UNIVERSIDADE DO ALGARVE

SISTEMA DE VISÃO INTELIGENTE DE BAIXO CUSTO PARA PARQUE DE ESTACIONAMENTO

DAVID VAZ DE SARAIVA

Dissertação

Mestrado em Engenharia Elétrica e Eletrónica

Trabalho efetuado sob a orientação de:
Professor Doutor João Miguel Fernandes Rodrigues

2015

SISTEMA DE VISÃO INTELIGENTE DE BAIXO CUSTO PARA PARQUE DE ESTACIONAMENTO

Declaração de autoria de trabalho

Declaro ser o autor deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

Resumo

Sistemas inteligentes para parque de estacionamento contribuem para a redução de tráfego, facilitando aos utilizadores a procura de um lugar de estacionamento, o que permite poupar tempo e dinheiro. No entanto, tais sistemas requerem, geralmente, instalações dispendiosas. Na presente dissertação, é proposto um sistema de visão computacional de baixo custo baseado no Raspberry Pi 2 e numa HD webcam para a deteção de lugares livres. O algoritmo de visão é baseado num detetor de arestas e num limite dinâmico atribuído a cada lugar para inferir sobre o estado do lugar. O sistema foi testado em ambiente real com diferentes condições atmosféricas e oclusões parciais entre os veículos. O algoritmo corre em tempo real e tem uma taxa de sucesso de 99.6%.

Palavras-chave: custo reduzido, sistema inteligente, visão computacional, estacionamento.

Abstract

Intelligent parking systems can reduce the traffic, making it easier for users to find free parking spaces, which saves time and money. However, these systems, often require expensive installations. In this project, a proposal for a low cost computer vision framework based on Raspberry Pi 2 and a HD webcam for the detection of free parking spaces is presented. The vision algorithm is based on edge detection and a dynamic threshold within each parking space to infer its status. The system was tested in a real environment with different weather conditions and partial occlusions. The algorithm runs in real time and has a success rate of 99.6%.

Keywords: low cost, intelligent system, computer vision, car parking.

Para a minha mãe que faz o impossível por mim...

Agradecimentos

Gostaria de agradecer e dedicar este trabalho minha à mãe e irmã, por toda a força e confiança que depositaram em mim. Gostaria de agradecer também, ao Dário Lourenço por todo o apoio que me tem dado. Quero também agradecer, a Bruno Costa, pela sua disponibilidade e incentivo em todas as etapas do trabalho.

Queria também, fazer um agradecimento ao meu orientador, o Prof. Doutor João Rodrigues, por ter aceite a orientar-me nesta dissertação.

Por fim, mas não menos importantes, agradeço a um conjunto de pessoas que influenciaram a minha vida positivamente de forma a chegar a esta etapa: Guliver Nunes, Albertina Nazário, Monika Bartosová, Joana Lopes, Hugo Santos, Hélio Correia, Vítor Cardeira, Jorge Saraiva e Mário Saraiva.

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de Abreviaturas

UWB	*University of West Bohemia*
HD	*High-Definition*
BD	Base de Dados
RoI	*Region of Interest*
API	*Aplication Programming Interface*
CPU	*Central Processing Unit*
RAM	*Random-Access Memory*
I/O	*Input/Output*
GPU	*Graphics Processing Unit*
OpenCV	*Open Source Computer Vision*
OpenCL	*Open Computing Language*
MOG	*Mixture of Gaussian*
BSD	*Berkeley Software Distribution*
CIFS	*Common Internet File System*
RPi	Raspberry Pi

1

Introdução

Sistemas inteligentes para parque de estacionamento podem reduzir o tráfego em mais de 30% nas áreas de implementação (SFpark, 2014a).

O sistema a implementar deve ser de fácil manutenção e com um custo o mais reduzido possível. Nesse sentido, foi escolhido um sistema de visão computacional baseado numa webcam HD e numa unidade de processamento de baixo custo (o Raspberry Pi 2). No entanto, esta opção introduz uma série de desafios ao nível do algoritmo de visão, uma vez que a Visão Computacional requere, em geral, elevado processamento que não se obtém neste tipo de plataformas de baixo custo.

O estudo iniciou-se com a investigação do estado da arte e com o teste de algoritmos de visão, primeiro utilizando o SimpleCV e posteriormente o OpenCV, bibliotecas

para visão computacional que se apresentam com mais detalhe no capítulo 2.

Em termos de aplicação, o estudo foi separado em dois períodos. Inicialmente o objetivo foi o estudo de algoritmos, sendo utilizado o PC, numa segunda fase, o código foi implementado no Raspberry Pi, a fim de avaliar a sua performance num sistema *low-cost*.

Para apoiar a criação dos algoritmos de visão e respetivos testes, foi criado e utilizado um conjunto de ferramentas que se apresenta na secção 3.2.

1.1 Enquadramento

O trabalho proposto nasceu da necessidade sentida pelo *staff* da University of West Bohemia, República Checa (UWB, 2015b), de forma a reduzir o tempo despendido na procura de um espaço para o carro no parque de estacionamento da Universidade. Integrando o programa de mobilidade internacional Erasmus e em colaboração direta com o Professor Doutor Vladimir Pavliček (página: UWB (2015a)), docente na *Faculty of Electrical Engineering*, deu-se início ao estudo para a presente dissertação em Pilsen, República Checa.

O sistema proposto é composto por dois módulos: (a) inicialização, onde os lugares de estacionamento são selecionados e (b) deteção do estado dos lugares, que é baseado em deteção de arestas, extração de regiões específicas de cada lugar e computação de um limite dinâmico em termos dos pixeis que representam as arestas e o número total de pixeis da região específica de cada lugar com o intuito de inferir sobre o seu estado.

Na presente dissertação apresenta-se o resultado obtido com a colocação de um módulo na secção H do parque da Universidade of West Bohemia (v.d. Fig. 3.1). O sistema a implementar irá basear-se na colocação de vários módulos, nas várias secções do parque.

O contributo deste trabalho é a obtenção de um sistema de baixo custo, simples, de fácil manutenção, preparado para funcionar com diferentes condições atmosféricas

(sol, chuva, nublado) e capaz de suportar oclusões parciais entre carros.

Em comparação com os atuais sistemas comercias para parque de estacionamento (SFpark, 2014b; Fastprk, 2015; Streetline, 2015), o sistema proposto apresenta a vantagem de não exigir qualquer custo com equipamento a ser colocado no terreno, dado que a prática usual é a colocação de um sensor, por vezes dois sensores para uma melhor precisão (por exemplo, ultrasons) em cada lugar de estacionamento; ao custo do equipamento, acresce o custo da manutenção desses sistemas.

1.2 Objetivos

Com o presente trabalho pretendeu-se criar um produto de custo reduzido no contexto de uma aplicação para parque de estacionamento.

Para cumprir este objetivo uma série de tarefas foi atribuída:

- Pesquisar e escolher uma plataforma para o sistema - *hardware* (Raspberry Pi);
- Estudar bibliotecas de visão computacional compatíveis com o *hardware*;
- Estudar e propor um sistema usando visão computacional para determinar o estado do parque;
- Implementar e testar os algoritmos no contexto real do parque de estacionamento;
- Avaliar resultados e fazer correções;
- Complementarmente, desenvolver um conjunto de *scripts* para correr em *background* para fazer a manutenção das fotos no servidor.

1.3 Contributos da dissertação

Existem três pontos a salientar:

- é proposto um sistema de parque de estacionamento com custo reduzido, utilizando o Raspberry Pi e visão computacional, resultando num produto de fácil implementação em comparação com soluções comerciais, que em geral requerem a instalação de equipamento adicional, com os inerentes encargos financeiros (Fastprk, 2015; SFpark, 2014a; TCS, 2015b; Streetline, 2015);

- os benefícios diretos da utilização do sistema de parque de estacionamento para os utilizadores (tempo e dinheiro);

- desenvolver uma "câmara IP inteligente"de baixo custo, baseada num Rasperry PI e uma webcam.

Em termos dos benefícios introduzidos com o sistema de parque de estacionamento proposto, é de salientar o tempo e o dinheiro que podem ser poupados aos utilizadores. Por exemplo, se o utilizador obtiver a informação da localização de um espaço ou secção livre, este pode dirigir-se diretamente para o mesmo. No caso de parque cheio, o utilizador conclui de imediato que não existe lugar. Se for considerado, todo o tempo, energia e combustível que é despendido nestas tarefas exaustivas, e multiplicados pelos dias úteis do ano, e ainda multiplicado por 3, 4, 10 anos, a conclusão é que um sistema de parque de estacionamento inteligente não é apenas beneficial, é necessário. De salientar, que um estacionamento mais organizado contribui para segurança de todos os utilizadores do espaço.

O sistema proposto é suportado por um baixo orçamento, uma vez que os encargos financeiros foram reduzidos ao máximo, com a utilização de um módulo de processamento de baixo custo (Raspberry Pi) e uma livraria *opensource* de visão computacional (OpenCV). Uma vez que não existe a necessidade de *hardware* adicional a ser instalado no parque, poupa-se também nos custos de manutenção.

No tempo em que as câmaras IP ganham mais funcionalidades, na presente dissertação propõe-se trazer mais "inteligência", pela integração dos seguintes componentes: Raspberry Pi + câmara USB + Sistema de Visão Computacional para parque de estaci-

onamento. Basicamente, está-se a desenvolver uma câmara inteligente de baixo custo, uma vez que este tipo de câmaras, atualmente, tem um preço demasiado elevado.

1.4 Vista Geral

Neste capítulo foi introduzido o tema e especificados os objetivos, enquadramento e os contributos da presente dissertação.

No Capítulo 2 é descrito o estado da arte, sendo apresentados alguns dos produtos comerciais e trabalhos de investigação que se encontram relacionados de alguma forma com esta dissertação.

No Capítulo 3 é apresentado o sistema, e especificado o conjunto de ferramentas criadas e utilizadas no decorrer desta dissertação e que podem ter aplicação em futuros projetos.

No Capítulo 4 encontra-se a análise dos resultados obtidos.

Por último, no Capítulo 5, apresentam-se as conclusões, considerações finais, bem como propostas para trabalhos futuros.

2

Estado da Arte e Conceitos Gerais

As atuais exigências monetárias, de energia (escassez e consequências da utilização dos combustíveis fósseis) e de tempo tornam imperativa a criação e implementação de produtos que promovam o aumento da eficiência dos condutores na busca por lugares em parques de estacionamento. No presente capítulo, são apresentados alguns conceitos gerais, técnicas e bibliotecas utilizadas nesta dissertação, bem como soluções já existentes no mercado.

2.1 A tecnologia para parque de estacionamento

Um sistema inteligente para parque de estacionamento consiste num conjunto de módulos funcionais, tais como: detetor, dispositivos de rede, servidores, interfaces de visualização, parquímetros (Fastprk, 2015). Dependendo da aplicação, certos módulos podem ser considerados opcionais.

O **detetor** é um dos componentes fundamentais num sistema de estacionamento. É ele que fornece a informação que permite inferir sobre o estado dos lugares. Um detetor pode ser baseado em diferentes fontes (ultrasónicos, magnéticos, óticos, infravermelhos, entre outros), a análise mais detalhada dos diferentes tipos de detetores é feita na secção 2.2. Dependendo do detetor/sensor assim é necessária a sua instalação sob ou sobre o lugar, ou no caso das câmaras, tem que se proceder à sua instalação num ponto estratégico. Entre outras, uma das vantagens da utilização de câmaras como detetores, é que estas abrangem um conjunto de carros, em vez da colocação personalizada de um ou dois sensores por lugar (SFpark, 2014a).

No caso das aplicações que utilizam a instalação de sensores (com ou sem fios), um conjunto destes comunica com um **dispositivo de rede**, que envia informações sobre o parque a fim de serem processadas pelo servidor central. Os dispositivos de rede encontram-se uniformemente distribuídos e colocados próximo dos sensores. Em SSET (2015), a comunicação é feita por fio utilizando o protocolo RS-485; em SFpark (2014a) é utilizada a comunicação sem fios com recurso ao GPRS; em Idris et al. (2009) é proposto um sistema que utiliza ZigBee. Conclui-se que relativamente ao que respeita às interfaces de comunicação estas variam de aplicação para aplicação, não havendo um standard.

Os **servidores** são responsáveis por receber os dados de vários dispositivos de rede, fazer a análise dos dados, armazená-los, e disponibilizar os resultados aos utilizadores, por exemplo, através de uma aplicação Web ou móvel, ou de painéis colocados que se encontrem à entrada do parque.

Existem diversos tipos de **interfaces de visualização** que podem ser utilizados no contexto de um sistema de parque de estacionamento, desde LEDs individuais sobre cada lugar de estacionamento (TCS, 2015b), a painéis de informação na entrada do parque (TCS, 2015d,c), bem como aplicações para telemóveis (SFpark, 2014a), serviços de mensagens (Short Message Service - SMS) ou mesmo uma página Web (SFpark, 2014a). Contudo, em SFpark (2014a) o serviço de SMS foi descontinuado por falta de aderência.

Os **parquímetros** são um dos módulos do sistema de parque de estacionamento que viu nos últimos anos, uma grande evolução a nível das suas capacidades e funcionalidades. Tem-se assistido à introdução de novos métodos de pagamento, substituindo as tradicionais moedas, por exemplo, o pagamento por telemóvel, em Portugal (v.d. Faro (2015)). No SFpark, o preçário varia dinamicamente, procurando dar resposta adequada à procura e oferta do estacionamento, e tendo em vista um lugar vago em cada região. Deste modo, em locais com muitos lugares vagos, os preços são mais baixos, por outro lado, em locais onde existe escassez de lugares de estacionamento os preços sobem, numa tentativa de promover outros locais de estacionamento menos congestionados. Contudo, o desenvolvimento de novas funcionalidades tem que ser acompanhado por desenvolvimento tecnológico, pelo que os pagamentos através de cartão de crédito ainda não se encontram, por norma implementados, devido às exigências em termos de segurança.

A nível de *hardware*, os parquímetros evoluem de forma a possibilitar outras interfaces de pagamento. Em termos energéticos, o desenvolvimento passa pela colocação de baterias e de painéis solares. Verifica-se contudo que, os novos parquímetros apresentam um esquema mais complexo e componentes mais sensíveis, v.d. SFpark (2014a).

A nível de *software*, o desenvolvimento faz-se em diversas áreas com a introdução de novas funcionalidades:

- preçários dinâmicos que reflitam o estado de ocupação de um parque;

- introdução de novas formas de pagamento (requer a implementação das comunicações e protocolos de segurança com diferentes operadores);
- compatibilidade com diferentes fornecedores de parquímetros;
- interface de visualização intuitiva de fácil utilização;
- modificações do sistema não necessitarem do acesso a cada parquímetro individualmente.

