



# **Universidade de Brasília**

**Instituto de Ciências Exatas  
Departamento de Ciência da Computação**

## **Detecção e monitoramento de vagas disponíveis em estacionamentos abertos através de processamento de imagens**

Vitor de Alencastro Lacerda

Monografia apresentada como requisito parcial  
para conclusão do Bacharelado em Ciência da Computação

Orientador  
Prof. Dr. Alexandre Zaghetto

Brasília  
2016

Universidade de Brasília — UnB  
Instituto de Ciências Exatas  
Departamento de Ciência da Computação  
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Rodrigo Bonifácio de Almeida

Banca examinadora composta por:

Prof. Dr. Alexandre Zaghetto (Orientador) — CIC/UnB  
Prof. Dr. Flávio Barros Vidal — CIC/UnB  
MsC. Luiz Henrique Morais Aguiar — CIC/UnB

### **CIP — Catalogação Internacional na Publicação**

Lacerda, Vitor de Alencastro.

Detecção e monitoramento de vagas disponíveis em estacionamentos abertos através de processamento de imagens / Vitor de Alencastro Lacerda. Brasília : UnB, 2016.

135 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2016.

1. Estacionamento, 2. Vagas livres, 3. Processamento de Imagens,
4. Redes Neurais, 5. Fluxo Óptico, 6. GLCM

CDU 004.4

Endereço: Universidade de Brasília  
Campus Universitário Darcy Ribeiro — Asa Norte  
CEP 70910-900  
Brasília-DF — Brasil



# **Universidade de Brasília**

**Instituto de Ciências Exatas  
Departamento de Ciência da Computação**

## **Detecção e monitoramento de vagas disponíveis em estacionamentos abertos através de processamento de imagens**

Vitor de Alencastro Lacerda

Monografia apresentada como requisito parcial  
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Alexandre Zaghetto (Orientador)  
CIC/UnB

Prof. Dr. Flávio Barros Vidal MsC. Luiz Henrique Morais Aguiar  
CIC/UnB CIC/UnB

Prof. Dr. Rodrigo Bonifácio de Almeida  
Coordenador do Bacharelado em Ciência da Computação

Brasília, 28 de Novembro de 2016

# Dedicatória

Dedico a todos os outros motoristas frustrados que gostariam de perder menos tempo estacionando seus carros.

# Agradecimentos

Agradeço a minha família e amigos pelo apoio durante toda a longa duração do curso e da elaboração deste trabalho e ao professor Alexandre Zaghetto pela orientação.

# Resumo

Encontrar vagas livres em grandes estacionamentos muitas vezes é uma tarefa frustrante e que consome muito tempo. Soluções que ajudam um motorista a encontrar uma vaga mais rapidamente economizam tempo e dinheiro, além de servir como fator diferencial entre estabelecimentos comerciais. Este trabalho apresenta um programa que analisa a imagem de uma câmera que filma um estacionamento e determina a ocupação das vagas presentes na imagem através da detecção de movimento no vídeo e uma rede neural artificial. Uma solução como esta, que utiliza técnicas de processamento de imagens, tem custo de instalação e manutenção muito menor do que o uso de sensores individuais nas vagas, além de ser muito mais adequada para uso em grandes estacionamentos descobertos.

**Palavras-chave:** Estacionamento, Vagas livres, Processamento de Imagens, Redes Neurais, Fluxo Óptico, GLCM

# Abstract

Finding free parking spaces in big parking lots is often a frustrating and time-consuming task. Solutions that help drivers find spaces faster save both time and money, and can help attract customers to commercial establishments. This work presents an approach that analyzes images from a camera filming a parking lot and determines parking space occupancy through movement detection and an artificial neural network. Solutions that use image processing techniques have much lower installation and maintenance costs than solutions that involve individual sensors. They are also more adequate for outdoor parking lots.

**Keywords:** Parking lots,Free Spaces, Image processing, Neural Networks, Optical Flow, GLCM

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Fundamentação Teórica</b>	<b>3</b>
2.1	Processamento de Imagens . . . . .	3
2.1.1	Imagens em nível de cinza . . . . .	3
2.1.2	Espaços de cores . . . . .	3
2.1.3	O espaço RGB . . . . .	4
2.1.4	Espaço YCbCr . . . . .	4
2.1.5	Descritores de textura . . . . .	5
2.1.6	GLCM . . . . .	6
2.2	Vídeos . . . . .	7
2.2.1	Fluxo Óptico . . . . .	7
<b>3</b>	<b>Redes Neurais Artificiais</b>	<b>9</b>
3.1	Neurônios . . . . .	9
3.2	Redes Neurais Feed-Forward . . . . .	11
3.3	Treinamento . . . . .	14
3.4	Classificação de padrões . . . . .	15
<b>4</b>	<b>Trabalhos Correlatos</b>	<b>17</b>
<b>5</b>	<b>Solução Proposta</b>	<b>19</b>
5.1	Aquisição . . . . .	19
5.2	Regiões de Interesse . . . . .	21
5.3	Classificação das seções . . . . .	22
5.3.1	Extração de características . . . . .	22
5.3.2	Classificação da rede neural artificial . . . . .	27
5.3.3	Ajuste gaussiano dos vizinhos . . . . .	27
5.3.4	Sistema de votação . . . . .	28
5.4	Extração do movimento . . . . .	30
5.4.1	Magnitudes do movimento . . . . .	31
5.4.2	Regiões de movimento . . . . .	31
5.5	Mapeamento de Vagas . . . . .	33
<b>6</b>	<b>Resultados Experimentais</b>	<b>36</b>
6.1	Aquisição de dados . . . . .	36
6.2	Treinamento da rede neural artificial . . . . .	37

6.3	Resultados Obtidos . . . . .	38
6.3.1	Vídeo 1 . . . . .	40
6.3.2	Vídeo 2 . . . . .	41
6.3.3	Vídeo 3 . . . . .	43
6.3.4	Vídeo 4 . . . . .	45
6.3.5	Vídeo 5 . . . . .	47
6.3.6	Vídeo 6 . . . . .	48
6.3.7	Vídeo 7 . . . . .	50
6.3.8	Vídeo 8 . . . . .	51
<b>7</b>	<b>Conclusão</b>	<b>54</b>
<b>Referências</b>		<b>56</b>

# Listas de Figuras

2.1	(a) Uma matriz com níveis de cinza e (b) a imagem correspondente.	4
2.2	(a) Uma imagem e cada um de seus canais RGB ordenados da esquerda para a direita e (b) os canais YCbCr da imagem ordenados da esquerda para a direita.	5
2.3	Exemplo da elaboração da GLCM	6
2.4	(a) Representação dos vetores de velocidade estimado pelo fluxo óptico. (b) Imagem em níveis de cinza onde a intensidade do <i>pixel</i> representa a magnitude do seu vetor de velocidade.	8
3.1	A estrutura básica de um neurônio humano.	10
3.2	A estrutura básica de um neurônio artificial.	10
3.3	(a) Gráfico da função degrau com limiar 0. (b) Gráfico da função logística com $a = 1$	12
3.4	A arquitetura feed-forward. Cada neurônio se comunica com os neurônios da camada seguinte até que a saída final seja produzida.	13
3.5	Um gráfico que mostra duas características $\tilde{x}_1$ e $\tilde{x}_2$ de um problema de classificação hipotético.	15
5.1	Um fluxograma com o funcionamento geral do <i>DVE</i>	20
5.2	Um quadro de um vídeo adquirido por uma câmera do sistema	21
5.3	O mesmo quadro da Figura 5.2, depois de definidas as regiões de interesse, marcadas de verde.	21
5.4	As ROIs determinadas na Figura 5.3 separadas em trinta seções verticais que ainda não foram classificadas. Nesta etapa os vetores $v$ de cada região é composto de trinta valores 2.	22
5.5	Uma imagem correspondente a uma das seções verticais definidas.	23
5.6	O histograma referente aos valores de contraste no conjunto analisado. Apesar de haver bastante sobreposição dos valores ainda é possível ver que há pouco ou nenhuma ocorrência de vagas após um certo valor.	24
5.7	O histograma referente aos valores de correlação no conjunto analisado. Aqui é possível ver um limiar inferior para a classe dos carros. Valores abaixo deste limiar provavelmente são vagas livres.	24
5.8	O histograma referente aos valores de energia no conjunto analisado. Há um ponto de divisão entre as duas classes, com pouca sobreposição de valores entre as classes.	25

5.9	O histograma referente aos valores de homogeneidade no conjunto analisado. Esse gráfico mostra mais sobreposição do que os anteriores, mas ainda contém bastante informação sobre a classe das vagas. . . . .	25
5.10	O histograma referente aos valores médios de crominância azul no conjunto analisado. Apesar de haver alta variedade nos valores encontrados nas imagens de carros, as imagens de faixa possuem valores médios em uma faixa estreita. . . . .	26
5.11	O histograma referente aos valores médios de crominância vermelha no conjunto analisado, mostrando comportamento semelhante ao histograma de valores médios de crominância azul. . . . .	26
5.12	Uma ilustração simplificada da influência que uma seção exerce sobre a classificação de suas vizinhas. . . . .	28
5.13	Uma imagem correspondente a uma seção dividida entre carro e vaga vazia, cuja classificação se beneficia do ajuste pelas vizinhas. . . . .	29
5.14	Um tipo de erro que o ajuste gaussiano ajuda a corrigir. As seções vermelhas são ocupadas e as verdes são livres. . . . .	29
5.15	O fluxograma de funcionamento do sistema de votação. . . . .	29
5.16	Um exemplo de quadro com as seções devidamente classificadas, as seções vermelhas representam espaço ocupados por carros, enquanto as verdes representam espaços livres. . . . .	30
5.17	Ilustração do processo de extração do movimento significativo no vídeo. (a) mostra o quadro analisado, (b) uma representação visual da matriz de magnitudes das velocidades e (c) uma representação da matriz depois de limiarizada. . . . .	31
5.18	Um retângulo que representa a estimativa da área ocupada por um objeto em movimento, representado em azul. . . . .	33
5.19	O retângulo em azul interceptando algumas seções verticais. As seções tocadas pelo retângulo serão reclassificadas ao final do movimento. . . . .	33
5.20	No quadro a esquerda, um movimento representado pelo retângulo azul intercepta três seções que pertenciam a vaga com os contornos pretos. As seções interceptadas passam a compor uma nova vaga, marcada de azul, enquanto a vaga marcada de preto agora é composta apenas da seção restante.	34
6.1	O drone utilizado para a gravação dos vídeos de teste. . . . .	36
6.2	(a) Um exemplo de imagem classificada manualmente na classe 1.(b) Um exemplo de imagem classificada manualmente na classe 2. . . . .	37
6.3	Figura mostrando a divisão das ROIs e numeração das seções nos vídeos mostrados para os observadores humanos. . . . .	38
6.4	(a) O momento inicial do vídeo 1; (b) O momento final do vídeo 1. . . . .	40
6.5	(a) O momento inicial do vídeo 2; (b) O momento final do vídeo 2. . . . .	42
6.6	(a) O momento inicial do vídeo 3; (b) O momento final do vídeo 3. . . . .	43
6.7	(a) O momento inicial do vídeo 4; (b) O momento final do vídeo 4. . . . .	45
6.8	(a) O momento inicial do vídeo 5; (b) O momento final do vídeo 5. . . . .	47
6.9	(a) O momento inicial do vídeo 6; (b) O momento final do vídeo 6. . . . .	49
6.10	(a) O momento inicial do vídeo 7; (b) O momento final do vídeo 7. . . . .	50
6.11	(a) O momento inicial do vídeo 8; (b) O momento final do vídeo 8. . . . .	52

7.1 Resultados preliminares de um possível trabalho futuro. . . . .	55
---	----

# Lista de Tabelas

# Capítulo 1

## Introdução

Atualmente, a frota de carros dos grandes centros urbanos fica cada vez maior. Esse aumento no número de veículos cria uma demanda por espaço de estacionamento, levando a construção de novos estacionamentos e uma dificuldade muito maior de se encontrar vagas disponíveis nos estacionamentos existentes nos estabelecimentos comerciais, áreas residenciais e centros urbanos. Dele De fato, todo ano, milhões de pessoas gastam milhares de horas rondando estacionamentos de supermercados, *shoppings* e prédios comerciais em busca de uma vaga de estacionamento vazia. Procurar por uma vaga em um estacionamento grande e cheio é uma tarefa cansativa e desagradável. Muitas vezes vagas desocupadas não são encontradas por motoristas que preferem parar seu carro e esperar que uma vaga próxima seja liberada. Outras vezes possíveis clientes desistem de ir a um determinado estabelecimento pois sabem da dificuldade de estacionar que enfrentarão. Esse problema é comum principalmente em *shopping centers* e grandes supermercados, que possuem seus próprios estacionamentos disponíveis para os clientes, mas que estão frequentemente muito cheios.

Facilitar a tarefa da busca de vagas em estacionamentos de estabelecimentos como estes, é benéfico então não só para os clientes, mas também para os donos e gerentes desses estabelecimentos. Sendo assim, esse trabalho tem como objetivo criar um sistema que seja capaz de determinar a ocupação das vagas de um estacionamento descoberto e informar quantas vagas estão disponíveis e a sua localização aproximada para os motoristas através do processamento de imagens do estacionamento. Tal solução facilitaria a procura de vagas, diminuindo o tempo gasto trafegando por fileiras de vagas ocupadas.

A solução apresentada utiliza redes neurais artificiais combinadas com técnicas descritoras de textura e características de crominância para diferenciar imagens de carros de imagens de vagas desocupadas. Cada região que contém vagas na imagem obtida é dividida em diversas seções verticais. Cada seção será classificada entre vaga vazia ou veículo, de forma que o programa então é capaz de determinar quantas vagas estão ocupadas nessa região e, consequentemente, quantas estão vazias. Algumas soluções propostas em trabalhos correlatos serão apresentadas no Capítulo 4.

No Capítulo 2 serão apresentados alguns conceitos importantes para o entendimento da solução que será apresentada posteriormente. Serão detalhados o que é uma imagem, como funciona um vídeo e outros elementos teóricos importantes como espaços de cores e descritores de textura. O Capítulo 3 apresenta uma fundamentação teórica de redes neurais artificiais. Esse assunto foi separado em um capítulo próprio por causa da grande

quantidade de conteúdo e por sua importância especial neste trabalho.

No Capítulo 5, a solução proposta é apresentada. Nesta seção será detalhado o funcionamento do sistema criado, o fluxo de execução do programa e os elementos que permitem o funcionamento do sistema.

Em seguida, o Capítulo 6 apresentará resultados obtidos em testes que comparavam o sistema a observadores humanos, seguidos de observações sobre os resultados do programa. Esse capítulo também apresenta os detalhes da rede neural artificial utilizada.

O Capítulo 7 possui comentários sobre a taxa de sucesso do trabalho, suas fragilidades, sua adequação para uso no estado atual e se o objetivo desejado foi alcançado. O capítulo finaliza o trabalho com um comentário sobre possíveis trabalhos futuros para a evolução do sistema.

# Capítulo 2

## Fundamentação Teórica

Esse capítulo tem como objetivo apresentar conceitos que facilitarão o entendimento do conteúdo do Capítulo 5, detalhando técnicas e conceitos utilizados na execução do trabalho através de exemplos e imagens. Serão apresentados conceitos relacionados ao processamento de imagens estáticas, seguidos de uma discussão breve sobre técnicas aplicadas em vídeos. O conteúdo deste capítulo continua no Cápítulo 3 que apresenta conceitos de redes neurais e classificação de padrões.

### 2.1 Processamento de Imagens

#### 2.1.1 Imagens em nível de cinza

Para que um computador seja capaz de operar sobre uma imagem, é preciso que seja utilizado um modelo de representação que traduza o que os nossos sentidos conseguem perceber em informações que podem ser interpretadas por uma máquina que não é dotada de visão. A forma mais simples de se representar uma imagem são as imagens em nível de cinza. Podemos definir imagens como uma função  $f(x,y)$  onde  $x$  e  $y$  são coordenadas espaciais e o valor de  $f(x,y)$  é a luminosidade, ou nível de cinza, da imagem naquele ponto. Quando esses valores são todos discretos, chamamos essa imagem de uma imagem digital [1]. Cada elemento individual dessa imagem, cada valor em cada coordenada, pode ser chamado de um *picture element* ou mais comumente *pixel*. Em imagens digitais em nível de cinza, cada *pixel* possui 1 *byte* de informação, ou seja, pode assumir valores entre 0 e 255, onde o valor 0 representa o preto e 255 representa o branco.

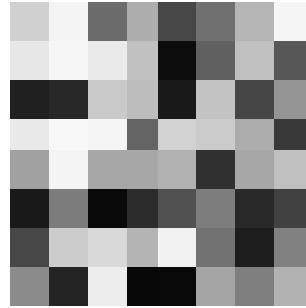
Uma imagem então, pode ser interpretada por um programa de computador como uma matriz  $M \times N$  de elementos de 1 *byte*, onde  $M$  é a largura da imagem e  $N$  é a sua altura. Cada elemento  $p_{i,j}$  da matriz possui a informação de luminosidade do pixel correspondente e o computador é capaz de exibir e interpretar esses valores apropriadamente. A Figura 2.1 exemplifica esse processo.

#### 2.1.2 Espaços de cores

Para que um sistema de processamento de imagens possa interpretar e processar cores, é preciso criar modelos apropriados para representá-las. Chamamos os modelos de representação das cores em uma imagem um espaço de cor. Existem diversos espaços de

208	244	108	173	71	112	181	245
231	246	234	193	12	97	192	87
32	40	202	189	25	195	70	149
233	248	245	100	210	203	173	57
161	244	167	167	177	48	167	192
25	124	9	44	81	125	41	65
71	204	217	180	242	114	30	129
139	36	238	8	9	165	127	178

(a)



(b)

Figura 2.1: (a) Uma matriz com níveis de cinza e (b) a imagem correspondente.

cor, cada um importante para a realização de tarefas e necessidades diferentes [2]. Para a compreensão deste trabalho, é necessário, porém, entender apenas dois: o espaço RGB e o espaço YCbCr.

