CENTRO UNIVERSITÁRIO DO ESTADO DO PARÁ
CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

Raul Vitor Mesquita da Silva
Sérgio Teixeira Corrêa Filho
Vitor Henrique Morais Brasil

OTIMIZAÇÃO DOS TEMPOS DE UM SEMÁFORO POR PROCESSAMENTO DE IMAGEM E CONTROLADORES.

Belém
2019

Raul Vitor Mesquita da Silva
Sérgio Teixeira Corrêa Filho
Vitor Henrique Morais Brasil

OTIMIZAÇÃO DOS TEMPOS DE UM SEMÁFORO POR PROCESSAMENTO DE IMAGEM E CONTROLADORES.

Trabalho de conclusão de curso de graduação apresentado ao Centro Universitário do Estado do Pará como requisito parcial para a obtenção do título de graduação em Engenharia da Computação.

Orientador (a): Prof. MSc. Polyana Fonseca

Belém
2019

Dados Internacionais de Catalogação-na-publicação (CIP)
Biblioteca do Cesupa, Belém – PA

Silva, Raul Vitor Mesquita da.

Otimização dos tempos de um semáforo por processamento de imagem e controladores / Raul Vitor Mesquita da Silva, Sérgio Teixeira Corrêa Filho, Vitor Henrique Morais Brasil; orientador Polyana Santos Fonseca Nascimento. – 2019.

Trabalho de Conclusão de Curso (Graduação) – Centro Universitário do Estado do Pará, Engenharia de Computação, Belém, 2019.

1. Processamento de imagem. 2. Sistemas especialistas. 3. Controladores. I. Corrêa Filho, Sérgio Teixeira. II. Brasil, Vitor Henrique Morais. III. Nascimento, Polyana Santos Fonseca, *orient*. IV. Título.

CDD 23ª ed. 006.6

Raul Vitor Mesquita da Silva
Sérgio Teixeira Corrêa Filho
Vitor Henrique Morais Brasil

OTIMIZAÇÃO DOS TEMPOS DE UM SEMÁFORO POR PROCESSAMENTO DE IMAGEM E CONTROLADORES.

Trabalho de conclusão de curso de graduação apresentado ao Centro Universitário do Estado do Pará como requisito parcial para a obtenção do título de Bacharel (a) em Engenharia da Computação.

Aprovado em: ____ de _____ de _____.

BANCA EXAMINADORA

_____

Itamar Jorge Vilhena De Brito- CESUPA

_____

Marcelo Pinto da Costa Mendes - CESUPA

_____

Polyana Santos Fonseca Nascimento - CESUPA (orientadora)

AGRADECIMENTOS

A deus, devemos entregar todos nossos méritos, conquistas e sonhos, mas dedico todos estes ao senhor Edilberto Palheta Brasil, que com toda certeza olha por nós todos os dias. Que foi um grande homem em vida e uma inspiração por diversas vezes.

Ser engenheiro é um sonho que se iniciou com a admiração ao grande conhecimento do senhor Claudio Elson Santos de Morais, que por muitas vezes me inspirou a escolher a profissão que em alguns meses se concretizará.

Não menos importante, gostaria de agradecer aos meus pais, Gleison Luiz da Silva Brasil e Claudiana Nazaré Morais Brasil, e a minha avó, Nester da Silvar Brasil, por serem meu pilar de sustentação, terem me criado e feito de mim o homem que sou hoje.

Sem citar nomes, um agradecimento especial a todos que estiveram presentes nesta trajetória, que de forma direta ou indireta deram sua contribuição para a conclusão deste trabalho.

Vitor Henrique Morais Brasil

AGRADECIMENTOS

À Deus, por ter me dado força, coragem e determinação para superar todas as dificuldades que surgiram durante esta jornada acadêmica.

Agradeço aos meus pais, Sérgio Teixeira Corrêa e Elcicarla Silva dos Santo Corrêa, assim como todos meus familiares, por todo o amor, incentivo e apoio que me foi dado.

Agradeço a minha avó Auristela Silva dos Santos, por todo o carinho, suporte e incentivo durante a minha carreira acadêmica.

Agradeço a minha namorada, Ítala Campos Nery, por todo o amor, apoio e colaboração durante este trabalho e por compreender meus momentos de ausência.

À minha orientadora Polyana Fonseca, pelo suporte, correções e incentivos que foram realizados com sabedoria e maestria durante o desenvolvimento deste trabalho.

Meus agradecimentos a todos os meus amigos, que me ajudaram em momentos de dificuldade e que fizeram parte da minha vida acadêmica.

À todos que direta ou indiretamente fizeram parte da minha formação, meus agradecimentos.

Sérgio Teixeira Corrêa Filho

AGRADECIMENTOS

Agradeço à minha mãe Sandra Mesquita que sempre esteve ao meu lado e foi a minha maior incentivadora.

Ao meu pai Ronaldo Araújo que batalhou por anos para proporcionar a melhor educação para seus filhos.

Aos meus amigos Sérgio Corrêa e Vitor Brasil que acreditaram no potencial do projeto.

Bernadete Pimenta, obrigado por ser minha companheira entender a minha dedicação e momentos de reclusão.

Agradeço aos meus queridos professores que se dedicaram a ensinar e compartilhar todo o seu conhecimento.

Um agradecimento especial a professora Polyana Fonseca que fez toda a diferença na orientação da nossa monografia.

Agradeço a Deus e a todos que contribuíram direta ou indiretamente para a realização dessa etapa.

Raul Vitor Mesquita da Silva

"Nem todos os que vagueiam estão perdidos". J. R. R. Tolkien

RESUMO

Este trabalho de conclusão de curso tem como principal objetivo de fornecer uma melhoria para o fluxo de transito com a otimização dos tempos de semáforos, através de uma aplicação de processamento digital de imagem e controladores. Para que assim seja possível a elaboração de semáforos inteligentes que possam contribuir para um trânsito melhor nas grandes cidades. O sistema todo conta com um conjunto de *scripts* que utilizam a linguagem Python e seus específicos módulos para executar diferentes funções, tais como processar digitalmente um vídeo de tráfego urbano, analisar o estado da via e determinar o seu estado como fluxo leve, moderado, intenso e por fim, realizar a otimização do tempo de sinal verde de um determinado semáforo.

Palavras-chave: *Semáforo; trânsito; processamento de imagem; sistemas especialistas; controladores.*

ABSTRACT

This course completion work has as main objective to provide an improvement to the flow of traffic with the optimization of traffic light times, through a digital image processing application and controllers. So that it is possible to create intelligent traffic lights that can contribute to a better traffic in big cities. The whole system has a set of scripts that use the Python language and its specific modules to perform different functions, such as digitally processing an urban traffic video, analyzing the state of the road and determining its state as light, moderate, intense flow and finally, to perform the optimization of the green signal time of a certain semaphore.

Keywords: *Traffic light; Traffic; image processing; expert systems; controllers.*

LISTA DE FIGURAS

Figura 1: Representação de quantidade de veículos por hora x horas do dia 25
Figura 2: Diagrama de bloco do sistema Scoot. 27
Figura 3: Cruzamento com a instalação dos sensores de veículos. 27
Figura 4: Representação da detecção 28
Figura 5: Representação da detecção. 28
Figura 6: Ilustração do conceito de pixels e do sistema de coordenadas para representação de imagem digital. 30
Figura 7: Primeira imagem digital registrada por Russel Kirsch. 32
Figura 8: Etapas de processamento de imagem digital. 32
Figura 9: Imagem original antes da amostragem. 34
Figura 10: Imagem original antes (a) e depois (b) de uma amostragem com frequência Fs = 1/32px^-1 nas direções espaciais. 35
Figura 11: (a) Imagem da figura 10 quantizada em 2 bits. Valores de entrada variam entre 0 a 15. (b) quantizada em 2 bits. Valores de entrada variam entre 0 a 3. 35
Figura 12: Imagem da figura 4 quantizada em 1 bits. Valores de entrada variam entre 0 ou 1. 36
Figura 13: Vizinhança de um pixel N8. 36
Figura 14: A imagem “Lena” (a) e seu histograma respectivo (b). 37
Figura 15: A imagem “Lena” com menos brilho (a) e seu histograma respectivo(b). 37
Figura 16: A imagem “Lena “com mais brilho (a) e seu histograma respectivo(b). 38
Figura 17: Diagrama mostrando as etapas de um sistema de processamento de imagens. 39
Figura 18: Histograma de intensidade sendo dividido por um limiar único. 40
Figura 19: (a) Transformação global para binarização. (b) Transformação em um histograma de perfil não bimodal. 41
Figura 20: Máscaras para o cálculo de gradientes utilizadas na detecção de: a) Prewitt; b) Sobel. 42
Figura 21: a) Recorte da Imagem original, b) bordas detectadas com o operador de Prewitt; c) bordas detectadas com o operador Sobel. 44
Figura 22: Imagem Lena em preto e branco – (a) imagem original, (b) imagem com ruído. 44
Figura 23: Imagem Lena colorida – (a) imagem original, (b) imagem com ruído. 44
Figura 24: Imagem original (a), e espectro discreto de potência localizado no centro da imagem (b). 45
Figura 25: Espectro discreto de potencias da imagem filtrada por um passa-baixas ideal (a) e a imagem reconstruída, onde é possível notar-se a suavização das bordas e a consequente perda de detalhes finos (b). 46
Figura 26: Espectro discreto da imagem digital filtrada por um passa-alta (a) e imagem reconstruída a onde é possível observar a evidenciação das bordas e destaque os detalhes (b).. 47
Figura 27: Espectro discreto da imagem digital filtrado por um passa-bandas (a) e imagem reconstruída e filtrada, onde se manteve os detalhes e o intervalo fechado das frequências (b). 47
Figura 28: Representação de Estrutura de Sistema Especialista. 51
Figura 29: Representação ilustrativa de um CLP. 58
Figura 30: Arquitetura de um CLP. 59
Figura 31: Módulo de fonte de alimentação para CLP. 60
Figura 32: Módulo de um processador para CLP 60
Figura 33: Fluxograma de um ciclo de scan de um CLP. 62

Figura 34: Controlador lógico programável e interface gráfica do programa. 63
Figura 35: Aplicação de um CLP de terminal único. 64
Figura 36: Ilustração de uma aplicação de CLP gerenciador de controle. 64
Figura 37: Fluxograma das etapas do projeto. 67
Figura 38: Código para funcionamento do filtro de Kalman. 69
Figura 39: Código para detecção de veículos. 70
Figura 40: Binarização sendo feita sobre as filmagens da avenida. 71
Figura 41: Código para cálculo de estimativa da velocidade. 72
Figura 42: Código para registrar a localização dos veículos e suas velocidades. 72
Figura 43: Código para calcular média de velocidade e definir estado da via. 73
Figura 44: Fluxograma para determinar estado da via. 74
Figura 45: Código do funcionamento do semáforo. 75
Figura 46: Código de atualização do tempo de verde. 76
Figura 47: Fluxograma de máquina de estados. 76
Figura 48: Fluxograma da máquina de estado 77
Figura 49: Código para inserir o valor para o banco de dados. 78
Figura 50: Configurações do banco de dados para inserir no script 79
Figura 51: Código para salvar valor da variável no micro controlador. 80
Figura 52: Bocais finalizados da maquete. 81
Figura 53: Circuito eletrônico inserido na maquete. 82
Figura 54: Diagrama do circuito eletrônico. 82
Figura 55: Base feita por canos PVC. 83
Figura 56: Maquete finalizada 83
Figura 57: Binarização realizada em imagens autorais. 84
Figura 58: Teste para detecção da velocidade dos veículos. 85
Figura 59: Variação das velocidades médias com cada vídeo. 86
Figura 60: Variação do tempo de verde para cada vídeo. 87
Figura 61: Variação da velocidade média com dados fictícios. 87
Figura 62: Variação do tempo de verde com dados fictícios. 88
Figura 63: Código para “desenhar” o semáforo. 95
Figura 64: Código do funcionamento do semáforo. 96
Figura 65: Demonstração do semáforo virtual. 96

Sumário

1 INTRODUÇÃO 15
1.1 OBJETIVO GERAL 16
1.2 OBJETIVOS ESPECÍFICOS 16
1.3 METODOLOGIA 17
1.4 ORGANIZAÇÃO DO TRABALHO 17
2 SEMÁFORO 19
2.1 COORDENAÇÕES DOS SEMÁFOROS 20
2.2 ELEMENTOS DA PROGRAMAÇÃO SEMAFÓRICA 21
2.3 MODOS DE OPERAÇÃO SEMAFÓRICA 24
2.3.1 Semáforos operados em tempo fixo 25
2.3.2 Semáforos operados em tempo real 26
3 IMAGEM DIGITAL 30
3.1 CONCEITOS DE PROCESSAMENTO DE IMAGEM DIGITAL 31
3.2 HISTÓRICO 31
3.3 ETAPAS DO PROCESSAMENTO DE IMAGEM 32
3.3.1 Aquisição 33
3.3.2 Armazenamento 33
3.3.3 Processamento 33
3.3.4 Transmissão 33
3.3.5 Exibição 33
3.5 AMOSTRAGEM E QUANTIZAÇÃO 33
3.6 VIZINHANÇA E COORDENADAS ESPACIAIS 36
3.7 HISTOGRAMA 36
3.8 SEGMENTAÇÃO 38
3.9 LIMIARIZAÇÃO 40
3.10 DETECÇÃO DE BORDAS 42
3.11 FILTRAGEM 44
3.11.1 Filtros passa-baixas 45
3.11.2 Filtros passa-altas 46
3.11.3 Filtro passa-bandas 47

3.11.4 Filtro de Kalman **48**
4 SISTEMAS ESPECIALISTAS **50**
4.1 VANTAGENS DO USO DE UM SISTEMA ESPECIALISTAS: 50
4.2 BASE DE CONHECIMENTO 51
4.3 NÚCLEO DO SISTEMA ESPECIALISTA 52
4.3.1 Módulo Coletor de Dados **52**
4.3.2 Módulo de Explicação **53**
4.4 MEMÓRIA DE TRABALHO 54
4.5 INTERFACE 54
4.6 CONHECIMENTO 54
4.7 AQUISIÇÃO DO CONHECIMENTO 55
4.7.1 Técnicas Manuais **55**
4.7.2 Técnicas semiautomáticas **56**
4.7.3 Técnicas automáticas **56**
5 CONTROLADORES **57**
5.1 VANTAGENS DOS CLPS 58
5.2 ARQUITETURA 59
5.3. FUNCIONAMENTO DE UM CLP 61
5.4 CLASSE E APLICAÇÃO DOS CLP 63
6 PROCEDIMENTOS METODOLóGICOS **66**
6.1 COLETA DE DADOS CONTÍNUOS 67
6.2 IMPLEMENTAÇÃO DO CÓDIGO FONTE PARA O TRABALHO 68
6.3 CRIAÇÃO DO BANCO DE DADOS 77
6.4 CONSTRUÇÃO DA MAQUETE 80
7 RESULTADOS OBTIDOS **84**
8 CONSIDERAÇÕES FINAIS **89**
8.1 TRABALHOS FUTUROS 89
9 REFERÊNCIAS BIBLIOGRÁFICAS **91**
Apêndice A – SCRIPT PARA IMPLEMENTAÇÃO DO SEMÁFORO VIRTUAL **95**

1 INTRODUÇÃO

O problema de congestionamentos no trânsito das grandes cidades é muito comum em nossa rotina, o que afeta de forma negativa a vida de uma grande parcela de moradores dos centros urbanos brasileiros. Entendendo este fato como um problema que precisa ser ao menos amenizado, este trabalho se propôs a buscar algum tipo de solução satisfatória que pudesse se encaixar neste cenário.

De acordo com Martinez (2008), As maiores cidades do Brasil têm um problema em comum: os congestionamentos diários. Nos últimos anos, a frota de veículos do país passou de 30 milhões para 50 milhões, um aumento de 66.6%. Os especialistas em trânsito avaliam vários motivos para a justificativa de tantos engarrafamentos, tais como a facilidade para a obtenção de crédito bancário e os planos de financiamento oferecidos pelas concessionárias de automóveis.

Existem também outros motivos que contribuem para esta realidade no trânsito das grandes metrópoles. Segundo Gaete (2016), uma das principais causas dos congestionamentos seria a falta da priorização do poder público para os meios de transporte públicos, pois nos setores onde o serviço não é oferecido com qualidade para os habitantes de um centro urbano, muitos acabam sendo forçados a dependerem do uso de automóveis próprios.

Tendo esta realidade em mente, o projeto desenvolvido neste trabalho tem como principal objetivo realizar uma proposta para tentar amenizar este problema, com a elaboração de uma solução que possa contribuir para a melhora da fluidez do trânsito nas principais vias urbanas, assim colaborando para a redução do tráfego e congestionamento.

Segundo Tanscheit (2016) é indiscutível o fato de que as novas demandas de infraestrutura, serviços e economia irão exigir mudanças significativas nas cidades. O uso de novos recursos pode garantir o sucesso das respostas que as cidades darão diante dessa rápida urbanização. E é por isso que uma cidade precisará ser inteligente.

De acordo com Richter (2017), as cidades inteligentes movimentam bilhões de dólares por ano, já que uma cidade conectada é também mais eficiente. Por isso que esta área é muito importante para ser investida, pois quando aplicada de forma correta, nota-se uma melhoria significativa no desenvolvimento econômico e social. Assim é bem

provável que futuramente as empresas se envolvam cada vez mais na área, devido à alta demanda por parte dos setores públicos.

O projeto que será desenvolvido por estes autores também se encaixa na área de cidades inteligentes, que têm como principal objetivo de oferecer uma qualidade de vida melhor para uma determinada população de uma cidade.

Segundo Drey (2015), o trânsito lento ocorre em boa parte do tempo em que os semáforos atuam, alguns deles mal regulados ou não sincronizados com o seguinte acabam prejudicando os motoristas, colaborando com possíveis engarrafamentos. Até alguns anos, havia apenas os semáforos que atuam com um temporizador individual. Hoje existem semáforos que se adaptam as condições do trânsito em tempo real, embora estes últimos sejam pouco utilizados no Brasil.

Analisando os fatos citados acima, os autores acreditam que semáforos adaptativos possam contribuir para uma melhor infraestrutura nas cidades brasileiras, já que o mesmo se empenhará para entregar aos motoristas um trânsito com mais fluidez e menos congestionamentos e engarrafamentos, além de gerar receita para os cofres públicos.

1.1 OBJETIVO GERAL

O objetivo geral do projeto consiste em desenvolver um sistema que fará uso de processamento de imagem digital com a aplicação de controladores para otimizar o tempo de um semáforo.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho serão:

- Entender o funcionamento dos tempos de um semáforo;
- Estudar sobre scripts e bibliotecas em *Python* que sejam apropriadas para o uso de processamento de imagem;
- Buscar compreensão sobre módulos e bibliotecas do *Python* que possam simular o funcionamento de um controlador;
- Desenvolver um script com a função de reconhecimento e detecção da velocidade dos veículos em uma determinada via;
- Elaborar um script com a função de um controlador para um semáforo inteligente;

- Integrar um código fonte responsável por aperfeiçoar e atualizar os tempos de verde de um semáforo;
- Escrever um código fonte para enviar e receber o tempo do ciclo verde para um banco de dados no *Firebase*;
- Criar uma maquete para simular o funcionamento de um semáforo.
- Medir os resultados obtidos com o projeto;
- Verificar quais melhorias podem ser alcançadas com sua implementação.

1.3 METODOLOGIA

Os autores deste trabalho deram seu primeiro passo neste projeto com um estudo aprofundado em scripts e bibliotecas da linguagem de programação *Python*, para a implementação de códigos capazes de identificar a quantidade de veículos e a sua velocidade em uma determinada via por processamento digital de imagem, assim como também simular o funcionamento de um controlador para um semáforo.

A equipe realizou filmagens de uma via do município de Belém para que fosse possível serem utilizadas como entrada para a aplicação, com a finalidade de testar o funcionamento da mesma. Depois de realizado este procedimento, foi necessário o desenvolvimento de um código para a otimização dos tempos de um semáforo, também incluindo testes e analises de funcionamento para verificar se o mesmo está sendo implementado de forma eficiente.

Os autores também criaram uma maquete para que fosse possível visualizar de forma prática o funcionamento da aplicação e testar os resultados obtidos, para mensurar quais benefícios o projeto pode trazer, assim como verificar se os resultados foram satisfatórios e determinar as futurar implementações para aperfeiçoamento do projeto.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido em capítulos, onde cada um terá sua determinada função, o primeiro capítulo teve a função de apresentar os problemas existentes não só em Belém, mas como também na maioria dos grandes centros urbanos brasileiros, que seriam os congestionamentos que interferem no fluxo de veículos. Outro ponto discutido são as justificativas para a implementação deste projeto, mostrando fatos e argumentos que comprovam o funcionamento de projetos voltados para mobilidade urbana e os benefícios trazidos após suas implementações. O capítulo encerra com a

exposição dos objetivos gerais e específicos, assim como também a metodologia usada pelos autores.

O segundo capítulo aborda o conceito de um semáforo de trânsito, assim como também a explicação de conceitos importantes, tais como os seus ciclos e explicação de cada tipo de sinalização, assim como seus possíveis modos de operação.

O terceiro capítulo terá como principal função de fundamentação teórica, realizando uma abordagem mais técnica e aprofundada em assuntos importantes para este trabalho, tais como processamento de imagem digital, controladores e sistemas especialistas.

O quarto capítulo terá seu foco voltado para a exposição detalhada dos procedimentos e métodos aplicados pelos autores para realizar a implementação deste trabalho. Assim, o mesmo irá ter a sua finalização expondo detalhadamente a análise dos resultados com a otimização dos tempos dos semáforos obtida.

O quinto capítulo será das considerações finais, onde serão mencionados brevemente todos os procedimentos que foram tomados pelos autores no decorrer do desenvolvimento do trabalho, assim como também os autores vão expor a sua conclusão, mostrando os benefícios que podem ser alcançados com base no que for mostrado na análise dos resultados e propostas de trabalhos futuros.

2 SEMÁFORO

O semáforo é um dispositivo eletromecânico de sinalização viária composta por indicações luminosas, estas que são acionadas de forma alternada ou intermitentemente por meio de um sistema eletrônico que possui a finalidade de transmitir diferentes mensagens aos motoristas inseridos em uma via pública, assim regulamentando o direito de passagem ou advertindo sobre outras situações especiais nas vias (CUCCI NETO, 2014).

O semáforo é um instrumento muito importante no trânsito, já que é um bom controlador de trânsito que possui a capacidade de amenizar os possíveis conflitos entre os veículos, aumentar a capacidade de vazão e organizar o trânsito, principalmente nos horários de pico, onde costumam haver um fluxo mais intenso (ARAUJO, 2006).

As ruas e avenidas podem ser definidas como o meio físico de circulação dos automóveis, assim, elas devem ser projetadas para se adaptar ao fluxo diariamente. Em um cruzamento entre duas ou mais vias existem movimentos que não podem ser realizados simultaneamente, pois são conflitantes entre si (DENATRAN, 1984). Para evitar os conflitos é preciso estabelecer regras de controle do direito de passagem, tanto para veículos como para pedestres.

Os conflitos entre veículos podem ser resolvidos pela regra do primeiro a chegar é o primeiro a atravessar o cruzamento, mas a mesma só pode se aplicar em vias que possuem um baixo fluxo de automóveis. Em ruas movimentadas é necessário estabelecer novas regras de prioridade entre as aproximações do cruzamento para permitir a travessia da interseção. Uma forma de implementar regras de autorização e proibição de movimentos no cruzamento seria com o suo de semáforos (DENATRAN, 1984).

