**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
2015

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

Coordenador: Prof. Dr. Coordenador

Banca examinadora composta por:

Prof. Dr. Alexandre Zaghetto (Orientador) — CIC/UnB
Prof. Dr. Professor I — CIC/UnB
Prof. Dr. Professor II — 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, 2015.
65 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2015.

1. Estacionamento, 2. Processamento de Imagens, 3. Vagas livres

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. Professor I
CIC/UnB

Prof. Dr. Professor II
CIC/UnB

Prof. Dr. Coordenador
Coordenador do Bacharelado em Ciência da Computação

Brasília, 10 de setembro de 2015

# Dedicatória

Dedico a....

# Agradecimentos

Agradeço a....

# Abstract

Soluções para a detecção e monitoramento do estado de vagas em estacionamentos tradicionalmente tem sido implementadas através do uso de sensores físicos posicionados nas vagas do estacionamento e costumam não estar presentes em estacionamentos abertos. Esse paper propõe um solução para esse problema através de processamento de imagens obtidas com uma câmera de vídeo posicionada em uma posição adequada sobre o estacionamento.

**Palavras-chave:** Estacionamento, Processamento de Imagens, Vagas livres

# Abstract

My abstract lorem ipsum

**Keywords:** Parking lot, Image processing, Free Spaces

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

## 1.1 Motivação

O crescimento constante das grandes cidades ao redor do mundo traz consigo diversos desafios relacionados à mobilidade urbana e gerência de veículos. A frota de veículos no mundo continua a crescer com o passar dos anos e vários problemas, das mais diversas naturezas, surgem desse aumento. É preciso então, cada vez mais, encontrar soluções inteligentes e eficientes para lidar com as situações que emergem. Programas de melhoria e estímulo ao transporte coletivo, redução de emissões de veículos e novas tecnologias de consumo de combustível têm sido eficazes em batalhar esses problemas, mas ainda há muito espaço a ser explorado para resolver os diversos desafios que aparecem com o crescimento da frota de veículos.

Um problema de particular interesse para os moradores de grandes cidades é o da disponibilidade de vagas de estacionamento em espaços comercias e residenciais. Com o número crescente de veículos, a demanda por espaços onde um cidadão possa parar seu carro aumenta exponencialmente e com o aumento dessa demanda, se inicia uma busca por mais espaços onde se possa construir estacionamentos e métodos para facilitar a tarefa de se encontrar vagas livres.

Muitos motoristas nos maiores centros metropolitanos perdem uma quantidade considerável de tempo todos os dias no seu deslocamento entre a sua casa e seu trabalho procurando uma vaga disponível para estacionar seu veículo. Além disso, o mesmo motorista ainda perde mais tempo buscando vagas em grandes estabelecimentos comerciais como *shoppings* ou supermercados.

Tempo perdido não é o único problema que surge na dificuldade de se estacionar o carro. Quanto mais tempo esses carros rodam atrás de espaços disponíveis, mais combustível é gasto e com isso mais dinheiro se perde e mais poluentes são liberados na atmosfera. É fácil então perceber que reduzir o tempo gasto na procura de uma vaga de estacionamento, tem impacto não só na vida de um indíviduo, mas também no meio ambiente e na vida financeira do cidadão. É claro que em uma escala individual essa redução tem pequeno impacto, mas em uma escala global, otimizar a busca por vagas traz grandes benefícios.

Os benefícios de um sistema que facilite a tarefa de estacionar vão ainda mais além. Otimizar estacionamentos pode ser uma poderosa jogada de marketing. Um negócio que possua um sistema facilitador para o estacionamento instalado atrai novos clientes, que

apreciam não ter que passar pela árdua tarefa de procurar vagas. Uma boa logística para estacionamentos pode ser um fator que diferência um estabelecimento dos seus concorrentes e o coloca acima dos demais.

Estacionamentos rotativos também têm grande interesse em sistemas dessa natureza. Eles permitiriam que o estacionamento informasse as vagas atualmente livres para os seus clientes. Mais ainda, se é possível detectar vagas livres, é possível também obter informações sobre as vagas ocupadas como tempo de permanência do veículo que a ocupa atualmente e tempo disponível restante. Sendo assim, é possível automatizar completamente o funcionamento do estacionamento rotativo, economizando uma grande quantidade de dinheiro. É possível também utilizar as informações obtidas pelo programa para encontrar padrões de estacionamento e horários de pico e ajustar preços de acordo com esses dados[12].

Outra dificuldade encontrada nos estacionamentos é a de veículos estacionados de forma imprópria[9]. Veículos que ocupam mais de uma vaga podem impedir que outros usuários se aproveitem do espaço disponível. Um sistema de monitoramento de ocupação das vagas poderia ajudar a minimizar esse problema.

É muito fácil então perceber que otimizar a tarefa de estacionar tem impacto não só no comforto individual, mas também econômico e ambiental. De fato, muitas soluções já foram propostas e implementados e serão apresentadas na seção 1.3 abaixo. Porém a maioria dessas soluções são caras e difíceis de implementar. Além disso, focam principalmente em otimizar estacionamentos fechados, como garagens. É preciso também encontrar soluções baratas e eficientes para facilitar a busca e monitoramento de vagas em estacionamentos abertos, como aqueles encontrados comumente em supermercados.

## 1.2 Objetivo

O objetivo deste trabalho é apresentar um *software* capaz de interpretar as imagens em câmera que está filmando um estacionamento e detectar quantas vagas estão disponíveis, a posição dessas vagas, a quanto tempo as vagas ocupadas estão ocupadas e disponibilizar essas informações. Além disso o programa será capaz de determinar padrões de uso do estacionamento depois de um determinado tempo de execução, como tempo médio de ocupação, horários de pico e vagas mais populares. Esse *software* deve ser capaz de auxiliar donos de estabelecimentos comerciais e de estacionamentos rotativos a melhorar os seus serviços nessa área.

## 1.3 Soluções existentes

Se sistemas de otimização de estacionamento trazem tantas vantagens, é de se imaginar que já se tenham sido buscadas e encontradas várias soluções para esse problema. De fato, é uma área de grande interesse e diversas alternativas já foram encontradas.

Os chamados sistemas inteligentes de estacionamento, ou *smart parking systems*(SPS) podem ser divididos em cinco grandes categorias[12]: sistemas de orientação e informação para estacionamento, sistemas baseados em informação de trânsito, sistemas de pagamento inteligente, E-parking e estacionamentos automatizados.

Embora a discussão aprofundada de cada uma desses tipos de sistema não seja de grande interesse para esse trabalho, todas os sistemas, independente da categoria a que pertecem compartilham uma necessidade: determinar a ocupação das vagas em um estacionamento. Existem várias formas de se detectar se um carro está estacionado em uma determinada vaga, incluindo sistemas de processamento de imagens e uma variedade enorme de sensores diferentes.

Os sistemas de obtem informação sobre o ocupação atual de estacionamentos normalmente são de uma de quatro formas[1]: aqueles que se usam de contadores, sensores ligados a fio, sensores *wireless* e sistemas baseados em imagem.

Podemos ainda dividir esses sistemas em duas categorias: os intrusivos e os não intrusivos[12]. Sistemas intrusivos normalmente estão instalados sob o asfalto ou dentro do concreto da estrutura do estacionamento. Esses sistemas são mais caros e de expansão muito mais difícil. Sistemas não-intrusivos são aqueles que podem ser instalados externamente. Eles são muito mais econômicos, porém normalmente exigem que sejam instalados sensores individuais em cada vaga e são portanto, pouco expasíveis.

Os sistemas baseados em contadores normalmente se utilizam de algum sensor nas entradas e saídas do estacionamento que detectam quando um veículo passa por eles. Esses sistemas, apesar de baratos e de fácil manutenção podem apenas determinar o número total de vagas livres ou ocupadas, sem indicar a posição dos espaços disponíveis para os usuários.

