SERVIÇO PÚBLICO FEDERAL
UNIVERSIDADE FEDERAL DE SANTA CATARINA
***CAMPUS* BLUMENAU**
Rua João Pessoa, 2750, Velha, Blumenau – SC.
www.blumenau.ufsc.br/blumenau@contato.ufsc.br

VISÃO COMPUTACIONAL

Relatório A2

Gabriel Alves Silvestre

Ronaldo William Baggio de Oliveira

Blumenau

Abril, 2018

1 INTRODUÇÃO

O presente trabalho trata da exposição dos aspectos técnicos do segundo trabalho apresentado para a disciplina de Visão Computacional, neste documento serão expostos conceitos e teorias trabalho que tem como tema o reconhecimento de placas de veículos.

O objetivo deste trabalho é elaborar uma função capaz de reconhecer e retornar os caracteres alfanuméricos a partir de imagens com placas de veículos. Esta função deve utilizar features de região para detectar cada caractere da placa, em seguida através do template matching comparar o caractere detectado com os templates respectivos a letra e número.

Esta função é implementada e construída por meio da biblioteca de visão computacional Machine Vision Toolbox (MVT) do software de cálculo numérico Matlab. Tal biblioteca fornece diversas ferramentas de visão computacional e controle baseado em visão, além de um completo guia para referência.

O trabalho está segmentado em quatro tópicos através do quais serão apresentadas uma breve contextualização, em seguida uma abordagem das metodologias utilizas bem como a lógica de funcionamento, por fim são apresentados os resultados e fechamento é realizado com a conclusão.

2 CONTEXTUALIZAÇÃO

Assim como qualquer cidadão deve possuir carteira de identidade e cadastro de pessoa física, os automóveis também devem possuir um código exclusivo respectivamente associado para auxiliar na identificação e tornar cada veículo único. Este código, popularmente chamado de placa, é uma combinação alfanumérica de três letras (alfabeto latino) e quatro números (números naturais no intervalo entre 0 e 9).

Identificar e reconhecer placas de veículos automaticamente através de imagens é útil em diversas situações como controle de estacionamento, lombadas eletrônicas, radares eletrônicos, inspeções veiculares e verificações de segurança.

Para utilizar técnicas de visão computacional não são necessárias alterações nos sistemas de monitoramento, na maior parte das vezes as câmeras dos sistemas de monitoramento são suficientes.

Em razão do avanço tecnológico no campo da visão computacional é possível tornar os sistemas de identificação mais eficientes, ou seja, obter resultados mais próximos a realidade de forma mais rápida mesmo com a presença de interferências como variação de brilho, redução de nitidez, variação do ângulo entre outros.

3 METODOLOGIAS UTILIZADAS

O primeiro passo na definição das metodologias utilizadas é avaliar os requisitos do trabalho, isto é, determinar e definir o que é necessário para atingir os objetivos com êxito. Ao analisar uma série de imagens de placas de automóveis ficam claros os seguintes requisitos:

Tabela I – Requisitos do sistema.

Nº	Requisito	Descrição
I	Identificar placa no ambiente	Identificar a placa do automóvel em relação ao ambiente como um todo, distinguir a placa dos demais objetos presentes na imagem.
II	Corrigir perspectiva	Corrigir de perspectiva, já que imagens diferentes podem trazer placas com posicionamentos diferentes.
III	Modelo dos caracteres	Criar um conjunto de modelos (templates) com os quais os caracteres da placa serão comparados.
IV	Calcular as dimensões da placa	Calcular as dimensões da placa e avaliar a qual tipo de veículo pertence a placa (carro/caminhão ou motos).
V	Detectar caracteres na placa	Detectar cada um dos caracteres presentes na placa, separar cada caractere dos demais (isolar).
VI	Identificar caractere	Calcular quanto o caractere encontrado é próximo (semelhante) de cada um dos modelos.

3.2 Corrigir Perspectiva

Para satisfazer o segundo requisito II é fundamental aplicar o conceito de Homografia, que é uma transformação capaz de mapear pontos de um plano para outro plano, sendo que tais planos estão em orientações diferentes.

Esta transformação da homografia relaciona a mudança entre dois planos, desta forma é possível relacionar imagens com posicionamentos diferentes a um mesmo referencial. Com a Homografia é possível definir os modelos como referência, e transformar as imagens das placas.

3.3 Modelo dos Caracteres

Para verificar que determinado caractere da placa corresponde a um determinado caractere alfanumérico, ou seja, para reconhecer o caractere é necessário antes criar modelos com os quais os caracteres serão comparados.

Assim para o primeiro passo para elaborar um modelo é verificar a fonte dos caracteres, isto é, o padrão com o qual são escritos as letras e o números. No Brasil as resoluções do CONTRAN, órgão nacional regulamentador do trânsito, estabelecem a fonte Mandatory como padrão.

Após uma pesquisa foram encontradas as fontes do padrão Mandatory:

Fig.1 – Letras do padrão Mandatory.

Fig.2 – Números do padrão Mandatory.

Através das figuras acima foram criados os modelos para cada respectivo caractere, todos modelo foram salvos no formato JPG (Joint Photografic Expert Group).

3.4 Calcular dimensão da placa

Dentre o padrão de placas brasileiro existem dois tipos principais de placas: as placas de motos e as placas dos demais veículos. As placas das motos tem 17 cm de altura e 20 cm de comprimento, enquanto as placas do demais veículos tem 13 cm de altura e 40 cm de comprimento.

