GIL ANDRADE GONTIJO
NATHAN SAMPAIO SANTOS

Algoritmo de reconhecimento de faixas de travessia e pedestres por meio do uso de Visão Computacional

São Paulo

2021

GIL ANDRADE GONTIJO
NATHAN SAMPAIO SANTOS

Algoritmo de reconhecimento de faixas de travessia e pedestres por meio do uso de Visão Computacional

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Engenheiro

Área de concentração:
Engenharia Elétrica e de Computação

Orientador: Prof. Dr. Ricardo Luis de Azevedo da Rocha
Co-orientador: Prof. Dr. Claudio Luiz Marte

São Paulo
2021

Prof. Dr. Ricardo Luis de Azevedo da Rocha

Agradecimentos

O desenvolvimento deste trabalho teve o apoio e o acompanhamento de pessoas que puderam tornar possível sua conclusão, dentre as quais agradeço:

Ao nosso professor orientador, Prof. Dr. Ricardo Luis de Azevedo da Rocha que nos acompanhou durante todo o percurso e de imediato aceitou nos guiar na ocasião da apresentação do tema. Por meio de direcionamentos, referências e apoio, ofereceu todo o auxílio necessário para a elaboração do projeto.

Ao nosso professor co-orientador, Prof. Dr. Claudio Luiz Marte, que se mostrou completamente aberto na nossa busca por temas ligados à sua área de atuação e nos auxiliou a encontrar um assunto que fosse uma ponte entre a Engenharia da Computação e a Engenharia de Transportes. Além de ter nos apresentado diversas outras pessoas que também contribuíram muito para o trabalho.

À Laisa, que nos acompanhou e dedicou seu tempo a promover a discussão do andamento do projeto em reuniões semanais.

Ao Casimiro, que com sua experiência em visão computacional pôde nos auxiliar quanto aos passos a serem seguidos e também às oportunidades de desenvolvimento do algoritmo.

À equipe da CET, que nos inspirou a aceitar este desafio e nos auxiliou com dados que puderam ser usados para fins de teste.

E finalmente, a todos que contribuíram de alguma forma para o projeto de priorização de pedestres, do qual somos muito gratos pela possibilidade de contribuição.

RESUMO

Com a universalização do acesso à internet, a chegada de redes 5G e o advento de um novo paradigma chamado cidades inteligentes, o mundo busca cada vez mais utilizar dados para otimizar os seus processos. Dentre esses processos, o controle semafórico que leva em consideração a prioridade que deve ser dada aos pedestres é um importante passo para conectar ainda mais as pessoas aos sistemas que as cerca.

Assim, esse projeto busca, principalmente, identificar a faixa de travessia de pedestres através do tratamento de imagens. A partir de então, é possível partir para a busca de dados relevantes para o tempo de semaforização, como a quantidade de pessoas aguardando e atravessando, sua idade, gênero, possíveis reduções de mobilidade e entres outros fatores que podem ser estimados em tempo real para que haja um tempo de abertura dos semáforos compatíveis com a demanda ao longo do dia.

Palavras-Chave – semaforização, faixa de pedestres, visão computacional, Open CV, tratamento de imagens, inteligência artificial

ABSTRACT

With the globalization of internet access, upcoming 5G networks, and the new smart cities paradigm, the world seeks to use data to optimize its daily processes. One example would be the controlling of traffic lights to give priority to pedestrians, which is an important step in connecting people to their surrounding systems even better.

Therefore, this project primarily pursues to identify crosswalks using computer imaging processes. From then on, we are able to gather other relevant data to control the timing of traffic lights, such as the number of people crossing and waiting on the sidewalks, their relative age, gender, possible mobility deficiencies, and other elements that can be estimated in real time for compatible cross walk timing according to the flexible demand throughout the day.

Keywords – traffic light control, crosswalk, computer vision, Open CV, image processes, artificial intelligence

LISTA DE FIGURAS

Figura 1: Exemplo de botoeira instalada em um cruzamento viário 12
Figura 2: Funcionamento do projeto de Cyrel O.Manlises utilizando imagens para o controle semafórico 13
Figura 3: Ilustração do projeto de Rosana Rego e Rodrigo Semente 14
Figura 4: Exemplo 1 dos testes iniciais para reconhecimento da faixa de pedestres 28
Figura 5: Exemplo 2 dos testes iniciais para reconhecimento da faixa de pedestres 28
Figura 6: Tratamento de escala de cinza aplicado à imagem 29
Figura 7: Imagem original utilizada para o tratamento em escala de cinza 29
Figura 8: Quadro de uma gravação obtida por uma câmera de monitoramento instalada em São Paulo 30
Figura 9: O distanciamento da câmera afeta a qualidade do reconhecimento de faixa de pedestres 30
Figura 10: Tratamento da imagem com a função Canny 31
Figura 11: Resultado do prolongamento do meio fio na imagem 31
Figura 12: Tentativa (1) falha no prolongamento das linhas do meio fio para reconhecer as faixas de pedestres 32
Figura 13: Tentativa (2) falha no prolongamento das linhas do meio fio para reconhecer as faixas de pedestres 32
Figura 14: Tentativa (3) falha no prolongamento das linhas do meio fio para reconhecer as faixas de pedestres 33
Figura 15: Reconhecimento das faixas utilizando a primeira abordagem com filtro de formas retangulares 34
Figura 16: Reconhecimento das faixas utilizando a primeira abordagem com filtro de formas retangulares com descarte dos quadrados 35
Figura 17: Resultado final da identificação das faixas de pedestres 36
Figura 18: Arquivos para a execução do script 37
Figura 19: Descrição da função "treat_img" 37

Figura 20: Exemplo de tratamento binário para encontrar limites utilizando o Open CV [5] 38
Figura 21: Descrição da função "draw_rectangles" 38
Figura 22: Descrição da função "draw_rectangles_centroid" 40
Figura 23: Esquema de funcionamento dos macroprocessos do modelo de identificação das faixas de pedestres 40
Figura 24:Teste final de reconhecimento 1 41
Figura 25: Teste final de reconhecimento 2 41
Figura 26: Teste final de reconhecimento 3 42
Figura 27: Teste final de reconhecimento 4 42
Figura 28: Teste final de reconhecimento 5 43

LISTA DE TABELAS

Tabela 1: Horários do pedido de vídeos *24*

SUMÁRIO

1. INTRODUÇÃO 11

1.1. Motivação 11

1.2. Objetivo 14

1.3. Justificativa 15

1.4. Organização do trabalho 15

2. ASPECTOS CONCEITUAIS 17

3. TECNOLOGIAS UTILIZADAS 19

4. METODOLOGIA DO TRABALHO 20

4.1 Concepção 20

4.2 Projeto 20

4.3 Aquisição e tratamento dos vídeos 21

4.4 Treinamento e validação do modelo 22

4.5 Manipulação para obtenção de novos dados 22

4.6 Avaliação da performance 23

5. ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA 24

5.1 Requisitos de aquisição dos dados de entrada 24

5.2 Requisitos de posicionamento da câmera 25

5.3 Requisitos de condição das faixas de travessia 26

6. PROJETO E IMPLEMENTAÇÃO 27

6.1 Reconhecimento de Faixas e Calçadas 27

6.1.1 Primeira Abordagem 27

6.1.2 Segunda Abordagem 30

6.1.3 Aperfeiçoamento da Primeira Abordagem 33

6.2 Visão Geral Sobre o Código de Implementação 36

7. TESTES E AVALIAÇÃO 41

8. CONSIDERAÇÕES FINAIS 44

8.1 Conclusões do Projeto de Formatura 44

8.2 Contribuições 45

8.3 Perspectivas de Continuidade 46

9. REFERÊNCIAS BIBLIOGRÁFICAS 47

1. INTRODUÇÃO

Nas grandes metrópoles, alguns dos principais problemas a serem enfrentados pelos seus respectivos cidadãos é o trânsito e o transporte de pessoas. Segundo a legislação brasileira, a forma como os modais se relacionam deve seguir uma ordem de proteção do maior sobre o menor, a saber, do mais frágil para o menos frágil, a lista: 1. Pedestres; 2. Bicicletas; 3. Motos; 4. Carros; 5. Caminhões; e assim por diante.

