Universidade de Aveiro Departamento de Engenharia Mecânica
2014

Ricardo Morais

Parametrização de Algoritmos para Deteção de Estrada a Bordo do ATLASCAR

Universidade de Aveiro Departamento de Engenharia Mecânica
2014

Ricardo Morais

Parametrização de Algoritmos para Deteção de Estrada a Bordo do ATLASCAR

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de Vítor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro.

O júri / The jury

Presidente / President **Prof. Doutor Jorge Augusto Fernandes Ferreira**
Professor Auxiliar da Universidade de Aveiro

Vogais / Committee **Prof. Doutor ??**
?? da Universidade ??

Prof. Doutor Vítor Manuel Ferreira dos Santos
Professor Associado da Universidade de Aveiro (orientador)

Agradecimentos / Acknowledgements

Deixo aqui o meu sincero agradecimento ao Prof. Dr. Vítor Santos, pela orientação, ajuda disponível, pela paciência, e pelos conselhos dados.

Um agradecimento especial os membros do Laboratório de Automação Robótica do Departamento de Engenharia Mecânica da Universidade de Aveiro. Em especial destaque ao Jorge Almeida , José Viana, Pedro Cruz e Emílio Estrelinha por tudo o que aturaram durante a minha passagem pelo laboratório, por todo o conhecimento partilhado e pelos inúmeros momentos de boa disposição.

Gostaria ainda de agradecer a todas as pessoas que permitiram indiretamente a realização deste trabalho.

Por fim, mas não menos importante, aos meus pais e á minha família um enorme obrigado pelo apoio incondicional e pelo esforço que fizeram para garantir a minha formação.

Palavras-chave AtlasCar, Visão artificial, Deteção de estrada, Linhas de marcação de estrada , Sistemas de apoio a condução, ROS

Resumo

O veículo AtlasCar é um protótipo desenvolvido pelo Laboratório de Automação e Robótica do Departamento de Engenharia Mecânica da Universidade de Aveiro, para o estudo de sistemas de segurança ativos e passivos e soluções de condução autónoma.

O objetivo deste trabalho é dotar o AtlasCar de um sistema de visão artificial capaz de reconhecer os limites da estrada, sendo para isso necessária a procura de marcas na estrada e/ou características das mesmas.

Este trabalho encontra-se dividido em duas partes principais: o desenvolvimento de módulos de software para a implementação de algoritmos com metodologias de deteção distintas e o estudo dos parâmetros que os compõem. O primeiro algoritmo baseia-se no método de RANSAC para efetuar a aproximação de uma *spline* aos limites da estrada, por sua vez o segundo algoritmo utiliza a transformada de Hough para efetuar esta aproximação. Os métodos apresentados foram testados em condições reais e foram analisadas as situações de falha, vantagens e desvantagens da aplicação de um método em relativamente ao outro.

Com a implementação deste sistema será possível a distinguir as zonas navegáveis das não navegáveis em contexto de estrada, permitindo assim a outros processos fazer um planeamento local da navegação.

Keywords Atlascar, Artificial vision, Road detection, Road lane markers ,Driver assistance systems, ROS

Abstract

The AtlasCar vehicle is a prototype developed by the Laboratory of Automation and Robotics, Department of Mechanical Engineering of the University of Aveiro, for the study of active and passive safety systems and autonomous driving solutions.
The objective of this work is to provide the AtlasCar an artificial vision system able to recognize road borders, identifying road lane markings and their characteristics.
This work is divided into two main parts: development of software modules to implement algorithms with different detection methodologies and the study of parameters that compose them. The first algorithm is based on RANSAC method for performing spline fitting to the road limits, the second algoritm uses the Hough transform to make this fitting. The methods performed were tested under real conditions, and failure situations, advantages and drawbacks of their applications are analysed and compared.
With the implementation of this system it will be possible to distinguish navigable from non-navigable road surface, thus endowing path planning algorithms to operate in local context.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 Enquadramento

O projeto ATLAS [2] surgiu no Laboratório de Automação e Robótica da Universidade de Aveiro com o objetivo de desenvolver soluções na área da condução autónoma. Inicialmente este projeto desenvolveu pequenos robôs (fig. 1.1), e devido aos sucessos obtidos nestas competições em que participaram, a equipa Atlas decidiu apostar num modelo á escala real [14] [16] [13]. Foi a partir dai que nasceu o AtlasCar (fig. 1.2).

