UNIP LIMEIRA – UNIVERSIDADE PAULISTA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CAIO YAGO VILELA F00JED7

RAFAEL H. DOS SANTOS LEMES F054802

DETECÇÃO E RECONHECIMENTO DE OBJETOS EM IMAGENS

LIMEIRA – SÃO PAULO
2021

CAIO YAGO VILELA F00JED7

RAFAEL H. DOS SANTOS LEMES F054802

DETECÇÃO E RECONHECIMENTO DE OBJETOS EM IMAGENS

Relatório de Atividade Prática Supervisionada (APS) para avaliação no 6ºsemestre letivo do curso de Ciência da Computação apresentado à UNIP – Universidade Paulista.

Orientador (es): Danilo Pereira

LIMEIRA – SÃO PAULO

2021

SUMÁRIO

1. OBJETIVO

Com base nos estudos que tivemos no decorrer dos anos na faculdade em relação à programação e com a APS na matéria Processamento de Imagem e Visão Computacional, iremos desenvolver uma melhoria da informação visual para a interpretação humana com Processamento automático de imagem (Visão Computacional), Criar um código que Leia a Imagem e faça a detecção do objeto. Já que atualmente a inteligência artificial é o futuro da Tecnologia, isso demonstra que estamos sempre em constante evolução nos meios de Informação e para o desenvolvimento desse trabalho usaremos a programação em Python.

2. INTRODUÇÃO

O mundo de hoje está cheio de dados, e as imagens constituem uma parte significativa desses dados. Porém, antes de poderem ser utilizadas, essas imagens digitais devem ser processadas, analisadas e manipuladas para melhorar sua qualidade ou extrair alguma informação que possa ser utilizada.

O Processamento Digital de Imagens (PDI) é uma importante área da computação, suas aplicações são diariamente utilizadas em diversos setores da indústria e comercio, como em sistemas médicos, reconhecimento de objetos e até na identificação de pessoas. Sabendo disso foi proposto um sistema que possa fazer o reconhecimento automático de placas de veículos. Sua utilização pode auxiliar na questão que envolve gerenciamento de entradas e saídas das empresas, visando diminuir os riscos de falhas de gerenciamento e podendo aumentar o nível de segurança dos ambientes.

Na questão logística do sistema, a aplicação de Processamento Digital de Imagens nesses tipos de processos, tem como objetivo uma otimização dos resultados no que se diz respeito ao gerenciamento. Como conceituado por

Ballou (1993), “Logística é o processo de planejamento do fluxo de materiais com o objetivo de entregar o necessário com qualidade e com melhor tempo, potencializando seus recursos e ganhando em qualidade de serviços”

Nesse sentido, uma das mais recentes aplicações desta solução tem sido sua instalação em portos marítimos para leitura e gerenciamento de placas de contêineres, e a bordo carros de polícia para capturar placas de dentro do veículo.

3. CONCEITOS BÁSICOS DE UTILIZAÇÃO

Está aplicação pode ser utilizada para Centros de Controle, gestão do tráfego da cidade, propriedades industriais, estradas ou fronteiras, bem como instalações críticas de controle de acesso, como aeroportos, estações de trem ou ônibus, solução de captura e gerenciamento de placas de veículo, geração de listas brancas/negras para acesso restrito, rastreamento e rastreabilidade de veículos através de desenhos em tempo real ou vídeo gravado, bem como consulta e gerenciamento de imagens associadas a vídeos do centro de controle ou de qualquer outro ponto conectado à Internet. gravação automática de vídeo em caso de incidentes e a notificação e gerenciamento de alarmes.

É um recurso que permite a identificação de uma placa de veículo no momento da sua identificação, que é efetuada através da leitura realizada por uma câmera, monitorando o tráfego e identificando veículos.

Pode-se configurar para que apenas seja necessário o reconhecimento da placa para a validação, ou o sistema faça uma dupla verificação, validando o veículo e mais um tipo de autenticação como, leitura facial, tag, cartão, senha, Fiscalização, Controle e estatísticas de acesso; Monitoramento de tráfego; Identificação de veículos roubados e irregulares; integrável aos sistemas, em blitz policiais, selecionando os veículos que realmente possuem alguma irregularidade, evitando assim constrangimento de cidadãos idôneos e aumentando a eficiência da fiscalização.