O semáforo faz uso do método de revezamento para realizar o controle do fluxo, ou seja, quando uma via possui permissão do semáforo para prosseguir com o fluxo, a outra via deve ter o seu fluxo interrompido e vice-versa. Para que haja um tráfego ordenado entre as vias, o semáforo faz uso de sinais visuais para regular a movimentação de veículos (CUCCI NETO, 2014). Estes dispositivos eletrônicos possuem três ciclos de tempo, ou seja, estados que são representados por suas respectivas cores:

- O primeiro ciclo tem a finalidade de permitir a passagem dos veículos que estão presentes em uma determinada via este que é representado pela cor verde no semáforo.
- O segundo ciclo tem como função de emitir um alerta para os condutores de que o tempo do sinal verde está prestes a se esgotar. Desse modo, o motorista tem tempo suficiente para realizar uma parada em segurança, devido à distância percorrida durante o tempo de frenagem, mas caso não seja possível, o motorista ainda possui permissão para realizar a passagem na via. Este ciclo é representado pela cor amarela.
- O terceiro ciclo é a tem como função de indicar a proibição da passagem dos condutores de uma determinada via, para que assim seja possível o fluxo de outra via. Este tempo do semáforo é representado pela cor vermelha.

2.1 COORDENAÇÕES DOS SEMÁFOROS

De acordo com Cucci Neto (2014), para a coordenação dos semáforos são utilizados equipamentos programáveis que realizam as trocas de cada sinalização, os controladores podem ser divididos em eletromecânicos e eletrônicos:

- Controladores eletromecânicos: Programação implementada a partir da junção de recursos eletrônicos e mecânicos. Possuem apenas uma programação semafórica e recursos operacionais limitados.

- Controladores eletrônicos: são constituídos por componentes elétricos e eletrônicos. Sua programação é implementada a partir de recursos computacionais do equipamento. Este tipo de tecnologia permite que os equipamentos disponham de recursos de programação que facilitam as soluções de engenharia.

A programação dos controladores do semáforo pode ser realizada de duas maneiras, centralizada e descentralizada. No modo de operação descentralizada a programação semafórica é implementada diretamente no controlador, em campo. Qualquer alteração que seja necessária realizar na programação deve ser feito usando os recursos disponíveis no controlador para a entrada de dados, sendo alterada de forma manual (CUCCI NETO, 2014, p. 67).

Segundo Cucci Neto (2014), no modo de operação centralizado, os controladores eletrônicos de tráfego são ligados a um computador central que utiliza programas próprios para gerenciamento da operação conjunta dos equipamentos. O controle centralizado é utilizado para agilizar a operação do sistema de interseções semaforizadas, admitindo vários níveis de funcionamento.

2.2 ELEMENTOS DA PROGRAMAÇÃO SEMAFÓRICA

Vários critérios podem ser usados para realizarem o controle dos tempos de um semáforo. O método mais utilizado é o de grau de saturação e este inicialmente vai determinar o grau de saturação das vias, que também podem ser chamadas de *links*, assim as mesmas irão ser controladas para determinar o tempo do ciclo verde e vermelho.

Segundo Vilanova (2005), podemos definir o grau de saturação como a relação entre o número de veículos que desejam passar e o número de veículos que pode conseguir passar durante certo período de tempo.

O grau de saturação costuma ser representado pela variável x e pode ser definido algébricamente da seguinte forma: Quando o grau de saturação está em 100% (x = 1), significa que o tempo do ciclo verde é suficiente para permitir a passagem de todos os veículos. Caso o grau seja menor que 100% (x<1), indica que um número maior de veículos pode passar durante o tempo do ciclo verde, já que teria mais tempo que o necessário. Por fim, quando o grau de saturação se encontra maior que 100% (x>1), significa que o tempo do sinal verde é insuficiente para permitir a passagem dos automóveis naquela via (VILANOVA, 2005).

Em termos matemáticos, o grau de saturação é definido pela equação 1:

$$x = \frac{\mathrm{F} * \mathrm{Tciclo}}{\mathrm{FS} * \mathrm{Tverde}} \qquad \text{eq.(1)}$$

Onde temos:

- x = grau de saturação do link;
- F= fluxo atual do link;
- FS = fluxo de saturação do link;
- Tciclo = Tempo de ciclo correspondente (verde amarelo e vermelho);
- Tverde = Tempo de verde correspondente.

Também pode ser representada por:

$$x = y * \frac{1}{p} \qquad \text{eq.(2)}$$

Onde:

- x = grau de saturação do link;
- y = taxa de ocupação;
- p = porcentagem de verde correspondente.

A partir do momento que é obtido o grau de saturação, podemos determinar qual será o tempo de cada ciclo de um semáforo, assim, precisamos primeiramente calcular a taxa de ocupação da via ao qual o semáforo está controlando o fluxo (VILANOVA, 2005). Este que é representado pela equação 3:

$$y = \frac{F}{FS} \qquad \text{eq.(3)}$$

Onde temos:

- y = taxa de ocupação da via;
- F = fluxo atual da via;
- FS = fluxo de saturação da via.

Vale ressaltar que as variáveis F e FS são medidas em veículos por hora (v/h), assim o fluxo de saturação será a quantidade de veículos que passam naquele determinado período (VILANOVA, 2005). O cálculo do fluxo de saturação será explicado mais a frente no trabalho. Depois que for realizado o cálculo da taxa de ocupação, a próxima variável para se calcular é do tempo verde pela equação 4:

$$p = \frac{y}{x} \qquad \text{eq.(4)}$$

Onde:

- p = porcentagem de verde do link;
- y = taxa de ocupação correspondente;
- x = grau de saturação definido.

Uma informação importante é que cada grau de saturação irá ter seus próprios valores de y e p.

Depois que for realizado o cálculo das taxas citadas acima, é necessário realizar o cálculo das variáveis que estão relacionadas com o ponto de intersecção entre as vias, ou seja, o seu cruzamento. A primeira variável é conhecida como tempo morto (tm) de um cruzamento, esta que é a soma dos tempos de amarelo (ta) e vermelho (tv) dos semáforos envolvidos no cruzamento (VILANOVA, 2005). Esta variável é calculada pela equação 5:

$$\mathrm{tm} = \sum(\mathrm{tai} + \mathrm{tvli}) \qquad \text{eq.(5)}$$

Onde:

- tm = tempo morto do cruzamento;
- tai = tempo de amarelo do semáforo i;
- tvi = tempo de vermelho de limpeza do semáforo i.

Após o cálculo do tempo morto, precisamos achar o valor da segunda variável que é o tempo de ciclo (tc). A função desta variável é indicar o tempo de ciclo total de um cruzamento e é representada pela equação 6:

$$\mathrm{tc} = \frac{\mathrm{tm}}{1-\sum(\mathrm{pi})} \qquad \text{eq.(6)}$$

Onde temos:

- tc = tempo de ciclo do cruzamento;
- tm = tempo morto do cruzamento;
- pi = porcentagem de verde do semáforo i.

Por fim, temos a última variável, que seria o tempo do ciclo verde de cada sinal (tvi), este que é calculado pela equação 7:

$$\mathrm{tvi} = yi * tc \qquad \text{eq.(7)}$$

Onde:

- tvi = tempo de verde do semáforo i;
- yi = taxa de ocupação da via do semáforo i;
- tc = tempo de ciclo do cruzamento.

A decisão do grau de saturação é uma das etapas mais importantes na implementação de um semáforo, pois o seu valor será utilizado para calcular e

determinar os tempos de todos os ciclos. Esta sessão terá como principal objetivo de demonstrar como são realizados os procedimentos para calcular matematicamente o grau, já que anteriormente foi demonstrada a forma de estimar o seu valor, assim o cálculo desta variável será definido por aquele que estiver realizando o cálculo dos ciclos do semáforo (VILANOVA, 2005).

Caso o grau de saturação tenha um valor muito elevado, pode ocorrer que não seja possível adotar o valor resultante, pois levaria a tempos de ciclo impraticavelmente altos. Outro fator que pode ser considerado prejudicial seria uma taxa de ocupação com um valor muito baixo, que teria como principal consequência um tempo de sinal verde insuficiente para oferecer a segurança durante a permissão da passagem dos veículos (VILANOVA, 2005).

É sugerido que o grau de saturação seja configurado no valor de faixa entre 80% a 90%. Com o seu valor nesta faixa percentual, é possível obter os menores valores possíveis para o atraso total do semáforo. Caso não haja fatores especiais que justifiquem outros valores, aconselha-se a adoção para o grau de saturação de 88% para todos os *links* críticos (VILANOVA, 2005).

É importante ressaltar que a escolha desta faixa de valor para o grau de saturação se dá pelo fato de que fatores externos podem levar a um fluxo de veículos mais intenso, como o horário, acidentes e outras ocorrências. No caso dos semáforos inteligentes, que trabalham em tempo real, tendem a manter o valor do grau de saturação o mais próximo possível do valor recomendando, sempre se adaptando às situações externas (VILANOVA, 2005).

2.3 MODOS DE OPERAÇÃO SEMAFÓRICA

Os semáforos de trânsito podem ser implementados para operarem de duas formas: em tempo fixo (o mais comum) e em tempo real.

O controle em tempo real é constituído por uma tecnologia complexa desconhecida para aqueles que estão acostumados com o tempo fixo. Alguns técnicos que atuam na área creem que após a implementação de tal sistema, seus trabalhos seriam desnecessários, já que o modo de operação do semáforo seria automático. Esta é uma informação errada, já que um sistema em tempo real exige atenção de uma equipe capacitada (MORENO; MAMADE; FILHO, 2014 apud PEREIRA; RIBEIRO, 2007).

2.3.1 Semáforos operados em tempo fixo

De acordo com Moreno, Mamade e Filho (2014), um conjunto semafórico que opera em uma via qualquer pode ser programado para operar em um tempo fixo específico para cinco estágios, pois existe uma variação de veículos para cada horário. A figura 1 demonstra a relação da quantidade de veículos por horas do dia.

Figura 1: Representação de quantidade de veículos por hora x horas do dia
Fonte: Moreno, Mamade e Filho, 2014.

Em razão da variação média do fluxo no tempo semafórico, é necessário integrar um tempo extra devido às aleatoriedades cotidianas. Quando houver uma demanda maior de veículos na região, o tempo destinado às aleatoriedades será maior e consequentemente o haverá um aumento no tempo perdido de escoamento de veículos (MORENO; MAMADE; FILHO, 2014).

Foi descrito no tópico acima que o semáforo pode ser programado para trabalhar em cinco estágios diferentes no decorrer do dia, como são mostrados abaixo:

- Estágio 1: Programação semafórica para um fluxo de 500 veículos por hora;
- Estágio 2: Programação semafórica para um fluxo de 1.000 veículos por hora;
- Estágio 3: Programação semafórica para um fluxo de 750 veículos por hora;
- Estágio 4: Programação semafórica para um fluxo de 600 veículos por hora;
- Estágio 5: Programação semafórica para um fluxo de 2.000 veículos por hora.

Um dos maiores obstáculos encontrados na programação de tempo fixo é a carência de atenção e constantes atualizações, já que todo ano ocorrem diversas modificações no fluxo de trânsito, tais como aumento da frota e extinção de polos geradores de tráfego (MORENO; MAMADE; FILHO, 2014).

Outro problema presente na programação semafórica em tempo fixo é a inexistência de reação no momento que acontece alguma situação atípica no trânsito, já

que a programação tem como dados de entrada pesquisas de contagem de carros históricas (MORENO; MAMADE; FILHO, 2014).

A programação semafórica em tempo fixo não deixa de ser importante, já que a mesma pode ser útil em diversas situações. Quando é necessário aumentar o fluxo de veículos em uma via, o controlador pode ser programado de 3 formas distintas: simultâneo, alternado e progressivo.

A. Simultâneo: abertura de um conjunto de sinais de trânsito de uma via ao mesmo tempo, com desvantagem de estimular altas velocidades.
B. Alternado: Os semáforos de transito acionam o ciclo de tempo verde de forma alternada, sendo eficientes quando houver uma distância considerável entre as interseções semafóricas.
C. Progressivo: Ajusta os tempos de cada semáforo, proporcionando a passagem de um grupo de veículos por toda a via, mantendo uma velocidade de progressão.

2.3.2 Semáforos operados em tempo real

Segundo Moreno, Mamade e Filho (2014), semáforos operados em tempo real são constituídos por sensores que atuam em uma via. Pela a passagem de veículos, a informação é transmitida para uma central, que vão adaptando o tempo dos semáforos para atender a demanda veicular.

Para Moreno, Mamade e Filho (2014), a grande maioria dos semáforos é programada pelo sistema de automatização Scoot, este que é capaz de efetuar o controle em tempo real e se adaptando às variações aleatórias que ocorrem de ciclo em ciclo.

Segundo Ming (1997, *apud* MORENO; MAMADE; FILHO, 2014), o sistema Scoot é constituído por cinco etapas:

- Sistema de detecção de veículos;
- Sistema de transmissão de dados;
- Computador central;
- Controladores;
- Grupos focais.

A figura 2 demonstra o diagrama de blocos do sistema Scoot.

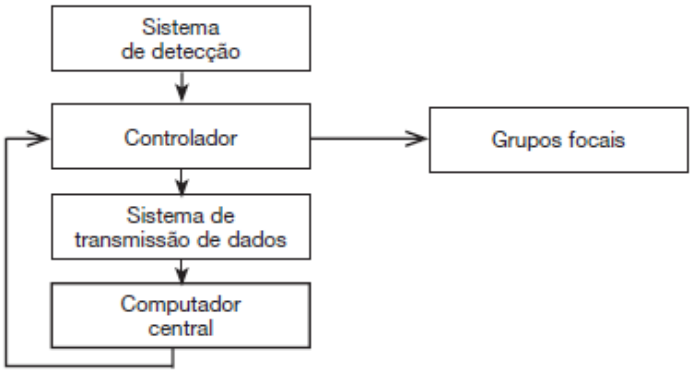


Figura 2: Diagrama de bloco do sistema *Scoot*.
Fonte: Moreno, Mamade e Filho, 2014.

O sistema de semáforos em tempo real é demonstrado na figura 3. Nela os sensores indicam o fluxo de veículos e os dados são coletados pelo controlador e os envia para um computador central. Este tem a função de processar as informações, calcular os ciclos dos semáforos e envia ao controlador que os programa nos grupos focais (MORENO; MAMADE; FILHO, 2014).

A detecção dos veículos deve ser realizada entre 8 a 12 segundos antes de qualquer veículo chegar na faixa de pedestre (conforme a figura 3), para que assim possa haver tempo suficiente para o sistema Scoot operar de forma eficiente.

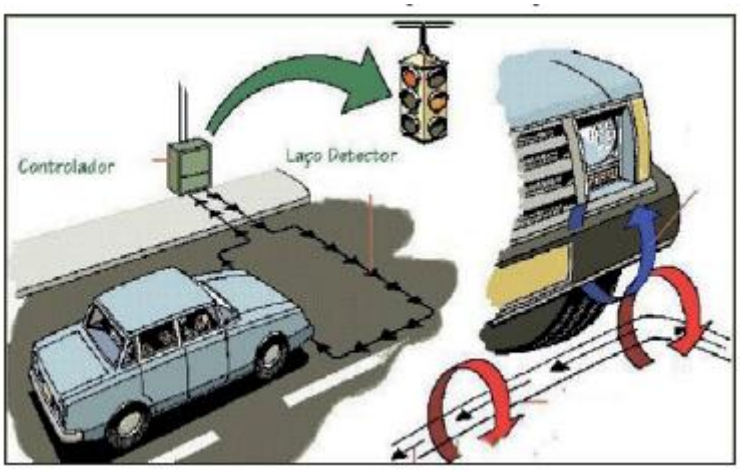


Figura 3: Cruzamento com a instalação dos sensores de veículos.
Fonte: Fonte: Moreno, Mamade e Filho, 2014.

Para que seja possível entender o funcionamento do sistema de detecção do veículo, é usado um exemplo numérico descrito por Ming (1997, apud MORENO; MAMADE; FILHO, 2014), onde a cada ¼ de segundo o sistema de detecção verifica se o laço está ocupado ou não (figura 4).

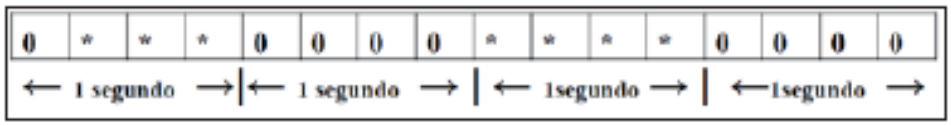


Figura 4: Representação da detecção
Fonte: Fonte: Moreno, Mamade e Filho, 2014.

Para Moreno, Mamade e Filho (2014), a detecção é realizada em intervalos de ¼ de segundo (250 milissegundos). O que se detecta é a junção do fluxo de veículos (em veículos por segundo ou por hora) e a porcentagem de ocupação do laço, cuja unidade é expressa em LPU (*link profile unit*). O número de LPU é obtido da seguinte forma:

- O 1° intervalo de ¼ de segundo "*" (ocupado) após um intervalo com "0" (não ocupado), equivale a 7 LPUs;
- O 2° é igual a 6 LPUs;
- O 3° é igual a 5 LPUs;
- O 4° é igual a 4 LPUs;
- O 5° é igual a 3 LPUs;
- O 6° equivale a 2 LPUs;
- O 7° equivale a 1 LPUs.

Para facilitar o entendimento deste cálculo, é mostrada na figura 5 de um exemplo numérico de detecção de 3 veículos em um tempo de 6 segundos:

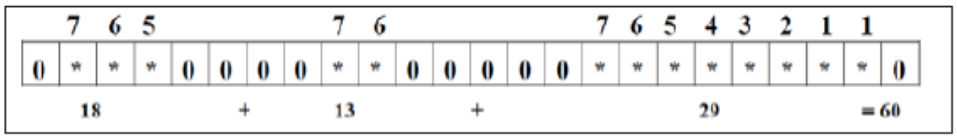


Figura 5: Representação da detecção.
Fonte: Moreno, Mamade e Filho, 2014.

No exemplo acima, os 3 veículos representam 20 LPU cada e com essas informações, pode-se determinar que:

- Em termos de fluxo de tráfego: fluxo = 3 veículos / 6 segundos = 0,5 veículos/segundo = 1.800 veículos/hora;
- Em termos de porcentagem de ocupação: ocupação = 13 intervalos / 24 intervalos = 54%;
- Em termos de LPU: 60 LPU / 6 segundos = 10 LPU/segundo.

A medida em LPU ocorre em função da vazão de veículos, da velocidade, do comprimento dos veículos e da disposição da via. Vale ressaltar que não existe uma

igualdade fixa e matemática entre o número de veículos e o número de LPU, pois a equivalência entre as duas medidas depende do local, do momento e do fluxo do tráfego (MORENO; MAMADE; FILHO, 2014).

Com base no que foi descrito no decorrer do texto, é possível chegar a uma conclusão de que a implementação de semáforos inteligentes pode trazer grandes melhorias para o trânsito, pois os mesmos são capazes de se adaptarem às situações externas e manter seu funcionamento de forma eficaz, assim evitando todo tipo de transtorno.

3 IMAGEM DIGITAL

Para que seja possível dar início à esta sessão, o primeiro procedimento será uma breve descrição do que seria uma imagem. Para Gonzáles e Woods (2011), uma imagem seria uma função bidimensional f (x, y), onde x e y seriam coordenadas espaciais e f seria representa a intensidade ou o nível de cinza de uma imagem neste ponto. A partir do momento que estas variáveis possuem seu determinado valor a imagem se torna digital. Assim, toda a imagem digital será composta por elementos, estes que são conhecidos como pixels, que terão suas próprias localizações e valores.

Os pontos da matriz são conhecidos como pixels (*picture elements*). Cada pixel tem a função de representar uma parte da cena real, desta forma a resolução espacial da imagem é proporcional aos valores de *M* e *N* correspondentes na matriz (BALAN, 2009).

A figura 6 irá demonstrar de forma mais clara como os pixels estão presentes em uma imagem.

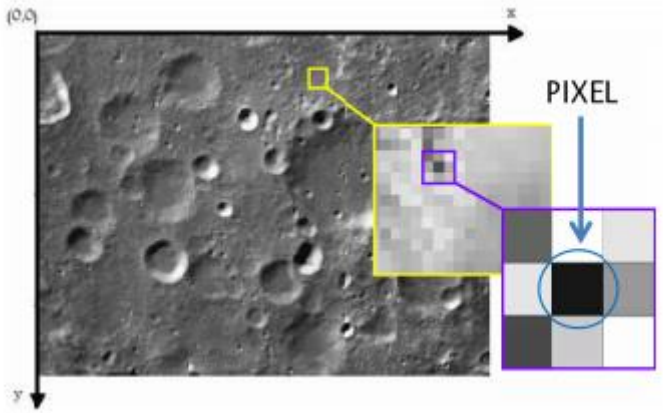


Figura 6: Ilustração do conceito de pixels e do sistema de coordenadas para representação de imagem digital.
Fonte: Pedrini; Schwartz, 2007.

Uma imagem digital é uma representação de uma imagem bidimensional usando números binários, estes que se encontram codificados para que seja possível seu armazenamento, transferência, impressão ou reprodução, e seu processamento por meios eletrônicos (FELISBERTO; BIANCA, 2014).

Pode ser definido que quando uma imagem é convertida para o formato digital, significa que os elementos que a compõem são transferidos para um pequeno formato representativo e original. O menor elemento da imagem é conhecido como pixel e é

identificado de acordo com seu nível de intensidade de cinza e cores correspondentes. Após a sua identificação, estes mesmos elementos são armazenados por códigos que podem ser reconhecidos e apresentados novamente por um dispositivo de visualização (BALAN, 2009).

3.1 CONCEITOS DE PROCESSAMENTO DE IMAGEM DIGITAL

Tendo este fato em mente, podemos considerar que toda imagem é formada por um conjunto de informações que estariam contidas na mesma. De acordo com Albuquerque (2000), toda imagem é composta por alguma informação, que pode estar associada à um nível cognitivo ou a uma medida. Assim, o ato de processar uma imagem seria extrair qualquer tipo de informação nela presente.

Reconstruir uma imagem qualquer pelo sistema visual humano é um procedimento muito complexo, pois a imagem possui muitas informações que são interpretadas por um conjunto de tarefas cognitivas e interpretativas para se extrair as informações transportadas. O processamento de imagem digital é uma tecnologia que permite modificar, analisar e manipular imagens digitais a partir de um computador. A maioria dos algoritmos de processamento de imagens é composta por diversas etapas realizadas com alta velocidade de cálculos que permitem aperfeiçoar as operações de tratamento de imagens (BALAN, 2009).

3.2 HISTÓRICO

Com o avanço das tecnologias, principalmente dos sistemas de comunicação, se tornou uma necessidade a criação de um meio de captar, armazenar e processar imagens.

Segundo Balan (2009), em 1957 foi realizado o primeiro registro de uma imagem digital, o responsável por tal ato foi Russel Kirsch no NBS (*National Bureau of Standards*) conhecido atualmente como NIST (*National Institute of Standards and Technology*). A imagem registrada era uma foto 5x5cm de um bebê, como mostrado abaixo:

Figura 7: Primeira imagem digital registrada por Russel Kirsch.
Fonte: BALAN, 2009

Kirsch e sua equipe também foram capazes de criar um escâner com tambor rotativo e com ele foi possível *scannear* a foto tamanho 4x3 do bebê, que foi um marco crucial para a história como a primeira imagem adquirida e armazenada por meio da representação digital da imagem. A foto foi registrada com 176 pixels em preto e branco (BALAN, 2009).