O AtlasCar é um protótipo onde são desenvolvidos e testados diversos sistemas que têm como principais objetivos a assistência à condução.

Este veiculo encontra-se equipado com diversos equipamentos e sensores que não se encontram na maioria dos carros, dos quais se destacam computadores, câmaras, sensores laser 2D e 3D, entre outros. Graças a estes sensores, em conjunto com atuadores e controladores, o AtlasCar tem a capacidade de detetar objetos e de saber onde se encontra relativamente a eles, tornando-o capaz de se movimentar de forma autónoma.

1.2 Apresentação do Problema

A necessidade de saber onde se encontram os limites da estrada é vital para que a condução assistida, ou autónoma, ocorra com o mínimo de risco. Por essa razão, a maioria das marcas de automóveis têm instalado nos veículos sistemas que permitem uma condução cada vez mais

Figura 1.1: Robôs ATLAS para condução autónoma.

Figura 1.2: AtlasCar.

controlada e segura.

No contexto do projeto AtlasCar, pretende-se abordar a temática do reconhecimento de estrada em dois níveis principais. Numa primeira etapa, pretende-se implementar vários algoritmos de deteção de estrada. Numa segunda fase do trabalho pretende-se fazer um estudo dos parâmetros usados pelos diferentes algoritmos.

A intenção é desenvolver um algoritmo capaz de segmentar a zona navegável; esta aplicação deverá ser capaz de fornecer informações vitais para o planeamento da trajetória a tomar pelo veículo, desde que esta trajetória se encontre ao alcance da câmara.

1.3 Objetivos

Este trabalho tem como objetivo o desenvolvimento de um módulo de software capaz de reconhecer os limites da estrada, com recurso a imagem.

Para determinar os limites da estrada é necessário a subscrição de uma imagem e nela determinar a localização desses limites.

Visto que o projeto AtlasCar é desenvolvido em ambiente *Robot Operation System* (ROS), é necessária a familiarização com o seu funcionamento.

1.4 *Robot Operation System*

O texto de Quigley [15] descre o ROS como um ambiente de desenvolvimento especialmente ligado a aplicações em robótica, este permite reduzir bastante a complexidade de projetos de maior dimensão devido à sua arquitetura modular. Este tipo de arquitetura permite reduzir um grande projeto em pequenos módulos, cada um com uma aplicação específica, facilitando o "debugging" e a sua compreensão. Para além de reduzir a complexidade de organização, permite a reutilização desses módulos em outras aplicações.

Uma outra caraterística do ROS é a de permitir que, para um mesmo projeto, se utilizem linguagens de programação distintas nos diversos módulos desenvolvidos. Neste ambiente, a comunicação entre processos é feita através de mensagens. As mensagens em ROS são estruturas predefinidas.

A comunicação entre módulos é realizada através da passagem de mensagens entre si. Um módulo envia uma mensagem através da publicação de um tópico ("string" que identifica a mensagem). Em projetos com alguma dimensão é comum existirem diversos tópicos a serem publicados e subscritos por um ou mais módulos.

Em sistemas robóticos há a necessidade de seguir e permitir efetuar transformações no espaço entre referenciais de vários sensores e entre robôs móveis e referenciais fixos. Por esse motivo, foi desenvolvido em ROS uma ferramenta que permite realizar e detetar essas transformações. Esta ferramenta relaciona todos os referenciais presentes num determinado sistema, não sendo necessária a definição das matrizes de transformação sempre que se deseja publicar uma transformação.