3.1 APLICAÇÃO

O primeiro passo para o desenvolvimento do artigo, foi a captura ou aquisição das imagens que posteriormente seriam processadas, lembrando que foram utilizadas imagens de placas não oficiais meramente para estudos, Porém que se utilizam dos padrões reais, tanto como cor, tamanho e fonte da letra. As imagens utilizadas foram obtidas de duas formas, a princípio imagens pré-carregadas no computador com diferentes condições de ruídos, iluminação, tamanhos e cores obtidas da Internet e a posteriormente utilizando-se de dispositivos de capturas como câmera e webcams. Para isso são necessários um sensor e um digitalizador. O sensor converterá a informação ótica em sinal elétrico e o digitalizador transformará a imagem analógica em imagem digital (KHOSHAFIAN & BAKER, 1996). No caso desse projeto, foi utilizado um dispositivo com resolução de captura 720x480, produzindo assim uma saída de imagem com resolução *High Definition* (HD)

Esse recurso também pode ser utilizado para emitir alertas sobre veículos cadastrados em uma black list, informando os operadores sobre alguma informação importante relacionado ao carro ou moto.

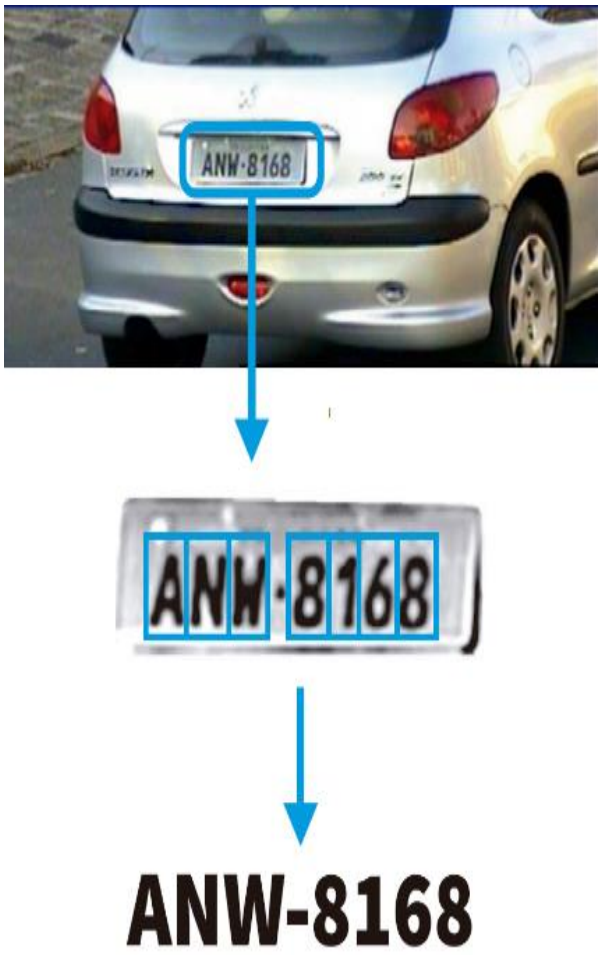
O sistema também permite obter relatórios, imagens associadas a vídeo e estatísticas graças a vários filtros personalizáveis, como o número da placa.

3.2 DESENVOLVIMENTO

Todas as funções de processamento das imagens descritas e utilizadas no desenvolvimento do projeto foram obtidas através da biblioteca OpenCV, disponível para diversas linguagens de programação, como foi disposto na fundamentação do artigo. Para a programação foi utilizado Python. Python é uma linguagem de programação multiplataforma desenvolvido sob uma licença de código aberto aprovada pela OSI20, tornando-o livremente utilizável e distribuível, mesmo para uso comercial. Sua licença é administrada pela *Python SoftwareFoundation*.

