Reconhecimento de placas de trânsito na plataforma Android para sistemas de assistência avançada de condução

Esdras Vitor Silva Pinto
Engenharia Elétrica - UFMG - esdrasvitorsi@gmail.com

1. Introdução

Muitos veículos atuais, contam com sistemas sofisticados que visão auxiliar a direção do motorista, aumentando a segurança e o conforto do veículo. Tais tecnologias são conhecidas como sistemas de assistência avançada de condução (ADAS - Advanced Driver Assistance Systems). Os ADAS fornecem informações relevantes aos motoristas sobre o ambiente ao redor do veículo, aumentando a sua segurança e dirigibilidade. Dentre os vários sistemas ADAS, como detectores de faixa [2] e estacionamento auxiliado [3], destacam-se os sistemas capazes de detectar e reconhecer placas de trânsito (TSR - Traffic Sign Recognition) [1].

Sistemas TSR captam imagens por meio de uma câmera apontada para frente do veículo, ou seja, a câmera possui a mesma visão que o motorista. O algoritmo dos sistemas TSR é composto basicamente por duas partes. A primeira parte consiste na identificação de potenciais regiões que podem conter placas de trânsito em um frame [1]. A segunda envolve a análise dessas regiões, reconhecendo e classificando a potencial placa. Portanto, o algoritmo é composto por uma fase de detecção e outra de reconhecimento.

Na etapa de detecção, são empregadas informações de cores para a seleção de áreas de interesse. Como as placas possuem formas e cores definidas, o uso de cores se apresentam convenientes em suas identificações. Por exemplo, a maioria das placas de trânsito brasileiras possuem as cores branca, azul, amarelo, verde e vermelho. Adicionalmente, as formas mais comumente utilizadas são circulares e retangulares. Com as informações de cores aliadas às informações de geometria, as regiões mais propensas de ser uma placa de trânsito são identificadas [4].

Na parte de reconhecimento, existem duas sub etapas. A primeira é a extração de características das regiões provenientes da fase de identificação. Essas características são pontos na imagem que são

suficientemente estáveis e característicos da imagem em questão. Vários algoritmos têm sido utilizados para extrair características de imagens, como histograma de gradientes orientados HOG (histogram of oriented gradients) [5,6], extração robusta acelerada (SURF - speeded-up robust feature), característica com invariância de escala (SIFT) [7], dentre outros. Uma vez que conhecemos os pontos característicos de um objeto, pode-se buscar por essas características em outra imagem. É exatamente isso que é feito na segunda etapa do reconhecimento, o qual envolve aprendizado de máquina.

Para que o algoritmo do TSR seja capaz de reconhecer placas de trânsito, ele precisa ser treinado. O algoritmo de extração de características é aplicado em um conjunto de imagens que representam genuinamente uma placa de trânsito. Essas características são tratadas e armazenadas em um banco de dados. Uma vez que se conhecem as características das placas de trânsito, o reconhecimento ocorre por meio de comparação das características salvas no banco de dados com aquelas que foram extraídas da região de interesse. Se houver uma correspondência entre essas características acima de um valor limiar estabelecido, a placa pôde ser reconhecida. Por outro lado, se não houver correspondência suficiente, nenhuma placa será reconhecida na área de interesse.

Uma característica importante para um sistema TSR é o grau de confiabilidade. Para que esse sistema seja parte dos ADAS, é crucial o reconhecimento correto de placas nas mais variadas situações do ambiente. Ou seja, esses sistemas devem ser invariantes à iluminação, onde serão capazes de fazer o reconhecimento tanto durante o dia quanto à noite. Além disso, esses sistemas precisam ser capazes de reconhecer placas parcialmente ocultadas, desgastadas com o tempo, e inclinadas. Outro desafio que os sistemas TSR enfrentam é a operação em tempo real. Para que o reconhecimento seja possível até mesmo em velocidades elevadas, existe um tempo máximo na ordem de milissegundos para o TSR identificar e reconhecer os sinais de trânsito em um frame. Como o algoritmo envolve varias etapas de processamento de imagem, operar em tempo real é um desafio.

1.1 Motivação

As placas de trânsito são importantes mecanismos para garantir a dinâmica e segurança no trânsito. Sempre que elas são ignoradas, como por exemplo, parada obrigatória e limite máximo de velocidade, aumenta-se o risco de um potencial acidente. A simples observância às placas de trânsito contribui significativamente para o aumento da segurança nas vias e rodovias do país. Neste contexto de segurança e auxilio ao motorista, sistemas TSR apresentam grande relevância. Tais sistemas matém o motorista ciente das informações visuais das placas, emitindo alertas no painel e auxiliando o motorista a observá-los. Por exemplo, o sistema TSR pode alertar que há uma parada obrigatória à frente ou se a velocidade do veículo está acima da permitida no local.

Além de ser um importante sistema de assistência avançada de condução, sistemas TSR são também parte chave dos carros autônomos. Também aliado à segurança, carros autônomos estão se tornando uma realidade. Empresas como Google, já estão em fase avançadas de testes, e a confiabilidade tem sido surpreendente. Portanto, o reconhecimento de sinais de trânsito também é utilizado em carros autônomos [1].

1.2 Objetivos

Nesse trabalho de final de curso, será implementado os algoritmos de detecção e reconhecimento na plataforma Android. A motivação pela escolha da plataforma Android se deve ao fato de ser uma plataforma com grande desempenho computacional, e compacto, podendo ser facilmente embarcado em um veículo. Adicionalmente, é possível desenvolver uma solução completa do sistema TSR, incluindo captura de imagem por câmera, processamento para detectar e reconhecer as placas, e mostrar os resultados no display. Para a extração de características, será utilizado o algoritmo SURF. Conforme [8], SURF possui invariância à escala, invariância a rotação e pode ser utilizado para reconhecimento de objetos parcialmente ocultos. Essas características do SURF, permitem a sua utilização na implementação de sistemas TSR robustos, que sejam capazes de lidar com condições variadas do ambiente.

2. Revisão bibliográfica

Sistemas de detecção e reconhecimento de placas de trânsito têm resultado em várias pesquisas na área de visão computacional. Várias plataformas de implementação foram propostas até o momento da escrita desse trabalho, incluindo computadores de uso geral, FPGA, chips customizados para processamento de imagens e processamento digital de sinal, sistemas embarcados dedicados, como Raspberry Pi, e também processadores gráficos (GPU - Graphic Processing Unit).

As implementações dos sistemas TSR em computadores de uso geral, têm se mostrado rápidas o suficiente para permitir detecção em tempo real. Conforme apresentado em [9], o processo de reconhecimento, o qual envolve extração de características da região de interesse por meio do algoritmo SIFT e o tempo de classificação, foi realizado com apenas 81ms em um computador de uso geral operando a 2.4GHz com dois núcleos e sistema operacional Linux. Além da rápida execução da fase de reconhecimento, a metodologia proposta neste trabalho atingiu uma confiabilidade de 95%. Embora tenham sido obtidos resultados promissores, tanto de desempenho quando de confiabilidade, a solução apresentada é difícil de ser embarcada em um veículo.