3.3 ETAPAS DO PROCESSAMENTO DE IMAGEM

O processamento de imagem é formado por vários elementos, como mostrados na figura 8. De acordo com Marques Filho e Vieira Neto (2000) Estes elementos são usados de forma geral em todos os sistemas onde é necessária uma intensa análise de várias imagens, sejam eles de baixo ou alto custo. Todas as etapas da figura 8 foram descritas por Marques Filho e Vieira Neto (2000).

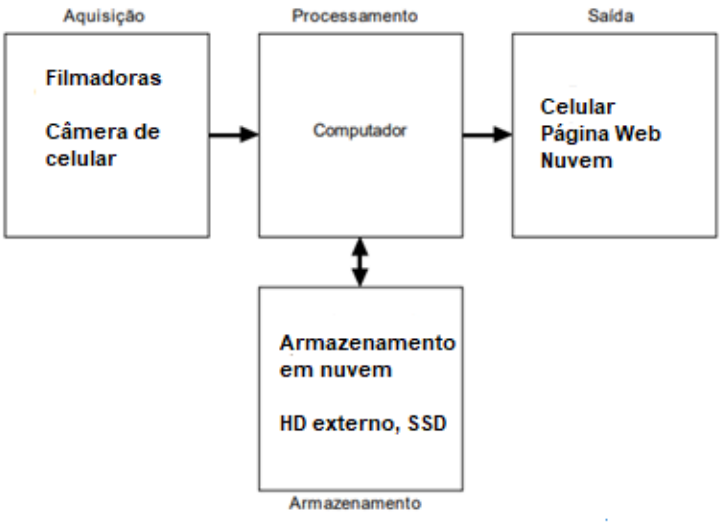


Figura 8: Etapas de processamento de imagem digital.
Fonte: Adaptado de Marques Filho; Vieira Neto, 2000.

3.3.1 Aquisição

Esta etapa tem como principal objetivo de converter uma imagem para uma representação numérica que seja adequada para o processamento digital. Para isso são utilizados dispositivos eletrônicos capazes de extrair do ambiente uma determinada grandeza e consequentemente converte-la em um sinal elétrico, tais como câmeras e *scanners*.

3.3.2 Armazenamento

Esta parte é responsável por armazenar no disco rígido ou em outro dispositivo de armazenamento permanente a saída oferecida pelo processamento de imagem digital. O armazenamento pode ser feito em modo de curta duração, utilizando a memória RAM do computador ou pelo uso de dispositivos de armazenamento.

3.3.3 Processamento

Esta etapa envolve procedimentos em forma algorítmica, assim sendo implementados por software. O uso de hardware é necessário quando o computador que estiver realizado o processamento possuir limitações para realizar tais tarefas.

3.3.4 Transmissão

A etapa de transmissão é responsável por transmitir a distância utilizando redes de computadores e protocolos de comunicação já existentes todos os dados que já foram extraídos do processamento de imagem, sendo necessário o uso de técnicas de compressão e descompressão de imagens, para facilitar a quantidade de dados necessários para serem transmitidos.

3.3.5 Exibição

A última etapa realiza a função de mostrar para o usuário os resultados obtidos pelo processamento de imagem digital, onde serão mostrados todos os dados e informações que foram extraídas das imagens analisadas.

3.5 AMOSTRAGEM E QUANTIZAÇÃO

Pelo fato de que computadores não possuem a capacidade de operar com dados analógicos, se torna necessário realizar a extração da informação desejada da imagem por processamento de imagem digital. Este processo envolve duas etapas, conhecidas como amostragem e a quantização.

Uma função contínua da reflexão de luz no espaço R^2 é representado por *f(x,y)*, assim temos que esta função representa uma imagem no espaço contínuo, onde o seu gráfico é uma superfície e o eixo de Z demonstra o nível de brilho da coordenadas x e y da imagem. Porém, na vida real é impossível a existência de uma imagem contínua devido ao fato de que o computador não possui a capacidade de mapear valores infinitos (MARTINS, 2016).

Para que seja possível a criação de uma imagem digital, a mesma deve ser digitalizada tanto espacialmente quanto em amplitude. A amostragem pode ser definida com o processo de adquirir apenas algumas amostras da função contínua de nível de luz, estas que representam a imagem. A amostragem da imagem *I(x,y)* nas direções de suas coordenadas resulta em uma matriz de NxM amostradas, seguidas pelas etapa de quantização do valor de *I(x,y)* em L níveis de cinza, como é ilustrada na figura 9 (MARTINS, 2016).

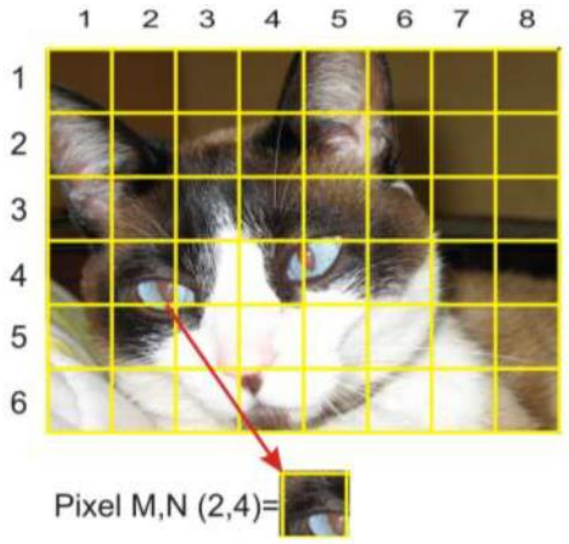


Figura 9: Imagem original antes da amostragem.
Fonte: Balan, 2009.

O processo de amostragem pode ser entendido como a divisão do plano *x,y* em uma grade onde x e y serão números inteiros. Os pontos da matriz são conhecidos como pixels (*picture elements*). Na malha de amostragem, o formato dos pixels pode ser retangular, triangular ou até mesmo possuir uma forma mais complexa. Os valores de cada elemento da matriz tais como coluna x linha (xy) que identifica um único pixel (M, N), devem ser escolhidos de forma a respeitar a relação qualidade da imagem x espaço de armazenamento, devido à aplicação para a qual a imagem está destinada (BALAN, 2009).

O próximo passo após a realização da amostragem é o de atribuição de um valor de brilho para cada amostra que foi coletada, este processo é conhecido como quantização. Nesta matriz, todos os elementos pertencentes as mesmas são conhecidos como pixels. Assim, pode ser definido que uma imagem digital é uma função continua de intensidade de luz, que passou pelas etapas de amostragem e quantização (MARTINS, 2016).

Os efeitos da amostragem podem ser mostrados na imagens abaixo (figura 10), onde será evidenciado o efeito de diferentes formas de quantização (figuras 11 e 12) para um melhor entendimento:

(a) (b)

Figura 10: Imagem original antes (a) e depois (b) de uma amostragem com frequência Fs = 1/32px^-1 nas direções espaciais.
Fonte: Albuquerque, 2015.

(a) (b)

Figura 11: (a) Imagem da figura 10 quantizada em 2 bits. Valores de entrada variam entre 0 a 15. (b) quantizada em 2 bits. Valores de entrada variam entre 0 a 3.
Fonte: Albuquerque, 2015.

Figura 12: Imagem da figura 4 quantizada em 1 bits. Valores de entrada variam entre 0 ou 1.
Fonte: Albuquerque, 2015.

3.6 VIZINHANÇA E COORDENADAS ESPACIAIS

Existem técnicas de processamento de imagem que estão baseadas na relação entre os pixels de uma determinada localização, assim é necessária a definição da vizinhança para tal etapa.

Segundo Albuquerque (2015), dada uma imagem digital u, temos que cada pixel Uij terá 8 vizinhos, sendo dois na vertical, 2 na horizontal e 4 nas diagonais, sendo esta relação chamada de N8. A figura 13 ilustra como funciona a vizinhança de um pixel qualquer:

(i-1,j-1)	(i-1,j)	(i-1,j+1)
(i,j-1)	(i,j)	(i,j+1)
(i+1,j-1)	(i+1,j)	(i+1,j+1)

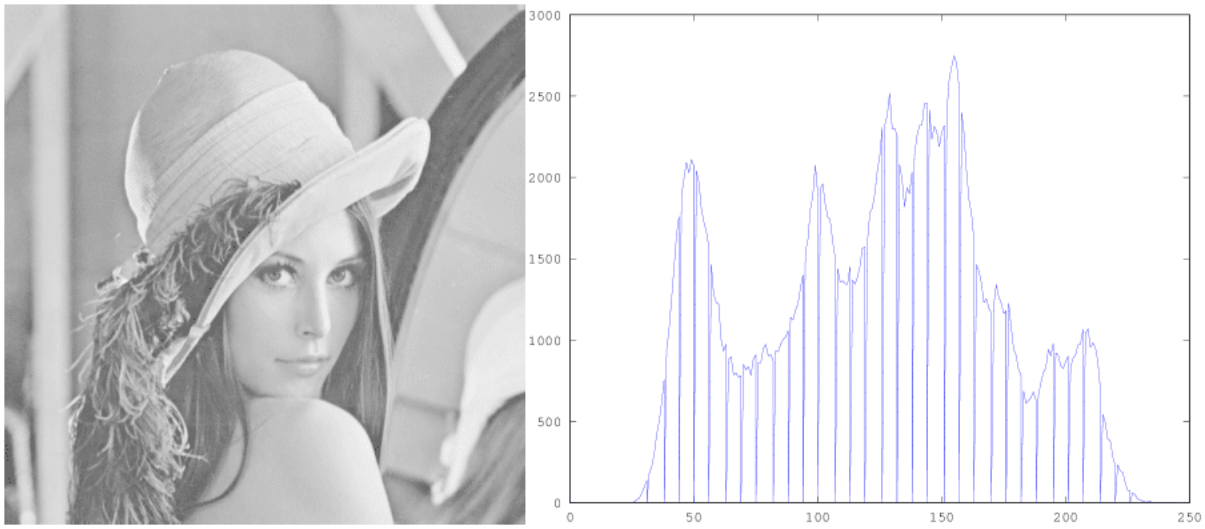
Figura 13: Vizinhança de um pixel N8.
Fonte: Albuquerque, 2015.

3.7 HISTOGRAMA

Um histograma de uma imagem digital pode ser definido como uma representação gráfica de um conjunto de dados que se encontram divididos em classes. Ela é composta por várias colunas onde a base de cada uma representa uma determinada classe e a sua altura representa a quantidade ou frequência com que o valor dessa classe ocorre naquela imagem, assim pode ser entendido que esta ferramenta é muito útil para se ter um entendimento aprofundado dos principais elementos e características contidos na imagem analisada. O histograma possui a habilidade de definir a frequência, assim como também a quantidade de valores de um determinado pixel, sendo capaz de

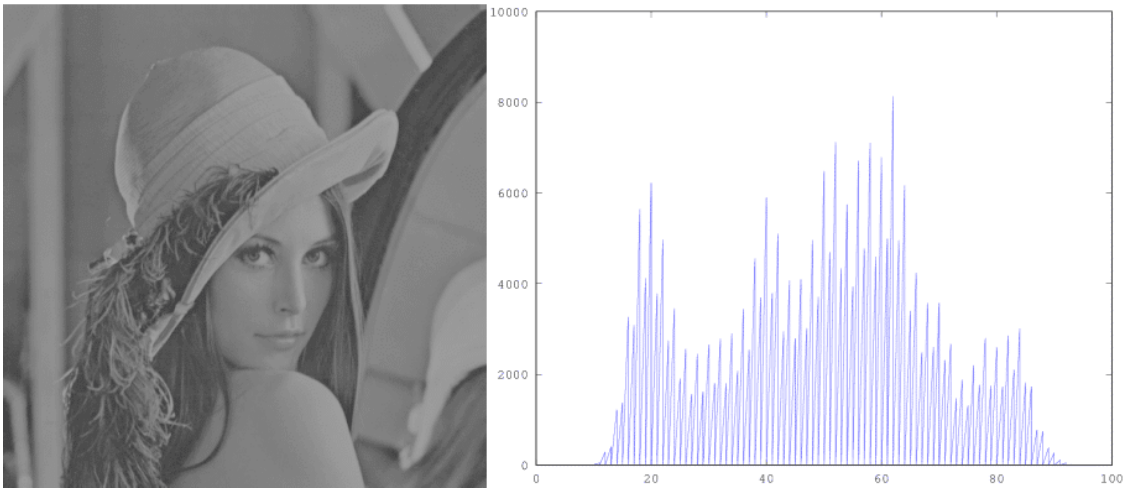
distinguir quando uma imagem está muito clara ou escura, guardando informações sobre o contraste de tal imagem (JO, 2015).

Como foi exposto acima, pode ser obter os níveis de contraste de uma imagem pelo uso de histograma. Nas figuras 14, 15 e 16 será demonstrada a mesma imagem, mas com níveis de brilho diferentes entre si para que seja possível observar melhor a atuação de um histograma.



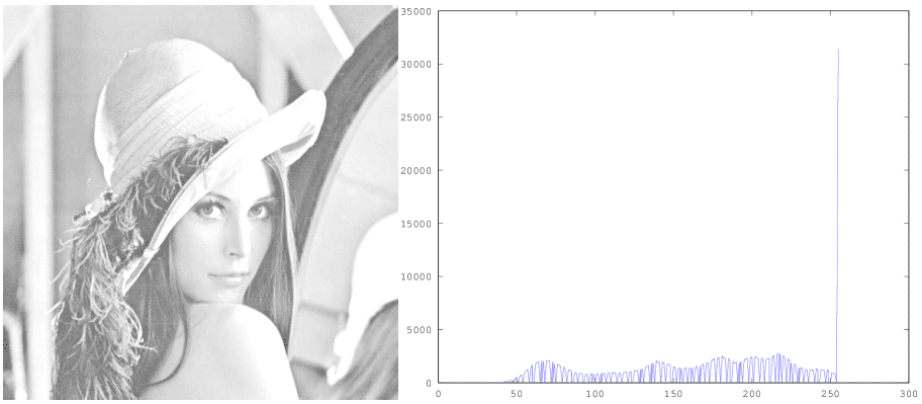
(a) (b)

Figura 14: A imagem “Lena” (a) e seu histograma respectivo (b).
Fonte: Jo, 2015.



(a) (b)

Figura 15: A imagem “Lena” com menos brilho (a) e seu histograma respectivo(b).
Fonte: JO, 2015.



(a) (b)

Figura 16: A imagem “Lena “com mais brilho (a) e seu histograma respectivo(b).
Fonte: JO, 2015.

O histograma é representado graficamente por dois eixos, onde o eixo x representa uma classe específica de pixel, enquanto que o eixo y representa sua respectiva quantidade. Pode ser observado que os pixels da figura 14 possuem valores numéricos na faixa de 25 a 240. O histograma da imagem fornece também dados na quantidade de pixels e seus atributos, por exemplo, temos que para os pixels com valor de 160 existem aproximadamente 2500 unidades.

Na figura 15, é possível observar que tal imagem possui pixels com valores abaixo de 90, isso se deve pelo fato de que como a intensidade do seu brilho é baixa. Enquanto que na figura 16 a imagem possui pixels na faixa de 45 a 255, porém com maior quantidade de no valor de 255, levando assim à uma imagem aparentemente mais pálida.

3.8 SEGMENTAÇÃO

De acordo com Albuquerque (2015), a ferramenta de segmentação tem como principal finalidade de dividir em seus componentes constituintes. O resultado obtido após este processo é o destaque dos objetos da imagem analisada em relação ao fundo. De um ponto de vista formal, uma imagem *U* idealmente segmentada pode ser compreendida como a união de *N* regiões *Ri* disjuntas entre si, conforme a equação 8:

$$u = \bigcup_{i=1}^{N} R_i \quad \text{e} \quad R_i \cap R_j = \emptyset, \text{ se } i \neq j.$$

eq.(8)

Um fato em relação a este processo seria que para que o mesmo consiga obter um bom desempenho, a imagem precisa ter uma boa definição. Com isso se tem em mente de que necessidade de uma definição de todos os objetos e do fundo de qualquer

imagem que for processada é uma dificuldade comum da segmentação. Com isso é possível afirmar que esta etapa é a mais crucial do processamento de imagem, pois a saída fornecida é que será a avaliada e processada por etapas anteriores (ALBUQUERQUE; PERSECHINO, 2015).

O diagrama representado na figura 17 retrata de forma clara os procedimentos necessários para o processamento de imagem:

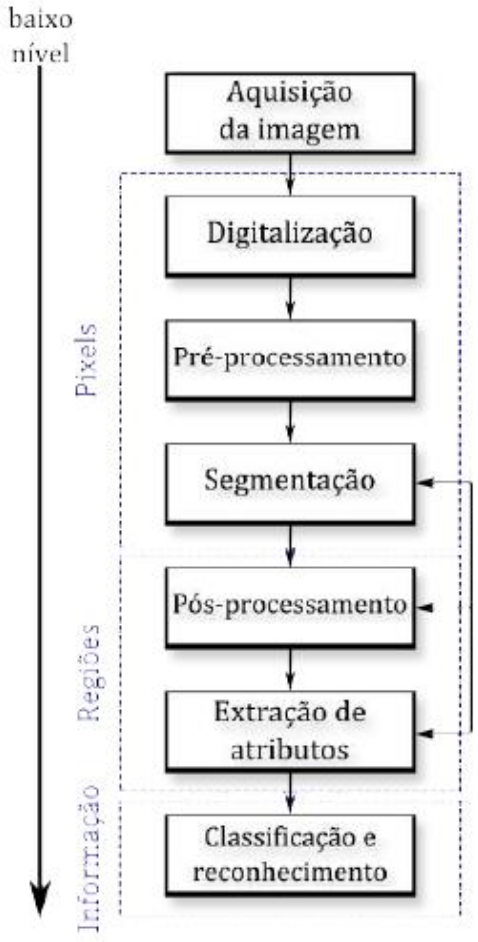


Figura 17: Diagrama mostrando as etapas de um sistema de processamento de imagens.
Fonte: Albuquerque; Persechino, 2015.

Existem duas abordagens básicas para segmentação de imagens, a primeira abordagem está associada às técnicas baseadas em igualdade entre pixels, enquanto que a outra se relaciona com técnicas baseadas em descontinuidade entre pixels. A primeira delas, limiarização, faz uso da semelhança entre os valores dos pixels, enquanto que a segunda, detecção de bordas, faz uso de descontinuidades entre pixels (ALBUQUERQUE; PERSECHINO, 2015).

3.9 LIMIARIZAÇÃO

Como foi demonstrado anteriormente no decorrer do texto, que o histograma de uma imagem pode apresentar diversas informações sobre a distribuição dos valores de intensidade dos pixels que compõem tal imagem.

Podemos supor que o histograma de intensidade da figura 18 esteja relacionado a uma imagem *f(x,y)*, constituída de objetos claros sobre um fundo escuro, de tal forma que os pixels do objeto e do fundo possuam valores de intensidade reunidos em dois grupos considerados dominantes (modos). Um modo capaz de extrair os objetos do fundo é escolher um limiar representado pela variável T, que seja capaz de separar estes modos. Assim, qualquer elemento *(x,y)* na imagem em que $f(x, y) > T$ é chamado de ponto do objeto, e quando $f(x, y) < T$ é conhecido como ponto de fundo (GONZALES; WOODS, 2010).

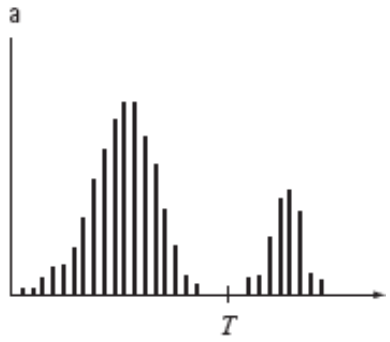


Figura 18: Histograma de intensidade sendo dividido por um limiar único.
Fonte: Gonzales, 2010.

A imagem segmentada *g (x,y)* é dada pela equação 9:

$$g(x,y) = \begin{cases} 1 & \text{se } f(x,y) > T \\ 0 & \text{se } f(x,y) \leq T \end{cases} \qquad \text{eq.(9)}$$

Vale ressaltar que é conveniente usar a intensidade 0 para fundo e 1 para objetos, porém qualquer valor pode ser utilizado para representar cada modo, deste que estes sejam distintos (GONZALES; WOODS, 2010).

Outra observação é que outra transformada pode ser aplicada na imagem:

$$T(u_{ij}) = \begin{cases} 0, & \text{se } u_{ij} \leq L \\ 1, & \text{caso contrário} \end{cases}$$

Onde o parâmetro L corresponde ao limiar de corte, dando assim origem ao nome da técnica Devido à forma da transformação T, esta operação é frequentemente chamada de binarização (ALBUQUERQUE; PERSECHINO, 2015).

Albuquerque e Persechino (2015) afirmam que a técnica de binarização funciona de forma eficiente desde que o histograma da imagem possua um perfil bimodal, pois isto representa o fato de que os valores de intensidade da imagem estão separados em duas classes distintas. A figura 20 demonstra a binarização sendo eficiente em uma imagem bimodal.

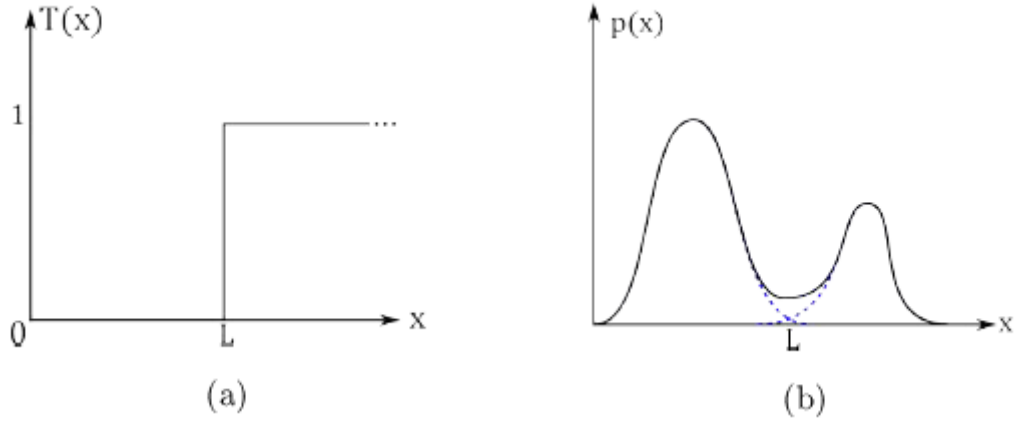


Figura 19: (a) Transformação global para binarização. (b) Transformação em um histograma de perfil não bimodal.
Fonte: Albuquerque; Persechino, 2015.

Segundo Albuquerque e Persechino (2015), a figura 19(a) mostra a transformação global para binarização, onde a mesma anula os pixels cujos valores estão abaixo do limiar L e conserva os demais. Assim o limar L é fácil de encontrar quando o histograma é bimodal, pois ele condiz com o valor de separação entre os dois pontos de máxima.

Albuquerque e Persechino (2015) concluem que imagens cujo histograma não possui um perfil bimodal, como mostrado na figura 19(b), são as mais comuns de serem trabalhadas e analisadas, sendo assim necessário a escolha de um ou mais limiares, esta que é uma tarefa pouco comum de ser executada.

Entre as mais variadas abordagens para determinação de um limiar ótimo, uma característica bastante comum é a busca por um limiar L0 que aperfeiçoe (maximize ou minimize) alguma função critério. A função critério pode ser entendida, por exemplo, o espaço entre duas distribuições (método do erro mínimo), à entropia do histograma (métodos entrópicos) ou à razão entre variâncias (ALBUQUERQUE; PERSECHINO, 2015).

