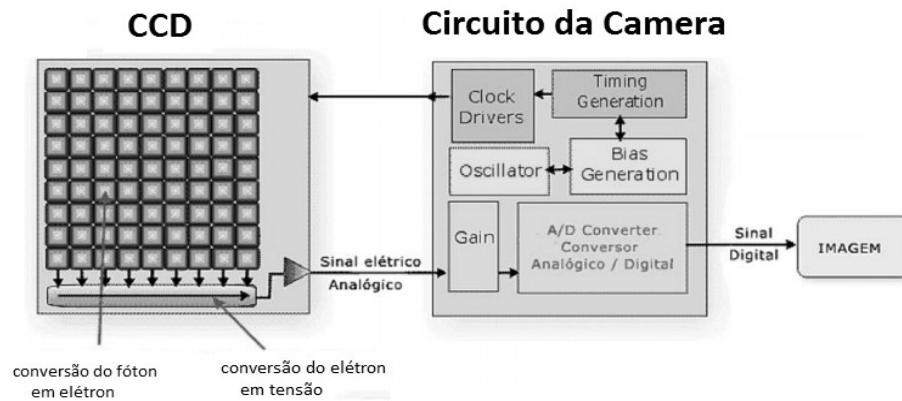
**Figura 3** – Representação de uma matriz do CCD.



Fonte: [Extraída de (<http://labculturviva.org/pontobrasil/materialdidatico/imagemeletronica.pdf>) Acessado 3/12/2013].

Esse sinal elétrico gerado pelo CCD é condicionado por um conversor ADC representado pela Figura 4, onde o número de bits utilizados para armazenar na memória a informação da matriz determina a resolução de intensidade da imagem digital, desse modo a resolução da imagem é determinada pelo número de elementos da matriz conhecido como elemento de imagem ou pixel, que é a menor unidade da imagem eletrônica (SCURI, 2002).

**Figura 4** – Processamento de imagem, ilustra o condicionamento do sinal obtido do CCD em imagem digital.



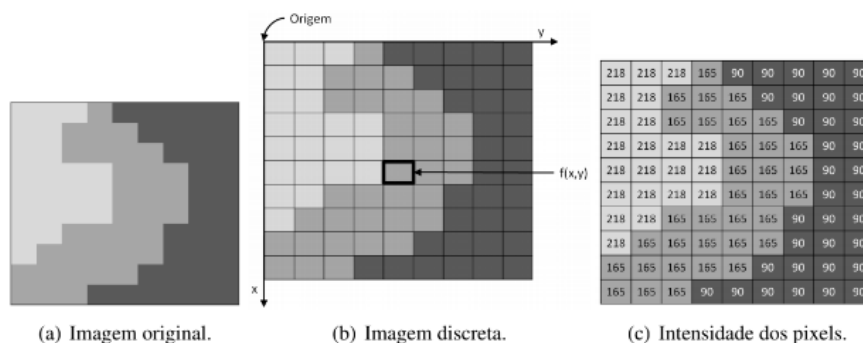
**Fonte:** [Extraída de (<http://labculturviva.org/pontobrasil/materialdidatico/imagemeletronica.pdf>) Acessado 3/12/2013].

Os pixels são dados por uma função bidirecional de intensidade da luz  $f(x, y)$ , onde  $x$  e  $y$  são as coordenadas espaciais e o valor de  $f$  é proporcional ao brilho da imagem naquele ponto especificamente (SCURI, 2002).

Como esse sinal analógico é contínuo no tempo, torna necessário o registro da imagem, para realizar a conversão digital, assim a reprodução do sinal digital e a percepção que se tem da imagem é afetada pela quantidade de pontos da amostragem e pelo intervalo de tempo que esse sinal é coletado, influenciando o dimensionamento do pixel ao longo dos eixos  $x$  e  $y$  (SCURI, 2002).

Em uma imagem monocromática, durante a sua digitalização o pixel varia seu valor de acordo com a intensidade da luz, representando a luz branca com o valor mais alto possível até a ausência total de luz representado pelo valor mais baixo possível do sistema (SCURI, 2002), relacionando esses valores obtidos com uma escala de cinza conforme a Figura 5.

**Figura 5** – Representação dos valores de cada elemento que forma um pixel de imagem.

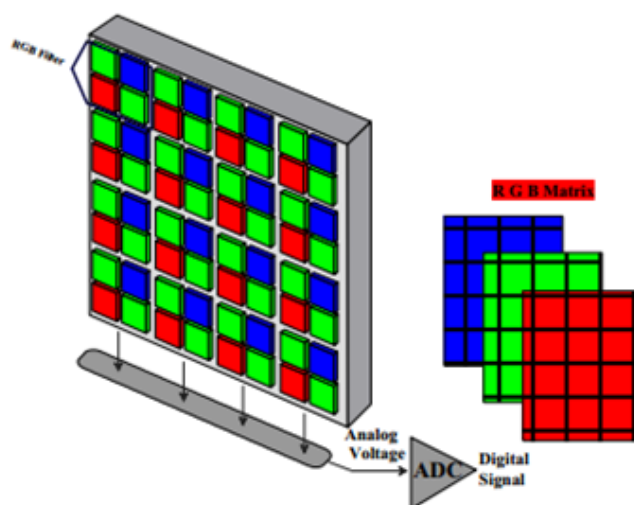


**Fonte:** [Extraída de (<http://labculturviva.org/pontobrasil/materialdidatico/imagemeletronica.pdf>) Acessado 3/12/2013].

### 2.4.1 AQUISIÇÃO DE IMAGENS COLORIDAS

O método mais utilizado para a aquisição de imagens coloridas é o filtro RGB onde uma imagem é representada por três matrizes que permite decompor a luz em cada um dos canais RGB conforme ilustrado pela Figura 6 (GOMEZ; HERNANDEZ, 2011).

**Figura 6** – Aquisição de imagens coloridas, ilustração de como o filtro RGB decompõe as cores de uma fonte luminosa.



**Fonte:**[Extraída de ([http://cdn.intechopen.com/pdfs/17632/InTech-Digital\\_image\\_processing\\_using\\_labview.pdf](http://cdn.intechopen.com/pdfs/17632/InTech-Digital_image_processing_using_labview.pdf)) Acessado 12/11/2013].

Em uma imagem monocromática o valor do pixel é dado em valores escalar, para a imagem colorida o valor do pixel está associado a vetores formado por uma sequência de imagens monocromáticas obtidas pela decomposição dos canais do filtro RGB, assim uma imagem colorida é denominada de imagem multibanda, onde a representação de cada vetor define a luminosidade, o comprimento de onda dominante, e a saturação ou grau de pureza da matriz (SCURI, 2002).

Em uma câmera digital a imagem é armazenada no quadro de vídeo e reproduz a imagem digital em dois tipos de captura a *Snap Short* e a *Grab Short*.

*Snap Short* adquire uma única imagem em um buffer de memória, neste modo obtemos apenas um único quadro, gerando uma foto.

*Grab Short* é um processo contínuo de aquisição e de alta velocidade das imagens, neste modo obteremos um vídeo (GOMEZ; HERNANDEZ, 2011).

## 2.5 REDES NEURAIIS ARTIFICIAIS

Redes Neurais Artificiais são um conjunto de modelos de processamento de dados baseado em equações matemática, dividido em diferentes camadas onde cada camada pode ter diferentes canais de comunicações entre cada uma delas, onde cada uma das combinações pode gerar resultados diferente ou afunilar vários dados de entradas em poucos resultados (MIYAZAKI,2017).

A rede neural artificial pode ser dividida basicamente em três partes:

1. Camada de entrada de dados
2. Camada de processamento
3. Camada de saída de resultados

O ponto forte das redes neurais é a capacidade de aprendizado. O processo de aprendizado é realizado através de um treinamento onde os dados são apresentados a rede e recebem uma tratativa diferente dependendo da influência que ele terá no resultado (MIYAZAKI,2017).