2.2 Detetores de veículos

Existem diferentes tipos de detetores, sendo os mais comuns para parques de estacionamento os seguintes:

(a) indução (Nortech, 2015);

(b) magnéticos (Buecheler, 2014; SFpark, 2014a; Nedap, 2014; Fastprk, 2015; Parksol, 2013a);

(c) ultra-som (TCS, 2015a; Parksol, 2013b);

(d) óticos- infravermelhos ou laser (Nedap, 2014; SmartParking, 2015);

(e) os baseados em vídeo e imagem (Prabha et al., 2014; Yamada et al., 2001).

Outros tipos de detetores para veículos (para tráfego, e outras aplicações) são baseados em:

(a) micro-ondas (Sedco, 2015);

(b) vetor de sensores acústicos (Kuhn et al., 1998);

(c) piezoelétricos (Diamond, 2015);

(d) foto-elétricos (Optex FA, 2014).

Certos detetores consistem em dupla deteção utilizando dois tipos de sensores: Magnético e Infravermelho, v.d. Nedap (2014), Magnético e Luminosidade, v.d. Buecheler (2014).

Os sensores de indução **(a)**, são compostos por três componentes: a malha ou *loop* (*performed* ou dente-de-serra), um cabo de extensão da malha e um detetor (esquema apresentado na referência Marsh (2000)). A malha é colocada sob a faixa de rodagem. As duas pontas do fio da malha são conectadas ao cabo de extensão, que por sua vez é ligado ao detetor. O detetor alimenta a malha criando um campo magnético à volta da área da malha. Uma frequência constante proveniente da malha é monitorizada pelo detetor, sendo a frequência base quando não existe um veículo sobre a malha. Quando um grande objeto de metal, tal como um veículo, se move sobre a malha, a frequência de ressonância aumenta. Este aumento de frequência é sentido e dependentemente do desenho do detetor, é responsável pela abertura ou fecho de um relé. O relé continuará fechado (ou aberto), enquanto o veículo se mantiver na malha, e a frequência não voltar à da base. O relé pode interagir com um diferente número de dispositivos, nomeadamente: portões, semáforos, etc. Este tipo de detetores é utilizado para controle de parques de estacionamento que utilizam cancelas, portões de segurança, em restaurantes com *drive-in*, entre outros. Apresentam como principais desvantagens (Kon, 1998) o facto de ter um processo de instalação muito sensível, só pode ser instalado em pavimento em "bom estado", e tem que ser reinstalado sempre que é colocada uma nova camada de pavimento.

Os sensores magnéticos **(b)** podem ser divididos em duas categorias (Kon, 1998): os passivos e os ativos. Tal como o princípio de funcionamento do detetor de indução (esquema representado na referência Daubaras et al. (2012)), ambos têm como base de funcionamento que a passagem de um grande objeto de metal perturba o campo magnético. O detetor do tipo ativo é denominado magnetómetro e funciona da mesma forma que o detetor de malha de indução, exceto que consiste numa bobina de fio em torno de núcleo magnético, medindo as mudanças do campo magnético causadas

pelo veículo. Estas mudanças podem ser utilizadas tanto para detetar a presença de um veículo, ou a sua passagem. O detetor de tipo passivo mede simplesmente a mudança de fluxo no campo magnético terrestre causado pela passagem do veículo. Estes detetores passivos só podem ser utilizados para veículos em movimento, não sendo portanto detetores de presença. Este tipo de detetores, ainda que largamente utilizado, apresenta as seguintes desvantagens:

- interferência eletromagnética, que afeta a precisão dos sensores;
- é necessária, por vezes, a instalação de 2 sensores por lugar (SFpark, 2014a);
- a verificação dos dados de cada sensor é um processo moroso, envolvendo levantamentos no local;
- menor precisão na deteção de carros de menores dimensões, ou de veículos que não sejam de 4 rodas;
- os sensores constituem um equipamento novo nas ruas, pelo que outras companhias ou serviços, podem inadvertidamente destruí-los;
- a bateria tem um período de vida entre 3 a 5 anos, sendo depois necessário proceder à sua manutenção;
- é necessária instalação de cada sensor individualmente.

Os detetores de ultra-som **(c)** transmitem ondas sonoras entre 25 kHz a 50 kHz, e utilizam a energia refletida para analisar e detetar o estado do lugar (Kianpisheh et al., 2012). Ondas ultra-sónicas são emitidas do detetor a cada 60 milissegundos, e a presença ou a ausência de veículos é determinada pela diferença de tempos entre os sinais de emissão e receção. Estes detetores podem servir para fazer a contagem de veículos, ou determinar o estado de ocupação de cada lugar. A sua instalação deve ser feita o mais perpendicularmente possível com o alvo e são sensíveis às mudanças

de temperatura e a fortes ventos. O princípio de funcionamento está ilustrado em Kianpisheh et al. (2012).

O princípio de funcionamento dos detetores de infravermelhos **(d)**, tal como descrito em Kon (1998), é baseado na iluminação do alvo por parte do emissor e pela receção da energia recebida num pixel ou conjunto de pixeis do detetor. Os dados são depois processados utilizando vários algoritmos de processamento de sinal para extrair a informação desejada. Algumas das vantagens são que este tipo de detetores podem funcionar tanto de dia como de noite e podem ser montados de lado, ou noutras configurações. Em termos de desvantagens, salienta-se que os detetores de infravermelhos são sensíveis a condições atmosféricas rigorosas.

Os detetores baseados em imagem/vídeo **(e)** podem ser descritos como a combinação de *hardware* e *software* para extrair informação de imagens obtidas através de uma câmara (Kon, 1998), como na presente dissertação. Estes dispositivos podem ir desde as câmaras de vigilância, às câmaras de TV convencionais, ou às câmaras de infravermelhos. Nesta dissertação, é proposta uma HD webcam integrada com um dispositivo de processamento, formando um produto que adquire e processa as imagens. As vantagens deste tipo de detetores, é que pode utilizar a infraestrutura já presente no local (CCTV), abrange múltiplos lugares, é de fácil implementação, e as zonas de deteção podem ser facilmente programadas. Contudo, é necessário ultrapassar certos desafios, como os impostos pelas diferentes condições meteorológicas, luminosidade, sombras e reflexos do pavimento.

O número de sensores utilizado no projeto piloto de São Francisco foi cerca de 11700, para 8000 lugares de estacionamento (SFpark, 2014a), uma vez que certos lugares requereram a instalação de dois sensores, para uma correta leitura. Foi ainda necessário instalar cerca de 300 dispositivos de rede, montados em postes. Como aspetos negativos desse projeto salienta-se: a instalação individual de cada sensor, o facto da bateria ter apenas um período útil de 3 anos, sendo depois necessário proceder à manutenção de todos os sensores, o que implica um orçamento elevado - foram

apresentados custos superiores a 46 milhões de euros (SFpark, 2014b), sendo que 5.7 milhões de euros foram despendidos com os sensores, o que se traduziu num custo aproximado de 700 euros por lugar de estacionamento.

A taxa de precisão dos sensores requerida no SFpark (se o lugar está ocupado ou livre), foi de 92% (SFpark, 2014b). Os algoritmos de visão como os apresentados em Yamada et al. (2001); Ichihashi et al. (2009); Almeida et al. (2013), entre outros, apresentam uma taxa de sucesso superior a 95%.

A solução utilizando uma câmara e visão computacional para sistemas de parques de estacionamento, conforme proposto na presente dissertação, apresenta vantagens nos vários domínios, nomeadamente nos aspetos financeiros e de rentabilização de recursos.

2.3 Como abordar um sistema de Visão Computacional

De acordo com Szeliski (2010), o melhor procedimento é aquele que inicia com o "pensar"o problema e as técnicas que melhor se adequam a ele ao invés de começar exaustivamente com o teste de algoritmos. Este procedimento, que parte dos problemas para as soluções, é a solução típica no estudo da visão computacional. Primeiro, deve-se especificar a definição do problema e decidir quais são as restrições e os requerimentos do problema. De seguida, deve-se procurar técnicas já conhecidas por resultarem, e implementar algumas delas, para avaliar a sua *performance* e fazer uma seleção final. Contudo, de forma a que se chegue a uma solução prática e que funcione, é necessário possuir uma amostra realística dos dados a analisar, sintética ou do terreno onde o sistema será implementado.

A descrição de um sistema de processamento de informação (visual), pode ser divido de forma simples (Szeliski, 2010):

- **Teoria computacional** - Qual é o objetivo da computação (tarefa) e quais são os constrangimentos conhecidos ou que podem ser relacionados com o problema?

- **Representações e algoritmos** - De que maneira os dados de entrada, de saída e intermédios são representados e quais os algoritmos utilizados para calcular o resultado?

- **Implementação no *hardware*** - Como é que essas representações são passadas para o *hardware*? De que forma as restrições/constrangimentos do *hardware* podem ser utilizados para guiar e escolher as representações e os algoritmos?

É da convicção de Szeliski (2010), que uma análise cuidada da especificação do problema e o conhecimento dos constrangimentos, desde a formação da imagem à abordagem científica e probabilística do problema, tem que estar conectada com algoritmos robustos e eficientes (abordagem de engenharia) para desenhar/criar algoritmos de visão com sucesso.

O método científico é definido em Dictionaries (2015) como o procedimento ou método que consiste na observação, experimentação, formulação, teste e modificação sistemática de hipóteses.

2.3.1 Dificuldades em desenvolver o sistema de visão computacional?

Embora por vezes os problemas de Visão Computacional aparentem ser conceptualmente simples, eles exibem grande complexidade. O presente trabalho não é diferente - o conceito de olhar para um lugar de estacionamento e inferir sobre o seu estado (livre ou ocupado) é simples, mas a transmissão desta noção para o "computador"cria um conjunto de questões e problemas, nomeadamente:

- Como descrever ao "computador"o que é um carro? E será essa descrição suficiente para distinguir o carro da sombra de uma árvore?

- Como distinguir entre um carro e o fundo?

- Como inferir a posição do carro no parque? No presente trabalho, uma das principais dificuldades teve a ver com a perspetiva/ângulo de visualização do parque, uma vez que não foi possível obter uma imagem do parque vista de cima (*birds view*). No parque estudado, os carros sobrepõem-se uns aos outros, existindo a possibilidade de oclusões (por exemplo, um carro tapar parcialmente a vista do lugar adjacente, não deixando portanto "ver"diretamente se este está ocupado ou livre). Um dos principais pontos fulcrais no trabalho desenvolvido, consiste na resolução deste problema.

- Quais são os passos necessários para o reconhecimento do estado do lugar?

- O algoritmo é adequado para os constrangimentos de *hardware*, considerando um CPU de baixa performance e de recursos limitados?

2.4 Produtos existentes no mercado

Os sistemas de parque de estacionamento inteligentes são um tópico amplamente estudado, e.g., Yamada et al. (2001); Yusnita et al. (2012); Parking (2015); SFPark (2015); Streetline (2015); Prabha et al. (2014); Fastprk (2015); True (2007).

Existe um conjunto de algoritmos como os de True (2007); Yamada et al. (2001); Yusnita et al. (2012), entre outros, que utiliza características do carro, tais como: capô, janelas, faróis, pára-choques, etc., para inferir sobre o estado de ocupação de um lugar, no entanto, o efeito das oclusões parciais reduz significativamente a região de interesse de cada lugar (RoI- *Region of Interest*), não sendo viável a extração e análise desses segmentos.

A utilização de algoritmos de subtração de fundo (*background subtraction*) é amplamente utilizada em aplicações de vídeo-vigilância (Brutzer et al., 2011; Reddy et al., 2013), no entanto, existem vários desafios a serem ultrapassados, nomeadamente variações graduais e repentinas de luminosidade, sombras, a distinção entre o objeto e a

sua sombra, fundo dinâmico (por exemplo, o efeito do vento nas árvores), as diversas velocidades dos carros, que traz implicações a nível dos parâmetros de atualização do fundo e do ritmo de aprendizagem. Nestes casos, os carros que circulam lentamente, são apenas parcialmente detetados, por outro lado, os carros que circulem demasiado rápido para o intervalo entre as imagens, podem também não ser detetados. A afinação dos parâmetros para um tipo de movimento, piora a deteção no outro. De salientar que, devido à capacidade reduzida de processamento do Raspberry Pi, somente a operação de atualizar o fundo, no modelo B+, demora cerca de 1.65 segundos e cerca de 0.41 segundos, no modelo 2 B. Torna-se portanto necessário encontrar um algoritmo robusto, que não seja sensível a oclusões parciais e que seja eficiente para correr num módulo de processamento de custo reduzido.

No campo científico, bem como no mercado são encontradas soluções no âmbito de sistemas de parques de estacionamento inteligentes, e que estão em alguns pontos relacionados com a implementação aqui proposta.

Em Prabha et al. (2014), é proposto um sistema com o objetivo de monitorização de lugares de estacionamento utilizando o Raspberry Pi modelo B+ com o Sistema Operativo Raspbian e com uma webcam conectada ao Raspberry Pi. Em termos de *software*, foi utilizado o OpenCV com os *bindings* de Python. Em computação, um *binding* de uma linguagem de programação para uma biblioteca ou Sistema Operativo, é uma *Application Programming Interface* (API) que fornece um meio ("cola") para utilizar essa biblioteca ou serviço numa particular linguagem de programação. Contudo, o referido projeto distancia-se do apresentado nesta dissertação pelo número reduzido de lugares que monitoriza (apenas 4), e pela perspetiva de visualização (visão de topo - *birds view*) que, geralmente, não se encontra disponível em espaços abertos (tal como na University of West Bohemia).

Em Yamada et al. (2001), o algoritmo utiliza os vários componentes constituintes do carro (capô, janelas, faróis, pára-choques, grelha frontal, matrícula) para inferir sobre o estado de cada lugar, atribuindo uma pontuação a cada componente/segmento. Se

o número de segmentos encontrado for superior a um dado limite, então o algoritmo conclui que o lugar está ocupado por um carro. Na figura 2 no artigo de Yamada et al. (2001) é possível observar a diferença entre o número de segmentos na presença ou na ausência de um carro. Na figura 3 do mesmo artigo, observam-se os resultados da pontuação atribuída pelo algoritmo, e constata-se que o número de segmentos na presença de carros é maior. O parâmetro g, que é calculado com base no número de segmentos e relacionado com as suas áreas, é também maior na presença de carros. O artigo mostra que o algoritmo funciona com a ocorrência de chuva, mas existem algumas deteções erradas quando o parque começa a secar, porque as zonas ainda molhadas têm formas irregulares que aumentam o número de segmentos quando o lugar está vazio. A taxa de deteção para este algoritmo foi de 98.7%, contudo a sua aplicação não é compatível com o parque estudado nesta dissertação, uma vez que não é possível ter uma representação individual dos carros dentro de cada lugar de estacionamento (parte inferior do parque), devido à perspetiva da câmara, sendo que os carros se sobrepõem uns aos outros e o número de segmentos visíveis é variável e depende se o carro adjacente se encontra presente ou não.