3.10 DETECÇÃO DE BORDAS

O princípio do funcionamento da detecção de bordas está ligado com a descontinuidade entre os pixels. Uma borda pode ser entendida como uma transição abrupta no decorrer de certa direção na imagem. Assim, com o uso de filtros diferenciadores, é possível a evidenciação das bordas, que em conjunto da operação de limiarização, permitem a segmentação de uma imagem digital (ALBUQUERQUE; PERSECHINO, 2015).

A maior parte dos processos de detecção de bordas baseia-se na análise de descontinuidades do nível de cinza no pixel observado em relação aos seus vizinhos. Assim, a diferença pode ser estabelecida pela derivada do sinal no pixel, cuja definição clássica é dada pela equação 10, que resulta em f (x+1) – f (x), quando h=1, ou seja, a derivada é alcançada com a diferença (gradiente) entre o valor da tonalidade do pixel x+1 e o pixel x (ARTERO; TOMMASELLI, 2009).

$$f'(x) = \lim_{h \to a} \frac{f(x+h) - f(x)}{h} \qquad \text{eq.(10)}$$

No processo de detecção de bordas, é comum o uso de máscaras de convolução, construídas de modo a se obter uma proximidade da primeira derivada. Este é o caso dos operadores de Prewitt (PREWITT, 1970, apud TOMMASELLI; ARTERO, 2009) e Sobel (ROSENFELD e KAK, 1982, apud TOMMASELLI; ARTERO, 2009). O primeiro operador faz uso de máscaras de convolução que aparecem na Figura 16(a), determinando os gradientes em x e y, respectivamente, como Gx e Gy. Estas máscaras são a expressão da média dos valores da primeira derivada na vizinhança do pixel central (ARTERO; TOMMASELLI, 2009).

De acordo com Tommaselli e Artero (2009), é declarado que os pixels mais concentrados no centro possuem uma maior influência sobre o mesmo, o operador de Sobel define suas máscaras com valores maiores na região central, como é exposto na Figura 20(b).

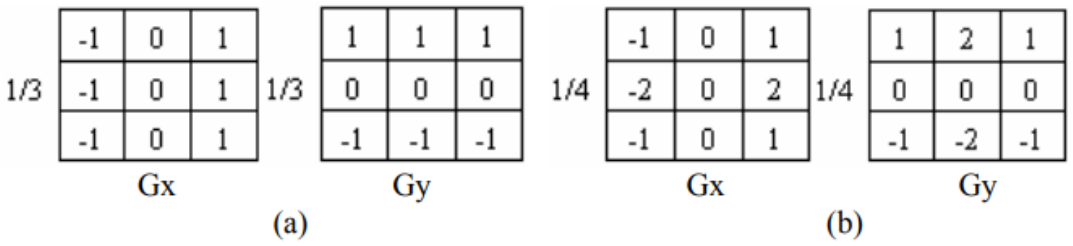


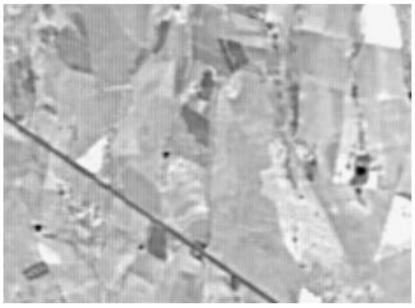
Figura 20: Máscaras para o cálculo de gradientes utilizadas na detecção de: a) Prewitt; b) Sobel.
Fonte: TOMMASELLI; ARTERO, 2009.

Segundo Tommaselli e Artero (2009), A partir de *Gx* e *Gy* obtidos por Prewitt ou Sobel, é realizável a obtenção da magnitude da borda e também a direção do gradiente em cada pixel da imagem, usando as equações 11 e 12, analisando que direção da borda = direção do gradiente + 90º.

$$magnitude = \sqrt{Gx^2 + Gy^2} \quad \text{eq.(11)}$$

$$\alpha = \arctan(\frac{Gy}{Gx}) \quad \text{eq.(12)}$$

Como as máscaras são planejadas baseadas em uma diferenciação no local, elas possuem a tendência de indicar baixas respostas em regiões homogêneas, ou seja, com baixa variação entre as tonalidades dos pixels e altas respostas em regiões de borda, que possuem uma alta variação entre as tonalidades dos pixels no local. Entretanto, quando aplicadas em regiões com muita variação, devido à presença de ruídos, os resultados acabam sendo prejudicados. Este obstáculo costuma ser superado com o processamento da imagem com um filtro passa-baixa, que realiza uma suavização no local, antes da operação de detecção de bordas. Outro caminho que pode ser tomado consiste no aumento do tamanho das máscaras utilizadas na detecção de bordas (ARTERO, 1999 apud TOMMASELLI; ARTERO, 2009), o que significa considerar uma maior vizinhança do pixel a ser observado, incluindo uma suavização no local. Porém, é necessário ponderar que um aumento exagerado no tamanho das máscaras pode diminuir a sensibilidade do detector em relação às pequenas variações de direção das bordas, sendo necessária a consideração do tamanho de máscara mais adequado aos variados casos. Uma escolha plausível nesta situação é a modificação dos tamanhos de acordo com o nível de ruído encontrados na região (TOMMASELLI; ARTERO, 2009).



(a)

(b)

(c)

Figura 21: a) Recorte da Imagem original, b) bordas detectadas com o operador de Prewitt; c) bordas detectadas com o operador Sobel.
Fonte: Tommaselli; Artero, 2009.

3.11 FILTRAGEM

Em qualquer imagem digital, sempre existe a possibilidade da ocorrência de ruídos, estes que podem ser entendidos como uma distorção indesejada que se encontra presente em uma determinada imagem. Para a eliminação deste problema, existem técnicas de processamento de imagem que visam a redução destes ruídos, conhecidas como filtragem. Os filtros são meios de transformação da imagem no domínio espacial (MENESES; ALMEIDA, 2012).

As figuras 22 e 23 demonstram como os ruídos podem estar presentes em uma imagem qualquer:

Figura 22: Imagem Lena em preto e branco – (a) imagem original, (b) imagem com ruído.
Fonte: Meneses; Almeida, 2012.

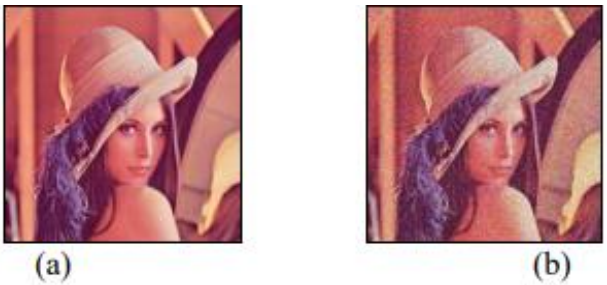


Figura 23: Imagem Lena colorida – (a) imagem original, (b) imagem com ruído.
Fonte: Meneses; Almeida, 2012.

Para que o processamento digital de imagens seja eficiente e consiga extrair informações e dados desejados de uma determinada imagem, muitas vezes é necessária

a retirada de qualquer tipo de distorção indesejada. Um dos métodos utilizados para que seja possível a remoção destes ruídos é a filtragem. Filtros possuem a finalidade de corrigir ou refinar os detalhes pertencentes à uma imagem. A correção seria a retirada das características indesejadas como ruídos e a melhoria dos elementos da imagem digital (FERNANDES; BARCELOS, 2012).

Segundo Meneses e Almeida (2012), para que seja possível a realização da etapa de filtragem, é necessário o conhecimento em relação ao tipo de filtro que se encaixa melhor para o resultado que se deseja obter. O tipo de um filtro, ou seja, a sua natureza é determinada pela sua configuração que, podem ser classificados em filtro passa-baixa, filtro passa-alta e filtro passa-bandas.

3.11.1 Filtros passa-baixas

O primeiro tipo de filtro que será exposto neste trabalho é o filtro passa-baixas, onde o mesmo tem a capacidade de remoção do sinal original, as componentes que possuem uma alta frequência, assim retirando ruídos, mas também sob penalidade de remoção de detalhes finos (ALBUQUERQUE; PERSECHINO, 2015).

A equação 13 representa um filtro passa-baixas:

$$|H(\omega 1, \omega 2)| = \begin{cases} 1; |r| \leq rc \\ 0; caso\ contrário \end{cases} \quad \text{eq.(13)}$$

Com base na equação 13, Albuquerque e Persechino (2015) opinam que a magnitude do filtro H é composta em um disco de raio rc que está localizado no centro da imagem.

A seguir, será ilustrada através das figuras 24 e 25 a atuação do filtro passa baixas em uma determinada imagem digital.

(a) (b)

Figura 24: Imagem original (a), e espectro discreto de potência localizado no centro da imagem (b).
Fonte: Albuquerque; Persechino, 2015.

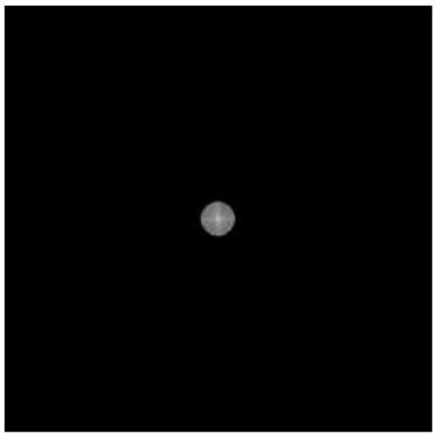



Figura 25: Espectro discreto de potencias da imagem filtrada por um passa-baixas ideal (a) e a imagem reconstruída, onde é possível notar-se a suavização das bordas e a consequente perda de detalhes finos (b).
Fonte: Albuquerque; Persechino, 2015.

Como é possível observar na figura 25, verifica-se que a atuação do filtro passa-baixas é capaz de conservar as componentes espectrais de baixa frequência e eliminando as mais altas, assim oferecendo à imagem uma suavização, a ondulação presente na imagem filtrada se deve ao fenômeno de Gibbs, devido às descontinuidades abruptas serem capazes de ser reveladas por meio de números infinitos de componentes espectrais. Como não é possível representar termos infinitos, acaba levando a existência de ondulações na imagem depois do processo de filtragem por passa-baixas (ALBUQUERQUE; PERSECHINO, 2015).

3.11.2 Filtros passa-altas

O filtro passa-altas é representado pela equação 14:

$$|H(\omega 1, \omega 2)| = \begin{cases} 1; |r| \leq rc \\ 0; caso\ contrário \end{cases} \quad \text{eq.(14)}$$

Em relação a este filtro, o seu disco delimitado por rc vai definir quais frequências internas relacionadas a ele serão anuladas, assim tornando evidentes regiões que se encontram com maior variação espacial, ou seja, com alta frequência. Com o uso deste filtro, alguns detalhes acabam sendo preservados, como é o caso de bordas, já que apresentam transições pouco intensas entre os tons de cinza, ao contrário de regiões com maior uniformidade espacial que é excluída (ALBUQUERQUE; PERSECHINO, 2015).

A figura 26 retrata o funcionamento do filtro passa-altas:

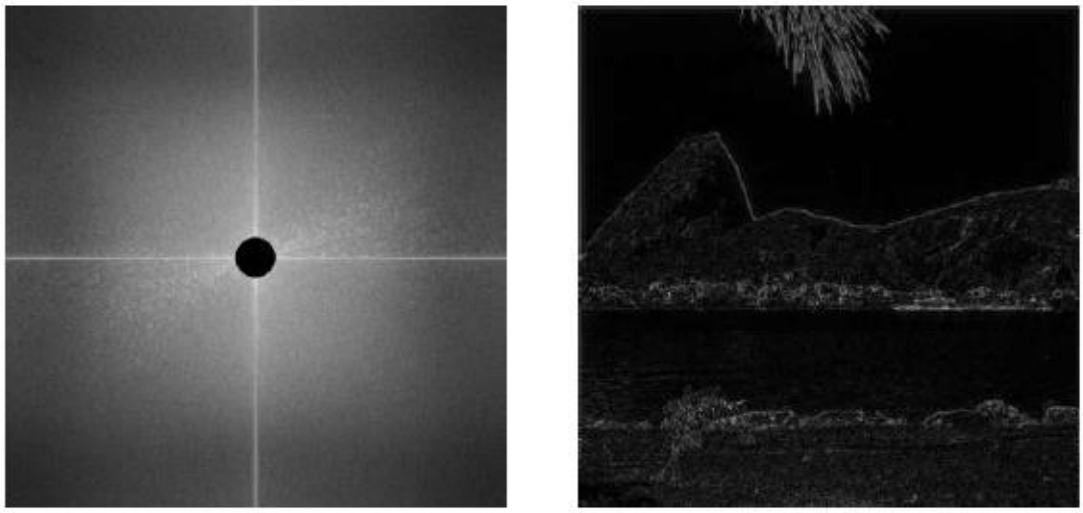


Figura 26: Espectro discreto da imagem digital filtrada por um passa-alta (a) e imagem reconstruída a onde é possível observar a evidenciação das bordas e destaque os detalhes (b).
Fonte: Albuquerque; Persechino, 2015.

3.11.3 Filtro passa-bandas

De acordo com Albuquerque e Persechino (2015), este filtro tem seu princípio de funcionamento diferente dos filtros passa-baixos e passa-altas, já que o mesmo é capaz de selecionar um período fechado específico de frequências, ou seja, de selecionar uma região em forma de anel no plano das frequências.

O filtro possui a forma de caso ideal mostrado na equação 15:

$$|H(\omega 1, \omega 2)| = \begin{cases} 1; rc1 \ \leq \ |r| \leq rc \\ 0; caso\ contrário \end{cases} \quad \text{eq.(15)}$$

Como este filtro possui a sua configuração em anel de largura, representado pela expressão rc1 - rc2, o filtro passa banda tem a finalidade de manter as características intermediárias do sinal filtrado, assim, a supressão de ruídos não se torna totalmente intenso, e não existe o detalhamento e evidenciação das bordas na imagem digital (ALBUQUERQUE; PERSECHINO, 2015).

A figura 28 mostra a atuação do filtro passa-bandas em uma imagem digital:

(a) (b)

Figura 27: Espectro discreto da imagem digital filtrado por um passa-bandas (a) e imagem reconstruída e filtrada, onde se manteve os detalhes e o intervalo fechado das frequências (b).
Fonte: Albuquerque e Persechino, 2015.

3.11.4 Filtro de Kalman

O Filtro de Kalman é amplamente estudado e aplicado na solução de diversos problemas, sendo o seu desenvolvimento matemático baseado em propriedades de Estimação e nas Equações Diferenciais de Riccati. A Filtragem de Kalman vem sendo aplicada em áreas tão diversas quanto: aeroespacial, navegação marítima, instrumentação de usinas nucleares, modelagem demográfica, astronomia, meteorologia, economia e indústria em geral. Este filtro é considerado por muitos um grande avanço da teoria de estimação do século vinte. As principais aplicações da filtragem de Kalman estão nos sistemas de controle modernos e na navegação e rastreamento de todos os tipos de veículos (SILVA, 2014).

O filtro de Kalman é um conjunto de equações matemáticas que constitui um processo recursivo eficiente de estimação, uma vez que o erro quadrático é minimizado. Através da observação da variável denominada "variável de observação" outra variável, não observável, denominada "variável de estado" pode ser estimada eficientemente. Podem ser estimados os estados passados, o estado presente e mesmo previstos os estados futuros. O filtro de Kalman é um procedimento aplicável quando os modelos estão escritos sob a forma espaço-estado. Além disso, o mesmo permite a estimação dos parâmetros desconhecidos do modelo através da maximização da verossimilhança via decomposição do erro de previsão. É a técnica de estimação de estado mais empregada, apesar de suas deficiências. O modelo do Filtro de Kalman Discreto (FKD) é descrito de acordo com o as equações 15 e 16:

$$\begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_{t-1} \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases} \quad \text{Eq.(15)}$$

$$\begin{cases} K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \\ \mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t) \bar{\Sigma}_t \end{cases} \quad \text{Eq.(16)}$$

Onde:

- s ϵ Rn é o vetor de estados;
- u ϵ RS é o vetor das entradas de controle;
- z ϵ Rm é o vetor de medições;
- a matriz n × n, A, é a matriz de transições de estados;

- B, n × l, é a matriz de coeficientes de entrada;
- a matriz C, m × n, é a matriz de observação;
- $\gamma \epsilon$ Rn representa o vetor de ruídos do processo e
- $\delta \epsilon$ Rm o vetor de erros de medição.

Vale ressaltar que os índices t e t− 1 representam os instantes de tempo atual e anterior respectivamente. E assim, o funcionamento do filtro realiza a predição e atualização de acordo com as propriedades estatísticas do ruído. Existe um modelo interno do sistema que é utilizado para a atualização e uma realimentação realiza as medições de laser. Os sistemas 2 e 3 descrevem as fases de predição e atualização respectivamente para o FKD (FAÇANHA; CARNEIRO; COSTA FILHO, 2013).

$$\begin{cases} s_t = A_t s_{t-1} + B_t u_t + \gamma_t \\ z_t = C_t s_t + \delta_t \end{cases} \quad \text{Eq.(17)}$$

Como foi visto anteriormente, o método de processamento de imagem digital é uma tecnologia que permite a extração de qualquer tipo de informação ou dados que estejam inseridos em uma imagem ou vídeo. Tal tecnologia foi muito útil neste trabalho, pois com ela foi possível detectar a velocidade dos veículos presentes em uma via e calcular suas velocidades de forma eficiente e rápida.

4 SISTEMAS ESPECIALISTAS

Sistemas especialistas podem ser definidos como sistemas de computadores com a habilidade de efetuar operações que se fossem realizadas pelos humanos, seriam considerados complexos e inteligentes, sendo que ainda possuem a capacidade de gerar explicações sobre a linha de raciocínio utilizada para a tomada de cada uma de suas decisões (SOLANGE, 2003, p.52 *apud* COSTA; SILVA, 2005, p.2).

Os sistemas especialistas são softwares que conseguem realizar a manipulação de conhecimento, dados e informações de maneira eficiente e inteligente. Eles são implementados para resolver problemas que exigem uma grande quantidade de inteligência e especialização. Todo sistema especialista é composto por uma base de conhecimento onde nela é armazenado todo o conhecimento humano sobre uma área especifica onde o sistema está destinado a operar (COSTA; SILVA, 2005, p.2).

Para Mendes (1997), sistemas especialistas são baseados em informações e construídos com regras capazes de reproduzir o conhecimento de um perito em uma determinada área, sendo utilizados para resolverem problemas em domínios singulares.

Sistemas especialistas são programas que se permitem um computador solucionar problemas vinculados a uma determinada área, por meio de armazenamento e sequenciamento de informações e autoaprendizagem (XAVIER PY, 2009 *apud* SPIRLANDELLI *et al.*, 2011).

Sistemas especialistas podem ser compostos por um nível de conhecimento análogo ao de especialistas, oferecendo resultados confiáveis, dando ao usuário a possibilidade de checar quais respostas possuem uma maior eficiente e importância. Os usos desses softwares proporcionam mais agilidade no trabalho exercido, assim como diminuir a ocorrência de erros (SPIRLANDELLI *et al*., 2011).

Com base nos conceitos expostos, os sistemas especialistas podem ser definidos como programas de computador capazes de solucionar problemas de uma determinada área de forma inteligente e rápida, fazendo uso de uma base de dados que seriam do mesmo nível de conhecimento de um especialista humano em uma determinada área.

4.1 VANTAGENS DO USO DE UM SISTEMA ESPECIALISTAS:

De acordo com Mendes (1997), as vantagens vindas da utilização de sistemas especiais são várias, dentre elas, podem ser destacadas:

a. O uso de um sistema especialista pode fornecer uma melhoria na produtividade e desempenho de seus usuários, já que o mesmo é capaz de achar melhores respostas e resolver problemas de maneira eficiente e rápida;

b. Sistemas especialistas podem reduzir a dependência que as empresam tem em situações críticas como, por exemplo, a falta de um especialista. Existem diversos motivos que podem levar à ausência de um funcionário com um cargo muito importante. Para evitar este tipo de dependência, é possível registrar o conhecimento de empregados nos sistemas especialistas, assim diminuindo o grau de dependência entre a empresa e a presença física de seus empregados;

c. Sistemas especialistas podem ser muito úteis para o treinamento de grupos de pessoas, de forma rápida e agradável. Podem servir para coleta de informações sobre o desempenho dos treinandos após o treinamento, sendo capaz de obter informações para reformulação das lições e assim obter o melhor desempenho para o treinamento.

4.2 BASE DE CONHECIMENTO

A figura 28 representa a maioria da estrutura dos sistemas especialistas.

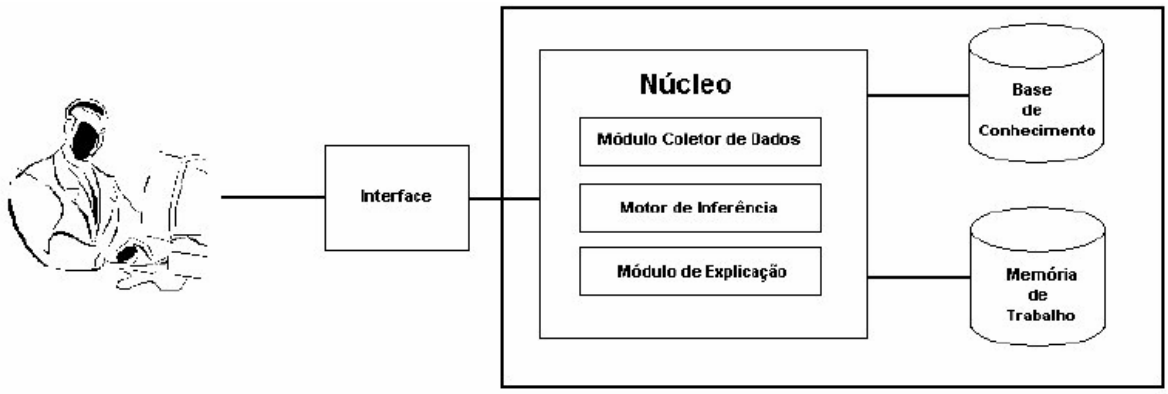


Figura 28: Representação de Estrutura de Sistema Especialista.
Fonte: Costa; Silva, 2005.

A base de conhecimento pode ser definida como um local de armazenamento de informações relacionado a um determinado domínio de atuação do sistema especialista. Essas informações podem ser representadas em várias linguagens, tais como regra de produção, lógica ou redes semânticas. Existem situações onde os dados existentes na base de conhecimento não são suficientes para que o sistema ache a resposta para o problema, por isso é recomendado que os sistemas especialistas possuíssem mecanismo

de inserção de dados, onde um usuário pode inserir informações para que o SE seja capaz de fornecer uma resposta satisfatória (COSTA; SILVA, 2005, p. 39).

A base de conhecimento é onde se armazena todo o conhecimento fornecido por um especialista da área, sendo construído conforme os objetivos e soluções que almejam ser alcançados pelo sistema especialista (FLORES, 2003 apud XAVIER PY, 2009).

De acordo com Xavier (2009, *apud* SPIRLANDELLI, 2011, p. 5), a base de conhecimento pode ser representada de 3 formas principais: lógicas, redes semânticas ou quadros.

- Lógica: É a forma mais básica para a maior parte dos formalismos de representação de informação, sendo de forma explicita como nos sistemas especialistas baseados na linguagem Prolog, ou de forma mascarada nas representações especificas que podem ser interpretadas como proposições ou predicados lógicos (MINSKY, 1975 *apud* XAVIER PY, 2009).
- Redes Semânticas: são definidos como um conjunto de nodos conectados por arcos. Nodos representam objetos, predicados, classes, palavras de cada linguagem e entre outras interpretações, enquanto que os arcos representam as relações binarias entre eles (XAVIER PY, 2009).
- Quadros: frames e roteiros foram inicialmente usados para que fosse possível a expressão das estruturas internas dos objetos, com o intuito de manter a possibilidade de representar heranças de propriedades como as redes semânticas (MINSKY, 1975 apud XAVIER PY, 2009).