### 2.5.1 APRENDIZADO PROFUNDO

Aprendizado profundo (Deep Learning) se refere a introdução de camadas ocultas dentro da rede neural artificial e métodos de treinamentos mais eficientes visando diminuir as falhas e aumentar a velocidade.

Essa ferramenta possibilita o processamento de um extenso banco de dados e inúmeros dados de entrada (MIYAZAKI,2017).

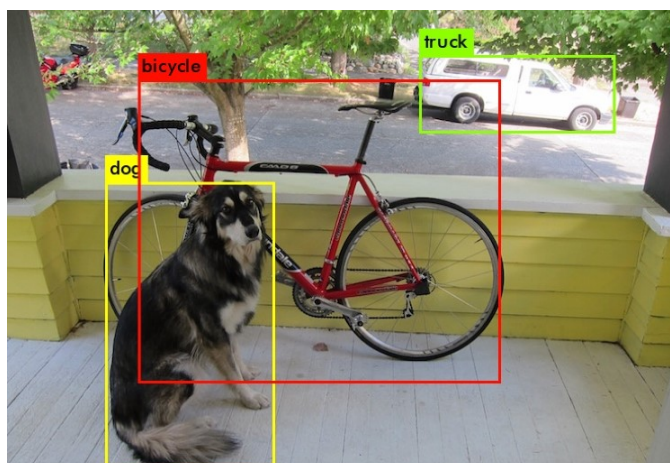
### 2.5.2 BIBLIOTECA OPENCV

*Open CV Library* (Open Source Computer Vision Library) é uma biblioteca de algoritmos de processamento visual e aprendizado de máquina, *machine learning*. A biblioteca é *open source* o que significa que é de livre uso e implementação da comunidade tecnológica, onde pessoas de diferentes formações e áreas do conhecimento contribuem para o desenvolvimento do conhecimento e livre divulgação. OpenCV tem como foco otimizar e aprimorar o processamento visual, o inclui imagens, moldes 3D, áudios (REDMON; FARHADI,2016).

### 2.5.3 ARQUITETURA DE REDE NEURAL

A arquitetura "Você olha só uma vez" (YOLO), ou seja, que utiliza apenas uma vez a rede neural na imagem completa, em comparação com outras arquiteturas de classificação de objetos de uma imagem, é a que mostra melhores resultados, processando imagens em até 30 quadros por segundo em *Grab Short*, aplicando o modelo de imagem em vários ângulos e posições diferentes, independente da escala desse objeto, e também marcando os objetos identificados com caixas delimitadoras de acordo com a classificação da classe dos elementos encontrados (REDMON; FARHADI,2016) conforme a Figura 7.

**Figura 7** – Representação da detecção de objetos com a arquitetura YOLO.



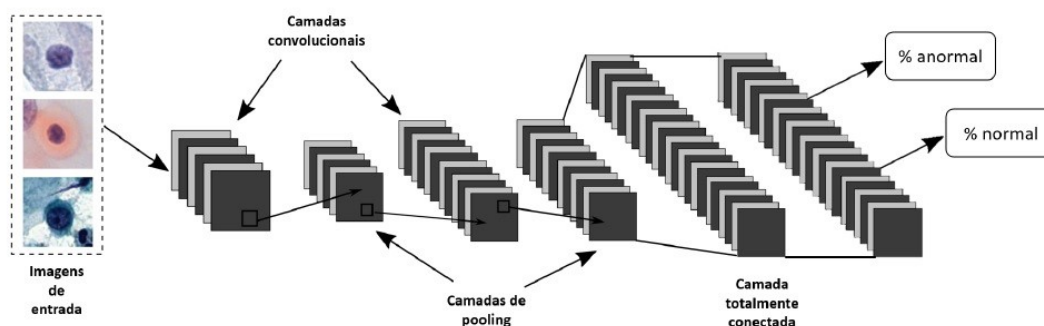
Fonte:[Extraída de (<https://pjreddie.com/darknet/yolo/>) Acessado em 28/09/2018].

### 2.5.4 REDE NEURAL CONVOLUCIONAL

Rede Neural Convolutacional (Convolutional Neural Network – CNN) é uma arquitetura de aprendizado profundo das redes neurais, que foca em características de cada amostra relevando variações de escala, posição ou rotação, tornando a identificação de objetos mais robusta (MIYAZAKI, 2017).

A imagem é dividida em camadas por meio de filtros sucessivos, onde cada filtro vai cada filtro é ajustado para expor características dos moldes como bordas, diferenças no tons das cores (MIYAZAKI,2017), representado pela Figura 8.

**Figura 8** – Arquitetura de uma Rede Neural Convolutiva.



Fonte:[Extraída de (MIYAZAKI,2017)].

As camadas de *pooling* são geradas para simplificar as camadas convolucionais, tornando menos os parâmetros a serem examinados em cada camada (MIYAZAKI,2017).

As camadas totalmente conectadas são as últimas camadas, onde elas são ligadas juntas para formar o objeto que será detectado (MIYAZAKI,2017).

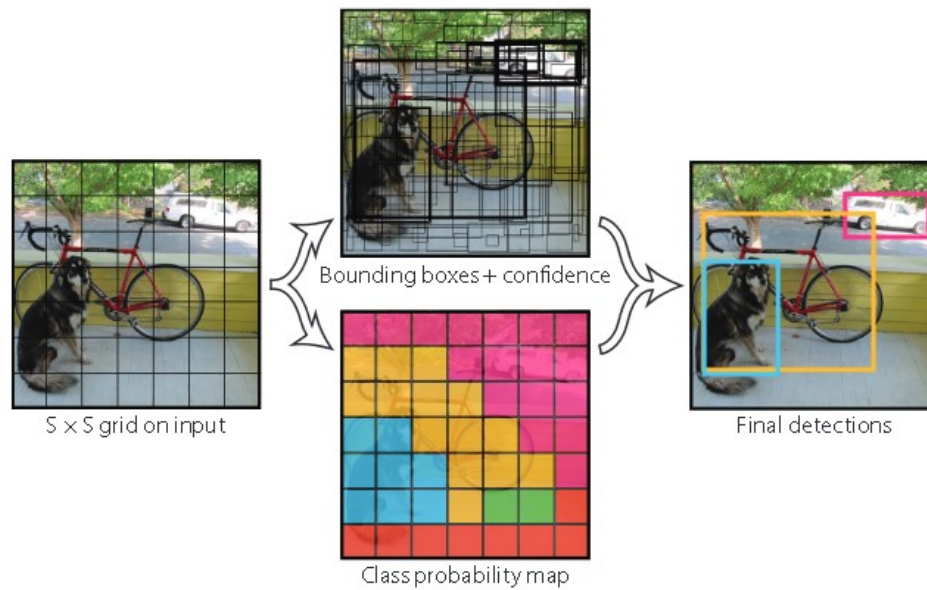
O resultado será classificado em uma porcentagem de normal e uma porcentagem de anormal, nessa etapa o objeto detectado é classificado de acordo com o índice dos moldes já treinados, quanto maior a taxa de normalidade, maior a confiança que a rede neural tem na detecção, em contra partida, quanto maior a taxa de anormalidade menor a confiança da rede na detecção (MIYAZAKI,2017), conforme a Figura 8.

### 2.5.5 PRECISÃO DA REDE NEURAL

O desempenho da detecção e classificação de objetos nas redes neurais do tipo *Deep Learning* variam de acordo com a resolução e o número de elementos contido na imagem. Aumentando a resolução e os quadros de detecção diminui a taxa de erro diminuindo junto a velocidade do algoritmo (REDMON; FARHADI,2016).