### 2.1.3 O espaço RGB

RGB é um acrônimo para *Red*, *Green*, *Blue*, vermelho, verde e azul em inglês. Essas cores são consideradas então cores primárias aditivas devido ao fato de pesquisas mostrarem que um grande gama de cores pode ser formado através de combinações aditivas das cores vermelho, verde e azul [3]. Nesse modelo, a cor de um *pixel* de uma imagem é representada através de três coeficientes que definem a influência de cada cor primária na combinação. Uma imagem no modelo RGB é então representada por três matrizes de níveis de cinza de dimensões iguais, chamadas canais, onde cada valor representado na matriz, representa o valor do coeficiente da cor correspondente na imagem final. Isto é, a imagem pode ser representada por uma matriz  $M \times N \times 3$  onde a cor final  $C_{i,j}$  de cada *pixel* da imagem é definida pela Equação 2.1. Se os coeficientes são representados por um *byte*, é possível então representar aproximadamente 16,7 milhões de cores com este modelo. A Figura 2.2a mostra os canais de uma imagem RGB separadamente.

Esse espaço de cor é o mais comumente usado para representar imagens que vemos no cotidiano, uma vez que é semelhante a visão humana e portanto é utilizado pelos dispositivos multimídia mais comuns.

$$C_{i,j} = p_{i,j,1} \cdot R + p_{i,j,2} \cdot G + p_{i,j,3} \cdot B, (1 \leq i \leq M, 1 \leq j \leq N) \quad (2.1)$$

### 2.1.4 Espaço YCbCr

Assim como o espaço de cor RGB, o espaço YCbCr também representa uma imagem através de três matrizes, porém, seus canais contém informações diferentes do modelo RGB. Esse modelo é muito utilizado para o armazenamento de vídeos, uma vez que o modelo tira vantagem de alguns aspectos da visão humana para poder armazenar menos dados, sem perda significativa de informação visual. Por exemplo, humanos são mais sensíveis à detalhes e variações em níveis de cinza do que detalhes em imagens coloridas. Além disso, o olho humano é mais sensível ao verde do que qualquer outra cor [4]. Com esse conhecimento, o espaço YCbCr representa as cores de uma imagem através de sua

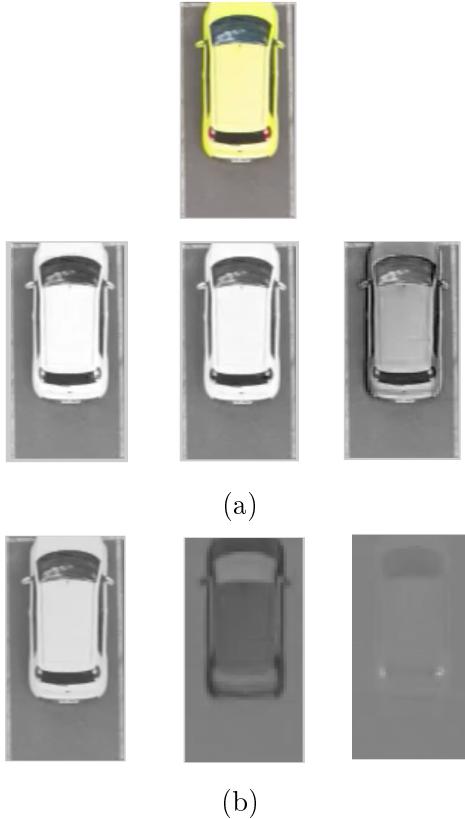


Figura 2.2: (a) Uma imagem e cada um de seus canais RGB ordenados da esquerda para a direita e (b) os canais YCbCr da imagem ordenados da esquerda para a direita.

luminosidade(Y) e os valores de crominância azul(Cb) e crominância vermelha(Cr). Cb e Cr são sinais de diferença de cor e são definidos pela subtração do valor de luminosidade do canal azul e vermelho da imagem, respectivamente. A luminosidade é definida pela Equação 2.2 [5] que reflete a sensibilidade maior a cor verde da visão humana.

Na Figura 2.2b estão exemplificados os canais de uma imagem no espaço YCbCr.

$$Y = 0,299.R + 0,587.G + 0,114.B \quad (2.2)$$

### 2.1.5 Descritores de textura

Descritores de textura são algoritmos que procuram fazer o que o olho humano faz com facilidade: distinguir tipos diferentes de objetos apenas por algumas de suas características visuais [6]. Estes algoritmos observam e analisam características pertinentes a imagem inteira e são capazes de identificar diferenças mais sutis entre imagens diferentes. Existem diversas técnicas de descrição de textura [7] que são utilizadas com sucesso em aplicações de processamento de imagem. Para esse trabalho, a técnica escolhida foi a conhecida como GLCM (*gray-level co-occurrence matrix*) descrita na Seção 2.1.6, por ter execução rápida e ser invariante quanto à escala de cinza.

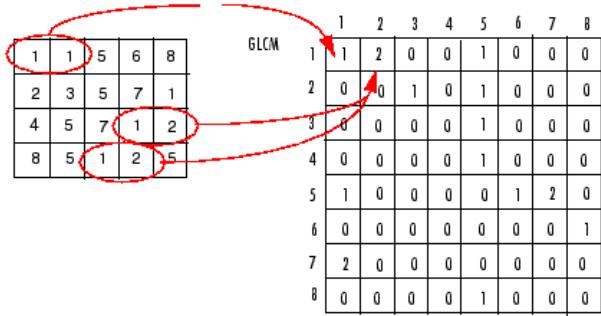


Figura 2.3: Exemplo da elaboração da GLCM<sup>1</sup>.

### 2.1.6 GLCM

GLCM, ou matriz de co-ocorrência de nível de cinza, é uma técnica de descrição de textura que visa extraír medidas estatísticas da imagem sendo analisada. Para tanto, é criada uma matriz quadrada de tamanho  $M \times M$  onde  $M$  é a quantidade de níveis de cinza possíveis na imagem em questão. Essa matriz armazena a probabilidade de dois *pixels* se relacionarem através de uma certa relação espacial [6].

Para esse trabalho, a relação observada é a da vizinhança entre dois *pixels*, sempre analisando o valor a direita de um determinado *pixel* da imagem. Os 255 valores de intensidade possíveis são divididos em 8 níveis. Cada elemento  $P_{i,j}$  na GLCM  $G$  conta a quantidade de vezes que um *pixel* com valor de intensidade  $j$  apareceu à direita de um *pixel* com valor  $i$ . A Figura 2.3 exemplifica a criação da GLCM a partir de uma imagem com 8 níveis possíveis.

Uma vez criada a matriz de co-ocorrência, diversas medidas podem ser extraídas. Para esse trabalho são extraídas as quatro características definidas abaixo. Nas equações apresentadas  $P_{(i,j)}$  representa o valor do elemento na posição  $(i,j)$  da GLCM,  $\mu_i$  e  $\mu_j$  representam a média dos valores de  $i$  e  $j$  respectivamente e  $\sigma_i$  e  $\sigma_j$  os desvios padrão destes valores.

- **Contraste:** mede o contraste entre um *pixel* e seu vizinho na imagem. Deve ser 0 para uma imagem completamente homogênea. Definido por:

$$C = \sum_{i,j} (i - j)^2 P_{(i,j)} \quad (2.3)$$

- **Correlação:** mede a taxa de correlação de um *pixel* e seu vizinho na imagem inteira. Definida por:

$$Co = \sum_{i,j} \frac{(i - \mu_i)(j - \mu_j)P_{(i,j)}}{\sigma_i \sigma_j} \quad (2.4)$$

- **Energia:** soma do quadrado dos elementos da imagem. Definida por:

$$E = \sum_{i,j} P_{(i,j)}^2 \quad (2.5)$$

---

<sup>1</sup>Extraída de <https://www.mathworks.com/help/images/ref/graycomatrix.html>.

- **Homogeneidade:** mede a proximidade da distribuição dos elementos à diagonal da matriz. Definida por:

$$H = \sum_{i,j} \frac{P_{(i,j)}}{1 + |i - j|} \quad (2.6)$$

## 2.2 Vídeos

Quando observamos uma cena no mundo real, a cena raramente é estática. Com o passar do tempo, os objetos presentes na cena se movem continuamente pelo espaço, criando infinitas imagens diferentes em nossas retinas. Podemos simular essa sensação de movimento na nossa visão através da exibição rápida de imagens que contém valores discretos. Esse é o conceito de um vídeo digital. Um vídeo digital é formado através da amostragem de imagens em três eixos: horizontal, vertical e temporal [5, 8]. A cada intervalo fixo de tempo, uma imagem representada por uma matriz bidimensional de valores de intensidade luminosa e cor é amostrada. Cada uma dessas imagens, ou quadros, representa o estado da cena naquele momento no tempo.

### 2.2.1 Fluxo Óptico

O movimento é um dos elementos principais para a extração de informações de um vídeo. O fluxo óptico [9], ou *optical flow*, é uma das técnicas para a estimativa de movimento em vídeos digitais. A técnica consiste em medir a projeção 2D no plano da imagem de um movimento real 3D. Essa projeção 2D mede a velocidade de cada *pixel* da imagem. Esses vetores de velocidade da imagem podem ser utilizados para a realização de diversas tarefas. Por exemplo, é possível extrair a magnitude dos vetores e criar uma nova imagem, aonde os valores de intensidade representam o movimento realizado pelo *pixel*. Nesse trabalho, essa técnica é utilizada para a segmentação de objetos em movimento contra um fundo estático, detalhada no Capítulo 5. A Figura 2.4 mostra os vetores de movimento calculados e uma imagem em níveis de cinza que representa as magnitudes das velocidades de cada pixel.

Para calcular o fluxo óptico, começamos assumindo que no intervalo entre dois quadros a intensidade de cada *pixel* não muda de forma significativa. Chamamos então a intensidade de cada *pixel* na posição  $(x,y)$  em um determinado momento  $t$  de  $I(x, y, t)$  e portanto, podemos dizer que:

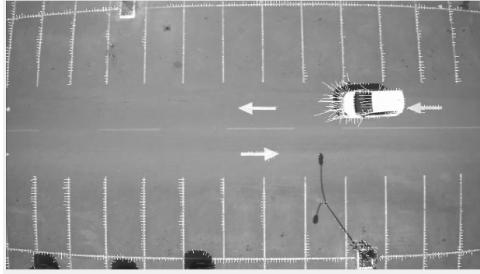
$$I(x, y, t) = I(x + dx, y + dy, t + dt) \quad (2.7)$$

onde  $dx$  e  $dy$  são o deslocamento que o *pixel* no intervalo  $dt$  entre dois quadros.

Queremos então encontrar a velocidade  $v = (\frac{dx}{dt}, \frac{dy}{dt})$  do *pixel*. Usando a expansão por série de Taylor e manipulações matemáticas na Equação 2.7, obtemos a Equação 2.8 [10].

$$\nabla I v + I_t = 0 \quad (2.8)$$

onde  $\nabla I$  é o gradiente da função intensidade. Essa equação descreve o chamado problema de Restrição do Fluxo Óptico [9]. Porém, há um problema. A partir da equação só é possível obter um dos componentes da velocidade. A solução completa da velocidade depende de um cálculo na vizinhança de cada ponto. Diversos métodos foram criados para



(a)



(b)

Figura 2.4: (a) Representação dos vetores de velocidade estimado pelo fluxo óptico. (b) Imagem em níveis de cinza onde a intensidade do *pixel* representa a magnitude do seu vetor de velocidade.

resolver a equação da Restrição do Fluxo Óptico, de naturezas distintas [9]. Para esse trabalho foi escolhido o método desenvolvido por Lucas e Kanade, detalhado em [9–11].

O método de Lucas-Kanade resolve a equação 2.8 a partir da suposição que o vetor do fluxo óptico se mantém constante entre os elementos de uma pequena janela imagem. Desta forma é possível elaborar um sistema determinado, que é resolvido por uma aproximação pela técnica dos mínimos quadrados. Apesar de criar uma malha de pontos do campo de velocidade menos densa que outros métodos, o método de Lucas-Kanade é mais resistente a ruídos. Por esse motivo este método foi escolhido para a elaboração do presente trabalho.

O Capítulo 3 continua a apresentação de conceitos importantes para o entendimento do trabalho, mas tem foco exclusivo em redes neurais artificiais.

# Capítulo 3

## Redes Neurais Artificiais

Comparado com os computadores mais avançados que existem hoje, o cérebro humano ainda se mostra muito mais poderoso e eficaz do que as máquinas [12]. O cérebro é capaz de processar informação a uma velocidade muito maior do qualquer computador convencional e realiza com facilidade tarefas como o reconhecimento de padrões e classificação de objetos, enquanto algoritmos tradicionais falham.

Redes neurais artificiais foram criadas com inspiração no funcionamento e na estrutura do cérebro, como uma alternativa poderosa para resolver problemas que as arquiteturas tradicionais não eram capazes de resolver eficientemente. Essas redes emulam a estrutura cerebral natural, se utilizando de elementos distintos de processamento (neurônios) que se comunicam para realizar a tarefa desejada.

Simon Haykin define uma rede neural como "...um processador distribuído massivamente paralelo composto por unidades simples de processamento, que possui uma propensidade natural a armazenar conhecimento experimental e torná-lo disponível para uso."

As redes neurais artificiais são utilizadas para resolver problemas como ajuste de funções, classificação de objetos e reconhecimento de padrões [13]. Este capítulo é uma continuação do Capítulo 2 e possui explicações e clarificações de conceitos importantes para o entendimento das redes artificiais, começando pelo seu elemento mais básico, o neurônio.

### 3.1 Neurônios

O neurônio é a estrutura básica do sistema nervoso [14]. É uma célula composta principalmente de três partes distintas: o corpo celular, os dendritos e o axônio. O corpo celular é a estrutura central da célula onde está contido o núcleo do neurônio e são executados as suas funções vitais. Os dendritos são prolongamentos do corpo celular responsáveis por receber sinais e o axônio é uma extensão maior do corpo celular, responsável por enviar sinais a outros neurônios. Dois neurônios interagem em apenas pontos de contato chamados sinapses. Nessas sinapses o axônio de um neurônio envia sinais para os dendritos de um segundo neurônio. A Figura 3.1 ilustra a estrutura básica dos neurônios.

O neurônio artificial também possui três estruturas básicas semelhantes àquelas do neurônio natural. Seus três elementos básicos são: as entradas(análogas aos dendritos), a

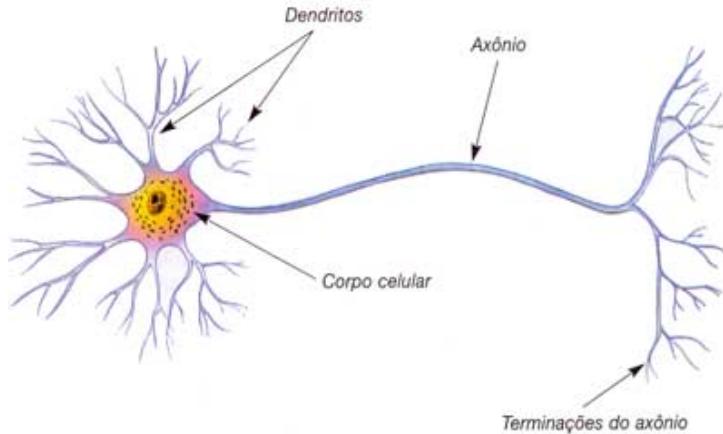


Figura 3.1: A estrutura básica de um neurônio humano.<sup>1</sup>

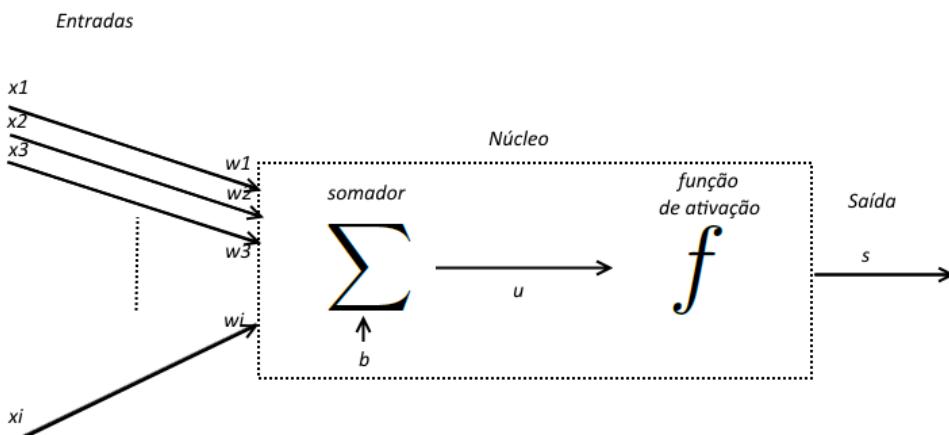


Figura 3.2: A estrutura básica de um neurônio artificial.

sáida(análoga ao axônio) e um núcleo de processamento. O neurônio pode ter uma ou mais conexões de entrada caracterizadas por um peso  $w_i$  e que recebe um valor  $x_i$ . No núcleo do neurônio artificial há um somador que calcula o valor  $u$  agregado das entradas, ponderadas pelo peso correspondente. O resultado dessa soma ponderada é então deslocado de um valor escalar e em seguida submetido a uma função chamada de função de ativação, que determina a saída final do neurônio. Em suma podemos descrever a saída de um neurônio  $n$  que possui  $i$  entradas através das seguintes equações:

$$u_n = \sum_{j=1}^i x_j w_j \quad (3.1)$$

$$s(n) = f(u_n + b) \quad (3.2)$$

Onde  $u_n$  é o valor agregado das entradas,  $x_j$  é a j-ésima entrada e  $w_j$  o peso da conexão associada,  $s(n)$  é a saída do neurônio,  $f$  é a função de ativação e  $b$  o deslocamento, ou *bias* do neurônio  $n$ . A Figura 3.2 ilustra a estrutura básica de um neurônio artificial.

<sup>1</sup>Extraída de <http://www.sobiologia.com.br/conteudos/FisiologiaAnimal/nervoso2.php>.

Claramente, a função de ativação do neurônio é o grande fator que define o seu comportamento e consequentemente sua funcionalidade. Existem dois tipos básicos de funções de ativação [12]: a função degrau e a função sigmóide.

O neurônio mais simples possível tem sua saída regida por uma função degrau e saída binária. Isto é, se o valor agregado  $u$  de suas entradas for maior que um determinado limiar, a sua saída será 1 e a saída será 0 caso contrário. Apesar de ser capaz de resolver alguns problemas, esse tipo de neurônio tem algumas desvantagens. Por vezes, alterações sutis podem representar a diferença entre os dois valores possíveis da saída. Além disso, um alcance limitado de valores na saída prejudica o treinamento.