Em grandes projetos (desenvolvidos em ROS), por vezes, torna-se complicado observar o estado de todo o sistema. Nesse sentido, existem diversas ferramentas que permitem analisar o estado das comunicações entre módulos, que transformações se encontram a ser realizadas, um osciloscópio virtual que permite observar os dados que se encontram a circular entre módulos, entre muitas outras. Estas ferramentas facilitam o desenvolvimento e o ”debugging”de sistemas.

Capítulo 2

Estado da Arte

A deteção de estrada com visão pode ser dividida em duas partes: deteção de estradas estruturadas, com marcações a delimitar as faixas de rodagem como, por exemplo, as estradas urbanas, e a deteção de estradas não estruturadas, sem marcações como por exemplo as estradas rurais [8]. Para resolver este problema foram tentadas várias abordagens. Neste capitulo serão apresentadas algumas das abordagens tomadas, incluindo algumas no contexto do próprio projeto Atlas.

2.1 General road detection from a single image [8]

O método utilizado por H.Kong em 2010 aborda a segmentação da estrada baseada no ponto de fuga, esta encontra-se dividida em duas etapas; na primeira fase é estimado o ponto de fuga que é associado à reta principal da estrada, seguido de uma segmentação da estrada baseada neste ponto.

Na primeira fase é feita uma estimativa da orientação da textura da imagem usando um filtro de Gabor a duas dimensões. Após a orientação da textura ter sido calculada em cada pixel da imagem é feita uma votação local que é feita através dos pixels que se encontram nas proximidades do pixel candidato.

Na segunda etapa é estimada a localização dos bordos da estrada com base no ponto encontrado previamente. Para tal, são considerados 29 segmentos de reta igualmente espaçados com origem no ponto de fuga.

O bordo dominante é aquele que maximiza a diferença de cor entre as regiões que separa, e também aquele que tem uma orientação mais consistente.

Figura 2.1: Segmentação da estrada baseada no ponto de fuga (adaptado de [8]).

Os autores testaram este algoritmo em 1003 imagens de estradas com uma larga variação de cor, textura, iluminação e ambiente envolvente. No que diz respeito à localização dos pontos de fuga o método proposto apresenta em média cerca de 9 pixels de erro em vez de 14 obtidos pelo método de C.Rasmussen[6], por sua vez a localização dos bordos dominantes e segmentação da estrada apresenta uma maior eficiência podendo assim ser efetuados em tempo real.

2.2 Adaptive fuzzy color segmentation with neural network for road detections [4]

Este método apresentado por Chieh-Li Chen e Chung-Li Tai em 2010 utiliza a segmentação por cor para fazer a deteção da estrada. Para tal, o método parte do pressuposto que a imagem atual já é de estrada, ou seja, a região inferior da imagem deve pertencer a estrada (fig. 2.2).

Figura 2.2: Referencia da estrada (retirado de [4]).

Esta amostra é usada para definir um intervalo de procura no estaço de cor HSV; esse intervalo é usado por uma rede neuronal previamente treinada para fazer a procura de estrada na imagem. Durantes os ensaios experimentais os autores verificaram que as cores com valores de intensidade perto de 0 e 1 eram de difícil deteção, desta forma foi usada um ajuste de intensidade da imagem (fig. 2.3).

Figura 2.3: Ajuste da intensidade das imagens (adaptado de [4]).

2.3 A Robust Approach for Road Detection With Shadow Detection Removal Technique [3]

Em 2014 N.Ahmed Salim, X.Cheng e X.Degui apresentaram um método que utiliza a remoção de sombras e a segmentação por cor para a deteção de estradas não estruturadas, e encontra-se dividido em duas partes: a primeira dedica-se à deteção e remoção de sombras, e á segunda parte à segmentação da estrada.