(REDMON; FARHADI,2016) destacam que o YOLO mantém uma precisão média por dividir a imagem que entra em várias grades de células, que são as responsáveis de identificar o objeto, conforme a Figura 9.

**Figura 9** – Representação da previsão de quais objetos estão imagem.



**Fonte:**[Extraída de (REDMON; FARHADI, 2016)].

Depois da identificação, as caixas de marcação e de confiança fazem a previsão de qual é o objeto identificado, por dar pontuação para cada célula, onde a somatória dessa pontuação é a base para que o mapa de classe classificar qual é o objeto da imagem (REDMON; FARHADI, 2016).

Como nosso projeto será testado no ambiente externo, os objetos identificados, não sendo semáforos serão classificados como falso positivo, e a falta de identificação de um semáforo vai ser classificado como falso negativo.

Este capítulo procurou destacar os principais pontos dos fenômenos ópticos, sistemas de cores e as ferramentas para o reconhecimento de imagem, iniciando uma breve introdução sobre Redes Neurais Artificiais.



### 3 METODOLOGIA

O desenvolvimento de um sistema de identificação e classificação de objetos em uma imagem consiste em obter uma imagem digital o mais fiel possível da fonte e também ao mesmo tempo ser de fácil processamento para a extração das informações, imagens de rápida atualização e com maiores resoluções requerem um sistema de alto processamento para uma interpretação precisa do algoritmo de imagem, nesse requisito o software deve ser desenvolvido para que a imagem capturada possa ser analisada por um computador de baixo processamento.

Neste capítulo estão descritos os processos para obter a imagem digital e fazer a identificação e classificação dos elementos desejados desses dados.

Para esse projeto foi analisado alguns modelos de câmeras digitais, mas devido a praticidade do manuseio do celular em um veículo automotivo, optamos para o celular Huawei Y6II CAM-L23 possui uma câmera de 13 megapixels que possuímos ilustrado na Figura 10.

**Figura 10** – Celular Huawei Y6 li



**Fonte:**[Extraída de ([https://www.magazineluiza.com.br/huawei-y6-2018-dual-android-8-0-tela-5-7-16gb-camera13mppreto/p/5353485/te/tcsp/?partner\\_id=19805&utm\\_source=google&seller\\_id=pagdepois&product\\_group\\_id=304686703092&ad\\_group\\_id=48543699755&aw\\_viq=pla&gclid=EAlaIQobChMI06\\_mvZmR3wIVkg2RCh1Exg-QEAYYASABEgKgovD\\_BwE](https://www.magazineluiza.com.br/huawei-y6-2018-dual-android-8-0-tela-5-7-16gb-camera13mppreto/p/5353485/te/tcsp/?partner_id=19805&utm_source=google&seller_id=pagdepois&product_group_id=304686703092&ad_group_id=48543699755&aw_viq=pla&gclid=EAlaIQobChMI06_mvZmR3wIVkg2RCh1Exg-QEAYYASABEgKgovD_BwE)) Acessado 02/12/18].

#### 3.1 AMBIENTE DE DESENVOLVIMENTO

Entre todas as linguagens de programação utilizadas para desenvolver um *Software* a mais funcional é o Python, com uma ampla compatibilidade entre outras plataformas, também sua simplicidade na linguagem reduz o número de comandos necessário para se realizar uma tarefa.

A linguagem Python possui compatibilidade com diversas plataformas diferentes e também ferramentas específicas para trabalhar com visão computacional no *Deep*

*Learning* como o OpenCV, e também o Numpy que é uma biblioteca de operações numéricas onde as estruturas de matriz OpenCV são convertidas ou combinadas para e de matrizes Numpy.

Por possuir um buscador e instalador próprio do OpenCV e Numpy, o ambiente de desenvolvimento escolhido foi o Pycharm, oferecendo apenas as atualizações que são compatíveis com o mesmo evitando erros de instalação quando é necessário adicionar uma nova ferramenta.

Após a instalação das bibliotecas, um pequeno teste deve ser realizado para testar se não tem erros executando os comandos abaixo:

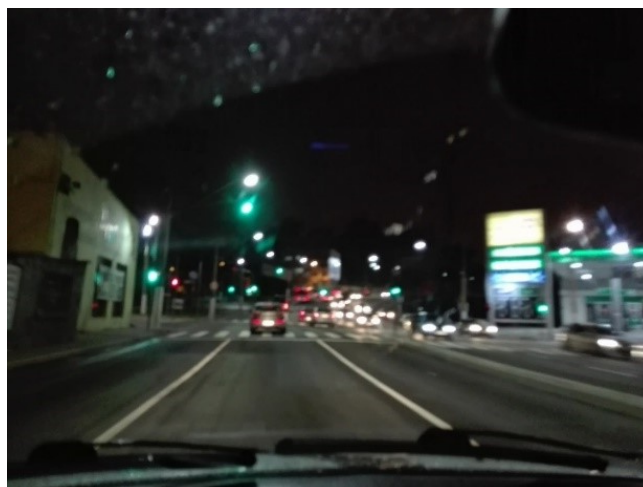
1. `import cv2`
2. `import numpy`

Não retornado nenhum erro, todas as ferramentas necessárias para que o ambiente de desenvolvimento reconheça e analise as imagens obtidas da câmera estão funcionando.

### **3.2 SISTEMA DE DETECÇÃO DO SEMÁFORO DE TRÁFEGO URBANO**

O desenvolvimento de um sistema de assistência ao condutor para a detecção do estado do semáforo de controle do tráfego urbano torna-se um projeto difícil devido a distância e o tamanho do equipamento quando comparado com os outros elementos encontrados no mesmo ambiente, como por exemplo os letreiros luminosos, propagandas com refletores entre outros, conforme foto da Figura 11.

**Figura 11** – Fotografia com alguns elementos do transito que interferem na detecção dos semáforos.



Fonte: Autor.

Os *Softwares* utilizados para a detecção e reconhecimento facial não fornecem um resultado satisfatório para sua adaptação e utilização em transito urbano, por dependerem de um tamanho mínimo do elemento capturado pela câmera e o mesmo não conseguir classificar a imagem durante movimentações repentinas, interferindo diretamente nos quadros de atualização de imagem, o que acontece frequentemente devido as condições acidentadas das vias de tráfego como lombadas, buracos e ressaltos no asfalto como os reparos de recapagem.

A solução mais satisfatória encontrada para atender a velocidade necessária de um sistema de assistência ao condutor é o *Deep Learning*, onde a detecção dos semáforos será realizado por um *Softwares* de rede neural treinado para reconhecer os semáforos distantes do para-brisa em diferentes condições climáticas.

Devido aos avanços das tecnologias das câmeras digitais as taxas de captura de imagem podem ser mais altas do que a frequência de mudança de estado do semáforo, quando isso acontece o semáforo parece estar desligado conforme Figura 12.

**Figura 12** – Fotografia retirada quando o semáforo estava na mudança de estado.



**Fonte:** Autor.

Quando isso acontece não é possível classificar essa imagem como verdadeiro positivo, nem falso positivo, tornando necessário excluir do banco de dados e refazer o trajeto novamente.

A detecção e classificação de objetos em imagens de um modo geral, nesses últimos anos, tem obtidos grandes avanços com melhorias significativas, mas para realizar o reconhecimento de semáforo é necessário realizar essa tarefa de modo mais confiável reduzindo a taxa de erro.

## 4 DESENVOLVIMENTO

A principal motivação para o desenvolvimento de um *software* de identificação e classificação do estado do semáforo, surgiu com a necessidade dos veículos autônomos de trafegarem em trânsito urbano de acordo com a sinalização viária correspondente, em uma parceria com a universidade de Heidelberg, a empresa Bosch desenvolveu e apresentou na conferência internacional de automação e robótica em 2017 um projeto baseado na arquitetura YOLO, utilizado como modelo para o desenvolvimento desse projeto.