A escolha mais comum de função de ativação para a construção de redes neurais é a família de funções sigmóides. Essas funções, que têm um gráfico com formato de  $S$ , assumem valores contínuos entre 0 e 1, o que faz com que alterações sutis na entrada representem alterações mais sutis na saída, aumentando a qualidade da informação gerada pelo neurônio. A mais comum das funções sigmóides utilizadas é a função logística, definida pela Equação 3.3 [15], onde  $a$  é uma constante que mede a declividade da curva.

$$f(u) = \frac{1}{1 + \exp -au} \quad (3.3)$$

Um dos motivos que tornou a função logística uma escolha popular para os neurônios artificiais foi a sua derivação simples, uma vez que algoritmos de treinamento muitas vezes usam a derivada da função de ativação.

É interessante reparar que quando  $a$  se aproxima do infinito, a função logística se comporta da mesma forma que a função degrau.

Outra função de interesse no estudo das redes neurais artificiais, é a função *softmax* ou exponencial normalizada [16]. Essa função é utilizada na teoria estatística para a determinação da distribuição da probabilidade. Isto é, a função *softmax* é uma função que define a probabilidade que uma variável discreta assuma um determinado valor [17]. Em termos simples, o seu papel é transformar um vetor de  $n$  elementos em um segundo vetor de mesmo tamanho, de valores no intervalo  $[0, 1]$  e que têm soma total igual a 1.

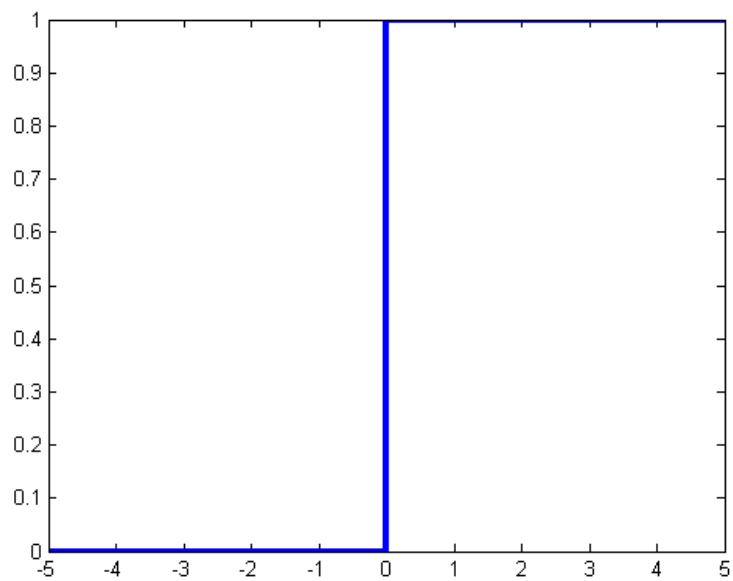
Essa função é muito utilizada como função de ativação nas camadas de saídas de redes treinadas para reconhecimento de padrões, por sua capacidade de determinar a probabilidade da entrada pertencer a uma determinada classe. A função *softmax* pode ser definida pela Equação 3.4, onde  $x$  é o vetor de tamanho  $n$  da entrada e  $x_i$  é o seu  $i$ -ésimo elemento.

$$p(x)_i = \frac{\exp x_i}{\sum_{j=1}^n \exp x_j} \quad (3.4)$$

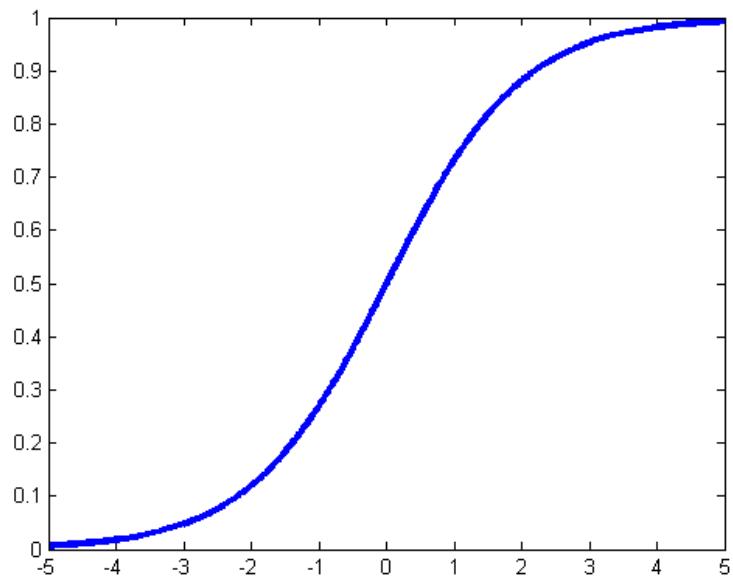
## 3.2 Redes Neurais Feed-Forward

Apesar de neurônios serem capazes de resolver alguns problemas sozinhos, o verdadeiro poder das redes neurais vem da interconexão entre os neurônios de forma a criar ligações semelhantes às sinapses dos neurônios naturais.

Existem várias arquiteturas para a formação destas redes de neurônios, porém neste trabalho a discussão será limitada àquela utilizada na implementação do programa: as chamadas redes neurais *feed-forward*.



(a)



(b)

Figura 3.3: (a) Gráfico da função degrau com limiar 0. (b) Gráfico da função logística com  $a = 1$

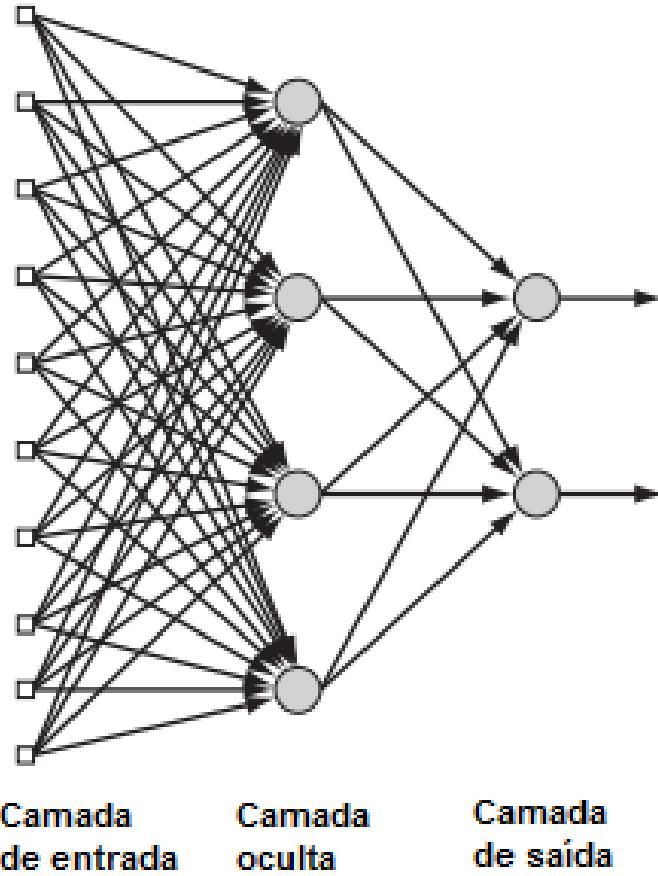


Figura 3.4: A arquitetura feed-forward. Cada neurônio se comunica com os neurônios da camada seguinte até que a saída final seja produzida.<sup>2</sup>

São necessárias ao menos três camadas para a construção da rede neste modelo: a primeira camada, chamada de camada de entrada, uma ou mais camadas intermediárias (ou ocultas) e a camada de saída. O papel das camadas ocultas é intermediar entre as entradas externas e a saída da rede, de forma a possibilitar que a rede extraia dados estatísticos mais significativos da sua entrada. Uma camada é composta de um número de neurônios que agem de forma paralela. Cada neurônio de uma camada que não é a de saída está conectado a todos os neurônios da camada seguinte, de forma que não há comunicação entre uma camada e camadas anteriores. Isto é, a informação flui na rede apenas no sentido entrada-saída, como ilustrado na Figura 3.4.

O vetor de entrada da rede é alimentado aos neurônios da primeira camada, cuja saída serve de entrada para a segunda camada e assim por diante, até que a última camada seja alimentada pela saída da camada anterior e produza a saída final da rede.

---

<sup>2</sup>Adaptada de [12].

### 3.3 Treinamento

Uma rede neural artificial precisa aprender a realizar a tarefa para qual foi criada antes que possa ser utilizada. Para isso a rede precisa ter duas habilidades importantes: de aprendizado e de generalização. Aprendizado é a capacidade da rede de aproximar o comportamento das entradas fornecidas durante o treinamento, enquanto generalização é a sua capacidade de prever e operar sobre dados além do conjunto com a qual foi treinada [18].

É necessário então que a rede passe por um processo de treinamento, onde os valores dos pesos  $w_i$  e os deslocamentos  $b$  de cada neurônio são definidos de forma que o funcionamento da rede seja ótimo para a tarefa que se deseja realizar. Para realizar o treinamento de uma rede, o primeiro passo é definir 3 conjuntos distintos de entradas:

- **Conjunto de treinamento:** A rede é submetida às entradas deste conjunto para que seja feita a calibração dos valores da matriz de pesos  $W$  e do vetor de deslocamentos  $B$ .
- **Conjunto de validação:** Após uma etapa de treinamento, a rede computa os dados deste conjunto de entradas para validar os valores da matriz  $W$  e do vetor  $B$  escolhidos até então.
- **Conjunto de testes:** Por fim, após o treinamento, a rede é submetida aos dados deste conjunto, a fim de verificar o seu funcionamento correto.

Isto é, a cada ciclo de treinamento, a rede ajusta os seus pesos e deslocamentos de acordo com o conjunto de treinamento, e em seguida processa os dados do conjunto de validação. Se for determinado que a rede teve sucesso no processo de validação, considera-se a rede pronta para passar pelo conjunto de testes, aonde a sua capacidade de generalização é testada. A matriz  $W$  de pesos gerada durante este processo é uma forma de representação do "conhecimento" da rede. Assim, o aprendizado não é uma característica de cada neurônio, mas sim um processo que ocorre na rede inteira como resultado do treinamento [15].

Existem duas principais maneiras de se executar o treinamento de uma rede.

- **Treinamento supervisionado:** Neste paradigma os conjunto de treinamento e de validação consistem em um grupo de vetores de entrada  $x$  e um gabarito de vetores de saída desejado  $y$ . O treinamento acontece até que o vetor de saída da rede para uma determinada entrada  $x_i$  seja suficientemente próximo do vetor gabarito  $y_i$  correspondente.
- **Treinamento não-supervisionado:** Neste paradigma a rede apenas recebe um conjunto  $x$  de entradas e aprende características intrínsecas dos dados contidos no conjunto.

Uma vez definido o paradigma de treinamento, é necessário definir uma técnica para a análise do erro e alteração dos valores da matriz de pesos  $W$  e do vetor  $B$  de deslocamentos. Uma das técnicas mais utilizadas é a chamada de *backpropagation* [18, 19]. A técnica consiste de uma análise do erro apresentado na saída da rede, que então é utilizada para se percorrer a rede no sentido contrário dos dados modificando os valores de  $W$  e  $B$  de

forma a diminuir o erro na próxima iteração. Essa técnica é muito utilizada para sistemas de treinamento supervisionado, onde o erro é avaliado através de uma comparação da saída da rede com o gabarito fornecido. Uma vez que os valores de  $W$  e  $B$  param de mudar, diz-se que a rede atingiu um estado de convergência [15] e o treinamento finaliza.

### 3.4 Classificação de padrões

Um problema muito comum que pode ser solucionado com redes neurais artificiais é o da classificação de padrões. Isto é, analisar um padrão ou um conjunto de características de um conjunto de dados e designá-lo a uma de várias classes pré-determinadas. Os nossos sentidos são muitas vezes capazes de fazer isso com extrema facilidade em uma fração de segundo. Por exemplo, se estamos andando pela rua, somos capazes de dizer que um som mais agudo e melódico é o canto de um pássaro, enquanto um som mais grave e contínuo provavelmente pertence ao motor de um carro passando. Essa habilidade é adquirida através de um aprendizado que ocorre durante as nossas vidas, e da mesma forma podemos ensinar uma rede neural artificial a reconhecer e classificar padrões.

Normalmente, uma rede que será utilizada para classificar padrões passa por um treinamento supervisionado, onde é alimentada com um vetor de entradas que contém características que descrevem os padrões junto com um gabarito que determina a classe a qual cada entrada pertence. Espera-se que depois do treinamento, ao ser apresentada a um novo padrão, a rede seja capaz de determinar a qual classe esse padrão pertence. Para que a rede demonstre boa generalização, é importante que a rede não aprenda uma representação exata do conjunto de treinamento, mas que construa um modelo estatístico do processo que gera os dados [20].

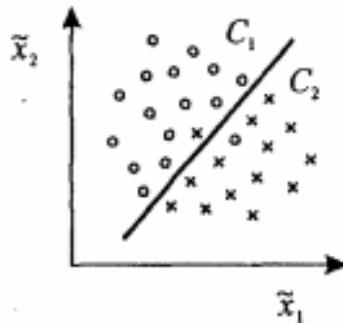


Figura 3.5: Um gráfico que mostra duas características  $\tilde{x}_1$  e  $\tilde{x}_2$  de um problema de classificação hipotético.<sup>3</sup> Círculos são padrões da classe  $C_1$ , enquanto os 'x' são padrões da classe  $C_2$ . O limiar de decisão, representado pela linha, consegue separar bem as duas classes quando as características são consideradas como um par. Porém haveria grande sobreposição se os valores fossem considerados separadamente.

A escolha das características a serem utilizados para descrever os padrões não pode ser aleatória. Deve se escolher um número de características suficiente para descrever todos os padrões que podem ser encontrados. Além disso é importante que as características sejam suficientemente discriminantes. Isto é, deve haver o mínimo possível de superposição

---

<sup>3</sup>Extraída de [20].

entre as características de classes distintas. Essas características podem ser escolhidos a mão, com base em um conhecimento prévio dos dados a serem classificados, ou extraídos automaticamente por um algoritmo desenvolvido para tal. Normalmente as características são processadas em conjunto e não consideradas individualmente. A Figura 3.5 mostra como isso é benéfico. Se as características hipotéticas  $\tilde{x}_1$  e  $\tilde{x}_2$  forem consideradas separadamente, há significativa superposição, de forma que é difícil separar com convicção as duas classes  $C_1$  e  $C_2$ , porém, quando consideradas juntas, é possível traçar uma linha que as separa facilmente, apesar de alguns poucos elementos se confundirem. O papel da rede é determinar a função que traça essa linha através do treinamento, e retornar uma probabilidade de que um conjunto de características que lhe foi fornecido faça parte de uma determinada classe.

Um exemplo: se queremos criar uma rede que diferencia imagens de peixes palhaço de imagens de tubarões, um conjunto adequado de características provavelmente envolveria a cor  $c$  predominante do animal, o tamanho do animal  $t$  e o tamanho  $b$  das suas barbatanas. A rede então seria submetida a entradas na forma de um vetor  $(c, t, b)^T$ , valores numéricos que caracterizam uma imagem a ser analisada. Podemos facilmente separar as classes através dessas características. Peixes com cor mais próxima do laranja e barbatanas pequenas são muito provavelmente peixes palhaços, enquanto aqueles que têm a barbatana grande seriam classificados como tubarões.

# Capítulo 4

## Trabalhos Correlatos

Nesse capítulo serão expostas soluções para o problema apresentado no Capítulo 1, acompanhadas de uma discussão sobre vantagens e desvantagens, e as razões pela escolha da solução proposta no Capítulo 5.

Podemos separar os métodos de detecção de vagas em duas grandes categorias: métodos intrusivos e não-intrusivos. Métodos intrusivos são caracterizados por exigirem instalações mais complexas, envolvendo a inserção de equipamento no asfalto do estacionamento ou em um estrutura de concreto. Os métodos não intrusivos normalmente se utilizam de equipamentos externos, que não exigem obras para a sua instalação.

Um exemplo comum de método intrusivo é o uso de sensores individuais em cada vaga do estacionamento. Diversos tipos de sensores podem ser utilizados, cada um com vantagens e desvantagens próprias [21]. Sensores infravermelhos versáteis e de simples instalação, mas podem ser afetados por condições ambientais. Tubos pneumáticos e sensores de peso sob o asfalto têm alta cofiabilidade, mas exigem grandes obras para a instalação.

Alguns tipos de sensores podem ser instalados de forma não intrusiva. Esses sensores podem ser afixados no teto ou em uma estrutura próxima a vaga. Sensores ultrassônicos por exemplo, pertencem a essa categoria. Esses sensores transmitem ondas sonoras de baixa frequência e usam a energia refletida para determinar a ocupação das vagas [22].

Intrusivos ou não, os métodos de detecção que utilizam sensores compartilham duas desvantagens: é necessário a instalação do sensor em cada uma das vagas, aumentando o custo do sistema e diminuindo sua escalabilidade, e eles não são adequados para estacionamentos descobertos, onde é impossível instalar sensores no teto e obras no asfalto rodoviário podem danificá-lo permanentemente [21].

Uma solução que se mostrou adequada para os estacionamentos descobertos foi o uso de *softwares* de visão computacional, que analisam imagens de câmeras de vídeo para determinar a ocupação das vagas do estacionamento. Essa opção tem um custo baixo e exige apenas que sejam instaladas câmeras em pontos estratégicos do estacionamento. Uma vez que as câmeras foram instaladas e as imagens estão sendo capturadas, resta que seja executado um programa que as analise.

Para a escolha da abordagem para este trabalhos foram escolhidos os seguintes critérios principais:

- O sistema deve poder ser instalado e começar sua execução em um estacionamento em qualquer estado, sem a necessidade de iniciar com todas as vagas descocupadas,

possibilitando uma instalação a qualquer momento, sem interrupção das operações do estacionamento.

- O sistema deve necessitar de interação humana mínima, se limitando apenas uma rápida calibração no início de sua execução a fim de evitar erro humano.
- O sistema deve ser capaz de identificar a posição das vagas do estacionamento após um certo tempo de execução, e não através de entrada manual, para compensar por erros na calibração.
- O sistema deve, além de determinar o número de vagas livres no estacionamento, ser capaz de indicar a posição aproximada de tais vagas de forma a facilitar ainda mais a busca por uma vaga desocupada.

Em [23], Delibaltov *et al* descrevem um método de detecção que estima um volume tridimensional para cada vaga marcada na imagem e depois se utiliza de redes neurais para determinar *pixels* da imagem pertencentes a veículos. Em seguida o sistema computa a probabilidade de que uma certa vaga esteja ocupada baseado em uma função de probabilidade e os *pixels* de veículo na região de interesse de cada vaga. Esse método se mostra preciso e eficaz, mas exige que o estacionamento esteja vazio e que cada vaga seja marcada individualmente no momento de execução, de forma que ele não é capaz de mapear as vagas do estacionamento ou detectar veículos estacionados irregularmente.