4.3 NÚCLEO DO SISTEMA ESPECIALISTA

O núcleo de qualquer sistema especialista é composto por 3 módulos, podendo ser independente ou não.

4.3.1 Módulo Coletor de Dados

Segundo Costa e Silva (2005), este tipo de módulo possui a função de interagir com o usuário, fazendo perguntas necessárias para o desenvolvimento do raciocínio. Um exemplo seria um sistema que faz perguntas para um paciente sobre quais sintomas ele está sentindo no momento, para que assim o sistema especialista consiga formular um diagnóstico de forma rápida, segura e eficiente.

4.3.2 Módulo de Explicação

Este módulo tem a função de fornecimento de explicações de alto nível, justificando o motivo de ter chegado à uma determinada resposta ou de porque está fazendo uma pergunta qualquer para o usuário (COSTA; SILVA, 2005).

4.3.3 Motor de Inferência

O módulo de inferência tem relação com a base de regras enviando e recebendo novas informações do sistema, desta forma sendo capaz de adquirir novos aprendizados. Tem o dever de enviar dados destinados aos usuários em forma algorítmica, permitindo a conversa com a interface do usuário e com a base de conhecimento (SPIRLANDELLI *et al*., 2011).

Para Xavier Py (2009) as características de um motor de inferência são: método de raciocínio, estratégia de busca, resolução de conflito e representação de incerteza.

De acordo com Xavier Py (2009), existem dois modos de raciocínio aplicados para as regras de produção: encadeamento progressivo (*foward chaining*) e encadeamento regressivo (*backward chaining*).

No encadeamento progressivo, também conhecido por encadeamento dirigido por dados, a parte que está na esquerda da regra é comparada com a descrição da situação atual, esta que se encontra contida na memória de trabalho. No momento em que as regras são capazes de satisfazer a descrição tem a sua parte direita executada, assim realizando a introdução de novos fatos na memória de trabalho (XAVIER PY, 2009).

No encadeamento regressivo, a forma como o sistema trabalha é monitorada por uma lista de objetivos. Um objetivo pode ser realizado diretamente por um elemento da memória de trabalho, ou podem existir regras que possibilitam a inferência de alguns dos objetivos correntes, ou seja, que possam conter uma descrição de tal objetivo em suas partes direitas (XAVIER PY, 2009).

A partir do momento em que é definido o tipo de encadeamento, o motor de inferência tem a necessidade de realizar uma busca para guiar a pesquisa na memória de trabalho e na base de regras (XAVIER PY, 2009 *apud* SPIRLANDELLI *et al*., 2011).

Depois que a busca é feita, o motor de inferência possui um conjunto de regras que satisfazem à situação do problema, também conhecido como conjunto de conflitos. Assim que o conjunto possuir um estado vazio, o sistema especialista tem a sua execução terminada. Quando o mesmo ainda possuir algum estado, é necessária a

escolha de regras que serão executadas, assim como sua ordem de execução (SPIRLANDELLI et al., 2011).

4.4 MEMÓRIA DE TRABALHO

A memória de trabalho de um sistema especialista pode ser definida como uma área destinada ao armazenamento de dados temporários. Todas as respostas que são fornecidas pelo usuário também são armazenadas, assim como todos os procedimentos executados e as respostas finais (COSTA; SILVA, 2005).

A memória de trabalho é uma ferramenta importante durante a atuação de um sistema especialista, pois é nela que o motor de inferência realiza a etapa de comparação, analisando os dados recebidos com as informações presentes no banco de conhecimento, permitindo a tomada de decisão pelo motor de inferência (SPIRLANDELLI et al., 2011).

Para Costa e Silva (2005), o uso da memória de trabalho traz diversas vantagens, tais como:

1. Evitar a mesma pergunta seja feita para o usuário;
2. Fornece ao usuário a linha de raciocínio utilizada para a elaboração das respostas finais obtidas;
3. Evita que buscas de informações se repitam na base de conhecimento para a obtenção de conclusões intermediárias.

4.5 INTERFACE

A interface é uma forma de comunicação do usuário com o sistema especialista. É pela interface que o sistema especialista faz perguntas para o usuário e assim recebe suas respectivas respostas. A partir do momento que o sistema especialista consegue resolver o problema, ele expõe por meio da interface o raciocínio utilizado para solucionar o problema, assim como também suas conclusões. Vale ressaltar que o sistema especialista deve utilizar uma linguagem natural e de fácil entendimento para que a comunicação com o usuário seja eficiente, por isso é recomentado que o sistema faça uso de imagens, vídeos, textos, áudios ou animações (COSTA; SILVA, 2005).

4.6 CONHECIMENTO

Conhecimento pode ser entendido como uma informação que é armazenada ou entendida tanto pelo ser humano quanto a máquina, que os auxiliam a interpretar,

analisar, compreender e responder de forma apropriada ao meio em que se encontram (FISCHLER; FIRSCHEIN, 1987 *apud* FERRARI, 2005).

4.7 AQUISIÇÃO DO CONHECIMENTO

Todo ser humano durante a sua vida acumula todo tipo de conhecimento, assim permitindo que hajam de maneira inteligente e possam interagir com o meio em que se encontram. Para que o aprendizado ocorra de maneira correta, é necessário um ciclo de processamento das informações, indo desde a coleta até o armazenamento definitivo no cérebro. Lembrando que o aprendizado varia para cada ser humano (RABUSKE, 1998 apud FERRARI, 2005).

Para que fosse possível a realização da aquisição do conhecimento, foram necessários o desenvolvimento de diversas técnicas para sistematizar e automatizar tal processo. Tais técnicas podem ser classificadas em manuais, semiautomáticas e automáticas. As mais utilizadas são as técnicas manuais, já as automáticas consistem em um processo onde o conhecimento é adquirido de forma autônoma, sem ou com pouca interferência humana, enquanto que as semiautomáticas são usadas em conjunto com os manuais. (COSTA; SILVA, 2005).

4.7.1 Técnicas Manuais

De acordo com Rezende (2002 *apud* COSTA; SILVA, 2005), a maior parte das técnicas se baseia na psicologia e análise de sistemas. Nessas técnicas a atuação principal é do engenheiro de conhecimento, este que é responsável por adquirir o conhecimento do Especialista e de outras fontes para assim codifica-lo na base de conhecimentos.

A primeira técnica manual é a baseada em descrições ou imersão na literatura, nela o engenheiro de conhecimento realiza um estudo elaborado sobre a área onde o problema está inserido para conseguir um conhecimento inicial sobre o assunto. Esta etapa é importante, pois facilita em futuras entrevistas com o Especialista, deixando o diálogo com o usuário mais acessível, eficiente e rápido (COSTA; SILVA, 2005).

A segunda técnica é a baseada em entrevistas feitas com o Especialista, onde as informações são coletadas e posteriormente analisadas para extrair todo o conhecimento desejado (COSTA; SILVA, 2005). Existem dois tipos de entrevistas: não estruturadas e estruturadas:

- Não estruturada: Entrevista visada para obter uma visão geral do domínio onde o problema está inserido.

- Estruturada: São entrevistas formais, focando na aquisição de conhecimentos específicos sobre o domínio, com perguntas feitas de forma cuidadosa para garantir uma coleta de dados eficiente.

4.7.2 Técnicas semiautomáticas

As técnicas manuais envolvem um maior número de pessoas, tais como engenheiros, especialistas e programadores, assim tendendo a se tornarem mais problemáticas durante o seu desenvolvimento. A etapa que mais ocorrem ruídos é na transmissão dos conhecimentos adquiridos pelo especialista para o código fonte implementado pelos programadores. Uma solução capaz de viabilizar estes problemas são as técnicas semiautomáticas, que consistem no uso de ferramentas computacionais que oferecem suporte para o engenheiro do conhecimento e codificação da base de conhecimento (COSTA; SILVA, 2005).

Com o uso destes programas, o engenheiro de conhecimento e o especialista obtêm respostas mais rápidas, pois como o número de pessoas envolvidas é menor, haverá menos ruídos e ainda ocorre uma aceleração no desenvolvimento na base de dados (COSTA; SILVA, 2005).

4.7.3 Técnicas automáticas

Existem várias técnicas que estão inseridas na inseridas na área de absorção de conhecimento automático, dentre eles podem ser citadas a mineração de dados e o aprendizado de máquina, tais como redes neurais, árvores de decisão e etc (COSTA; SILVA, 2005).

Como foram expostos nos tópicos anteriores, os sistemas especialistas possuem a finalidade de simular o raciocínio de um profissional pertencente a alguma área especifica. Como podem ser aplicados em diversas situações, houve a oportunidade de utilizar um sistema especialista para delimitar quando foi necessário realizar uma otimização no tempo de ciclo verde do semáforo inteligente.

5 CONTROLADORES

ProMotion (S.D) define que os controladores lógicos programáveis (também conhecidos como CLP) são adventos da automação industrial e possuem a função de controlar processos industriais ou parte deles por meio de algoritmos programáveis de controles específicos. Entretanto, esses dispositivos não atuam sozinhos, pois precisam de outros coadjuvantes, como atuadores e sensores.

Para Coimbra (2009), os controladores lógicos programáveis são dispositivos eletrônicos responsáveis pelo controle de processos e utilizados em diversas áreas, tais como robótica e automação industrial.

O Controlador lógico programável é um dispositivo eletrônico capaz de armazenar instruções para a implementação de funções de controle, tais como sequência lógica, temporização e contagem, além de efetuar operações lógicas e aritméticas, manipulação de dados e comunicação em rede (COSTA, 2006).

O surgimento dos primeiros controladores está registrado no século 20, quando o matemático engenheiro James Watt projetou um regulador centrífugo para controlar velocidade das máquinas a vapor, na época da revolução industrial. Os controladores ainda eram totalmente operados de forma manual, somente entre 1915 e 1930 surgiram os controladores proporcionais e os registradores gráficos montados em campo. Após esse período, foram desenvolvidos os controladores de ganho ajustável, devido à necessidade do envio de informações para uma central industrial (PROMOTION, S.D).

No final dos anos 40 com o impulso produzido pelos transistores, houve o surgimento dos controladores eletrônicos analógicos e a transmissão de sinais em corrente. Na década de 50, foi criado o circuito integrado, quando foram usados os primeiros sistemas de controle por computador. Ainda no final dos anos 50, começaram a serem utilizados também os padrões de transmissão de sinais analógicos e digitais (PROMOTION, S.D).

Os Controladores Lógicos Programáveis surgiram no final da década de 60, sendo criados devido a uma solicitação da General Motors nos Estados Unidos, que estava a procura de uma alternativa que fosse capaz de simplificar a atualização dos painéis elétricos de comando, processo que consumia tempo. Sua função inicial não difere muito da atual, pois os controladores realizam o aperfeiçoamento e expansão das

possibilidades de controle de máquinas, autômatos e outros sistemas presentes nas indústrias. Com o avanço tecnológico que houve com o passar dos anos, várias funções foram atribuídas a estes dispositivos, e um número cada vez maior de empresas começou a oferecer modelos de controladores lógicos, com diferentes características e para aplicações específicas (PROMOTION, S.D).

De acordo com ProMotion (S.D), com o uso dos microprocessadores na década de 70, os Controladores Lógicos Programáveis (CLP) passaram a ser utilizados em várias aplicações para automação de processos industriais e não industriais.

Os controladores lógicos programáveis (CLPs) são hoje a tecnologia de controle de processos industriais mais amplamente utilizada. Um CLP é um computador industrial que pode ser programado para executar funções de controle, como é ilustrado na figura 30. Esses controladores reduziram de forma drástica a afiação usada obrigatoriamente em circuitos de controle convencional a relé, além de apresentar outros benefícios, como a facilidade de programação e instalação, controle de alta velocidade, compatibilidade de rede, verificação de defeitos e conveniência de teste e alta confiabilidade (PETRUZELLA, 2014).

Figura 29: Representação ilustrativa de um CLP.
Fonte: Petruzelli, 2014.

5.1 VANTAGENS DOS CLPS

De acordo com Petruzelli (2014), controladores lógicos programáveis oferecem diversas vantagens, tais como:

- **Maior confiabilidade:** Uma vez escrito e testado, o programa pode ser facilmente transferido para outros CLPs. Como toda a lógica está contida em sua memória, não há chance de cometer erro lógico na fiação. Outro detalhe é em relação à fiação, que embora ainda seja necessária para conectar os dispositivos de campo, torna-se menos volumosa.

- **Mais flexibilidade:** É mais fácil criar e modificar um programa em um CLP do que ligar e religar os fios conectores em um circuito. Com um CLP, as relações entre as entradas e as saídas podem ser determinadas pelo usuário do programa. Os fabricantes de equipamentos podem atualizar o sistema simplesmente enviando um novo programa e usuários finais podem modificá-lo no campo ou providenciar segurança de acordo com as características do equipamento, como travas e senhas.

- **Menor custo:** Os CLPs foram projetados para substituir o controle lógico por relé e a redução de custos tem sido tão significativa que tornou tal tecnologia obsoleta, exceto para aplicações de potência.
- **Capacidade de comunicações:** Um CLP possui a capacidade de comunicação com outros controladores para realizar funções como supervisão do controle, coleta de dados, dispositivos de monitoramento e parâmetros do processo, além de baixar e transferir programas.
- **Tempo de resposta rápido**: O controlador Lógico Programável opera em tempo real, o que significa que qualquer evento que ocorre no campo resultará na execução de uma operação ou saída, de forma rápida e eficiente.

5.2 ARQUITETURA

A figura 30 abaixo representa o desenho esquemático da arquitetura de um CLP:

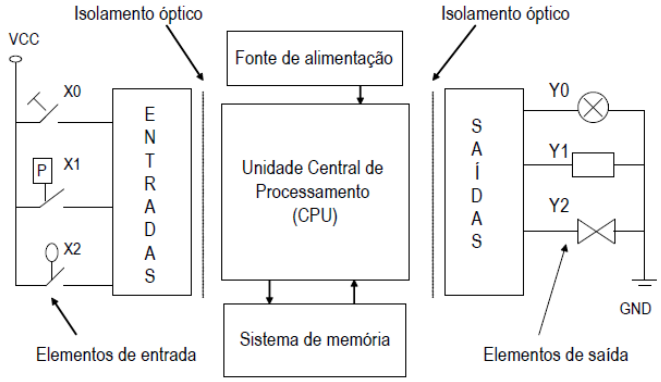


Figura 30: Arquitetura de um CLP.
Fonte: Costa, 2006

De acordo com da Costa (2006), a fonte de alimentação (figura 32) é o componente responsável por fornecer a energia elétrica adequada para o funcionamento da CPU e dos circuitos de entrada e saída.

Figura 31: Módulo de fonte de alimentação para CLP.
Fonte: Petruzelli, 2014.

A Unidade de Processamento Central (CPU) consiste em um microprocessador, para a implementação lógica e controle das comunicações entre os módulos e precisa de uma memória para armazenar as saídas fornecidas pelas operações lógicas executadas pelo microprocessador. A CPU (figura 33) controla todas as atividades e é projetada de modo que o usuário possa introduzir o programa desejado usando lógica *ladder* (PETRUZELLA, 2014).

Figura 32: Módulo de um processador para CLP.
Fonte: Petruzelli, 2014.

Para Da Costa (2006) os circuitos de entrada formam a interface pela qual os dispositivos enviam informações de campo para o CLP. As entradas podem ser digitais ou analógicas e são provenientes de elementos de campo, como sensores, botões e chaves fim-de-curso.

Os dispositivos de saída, tais como solenóides, relés, contatores, válvulas, luzes indicadoras e alarmes, estão conectados aos circuitos de saída do CLP. As saídas, de maneira similar às entradas, podem ser digitais ou analógicas e são geralmente isoladas

do campo por meio de isoladores galvânicos, como acopladores ópticos e relés (Costa, 2006).

De acordo com Da Costa(2006), o sistema de memórias é constituído por memórias ROM e RAM. O programa e os dados armazenados no sistema de memória são geralmente descritos a partir dos seguintes conceitos:

- Memória residente (ROM): Contém os programas que são permanentemente armazenados, que supervisionam e executam a sequência de operações, as atividades de controle e a comunicação com os dispositivos periféricos.
- Memória do usuário (RAM): Armazena o programa aplicativo do usuário, ou seja, o programa de aplicação para que o CLP possa atuar em campo.
- Memória de dados ou tabela de dados (RAM): São armazenados os dados associados ao programa de controle, tais como valores de temporizadores, contadores, constantes etc.
- Memória imagem das entradas e saídas (RAM): Área que reproduz o estado (ligado ou desligado) em tempo real de todos os dispositivos de entrada e saída integrados ao CLP.

5.3. FUNCIONAMENTO DE UM CLP

O CLP funciona de acordo com o programa que for armazenado em sua memória ROM, que executa um ciclo de *scan*, que consiste de uma série de operações realizadas de forma sequencial e repetida (COSTA, 2006). A Figura 34 apresenta, em forma de fluxograma, as principais fases do ciclo de *scan* de um CLP.

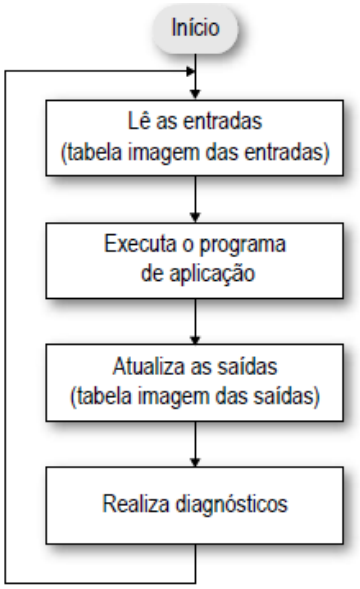


Figura 33: Fluxograma de um ciclo de scan de um CLP.
Figura 33. Fonte: Costa, 2006.

Na área da programação dizemos que cada informação programada é uma "linha". Assim, o *scan* realiza uma leitura da informação programada. Após essa leitura ou varredura, se houver uma segunda linha de programação, o *scan* passa para a próxima e faz a leitura da esquerda para direita e assim sucessivamente (ROCHA, 2013).

De acordo com Da Rocha (2013), o *scan* por diversas vezes vai realizar este tipo de varredura de forma sequencial, mesmo que não tenha nada de atual para executar. Isso quer dizer que o equipamento fica funcionando continuamente.

A primeira etapa em um ciclo de *scan*, ou seja, do funcionamento de forma sequencial e repetida de um CLP é a atualização das entradas, onde o CLP verifica os dispositivos de entrada quanto ao seu estado. O estado das entradas é atualizado e armazenado temporariamente em uma região da memória chamada "tabela imagem das entradas" (COSTA, 2006).

A segunda etapa é a execução do programa, o CLP verifica as instruções do programa de controle armazenadas na memória RAM e utiliza o estado das entradas armazenadas, assim determinando se uma saída será ou não energizada. O estado resultante das saídas é armazenada em uma região da memória RAM chamada "tabela imagem das saídas" (COSTA, 2006).

De acordo com a CLPREDES (2010), a programação do CLP é feita por meio de uma Ferramenta que pode ser um programador manual ou um computador com software de programação específico para controladores lógicos programáveis, ou seja, que faça uso da linguagem *ladder*.

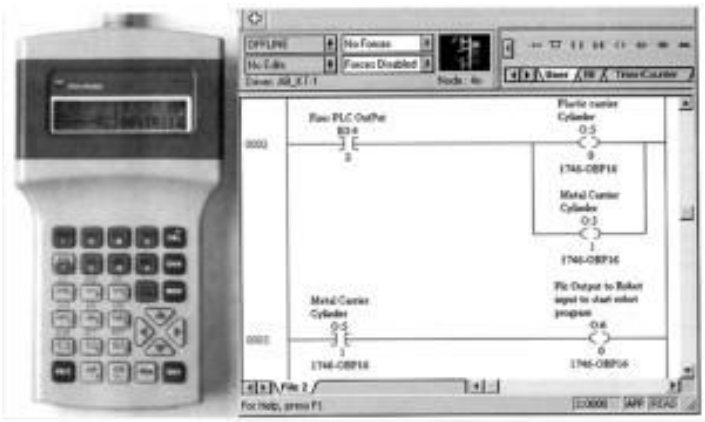


Figura 34: Controlador lógico programável e interface gráfica do programa.
Fonte: Clpredes, 2010.

Segundo da Costa (2006), a terceira etapa é a atualização das saídas, esta que é baseada nos estados dos bits da "tabela imagem das saídas", assim o CLP altera o estado seus circuitos de saída, que exercem controle sobre dispositivos externos.

A última etapa do funcionamento do CLP é a realização de diagnósticos, onde no final de cada ciclo de *scan*, a *CPU* examina as condições do CLP, ou seja, se ocorreu qualquer falha em algum de seus componentes internos, tais como fonte, circuitos de entrada e saída, memória e entre outros (COSTA, 2006).

5.4 CLASSE E APLICAÇÃO DOS CLP

Para que seja possível a classificação dos CLPs, vários atributos são levados em conta, que são a funcionalidade, número de entradas e saídas, custo e tamanho Físico (Figura 1.28). Desses fatores, a quantidade de entradas e saídas é a mais importante. Geralmente, o tipo nano é o de menor tamanho, com menos de 15 pontos de E/S. Logo depois vem os tipos micro (15 a 128 pontos de entrada e saída), os de porte médio (128 a 512 pontos de entrada e saída) e os de grande porte (mais de 512 pontos de entrada e saída) (PETRUZELLA, 2014).

De acordo com Petruzella (2014), fazer a combinação do CLP com a aplicação é um fator chave na seleção e existem três tipos de aplicação voltadas para tal tecnologia.

A primeira aplicação é a de terminal único, esta envolve apenas um CLP responsável por controlar um determinado processo, como mostra na figura abaixo. O dispositivo eletrônico deve ser uma unidade simples e não deve ser utilizado para se comunicar com outros computadores ou CLPs. (PETRUZELLA, 2014).

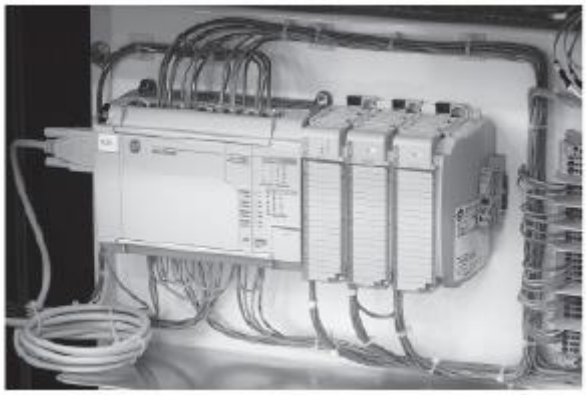


Figura 35: Aplicação de um CLP de terminal único.
Fonte: Petruzelli, 2014.

A segunda aplicação é a multitarefa, onde CLP é responsável pelo controle de vários processos, e a capacidade adequada de entradas e saídas é um atributo importante a ser levado em consideração neste tipo de instalação (PETRUZELLA, 2014).

A última aplicação é o gerenciador de controle, composta por um CLP controlando vários outros e precisa de um processador eficiente para se comunicar com outros CLPs e, possivelmente, com um computador. O gerenciador de controle supervisiona vários controladores, baixando programas que determinam funções para os outros (PETRUZELLA, 2014).