Os sistemas baseados em sensores, com ou sem fio, normalmente contam com um sensor individual para cada vaga. Esses sensores podem ser de diversos tipos: balanças, tubos pneumáticos, sensores infravermelhos, magnéticos ou de ultrasom, entre outros. Muitos fatores diferentes afetam a escolha do sensor ideal para um determinado caso, como facilidade de instalação, preço e facilidade de expansão e manutenção. Os sistemas de sensores cabeados têm sido substituídos por sistemas de tecnologia *wireless* a medida que esse tecnologia cresce.

Todas essas soluções porém, tem sido normalmente implementadas em garagens e outros estacionamentos fechados. Alguns exemplos, principalmente aqueles que são instalados no teto, como os sensores de ultrasom, só podem ser utilizados nesse tipo de estacionamento. Nesses casos, é possível instalar os sensores durante a construção da estrutura. Estacionamentos fechados estacionamentos também não costumam aumentar seu tamanho ou número de vagas disponíveis com o passar do tempo. Para estacionamentos abertos, é preciso encontrar outra solução.

Os sistemas que usam processamento de imagem normalmente se aproveitam da imagem capturada por uma câmera, que pode fazer um papel dobrado como câmera de segurança, para determinar o estado das vagas do estacionamento. Uma câmera normalmente é responsável por um grande conjunto de vagas e pode ser instalada em poste de luz ou em um outro espaço construído especificamente para ela. A câmera envia as imagens para um computador central que possui um *software* capaz de interpretar essas imagens para determinar quais vagas estão vazias. Um sistema desse tipo é ideal para o objetivo deste trabalho. Além disso são baratos e de fácil manutenção e escalabilidade.

Existem diversas soluções para sistemas baseados em visão computacional para a determinação de ocupação em estacionamento. Alguns *softwares* se utilizam de modelagem tridimensional e cálculos de probabilidade para detecção contínua em vídeo.[4]. Outros são aplicados em imagens estáticas e se utilizam de classifições de pixels para determinar

a posição de veículos [14]. Soluções diferentes podem detectar as vagas automaticamente ou através de *input* de usuários.

O sistema proposto nesse trabalho se utiliza de geração dinâmica de fundo e técnicas de subtração de *background* para determinar o estado de cada vaga presente na imagem. Para a determinação das vagas, basta uma calibração inicial no momento da instalação.

# Capítulo 2

# Fundamentação teórica

## 2.1 Processamento de imagens e Visão Computacional

A visão humana é considerada por muitos um sentido complexo e muito poderoso. Além disso, as imagens podem ser a parte mais importante da nossa percepção. É natural então imaginar que cientista queiram reproduzir as características e capacidades da visão através de programas de computador. Mas enxergar, ou ver, é muito mais do que detectar obejtos em um ambiente. Envolve também analisar e encontrar características importantes e derivar informação do que foi percebido.[6] Reproduzir esse processo é uma tarefa de extrema dificuldade e com esse intuito surgiu a área de Processamento de imagens.

O Processamento de imagens é uma disciplina que vem do processamento de sinais. Afinal de contas, uma imagem não passa de um conjunto de sinais luminosos detectados por um sistema de visão. 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.[5] Cada elemento individual dessa imagem, cada valor em cada coordenada, pode ser chamado de um *picture element* ou mais comumente *pixel*.

O interesse em Processamento de imagens normalmente se aplica em duas grandes áreas. Aprimoramento e correção de imagens para fins de interpretação humana e algoritmos para análise automática de informações contidas em imagens. Alguns autores chamam a primeira área de Processamento de imagens e a segunda de visão computacional. Essa distinção é restritiva, pois implica de certa forma que um programa na área de Processamento de imagens deve ter uma imagem como entrada e saída, mas nesse caso, nem uma tarefa simples como encontrar a cor predominante em uma imagem estão

Diferente dos humanos, softwares de processamento de imagem ou visão computacional, são capazes de extrair informações de imagens em espectros de frequência diferente, como o raio-x ou raios gamma. Computadores podem processar imagens de fontes que os olhos humanos não estão acostumados a processar.

Um sistema de Processamento de imagens normalmente possui dois componentes crucias: o equipamento para aquisição da imagem e o *software* de processamento propriamente dito. É importante lembrar que a aquisição da imagem não vem apenas de câmeras fotográficas ou de vídeo. Qualquer dispositivo físico sensível a alguma faixa do espectro eletromagnético capaz de adquirir um sinal nessa faixa e digitalizá-lo pode servir para

adquirir a imagem. O processamento normalmente é determinado por algoritmos e formulações matemáticos e estatísticas. Porém intuição humana ainda é de grande importância na escolha das técnicas a serem utilizadas.

O Processamento de imagens começou a ser aplicado na área de Jornalismo, na busca de aprimorar as imagens transmitidas por cabos submarinos entre Londres e Nova York. [11] O grande salto na área, porém, ocorreu com o advento dos primeiro computadores digitais, trinta anos depois. Com o aumento das pesquisas inúmeras novas técnicas foram criadas nos laboratórios de computação da época.

Desde então, o Processamento de imagens e a visão computacional tem sido usados em diversas áreas. De fato, quase não existe uma área de implementação tecnológica que não tenha sido impactada pelo avanço no Processamento de imagens. Na medicina, imagens são rotineiramente utilizadas para elaborar diagnósticos e é possível utilizar programas de computador para encontrar anomalias em imagens obtidas por máquinas como o raio-x. É possível contar células em uma imagem de microscópio ou galáxias em uma imagem de um telescópio espacial. Sistemas de identificação biométrica por impressão digital ou reconhecimento de retina são cada vez mais abrangentes pelo mundo. Técnicas de transmissão de sinal de imagem e compressão de vídeo possibilitam alguns dos serviços digitais mais utilizados atualmente. E finalmente, programas de visão computacional são capazes de detectar posição de veículos em um estacionamento.

Nas seções seguintes deste trabalho, descreverei alguns conceitos e técnicas de Processamento de imagens importantes para a compreensão do funcionamento do programa proposto.

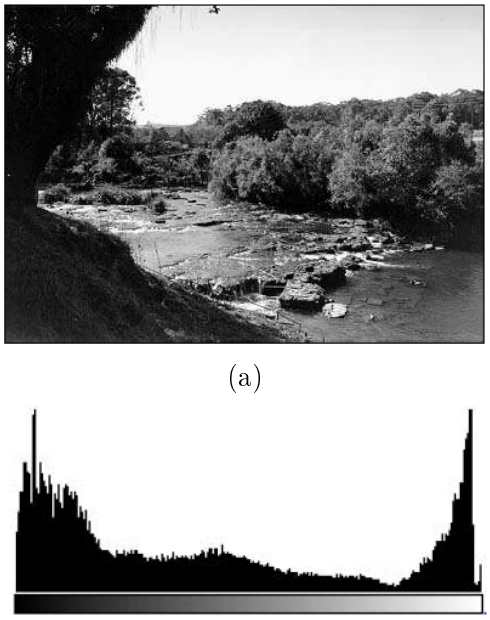
## 2.2 Histogramas

O histograma de uma imagem com níveis de cinza entre 0 e 255 é uma função discreta $h(nc) = N$. Onde nc é um determinado nível de cinza e N é o número de *pixels* na imagem que possuem esse nível.[5] O histograma nada mais é que a representação gráfica da frequência da distribuição dos níveis de uma imagem.[8]. Ele portanto informa a probabilidade de que um determinado pixel possua um certo nível de cinza. A imagem 2.1 mostra uma imagem e um gráfico de barras que representa seu histograma.

Histogramas também podem ser extraídas de imagens coloridas. Nesse caso, normalmente cada canal RGB é separado e um histograma para cada um desses canais é gerado. É comum que programas que usam o histograma de uma imagem dividam os valores encontrados em cada nível pelo número total de *pixels*, obtendo assim um histograma normalizado.