Ainda no contexto dos computadores de uso geral, algumas implementações utilizam processadores gráficos (GPU) para paralelizar maciçamente os algoritmos de detecção e reconhecimento. Essa paralelização aumenta consideravelmente o desempenho. No trabalho [10], um sistema TSR foi implementado na plataforma Tesla K20 GPU da NVIDIA. A execução da detecção e reconhecimento levou entre 13 a 17 ms, apresentando uma taxa média de 21 frames por segundo. Entretanto, o uso de GPU só se justifica para imagens de alta resolução, conforme mencionado pelos autores. Isto ocorre devido o overhead da linguagem CUDA necessário para instanciar de núcleos na GPU, o que impacta no tempo de processamento. Para compensar esse overhead, a imagem precisa ser de alta resolução, como 1236 x 1628 utilizada pelos autores. Por fim, os resultados obtidos nesse trabalho do ponto de vista de confiabilidade são bastante promissores, onde atingiu-se uma taxa de sucesso de reconhecimento de 93,77%.

Soluções alternativas aos computadores de uso geral e GPU, empregam hardware específico e possuem a vantagem de serem mais viáveis para uma situação real de uso do TSR em um veículo. Muitas dessas soluções se baseiam em hardware reconfigurável, como FPGA (Field-Programmable Gate Array). O trabalho [11] apresenta um sistema TSR implementado em FPGA, em específico, na FPGA Spartan-6. A fase de identificação de áreas de interesse é baseada em cores, e a parte de reconhecimento é baseada na extração e classificação de características. Embora o sistema seja passível de ser embarcado em um veículo, ele é capaz de reconhecer apenas placas de limite de velocidade.

O grande benefício da FPGA é a flexibilidade. Caso haja alguma melhoria ou modificação nos algoritmos, é possível reconfigurar a FPGA. Adicionalmente, as FPGAs modernas compartilham o interior do chip com um ou mais núcleos de processadores físicos. Isso permite obter um equilíbrio entre desempenho e flexibilidade, onde partes mais lentas do algoritmo podem ser realizadas em hardware, e partes que necessita de flexibilidade podem ser implementada em software. De fato, uma abordagem envolvendo hardware e software embarcado é apresentado em [1].

Alguns sistemas TSR foram implementados em chips dedicados de processamento digital de sinal (DSP - Digital Sign Processing). Conforme apresentado em [2], o chip de processamento digital de sinal OMAL L138 da Texas Instruments (TI) foi empregado em um sistema TSR, onde foi obtido um tempo total de processamento de 300ms por frame. Ainda que o tempo seja promissor para aplicação em tempo real, o algoritmo de reconhecimento utilizado, baseado em comparação de modelos, não é robusto o suficiente para detectar placas de trânsito parcialmente ocultadas ou rotacionadas [1].

Duas qualidades importantes para sistemas TSR é a operação em tempo real e robustez na detecção e reconhecimento. A maioria das soluções para sistemas TSR, seja em software ou hardware, operam em tempo real. Assim, é possível garantir que a detecção ocorrerá mesmo quando o veículo estiver em alta velocidade. Por alto lado, a confiabilidade ou robustez de detecção ainda está abaixo do desejado. O grau de confiabilidade dos sistemas TSR analisados variam entre 90 e 96%. Por se tratar de

uma aplicação crítica, onde uma classificação errada ou simplesmente o não reconhecimento do sinal pode causar graves consequências, como acidentes ou comportamento inesperado de um veículo autônomo, espera-se que a confiabilidade seja ainda superior a 96%.

Complementar os requisitos de operar em tempo real e ter confiabilidade de tecnicamente 100%, os sistemas TSR precisam ser embarcados em um veículo para terem utilidade no mundo real. As soluções discutidas envolvendo computadores de uso geral, embora possam ser embarcadas no veiculo, não são interessantes do ponto de vista de produção, devido ao alto custo e espaço ocupado. Por outro lado, soluções envolvendo sistemas embarcados dedicados, como FPGAs, são mais atraentes do ponto de vista de serem integradas aos veículos, pois são compactos e dependendo do tipo de FPGA usado, o custo de produção é menor do que computadores de uso geral. Essa é uma das principais razões na qual grande parte das pesquisas destinada a sistemas TSR serem desenvolvidas em sistemas embarcados.

Esse presente trabalho é inspirado em um trabalho similar desenvolvido durante meu intercâmbio acadêmico no Instituto de Tecnologia de Illinois [1], no qual foi implementado o algoritmo de reconhecimento do TSR em um sistema embarcado composto por uma FPGA no kit de desenvolvimento ZedBoard da Xilinx O algoritmo de reconhecimento usa o SURF para extrair características das imagens. Todo algoritmo de reconhecimento é implementado em software, embora se tenha a possibilidade de uso da FPGA para desenvolver aceleradores de hardware. Os resultados obtidos não são suficientes para operação em tempo real. Em algumas situações, o tempo total para o reconhecimento é maior do que 2s. Devido ao elevado custo e à dificuldade de se encontrar uma câmera destinada à Zedboard, optou-se por implementar todo algoritmo na plataforma Android.

3. Metodologia

Nessa seção, é apresentada a ideia geral do funcionamento dos algoritmos de detecção e reconhecimento utilizados neste trabalho.

3.1 Detecção

As cores dos pixels de uma imagem são determinadas de acordo com o espaço de cor no qual a imagem é definida. Um espaço de cor bastante comum é o RGB. Este espaço baseia-se na combinação das cores vermelha (R-Red), verde (G-Green) e azul (B-Blue) para produzir diferentes cores para os pixels. Outro espaço de cor é o HSI (H - Hue, S - Saturation, I - Intensity). A componente hue especifica a cor do pixel em forma de angulo entre 0º e 360º. Por exemplo, a cor vermelha é representada em 0º, a cor verde em 120º, e a cor azul é especificada em 240º. A saturação indica a quantidade de luz branca presente na cor, variando na faixa de 0 a 1. A intensidade também se encontra na faixa de 0 a 1, onde 0 resulta em preto e 1 em branco. A figura X ilustra a comparação entre os espaços RGB e HSI.