Bong [24] propõe um sistema que se utiliza de subtração de imagens e o uso de detecção automática de coordenadas aproximadas das vagas para resolver este problema. Porém esse sistema exige que seja armazenada uma imagem do estacionamento completamente vazio, além de que seja criada uma imagem que marca cada vaga para o processo de inicialização do sistema.

Um sistema que se utiliza de uma técnica de classificação de histogramas de crominância em áreas de interesse e um algoritmo de que encontra pontos com *features* relevantes foi proposto em [25]. Esse sistema tem a vantagem de ser relativamente invariante quanto à iluminação da imagem, por utilizar os canais de crominância para a análise. Também é bastante capaz de resolver o problema de oclusão de veículos atrás de outros na imagem de câmeras com uma angulação maior.

A solução proposta neste trabalho procura compensar alguns dos problemas presentes nos trabalhos mencionados, enquanto apresenta uma nova abordagem a ser considerada e aprimorada por trabalhos futuros.

# Capítulo 5

## Solução Proposta

Uma vez que o conhecimento teórico necessário para o entendimento completo do trabalho já foi apresentado, neste capítulo será definida a solução proposta.

A solução desenvolvida consiste em um algoritmo que analisa imagens de vídeo de um estacionamento descoberto e determina o número de vagas livres na imagem, além da sua localização aproximada. O sistema funciona bem em imagens de menor qualidade, mas é necessário que as imagens que compõem o vídeo estejam no espaço de cor RGB. Chamaremos a solução apresentada a seguir de *Dectector de vagas em estacionamento* ou *DVE*. A partir deste momento no trabalho, referências ao programa se referem ao *DVE*.

O trabalho se preocupa em cumprir os critérios definidos no Capítulo 4. Além disso, o sistema foi desenvolvido de forma que pudesse processar as imagens adquiridas de forma mais próxima possível do tempo real, causando o mínimo de atrasos para o processamento de quadros subsequentes do vídeo.

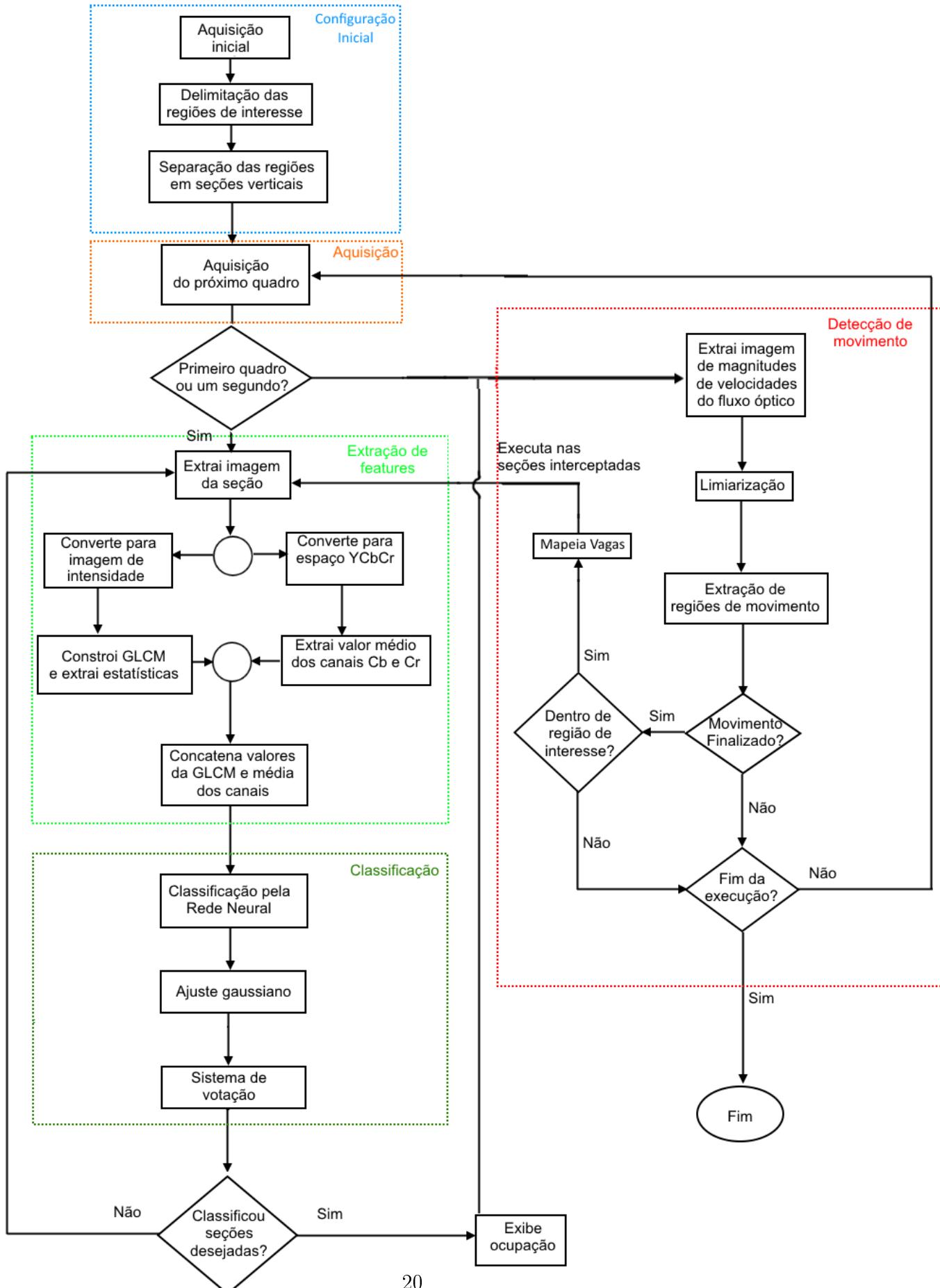
O sistema recebe como entrada um vídeo em cores capturado em um certo ângulo e um vetor de regiões de interesse no vídeo. A saída do *DVE* a cada quadro é o número de vagas livres em cada região de interesse no vídeo.

A Figura 5.1 contém um fluxograma que mostra as etapas do processamento de cada quadro do vídeo adquirido. No decorrer deste capítulo cada um desses passo será discutido com mais detalhes.

### 5.1 Aquisição

Uma câmera montada em um poste de luz ou outro ponto similar captura as imagens utilizadas pelo *DVE*. A câmera é montada de forma que o seu campo de visão contenha o máximo de vagas possível, porém que ainda seja possível visualizar o asfalto das vagas desocupadas e ocorra o mínimo de oclusão de veículos. A Figura 5.2 mostra um quadro de uma aquisição em ângulo ideal.

Neste trabalho, a preocupação foi implementar um programa que analisa as imagens de uma única câmera de vídeo. Em uma aplicação no mundo real, diversas câmeras seriam instaladas para aumentar a cobertura do estacionamento. Neste caso, cada vídeo seria processado por uma cópia diferente do sistema, responsável pela área coberta pela câmera correspondente. As saídas de cada cópia do programa correspondem a ocupação das vagas em um setor diferente do estacionamento.

Figura 5.1: Um fluxograma com o funcionamento geral do *DVE*

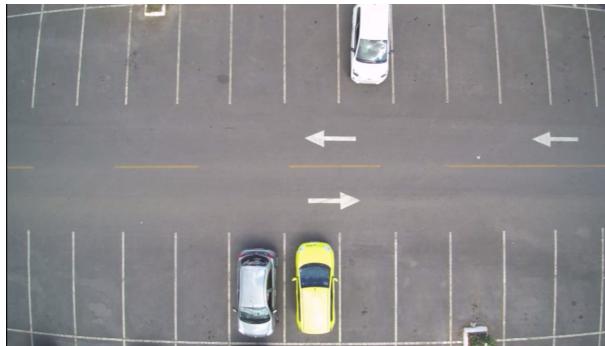


Figura 5.2: Um quadro de um vídeo adquirido por uma câmera do sistema

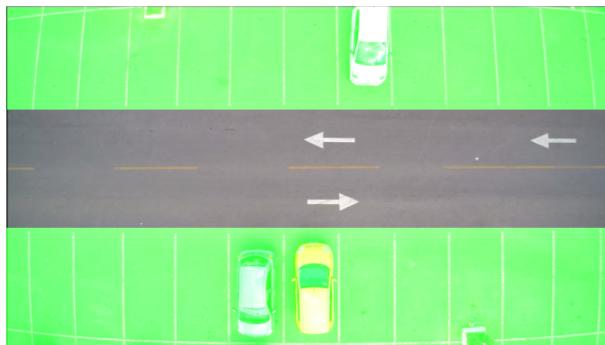


Figura 5.3: O mesmo quadro da Figura 5.2, depois de definidas as regiões de interesse, marcadas de verde.

## 5.2 Regiões de Interesse

No momento da instalação do *DVE*, é necessário definir um número qualquer de regiões de interesse(*Regions of interest* ou ROIs). Essas regiões determinam a área da imagem onde existem vagas. Além de determinar as regiões, deve-se informar ao programa o número de vagas existente em cada região de interesse.

As ROIs devem ser retangulares e determinadas de forma a não haver interseção entre as regiões, como exemplificado na Figura 5.3 que mostra as regiões determinadas para o quadro da Figura 5.2.

Depois de determinadas as regiões de interesse, cada uma das regiões é dividida em um número igual de seções verticais como ilustrado na Figura 5.4. Como as vagas não são determinadas individualmente no momento da instalação do *DVE*, é necessário que seja feita alguma divisão das ROIs. Cada uma destas seções verticais é classificada separadamente em uma etapa futura do processamento. O resultado da classificação de cada seção de uma ROI determina um vetor  $v$  de  $n$  elementos, onde  $n$  é o número de seções em que a região foi dividida e cada elemento indica a ocupação de uma seção, sendo o valor 1 correspondente a uma seção ocupada e o valor 2 correspondente a uma seção livre. Através da análise deste vetor é que o *DVE* determina o número de vagas livres em cada região. Munido do número de vagas que cada região contém e um valor aproximado do número de seções que um carro ocupa, o programa estima quantas vagas estão ocupadas, e encontra o número de vagas livres através de simples subtração e a posição aproximada destas vagas através da posição das seções livres no vetor.

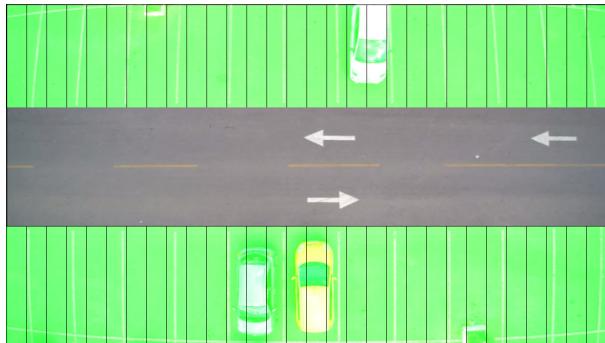


Figura 5.4: As ROIs determinadas na Figura 5.3 separadas em trinta seções verticais que ainda não foram classificadas. Nesta etapa os vetores  $v$  de cada região é composto de trinta valores 2.

Uma vez que as regiões de interesse foram definidas e divididas em seções, a etapa de configuração inicial é finalizada e o programa pode iniciar o processo de determinação da ocupação das vagas.

## 5.3 Classificação das seções

O primeiro passo na execução do *DVE* depois da configuração inicial é a classificação de cada uma das sessões verticais criadas na imagem. Essa classificação é feita assim que a execução começa, no primeiro quadro do vídeo e é repetida a cada segundo de execução do programa. Além disso, quando se detecta movimento em uma das ROIs as sessões interceptadas pelo movimento detectada são classificadas. O processo de detecção de movimento e determinação das sessões a serem classificadas excepcionalmente é detalhado na Seção 5.4.2.

Para que seja feita a extração das características e a classificação de uma seção, uma imagem  $I_s$  é extraída do quadro de vídeo original capturado. Essa imagem corresponde à fração do quadro contida dentro dos limites da seção a ser analisada. A extração de  $I_s$  consiste simplesmente em copiar os valores da matriz que representa o quadro que estavam dentro das bordas da seção em uma nova matriz de dimensões iguais às da seção. A Figura 5.5 mostra um exemplo de uma  $I_s$ . Cada uma dessas imagens é submetida separadamente ao processo de classificação.

### 5.3.1 Extração de características

O processo de classificação de uma seção começa com a extração das características que definem o padrão de cada classe e a criação do vetor de entrada da rede neural artifical. Dois conjuntos de características principais são extraídos: as quatro medidas extraídas da GLCM da imagem apresentadas na Seção 2.1.6 e dois valores que descrevem a crominância azul e vermelha da seção.

Para a extração da GLCM é preciso primeiro obter uma imagem de níveis de cinza que represente  $I_s$ . Para isso, a imagem é convertida para o espaço  $YCbCr$  como descrito na Seção 2.1.4. A imagem referente ao canal  $Y$  resultante é uma imagem de intensidade e por isso é utilizada para o cálculo da GLCM. Como mencionado na Seção 2.1.6, a construção



Figura 5.5: Uma imagem correspondente a uma das seções verticais definidas.

é feita com base na vizinhança a direita de cada *pixel* da imagem. Cada valor dentre os 256 possíveis é dividido dentro de 8 níveis distintos, e sempre que um valor presente em nível  $N$  aparece a direita de um valor de um nível  $M$ , o elemento  $(N, M)$  da GLCM é incrementado. A Figura 2.3 mostra esse processo.

Uma vez construída a GLCM referente a sessão, quatro medidas estatísticas são extraídas: contraste, correlação, energia e homogeneidade. Essas medidas são calculadas pelas Equações 2.3, 2.4, 2.5 e 2.6 respectivamente. Esses valores formam um vetor  $g = (\text{contraste}, \text{correlação}, \text{energia}, \text{homogeneidade})^T$  que é parte da entrada final da rede neural artificial. O vetor na Equação 5.1 representa o vetor  $g$  da Figura 5.5.

$$g = \begin{pmatrix} 0,2170 \\ 0,9208 \\ 0,2084 \\ 0,9019 \end{pmatrix} \quad (5.1)$$

As outras duas características extraídas de cada sessão vêm dos canais  $Cb$  e  $Cr$  de  $I_s$ . A média de valores da matriz de cada canal é calculada através da Equação 5.2. Onde  $N$  é o número total de *pixels* de  $I_s$  e  $v_i$  é o valor do  $i$ -ésimo *pixel*. Esses valores então formam o vetor  $c = (M_{Cb}, M_{Cr})^T$ .

$$M = \frac{\sum_{i=1}^N v_i}{N} \quad (5.2)$$

O vetor  $c$  da Figura 5.5 é:

$$c = \begin{pmatrix} 130,7157 \\ 127,0190 \end{pmatrix} \quad (5.3)$$

Essas características foram escolhidos por terem se mostrado suficientemente descriptivas e distintivas após uma análise de um conjunto de 90 imagens. As Figuras 5.6, 5.7, 5.8, 5.9, 5.10 e 5.11 mostram histogramas que descrevem a distribuição das características no conjunto de imagens. Em cada um dos gráficos, o eixo  $x$  corresponde a valores obtidos para característica em questão e o eixo  $y$  indica o número de imagens que exibiram aquele valor. As barras azuis são referentes as imagens de carros ou vagas ocupadas e as barras vermelhas referentes a vagas vazias.

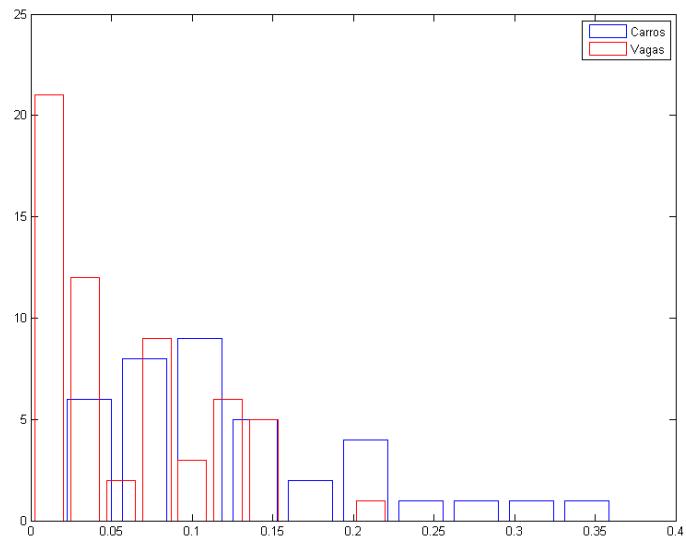


Figura 5.6: O histograma referente aos valores de contraste no conjunto analisado. Apesar de haver bastante sobreposição dos valores ainda é possível ver que há pouco ou nenhuma ocorrência de vagas após um certo valor.

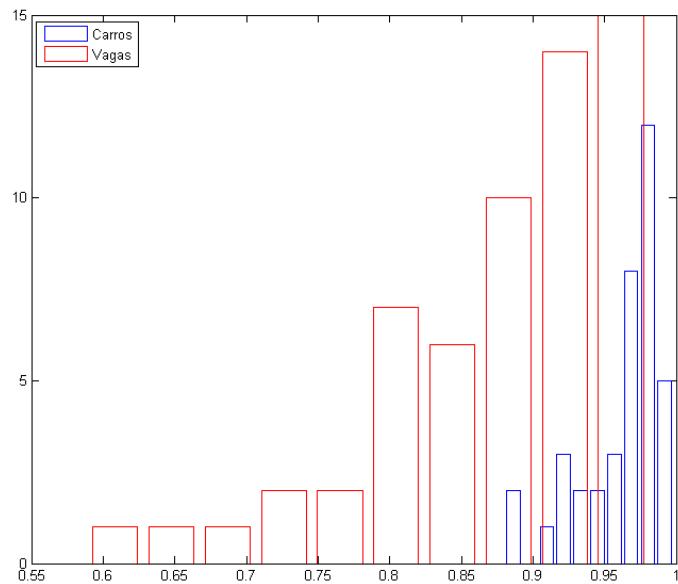


Figura 5.7: O histograma referente aos valores de correlação no conjunto analisado. Aqui é possível ver um limiar inferior para a classe dos carros. Valores abaixo deste limiar provavelmente são vagas livres.