A linguagem permite que programador expresse suas ideias em menos linhas de código sem reduzir sua legibilidade, sendo totalmente compatível com o

OpenCV. Foram apresentadas algumas dificuldades como permitir a captura de várias placas de veículos em pé e em movimento, fazer o Algoritmo baseado em localização de caracteres independente de cor; Alta confiabilidade e precisão na leitura das placas; Robustez de reconhecimento nas mais variadas condições de imagem: luminosidade, clima e condições da placa, Reconhecimento em tempo real.



3.3 CÓDIGOS

```
try :
    from  PIL  import  Image
exceto  ImportError :
    import  Image
import  pytesseract

# Se você não tiver o executável tesseract em seu PATH, inclua o seguinte:
pytesseract . pytesseract . tesseract_cmd = r '<full_path_to_your_tesseract_executable>'
# Exemplo tesseract_cmd = r'C: \ Arquivos de programas (x86) \ Tesseract-OCR \ tesseract '

# Imagem simples para
imprimir string ( pytesseract . Image_to_string ( Image . Open ( 'test.png' )))

# A fim de ignorar as conversões de imagem de pytesseract, apenas use o caminho de imagem relativo ou absoluto
# NOTA: Neste caso, você deve fornecer imagens compatíveis com tesseract ou o tesseract retornará um erro de
impressão ( pytesseract . Image_to_string ( 'test.png' ))

# Lista de idiomas disponíveis para
impressão ( pytesseract . Get_languages ( config = '' ))
```

```
# Imagem de texto em francês para
imprimir string ( pytesseract . Image_to_string ( Image . Open ( 'test-european.jpg' ), lang = 'fra' ))

# Processamento em lote com um único arquivo contendo a lista de vários caminhos de arquivo de imagem
imprimir ( pytesseract . Image_to_string ( 'images.txt' ))

# Timeout / terminar o trabalho tesseract após um período de tempo
try :
    print ( pytesseract . Image_to_string ( 'test.jpg' , timeout = 2 )) # Timeout após 2 segundos
    print ( pytesseract . Image_to_string ( 'test.jpg' , timeout = 0,5 )) # Tempo de espera depois de meio segundo
, excepto RuntimeError como timeout_error :
    processamento # Tesseract é terminada
    passe

# Obtenha estimativas de caixa delimitadora
print ( pytesseract . Image_to_boxes ( Image . Open ( 'test.png' )))

# Obtenha dados detalhados, incluindo caixas, confidências, linha e números de página
imprimir ( pytesseract . Image_to_data ( Image . Open ( 'test.png' )))

# Obtenha informações sobre orientação e
```

impressão de detecção de script (pytesseract . Image_to_osd (Image . Open ('test.png')))

Obtenha um PDF pesquisável

pdf = pytesseract . image_to_pdf_or_hocr ('test.png' , extension = 'pdf')

com open ('test.pdf' , 'w + b') como f :

f . escrever (pdf) # tipo de pdf é bytes por padrão

Obtenha a saída

HOCR hocr = pytesseract . image_to_pdf_or_hocr ('test.png' , extension = 'hocr')

Obtenha saída ALTO XML

xml = pytesseract . image_to_alto_xml ('test.png')

Suporte para imagem OpenCV / objetos de matriz NumPy

import cv2

img_cv = cv2 . imread (r '/ <caminho_para_imagem> /digits.png')

Por padrão, o OpenCV armazena imagens no formato BGR e, uma vez que o pytesseract assume o formato RGB,

precisamos converter de BGR para o formato / modo RGB:

img_rgb = cv2 . cvtColor (img_cv , cv2 . COLOR_BGR2RGB)

print (pytesseract . image_to_string (img_rgb))

```
# OU

img_rgb = Imagem . frombytes ( 'RGB' , img_cv . forma [: 2 ], img_cv , 'bruto' , 'BGR', 0 , 0 )

imprimir ( pytesseract . Image_to_string ( img_rgb ))
```

- **Funções**

get_languages Retorna todos os idiomas atualmente suportados pelo Tesseract OCR.