Tendo conhecimento disto, é possível definir duas relações:

- $Placas\ de\ Motos$: 1.1765
- $Placas\ Demais\ Veículos$: 3.0769

Então para distinguir entre os dois tipos de placa basta verificar a relação da placa a ser identificada e ver qual o valor mais próximo.

3.5 Detectar caracteres na placa

Para detectar cada um dos caracteres presentes na placa é necessário utilizar o conceito de features de região, através do qual é possível encontrar tamanho, formas e até mesmo orientação de regiões presentes em imagens

As regiões são separadas por meio das características comuns, estas regiões normalmente são chamadas de Blobs, dentre a característica comum mais utilizada para classificar regiões está a cor. Uma região que contém um grupo de pixeis da mesma cor associados (unidos) contêm a mesma etiqueta, tal etiqueta é responsável por identificar a região, a cada região distinta é associada uma etiqueta diferente.

Para descobrir os vários blobs de uma imagem basta fazer uma análise de conectividade, que é capaz de descrever quantas regiões distintas existem dentro de uma imagem. Ainda assim, é importante ressaltar que um blob não está necessariamente associado um objeto do mundo real, um blob pode ou não representar um objeto.

Depois de detectar cada um dos blobs presentes na imagem é possível extrair informações dos mesmos, dentre as propriedades mais comuns estão área, perímetro, centro de massa, forma, posição e orientação dentre outras.

Para detectar caracteres as propriedades mais relevantes são posição e área, isto porque através da definição de um valor limite é possível encontrados blobs com áreas relevantes, ou seja, blobs com áreas muito pequenas ou muito grandes são descartados e apenas blobs que representam os caracteres da placa serão considerados.

Além disso, através da posição é factível descobrir as coordenadas dos quadrados que contem cada um dos blobs, e assim é possível selecionar cada blob separadamente, em outras palavras é possível selecionar (detectar) cada caractere da placa separadamente. Também com posição de cada blob é obtida a organização (sequência) correta das letras e números, isto é, a função torna-se capaz de colocar as letras e números na mesma ordem da placa.

3.6 Identificar caracteres na placa

Após detectar cada um dos números e letras da placa o passo seguinte é identificar que caractere é este, para isto é preciso calcular o quanto o caractere encontrado é próximo (semelhante) de cada um dos modelos elaborados anteriormente.

Existem diversas maneiras de calcular a semelhança entre duas imagens (template matching), dentre estas maneiras uma das mais eficientes são as funções de similaridade, equações que calculam o erro entre o pixel da posição (u,v) de uma imagem com o pixel (u,v) de outra imagem.

Entre as funções de similaridade existentes será utilizada a função ZNCC (Zero Mean Normalized Cross-Correlation), que retorna valores no intervalo -1 a 1, onde valores menores que zero indicam uma relação negativa (-1 indica que uma imagem é o inverso da outra) e valores maiores que zero indicam alguma semelhança, quanto maior a semelhança mais próximo de 1 (onde 1 significa que as imagens são idênticas).

3 LÓGICA DE FUNCIONAMENTO

Por meio das funções do Machine Vision Toolbox do MATLAB e da tabela de requisitos do sistema é foi elaborada a função de identificação de placas de automóveis, tal função tem o funcionamento segundo as referidas etapas:

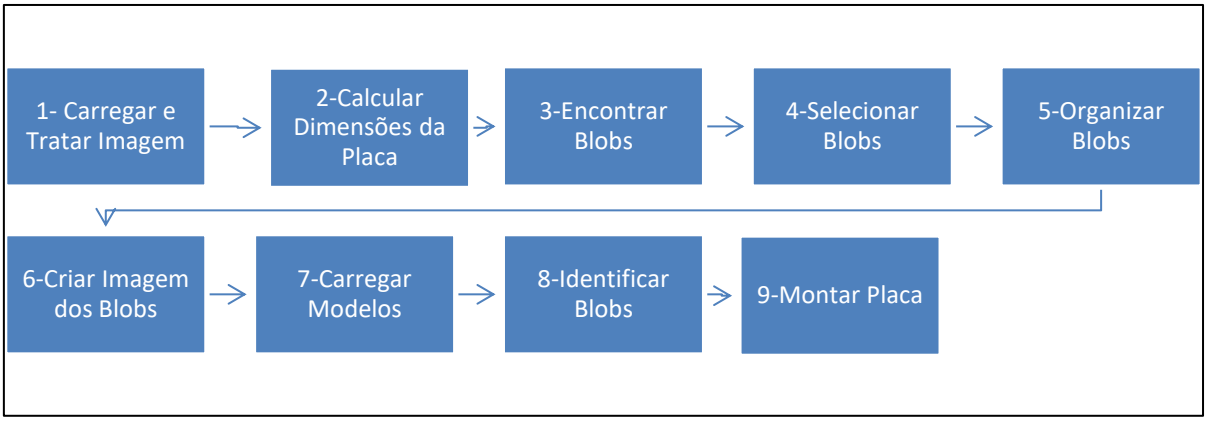

Fig.3 – Fluxograma do funcionamento da função de identificação de placas.