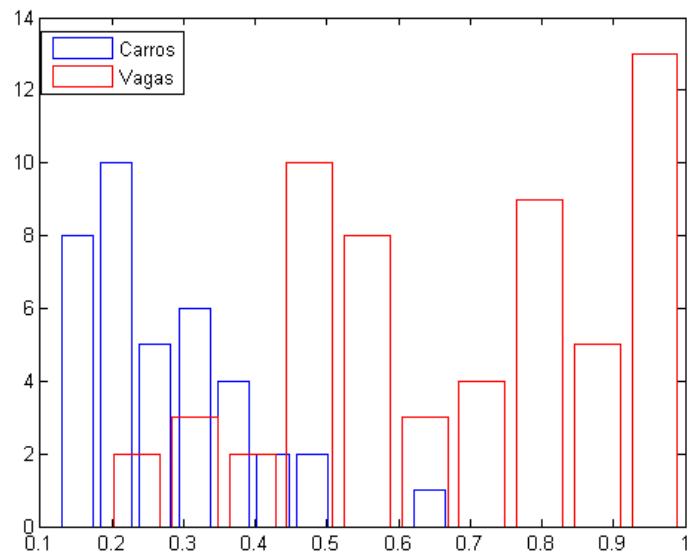


Figura 5.8: O histograma referente aos valores de energia no conjunto analisado. Há um ponto de divisão entre as duas classes, com pouca sobreposição de valores entre as classes.

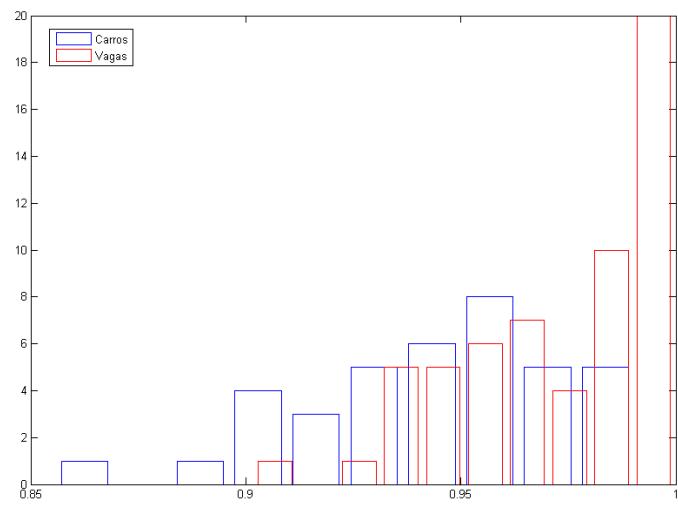


Figura 5.9: O histograma referente aos valores de homogeneidade no conjunto analisado. Esse gráfico mostra mais sobreposição do que os anteriores, mas ainda contém bastante informação sobre a classe das vagas.

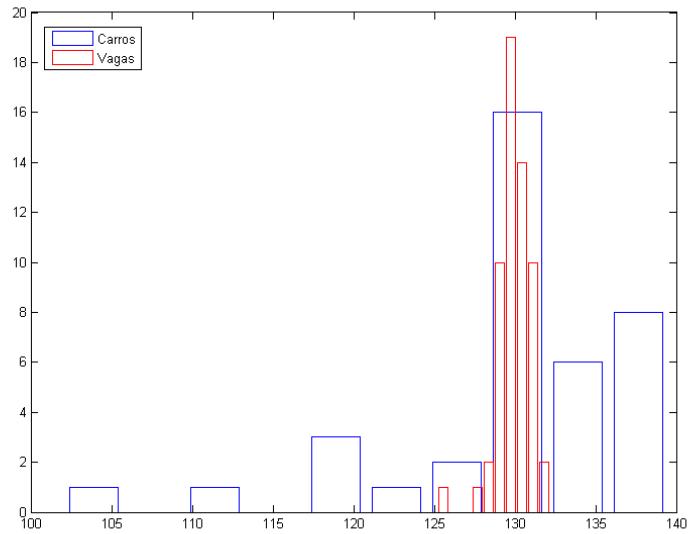


Figura 5.10: O histograma referente aos valores médios de crominância azul no conjunto analisado. Apesar de haver alta variedade nos valores encontrados nas imagens de carros, as imagens de faixa possuem valores médios em uma faixa estreita.

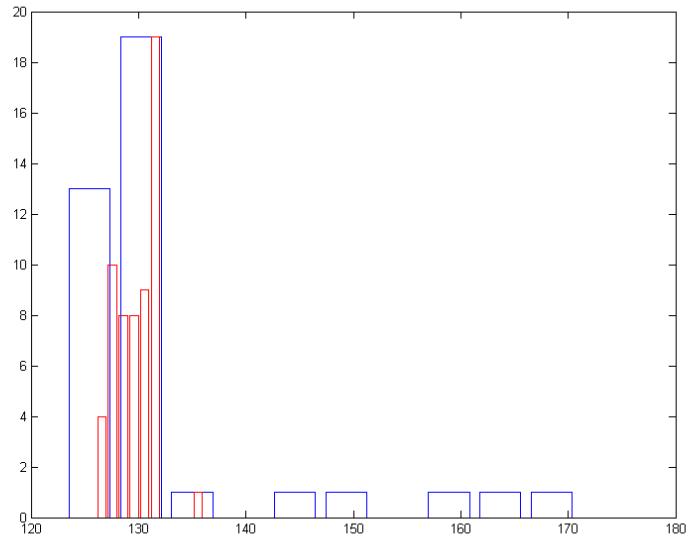


Figura 5.11: O histograma referente aos valores médios de crominância vermelha no conjunto analisado, mostrando comportamento semelhante ao histograma de valores médios de crominância azul.

### 5.3.2 Classificação da rede neural artificial

Depois de extraídas as características das imagens e construídos os vetores  $g$  e  $c$  com os valores das características, a entrada para a rede neural artificial é construída. O vetor de entrada  $x_s$  referente a seção a ser classificada é a concatenação de  $g$  e  $c$  de forma que:

$$x_s = (\text{contraste}, \text{correlação}, \text{energia}, \text{homogeneidade}, M_{Cb}, M_{Cr})^T \quad (5.4)$$

O vetor  $x_s$  é alimentado à rede neural artificial que retorna um vetor  $y_s$  de saída que possui dois elementos com valores reais entre 0 e 1. Esses valores sempre somam 1 e representam o grau de certeza que a rede tem de que a seção  $x_s$  representa um elemento da classe correspondente ao seu índice. Isto é, o primeiro elemento,  $y_s(1)$  representa o grau de certeza que a rede tem que a seção pertence à classe 1, os carros, e o segundo elemento,  $y_s(2)$  representa o grau de certeza que a rede tem que a seção pertence à classe 2, das vagas livres. O vetor  $y_s$  é armazenado e processado futuramente para decidir a classe final a que a seção pertence. Os vetores da Equação 5.5 correspondem aos vetores  $x_s$  e  $y_s$  da seção mostrada na Figura 5.5. Nesse exemplo o vetor  $y_s$  tem valor 1 no seu primeiro elemento, indicando certeza total da rede de que a imagem corresponde a uma seção ocupada por um veículo. É importante lembrar porém, que este não é sempre o caso e frequentemente o vetor de saída possui valores diferentes de 1 e 0.

$$x_s = \begin{pmatrix} 0,2170 \\ 0,9208 \\ 0,2084 \\ 0,9019 \\ 130,7157 \\ 127,0190 \end{pmatrix} \quad y_s = \begin{pmatrix} 1,0 \\ 0,0 \end{pmatrix} \quad (5.5)$$

### 5.3.3 Ajuste gaussiano dos vizinhos

O vetor  $y_s$  resultante da classificação da rede neural artificial não define a classe final da seção avaliada. Partindo do princípio que na maioria dos casos, uma seção vai possuir a mesma classe de suas vizinhas, esta informação também cumpre um papel na classificação de uma seção.

Depois de extraído o vetor  $y_s$  de uma seção vertical, o programa observa as suas duas vizinhas mais próximas a esquerda e a direita se existirem e para cada vizinha, calcula uma quantidade em que deve aumentar ou diminuir os valores de  $y_s$ . Uma seção vizinha sempre é responsável por aumentar o valor referente a sua classe e diminuir o outro valor. Isto é, se uma das vizinhas da seção avaliada é uma seção ocupada (classe 1), o valor de  $y_s(1)$  aumenta ligeiramente e o valor de  $y_s(2)$  diminui na mesma proporção.

A influência de cada vizinha nos valores de  $y_s$  da seção sendo classificada é definida pela sua distância no vetor de seções e ilustrada aproximadamente pela Figura 5.12. Para compreender de forma mais clara, considere a posição de cada seção como o seu índice no vetor  $v$  de todas as seções. Se estamos avaliando a seção de índice 12, as seções de índice 11 e 10 são as duas vizinhas a esquerda avaliadas e as seções de índice 13 e 14 são as suas vizinhas a direita avaliadas. A quantidade da influência de cada vizinha é definida pela Equação 5.6, onde  $\sigma = 0,5$ ,  $i_s$  é o índice da seção sendo classificada e  $i_v$  é o índice da vizinha. Note que essa equação representa o valor de uma curva gaussiana calculada

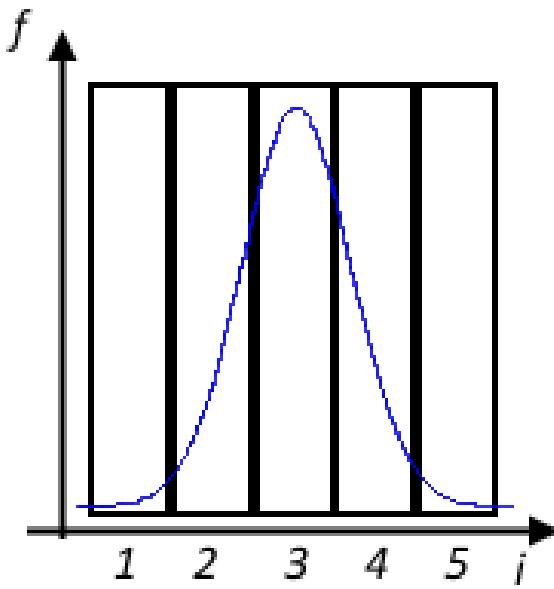


Figura 5.12: Uma ilustração simplificada da influência que uma seção exerce sobre a classificação de suas vizinhas.

no ponto  $i_s$  com média igual a  $i_v$  e desvio padrão igual a  $\sigma$ . Se chamarmos o valor de  $y_s$  referente a classe da vizinha de  $c_v$  e o outro valor de  $\bar{c}_v$ , cada vizinha modifica  $y_s$  de acordo com as Equações 5.7 e 5.8.

$$f(s, v) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{(i_s - i_v)^2}{2\sigma^2}} \quad (5.6)$$

$$c_v = c_v(1 + f(s, v)) \quad (5.7)$$

$$\bar{c}_v = \bar{c}_v(1 - f(s, v)) \quad (5.8)$$

O propósito desta etapa é evitar possíveis erros de classificação que podem ocorrer por pequenas anomalias na imagem da seção que possam confundir a rede e principalmente ajudar a classificar seções como as da Figura 5.13 que possuem partes que representam veículos e outras partes vazias. Como um benefício adicional, esse ajuste gaussiano diminui a chance de o DVE não detectar que uma vaga foi ocupada por que a seção central ocupada pelo carro foi classificada incorretamente, como ilustrado na Figura 5.14, uma vez que é comum que uma das seções adjacentes seja classificada como ocupada e colabore com a correção da classificação da seção central.

### 5.3.4 Sistema de votação

Além do sistema de classificação momentânea, o programa integra um componente que determina a classe majoritária de uma seção durante um intervalo de tempo. Para o DVE uma seção pode estar em um de dois estados: estabilizada ou não-estabilizada. No começo da execução do programa, nenhuma seção está estabilizada.



Figura 5.13: Uma imagem correspondente a uma seção dividida entre carro e vaga vazia, cuja classificação se beneficia do ajuste pelas vizinhas.

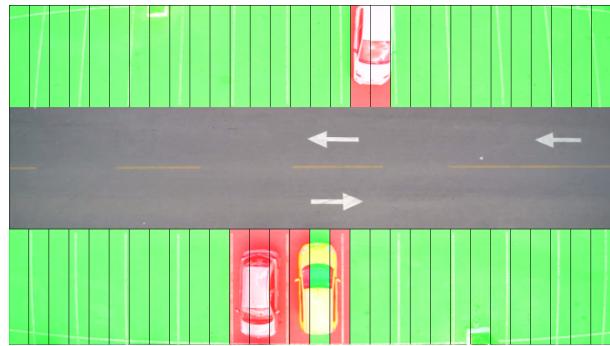


Figura 5.14: Um tipo de erro que o ajuste gaussiano ajuda a corrigir. As seções vermelhas são ocupadas e as verdes são livres.

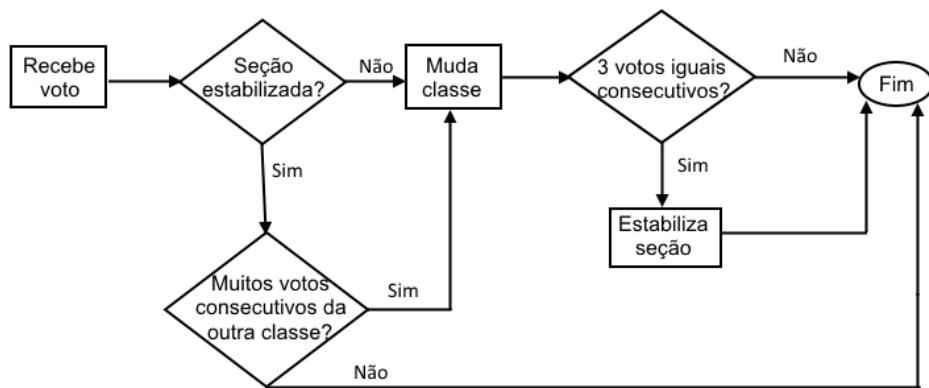


Figura 5.15: O fluxograma de funcionamento do sistema de votação.

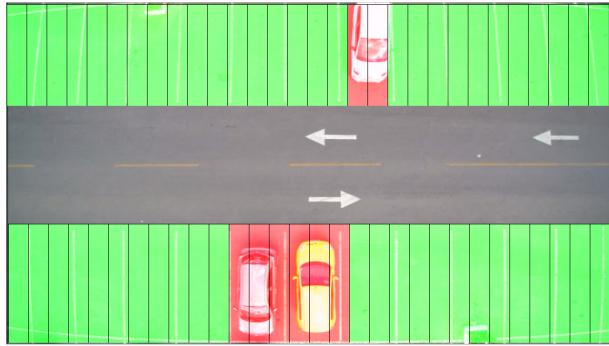


Figura 5.16: Um exemplo de quadro com as seções devidamente classificadas, as seções vermelhas representam espaço ocupados por carros, enquanto as verdes representam espaços livres.

Depois de determinar o vetor  $y_s$ , o sistema determina um voto definido pela Equação 5.9. Se a seção não estiver estabilizada, sua classe se torna a classe representada pelo voto. Se uma seção receber 3 votos iguais consecutivamente, a seção se torna estabilizada. Nesse estado, a sua classe não muda mais, mas a seção continua a receber votos. Uma seção deixa de estar estabilizada e volta a poder ser reavaliada se receber um número suficientes de votos consecutivos da classe diferente da sua classe atual ou, mais comumente, quando o *DVE* detecta que houve movimento dentro da área de interesse que intercepta a seção. Esse processo está ilustrado no fluxograma da Figura 5.15.

Essa etapa do processamento tem o propósito de evitar que flutuações nas imagens das seções causem mudanças frequentes na sua classificação e fazer com que na maioria dos casos a classe de uma seção só mude quando houver movimento detectado na região que ocupada pela seção, trazendo estabilidade para os resultados do programa.

$$voto = \begin{cases} 1, & \text{se } y_s(1) \geq 0,6 \\ 2, & \text{caso contrário} \end{cases} \quad (5.9)$$

Ao final do processo de classificação, o vetor  $v$  de seções é representado visualmente em uma imagem semelhante a Figura 5.16, onde as seções vermelhas são da classe 1 e representam veículos e as seções verdes são da classe 2 e representam espaços desocupados.

## 5.4 Extração do movimento

Outra ferramenta utilizada pelo *DVE* para a análise do vídeo capturado é a extração do movimento que ocorre na imagem. A cada quadro o programa detecta o movimento ocorrido desde o quadro anterior e processa essa informação para determinar se houve movimento de veículos em alguma das regiões de interesse delimitadas. Mais especificamente, o programa está interessado nos momentos quando um objeto que antes se movimentava termina seu movimento em uma região de interesse ou quando um objeto que iniciou seu movimento dentro de uma ROI finaliza o movimento. Essas situações são entendidas como um veículo estacionando em uma vaga e um veículo saíndo de uma vaga respectivamente.

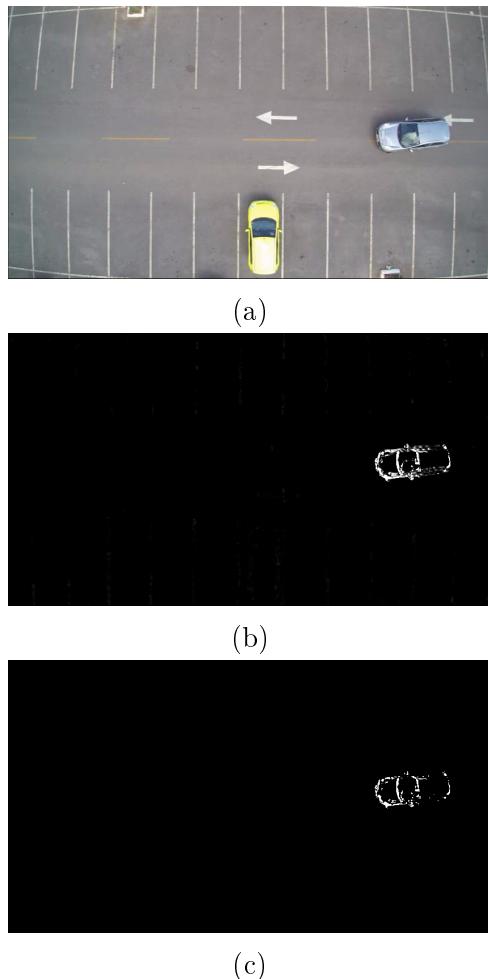


Figura 5.17: Ilustração do processo de extração do movimento significativo no vídeo. (a) mostra o quadro analisado, (b) uma representação visual da matriz de magnitudes das velocidades e (c) uma representação da matriz depois de limiarizada.