A extração de histogramas é a base para diversas operações de processamento de imagem. Além de prover estatísticas importantes sobre a imagem, os cálculos de histogramas são pouco exigentes tanto para o *software* quando para o *hardware* e portanto são muito importantes para processamento em tempo real de vídeo.

Alguns conceitos de qualidade de imagem, como constraste e brilho estão associados a histogramas. A forma do histograma sozinha pode servir para se determinar vários aspectos da imagem. O constraste de uma imagem é diretamente proporcional dos níveis de cinza da imagem, ou seja, é proporcional a largura do gráfico do histograma. O brilho por outro lado pode ser indicado pela média dos valores do histograma ou pelos picos



(a)

(b)

Figura 2.1: (a) Uma imagem; (b)O seu histograma [11]

mais a direita ou mais a esquerda do histograma, representando respectivamente brilho mais alto ou mais baixo.

Uma utilidade comum do histograma são as chamadas técnicas de transformação de histograma. Nessas técnicas, uma função de transformação é aplicada sobre a função h(nc) do histograma a fim de deslocar os valores para níveis mais desejados. Uma função de transformação que torne a curva do histograma mais uniforme aumenta o contraste da imagem, enquanto outras podem aumentar a probabilidade de um determinado valor que se deseja dar destaque.

Além disso, os histogramas podem ser utilizados para auxiliar em técnicas de segmentação, como será discutido na seção 2.3. A análise de um histograma pode ser crucial para a determinação de um limiar adequado para binarização. Na seção 3.1 eu descrevo como o histograma de cada quadro do vídeo é utilizado para separar os veículos em movimento na imagem dos outros elementos estáticos.

## 2.3 Segmentação de objetos

Um processo importante para vários sistemas de visão computacional é o da extração de características ou *features* importantes de uma ou mais imagens. Através destes *features*o computador deve ser capaz de extrair uma informação ou conhecimento desejado, ou até mesmo mostrar na saída uma imagem diferente que possa ser analisada por humanos a fim de determinar essa informação.

Quando o objetivo do processo é de localizar e extrair a posição de objetos específicos em uma cena, chamamos esse processo de segmentação de objetos. Ele consiste basicamente em isolar e selecionar *pixels* da imagem de entrada que possam fazer parte da representação visual do objeto[6]. A saída desses programas pode ser uma imagem que

contem apenas os objetos desejados, ou um arquivo com informações como o número de objetos encontrados diferentes encontrados.

Existem diversas técnicas para segmentação de objetos descritas na literatura. Cada uma dessas técnicas é ideal para um tipo diferente de imagem ou de objeto a se encontrar. Assim como em muitas áreas da computação, a decisão da técnica a ser utilizada é tão importante quando a implementação do programa propriamente dito.

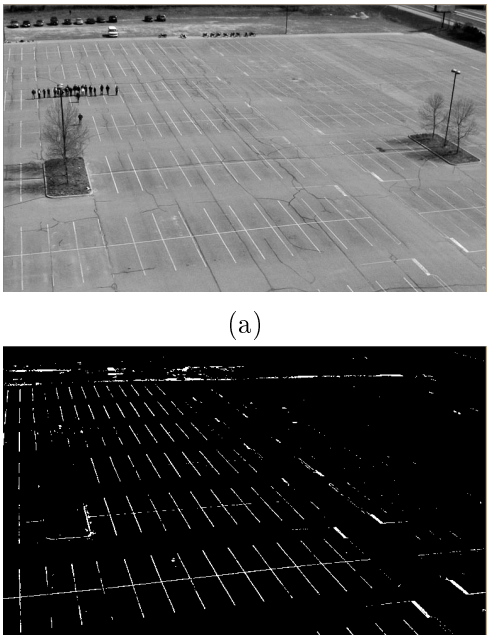
É interessante apresentar alguns conceitos importantes antes de iniciar uma discussão sobre técnicas de segmentação. O primeiro deles é o conceito de vizinhaça do *pixel*. A vizinhança de um ponto de uma imagem é um objeto de estudo importante para determinar objetos. Ela consiste essencialmente dos valores dos *pixels* ao redor do *pixel* sendo analisado. Chamamos de vizinhança-4 o conjunto dos quatro *pixels* que ficam diretamente acima, abaixo, à direita e à esquerda do ponto em questão. A vizinhança-8, por sua vez, é então o conjunto que consiste da vizinhança-4 e mais os quatro *pixels* nas diagonais do ponto sendo analisado. Chamamos os *pixels* que fazem parte de alguma vizinhança de outro *pixel* de vizinho.

O segundo conceito importante é a definição de bordas. As bordas em uma imagem representam os limites de cada objeto e são, portanto, de extrema importância na segmentação de objetos em imagens. Pontos pertencentes a bordas são aqueles que possuem uma mudança brusca nos níveis de cinza. Por exemplo, podemos definir um ponto de uma borda em uma imagem binária - que possui apenas pontos brancos ou pretos - como um ponto preto que contenha pelo menos um vizinho branco.[15]

Uma técnica simples e muito utilizada para segmentação é da limiarização, ou *thresholding*. Essa técnica consiste na categorização dos *pixels* de uma imagem através de um limiar $L$. Aqueles que possuirem valores maiores que $L$ terão seu valor configurado como o valor máximo, representado pela cor branca. Normalmente o valor do limiar é escolhido de acordo com a expectativa dos valores de cinza dos obejtos do conjunto de interesse da imagem, mas é possível também implementar formas de detectar um limiar apropriado computacionalmente. Na seção 3.1 é descrito como o histograma de cada quadro foi usado para determinar um limiar global para a execução do *thresholding* de cada *frame*. O resultado é visível na figura 2.2. É possível também, e apropriado em vários casos, determinar limiares locais para a imagem. Nesse caso, cada fragmento da imagem, como um quadrante, possui um valor diferente de $L$. O resultado desse processo de limiarização é normalmente uma imagem binária.

Uma vez que temos uma imagem binarizada podemos utilizar várias técnicas para separar os objetos. A mais simples de todas, nos casos onde a imagem contém apenas um objeto de interesse que possui valores brancos na imagem, podemos simplesmente encontrar todos os *pixels* brancos. Em imagens binárias que possuem mais de um objeto de interesse é preciso implementar uma maneira de se separar esses objetos. Existem técnicas que analisam a conectividade dos pontos a fim de extrair limites dos objetos.[15] Outros algoritmos classificam pontos com o valor de interesse em classes diferentes. Pontos com o valor interessante não-classificados que sejam vizinhos de um ponto de uma classe passam a fazer parte daquela classe ou passam a integrar uma classe nova caso contrário. Ao final da execução, cada classe contém os *pixels* de um objeto diferente.

Outra técnica interessante de separação de objetos em imagens é o algoritmo chamado *watershed*. *Watersheding* é um algoritmo morfológico que permite a detecção de bordas em imagens em níveis de cinza. É uma técnica muito eficiente e que produz contornos



(a)

(b)

Figura 2.2: (a) Uma imagem de um estacionamento; (b) A mesma imagem binarizada através da técnica utilizada na seção 3.1

bastante precisos. Esse algoritmo possue uma desvantagem particular em relação a outros algoritmos de segmentação de imagens. Por vezes, se mal utilizado a técnica de *watershe-ding* pode segmentar demais a imagem, reconhecendo objetos não existentes. Existem então, diversas pesquisas para minimizar esse problema, como a apresentada em [10].

O algoritmo de *watershed* consiste em interpretar a imagem como um mapa topológico de uma região, onde os pontos mais escuros representam pontos mais baixos, ou "vales". Uma vez detectados esses vales, o algoritmo começa um processo de inundação do vale, daí o nome. Computacionalmente, esse processo de inundação consiste na atribuição de classes a pixels adjacentes ao vale que se propaga a cada iteração. Os pontos onde a água se encontraria - ou pontos que seriam atribuídos a duas classes - são os pontos de borda da imagem. Essa técnica é muito usada no ramo da Biologia, para a segmentação de células em imagens de microscópio.

