

**UNIVERSIDADE FEDERAL DE SANTA MARIA  
CENTRO DE TECNOLOGIA  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA  
COMPUTAÇÃO**

Leonardo de Abreu Schmidt

**AUTOMATIZAÇÃO DO RECONHECIMENTO DE BURACOS EM  
RODOVIAS USANDO INTELIGÊNCIA COMPUTACIONAL**

Santa Maria, RS

2020

# **AUTOMATIZAÇÃO DO RECONHECIMENTO DE BURACOS EM RODOVIAS USANDO INTELIGÊNCIA COMPUTACIONAL**

**Leonardo de Abreu Schmidt**

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Ciência da Computação (PPGCC), Área de Concentração em Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de  
**Mestre em Ciência da Computação**

**Orientadora: Prof<sup>a</sup>. Dr<sup>a</sup>. Marcia Pasin**

**Santa Maria, RS, Brasil**

**2020**

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

Schmidt, Leonardo de Abreu  
Automatização do reconhecimento de buracos em rodovias  
usando inteligência computacional / Leonardo de Abreu  
Schmidt.- 2020.  
76 p.; 30 cm

Orientadora: Marcia Pasin  
Dissertação (mestrado) - Universidade Federal de Santa  
Maria, Centro de Tecnologia, Programa de Pós-Graduação em  
Ciência da Computação , RS, 2020

1. Detecção de buracos 2. Classificadores 3. Redes  
Neurais Artificiais 4. Máquinas de Vetores de Suporte I.  
Pasin, Marcia II. Título.

Sistema de geração automática de ficha catalográfica da UFSM. Dados fornecidos pelo autor(a). Sob supervisão da Direção da Divisão de Processos Técnicos da Biblioteca Central. Bibliotecária responsável Paula Schoenfeldt Patta CRB 10/1728.

Declaro, LEONARDO DE ABREU SCHMIDT, para os devidos fins e sob as penas da lei, que a pesquisa constante neste trabalho de conclusão de curso (Dissertação) foi por mim elaborada e que as informações necessárias objeto de consulta em literatura e outras fontes estão devidamente referenciadas. Declaro, ainda, que este trabalho ou parte dele não foi apresentado anteriormente para obtenção de qualquer outro grau acadêmico, estando ciente de que a inveracidade da presente declaração poderá resultar na anulação da titulação pela Universidade, entre outras consequências legais.

**Universidade Federal de Santa Maria  
Centro de Tecnologia  
Programa de Pós-Graduação em Ciência da Computação**

A Comissão Examinadora, abaixo assinada,  
aprova a Dissertação de Mestrado

**AUTOMATIZAÇÃO DO RECONHECIMENTO DE BURACOS EM  
RODOVIAS USANDO INTELIGÊNCIA COMPUTACIONAL**

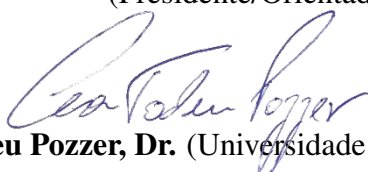
elaborada por  
**Leonardo de Abreu Schmidt**

como requisito parcial para obtenção do grau de  
**Mestre em Ciência da Computação**

**COMISSÃO EXAMINADORA:**



**Marcia Pasin, Dr<sup>a</sup>.**  
(Presidente/Orientadora)



**Cesar Tadeu Pozzer, Dr.** (Universidade Federal de Santa Maria)



**Ana Trindade Winck, Dr.** (Universidade Federal de Ciências da Saúde de Porto Alegre)

Santa Maria, 27 de Março de 2020.

## AGRADECIMENTOS

Agradeço à minha família, em especial aos meus pais Silaine e Wolnei que sempre me incentivaram à correr atrás das oportunidades, me deram apoio e força, sem eles isso não seria possível, são minha fonte de inspiração, de comprometimento e caráter.

Também agradeço ao meu irmão Gustavo, que é uma pessoa muito especial na minha vida.

Agradeço igualmente aos meus avós, que são uma parte muito importante da minha vida, e sempre estiveram ao meu lado.

À minha namorada Carin que sempre me ajudou, me incentivou a continuar e me esforçar em tudo.

Aos meus tios por todo o apoio em todos esse tempo.

Também quero agradecer aos meus professores, por todos os ensinamentos, pelos conhecimentos passados, por todo o crescimento, acadêmico e pessoal, que me foi concedido através desse tempo em que estive na instituição.

Enfim, meu sincero muito obrigado por tudo que todos vocês representam para mim, espero levar sempre comigo um pouco de cada um de vocês.

*“Você vive e aprende. De qualquer forma, você vive. ”*

— DOUGLAS ADAMS

# RESUMO

Dissertação de Mestrado  
Programa de Pós-Graduação em Ciência da Computação  
Universidade Federal de Santa Maria

## **AUTOMATIZAÇÃO DO RECONHECIMENTO DE BURACOS EM RODOVIAS USANDO INTELIGÊNCIA COMPUTACIONAL**

AUTOR: LEONARDO DE ABREU SCHMIDT

ORIENTADORA: MARCIA PASIN

Local da Defesa e Data: Santa Maria, 27 de Março de 2020.

Buracos no pavimento de ruas e rodovias são um problema bem conhecido, que é agravado diante do crescente desenvolvimento da tecnologia de veículos autônomos. Essa tecnologia, portanto, necessita incorporar sistemas eficientes de detecção automática de buracos. De fato, existem diferentes métodos para a detecção automática de buracos, porém, esses métodos ainda precisam ser mais profundamente avaliados tanto em relação à sua precisão quanto à sua velocidade de resposta para que sejam usados em contextos reais. A fim de contribuir com um passo na direção dessa avaliação, este trabalho apresenta a proposta de uma solução para detecção de buracos em imagens, sendo o objetivo a detecção da existência de um buraco e a determinação do mesmo na imagem. O trabalho apresenta duas partes principais, sendo a primeira uma arquitetura de classificação das imagens, a qual utiliza a técnica de Histograma de Gradientes Orientados (HOG) como extração de características da imagem, e dois classificadores; Redes Neurais Artificiais (RNA) e Máquinas de Vetores de Suporte (SVM). O objetivo é a otimização de parâmetros e a determinação do melhor classificador para o problema. A segunda parte apresenta uma proposta de arquitetura de detecção de buracos em imagens, determinando não só a existência mas também a localização de um buraco na imagem. Essa tarefa utiliza descritores de textura de *Haralick* em um sistema de *grid* aplicado as imagens, somado a arquitetura de classificação da primeira parte. Também foram avaliados os tempos de processamento dessas arquiteturas. Os resultados apresentados mostram que a arquitetura usando o classificador de Rede Neural Artificial é a melhor opção para detecção de buracos, atingindo 83% de *IoU* (*Intersection of Union*) além de 73% de acurácia e 93% de precisão. Os resultados sobre os tempos de processamento também foram favoráveis ao classificador de Rede Neural chegando a 9 *frames* por segundo.

**Palavras-chave:** Redes Neurais Artificiais. Máquinas de Vetores de Suporte. Detecção de Buracos. Pavimentação de Rodovias.

# ABSTRACT

Master's Dissertation  
Undergraduate Program in Computer Science  
Federal University of Santa Maria

## **AUTOMATIZAÇÃO DO RECONHECIMENTO DE BURACOS EM RODOVIAS USANDO INTELIGENCIA COMPUTACIONAL**

**AUTHOR: LEONARDO DE ABREU SCHMIDT**

**ADVISOR: MARCIA PASIN**

Defense Place and Date: Santa Maria, March 27<sup>th</sup>, 2020.

Potholes in asphalt pavements and highways are a well-known problem, which is aggravated by the growing development of autonomous vehicle technology. This technology, therefore, needs to incorporate efficient automatic pothole detection systems. In fact, there are different methods for automatic pothole detection, however, these methods still need to be further evaluated both in terms of their accuracy and speed of response in order to be used in real contexts. In order to contribute with a step in the direction of this evaluation, this work presents the proposal of a solution for the detection of potholes in images, being the detection of the existence of a pothole and the determination of the same in the image, the objective of this work. The work has two main parts, the first being an image classification architecture, which uses the Histogram of Oriented Gradients (HOG) technique as an extraction of image characteristics, and two classifiers; Artificial Neural Networks and Support Vector Machines. The objective is to optimize parameters and determine the best classifier for the problem. The second part presents a proposal for architecture to detect potholes in images, determining not only the existence but also the location of a pothole in the image. This task uses texture descriptors from *Haralick* in a system of *grid* applied to the images, added to the classification architecture of the first part. The processing times of these architectures were also evaluated. The results presented show that architecture using the Artificial Neural Network classifier is the best option for pothole detection, reaching 83% *IoU* (*Intersection of Union*) in addition to 73% accuracy and 93% precision. Processing time is also favorable to the Neural Network classifier reaching about 9 *frames* per second (FPS).

**Keywords:** Artificial Neural Network, Support Vector Machine, VANETs.

## LISTA DE FIGURAS

Figura 1.1 – Exemplo um cenário para um sistema de informação para a detecção de buracos em asfalto. ....	16
Figura 3.1 – Histograma dos gradientes orientados com vetores de tamanho 6 por célula. ....	28
Figura 3.2 – Esquema de normalização de valores com agrupamento de células em blocos. ....	29
Figura 3.3 – Aplicação do HOG em uma imagem de buraco. ....	30
Figura 3.4 – Gráficos das funções de ativação e suas respectivas derivadas. ....	33
Figura 3.5 – Multi-Layer Perceptron com três camadas. ....	34
Figura 3.6 – Matriz de coocorrência. ....	37
Figura 4.1 – Arquitetura para análise de viabilidade. ....	38
Figura 4.2 – Extrato do primeiro <i>dataset</i> contendo 184 imagens. Sendo as três primeiras, amostras negativas e as três últimas amostras positivas. ....	40
Figura 4.3 – Extrato do segundo <i>dataset</i> contendo 3200 imagens. Sendo as três primeiras, amostras negativas e as três últimas amostras positivas. ....	40
Figura 4.4 – Médias para todas as configurações $C_n$ . ....	45
Figura 4.5 – Resultados obtidos para a RNA usando HOG de 2 e 5 orientações. ....	46
Figura 4.6 – Resultados obtidos para o SVM usando HOG de 2 e 5 orientações. ....	47
Figura 4.7 – Tempos de execução para o HOG. ....	48
Figura 4.8 – Tempos de treinamento para a RNA e o SVM. ....	48
Figura 4.9 – Médias para todas as configurações e validação cruzada para k-fold=10. ....	49
Figura 4.10 – Resultados obtidos para o SVM usando HOG de 2 e 5 orientações. ....	50
Figura 4.11 – Curvas ROC SVM. ....	50
Figura 4.12 – Resultados obtidos para a RNA usando HOG de 2 e 5 orientações. ....	51
Figura 4.13 – Curvas ROC RNA. ....	52
Figura 4.14 – Resultados de classificação entre os <i>datasets</i> . ....	53
Figura 4.15 – Curvas ROC dos resultados entre os datasets D1 e D2 para RNA e SVM. ...	54
Figura 5.1 – Arquitetura proposta para a detecção de buracos em vídeos utilizando descritores de textura de Haralick em um grid de imagem, aplicação da técnica de HOG como descritor da imagem e classificação. ....	55
Figura 5.2 – Grid de imagem com $M \times N$ células, onde cada célula possui o tamanho de $m_1 \times n_1$ pixels. ....	56
Figura 5.3 – Gráficos gerados pelos diferentes descritores de Haralick em relação à imagem da Figura 5.2 usando um <i>grid</i> de $20 \times 8$ , com células de tamanho $25 \times 25$ . ....	57
Figura 5.4 – Sinal gerado pelo valor do descritor de correlação de Haralick para cada célula do <i>grid</i> . ....	58
Figura 5.5 – Agrupamento de células do <i>grid</i> que apresentam valor do descritor acima do <i>threshold</i> calculado. ....	61
Figura 5.6 – <i>Threshold</i> Média. ....	63
Figura 5.7 – <i>Threshold</i> Mediana. ....	63
Figura 5.8 – <i>Threshold</i> Desvio Padrão. ....	63
Figura 5.9 – Exemplo de localização e marcação de um buraco em imagem. Em verde, a caixa de <i>ground-truth</i> , e em vermelho a caixa predita. ....	65
Figura 5.10 – Formula para cálculo da interseção da união. ....	65
Figura 5.11 – Amostras de <i>frames</i> resultantes do processamento com a marcação em verde das <i>bounding-boxes</i> dos buracos ....	66
Figura 5.12 – IoU, acurácia e precisão referente ao processamento do vídeo. ....	67



Figura 5.13 – Tempos totais de execução de RNA.....	68
Figura 5.14 – Tempos totais de execução de SVM. ....	68

## LISTA DE TABELAS

Tabela 4.1 – Configurações HOG testadas. ....	43
Tabela 4.2 – Configurações da rede neural. ....	44

## LISTA DE ABREVIATURAS E SIGLAS

AA	Automobile Association
AUC	Area Under Curve
RBF	Função de Base Radial ou <i>Radial Basis Function</i>
CNT	Confederação Nacional de Transportes
CV	<i>Cross-Validation</i>
FCM	Fuzzy C-Means
FN	False Negative ou Falso Negativo
FP	False Positive ou Falso Positivo
HOG	Histogram of Oriented Gradients
IoU	Intersection of Union
ML	Machine Learning
MLP	Perceptron de Múltiplas Camadas ou <i>Multilayer Perceptron</i>
RNA	Rede Neural Artificial
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TN	True Negative ou Verdadeiro Negativo
TP	True Positive ou Verdadeiro Positivo

# SUMÁRIO

<b>1 INTRODUÇÃO</b> .....	15
<b>1.1 Objetivos</b> .....	17
1.1.1 Objetivos gerais .....	17
1.1.2 Objetivos específicos .....	17
<b>1.2 Metodologia</b> .....	18
<b>1.3 Estrutura do texto</b> .....	18
<b>2 REVISÃO BIBLIOGRÁFICA</b> .....	20
<b>2.1 Detecção de buracos baseada em vibração</b> .....	20
<b>2.2 Detecção de buracos baseada em processamento de imagens</b> .....	21
<b>2.3 Detecção de buracos baseada em visão computacional</b> .....	23
<b>3 FUNDAMENTAÇÃO TEÓRICA</b> .....	25
<b>3.1 Histograma de Gradientes Orientados</b> .....	25
3.1.1 Gradientes orientados .....	26
3.1.2 Histogramas.....	27
3.1.2.1 Preenchimento do Histograma .....	27
3.1.3 Normalização .....	29
3.1.4 Vetor Final .....	29
<b>3.2 Aprendizado de Máquina</b> .....	30
3.2.1 Aprendizado supervisionado .....	31
3.2.2 Aprendizado não supervisionado .....	31
3.2.3 Rede Neural Artificial (RNA) .....	32
3.2.4 Máquinas de Vetores de Suporte (SVM) .....	35
<b>3.3 Descritores de Textura</b> .....	36
3.3.1 Descritores de Haralick .....	36
<b>4 METODOLOGIA PARA CLASSIFICAÇÃO DE IMAGENS DE BURACOS</b> .....	38
<b>4.1 Ambiente de experimentação</b> .....	39
<b>4.2 Conjunto de dados</b> .....	39
<b>4.3 Métricas e modelos de avaliação</b> .....	40
4.3.1 Curva ROC .....	42
4.3.2 Validação Cruzada.....	42
<b>4.4 Parametrizações</b> .....	43
4.4.1 Parametrização HOG .....	43
4.4.2 Rede Neural Artificial .....	43
4.4.3 SVM.....	44
<b>4.5 Resultados <i>dataset D1</i></b> .....	45
4.5.1 Resultados .....	45
4.5.2 Tempo de processamento HOG.....	47
4.5.3 Tempo de treinamento .....	48
<b>4.6 Resultados <i>dataset D2</i></b> .....	49
<b>4.7 Validação entre os <i>datasets</i></b> .....	52

<b>5 METODOLOGIA PARA DETECÇÃO DE BURACOS EM IMAGENS DE VÍDEO</b>	<b>55</b>
<b>5.1 Divisão da imagem em <i>grid</i></b>	<b>56</b>
<b>5.2 Descritor de textura de Haralick</b>	<b>56</b>
<b>5.3 Thresholds</b>	<b>59</b>
<b>5.4 Agrupamento</b>	<b>60</b>
<b>5.5 HOG e Classificação</b>	<b>61</b>
<b>5.6 Experimentos</b>	<b>62</b>
5.6.1 Escolha do <i>threshold</i>	62
<b>5.7 Avaliação do modelo de detecção</b>	<b>64</b>
5.7.1 IoU - <i>Intersection of Union</i>	64
<b>5.8 Resultados</b>	<b>65</b>
5.8.1 Tempo de Processamento	67
<b>6 CONCLUSÕES</b>	<b>70</b>
<b>6.1 Observações sobre os resultados</b>	<b>70</b>
<b>6.2 Trabalhos futuros</b>	<b>71</b>
<b>6.3 Apontamento final</b>	<b>72</b>
<b>REFERÊNCIAS</b>	<b>73</b>

# 1 INTRODUÇÃO

A ocorrência de buracos em estradas e rodovias é um problema recorrente em nosso país. Buracos nos asfalto são responsáveis por causar problemas como interrupção do tráfego, atrasos, desconforto para passageiros, danos a veículos que demandam gastos extras com reparos, além de colaborarem para o acontecimento de acidentes de trânsito. Em especial, no Brasil, onde transporte rodoviário constitui o principal sistema logístico, manter rodovias em condições é imprescindível. Pavimentos com problemas danificam os veículos de carga, aumentam o consumo de combustível, o tempo das viagens e, o conseqüentemente, custo operacional do transporte.

No Brasil, a Confederação Nacional de Transportes (CNT) (CNT et al., 2018) analisou 107 mil quilômetros de estradas e classificou mais de 50% delas como regular, ruim ou muito ruim. Porém, buracos no asfalto não é um problema apenas no Brasil. De acordo com uma pesquisa publicada em março de 2016 pela UK Automobile Association (AA) (ALLIANCE, 2016) com a participação de 25 mil motoristas britânicos, danos causados por buracos nas estradas afetam 39% deles. Pneus, carrocerias ou outras peças do veículo foram danificados. Normalmente, os buracos são causados devido a depressões nas faixas ou rachaduras no asfalto, que se deterioram, formando desde pequenos buracos até grandes depressões na pista. Adicionalmente, condições climáticas, como chuvas fortes e neve, podem exercer grande influência sobre a situação de buracos e deterioração das estradas.

Com a popularização de sensores, dispositivos móveis com acesso à internet e a grande difusão da Internet das Coisas (Internet of Things ou IoT), ocorre também um aumento gigantesco diário na geração de dados, os quais, de alguma forma necessitam de processamento e análise (RIBEIRO; GROLINGER; CAPRETZ, 2015). Com a disseminação de novas tecnologias, cada vez mais dados (fotos, vídeos, texto, sinais) são gerados por dispositivos móveis ou embarcados, e a partir daí também surgem novas oportunidades de criação de aplicações para processamento desses dados orientados à natureza multimídia. Tudo isso proporcionou um ambiente propício ao desenvolvimento de sistemas capazes de agir inteligentemente sobre uma grande variedade de domínios. Segundo (KOKAR; ENDSLEY, 2012) é preciso coletar informações sobre o ambiente (geralmente de diferentes fontes), tomar decisões baseadas nessa coleta e no conhecimento, agir de acordo com as decisões, coletar *feedbacks* do ambiente para complementar o conhecimento existente e, dessa forma, tomar decisões melhores no futuro.

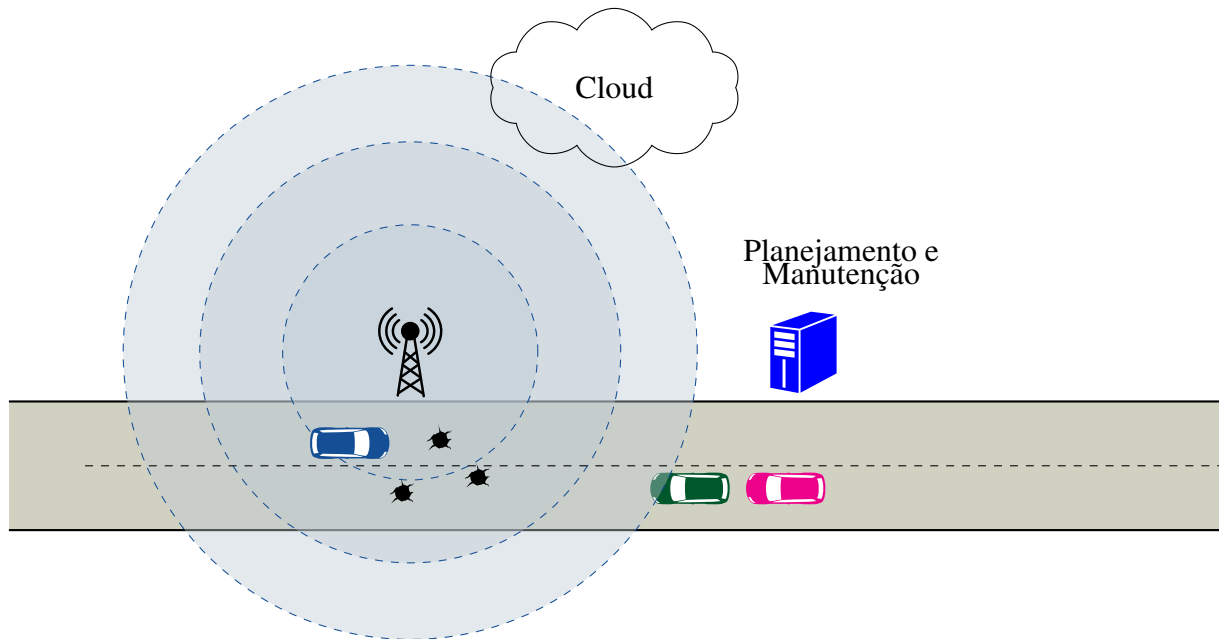


Figura 1.1 – Exemplo um cenário para um sistema de informação para a detecção de buracos em asfalto.