### 5.4.1 Magnitudes do movimento

O movimento é detectado através do método de cálculo do fluxo óptico utilizando o método de Lucas-Kanade [9–11] para resolver a Equação 2.8. O resultado deste processo é uma matriz  $M$  de dimensões iguais as do quadro onde cada elemento corresponde à magnitude do vetor velocidade do seu movimento. A direção e o sentido do vetor velocidade não são de interesse do *DVE*, uma vez só os pontos de início e fim do movimento são relevantes para o processamento. A matriz  $M$  então passa por um processo de limiarização, onde cada elemento de magnitude menor do que um limiar  $t$  tem seu valor redefinido como 0. O resultado deste processo é uma matriz que mostra as posições do quadro aonde ocorreu movimento significativo.

### 5.4.2 Regiões de movimento

Uma vez que a matriz  $M$  de magnitudes das velocidades do movimento foi limiarizada, a etapa seguinte é determinar as regiões do quadro que correspondem ao objeto em

movimento. Essas regiões são determinadas encontrando pontos na matriz  $M$  que correspondem a estimativas dos centros dos objetos móveis, e a partir desses pontos construir um retângulos que representam os limites superiores e inferiores das áreas que os objetos ocupam no quadro.

$$C = \begin{pmatrix} X_c \\ Y_c \end{pmatrix}, \text{ onde } X_c = \frac{\sum_i x_i v_i}{\sum_i v_i} \text{ e } Y_c = \frac{\sum_i y_i v_i}{\sum_i v_i} \quad (5.10)$$

Para se encontrar esses centros, o *DVE* começa separando um vetor  $B$  que contém as coordenadas dos pontos diferentes de 0 de  $M$  e os valores destes pontos. A posição do centro de cada objeto em movimento é um vetor de dois elementos onde cada elemento é a média das coordenadas correspondentes dos pontos que compõe o objeto ponderadas pelos valores dos pontos. A Equação 5.10 representa esse cálculo, onde  $C$  é a posição do centro e  $x_i, y_i$  e  $v_i$  as coordenadas e valores dos pontos que compõe o objeto. A construção dos conjuntos de pontos que representam cada objeto em movimento é feita seguindo os passos abaixo.

1. Para cada ponto de  $B$  faça:
  - Verifique se o ponto está próximo o suficiente de um centro já encontrado através da Equação 5.11 de distância.
  - Caso esteja, associe esse ponto ao conjunto de pontos do objeto e recalcule o centro usando a Equação 5.10.
  - Caso contrário, inicie um novo conjunto de pontos referente a um novo objeto. O centro deste novo conjunto é o ponto.
2. Percorra cada conjunto criado no passo 1 faça e una todos os pares de conjuntos que possuem centros suficientemente próximos de acordo com a Equação 5.11;
3. Exclua todos os conjuntos com um número muito pequeno de elementos.
4. Para cada conjunto restante, determine os elementos com menores e maiores valores nas coordenadas  $x$  e  $y$  em  $B$ . Esses pontos representam os limites inferiores e superiores do espaço que cada objeto ocupa no quadro.

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (5.11)$$

Munido dos limites da área ocupada por cada objeto em movimento na cena, o programa constrói retângulos que representam uma estimativa da posição desses objetos no quadro. Se  $x_{min}$ ,  $x_{max}$ ,  $y_{min}$  e  $y_{max}$  são, respectivamente, os valores das menores e maiores coordenadas  $x$  e  $y$  de um objeto, um retângulo na coordenada  $(x_{min}, y_{min})$  de largura  $x_{max} - x_{min}$  e altura  $y_{max} - y_{min}$  é criado. A Figura 5.18 mostra um exemplo de uma destes retângulos, representado em azul.

O programa acompanha a posição destes retângulos para determinar o comportamento dos objetos em movimento. O *DVE* armazena a posição inicial de cada retângulo e a sua posição atual. Quando ocorre de um quadro possuir um número menor de retângulos que o anterior, isso significa que o movimento de um objeto se encerrou. Nesse momento, o programa identifica se o início ou o final do movimento encerrado estavam dentro de uma

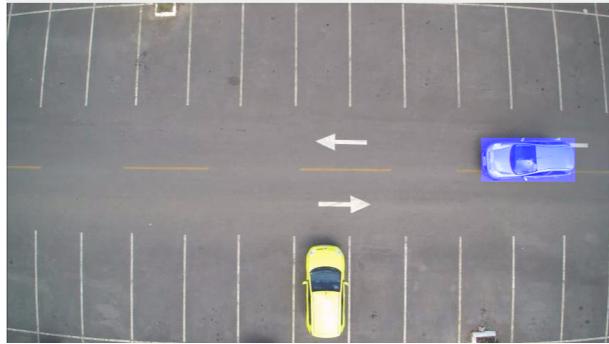


Figura 5.18: Um retângulo que representa a estimativa da área ocupada por um objeto em movimento, representado em azul.

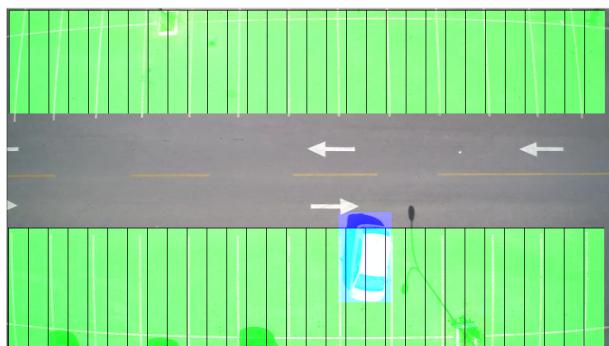


Figura 5.19: O retângulo em azul intereceptando algumas seções verticais. As seções tocadas pelo retângulo serão reclassificadas ao final do movimento.

ROI. Se o movimento tiver terminado dentro de uma ROI, as seções interceptadas pelo retângulo referente a sua última posição são reclassificadas. Se o movimento tiver iniciado dentro de uma ROI o mesmo ocorre para as seções interceptadas pelo retângulo referente ao início do movimento. A Figura 5.19 ilustra o retângulo interceptando as seções.

## 5.5 Mapeamento de Vagas

Para determinar a quantidade e a posição aproximada das vagas, o programa faz um mapeamento do estacionamento através dos movimentos capturados. Quando um movimento começa ou termina dentro de uma ROI, além de marcar as seções contidas na região deste movimento como não-estabilizadas da forma descrita na Seção 2.2, o programa também determina que estas seções compõem uma vaga. Com o tempo, o programa terá determinado quais conjuntos de seções determinam as vagas do estacionamento.

As vagas determinadas desta forma não são estáticas. Na medida que veículos ocupam as seções verticais, as posições das vagas se modificam de acordo com a nova informação. Por exemplo, se uma área de movimento termina sobre duas seções que compõem uma vaga e uma terceira ainda não designada, o programa entende que todas as três seções compõem a mesma vaga. De forma semelhante, se as seções interceptadas por um movimento novo são um subconjunto das seções de uma vaga, estas seções passam a compor uma nova vaga, enquanto a vaga que existia antes passa a ser composta pelas seções res-

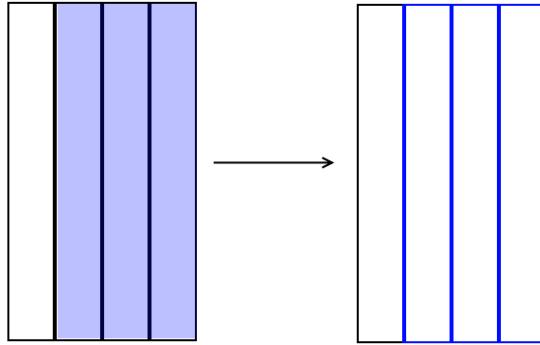


Figura 5.20: No quadro a esquerda, um movimento representado pelo retângulo azul intercepta três seções que pertencem a vaga com os contornos pretos. As seções interceptadas passam a compor uma nova vaga, marcada de azul, enquanto a vaga marcada de preto agora é composta apenas da seção restante.

tantes. Esse dinamismo permite que erros no mapeamento causados, por exemplo, por veículos mal estacionados ou por alguma falha no próprio programa sejam naturalmente corrigidos com o passar do tempo.

O programa determina que uma vaga foi ocupada se a maioria das seções que a compõem estão classificadas como ocupadas. Por isso, é importante que nenhuma seção pertença a mais de uma vaga. Se isso acontecer, o programa considera a ocupação desta seção mais de uma vez, causando erros na contagem das vagas ocupadas. Além disso, é importante que as seções que compõe uma vaga refletem o mais corretamente possível, evitando principalmente que duas vagas distintas estejam marcadas na região de uma única vaga.

Com esses critérios em mente, dois métodos de mapeamento foram desenvolvidos: o modo simples e o modo complexo. No modo simples o programa designa todas as seções interceptadas por um movimento a uma nova vaga, mesmo que essas seções já pertencessem a alguma outra vaga. Apesar de garantir a que nenhuma seção fará parte de duas vagas simultaneamente, essa simplificação traz a desvantagem de ocasionalmente determinar vagas que não refletem corretamente os movimentos. A mais comum destas eventualidades é a criação de vagas compostas de apenas uma seção, exemplificada na Figura 5.20. Essas vagas oferecem um risco de erro, uma vez que basta que a seção seja classificada como ocupada para que a vaga também o seja, diminuindo a precisão da consideração. Porém, um novo movimento que intercepte esta seção normalmente a integra em uma vaga que melhor reflete a configuração do estacionamento. Este tipo de situação de erro então tem uma probabilidade muito maior de corrigir com o passar do tempo do que uma seção que pertença a duas vagas, além de menos possibilidade de causarem erros na contagem de vagas ocupadas, já que comumente estas seções estão presentes em regiões desocupadas. Esse método é mais adequado quando o programa está executando com um número elevado de seções verticais. Além deste modo exibir melhor performance, o número elevado de seções ajuda a amenizar o problema mencionado.

O modo complexo por outro lado segue um conjunto de regras mais elaborado para determinar as vagas a que cada seção pertence. Quando um movimento intercepta um conjunto de seções que chamaremos de  $S$ , novas vagas são formadas de acordo com os

seguintes critérios:

- Caso nenhuma seção de  $S$  esteja associada a alguma vaga,  $S$  passa a compor uma nova vaga.
- Caso as seções de uma certa vaga  $V$  sejam um subconjunto de  $S$ ,  $V$  passa a ser composto por  $S$ .
- Caso  $S$  seja um subconjunto de uma vaga  $V$ , uma nova vaga com as seções de  $S$  é criada. Caso as seções de  $V$  que não pertenciam a  $S$  formem um conjunto contínuo com número de seções suficientemente grande, uma segunda vaga  $V_2$  é criada.
- Caso a intersecção entre as seções de  $S$  e de uma outra vaga  $V$  seja não-nula e nenhum dos casos anteriores ocorra, as seções da intersecção são divididas entre as duas vagas. Se houver um número ímpar de seções a dividir,  $S$  recebe a seção sobresalente.

Esse modo de mapeamento não garante naturalmente que as seções pertençam a uma única vaga, apesar da ocorrência ser rara. Porém, as vagas criadas através deste modo representam melhor a configuração real do estacionamento. Ao contrário do método simples, este modo de mapeamento é mais adequado quando o número de seções verticais em cada área de interesse é menor.

Independentemente do modo escolhido para a execução, o programa determina as vagas no momento inicial do vídeo de forma diferente do que durante o resto da execução. Como neste momento não há movimento algum, as posições das vagas são determinadas apenas pela ocupação das seções verticais. Cada conjunto de seções verticais consecutivas classificadas como ocupadas configuram uma vaga.

# Capítulo 6

## Resultados Experimentais

### 6.1 Aquisição de dados

Para os experimentos realizados neste trabalho foi capturado um conjunto de vídeos no estacionamento do pavilhão João Calmon na Universidade de Brasília. Foram feitas duas seções de filmagens contínua, aonde três veículos se deslocavam pelo estacionamento e ocupavam vagas escolhidas arbitrariamente. O primeiro vídeo capturado foi utilizado para o treinamento da rede. O segundo vídeo foi dividido em oito vídeos de menor duração sobre os quais foram realizados os testes.

As filmagens foram realizadas por um drone modelo *Yuneec Typhoon Q500+* (Figura 6.1). O drone foi controlado para que pairasse no ar em uma altura semelhante a de um poste de luz, a fim de capturar imagens de forma mais próxima possível de uma câmera de vídeo instalada em um poste de luz.

Para a aquisição das imagens utilizadas para o treinamento da rede, o primeiro vídeo passou por um processo semelhante ao processo de configuração inicial do programa final descrito no Capítulo 5. Duas áreas de interesse foram determinadas de forma idêntica ao processo de escolha de ROIs do *DVE*. Em seguida cada ROI foi dividida em trinta seções verticais. Três quadros do vídeo foram escolhidos de forma aleatória. As imagens correspondentes a cada seção de cada quadro escolhido foram extraídas em arquivos separados. Dessa maneira, as imagens utilizadas para o treinamento da rede neural foram construídas da mesma maneira que as imagens que seriam alimentadas a rede para classificação no futuro.



Figura 6.1: O drone utilizado para a gravação dos vídeos de teste.



Figura 6.2: (a) Um exemplo de imagem classificada manualmente na classe 1.(b) Um exemplo de imagem classificada manualmente na classe 2.

## 6.2 Treinamento da rede neural artificial

Uma vez adquiridas as imagens a serem utilizadas para o treinamento, cada uma das imagens foi separada manualmente em uma das duas categorias: veículo ou vaga vazia. Ao todo foram utilizadas 57 imagens classificadas como veículo(categoria 1) e 59 imagens classificadas como vaga vazia(categoria 2) totalizando 116 imagens. Foram extraídas as características de cada imagem da forma descrita na Seção 5.3.1. Os vetores  $x_s$  obtidos foram então concatenados de forma a criar duas matrizes. A primeira de dimensões  $6 \times 57$  representava as características das imagens de carros e a segunda de dimensões  $6 \times 59$  as características das imagens de vagas desocupadas.

Para que pudesse ser feito o treinamento supervisionado da rede neural, duas matrizes de alvos foram construídas. Uma com dimensões  $2 \times 57$  e a outra com dimensões  $2 \times 59$ . A primeira matriz com todas as colunas iguais a  $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$  e a segunda com as colunas da forma  $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ . Essas matrizes vão compor o gabarito utilizado para o treinamento supervisionado da rede.

As matrizes de características de cada classe foram então concatenadas formando uma matriz de entrada  $In_{6 \times 116}$ . O mesmo foi feito com as matrizes do gabarito, resultando na matriz  $Target_{2 \times 116}$ . Em seguida as colunas dessas duas matrizes foram reorganizadas aleatoriamente, de forma que as mudanças feitas em  $In$  fossem refletidas em  $Out$ , garantindo que não fosse perdida a relação entre as colunas de mesmo índice das matrizes. O resultado final deste processo é que cada coluna de  $Target_{2 \times 116}$  indica a classe da coluna correspondente de  $In_{6 \times 116}$ .

Uma vez criadas essas duas matrizes, as entradas e seus respectivos alvos são separados em três conjuntos: o conjunto de treinamento, de validação e de testes. A divisão é feita de forma que 70% das entradas são designadas ao conjunto de treinamento, 15% designadas ao conjunto de validação e os 15% restantes ao conjunto de testes. O conjunto de treinamento então é composto por duas matriz  $Ti_{6 \times 81}$  e  $To_{2 \times 81}$  que são iguais as primeira 81 colunas de  $In_{6 \times 116}$  e  $Target_{2 \times 116}$  e representam os vetores descritores das entradas e seus gabaritos respectivamente. Os outros dois conjuntos são construídos de forma similar utilizando. O conjunto de validação é composto por matrizes de 18 colunas e o de teste por matrizes de 17 colunas.

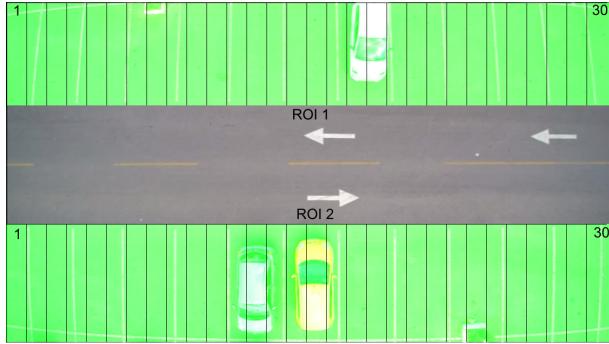


Figura 6.3: Figura mostrando a divisão das ROIs e numeração das seções nos vídeos mostrados para os observadores humanos.

A rede neural utilizada foi uma rede *feed-forward* com três camadas. A camada oculta possui 15 neurônios com função de ativação logística (Equação 3.3) e a camada de saída 2 neurônios com função de ativação *softmax* (Equação 3.4). A rede é submetida a um treinamento supervisionado com um limite de 1000 iterações. Quando a situação de convergência é atingida, o treinamento para e o resultado é a rede final utilizada.

### 6.3 Resultados Obtidos

Para determinar a capacidade do *DVE* de determinar se as seções verticais estão ocupadas ou livres, os resultados obtidos pelo programa foram comparados com os resultados que eram esperados de observadores humanos. Foram apresentados a três observadores, que chamaremos pelas iniciais F, M e P, um conjunto de oito vídeos que mostravam veículos estacionando ou saindo de vagas em um estacionamento descoberto. As áreas de interesse do vídeo foram definidas previamente e divididas em 30 seções verticais. Dessa forma, todos os observadores e o *DVE* analisaram seções verticais idênticas.

Cada um dos observadores receberam as seguintes instruções:

- As regiões de interesse e as seções são numeradas como indicado na Figura 6.3.
- Uma seção ocupada é aquela cuja maior parte de sua área está ocupada por um veículo.
- No momento inicial do vídeo (0 segundos), indique quais seções verticais estão ocupadas através do número das ROI e das seções, as outras serão assumidas como livres. Indique também o número de vagas ocupadas.
- Se a qualquer momento o estado de ocupação de uma seção mudar, indique o tempo da mudança, a seção onde ocorreu a mudança e a natureza da mudança (ocupada ou liberada).

Cada observador assistiu aos vídeos sozinho, sem interferência externa e sem conhecimento dos resultados do programa. De fato, a fim de evitar interferência, o *DVE* só foi executado sobre os casos de teste após todos os observadores escolhidos terem entregado os resultados que obteram.