get_tesseract_version Retorna a versão do Tesseract instalada no sistema.

image_to_string Retorna a saída não modificada como string do processamento de OCR do Tesseract

image_to_boxes Retorna o resultado contendo caracteres reconhecidos e seus limites de caixa

image_to_data Retorna o resultado contendo os limites da caixa, confidências e outras informações. Requer Tesseract 3.05+. Para obter mais informações, verifique a documentação do Tesseract TSV

image_to_osd Retorna o resultado contendo informações sobre orientação e detecção de script.

image_to_alto_xml Retorna o resultado na forma do formato ALTO XML do Tesseract.

run_and_get_output Retorna a saída bruta do Tesseract OCR. Oferece um pouco mais de controle sobre os parâmetros enviados para o tesseract.

- **Parâmetros**

image_to_data(image, lang=None, config='', nice=0, output_type=Output.STRING, timeout=0, pandas_config=None)

image Object or String - PIL Image / NumPy array ou caminho de arquivo da imagem a ser processada pelo Tesseract. Se você passar o objeto em vez do caminho do arquivo, o pytesseract converterá implicitamente a imagem para o modo RGB .

lang String - string do código de idioma do Tesseract. O padrão é engse não for especificado! Exemplo para vários idiomas:lang='eng+fra'

config String - Quaisquer sinalizadores de configuração personalizados adicionais que não estão disponíveis por meio da função pytesseract. Por exemplo:config='--psm 6'

nice Integer - modifica a prioridade do processador para a execução do Tesseract. Não compatível com Windows. Nice ajusta a gentileza de processos do tipo Unix.

output_type Atributo de classe - especifica o tipo de saída, o padrão é string. Para obter a lista completa de todos os tipos suportados, verifique a definição da classe pytesseract.Output .

timeout Integer ou Float - duração em segundos para o processamento de OCR, após o qual o pytesseract terminará e gerará RuntimeError.

pandas_config Dict - apenas para o tipo Output.DATAFRAME . Dicionário com argumentos personalizados para pandas.read_csv . Permite que você personalize a saída de image_to_data .

- **Uso da CLI:**

pytesseract [-l lang] arquivo_de_imagem

3.3.1 FUNÇÃO FINDCONTOURS

O *findContours* é uma função utilizada para encontrar contornos e bordas, podem ser explicados com uma curva que une todos os pontos contínuos ao longo do limite com a mesma cor e intensidade. É uma técnica bastante utilizada para se encontrar padrões e objetos em uma imagem digital.
A Função recebe alguns argumentos como parâmetros, o primeiro é a imagem de origem, o segundo é o modo de recuperação de contorno, o terceiro é o método de aproximação de contorno. O método retorna uma imagem, contornos e suas hierarquias. Esses contornos na verdade são uma lista de todos os contornos obtidos da imagem. Cada um desses contornos individualmente é representado por uma matriz de coordenadas (x, y) de pontos de fronteira do objeto. Veja o trecho do código:

“imagem, contornos, hierarquia

cv2.findContours(img_result, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)”

Aplicando esse resultado com a função *drawContours* é possível desenhar esses contornos obtidos na imagem.

3. 3.2 DRAWCONTOURS

Os contornos ficaram espalhados por toda a imagem, no entanto a característica buscada é o retângulo. Para a resolução desse problema, é necessário especificar para a função *drawContours* apenas os contornos que desejo pintar, nesse caso somente os que formam um retângulo. Por fim utilizando de outras duas funções da biblioteca, as funções *arcLength* e a *approxPolyDP.*

A função *arcLength,* também conhecida como comprimento de arco, verificará se os contornos encontrados formam um perímetro, ou seja, se ele é fechado. “Ele se aproxima de uma forma de contorno a outra forma com menos número de vértices, dependendo da precisão que forem especificados”.