O sistema de parque de estacionamento SFpark (2015) é baseado no princípio de que se as tarifas forem bem ajustadas, pelo menos um lugar estará disponível em cada secção em dado momento. Para determinar as tarifas, o sistema necessita de obter a informação sobre a disponibilidade dos lugares. O sistema utiliza sensores sem fios (um ou dois por lugar) para indicar se um espaço está ocupado ou não. Estes dados são disponibilizados em tempo real aos condutores, e são revistos mensalmente a fim de determinar os locais onde os preços de estacionamento devem ser aumentados ou reduzidos. O sistema transmite de seguida a informação dos preços ajustados aos parquímetros. O projeto piloto distingue-se pela implementação em larga escala na cidade de São Francisco, monitorizando cerca de 8000 lugares em diferentes áreas da cidade, conforme se observa na referência SFpark (2014b). No entanto, foram despendidos neste projeto mais de 46 milhões de dólares, sendo que cerca de 5.7 milhões de

dólares foram direcionados para os sensores. A instalação de um sensor não foi sempre suficiente para alcançar uma taxa de precisão de 92%, pelo que, em certos lugares foram instalados dois sensores.

Streeline (Streetline, 2015), é uma empresa de nível mundial que trabalha na área do estacionamento inteligente e que oferece um produto com ferramentas para análise de dados para resolver os problemas do estacionamento. A sua área de intervenção abrange garagens, universidades, empresas e a cidades. Oferece diferentes produtos, baseados na existência ou não de parquímetros, utiliza as novas tecnologias (aplicações e serviços Web), para informar os seus clientes e para a consulta e análise dos dados. O seu produto, à semelhança do projeto SFpark, consiste na colocação de sensores em cada lugar, o que requer encargos futuros em termos de manutenção.

A colocação de câmaras, ou a utilização das já existentes, apresenta-se como uma forma eficiente de reduzir os custos neste tipo de projetos.

Os artigos e projetos acima analisados constituíram uma mais valia a este trabalho, tendo servido de base para analisar o problema e apontar possíveis soluções. O detetor selecionado é baseado em imagem, e sustenta a criação de um produto de baixo custo, que permite evitar encargos financeiros com a implementação e manutenção do projeto .

Em Prabha et al. (2014), foi utilizado o Raspberry Pi como plataforma de desenvolvimento de baixo custo, tal como no projeto que serviu de base à presente dissertação (nesta é utilizada a versão mais recente, modelo 2) e o OpenCV como biblioteca para a criação do algoritmo de visão, contudo em Prabha et al. (2014) não é apresentada a taxa de sucesso do algoritmo e este, foi implementado num parque com vista *bird view* em apenas 4 lugares, o que facilita bastante a distinção entre um parque ocupado ou livre.

Anteriormente, em Yamada et al. (2001), é feita uma análise exaustiva do algoritmo no parque de estacionamento com diferentes condições atmosféricas, contudo este algoritmo não é aplicável ao parque de estacionamento referido nesta dissertação, uma

vez que o efeito da perspetiva leva a que a representação de um carro se sobreponha ao lugar adjacente, não permitindo distinguir os contornos do carro dentro das linhas de separação dos lugares como acontece no artigo Yamada et al. (2001).

Por último, importa referir que alguns deste algoritmos foram totalmente ou parcialmente implementados e testados no contexto do parque em análise. Os testes podem ser encontrados em Sec. 4.2.

2.5 Softwares e Biliotecas existentes no mercado

O SimpleCV (SimpleCV, 2015) é uma *framework* para a construção de aplicações de visão computacional, sendo que é possível utilizar várias funções do OpenCV sem ser necessário ter em atenção certos detalhes, isto é, proporciona uma camada de abstração para a utilização de certas funções e algoritmos. Contudo, mais controlo foi sendo necessário introduzir na implementação dos testes a realizar, o que levou à utilização direta do OpenCV.

O OpenCV (OpenCV, 2014b) é uma biblioteca de funções para Visão Computacional. A descrição que se segue é baseada em Pajankar (2015). O OpenCV é uma biblioteca multi-plataforma, o que significa que pode ser implementada e utilizada com diferentes sistemas operativos: Windows, Android e sistemas operativos do tipo Unix. É focado principalmente em processamento de imagem e vídeo. Adicionalmente, possui diversas ferramentas e eventos para interação com o utilizador. Dado que foi lançado sob uma licença BSD, pode ser utilizado livremente para fins académicos ou comerciais. Possui interfaces de programação para linguagens populares, tais como C/C++, Python e Java. No momento da redação desta dissertação (2015) o OpenCV encontra-sena versão 3.0, contudo, a versão disponível nos repositórios é a 2.4, tendo sido esta a selecionada para a implementação.

2.6 Raspberry Pi

O Raspberry Pi é uma unidade de processamento *single-board* de baixo custo. Em conexão com a HD webcam, o Raspberry Pi 2, será responsável pela aquisição de imagem e da sua análise através do algoritmo de visão criado no âmbito desta dissertação.

Uma plataforma *single-board* não é modular como os tradicionais computadores, sendo composta por uma única "placa"que inclui processador, RAM, I/O e portos para interação com outros dispositivos (Pajankar, 2015). Estas plataformas *single-board* são utilizados como computadores de baixo custo para a área académica e de investigação. Outros dispositivos semelhantes ao Raspberry Pi disponíveis no mercado são, por exemplo: o Banana Pi, o BeagleBone e o CubieBoard, entre outros.

O Raspberry Pi possui o tamanho de um cartão de crédito e a sua criação teve como principal objetivo promover o ensino e as competências informáticas nas escolas. A sua utilização, no entanto, ultrapassou o seu propósito de criação e expandiu-se para os mercados dos sistemas embebidos e para a investigação, tendo sido também adotado em diversas aplicações e produtos.

O Raspberry Pi utiliza o Python como principal linguagem de programação para desenvolvimento de aplicações (Pajankar, 2015). A utilização de Python para o OpenCV no Raspberry Pi é o recomendado para os trabalhos em visão computacional (Pajankar, 2015).

O Raspberry Pi não oferece suporte ao OpenCL, pelo que todo o processamento é realizado pelo CPU. Esta situação revela-se como uma forte restrição aos algoritmos de visão, pois o CPU do Raspberry Pi tem pouco poder de computação para as exigências dos algoritmos de processamento de imagem.

A descrição das especificações dos modelos mais recentes do Raspberry Pi, pode ser consultada na referência (Element14, 2015). As principais características do Raspberry Pi modelo B, modelo 2 B, e um PC, encontram-se discriminadas na Tab. 2.1, de forma resumida.

Tabela 2.1: Características Raspberry Pi (modelo B, modelo 2 B) e de um computador pessoal.

	RPi Modelo B	RPi Modelo 2 B	PC
CPU	BroadCom BCM2835 ARMv6	BroadCom BCM2836 ARMv7	Intel Core i7-4710HQ
Frequência	700 MHz	900 MHz	2.5 GHz
Nº núcleos	1	4	4
RAM	512 MB	1 GB	8 GB
GPU	BroadCom VideoCore IV @ 250 MHz	BroadCom VideoCore IV @ 250 MHz	NVIDIA GeForce GTX 850 MHz

Existem algumas especificações a ter em atenção. Estes dois sistemas - Raspberry Pi vs computador pessoal (PC) - são completamente distintos, e percebe-se pela Tab. 2.1 que o Raspberry Pi, tem um poder de computação significativamente menor quando comparado com o PC, tal como pode ser observado pelos 1,600,000,000 ciclos de relógio por segundo que o Raspberry Pi 2 B tem a menos.

O Raspberry Pi não suporta o OpenCL, pelo que os algoritmos de visão criados irão correr no CPU. Adicionalmente à presente dissertação, foi elaborado um manual com um conjunto de tutoriais de como utilizar o Raspberry Pi e o OpenCV, nomeadamente: a instalação do Sistema Operativo, as várias formas de comunicar com o Raspberry Pi, as livrarias a instalar após o primeiro *boot*, a instalação do OpenCV dos repositório ou da *source*.

2.7 Breve Discussão

Os sistemas inteligentes para parque de estacionamento são necessários para a redução do tráfego, o que consequentemente possibilita poupança de tempo e de dinheiro, a redução da emissão de gases de efeito estufa e o aumento da segurança de todos os utilizadores das vias de circulação.

Atualmente, os referidos sistemas encontram-se em fase de evolução e de teste. Por um lado, procura-se explorar as novas tecnologias (aplicações para telemóvel, páginas Web) para melhor direcionar e informar os condutores e outros utentes do espaço público e privado sobre o estado dos lugares a que se dirigem. Por outro lado, procura-se monitorizar e armazenar a informação sobre a presença de veículos e seu tempo de

permanência, a fim de detetar padrões de tráfego e de comportamentos. O problema do estacionamento é transversal a todas as cidades e estima-se que 30% do tráfego das cidades tem origem na procura de estacionamento (Buecheler, 2014). No sentido de fazer uma gestão inteligente dos espaços disponíveis, uma das respostas implementadas consiste na aplicação de um preçário dinâmico que reflita a procura e a oferta do parque.

Sistemas que utilizam a imagem como meio para a deteção do estado do parque são caracterizados por apresentar um baixo custo de implementação, tendo sido por esse motivo este o sistema selecionado para o trabalho apresentado nesta dissertação.

O OpenCV é uma biblioteca de visão computacional *opensource*, adequada a ser utilizada no Raspberry Pi e com uma comunidade ampla e recetiva a auxiliar nas questões de outros membros, pelo que a sua utilização foi adotada.

Em termos da unidade de processamento, o Raspberry Pi modelo 2, é uma plataforma de baixo custo, com uma comunidade também muito ampla e aberta, uma plataforma de referência no que respeita à qualidade/preço e das características que apresenta.

3

Desenvolvimento

3.1 Sistema

Como referido na Introdução, Sec. 1, o presente trabalho tem como objetivo o desenvolvimento de um sistema para parques de estacionamento de custo reduzido, com recurso a visão computacional. A proposta apresentada nesta dissertação é a integração de uma webcam com um Raspberry Pi 2, que irá executar o algoritmo de visão, especialmente criado para correr numa plataforma de baixo custo, para inferir sobre o estado do parque. O produto pode ser utilizado isoladamente ou em conjunto, para fazer a cobertura de um parque de estacionamento.

As soluções comercias existentes no mercado são viáveis do ponto de vista de pre-

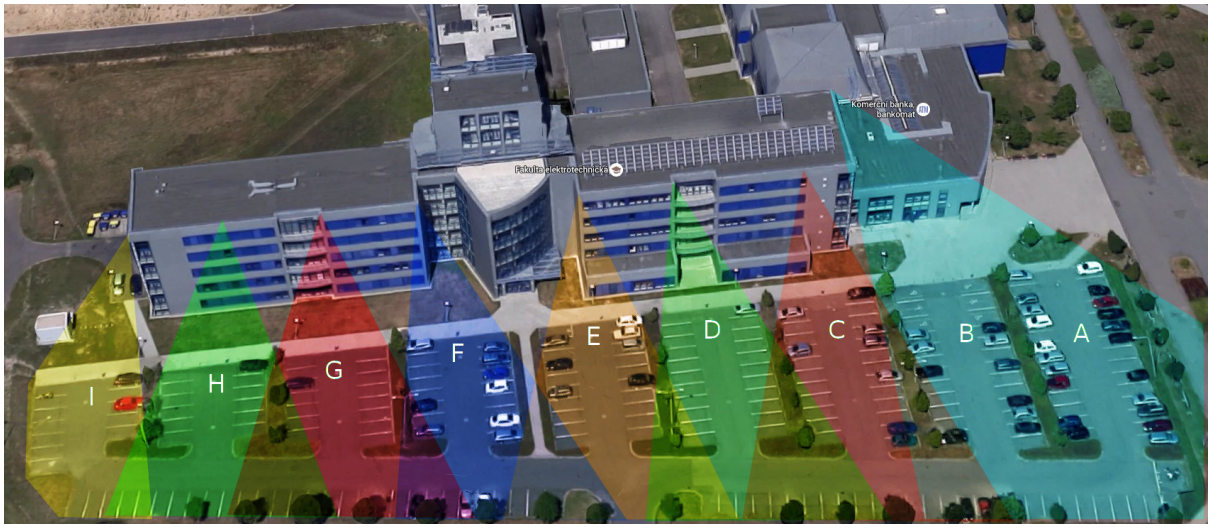


Figura 3.1: Setores do parque.

cisão do sistema, mas economicamente dispendiosas (SFpark, 2014a; Streetline, 2015), uma vez que incluem a colocação de sensores em cada lugar de estacionamento e a sua posterior manutenção, implicando também projetar a comunicação da rede de sensores com os dispositivos de rede que encaminham os dados para um servidor de modo a serem processados.

O sistema proposto tem como princípio o processamento de imagens adquiridas por uma ou mais câmaras para inferir sobre o estado dos lugares do parque de estacionamento. Dada a extensão dos parques de estacionamento, em geral, torna-se evidente que uma só câmara não será suficiente para cobrir toda a área, pelo que o sistema será, geralmente, composto por um conjunto dos módulos propostos. A integração de câmaras IP, também não é sustentável financeiramente, uma vez que estas ainda apresentam um custo elevado por unidade. A colocação de câmaras IP, implica também usualmente a utilização de um PC para fazer o processamento de múltiplas *live feeds*.

Também como já referido o sistema foi testado no parque de estacionamento da University of West Bohemia, o qual é amplo e encontra-se dividido em 9 sectores (Fig. 3.1), sendo que em cada sector será instalada uma câmara (HD Webcam, 720p

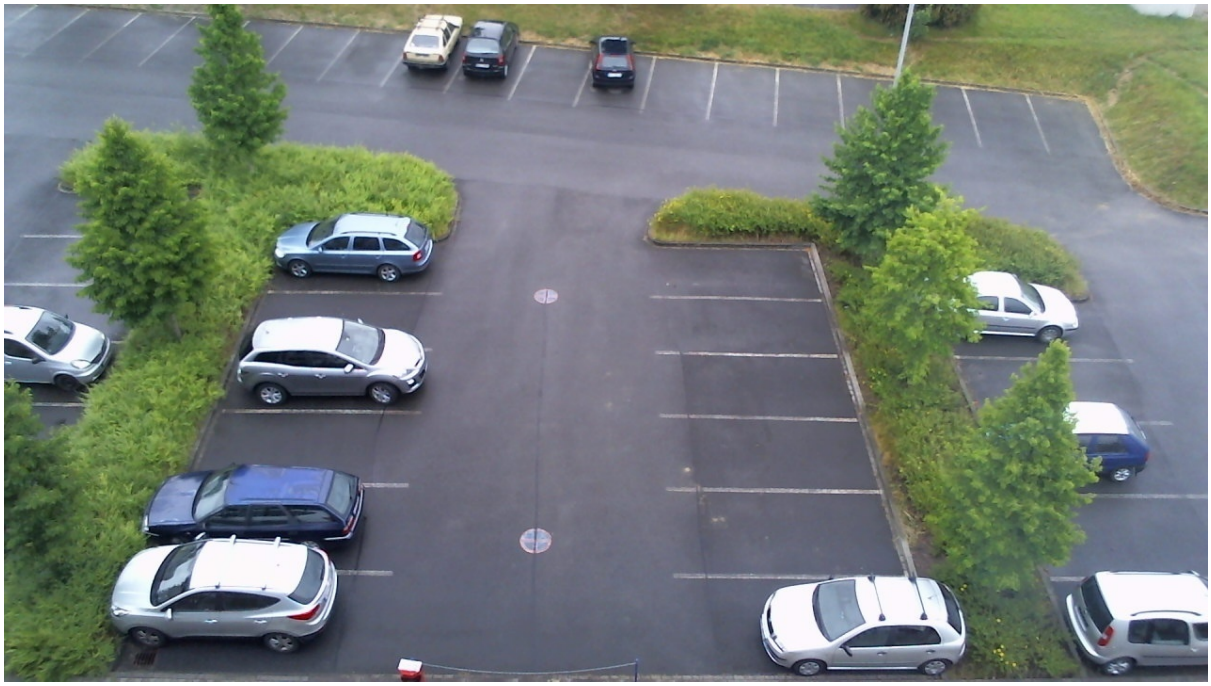


Figura 3.2: Imagem (I) da secção H.

com 30 fps) e um Raspberry Pi 2; cada cor representa o campo de visão de uma câmara, e a interseção é representada com uma cor diferente. O custo total do sistema por sector ronda os 70€. As câmaras serão colocadas no topo do edifício, na cobertura. No âmbito desta dissertação, foi utilizado para explicação e ilustração a secção H, ver Fig. 3.2.