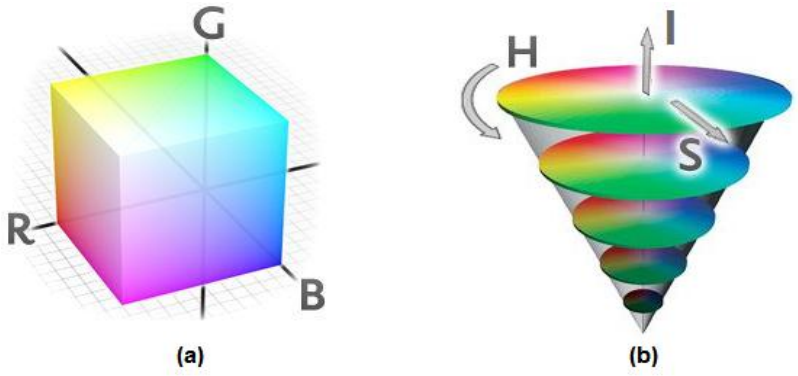


Figura 1: Ilustração do espaço de cor RBG (a) e HSI (b)

A detecção de placas de trânsito em uma imagem é feita usando-se segmentação de cores, isto é, certas cores de interesse são extraídas e posteriormente analisadas. O espaço RGB não é apropriado para segmentação por cores, já que a informação de cor está misturada em três componentes. Por outro lado, o espaço HSI é mais indicado para esta finalidade, já que a componente de cor (hue) pode ser separada da componente que indica o grau de sua pureza (saturação). Como as imagens de entrada utilizadas neste projeto estão no formato RGB, é necessária uma etapa inicial de conversão para HSI, o qual é feito pelas relações a seguir [1].

$$H = \begin{cases} 0, R = G = B \\ \frac{(G-B)60°}{max(R,G,B) - min(R,G,B)} mod\ 360°, R \geq G, B \\ \frac{(B-R)60°}{max(R,G,B) - min(R,G,B)} + 120°, G \geq R, B \\ \frac{(R-G)60°}{max(R,G,B) - min(R,G,B)} + 240°, B \geq R, G \end{cases}$$

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)]$$

$$I = \frac{1}{3}(R+G+B)$$

Figura 2: Relações de conversão de RGB para HSI

As placas de trânsito possuem formas e cores bem definidas. O processo de segmentação usa o fato da maioria delas possuírem as cores vermelha, amarela, branca, verde e azul para definir regiões de interesse. Neste trabalho, apenas as placas de regulamentação do trânsito brasileiro serão detectadas. Como essas placas possuem bordas vermelhas ou são completamente vermelhas como no caso da placa parada obrigatória, a detecção se limitará em identificar regiões na imagem que são vermelhas. Conforme apresentado em [12], os limiares no espaço HSI para extrair a cor vermelha são os mostrados a seguir.

$$Pixel(i,j) = \begin{cases} Vermeho, se\ H(i,j) \leq 10\ ou\ H(i,j) \geq 300 \\ Outra\ cor, caso\ contrário \end{cases}$$

Trabalhando com a componente hue, torna-se a detecção mais robusta aos efeitos de variação de iluminação do ambiente, pois o espaço HSI separa o nível de luz branca presente na cor da cor propriamente dita.

Uma vez definido os limiares para a cor vermelha, a imagem em HSI é transformada em uma imagem binária, de modo que todos os pixeis que possuem valores de hue dentro dos limiares estabelecidos sejam '1' (branco) e os demais pixeis sejam '0' (preto). A figura 3 a seguir mostra a segmentação de um frame contendo uma placa de trânsito.

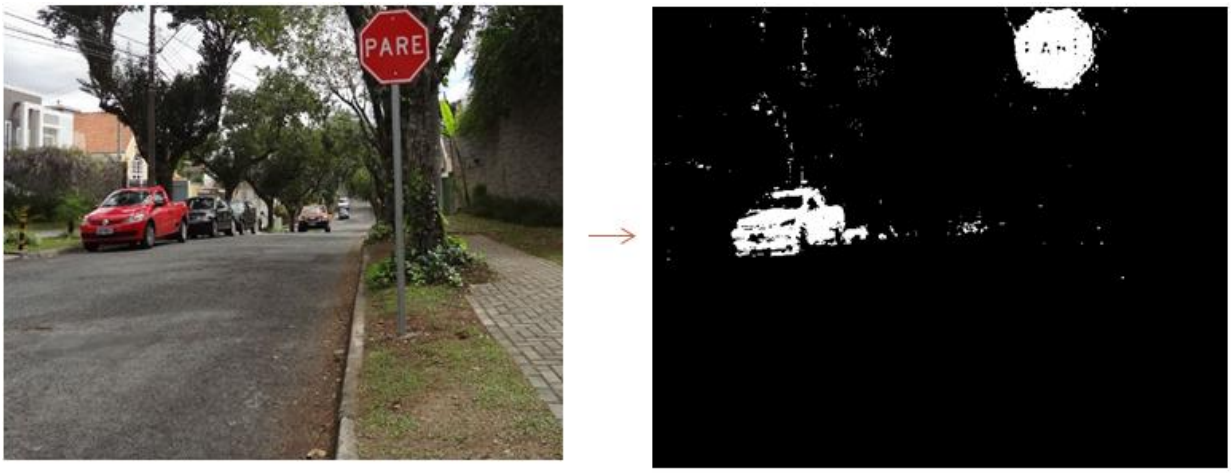


Figura 3: Conversão de um frame para imagem binária, onde apenas pixeis com as cores desejadas na imagem de entrada possuem a cor preta na imagem binária, e as demais regiões possuem cor branca

É possível observar que a imagem binária acima possui pequenos agrupamentos de pixels que não são capazes de fornecerem informações suficientes para a etapa de reconhecimento. Para eliminar parte desses ruídos, são aplicados filtros morfológicos [1]. Dois filtros morfológicos são utilizados nesse trabalho: erosão e dilatação.

O filtro de erosão tem o efeito de comprimir os agrupamentos de pixels, eliminando assim, os pequenos agrupamentos. O processo de filtragem é baseado no elemento estrutural mostrado na figura 4. Este elemento é posicionado sobre cada pixel da imagem binária. Se os pixels imediatamente a cima, em baixo, à esquerda e à direita forem todos '1', o pixel que esta sendo analisado (pixel central do elemento estrutural) é atribuído o valor '1'. Caso contrário, é atribuído o valor '0'.

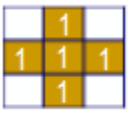


Figura 4: Elemento estrutural utilizado nos filtros de erosão e dilatação [13]

A figura 5 ilustra o processo de filtragem de erosão. O elemento estrutural é colocado sobre a imagem binária. Caso os pixels na imagem binária não apresentar valor '1' nas mesmas posições do elemento estrutural, como ilustrado na situação (1), o pixel na imagem filtrada é atribuída o valor '0'. Por outro lado, se os pixels na imagem binária possuem valor '1' em todas posições do elemento estrutural, o

pixel correspondente à posição central do elemento é atribuído o valor '1' na imagem filtrada (situação (5)). O elemento estrutural é deslocado pixel a pixel, percorrendo toda imagem, sendo obtida a imagem na figura 6 no fim do processo. Nota-se que de fato o filtro de erosão comprimiu a imagem original.