Para esse projeto ficou estabelecido que o *software* possuiria várias etapas onde cada uma delas teria um aprimoramento com novas funções e desempenho, portanto nesse trabalho será apresentado a primeira versão do projeto, com as condições mínimas de identificação e classificação do estado do semáforo.

### 4.1 DESCRIÇÃO DO ALGORITMO

Primeiramente foi elaborado os Fluxogramas do projeto com o escopo e as funções principais descritos no APÊNDICE - FLUXOGRAMAS DO PROJETO

Em seguida a descrição dos principais comandos utilizados na programação do projeto:

A obtenção da amostra pode ser realizada de duas maneiras, em tempo real ou de um arquivo. Utilizando o seguinte comando será em tempo real:

```
camera = cv2.VideoCapture(0)
```

Onde a imagem é extraída automaticamente de uma câmera conectada ao computador com *delay* valor zero e atribuída a uma variável, *frame* por *frame*.

Para realizamos os testes em bancada optamos por retirar amostras pré-definidas de um arquivo com o seguinte comando:

```
original = cv2.imread('C:\\Users\\Suporte\\Desktop\\fotos\\teste27.jpg')
```

O comando acima lê um arquivo de imagem, podendo ser de diferentes tipos de arquivo de imagem, do endereço no banco de dados.

```
camera = cv2.resize(original, (416,416), cv2.INTER_LINEAR)
```

Este comando redefine o tamanho da imagem em pixel, neste caso 416x416, e o INTER\_LINEAR é opção para fazer a interpolação linear.

Os comandos que se seguem logo abaixo são carregados a uma variável classe os nomes de cada item que será inserido dentro da rede neural, onde serão separados e classificados por nome.

```
1. classes=None
2. with open('C:\\Users\\Suporte\\PycharmProjects\\reconhecimento2\\catkin_create_pkg yolo_light\\texto1.txt', 'r') as f:
3.     classes = [line.strip() for line in f.readlines()]
```

O arquivo com extensão txt contém o identificador com os nomes dos objetos, com base na quantidade de nomes na lista será criada um banco de dados do mesmo tamanho, se houver divergência entre o número de nomes e o número de objetos introduzidos no pré treino o sistema retornará um erro.

Os nomes de cada objeto serão usados posteriormente para realizar a análise de cada identificação. O segundo argumento do *with open* é o modo de leitura que o *Python* utilizará, 'r' significa que o arquivo deve ser lido sem nenhuma modificação. Cada linha será lida e atribuída a variável classe.

```
COLORS = np.random.uniform(0, 255, size=(len(classes), 3))
```

Utilizando este método para realiza um número aleatório no range 0 a 255, segundo a escala RGB de cores, para cada retorno da função **len** que o número da posição de cada objeto dentro da variável classes, onde o salto é a soma da posição da classe +3.

```
net=cv2.dnn.readNet('C:\\Users\\Suporte\\Documents\\Python Scripts\\yolov3.weights', 'C:\\Users\\Suporte\\PycharmProjects\\reconhecimento2\\catkin_create_pkg yolo_light\\yolo1.cfg')
```

Nas redes neurais, os parâmetros são atribuídos por dois arquivos, o *yolov3.weights* e o *yolo1.cfg*.

A função *yolov3.weights* é onde se obtém um modelo definido como treino onde ele retorna uma flag quando o sistema estiver livre para uma nova tarefa. Os pesos são os arquivos que contém as dimensões de cada modelo, eles são agrupados em *layers*, onde cada *layer* posteriormente terá uma classe atribuída. Os modelos são imagens onde são carregadas de algum determinado objeto, onde é retirada a altura pela largura dos pixels em uma escala de cinza.

O arquivo *yolo1.cfg* contém a arquitetura da rede para depois realizar a leitura dos pesos, nesse trabalho não será abrangido a criação dos modelos e configuração da

rede neural, será focado no sistema de reconhecimento de imagem no método Yolo. Em seguida temos a estrutura de repetição *while(true)* que consiste em colocar a aquisição e classificação de imagem em execução sem interrupção conforme o comando abaixo:

```
1. while True:
2.     ret,image = camera.read()
3.     Width=image.shape[1]
4.     Height=image.shape[0]
5.     scale = 0.00392
```

Para otimizar nossos testes em bancada, está função será desativada, mas mantida para futuras atualizações do *software* e testes em campo.

```
blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True,crop=False)
```

As variáveis de entrada são as imagens, a escala que será aplicada, o tamanho original da imagem em pixels, os valores em RGB para aplicação na imagem, sinal indicando se a imagem será dividida em diferentes camadas visando facilitar a compreensão da imagem, ultima indicação se alguma parte cortada da imagem deve ser salva.

```
draw_bounding_box(image, class_ids[i], confidences[i], round(x), round(y), round(x+w), round(y+h))
```

Esta função tem como objetivo desenhar as caixas e o nome da identificação dos modelos e das cores.

Apenas é inicializada se a variável classes receber a identificação “traffic light” da rede neural.

```
if classes[class_id] == "traffic light":
```

As coordenadas do modelo identificado são divididas em três partes na vertical.

```
1. h0 = int(y_plus_h/3)
2. h1 = h0 + int(y_plus_h/3)
3. h2 = y_plus_h
4.
5. vermelho = np.sum(img[x:x_plus_w, y:h0])
6. amarelo = np.sum(img[x:x_plus_w, h0:h1])
7. verde = np.sum(img[x:x_plus_w, h1:h2])
```

A função sum faz o cálculo de divisão da matriz da imagem que sendo processada na rede neural, onde a imagem é dividida em três partes, seguindo a ordem vertical do semáforo, e cada uma das partes é alocada em uma variável.

```
1. cores = [vermelho, amarelo, verde]
2. cor_final = max(cores)
```

Visto que a imagem recebeu um filtro de cinza, as três variáveis são organizadas e depois comparadas e a que tiver o maior valor de pixel indicará em qual estado está o semáforo.

Após uma sequência de if else, as caixas de identificação são desenhadas e a cor é escrita na imagem.

```
1. cv2.rectangle(img, (x, y), (x_plus_w, y_plus_h), color, 2)
2. cv2.putText(img, label, (x - 10, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
```

Função para alocação dos objetos identificados em caixas e suas coordenadas.

```
indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)
```

Nesse capítulo foi descrito como foi realizado a utilização das ferramentas para extrair as características necessárias do objeto detectado e classificá-lo por meio de equações dos valores do pixel para se obter a intensidade da luz, e a profundidade dos objetos encontrados na imagem.



## 5 RESULTADOS E DISCUSSÃO

O sistema de identificação assistida do controle de tráfego veicular urbano desenvolvido nesse projeto constitui em um *software* que através de uma câmera ou um arquivo de foto seja possível a identificação do estado do semáforo auxiliando os condutores com deficiência na percepção das cores e também os condutores que mesmo sem nenhuma deficiência podem perder essa percepção devido a exposição de outros elementos encontrados nas vias que não tem a finalidade de sinalizar e manter o fluxo do trânsito.

Nesse capítulo é apresentado os resultados obtidos realizando um comparativo entre o realizado e o ideal, analisando os fatores de erros e acertos, também verificando se a metodologia desse projeto atende as necessidades exigidas para auxiliar os condutores de veículos automotivos.