Seguindo essa ordem, uma das formas de priorizar e, por consequência, proteger o pedestre, o qual é o ator mais frágil nessas interações, é a de dar um intervalo de tempo adequado para a travessia da faixa de pedestres. Isso quer dizer que o tempo deve ser suficiente para que a demanda de pedestres de uma via seja capaz de atravessar até o outro lado enquanto seu sinal ainda permanece verde, bem como a frequência de permissão para que ele atravesse seja bem mensurada de tal forma que não seja muito grande, fazendo com que o trânsito de carros seja desnecessariamente interrompido, e nem muito pequena, de modo que o pedestre tenha que esperar muito tempo na calçada ou que arrisque a travessia da via mesmo sem ter a prioridade naquele momento.

Como parte da solução desse problema maior, está a percepção pelo sistema semafórico de quantos pedestres há nas calçadas aguardando a travessia e o perfil desses pedestres, uma vez que o tempo de travessia pode ser alterado de acordo com alguns fatores como gênero, faixa etária e limitações físicas.

1.1. Motivação

Para darmos prosseguimento ao projeto, precisamos primeiro entender o que já foi estudado e aplicado no controle semafórico no que diz respeito à identificação de pedestres.

Hoje, nas cidades brasileiras, podemos identificar duas políticas de tempo de passagem para pedestre: (1) Tempo Fixo e (2) Botoeira.

A política de tempo fixo é aquela na qual nenhuma entrada de dados em tempo real altera o período em que o semáforo está aberto ou não para os carros. Isso não significa que o tempo semafórico não possa se modificar ao longo do dia, porém todas essas variações já foram previamente programadas pelo controlador semafórico.

Já a botoeira (Figura 1) é um meio pelo qual os pedestres, ao apertar o botão designado, podem informar o sistema que desejam atravessar a via. Assim, de acordo com as prioridades estabelecidas pelo projetista, o semáforo considerará as informações fornecidas para modificar - ou não - o tempo de abertura do sinal.

Figura 1: Exemplo de botoeira instalada em um cruzamento viário

Ambas as políticas apresentam seus defeitos e, à medida que as metrópoles caminham para o conceito de cidades inteligentes, elas também vão tornando-se cada vez mais obsoletas. De um lado, a política de tempo fixo não considera as demandas em tempo real da travessia, fazendo com que possa haver desnecessária espera, seja por parte dos veículos, seja por parte dos pedestres. Por outro lado, a botoeira, além de, em muitas vezes, não ter o seu acesso facilitado para pessoas com necessidades especiais, também está sujeita ao natural desgaste mecânico,

intempéries climáticas, vandalismos e falta de manutenção, tornando, assim, o seu uso impossibilitado ou ineficaz.

Tendo em vista esses problemas, as tecnologias de hoje nos permitem tirar vantagem dos diferentes tipos de sensores disponíveis e, então, poder utilizar esses dados para tornar o controle semafórico mais apto a se adaptar aos diferentes cenários de demanda ao longo do tempo.

Assim, já há no mundo algumas pesquisas que apontam para essa aplicação. É importante mencionar que o reconhecimento de pessoas por imagem já é um campo bem desenvolvido, principalmente por sua utilidade nos carros autônomos. Contudo, o emprego dessa tecnologia com o objetivo de realizar o controle semafórico ainda é escasso.

Ainda assim, um dos trabalhos relevantes que podemos citar foi feito por Cyrel O. Manlises [1] no qual ele utilizou o processamento de imagens para identificar pedestres e, quando houvesse um sinal positivo, o tempo de abertura para a passagem do pedestre era expandido para 45 segundos. Caso contrário, o tempo era mantido em 30 segundos.

Na Figura 2 podemos observar o modelo esquemático para esse projeto.

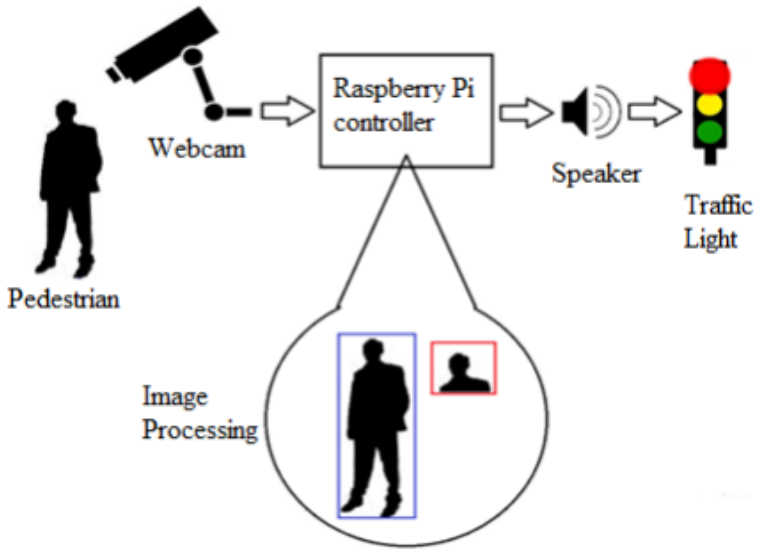

Figura 2: Funcionamento do projeto de Cyrel O.Manlises utilizando imagens para o controle semafórico

Já no Brasil, há muitas poucas iniciativas nesse sentido, sendo uma delas [2] proposta por Rosana Cibely B. Rego e Rodrigo S. Semente, ambos da Universidade Federal Rural do Semi-Árido no Rio Grande do Norte, os quais construíram um equipamento protótipo que identifica a presença de pedestres aguardando a travessia em um cruzamento através de sensores infravermelho e um detector de peso no solo.

A Figura 3 demonstra como seria o funcionamento do protótipo.

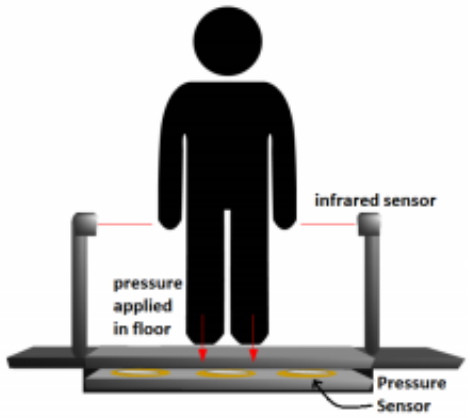

Figura 3: Ilustração do projeto de Rosana Rego e Rodrigo Semente

Rego e Semente justificaram sua escolha por um sistema com sensores simples por causa do alto custo de equipamentos e de processamento que a análise de imagens exige. De certa forma, o protótipo deles apresenta vantagens em relação ao de O.Manlises, pois ambos fazem uma detecção trivial da presença ou não de pedestres em um cruzamento, logo, aquele sistema que apresenta menor custo e menor complexidade tem vantagem sobre o outro.