Para os testes apresentados a seguir, serão observadas somente aquelas seções que o *DVE* ou os observadores determinaram como ocupadas. As demais seções serão consideradas livres. Chamaremos de um *acerto* sempre que em um determinado segundo  $t$ , o programa e um observador concordam quanto a ocupação de uma seção vertical. Sendo assim, o programa pode atingir um máximo de 60 acertos por segundo. A indicação da ocupação das seções pelos observadores não foi definida a cada segundo, mas considera-se que entre duas acusações de mudança de estado a ocupação das seções permanece a mesma. Sendo assim, podemos utilizar esses momentos para definir a ocupação em cada segundo do vídeo. Por exemplo, se um observador disser que a seção 10 estava ocupada no momento inicial do vídeo e depois indicar que a mesma sessão foi liberada aos 8s, a seção será considerada ocupada durante todos os momentos deste intervalo.

Além da capacidade de classificar cada seção, a precisão do *DVE* em determinar o número de vagas ocupadas a cada momento do vídeo também foi analisada. Para este teste, os observadores foram instruídos a informar o número de vagas ocupadas no início do vídeo. Sempre que considerassem que houve uma mudança neste número, deveriam informar o momento da mudança e o novo número de vagas ocupadas. Para esta análise, um *acerto* é considerado a cada segundo que o programa indica o mesmo número de vagas ocupadas que os observadores. Para a determinação das vagas o programa foi executado no modo complexo de mapeamento.

Nas seções seguintes serão apresentados cada um dos vídeos utilizados para os testes. Cada caso iniciará com uma breve descrição do movimento dos veículos no vídeo, a duração do vídeo e o número de acertos possíveis para as seções verticais, seguidos de cinco tabelas, uma para cada observador e duas para o *DVE*. As tabelas indicam o tempo onde foram acusadas mudanças de estado nas seções e no número de vagas ocupadas. A primeira linha da tabela indica as seções ocupadas no momento inicial do vídeo e cada linha subsequente indica um momento em segundos quando houve mudança de estado de pelo menos uma seção ou vaga, a ROI e o número das seções modificadas e o número de vagas ocupadas naquele momento no tempo. Finalmente, será calculada uma taxa de acerto que compara o desempenho do programa na classificação a cada observador e uma taxa final de acerto média para o caso de teste. A taxa de acerto para cada observador é calculada pela razão entre o número de acertos do programa e o número de acertos possíveis para o caso de teste e a taxa de acerto média é a média aritmética entre estes valores. Para que o leitor possa comparar o desempenho do *DVE* com suas próprias impressões, uma imagem dos momentos iniciais e finais de cada vídeo será mostrada junto de cada análise.

A leitura das tabelas que representam as taxas de acerto devem ser feitas com um certo cuidado. Por causa da grande quantidade de seções e de possíveis acertos, um erro na classificação de uma seção representa uma mudança pequena na taxa de acerto. Em cada caso de testes há um grande número de seções aonde não há movimento algum, que são classificadas corretamente como vazias, elevando a taxa de acertos do teste. Enquanto esses acertos são relevantes, a classificação correta das seções onde há movimento de veículos representa uma diferença muito maior no funcionamento do *DVE*.

Por outro lado, a taxas de acerto das ocupações das vagas são muito mais afetadas por erros. Se o *DVE* indicar uma vaga ocupada a mais que um observador por apenas 1 segundo, um erro quase insignificante em termos humanos, isso pode representar uma diminuição de até 10% na taxa de acerto.

Por isso, as taxas de acerto não são fatores exclusivos na determinação da eficácia do

programa. Assim, uma análise rápida do comportamento do *DVE* em cada caso de testes será apresentada após as tabelas.

### 6.3.1 Vídeo 1

No primeiro caso de testes, o vídeo começa com um estacionamento vazio. Depois de alguns segundos um único veículo de cor branca entra na cena pela direita e estaciona em uma vaga na parte inferior da imagem. O vídeo tem 15 segundos de duração, totalizando 900 acertos possíveis.

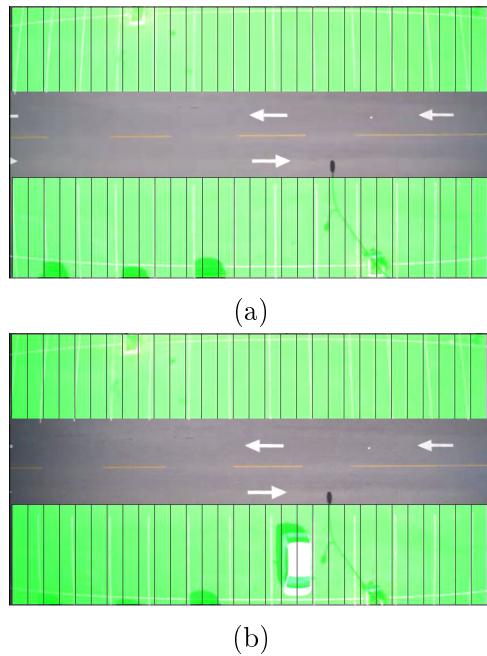


Figura 6.4: (a) O momento inicial do vídeo 1; (b) O momento final do vídeo 1.

Observador F		
Tempo(s)	Acontecimento	Vagas ocupadas
0	Nenhuma seção ocupada	0
10	ROI 2: Seções 18 e 19 ocupadas.	1

Observador P		
Tempo(s)	Acontecimento	Vagas ocupadas
0	Nenhuma seção ocupada	0
10	ROI 2: Seções 18 e 19 ocupadas.	1

Observador M		
Tempo(s)	Acontecimento	Vagas ocupadas
0	Nenhuma seção ocupada	0
9	ROI 2: Seções 18 e 19 ocupadas.	1

DVE		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seção 3 ocupada.	1
10	ROI 2: Seções 18 e 19 ocupadas.	2
14	ROI 2: Seção 3 liberada.	1

Acertos - seções		
Observador	Acertos	Taxa de acertos
F	886	98,44%
P	886	98,44%
M	884	98,22%
Média	885,3	98,37%

Acertos - vagas		
Observador	Acertos	Taxa de acertos
F	1	7%
P	1	7%
M	1	7%
Média	1	7%

Neste caso de testes, o programa classifica errôneamente a seção 3 da ROI 2 durante quase toda a duração do vídeo. Como o erro se inicia logo no primeiro momento do vídeo, esta seção é definida como uma vaga que conta como ocupada enquanto o erro de classificação persiste, fazendo com que o programa indique um número de vagas ocupadas errado durante toda a sua execução. Esse caso ilustra uma possível consequência de um erro de classificação. Apesar do erro, o reconhecimento da vaga ocupada pelo veículo nas seções 18 e 19 ocorre sem problemas.

### 6.3.2 Vídeo 2

Neste vídeo, um carro branco está estacionado no conjunto inferior de vagas. Um veículo cinza entra pela direita e estaciona no conjunto superior de vagas. O vídeo tem uma duração de 10 segundos, totalizando 600 acertos possíveis.

Observador F		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 19 e 20 ocupadas	1
6	ROI 1: Seções 21,22 e 23 ocupadas.	2

Observador P		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 19 e 20 ocupadas	1
6	ROI 1: Seções 20,21,22 ocupadas.	2

Observador M		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 19 e 20 ocupadas	1
6	ROI 1: Seções 21,22 e 23 ocupadas.	2

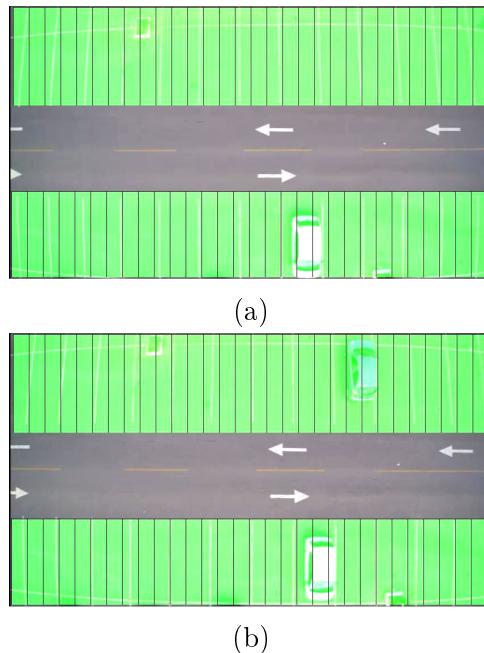


Figura 6.5: (a) O momento inicial do vídeo 2; (b) O momento final do vídeo 2.

DVE		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 18,19 e 20 ocupadas.	1
6	ROI 1: Seções 21 e 22 ocupadas.	2

Acertos - seções		
Observador	Acertos	Taxa de acertos
F	586	97,66%
P	586	97,66%
M	586	97,66%
Média	586	97,66%

Acertos - vagas		
Observador	Acertos	Taxa de acertos
F	10	100%
P	10	100%
M	10	100%
Média	10	100%

Neste caso de testes, houve discordância entre os observadores sobre as seções ocupadas pelo carro cinza na parte superior do vídeo. Porém o *DVE* classificou como ocupadas as seções onde houve consenso entre os observadores.

O programa também classificou errôneamente a seção 18 do ROI 2 no momento inicial do vídeo. Novamente o *DVE* acerta nas seções onde há consenso entre os observadores. O erro é mais aceitável que o erro que ocorreu no vídeo 1, pois a determinação da vaga ainda ocorre de maneira que a sua ocupação é reconhecida.

### 6.3.3 Vídeo 3

Neste vídeo, dois carros se encontram no estacionamento. Um veículo amarelo entra pela direita e estaciona na seção superior das vagas. O veículo branco sai pela parte de baixo da tela. O vídeo tem duração de 20 segundos totalizando 1200 acertos possíveis.

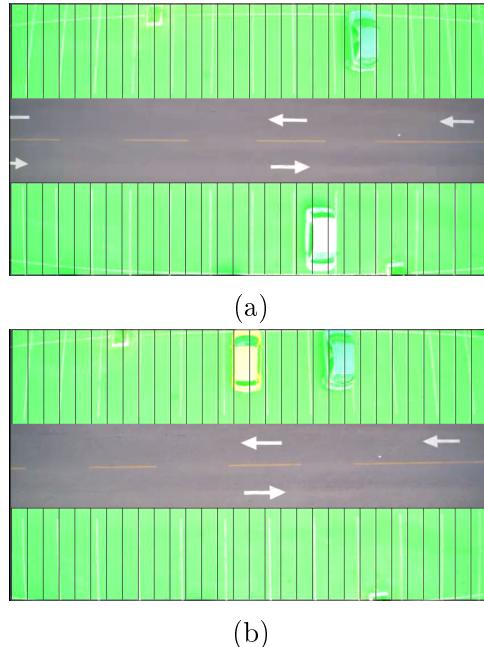


Figura 6.6: (a) O momento inicial do vídeo 3; (b) O momento final do vídeo 3.

Observador F		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 22 e 23 ocupadas. ROI 2: 19,20 e 21 ocupadas	2
8	ROI 1: Seções 16 e 17 ocupadas.	3
13	ROI 2: Seções 18 e 19 liberadas.	2

Observador P		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 22 e 23 ocupadas. ROI 2: 20 e 21 ocupadas	2
9	ROI 1: Seções 15,16 e 17 ocupadas.	3

Observador M		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 22 e 23 ocupadas. ROI 2: 20 e 21 ocupadas	2
8	ROI 1: Seções 15,16 e 17 ocupadas.	3
14	ROI 2: Seções 17,18 e 19 liberadas	2

DVE		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 22 e 23 ocupadas. ROI 2: 19,20 e 21 ocupadas	2
1	ROI 1: Seção 23 liberada.	2
2	ROI 1: Seção 23 ocupada.	2
4	ROI 1: Seção 23 liberada.	2
8	ROI 1: Seções 16 e 17 ocupadas.	3
9	ROI 2: Seção 21 liberada. ROI 2: Seções 18 e 24 ocupadas	3
10	Sem acontecimentos	2
11	ROI 1: Seção 20 ocupada. ROI 2: Seção 24 liberada.	3
13	ROI 1: Seção 20 liberada. ROI 2: Seção 18,19 e 20 liberada.	2
16	ROI 1: Seção 20 ocupada.	2
19	ROI 1: Seção 20 liberada.	2

Observador	Acertos	Taxa de acertos
F	1165	97,08%
P	1113	92,75%
M	1136	94,66%
Média	1138	94,83%

Acertos - vagas		
Observador	Acertos	Taxa de acertos
F	19	95%
P	11	55%
M	18	90%
Média	16	80%

Esse caso de teste possui alguns problemas causados principalmente por causa da movimentação do *drone* no momento da gravação. Essa movimentação faz com que partes da imagem passem a ocupar seções diferentes, criando algumas inconsistências. Repare que por volta dos 14s dois dos observadores disseram que a seção 18 da ROI 2 foi liberada, apesar de nunca terem acusado que a seção estava ocupada anteriormente. Essa inconsistência ocorre porque entre o início do vídeo esse momento, o *drone* utilizado para a gravação se movimenta e faz com que o veículo que estava ocupando as seções 19 e 20 ocupe as seções 18 e 19. Esse problema acontece em outros casos de teste em menor proporção.

O teste sobre esse vídeo ainda possui resultados interessantes e úteis. Uma vantagem do uso de um sistema automatizado se mostra neste caso de teste. O observador *P* não percebeu que um veículo saia da tela por volta dos 14 segundos e por isso não acusou a mudança de estado de nenhuma seção nesse momento. Uma falha que não ocorre no programa.

Através deste caso também percebe-se que o *DVE* tem dificuldade em classificar corretamente as seções ocupadas pelo veículo cinza na ROI 1, evidenciada principalmente pela flutuação da classificação das seções 23 e 20.

Mesmo com os problemas, o mapeamento e determinação das ocupações das vagas mostra uma alta taxa de acerto. Neste caso de teste, a determinação inicial das vagas funciona a favor do programa. Apesar de ter dificuldades em classificar o carro cinza, o *DVE* identifica uma das seções ocupadas pelo veículo e determina que esta seção pertence a uma vaga. Por isso, o número de vagas ocupadas se mantém correto durante a execução do teste. Uma flutuação na classificação da seção causa um erro por um segundo, mas que é rapidamente corrigido. Além disso o sistema de mapeamento se mostrou eficaz e seguro. Quando o deslocamento do drone ocorre, diversos movimentos são detectados no vídeo, causando a marcação de várias vagas. Apesar disso, nenhuma seção fica marcada em duas vagas.

### 6.3.4 Vídeo 4

Neste vídeo, o veículo branco entra em cena pela parte superior da tela e estaciona entre os outros dois carros. O carro cinza sai da vaga que ocupava e estaciona em uma área inferior. O vídeo possui uma duração de 30 segundos, portanto 1800 possíveis acertos.

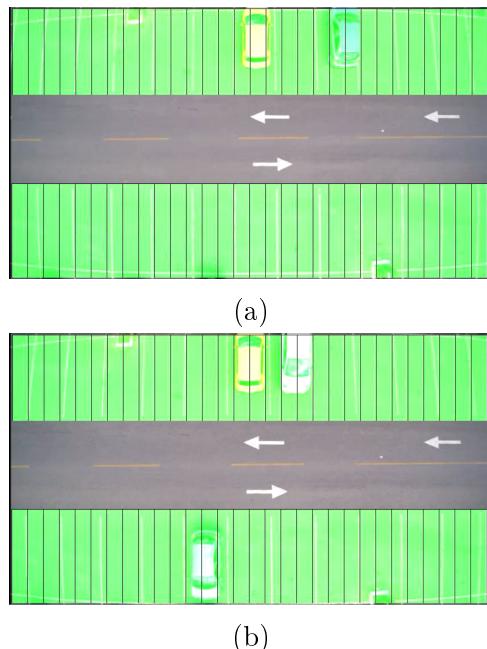


Figura 6.7: (a) O momento inicial do vídeo 4; (b) O momento final do vídeo 4.

Observador F		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 15,16,21 e 22 ocupadas.	2
7	ROI 1: Seções 19 e 20 ocupadas.	3
16	ROI 1: Seções 21 e 22 liberadas.	2
28	ROI 2: Seções 12 e 13 ocupadas.	3

Observador P		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 15,16,17,20 e 21 ocupadas.	2
8	ROI 1: Seções 18 e 19 ocupadas.	3
18	ROI 1: Seções 21 e 22 liberadas	2
29	ROI 2: 12,13 e 14 ocupadas	3

Observador M		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 15,16,20 e 21 ocupadas.	2
6	ROI 1: Seções 18 e 19 ocupadas.	3
17	ROI 1: Seções 21 e 22 liberadas	2
28	ROI 2: Seções 12,13 e 14 ocupadas	3

DVE		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 15,16,20 e 21 ocupadas.	2
6	ROI 1: Seções 18 e 19 ocupadas.	3
11	ROI 1: Seção 22 ocupada.	3
14	ROI 1: Seção 22 liberada.	3
23	ROI 1: Seção 20 liberada.	2
28	ROI 2: Seções 12 e 13 ocupadas.	3

Acertos - seções		
Observador	Acertos	Taxa de acertos
F	1750	97,22%
P	1749	97,16%
M	1772	98,44%
Média	1757	97,61%

Acertos - vagas		
Observador	Acertos	Taxa de acertos
F	22	73,33%
P	22	73,33%
M	24	80%
Média	22,66	75,55%

Este vídeo reforça a dificuldade do programa de classificar corretamente as seções ocupadas pelo veículo cinza. Porém a dificuldade parece não acontecer quando este carro estaciona na região inferior, quando o programa determina que o veículo ocupa as mesmas seções que os observadores humanos.

Há um atraso na desocupação de uma vaga, indicada por volta dos 18 segundos pelos observadores mas aos 23 segundos pelo programa. O erro ocorre por causa da criação de uma vaga composta apenas pela seção 20. Apesar dos cuidados que o modo complexo toma para evitar este acontecimento, quando uma nova vaga é demarcada pelas seções 21,

22 e 23, que se sobrepõe parcialmente a vaga detectada no início do vídeo, composta pelas seções 20 e 21. Como neste caso a seção da intersecção passa a fazer parte da vaga nova, a seção 20 passa a representar uma vaga. Como a liberação desta seção só ocorre aos 23 segundos, o número de vagas ocupadas indicado é errado por este intervalo de tempo.