### 5.1 TESTES REALIZADO

Como mencionado no capítulo três o *software* foi desenvolvido no compilador *pycharm* alterando as fotos de teste na linha de comando abaixo:

1. `original=cv2.imread('C:\\Users\\Suporte\\Documents\\PythonScripts\\positivo2.jpg')`
2. `camera = cv2.resize(original, (416,416), cv2.INTER_LINEAR)`

Selecionado a foto e apertando as teclas Alt+Shift+F10 o compilador executará o programa retornando a imagem com o resultado conforme a Figura 13.

**Figura 13** – Foto com o semáforo e seu estado identificado e classificado.



**Fonte:** Autor.

Na Figura 13 é ilustrado quando o sistema detecta o semáforo, os títulos das caixas delimitadoras variam de acordo com o estado em que o semáforo se encontra.

## 5.2 RESULTADOS OBTIDOS

Após realizar o teste descrito no item anterior com 206 fotos obtemos as informações descrita na Tabela 1.

**Tabela 1** – Análise das amostras em bancada.

	total	%	Fotos dia	Fotos noite sem chuva	Fotos noite com chuva	Fotos com mais elementos	Fotos com poucos elementos
Positivo 100%	33	16,01	27	2	7	13	20
cor errada	66	32,03	42	13	10	43	23
negativo	26	12,62	13	2	11		
Falso positivo	17	8,25	17	0	0	10	7
Falso negativo	64	31,06	27	10	26	47	17
total	206	99,97	126	27	54	113	67
%	100		61,16	13,1	26,21	52,42	32,52

**Fonte:** Autor.

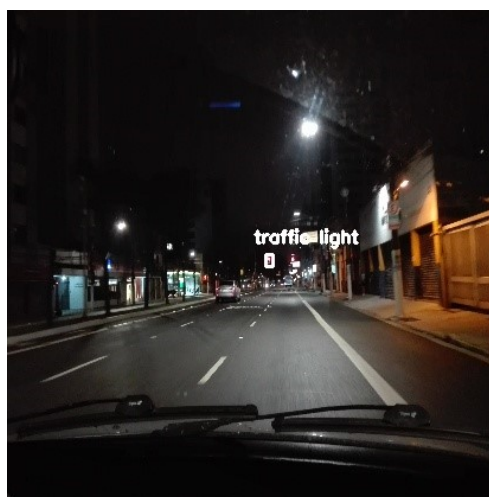
Como as fotos foram tiradas no mesmo trajeto de São Paulo a Santo André durante vários dias e períodos diferentes, obtivemos os mesmos semáforos em ângulos e luminosidades diferentes, e com esses dados, observamos que a eficiência da detecção ficou em 16,01% das 206 fotos analisadas devido a seguintes problemas encontrados:

- a) Distorções da câmera de baixa resolução e foco;
- b) Falta de identificação na cor vermelha;
- c) Para brisa do veículo com resíduos pós chuva;
- d) Para brisa com o reflexo painel do veículo devido a iluminação noturna e o ângulo errado da câmera;
- e) Baixa resolução de imagem processada pela rede neural;
- f) Excesso de elementos que não fazem parte da sinalização de trânsito.

### 5.3 ANÁLISE DOS RESULTADOS

Os resultados desse projeto foram bem satisfatórios quanto a escolha da metodologia que foi usado em nossos experimentos reais aonde testamos o sistema em condições que os condutores de veículos encontram diariamente, em dias e noites com ou sem chuva, com baixa ou excesso de iluminação externa, segundo o quadro 4 na linha 2, o maior erro encontrado, com 32% foi a falta de classificação das cores, onde o algoritmo consegue identificar um semáforo com uma distância considerável sem classificar a cor, conforme a Figura 14.

**Figura 14** – Foto onde o algoritmo detecta o semáforo sem a cor.

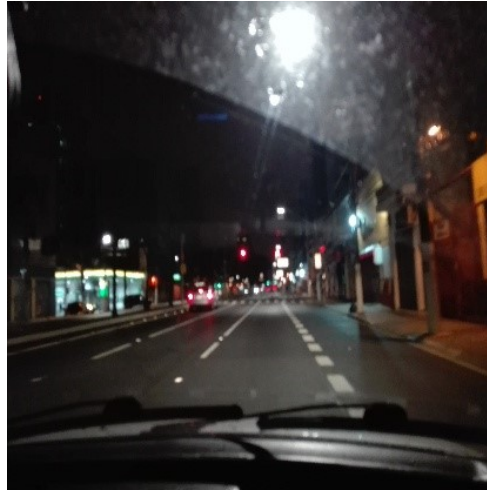


**Fonte:** Autor.

Outro problema encontrado foi o excesso de falso negativo, com 31%, onde tem o semáforo mais não é detectado, na nossa opinião isso aconteceu pela escolha da câmera, onde usamos uma que tínhamos a disposição, com um foco insatisfatório para um veículo em movimento, também erramos o posicionamento da câmera que permitiu que o reflexo do painel aparecesse em algumas amostras de noite, como

aconteceu quando chegamos mais perto do semáforo, conforme a Figura 15.

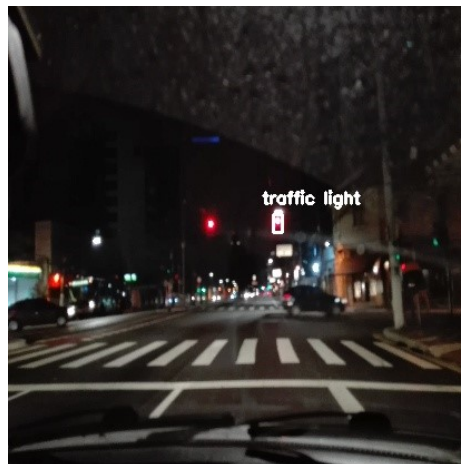
**Figura 15**– Foto com falta de foco e reflexo do painel.



**Fonte:** Autor

Quando chegamos mais perto ainda do semáforo acima o algoritmo voltou a detectar novamente, mas não totalmente conforme pode ser visto na Figura 16.

**Figura 16** – Foto com a identificação parcial do semáforo.



**Fonte:** Autor.

Neste capítulo foi descrito como foi realizado os nossos testes e nossa análise das dificuldades e problemas encontrados. No próximo capítulo será descrito as conclusões do nosso projeto e as propostas futuras.

## 6 CONSIDERAÇÕES FINAIS

Atualmente temos vistos os avanços do carro autônomo bem como o surgimento de novas tecnologias para essa finalidade e um crescimento muito significativo de novas pesquisas nesse seguimento. O fato é que esses novos conceitos podem ajudar os condutores de veículos automotores na identificação da sinalização, ainda mais os portadores de alguma deficiência na percepção das cores que é o foco desse projeto.

Apesar de uma baixa taxa de acerto para um sistema que tem como objetivo auxiliar os portadores de deficiência, o Sistema de identificação assistida do controle de tráfego veicular urbano, dentro da proposta inicial, teve uma contribuição para o desenvolvimento de novos sistemas de identificação e classificação de objetos com *Deep Learning*, usando a arquitetura YOLO, e uma nova proposta de solução de baixo custo mais abrangente para a acessibilidade de todas as pessoas que tem necessidades especiais.

### 6.1 PROPOSTAS FUTURAS

Durante o desenvolvimento desse projeto foram observados a necessidade de algumas melhorias no projeto.

Como sugestão para trabalhos futuros:

- a) Fazer um estudo e testes com mais variações de configurações e conjuntos de dados de pré-treino, como por exemplo o YOLO9K e o TinyYOLO;
- b) Realizar o treinamento da rede neural com mais modelos de semáforos e variações de região e de tempo;
- c) Aprimorar o software para trazer mais funções, como por exemplo aviso sonoro;
- d) Aumentar a taxa de acertos e confiabilidade do sistema;
- e) Melhorar a velocidade da aquisição de imagens em tempo real do software;
- f) Realizar testes de aquisição de imagens em tempo real com o sistema em um veículo automotivo;
- g) Desenvolver modelos de pós-treino;
- h) Desenvolver novas configurações do framework contidas no arquivo cfg.
- i) Fazer um conjugado com GPS (*Maps, Waze*) para localizar os semáforos, e depois usar a rede neural para identificar o estado.