Antes de iniciar o tratamento de imagem é essencial carregar a imagem que contem a placa a ser identificada, após carregar a imagem é convertida para o espaço de cor escala de cinza e invertida:

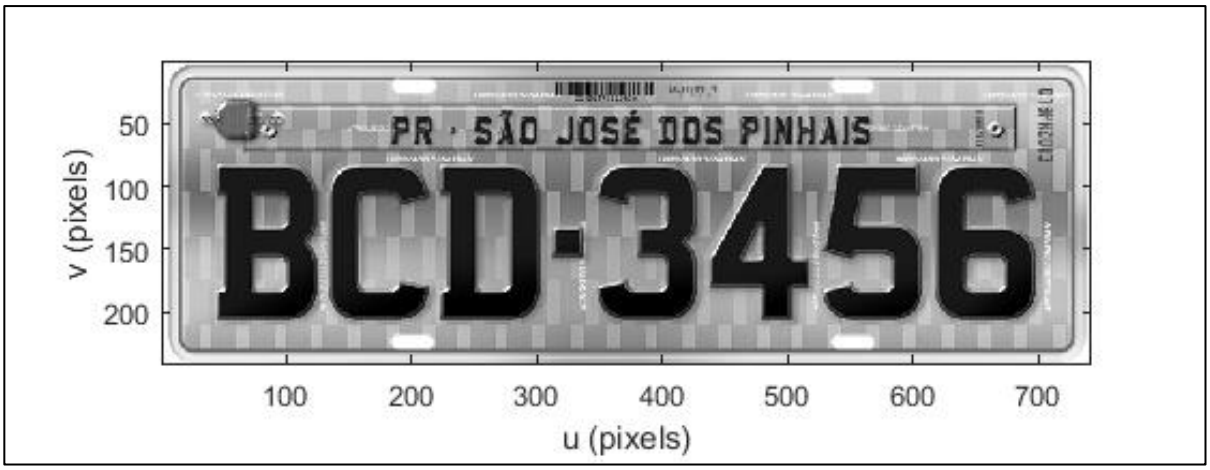

Fig.4 – Imagem Placa 1.

Fica evidente na imagem acima que os caracteres estão próximos de tons pretos, porém pixeis da cor preto tem valor zero na escala cinza, o que tornaria inviável calcular a área através das features de região. Para resolver isto basta criar uma máscara e definir um valor limiar (treshold) no qual todos os pixeis da cor preta ou próximos a esta cor recebem valor um, valor este que representa a cor branca, enquanto os demais pixeis (distantes da cor preta) recebem valor zero.

$$Treshold = 0.3$$

Fig.5 – Imagem Placa 1 após aplicação da máscara – resultado obtido através das funções da biblioteca Vision do Matlab.

Depois de tratar e corrigir a imagem a etapa seguinte é encontrar as regiões comuns com mesma cor, ou seja, encontrar os diferentes blobs que irão representar as letras e números da placa.

Encontrar os blobs é uma tarefa simples no MATLAB pois existe uma função chamada *iblob*, que retorna um vetor de objetos do tipo feature de região onde estão representadas cada uma das regiões conectadas na imagem. Este objeto contém várias informações acerca dos blobs como área (número de pixels), etiqueta (número de identificação da região), centroide (vertical e horizontal), os limites máximo e mínimo da caixa que contém a região (limite vertical e horizontal) dentre outros.

Ao aplicar a função *iblob* na imagem da figura 4 são encontrados 293 blobs diferentes, um número elevado visto que o objetivo é detectar apenas os blobs referentes a placa, porém ao examinar a área dos blobs conforme tabela II:

Tabela II – Blobs Encontrados.

Área do Blob	Frequência	Frequência (%)
0	145	49,49
1	70	23,89
1 < área ≤ 200	70	23,89
200 < área ≤ 1000	1	0,34
1000 < área ≤ 10000	7	2,39

Ao observar os dados da tabela II:

- Aproximadamente metade dos blobs tem área igual a zero, estes são os blobs referentes as regiões pretas da imagem, que pertencem em sua maioria ao fundo da placa.
- Cerca de 24% dos blobs tem área igual a 1, este são blobs referentes aos ruídos presentes na imagem.
- Setenta blobs tem área no intervalo 1 a 200, estes blobs representam as manchas brancas maiores (como as localizadas acima da letra B – figura 4), as barras do código de barras (localizadas acima do nome da cidade) e os caracteres referentes ao nome da cidade e do estado.

- Um blob tem área no intervalo 200 a 1000, avaliando a figura 4 é perceptível que este blob é referente ao hífen utilizado para separar as letras e números da placa.
- Sete blobs tem área no intervalo 1000 a 7000, estes são os blobs que representam as letras e números de identificação da placa.

Neste caso os sete blobs com maiores áreas representam os caracteres da placa, porém isso pode não pode ser generalizado já em alguns casos a máscara criada não fornece um resultado tão bom quanto o da imagem 4. A fim de tornar o algoritmo mais robusto é interessante definir verificações para verificar quando um blob realmente representa um caractere.

Sendo assim foram definidas três etapas de verificação:

- Área do Blob: como mencionado anteriormente blobs com área zero ou com áreas pequenas (intervalo 0 a 1) e média (intervalo 1 a 1000) são referentes a ruídos e manchas que não representam caracteres, então os blobs com área inferior a 1000 serão descartados.
- Localização: é calculada a média do limite vertical das caixas que contém os maiores blobs, após este cálculo o valor do limite vertical de cada blob é comparado com a média e caso a diferença seja superior a vinte o blob é descartado.
- Altura: é calculada a média da altura das caixas que contém os blobs e após este cálculo a altura de cada blob é comparada com a média, caso a altura do blob não esteja dentro do limite inferior de 80% da altura média e do limite superior a 120% da altura média o blob é descartado.