Entretanto, para criarmos um sistema mais inteligente de controle semafórico, precisamos reunir dados sobre os pedestres que um sistema infravermelho não é capaz de fornecer. Assim, ao utilizarmos imagens, além de detectarmos a presença de pessoas, ainda podemos extrair a contagem de pessoas aguardando, qual é o tipo de pessoa e entre outros fatores que podem otimizar o tráfego nas grandes cidades.

Um dado interessante trazido por Rego e Semente foi de que a otimização do tempo de travessia de pedestres também traz efeitos positivos sobre o congestionamento de carros,

Isto posto, o presente projeto parece ter um bom horizonte de possibilidades e de resultados a partir do seu bom funcionamento, sem contar que o Brasil, especificamente, carece de pesquisas nessa área, o que agregará um bom valor à matéria.

1.2. Objetivo

Este trabalho busca caminhar em direção a uma solução para o problema exposto a partir do estudo de técnicas de Visão Computacional e posterior desenvolvimento de softwares e/ou algoritmos, por meio de ferramentas como o OpenCV, capazes de reconhecer e categorizar diferentes faixas de travessias e perfis de pedestres a partir de vídeos e imagens.

1.3. Justificativa

Assim, uma maneira de cumprir o objetivo proposto é processar imagens de câmeras instaladas nos cruzamentos semafóricos a fim de fornecer ao sistema as entradas de dados citadas anteriormente, ou seja, o número de pessoas aguardando a abertura do sinal e o perfil de cada uma delas. Tal processamento exige o uso de técnicas tais como reconhecimento de imagens em tempo real, a partir de inteligência artificial e o tratamento de um grande volume de dados.

Para a conclusão desse trabalho, contamos não somente com nossos conhecimentos prévios e a serem desenvolvidos, mas também com uma equipe que já se debruça sobre este problema há um bom tempo, dentre os quais podemos mencionar nosso co-orientador, o professor Claudio Marte, do Departamento de Engenharia de Transportes da Engenharia Civil - EPUSP, um grupo técnico da Companhia de Engenharia de Tráfego (CET) da cidade de São Paulo, alunos de graduação e pós-graduação da EPUSP e nosso professor orientador que é especialista em Inteligência Artificial.

Com todos esses recursos e motivados a criar uma solução robusta para essa questão, cremos que esse trabalho será útil não apenas para a conclusão de curso, mas também contribuirá para a construção de uma harmonia mais bem ajustada entre os modais de transporte e, por conseguinte, para a melhora da qualidade de vida dos habitantes das cidades.

1.4. Organização do trabalho

O presente documento é estruturado nas seguintes seções, onde faremos a descrição funcional do projeto:

- Aspectos Conceituais
- Tecnologias Utilizadas
- Metodologia Do Trabalho
- Especificação de Requisitos do Sistema
- Projeto e Implementação
- Testes e Avaliação
- Considerações Finais

2. ASPECTOS CONCEITUAIS

Para poder repassar os dados das leituras de imagens ao sistema de controle semafórico, será necessário identificar três categorias de “objetos” em cada frame. São elas: Faixa de Pedestres, Pessoas (e suas respectivas classificações por gênero, faixa etária e limitações físicas) e Carros.

A ordem de prioridade do trabalho também se dará na ordem supracitada, afinal, a relevância dos dados da faixa de pedestres é maior do que aquela vinda dos dados de identificação de automóveis.

Ao usarmos algoritmos de visão computacional como parte da solução desse problema, precisamos entender que um computador não sabe, por si só, classificar a natureza de um objeto apresentado na tela. Para isso, nós precisamos indicar para ele o que são as classificações desejadas, como por exemplo, pedestres, carros, etc. A partir dessa identificação, o computador reconhece o padrão daquele objeto e, quando detectar tal padrão em uma nova imagem, poderá indicar que aquele padrão pertence à classe previamente definida.

A esse procedimento descrito, chamamos de “treinamento”.

Para realizar um treinamento adequado e que seja genérico o suficiente com o intuito de a máquina poder identificar padrões futuros, necessitamos de uma base de imagens que sejam diversificadas o bastante. Caso contrário, corremos o risco de termos o efeito de overfitting (sobreajuste, em potuguês), o qual a inteligência se ajusta muito bem à base de treinamento, mas é péssima em identificar novos padrões.

Na aplicação desse conceito ao nosso problema, isso significa que a base de treinamento deve conter imagens com resoluções não tão altas, em cenários distintos, como diferentes locais, horários e luminosidades.

Uma vez que temos a posse de um banco de imagens satisfatório, precisamos agora ensinar o computador a identificar uma classe de objetos e isso se dará através do

apontamento dessa classe em centenas de imagens. Chamamos isso de criar anotações na imagem.

Com a nossa base efetivamente pronta, podemos treinar a inteligência do computador e, enfim, colocá-la em uso em produção.

Esse, portanto, será o fluxo do nosso trabalho até que ele seja concluído.

3. TECNOLOGIAS UTILIZADAS

A implementação do algoritmo se dará a partir do uso da linguagem de programação Python e por meio do de bibliotecas de visão computacional como o OpenCV, YOLOv3 e DarkFlow.

A escolha foi feita seguindo a ampla disponibilidade de documentação para a linguagem e a biblioteca em questão e devido também à alta portabilidade e compatibilidade da biblioteca com diversas outras linguagens. Além disso, a presença de inúmeras outras bibliotecas auxiliares para o Python nos possibilitará desenvolver melhor outras tarefas atreladas à extração, tratamento e visualização dos dados e resultados.

Essa alta compatibilidade, por meio de APIs oficiais, facilitará o processo de incorporação do algoritmo em dispositivos com menor capacidade de processamento, já que um dos objetivos do projeto está relacionado à aplicação em semáforos, os quais não dispõem de processadores robustos para a execução do algoritmo em tempo real.

4. METODOLOGIA DO TRABALHO

4.1 Concepção

O trabalho foi concebido a partir de reuniões com o professor coorientador, nas quais foi expressa a vontade do grupo em trabalhar em um tema ligado ao assunto de cidades inteligentes e que pudesse ter relação com alguma aplicação da área do conhecimento da engenharia da computação.

A partir disso, foi notada a necessidade de quantização e extração de dados da realidade para alimentação do modelo de priorização de pedestres, o qual já se encontrava em desenvolvimento, para que sua concretização se tornasse possível.

Este trabalho também tem o objetivo de complementar o projeto de priorização de pedestres coordenado pelo Professor Claudio Luiz Marte e busca trazer métricas úteis e de difícil obtenção.

4.2 Projeto

O foco do trabalho foi definido para prover dados em tempo real que incluem, mas não se limitam a:

- quantidade de pedestres atravessando e à espera na área de armazenagem,
- suas velocidades de travessia,
- suas classificações etárias, de velocidade ou mesmo em categorias pré-definidas.

Dessa forma, a partir de estudos do estado da arte para aplicações de contagem em tempo real, foi visto que o uso de técnicas de visão computacional avançadas, que se encontram num crescente aprimoramento, seria ideal para o atingimento dos objetivos supracitados.

Portanto, foi realizada a definição das etapas de desenvolvimento do projeto que antecedem e sucedem o próprio treinamento do modelo. São essas:

- Aquisição dos dados de entrada esperados do modelo,

- Tratamento e anotação dos vídeos obtidos,
- Treinamento do modelo,
- Validação inicial da acurácia da predição do modelo,
- Manipulação dos resultados para obtenção de novos dados,
- Avaliação da performance geral do modelo.