No segundo caso a função *approxPolyDP* conhecida como aproximação de contorno, fará como que no caso de perímetros de contornos forem problemáticos ou seja com menos número de vértices, haja uma aproximação para seu formato mais semelhante. Por exemplo, no caso do perímetro for um quadrado, e devido a problemas na imagem se obteve uma forma ruim do quadrado, essa função pode-se configurada com os parâmetros para se conseguir o formato geométrico mais semelhante ao que se tem originalmente. A função de segmentação por retângulos proposta pelo algoritmo resulta em um percentual de acerto satisfatório para a implementação de um protótipo para essa pesquisa.

Utilizando das funções citadas é possível se obter uma segmentação básica do local da placa. Para isso foi desenvolvido uma implementação de um código em Python para a demonstração dos resultados.

```
12  def desenhaContornos(contornos, imagem):
13
14      for c in contornos:
15          # perimetro do contorno, verifica se o contorno é fechado
16          perimetro = cv2.arcLength(c, True)
17          if perimetro > 120:
18              #aproxima os contornos da forma correspondente
19              approx = cv2.approxPolyDP(c, 0.03 * perimetro, True)
20              #verifica se é um quadrado ou retangulo de acordo com a qtd de vertices
21              if len(approx) == 4:
22                  (x, y, alt, lar) = cv2.boundingRect(c)
23                  cv2.rectangle(imagem, (x, y), (x + alt, y + lar), (255, 0, 0), 2)
24                  #cv2.drawContours(imagem, [c], -1, (0, 255, 0), 1)
25                  roi = imagem[(y+15):y + lar, x:x + alt]
26                  #amplia a imagem 4x
27                  cv2.imwrite("C:/Tesseract-OCR/saidas/roi-img.jpg", roi)
28
29      return imagem
```

3. 3.3 RESULTADOS OBTIDOS

O código descrito na imagem, é simplesmente a nível de demonstração das funções descritas anteriormente, a fim de evidenciar no contexto da imagem a localização da placa, através de retângulos. A função recebe uma matriz de contornos, a partir disso será desenhada os contornos na imagem original.

Para essa operação é utilizado um laço “for” como uma estrutura de repetição

(linha 14), a fim de que a cada interação verifique através das funções arcLength (linha16) e approxPolyDP (linha 19), o perímetro, e seja constatado que ele se trata de um retângulo validando a característica buscada.

Deve-se observar a condição perímetro > 120, (linha 17), onde acontece um teste logico no algoritmo. Este teste trata-se de um filtro de contornos que serão desenhados na imagem. Este valor de 120 é um parâmetro testado para um retângulo satisfatório, representando todos os retângulos encontrados na imagem com dimensões de objetos vistos a uma distância entre 3 e 5 metros do observador, que neste caso e a câmera.

Eliminando todos os perímetros fora da condição, é possível fazer uma verificação dos retângulos que possivelmente poderão ser a placa.

Fazendo isso, o retorno da função evidenciará os contornos mais fiéis possíveis, resultando em uma saída limpa com possível o alvo desejado, fazendo assim a segmentação da placa.

Concluído a localização da placa, é seguido para a operação de recorte do perímetro encontrado, para esse passo e utilizado a função ROI passando as coordenadas (x, y) encontradas nos contornos. Veja o trecho do código:

“roi = imagem[y:y+alt, x:x+lar]“

4. PROJETO

Na computação tudo pode ser abstraído a partir de uma sequência de números binários, ou seja 0 e 1, a partir disso com o avanço das tecnologias e o aumento gradativo dos dados e consecutivamente informações, foi indagado como poderiam ser representados alguns tipos de dados complexos, como as imagens. Imagens são representações visuais de objetos através de alguma técnica ou processo.

No ambiente computacional às imagens são representadas digitalmente por uma matriz bidimensional de pontos f (x, y) de bits9 ou binários (0, 1), processadas e transformadas em representações conhecidas como pixels, a junção de todos os pontos forma o que conhecemos como imagem. Na computação existem dois tipos dessas representações, as imagens vetoriais e matriciais. Imagens vetoriais são uma descrição dos dados (imagens ou abstrações) na forma de um espaço contínuo, conhecido como descrição vetorial, são formas que descrevem o início e fim de cada segmento de reta, ou os pontos de controle de uma curva, ou ainda elementos que definem um sólido, como por exemplo o lado de um cubo e o raio de uma esfera.