Soma-se a este cenário, a crescente implantação de veículos com novas tecnologias incorporadas e com a iminente ampla disponibilidade de veículos autônomos e colaborativos, com sistemas de informação embarcados, representando soluções para automação do processo de detecção buracos, ganharam atenção. Nesse sentido, a detecção de buracos pode ser um serviço oferecido automaticamente pelos carros, para não apenas veículos autônomos mas também veículos tradicionais. A informação de um buraco pode ser disseminada para os demais motoristas, se o serviço estiver disponível em sistema de informação distribuído, conforme ilustra o cenário da Figura 1.1. Além disso, possivelmente, as empresas de manutenção de estradas e o governo podem compilar dados obtidos por sensores embarcados em veículos em um sistema de informação para planejar ações de manutenção e melhorar as condições de pavimentação em rodovias e estradas.

Nesse contexto, técnicas de processamento de imagens e de classificação, como Redes Neurais Artificiais (RNAs) e Máquinas de Vetores de Suporte (SVMs), vêm se destacando cada vez mais, e chamando a atenção para as possibilidades de tudo que pode ser desenvolvido. Imagens coletadas por câmeras instaladas nos veículos podem ser analisadas com o objetivo de detecção de buracos no asfalto.

## 1.1 Objetivos

### 1.1.1 Objetivos gerais

O objetivo geral deste trabalho é o desenvolvimento de uma arquitetura de detecção de buracos em pavimentos de asfalto usando técnicas de processamento de imagens e aprendizado de máquina, mais precisamente aprendizagem e classificação. Também é objetivo do trabalho explorar a velocidade de processamento da arquitetura proposta, de forma a avaliar a possibilidade de aplicação em tempo real.

### 1.1.2 Objetivos específicos

São objetivos específicos deste trabalho:

- Montar uma base de dados de imagens para o processo de aprendizagem e classificação,
- Realizar a classificação das imagens da base de dados entre imagens de buracos e imagens sem buraco,
- Desenvolver uma proposta de arquitetura de detecção para localização de buracos em imagens de vídeo,
- Realizar *Tunning* das técnicas utilizadas para classificação e para localização,
- Avaliar a medida de custo-qualidade das técnicas utilizadas e a possibilidade da aplicação em tempo real.

(ENIGOA et al., 2016) descreve diferentes técnicas para a detecção automatizada de buracos: mecanismos baseados em vibração, abordagens de detecção em imagens e análise baseada em visão computacional. As abordagens baseadas em vibração exigem que o veículo entre no buraco para permitir a detecção. As técnicas de reconstrução 2D/3D baseadas em visão computacional permitem a detecção de buracos sem a necessidade de o carro bater no buraco, mas requerem mais poder computacional do que as abordagens baseadas em vibrações.

Este trabalho segue a técnica de análise baseada em imagem. Esta técnica permite a detecção de buracos sem a necessidade do veículo atingir o buraco e também normalmente requer menos processamento do que as soluções de reconstrução 2D/3D. Basicamente, com a técnica de análise baseada em imagem, uma imagem capturada por uma câmera é analisada.



Se um buraco for detectado em tempo real, este poderá ser evitado pelo carro. No entanto, o processamento precisa ser rápido o suficiente para que o carro/motorista tenha a chance de evitar o buraco. O custo computacional da solução não deve inviabilizar a aplicação da técnica em tempo real. Este trabalho oferece um passo nessa direção, ao apresentar um estudo de viabilidade da abordagem baseada em processamento de imagens e classificação.

## 1.2 Metodologia

Assim, este trabalho segue a seguinte metodologia:

- revisão do estado da arte, enfocando detecção de buracos em imagens de pavimentos de asfalto,
- proposição de um sistema para detecção automática de buracos em imagens no asfalto,
- avaliação de dois classificadores de imagens para fins de detecção de buracos, no que diz respeito à relação qualidade do resultado produzido e custos de processamento para verificar a viabilidade de aplicação de solução proposta em tempo real, e
- desenvolvimento de um sistema de localização/marcação de buracos em imagens.

Na abordagem aqui seguida, em contraste com outros trabalhos (HOU; WANG; GONG, 2007), (LI et al., 2009), (MOAZZAM et al., 2013), (NIENABER; BOOYSEN; KROON, 2015), (OUMA; HAHN, 2017), (VIGNESHWAR; KUMAR, 2016), (WANG et al., 2017), são usados algoritmos de processamento de imagens e descritores de textura com assistência de uma RNA e de SVM para permitir a detecção de buracos em imagens de pavimentos de asfalto.

## 1.3 Estrutura do texto

O resto deste texto está organizado da seguinte forma:

- Capítulo 2 apresenta uma revisão bibliográfica dos trabalhos relacionados à detecção automatizada de buracos com a utilização de diferentes técnicas,
- Capítulo 3 apresenta a fundamentação teórica das técnicas e algoritmos utilizados no presente trabalho,

- Capítulo 4 descreve a metodologia de classificação nas imagens de buracos empregada nesse trabalho bem como seus métodos de avaliação,
- Capítulo 5 apresenta a metodologia utilizada para realizar a localização e marcação de buracos em imagens, bem como os métodos utilizados para avaliação da solução proposta,
- Capítulo 6 apresenta as conclusões sobre os resultados obtidos e os trabalhos futuros.

## 2 REVISÃO BIBLIOGRÁFICA

Com o desenvolvimento de novas tecnologias para o setor automotivo, incluindo veículos autônomos e assistência avançada ao motorista, a detecção de buracos é hoje um desafio relevante. De fato, no futuro, a detecção de buracos será um serviço básico para carros autônomos e para carros colaborativos. Assim, a indústria automobilística já trabalha com o desenvolvimento de sistemas para tratar a existência de buracos no asfalto. Por exemplo, a tecnologia de detecção de buracos do Ford Focus<sup>1</sup> permite o ajuste automático da suspensão do carro para proporcionar conforto ao usuário, dada a gravidade dos danos no asfalto. Para os carros Jaguar Land Rover<sup>2</sup> equipados com os recursos de comunicação, um sistema de informações fornece previamente aos motoristas, informações sobre buracos para ajudá-los a evitar acidentes ou possibilitar o uso de rotas alternativas. No entanto, o sistema precisa ser implementado com um serviço de localização eficiente, para que o buraco possa ser localizado corretamente e com precisão adequada.

Na academia, muito esforço tem sido feito na direção da detecção automatizada de buracos nos últimos anos (KIM; RYU, 2014). No geral, os métodos existentes são divididos em métodos baseados em vibração, métodos de detecção em imagens e métodos baseados em visão computacional. Esses métodos são descritos resumidamente a seguir, usando como base uma pesquisa bibliográfica que descreve o estado da arte.

### 2.1 Detecção de buracos baseada em vibração

Nas abordagens de detecção de buracos baseadas em vibrações (ERIKSON; GIROD; HULL, 2008), (MEDNIS et al., 2011), (YU; SALARI, 2011), (ZOYSA et al., 2007), os veículos usam sensores incorporados, como codificadores e acelerômetros, para detectar buracos. O codificador permite detectar posições e velocidade do carro. O acelerômetro acoplado ao veículo mede a sua aceleração nos três eixos. Assim, quando um carro atinge um buraco, que é uma variação na condição da superfície, o acelerômetro indica uma variação em um ou mais eixos. No entanto, o resultado dessa técnica pode variar de acordo com: qualidade dos sensores, limites aplicados – que são normalmente escolhidos manualmente (e, portanto, suscetíveis

<sup>1</sup> <https://media.ford.com/content/fordmedia/feu/en/news/2018/06/28/innovative-pothole-detection-system-irons-out-the-bumps-for-all-.html>

<sup>2</sup> <https://www.landrover.com/experiences/news/pothole-detection.html>

a erros) – e qualidade do esquema de suspensão do veículo – que pode influenciar o resultado já que um sistema de suspensão eficiente pode mascarar um pequeno buraco por exemplo). Além disso, as abordagens baseadas em vibrações só permitem identificar um buraco quando um veículo o atinge. Se o carro/motorista desviar do buraco, o buraco não será detectado, o que inviabiliza posterior mapeamento do buraco em um sistema de informação para demais motoristas. Como vantagem, as abordagens baseadas em vibração podem ser implementadas usando a tecnologia já disponível e normalmente com baixo custo monetário, uma vez que os acelerômetros estão amplamente disponíveis em *smartphones*, permitindo, por exemplo, a implementação de soluções de detecção participativa, além de não apresentarem um elevado custo computacional.

(Silveira Rodrigues et al., 2019) apresenta uma proposta para detecção de buracos baseada em *threshold* adaptativo, utilizando transformadas Wavelet de Haar, processando sinais obtidos com acelerômetro de um *smartphone*.

## 2.2 Detecção de buracos baseada em processamento de imagens

Abordagens baseadas em detecção de buracos em imagens (NIENABER; BOOYSEN; KROON, 2015), (WANG et al., 2017), (HOU; WANG; GONG, 2007), (LI et al., 2009), (MO-AZZAM et al., 2013), (YU; SALARI, 2011) permitem identificar buracos e outras falhas no asfalto usando imagens capturadas por câmeras instaladas em carros. Com o objetivo de melhorar a qualidade da solução, este procedimento pode ser combinado com outras técnicas. Por exemplo, (WANG et al., 2017) propuseram uma abordagem de detecção de buracos com base na decomposição de *wavelets* e processamento morfológico usando segmentação e extração de imagens por bordas. No entanto, o aumento da qualidade da solução pode afetar o tempo de execução dos algoritmos utilizados, inviabilizando a detecção em tempo real (dado a natureza do problema).

Segundo (VIGNESHWAR; KUMAR, 2016), a detecção de buracos em imagens pode envolver diferentes métodos de processamento, como filtragem de imagens, segmentação e agrupamento de imagens (com algoritmos como K-Means ou Fuzzy C-Means (FCM), por exemplo), detecção de bordas e uso de limiares. No entanto, algumas técnicas introduzem falhas no processo de detecção de objetos. Por exemplo, o processo de detecção de borda usa filtros de imagem para destacar as alterações na textura, indicando um possível objeto na cena. Se o filtro não é aplicado corretamente, essa técnica pode remover mais informações da ima-

gem do que o necessário, ou as bordas detectadas podem ser apenas parte do objeto de interesse, dificultando a detecção do objeto por um classificador.

(OUMA; HAHN, 2017) propuseram uma solução de baixo custo aplicando a transformada *wavelet* à filtragem de imagens baseada em textura de várias escalas e usando um algoritmo FCM para construir um classificador capaz de identificar o pavimento com defeito e sem defeito.

Em (LIN; LIU, 2010), a medida de textura baseada no HOG é extraída como os recursos da região da imagem e a SVM não linear é usado para identificar se uma região de destino é um buraco. No experimento realizado pelos autores, 50 imagens foram usadas como amostras de treinamento e as outras 30 imagens foram usadas como amostras de teste. Os resultados experimentais mostraram que, após o treinamento dos dados normalizados pela SVM, os modelos propostos reconheceram corretamente as 30 amostras de imagens de teste.

(YU; SALARI, 2011) propuseram uma abordagem para melhorar a qualidade da imagem da câmera, uma vez que a qualidade das imagens é limitada pela falta de iluminação e por sombreamento. Assim, um sistema de iluminação artificial foi usado para permitir a captura de imagens de mais alta qualidade. O método proposto consiste em uma fonte de luz ativa que projeta um padrão de linha de raios laser na superfície do pavimento, uma câmera para capturar imagens e os algoritmos de processamento de imagem para identificar buracos. Após a captura das imagens do pavimento, as regiões correspondentes aos buracos são representadas por uma matriz de ladrilhos quadrados e a forma estimada do buraco é determinada. As medidas verticais e horizontais de estresse da pista, o número total de ladrilhos e informações de índice de profundidade são calculadas, fornecendo entrada para uma RNA de três camadas para classificação da magnitude do buraco e classificação do tipo de fissura. O sistema foi validado usando um conjunto de dados com 100 imagens. Os resultados experimentais demonstraram que o modelo proposto é eficiente para a detecção de buracos e fissuras.

Finalmente, a detecção de objetos é um assunto de grande relevância para a pesquisa em análise de imagens e também em inteligência artificial. Sua aplicação em diversas áreas automatiza diferentes tarefas e auxilia no processo de tomada de decisão. No contexto específico de redes de transporte e sistemas de transporte inteligentes, as técnicas de reconhecimento de imagens motivaram a pesquisa sobre o reconhecimento de sinais de trânsito, veículos, pedestres, buracos e outros objetos relativos ao tráfego. No geral, um estudo mais detalhado do esforço computacional versus a qualidade das soluções ainda precisa ser feito com relação às

abordagens baseadas na detecção de imagens aplicadas à detecção de buracos.

### 2.3 Detecção de buracos baseada em visão computacional

A visão computacional é o processo de modelagem e replicação da visão humana usando software e hardware. Soluções baseadas em visão computacional (BUZA; OMANOVIC; HUSEINOVIC, 2013), (JOG et al., 2012), (KOCH; BRILAKIS, 2011), (KOCH; JOG; BRILAKIS, 2013), (LOKESHWOR; DAS; SUD, 2013) usam sinais de um LiDAR (Detecção por Variação da Luz) ou imagens de uma câmera estéreo para detectar buracos. Essas soluções exigem que os veículos sejam equipados com essas tecnologias. Além disso, é necessária uma alta capacidade computacional para que o resultado gerado esteja pronto com rapidez suficiente para permitir que os veículos/motoristas desviem dos buracos detectados.

Em (AZHAR et al., 2016), a detecção de buracos é abordada do ponto de vista da visão computacional. Diferentes tipos de imagens, com buracos e sem buracos, foram considerados para experimentos. Levando em conta a natureza baseada na forma da aparência dos buracos, os recursos de Histogramas de Gradientes Orientados (HOG) foram calculados para um conjunto de dados de 50 imagens de entrada. As características foram treinadas e classificadas usando o classificador Naïve Bayes, resultando na identificação da entrada como buracos ou sem imagens de buracos. Esquema de segmentação de corte de gráfico normalizado foi empregado para localizar buracos nessas imagens. O esquema proposto foi testado em um conjunto de dados de 70 imagens de pavimento. Os resultados dos experimentos alcançaram 90% de acurácia, 86,5% de precisão e 94,1 % de *recall*.

(KANG; CHOI, 2017) desenvolveram um sistema de detecção de buracos usando uma câmera e dispositivos 2D LiDAR. A combinação de sistema de sensor heterogêneo permite melhorar a precisão da detecção de buracos. Juntamente com esse conjunto de sensores, outras técnicas são usadas para melhorar a qualidade da solução - processamento para redução de ruído, agrupamento, extração de segmento de linha e função de gradiente de dados de buracos nas imagens do LiDAR, controle de brilho, binarização, extração de bordas e extração de objetos nas imagens da câmera.

Em (BUZA; OMANOVIC; HUSEINOVIC, 2013) foi proposto um modelo de processamento de imagens e clusterização espectral para detecção de buracos usando segmentação de imagem pelo método de *thresholding* de Otsu (Otsu, 1979) e extração de formas. A abordagem proposta no trabalho foi testada com 50 diferentes amostras de imagens de buracos. A acu-

rácia do modelo proposto e testado com as 50 imagens foi calculada manualmente atingindo aproximadamente 81%.

Embora as técnicas baseadas em visão computacional possibilitem desempenho adequado em relação à qualidade da solução, elas também apresentam alto custo na aquisição de equipamentos e de sensores. No *Autonomy Day* da Tesla em 2019, Elon Musk ganhou as manchetes ao atacar a tecnologia LiDAR (*Light Detection and Ranging*). Segundo ele "qualquer um que confie no LiDAR está condenado". Embora o LiDAR tenha sido amplamente adotado por desenvolvedores de veículos autônomos por mais de uma década, Musk declarou que o único hardware que a Tesla precisa é do conjunto de câmeras e sensores existentes já instalados em seus veículos. Porém, segundo publicado em (HAYFLICK, 2019), a predição de dimensões 3D de um veículo apresentam erros discrepantes quando da utilização unicamente de modelos 2D (Câmeras). Segundo a publicação, a utilização de câmeras em conjunto com LiDARs é a melhor solução para uma correta detecção do objeto em cena e a determinação de suas dimensões. Em seus resultados, o estudo ainda aponta uma *Intersection of Union* (IoU), métrica para mediar a acurácia de um detector de objetos, total de 32.1% em todo *dataset* testado apenas com a utilização de câmeras, quando na verdade o ideal seria algo próximo à 90%.

### 3 FUNDAMENTAÇÃO TEÓRICA

Existem diversas técnicas de processamento de imagens, descritores de texturas e algoritmos de aprendizado. Neste trabalho técnicas como Histogramas de Gradientes Orientados, classificadores e descritores de textura de *Haralick* são utilizadas. Este capítulo faz um estudo a respeito dessas técnicas e suas utilizações em arquiteturas de detecção e reconhecimento de objetos em imagens. A seção 3.1 explora o funcionamento do algoritmo de Histograma de Gradientes Orientados (HOG) como descritor de características de imagens. A seção 3.2 descreve os algoritmos de aprendizado supervisionado, sendo Redes Neurais Artificiais e Máquinas de Vetores de Suporte dois dos principais algoritmos utilizados na exploração de problemas de classificação. Finalmente, a seção 3.3 descreve o funcionamento de descritores de textura de Haralick e sua aplicação no problema deste trabalho sobre detecção de buracos.

#### 3.1 Histograma de Gradientes Orientados

A técnica de Histograma de Gradientes Orientados, comumente chamada de HOG, é um descritor de características de imagens. Esse descritor representa imagens sob outro aspecto e é representado em formato de um vetor contendo as características dessa imagem. Além disso, o descritor HOG gera o vetor com quantidade de informações menor do que o requerido pela imagem inteira, sendo necessário menos espaço de armazenamento.

Pode-se dizer que se uma imagem for visualizada como um descritor HOG, o que seria visto é uma representação de variação entre pixels e bordas representadas por vetores com determinadas direções e intensidades. Nessa técnica existem essencialmente Três parâmetros principais que regem a qualidade e quantidade de informações do vetor de características, além de três passos principais que são computados para a geração do vetor HOG final. Em relação aos parâmetros, pode-se configurar o tamanho das células usadas na técnica, a quantidade de orientações nos vetores de histograma, e também o tamanho dos blocos na parte final de normalização. Todos esses parâmetros são utilizados ao longo dos passos da técnica. Em relação aos passos, o primeiro é o cálculo dos gradientes orientados, que é explicado na seção 3.1.1, posteriormente a seção 3.1.2 mostra como é executado o cálculo do vetor HOG propriamente dito e, por último, na seção 3.1.3 é realizada uma normalização desse vetor.



### 3.1.1 Gradientes orientados

Um gradiente é um vetor que indica a direção e sentido no qual se obtém maior variação de uma dada grandeza. No caso dos gradientes orientados na técnica de HOG, existe anexado à esse vetor uma magnitude. No caso de processamento de imagens, as grandezas são referentes aos valores assumidos por um pixel e os gradientes representam as variações entre os pixels de uma imagem, sendo possível calcular a direção e magnitude dessas variações. O primeiro passo para gerar um vetor HOG é o cálculo dos gradientes horizontais e verticais de uma imagem. Esses gradientes são calculados filtrando a imagem com *kernels* (filtros) horizontais e verticais, ambos compostos pelo mesmo conjunto de valores  $k = \{-1, 0, 1\}$ , sendo cada *kernel* processado separadamente. Para cada *kernel*, percorre-se a imagem aplicando-o em cada posição de pixel da imagem. Isso é feito de acordo com as Equações 3.1 e 3.2:

$$g_x(i, j) = p(i, j - 1) * -1 + p(i, j) * 0 + p(i, j + 1) * 1, \quad (3.1)$$

$$g_y(i, j) = p(i - 1, j) * -1 + p(i, j) * 0 + p(i + 1, j) * 1. \quad (3.2)$$

em que  $i$  e  $j$  são as posições de pixel na imagem original e  $g_x$  e  $g_y$  são os valores dos componentes dos gradientes horizontal e vertical respectivamente calculados para cada pixel da imagem.

Depois de filtrar a imagem com os dois *kernels* e gerar os valores de gradientes horizontais e verticais, o próximo passo é calcular a magnitude resultante do gradiente e a direção em que esse gradiente se apresenta. Ambos são importantes pois discriminam bordas, regiões de grande variação, ou até mesmo homogeneidade em uma regiões onde há continuidade de cor. Isso é feito aplicando as Equações 3.3 e 3.4 para cada gradiente de cada pixel  $(g_x, g_y)$ . A equação 3.3 apresenta a magnitude do gradiente  $G_{i,j}$  e é dada pela raiz quadrada do somatório dos quadrados das componentes horizontal e vertical do gradiente, enquanto que a direção (orientação) desse vetor é dada pelo arcotangente da divisão da componente vertical pela horizontal.

$$G_{i,j} = \sqrt{g_x^2 + g_y^2}, \quad (3.3)$$

$$\theta_{i,j} = \arctan \frac{g_y}{g_x}. \quad (3.4)$$

Esses valores são utilizados pelo próximo passo da abordagem, o cálculo dos histogramas, descrito na próxima seção.