É importante também diferenciar objetos encontrados em uma imagem um do outro. Muitas vezes a imagem binária resultante de um processo de segmentação contém diversos objetos. Em alguns casos, não temos interesse em todos eles e é importante que o computador seja capaz de diferenciar um do outro. Existem diversas técnicas que visam esse objetivo. É possível analisar o histograma apenas da região onde o programa sabe que existe um objeto e comparar com um histograma esperado. Outros algoritmos se utilizam de operações matricias que se aproximam de uma derivação a fim de encontrar a orientação das bordas do objeto do objeto detectado.

Outra abordagem bastante utilizada é a de redes neurais treinadas para identificar esses *features* de objetos de interesse. Objetos encontrados são analisados por uma técnica qualquer e suas características são encontradas. As redes são então treinadas para classificar objetos com as mesmas características junto com os seus semelhantes. Esse

tipo de implementação já foi utilizada com sucesso em aplicações para gerenciamento de estacionamento, como a apresentada em [14].

Para a implementação programa apresentado nesse trabalho são de particular interesse as técnicas de segmentação baseadas em *thresholding* e subtração de fundo. As técnicas de geração e subtração de fundo serão descritas mais adiante na seção 2.6. Além delas, as técnicas de reconhecimento de *features* de objetos através de comparação de histogramas serão bastante utilizadas para a confirmação de que o objeto encontrado é um veículo.

## 2.4 Subtração de imagens

A diferença entre duas imagens pode ser obtida através de uma subtração simples dos valores dos elementos da primeira imagem com os elementos correspondentes da segunda imagem. Matematicamente podemos expressar essa operação pela equação:

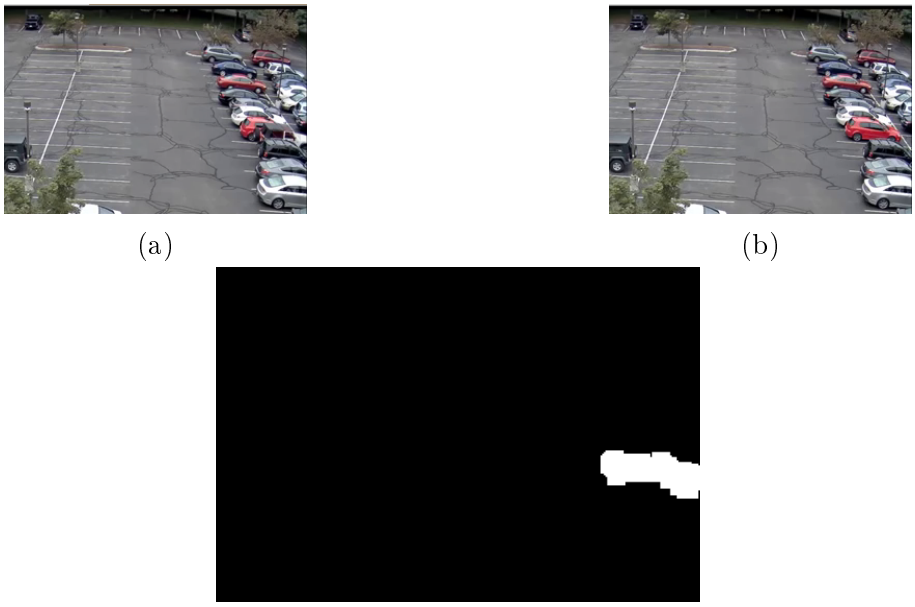
$$D(x, y) = A(x, y) - B(x, y) \tag{2.1}$$

Onde $D(x, y)$, $A(x, y)$ e $B(x, y)$ são os valores dos *pixels* na posição (x,y) das imagens de diferença obtida e as imagens A e B originais. Podemos subtrair imagens RGB fazendo a subtração normal de cada canal da primeira imagem com o canal correspondente da segunda.

A grande utilidade de se subtrair duas imagens é justamente realçar as diferenças entre estas duas imagens. Obter a diferença de duas imagens pode ser útil para facilitar a visualização de resultados da aplicação de uma algoritmo ou para encontrar mudanças que ocorrem entre dois quadros distintos de um vídeo capturado. A diferença entre dois quadros de um vídeo contém a posição de um objeto que se moveu na cena, mais o ruído da imagem. Essa segunda aplicação mencionada é de grande relevância para este trabalho. Na seção 3.1 mais a frente, eu descrevo como utilizei a diferença entre dois quadro consecutivos para encontrar uma região que contenha um veículo em movimento, e na seção 3.3 mostro como utilizei essa região para separar dinamicamente o *background* da imagem dos elementos de *foreground*. Além disso, no algoritmo descrito na seção 3.4 utilizo a diferença entre o fundo da imagem em dois momentos diferentes para detectar veículos que estacionaram nesse período de tempo e passaram a fazer parte do fundo.

Se uma imagem composta por valores de níveis de cinza tem o valor de cada um dos seus *pixels* representado por 8 bits, os valores que podem ser representados estarão então entre o intervalo de 0-255. Mas imagens resultantes da subtração de outras duas imagens terão seus valores representados na faixa de -255 a 255. É preciso então processar esses valores para trazê-los de volta para o intervalo que podemos representar. Existem diversas soluções para esse problema. Os algoritmos deste trabalho, porém, se importam apenas com a existência de uma diferença entre duas imagens e a posição dos *pixels* que possuem valor maior que zero na imagem de diferença e não os valores específicos de cada um. Sendo assim, a abordagem utilizada para resolver essa situação foi apenas a extração do valor absoluto do resultado da diferença.

A figura 2.3 ilustra a subtração de duas imagens. O resultado apresentado já passou por um processo de limiarização e portanto é uma imagem binária. As imagens utilizadas são fundos gerados dinamicamente. A imagem obtida pela subtração mostra uma diferença em uma região onde é sabido que há vagas no estacionamento. Na seção 3.4 é



(a) (b)

(c)

Figura 2.3: (a) O fundo em um determinado quadro em um vídeo; (b) O fundo em um outro quadro do mesmo vídeo alguns segundos depois; (c) A imagem resultante da subtração entre (a) e (b) binarizada.

descrito como essa informação é utilizada para determinar que um carro estacionou em uma vaga.

## 2.5 Operações Morfológicas

O principio básico da morfologia matemática consiste em extrair as informações relativas à geometria e à topologia de um conjunto desconhecido (uma imagem), pela transformação através de outro conjunto completamente definido, chamado elemento estruturante.[11] Para aplicar operações morfológicas então, precisamos primeiramente definir conjuntos sobre as imagens. Normalmente aplicamos as operações descritas a seguir em imagens binárias e essas imagens podem ser descritas completamente pelo conjunto A definido por:

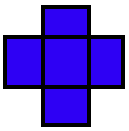
$$A = (x, y)|P(x, y) > 0 \tag{2.2}$$

Ou, mais simplesmente, o conjunto dos *pixels* $P(x, y)$ que não são pretos.

O elemento estrutrante $B$ para as operações morfológicas é uma segunda matriz, de tamanho geralmente menos do que o da imagem original que deve ser escolhida cuidadosamente para que se obtenha os resultados desejados.

As duas operações morfológicas básicas para o Processamento de imagens são: a dilatação, que aumenta os elementos de uma imagem alterando a área ao redor de um determinado *pixel* para conformar a um dado padrão e a erosão, que se utiliza do elemento estruturante para remover objetos indesejados e diminuir elementos.[3].