## REFERÊNCIAS BIBLIOGRÁFICAS

Bosch Small Traffic Lights Dataset disponível em <http://k0b.de/bstld> - Acessado em 21/10/2018.

BRUNI, I. F.; CRUZ, A. A. V. sentido cromático: tipos de defeitos e testes de avaliação clínica. Arq. Bras. Oftalmol., São Paulo, v. 69, n. 5, p. 766-775, 2006.

GOMEZ R.P.; HERNANDEZ G.A. Digital Image Processing Using LabView, 2011.

LEMT-Laboratório de Estudos Metodológicos em Tráfego e Transportes. Disponível em [http://www.ptr.poli.usp.br/index.php?option=com\\_content&view=article&id=9&Itemid=49](http://www.ptr.poli.usp.br/index.php?option=com_content&view=article&id=9&Itemid=49) -Acessado em 05/09/2012.

MIYAZAKI C.K. Redes Neurais Convolucionais para Aprendizagem e reconhecimento de objetos 3D, 2017

NETO L. Q. – Disponível em <http://g1.globo.com/Noticias/Carros/0,,MUL1527656-9658,00FALTA+DE+PADRONIZACAO+DAS+SINALIZACOES+PREJUDICA+DALTONICOS+NO+TRANSITO.html>- Acessado em 01/09/2013.

Os Fundamentos da Física 2; Ramalho Nicolau Toledo; Editora Moderna, 2009.

PEDROSA, I. Da Cor à Cor Inexistente. Rio de Janeiro: Editora Universidade de Brasília, 1982.

PIETRANTONIO H. – Análise da Relação entre a Observância aos Padrões de Projeto de Sinalização e a Ocorrência de Acidentes de Trânsito em Rodovias, 2005.

RAL, W. D.; THOMAZ, C. E. Normalização espacial de imagens frontais de face. Centro Universitário da FEI - Departamento de Engenharia Elétrica. SBC, p. p. 13. 2008.

REDMON J.; FARHAD A. YOLOv3: An Incremental Improvement, 2016

RENTERÍA A. R. - CONTROLE DE SEMÁFOROS POR LÓGICA FUZZY E REDES NEURAI, 2002.

ROCHA J. C.. Cor luz, Cor Pigmento e os sistemas RGB e CMY, 2010.

SCURI A. E. Fundamentos da imagem digital, 2002

SOARES, R. Avaliação dos Condutores Portadores de Discromatopsia Congênita na Percepção Cromática da Sinalização Viária, 2009.

<https://www.infowester.com/imagens.php> - Acessado em 12/05/2013.

<http://labculturaviva.org/pontobrasil/materialdidatico/imagemeletronica.pdf>- acessado em 3/12/2013.

<https://processing.org/tutorials/color/> - Acessado em 26/07/18

[https://opencvpythontutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_setup/py\\_setup\\_in\\_windows/py\\_setup\\_in\\_windows.html](https://opencvpythontutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_setup_in_windows/py_setup_in_windows.html) - Acessado em 11/08/2018.

<https://pjreddie.com/darknet/yolo/> - Acessado em 28/09/2018.

<https://www.arunponnusamy.com/yolo-object-detection-opencv-python.html>-  
Acessado em 28/09/2018.

<https://medium.com/@madhawavidanapathirana/not-just-another-yolo-v3-for-python-79da6c3af082>- Acessado em 28/09/2018.

<https://www.iotforall.com/objects-recognition-live-stream-yolo-model/>- Acessado em 28/09/2018.

<https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006> - Acessado em 10/10/2018.

# APÊNDICE

## CÓDIGO FONTE COMPLETO

```

1. # import required packages
2. import cv2
3. import argparse
4. import numpy as np
5. #camera = cv2.VideoCapture(0)
6. original = cv2.imread('C:\\Users\\Suporte\\Documents\\Python Scripts\\positivo2.jpg')
7. camera = cv2.resize(original, (416,416), cv2.INTER_LINEAR)
8. # read class names from text file
9. classes = None
10. with open('C:\\Users\\Suporte\\PycharmProjects\\reconhecimento2\\catkin_create_pkg
    yolo_light\\texto1.txt', 'r') as f:
11.     classes = [line.strip() for line in f.readlines()]
12. # generate different colors for different classes
13. COLORS = np.random.uniform(0, 255, size=(len(classes), 3))
14. # read pre-trained model and config file
15. net = cv2.dnn.readNet('C:\\Users\\Suporte\\Documents\\Python
    Scripts\\yolov3.weights', 'C:\\Users\\Suporte\\PycharmProjects\\reconhecimento2\\catkin_creat
    e_pkg yolo_light\\yolo1.cfg')
16. #while True:
17.     # read input image
18.     #ret,image = camera.read()
19. image = camera
20.
21. Width = image.shape[1]
22. Height = image.shape[0]
23. scale = 0.00392
24. # create input blob
25. blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True, crop=False)
26. # set input blob for the network
27. net.setInput(blob)
28. # function to get the output layer names
29. # in the architecture
30. def get_output_layers(net):
31.     layer_names = net.getLayerNames()
32.     output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
33.     return output_layers
34. # function to draw bounding box on the detected object with class name
35. def draw_bounding_box(img, class_id, confidence, x, y, x_plus_w, y_plus_h):
36.     if classes[class_id] == "traffic light":
37.         #teste = img[x:x_plus_w, y:y_plus_h ]
38.         h0 = int(y_plus_h/3)
39.         h1 = h0 + int(y_plus_h/3)
40.         h2 = y_plus_h
41.         vermelho = np.sum(img[x:x_plus_w, y:h0])
42.         amarelo = np.sum(img[x:x_plus_w, h0:h1])
43.         verde = np.sum(img[x:x_plus_w, h1:h2])
44.         cores = [vermelho, amarelo, verde]
45.         cor_final = max(cores)
46.         if cor_final == vermelho:
47.             print("Vermelho")
48.         elif cor_final == amarelo:
49.             print("Amarelo")
50.         else:
51.             print("Verde")
52.         label = str(classes[class_id])
53.         color = COLORS[class_id]
54.         cv2.rectangle(img, (x,y), (x_plus_w,y_plus_h), color, 2)
55.         cv2.putText(img, label, (x-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
56.     # run inference through the network