### 3.1.2 Histogramas

Até então o cálculo das magnitudes e orientações para cada pixel gera adicionalmente duas matrizes de mesmo tamanho que a imagem, uma para as magnitudes e outra para as orientações de cada magnitude respectivamente. Porém, como dito no início dessa seção, o HOG nesse passo diminui a quantidade de informações necessárias para representar a imagem.

Nessa etapa, são calculados os histogramas para os gradientes orientados ( $G_{i,j}$  e  $\theta_{i,j}$ ), porém esses histogramas não são calculados para a imagem inteira de uma só vez. Nesse ponto, a imagem é dividida em sub-imagens de tamanhos iguais (ex:  $8 \times 8$  pixels), nas quais serão calculados os histogramas gerando os vetores HOG. Esses tamanhos são um dos parâmetros que são definidos para a técnica, conforme descrito no início dessa seção. Cada sub-imagem, aqui chamada de célula, é representada por um vetor de tamanho  $\theta$ , onde  $\theta$  representa a quantidade de orientações que são amostradas por célula, e também se configura como o segundo parâmetro a ser definido na técnica.

O histograma é então calculado para cada célula das matrizes de magnitude e orientações gerando um vetor HOG por célula, ou seja, o tamanho final desses vetores sofre influência do tamanho definido para células e da quantidade de orientações amostradas por histograma, e é claro pelo tamanho da imagem. Cada histograma é um vetor com  $\theta$  posições que representa a quantidade de ângulos amostrados de  $0$  à  $180^\circ$ . A Figura 3.1 mostra um exemplo de funcionamento da geração dos vetores de histograma onde são amostrados seis orientações por célula e cada posição representa respectivamente as orientações  $0$ ,  $30$ ,  $60$ ,  $90$ ,  $120$  e  $150$ . Como existem 16 células, o vetor de saída dessa etapa terá tamanho 96.

Os vetores são amostrados de  $0$  à  $180^\circ$  e não de  $0$  à  $360^\circ$  pois a representação em  $45^\circ$ , por exemplo, corresponde a mesma representação do vetor em  $225^\circ$ . O que importa apenas são as direções, os sentidos e a magnitude.

#### 3.1.2.1 Preenchimento do Histograma

Para preencher o histograma que possui  $\theta$  posições, utilizam-se as matrizes de orientação e magnitudes. A matriz de orientação da célula de imagem na Figura 3.1 indica em qual posição do histograma o valor respectivo de magnitude na outra matriz será colocado. Nesse ponto, podem ocorrer três casos.

- Caso 1: O primeiro caso diz respeito à orientação ter correspondente direto no vetor

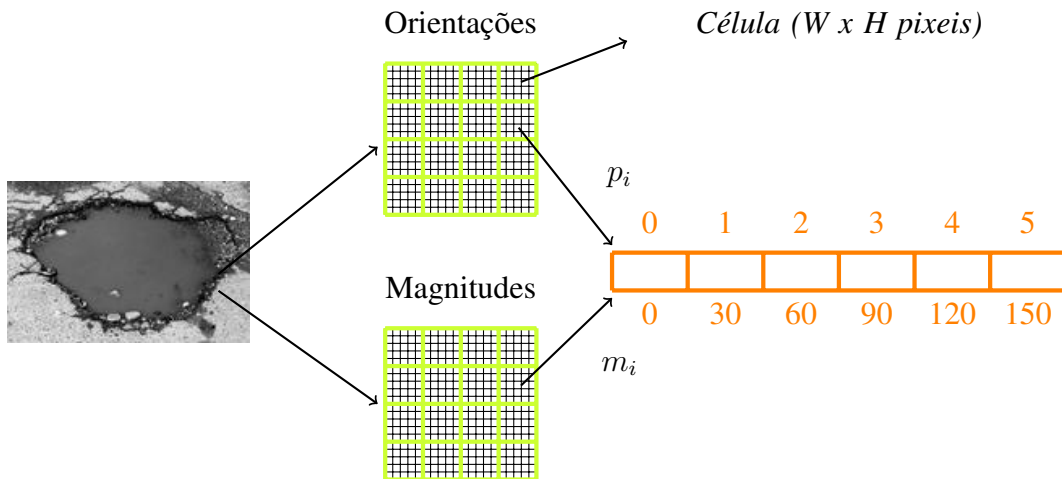


Figura 3.1 – Histograma dos gradientes orientados com vetores de tamanho 6 por célula.

histograma, no caso a Figura 3.1 tem uma orientação 60, por exemplo. Nesse caso coloca-se o valor de magnitude inteiro na posição correspondente no histograma,

- Caso 2: o segundo caso é a orientação estar exatamente entre duas posições do vetor. Por exemplo, uma orientação 45 está entre o 30 e o 60. Nesse caso o valor é dividido entre as células. Por exemplo, o 45 seria dividido, 22.5 para a posição 1 (correspondente a orientação 30) do histograma e 22.5 para a posição 2 (correspondente ao 60),
- Caso 3: o terceiro caso não corresponde a nenhum dos anteriores e, portanto, o valor de magnitude está entre duas orientações, porém com diferentes distância entre elas. Por exemplo, o valor 42 está distante 12 unidades do 30 e 18 unidades do 60. O valor seria dividido proporcionalmente entre as posições. Por exemplo, o 42 que está à 18 unidades do 30, teria 60% de seu valor (25.2) colocado na posição 1 do histograma, enquanto que o restante (19.8) seria colocado na posição 2.

Exemplificando, uma célula de  $8 \times 8$  de uma imagem possui 192 valores, sendo os 64 pixels com 3 canais de cores. Essa mesma célula tem matrizes de gradientes orientados de tamanho  $8 \times 8 \times 2$  devido às matrizes de direção e magnitude, gerando 256 valores por célula (128 por matriz). E, finalmente, o histograma com  $\theta$  orientações amostradas na célula fornece um vetor que reduz significativamente as informações da célula original. Após a construção do histograma em cada célula, as matrizes auxiliares de orientações e magnitudes não são mais necessárias e podem ser descartadas. O próximo e último passo é a normalização dos histogramas.

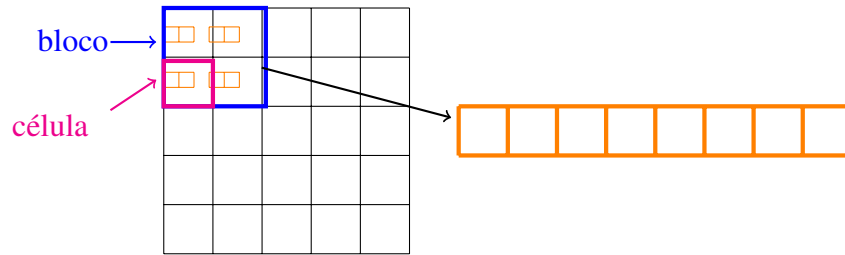


Figura 3.2 – Esquema de normalização de valores com agrupamento de células em blocos.

### 3.1.3 Normalização

O processo do HOG gera um vetor de histograma por célula da imagem, sendo o tamanho dos histogramas e o tamanho das células parâmetros pré-definidos. A normalização tem o objetivo principal de retirar componentes que tem interferência direta na imagem, como por exemplo a luz. Mais especificamente, esta etapa normaliza os vetores calculados por célula na etapa anterior, agrupando elas em conjuntos com  $n \times n$  células chamadas de blocos. Esse agrupamento é a concatenação de cada vetor HOG de células contidas no bloco e o tamanho  $n$  é o último dos três parâmetros mencionados no início da seção e que são definidos na técnica de HOG. Na Figura 3.2, por exemplo, existe uma amostragem de orientações com  $\theta = 2$  e tamanho de bloco de  $2 \times 2$ , gerando um vetor histograma concatenado por bloco de tamanho 8. O processo de normalização em si primeiramente concatena os histogramas das células que compõem o bloco conforme mostra a Figura 3.2 e posteriormente divide os valores de cada posição por um valor de normalização. Esse valor é encontrado a partir da Equação 3.5, em que  $V$  é o vetor concatenado no bloco e  $i$  é o índice do elemento desse vetor:

$$valor\_normalizado = \sqrt{\sum_{i=0}^N V_i^2}. \quad (3.5)$$

### 3.1.4 Vetor Final

Finalmente, todos os vetores histograma dos blocos da normalização são concatenados, resultando em um único vetor final com tamanho definido na Equação 3.6:

$$tamanho = \theta \cdot \alpha^2 \cdot \left( \left( \frac{W}{\lambda} - 1 \right) \cdot \left( \frac{H}{\lambda} - 1 \right) \right), \quad (3.6)$$

em que  $W$  é a largura da imagem,  $H$  é a altura da imagem,  $\theta$  é o número de orientações amostrados por célula,  $\lambda$  é o número de pixels por célula, e  $\alpha$  é o número de células por bloco fornecido pela etapa de normalização.

Exemplificando, considerando como entrada, uma imagem representada na Figura 3.3(a), o resultado após todo o processo HOG é representado na Figura 3.3(b). Entretanto, vale ressaltar que a Figura 3.3(b) apresenta apenas uma forma de visualização do vetor HOG como imagem, mostrando as direções e magnitudes dos vetores, não sendo considerada para fins de processamento.

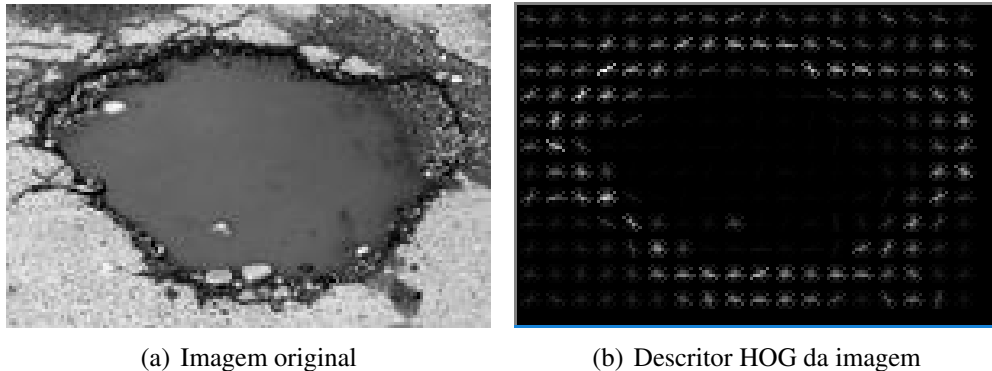


Figura 3.3 – Aplicação do HOG em uma imagem de buraco.

### 3.2 Aprendizado de Máquina

Algoritmos de aprendizado de máquina são cada vez mais ferramentas importantes em *softwares* inteligentes (CHAN et al., 2013). Tarefas como detecção e tradução entre idiomas, reconhecimento de objetos e sistemas de recomendação de produtos dependem dessas técnicas para descoberta de conhecimento, detecção de padrões e criação de regras de associação (BIERZYNSKI; ESCOBAR; EBERL, 2017).

Quando se trabalha com imagens, a aplicação de técnicas de processamento, assim como a técnica de HOG mostrada na seção anterior são úteis para representar imagens sob outras formas, ressaltando características que definem aspectos importantes de um objeto por exemplo. Contudo, para realização de tarefas de classificação existem outros algoritmos que podem ser utilizados, dependendo da tarefa ou natureza dos dados.

O aprendizado automático explora o estudo e construção de algoritmos que podem aprender de seus erros e fazer previsões sobre dados. Esses algoritmos de classificação são modelos matemáticos que descobrem relações (características em comum) entre objetos e dessa forma conseguem agrupá-los junto como pertencendo à uma mesma classe, e processamentos como os descritos na seção anterior podem auxiliar esses algoritmos a entender melhor tais objetos.

(Samuel, 1959) definiu aprendizado de máquina como o campo de estudo que permite aos computadores a habilidade de aprender sem serem explicitamente programados. Segundo (SHALEV-SHWARTZ; BEN-DAVID, 2014), o aprendizado pode ser visto com um processo de uso da experiência para o ganho de expertise. Quanto ao paradigma de aprendizado, esses algoritmos podem ser supervisionados ou não-supervisionados (ARTIFICIAL, 2011), dependendo da disponibilidade de dados e do problema. Esses conceitos são descritos a seguir. Nesse trabalho, apenas são utilizados dois algoritmos com paradigma de aprendizado supervisionado.

### 3.2.1 Aprendizado supervisionado

Para que um algoritmo crie relações entre objetos e separe eles em classes, é necessário que ele aprenda características sobre tais objetos. O processo de aprendizado dessas características é chamado de treinamento, ou seja, um algoritmo treina para aprender sobre determinados dados, e a partir deles generalizar sobre outro dados de mesma natureza.

Segundo (HURWITZ; KIRSCH, 2018), o processo de aprendizado supervisionado começa tipicamente com um conjunto estabelecido de dados e seus respectivos rótulos, os quais indicam o significado desses dados, sendo sua tarefa, encontrar padrões neles que podem ser aplicados em um processo analítico.

O processo de treinamento de um classificador é análogo ao aprendizado humano. Basicamente, para cada objeto apresentado ao algoritmo deve-se ter juntamente a classe à qual ele pertence. Dessa forma, o algoritmo consegue saber que características definem cada classe. Portanto, se a imagem de uma maçã é apresentada ao classificador, e é informado ao classificador que esse objeto é uma maçã, isso caracteriza-se como um treinamento supervisionado, pois o treinamento sabe previamente à qual classe pertence tal objeto. Em um treinamento não supervisionado, o classificador teria a tarefa de agrupar objetos semelhantes sem saber à qual classe ele pertence. As subseções 3.2.3 e 3.2.4 exploram dois classificadores diferentes usando aprendizado supervisionado.

### 3.2.2 Aprendizado não supervisionado

Diferentemente do paradigma anterior, o aprendizado não supervisionado não possui um rótulo explicitamente declarado para cada amostra da base de dados de treino. Nesse ponto dados de treino e teste também não são distintos pois todo o conhecimento que deve ser adquirido para realizar a tarefa de classificação deve advir dos próprios dados. Ou seja, não há

determinação prévia de que classe pertence determinada amostra de dados, e o algoritmo deve descobrir as relações entre os objetos para determinar características em comum e agrupá-los em conjuntos.

Segundo (BRAGA; CARVALHO; LUDERMIR, 1998), a partir do momento em que o algoritmo estabelece uma harmonia com as regularidades estatísticas da entrada de dados, desenvolve-se nele uma habilidade de formar representações internas para codificar características da entrada e criar novas classes ou grupos automaticamente.

### 3.2.3 Rede Neural Artificial (RNA)

Um dos modelos mais conhecidos em aprendizagem de máquina é a de uma Rede Neural Artificial (RNA). Uma RNA é uma arquitetura composta de elementos análogos aos que compõem o sistema cerebral humano, e portanto busca imitar o seu funcionamento. O modo de aprendizado de uma RNA supervisionada é baseado em um conjunto de pares amostra-classe por meio da qual se constrói seu aprendizado. O treinamento é feito de acordo com modelos matemáticos que, através de representações numéricas de dados, consegue separar objetos em suas respectivas classes.

Uma Rede Neural Artificial (RNA) é baseada em conceitos que incluem elementos que processam informações (neurônios), conexões ponderadas por pesos que transmitem informações entre neurônios (sinapses), camadas que organizam os neurônios em uma arquitetura específica e funções de ativação que determinam a resposta (saída) de cada neurônio (FAUSETT, 1994), (FEITOSA et al., 2018). Cada elemento da RNA é descrito nos itens à seguir.

- Neurônios: são a unidade básica de processamento de dados em uma RNA. São neles que os sinais de entrada de uma rede são processados, gerando uma saída. Possuem essencialmente uma função de ativação atrelada.
- Função de ativação: são responsáveis por converter um sinal de entrada em uma saída em um neurônio. Uma função de ativação é simplesmente uma função matemática. Funções Sigmoide, Tangente Hiperbólica (Tanh), ReLU, Escada e Softmax são alguns exemplos de funções de ativação (KOSKO, 1992). A escolha de uma função de ativação é algo importante em uma RNA pois define o comportamento do resultado computado por ela e também como funcionará o algoritmo de correção de erros. A Figura 3.4 mostra os gráficos apresentados pelas funções ReLU, Sigmoid e Tangente Hiperbólica respectivamente,

e suas derivadas.

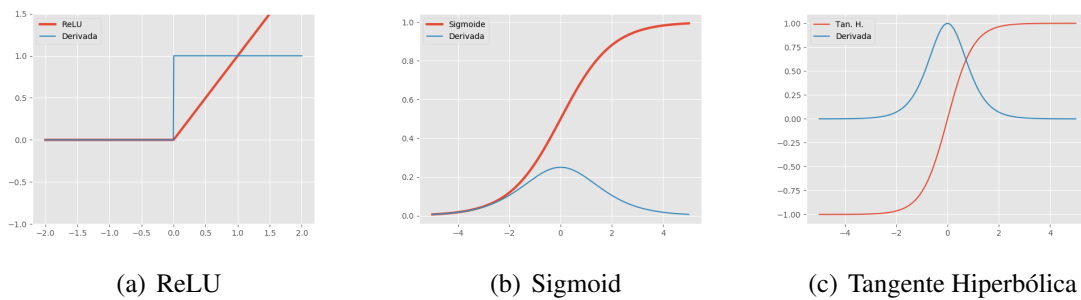


Figura 3.4 – Gráficos das funções de ativação e suas respectivas derivadas.

Fonte: <https://matheusfacure.github.io/2017/07/12/ativ-func/>.

- **Camadas:** são organizações lógicas de neurônios em uma RNA onde os mesmos são aninhados em um mesmo nível hierárquico da arquitetura. Normalmente uma RNA possui uma Camada de Entrada, uma Camada de Saída e uma camada intermediária conhecida como Camada Oculta, mas podem haver mais camadas intermediárias.
- **Pesos:** são onde está de fato o conhecimento de uma rede. Quando uma RNA é treinada para classificar dados entre duas ou mais classes, tudo que é aprendido pela RNA durante o treinamento tem de ser salvo em algum lugar. E é nos pesos que isso se encontra. Basicamente, pesos são como arestas que ligam um sinal de entrada até um neurônio ou neurônio à outro, e isso é o que define o quanto uma entrada influencia na ativação de um neurônio.
- **Algoritmos de treinamento:** para que uma RNA aprenda com os dados à ela apresentados, é necessário que correções sejam feitas nos valores armazenados nos pesos da rede. Algoritmos que realizam correções de erro na rede são utilizados para essa tarefa. Um dos mais conhecidos é o algoritmo de Descida Estocástica do Gradiente (SGD - *Stochastic Gradient Descent*).
- **Tamanho de Batch:** quando uma RNA é treinada, amostras de treinamento são apresentadas à rede, e ao final o algoritmo de treinamento é responsável por calcular o erro da rede em relação àquela amostra e corrigir seus pesos. Porém é possível apresentar mais de uma amostra à rede antes de realizar seu ajuste, à isso dá-se o nome de Tamanho de Batch. Existem essencialmente três estratégias:



- Modo *Batch*: todas as amostras são apresentadas à rede antes de cada ajuste. Nesse caso, o número de iterações é igual ao número de épocas de uma rede. Apresenta ser mais rápido que os demais pois ocorre apenas um ajuste por época, embora essa estratégia possa acarretar problemas no treinamento.
- Estocástico ou Incremental: é o modelo tradicional, onde se apresenta uma amostra por vez à rede e em seguida a mesma é ajustada. É mais lenta que o modelo em *Batch* pois para cada amostra ocorre um ajuste na rede, contudo o treinamento não é tão prejudicado.
- *Mini-batch*: é um dos modelos mais utilizados. Apresenta à rede um conjunto de amostras de tamanho maior que um e menor que o número total de dados, e ao final da iteração a rede é ajustada. Concentra o melhor de cada um dos modelos anteriores pois o treinamento é rápido e não é tão prejudicado.

Uma das arquiteturas mais conhecidas de RNA é a MLP (*Multi-Layer Perceptron*, ou *Perceptron* de Múltiplas Camadas). Essas são compostas por uma camada de entrada, onde entram os sinais sensoriais, uma ou mais camadas intermediárias chamadas camadas ocultas e uma última camada de saída. A Figura 3.5 é um exemplo de MLP. A qual possui duas entradas, 5 neurônios na camada oculta e apenas um neurônio na camada de saída, sendo todas as camadas interligadas por pesos. Essa rede pode, por exemplo, realizar classificações binárias.

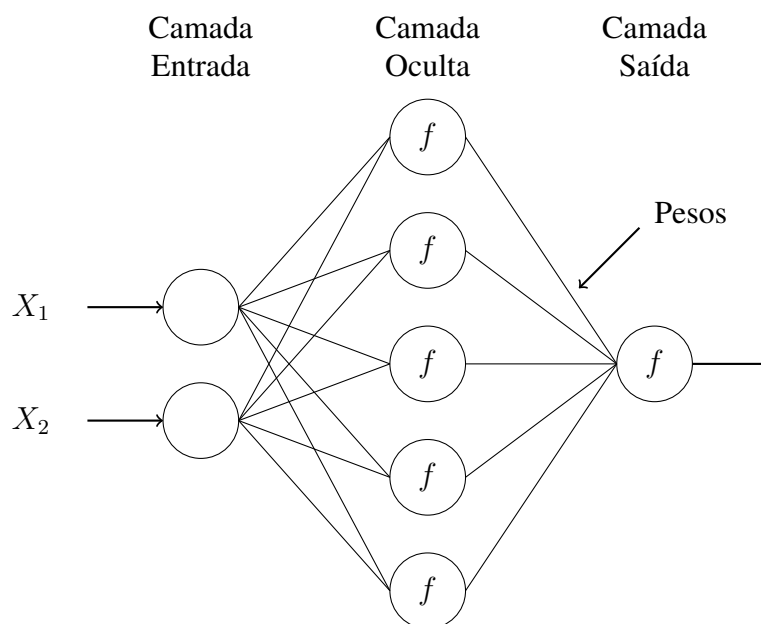


Figura 3.5 – Multi-Layer Perceptron com três camadas.