(a)



(b)

(c)

Figura 2.4: (a) Uma imagem binária; (b) O elemento estrutrante a ser utilizada na dilatação; (c) A imagem resultando do processo de dilatação, com os pixels adicionados pintados de azul para facilitar a visualização

A dilatação faz com que um objeto presente na imagem cresça de tamanho. Ela é utilizada também para elminiar buracos em componentes segmentados, pois falhas menores do que o elemento estruturante B são completamente cobertas ao final da operação. Se chamarmos o conjunto de todos os pontos da imagem de $I$, podemos definir o conjunto resultante da operação de dilatação como:

$$D(A,B) = A \oplus B = \{(x,y) \in I | B_{(x,y)} \cap A \neq \emptyset\} \tag{2.3}$$

Isto é, todos os elementos de $I$ aonde $B$, se deslocado para a posição (x,y) do elemento, intercepta o conjunto dos pontos brancos, $A$. A figura 2.4 ilustra o resultado do processo de dilatação.

Nesse trabalho, a dilatação é utilizada na seção 3.1 para aumentar a área da mancha encontrada pela diferença dos quadros consecutivos, a fim de determinar uma maior área que certamente pertence ao *foreground*.

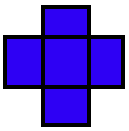
A erosão por sua vez faz o processo contrário da erosão. Ela dimunui o contorno dos objetos, pode diminuir o número de componentes segmentados e até remover elementos que sejam menores do que o elemento estruturante, uma vez que componentes que sejam menores serão eliminados da imagem $I$. Uma aplicação interessante da operação de erosão é na detecção de contornos. Podemos subtrair uma imagem de sua versão erodida para encontrar os contornos dos objetos presentes na imagem.

Da mesma forma que com a dilatação, podemos definir a erosão matematicamente como:

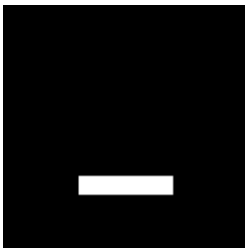
$$D(A,B) = A \ominus B = \{(x,y) \in I | B_{(x,y)} \subseteq A\} \tag{2.4}$$

Ou seja, os pontos em $I$ tais que o elemento estruturante $B$ deslocado para a posição (x,y) do ponto, está contido completamente em A. Mais especificamente, apenas os pontos

(a)



(b)



(c)

Figura 2.5: (a) Uma imagem binária; (b) O elemento estrutrante a ser utilizada na erosão; (c) A imagem resultando do processo de erosão

onde se sobrepusermos o $B$ só haverá pontos brancos sob os pontos de B. A figura 2.5 ilustra o processo de erosão sobre uma imagem binária.

É possível também combinar essas operações entre si para criar novas operações morfológicas sobre as imagens. As combinações mais comuns formam as operações de abertura e fechamento. A abertura é a aplicação de uma dilatação seguida de uma erosão. A operação da abertura suaviza os contornos dos objetos na imagem, ou seja, ela remove pequenas protuberâncias. O fechamento por sua vez, é a combinação de uma erosão seguida de uma dilatação. Essa operação é utilizada para fechar pequenos buracos ou falhas contidos no interior de um objeto na imagem binária.

## 2.6 Geração de background

Uma área de estudos de muita utilidade na segmentação de objetos é de geração de fundo. Em particular, segmentar os objetos que pertencem ao *foreground* dos objetos que pertencem ao fundo da imagem é uma etapa crucial nos programas de segmentação ou rastreamento de objetos em vídeo. De fato, o primeiro passo nesses algoritmos normalmente é encontrar um fundo para servir de imagem de referência e possibilitar que sejam aplicadas técnicas de subtração de fundo com a finalidade de encontrar objetos que estejam em movimento.

A abordagem típica de determinação de *background* é obter a imagem de referência para o fundo quando a cena está estática.[13]. Na prática essa abordagem não é totalmente eficiente, principalmente em aplicações que dependem de imagens obtidas de câmeras de segurança ou de ambientes que não costumam ficar totalmente sem movimento. Em programas que obtém as imagens através de câmeras que filmam o mesmo ambiente durante um dia inteiro, mudanças na iluminação, tanto natural quanto artificial,

modificam o fundo da imagem. Além disso, objetos que estavam em movimento podem se tornar objetos estáticos e devem ser agregados ao *background*.

É preciso então encontrar técnicas para determinar o fundo da imagem em tempo de execução, de forma adaptativa e dinâmica. Alguns programas se utilizam da média aritmética ou dos quadros durante o decorrer do vídeo. Similarmente a observar apenas uma imagem, pode-se observar a mediana dos valores de cada *pixel* dos quadros. O algoritmo apresentado em [13] se utiliza de temporizadores para atualizar os valores dos pixels de fundo e em [2] é apresentado ainda um outro método.

Esse trabalho se utiliza de uma técnica de geração de fundo similar àquela apresentada em [7] que provou ter grande sucesso na determinação de imagens de fundo em câmeras de monitoramento de trânsito. O algortimo é descrito com mais detalhes na seção 3.3.

# Capítulo 3

# Metodologia

## 3.1 Segmentação dos veículos

A primeira etapa na implementação do programa apresentado nesse trabalho é a da segmentação dos veículos em movimento durante o decorrer do vídeo obtido pela câmera estática. Para detectar esse veículos, esse projeto utiliza uma técnica de segmentação baseada naquela apresentada em [7]. Segmentar corretamente os carros é crucial para o procedimento descrito na seção 3.3. Nessa seção, descrevo como uma imagem com apenas o contorno de um carro é obtida.

Primeiramente, é extraída a imagem de diferença entre dois quadros consecutivos. Chamamos essa imagem de $D$. Essa diferença é trivial, nada mais é do que o valor absoluto da diferença entre o valor do mesmo *pixel* em cada um dos dois quadros. Pixels que representam objetos que se mantiveram estáticos durante os dois quadros originais, como o asfalto e carros estacionados, devem ter então valor zero em $D$. Portanto, é de se esperar que a imagem $D$ tenha uma grande quantidade de valores próximos ou iguais a zero.

Após extraída a diferença entre os dois quadros, resta binarizar $D$. Como descrito na seção 2.3, é preciso encontrar um limiar apropriado para cada imagem de diferença obtida. Como elas são obtidas dinamicamente, determinar esse limiar de forma estática, não traz resultados satisfatórios. Pares de quadros que apresentam menos movimento possuem valores de diferença menores e portanto, as regiões que devem ser tornadas brances têm valores inicialmente mais baixos.

Sabendo que a grande maioria dos pixels de $D$ possui valor próximo a zero, podemos determinar um limiar $T$ de forma dinâmica. Para encontrar o valor ideal de $T$, o programa primeiro encontra o pico do histograma de $D$. O nível de cinza correspondente a esse valor é o que mais ocorre na imagem e será sempre um valor bem próximo de zero. O valor escolhido para $T$ é o terceiro nível de cinza cujo valor no histograma é inferior a 1% do valor do pico. A escolha desse valor ao invés do primeiro vem de um problema gerado por ruído na captura da imagem.

Por conta de ruído na imagem, valores de pixels em pontos estáticos, como por exemplo, parte do asfalto, apresentam pequenas diferenças que poluem a imagem de diferença. Esses pequenos pontos com difereça maior que zero podem aparecer na imagem binarizada, gerando regiões indesejadas. Para minimizar esse problema, duas medidas são tomadas. A primeira é a aplicação do parâmetro do limiar mínimo $lm$. Esse valor depende do vídeo sendo analizado e deve ser maior quanto maior for o ruído da câmera.

Antes do cálculo do histograma da imagem de diferença o programa configura todos os valores abaixo desse limiar para o valor mínimo. Isso desvia o histograma para valores menores e gera uma binarização mais limpa. A determinação correta de *lm* é essencial para uma boa segmentação. A segunda medida é a escolha do terceiro valor inferior a 1% do pico. Isso evita apenas que o ruído não seja responsável ao acaso pelo limiar escolhido estar abaixo do desejado. A figura 3.1 mostra o resultado do processo.