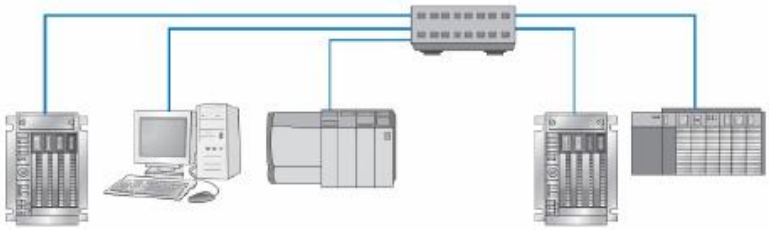


Figura 36: Ilustração de uma aplicação de CLP gerenciador de controle.
Fonte: Petruzelli, 2014.

Os controladores lógicos programáveis são essenciais em diversas áreas da indústria, devido a sua capacidade tecnológica de controlar outros dispositivos eletrônicos ligados ao mesmo. Tal tecnologia será muito importante neste trabalho, pois

será responsável por realizar o controle dos tempos de cada ciclo do semáforo inteligente.

6 PROCEDIMENTOS METODOLÓGICOS

A idéia do projeto desenvolvido pela equipe tem como principal objetivo de atuar como um semáforo inteligente capaz de analisar em tempo real, o fluxo de uma determinada via e melhorar os seus tempos, para que possa oferecer aos motoristas um fluxo de veículos mais rápido, organizado e eficiente.

Toda a implementação do projeto se deu por meio de programação. Então se optou pelo uso da linguagem *Python* junto com a biblioteca de computação visual *Opencv*, devido a sua facilidade de programação e a extensão de seus módulos. Foram necessários muitos estudos para que fosse possível compreender os princípios de funcionamento da biblioteca já citada. Depois que os estudos foram finalizados, outra decisão tomada foi na forma como o processamento de imagem iria afirmar quando o tráfego de uma via estivesse engarrafado ou com fluxo livre. Em um primeiro momento, decidiu-se detectar em tempo real a quantidade de veículos presentes naquela via, assim, a partir do momento que o sistema identificasse a presença de muitos veículos, o controlador aumentaria o tempo de ciclo verde para melhorar o fluxo de veículos.

Foi verificado posteriormente que ao utilizar tal método, a aplicação em si teve vários problemas que não puderam ser resolvidos, todos esses problemas serão explicados no decorrer deste capítulo. Assim, a melhor solução foi tentar detectar o estado do fluxo de via pela velocidade dos veículos em tempo real.

Depois que o objeto da detecção foi alterado, iniciaram-se vários estudos para o desenvolvimento de um código em *Python* integrado com a biblioteca de processamento de imagem *Opencv*, que fosse capaz de detectar a velocidade dos veículos em tempo real e fornecesse um *feedback* para o usuário, demonstrando através de telas em tempo real.

O capítulo de procedimentos metodológicos será divido em subtópicos, onde será explicada detalhadamente cada etapa que foi necessária ser finalizada para a conclusão deste trabalho. Cada procedimento do projeto pode ser visto na figura 37.

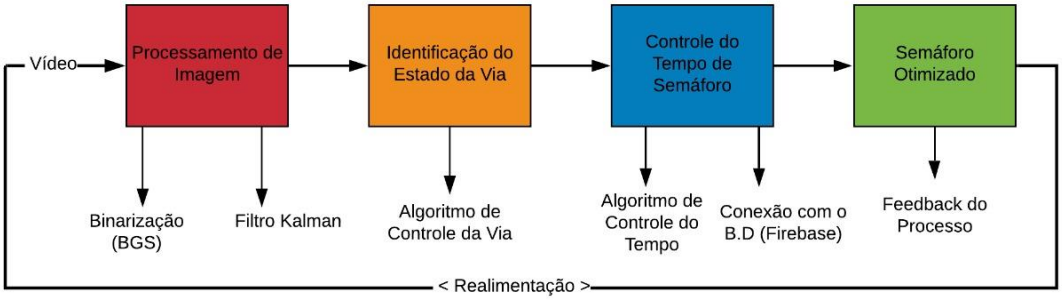


Figura 37: Fluxograma das etapas do projeto.
Fonte: Autores (2019).

6.1 COLETA DE DADOS CONTÍNUOS

Para o bom funcionamento do algorítmo, o mesmo precisava de dados contínuos ao longo do tempo para que fosse possível analisar e extrair informações da via, sendo o vídeo, a melhor opção. Durante as fases iniciais de teste, foram utilizados vídeos de tráfego da internet para validar o funcionamento do algorítmo de detecção de automóveis e velocidade. Posteriormente, decidiu-se capturar vídeos próprios de fluxo de veículos para os testes do trabalho.

Para a captura de imagens foi utilizada uma câmera de celular com uma resolução de 13 megapixels. Foram realizadas várias filmagens de aproximadamente 1 minuto de duração na Avenida Almirante Barroso entre a Travessa Timbó e Travessa Estrella, no horário de 6 a 8:30 da manhã em um dia útil da semana. Os vídeos foram feitos em um ponto onde era possível captar as imagens da avenida de forma clara, ou seja, em uma passarela localizada em frente a parada de ônibus que fornecia uma visão acima do fluxo de veículos.

Para minimizar interferências ou ruídos na realização das filmagens da avenida, foi utilizado um tripé de câmera para que as imagens não ficassem tremidas e consequentemente prejudicassem o processo de binarização da imagem digital.

No próximo sub-tópico, será realizada uma explicação detalhada da implementação e funcionamento do código, expondo as partes mais cruciais e suas determinadas funções, acompanhadas de imagens e fluxograma para facilitar o entendimento do mesmo.

6.2 IMPLEMENTAÇÃO DO CÓDIGO FONTE PARA O TRABALHO

Para iniciar a etapa de elaboração do script para o processamento de imagem, foi necessário determinar qual grandeza seria analisada durante a fase processamento de imagem digital para detectar quando a via estaria com fluxo bom ou congestionado. A primeira ideia foi de apenas detectar e contar a quantidade de veículos que se encontrariam presentes em um determinado trecho daquela via.

A principal falha dessa abordagem seria a incapacidade de calcular o estado da via pela contagem da quantidade de carros, pois além de ser imprecisa, ocasionava em situações onde seria incapaz de determinar o estado da via. Por exemplo, se a contagem de carros fosse elevada num curto intervalo de tempo, isso significaria que há muitos carros parados na tela (caracterizando congestionamento) ou que há muitos carros passando, mas em boa velocidade (caracterizando bom fluxo de veículos)? Assim, foi elaborada outra estratégia que fosse mais precisa para o projeto: Calcular e estimar a velocidade dos veículos para determinar a velocidade média da pista.

No script da figura 38 é determinado funcionamento do filtro Kalman para que o mesmo possa ser importado para o código principal e ser executado. O filtro Kalman é importante para este trabalho, pois como já foi explicado anteriormente, o seu papel será realizar o procedimento de filtragem para medir as grandezas realizadas no decorrer do tempo e gerar resultados que tendem a se aproximar dos valores reais das grandezas medidas, ou seja, o filtro calcula a velocidade estimada dos veículos detectados pelo sistema que seja satisfatória para se comparar com a velocidade real dos mesmos.

```
import numpy as np
from filterpy.kalman import KalmanFilter
from filterpy.common import Q_discrete_white_noise

f = KalmanFilter (dim_x=2, dim_z=1)
f.x = np.array([[2.],    # posição
                [0.]])   # velocidade
f.x = np.array([2., 0.])
f.F = np.array([[1.,1.],
                [0.,1.]])
f.H = np.array([[1.,0.]])
f.P *= 1000.
f.P = np.array([[1000.,    0.],
                [   0., 1000.] ])
f.R = 5
f.R = np.array([[5.]])
f.Q = Q_discrete_white_noise(dim=2, dt=0.1, var=0.13)

while True:
    f.predict()
    f.update(get_some_measurement())

    # saída
    x = f.x
```

Figura 38: Código para funcionamento do filtro de Kalman.
Fonte: Autores (2019).

O filtro de Kalman vai realizar a descrição da trajetória de todos os veículos que forem detectados pelo sistema, usando o plano cartesiano nas coordenadas do eixo x e y para determinar a localização de cada automóvel, assim a variável onde serão atribuídos tais valores será do tipo *array*, já que nela estarão contidos vários valores correspondentes para cada objeto detectado no vídeo.

O código da figura 39 possui a função de programar a ferramenta de *Background Subtraction*. Como explicado anteriormente, tal instrumento tem a função de detectar as mudanças em uma sequência de imagens, permitindo que o primeiro plano da imagem seja extraído para o processamento de imagem, assim auxiliando no reconhecimento e detecção de objetos inseridos em um conjunto de imagens. O método de *Background Subtraction* é muito útil também para a detecção de objetos em movimento, por isso a equipe decidiu utilizar e programar tal ferramenta.

O método de Backgroud Suctraction permite a detecção de objetos que se encontram em movimento em um conjunto de imagens. A detecção se dá pela conversão de todos os objetos em movimento para a cor branca, enquanto que o cenário e objetos parados recebem a cor preta, assim contribuindo de forma eficiente para a detecção dos veículos.

```
1  a = []
2  model_dir = ''
3  bgsMOG = cv2.createBackgroundSubtractorMOG2(history=2, varThreshold = 50, detectShadows=0)
4  if cap:
5      while True:
6          ret, frame = cap.read()
7          if ret:
8              fgmask = bgsMOG.apply(frame, None, 0.01)
9              # To find the contours of the objects
10             _, contours, hierarchy = cv2.findContours(fgmask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
11             # cv2.drawContours(frame,contours,-1,(0,255,0),cv2.cv.CV_FILLED,32)
12             try: hierarchy = hierarchy[0]
13             except: hierarchy = []
14             a = []
15             for contour, hier in zip(contours, hierarchy):
16                 (x, y, w, h) = cv2.boundingRect(contour)
17                 if w > 30 and h > 30:
18                     cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
19                     (x, y, w, h) = cv2.boundingRect(contour)
20                     xl = w / 2
21                     yl = h / 2
22                     cx = x + xl
23                     cy = y + yl
24                     a.append([cx, cy])
25                     # print(len(a))
26             cv2.imshow('BGS', fgmask)
27             cv2.imshow('Ori+Bounding Box', frame)
28             key = cv2.waitKey(100)
29             if key == ord('q'):
30                 break
```

Figura 39: Código para detecção de veículos.
Fonte: Autores (2019).

Em outras palavras, o método de *Background Subtraction* vai apenas detectar os objetos que se encontram em movimento no vídeo, enquanto que o cenário e tudo que se encontra estático será excluído do processamento de imagem, ou seja, nas filmagens realizadas, apenas os carros serão analisados e detectados, enquanto que o ambiente e os elementos estáticos presente nele serão totalmente excluídos.

Dentre os vários métodos para a segmentação, a binarização foi escolhida por se adequar melhor para a detecção de objetos em movimentação. Na binarização, todos os pixels que estão se movimentando no vídeo receberão a cor correspondente ao valor 1 (branco), enquanto que tudo que se encontra parado recebe o valor 0 (preto), desta forma, é exibida uma janela pelo comando “cv2.imshow” (figura 39, linha 26). A figura 40 mostra o funcionamento da binarização sobre a filmagem realizada.

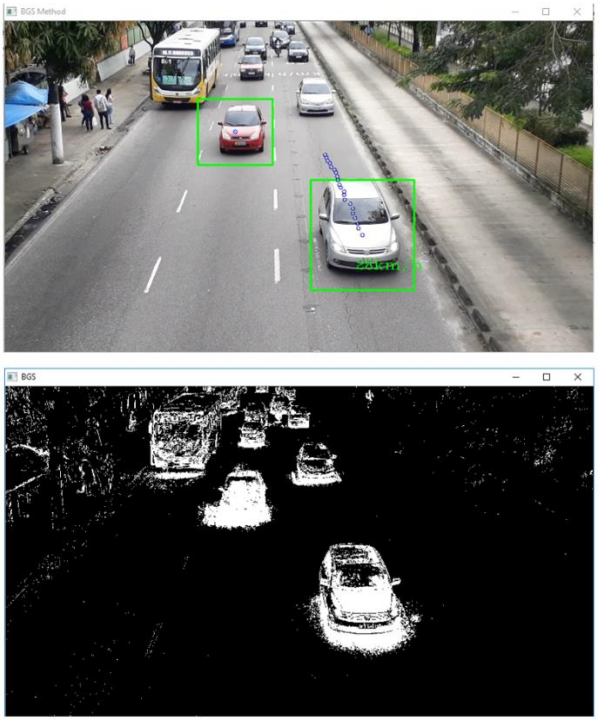


Figura 40: Binarização sendo feita sobre as filmagens da avenida.
Fonte: Autores (2019).

Vale ressaltar que o início do código principal se dá com a definição de alguns parâmetros necessários para a compilação do mesmo, sendo mais importante a taxa de sensibilidade do *Threshold*, ou seja, o quão sensível será o processamento de imagem para detectar os objetos nas imagens.

Uma dificuldade apresentada durante a fase de teste do sistema com as imagens filmadas foi que em alguns trechos do vídeo surgiam algumas interferências que ocasionavam em ruídos durante o processamento de imagem. Um dos tipos de ruído apresentado era em relação aos pedestres que atravessavam a rua fora da faixa de pedestre, pois como os mesmos se encontravam em movimento, o sistema acabava detectando os seus movimentos.

Outro ruído que presente no processamento das imagens era em relação aos ônibus, pois como são veículos grandes que possuem vários formatos devido às janelas, painéis, letreiros, escotilhas e etc., o sistema acabava detectando a borda de cada um desses elementos como se fossem carros, consequentemente gerando uma certa imprecisão para medir a velocidade média dos veículos na via.

O método que foi utilizado para contornar estes ruídos foi diminuir a sensibilidade da taxa de binarização de *Threshold*, assim o script ainda era capaz de detectar os veículos e ignorava as interferências que foram mencionadas acima.

```
def calculate_speed (trails, fps):
        # distance: distance on the frame
        # location: x, y coordinates on the frame
        # fps: framerate
        # mmp: meter per pixel
        dist = cv2.norm(trails[0], trails[10])
        dist_x = trails[0][0] - trails[10][0]
        dist_y = trails[0][1] - trails[10][1]
        mmp_y = 0.2 / (3 * (1 + (3.22 / 432)) * trails[0][1])
        mmp_x = 0.2 / (5 * (1 + (1.5 / 773)) * (width - trails[0][1]))
        real_dist = math.sqrt(dist_x * mmp_x * dist_x * mmp_x + dist_y * mmp_y * dist_y * mmp_y)
        return real_dist * fps * 250 / 3.6
```

Figura 41: Código para cálculo de estimativa da velocidade.
Fonte: Autores (2019).

O cálculo da velocidade (figura 41) é feito através da definição de uma matriz dentro da execução do vídeo. Essa matriz vai delimitar um vetor posição para aquele objeto que está sendo identificado no momento. Após a definição desse vetor posição o código utilizará uma definição de polegadas por *pixel* que é utilizada normalmente em reconhecimento de vídeo, para definir a distância entre determinados objetos dentro daquele vídeo especifico.

Esse cálculo utiliza a definição de 0.2 polegadas para cada pixel definido no vídeo, então se a resolução do vídeo for de 1080p, ele estará utilizando 1080 pixels na vertical e 1920 pixels na horizontal, sendo utilizados esses parâmetros para o cálculo da distância percorrida. Após a obtenção da distância o sistema utilizara também a taxa de quadros por segundo daquele vídeo para definir o tempo em que o objeto reconhecido utilizará para percorrer aquela distância durante a execução. Após obtermos a distância e o tempo podemos obter a velocidade daquele objeto.

```
blob['speed'] = [item for item in blob['speed'] if item != 0.0]
print ('========= speed list =========', blob['speed'])
ave_speed = np.mean(blob['speed'])
print ('========= ave_speed =========', ave_speed)
cv2.putText(frame, str(int(ave_speed)) + 'km/h', (blob['trail'][0][0]
arquivo = open('velocidades.txt', 'a')
arquivo.write('%s' %ave_speed + "\n")
arquivo.close()
```

Figura 42: Código para registrar a localização dos veículos e suas velocidades.
Fonte: Autores (2019).

No *script* mostrado na figura 42 é escrito no *log* do compilador a posição de cada veículo referente aos eixos x e y, através da variável *blob['trail']* (figura 42, linha 5). Outra variável muito importante que também é registrada no *log* é a *ave_speed* (figura 42, linha 3), pois ela funciona como um *array* onde são registradas todas as velocidades em quilômetros por hora que são detectadas durante o processamento de imagem. Após este processo, é criado um arquivo em txt com o nome de "velocidades", onde todos os valores que se encontram no *array ave_speed* são escritos e salvos em formato de uma linha e uma coluna, para que possam ser analisados e usados nas próximas etapas.

```
2 y = 0
3 soma = 0
4 arquivo_velocidades = open("velocidades.txt","r"
5 x = arquivo_velocidades.readlines()
6 for i in range(len(x)):
7
8     soma = float(x[i]) + soma
9     y= y+1
10    #print(soma)
11    vel_media =soma/y
12 arquivo_velocidades.close()
13 print("\n\n       MEDIA DAS VELOCIDADES")
14 print"\nSoma total das velocidades:", soma
15 print"\nQuantidade de veiculos:", y
16 print"\nMedia de velocidade na via:", vel_media
17
18 if vel_media < 30:
19    arquivo_vel_media = open("vel_media.txt","w")
20    arquivo_vel_media.write("1")
21    print ("\nEstado da via = 1")
22 if vel_media >= 30 and vel_media<= 45:
23    arquivo_vel_media = open("vel_media.txt","w")
24    arquivo_vel_media.write("2")
25    print ("\nEstado da via = 2")
26 if vel_media > 45:
27    arquivo_vel_media = open("vel_media.txt","w")
28    arquivo_vel_media.write("3")
29    print ("\nEstado da via = 3")
```

Figura 43: Código para calcular média de velocidade e definir estado da via.
Fonte: Autores (2019).

O *script* da figura 43 realiza uma leitura dos valores de cada linha que foram escritos no arquivo de texto "velocidades.txt" (figura 43, linha 4) e vai somar cada um deles e calcular a média geral dos mesmos e em seguida, será mostrado no log do compilador a soma total das velocidades, a estimativa da quantidade de veículos que foram detectados pelo sistema e a média da velocidade dos carros inseridos na via.

A partir do momento em que da etapa mencionada acima estiver concluída, o script vai executar a função que está ilustrada na figura 44:

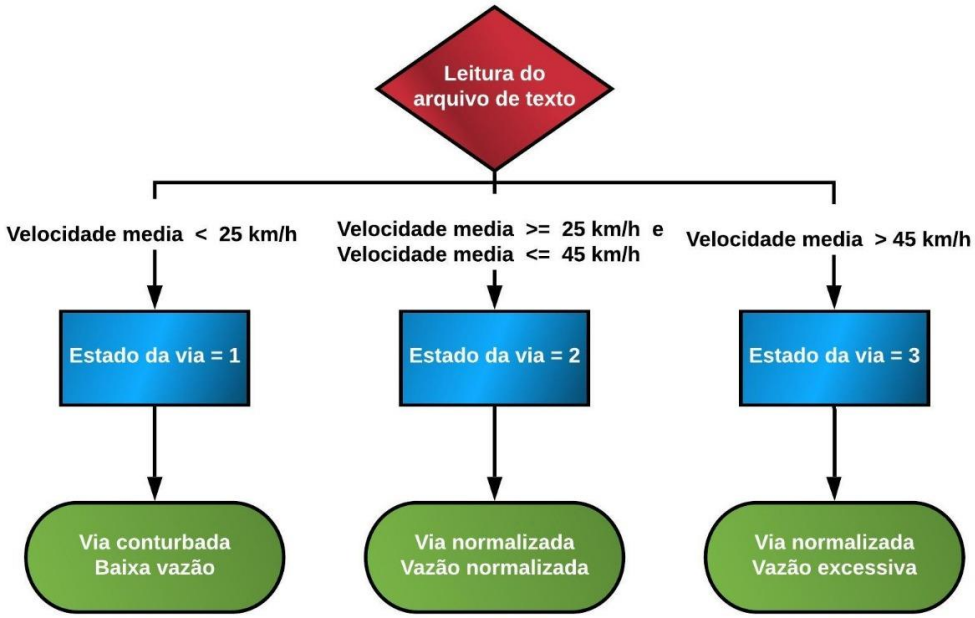


Figura 44: Fluxograma para determinar estado da via.
Fonte: Autores (2019).

Depois que o *script* realizar a média da velocidade dos carros inseridos na via com os valores que estavam inseridos no arquivo de texto, o mesmo será capaz de definir o estado da via de acordo com a velocidade média calculada, assim temos:

- Estado da via será 1 quando a via estiver conturbada.
- Estado da via será 2 quando a via estiver normalizada com vazão também normalizada.
- Estado da via será 3 quando a via estiver normalizada com vazão excessiva.

A próxima etapa foi utilizar a biblioteca *turtles* do *python* para realizar a simulação virtual de um semáforo de trânsito e assim demonstrar o seu funcionamento com base no valor que a variável do estado da via recebe.

Turtle é um módulo *Python* que oferece funcionalidades para fazer desenhos na tela, com comandos simples e rápidos. Esse módulo segue a ideia da linguagem de programação Logo, cuja ideia é de um robô (representado na tela por uma tartaruga), onde o usuário pode controla-lo através de comandos simples de movimentação (STUMM JR, 2011).

A ilustração do semáforo virtual e o seu script não são essenciais para o texto, por isso os mesmos serão adicionados na sessão de apêndice A deste trabalho.

```
1  if state_num == 0:  # Transition from state 0 to state 1
2      henry.color('darkgrey')
3      alex.color('darkgrey')
4      tess.color('green')
5      wn.ontimer(advance_state_machine)
6      arquivo_tempo_verde = open("tempo_verde.txt","r")
7      tempstr = arquivo_tempo_verde.read()
8      temp = float(tempstr)
9      print(" _____")
10     print ("|       ESTADO          |")
11     print ("|-----------------------|")
12     temp= round(temp,3)
13     print "| VERDE:   ", temp , " seg |"
14     time.sleep(float(temp))# set the timer to explode in 3000 milliseconds (3 seconds)
15     state_num = 1
16 elif state_num == 1:  # Transition from state 1 to state 2
17     henry.color('darkgrey')
18     alex.color('orange')
19     tess.color('darkgrey')
20     wn.ontimer(advance_state_machine)
21     print("| AMARELO:  5 seg       |")
22     time.sleep(5)
23     state_num = 2
24 elif state_num == 2:    # Transition from state 3 to state 0
25     henry.color('red')
26     alex.color('darkgrey')
27     wn.ontimer(advance_state_machine)
28     print("| VERMELHO: 5 seg       |")
29     print("|_____|\n")
30     time.sleep(5)
31     state_num = 0
```

Figura 45: Código do funcionamento do semáforo.
Fonte: Autores (2019).

A figura 45 mostra o script que determina o comportamento do semáforo virtual, onde o mesmo se baseia no princípio de máquina de estado e cada condição corresponde a um templo de ciclo do semáforo.

A primeira condição determina o tempo necessário para o ciclo verde do semáforo, onde o script lê o valor escrito no arquivo de texto “tempo_verde.txt” (figura 45, linha 6) e vai atribuir o mesmo para o ciclo, enquanto que as outras condições determinam o tempo fixo de amarelo e vermelho para cinco segundos.