Na Fig. 3.2, observa-se dois tipos de estacionamento distintos: em *baixo*, onde se encontram os lugares de estacionamento posicionados de forma transversal à câmara e com possibilidade de oclusões parciais entre os carros, e em *cima*, onde se encontram os lugares longitudinalmente em relação à câmara, mas com uma "rotação".

Tal como mencionado na Sec. 1.1, o sistema é dividido em dois módulos: (i) inicialização do sistema e (ii) deteção do estado dos lugares. Os módulos e os seus submódulos encontram-se representados no diagrama apresentado na Fig. 3.3

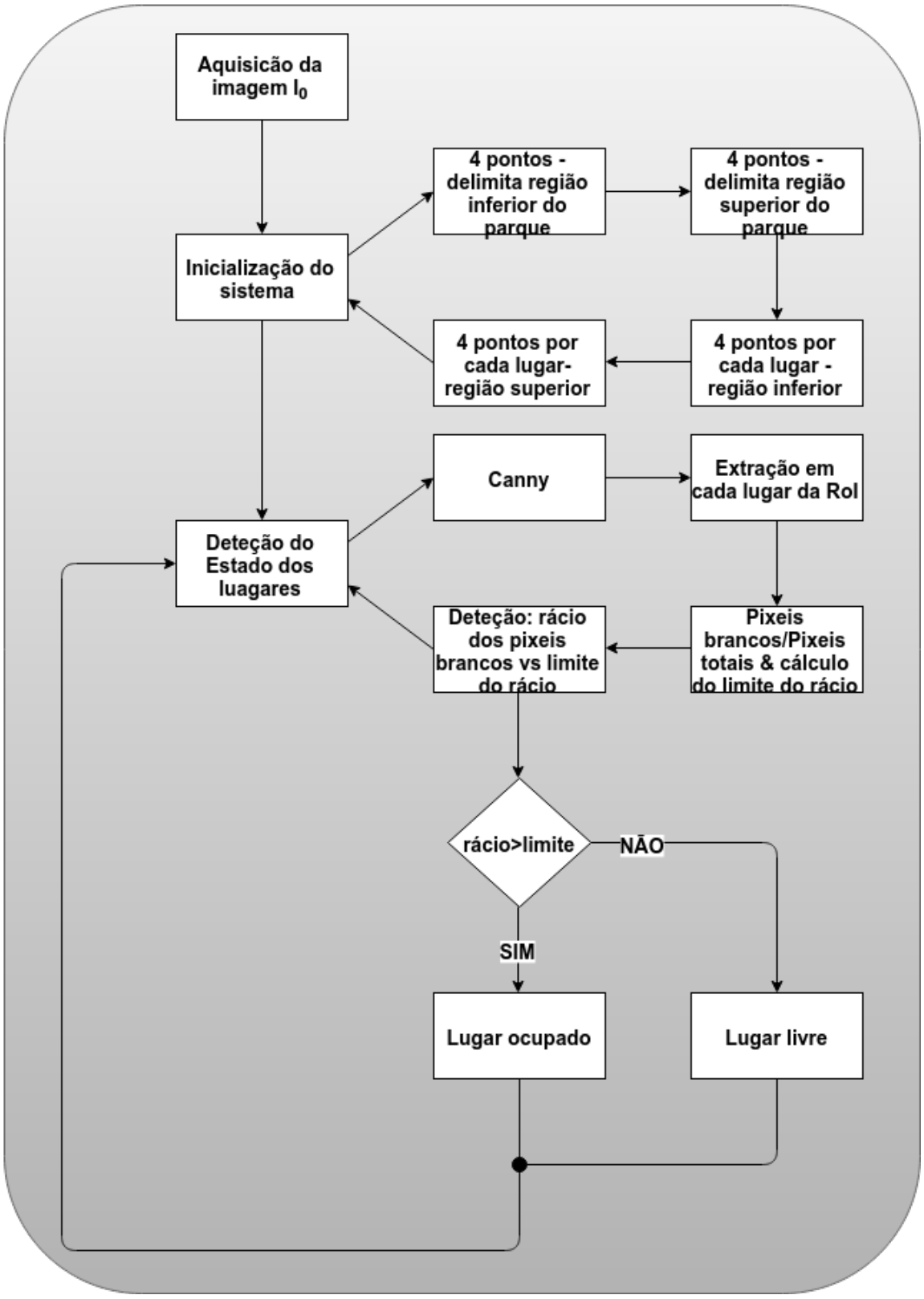


Figura 3.3: Diagrama do sistema.

3.1.1 Inicialização do Sistema

A inicialização do sistema (i), é feito apenas uma vez na configuração inicial do sistema. Considere-se $I_t(x, y)$ como a imagem adquirida a cada Δt = 1s (este valor pode ser alterado na base de dados (BD) da aplicação). Com a imagem inicial do parque adquirida I_0, o primeiro passo é selecionar manualmente um conjunto de pontos que limitam os lugares de estacionamento, utilizando a aplicação fornecida ao administrador do parque:

(a) Quatro pontos ($P_{\{b1,\ldots,b4\}}$) a laranja que definem os cantos da região inferior do parque, ver Fig. 3.4. Posteriormente, esses pontos serão utilizados para a obtenção de um rácio entre a distância à câmara e o tamanho do carro.

(b) Quatro pontos ($P_{\{u1,\ldots,u4\}}$) a vermelhos definem a região superior do parque, ver na Fig. 3.4. Posteriormente, estes pontos serão utilizados para realizar uma transformação de perspetiva (OpenCV, 2015) da região superior do parque, como se observa na Fig. 3.5.

Figura 3.4: Seleção das regiões de referência para o estacionamento: em *cima*, pontos vermelhos e de *baixo*, pontos laranja.

(c) Um conjunto (N_{pb}) de 4 pontos ($p_{\{1,...,4\}}$), delimitam os lugares de estacionamento na região inferior do parque, $PP_{\{i,p\}}$, com $i = \{1, ..., N_{pb}\}$. As coordenadas de cada lugar são selecionadas a partir da linha de baixo que define o lugar, até onde o topo do carro/jipe/carrinha/etc. é esperado, v.d. Fig. 3.6b) (e não a segunda linha que define o lugar, Fig. 3.6a)). Desta forma, o "centro"do lugar irá subir, não sendo suposto estar coberto pelo carro abaixo, quando acontecem as oclusões parciais (em geral, o ângulo da câmara para o parque tem que ser tal que permita esta premissa). Na Fig. 3.7, é possível observar a marcação destes pontos a cores. Como se pode notar, não é necessária grande precisão na marcação dos pontos, apenas uma projeção onde o carro/jipe irá estacionar.

(d) O mesmo do que em c), mas agora para a parte cima do parque de estacionamento, após a transformação de perspetiva definida em b), ver os pontos coloridos na

Figura 3.5: Detalhe da região de *cima* do parque de estacionamento após transformação de perspetiva, a verde encontram-se as RoI de cada lugar (R_i) e a cores os pontos que definem os lugares.

(a) Marcações seguindo linhas (não usadas). (b) Marcações dado a altura do carro (usadas).

Figura 3.6: Análise da marcação do lugar, para a secção de *baixo* do parque.

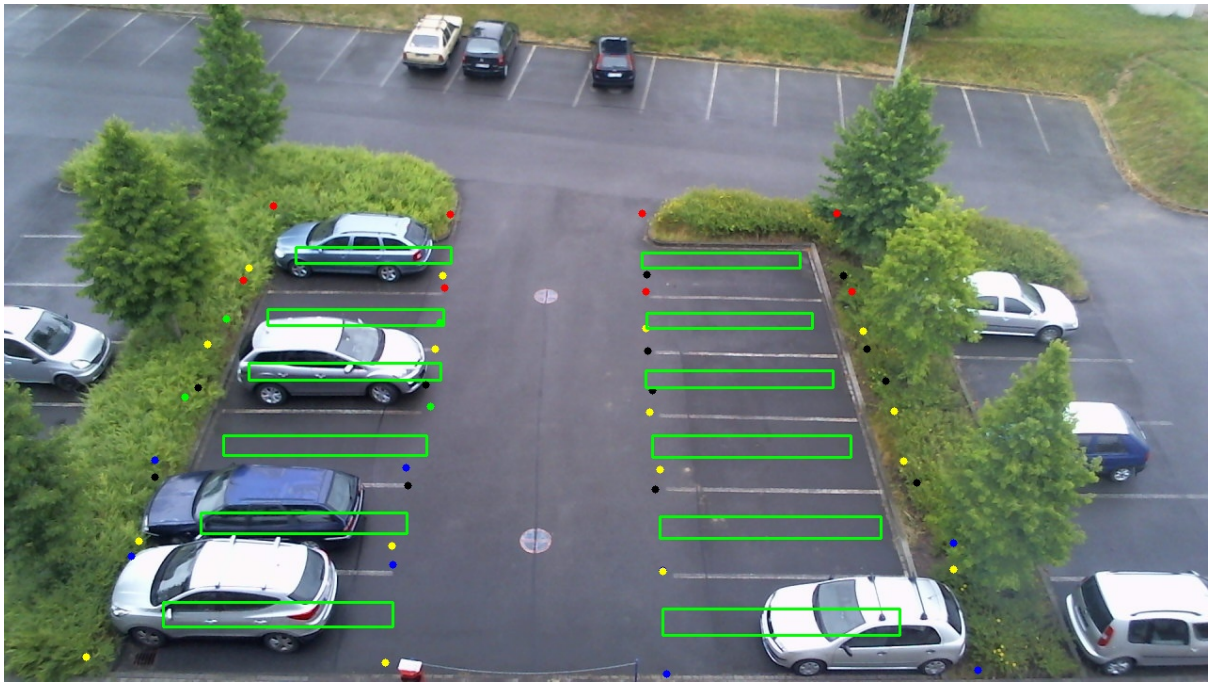


Figura 3.7: Região de *baixo* do parque de estacionamento. A verde encontram-se as RoI de cada lugar (R_i) e a cores os pontos que definem os lugares.

Fig. 3.5. Os N_{pu} lugares em *cima*, $PP_{\{i,p\}}$, são igualmente definidos por 4 pontos, agora com $i = \{N_{pb}+1, ..., N_{pb}+N_{pu}\}$.

O número total de lugares em cada secção é dado por $N_{ps_s} = N_{pb_s} + N_{pu_s}$, com s sendo a letra/número da secção. A notação s é apenas utilizado neste parágrafo e removido dos restantes parágrafos para simplificar a notação (os restantes parágrafos referem-se sempre à secção individual H).

Para a secção H, $N_{pb_H} = 12$, $N_{pu_H} = 8$ e $N_{ps_H} = 20$. O número total de lugares é dado pela soma destes em todas as secções $N_{ps} = \sum_{s=\{A,...,I\}} N_{ps_s}$.

3.1.2 Deteção do Estado dos Lugares

A Deteção do Estado dos Lugares (ii), é composta por 4 passos:

(a) o primeiro, consiste na aplicação do detetor de arestas Canny a cada imagem I_t, devolvendo I_c (os limites utilizados seguem a recomendação de Bradski and Kaehler (2008)). O objetivo é apenas a devolução das arestas dos carros (ver Fig. 3.8),

Figura 3.8: Mapa de arestas (I_c).

no entanto se aparecerem arestas de poças de água, ruído ou sombras das árvores ou outros carros, estas não têm um impacto significativo no sistema em termos globais.

(b) Para cada imagem de um lugar, PP_i, corresponde uma região de interesse (RoI) R_i extraída a partir de I_c, com $i = \{1, .., N_{ps}\}$. Cada R_i permite a determinação da presença de um carro. Existem dois métodos para o cálculo destes RoIs, dependendo se é a parte superior ou inferior do parque. Considerando os pontos $PP_{\{i,p\}}$: (b.1) nos RoIs situados na zona inferior do parque, verifica-se que os carros mais perto da câmara apresentam um tamanho maior que os carros mais distantes, pelo que, para a computação do RoI (retângulos verdes na Fig. 3.7) a altura é dada por 20% da altura de cada lugar, e um comprimento de 3/4 do lugar para a remoção de vegetação, etc. Um exemplo de R_i de todos os lugares na parte inferior do parque pode ser observado na Fig. 3.9, com as primeiras 6 linhas a corresponderem ao lado esquerdo do estacionamento (A1,...,A6), as restantes ao lado direito do estacionamento (B1,...,B6).

Figura 3.9: Exemplos de R_i, na parte de *baixo* do parque (à direita) e respectivo resultado do Canny (à esquerda). De cima para baixo, as primeiras 6 linhas correspondem ao lado esquerdo do estacionamento, A1 a A6, as restantes ao lado direito do estacionamento, B1 a B6.

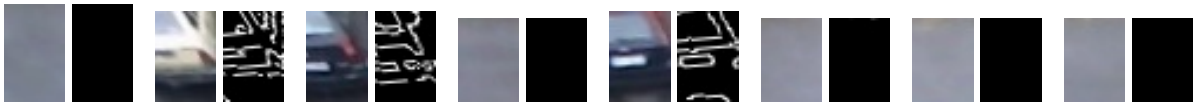


Figura 3.10: Exemplos de R_i, na parte de *cima* do parque e resultado do Canny. Da esquerda apra a direita respectivamente C1 a C8.

(b.2) Nos RoIs situados na parte superior do parque, após a transformação de perspetiva, apresentam tamanhos aproximadamente iguais (retângulos na Fig. 3.5). Os RoIs foram obtidos utilizando 3/4 da altura e 40% da largura, ver exemplos na Fig. 3.10.