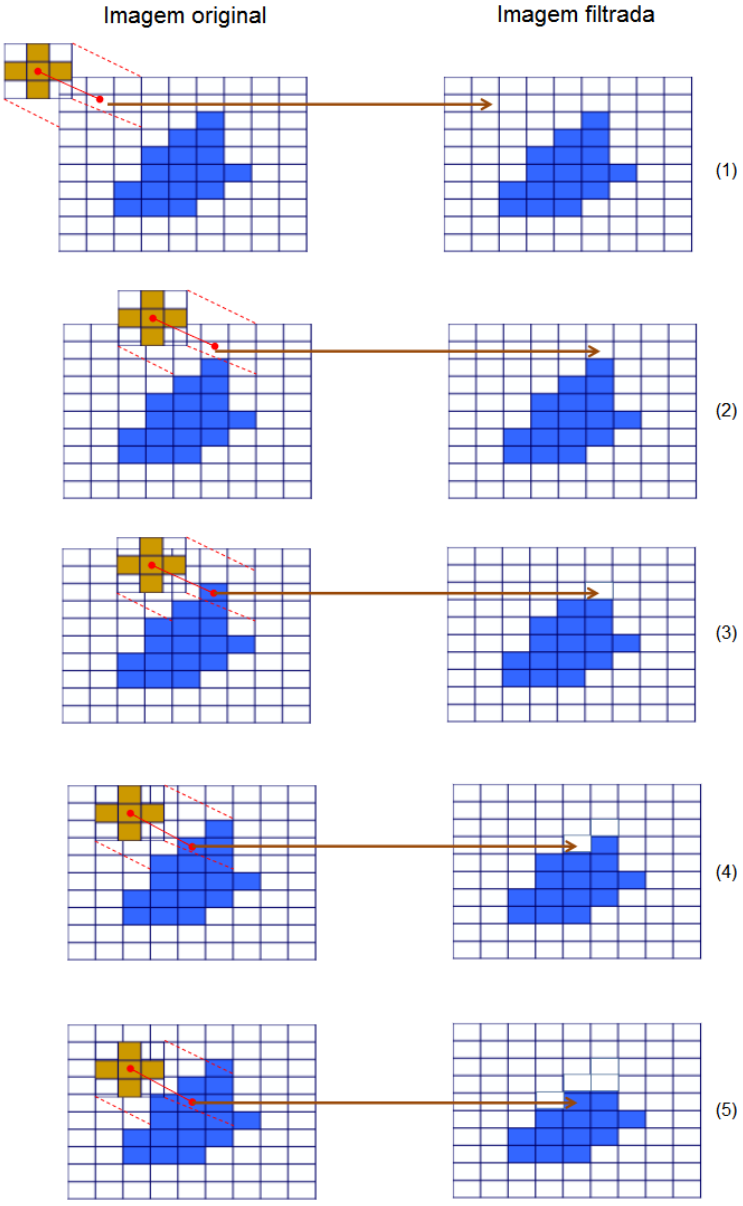


Figura 5: Ilustração do processo de filtragem por erosão [13]

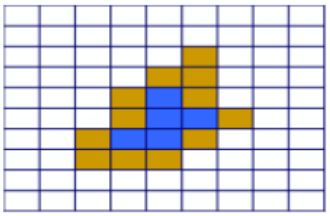


Figure 6: Resultado após filtro de erosão [13]

O filtro de dilatação tem o efeito oposto ao de erosão, isto é, os agrupamentos de pixeis são expandidos. O elemento estrutural utilizado na dilatação é o mesmo no caso da erosão, porém, para o pixel ser atribuído valor '1' na imagem filtrada, basta que pelo menos um pixel da imagem binária coincida com o elemento estrutural. A figura 7 ilustra o processo de filtragem de dilatação. Quando nenhum pixel da imagem binária coincide com os valores do elemento estrutura, como na situação (1), o pixel na imagem filtrada é atribuído o valor '0'. Por outro lado, se pelo menos um dos pixeis coincidem com o elemento estrutural, como na situação (2), o pixel na imagem filtrada é atribuído o valor '1'. O elemento estrutural é deslocado pixel a pixel de modo a percorrer toda imagem. O resultado da filtragem é mostrado na figura 8. Pode-se perceber que a imagem original foi expandida.

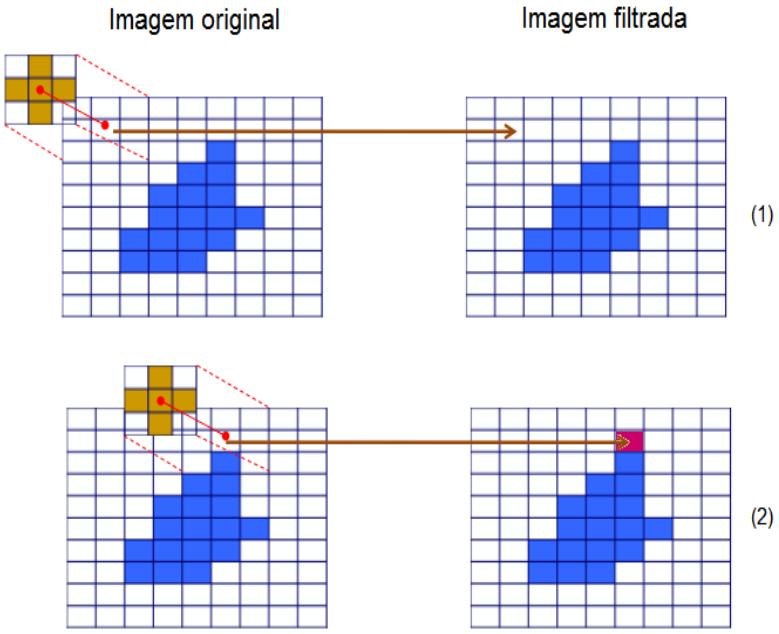


Figure 7: Ilustração do processo de filtragem por dilatação [13]

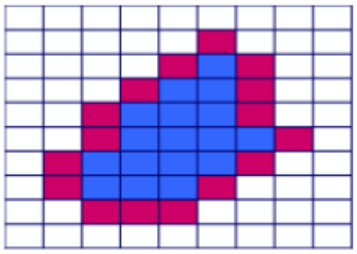


Figure 8: Resultado após filtro de erosão [13]

A redução de ruído é importante para evitar falsos positivos e economizar tempo de processamento na fase de reconhecimento. O filtro de erosão é aplicado na imagem binária seguida do filtro de dilatação. A erosão elimina pequenos grupos de pixels e a dilatação reforça os agrupamentos que restaram. A figura 9 mostra o resultado da aplicação de filtros morfológicos para redução de ruídos na imagem binária da figura 3.

Figura 9: Redução de ruídos na imagem binária através da aplicação de filtros morfológicos

Uma vez removido os ruídos, é preciso definir coordenadas para cada agrupamento de pixels. O algoritmo para realizar essa tarefa percorre toda a imagem, identificando grupos de pixels e atribuindo coordenadas para cada um deles. O algoritmo inicia a varredura de baixo para cima e da esquerda para direita. Para cada pixel, é atribuído um identificador numérico, cujo valor depende dos identificadores dos quatro vizinhos mais próximos do pixel que já foram analisados.

Por exemplo, considere a imagem da figura 10-(a). Os pixels ativos na imagem (brancos) precisam ser identificados. Ao percorrer a imagem, o algoritmo chegará no pixel P1. Os quatro vizinhos mais próximos desse pixel que já foram percorridos é mostrado na figura 10-(b). Como nenhum dos seus

quatro vizinhos possuem identificador, é atribuído o identificador 1 para este pixel. No caso do pixel P2, os quatro pixels vizinhos são os mostrados na figura 10-(c). Neste caso, um dos seus vizinhos já possuem identificador, e, portanto, o identificador atribuído ao pixel P2 também será 1. Já o pixel P5, nenhum dos seus quatro vizinhos foram identificados, e, portanto, será atribuído o identificador 2, uma vez que o identificador 1 já foi atribuído anteriormente. A figura 10-(d) mostra como fica as designações dos identificadores quando toda imagem for percorrida. Ao decorrer desse processo, as coordenadas Xmin, Xmax,Ymin e Ymax de cada identificador é armazenada. Quando toda imagem for varrida, tem-se às coordenadas dos retângulos que contêm os agrupamento de pixels, isto é, retângulos que envolvem as áreas de interesse.