Uma informação que vale a pena ser exposta foi que no início da implementação as otimizações eram realizadas de forma empírica, pois a ideia adotada inicialmente era de apenas de adicionar entre 5 a 10 segundos no tempo de sinal verde para realizar a otimização. Para deixar o sistema de adição do ciclo verde mais dinâmico, foi adotada uma nova ideia de implementação, onde foi escrito um algoritmo que atualiza o valor do tempo do ciclo verde, já com o acréscimo percentual da otimização baseado no monitoramento constante do fluxo de veículos.

A figura 46 demonstra o script de atualização do tempo verde.

```
arquivo_tempo_verde = open("tempo_verde.txt","r")
tempstr = arquivo_tempo_verde.read()
temp = float(tempstr)
print"-----------------------------------------------------------------"
temp= round(temp,3)
print"\n Tempo no txt:", temp
arquivo_vel_media = open("vel_media.txt","r")
x = arquivo_vel_media.read()
y = int(x)
print " Valor de y:",y,"\n"
if y == 1:
    print " Tempo de verde:", temp, "seg"
    temp10 = (float(temp)+float(temp)*float(10)/100)
    arquivo_tempo_verde = open("tempo_verde.txt","w")
    arquivo_tempo_verde.write(str(temp10))
    print " Tempo de verde + 10%:", temp10, "seg"
    db.child().update({"tempo_verde": temp10})
    print "\n\n Aguardando", (temp10 +4), "seg para a proxima execucao.\n\n"
    time.sleep(temp10 + 4)
if y == 2:
    print( "\n ********** Velocidade media compativel com o padrão ********** \n")
    print "                  Tempo de verde:" , temp, "seg"

    print( "\n                  Aguardando proxima execucao...\n\n")
    time.sleep(temp)
if y == 3:
    print " Tempo de verde:", temp,"seg"
    temp10 = (float(temp)-float(temp)*float(10)/100)
    arquivo_tempo_verde = open("tempo_verde.txt","w")
    arquivo_tempo_verde.write(str(temp10))
    db.child().update({"tempo_verde": temp10})
    print " Tempo de verde + 10%:", temp10,"seg"
    print "\n Aguardando " , (temp10 +10), " seg para a proxima execução.\n\n"
    time.sleep(temp10 + 10)
```

Figura 46: Código de atualização do tempo de verde.
Fonte: Autores (2019).

Com a implementação deste *script* o sistema é capaz de atualizar o tempo de ciclo de verde após cada execução do mesmo, onde será lido o valor que está escrito no arquivo “vel_media”, assim o script novamente trabalhará com laços de condição de *if-else*, onde podem ocorrer 3 situações que vão ocorrer de acordo com o valor que está escrito no arquivo já mencionado, como é mostrado na figura 47.

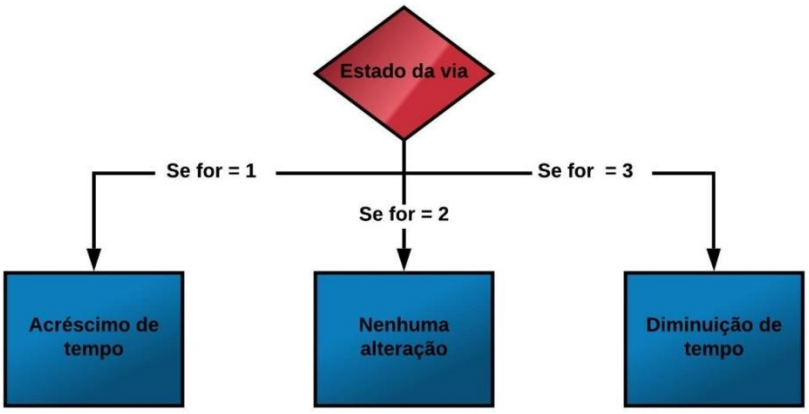


Figura 47: Fluxograma de máquina de estados.
Fonte: Autores (2019).

Como é mostrada na figura 47, a tomada de decisão do sistema será feita de acordo com o valor do estado que for determinado para a via. Quando o estado da via possuir o valor 1, o *script* vai otimizar o tempo de sinal verde, incrementando cerca de 10% do seu valor.

Quando o sistema detectar o valor 2 para o estado da via, ele entende que as velocidades estão compatíveis com a via e não realizará nenhuma mudança, assim deixando o tempo de verde inalterado.

Para o valor 3 do estado da via, o sistema entende que os veículos estão em uma velocidade acima do permitido naquela via e tentará realizar um controle sobre os mesmos, assim diminuindo o tempo de verde, fazendo a subtração do seu tempo pela porcentagem de 10% do seu valor.

Assim que todos os procedimentos mencionados acima forem finalizados, a aplicação irá aguardar uma próxima execução do seu algoritmo para verificar se ainda será necessária outra otimização do tempo de ciclo verde do semáforo, como é mostrado na figura 48.

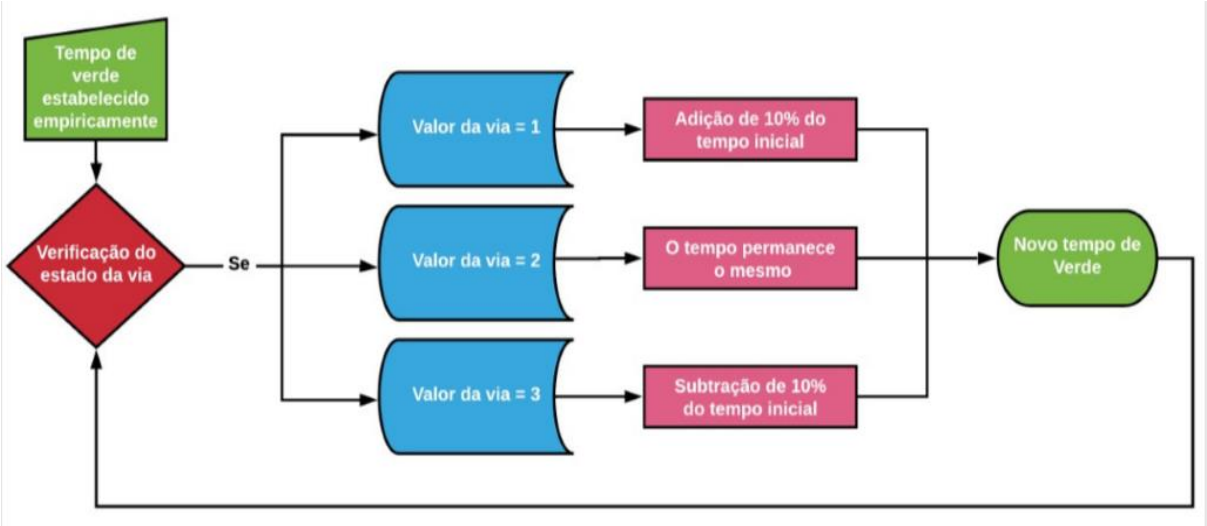


Figura 48: Fluxograma da máquina de estado
Fonte: Autores (2019).

6.3 CRIAÇÃO DO BANCO DE DADOS

Depois dos procedimentos acima, a próxima etapa seria decidir a forma como o arquivo de atualização do tempo de ciclo verde seria capaz de migrar para o micro controlador *Raspberry*, pois a demonstração do funcionamento do sistema não poderia

ficar limitada apenas a uma simulação virtual. Assim, fez-se necessária à montagem de uma maquete para simular o funcionamento de um semáforo em tempo real, assim todo o circuito eletrônico seria implementado pelo micro controlador, como já foi citado.

Antes de iniciar a construção da maquete, foi necessário verificar a melhor forma de transferir o valor da variável de tempo do semáforo verde, assim foi escolhido que a melhor estratégia seria a criação de um banco de dados para salvar o valor da variável.

O banco de dados escolhido para ser utilizado no projeto foi o *Firebase*, que é *Baas* (*Backend as a Service*) para aplicações *Web* e *Mobile* do *Google*, ou seja, ele disponibiliza serviços como configuração de servidor, integração com banco de dados, sistema de *push notification* e entre outros (ORLANDI, 2019).

Outro motivo pelo *Firebase* ter sido escolhido para o projeto, além de oferecer uma parte do seu serviço de forma gratuita, foi por oferecer uma compatibilidade muito eficiente com a linguagem *Python*, assim sendo possível qualquer tipo de alteração no banco de dados por linhas de comando escritas na linguagem.

A figura 49 mostra o código em *Python* para a implementação do banco de dados no *Firebase*:

```
import pyrebase

config = {
  "apiKey": "AIzaSyD0zboSWeuHyqOkh297nr92oW5cno_r3go",
  "authDomain": "teste-3f889.firebaseapp.com",
  "databaseURL": "https://teste-3f889.firebaseio.com",
  "projectId": "teste-3f889",
  "storageBucket": "teste-3f889.appspot.com",
  "messagingSenderId": "206922600117",
  "appId": "1:206922600117:web:a06068c20acaaeb7"
}

firebase = pyrebase.initialize_app(config)

db = firebase.database()

arquivo_tempo_verde = open("tempo_verde.txt","r")
tempstr = arquivo_tempo_verde.read()
temp = float(tempstr)
db.child().update({"tempo_verde": temp})
```

Figura 49: Código para inserir o valor para o banco de dados.
Fonte: Autores (2019).

O *script* faz uso do módulo *Pyrebase* para que seja possível qualquer alteração no banco de dados pela escrita do *Python*, assim o mesmo se inicia com a variável *config* que sincroniza o *script* com as configurações do banco de dados criados na aplicação web, como é demonstrado na figura 50.

```
const firebaseConfig = {
  apiKey: "AIzaSyD0zboSWeuHyqOkh297nr92oW5cno_r3go",
  authDomain: "teste-3f889.firebaseapp.com",
  databaseURL: "https://teste-3f889.firebaseio.com",
  projectId: "teste-3f889",
  storageBucket: "teste-3f889.appspot.com",
  messagingSenderId: "206922600117",
  appId: "1:206922600117:web:a06068c20acaaeb7"
};
```

Figura 50: Configurações do banco de dados para inserir no script.
Fonte: Autores (2019).

Depois que o banco de dados foi criado na ferramenta, basta usar a configuração do mesmo e inserir na variável *config*, para que assim seja possível realizar qualquer tipo de alteração por linhas de comando, tais como inserir, apagar ou alterar qualquer elemento contido no mesmo.

O *script* da figura 51 apenas precisa estar salvo em qualquer pasta do micro controlador *Raspberry*, Ao ser executado, o *script* vai ler o valor escrito no arquivo de texto “tempo_verde.txt” (figura 51, linha 14) e vai inserir o mesmo no banco de dados criado. Assim, o script da figura 51 apenas lê o valor que se encontra no banco de dados, escreve no *log* do compilador e salva novamente em formato de arquivo *txt*.

```
import pyrebase
config = {
  "apiKey": "AIzaSyD0zboSWeuHyqOkh297nr92oW5cno_r3go",
  "authDomain": "teste-3f889.firebaseapp.com",
  "databaseURL": "https://teste-3f889.firebaseio.com",
  "projectId": "teste-3f889",
  "storageBucket": "teste-3f889.appspot.com",
  "messagingSenderId": "206922600117",
  "appId": "1:206922600117:web:a06068c20acaaeb7"
}
firebase = pyrebase.initialize_app(config)
db = firebase.database()
while(True):
    x = 'tempo_verde';
    data = db.child(x).get()
    print(data.val())
    x = data.val()
    y = str(x)
    arquivo = open('tempoverderasp.txt', 'w')
    arquivo.write(y)
    arquivo.close()
    if data.val() != None:
       print('valor encontrado')
```

Figura 51: Código para salvar valor da variável no micro controlador.
Fonte: Autores (2019).

Depois que a fase de implementação do banco de dados foi finalizada, a última etapa para a finalização do projeto foi a elaboração da maquete para demonstrar o funcionamento do sistema implementado.

6.4 CONSTRUÇÃO DA MAQUETE

A última etapa necessária para a conclusão dos procedimentos metodológicos do projeto seria a construção de uma maquete que teria a função de simular o funcionamento de um semáforo implementado com o sistema. Para a criação da maquete, foram usados os itens da Tabela 1:

Material	Quantidade
Bocal e-27 p/ abajur	3
Cabo paralelo preto 5 metros	1
Lâmpadas (verde, amarela e vermelha)	3
Módulo relé	1
Cano PVC de 1 metro	1
Joelho PVC 20 mm	4
Cano PVC formato T 20 mm	3
Caixa de papelão para sapato	1
Papel de EVA cor preta	3
Raspberry Pi Model 3b	1

Tabela 1: Itens utilizados para construção da maquete.
Fonte: Autores (2019).

Na confecção do semáforo, foi utilizada a caixa de papel junto com o material EVA para cobri-la, assim deixando o mais próximo possível da realidade. Foram realizados 3 cortes para que fosse possível encaixar os 3 bocais para as lâmpadas de cores verde, amarela e vermelha, como é mostrado na figura 52.

Figura 52: Bocais finalizados da maquete.
Fonte: Autores (2019).

Foram soldados os cabos nos terminais dos bocais para que fosse possível a conexão dos mesmos com seus respectivos terminais do módulo relé e para a tomada macho, assim terminado a ligação das lâmpadas em modo paralelo, assim todo o funcionamento do circuito era controlado pela programação do micro controlador *Raspberry Pi* junto com o módulo rele, este que é responsável pela comutação dos estados das lâmpadas, como é mostrado na figura 53 e 54.

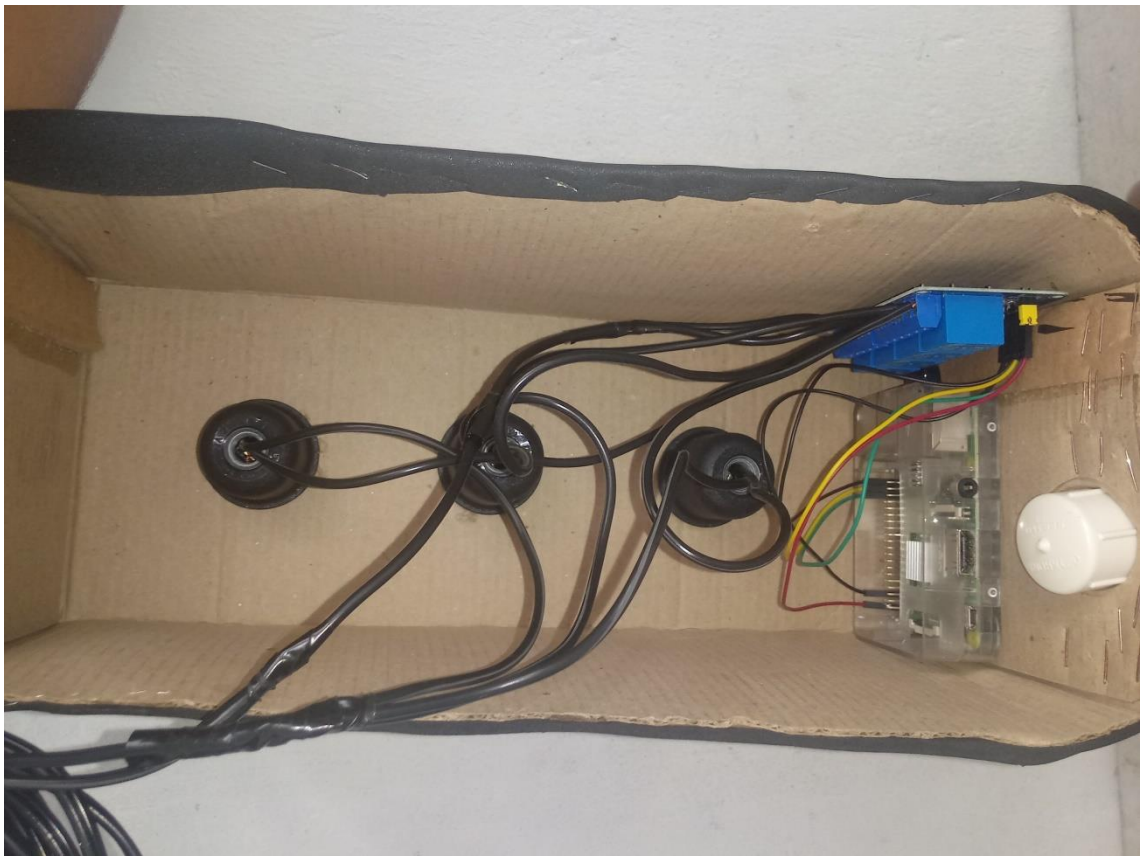


Figura 53: Circuito eletrônico inserido na maquete.
Fonte: Autores (2019).

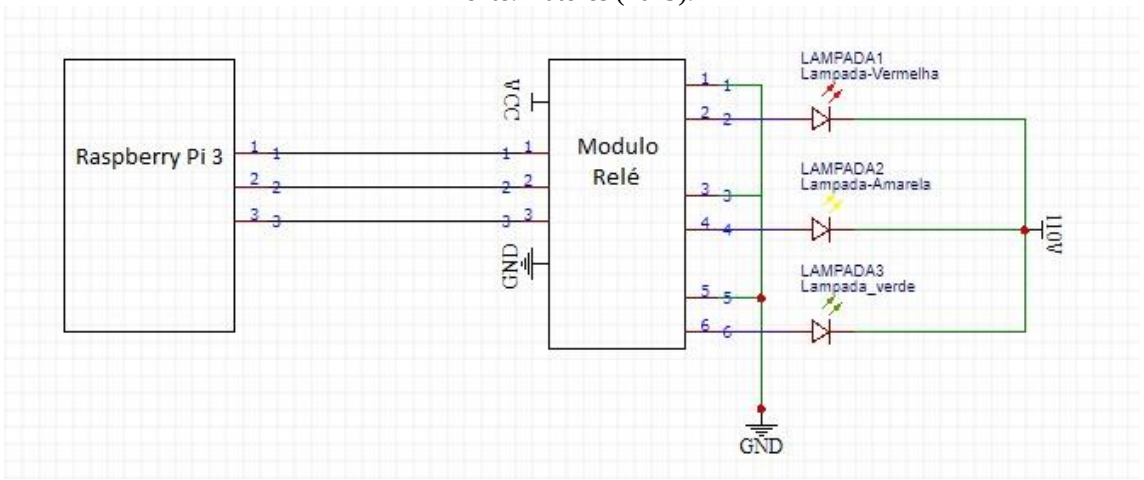


Figura 54: Diagrama do circuito eletrônico.
Fonte: Autores (2019).

Depois que finalizada a implementação do circuito elétrico, a próxima etapa foi construir a base de sustentação para o semáforo, assim foram utilizados canos de PVC para a criação do suporte, como está mostrado na figura 55 abaixo:

Figura 55: Base feita por canos PVC.
Fonte: Autores.

A maquete finalizada pode ser vista na figura 56.

Figura 56: Maquete finalizada.
Fonte: Autores.

Com a finalização da maquete para simulação do funcionamento do projeto, assim como o término da programação dos scripts necessários para cada função, os procedimentos metodológicos foram encerrados e realizados vários testes, com a finalidade de verificar se os resultados obtidos foram satisfatórios ou não.

7 RESULTADOS OBTIDOS

Depois que os procedimentos metodológicos mencionados no capítulo anterior foram concretizados, foram necessários diversos testes com a finalidade de verificar se o desempenho do sistema é considerado satisfatório.

Em relação à etapa de binarização, necessária para destacar nas imagens os objetos que estão em movimento, pode-se observar que o sistema consegue identificar de forma eficiente todos os veículos que se encontram presentes na via, como é mostrado na figura 57.

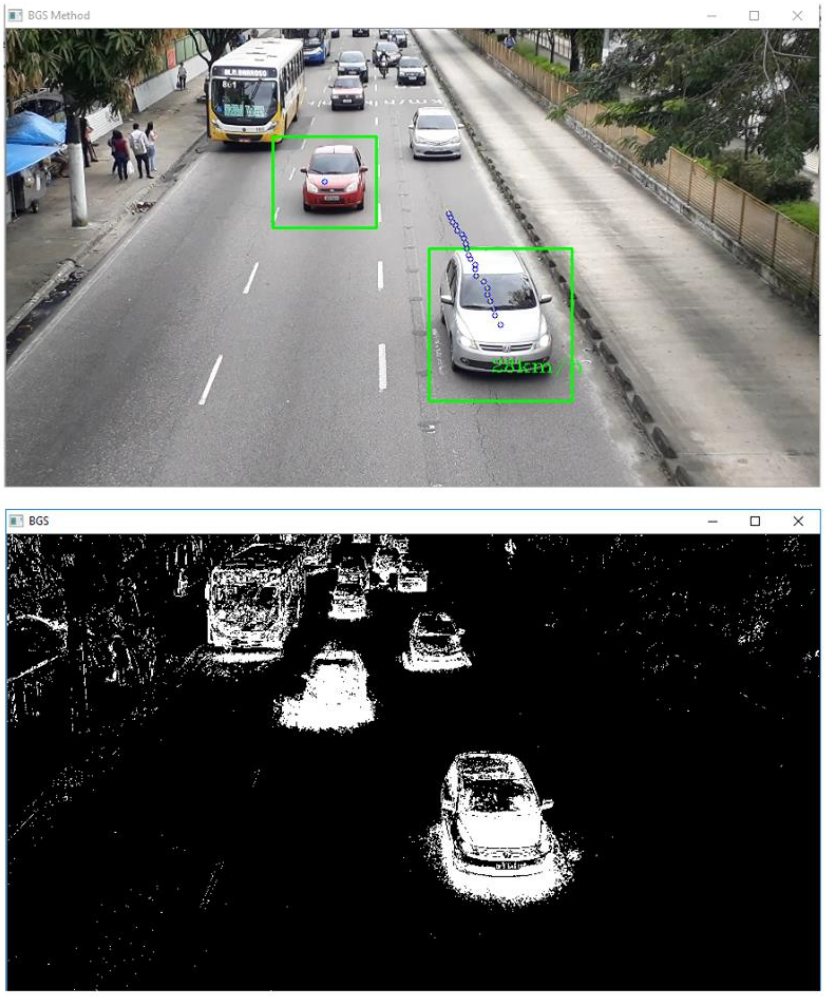


Figura 57: Binarização realizada em imagens autorais.
Fonte: Autores.

Em relação à contagem de veículos, em alguns momentos no vídeo existe a presença de ruídos, o que acaba interferindo nesta etapa. Para verificar a veracidade desta informação, foram realizados testes de contagem manual dos veículos pelos autores, para que assim a média da quantidade dos veículos fosse comparada com a

quantidade medida pelo sistema, assim obtivemos os seguintes resultados mostrados na tabela 2:

Vídeo	Carros contados manualmente	Carros contados pelo sistema
1	24	253
2	56	491
3	41	359
4	65	585
5	95	513
6	98	902

Tabela 2. Dados obtidos para teste de contagem de veículos.
Fonte: Autores.

Como podemos observar nos dados contidos na tabela 2, o sistema não é eficiente para contar os veículos presentes na rua, pelo fato de que durante a execução do algoritmo, o mesmo acaba contando o mesmo veículo por diversas vezes. Esta constatação foi o que provocou a mudança na estratégia de medição: em vez de contar o número de carros, passou-se a contabilizar a velocidade média da via. Na nova abordagem, esta adversidade não interfere na medição da velocidade média final, visto que cada objeto será registrado com velocidades médias aproximadas, assim durante a etapa de cálculo final, a velocidade média ainda estará com o seu valor aproximado do real.