Depois verificar e selecionar apenas os blobs correspondentes as letras e números da placa a próxima etapa é organizar os blobs de forma que estejam coerentes com a sequência na qual aparecem na placa.

Para tanto basta utilizar limite horizontal mínimo de cada blob e comparar todos os blobs, o blob que tem o menor limite horizontal mínimo é referente ao primeiro caractere da placa (por exemplo, na figura 4 a letra B), o blob que tem o segundo menor limite horizontal mínimo é o segundo caractere (por exemplo, na figura 4 a letra C) e assim sucessivamente até o último caractere.

Logo após selecionar e ordenar blobs pode-se obter a imagem referente a cada um dos blobs por intermédio dos valores dos limites horizontais e verticais máximos e mínimos, ou seja, a partir da imagem com a máscara (figura 4) serão recortadas imagens de acordo com os valores dos limites.

A função ZNCC implementada no Matlab requer que as imagens comparadas possuam o mesmo tamanho, em razão disto é indispensável aplicar a função *isamesize* que é capaz de alterar o tamanho de uma imagem. Finalmente com as imagens do mesmo tamanho é possível aplicar a função ZNCC e comparar cada caractere com todos os modelos construídos, o modelo que retornar o valor mais próximo de 1 será definido como valor do caractere:

3 TESTES E RESULTADOS

Após organizar e adequar requisitos e a lógica de funcionamento de todas as operações é importante submeter a função de identificação a testes com imagens de placas em diferentes situações.

3.1 Teste 1 – Número e Letra da placa isolada

O primeiro teste para testar a eficiência da função é com uma imagem que contém apenas a placa a ser identifica:

Fig.6 – Imagem Placa 2.

Após carregar a imagem e aplicar a máscara é obtido o seguinte resultado:

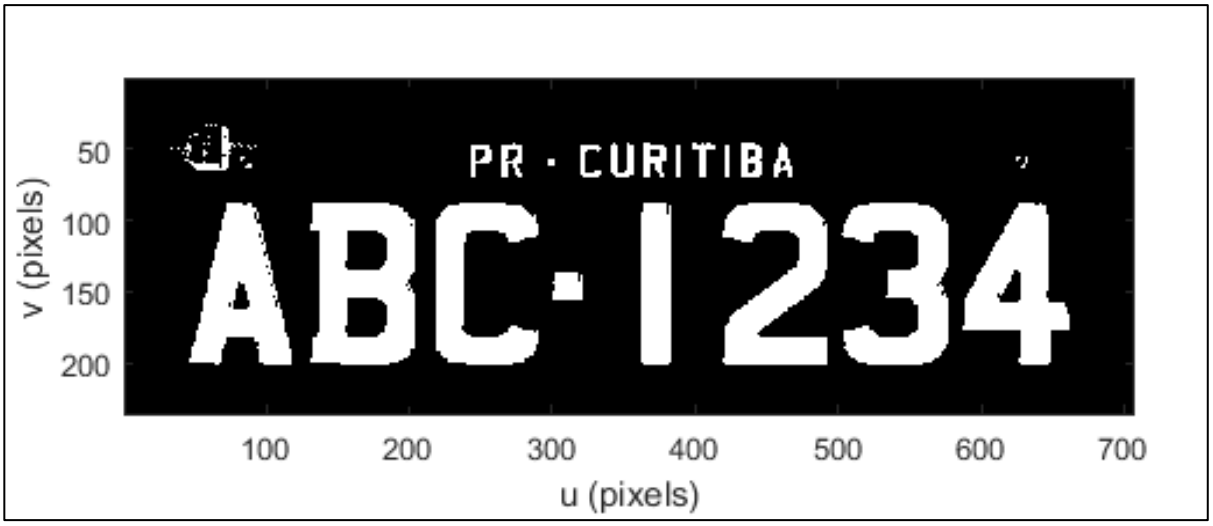

Fig.7 – Imagem Placa 2 após aplicação da máscara – resultado obtido através das funções da biblioteca Vision do Matlab.

Ao analisar a figura 7 é notável a presença de pontos e manchas brancas em diversos locais da placa, tais corpos e pontos estão presentes devido a valor limiar (treshold) definido para aplicar a máscara. Foram realizados vários testes para tentar encontrar valores mais adequados para o treshold, ao reduzir este valor o número de manchas brancas é reduzido, porém ocorre o aparecimento de pontos brancos no interior dos caracteres. Sendo assim, o valor de treshold não sofreu alteração (mesmo valor definido no tópico anterior).

Após transformar a imagem é o momento de aplicar a função *iblob*, novamente a função foi capaz de encontrar um número elevado de blobs diferentes, visto que o objetivo é detectar apenas os sete blobs referentes as letras e números da placa, porém ao examinar as áreas dos blobs:

Tabela III – Blobs Encontrados Placa 2.

Área do Blob	Frequência	Frequência (%)
0	46	37,70
1	37	30,33
1 < área ≤ 200	26	21,31
200 < área ≤ 1000	6	4,92
1000 < área ≤ 10000	7	5,74

Como descrito no tópico anterior foram implementadas verificações para considerar apenas áreas válidas, apenas áreas que realmente representam blobs referentes as letras e números. Após selecionar apenas os blobs válidos a função organiza os blobs de acordo com a posição do limite vertical mínimo da caixa que contém o blob.

Então com os blobs selecionados e organizados, basta utilizar os limites máximos e mínimos verticais e horizontais para construir novas imagens a partir da imagem transformada (figura 7), em outras palavras os limites serão utilizados para recortar imagens dos caracteres.