4.3 Aquisição e tratamento dos vídeos

Ao final destas etapas é esperado que tenhamos disponíveis uma quantidade considerável de vídeos, os quais devem estar tratados de forma a obedecer a restrições de resolução, tamanho e requisitos de entrada para o correto treinamento do modelo.

Foram adquiridos vídeos em quantidade e variabilidade consideradas suficientes para o correto treinamento do modelo e obtenção de sua esperada generalização para os diversos cenários.

O tratamento dos vídeos envolve a redução de seus tamanhos, em questão de capacidade computacional necessária para sua armazenagem, mas também da anotação das imagens (cada quadro do vídeo será uma imagem) e a definição dos elementos que pretendemos identificar, assim como sua correta classificação.

A redução dos tamanhos deve ser realizada para a totalidade dos vídeos coletados, tendo em vista que este será um tratamento reproduzido também na aplicação final, a fim de minimizar custos com o posterior processamento destes.

Em contrapartida, a anotação de cada quadro só será necessária para um recorte predefinido de todo o material disponível. Isso se deve ao fato de que adotaremos a prática de treinamento de modelos de aprendizado de máquina no qual é esperada uma separação dos dados de entrada entre dados de treino, validação e, possivelmente, de teste.

Essa prática prevê o uso de diferentes dados para os diferentes processos com o objetivo de mitigar efeitos como o sobreajuste do modelo e garantir que a

generalização e a predição será feita com dados não antes vistos na etapa de construção e treinamento do algoritmo.

4.4 Treinamento e validação do modelo

O treinamento do modelo possui como requisitos a correta execução das etapas anteriores, visto que o formato dos dados de entrada é determinante para como o modelo será treinado e, consequentemente, a acurácia do seu resultado.

Dito isso, é esperado que estas etapas (tratamento, treinamento e validação) sejam recorrentes no processo de desenvolvimento, justificado pela alta dependência do sucesso da predição com os dados de entrada e pela já prevista revisão das etapas anteriores. Logo, serão realizadas iterações que compreendem as etapas supracitadas até a obtenção de métricas satisfatórias, correspondendo aos resultados esperados para o sucesso do projeto.

Dessa forma, é importante a definição, anterior à primeira validação, das métricas, com o intuito de que se mantenha uma regularidade das análises, possibilitando uma linearidade e melhor visualização dos avanços ou retrocessos após cada iteração do desenvolvimento.

4.5 Manipulação para obtenção de novos dados

O primeiro passo, após a obtenção da quantidade de pedestres previsto na etapa anterior, será o processamento das identificações obtidas para o cálculo de métricas como velocidade de travessia e também para uma categorização detalhada dos pedestres identificados, a fim de testar a possibilidade de prever a velocidade já a partir da categorização inicial do pedestre.

O processo de medir a velocidade do indivíduo consiste em acompanhar a marcação e garantir que o algoritmo entenda que, por sucessivos quadros do vídeo, marcações sequenciais e próximas espacialmente na imagem tratam da mesma pessoa.

Por sua vez, o processo de categorização inicial do pedestre prevê uma discriminação prévia de categorias ou patamares de velocidades dos pedestres e o correto treinamento da predição do modelo, de forma que ele identifique os pedestres e o classifique em uma das determinadas categorias, a qual terá uma velocidade de travessia já estabelecida e, a partir dela, calcular o tempo da travessia.

Ambos os processos são concorrentes e têm como objetivo em comum estimar o tempo mínimo necessário para a travessia. Além disso, ambos os cálculos estão sujeitos a variações na posição da câmera, que impacta diretamente a escala e proporção entre distância real percorrida e quantidade de pixels deslocados no vídeo.

Desta forma, ambos serão testados e o que apresentar melhor assertividade será utilizado na versão final do algoritmo.

4.6 Avaliação da performance

Finalmente, a última etapa consiste na avaliação dos resultados obtidos e sua aderência à contagem e cálculo da velocidade manuais.

A hipótese é a de que os dados obtidos manualmente serão mais precisos, portanto, servirão de base para avaliar a acurácia da predição do modelo.

Como explicado anteriormente, um recorte dos vídeos será utilizado para o teste do modelo e isso inclui os que serão sujeitos à análise manual e dos quais serão extraídos os dados que serão comparados à saída do algoritmo.

5. ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

5.1 Requisitos de aquisição dos dados de entrada

Dentre os requisitos levantados, encontramos como primeira providência a aquisição de vídeos em quantidade suficiente para a correta generalização do algoritmo nas diversas situações de clima, luminosidade, fluxo e componentes da via.

Dessa forma, listamos, em um pedido encaminhado à Companhia de Engenharia de Tráfego de São Paulo, os pontos a serem atendidos, assim como o elevado número de vídeos dos quais necessitamos para o treinamento do modelo.

A quantidade de vídeos, estimados para uma duração de 15 (quinze) minutos cada, terá de ser da ordem de 600 exemplares. Dentre as características dos locais cobertos pelos vídeos, destacamos os seguintes pontos:

- Devem apresentar a faixa de pedestres;
- Devem conter também a área de espera dos pedestres antes destes realizarem a travessia;
- Preferencialmente, devem conter semáforos de pedestres;
- Devem, em sua maioria, conter cruzamentos com carros;
- Devem, em alguns casos, conter ciclofaixas/ciclovias.

Além disso, para uma melhor adesão do modelo ao cenário real, é necessária uma diversidade de horários e climas. Portanto, estipulamos os seguintes horários, adquiridos em dias úteis:

Tabela 1: Horários do pedido de vídeos

6h-6h15	opcional
8h-8h15	obrigatório
9h-9h15	opcional
12h-12h15	obrigatório
13h13h15	opcional
15h-15h15	opcional

17h-17h15	opcional
18h-18h15	obrigatório
20h-20h15	obrigatório
00h-00h15	opcional

Como já mencionado, são necessários vídeos em condições atmosféricas diferentes para cada horário escolhido, tendo a seleção de três dias distintos, cada um com uma destas situações climáticas: ensolarado; chuvoso e nublado.

É importante a seleção de 50 a 200 localizações distintas, conforme características mencionadas. Destes locais é importante a presença de pelo menos três dias úteis nas condições climáticas citadas (ensolarado, nublado e com chuva). Para cada dia, trechos de pelo menos 15 min em horários com luminosidade e movimentação variados (pelo menos às 8h, 12h, 18h e 20h). Sendo assim seriam no mínimo 600 trechos de 15 min de vídeo, o que consideramos suficiente para o treinamento das redes neurais do projeto.

5.2 Requisitos de posicionamento da câmera

Após testes em diferentes imagens em situações diversas de enquadramento, elevação, resolução e distância da via, foi possível determinar um posicionamento adequado para a melhor performance do algoritmo desenvolvido neste trabalho.

É recomendado que a câmera se encontre a pelo menos 2,5 metros de distância do chão, a pelo menos 1 metro de distância do meio-fio, e que o centro da via corresponda ao centro do enquadramento da gravação ou imagem. As medidas de distância são mínimos aceitáveis para um adequado funcionamento da identificação, porém, medidas maiores, desde que não ultrapassem demasiadamente do mínimo, são encorajadas pois melhoram a visão panorâmica da via como um todo.

Estas especificações atendem, mas não estão limitadas, aos objetivos que incluem a identificação de faixas mais afastadas do ponto de vista da câmera e a minimização do impacto de obstáculos oclusivos.