Por se utilizar de vetores e se transformarem utilizando funções matemáticas são mais leves e ocupam menos espaço sem perca de qualidade.

Diferentemente das imagens vetoriais, uma descrição matricial ou raster (bitmap) também conhecido como mapa de bits, as imagens matriciais são imagens que contêm a descrição em uma escala de cor de cada ponto ou pixel. Sabe-se que o pixel é a menor medição de uma imagem os pixels ou pels como também são conhecidos, correspondem a qualquer valor em uma escala de cores RGB (vermelho, verde, azul) podendo ser alternado sua ordem. Em uma imagem por exemplo com escala de 8 bits por exemplo, onde valor do pixel está entre 0 e 255, sendo 0 (preto) e 255 (branco) e o entremeio a mistura das cores, é representado uma imagem em tons de cinza.

Visto que as imagens são pontos ou pixels onde cada um representa uma cor, sabendo disso como o que foi descrito por Scaño (2002), “Cores de um ponto de vista da física nada mais é do que luz. A luz percorre o espaço, podendo se comportar de duas formas distintas, como uma onda e como uma partícula, isso caracteriza a natureza dual da luz”. No caso das cores, deve-se se estudar o comportamento na forma de onda, neste caso ela se transforma em um tipo de energia, sendo uma radiação eletromagnética para cada comprimento de onda. Esta função toma o nome de Distribuição Espectral de Potência. A faixa dimensional do espectro visível pelo olho humano é delimitada a partir baixa frequência percebida como vermelha ou faixa de radiação infravermelha até o lado de mais alta frequência perceptível como a cor violeta conhecida ou sugestivamente como faixa de radiação ultravioleta. Para cada comprimento de onda de luz visível é associado a percepção de uma cor.

Sabendo sobre a forma que a luz atua e sobre sua distribuição espectral, pode-se então entender que a mesma pode ser moldada por funções matemáticas unidimensionais, dependendo do seu comprimento de onda, ou seja, cores são as sensações humanas de diferentes porções finitas do espectro de luz. Sua principal definição seria uma característica perceptual da espécie humana, onde cada espécie de possui uma percepção de comprimento de onda diferente. Imagens digitais, também possuem cores digitais, onde cada nível de cor é representado por um sistema de três cores o mais utilizado em cores digitais é o padrão RGB,

onde cada cor é definida pela quantidade de azul (Blue), verde (Green), vermelho (Red) que a compõem, coexistindo os modelos como por exemplo, HSV10, YUV11, XYZ12, etc.

Neste contexto utilizado tem-se cada cor sendo representada por uma tripla de inteiros que variam de acordo com, $0 \leq R \leq 255$, $0 \leq G \leq 255$ e $0 \leq B \leq 255$. Por exemplo a cor cinza (ciano), digitalmente seria representada por (128, 128, 128) no sistema RGB. Sabendo como são representadas as imagens digitais, tal como suas características e como atua no contexto computacional, uma das suas utilizações são na etapa de processamento digital, na qual serão aplicados correções e tratamentos.

5. RELAÇÃO ENTRE PROCESSAMENTO DE IMAGENS E COMPUTAÇÃO GRÁFICA

Em geral, autores de livros em Computação Gráfica (CG) e Processamento de Imagens (PDI) vêm tratando as duas áreas como distintas. O conhecimento em ambas as áreas tem crescido consideravelmente, o que tem permitido a resolução de problemas cada vez mais complexos. Numa visão simplificada, CG busca imagens fotos-realísticas de cenas tridimensionais geradas por computador, enquanto PDI tenta reconstruir uma cena tridimensional a partir de uma imagem real, obtida através de uma câmera. Neste sentido, PDI busca um procedimento inverso ao de CG, análise ao invés de síntese, mas ambas as áreas atuam sobre o mesmo conhecimento, o qual inclui, dentre outros aspectos, a interação entre iluminação e objetos e projeções de uma cena tridimensional em um plano de imagem. O cenário envolvendo todas as disciplinas que tenham algum ingrediente de processamento da informação visual, dentre as quais a CG e o PDI ocupam posição de destaque, é definido por alguns autores como Computação Visual.