```



```

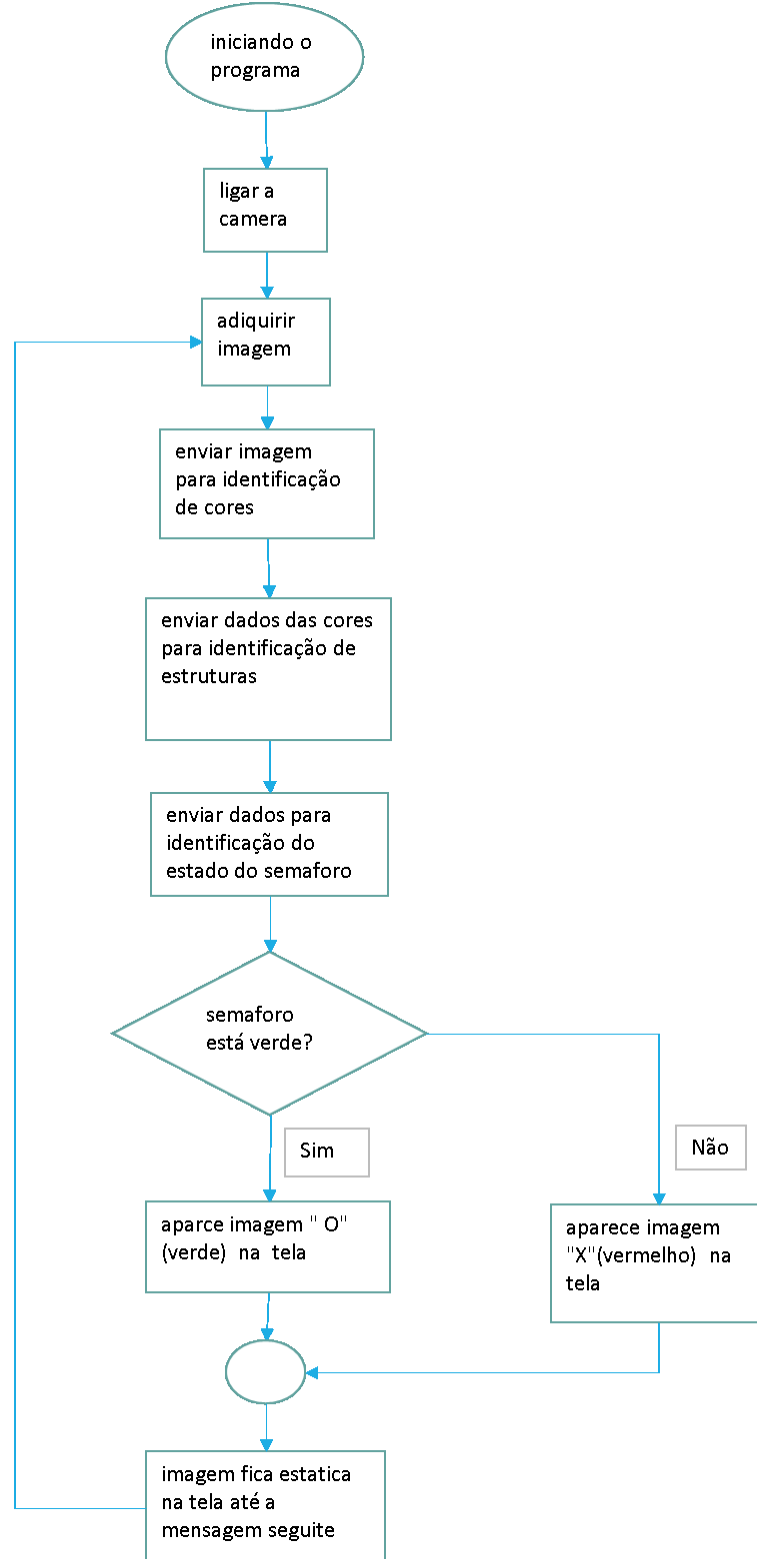
57.     # and gather predictions from output layers
58. outs = net.forward(get_output_layers(net))
59. # initialization
60. class_ids = []
61. confidences = []
62. boxes = []
63. #conf_threshold = 0.5
64. conf_threshold = 0.5
65. #nms_threshold = 0.4
66. nms_threshold = 0.4
67. # for each detetion from each output layer
68. # get the confidence, class id, bounding box params
69. # and ignore weak detections (confidence < 0.5)
70. for out in outs:
71.     for detection in out:
72.         scores = detection[5:]
73.         class_id = np.argmax(scores)
74.         confidence = scores[class_id]
75.         if confidence > 0.5:
76.             center_x = int(detection[0] * Width)
77.             center_y = int(detection[1] * Height)
78.             w = int(detection[2] * Width)
79.             h = int(detection[3] * Height)
80.             x = center_x - w / 2
81.             y = center_y - h / 2
82.             class_ids.append(class_id)
83.             confidences.append(float(confidence))
84.             boxes.append([x, y, w, h])
85. # apply non-max suppression
86. indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)
87. # go through the detections remaining
88. # after nms and draw bounding box
89. for i in indices:
90.     i = i[0]
91.     box = boxes[i]
92.     x = box[0]
93.     y = box[1]
94.     w = box[2]
95.     h = box[3]
96.
97.     draw_bounding_box(image, class_ids[i], confidences[i], round(x), round(y), round(x+w), round
98. (y+h))
99. # display output image
100. cv2.imshow("object detection", image)
101. #if cv2.waitKey(1) & 0xFF == ord('q'):
102.     #break
103.     cv2.waitKey(0)
104.     # save output image to disk
105.     # cv2.imwrite("object-detection.jpg", image)
106.     # release resources
107.     cv2.destroyAllWindows()

```

## FLUXOGRAMAS DO PROJETO

Planejamento do Fluxograma do escopo do projeto, conforme a Figura 17.

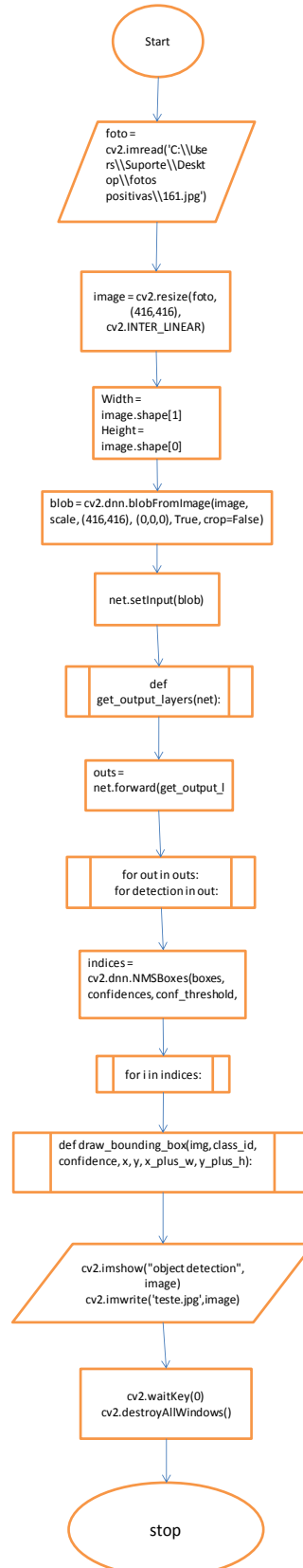
**Figura 17** – Fluxograma do projeto



Fonte: Autor.

Durante o desenvolvimento do software foi definidas funções específicas para cada processo descrito na sequência, podemos observar na Figura 18 a função principal.

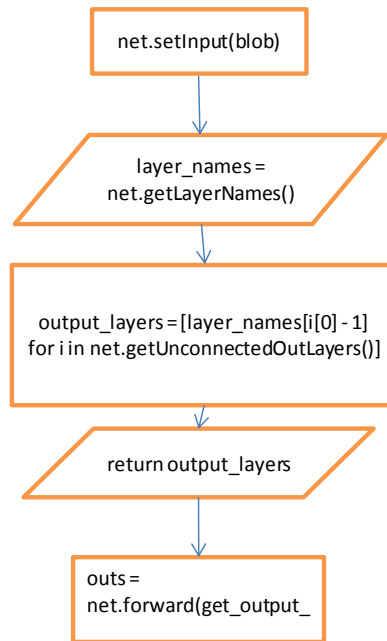
**Figura 18** – Fluxograma do funcionamento da função principal.



Fonte: Autor.

Na arquitetura YOLO as camadas são responsáveis por extrair as características da imagem de entrada, na Figura 19 demonstra a função de camadas.