Segundo (HAYKIN, 1994), as redes do tipo MLP têm sido utilizadas com sucesso para a solução de vários problemas envolvendo altos graus de não-linearidade. Seu treinamento é do tipo supervisionado e utiliza um algoritmo muito popular chamado retro-propagação do erro (*error backpropagation*). Este algoritmo é baseado numa regra de aprendizagem que “corrige” o erro durante o treinamento.

### 3.2.4 Máquinas de Vetores de Suporte (SVM)

Máquinas de Vetores de Suporte (SVM - *Support Vector Machine*) são modelos matemáticos de aprendizado supervisionado projetados para maximizar a capacidade de generalização de um modelo e evitar sobreajuste dos modelos aos dados de treinamento (MADEO; PERES; LIMA, 2016). Elas foram originalmente propostas para resolver tarefas de classificação binária, e portanto podem ser usadas para realizar a detecção de buracos na classificação de imagens. Por exemplo, dada uma imagem de entrada, a tarefa de classificação decide se uma imagem é um buraco ou não.

Em teoria, a SVM realiza um mapeamento não linear nos vetores de entrada do seu espaço de entrada original para um espaço de característica multi-dimensional; e otimiza hiperplanos capazes de separar dados nesse espaço de características de alta dimensão. Ao resolver tarefas de classificação, a SVM considera um conjunto de treinamento com  $N$  amostras, definido por  $\{\mathbf{x}_i, y_i\}_{i=1}^N$  com  $\mathbf{x}_i \in \mathbb{R}^m$  e saída  $y_i \in \{-1, +1\}$ , sendo  $+1$  para imagens com buraco e  $-1$  para imagens sem buraco. O objetivo da SVM é encontrar um hiperplano ideal que separa os pontos de dados no espaço de características. Esse hiperplano é dado pela Equação 3.7:

$$f(x) = \langle \mathbf{w} \cdot \phi(\mathbf{x}_i) \rangle + b, \quad (3.7)$$

em que  $\mathbf{w}$  é o conjunto ideal de pesos,  $\phi(\mathbf{x}_i)$  representa um mapeamento não-linear em  $\mathbf{x}_i$ ,  $b$  é o viés ideal e  $\langle \cdot \rangle$  é um produto escalar (MADEO; LIMA; PERES, 2012; HAYKIN, 2010). A SVM otimiza o hiperplano de separação maximizando a distância entre o hiperplano e os pontos de dados mais próximos (a margem), o que corresponde a minimizar o conjunto de pesos  $\mathbf{w}$  na Equação 3.7. Com relação ao termo  $\phi(\mathbf{x}_i)$ , a SVM realiza um mapeamento não-linear implícito para um espaço de recursos de alta dimensão através da aplicação de funções *kernel*.

### 3.3 Descritores de Textura

Muitas imagens podem ser descritas segundo características inerentes à relação entre seus pixels. Variações, bordas, cores, formas e padrões de repetição são características que podem definir o que uma imagem está mostrando.

Descritores de textura são ferramentas que auxiliam na definição, por exemplo, de uma imagem. Existem inúmeros descritores de textura que servem aos mais diferentes propósitos. Alguns dos descritores mais conhecidos são os descritores de textura de Haralick (Haralick; Shanmugam; Dinstein, 1973).

#### 3.3.1 Descritores de Haralick

Em seu trabalho Haralick propôs 14 diferentes medidas de textura que são representados por descritores matemáticos baseados em matrizes de coocorrência de imagens. Uma matriz de coocorrência (também chamada de ocorrência simultânea) é uma tabulação de quantas combinações diferentes de valores de intensidade dos pixels (níveis de cinza) ocorrem em uma imagem, essa matriz tem tamanho  $t \times t$ , onde  $t$  é a quantidade de tons de cinza presentes na imagem. A quantidade de combinações é de no máximo 8 por pixel, ou seja, o pixel central pode ser comparado com cada um dos 8 vizinhos. Nesse sentido, uma matriz de correlação é sempre computada em relação à uma combinação. Basicamente, a matriz considera a relação entre dois pixels por vez, um chamado de pixel referência e o outro de pixel vizinho. Se os pixels de referência e vizinho têm os valores 1 e 5 respectivamente, o número que indica a quantidade de ocorrências iguais à essa na matriz é armazenado na linha 1 e coluna 5 da matriz de ocorrência. A Figura 3.6 mostra um exemplo do funcionamento desse processo com uma imagem contendo três níveis à esquerda e uma matriz de correlação consequentemente de tamanho  $3 \times 3$  à direita. Na matriz à esquerda existe apenas dois valores 0 consecutivos, portanto a matriz à direita tem a coluna 0 linha 0 com o valor 1.

1	0	2	0	2
2	1	0	2	2
0	2	0	1	0
2	0	1	1	2
1	2	0	0	1

→

	0	1	2
0	1	3	4
1	3	1	2
2	4	1	1

Figura 3.6 – Matriz de coocorrência.

Alguns dos principais descritores de Haralick são: Homogeneidade, Entropia, Energia, Contraste, Variância e Correlação. Esses descritores estão mostrados nas equações 3.8 à 3.12. O primeiro descritor se refere à intensidade de contraste entre o pixel e seus vizinhos. O segundo descritor é a entropia, basicamente representa a medida da aleatoriedade da intensidade de uma imagem. O terceiro descritor é a medida de energia que representa a uniformidade local dos níveis de cinza. O quarto descritor é o Contraste, essencialmente será igual a 0 para uma imagem constante e aumentará à medida que as intensidades de pixel em uma região da imagem se tornarem mais diferentes. E por fim a correlação mede o quão correlacionado um pixel está com sua vizinhança.

$$Inercia = \sum \sum (i - j)^2 p(i, j), \quad (3.8)$$

$$Entropia = \sum \sum p(i, j) \log(p(i, j)), \quad (3.9)$$

$$Energia = \sum \sum p^2(i, j), \quad (3.10)$$

$$Contraste = \sum \sum |(i - j)|^k p^n(i, j), \quad (3.11)$$

$$Correlacao = \frac{\sum_i \sum_j (i - \mu)(j - \mu) p(i, j)}{\sigma^2}, \quad (3.12)$$

Onde  $p(i, j)$  representa uma posição na matriz de correlação nas posições  $i$  e  $j$ .

## 4 METODOLOGIA PARA CLASSIFICAÇÃO DE IMAGENS DE BURACOS

Neste capítulo, é apresentado uma comparação do custo e da qualidade de diferentes configurações de uma arquitetura para classificação de imagens de asfalto quanto à existência ou não de buracos, baseada em técnicas de processamento de imagens e aprendizado de máquina. Inicialmente, este capítulo explica a arquitetura desenvolvida em mais detalhes, no contexto da proposta, posteriormente são definidos os conjuntos de dados utilizados bem como as métricas e modelos de avaliação por meio das quais a qualidade da arquitetura proposta é medida.

O principal foco neste capítulo é a variação de parâmetros das técnicas utilizadas na arquitetura proposta como forma de realizar um *tunning* da arquitetura com as melhores configurações possíveis. A arquitetura geral do sistema proposto para detecção de buracos baseada em imagens contém basicamente dois módulos, pré-processamento de imagem e classificação, conforme ilustrado na Figura 4.1.

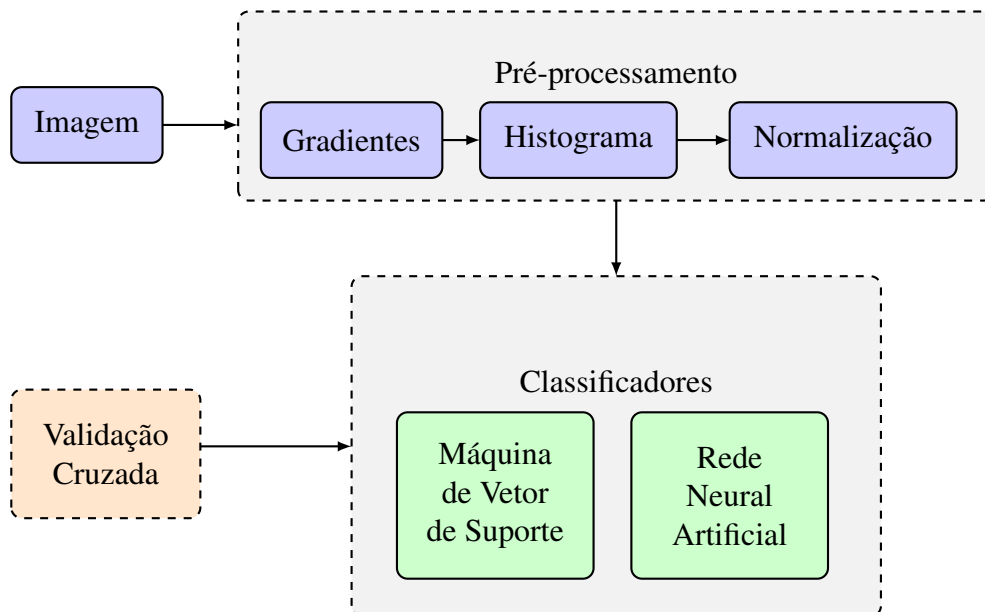


Figura 4.1 – Arquitetura para análise de viabilidade.

Durante a fase de pré-processamento, informações de pouca ou nenhuma relevância são removidas da imagem. As informações restantes são usadas para treinamento e teste (predição) pelos classificadores. O módulo de pré-processamento usa o Histograma de Gradientes Orientados (HOG) (DALAL; TRIGGS, 2005), que é um descritor de características de imagens. Os recursos capturados na imagem pelo descritor HOG são os histogramas (distribuição)

das orientações dos gradientes (orientados). Os gradientes em uma imagem são as derivadas nas coordenadas  $(x, y)$  e são úteis porque as magnitudes desses gradientes são altas em torno de regiões onde a variação entre as tonalidades de cor é grande. Como saída, o módulo de processamento gera um vetor de características que é a entrada para o módulo de classificação.

O módulo de classificação processa os dados de treinamento para classificar cada entrada (descriptor HOG) em sua classe correta, ou seja, contendo ou não um buraco. O módulo contém dois classificadores que processam a entrada separadamente e concorrentemente: uma Máquina de Vetor de Suporte (SVM) e uma Rede Neural Artificial (RNA), as quais foram treinadas e validadas utilizando o método de Validação Cruzada (*Cross Validation* - CV). Neste trabalho, o desempenho dos classificadores em conjunto com a técnica de HOG é analisado a fim de indicar qual o mais adequado para a tarefa de detecção de buracos, e com qual configuração. As próximas seções descrevem respectivamente o ambiente de experimentação, os conjuntos de dados utilizados para treinamento e teste, os métodos de avaliação, as parametrizações testadas e os resultados obtidos com diferentes *datasets*.

#### 4.1 Ambiente de experimentação

Os experimentos realizados para classificação das imagens de asfalto foram implementados usando linguagem Python versão 3.6 em conjunto com as bibliotecas *scikit-learn*, a qual implementa o algoritmo de classificação SVM utilizado neste trabalho, o algoritmo de Histograma dos Gradientes Orientados - HOG, bem como as funções para cálculo das métricas definidas. O algoritmo de RNA foi implementado usando a biblioteca *Keras* com a biblioteca *Tensorflow* como *backend*. Os experimentos foram realizados na plataforma Google Collaboratory contado com um hardware que possui uma GPU Tesla K80, com 2496 núcleos CUDA com 12GB de memória, e processadores Xeon @2.3Ghz com 12GB de memória RAM.

#### 4.2 Conjunto de dados

A arquitetura descrita na Figura 4.1 foi avaliada para efeitos de classificação utilizando dois *datasets* distintos:

- **dataset D1**: assim como em (KOCH; BRILAKIS, 2011), o conjunto de dados usado neste *dataset* tem tamanho modesto, sendo composto por 184 imagens, das quais 127 foram extraídas usando o mecanismo de busca da Google e 57 foram manualmente coletadas em

uma via de acesso secundário à Universidade Federal de Santa Maria, em agosto de 2018, sendo 92 amostras positivas (imagens de buracos no asfalto) e 92 amostras negativas (imagens de asfalto). Esse *dataset* possui grande variedade de buracos ao que se refere a formatos, tamanhos, condições climáticas (com ou sem água empossada por exemplo), com ou sem craquelados no entorno, também existem imagens com e sem sombreamento, além de sinais asfálticos presentes na imagem.

- **dataset D2:** contem 3200 imagens no total, sendo 1600 amostras positivas (contendo buracos) e 1600 amostras negativas (imagens de asfaltos) (ALZOUBI, 2018).

A ideia de avaliação em dois *datasets* diferentes é não só confirmar a veracidade da arquitetura proposta como também avaliar o comportamento da classificação sobre diferentes tipos de buracos e pavimentos de asfalto, dado que os dois *datasets* possuem origens geográficas distintas.

As Figuras 4.2 e 4.3 mostram exemplos de imagens extraídas de cada *dataset*. Pode-se notar que existe uma significativa variação entre texturas, elementos, e tipos de buracos.



Figura 4.2 – Extrato do primeiro *dataset* contendo 184 imagens. Sendo as três primeiras, amostras negativas e as três últimas amostras positivas.

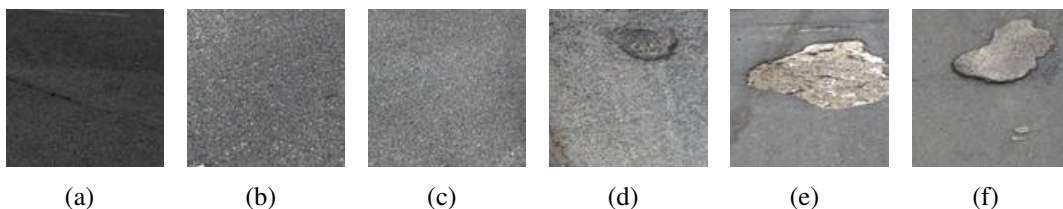


Figura 4.3 – Extrato do segundo *dataset* contendo 3200 imagens. Sendo as três primeiras, amostras negativas e as três últimas amostras positivas.

### 4.3 Métricas e modelos de avaliação

A arquitetura proposta na Figura 4.1 tem como etapa final a classificação de buracos implementado por meio de classificadores binários, cuja saída é a indicação de existência, ou

inexistência, de um buraco em uma imagem. As imagens de entrada para esse sistema (imagens de ruas e estradas, em cenários urbanos ou fora das cidades, em diferentes horários do dia) apresentam uma quantidade de informação que não é de interesse para o classificador (fundos variados, sobras, variação de iluminação, por exemplo). Neste cenário, o classificador pode apresentar quatro tipos de saída:

- Verdadeiro Negativo (True Negative ou TN): não há buraco na imagem e o diagnóstico é correto;
- Falso Negativo (False Negative ou FN): há um buraco na imagem mas ele não é detectado pelo classificador;
- Falso Positivo (False Positive ou FP): não há buraco na imagem, mas o classificador detecta um buraco;
- Verdadeiro Positivo (True Positive ou TP): existe um buraco na imagem e o diagnóstico é correto.

Esses possíveis resultados permitem avaliar a qualidade dos classificadores em termos das métricas indicadas nas Equações 4.1–4.4, em que  $N$  é a quantidade de imagens analisadas.

$$Acurácia = \frac{VP + VN}{N}, \quad (4.1)$$

$$Precisão = \frac{VP}{VP + FP}, \quad (4.2)$$

$$Revocação = \frac{VP}{VP + FN}, \quad (4.3)$$

$$F - score = 2 \cdot \frac{precisão \cdot revocação}{precisão + revocação} \quad (4.4)$$

A acurácia indica o desempenho geral do modelo. Dentre todas as classificações, ela indica em quantas o modelo classificou corretamente. A precisão indica, dentre todas as classificações na classe positiva (detecção de buracos), quantas estão corretas. A revocação (*recall*) indica, dentre todas as situações de classe positiva (existência de buracos), quantas foram detectadas pelo classificador. O  $f_{score}$  é a média harmônica entre precisão e revocação, com pesos iguais para os dois tipos de erro (falsos positivos e falsos negativos). Além disso, neste trabalho, os tempos de execução da técnica de HOG e dos classificadores também são fatores de qualidade dos algoritmos.



### 4.3.1 Curva ROC

Os experimentos também foram avaliados quanto a Curva ROC (*Receiver Operating Characteristic*) apresentada por cada configuração de classificação. A curva ROC mostra o quão bom o modelo criado pode distinguir amostras entre duas classes, e possui basicamente dois parâmetros, a taxa de verdadeiros positivos e a taxa de falsos positivos, os dois parâmetros estão descritos nas Equações 4.5 e 4.6.

$$\text{Taxa de Verdadeiros Positivos} = \frac{VP}{VP + FN}, \quad (4.5)$$

$$\text{Taxa de Falsos Positivos} = \frac{FP}{FP + VN}. \quad (4.6)$$

Basicamente um gráfico ROC traça uma curva entre a taxa de verdadeiros positivos e a taxa de falsos positivos, em diferentes *thresholds* de classificação. Ao final o que se deseja alcançar é uma curva que passe o mais próximo possível à esquerda e acima no gráfico, indicando uma baixa taxa de falsos positivos e uma alta taxa de verdadeiros positivos.

Como forma de simplificar o resultado observado em uma curva ROC pode-se utilizar o valor de AUC (*Area Under Curve*) que essencialmente é a área abaixo da curva, e seu valor pode variar de 0 até 1, respectivamente 0 e 100% de corretude na classificação.

### 4.3.2 Validação Cruzada

Quando são utilizados algoritmos ou arquiteturas de classificação, sempre são necessários dados para treinamento e classificação respectivamente, como forma de treinar essas arquiteturas e aplicar as métricas definidas na seção anterior aos resultados apresentados após classificação dos dados de teste. Para que esses pares de conjunto de dados treinamento-teste não sejam definidos empiricamente, utilizam-se técnicas que modelam esses conjuntos.

Neste trabalho, um esquema de validação cruzada é implementado. A técnica de validação cruzada divide o conjunto de dados em  $k$  grupos, chamados *k-folds*. A classificação é executada por  $k$  vezes. Em cada execução, um *fold*  $K_i$  diferente é usado para teste do classificador, enquanto os demais são usados para o treinamento, garantindo que todas as amostras serão testadas em algum momento. Ao final, é feita uma média dos resultados. Nos experimentos realizados, foram utilizados  $k = 5$  para o *dataset* D1 e  $k = 10$  para o *dataset* D2. Essa

diferença se deve ao fato de o *dataset* D1 ser muito menor que o segundo, e um grande número de grupos iria deixar os grupos de teste da validação cruzada muito pequenos.

#### 4.4 Parametrizações

Conforme objetivado no início deste capítulo, a parametrização das técnicas utilizadas na arquitetura proposta são parte fundamental na definição das melhores configurações para tarefa de classificação das imagens de buracos.

##### 4.4.1 Parametrização HOG

A técnica de HOG fornece os vetores de características das imagens que são entradas para os classificadores. Como visto no capítulo 3, existem três parâmetros que podem ser variados, sendo eles pixels por célula, células por bloco, e a quantidade de orientações amostradas em cada célula. A Tabela 4.1 apresenta quatro configurações  $C_i$  testadas com diferentes combinações de parâmetros de pixels por célula e de células por bloco. Para cada configuração  $C_i$  da tabela, foi variado também o parâmetro de orientações com valores 2, 5, 7, 9 e 11 no vetor de características HOG, implicando respectivamente em menos ou mais detalhes no vetor HOG. O objetivo desse ajuste é descobrir se a disponibilidade de mais dados melhora a qualidade dos classificadores, bem como qual é a configuração ideal da técnica de HOG para cada classificador. O resultado de cada configuração  $C_i$  testada para 2, 5, 7, 9 e 11 orientações gera 20 diferentes configurações possíveis de HOG.

Configuração	Células (pixels)	Blocos (células)
$C_1$	$8 \times 8$	$2 \times 2$
$C_2$	$8 \times 8$	$3 \times 3$
$C_3$	$16 \times 16$	$2 \times 2$
$C_4$	$16 \times 16$	$3 \times 3$

Tabela 4.1 – Configurações HOG testadas.

##### 4.4.2 Rede Neural Artificial

O primeiro classificador aplicado nesse trabalho é a Rede Neural Artificial (RNA). Para aplicação nos experimentos discutidos neste trabalho, foi utilizada a arquitetura de Perceptron de Múltiplas Camadas (MLP - Multilayer Perceptron). É uma rede com arquitetura *feed-forward* de múltiplas camadas que é treinada de modo supervisionado. A RNA aqui considerada

é composta por uma camada de entrada, uma camada oculta e uma camada de saída com um neurônio único, o qual classifica a imagem de entrada como sendo um buraco ou não. As configurações usadas na RNA estão descritas na Tabela 4.2.

<b>Rede Neural Artificial</b>		
	Camada Oculta	Camada de Saída
Número de neurônios	1/2	1
Função de ativação	ReLu	Sigmoid
Tamanho de <i>batch</i>	16	
Algoritmo	Stochastic Gradient Descent (SGD)	

Tabela 4.2 – Configurações da rede neural.

A entrada da RNA, assim como da SVM, é o vetor do algoritmo HOG. No entanto, a RNA usa as funções de ativação ReLu e Sigmoid na arquitetura e recebe valores de entrada entre 0 e 1. Como o vetor HOG também contém valores fora desse intervalo, a entrada de dados deve ser dimensionada. A Equação 4.7 representa a normalização dos dados antes do treinamento da RNA, em que  $V$  é o vetor no índice  $i$ :

$$\forall v \in V : v = v/mx, \quad (4.7)$$

onde  $mx$  representa o valor máximo no vetor HOG. Com esta equação, todos os valores em cada posição do vetor são redimensionados para variar entre 0 e 1.