Imagem Original

Passo 1: Gray Scale (escala de cinza)

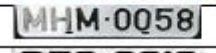
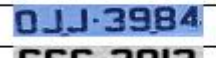
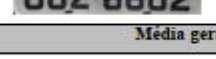
Passo 2: Thresholding (Limiarização)

Passo 3: Glaussian Blur (desfoque)

6. CONCLUSÃO

O que pode ser visto é que o reconhecimento dos caracteres nem sempre é de 100% de acerto, é comum que a interface não se saia bem em alguns casos de má formação dos caracteres e ou modelo da fonte, foram submetidas placas de numerações iguais, porém com fontes diferentes, o resultado foi a interpretação equivocada pela interface nos caracteres “A” e “I”. No primeiro caso a letra “A” foi confundida pela letra “H”, já no segundo caso o problema acontece nos caracteres “C” e no digito “1”, confundidos por “D e I” respectivamente.

Pode-se concluir que, a resolução de todas as variáveis se torna algo impossível para a interface. Contudo, tendo em vista que, podendo ser feito um processamento mais avançado, uma melhor parametrização da interface, somado a utilização de logicas básica na logística do reconhecimento, como percentuais mínimos de acertos, se o veículo se encontra posicionado corretamente, entre outros, com isso é possível obter resultados consistente e bastante satisfatórios no reconhecimento do veículo.

Indices	Origem	Alvo	Resultado	Acertos %
1		ATF-7354	_AT:F-7354	100 %
2		ABC-1234	HBC-1234	90 %
3		ABC-1234	ABD I234	80 %
4		BFG-8663	BF_O-86.63	90 %
5		FOR-1904	FUR- I904	80 %
6		HMG-0248	HMG-0248	100 %
7		ICE-2973	|E_E-2973	80 %
8		JDR-0312	JIIR-D3I2	40 %
9		JJK- 1960	JJK-I960	90 %
10		JSQ-7436	JED-74_3B	30 %
11		LPT-4625	LPT-4625	100 %
12		NNS-4646	NNS-4B_4B	80 %
13		MHM-0058	AI1M-0058	70 %
14		PEC-2013	PEC-2D|3	80 %
15		OJJ-3384	!_J-3384	70 %
16		CSC-2013	CSC-20I3	90 %
17		VCR-0000	VCR-D000	90 %
18		AXN-8888	AXN-3888	90 %
19		ASY-3826	ASY-3825	90 %
20		OOZ-8802	UUZ-BBJJZ	10 %
Média geral (%)				77.5%

7. BIBLIOGRÁFIA

PADILHA, A.J.: **Simulation of Data Distribution Strategies for LU Factorization on Heterogeneous Machines**. In: Proceedings of 17 International Parallel and Distributed Processing Symposium (IPDPS 2003), IEEE Computer Society Press. Los Alamitos (2003).

PAZ, E. P.; CUNHA, T. N. **Iniciação ao Processamento Digital de Imagens**. Rio de Janeiro: UFRJ, 1988 PROCESSAMENTO de Imagem Digital: Conceito de Pixel. Disponível em: <http://www.w3ii.com/pt/dip/concept_of_pixel.html>. Acesso em: 11 abr. 2017.

CABRAL, Fabio Augusto; PEREIRA MACHADO, Victor Hugo.**IDENTIFICADOR DE PLACA VEICULAR**: alternativa para segurança em escolas. 2014. 52 f. Monografia (PósGraduação Em Tecnologia) - Universidade Tecnológica Federal Do Paraná, Curitiba, 2014. 1. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/>. Acesso em: 29 mar. 2017.