O próximo teste foi verificar se a velocidade calculada pelo sistema seria compatível com a velocidade real dos veículos que estão localizados na via. Para isso foi necessário que um dos integrantes da equipe utilizasse seu carro para dirigir na mesma via que estava sendo analisada no trabalho, como é mostrado na figura 58:

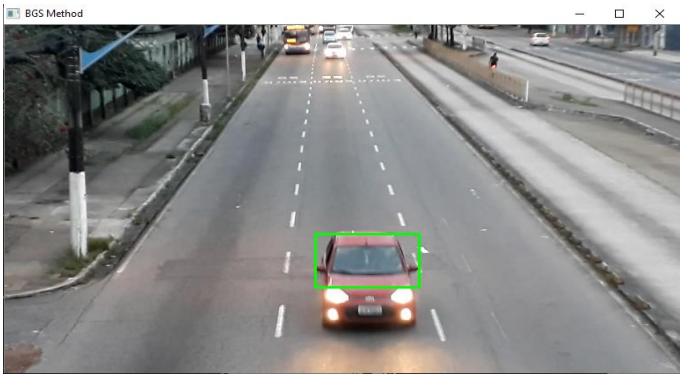


Figura 58: Teste para detecção da velocidade dos veículos.
Fonte: Autores.

Para que fosse possível a comparação da velocidade medida pelo algoritmo com a velocidade real do veículo, a estratégia adotada foi que um dos autores deste trabalho se locomovesse com um veículo próprio, este que seria o carro Ford Fiesta Sedan de cor vermelha (conforme a figura 58) e que estivesse com velocidade constante de 40 km/h. Assim, foi verificado que o sistema mediu a velocidade do mesmo carro em 22 km/h, ou seja, a saída fornecida pelo sistema estava inferior ao valor real da velocidade do carro.

O motivo para o surgimento desta adversidade se dá pela diminuição do processamento de frames que ocorre devido à etapa de binarização que acontece ao mesmo tempo em que o vídeo pós-processado é exibido. A estratégia adotada para resolver este problema foi realizar uma compensação matemática no valor da variável responsável por registrar e retornar para o usuário a velocidade dos veículos que estão sendo analisados pelo sistema, dessa forma compensando pela queda correspondente à análise de frames por segundo e aproximando a medição da velocidade média do seu valor real.

O teste seguinte teve a finalidade de verificar a eficiência do sistema de otimização do tempo de sinal verde do semáforo depois que é calculado a velocidade média dos veículos na via. Vale ressaltar que o tempo inicial do ciclo de verde do semáforo foi definido para 5 segundos de forma arbitrária.

Para coletar os dados para este teste, todos os vídeos autorais passaram pelo método de processamento de imagem e cada vídeo obteve uma média de velocidade própria, assim como a variação do tempo de ciclo verde, como é mostrado em formato de gráfico nas figuras 59 e 60:

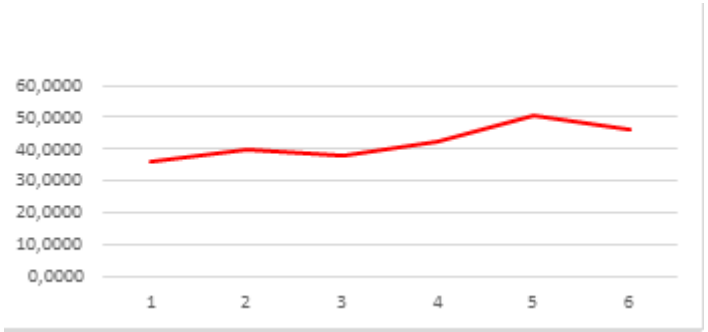


Figura 59: Variação das velocidades médias com cada vídeo.
Fonte: Autores (2019).

Figura 60: Variação do tempo de verde para cada vídeo.
Fonte: Autores (2019).

Com base nos resultados obtidos na execução de cada um dos vídeos autorais, podemos observar que em sua maioria, foram encontradas situações onde foi necessária a otimização do tempo de ciclo do verde, acrescentando 10% de seu valor anterior para os novos tempos, como é mostrado nos vídeos 1, 2, 3, 4 e 6. Isso se deve ao fato de que como as velocidades médias se encontram incompatíveis com um fluxo moderado, o sistema entende que é necessário realizar a otimização dos tempos do semáforo verde.

Do vídeo 4 para o 5 é notável que ocorreu uma diminuição do valor do tempo de verde, pois o sistema compreendeu que como a velocidade média nesta situação em específico já se encontrava em vazão excessiva, houve um decréscimo no tempo de verde para controlar a velocidade excessiva na via .

O mesmo teste foi repetido, mas desta vez utilizando dados simulados, a fim de obter uma maior variabilidade de velocidades médias registradas pelo sistema, e usando novamente o algoritmo para determinar o tempo de semáforo verde, foram obtidos novos resultados, estes que podem ser vistos em formato de gráfico nas figuras 61 e 62.

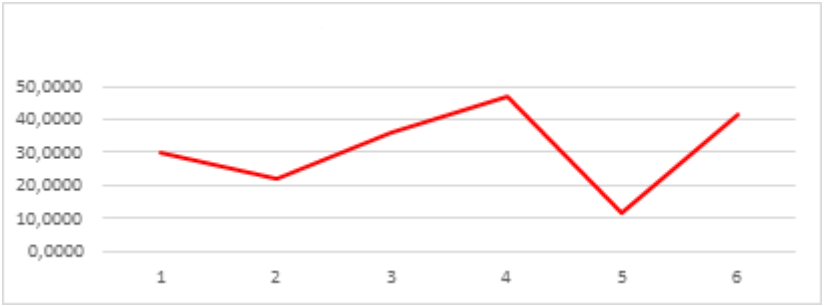


Figura 61: Variação da velocidade média com dados fictícios.
Fonte: Autores (2019).

Figura 62: Variação do tempo de verde com dados fictícios.
Fonte: Autores (2019).

Utilizando o segundo conjunto de dados, podemos observar que o algoritmo continua se comportando de maneira eficiente, pois o mesmo possui a capacidade de realizar a otimização dos tempos de ciclo verde do semáforo de acordo com a velocidade média que é registrada em cada execução do sistema, adequando-se dinamicamente à situação.

É importante ressaltar que os dados fictícios propositalmente ocasionam em variações bruscas. Isso se deve a necessidade de averiguar o tempo de resposta do sistema para esse tipo de situação, onde o mesmo mostrou ter uma resposta rápida, eficiente e dinâmica.

8 CONSIDERAÇÕES FINAIS

Processamento de imagem é uma tecnologia que permite a extração de qualquer tipo de informação que esteja inserida em uma imagem ou vídeo, podendo ser muito útil em diversas áreas da engenharia da computação. A proposta deste trabalho teve como principal foco esta tecnologia, com a integração de controladores e sistemas inteligentes para que fosse possível o seu aprimoramento e eficácia.

O tema que foi abordado neste trabalho é muito importante, pois o mesmo possui uma alta gama de aplicações, podendo ser útil em resolver problemas em diversas áreas, tais como agronegócio, medicina, mobilidade urbana e etc.

Em relação aos objetivos propostos neste trabalho, pode se dizer que os resultados obtidos foram considerados bastante satisfatórios, pois apesar de algumas adversidades que surgiram, o sistema como um todo foi capaz de executar as suas principais funções de maneira confiável e satisfatória, respondendo de acordo com o esperado.

Apesar do sistema não ser capaz de identificar veículos grandes como se fosse apenas um veículo, o mesmo se mostrou bastante viável para a detecção da velocidade de veículos presentes em uma via, com o auxílio do processamento de imagem integrado em controladores.

8.1 TRABALHOS FUTUROS

Para uma melhoria mais intensiva neste projeto, é necessária à implementação de futuras melhorias em relação ao modo de detecção de velocidade utilizado pelo sistema para o cálculo da velocidade dos veículos. Todas as possíveis implementações serão descritas neste tópico.

Entre as possíveis melhorias para o projeto, uma delas seria um aprimoramento no sistema de detecção de veículos, visto que como foi citado anteriormente, o processamento de vídeo apresentam em alguns momentos certos ruídos na detecção de objetos.

A fim de tornar o projeto mais aplicável seria essencial a capacidade de o sistema analisar imagens de vídeo em tempo real, podendo ser via *streaming* por

câmeras funcionando 24 horas por dia, assim como a utilização do sistema de araras da SEMOB (Secretaria de Mobilidade Urbana), onde são usados sensores em campo inseridos na via para calcular a velocidade que um determinado veículo consegue percorrer em uma distância específica, assim tornado a medição mais precisa e confiável.

Para melhorar a variação do tempo verde, um sistema especialista com lógica *fuzzy* poderia ser usado para substituir o cálculo de tempo, fazendo com que esta variação fosse mais tolerante a falhas e tornando esta variação mais dinâmica e suave.

9 REFERÊNCIAS BIBLIOGRÁFICAS

ALBUQUERQUE, Márcio Pontes de; PERSECHINO, André. Processamento digital de imagens: conceitos fundamentais. **Centro Brasileiro de Pesquisas Físicas - Coordenação de Atividades Técnicas**, Rio de Janeiro, p. 1-41, 15 out. 2015.

ALBUQUERQUE, Márcio; ALBUQUERQUE, Marcelo. Processamento de Imagens: Métodos e Análises. **Centro Brasileiro de Pesquisas Físicas – CBPF/MCT**, Rio de Janeiro, p. 1-12, 20 nov. 2000. Disponível em: http://www.cbpf.br/cat/pdsi/pdf/ProcessamentoImagens.PDF. Acesso em: 22 jan. 2019.

Araújo, S. C. (2006). Controlador de tráfego: semáforo inteligente.

ARTERO, A. O.; TOMMAZELLI, A. M. G., 2009. **Detecção e afinamento de bordas em direções previamente conhecidas**. Boletim de Ciências Geodésicas, v. 15, p. 157-177.

BALAN, Willians. **A IMAGEM DIGITAL**. Disponível em: http://willians.pro.br/disciplinas/A%20Imagem%20Digital%20-%20Willians%20Cerozzi%20Balan.pdf. Acesso em: 9 abr. 2019.

COIMBRA, Rodrigo. **O que são Controladores Lógicos Programáveis e suas diferentes linguagens de programação**. [S. l.], 24 set. 2009. Disponível em: https://projetoseti.com.br/o-que-so-controladores-lgicos-programveis-e-suas-diferentes-linguagens-de-programao/. Acesso em: 26 abr. 2019.

COMO FUNCIONA o CLP? [*S. l.*], 31 maio 2010. Disponível em: https://clpredes.wordpress.com/2010/05/31/como-funciona-o-clp/. Acesso em: 3 maio 2019.

COSTA, Cesar. **Projetando Controladores Digitais com FPGA**. [S. l.]: Novatec, 2006.

COSTA, Welbson Siqueira; SILVA, Shirlly Christiany Macedo. **Aquisição de conhecimento: O grande desafio na concepção de sistemas especialistas**. Rio Grande do Norte: UFRN, 2005.

COSTA, Welbson; SILVA, Shirlly. **Aquisição de conhecimento: o grande desafio na concepção de sistemas especialistas**. Holos, Rio Grande do Norte, p. 1-11, 13 set. 2005. Disponível em: http://www2.ifrn.edu.br/ojs/index.php/HOLOS/article/viewFile/71/77. Acesso em: 9 abr. 2019.

CUCCI NETO, João. **Manual brasileiro de sinalização de trânsito volume v – sinalização semafórica**. [S. l.: s. n.], 2014.

DENATRAN. **Manual de Semáforos**. Brasília: [*s. n.*], 1984. Disponível em: https://wp.ufpel.edu.br/csttt/files/2013/05/Manual-Semaforos-Denatran-1984.pdf. Acesso em: 10 jan. 2019.

DREY, Ramiro. **Semáforos Inteligentes**. Ti Especialistas. 2015. Disponível em: https://www.tiespecialistas.com.br/semaforos-inteligentes/. Acesso em: 21/11/18.

FAÇANHA, Tiago; CARNEIRO, André; COSTA FILHO, José. Filtro de Kalman via programação quadrática. **CENTAURO, Departamento de Engenharia de Teleinformática**, Fortaleza, p. 1-6, 6 fev. 2015.

FELISBERTO, Bianca. **Conceito e utilização de imagens digitais**. [S. l.], 5 nov. 2014. Disponível em: http://felisberto-bianca.blogspot.com/2014/11/conceito-e-utilizacao-de-imagens.html. Acesso em: 2 abr. 2019.

FERNANDES, Henrique; BARCELOS, Celia. REMOÇÃO DE RUÍDO EM IMAGENS COLORIDAS. *In*: IX ENCONTRO INTERNO & XIII SEMINÁRIO DE INICIAÇÃO CIENTIFICA, 2009, Uberlândia. Anais [...]. Uberlândia: [*s. n.*], 2009.

FERRARI, GISELLE LOPES. **INTELLEC: SHELL PARA DESENVOLVIMENTO DE SISTEMAS ESPECIALISTAS GISELLE LOPES FERRARI FLORIANÓPOLIS 2005**. 2005. Dissertação (Mestre Engenharia Elétrica) - Universidade Federal de Santa Catarina, Florianópolis, 2005.

FLÁVIO, Luís. **Belém tem um veículo para cada três pessoas**. Belém, 26 nov. 2017. Disponível em: https://www.diarioonline.com.br/noticias/para/noticia-468578-belem-tem-um-veiculo-para-cada-tres-pessoas.html. Acesso em: 31 maio 2019.

FLÁVIO, Luís. **Ranking coloca Belém como a 25ª cidade mais congestionada do mundo**. Belém, nove abr. 2018. Disponível em: http://m.diarioonline.com.br/noticias/para/noticia-499796-ranking-coloca-belem-coma-25%C2%AA-cidade-mais-congestionada-do-mundo.html. Acesso em: 31 maio 2019.

GAETE, Constanza. **As 7 causas mais comuns de congestão viária e as estratégias (exitosas) para enfrentá-la**. Arch Daily. 2016. <Disponível em: https://www.archdaily.com.br/br/799252/as-7-causas-mais-comuns-de-congestao-viaria-e-as-estrategias-exitosas-para-enfrenta-la>. Acesso em: 03/09/2018.

GONZALES, Rafael Gonzales; WOODS, Richard. **Processamento Digital de Imagem**. São Paulo: Pearson, 2011.

JO, Marcelo. **Histogramas**. [S. l.], 3 mar. 2015. Disponível em: https://www.embarcados.com.br/histograma/. Acesso em: 13 mar. 2019.

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

MARTINEZ, Manuela. **Trânsito: Congestionamentos se agravam nas metrópoles do país**. Uol Online. 2008. Disponível em: <https://vestibular.uol.com.br/resumo-das-disciplinas/atualidades/transito-congestionamentos-se-agravam-nas-metropoles-do-pais.htm?cmpid=copiaecola. >. Data de acesso: 29/10/2018.

MARTINS, SAMUEL BOTTER. Introdução ao Processamento Digital de Imagens Parte 1. **Instituto de Computação Universidade Estadual de Campinas**

(UNICAMP), São Paulo, p. 1-15, 26 jul. 2016. Disponível em: https://www.ic.unicamp.br/~ra144681/misc/files/ApostilaProcDeImagesParteI.pdf. Acesso em: 6 abr. 2019.

MENDES, R. D. **Inteligência artificial; sistemas especialistas no gerenciamento da informação**. Ciência da Informação. 1998. Disponível na Internet: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19651997000100006. Acesso: 20 abr., 2019.

MENESES, Paulo; ALMEIDA, Tati. **Introdução ao Processamento de Imagens de Sensoriamento Remoto**. Brasilia: [*s. n.*], 2012.

MING, S. H. Uma breve descrição do sistema Scoot. Notas Técnicas (NT 201), 1997. Companhia de Engenharia de Tráfego de São Paulo, Disponível em: http://www.cetsp.com.br/consultas/notas-tecnicas.aspx. Acesso em: 10/02/19.

MORENO, L. D.; MAMEDE, B. B.; PINA FILHO, A. C. de. **Automação de semáforos para uma melhor dinâmica urbana. Revista dos Transportes Públicos**, v. 138, ANTP, 2014, p. 11-25.

MOTORISTAS reclamam de falta de sincronia nos semáforos. Mato Grosso, 12 abr. 2018. Disponível em: https://www.atribunamt.com.br/2018/04/12/motoristas-reclamam-de-falta-de-sincronia-nos-semaforos/. Acesso em: 31 maio 2019.

PROMOTION**. O QUE SÃO CONTROLADORES NA AUTOMAÇÃO INDUSTRIAL?** [S. l.], [S.d]. Disponível em: http://www.group-promotion.com/o-que-sao-controladores-na-automacao-industrial/. Acesso em: 24 abr. 2019.

ORLANDI, Cláudio. **Firebase: serviços, vantagens, quando utilizar e integrações**. [*S. l.*], Janeiro 2019. Disponível em: https://blog.rocketseat.com.br/firebase/. Acesso em: 8 maio 2019.

PEDRINI, H.; SCHWARTZ, W.R. **Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações**. Editora Thomson Learning, 2007.

PEREIRA, G. e RIBEIRO, M. V. M. Controle de tráfego em tempo real: novos paradigmas, dificuldades e primeiros resultados. O caso do controle inteligente de tráfego (CIT). In: 16° CONGRESSO BRASILEIRO DE TRANSPORTE E TRÂNSITO, 2007. Associação Nacional de Transportes Públicos – ANTP. Maceió-AL, 2007.

PETRUZELLI, Frank. **Controladores Lógicos Programáveis**. São Paulo: AMGH, 2014.

RICHTER, José. **Cidades inteligentes: importância de oferecer um espaço humanizado.** Richter Gruppe. 2017. Disponível em:<http://richtergruppe.com.br/cidades-inteligentes-importancia-de-oferecer-um-espaco-humanizado/>. Acesso em: 21/11/18.

ROCHA, Jordão. **COMO FUNCIONA O CLP?** [*S. l.*], 11 ago. 2013. Disponível em: http://saladaautomacao.com.br/como-funciona-o-clp/. Acesso em: 2 maio 2019.

SILVA NETO, Adalberto. **Comparação entre o Filtro de Kalman e Filtro de Partículas Aplicadas na Robótica Móvel**. UNICAMP, São Paulo, p. 1-6, 7 jan. 2015. Disponível em: http://www.dca.fee.unicamp.br/~gudwin/courses/IA889/2014/IA889-15.pdf. Acesso em: 17 abr. 2019.

SPIRLANDELLI, L. P. **Sistemas Especialistas: Um Estudo de Caso Com o Expert Sinta. Revista Eletrônica de Sistemas de Informação e de Gestão Tecnológica**, v. 1, n. 1, p. 1–16, 2011.

STUMM JR., Valdir. **O MÓDULO TURTLE**. [*S. l.*], 9 jul. 2011. Disponível em: https://pythonhelp.wordpress.com/2011/07/09/o-modulo-turtle/. Acesso em: 16 maio 2019.

TANSCHEIT, Paula. **O que torna uma cidade inteligente?**, [S. l.], 16 ago. 2016. Disponível em: https://thecityfixbrasil.com/2016/08/16/o-que-faz-de-uma-cidade-inteligente/. Acesso em: 10 out. 2018.

VILANOVA, Luis. **Programação de um Semáforo Usando o Método do Grau de Saturação**. Disponível em: http://www.sinaldetransito.com.br/artigos/saturacao.pdf. Acesso em: 10/02/19.

XAVIER PY, Mônica. **Sistemas especialistas: uma introdução. Instituto de informática**, Universidade federal do Rio Grande do Sul. 2009. Disponível em <http://www.inf.ufrgs.br/gppd/disc/cmp135/trabs/mpy/sistemasespecialistas.pdf > Acesso em 18 abr. 2019

ZANOTTO, Henrique. **Falta de sincronia nos semáforos da Beira-Mar Norte causa trânsito**. Florianópolis, 27 set. 2017. Disponível em: https://ndmais.com.br/videos/balanco-geral-florianopolis/falta-de-sincronia-nos-semaforos-da-beira-mar-norte-causa-transito/. Acesso em: 31 maio 2019.

APÊNDICE A – SCRIPT PARA IMPLEMENTAÇÃO DO SEMÁFORO VIRTUAL

As figuras 63,64 e 65 mostram o script para a implementação do semáforo virtual para teste.

```
import turtle
import time
# escolhe a cor de fundo da janela
wn = turtle.Screen()
wn.bgcolor('skyblue')
# cria as "tartarugas"
tess = turtle.Turtle()
alex = turtle.Turtle()
henry = turtle.Turtle()

def draw_housing():
    """ Desenha o formato do semáforo"""
    tess.pensize(3)
    tess.color('black', 'white')
    tess.begin_fill()
    tess.forward(80)
    tess.left(90)
    tess.forward(157)
    tess.circle(40, 180)
    tess.forward(157)
    tess.left(90)
    tess.end_fill()
draw_housing()
def circle(t, ht, colr):
    """Posicionamento de cada tartaruga pa
    t.penup()
    t.forward(40)
    t.left(90)
    t.forward(ht)
    t.shape('circle')
    t.fillcolor(colr)
circle(tess, 40, 'green')
circle(alex, 100, 'orange')
circle(henry, 160, 'red')
```

Figura 63: Código para “desenhar” o semáforo.
Fonte: Autores.

```
state_num = 0
def advance_state_machine():

    global state_num

    if state_num == 0:  # Transição do estado 0 para 1
        henry.color('darkgrey')
        alex.color('darkgrey')
        tess.color('green')
        wn.ontimer(advance_state_machine)
        arquivo_tempo_verde = open("tempo_verde.txt","r")
        tempstr = arquivo_tempo_verde.read()
        temp = float(tempstr)
        print(" _____________________")
        print ("|       ESTADO        |")
        print ("|---------------------|")
        temp= round(temp,3)
        print "| VERDE:   ", temp , " seg |"

        time.sleep(float(temp))# determina o tempo do ciclo
        state_num = 1


    elif state_num == 1:  # Transição do estado 1 para 2
        henry.color('darkgrey')
        alex.color('orange')
        tess.color('darkgrey')
        wn.ontimer(advance_state_machine)
        print("| AMARELO:  5 seg     |")
        time.sleep(5)
        state_num = 2

    elif state_num == 2:    # Transição do estado 2 para 0
        henry.color('red')
        alex.color('darkgrey')
        wn.ontimer(advance_state_machine)
        print("| VERMELHO: 5 seg     |")
        print("|_____________________|\n")
        time.sleep(5)
        state num = 0
```

Figura 64: Código do funcionamento do semáforo.
Fonte: Autores.

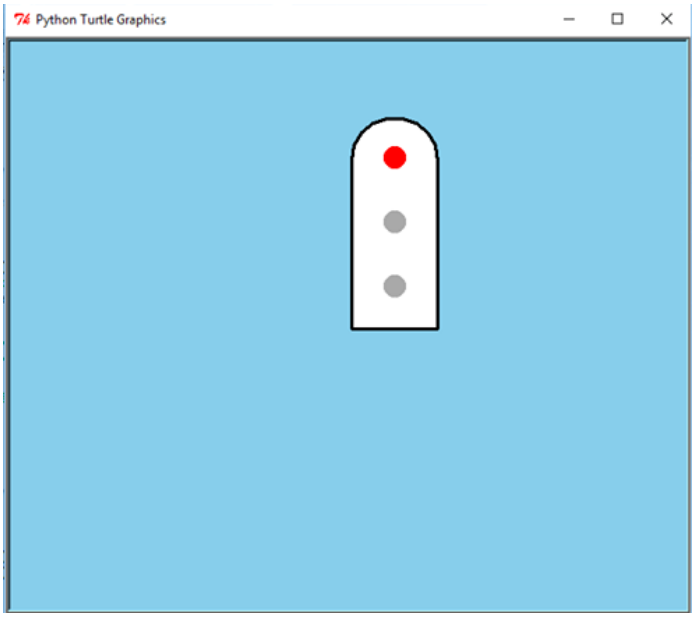


Figura 65: Demonstração do semáforo virtual.
Fonte: Autores.