5.3 Requisitos de condição das faixas de travessia

Como será visto nos testes, é primordial que as faixas de pedestres possuam uma coloração contrastante em relação à via, ou seja, ao asfalto.

Isso se faz necessário pois o algoritmo possui, dentro de suas etapas de tratamento de imagem, processamento de hierarquização das cores na escala de cinza, portanto, se a faixa não tiver sua cor suficientemente oposta ao escuro do asfalto, sua identificação será comprometida.

Por consequência, é necessária uma manutenção da pintura das faixas, principalmente as mais próximas às calçadas, para uma melhor percepção, por parte do algoritmo, e sua posterior identificação.

6. PROJETO E IMPLEMENTAÇÃO

6.1 Reconhecimento de Faixas e Calçadas

A primeira etapa do presente trabalho está relacionada ao reconhecimento de faixas de pedestre, calçadas e áreas de armazenamento de pedestres. Estas áreas serão usadas posteriormente para calcular parâmetros como densidade e fornecer informações para que o algoritmo possa entender se o indivíduo está esperando ou atravessando a via.

Desta forma, e tendo em vista que esta primeira etapa diz respeito a objetos ou itens estáticos em relação à totalidade dos frames que compõem o vídeo de entrada do sistema, ou seja, as gravações não sofrerão mudança do local de filmagem, foi possível fazer uso da opção menos custosa à capacidade de processamento da máquina: a biblioteca OpenCV 2.

A decisão leva em conta que não é necessário processar todos os frames que compõem o vídeo, pelas razões já descritas anteriormente. Assim sendo, é possível processar somente alguns frames do vídeo em questão, espaçados temporalmente, para recuperar a localização dos objetos procurados. Espera-se que os resultados não mudem, já que tratam do mesmo espaço, mas que eventualmente, na ocasião de diferenças, possam ser utilizados em conjunto para melhorar a acurácia da identificação sob diferentes circunstâncias.

Com a ferramenta escolhida, foi possível traçar estratégias de tratamento de imagem para que as formas geométricas referentes às faixas de pedestres pudessem ser reconhecidas pelo algoritmo.

6.1.1 Primeira Abordagem

A primeira forma encontrada para o reconhecimento da região de travessia dos pedestres inclui, resumidamente, a detecção de formas geométricas semelhantes a retângulos, os quais seriam delimitados por caixas desenhadas pelo próprio código, por cima da imagem analisada.

Ao longo do desenvolvimento da solução, foi observado que a primeira abordagem, a qual busca destacar o contraste da imagem, obteve resultados satisfatórios quando aplicadas em imagens coletadas para a realização dos testes iniciais, conforme mostram a Figura 4 e a Figura 5.

Figura 4: Exemplo 1 dos testes iniciais para reconhecimento da faixa de pedestres

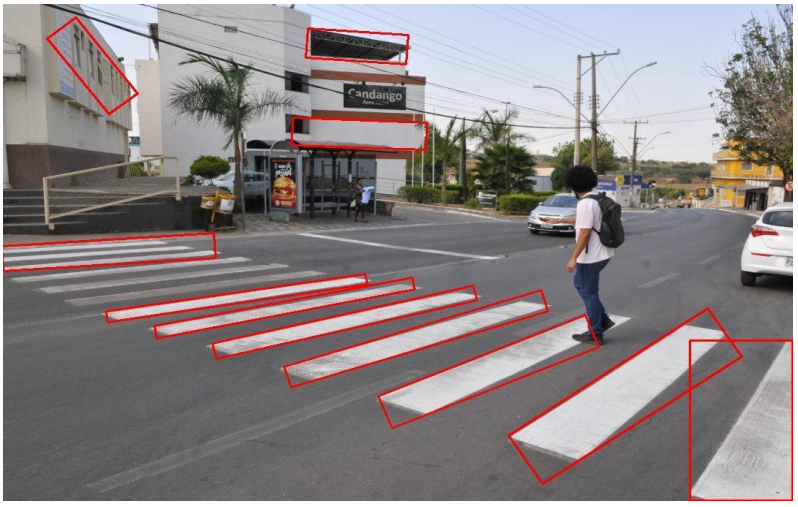

Figura 5: Exemplo 2 dos testes iniciais para reconhecimento da faixa de pedestres

No tratamento, além de trabalharmos as imagens na escala de cinza, é aplicado um filtro que ofusca a totalidade da foto, para que então seja aplicado um filtro final que distingue manchas acima de um determinado patamar na escala de cinza das

abaixo. Ao final da etapa de tratamento, observamos um exemplo a partir das figuras (imagem das manchas ao final do tratamento)

Figura 6: Tratamento de escala de cinza aplicado à imagem

Figura 7: Imagem original utilizada para o tratamento em escala de cinza

Em cima dessa imagem tratada, o algoritmo reconhece todos os contornos visíveis e elimina aqueles que não obedecem a algumas regras relacionadas à magnitude da área em relação à área de imagem e à proporção entre comprimento e altura, visto que as faixas não podem assumir a forma de um quadrado ou similar.

Em contrapartida, resultado semelhante não pôde ser alcançado na ocasião do recebimento dos primeiros vídeos, os quais são um pequeno exemplar do modelo

que será utilizado como entrada para o sistema de reconhecimento na aplicação real.

Tais vídeos apresentam qualidade muito inferior às imagens de teste, fato que agrava ainda mais a característica de estarem posicionados muito distantes do local de travessia dos pedestres, conforme a figura (frame de algum dos vídeos com poucos contornos identificados).

Figura 8: Quadro de uma gravação obtida por uma câmera de monitoramento instalada em São Paulo

Essa característica, já em algumas imagens utilizadas no teste inicial, afeta negativamente o desempenho do reconhecimento, conforme demonstra a Figura 9.

Figura 9: O distanciamento da câmera afeta a qualidade do reconhecimento de faixa de pedestres

6.1.2 Segunda Abordagem

A partir dos resultados preliminares obtidos, foi adotada outra abordagem, a qual consiste em identificar o início e o fim da faixa de pedestres, uma vez que a área da faixa é constante. Assim, contornamos a dificuldade de identificarmos cada ponto da faixa, bem como temos uma referência mais precisa, a qual é o meio fio da calçada.

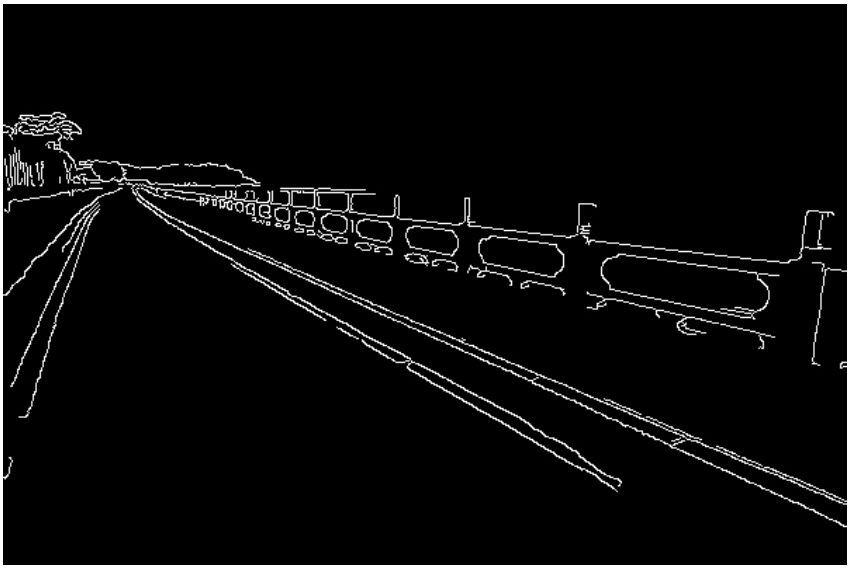

Figura 10: Tratamento da imagem com a função Canny

Figura 11: Resultado do prolongamento do meio fio na imagem

Na Figura 10, podemos observar o tratamento inicial planejado que, dentre outros, inclui a função “Canny”, nativa do OpenCV2, o qual nos auxiliará a desenhar retas e prolongá-las a fim de alcançarmos o resultado, conforme ilustrado na Figura 11

Nessa abordagem, podemos varrer tudo o que há entre as retas finais e iniciais, as quais serão os meios-fios, e, a partir de então, identificar a densidade de pedestres sobre a faixa e o seu respectivo tempo de travessia.