(c) O próximo passo consiste no cálculo do rácio (r_i) dos pixeis brancos (ruído ou arestas dos carros) pelo total número de pixeis em cada RoI, i.e., $r_i = \sum_{R_i=255} R_i(x,y) / \sum R_i(x,y)$. Na presença de um carro, a contagem de pixeis brancos é esperada ser elevada. Dois limites foram calculados (empiricamente) - os limites de deteção para as regiões superior e inferior do parque: (c.1) para a parte superior do parque, um limite do rácio $r_{tu} = 0.2$ foi utilizado.

(c.2) Para a parte inferior do parque, um limite estático não seria adequado, devido ao impacto da perspetiva da câmara, i.e., o efeito do tamanho do carro e a "largura"das arestas. Neste caso, uma equação linear foi calculada para expressar o rácio entre a distância e o tamanho do carro, utilizando os pontos $P_{\{b1,\ldots,b4\}}$ (ver Fig. 3.4), i.e., $b = (((P_{b3x} - P_{b2x}) - (P_{b4x} - P_{b1x})) \times (a + (P_{b4x} - P_{b1x})))/(P_{b3y} - P_{b4y})$. Nesta equação, o b representa a largura do parque, para uma dada altura a na imagem I_t. Assumindo como referência (unidade: 1) o tamanho do carro, onde a largura é dado por $P_{b1x} - P_{b4x}$, o rácio que compara o tamanho do carro com a altura, em respeito da largura é: $1.0/r_c = (P_{b4x} - P_{b1x})/b \Leftrightarrow r_c = b/(P_{b4x} - P_{b1x})$. Relacionando os limites das regiões superior e inferior é calculado o limite $r_{tb} = r_{tu}/(4 \times r_c)$.

(d) Finalmente, o limite superior (r_{tu_i}) e o inferior (r_{tb_i}), que depende do R_i, são comparados com r_i, e se forem menores, é a indicação de que um carro não está presente, ver Fig. 3.11, pontos verdes. Se um mesmo lugar se apresentar $5\times$ sequencialmente como lugar "ocupado"(correspondendo a 5 segundos), o lugar é considerado como ocupado e marcado com os pontos a vermelho na imagem. Da mesma forma, se um determinado lugar se apresentar $5\times$ como lugar "livre", o lugar é assinalado a verde, e dado como livre. Se dentro dos últimos 5 segundos existir alguma variação do estado, o lugar passa para o estado de "transição", e será assinalado a amarelo (Fig. 3.11 em baixo à esquerda).

3.2 Ferramentas

O sistema encontra-se preparado para funcionar com diversos tipos de fontes, quer seja, uma *live feed*, ou um conjunto de imagens para teste. A forma organizada das imagens na base de dados (BD), permitiu o teste automático do sistema para um conjunto de imagens distintas nomeadamente em termos do estado do parque (lugares

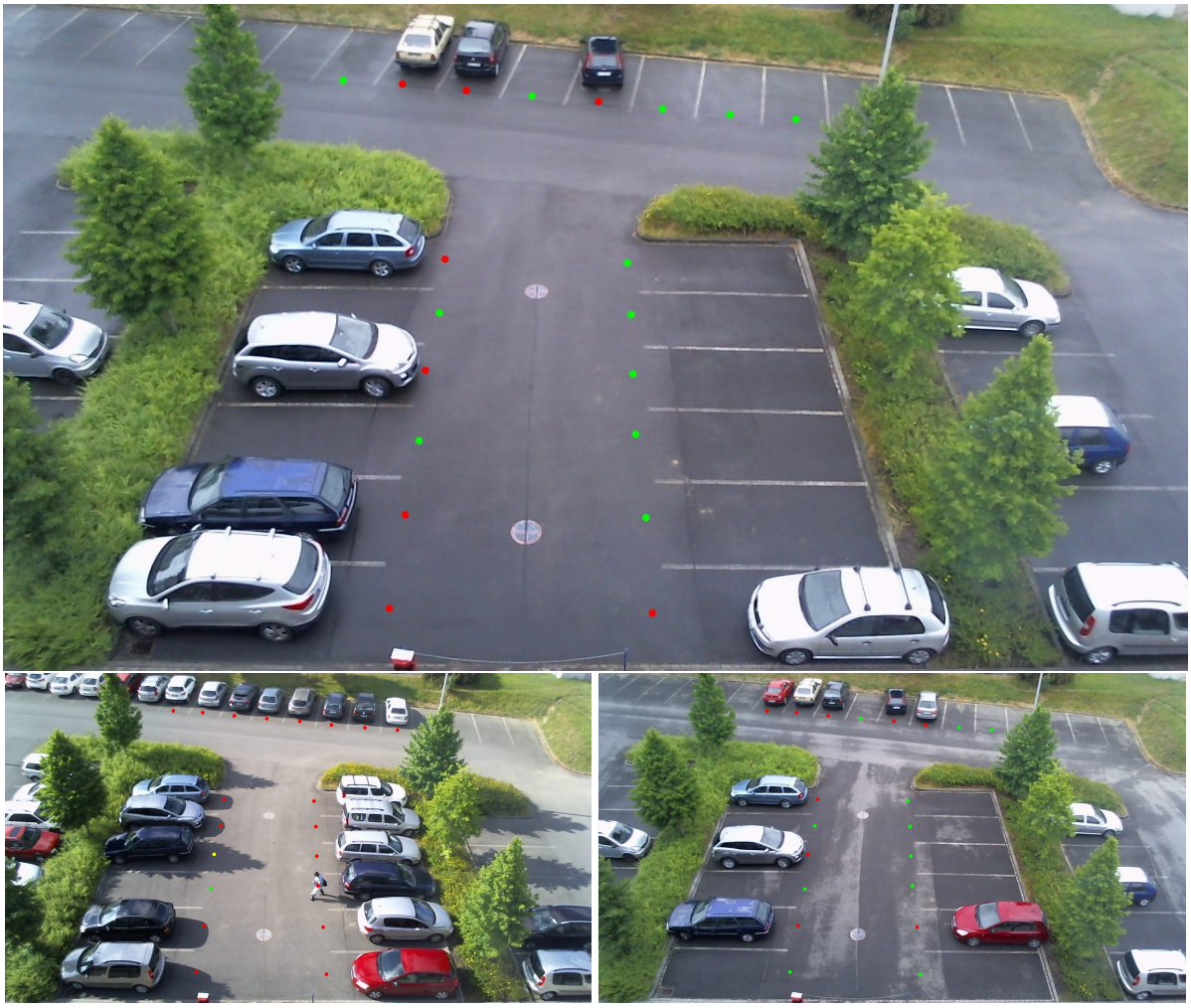


Figura 3.11: Resultados. Pontos vermelhos são espaços *ocupados*, pontos verdes são *livres* e a amarelo *ocupados* mas nas imagens anteriores devolveram *livre*.

vazios e ocupados) e em termos das diferentes condições meteorológicas que se fizeram sentir durante o período de aquisição de imagens (chuva, períodos de sol e nuvens).

3.2.1 Seleção dos lugares

Para o administrador do parque, uma das ferramentas mais úteis da aplicação no procedimento de inicialização do parque, é a seleção dos lugares. A extração da informação consiste no armazenamento das coordenadas x e y de cada ponto, sendo cada um guardado com uma *label* referente ao lugar a que pertence (ver Fig. 3.12. A *label*,

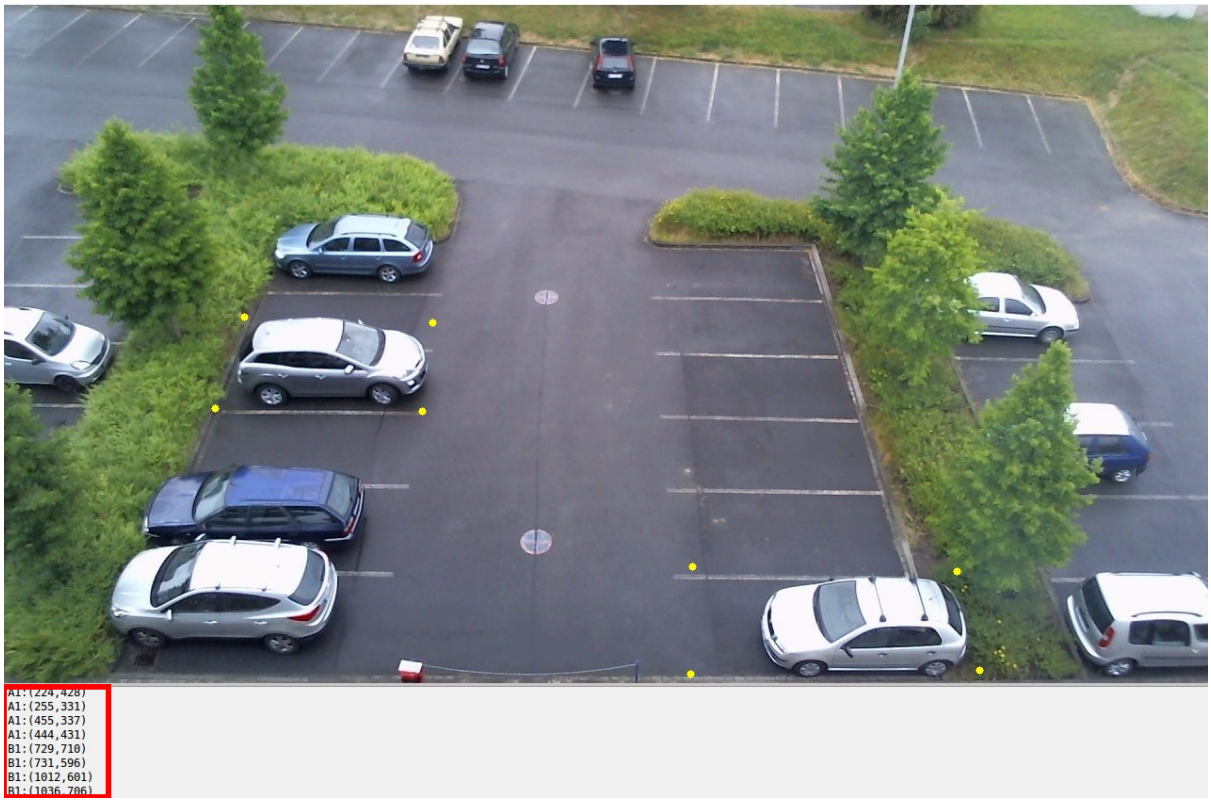


Figura 3.12: Seleção dos lugares

contém informação sobre a fila (A-Z) e o número do lugar na fila. No caso de lugares adjacentes, e sendo que dois pontos são comuns, basta premir a tecla CTRL e clicar duplamente nos outros dois pontos. Para mudar a fila, ou seja, no que se refere à *label* - A, B, C, e por ai adiante, basta premir a tecla SHIFT seguida de duplo clique num ponto do outro lugar da fila. Observe-se na figura Fig. 3.12, a seleção de dois lugares.

A aplicação extrai a informação do ficheiro com as coordenadas de cada lugar, para uma estrutura de dados adequada e pronta a utilizar (dicionário em Python), que será utilizada no módulo de *Deteção do Estado dos Lugares*.

3.2.2 Módulo de armazenamento e *backup* - opcional

Outra ferramente que pode ser util ao gestor do parque de estacionamento é a base de dados (DB). Esta pode ser utilizada e consultada como historial do que se passou no estacionamento, e tem como base o armazenamento de imagens do parque. Con-

siderados os recursos de memória do cartão SD (8 GB no cartão utilizado) existente no Raspberry Pi, este módulo pressupõe a existência de um servidor ou de um PC para o qual possa comunicar utilizando o protocolo CIFS e enviar as imagens. Desta forma, torna-se também possível que outras aplicações tenham acesso às imagens, sem lhes dar acesso direto ao sistema do parque. Informações de como montar o diretório remoto, podem ser encontradas na referência Ubuntu (2015).

No presente trabalho, verificou-se que após a implementação da pasta partilhada, que esta por vezes não era montada com sucesso quando o produto era ligado. A solução passou por colocar a "montagem"da pasta e sub-pastas para o *login*, em vez de durante o *boot* (opção por defeito).

A aplicação oferece ao utilizador um *ficheiro de configuração* onde pode selecionar o número de imagens adquiridas por segundo e a resolução de imagem que pretende. No caso do utilizador pretender saber o número de imagens por segundo que o sistema suporta, tem que alterar o parâmetro de configuração (*test_max_fps*).

Foi também desenvolvido um módulo para fazer o armazenamento de imagens de forma dinâmica, isto é, o sistema mantém no diretório duas horas de imagens anteriores à hora atual no servidor (ou mais, dependendo da configuração). Estas fotos podem ser utilizadas a fim de analisar algum evento, por segurança, ou para teste do sistema. As fotos são armazenadas com a hora, minutos, segundos e milésimos de segundo no seu nome de ficheiro. O trabalho de remoção das fotos é feito a cada mudança de hora, com recurso aos *cron job*, que são programas que correm em segundo plano. A estratégia passou por organizar as fotos em diretórios, atendendo à sua hora, deste modo, a solução passa por a cada mudança de hora, apagar a pasta que tenha mais de duas horas. A Fig. 3.13 apresenta o diagrama de funcionamento.

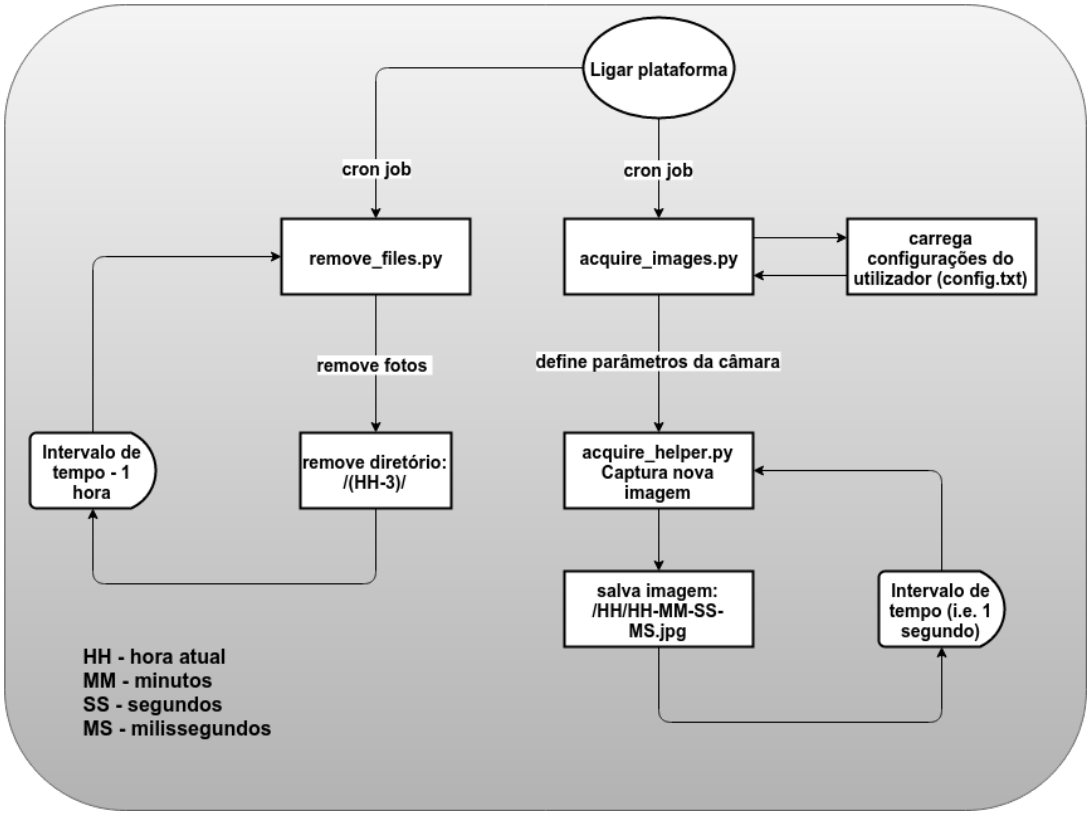


Figura 3.13: Diagrama de funcionamento.