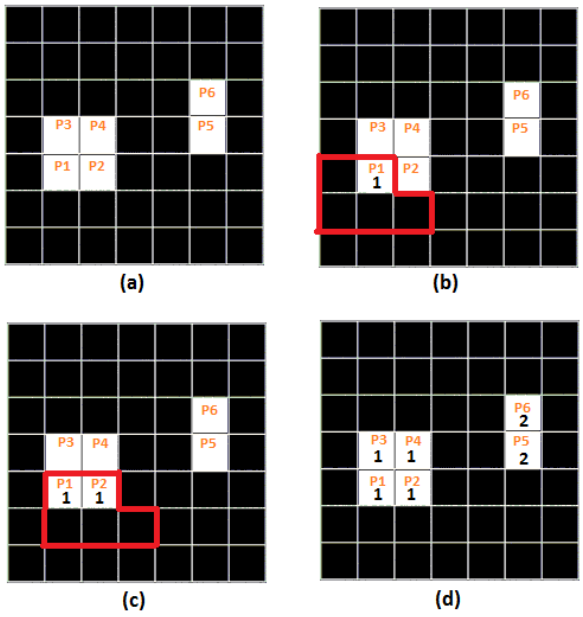


Figure 10: lustração do processo de identificação dos agrupamentos de pixels

Caso o agrupamento seja menor que um valor previamente determinando, certamente não será possível reconhecer placas nesta região, e ele será tratado como ruído. Por outro lado, caso o agrupamento seja denso o suficiente para realizar a detecção, definimos essa região como um candidato à placa de trânsito, ou seja, esse agrupamento se torna uma área de interesse. Todas as áreas de interesse são enviadas para a etapa de reconhecimento. A figura 11 resume as etapas envolvidas no algoritmo de detecção.

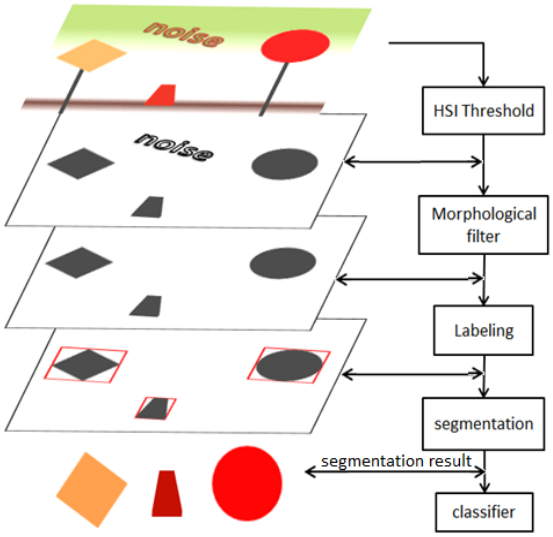


Figura 11: Sequência de operações na fase de detecção de placas de trânsito

3.2 Reconhecimento

Na etapa de reconhecimento, é utilizada a metodologia de extração de características. Essas características podem ser vistas como pontos matemáticos (pixeis) na imagem que são possíveis de serem encontrados em uma imagem similar. Neste caso, as características das regiões de interesse provenientes da etapa de detecção são extraídas e posteriormente comparadas com características dos modelos de placas de trânsito no qual o sistema foi treinado. Ou seja, características de imagens genuínas de placas de trânsito são extraídas na fase de treinamento do algoritmo e salvas no banco de dados. Na etapa de classificação, as características extraídas das áreas de interesse são comparadas com as características salvas no banco de dados.

Como o reconhecimento se baseia na extração de características, o tempo de execução na parte de reconhecimento é fortemente afetado pelo desempenho do algoritmo de extração. Nesse projeto, é utilizado o algoritmo SURF. Conforme [8], SURF é dito invariante à escala, rotação e oclusão parcial. Portanto, SURF é promissor para sistemas TSR, já que tais sistemas precisam ser capazes de reconhecer placas em diferentes escalas e também saber lidar com placas parcialmente ocultadas ou não alinhadas corretamente.

Na construção do banco de dados, o algoritmo SURF é aplicado sobre imagens genuínas de placas de trânsito e suas características são extraídas. A figura 12 mostra as características obtidas (destacadas pelos círculos verdes) para a placa pare.

Figura 12: Círculos verdes indicam as características extraídas pelo SURF para a placa PARE

Cada característica representada pelos círculos verdes na imagem acima, são descritos por um vetor de números reais de 64 posições. A comparação entre características é feita calculando-se a distância euclidiana entre elas. Por exemplo, se C1[64] e C2[64] representam duas características, a distância entre elas é dado por

$$D = \sqrt{(C1[0] - C2[0])^2 + (C1[1] - C2[1])^2 + \cdots + (C1[64] - C2[64])^2}$$

Se a distância D for menor que um limiar preestabelecido, as características podem ser consideradas semelhantes.

É bastante comum que o banco de dados seja composto por muitas características. Portanto, a comparação pelo método de força bruta, o qual compara cada característica do candidato com todas as características do banco de dados, é demasiadamente lenta e ineficiente.

Uma forma interessante de otimizar o processo de comparação é através da estruturação do banco de dados em árvore. Neste trabalho, o algoritmo K-means é utilizado neste processo de estruturação. Este

algoritmo analisa todas as características do banco de dados e calcula dois pontos, também conhecidos como centróides, baseados nessas características. Os centróides são utilizados como referência para dividir as características em dois grupos. Para isto, calcula-se a distância de todas as características do banco de dados em relação aos centróides. Em seguida, as características são inseridas no grupo cuja distância ao centróide é menor. A figura 13 ilustra a primeira divisão do banco de dados.

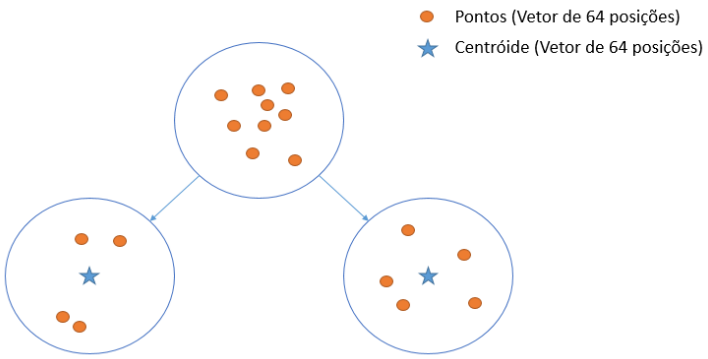


Figure 13: Ilustração da primeira divisão do banco de dados

À medida que o processo descrito acima é aplicado novamente nos subgrupos, obtém-se uma estruturação em árvore do banco de dados. Com o banco estruturado desta forma, a verificação de similaridade entre características é feita inicialmente calculando-se a distância das características dos candidatos aos centróides. O grupo cuja distância da característica ao centroid for menor é escolhido. Em seguida, a característica é comparada novamente com os centróides dos subgrupos do grupo escolhido. Quando for atingido o último nível da árvore, a característica do candidato à placa de trânsito é comparada com todas as características do grupo. Percebe-se então que a estruturação do banco de dados reduz consideravelmente a complexidade do algoritmo de reconhecimento, tornado a busca muito mais ágil e eficiente.