No entanto, a partir dos resultados preliminares obtidos seguindo esta abordagem, não foi possível, a partir dos tratamentos utilizados e testados, identificar as retas que correspondem à transição da via para a calçada.

Isso se deve à presença, em planos mais próximos, de elementos que também possuem o formato semelhante a uma reta, como postes, letreiros, cabeamentos e até mesmo carros, conforme ilustra a Figura 12, Figura 13 e Figura 14

Figura 12: Tentativa (1) falha no prolongamento das linhas do meio fio para reconhecer as faixas de pedestres

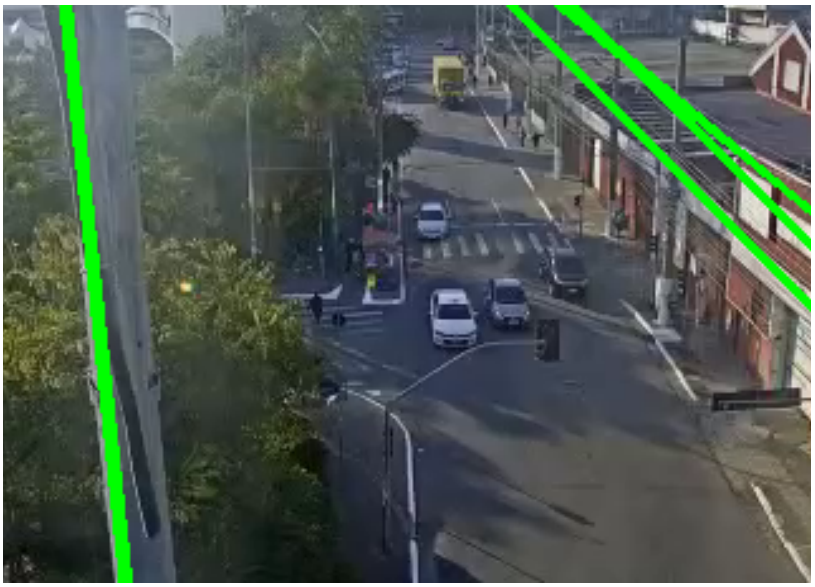

Figura 13: Tentativa (2) falha no prolongamento das linhas do meio fio para reconhecer as faixas de pedestres

Figura 14: Tentativa (3) falha no prolongamento das linhas do meio fio para reconhecer as faixas de pedestres

Isso se deve ao fato de que o contraste de cores na escala de cinza é mais forte entre os elementos supracitados, em comparação com a o meio-fio e o asfalto, ou mesmo entre as faixas de travessia e o asfalto.

6.1.3 Aperfeiçoamento da Primeira Abordagem

Tendo os resultados anteriores e os mais recentes em mãos, a decisão, em relação a qual abordagem seguir, foi de retomar a primeira abordagem, aplicando os filtros necessários para retirar as marcações que não identificam faixas de pedestres ou que identificam conjuntos de faixas fora do escopo da aplicação.

Esta decisão levou em conta o fato de que a primeira abordagem testada demonstrou maior potencial de identificação, pois, apesar de conter muitos ruídos e marcações indesejadas, mostrou-se capaz de também identificar os objetos-alvo. Portanto, o método de aperfeiçoamento teria como foco desenvolver uma etapa de

filtragem dos elementos ruidosos, restando apenas os retângulos que contenham as faixas da travessia da aplicação.

Para esta nova etapa, foram coletados novos vídeos gravados tendo em vista o objetivo da implementação de priorização de pedestres, logo, são gravações que se encaixam nos requisitos descritos no capítulo anterior e, portanto, serão utilizados para validar os resultados obtidos, os quais serão apresentados adiante.

Na Figura 15 abaixo, observamos o resultado utilizando o código construído até o final da primeira seção. É possível observar uma grande quantidade de marcações, inclusive ruidosas, mas, mais importante, uma boa quantidade das faixas é satisfatoriamente identificada também.

Figura 15: Reconhecimento das faixas utilizando a primeira abordagem com filtro de formas retangulares

O primeiro filtro pensado tem relação com a forma dos polígonos desenhados, visto que é possível prever a proporção mínima entre os lados. Dessa forma, polígonos que se aproximam mais à forma de um quadrado não se encaixam na descrição de uma faixa de travessia, portanto, deve ser ignorada pelo algoritmo.

O resultado, ainda com bastante marcações indesejadas, pode ser observado na Figura 16.

Figura 16: Reconhecimento das faixas utilizando a primeira abordagem com filtro de formas retangulares com descarte dos quadrados

É notável que uma melhoria, embora não suficientemente satisfatória, pode ser notada, principalmente em relação a marcações na porção superior da imagem, as quais fariam grande diferença na próxima etapa de filtros desenvolvidos, sendo de extrema importância que esses elementos já sejam excluídos do resultado final.

O próximo nível de filtragem dos elementos leva em conta a posição dos elementos dessa primeira iteração de identificação.

Cada um desses polígonos terá seu centroide, ou seja, a coordenada ou pixel que se encontra à sua metade da altura e largura, calculado. Posteriormente, será feita uma média aritmética dessas coordenadas, a fim de achar o ponto que se aproxima melhor da região de maior densidade de polígonos.

Esta etapa se assemelha, em alguma medida, a desenhar um raio em volta do centro de massa dos polígonos e manter somente as marcações que se encontram dentro desse raio de análise. No nosso caso, com apenas um centroide e somente

uma iteração, é esperado que esse ponto se encontre suficientemente próximo das marcações das faixas, visto que, inevitavelmente, há ruídos, mas há também identificações corretas e, crucial para o sucesso desse filtro, em maior número.

Na Figura 17 é observado o resultado final do aperfeiçoamento da identificação das faixas.

Figura 17: Resultado final da identificação das faixas de pedestres

Conforme ilustrado, todos os ruídos são eliminados após essa etapa final de filtragem de ruídos e marcações indesejadas. É importante notar também que algumas faixas, corretamente identificadas, também são eliminadas, inclusive faixas que não estão no escopo da gravação do vídeo e se encontravam no outro sentido da via.

Este menor número de identificações não compromete o sucesso do algoritmo, contanto que as presentes tenham sido identificadas corretamente e em quantidade suficiente para que a próxima etapa de construção da área de armazenamento de pedestres seja desenvolvida em sua totalidade.

6.2 Visão Geral Sobre o Código de Implementação

Como dito anteriormente, o código foi desenvolvido utilizando a biblioteca Open CV em linguagem Python.