#### 4.4.3 SVM

No classificador Support Vector Machine (SVM), os parâmetros são  $C$  e  $\gamma$  (*gamma*).  $C$  descreve a função de custo da margem flexível, que controla a influência de cada vetor de suporte individual. Este processo de controle envolve a penalidade de erro por estabilidade.  $\gamma$  é o parâmetro livre da função de *kernel*. Neste trabalho, a Função de Base Radial (Radial Basis Function ou RBF) é aplicada como *kernel*. À respeito dos parâmetros da SVM, os experimentos utilizaram  $\gamma = 0.01$  e  $C = 1$ .

## 4.5 Resultados *dataset D1*

### 4.5.1 Resultados

Inicialmente, o foco dos experimentos foi no ajuste dos parâmetros da técnica de HOG que é entrada para os classificadores RNA e SVM, buscando pelas melhores configurações da técnica em cada classificador e medindo seu desempenho em cada métrica. A Figura 4.4 apresenta a média das métricas consideradas para os dois classificadores com todas as configurações HOG  $C_i$  descritas na Tabela 4.1 agrupadas pelas métricas definidas na seção 4.2 para cada orientação  $O_n$ .

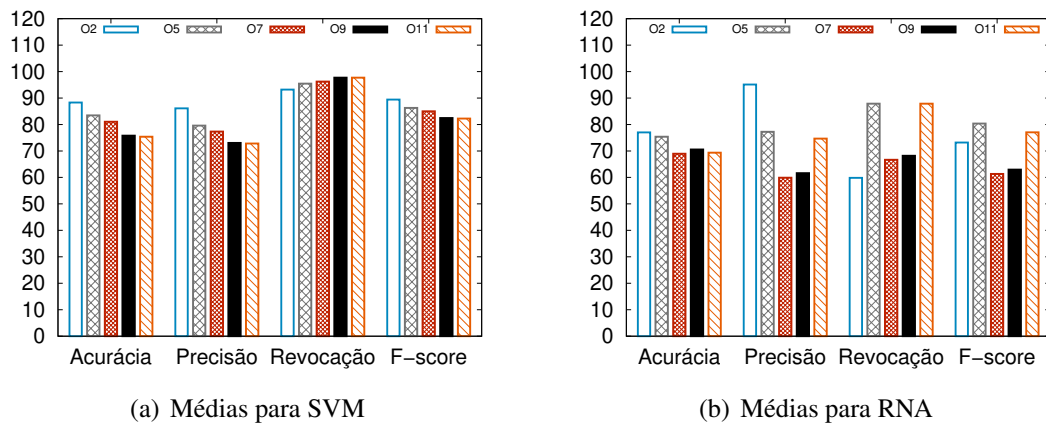


Figura 4.4 – Médias para todas as configurações  $C_n$ .

Os resultados apresentados no gráfico indicam que para ambos os classificadores, RNA e SVM, as melhores configurações médias de HOG para vetores de entrada utilizam 2 e 5 orientações por célula. Para SVM, esses valores de orientações HOG são as melhores opções em acurácia, precisão e f-score, sendo menores apenas na métrica de revocação, o que indica que ocorreram menor taxa de acerto nas amostras positivas do que nas demais orientações. É possível perceber que no geral a SVM tende a piorar seus resultados com mais informação de entrada, ou seja, mais orientações amostradas por célula no HOG. Isso indica que o algoritmo não precisa de muita informação para atingir bons resultados.

No caso da RNA, as configurações médias indicam novamente que 2 e 5 orientações são as melhores para configurações de entrada nesse classificador. No caso da acurácia e precisão, o valor de orientação 2 se mostra a melhor escolha, e no caso de revocação e f-score, 5 orientações apresentam melhores resultados. Esses valores tanto para RNA e SVM são a base para os testes

das demais configurações de pixels por célula e de células por blocos definidas na tabela 4.1. Sendo assim, os próximos resultados utilizam 2 e 5 orientações para teste das  $C_i$  configurações. A Figura 4.5 retrata os resultados do classificador RNA nesses casos.

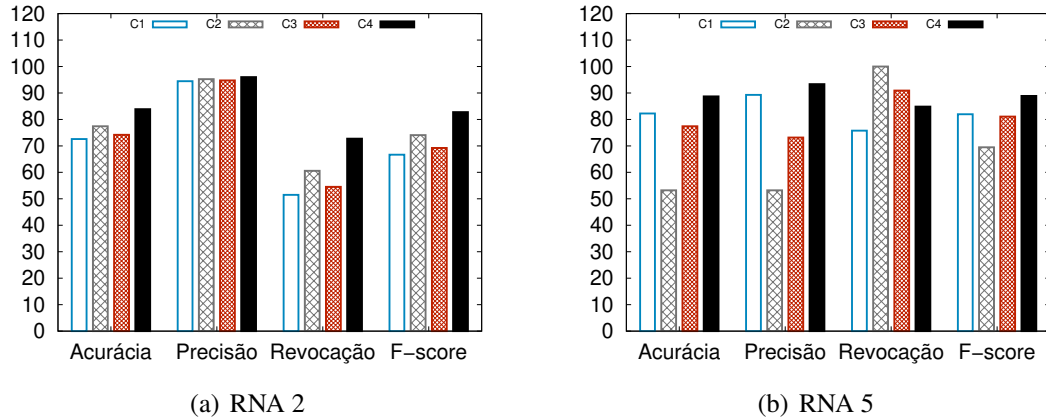


Figura 4.5 – Resultados obtidos para a RNA usando HOG de 2 e 5 orientações.

No primeiro gráfico, relacionado à RNA utilizando configurações HOG com 2 orientações segue uma tendência padronizada entre  $C_1 - C_2$  e  $C_3$  e  $C_4$ , onde o primeiro desses é menor que o segundo. Dadas as características das configurações  $C_i$ , pode-se dizer que a utilização de blocos de  $3 \times 3$  no processo de normalização HOG é a melhor escolha, e juntamente com o uso de células de tamanho  $16 \times 16$  constitui  $C_4$  como a melhor escolha, quando da utilização de 2 orientações como vetor de entrada da RNA. Destaque para a precisão que alcançou um percentual médio de 95%, indicando que de todas as classificações positivas feitas, quase todas estavam corretas.

Quanto à configuração utilizando 5 orientações, os resultados se apresentam melhores conseguindo atingir uma acurácia próxima à 90%. Destaque para a configuração  $C_4$  novamente pois apresentou em geral os melhores resultados em quase todas as métricas.

A Figura 4.6 apresenta os resultados do classificador SVM, também com 2 e 5 orientações.

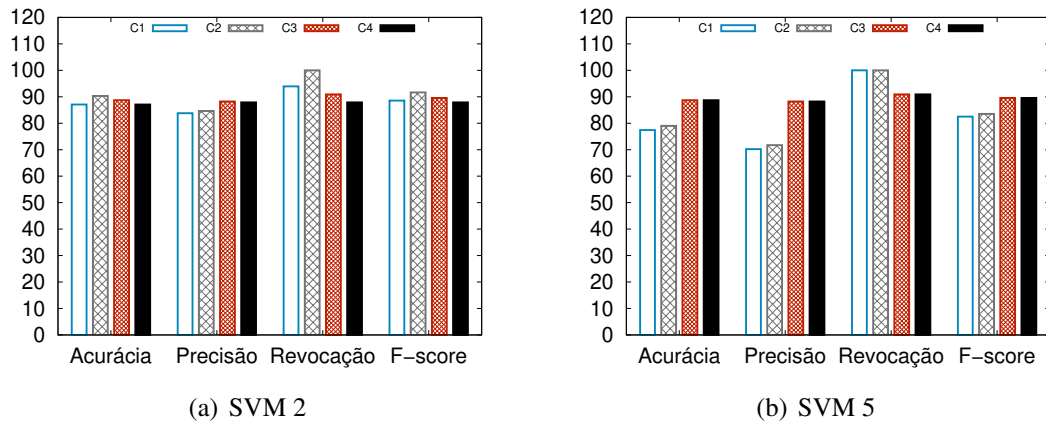


Figura 4.6 – Resultados obtidos para o SVM usando HOG de 2 e 5 orientações.

No caso da SVM com entrada HOG de 2 orientações, os resultados para cada métrica estabilizaram com valores aproximados, com pequena diferença para a configuração  $C_2$  chegando à 90% de acurácia e 100% de revocação. Em contrapartida, no gráfico dos resultados utilizando 5 orientações de HOG como entrada, as configurações  $C_3$  e  $C_4$  se mostram melhores que as configurações  $C_1$  e  $C_2$ . Pode-se observar aqui uma relação inversa entre quantidade de orientações por célula, e o tamanho em pixels dessa célula, pois enquanto no primeiro gráfico a melhor configuração era  $C_2$  com 2 orientações, ou seja, menos informações por células, porém contendo mais células. As configurações  $C_3$  e  $C_4$  têm menos células e mais orientações por célula atingindo resultados muito semelhantes à  $C_2$ .

O resultado do classificador RNA com  $C_4$  como melhor configuração HOG de entrada indica que a RNA precisa de menos dados que a SVM para predição de eficiente. Se, por exemplo, uma imagem de buraco com  $48 \times 48$  pixels é categorizada por ambos classificadores, a melhor configuração HOG para SVM ( $C_2$  com 2 orientações) resultaria em um vetor de entrada de tamanho 72, enquanto para a RNA a melhor configuração HOG de entrada ( $C_4$  com 5 orientações) gera uma entrada com vetor de tamanho 45 e mantém as taxas das métricas avaliadas, bem próximas as obtidas pela SVM.

#### 4.5.2 Tempo de processamento HOG

Finalmente, o tempo de processamento foi avaliado para verificar a viabilidade de aplicar os classificadores em cenários com demanda em tempo real. A Figura 4.7 mostra o tempo de processamento da técnica de HOG. Através dos experimentos, foi verificado que o tempo de processamento é maior nas configurações  $C_1$  e  $C_2$  e menor nas configurações  $C_3$  e  $C_4$ . Isso

porque o tamanho das células em  $C_3$  e  $C_4$  é maior, e portanto apresentam menos células por imagens para serem processadas. E, conseqüentemente, como a RNA apresenta os melhores resultados usando a configuração  $C_4$  do HOG, também se torna a opção mais rápida no processo de classificação.

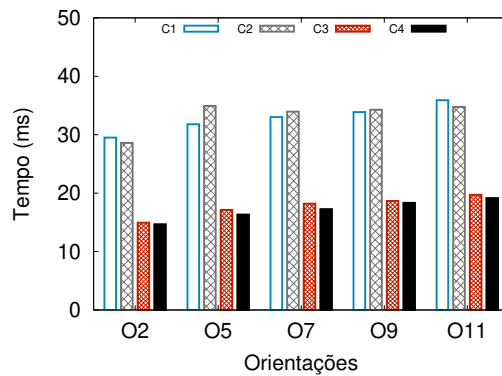


Figura 4.7 – Tempos de execução para o HOG.

#### 4.5.3 Tempo de treinamento

A Figura 4.8 retrata o tempo de treinamento para a SVM e a RNA. O classificador SVM é eficiente em termos de tempo para treinamento para classificar imagens de buracos representados por vetores HOG.

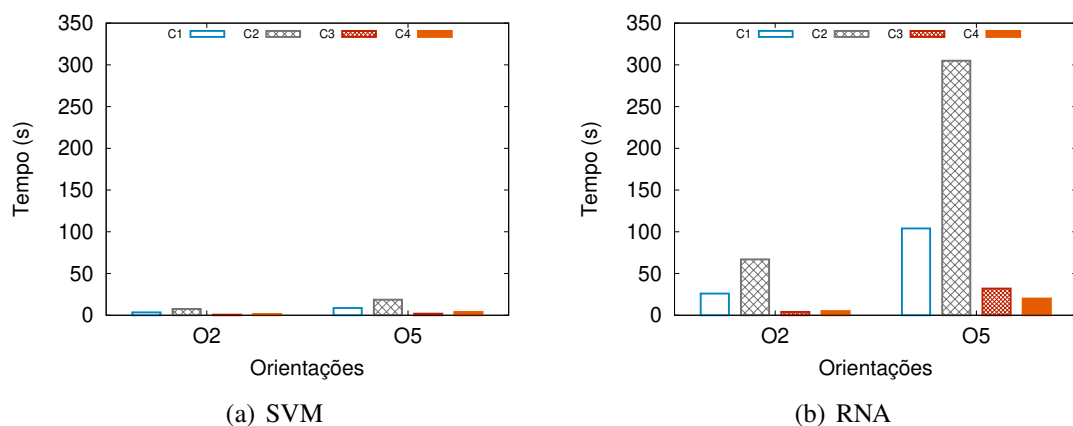


Figura 4.8 – Tempos de treinamento para a RNA e o SVM.

Entretanto, quando existe um problema de classificação de imagens, apenas o tempo de predição da técnica escolhida é levado em consideração. Os resultados indicam que a SVM possibilita treinamento em tempo real, mesmo aumentando continuamente o conjunto de dados

para fazer previsões mais precisas. De fato, o tempo de treinamento na RNA é muito maior quando comparado ao tempo de treinamento da SVM.

#### 4.6 Resultados *dataset D2*

Assim como na avaliação dos resultados do primeiro *dataset* (seção 4.5), neste trabalho também foram realizados os ajustes de parâmetros em relação as melhores métricas buscando pela melhores configurações.

A Figura 4.9 mostra o desempenho médio de todas as configurações HOG agrupadas pelas métricas definidas na seção 4.2 para cada orientação  $O_n$ . O gráfico indica que, assim como nos resultados obtidos com o primeiro *dataset*, os vetores HOG com 2 e 5 orientações são os que apresentam as melhores configurações e portanto melhores opções de entrada para os classificadores. Nos resultados em relação à SVM, os vetores com 2 orientações apresentam predominância dos melhores resultados em todas as métricas. Na RNA, os vetores com 5 orientações são os que apresentam os melhores resultados.

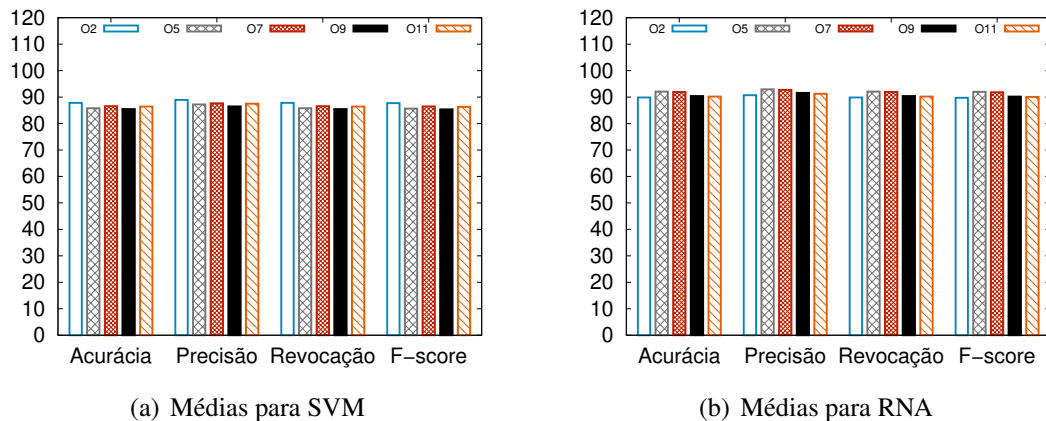


Figura 4.9 – Médias para todas as configurações e validação cruzada para k-fold=10.

A Figura 4.10 apresenta os resultados obtidos com o classificador SVM usando 2 e 5 orientações em relação às configurações  $C_n$ .



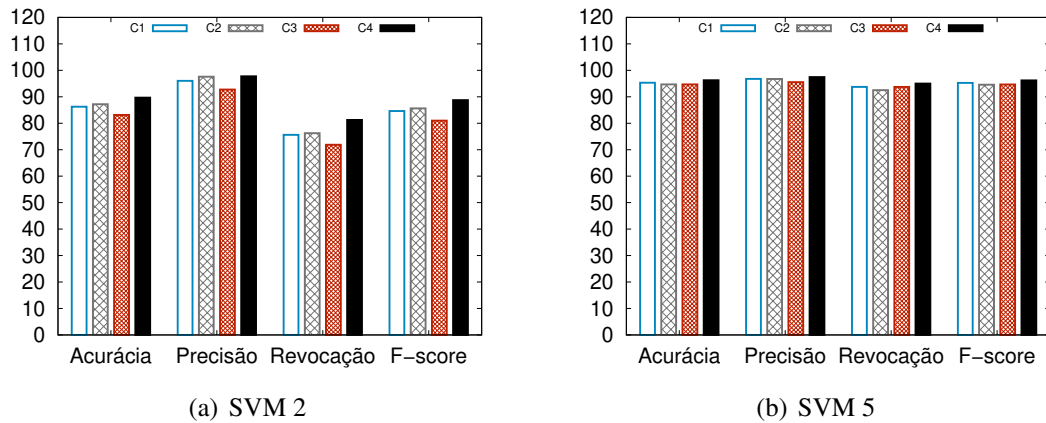


Figura 4.10 – Resultados obtidos para o SVM usando HOG de 2 e 5 orientações.

Primeiramente, sobre o gráfico da Figura 4.10(a), o qual representa os resultados da SVM com vetores HOG de entrada de 2 orientações, é possível perceber que os resultados se mantêm entre 80 e 100% das métricas definidas utilizando a configuração  $C_4$ . Em contrapartida, avaliando os resultados do gráfico da Figura 4.10(b) utilizando como entrada o HOG com 5 orientações, ocorre uma estabilização entre as configurações  $C_i$  em todas as métricas, com apenas pequenas diferenças. É importante observar que, no caso do segundo *dataset*, o aumento no número de orientações amostradas é um fator impactante na capacidade de aprendizado da SVM, podendo esse ser um fato causado pelo tamanho do segundo *dataset*.

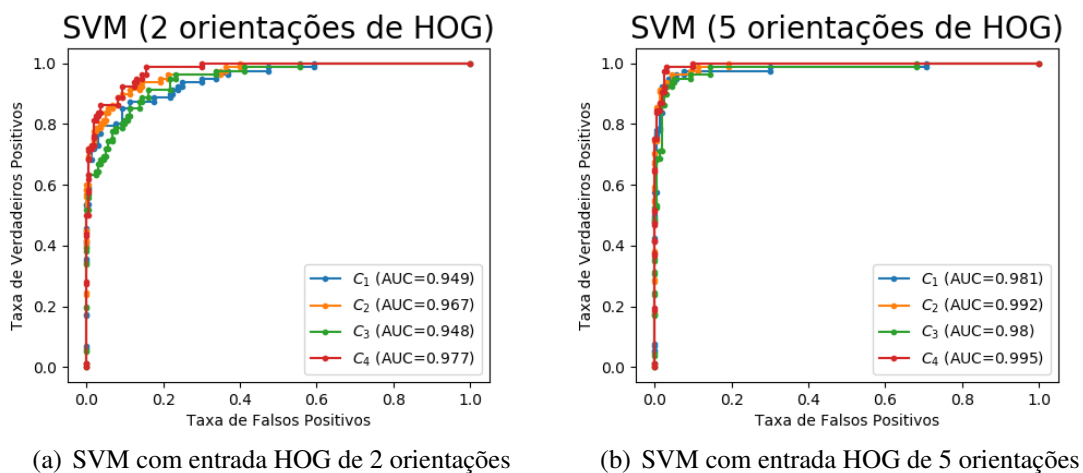


Figura 4.11 – Curvas ROC SVM.

As curvas ROC apresentadas nos gráficos da Figura 4.11 indicam de forma mais clara que a melhor configuração HOG é a  $C_4$  quando da utilização de 5 orientações por célula. A curva relativa à essa configuração mostra exatamente o que se espera como resultado conforme

descrito na subseção 4.3.1, ou seja, uma curva mais perto da parte superior esquerda, indicando uma baixa taxa de falsos positivos e uma alta taxa de verdadeiros positivos.

Na Figura 4.12, estão apresentados os resultados referentes à RNA usando respectivamente entradas HOG com 2 e 5 orientações. No primeiro gráfico, assim como na avaliação do primeiro *dataset*, as métricas indicam que as configurações  $C_3$  e  $C_4$  são as melhores, atingindo aproximadamente 90% em acurácia e f-score, 80% em revocação e 100% em precisão. Quando da análise do segundo gráfico pode-se perceber o mesmo fenômeno ocorrido com a SVM. O número de orientações amostradas aumenta e a capacidade da rede em classificar exemplos corretamente também aumenta, principalmente quando do uso da configuração  $C_4$ , chegando à atingir 95% em todas as métricas.