Figura 3.1: A esquerda: a imagem de diferença D. Ao centro: D binarizada com a técnica descrita acima. A direita: o histograma de D

## 3.2 Rastreamento dos veículos

## 3.3 Geração de Fundo Dinâmico

Uma etapa importante do trabalho é a geração de uma imagem de fundo, separada da imagem do vídeo original. Para tal, é preciso implementar um método que separe os elementos do *background* daqueles do *foreground*. Determinamos que os elementos do fundo são aqueles que não estão se movimentando no momento, enquanto os elementos do *foreground* são os objetos que se movem na imagem, em particular, os carros. É importante construir essa imagem, pois é essa segmentação de fundo e plano principal que permitirá mais tarde que o programa determine se uma vaga foi ocupada ou desocupada por um veículo.

Gerar essa imagem de fundo não é uma tarefa trivial. Devido a própria natureza do espaço sendo filmado, veículos que antes compunham o fundo da imagem podem passar a se mover e veículos que estão em movimento em um determinado momento eventualmente vão estacionar e devem passar a integrar o *background*. É preciso então ter muito cuidado na determinação do fundo do vídeo.

Antes de mais nada, é preciso determinar uma imagem de fundo inicial. Esse fundo inicial será usado pelo processo de geração dinâmica de fundo descrito mais abaixo. A imagem de fundo base é construída no momento da instalação do *software* deste trabalho, e pode ser reconstruída sempre que desejado, como no início do dia ou no horário que as atividades do estacionamento se iniciam.

A imagem de fundo inicial *Fi* é formada usando os 30 primeiros quadros capturados em vídeo a partir do momento que sua formação se inicia. O valor de cada *pixel* $i$ na imagem tem valor igual a média do valor desse *pixel* em cada um desses quadros, calculada por:

$$V_i = \frac{1}{30} \sum_{t=1}^{30} V_{it} \tag{3.1}$$

Onde $V_i$ é o valor do *pixel* $i$ em $Fi$ e $V_{it}$ é o valor do *pixel* $i$ no frame de número $t$.

Uma vez obtida $Fi$ não podemos simplesmente utilizá-la como um fundo definitivo. A determinação do *background* deve ser feita de forma dinâmica e constante por conta de diversos fatores. Alguns deles são:

- Mudanças na iluminação: com o passar do dia, tanto a iluminação natural quanto a artificial sofrem mudanças. Essas alterações podem fazer com que a cor do asfalto percebida pela câmera mude. É importante adaptar o fundo para que isso não surja como uma diferença entre dois quadros consecutivos
- Comportamente dos carros: veículos anteriormente parados podem começar a se mover e veículos em movimento vão estacionar. É preciso adaptar o *background* a esses eventos.
- Ruído da câmera: um fundo adaptativo e dinâmico minimiza os problemas causados por ruído na captação do vídeo.

Para atualizar corretamente o fundo do vídeo, precisamos primeiramente determinar a região do *foreground* da imagem. Para tal, o programa utiliza a imagem D obtida na seção 3.1. A versão binarizada de D passa por uma operação de dilatação que transforma o carro segmentado em uma mancha branca. Essa mancha representa uma região onde o programa sabe com certeza que existe um carro em movimento e será usada como uma máscara para a geração dinâmica do fundo. Chamaremos essa máscara de $Fg$.

O programa usa essa máscara para atualizar a imagem de fundo para um estado mais apropriado. Para cumprir essa tarefa, primeiro é extraída uma imagem $T$ que contém apenas o conteúdo do fundo atual na região da mancha branca $Fg$. Em seguida, $Fg$ é invertida e se torna uma mancha preta sobre um fundo branco, chamada $Fg'$. Essa segunda máscara é utilizada para apagar do quadro atual a região de *foreground* determinada pela mancha preta. Em seguida os valores dos pixels diferentes de zero em $T$ são inseridos nos pixels da mancha negra sobre a imagem do quadro atual. A média aritmética da soma dos demais valores da imagem do quadro atual e do fundo atual é extraído passa a ser o valor desses pixels no novo fundo.

O processo inteiro é mais facilmente compreendido através da equação que o define:

$$Bgi = \begin{cases} Ba_i & , Fgi = 255 \\ \frac{Bai}{2} + \frac{Qi}{2} & , Fgi = 0 \end{cases} \tag{3.2}$$

Onde $Bg_i$ é o valor do *pixel* $i$ na nova imagem de fundo, $Ba_i$ é o valor do *pixel* $i$ na imagem de fundo anterior, $Fg_i$ é o valor deste mesmo *pixel* em $Fg$ e $Q_i$ no quadro atual. Resumidamente, o processo mantém os valores do fundo anterior nos pontos onde se tem certeza que existe um elemento do *foreground* e faz uma média entre o fundo anterior e o quadro atual para os demais valores. A figura 3.2 ilustra o processo da geração dinâmica de fundo.

A imagem de fundo obtida através do processo que foi descrito nessa seção é o objeto de estudo mais importante deste trabalho. Através dela são obtidas as informações relevantes