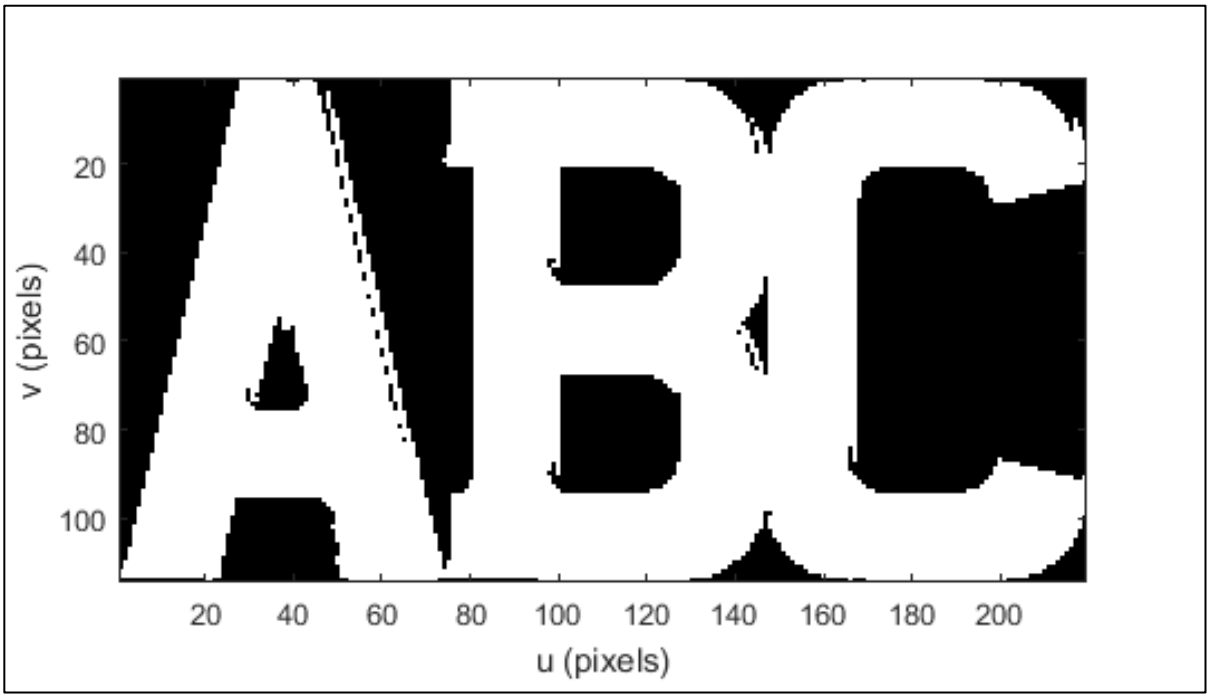

Fig.8 – Blobs referentes as letras da Placa 2.

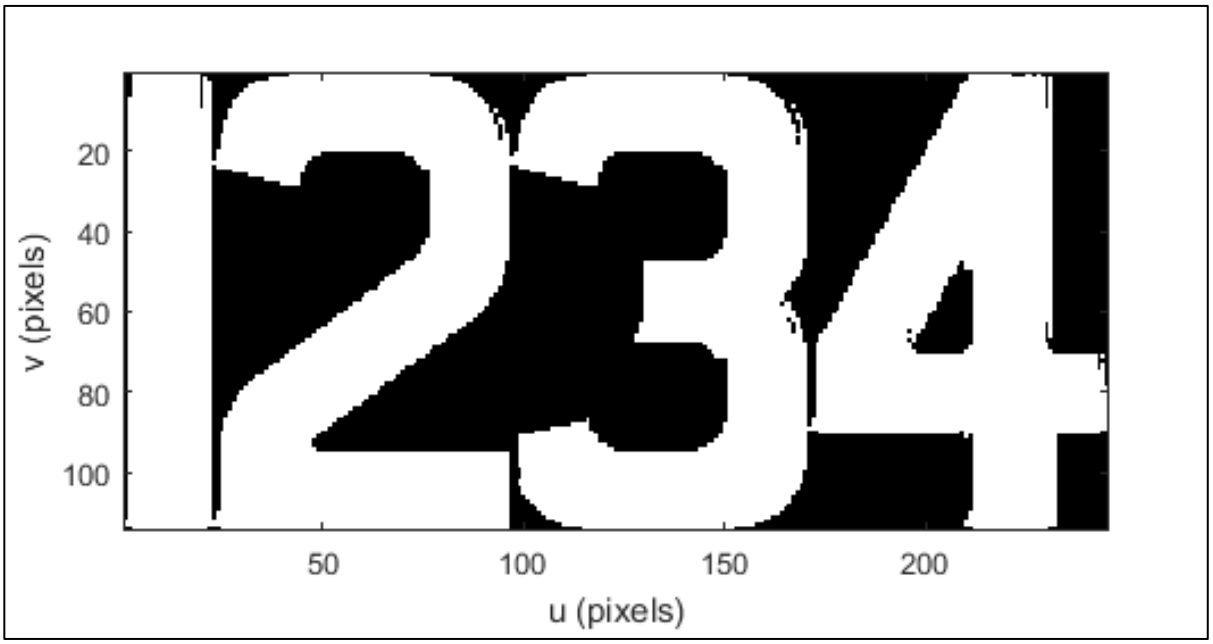

Fig.9 – Blobs referentes aos números da Placa 2.