4

Resultados Obtidos

4.1 Enquadramento

Como referido, o sistema proposto, Raspberry Pi modelo 2, HD webcam e módulo de Visão Computacional, revela-se como um produto de baixo custo (cerca de 70 euros) para aplicações de parque de estacionamento, com uma taxa de sucesso de 99.6% para o caso estudado.

Um conjunto de imagens foram capturadas desde o nascer do sol (por volta das 6h00) até ao pôr do sol (por volta das 20h00), durante 8 dias, de 15 a 22 de Junho de 2015, no parque de estacionamento da University of West Bohemia, com diferentes condições atmosféricas (nuvens, períodos de sol, e chuva). O intervalo definido para

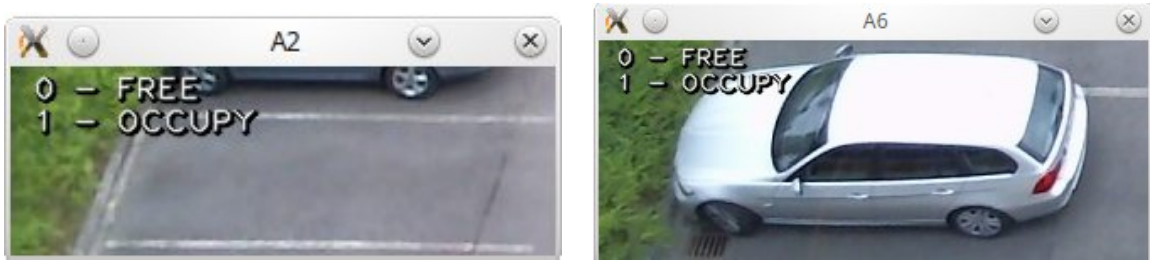
cada imagem foi de 1 segundo, mas este intervalo pode ser ajustado na aplicação. Na maioria do tempo (minutos) não existem mudanças no parque (carros a entrar ou a sair). Por essa razão, um conjunto de 82 sequências de imagens foi selecionado. Para teste, foi selecionada uma imagem de cada conjunto, 82 imagens no total, o que corresponde a 1640 lugares de estacionamento (82×20; secção H). Na validadação da precisão do algoritmo, os resultados serão comparados com o *ground truth* (ver Secção 4.1.1) do parque para aquele instante.

A estrutura dos testes consiste num ficheiro (run_tests.py) que é responsável por receber o teste a executar (teste1, teste2, etc.), e qual o argumento de entrada do teste (foto, conjunto de fotos, *live feed*, conjunto de pastas para analisar recursivamente) e posteriormente chama o respetivo teste.

4.1.1 *Ground Truth*

Procurou-se a implementação de uma estrutura que permitisse efetuar testes sobre todas as amostras, automaticamente. Dado que na maioria do tempo, as imagens são praticamente estáticas (entenda-se como não havendo movimento de carros) foram selecionados os conjuntos de fotos que correspondem a movimentos de veículos, e agrupados em duas pastas distintas - *entering* e *leaving*. De seguida, organizaram-se os movimentos pelo lugar onde estes se efetuaram, tendo sido criadas pastas com as designações a1/, a2/, etc. Dentro de cada uma dessas pastas por sua vez, foi gravada a realização de cada movimento em pastas diferentes - 1/, 2/, etc. Desta forma, é possível definir testes específicos para correrem quando se trata da saída ou da entrada de um carro. É ainda possível testar quais são os lugares que introduzem mais erros e isolar esses lugares nos testes.

A supervisão e análise dos resultados utilizou os respectivos dados como modelo de *ground truth*, no qual se compara os resultados obtidos com a solução. Com este propósito, foi criado um *script*, que recebe um ficheiro com as coordenadas de cada lugar e efetua a sua segmentação, apresentando de seguida esse lugar ao administrador



(a) lugar livre (b) lugar ocupado

Figura 4.1: Estado dos lugares

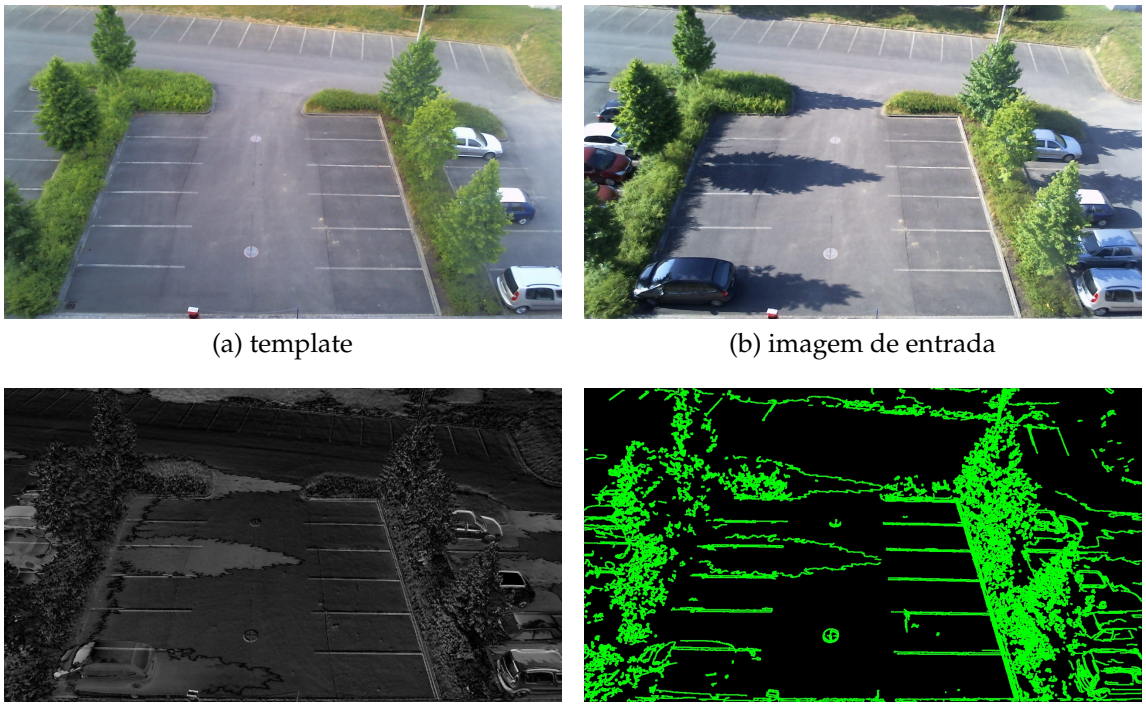
do parque, o qual é solicitado a introduzir o estado do lugar (premir tecla 1 - ocupado, premir tecla 0 - livre) - ver Fig. 4.1. O *script*, percorre recursivamente todas as pastas.

4.2 Discussão comparativa de outros algoritmos

O algoritmo proposto em Prabha et al. (2014) é composto pelos seguintes sub-modulos: (a) foto do parque vazio (converter para a escala de cinzentos); (b) foto do parque (em cada instante, converter para a escala de cinzentos); (c) diferença absoluta entre as duas imagens (a diferença são os carros); (d) deteção de edges utilizando o detector de arestas Canny (OpenCV, 2015); (e) aplicação dos contornos das imagens; (f) se o contorno é maior que determinado limite/*threshold*, o contador de lugares é incrementado de um; (g) aplicação do conceito de centróide.

A comparação (na realidade, a diferença) entre a imagem atual e uma imagem do parque vazio (*template*), não é adequada a um ambiente aberto, uma vez que o meio está sujeito a condições meteorológicas e às variações de luminosidade, que se traduzem em diferenças na imagem resultante (ver Fig. 4.2a) e b)).

A imagem de entrada, Fig. 4.2a), mostra um instante com a presença de fortes sombras. Pela observação do resultado, a localização do carro não é evidente e não possui "atributos"suficientes (arestas) para ser identificado como sendo um carro. De notar que, fazendo uma pesquisa exaustiva de combinações dos limites na operação de cálculo de arestas (Canny), o resultado poderá ser melhorado, contudo, essa combinação



(a) template (b) imagem de entrada

(c) diferença absoluta (d) resultado

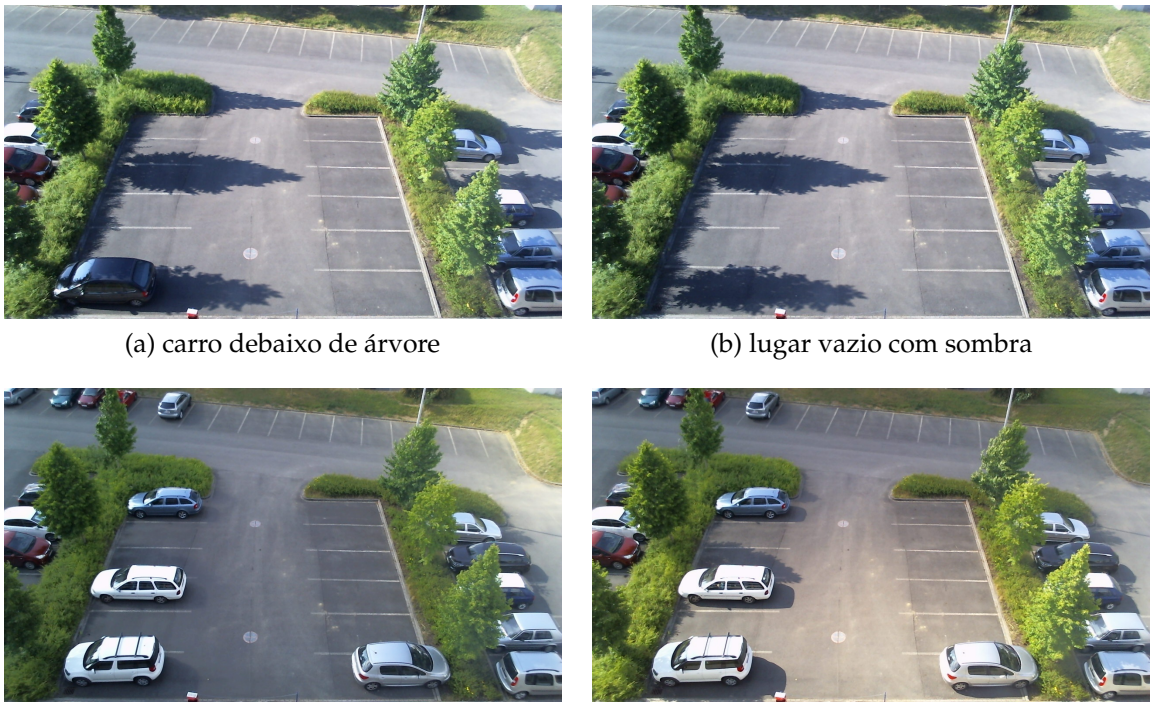
Figura 4.2: Resultado do algoritmo especificado em Prabha et al. (2014).

também poderá não funcionar na presença de condições meteorológicas diferentes.

Em Prabha et al. (2014), são utilizados somente 4 lugares para o teste e em ambiente fechado, o qual apresenta menos complexidade que um meio aberto.

O meio aberto, o estudado nesta proposta de dissertação, por seu lado, é caracterizado por grandes diferenças de luminosidade durante o decorrer do dia. Essas diferenças são provenientes quer do movimento solar ao longo do dia, quer pelas condições atmosféricas, nomeadamente, nuvens. Essas diferenças são responsáveis pelo aparecimento momentâneo ou prolongado de sombras, e que dificultam a compreensão da forma de determinado objeto, constituindo uma fonte promissora de erros. Por outro lado, as diferentes condições climatéricas dificultam a criação de um modelo robusto e que apresente uma boa resposta, independentemente de quais e como as condições atmosféricas se fazem sentir.

Embora em muitos casos, a diferença do fundo (*background*), em comparação com a cor e forma dos carros seja evidente, existem outros em que isto não é verdade, pelo



(a) carro debaixo de árvore

(b) lugar vazio com sombra

(c) nuvem sobre o parque

(d) efeito do movimento da nuvem

Figura 4.3: Em cima, efeito das sombras. Em baixo, aparecimento de uma nuvem sobre o parque.

que a perceção de contornos e de cores é limitada e deve ser tomada em consideração. Veja-se o exemplo das Fig. 4.3, 1ª linha.

Tal como mencionado em cima, o ambiente real aberto está sujeito a mudanças abruptas de luminosidade. Na Fig. 4.3 2ª linha, observa-se o efeito do aparecimento de uma nuvem, num intervalo de 10 segundos entre as fotos.

Na diferença absoluta das imagens, o fundo aparece na imagem resultante, uma vez que tem uma cor diferente do *template* utilizado, ver Fig. 4.2, imagem da 2ª linha à esquerda. Na Fig. 4.4, apresentam-se diferentes cores que o fundo pode assumir dependendo das condições atmosféricas.

O algoritmo em Prabha et al. (2014) necessita ser acompanhado por técnicas que permitam ultrapassar o efeito das sombras, da variação de luminosidade e de pequenos movimentos dos ramos das árvores e vegetação, como por exemplo, as técnicas das seguintes referências Horprasert et al. (1999) e Elgammal et al. (2000).

Figura 4.4: Diferenças no fundo.