Com isso, podemos entender um pouco mais do código que foi implementado para fazer a leitura de imagens e reconhecimento da faixa de pedestres.

Existem dois arquivos principais que podem ser utilizados tanto para a leitura de vídeos, tanto para a leitura de imagens, conforme a nomenclatura. Ambos possuem uma estrutura muito similar, sendo que a diferença entre eles é que um está preparado para receber fotos estáticas e outro vídeos como entrada de dados.

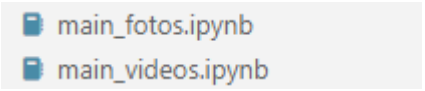

Figura 18: Arquivos para a execução do script

A partir da leitura de uma imagem estática ou de um quadro de vídeo, existem três funções principais para o reconhecimento das faixas de pedestres. São elas `treat_img` e `draw_rectangles` e `draw_rectangles_centroid`.

A função `treat_img` recebe uma variável que é a própria imagem lida previamente.

```
def treat_img(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray,
                               (15, 15), 4)

    ret, thresh = cv2.threshold(blurred,
                                140, 255,
                                cv2.THRESH_BINARY)

    contours, hier = cv2.findContours(thresh.copy(),
                                      cv2.RETR_TREE,
                                      cv2.CHAIN_APPROX_SIMPLE)

    show_image(thresh)

    return contours
```

Figura 19: Descrição da função "treat_img"

Dentro dessa função, são utilizadas outras quatro funções da própria biblioteca do Open CV que são:

`cvtColor`: O objetivo dessa função é modificar a escala de cores de uma imagem e, uma vez que queremos que a faixa de pedestres se destaque na imagem, a transformamos em uma figura preto e branco a partir do parâmetro `cv2.COLOR_BGR2GRAY`

- `GaussianBlur`: Como não podemos garantir a qualidade da imagem recebida, podemos tratá-la através dessa função que retira ruídos indesejados de uma forma gaussina. Ela recebe três parâmetros que são a própria imagem, o tamanho da matriz kernel da imagem e o desvio padrão. Os parâmetros utilizados foram (15,15) e 4 e foram encontrados empiricamente para obtermos um melhor resultado.
- `Threshold`: Agora, utilizamos essa função para encontrar os limites de objetos dentro da imagem. Enviamos o parâmetro `cv2.THRESH_BINARY` que vai delimitar de forma binária o início e o fim de um elemento, conforme a Figura 20.

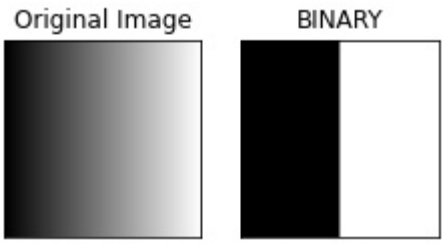

Figura 20: Exemplo de tratamento binário para encontrar limites utilizando o Open CV [5]

- `findContours`: Com a imagem devidamente tratada, finalmente utilizamos a função que encontra os contornos de elementos dentro da imagem. Aqui, temos o contorno de diversos elementos, dentre eles a da faixa de pedestres. Portanto, na função a seguir, iremos tratar esses dados para poder saber quais desses elementos se encaixam no critério de faixa de pedestres

Com os contornos dos elementos dentro da imagem devidamente identificados, podemos agora separar o que são faixas de pedestres e o que não é. Para isso, utilizamos a função `draw_rectangles`, a qual recebe a imagem em questão, os contornos encontrados anteriormente e um índice i que servirá para identificar a imagem de saída.

```
def draw_rectangles(img,i):
    H,W,colors = img.shape #get width and height

    for c in contours:
        # if the contour is not sufficiently large, ignore it
        if  cv2.contourArea(c) < (H*W)/300:
            continue

        if  cv2.contourArea(c) > (H*W)/10:
            continue

        # get the min area rect
        rect = cv2.minAreaRect(c)
        (x, y), (w, h), angle = rect
        aspect_ratio = max(w, h) / min(w, h)

        # Assume zebra line must be long and narrow (long part must be at lease 1.5 times the narrow part).
        if (aspect_ratio > 1.5):
            box = cv2.boxPoints(rect)
            # convert all coordinates floating point values to int
            box = np.int0(box)
            # draw a red 'nghien' rectangle
            cv2.drawContours(img, [box], 0, (0, 0, 255), 2)


        #cv2.imwrite('resultados/resultado_{}.jpg'.format(i), img)
        cv2.imshow("contours", img)
```

Figura 21: Descrição da função "draw_rectangles"

Primeiramente, realizamos um filtro de tamanho, uma vez que as linhas das faixas de pedestres não serão nem muito pequenas, nem muito grandes. Assim, filtramos os contornos que possuem áreas maiores do que um 300 avos da área total da imagem e menores do que um décimo. Esses parâmetros são um ajuste fino para o tipo de imagens que utilizamos e podem ser modificados de acordo com a posição da câmera de observação, o que pode modificar a proporção de ocupação da faixa de pedestres na imagem. Caso o contorne não se enquadre nesses parâmetros, o ignoramos.

Após isso, identificamos contornos que possuem uma proporção entre sua largura e altura maior do que 1,5, uma vez que as faixas de pedestres são retângulos esbeltos.

Passados esses filtros, podemos dizer que todos os contornos restantes são as próprias faixas de pedestres e, portanto, podemos desenhar esses contornos sobre imagem original e criar um arquivo de saída.

Com os retângulos filtrados, podemos dizer que a maior parte deles é parte de uma faixa de pedestres. Contudo, ainda pode haver na imagem elementos que se encaixam nos filtros anteriores e são indesejáveis, portanto, a próxima função faz um ajuste fino no método anterior.

Assim, realizamos um novo filtro de área e calculamos o centro do retângulo identificado e o centroide da faixa de pedestres inteira. Caso a distância entre os dois seja menor do que um décimo do máximo entre a altura e largura total da imagem, consideramos que esse retângulo faz parte de uma faixa de pedestres.

Finalmente, refazemos o mesmo processo final que realizamos na função anterior e finalizamos a identificação da faixa de pedestres na imagem.

Com isso, além da imagem demarcada, o programa consegue retornar um conjunto de contornos demarcando a faixa de pedestres, os quais podem ser utilizados para calcular a área de interesse e, posteriormente, com a identificação de pedestres, é possível analisar importantes parâmetros, como a densidade de ocupação dela, bem como a velocidade de travessia e outros parâmetros que sejam relevantes para a semaforização.

```python
def draw_rectangles_centroid(img,i,centroid):
    H,W,colors = img.shape #get width and height

    for c in contours:
        # if the contour is not sufficiently large, ignore it
        if  cv2.contourArea(c) < (H*W)/2000: #lower limit
            continue

        if  cv2.contourArea(c) > (H*W)/100: #upper limit
            continue

        # get the min area rect
        rect = cv2.minAreaRect(c)
        (x, y), (w, h), angle = rect
        aspect_ratio = max(w, h) / min(w, h)
        center = [x + 0.5*w,y + 0.5*h]


        # Assume zebra line must be long and narrow (long part must be at lease 1.5 times the narrow part).
        if (aspect_ratio > 6):
            if dist(center,centroid) < (max(H,W)/10):
                box = cv2.boxPoints(rect)
                # convert all coordinates floating point values to int
                box = np.int0(box)
                # draw a red 'nghien' rectangle
                cv2.drawContours(img, [box], 0, (0, 0, 255), 2)


    cv2.imwrite('resultados/resultado_{}.jpg'.format(i), img)
    #cv2.imshow("rectangles", img)
```

Figura 22: Descrição da função "draw_rectangles_centroid"

Por fim, temos um diagrama esquemático das funções que compõem o processo de identificação de pedestres.