Em sequência é necessário identificar qual caractere determinado blob representa, em outras palavras calcular o quanto o caractere representado pelo blob é próximo (semelhante) de cada um dos modelos alfanumérico. Como dito anteriormente o método utilizado será a comparação por meio da função de similaridade ZNCC, que retorna valores no intervalo -1 a 1 sendo que quanto maior a semelhança (quanto mais parecidas as imagens) mais próximo de 1 será o valor retornado.

Tabela IV – Valores ZNCC Placa 2.

Caractere Placa	Maior Valor ZNCC (Caractere Correspondente)	Segundo Maior Valor ZNCC (Caractere Correspondente)	Relação Maior Valor/Segundo Maior Valor
A	0.7178 (A)	0.2976 (c)	41,46%
B	0.8468 (B)	0.7414 (E)	87,55%
C	0.9390 (C)	0.6948 (G)	73,99%
1	0.1096 (1)	0.0846 (4)	77,19%
2	0.6601 (2)	0.4624 (3)	74,05%
3	0.7199 (3)	0.6395 (9)	88,83%
4	0.8297 (4)	0.2485 (1)	29,95%

Na tabela III estão representados os valores obtidos para a função de similaridade ZNCC, na primeira coluna estão contidos os caracteres reais da placa da figura XX enquanto na segunda coluna estão os maiores valores do ZNCC e os respectivos caracteres, em outras palavras a primeira coluna apresenta o caractere da placa enquanto a segunda apresenta a caractere estimado pelo algoritmo.

Neste caso tanto as letras quanto os números foram estimados de maneira correta, isto é, o caractere encontrado pela função é correspondente ao caractere contido na placa, o que

converge com o objetivo deste trabalho, que era de elaborar uma função para identificar caracteres das placas de veículos.

Ainda que o objetivo tenho sido alcançado existem ressalvas, o primeiro ponto pode ser analisado através da terceira e quarta colunas da tabela III, a terceira coluna ("Segundo Maior Valor ZNCC (Caractere Correspondente)") revela o segundo maior valor obtido com o ZNCC, isto é, o segundo caractere mais próximo do caractere real. Em um sistema eficiente, quanto maior a diferença do maior valor para o segundo maior valor melhor será a identificação, ou seja, menor a chance de um caractere ser identificado de maneira errônea.

A quarta coluna da tabela III apresenta a relação entre o maior e o segundo maior valor do ZNCC, em cinco dos sete caracteres a relação é próxima ou superior a 70% o que demonstra que em cinco das sete vezes a função esteve próxima de identificar um caractere de forma incorreta.

O segundo ponto a ser destacado no que diz respeito a eficiência da função é forma de aplicação da máscara na imagem da placa, isto porque conforme a variação do valor limiar (treshold) a função *iblob* encontra blobs diferentes com áreas diferentes. Também ao variar o valor do treshold podem ser reduzido ou elevado o número de corpos brancos na imagem transformada, o que tem impacto direto no resultado da identificação e pode reduzir a eficiência da função ZNCC.

Para contornar o problema no valor limiar pode ser aplicada uma função de cálculo automático de treshold, no Matlab a função *otsu* é capaz de realizar isto, como parâmetro basta passar a imagem a ser transformada e a função retorna um valor. Para o caso da imagem 9 o valor de treshold calculado pela função *otsu* é de 0.3895, valor aproximadamente 30% maior que o valor anterior que era de 0.3.

Ao realizar o teste com este novo valor (0.3895) a função de identificação deparou-se com alguns problemas, já que ocorreu o aumento no número de corpos brancos. Para otimizar a utilização da função *otsu* definiu-se que o valor de treshold é 80% do valor retornado pela função *otsu*. Assim, os resultados obtidos com a função *otsu* e com o valor configurado manualmente (0.3) foram muito próximos.

3.2 3.1 Teste 2 – Identificação da Cidade e Estado da placa isolada

O segundo teste para testar a eficiência da função, ainda na imagem que contém apenas a placa (figura 6), é identificar os caracteres correspondentes a cidade e ao estado.

Este é um caso mais extremo, já que os blobs que representam os caracteres da cidade e do estado tem área menor, ainda assim procedimento para identificar tais caracteres é muito semelhante mesmo que para identificar as letras e números da placa. A primeira diferença é que

neste caso, o limite vertical mínimo do blob que representa a primeira letra é utilizado para recorta a imagem original e criar uma nova imagem:

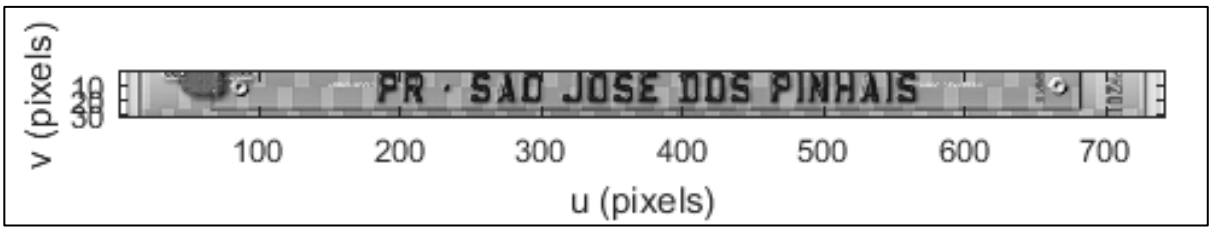

Fig.10 – Corte da imagem da Placa 2.