O *background subtraction* é uma conhecida técnica das áreas do processamento de imagem e visão computacional, para gerar uma máscara do *foreground* (mais precisamente, uma imagem binária que contém os pixeis que pertencem aos objetos em movimento na câmara) utilizando câmaras estáticas (OpenCV, 2014a). Tal como o nome sugere, o *background subtraction* calcula a máscara de *foreground*, realizando a subtração entre a imagem atual com o modelo do *background*, o qual contém a parte estática da imagem ou, generalizando, tudo o que pode ser considerado como fundo dadas as características observadas na imagem. Existem modelos mais simples Horprasert et al. (1999), Karaman et al. (2005) ou modelos mais avançados capazes de lidar com fundos dinâmicos, tais como o MOG (Mixture of Gaussians) KaewTraKulPong and Bowden (2002). A técnica de *background subtraction* consiste em dois passos principais: inicialização do fundo e atualização do fundo. No primeiro passo, um modelo inicial do

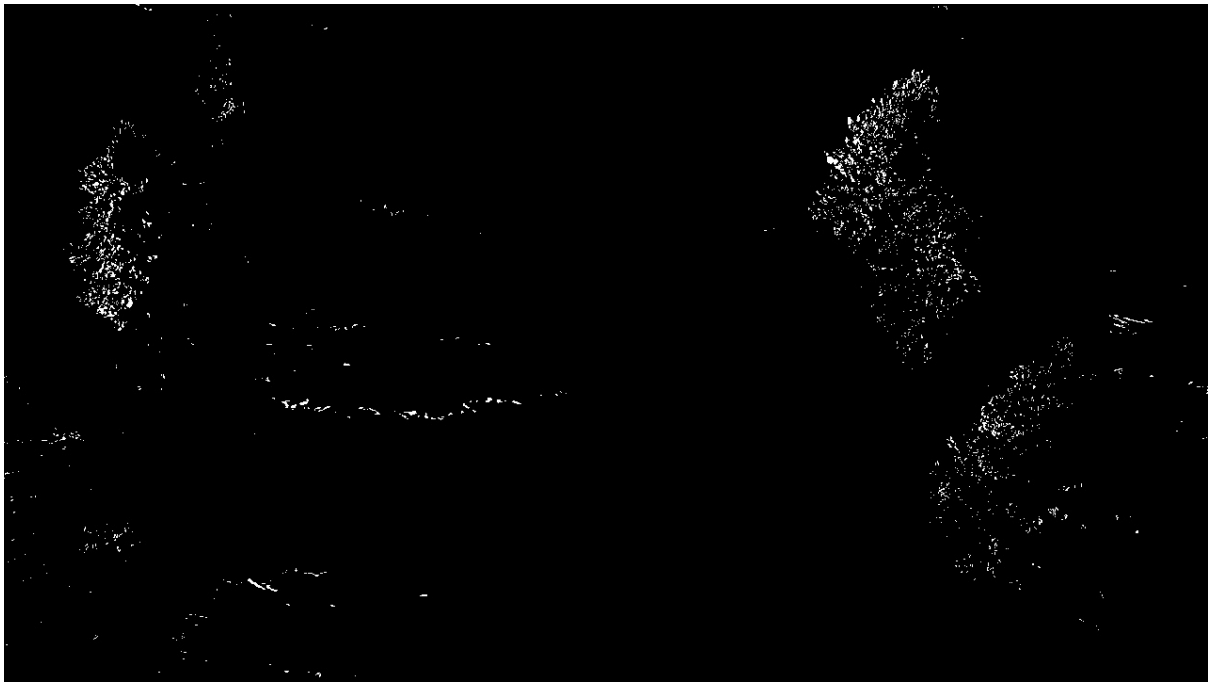


Figura 4.5: *Background subtraction*, efeito do vento.

fundo é calculado, enquanto que no segundo passo o modelo é atualizado no sentido de se adaptar a possíveis mudanças na imagem (OpenCV, 2014a).

No entanto, estes algoritmos têm de lidar com problemas clássicos (mudanças de luminosidade súbitas ou graduais, movimento dos objetos do *background*, movimentos repetidos, etc) (Fernandez-Sanchez et al., 2013). No parque estudado, o efeito do vento é notório junto às árvores e vegetação à volta dos lugares, como se pode confirmar na Fig. 4.5, utilizando o *background subtraction*. É possível observar pixeis brancos na região onde estão os lugares, sem ter existido movimento de carros, o que resulta apenas do movimento das sombras das árvores.

Por esta razão, vários autores têm proposto a fusão de diferentes recursos/características, incluindo a intensidade, arestas e informações da textura. Nesse sentido, foi desenvolvido um algoritmo, diagrama apresentado na Fig. 4.6, que adiciona a análise do contorno dos objetos do *foreground* e operações morfológicas, no sentido de eliminar o ruído e movimento de pequenos objetos, e calcular o centróide do contorno com o intuito de se perceber se o carro se encontra a entrar ou a sair do lugar de estaciona-

mento.

Na Fig. 4.7, é possível observar a deteção do movimento do carro. O algoritmo baseia-se em *background subtraction* e analisa o que mudou na imagem atual em relação às anteriores. Este algoritmo é bastante útil para este tipo de aplicação, uma vez que a câmara se encontra estática, e o que se pretende detetar é a mudança do estado dos lugares, que por sua vez, implica o movimento de entrada ou saída dos carros. O *background subtraction*, permite a cada iteração comparar a imagem atual com as anteriores, e o que mudou na imagem é passado para o *foreground*. O algoritmo, possui parâmetros que definem o ritmo de aprendizagem, isto é, em quantas iterações é que um determinado objeto que foi detetado, irá passar do *foreground* para o *background* (parte do fundo).

Testes utilizando o algoritmo de *background subtraction* MOG, mais concretamente no que se refere ao tempo de execução da operação de atualização do *foreground*, mostraram que o PC levou cerca de 0.03 segundos a executar essa operação, enquanto o Raspberry Pi 2 necessitou de 0.4 segundos (cerca de 13 vezes mais lento).

Outra dificuldade sentida, é relativa a mudanças de luminosidade ou de focus, onde o algoritmo é bastante sensível, pois nestes casos, grande parte do fundo (*background*) é considerado diferente do fundo das imagens anteriores e todas essas regiões são detetadas pelo algoritmo, apesar de não ter existido movimento. Neste teste, em particular, o algoritmo é ainda mais sensível, pois a modificação que se faz no *foreground* é no sentido de aumentar o que foi detetado por meio da operação morfológica dilatação. A dilatação foi realizada com o objetivo de unir os vários buracos (*holes*, o que é detetado como movimento/mudança), em especial nos casos onde o movimento do carro se faz de forma lenta. Nessas situações, o movimento do carro é representado por pequenos blocos separados entre si. Por exemplo, se se considerar um retângulo de cor uniforme, e se este deslizar um pouco para o lado, no meio do retângulo, é como se não tivesse acontecido movimento, pois não existe diferença com a imagem anterior. O mesmo acontece com o carro, considerando o topo, o capô e a mala, onde

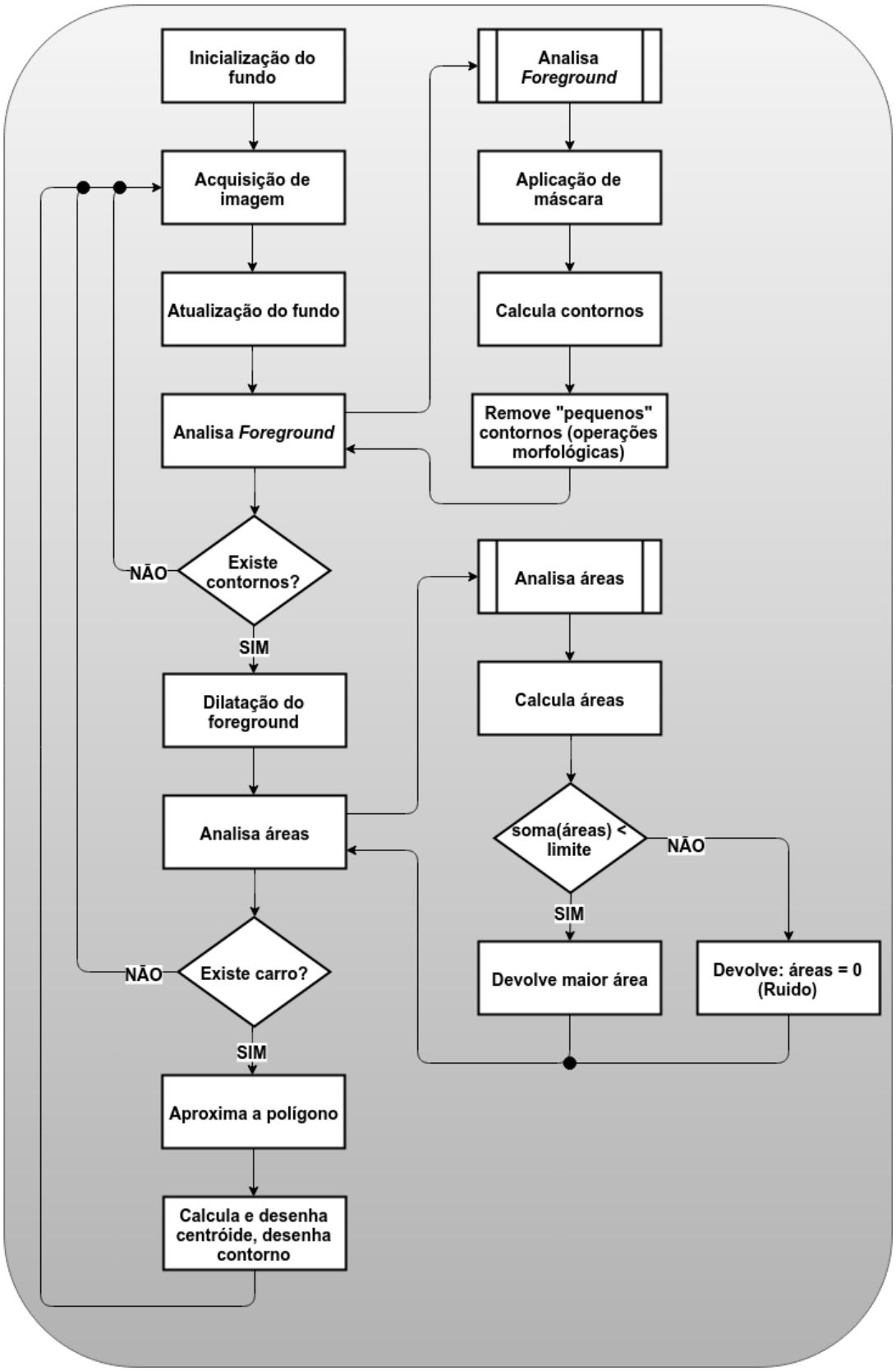


Figura 4.6: Algoritmo com recurso a *background subtraction*.

Figura 4.7: Deteção de carro em movimento.

eles são praticamente uniformes. Uma representação deste fenómeno é ilustrado na Fig. 4.8.

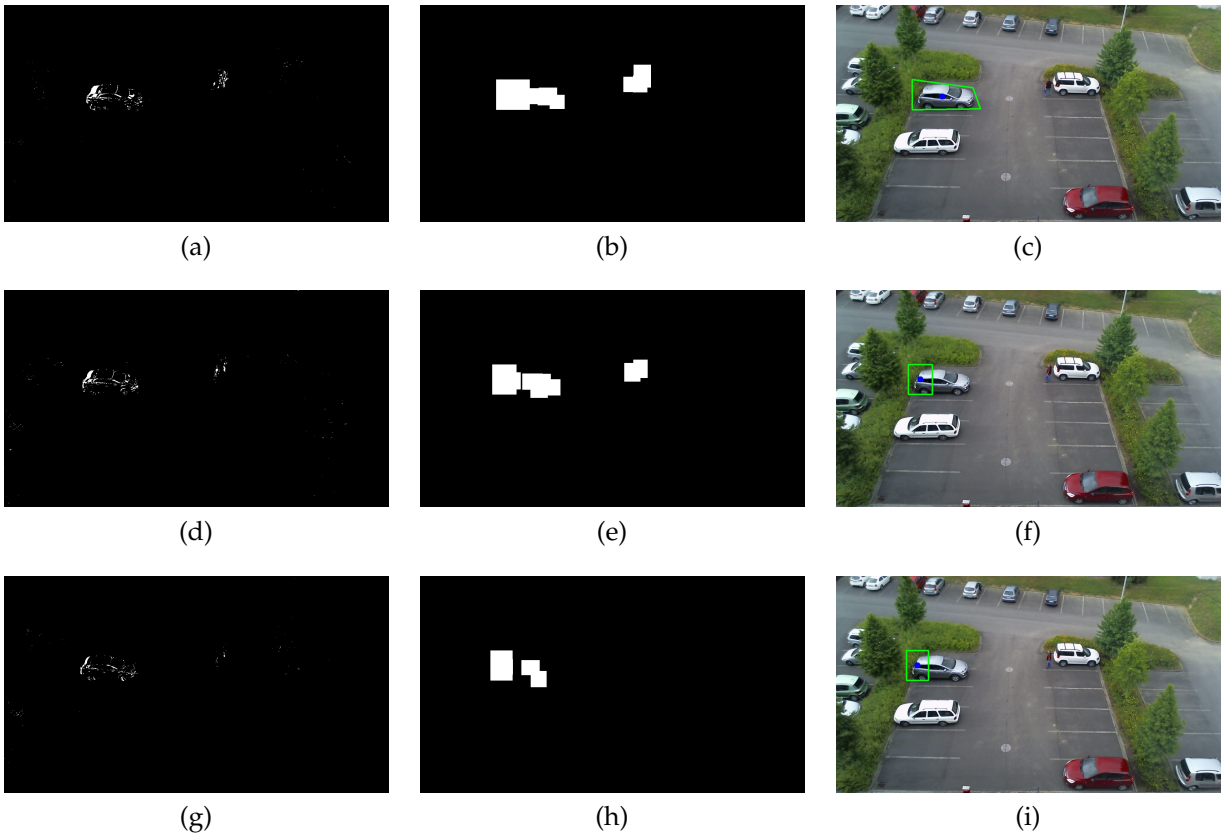


Figura 4.8: *Foreground*, dilatação, resultado.

Após a implementação, conclui-se que devido às dificuldades inerentes a este tipo de algoritmo, mencionadas acima, devido ao baixo processamento do Raspberry Pi, à dificuldade em traçar a trajetória do carro, e distinguir todo o contorno do carro, este tipo de algoritmo não é adequado para o produto a ser desenvolvido.

4.3 Performance do algoritmo proposto

O sistema foi testado com uma taxa de sucesso de 99.6% foi obtida em termos da precisão do estado dos lugares. As deteções incorretas deveram-se principalmente às fortes sombras, aos carros escuros sob uma sombra escura, no caso de chuva a deteção de manchas secas do carro após a saída dos carros, e ainda nos casos onde o topo do carro ultrapassa o previsto.