A deteção da estrada é feita recorrendo ao método Otsu no espaço HSV, que é baseado na matiz(H), saturação(S) e valor de brilho(V). Este espaço de cor é vantajoso pois as sombras alteram o brilho do fundo da imagem mas não afetam a matiz nem a saturação [17]. Depois de obtidas as sombras, é necessário removê-las: para tal é usada uma área de "buffer", que é uma área à volta da área da sombra que serve para compensar os valores da média e variância da região da sombra. O valor do pixel é compensado pela expressão (2.1).

$$I'(i,j) = \mu_{buff,n} + \frac{I_n(i,j) - \mu_n}{\sigma_{buff,n}}\sigma_n \tag{2.1}$$

onde $I'_n(i,j)$ é o valor compensado do pixel da sombra, μ_{buff}, e σ_{buff} são a média e a variância dos pixels da imagem I na localização $I_{buff,n}$, e ainda μ_n e σ_{buff} a média e variância dos pixels da imagem I na localização I_n

Na segunda parte, é feita a segmentação da estrada recorrendo ao classificador de Bayes, onde as distribuições de probabilidade são aproximadas por histogramas. Este classificador contém três fases distintas: estimativa de probabilidade, filtragem e decisão.

De acordo com o classificador de Bayes a decisão pode ser tomada comparando o quociente da probabilidade Hr e Hn para cada pixel da imagem de input Im

$$H_r[r,g,b] = \alpha H_r[r,g,b] + (1+\alpha)H_t[r,g,b] \tag{2.2}$$

$$H_r \approx (P(Color) \mid Road)) \tag{2.3}$$

$$H_n \approx (P(Color \mid Non - road)) \tag{2.4}$$

como a probabilidade de um pixel ser parte da estrada não depende só da sua cor mas também dos seus vizinhos, é aplicado um filtro de mediana a uma dada vizinhança.

Figura 2.4: Demonstração da segmentação de estrada usando este método (retirada de [3]).

As experiencias realizadas por este autor focaram-se em cenários de auto-estrada, estradas urbanas, retas e curvas, foram ainda testados alguns cenário fora de estrada. Para a avaliação dos resultados obtidos foi usado o método *three pixel wise measure* (qualidade), assim como o rácio de deteção e precisão de deteção. Estes revelaram que este método é robusto e eficiente.

2.4 Real time detection of Lane Markers in Urban Streets [10]

O método descrito por Mohamed Aly para a deteção das linhas da estrada encontra-se dividido em duas etapas. Numa primeira fase é utilizada a técnica de IPM (Inverse Perspective Mapping), que consiste na correção de perspetiva da imagem. Após a correção de perspetiva da imagem, esta é filtrada e binarizada, recorrendo ao uso de filtros gaussianos.

Para fazer a deteção das linhas são usadas duas técnicas, é construído um histograma com a soma dos valores de cada coluna na imagem, de onde os máximos do histograma são uma boa referencia para a posição das linhas. É então definida uma região de procura em torno da posição estimada para as linhas, sendo utilizado RANSAC nesta região para fazer um ajuste adequado.

Após terem sido encontradas, as linhas são transpostas para a imagem original, sem correção de perspetiva, onde é feito o seu prolongamento e um ajuste para compensar erros na inversão da perspetiva.

Figura 2.5: Sequência de imagens do processamento do algoritmo. (Adaptada de [5]).

Para obterem resultados, os autores usaram uma sequências de imagens de estradas urbanas com e sem sombras bem como com e sem curvaturas. Os resultados obtidos sugerem que o método proposto tem taxas altas de deteção trabalhando em torno dos 50hz.

2.5 Perception and Navigation for the DARPA Urbam Chalenge [5]

O texto de D. C.Moore, J. Leonard, e S. Teller descreve um método para deteção de estrada que se encontra dividido em quatro passos: filtro de correspondência, deteção de "features", uso de RANSAC e o seguimento de hipóteses.