O objetivo ao recortar a imagem é reduzir possíveis interferências, assim novamente, aplicando uma máscara (com o mesmo treshold da máscara anterior):

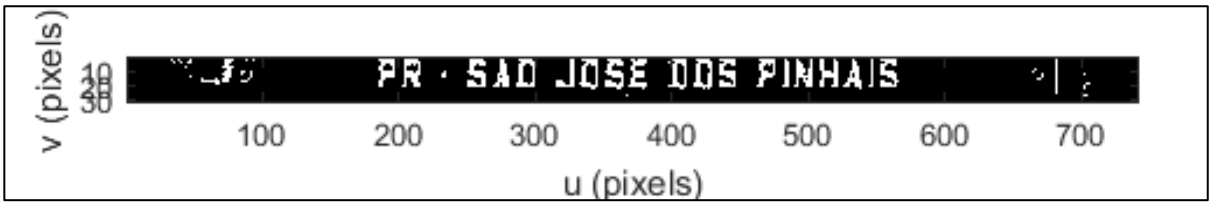

Fig.11– Máscara no corte da imagem da Placa 2 - resultado obtido com as funções do Machine Vision Toolbox.

Após organizar e aplicar as mesmas etapas de verificação que o procedimento anterior, surge a segunda diferença para o procedimento anterior (teste 1), a existência de uma etapa adicional de verificação, tal etapa consiste em calcular a distância do blob anterior e do blob posterior, caso esta distância seja superior a trinta o blob é rejeitado. Duas exceções a esta etapa são o primeiro e o último blob, já que o primeiro blob não possui blob anterior e o último blob não possui blob posterior.

Com o número correto de blobs, dois para o estado e o restante para a cidade:

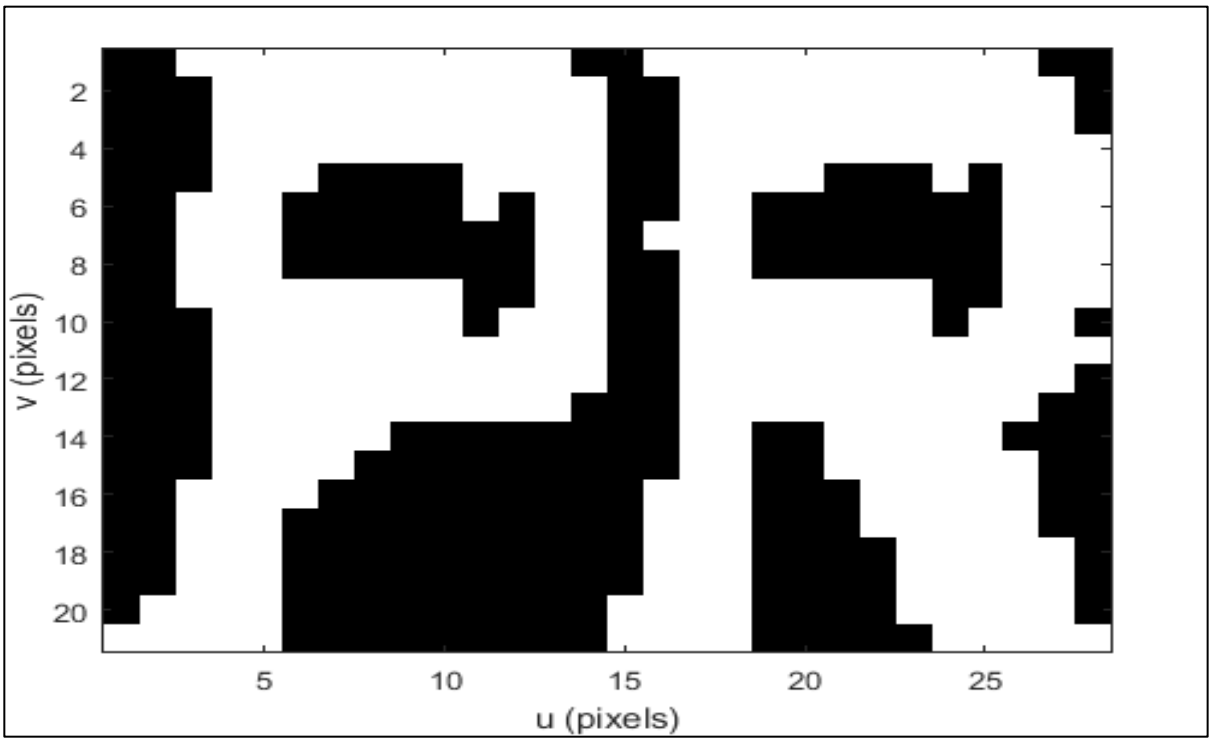

Fig.12 – Blobs referentes a sigla do estado da Placa 2.