Figura 23: Esquema de funcionamento dos macroprocessos do modelo de identificação das faixas de pedestres

7. TESTES E AVALIAÇÃO

A partir de testes em imagens fora das especificações da aplicação, foi possível entender com mais detalhes os casos que serão melhor atendidos pelo algoritmo.

Como observado nos resultados abaixo, o algoritmo é muito sensível a obstáculos que obstruem a visão completa das demarcações da faixa de pedestre, como pessoas, carros, postes e até mesmo sombras.

A distância e a posição da câmera em relação à via também impactam diretamente na performance do algoritmo, visto que, uma visão elevada e/ou mais afastada da via facilita a identificação de faixas que estão mais distantes, assim como minimiza a oclusão das mesmas pelos obstáculos supracitados.

Figura 24: Teste final de reconhecimento 1

Figura 25: Teste final de reconhecimento 2

Figura 26: Teste final de reconhecimento 3

Figura 27: Teste final de reconhecimento 4

Figura 28: Teste final de reconhecimento 5

Desta forma, é importante ressaltar o atendimento às especificidades de posição e resolução da câmera, bem como da qualidade e condição da pintura das faixas, para que o algoritmo possa performar da melhor forma possível.

8. CONSIDERAÇÕES FINAIS

8.1 Conclusões do Projeto de Formatura

Neste momento, cabe aqui revisitarmos o objetivo inicial do projeto, que era o estudo de técnicas de Visão Computacional e posterior desenvolvimento de softwares e algoritmos de Deep Learning, por meio de ferramentas como o OpenCV, capazes de reconhecer e categorizar diferentes perfis de pedestres a partir de vídeos.

Baseado nisso, caminhamos parcialmente em direção a esse objetivo, que foi a efetiva identificação da faixa de pedestres. Por isso, é importante entender porque o projeto se desenvolveu nessa direção e quais são as consequências disso.

Ao longo do projeto, foi percebido que a identificação da faixa de pedestres traria uma contribuição mais rápida e com maior impacto do que a criação do perfil de pedestres, em um primeiro momento. Isso acontece porque ao identificar a área de travessia, é possível então analisar a densidade de ocupação nela, bem como a velocidade de travessia dos pedestres.

Conforme as imagens reais de cruzamentos em São Paulo foram nos dadas, percebeu-se que as condições de ambiente, como posição da câmera, luminosidade, resolução das imagens e entre outros fatores, eram limitantes para a identificação dos elementos na figura. Assim, ao invés de tentarmos identificar os detalhes de cada pedestre, seria mais relevante caracterizar uma grande área de interesse, que é a faixa de pedestres.

Ainda que com esse objetivo redefinido ao longo do projeto, percebemos que ainda haveria dificuldades relacionadas às condições de imagens, já supracitadas.

Além disso, foram encontradas limitações quanto à tecnologia de visão computacional utilizada, visto que atualmente há opções mais avançadas e que, para casos similares ao deste trabalho, apresentam resultados mais precisos e com menos ruídos. No entanto, conforme mencionado anteriormente, as especificações do hardware em que o algoritmo desenvolvido será executado limitam a capacidade

de processamento destas alternativas mais sofisticadas, pois apesar de apresentarem resultados melhores, também requerem mais recursos computacionais para seu uso.

A partir de então, o projeto se desenvolveu atingindo os resultados comentados nos itens anteriores.

8.2 Contribuições

Aquele que tiver interesse na análise deste presente projeto, poderá encontrar duas contribuições fundamentais para a área de reconhecimento de objetos aplicado ao trânsito: estado da arte e aplicações iniciais.

Primeiramente, no estado da arte, foi realizado um levantamento amplo sobre o que já existe a respeito do tema no Brasil e no Mundo e observamos que essa área ainda carece de um amplo desenvolvimento. Apesar de o reconhecimento de máquina e o Deep Learning já ser avançado, há uma interface muito baixa entre a aplicação desse conhecimento e o controle de tráfego, muito menos o controle semafórico.

Logo, o trabalho trouxe à luz o grande vácuo que existe na área e, aqueles que porventura tiverem interesse nesse campo, terão muito a contribuir para a construção do conhecimento.

Já nas aplicações iniciais, as dificuldades que foram encontradas trouxeram outras duas contribuições nessa alçada. Primeiro, as imagens disponíveis para o monitoramento de tráfego são fatores limitantes para a análise e devem ser levadas em consideração ao se propor trabalhar nessa área. Deve-se entender que caso se deseje trabalhar em cooperação com órgãos com uma infraestrutura já existente, como foi o caso com a CET de São Paulo, haverá restrições na qualidade do material fonte a ser trabalhado.

Passada a fase de aquisição de dados e seus fatores delimitantes, a segunda contribuição da aplicação que foi feita é o entendimento de como trabalhar no tratamento de imagens. Nos itens 6 e 7 foram desenvolvidos e descritos o tratamento de imagens com os seus devidos parâmetros e testes otimizados para melhor identificar as faixas de pedestres. A partir desses experimentos, o leitor pode ser capaz de começar a desenvolver um outro projeto que desejar já tendo em mente as melhores práticas e técnicas para se obter uma otimização na identificação desse elemento nas figuras recebidas.

8.3 Perspectivas de Continuidade

O conceito do projeto ainda não pôde ser finalizado e para isso, ele poderia se seguir em outros cinco passos:

1. Melhora das especificações da coleta de imagens, fazendo com que as câmeras fossem posicionadas de modo específico para maximizar a efetividade da leitura das imagens pelo computador, bem como melhorar a resolução das imagens recebidas
2. Melhoria dos parâmetros de tratamento das imagens e vídeos para uma melhor aderência a uma maior gama de casos
3. Avaliar a densidade de pessoas em uma faixa e a sua velocidade de travessia até o final dela
4. Enviar os dados para um controle semafórico e analisar a sua melhora a partir das novas informações
5. Partir para a identificação de pedestres e sua classificação, fazendo com que o controle semafórico seja de fato ajustado em tempo real de acordo com a demanda instantânea.

9. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Manlises,Cyrel O., Martinez Jr.,Jesus M., Belenzo,Jackson L., Perez, Czarleine K. e Postrero, Maria Khristina Theresa A.; “Real-Time Integrated CCTV Using Face and Pedestrian Detection Image Processing Algorithm For Automatic Traffic Light Transitions”; 8th IEEE International Conference Humanoid, Nanotechnology, Information Technology Communication and Control, Environment and Management (HNICEM); 2015

[2] Rego, Rosana Cibely B. e Semente, Rodrigo S., “Presence sensor for people detection and reduction of pedestrian waiting time in traffic light”. Anais do Encontro de Computação do Oeste Potiguar ECOP/UFERSA, 2017

[3] TECHTUTORIALSX. “Python OpenCV: Converting an image to gray scale”, 2021. Disponível em: *https://techtutorialsx.com/2018/06/02/python-opencv-converting-an-image-to-gray-scale/.* Acesso em: 01 dez. 2021

[4] OPENCV. “Smoothing Images”, 2021. Disponível em: *https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html*. Acesso em: 05 nov. 2021

[5] OPENCV. “Image Thresholding”, 2021. Disponível em: *https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html*. Acesso em: 15 nov. 2021