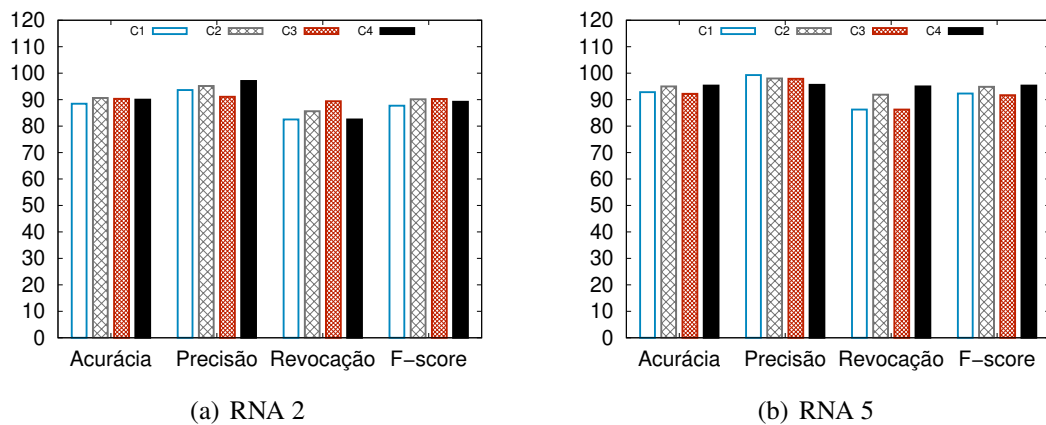


Figura 4.12 – Resultados obtidos para a RNA usando HOG de 2 e 5 orientações.

As curvas ROC, assim como a SVM, também chegam aos mesmos resultados concluindo que configuração  $C_4$  com 5 orientações é a melhor opção de configuração para a técnica de HOG, como forma de criar um vetor descritor de características de entrada para a RNA.

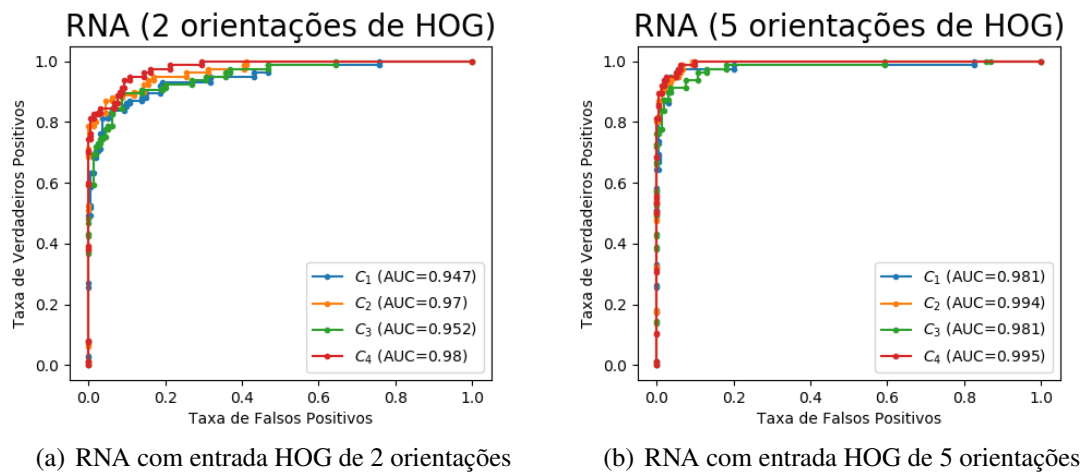


Figura 4.13 – Curvas ROC RNA.

#### 4.7 Validação entre os *datasets*

Nesta seção, os melhores modelos treinados para cada classificador são avaliados com dados não conhecidos previamente. Ou seja, cada classificador treinado com um *dataset* é testado com o outro, ambos configurados respectivamente de acordo com suas melhores configurações HOG definidas pelos resultados das subseções 4.5 e 4.6 anteriores. O objetivo desta validação é avaliar a capacidade do classificador em generalizar sobre o problema de buracos em estradas em relação à heterogeneidade das características de diferentes buracos. São dois casos de teste T1 e T2.

- T1: classificador treinado com o *dataset* D2 (3200 imagens), testado com o *dataset* D1 (184 imagens), e
- T2: classificador treinado com o *dataset* D1 (184 imagens), testado com o *dataset* D2 (3200 imagens).

A Figura 4.14 apresenta os resultados da RNA e SVM com a avaliação entre os *datasets*. Pode-se observar que, principalmente no caso da RNA (Figura. 4.14(a)), a rede treinada com o *dataset* D1 tem dificuldades de reconhecer os dados do *dataset* D2 e, portanto, apresenta valores menores para as métricas avaliadas. Nesse caso, as métricas da RNA alcançam em média um score médio de 84% em relação à T1, enquanto T2 sofre grande variação entre as métricas alcançando média de 66% entre as métricas. Contudo, vale ressaltar que a precisão alcançada em T2 é de 99% e portanto a rede treinada com o *dataset* D1, não reconheceu praticamente

nenhum falso positivo no *dataset* D2, o que indica que o treinamento foi bem sucedido, mesmo com poucas amostras. A revocação apresenta um valor baixo pois o classificador acabou por separar muitas amostras positivas como negativas, gerando muitos falsos negativos, provavelmente justificado pela baixa quantidade de amostras no treinamento.

A Figura 4.14(b) apresenta os resultados obtidos pelo classificador SVM avaliado entre os *datasets*. Em contraste aos resultados apresentados pela RNA na Figura 4.14(a), a SVM alcança melhores taxas principalmente em acurácia e precisão tanto em T1 quanto em T2, chegando a scores médios de 85% em T1 e de 83% em T2. Isso mostra que, no caso de T2, mesmo com poucas amostras o classificador conseguiu prever corretamente a grande maioria das 3200 imagens do *dataset* maior, ou seja, generalizando melhor do que a RNA. Vale ressaltar ainda que tanto na RNA quanto na SVM a métrica de precisão alcançada em T2 possui melhores resultados que em T1, portanto de todos os verdadeiros positivos preditos, a grande maioria estava correta, isso mostra que a diversidade de amostras positivas no *dataset* D1 foi o suficiente para prever e acertar a grande maioria das amostras positivas no *dataset* D2.

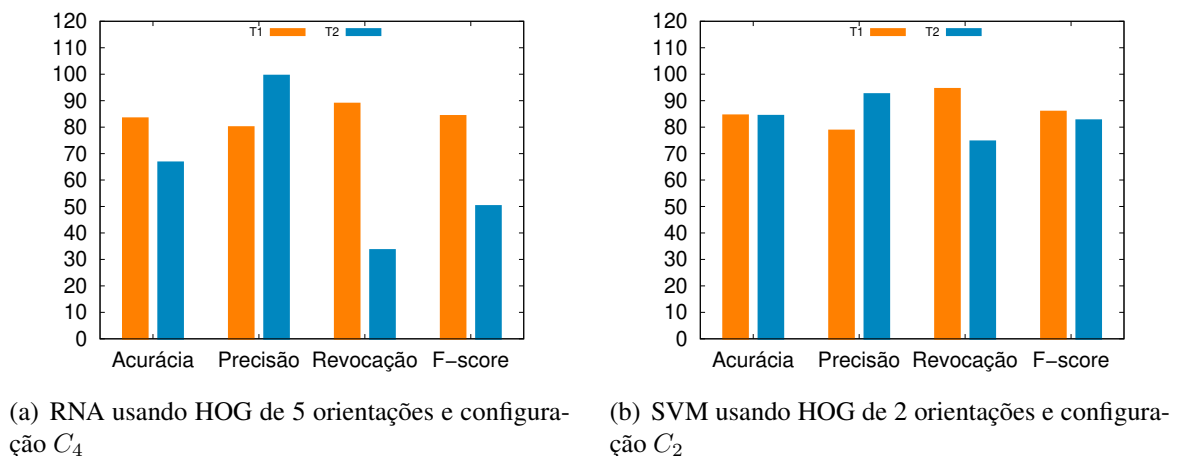


Figura 4.14 – Resultados de classificação entre os *datasets*.

A Figura 4.15 apresenta as curvas ROC para os casos de teste T1 e T2. Em relação à RNA os valores de de AUC da curva ROC indicam que o cenário T1 foi o melhor, ou seja, a rede conseguiu prever melhor o segundo *dataset* sendo treinada com o primeiro, o que mostra uma carência da RNA por quantidade de dados. Enquanto para a SVM os dois cenários foram equivalentes segundo a curva ROC, o que mostra a capacidade de aprendizado da SVM e principalmente de generalização sobre os dados.

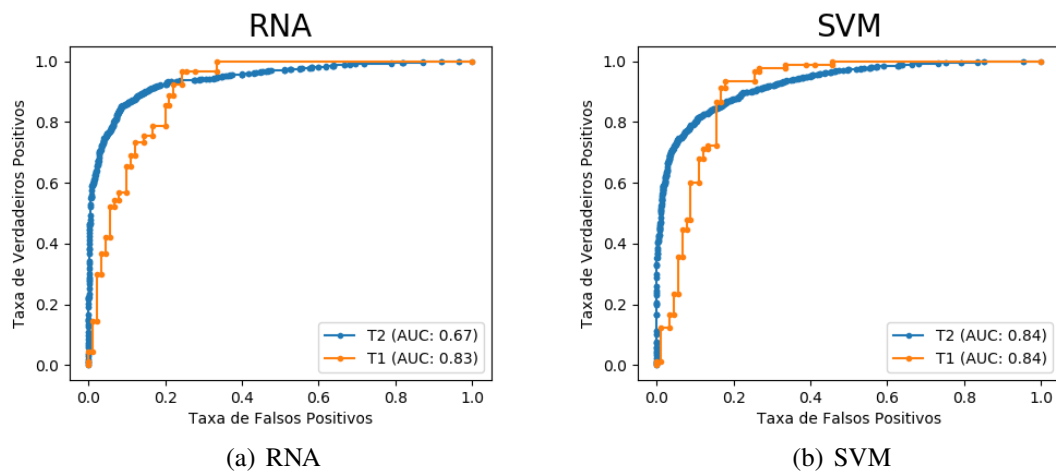


Figura 4.15 – Curvas ROC dos resultados entre os datasets D1 e D2 para RNA e SVM.

Este capítulo apresentou uma arquitetura para classificação de imagens de buracos em pavimentos de asfalto utilizando a técnica de Histograma de Gradientes Orientados juntamente com dois classificadores, RNA e SVM, realizando um *tunning* das parametrizações da arquitetura. Nesse sentido, a conclusão parcial desse trabalho indica que a SVM apresenta um comportamento mais robusto em relação à RNA, conseguindo generalizar de forma melhor, e atingindo resultados mais satisfatórios, sendo  $C_2$  com 2 orientações a melhor configuração HOG de entrada para esse algoritmo.

O próximo capítulo apresenta uma metodologia proposta para detectar buracos (objeto alvo) em imagens, ou seja, realizar a localização e determinação de onde o buraco está na imagem. Para isso, a arquitetura descrita nesse capítulo é utilizada em conjunto com descritores de texturas para determinar regiões de imagens propensas a conterem buracos, e então realizar a classificação.

## 5 METODOLOGIA PARA DETECÇÃO DE BURACOS EM IMAGENS DE VÍDEO

A tarefa de classificação é importante quando a necessidade de separação de objetos em diferentes classes se faz necessária. Contudo, a determinação (localização) de onde esses objetos estão em uma imagem de vídeo por exemplo é uma tarefa diferente e também um desafio. Neste capítulo, é apresentado como é realizado o processo de localização de um buraco em imagens de vídeo, bem como a sua marcação utilizando descritores de textura aplicados em um *grid* de sub-imagens.

A arquitetura de detecção aqui proposta é dividida em duas partes. A primeira parte consiste na divisão da imagem em um *grid*, a aplicação de um dos descritores de textura de Haralick em cada uma das células desse *grid*, aplicação de *threshold* e posterior agrupamento de células que apresentem altos valores pelo descritor. Como os descritores geram um único valor por imagem, um *grid* de células da imagem original foi criado, de forma a gerar um conjunto de valores para representar diferentes regiões da imagem.

A segunda parte usa a técnica de Histograma de Gradientes Orientados (descrita nas seções 3.1.1 e 3.1.2) em conjunto com os classificadores testados e parametrizados no capítulo anterior para determinar se existe ou não um buraco na imagem, conforme apresenta o esquema da Figura 5.1. O procedimento completo é explicado no texto que segue.

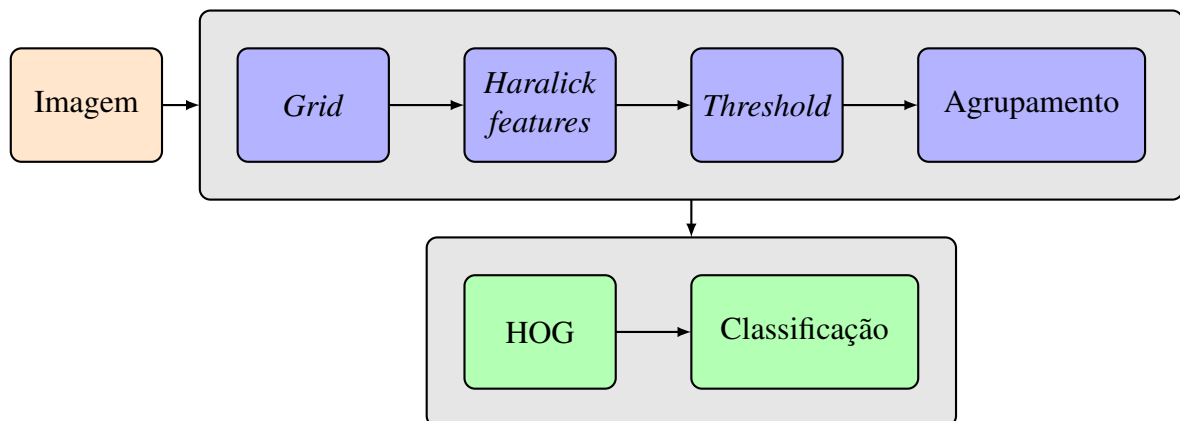


Figura 5.1 – Arquitetura proposta para a detecção de buracos em vídeos utilizando descritores de textura de Haralick em um *grid* de imagem, aplicação da técnica de HOG como descritor da imagem e classificação.

### 5.1 Divisão da imagem em *grid*

O processamento de um vídeo é basicamente feito pelo processamento de cada um de seus *frames* separadamente. Para cada *frame*, o processo consiste em dividi-lo em  $M \times N$  células, onde cada célula possui o mesmo tamanho de  $m_1 \times n_1$  pixels, de acordo com a Figura 5.2.

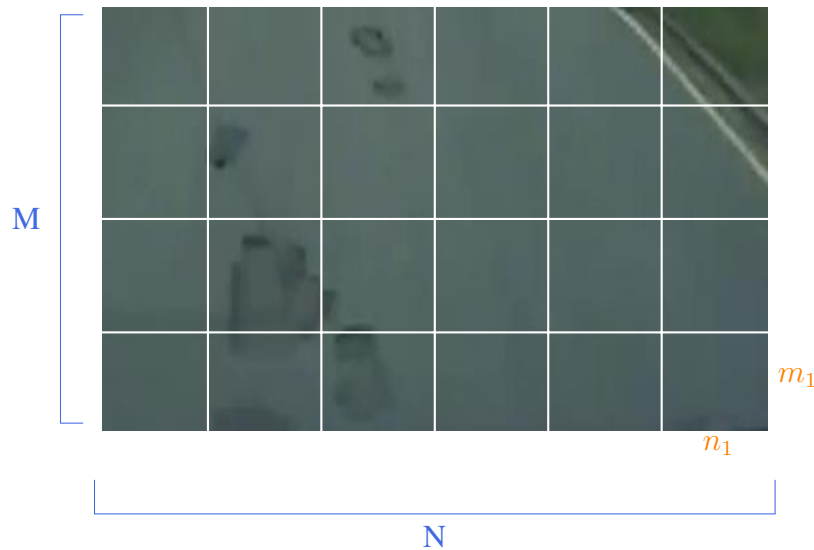


Figura 5.2 – Grid de imagem com  $M \times N$  células, onde cada célula possui o tamanho de  $m_1 \times n_1$  pixels.

Para cada célula do *grid*, um valor de característica é calculado por um dos descritores de Haralick. Cada valor calculado auxilia na determinação de regiões homogêneas e heterogêneas, ou seja, com baixa ou alta variação de textura. Dado que um buraco apresenta uma quebra na uniformidade da textura do asfalto, o descritor calculado para cada célula pode discriminar quais regiões são propensas a conter um buraco, uma parte de um buraco, ou algum outro objeto. O descritor Haralick é explicado em mais detalhes na seção 5.2.

### 5.2 Descritor de textura de Haralick

Os descritores de textura de Haralick (Haralick; Shanmugam; Dinstein, 1973) são usados como forma de obter informações acerca de relações contidas na textura de uma imagem. Com os descritores de uma imagem podem ser calculadas características como Correlação, Entropia, Energia, Inercia entre outras.

Neste trabalho, foi utilizado o descritor de Correlação. O descritor de correlação apre-

sentam valores de picos de variância quando à mudança abrupta de uma textura, com significativa diferenciação entre regiões homogêneas e heterogêneas. Uma mudança abrupta de uma textura pode estar relacionada com a ocorrência de um buraco, parte de um buraco ou outro objeto na pista. O valor de característica calculado tem o propósito então de definir quais células do *grid* tem alta variação na textura.

Na Figura 5.3, pode ser observado que os sinais gerados pelos descritores Correlação (a) e Entropia (b) existem picos no sinal, mostrando diferença entre regiões mais homogêneas e menos homogêneas. Em contraste, Energia (c) e Inercia (d) apresentam valores embaralhados, sem clara distinção de variação na textura. Contudo, a Entropia pode resultar em valores negativos, o que não é útil ao problema, sendo então a Correlação o descritor mais adequado a ser utilizado.

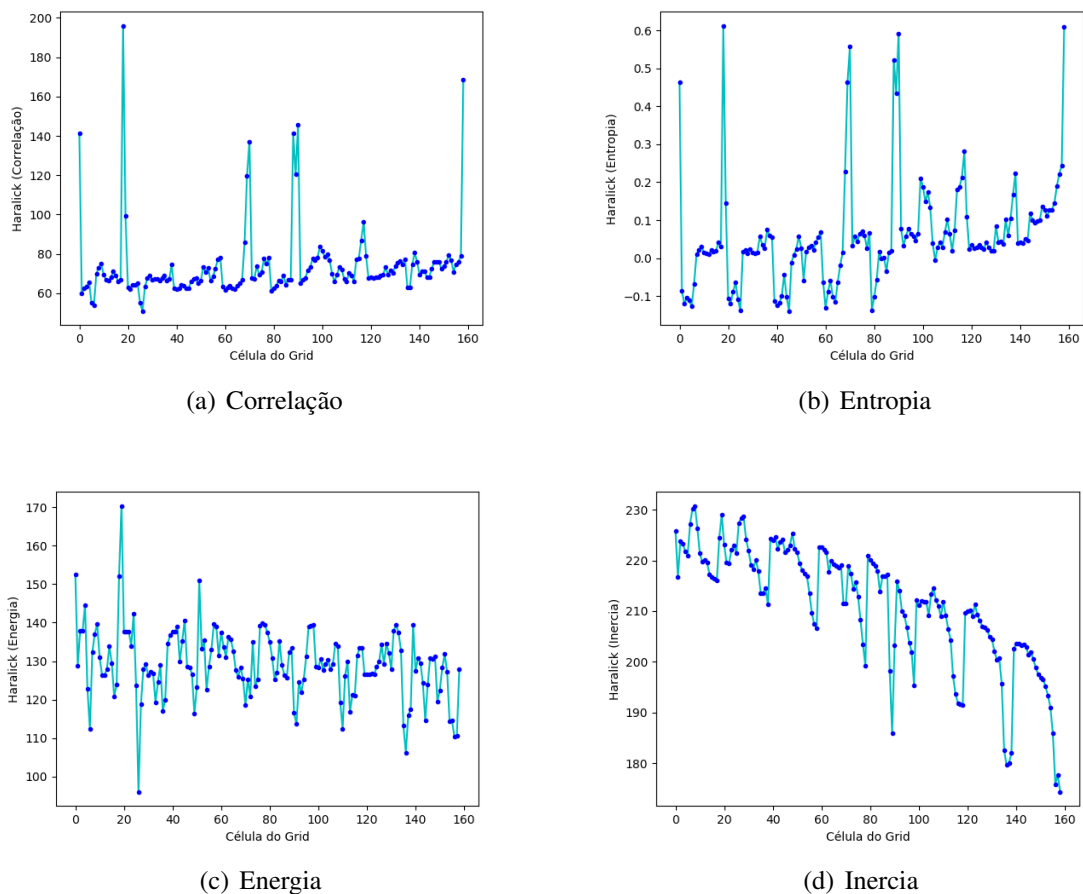


Figura 5.3 – Gráficos gerados pelos diferentes descritores de Haralick em relação à imagem da Figura 5.2 usando um *grid* de  $20 \times 8$ , com células de tamanho  $25 \times 25$ .

Esta mudança permite justamente detectar um novo padrão em uma imagem que pode, possivelmente, ser associado com a existência de um buraco. Matematicamente, este descritor



é representado pela Equação 5.1, onde  $\sigma$  e  $\mu$  representam respectivamente o desvio padrão e a média dos elementos computados.

$$Correlacao = \frac{\sum_i \sum_j (i - \mu)(j - \mu)p(i, j)}{\sigma^2} \quad (5.1)$$

Como mostrado na subseção 5.1, cada *frame* é dividido em células, e para cada célula é calculado um valor com o descritor de Correlação de Haralick. Assim, se um *frame* de  $500 \times 200$  pixels dividido em células de  $50 \times 50$  pixels haverá consequentemente 40 ( $10 \times 4$ ) células. Cada célula tem seu valor do descritor de Correlação calculado, e assim um sinal bidimensional é montado como resultado, onde cada ponto representa o valor de Correlação de uma célula do *frame*.

A Figura 5.4 apresenta um exemplo de sinal gerado pelos descritores de células de uma imagem. Ela contém um sinal bidimensional gerado com pontos referentes a cada célula do *grid*. Cada sub-imagem abaixo do gráfico da figura representa uma célula de um mesmo *frame*, e cada ponto do gráfico representa o valor de correlação de cada uma dessas células. No gráfico, as variações da textura são representados por picos, e normalmente pontos de baixa altura representam pouca ou nenhuma variação.