A figura 14 ilustra a correspondência entre o modelo da placa proibida virar à esquerda no banco de dados e um candidato à placa de trânsito. Nesse trabalho, considera-se que o mínimo de correspondência para validar uma placa é 10.

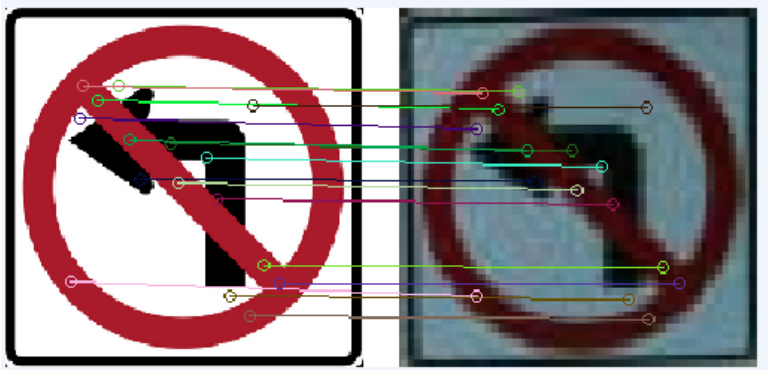


Figure 14: Correspondência entre as características do modelo no banco de dados para a placa proibido virar à esquerda (esquerda) e as características do candidato a ser classificado como placa de trânsito (direita).

A figura 15 resume as etapas envolvidas no algoritmo de reconhecimento.

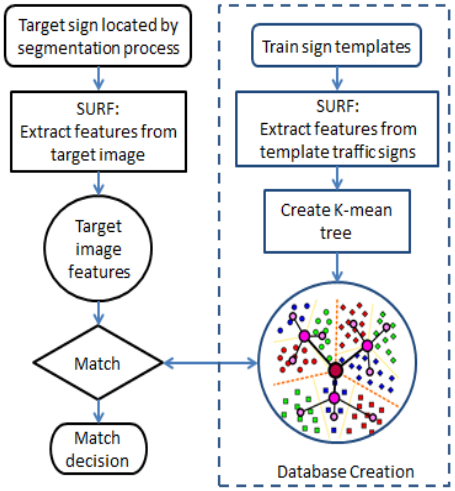


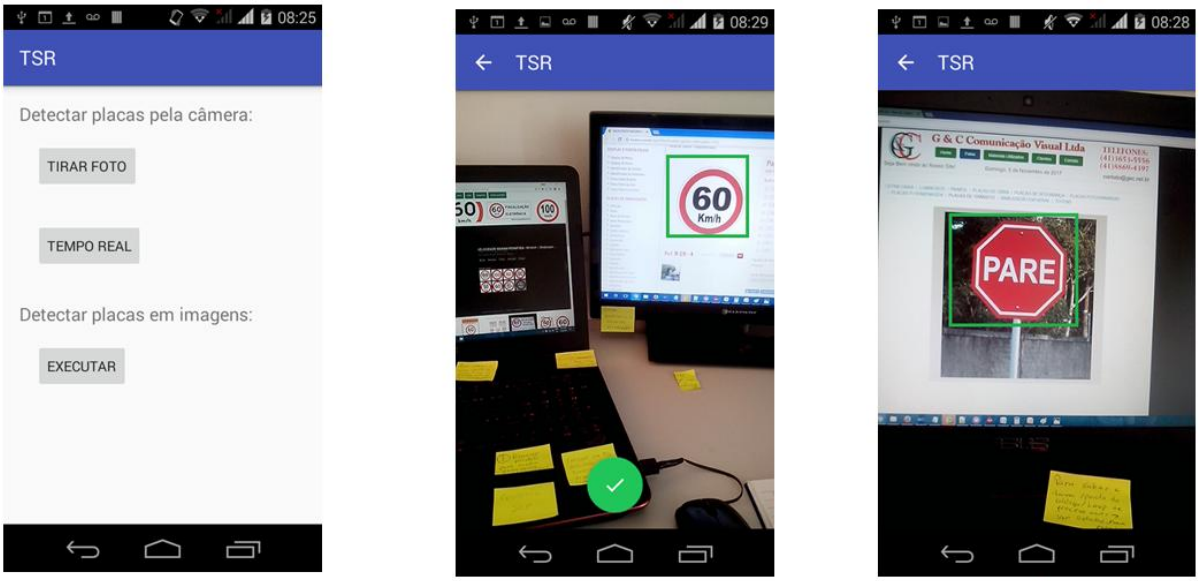
Figure 15: Sequência de operações na fase de reconhecimento

4. Resultados

Os algoritmo de detecção e reconhecimento descritos nas seções 3.1 e 3.2, respectivamente, foram implementados para rodar na plataforma Android e desenvolvidos no ambiente Android Studio. O algoritmo de detecção foi codificado nativamente sem o uso de bibliotecas de processamento de imagens e a implementação do SURF foi baseada na versão do OpenSURF. A versão original do OpenSURF utiliza várias funções e estruturas de dados da biblioteca de visão computacional OpenCV. Como a princípio optou-se em não utilizar esta biblioteca, o código original do OpenSURF foi portado para

Android, onde todas as funções do OpenCV foram removidas do código e substituídas por funções e estruturas próprias.

O aplicativo desenvolvido possui três funcionalidades principais, como pode ser visto na sua tela principal na figura 16-(a).



(a) (b) (c)

Figura 16: Telas do aplicativo desenvolvido

O recurso "Tirar Foto" permite a obtenção de placas de trânsito para testes dentro das dimensões mínimas exigidas pelo sistema. A tela mostrada na figura 16-(b) é referente ao uso deste recurso. O quadrado verde especifica as dimensões mínimas e uma foto pode ser armazenada através do botão verde na parte inferior da tela. O recurso "Tempo Real" executa a detecção e reconhecimento em tempo real de imagens provenientes da câmera do dispositivo. A tela na figura 16-(c) mostra a detecção em tempo real de uma placa. Por fim, o ultimo recurso é voltado para teste das imagens em geral, sendo útil para obtenção de resultados de confiabilidade e tempo de execução mostrada nas próximas seções.

O algoritmo foi treinado apenas para reconhecer placas de parada obrigatória. Novas placas podem ser adicionadas ao sistema bastando inserir suas características ao banco de dados.

4.1 Algoritmo de detecção

A imagem mostrada na figura 17 foi utilizada como imagem de entrada do algoritmo.