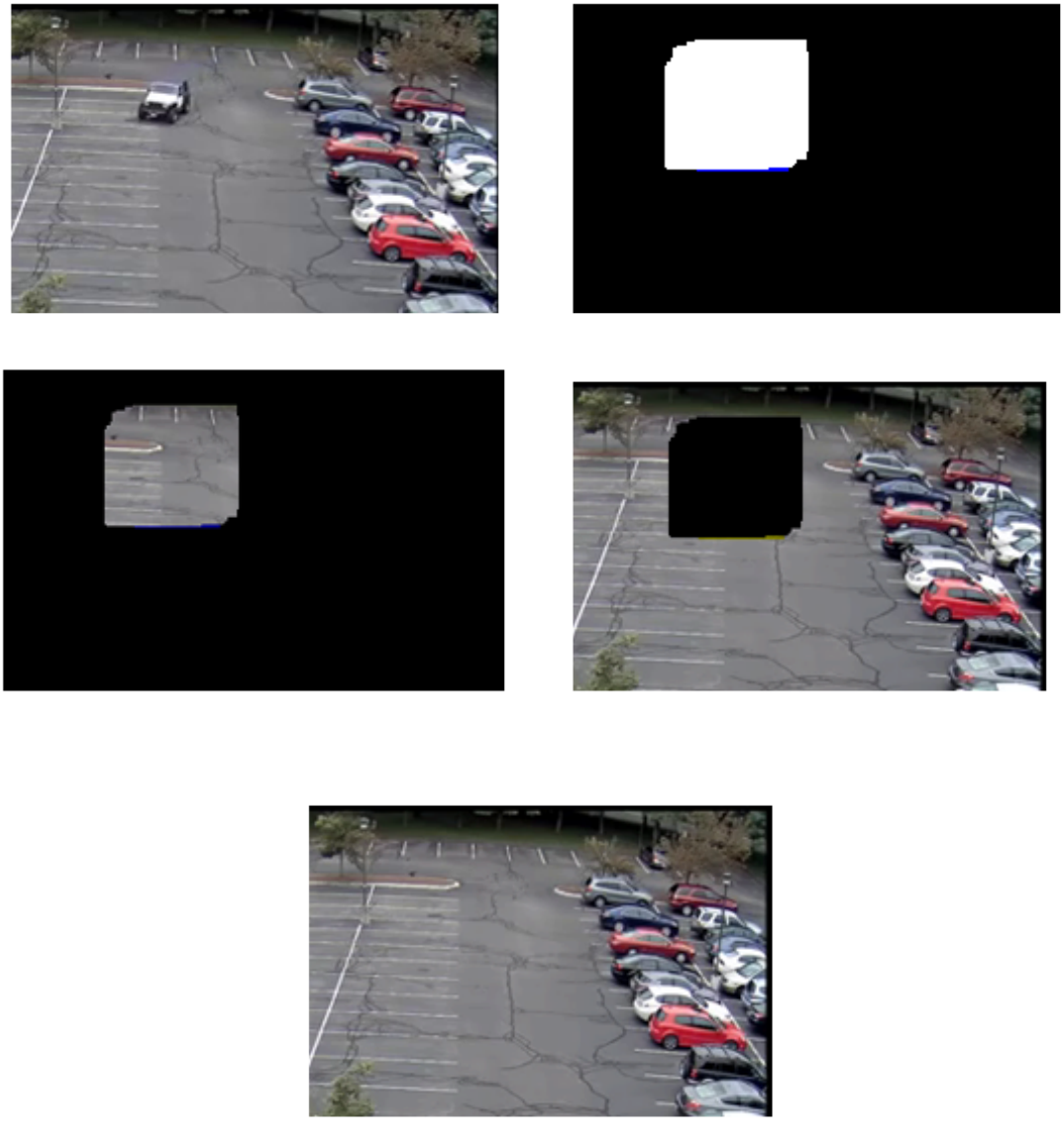


Figura 3.2: Imagens representando o processo de geração do fundo dinâmico. No topo o quadro do vídeo e a máscara obtida. No centro a imagem do fundo anterior na região da máscara e a média entre os valores do quadro atual e do fundo anterior nos outros pontos e abaixo o fundo novo construído.

para cumprir os objetivos desejados. Um objeto que aparece na imagem de fundo, é um objeto que parou de se mover e passou a ser um elemento estático do ambiente que está sendo filmado. No contexto do trabalho, isso significa que provavelmente um veículo estacionou. A região obtida pela subtração do fundo em momentos diferentes indica a posição de novos objetos estáticos. Esse processo é descrito na seção 3.4.

## 3.4 Diferenças entre fundos

Como foi mencionado ao final da seção 3.3, o programa identifica que um objeto que estava em movimento passou a estar parado quando este objeto aparece na imagem de fundo do vídeo. Analogamente, é possível identificar que um objeto que estava parado passou a se mover quando ele desaparece do *background* dinâmico. Fica clara então a importância de se reconhecer quando elementos são adicionados ou removidos do fundo.

Para identificar mudanças significativas na imagem de fundo do vídeo sendo observado, este programa utiliza o processo de subtração de imagens descrito na seção 2.4 com algumas modificações. A $t$ quadros do vídeo - onde $t$ é um valor determinado na etapa de calibração do sistema - o programa gera uma imagem de diferença entre o *background* atual e o *background* a $t$ quadros atrás. Essa imagem $Df$ de diferença é obtida através do mesmo processo utilizado para se obter a imagem $D$ de diferença entre dois *frames* na seção 3.1. Essa imagem é submetida a um processo de fechamento, ou seja, a aplicação de uma erosão seguida de uma dilatação. Essa operação tem como objetivo uniformizar os contornos na imagem $Df$ e pricipalmente, eliminar manchas que são pequenas demais para serem consideradas. Por causa disso, a escolha do elemento estruturante para a erosão nessa etapa é muito importante. Esse elemento é o parâmetro que discerne entre os contornos relevantes e os irrelevantes em $Df$.

Depois que os contornos que não são de interesse para o programa são eliminados de $Df$ é preciso submeter a imagem por um processo de limiarização para que ela seja compatível com as técnicas que serão aplicadas em seguida. Se a imagem do vídeo tem apenas um canal, o *thresholding* aplicado é trivial, cada ponto que não é preto da imagem se torna branco, e os outros continuam pretos como antes. Quando o vídeo possue três canais, é preciso aplicar essa limiarização em cada canal e depois criar uma única imagem binária com as informações de cada canal através da operação lógica *or*. Essa operação faz com que um determinado ponto da imagem resultando seja branco se o ponto de uma das duas imagens sendo operadas for branco e preto caso contrário. Ela é descrita pela equação 3.3:

$$Or(x,y) = \begin{cases} A(x,y) & ,A(x,y) = 255 \\ B(x,y) & ,B(x,y) = 255 \\ 0 & ,A(x,y) = 0 e B(x,y) = 0 \end{cases} \tag{3.3}$$

Onde $Or(x,y)$ é o valor do *pixel* na posição $(x,y)$ da matriz resultante e $A(x,y)$ e $B(x,y)$ são os valores dos *pixels* nas duas matrizes sobre as quais a operação está sendo aplicada.

O resultado dessa operação sobre as imagens de binarizadas dos canais R,G e B dois a dois é uma imagem completamente preta, exceto por manchas brancas que representam regiões onde houve uma diferença entre os fundos nos dois momentos. Nessas regiões se

pode assumir que ou um objeto novo se integrou ao fundo ou um objeto que antes fazia parte do *background* deixou de fazer.

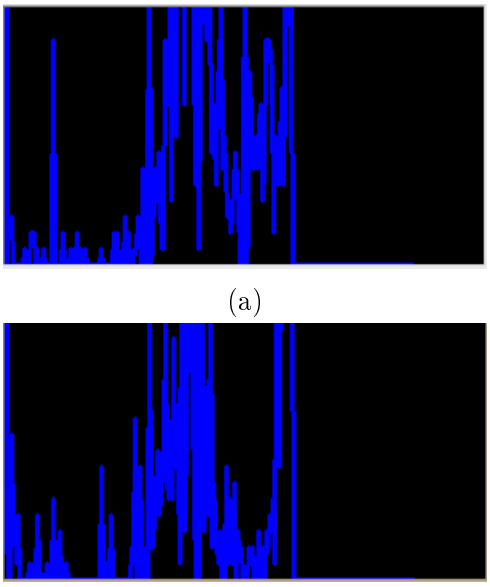
## 3.5 Identificação de objetos no fundo

Apenas encontrar as regiões aonde houve uma diferença entre os fundos não é suficiente para atingir os objetivos do trabalho. Apesar destas regiões indicarem bem precisamente os locais da imagem onde um objeto apareceu ou desapareceu no fundo, elas não são capazes de fornecer algumas informações cruciais. Por exemplo, como saber se uma determinada mancha em $Df$ representa um novo objeto no fundo ou um carro que saiu de uma vaga aonde estava estacionado? Como determinar se o objeto que causou essa diferença sequer é um veículo? É preciso então voltar a analisar as informações contidas na imagem original do quadro do vídeo para identificar corretamente o que está acontecendo na região indicada. Dessa vez porém, o programa pode analisar apenas as pequenas regiões correspondentes àquelas em $Df$.

Para responder a primeira pergunta apresentada no parágrafo anterior, o programa se utiliza de uma comparação de histogramas. Primeiramente é preciso extrair um histograma de uma região da imagem que seja suficientemente semelhante a uma vaga vazia. Esse histograma $H_0$ serve como um histograma de "controle", é o histograma que indica uma vaga vazia. É importante determinar $H_0$ de forma dinâmica, a fim de evitar que mudanças na iluminação do vídeo causem resultados incorretos. Para isso, o programa extrai esse histograma em intervalos de tempo regulares, de um espaço na imagem onde é sabido que não há carros. A determinação desse espaço pode ser feita manualmente no momento de instalação do programa ou posteriormente através de uma função de calibração. É importante escolher uma região onde ocorrem muito poucas mudanças na imagem de fundo e que seja semelhante ao fundo da imagem sem nenhum elemento. Em outras palavras, para o caso desse programa, deve-se escolher uma região que representa o chão do estacionamento e que não se espera que seja ocupada por objetos.

Munido com $H_0$ o programa extrai o histograma $H_1$ da área do fundo atual correspondente à cada mancha branca presente em $Df$. Em seguida o histograma $H_2$ da mesma região no fundo a $t$ quadros atrás é extraído. O programa faz uma comparação entre $H_1$ e $H_2$. Se os histograma forem suficientemente distintos, é considerado que houve uma mudança significativa no *background*. Nesse caso, o programa compara $H_1$ e $H_2$ com $H_0$. Essa segunda comparação tem como objetivo determinar se um objeto foi adicionado ou removido do fundo. Quando o $H_1$ é semelhante a $H_0$, pode-se assumir que a região observada mostra uma vaga vazia. Por outro lado, quando $H_2$ é semelhante a $H_0$, é muito provável que a região observada mostra um objeto novo que se integrou ao fundo.