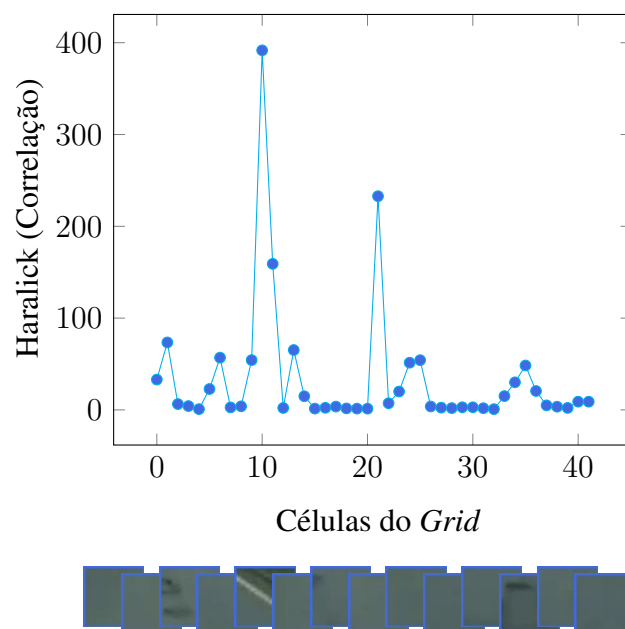


Figura 5.4 – Sinal gerado pelo valor do descritor de correlação de Haralick para cada célula do *grid*.

No caso do problema de detecção de buracos, valores baixos no gráfico indicam que provavelmente existe uniformidade no asfalto naquela parte, e portanto não existe nenhum in-

dicativo de haver um buraco, não sendo necessário nenhum processamento adicional. Em contrapartida valores altos de grande variação em relação aos demais podem indicar a existência de um buraco, um pedaço de um buraco ou demais elementos na pista.

Algo interessante na técnica é que, como é gerado um sinal por *frame*, pode-se ter independência de fatores como diferenças de luminosidade, diferentes texturas de asfalto, entre outros, pois a variação de um pico, por exemplo, pode ser sempre analisado em relação aos demais pontos do mesmo *frame*. Pode-se, por exemplo, tomar a uniformidade dos pontos com baixa amplitude como definição de valores que representam áreas onde há apenas asfalto sem buracos ou demais elementos.

Esses valores facilitam o processamento ao passo que são determinadas características de áreas onde há pouca variação e, portanto, não existe necessidade de executar a etapa de pré-processamento nem a parte de classificação, diminuindo o processamento.

O descritor de textura tem duas funções principais na arquitetura de detecção. A primeira é a determinação das células que apresentam e que não apresentam variância. A segunda função é que essa técnica permite que posteriormente, células que apresentam variância e que possuem algum tipo de vizinhança entre si possam ser agrupadas, ou seja, a determinação de haver ou não um buraco é feita pela classificação de um conjunto de células que forma uma sub-imagem dentro do *frame*. Assim, é possível marcar a ocorrência do buraco especificamente dentro do *frame*. Antes de realizar esse agrupamento, *thresholds* são aplicados nesse sinal gerado como forma de realizar a determinação do que é variação e o que não é.

### 5.3 Thresholds

Depois da fase de divisão e aplicação do descritor de Haralick, ocorre um processo de *thresholding* sobre o sinal gerado, para que possam ser indicadas quais células do *grid* não precisam passar para as fases posteriores de pré-processamento e classificação. Aqui serão testadas três técnicas de *threshold*, que posteriormente serão comparadas e avaliadas. Sendo elas Média (M), Mediana (MD) e desvio padrão (DP), definidas pelas Equações 5.2, 5.3 e 5.4 a seguir:

$$M = \frac{X_0 + X_1 + \dots + X_n}{n} \quad (5.2)$$

$$MD = \begin{cases} L(\frac{n}{2} + 1), & \text{sendo } n \text{ impar} \\ L(\frac{n}{2}) + L(\frac{n}{2} + 1), & \text{sendo } n \text{ par} \end{cases} \quad (5.3)$$

$$DP = \sqrt{\frac{\sum_{i=1}^n (x_i - M_a)^2}{n}} \quad (5.4)$$

Nas fórmulas, cada elemento  $X_i$  é um valor de Haralick computado para cada célula do *grid*, enquanto  $n$  é a quantidade de células do *grid* e  $L$  representa a lista contendo cada valor  $X_i$ . Por último,  $M_a$  é a média calculada dos elementos computados no desvio padrão.

Nesta fase, é indicado se existe uma variação ou não em cada célula do *grid*. Essa variação é dada pelo valor de Haralick computado para cada célula em relação ao *threshold* aplicado ao sinal geral. Quando uma célula satisfaz os requisitos do *threshold* aplicado, ela deve ser passada ao próximo estágio, o qual irá agrupar células próximas.

#### 5.4 Agrupamento

Células filtradas para classificação devem primeiro ser agrupadas como meio de criar objetos completos. Ou seja, células vizinhas são mescladas de forma a formar um único objeto maior e completo, para que seja possível o classificador realizar a predição correta.

Basicamente, o algoritmo une todas células que estão em vizinhança e determinar um retângulo no *frame* original iniciando na coordenada  $x$  e  $y$  da célula mais acima à esquerda, e terminando nas coordenadas  $x_1$  e  $y_1$  da célula mais abaixo à direita. As novas regiões de interesse formadas por essas coordenadas no *frame* são usadas para o cálculo do HOG e posteriormente pela fase de classificação. A regra que define o agrupamento entre células é de que, em uma célula onde existe variação, só poderá ser anexada a outra célula ou grupo de células se houver vizinhança horizontal ou vertical direta com outra célula também com variação. Células em diagonal não são agrupadas. Um exemplo do uso das regras descritas para realizar o agrupamento pode ser visualizado nas imagens da Figura 5.5.

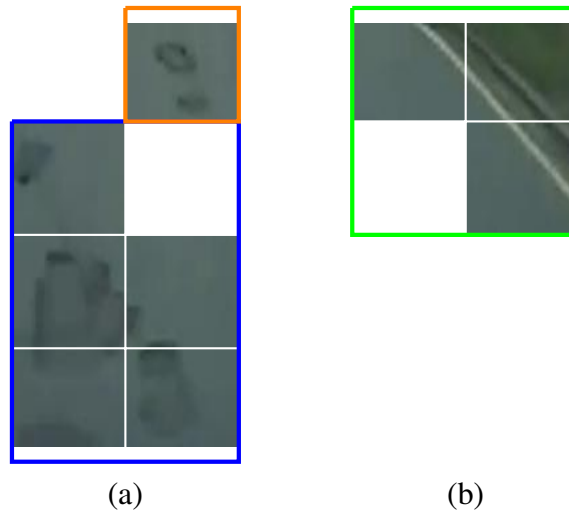


Figura 5.5 – Agrupamento de células do *grid* que apresentam valor do descritor acima do *threshold* calculado.

Na Figura 5.5, o agrupamento azul de células possui cinco células com variação acima do *threshold*, e uma célula não selecionada que se torna parte do conjunto como complemento. A célula em laranja da Figura 5.5(a) não configura nenhum agrupamento com as células do grupo azul pois está em diagonal com a célula mais próxima, contudo ainda sim representa um objeto selecionado. Por terceiro, o grupo verde apresenta o mesmo caso do grupo azul, onde uma célula não selecionada é incluída no agrupamento como complemento. Porém, a Figura 5.5(b) não representa um buraco. Nesses casos a tarefa final fica a cargo do pré-processamento com HOG e classificação desses objetos completos, para determinar o que é e o que não é buraco.

## 5.5 HOG e Classificação

Após a conclusão do agrupamento das células, cada sub-imagem é processada pela técnica de HOG e então passada como entrada ao classificador que define se a imagem é ou não um buraco. Caso positivo, uma *bounding-box* com as dimensões do agrupamento é colocada na imagem original para determinar a ocorrência do buraco. Uma *bounding-box* nada mais é que uma caixa na imagem que contem o buraco classificado positivamente. Ressalta-se ainda que as melhores configurações das técnicas testadas nos experimentos da Seção 4 são utilizadas aqui para a predição.

O processo completo pode ser descrito conforme o Algoritmo 1. Primeiramente, a imagem é dividida em um *grid* (linha 2), retornando as células (subimagens) e as coordenadas de

cada célula, em seguida o valor de correlação de Haralick é calculado para cada célula (linha 3). O *threshold* de Média é então calculado para todos os valores de correlação, retornando apenas as células que estão acima do *threshold* (linha 4) e, posteriormente, as células restantes são agrupadas de modo a formar objetos completos (buracos ou não) (linha 5). Em seguida, todas as imagens resultantes do agrupamento passam pela fase de HOG ao final pela classificação (linhas 8 e 9), e caso positivo a região determinada é marcada como contendo um buraco.

---

**Algoritmo 1:** Algoritmo para detecção de buracos em imagens de asfalto.

---

**Entrada:** Imagem

```

1 início
2   grid, coords ← split(Imagem)
3   cell_descriptor_values ← correlacao(grid)
4   cells_indexes ← threshold(cell_descriptor_value)
5   sub_images, coords ← group(grid,cells_indexes)
6   i ← 0
7   repita
8     subimagem ← HOG(sub_images[i])
9     isPothole ← classificador.predict(subimagem)
10    if isPothole then
11      draw_box(Imagem,coords[i])
12    i ← i + 1
13  até i < sub_images.quantidade;
14 fim
```

---

## 5.6 Experimentos

Como forma de avaliação da metodologia proposta para a detecção de buracos, foram utilizados *frames* de um vídeo extraído da internet no endereço <https://www.youtube.com/watch?v=8be0v041R5E>. Cada *frame* do vídeo foi processado e avaliado para determinação da viabilidade do uso do método proposto neste trabalho.

### 5.6.1 Escolha do *threshold*

Como descrito na seção 5.3, três *thresholds* foram testados para realizar o corte do sinal. Entretanto, apenas o *threshold* da Média foi escolhido, pois apresenta melhor distinção entre regiões de baixa e grande variação nas texturas. As Figuras 5.6, 5.7 e 5.8 mostram os resultados da utilização dos *thresholds* de Média, Mediana e Desvio Padrão (DP) respectivamente. É possível observar nessas figuras que a utilização da Mediana seleciona muitas células que possuem baixa variação, enquanto que Média e Desvio Padrão se mostram propensas a uma

melhor seleção dessas regiões.

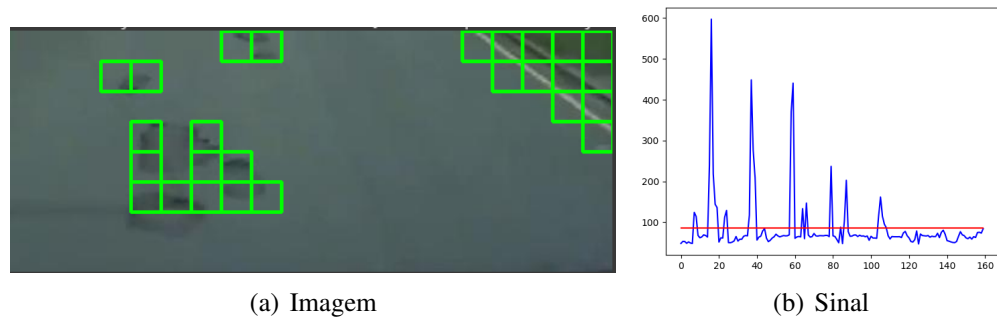


Figura 5.6 – *Threshold Média.*

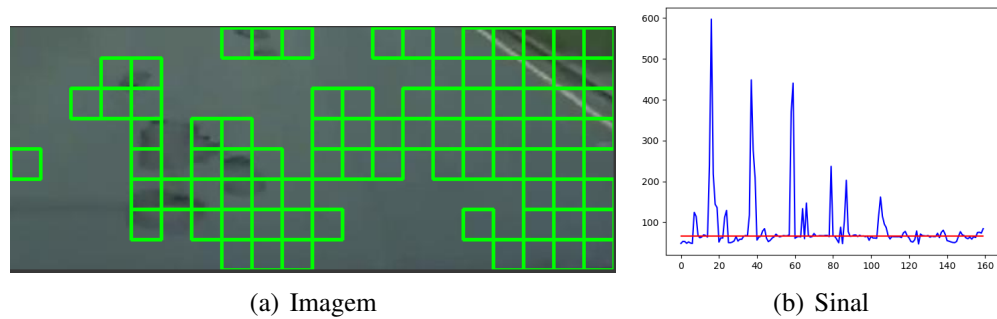


Figura 5.7 – *Threshold Mediana.*

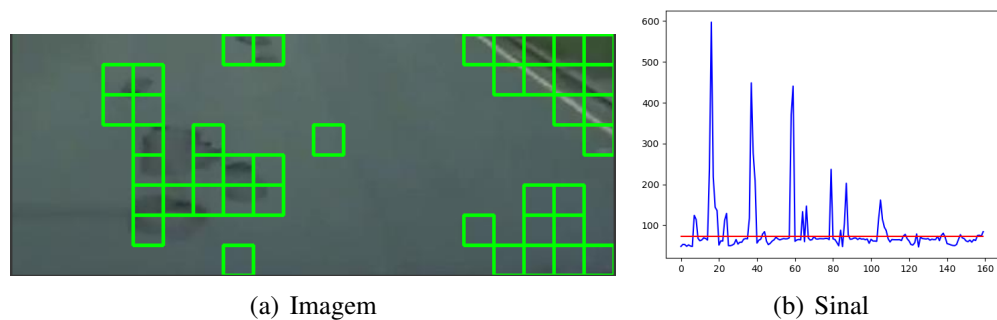


Figura 5.8 – *Threshold Desvio Padrão.*

O que pode ser notado, como diferença entre Média e Desvio Padrão, é que a Média ainda apresenta melhor separação entre diferentes buracos localizados na imagem, enquanto o Desvio Padrão unifica diferentes buracos além de incorrer em alguns casos no mesmo problema da Mediana.

## 5.7 Avaliação do modelo de detecção

No caso de um modelo para detecção de objetos em imagens, não é mais de interesse a definição de amostras negativas, ou seja, não é necessário dizer o que é asfalto, calçada, faixas, entre outros elementos. O foco de interesse é mostrar onde está localizado o objeto alvo - os buracos - ou seja, amostras positivas. Neste caso, a avaliação se torna um pouco mais qualitativa e menos quantitativa, com duplo objetivo: a arquitetura deve reconhecer determinado buraco e as dimensões desse buraco devem ser determinadas na imagem com precisão.

### 5.7.1 IoU - *Intersection of Union*

A interseção da união (IoU - *Intersection of Union*) é uma métrica de avaliação usada para medir a acurácia de um detector de objetos. Essa métrica é utilizada em muitos desafios de detecção de objetos como por exemplo, o popular *PASCAL VOC challenge*. Essa métrica é independente de qualquer técnica ou metodologia utilizada na detecção. De modo mais formal, para aplicar a IoU na avaliação de qualquer detector de objetos é necessário que se tenha:

- *ground-truth bounding boxes* para cada imagem avaliada: isso serve como um gabarito alvo aproximado de onde o objeto realmente está na imagem. Para cada objeto na imagem deve haver as coordenadas de sua *bouding box* no formato  $(P_1, P_2)$  onde  $P_1(x_1, y_1)$  e  $P_2(x_2, y_2)$  representam os pontos de início e fim respectivamente da *bounding-box*, ou no formato  $(x, y, w, h)$  onde  $x$  e  $y$  são as coordenadas do ponto central dela e  $w$  e  $h$  são as larguras e alturas. De forma simples, as *ground-truth bounding boxes* são as localizações verdadeiras de objetos na imagem, delimitadas por um quadrado.
- *bounding boxes*: preditas pelo detector, as quais, serão comparadas com as *ground-truth bounding boxes* afim de verificar o quanto de acurácia existe na predição.

A Figura 5.9 exemplifica o funcionamento da avaliação. O buraco está verdadeiramente contida dentro da caixa de borda verde, chamado de fundo de verdade (gabarito), que é a *ground-truth bounding box*. A caixa com borda vermelha é a predição (caixa predileta) *bounding box*. Portanto, o melhor caso ocorre quando o quadrado vermelho é o mais aproximado possível da caixa verde.

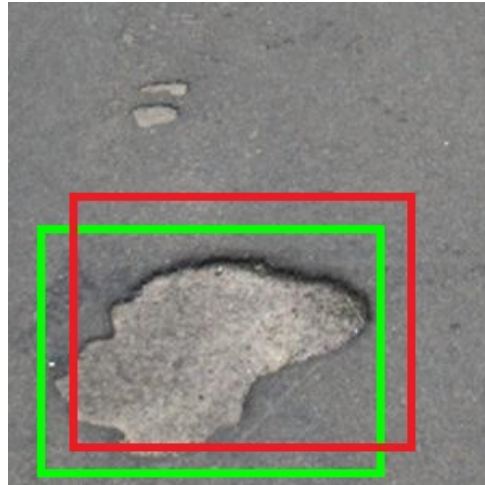


Figura 5.9 – Exemplo de localização e marcação de um buraco em imagem. Em verde, a caixa de *ground-truth*, e em vermelho a caixa predita.

O cálculo dessa métrica mostra que conforme se tem uma caixa mais próxima à outra, a interseção e a união dos dois conjuntos tem seus valores aproximados fazendo com que a divisão da IoU se torne mais próxima de 1, e quando se afastam a interseção se torna cada vez menor e a união maior, gerando valores de IoU mais próximos de 0. A Figura 5.10 mostra como funciona o cálculo do valor para a técnica de IoU.

$$\text{IoU} = \frac{\text{Interseção}}{\text{União}}$$

Figura 5.10 – Formula para cálculo da interseção da união.

## 5.8 Resultados

Como resultado final da arquitetura proposta neste capítulo para detecção de buracos em imagens, tem-se a indicação de onde existe buraco na imagem por meio da marcação do mesmo por uma *bounding-box*. A Figura 5.11 mostra exemplos de imagens resultantes do



processamento.

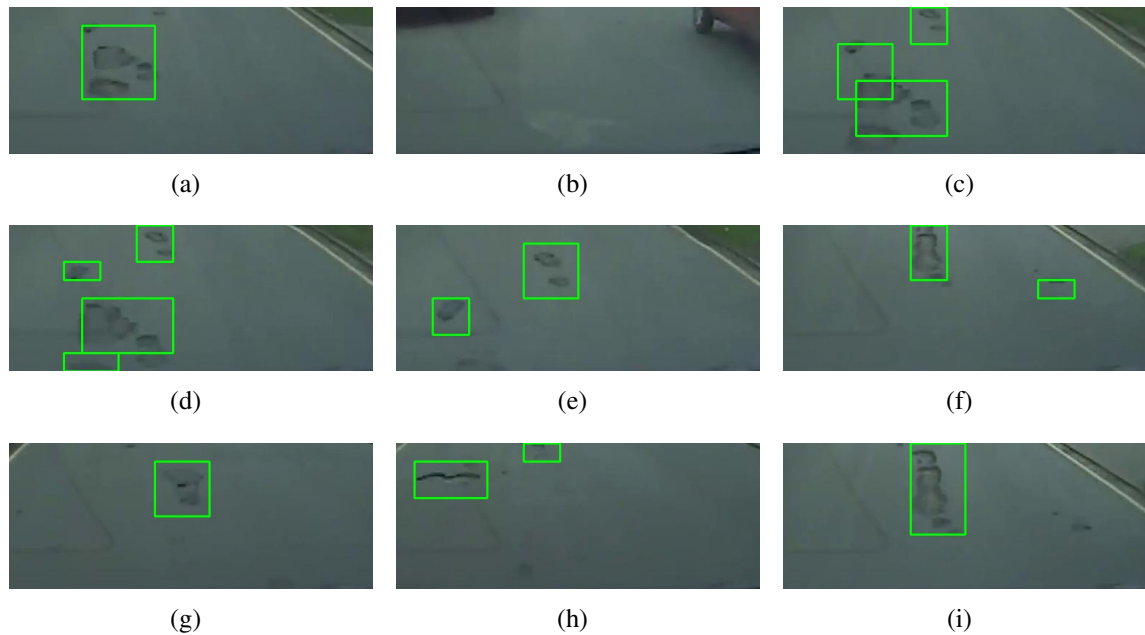


Figura 5.11 – Amostras de *frames* resultantes do processamento com a marcação em verde das *bounding-boxes* dos buracos

Como forma de avaliar essas marcações feitas pela arquitetura de detecção quanto à sua qualidade, o experimento realizado foi avaliado segundo sua acurácia, precisão e pela métrica da Interseção da União (IoU). As métricas de revocação e *f-score* não foram avaliadas pois dependem de falsos negativos e verdadeiros negativos, que não se aplicam nesse caso, pois busca-se apenas resultados positivos. A Figura 5.12 mostra os resultados obtidos em relação à essas métricas, e as métricas de acurácia e precisão para os seguintes casos.

- RNA25: RNA utilizando a configuração HOG  $C_4$  com 5 orientações, e células de *grid* com  $25 \times 25$ ,
- RNA50: RNA utilizando a configuração HOG  $C_4$  com 5 orientações, e células de *grid* com  $50 \times 50$ ,
- SVM25: SVM utilizando a configuração HOG  $C_2$  com 2 orientações, e células de *grid* com  $25 \times 25$ ,
- SVM50: SVM utilizando a configuração HOG  $C_2$  com 2 orientações, e células de *grid* com  $50 \times 50$ .