De forma genérica, o teste do algoritmo foi feito tal como representado no dia-

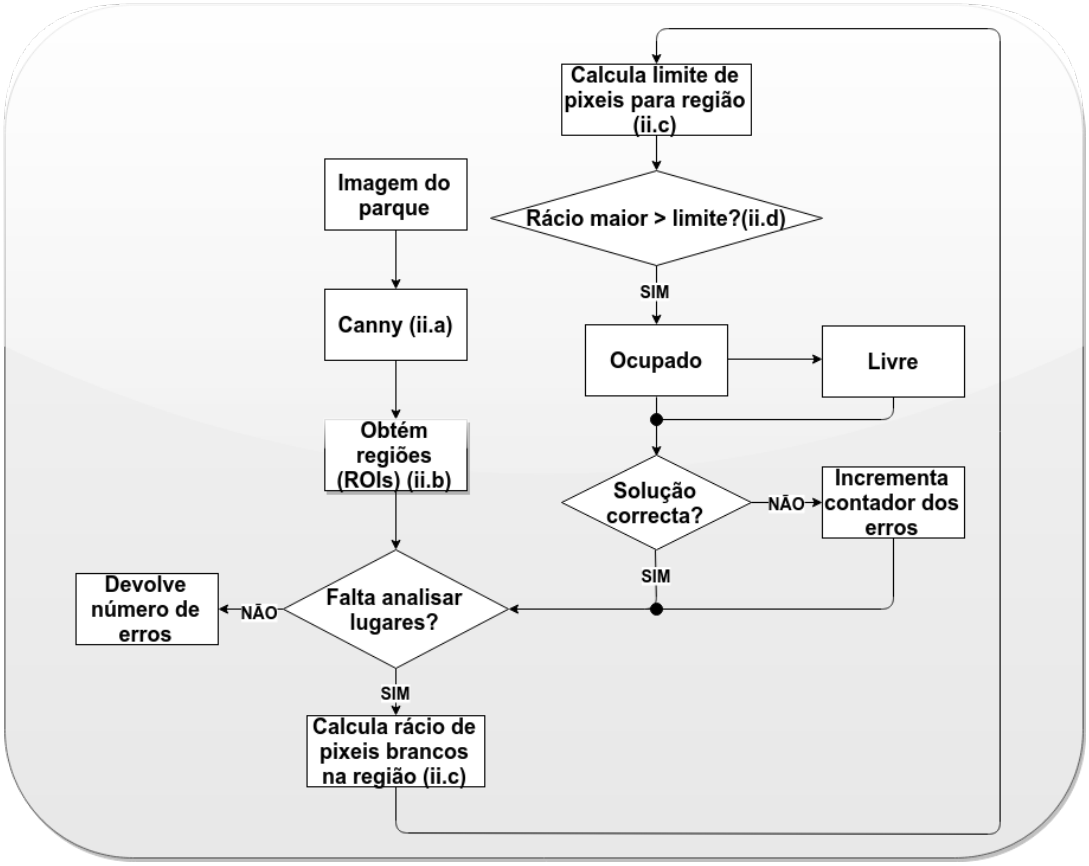


Figura 4.9: Teste do algoritmo.

(a) Canny depois de transformação de perspectiva - menos arestas

(b) Canny antes de transformação de perspetiva - mais arestas

Figura 4.10: Resultados do Canny

grama da Fig. 4.9. Relativamente à parte superior do parque de estacionamento, testes mostram que a realização do Canny deve ser feita antes da transformação de perspetiva, de forma a conservar o maior número de arestas - observem-se as imagens da Fig. 4.10.

Desta forma, existe uma maior contagem de pixeis brancos na presença de car-

total = 1640	Insucesso	Insucesso/total(%)
Deteção c/ rácio	6	0.3685
Deteção s/ rácio	24	1.4634

Tabela 4.1: Taxa de sucesso do algoritmo proposto.

Dispositivo	Duração do algoritmo (s)
PC	0.04
Raspberry Pi 2	0.76
Raspberry Pi modelo B	1.74

Tabela 4.2: Duração da análise de uma imagem.

ros, aumentando portanto a diferença com o limite de deteção, permitindo reduzir o número de falsos negativos (a não deteção de um carro, quando este se encontra presente).

A implementação do algoritmo com e sem rácio no limite de deteção, permite obter os resultados da Tab. 4.1. O algoritmo proposto tem como finalidade correr no Raspberry Pi 2, sendo o seu desempenho crucial. A Tab. 4.2, apresenta os resultados obtidos pelo algoritmo proposto para os diferentes dispositivos.

Em suma, com a taxa de sucesso de 99.6%, o tempo de execução do algoritmo no Raspberry Pi 2 (cerca de 0.76 segundos) e o baixo custo do produto (webcam HD e Raspberry Pi 2, cerca de 70 euros) conclui-se que a presente proposta, constitui um produto viável e com diversas vantagens em aplicações para parques de estacionamento.

5

Conclusões

Na presente dissertação é apresentada uma proposta de um sistema de estacionamento inteligente que se baseia na utilização de um produto de baixo custo. Do produto criado, salientam-se os seguintes contributos:

1. criação de um algoritmo de visão eficiente para uma plataforma de baixo custo e com baixos recursos de processamento;
2. deteção em diferentes tipos de estacionamento (longitudinal e transversal);
3. deteção em casos de ocorrência de oclusões parciais;
4. possibilidade de teste do algoritmo com diferentes entradas: fotos, conjuntos de fotos, *live feed*, etc;

5. deteção em parques que se encontram com uma rotação em relação à câmara, através de uma transformação de perspetiva;

6. seleção interativa dos lugares e carregamento automático das coordenadas, assim como o cálculo do seu comprimento, altura e centro;

7. armazenamento e *backup* de imagens num servidor.

Do produto criado, resultam os seguintes modelos:

1. segmentação de diferentes tipos de estacionamento;

2. funcionamento na ocorrência de oclusões parciais;

3. criação de um produto de baixo custo no contexto de sistemas de parque de estacionamento inteligentes;

4. criação do *ground truth* do parque.

Conclui-se, que os objetivos desta dissertação foram alcançados, dado ter-se conseguido criar um sistema de custo reduzido capaz de detetar o estado do parque de estacionamento, e com diversas condições meteorológicas. O sistema proposto oferece ainda um conjunto de ferramentas para a inicialização do sistema e teste do algoritmo.

O algoritmo proposto apresenta um método adequado para lidar com os casos em que ocorre oclusão parcial dos carros, recorrendo à região central do lugar para analisar a presença das arestas do carro.

As técnicas utilizadas são eficientes no que respeita aos recursos utilizados e podem ser implementados num módulo de processamento de baixo custo, como o Raspberry Pi 2, com uma HD webcam conectada. Nos casos onde a deteção falhe, o impacto será reduzido e o sistema recuperará em alguns segundos (conforme testes efetuados), com recurso à análise do momento atual e 4 instantes anteriores sobre o estado do parque.

Em parques de estacionamento amplos, o procedimento passa por dividir o parque em setores e atribuir um produto (Raspberry Pi + HD câmara) a cada setor. O acesso aos produtos é feito através do seu IP.

O trabalho futuro passa pela solução do problema das sombras e pela solução quando ocorrem oclusões totais. Ficará para novas abordagens o desenvolvimento de uma plataforma Web, e de um painel de visualização à entrada do parque. O sistema necessita ainda de ser testado com condições de inverno, mais especificamente neve. Nos diferentes passos do algoritmo, podem ainda ser propostas melhorias, assim como se poderão ser acrescentadas novas técnicas ou aperfeiçoadas as atuais. Entre os pontos a desenvolver no futuro, destacam-se: (a) funcionamento na ocorrência de oclusões totais; (b) *user interface* da aplicação e com suporte a múltiplas câmaras; (c) automatização do processo de inicialização e dos limites de deteção dos lugares.

Pode afirmar-se, como conclusão final, que os objetivos do trabalho foram cumpridos e que a taxa de sucesso de 99.6% é muito satisfatória na implementação deste algoritmo. Importa ainda destacar que foi submetido um artigo científico neste âmbito, o qual aguarda avaliação (Saraiva, D., Rodrigues, J. (2015) ***Low cost vision-guided parkink system***, submitted to 21th edition of the Portuguese Conference on Pattern Recognition, Faro, Portugal).

Referências bibliográficas

Almeida, P., Oliveira, L. S., Silva, E., Britto, A., and Koerich, A. (2013). Parking space detection using textural descriptors. In *IEEE International Conference onSystems, Man, and Cybernetics (SMC)*, pages 3603–3608. IEEE.

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc.

Brutzer, S., Höferlin, B., and Heidemann, G. (2011). Evaluation of background subtraction techniques for video surveillance. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 1937–1944. IEEE.

Buecheler, K. (2014). Emerging parking trends: Cities, universities, & corporate campuses, http://goo.gl/smb5kN (consultado 2015/09/23). Online.

Daubaras, A. et al. (2012). Vehicle detection based on magneto-resistive magnetic field sensor. *Elektronika ir Elektrotechnika*, 118(2):27–32.

Diamond (2015). Piezo sensors, , http://diamondtraffic.com/technicaldescription/125 (consultado 2015/09/23). Online.

Dictionaries, O. (2015). Scientific method. Oxford University Press.

Element14 (2015). Raspberry pi model B+, http://goo.gl/KzXGfl (consultado 2015/09/23). Online.

Elgammal, A., Harwood, D., and Davis, L. (2000). Non-parametric model for background subtraction. In *Computer Vision—ECCV 2000*, pages 751–767. Springer.

Faro, C. M. (2015). Pagamento de parquímetro por telemóvel chegou a faro, http://goo.gl/LZpSnA (consultado 2015/09/23). Online.

Fastprk (2015). How does it work?, http://goo.gl/vULJuc (consultado 2015/09/23). Online.

Fernandez-Sanchez, E. J., Diaz, J., and Ros, E. (2013). Background subtraction based on color and depth using active sensors. *Sensors*, 13(7):8895–8915.

Horprasert, T., Harwood, D., and Davis, L. S. (1999). A statistical approach for real-time robust background subtraction and shadow detection. In *IEEE ICCV*, volume 99, pages 1–19.

Ichihashi, H., Notsu, A., Honda, K., Katada, T., and Fujiyoshi, M. (2009). Vacant parking space detector for outdoor parking lot by using surveillance camera and fcm classifier. In *IEEE International Conference on Fuzzy Systems*, pages 127–134. IEEE.

Idris, M., Tamil, E., Razak, Z., Noor, N., and Kin, L. (2009). Smart parking system using image processing techniques in wireless sensor network environment, http://goo.gl/xzn3yt (consultado 2015/09/23). Online.

KaewTraKulPong, P. and Bowden, R. (2002). An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*, pages 135–144. Springer.

Karaman, M., Goldmann, L., Yu, D., and Sikora, T. (2005). Comparison of static background segmentation methods. In *Visual Communications and Image Processing 2005*, page 596069. International Society for Optics and Photonics.

Kianpisheh, A., Mustaffa, N., Limtrairut, P., and Keikhosrokiani, P. (2012). Smart parking system (SPS) architecture using ultrasonic detector. *International Journal of Software Engineering and Its Applications*, 6(3):55–58.

Kon, T. (1998). *Collision warning and avoidance system for crest vertical curves*. PhD thesis, Virginia Tech.

Kuhn, J. P., Bui, B. C., and Pieper, G. J. (1998). Acoustic sensor system for vehicle detection and multi-lane highway monitoring. US Patent 5,798,983.

Marsh (2000). The basics of loop vehicle detection, http://goo.gl/D6Bi2S (consultado 2015/09/23). Online.

Nedap (2014). Sensit - wireless vehicle detection sensors, http://goo.gl/cC4NLc (consultado 2015/09/23). Technical report, Nedap - identification systems.

Nortech (2015). Vehicle detectors, http://nortechdetection.com.au/detectors/ (consultado 2015/09/23). Online.

OpenCV (2014a). How to use background subtraction methods, http://goo.gl/ee4x7t (consultado 2015/09/23). Online.

OpenCV (2014b). Morphology operations, https://goo.gl/nwzajc (consultado 2015/09/23). Online.

OpenCV (2015). Geometric transformations of images, http://goo.gl/6b6FM5 (consultado 2015/09/23). Online.

Optex FA (2014). Photoelectric sensor process improvement, http://goo.gl/7lbeIp (consultado 2015/09/23). Online.

Pajankar, A. (2015). *Raspberry Pi Computer Vision Programming*. Packt Publishing.

Parking, S. (2015). Smart parking - a global parking business, http://www.smartparking.com/ (consultado 2015/09/23). Online.

Parksol (2013a). Sensor magnético, http://goo.gl/GMO8tH (consultado 2015/09/23). Online.

Parksol (2013b). Sensor ultra-sônico com indicação, http://goo.gl/7jQNxm (consultado 2015/09/23). Online.

Prabha, C. R. et al. (2014). Car parking management system. *ISSN (PRINT) : 2320 – 8945, Volume -2, Issue -3*.

Reddy, V., Sanderson, C., and Lovell, B. (2013). Improved foreground detection via block-based classifier cascade with probabilistic decision integration. *IEEE Tr. Circuits and Systems for Video Technology*, 23(1):83–93.

Sedco, M. (2015). Tc26-b - microwave vehicle motion sensor, http://goo.gl/JnYfiY (consultado 2015/09/23). Online.

SFpark (2014a). Sfpark - putting theory into practice. Technical report, San Franscisco Municipal Transportation Agency (SFMTA).

SFpark (2014b). *SFpark - Technical Manual*. San Franscisco Municipal Transportation Agency (SFMTA).

SFpark (2015). How sfpark works, http://sfpark.org/about-the-project/faq/how-sfpark-works/ (consultado 2015/09/23). Online.

SFPark (2015). Sfpark - about, http://sfpark.org/about-the-project/ (consultado 2015/09/23). Online.

SimpleCV (2015). Computer vision platform using python, http://simplecv.org/ (consultado 2015/09/23). Online.

SmartParking (2015). Smarteye, http://www.smartparking.com/technologies/smarteye (consultado 2015/09/23). Online.

SSET (2015). Introduction of ultrasonic parking detector/sensor, http://goo.gl/Fng2Uq (consultado 2015/09/23). Online.

Streetline (2015). Parking reinvented, http://goo.gl/A7H6c9 (consultado 2015/09/23). Online.

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

TCS (2015a). Single space monitoring, http://goo.gl/4c4CHs (consultado 2015/09/23). Online.

TCS, Q.-F. (2015b). 7 Stars Mall - Herzliya, Israel, http://goo.gl/CMvnz8 (consultado 2015/09/23). Online.

TCS, Q.-F. (2015c). Old Dominion University – Norfolk, VA, http://goo.gl/Qat6Zt (consultado 2015/09/23). Online.

TCS, Q.-F. (2015d). University of Louisville – Louisville, KY, http://goo.gl/8nDbXS (consultado 2015/09/23). Online.

True, N. (2007). Vacant parking space detection in static images. *University of California, San Diego*.

Ubuntu, W. (2015). Mount windows shares permanently, https://goo.gl/vEiHht(consultado 2015/09/23). Online.

UWB (2015a). Ing. Vladimír Pavliček, Ph.D., https://goo.gl/mNO9wV (consultado 2015/09/23). Online.

UWB (2015b). University of West Bohemia, https://www.zcu.cz/en/ (consultado 2015/09/23). Online.

Yamada, K. et al. (2001). A vehicle parking detection method using image segmentation. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 84(10):25–34.

Yusnita, R., Norbaya, F., and Basharuddin, N. (2012). Intelligent parking space detection system based on image processing. *International Journal of Innovation, Management and Technology*, 3(3):232–235.