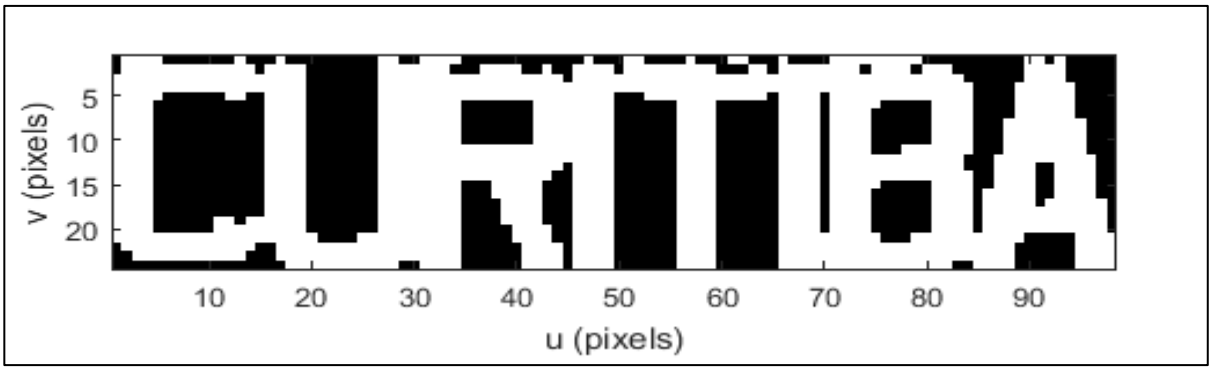

Fig.13 – Blobs referentes ao nome da cidade da Placa 2.

O próximo passo é aplicar a função de similaridade ZNCC, assim o seguinte resultado é obtido:

“PR-CURITIBA”

Ao examinar a placa é constatada a eficiência da função para identificar os caracteres da cidade e do estado. Porém como no teste anterior deve ser avaliada a eficiência da função, neste caso o ponto crítico são as verificações para selecionar os blobs válidos, como os blobs referentes a cidade e ao estado tem área menor estes podem ser interpretados como manchas ou pontos brancos, e consequentemente serão rejeitados.

Em casos extremos em razão do valor do treshold podem aparecer corpos brancos de tamanho próximo ao tamanho dos caracteres, e pior ainda, estes corpos podem também podem ter posição próxima aos corpos brancos. Esta combinação de fatores pode fazer as etapas de verificação inúteis, visto que estas levam em consideração tamanho e localização, e blobs errados sejam considerados caracteres da placa

3.3 Algoritmo desenvolvido

O algoritmo desenvolvido bem como o conjunto de imagens do teste e uma breve explicação podem ser encontrados no GitHub através do link:

- https://github.com/krausloko/TrabalhoA2

7 CONCLUSÃO

Inúmeros aplicações buscam na tecnologia uma forma de melhorar produtos e processos, isto porque é possível através dela é possível automatizar tornar atividades repetitivas e cansativas o que acaba por otimizar tempo e outros diversos recursos. Dentre as alternativas tecnológicas os algoritmos de visão computacional devem ganhar cada vez mais espaço, especialmente o campo de detecção e reconhecimento de caracteres tema deste trabalho.

Após levantar os requisitos, buscar na literatura o embasamento teórico, elaborar uma sequência lógica, implementar a função e realizar os devidos testes é interessante fazer uma avaliação para listar pontos relevantes.

A primeira avaliação é em relação a competência da função para reconhecer os caracteres, visto que o objetivo de trabalho é desenvolver uma função que a partir da imagem de uma placa fosse capaz de reconhecer e descrever os caracteres alfanuméricos inscritos na placa, utilizando para tanto técnicas de template matching e features de região.

Dentro do conjunto de imagens de placas testadas a função é capaz de reconhecer letras e números das placas de automóveis, bem como a sigla do estado e o nome da cidade, sendo assim em um âmbito geral a função é capaz de cumprir o objetivo. Apesar disso, devido a um conjunto de fatores limitadores a função pode apresentar resultados indesejados com a utilização de imagens muito diferentes da placa usada nos testes.

O primeiro ponto do conjunto de limitadores é a configuração com a qual a máscara é aplicada na imagem da placa, como descrito ao longo do trabalho alguns valores de treshold podem resultar no surgimento de corpos brancos (manchas), e estes podem afetar a função no momento de selecionar quais blobs são realmente correspondentes aos caracteres. Uma das alternativas apresentadas é a função automática para calcular treshold, porém em alguns casos nem mesmo esta função é capaz de auxiliar. Além disso, ajustar função *iblob* para considerar apenas blobs com valor 1, apenas partes brancas seriam computadas no cálculo da área), poderia trazer alguma melhoria.

Outro fator limitador é variação no tamanho em decorrência da função *isamesize*, isto porque esta função altera o tamanho da imagem através do corte e dimensionamento. Em razão destas operações (corte e dimensionamento) a imagem perde parte da resolução o que acaba por influenciar o template matching.

O terceiro fator limitador é o próprio template matching, além dos dois fatores limitadores anteriores afetarem diretamente o template matching existem outros pontos críticos. A construção dos modelos é um destes pontos, já que ao definir o tamanho e formato diferentes os resultados serão alterados, a proximidade entre os valores calculados pelo ZNCC também é importante, pois a função pode interpretar um caractere como outro.

Por fim, vale destacar a especificidade da função, a mesma está limitada a identificação de placas do padrão brasileiro, mais do que isso, a função conta com alguns parâmetros ajustadas manualmente o que pode afeta diretamente a capacidade identificar placas em condições muito divergentes das placas testadas.

REFERÊNCIAS

E-DISCIPLINAS USP. Visão computacional detecção de cores e “blobs ”. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/4112486/mod_resource/content/0/v14%20-%20detccao%20de%20cores%20e%20blobs.pdf>. Acesso em: 14 mai. 2018.

PETER CORKE. Machine vision toolbox. Disponível em: <http://petercorke.com/wordpress/toolboxes/machine-vision-toolbox>. Acesso em: 09 abr. 2018.