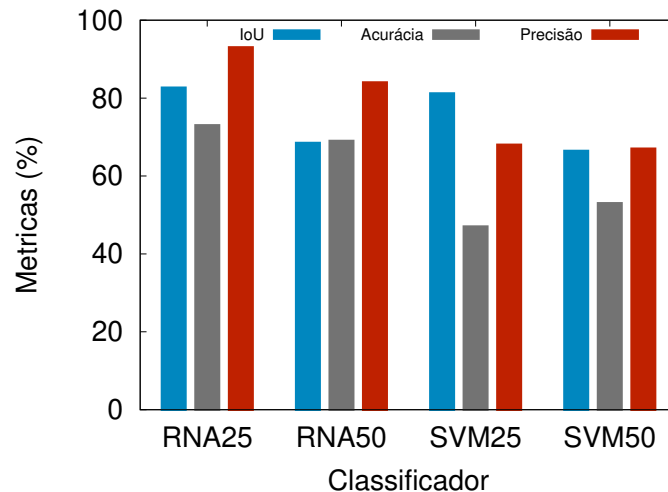


Figura 5.12 – IoU, acuracia e precisao referente ao processamento do vídeo.

Os resultados apresentados mostram que a utilização da RNA com sua melhor configuração HOG ( $C_4$  com 5 orientações) e *grid* de células  $25 \times 25$  é a melhor opção para a arquitetura, sendo superior às demais, com destaque para a precisão, atingindo 93%, acurácia com 73%, e a métrica de IoU com 83%.

Vale ressaltar que, embora o sistema de detecção tenha atingido um percentual alto em IoU, a arquitetura proposta depende do tamanho das células do *grid* para refinar a precisão da marcação, que é o que define a qualidade do sistema. Portanto, algumas marcações embora corretas podem ficar por vezes deslocadas da região verdadeira de marcação.

### 5.8.1 Tempo de Processamento

Para a validação da arquitetura da Figura 5.1 foram realizados experimentos utilizando 1, 2, 4 e 8 *threads* respectivamente, com o objetivo de avaliar o tempo de processamento. Foram avaliados os casos definidos na subseção anterior.

As Figuras 5.13 e 5.14 apresentam os resultados. No caso 5.13 (a), existe uma queda de desempenho em relação a células de  $50 \times 50$  pixels, apresentado em 5.13 (b). Isso ocorre devido ao tempo para cálculo da correlação de cada célula. Percebe-se também que, em geral, tanto em 5.13(a) quanto em 5.13(b) existe um pico de desempenho utilizando-se 4 *threads*, atingindo aproximadamente 9 FPS, resultados esses relativos à utilização da RNA.

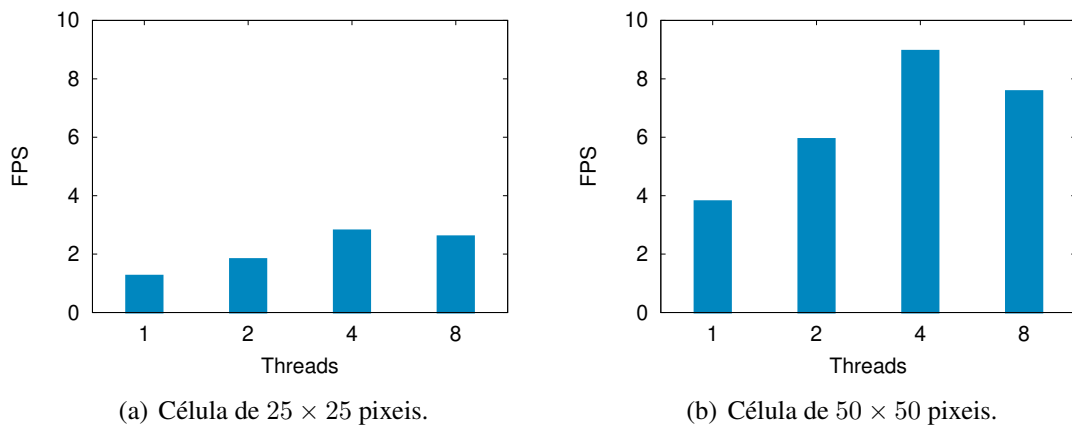


Figura 5.13 – Tempos totais de execução de RNA.

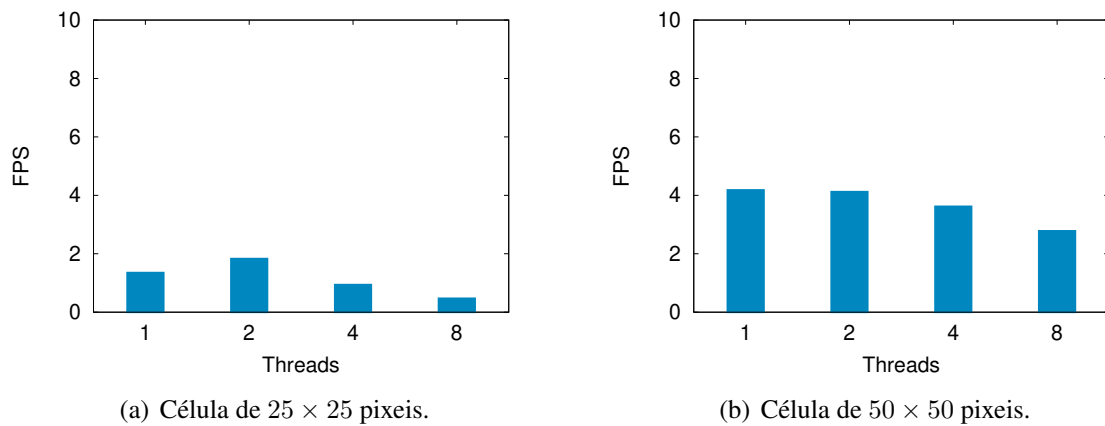


Figura 5.14 – Tempos totais de execução de SVM.

Portanto, se for considerado um veículo à 80 Km/h (ou 22 m/s), a velocidade de processamento será de aproximadamente um *frame* processado para cada 2.5m. Considerando que, os buracos são detectados desde o início de seu surgimento no campo de visão das imagens conforme observado nas Figuras 5.11 (c), (d) e (h) (buracos localizados no topo das imagens), pode-se dizer que 1 *frame* para cada 2.5 metros é o suficiente para que os buracos sejam detectados pelo menos uma vez antes de saírem novamente do campo de visão. Esses resultados apontam que embora a técnica ainda não possa ser aplicada exatamente em tempo real, é possível mapear as rodovias e servir inclusive como alerta entre motoristas, indicando qual o estado de conservação de determinada pista ou trecho de rodovia, se está em boas condições ou não. Ressalta-se ainda que, a relação entre velocidade de veículos, e possíveis reflexos relativos ao desvio de um buraco por parte de motoristas dado um alerta pelo sistema não é objeto de estudo

desse trabalho, portanto o comparativo de tempo feito se restringe estritamente ao cálculo de tempo de processamento por distância percorrida.

Outro fator que pode interferir na qualidade de *frames* computados é a angulação da imagem em relação à pista. Ou seja, angulações mais perpendiculares à pista tendem a capturar melhor as imagens de buracos, e com maior qualidade é possível melhor predição. Porém como já mencionado, tão logo um buraco que entra no campo de visão da imagem capturada é detectado, menos necessário se torna a quantidade de *frames* computados pela distância percorrida, que é o caso de imagens com angulações que tendem a ser mais paralelas em relação à pista. Contudo, a detecção se torna menos precisa quando buracos estão mais longe da fonte de captura da imagem.

Em resumo, a arquitetura tem seu melhor desempenho utilizando o classificador RNA, tanto qualitativamente realizando a predição de buracos, quanto em relação ao desempenho temporal. Contudo deve-se observar que a configuração usando grid com células de  $25 \times 25$  pixels os resultados em relação as métricas de acurácia, precisão e IoU são melhores, enquanto que células de  $50 \times 50$  pixels entregam resultados mais rápidos, porém de menor qualidade. Portanto é necessário investir na melhoria da implementação da técnica, deixando seu processamento mais rápido.

## 6 CONCLUSÕES

Este trabalho apresentou uma arquitetura computacional para detecção de buracos no asfalto, primeiramente realizando uma classificação de imagens em dois conjuntos: aqueles que contêm buracos e aqueles que não contêm buracos, e posteriormente realizando a marcação desses buracos em imagens com a construção de uma arquitetura de detecção. Foram avaliados diferentes classificadores, RNA e SVM, para detecção de buracos em asfalto, além de técnicas de pré-processamento e análise de texturas e seus resultados foram comparados com base nas métricas de acurácia, precisão, revocação e f-score, incluindo tempos de treinamento e processamento e a métrica de IoU para a detecção.

### 6.1 Observações sobre os resultados

Nesta abordagem, em contraste com outros trabalhos, foram usados algoritmos de processamento de imagens com o suporte de RNA e SVM para permitir a detecção de buracos em imagens capturadas por veículos. Também foi explorado o ajuste de parâmetros da técnica de pré-processamento que influenciou o desempenho a qualidade do resultado. Além disso, foram avaliados os tempos de execução nessa etapa também.

No geral, foi observado que a SVM fornece melhores resultados na classificação que a RNA, oferecendo taxas adequadas em todas as métricas e configurações. Por outro lado, a RNA requer menos quantidade de dados para processamento, o que pode ser uma vantagem em termos de armazenamento.

É possível notar que, os melhores resultados da RNA, obtidos com as configurações  $C_3$  e  $C_4$  do HOG, embora menos precisos que a SVM, têm tempos de treinamento semelhantes e necessitam de vetores HOG menores que a SVM.

Outro fato é o alto nível da métrica revocação na SVM, que revela a capacidade da técnica em aprender o que não é um buraco e que a estratégia de configuração  $C_2$  também foi mais eficiente na eliminação de falsos positivos.

Finalmente, a melhor solução encontrada é a combinação de HOG com a configuração  $C_2$  e 2 orientações, juntamente com a técnica SVM, atingindo 90,3% de acurácia, 100% de revocação e 91% de  $f_{score}$ . A RNA também alcançou resultados adequados com orientações e configuração de 5 HOG  $C_4$  atingindo 88% de acurácia, 93% de precisão, 84% de revocação e 88,8% de  $f_{score}$ . Nesse sentido pode-se ressaltar a importância das métricas de precisão e

revocação, pois a não marcação de buracos onde realmente há é um grande problema visto que o buraco não seria detectado. O contrário também poderia ser desastroso, uma vez que desviar de um buraco que não está lá poderia ser acarretar até mesmo em acidentes. Além disso, a solução final também pode ser selecionada levando em consideração o tempo de treinamento do conjunto de imagens.

Em relação a arquitetura de detecção, responsável por marcar os buracos nas imagens, verificou-se que tanto nas imagens de resultado quanto nas métricas de acurácia, precisão e IoU avaliadas os resultados foram satisfatórios. A métrica IoU atingiu 83%, o que representa o percentual com a qual o modelo conseguiu realizar a marcação de buracos em imagens corretamente em relação à localização verdadeira. Tanto SVM quanto a RNA nesse ponto atingiram valores bem próximos, sendo diferenciados pelas suas respectivas acurácias, precisões e tempos de processamento. Vale ressaltar que a qualidade das imagens pode impactar de forma significativa no resultado da detecção pelo modelo proposto.

Os tempos de processamento também foram avaliados e em conclusão, embora ainda não sejam exatamente adequados à execução em tempo real, podem ser processados e servir como alerta à outros motoristas ou para mapeamento das condições de tráfego de uma rodovia visando auxiliar mantenedores e investimentos governamentais

## 6.2 Trabalhos futuros

Como trabalhos futuros, pretende-se melhorar algumas partes da arquitetura proposta:

- melhorar a qualidade do mecanismo de *tracker* dos buracos em imagens, avaliando diferentes métodos de *thresholds* para melhor eleição de células com variação de textura, e também avaliar a criação de uma janela deslizante na imagem, em contraste com o modelo de *grid* rígido proposto nesse trabalho, onde as células estão lado a lado e não se sobrepõem. Isso poderia gerar maior precisão do *tracker* ao delimitar a região de um buraco, já que buracos localizados entre duas células na imagem seriam melhor demarcado, e consequentemente melhorando a acurácia e a IoU relativo ao problema.
- também pretende-se melhorar o tempo de execução da arquitetura de detecção, tendo em vista o aumento de *frames* processados e a possibilidade de utilização da arquitetura em tempo real.
- outro objetivo futuro é relativo ao aumento da base de imagens que servem como entradas

para os algoritmos de pré-processamento de imagens e posteriormente para os classificadores, de forma à aumentar a diversidade dos dados e conseqüentemente criar algoritmos capazes de reconhecer cada vez mais tipos distintos de buracos, ou até mesmo texturas asfálticas diferentes.

### **6.3 Apontamento final**

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

## REFERÊNCIAS

- ALLIANCE, A. I. Annual Local Authority Road Maintenance Survey. **Asphalt Industry Alliance, London, UK**, [S.l.], 2016.
- ALZOUBI, A. PotDataset. , [S.l.], 4 2018.
- ARTIFICIAL, I. Uma Abordagem de Aprendizagem de Máquina/Katti Faceli...[et al.]. **Rio de Janeiro: LTC**, [S.l.], 2011.
- AZHAR, K. et al. Computer vision based detection and localization of potholes in asphalt pavement images. **IEEE Canadian Conf. on Electrical and Computer Engineering (CCECE)**, [S.l.], p.1–5, 2016.
- BIERZYNSKI, K.; ESCOBAR, A.; EBERL, M. Cloud, fog and edge: cooperation for the future? In: SECOND INTERNATIONAL CONFERENCE ON FOG AND MOBILE EDGE COMPUTING (FMEC), 2017. **Anais...** [S.l.: s.n.], 2017. p.62–67.
- BUZA, E.; OMANOVIC, S.; HUSEINOVIC, A. Stereo vision techniques in the road pavement evaluation. **Proceedings of the 2nd International Conference on Information Technology and Computer Networks**, [S.l.], p.48–53, 2013.
- BUZA, E.; OMANOVIC, S.; HUSEINOVIC, A. Pothole detection with image processing and spectral clustering. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY AND COMPUTER NETWORKS, 2. **Proceedings...** [S.l.: s.n.], 2013. v.810, p.4853.
- CHAN, S. et al. PredictionIO: a distributed machine learning server for practical software development. In: ACM INTERNATIONAL CONFERENCE ON INFORMATION & KNOWLEDGE MANAGEMENT, 22. **Proceedings...** [S.l.: s.n.], 2013. p.2493–2496.
- CNT et al. **Anuário CNT do Transporte 2018**: estatísticas consolidadas. [S.l.: s.n.], 2018. 234p.
- DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR'05), 2005. **Anais...** [S.l.: s.n.], 2005. v.1, p.886–893.



- ENIGOA, V. S. F. et al. CrowdSourcing based online petitioning system for pothole detection using Android platform. **Procedia Computer Science, Volume 87**, [S.l.], p.316–321, 2016.
- ERIKSON, J.; GIROD, L.; HULL, B. The pothole patrol: using a mobile sensor network for road surface monitoring. **Proc. 6th International Conference on Mobile Systems, Applications, and Services**, [S.l.], p.29–39, 2008.
- FAUSETT, L. **Fundamentals of Neural Networks: architectures, algorithms, and applications**. [S.l.]: Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.
- FEITOSA, R. A. et al. Multidimensional Representations for the Gesture Phase Segmentation Problem - An Exploratory Study using Multilayer Perceptrons. In: INTERNATIONAL CONFERENCE ON AGENTS AND ARTIFICIAL INTELLIGENCE, 10. **Proceedings...** [S.l.: s.n.], 2018. p.347–354.
- Haralick, R. M.; Shanmugam, K.; Dinstein, I. Textural Features for Image Classification. **IEEE Transactions on Systems, Man, and Cybernetics**, [S.l.], v.SMC-3, n.6, p.610–621, Nov 1973.
- HAYFLICK, N. Is Elon Wrong About LiDAR? , [S.l.], August 2019.
- HAYKIN, S. **Neural networks: a comprehensive foundation**. [S.l.]: Prentice Hall PTR, 1994.
- HAYKIN, S. **Neural Networks and Learning Machines**. 3.ed. [S.l.]: Pearson, 2010. 936p.
- HOU, Z.; WANG, K. C. P.; GONG, W. Experimentation of 3D pavement imaging through stereovision. **Proc. Int. Conference on Transportation Engineering**, [S.l.], p.376–381, 2007.
- HURWITZ, J.; KIRSCH, D. Machine learning for dummies. **IBM Limited Edition**, [S.l.], v.75, 2018.
- JOG, G. M. et al. Pothole properties measurement through visual 2D recognition and 3D reconstruction. **Proceedings of the ASCE Int. Conference on Computing in Civil Engineering**, [S.l.], p.553– 560, 2012.
- KANG, B.; CHOI, S. Pothole detection system using 2D LiDAR and camera. In: INT. CONF. ON UBIQUITOUS AND FUTURE NETWORKS (ICUFN), 9. **Anais...** [S.l.: s.n.], 2017. p.744–746.

- KIM, T.; RYU, S.-K. Review and analysis of pothole detection methods. **Journal of Emerging Trends in Comp. and Information Sciences**, [S.l.], v.5, n.8, p.603–608, 2014.
- KOCH, C.; BRILAKIS, I. Pothole detection in asphalt pavement images. **Advanced Engineering Informatics**, [S.l.], v.25, n.3, p.507–515, 2011.
- KOCH, C.; JOG, G. M.; BRILAKIS, I. Pothole detection with image processing and spectral clustering. **Journal of Computing in Civil Engi.**, [S.l.], v.27, n.4, p.370–378, 2013.
- KOKAR, M. M.; ENDSLEY, M. R. Situation awareness and cognitive modeling. **IEEE Intelligent Systems**, [S.l.], v.27, n.3, p.91–96, 2012.
- KOSKO, B. **Neural networks for signal processing**. [S.l.]: Prentice-Hall, Inc., 1992.
- LI, Q. et al. A real-time 3D scanning system for pavement distortion inspection. **Measurement Science and Technology**, Vol. 21, No. 1, [S.l.], p.15702–15709, 2009.
- LIN, J.; LIU, Y. Potholes Detection Based on SVM in the Pavement Distress Image. **9th International Symposium on Distributed Computing and Applications to Business, Engineering and Science**, [S.l.], p.544–547, 2010.
- LOKESHWOR, H.; DAS, L. K.; SUD, S. K. Method for automated assessment of potholes, cracks and patches from road surface video clips. **Procedia – Social and Behavioral Sciences**, [S.l.], v.104, p.312–321, 2013.
- MADEO, R. C. B.; LIMA, C. A. M.; PERES, S. M. A Review on Temporal Reasoning Using Support Vector Machines. In: INTERNATIONAL SYMPOSIUM ON TEMPORAL REPRESENTATION AND REASONING, 19. **Anais...** [S.l.: s.n.], 2012. p.114–121.
- MADEO, R. C. B.; PERES, S. M.; LIMA, C. A. d. M. Gesture Phase Segmentation Using Support Vector Machines. **Expert Syst. Appl.**, [S.l.], v.56, n.C, p.100–115, Sept. 2016.
- MEDNIS, A. et al. Real time pothole detection using Android smartphones with accelerometers. **Int. Conf. on Distributed Computing in Sensor Systems and Workshops (DCOSS)**, [S.l.], p.1–6, 2011.
- MOAZZAM, I. et al. Metrology and visualization of potholes using the Microsoft Kinect sensor. **Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems**, [S.l.], p.1284–1291, 2013.

NIENABER, S.; BOOYSEN, J. M.; KROON, R. Detecting potholes using simple image processing techniques and real-world footage. **34th Southern African Transport Conference (SATC 2015)**, [S.l.], p.153–164, July 2015.

Otsu, N. A Threshold Selection Method from Gray-Level Histograms. **IEEE Transactions on Systems, Man, and Cybernetics**, [S.l.], v.9, n.1, p.62–66, Jan 1979.

OUMA, Y.; HAHN, M. Pothole detection on asphalt pavements from 2D-colour pothole images using fuzzy c-means clustering and morphological reconstruction. **Automation in Construction**, [S.l.], v.83, p.196–211, 2017.

RIBEIRO, M.; GROLINGER, K.; CAPRETZ, M. A. Mlaas: machine learning as a service. In: IEEE 14TH INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND APPLICATIONS (ICMLA), 2015. **Anais...** [S.l.: s.n.], 2015. p.896–902.

Samuel, A. L. Some Studies in Machine Learning Using the Game of Checkers. **IBM Journal of Research and Development**, [S.l.], v.3, n.3, p.210–229, July 1959.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding machine learning: from theory to algorithms**. [S.l.]: Cambridge university press, 2014.

Silveira Rodrigues, R. et al. Pothole Detection in Asphalt: an automated approach to threshold computation based on the haar wavelet transform. In: IEEE 43RD ANNUAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE (COMPSAC), 2019. **Anais...** [S.l.: s.n.], 2019. v.1, p.306–315.

VIGNESHWAR, K.; KUMAR, B. H. Detection and counting of pothole using image processing techniques. **IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)**, [S.l.], p.1–4, 2016.

WANG, P. et al. Asphalt Pavement Pothole Detection and Segmentation Based on Wavelet Energy Field. **Mathematical Problems in Engineering**, [S.l.], v.2017, p.1–13, 2017.

YU, X.; SALARI, E. Pavement Pothole Detection and severity measurement using laser imaging. **IEEE International Conference on Electro/Information Technology (EIT)**, [S.l.], p.1–5, 2011.

ZOYSA, K. D. et al. A public transport system based sensor network for road surface condition monitoring. **Proc. Workshop on Networked Sys. for Developing Regions**, [S.l.], p.1–6, 2007.