### 6.3.5 Vídeo 5

Um veículo desocupa a sua vaga saíndo pela parte superior da tela. O vídeo possui uma duração de 10 segundos ou 600 possíveis acertos.

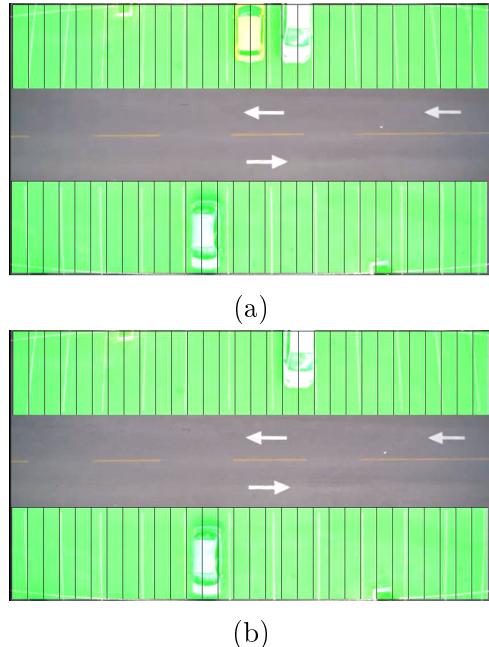


Figura 6.8: (a) O momento inicial do vídeo 5; (b) O momento final do vídeo 5.

Observador F		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 15,16,18 e 19 ocupadas. ROI 2: Seções 12 e 13 ocupadas.	3
3	ROI 1: Seções 15 e 16 liberadas.	2

Observador P		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 15,16,18 e 19 ocupadas. ROI 2: Seções 12,13 e 14 ocupadas.	3
4	ROI 1: Seções 15 e 16 liberadas.	2

Observador M		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 15,16,18 e 19 ocupadas. ROI 2: Seções 12,13 e 14 ocupadas.	3
3	ROI 1: Seções 15 e 16 liberadas.	2

DVE		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 15,16,18 e 19 ocupadas. ROI 2: Seções 12 e 13 ocupadas.	3
1	ROI 1: Seção 17 ocupada.	3
3	ROI 1: Seções 15 e 16 liberadas.	2
4	ROI 1: Seção 17 liberada.	2

Acertos - seções		
Observador	Acertos	Taxa de acertos
F	597	99,50%
P	585	97,50%
M	587	97,83%
Média	1757	97,61%

Acertos - vagas		
Observador	Acertos	Taxa de acertos
F	10	100%
P	9	90%
M	10	100%
Média	9,66	96,66%

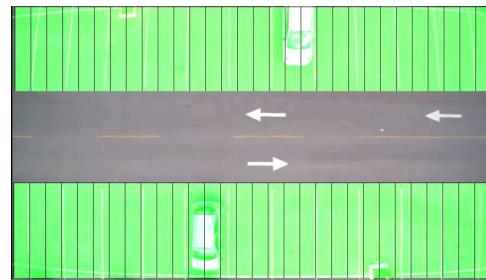
Um caso de testes simples, onde os erros ocorreram principalmente por causa da natureza subjetiva do que configura uma seção ocupada. O *DVE* classifica a seção 17 como ocupada e os observadores não. Por outro lado, dois dos observadores acharam que a seção 14 da ROI 2 estava ocupada, discordando do programa. Contudo, essas discordâncias não afetam a quantidade de vagas ocupadas.

### 6.3.6 Vídeo 6

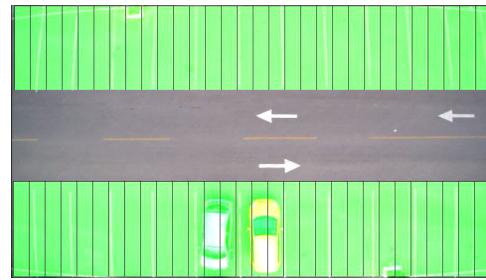
Neste vídeo o veículo amarelo entra na cena pela parte inferior da tela. O veículo branco desocupa sua vaga e sai da tela pelo lado esquerdo. O vídeo tem 13 segundos totalizando 780 acertos possíveis.

Observador F		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 18 e 19 ocupadas. ROI 2: Seções 12 e 13 ocupadas.	2
5	ROI 2: Seções 14 e 15 ocupadas.	3
8	ROI 1: Seções 18 e 19 liberadas.	2

Observador P		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 18 e 19 ocupadas. ROI 2: Seções 12 e 13 ocupadas.	2
5	ROI 1: Seções 15,16 e 17 ocupadas.	3
9	ROI 2: Seções 18 e 19 liberadas.	2



(a)



(b)

Figura 6.9: (a) O momento inicial do vídeo 6; (b) O momento final do vídeo 6.

Observador M		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 18 e 19 ocupadas. ROI 2: Seções 12 e 13 ocupadas.	2
4	ROI 1: Seções 15,16 e 17 ocupadas.	3
8	ROI 2: Seções 18,19 liberadas.	2

DVE		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 1: Seções 18 e 19 ocupadas. ROI 2: Seções 12 e 13 ocupadas.	2
5	ROI 2: Seções 15 e 16 ocupadas.	3
7	ROI 2: Seção 17 ocupada.	3
11	ROI 1: Seções 18 e 19 liberadas.	2

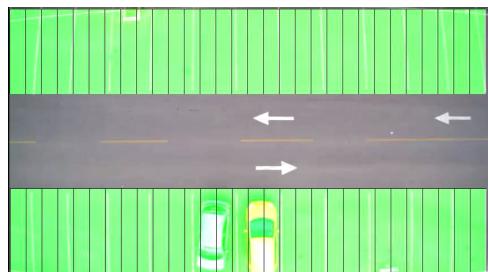
Acertos - seções		
Observador	Acertos	Taxa de acertos
F	752	96,40%
P	774	99,23%
M	767	98,33%
Média	764,33	97,99%

Acertos - vagas		
Observador	Acertos	Taxa de acertos
F	10	76,92%
P	11	84,61%
M	9	69,23%
Média	10	76,92%

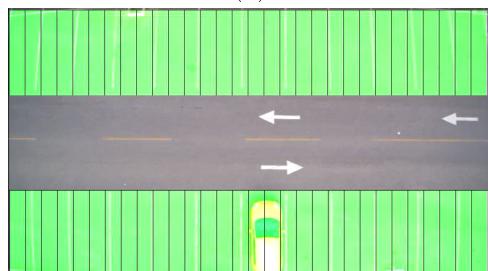
Um caso de testes aonde a avaliação do *DVE* se mantém estável e concorda quase plenamente com a avaliação dos humanos. Os atraso da detecção da liberação das seções 18 e 19 da ROI 2 e da desocupação da vaga representada por estas seções se deve ao fato de que o movimento do veículo saindo só é considerado finalizado depois que o veículo sai do campo de visão da câmera, o que ocorre alguns momentos depois que os observadores humanos consideraram que o veículo saiu da vaga.

### 6.3.7 Vídeo 7

Neste vídeo um dos carros sai da cena pela esquerda. O vídeo tem 17 segundos de duração e portanto 1020 acertos possíveis.



(a)



(b)

Figura 6.10: (a) O momento inicial do vídeo 7; (b) O momento final do vídeo 7.

Observador F		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 13,14,16 e 17 ocupadas.	2
4	ROI 2: Seções 13 e 14 liberadas.	1

Observador P		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 12,13,14,16 e 17 ocupadas.	2
6	ROI 2: Seções 12,13,14 liberadas.	1

Observador M		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 12,13,14,16 e 17 ocupadas.	2
5	ROI 2: Seções 12,13,14 liberadas.	1

DVE		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 13,14,16 e 17 ocupadas.	2
1	ROI 2: Seção 12 ocupada.	2
2	ROI 2: Seção 14 liberada.	2
5	ROI 2: Seções 12 e 13 liberadas.	1

Acertos - seções		
Observador	Acertos	Taxa de acertos
F	1013	99,31%
P	1014	99,41%
M	1016	99,60%
Média	1014,33	99,44%

Acertos - vagas		
Observador	Acertos	Taxa de acertos
F	16	94,11%
P	16	94,11%
M	17	100%
Média	16,33	96,07%

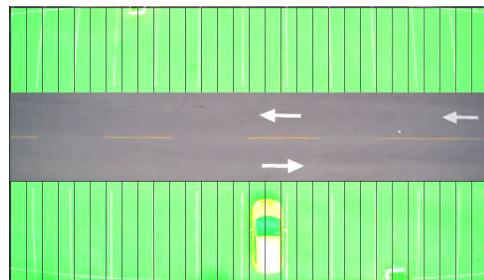
O programa concordou com os observadores nesse vídeo, exibindo apenas um leve atraso para a classificação correta da seção 12 e uma liberação levemente precipitada da seção 14. A ocupação das vagas é determinada de forma quase idêntica a dos observadores.

### 6.3.8 Vídeo 8

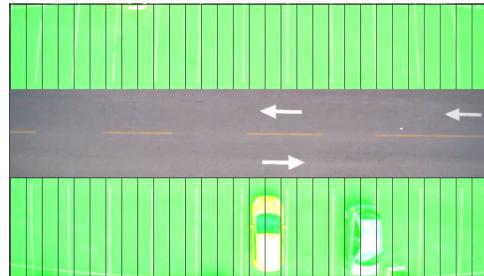
No oitavo e último caso de testes, um carro branco passa pela região central da imagem sem estacionar em nenhuma vaga e depois um carro cinza estaciona na região inferior. O vídeo tem 23 segundos de duração e 1380 acertos possíveis.

Observador F		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 16 e 17 ocupadas.	1
20	ROI 2: Seções 22 e 23 ocupadas.	2

Observador P		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 16 e 17 ocupadas.	1
20	ROI 2: Seções 22 , 23 e 24 ocupadas.	2



(a)



(b)

Figura 6.11: (a) O momento inicial do vídeo 8; (b) O momento final do vídeo 8.

Observador M		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 16 e 17 ocupadas.	1
20	ROI 2: Seções 22,23 e 24 ocupadas.	2

DVE		
Tempo(s)	Acontecimento	Vagas Ocupadas
0	ROI 2: Seções 16 e 17 ocupadas.	1
20	ROI 2: Seções 21,22,23 e 24 ocupadas.	2
21	ROI 2: Seção 21 liberada.	2

Acertos - seções		
Observador	Acertos	Taxa de acertos
F	1376	99,71%
P	1379	99,92%
M	1379	99,92%
Média	1378	99,85%

Acertos - vagas		
Observador	Acertos	Taxa de acertos
F	23	100%
P	23	100%
M	23	100%
Média	23	100%

O programa avalia as seções de forma quase idêntica aos humanos. Aos 20s, a classificação que ocorre regularmente determina que quatro seções estão sendo ocupadas pelo

veículo. Um segundo depois porém, quando o movimento do veículo termina, o erro é corrigido e o programa volta a acertar completamente.

# Capítulo 7

## Conclusão

A tarefa de se encontrar vagas em estacionamentos tem se tornado cada vez mais difícil com o aumento da frota de veículos das grandes cidades. Essa busca por um espaço livre para estacionar o carro pode durar muito tempo, criando gastos que se acumulam até valores altíssimos.

Buscando minimizar estes problemas, diversos sistemas de mapeamento de estacionamento e detecção de vagas livres foram desenvolvidos. Algumas soluções são implementadas nos próprios veículos [26], mas as mais interessantes para uma solução geral do problema são aquelas que detectam a quantidade de vagas livres em uma região do estacionamento e disponibilizam essa informação para todos os seus usuários. Em garagens, sensores de vários tipos podem ser utilizados [22, 27, 28], mas estas soluções não são muito adequadas para aplicação em estacionamentos descobertos.

Para estes casos, uma solução que se mostrou adequada e de fácil implementação foi o uso de câmeras de vídeo e algoritmos de processamento de imagens e visão computacional. Este trabalho apresentou um algoritmo para ser utilizado em um sistema como estes, chamado de *Detector de vagas em estacionamentos abertos*, abreviado como *DVE*. O algoritmo recebe as imagens de câmera de vídeo coloridas montadas em postes de luz e usa uma rede neural artificial combinada com uma técnica de rastreamento dos veículos através do fluxo óptico para mapear e determinar a ocupação das vagas que aparecem na imagem.

O *DVE* é especialmente adequado para estacionamentos descobertos com postes de luz em intervalos de distância regular e com poucas ou nenhuma obstrução visual. Quando testado em vídeos que simulavam a captura de uma câmera instalada em um destes postes, o programa apresentou resultados bastante semelhantes a resultados determinados por um observador humano. O *DVE* apresentou dificuldade em classificar corretamente regiões ocupadas por carros com coloração semelhante a do asfalto quando iluminados de certa maneira. Além disso apresentou a possibilidade de detecção errônea de vagas, o que fazia que fossem acusadas uma ocupação maior do que a real nos vídeos.

Apesar de alguns erros, o *DVE* é resistente a pequenos erros de classificação da rede neural na maioria dos casos e até aos movimentos do equipamento de captura, já que na maioria das vezes continuou detectando o número correto de vagas ocupadas na imagem apesar destes empecilhos.

Por fim, o programa se mostrou promissor e capaz de ajudar de forma simples usuários de estacionamento a encontrarem vagas com mais facilidade. Ainda é necessário, porém,

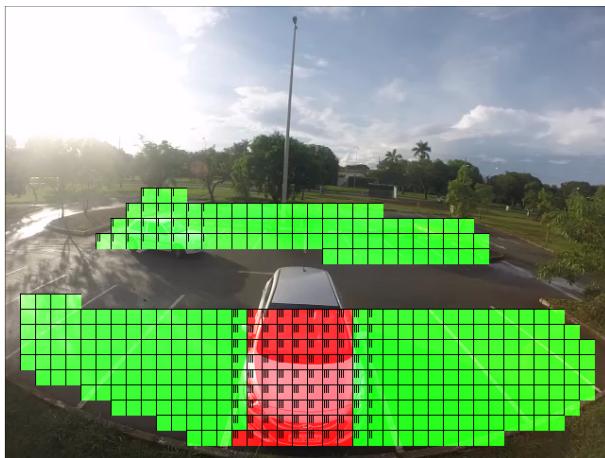


Figura 7.1: Resultados preliminares de um possível trabalho futuro.

fazer um refinamento do sistema de mapeamento do *DVE*, de forma que seja possível criar um mapa mais preciso do estacionamento monitorado e disponibilizar mais informações aos usuários, como o tempo médio de espera ou a posição exata das vagas livres. Um outro possível trabalho futuro é uma adaptação do *DVE* que o torne capaz de funcionar com imagens de câmeras em ângulos oblíquos ao estacionamento, permitindo uma instalação ainda mais fácil e em estacionamentos de tipos mais variados. Já foram feitos experimentos neste sentido usando seções quadradas menores ao invés das seções verticais utilizadas atualmente e a mesma rede neural com resultados preliminares promissores, como exemplificado na Figura 7.1.

# Referências

- [1] Rafael C Gonzalez. *Digital image processing*. Pearson Education India, 2009. 3
- [2] Tinku Acharya and Ajoy K Ray. *Image processing: principles and applications*. John Wiley & Sons, 2005. 4
- [3] IBGE. *Introdução ao processamento digital de imagens*. IBGE, 2000. 4
- [4] R Gaunt. Color spaces in digital video. Technical report, Lawrence Livermore National Lab., CA (United States), 1997. 4
- [5] Charles A. Poynton. *A Technical Introduction to Digital Video*. John Wiley & Sons, Inc., New York, NY, USA, 1996. 5, 7
- [6] Jefferson Gustavo Martins, YMG Costa, D Bertolini, and LS Oliveira. Uso de descriptores de textura extraídos de glcm para o reconhecimento de padroes em diferentes domínios de aplicaçao. In *XXXVII Conferencia Latinoamericana de Informática*, pages 637–652, 2011. 5, 6
- [7] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002. 5
- [8] Alan C Bovik. *Handbook of image and video processing*. Academic press, 2010. 7
- [9] Virgínia Fernandes Mota. *Tensor baseado em fluxo óptico para descrição global de movimento em vídeos*. PhD thesis, Universidade Federal de Juiz de Fora, 2011. 7, 8, 31
- [10] Alexandre Wagner Chagas Faria. Fluxo óptico. *Universidade Federal de Minas Gerais, ICEx-DCC-Visão Computacional*, 1992. 7, 8, 31
- [11] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005. 8, 31
- [12] Simon Haykin. *Neural Networks*. Prentice Hall, 2 edition, 1999. 9, 11, 13
- [13] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990. 9
- [14] Daniel D Chiras et al. *Human biology*. Jones & Bartlett Publishers, 2013. 9

- [15] Nikola K Kasabov. *Foundations of neural networks, fuzzy systems, and knowledge engineering*. Marcel Alencar, 1996. 11, 14, 15
- [16] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006. 11
- [17] DG Altman. Bs everitt (1998) cambridge dictionary of statistics. 11
- [18] Guoqiang Peter Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000. 14
- [19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016. 14
- [20] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995. 15
- [21] Idris M.Y.I, Leng Y.Y, et al. Car park system: A review of smart parking system and its technology. *Information Technology Journal*, 8(8):101–113, June 2009. 17
- [22] Amin Kianpisheh, Norlia Mustaffa, Pakapan Limtrairut, and Pantea Keikhosrokiani. Smart parking system (sps) architecture using ultrasonic detector. *International Journal of Software Engineering and Its Applications*, 6(3):55–58, 2012. 17, 54
- [23] Diana Delibaltov, Wencheng Wu, Robert P Loce, Edgar Bernal, et al. Parking lot occupancy determination from lamp-post camera images. In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pages 2387–2392. IEEE, 2013. 18
- [24] DBL Bong, KC Ting, and KC Lai. Integrated approach in the design of car park occupancy information system (coins). *IAENG International Journal of Computer Science*, 35(1):7–14, 2008. 18
- [25] Nicholas True. Vacant parking space detection in static images. *University of California, San Diego*, 2007. 18
- [26] Matthias R Schmid, Savas Ates, Jürgen Dickmann, Felix von Hundelshausen, and H-J Wuensche. Parking space detection with hierarchical dynamic occupancy grids. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 254–259. IEEE, 2011. 54
- [27] Sangwon Lee, Dukhee Yoon, and Amitabha Ghosh. Intelligent parking lot application using wireless sensor networks. In *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*, pages 48–57. IEEE, 2008. 54
- [28] Joerg Wolff, Thomas Heuer, Haibin Gao, Michael Weinmann, Stefan Voit, and Uwe Hartmann. Parking monitor system based on magnetic field senso. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 1275–1279. IEEE, 2006. 54