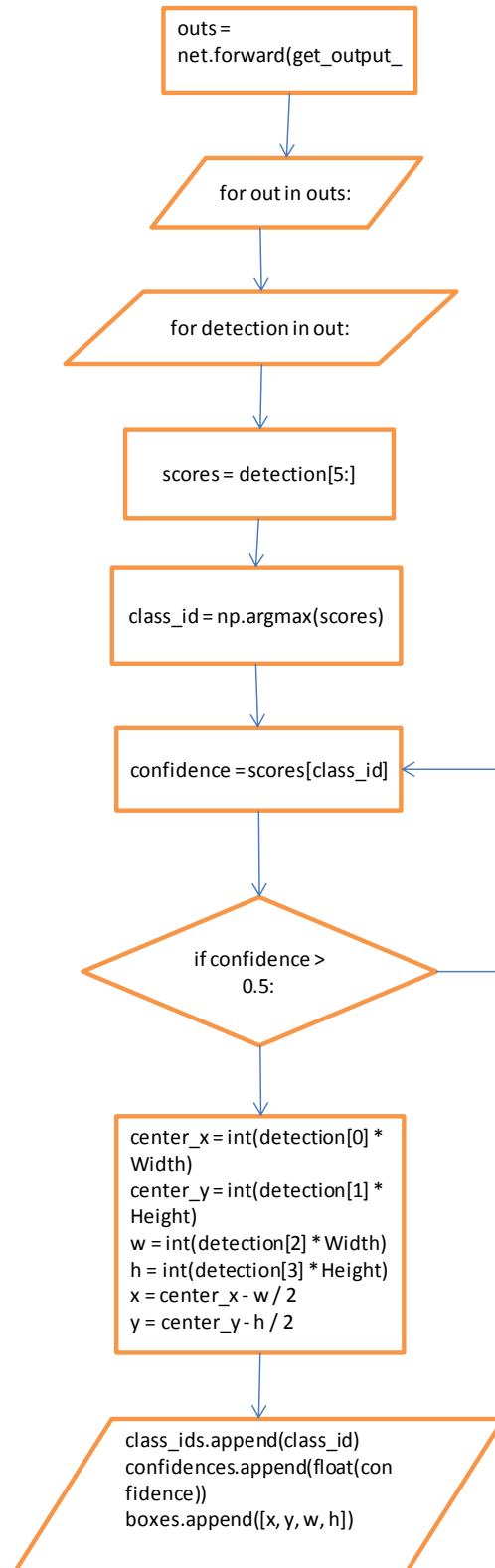
**Figura 19** – Fluxograma da função que identificação das camadas.



**Fonte:** Autor.

Depois uma função de filtro que ativa na presença de características relevantes, como as bordas do objeto e as cores, conforme a Figura 20.

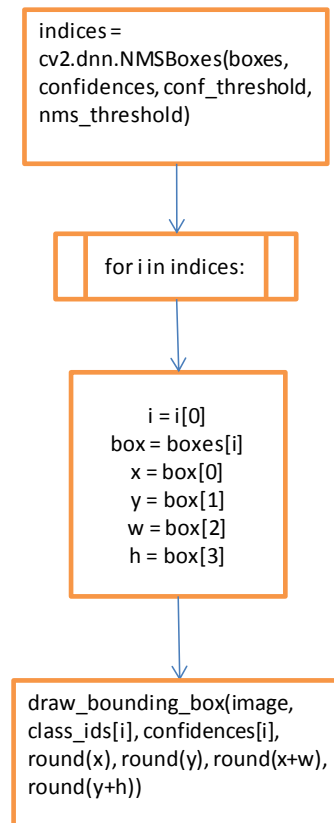
**Figura 20** – Função de validação do objeto detectado da imagem.



**Fonte:** Autor.

Quando são detectados mais de um objeto na mesma imagem independentemente da distância e do tamanho desses objetos, a função identifica simultaneamente conforme a Figura 21.

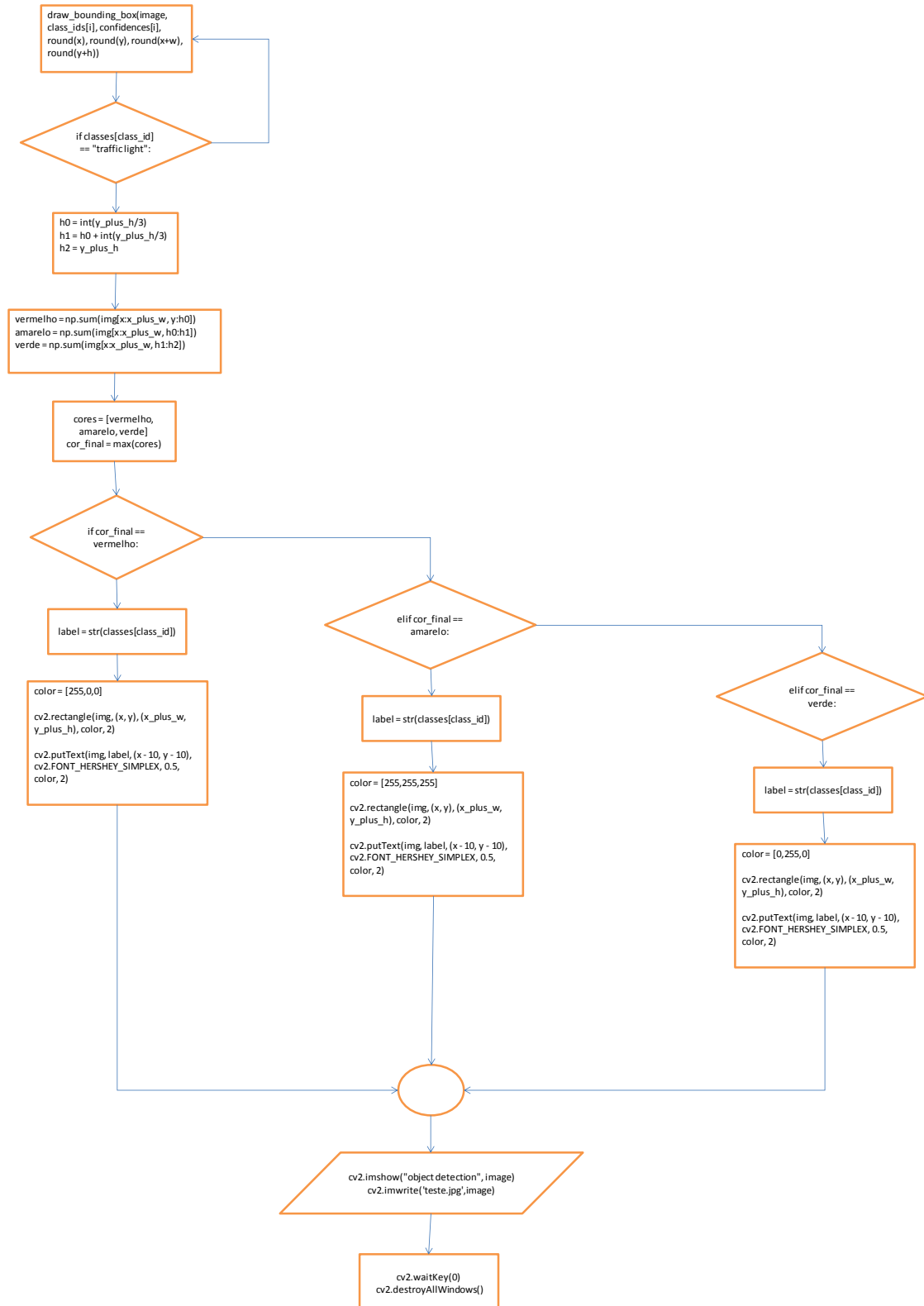
**Figura 21** - Função de classificação de múltiplas detecções.



**Fonte:** Autor.

Depois de percorrer as funções anteriores a função de saída marca com *box* os objetos, e os nomeia com o estado do semáforo, conforme a Figura 22.

**Figura 22** - Função delimitadora dos objetos identificado e informação do estado do semáforo.



Fonte: Autor.