Figura 17: Imagem de entrada do algoritmo de detecção

A obtenção de regiões de interesse é feita aplicando-se um processamento de segmentação na imagem de entrada. Essa segmentação tem por objetivo identificar todos os pixels na imagem que são vermelhos. A figura 18 mostra a imagem binária resultante, onde os pixels brancos representam as áreas vermelhas.

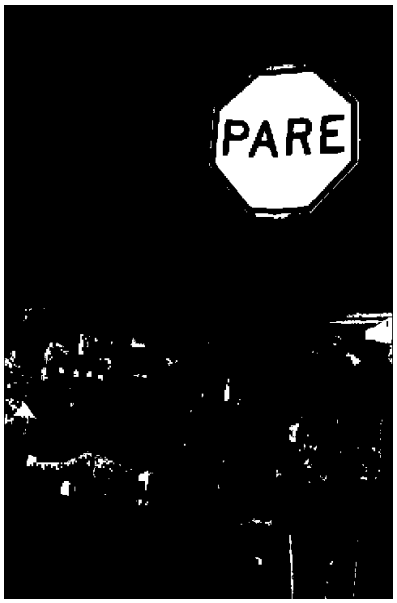


Figura 18: Imagem binária resultante da segmentação

Os ruídos na imagem de entrada são reduzidos com aplicação de filtros morfológicos de erosão e dilatação. Inicialmente, aplica-se o filtro de erosão, seguido pelo filtro de dilatação. A figura 19 mostra como pequenos agrupamentos de pixels são eliminados com a filtragem.

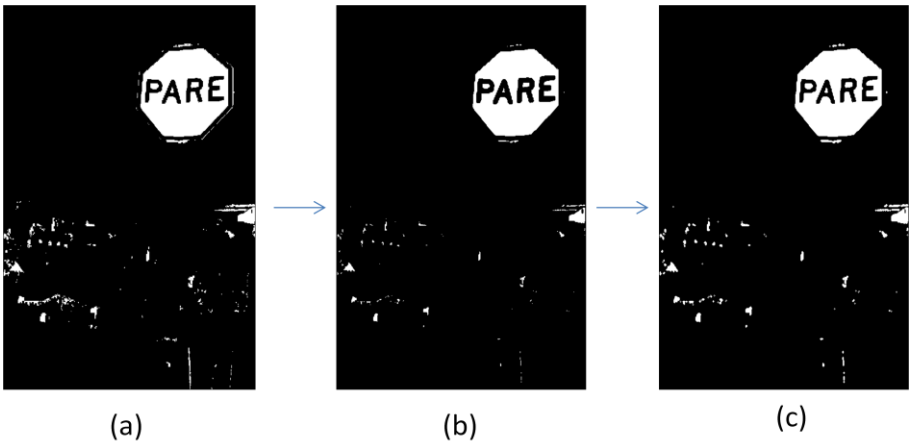


Figura 19: Imagem binária (a), imagem após filtro de erosão (b), e imagem após filtro de dilatação

Após a remoção dos ruídos, são obtidas as coordenadas dos retângulos que envolvem as regiões de interesse, conforme a figura 20.

Figura 20: Imagem com as áreas de interesse encontradas (indicada pelo quadrado verde)

Foram testadas 100 imagens contendo placas de parada obrigatória. Em 9 placas, o algoritmo não conseguiu fazer a detecção, resultado em uma taxa de sucesso de 91%. A figura 21 mostra algumas dessas imagens utilizadas no teste.

Figura 21: Exemplos de algumas imagens utilizadas do teste de detecção

4.2. Algoritmo de reconhecimento

As regiões de interesse da figura 17 foram enviadas para o algoritmo de reconhecimento. As características extraídas dessa região são mostradas na figura 22.

Figura 22: Características extraídas da região de interesse (indicada pelos círculos verdes)

As características acima são então comparadas com aquelas do banco de dados, sendo suas correspondências mostradas na figura 23.

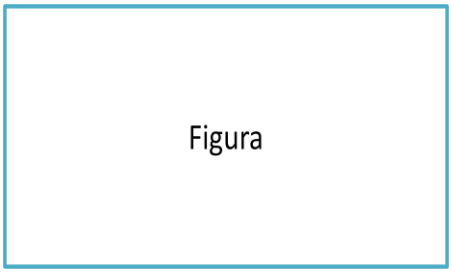


Figura 23: Ilustração de correspondência entre o candidato e a placa no banco de dados

Como foi encontrada mais de 10 correspondências, o candidato pôde ser reconhecido como uma placa de parada obrigatória.

Foram testadas no total 100 imagens contendo parada obrigatória. Essas imagens foram coletadas nas ruas de Belo Horizonte na região da Pampulha. Foi obtida uma taxa de sucesso de reconhecimento de 87%, sendo que em 13 imagens o algoritmo não conseguiu classificar a placa.

4.3. Tempo de execução

Para analisar o tempo de execução do sistema, os tempos de todas as etapas envolvidas nos algoritmos de detecção e reconhecimento foram mensurados, sendo mostrados nas tabelas 1 e 2.

Tabela 1: Tempo de execução nas etapas de detecção

Etapas do algoritmo de detecção	Tempo de execução (ms)
Conversão RGB para HSI	50
Segmentação (Criação da imagem binária)	90
Filtro de erosão	85
Filtro de dilatação	92
Obtenção das coordenadas das áreas de interesse	85
Tempo total	402

Tabela 2: Tempo de execução nas etapas de reconhecimento

Etapas do algoritmo de reconhecimento	Tempo de execução (ms)
Extração de características dos candidatos	300
Comparação de características (com Banco de dados estruturado em árvore)	60
Tempo total com banco estruturado em árvore	360

4.3. Resumo dos resultados de detecção e reconhecimento

A tabela 3 resume os resultados obtidos na detecção e reconhecimento.

Tabela 3: Resultados de confiabilidade da detecção e reconhecimento

Placas testadas	Placas detectadas	Placas reconhecidas
100	91	87
Taxa de sucesso	91%	87%

5. Analise dos resultados

O algoritmo de detecção foi capaz de identificar placas de trânsito corretamente em 91% das imagens testadas. Essa taxa de sucesso está dentro das expectativas do sistema, embora ainda seja necessário aumentar a confiabilidade para tornar viável a sua aplicação em veículos. Embora a taxa de sucesso tenha sido maior que 90%, o algoritmo não se mostrou robusto na detecção de placas com pinturas desgastadas, por exemplo, devido à corrosão, poeira e desgaste com o tempo. A maioria das placas não identificadas pelo algoritmo de detecção foram exatamente placas nas quais a cor já não possui o tom avermelhado. Como a detecção é baseada exclusivamente por cor, placas que perderam suas cores características não são percebidas pelo algoritmo. A figura 24 mostra um exemplo dessa situação.

Figura 24: Placa desgastada com o tempo não foi detectada pelo algoritmo

Apesar da baixa taxa de detecção de placas desgastadas com o tempo, empoeiradas e corroídas, a detecção mostrou uma grande imunidade em relação à iluminação. Ou seja, como mostrado na figura 21, a detecção foi possível em placas muito iluminadas e em placas na sombra.