Os histogramas são comparados através de um pequeno conjunto de parâmetros. Os parâmetros escolhidos não podem ser estritamente dependentes do tamanho da região analisada, já que não há garantia de que a região de controle terá o mesmo tamanho da região indicada em $Df$. O primeiro desses parâmetros é o nível de cinza que ocorre com mais frequência na região. Esse parâmetro é suficiente para determinar se as áreas analisadas são diferentes demais umas das outras, uma vez que imagens com níveis de cinza predominante demasiado diferentes certamente serão consideravelmente distintas. Porém, não se pode assumir o contrário, que uma coincidência nesse valor indica imagens semelhantes. Para complementar esse parâmetro o programa usa a distribuição dos níveis



(a)

(b)

Figura 3.3: (a) O histograma $H_1$ da figura 2.3a na área indicada por 2.3c; (b)O histograma $H_2$ da figura 2.3b na área indicada por 2.3c

de cinza. Se dois histogramas apresentam um distribuição de valores proporcional, é muito provável que essas duas imagens sejam parecidas.

A figura 3.3 mostra esses dois histogramas para os dois momentos representados na figura 2.3. Nela é visível a diferença da distribuição dos valores de cinza na imagem.

## 3.6 Verificação de objetos no fundo

Uma vez determinado que um objeto novo se integrou ao fundo da imagem, é preciso garantir então que o objeto que se integrou ao fundo na posição da vaga é realmente um veículo. O sistema não deve acusar que uma vaga foi ocupada quando algum outro corpo está sobre a vaga, como por exemplo uma pessoa que interrompeu seu trajeto sobre uma vaga ou a porta de um carro na vaga adjacente que ficou aberta por muito tempo. Observe que esse cuidado e suficiente para garantir que o sistema não libere vagas incorretamente quando um objeto que não é um carro sai dessa vaga, uma vez que a vaga nunca é marcada como ocupada nesse caso.

Para identificar o objeto novo que apareceu no fundo da imagem, o sistema se utiliza pricipalmente de um recurso, a comparação da posição de um objeto identificado com os finais dos rastros identificados na seção 3.2. Antes de afirmar que uma vaga foi ocupada por um veículo quando identifica uma mudança no fundo sobre a área de uma vaga, o programa percorre o vetor de rastros obtidos e identifica se algum deles tem início em um ponto fora de qualquer vaga e o final dentro da região do objeto identificado. Se esse for o caso, garante-se que esse objeto que apareceu estava anteriormente em movimento. Isso significa que a diferença na região não foi causada por um objeto que surgiu subitamente na cena naquela posição, como um grande pico de ruído, uma porta que se abriu em um

carro adjacente ou alguma mudança de cor sobre um carro parado, possivelmente causada por um vidro abrindo ou o reflexo de uma luz.

Como um complemento a esse recurso, o programa se utiliza de um algoritmo simples de classificação para determinar a probabilidade de que o objeto seja de fato um carro. Esse algoritmo usa informações sobre as bordas presentes nas imagens para diferenciar principalmente veículos de pessoas. Características utilizadas para esse sistema de classificação incluem a largura, o comprimento, quanto uma determinada cor ocupa no objeto e a presença de alguns componentes como pára-brisas que ajudam a confirmar que o objeto é um veículo. Esse sistema será descrito com mais detalhes posteriormente.

# Capítulo 4

# Resultados Parciais

## 4.1 Testes em imagens estáticas

Os primeiros testes aplicados no programa foram feitos com imagens estáticas ao invés de imagens de vídeo. O propósito destes testes era verificar a eficácia do algoritmo de diferenciação de fundos detalhado na seção 3.4. Cada imagem estática fazia o papel de uma imagem de fundo que teria sido extraída de um vídeo da câmera no estacionamento. As imagens para os primeiros testes foram obtidas com o auxílio de um *drone* de controle remoto que fez fotografias aéreas do estacionamento próximo aos prédios PAT e PJC no campus da Universidade de Brasília. A fim de simular melhor a imagem que seria obtida por uma câmera instalada em um poste no estacionamento, um programa de edição de imagens foi utilizado para recortar pedaços da imagem com um nível maior de aproximação. Essas foram as imagens utilizadas para os testes.

A primeira etapa dessa bateria de testes foi testar o caso mais simples de todos: comparar uma imagem de fundo com a imagem do estacionamento completamente desocupado. Através de manipulação de imagens, foram compostas diversas imagens com carros de cores diferentes, estacionados em várias combinações distintas de vagas. Cada imagem é comparada a imagem da seção de estacionamento correspondente com todas as vagas desocupadas. Aqui, uma diferença entre as duas imagens que esteja em uma região que se sabe que é uma vaga sempre representa um carro que passou a ocupar aquela vaga. Uma terceira imagem mostra através de círculos coloridos quais vagas estão ocupadas na imagem de teste. Nessa etapa de testes, o programa obteve uma taxa de acerto de 100%

O bom funcionamento nesse caso simples é de grande importância. Se o programa for inicializado com o estacionamento vazio, a primeira comparação de fundo será dessa maneira. Se o algoritmo acerta quais vagas estão ocupadas, ele agora está munido da informação completa dos estados das vagas. A partir desse momento, é possível interpretar de forma mais simples as diferenças na imagens de fundo obtidas no vídeo. Uma diferença significativa sobre uma região de vaga livre, significa que a vaga foi ocupada. Se a vaga estava ocupada, ela foi liberada entre uma verificação e outra. Assim, o algoritmo consegue indicar o estado das vagas com uma taxa muito alta de acerto através de operações simples, desde que o estado inicial seja conhecido.

# Referências


[1] 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. 3

[2] Wen-Yuan Chen and Chuin-Mu Wang. The dynamic background generation scheme using an image frame statistical comparison method. *International Journal of Advancements in Computing Technology*, 4(18), 2012. 14

[3] José Eustáquio Rangel de Queiroz and Herman Martins Gomes. Introdução ao processamento digital de imagens. *RITA*, 13(2):11–42, 2006. 11

[4] 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. 3

[5] Rafael C Gonzalez. *Digital image processing*. Pearson Education India, 2009. 5, 6

[6] Ana Beatriz Vicentim Graciano. *Rastreamento de objetos baseado em reconhecimento estrutural de padroes*. PhD thesis, Universidade de São Paulo, 2007. 5, 7

[7] Zhai Hai-tao, Wu Jian, Xia Jie, and Cui Zhi-ming. Self-adaptive detection of moving vehicles in traffic video. In *The 2009 International Symposium on Web Information Systems and Applications (WISA 2009)*, page 449, 2009. 14, 15

[8] IBGE. *Introdução ao processamento digital de imagens*. IBGE, 2000. 6

[9] 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. 2

[10] Norberto Malpica, Carlos Ortiz de Solorzano, Juan José Vaquero, Andrés Santos, Isabel Vallcorba, José Miguel Garcia-Sagredo, and Francisco del Pozo. Applying watershed algorithms to the segmentation of clustered nuclei. 1997. 9

[11] Ogê Marques Filho and Hugo Vieira Neto. *Processamento digital de imagens*. Brasport, 1999. vi, 6, 7, 11

[12] 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. 2, 3

[13] B Shoushtarian. A practical approach to real-time dynamic background generation based on a temporal median filter. *Journal of Sciences, Islamic Republic of Iran*, 14(4):351–362, 2003. 13, 14

[14] Nicholas True. Vacant parking space detection in static images. *University of California, San Diego*, 2007. 4, 10

[15] AI VKL. Jain,"fundamentals of digital image processing,", 1989. 8