DE CAMPOS, Tatiane Jesus. **Reconhecimento de caracteres alfanuméricos de placas em imagens de veículos**. 2001. 120 p. Dissertação de Mestrado (Mestrado em Ciência da Computação) - Universidade Federal do Rio Grande do Sul, Porto Alegre, [2002]. Disponível em:<http://www.lume.ufrgs.br/handle/10183/2329>. Acesso em: 04 mar. 2017

8. FICHA DE ATIVIDADES

FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, iniciação Científica, trabalhos Individuais e em grupo, práticas de ensino e outras)

NOME: Caio Yoon Tsilva

RA: F00JED-1 CURSO: Ciência da Computação

CAMPUS: Limeira - SP SEMESTRE: 6º Semestre TURNO: Noturno

DATA	ATIVIDADE	TOTAL DE HORAS	ASSINATURA ALUNO	ASSINATURA PROFESSOR
10/09	Reunião em grupo para discutir o trabalho	6	Caio.Y	
11/09	Início da escrita do algoritmo do trabalho	6	Caio.Y	
12/09	Reunião e apresentação inicial do projeto	4	Caio.Y	
13/09	Finalização do algoritmo para código	3	Caio.Y	
24/09	Implementação do algoritmo no Spyder e Pycharm	3	Caio.Y	
25/09	Correção dos erros nos Código 1º fase	4	Caio.Y	
01/10	Finalização e teste 1º fase concluída com exito	3	Caio.Y	
02/10	Início da 2º fase do projeto como proposto	3	Caio.Y	
08/10	implementação do algoritmo no pycharm	4	Caio.Y	
09/10	Testes com o projeto 2º fase	4	Caio.Y	
15/10	Finalização e teste 2º fase concluída com exito	3	Caio.Y	
16/10	Tirando duvidas com o professor Danilo	3	Caio.Y	
17/10	Testes com o projeto 3º fase	2	Caio.Y	
18/10	Finalização do projeto fases 1º,2º,3º com sucesso	4	Caio.Y	
19/10	Elaboração dos documentos do projeto	2	Caio.Y	
20/10	Reunião para correção de erros dos documentos	3	Caio.Y	
21/10	Finalização do projeto e apresentação para o grupo	2	Caio.Y	
22/10	Apresentação	1	Caio.Y	

TOTAL DE HORAS: 60 Horas

FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, iniciação Científica, trabalhos Individuais e em grupo, práticas de ensino e outras)

NOME: Rafael Henrique dos Santos Lemes

RA: F054802 CURSO: Ciência da Computação

CAMPUS: Limeira - SP SEMESTRE: 6º Semestre TURNO: Noturno

DATA	ATIVIDADE	TOTAL DE HORAS	ASSINATURA ALUNO	ASSINATURA PROFESSOR
10/09	Reunião em grupo para discutir o trabalho	6	RafaelHSLemes	
11/09	Início da escrita do algoritmo do trabalho	6	RafaelHSLemes	
12/09	Reunião e apresentação inicial do projeto	4	RafaelHSLemes	
13/09	Finalização do algoritmo para código	3	RafaelHSLemes	
24/09	Implementação do algoritmo no Spyder e Pycharm	3	RafaelHSLemes	
25/09	Correção dos erros nos Código 1º fase	4	RafaelHSLemes	
01/10	Finalização e teste 1º fase concluída com exito	3	RafaelHSLemes	
02/10	Início da 2º fase do projeto como proposto	3	RafaelHSLemes	
08/10	implementação do algoritmo no pycharm	4	RafaelHSLemes	
09/10	Testes com o projeto 2º fase	4	RafaelHSLemes	
15/10	Finalização e teste 2º fase concluída com exito	3	RafaelHSLemes	
16/10	Tirando duvidas com o professor Danilo	3	RafaelHSLemes	
17/10	Testes com o projeto 3º fase	2	RafaelHSLemes	
18/10	Finalização do projeto fases 1º,2º,3º com sucesso	4	RafaelHSLemes	
19/10	Elaboração dos documentos do projeto	2	RafaelHSLemes	
20/10	Reunião para correção de erros dos documentos	3	RafaelHSLemes	
21/10	Finalização do projeto e apresentação para o grupo	2	RafaelHSLemes	
22/10	Apresentação	1	RafaelHSLemes	

TOTAL DE HORAS: 60 Horas