O algoritmo de reconhecimento também apresentou uma taxa de sucesso dentro da faixa esperada, onde foi possível reconhecer corretamente em 87% das imagens testadas. Por se tratar de uma aplicação crítica, o reconhecimento precisa ser ainda maior para tornar o sistema prático.

Os tempos de processamento dos algoritmos em si foram de fato o grande gargalo da viabilidade do projeto. Os dados das tabelas 1 e 2 revelam a incapacidade do sistema em executar em tempo real. Muito tempo está sendo consumido para detectar e reconhecer as placas, sendo necessário pouco menos de 1 segundo para processar completamente uma imagem. Este tempo é demasiadamente longo e impossibilita a detecção em tempo real.

Como a duração do processamento esta atrelada com a resolução da imagem, optou-se em utilizar uma resolução de 480x640 para capturar imagens da câmera. Nessas dimensões, os tempos obtidos são aqueles da tabelas 1 e 2 . Quando menor a resolução, menor o tempo de processamento. Por outro lado, isto implica que o dispositivo de detecção, por exemplo, o smatphone, deve ficar mais próximo da placa. Nos testes realizados, a distância média entre o smartphone e a placa para ser possível a detecção foi de

aproximadamente 1 metro. Esta é outra limitação do sistema desenvolvido. Para detectar placas em uma distância maior, é preciso aumentar a resolução, tendo como efeito o aumento do tempo de processamento.

6. Conclusão

Esse projeto de conclusão de curso abordou a implementação completa de um sistema de detecção e reconhecimento de placas de trânsito na plataforma Android. A detecção é feita baseada em segmentação de cores para extração de regiões vermelhas, já que as placa de trânsito de regulamentação possuem bordas dessa cor. Após a segmentação, eliminam-se os ruídos e são determinadas as regiões de interesse. O algoritmo de reconhecimento recebe essas regiões, extrai suas características, e busca similaridades entre elas e o banco de dados formado por características próprias de placas de trânsito. Dependendo da quantidade de similaridade, o algoritmo é capaz fazer o reconhecimento.

As taxas de sucesso de detecção e reconhecimento foram de 91% e 87% respectivamente, o que está dentro do esperado, mas ainda aquém do desejado. As imagens de testes utilizadas no algoritmo foram obtidas de situações reais do dia a dia nas ruas de Belo Horizonte em condições de tempo ensolarado. Ainda é preciso testar o sistema em outras condições, como à noite, com chuva, neblina, e dentre outras, para validar o seu desempenho. Um bom sistema TSR precisa ser capaz de lidar com as mais variadas situações do ambiente.

Os resultados mostram que ambos os algoritmos possuem suas virtudes e pontos fracos. O algoritmo de detecção, por exemplo, não é capaz de detectar placas que não possuem o tom avermelhado devido corrosão, poeira e desgaste com o tempo. Por outro lado, se mostrou confiável na detecção de placas com diferentes intensidades de iluminação. Já o algoritmo de reconhecimento, apresentou um elevado tempo de processamento, comprometendo a execução em tempo real do sistema. No entanto, apresentou robustez no reconhecimento de placas inclinadas e parcialmente ocultadas.

O tempo de processamento dos algoritmos também deixou a desejar, impossibilitando o seu funcionamento em tempo real. Algumas melhorias podem ser feitas para melhorar consideravelmente esses tempos. Por exemplo, existem bibliotecas específicas de visão computacional, como a OpenCV, que são otimizadas para processamento de imagem. Como toda implementação dos algoritmos foram feitas nativamente sem a preocupação com otimização, é provável que a substituição de funções no código atual para funções da OpenCV produza melhores resultados. Adicionalmente, como os processadores atuais tendem a possuir vários núcleos, a utilização de threads pode melhorar significativamente o tempo de execução. A ideia é paralelizar os algoritmos de modo a se beneficiar dos vários núcleos dos processadores.

Embora ainda não seja viável a sua aplicação em veículos, esse trabalho mostrou uma metodologia para detecção e reconhecimento de placas de trânsito. Muitas melhorias ainda precisam ser realizadas até atingir os critérios desejados de confiabilidade e velocidade de processamento.

7. Referências

[1] Yan, H.; Virupakshappa, K.; Pinto, E. V. S.; Oruklu, E. Hardware/Software Co-Design of a Traffic Sign Recognition System Using Zynq FPGAs. Electronics (Basel), 01 December 2015, Vol.4(4), pp.1062-1089.

[2] Al Smadi, T. Real-Time Lane Detection for Driver Assistance System. Circuits and Systems, 2014, 5, 201-207.

[3] Kaur, R.; Singh, B. Design and implementation of car parking system on FPGA. International Journal of VLSI design & Communication Systems (VLSICS) Vol.4, No.3, June 2013

[4] Waite, S.; Oruklu, E. FPGA-Based Traffic Sign Recognition for Advanced Driver Assistance Systems. Transp. Technol.2012, 3, 1–16.

[5] Liang, M.; Yuan, M.; Hu, X.; Li, J.; Liu, H. Traffic sign detection by ROI extraction and histogram features-based recognition. In Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, USA, 4–9 August 2013; pp. 1–8.

[6] Wang, G.; Ren, G.; Wu, Z.; Zhao, Y.; Jiang, L. A robust, coarse-to-fine traffic sign detection method. In Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, USA, 4–9 August 2013; pp. 1–5.

[7] Lowe, D.G. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Corfu, Greece, 30 September–2 October 1999; Volume 2, pp. 1150–1157.

[8] Bay, H.; Tuytelaars, T.; Gool, L.V. SURF: Speeded up robust features. Comput. Vis. Image Underst. 2008, 110, 346–359.

[9] Ren, F.; Huang, J.; Jiang, R.; Klette, R. General traffic sign recognition by feature matching. Image and Vision Computing New Zealand, 2009. IVCNZ'09. In Proceedings of the 24th International Conference, Willington, New Zealand, 23–25 November 2009; pp. 409–414.

[10] Chen, Z.; Huang, X.; Ni, Z.; He, H. A GPU-based real-time traffic sign detection and recognition system. In Proceedings of the 2014 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS), Orlando, FL, USA, 9–12 December 2014; pp. 1–5.

[11] Schwiegelshohn, F.; Gierke, L.; Hubner, M. FPGA based traffic sign detection for automotive camera systems. In Proceedings of the 2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Bremen, Germany, 29 June–1 July 2015; pp. 1–6.

[12] Gómez-Moreno, H.; Maldonado-Bascón, S.; Gil-Jiménez, P.; Lafuente-Arroyo, S. Goal Evaluation of Segmentation Algorithms for Traffic Sign Recognition. IEEE Trans. Intell. Transp. Syst. 2010, 11, 917–930.

Imagens retiradas da internet

-> HSI e RGB

https://gerardnico.com/wiki/data/type/color/hsv

http://learn.colorotate.org/color-models/#.WgOOymiPKUk

-> Erosao e dilatacao

http://slideplayer.com/slide/7444081/