No primeiro passo, a imagem é filtrada usando um elemento estruturante horizontal ajustado ao tamanho esperado da largura da linha, baseado na localização do plano do chão e nas características intrínsecas e extrínsecas da câmara. No passo seguinte, a deteção de "features" das linhas é feita usando os picos obtidos no passo anterior. Bons picos são obtidos da resposta do filtro quando este coincide com elevados gradientes da imagem perpendiculares a direção esperada para a estrada.

Seguidamente, é usado o método de RANSAC (*Randon Samplig Concensus*), para selecionar grupos com características das linhas, de tal forma que estas indiquem uma única marcação de acordo com um modelo especifico de linha.

Por ultimo, as linhas detetadas são seguidas ao longo do tempo, para assegurar que são consistentes.

Figura 2.6: Demonstração da deteção das linhas (retirada de [5]).

2.6 Real time extraction of road border lines using simple statistical descriptors [11]

M.Oliveira e V.Santos apresentaram em 2008 um algoritmo onde é usada uma imagem de perspetiva invertida ou *Bird View* (fig. 2.7), á qual é aplicada a um *TopHat* para facilitar a extração de linhas em condições de luminosidade adversas, como por exemplo sombras.

Figura 2.7: Imagem de perspetiva normal e perspetiva invertida (retirado de [11]).

Admitindo que as linhas delimitadoras da estrada interceptam sempre a parte inferior da imagem, é traçada uma linha de procura junto a esta parte, para a qual é traçado um perfil de brilho (fig. 2.8).

Figura 2.8: Resultado do *TopHat* na imagem de perspetiva invertida. A linha a vermelho representa a posição da linha de procura (retirado de [11]).

Este perfil é dividido em grupos com base no gradiente máximo local, sendo descartados os grupos que tem uma quantidade de pixels superior à esperada. Cada grupo de pixels representa uma possível interseção entre a linha de procura e as linhas de marcação da estrada. Com base no ponto central destes grupos é feita a reconstrução da linha candidata.

Para cada candidato é feita uma procura de pontos de interesse sobre a linha, que consiste na deteção do primeiro pixel branco numa procura da direita para a esquerda num conjunto de tentativas igualmente espaçadas verticalmente. Daqui resulta um vector de pontos que se encontram no bordo direito da linha. São calculados os segmentos definidos por cada dois pontos consecutivos assim como a sua orientação e respetivos ângulos.

Finalmente, para cada vector normal ao ponto são definidos iterativamente segmentos com a largura de um pixel, que proporcionam uma indicação para a largura da linha em cada ponto (fig. 2.9).

Figura 2.9: Representação da procura de pontos sobre o bordo da linha (retirado de [11]).

Este algoritmo foi testado pelos autores em 45 imagens de estrada retiradas aleatoriamente de um troço com cerca de 200 km, obtendo uma taxa de sucesso de 76.1% e 32.6% de falsos positivos.

2.7 "Short term path planning using a multiple hypothesis evaluation approach for an autonomous driving competition"[12]

Em 2011 M. Oliveira, V. Santos, e A. D. Sappa para obter informações sensoriais usaram a técnica de *Inverse Perspective Mapping (IPM)*, que consiste no mapeamento de pontos na imagem de perspetiva corrigida.

O uso da IPM facilita significativamente a escolha dos elementos estruturantes dos filtros de convolução e a deteção de caraterísticas de interesse. Esta técnica necessita de algum conhecimento prévio, nomeadamente, a orientação e os parâmetros extrínsecos da câmara. O uso desta técnica assume que toda a estrada à frente é plana; caso existam obstáculos ou desníveis na estrada, estes serão representandos distorcidos na IPM. Para fazer a deteção destes é usado um *laser range finder (LRF)* paralelo ao plano da estrada.

A deteção das linhas da estrada é feita com base em trabalho anterior, mas a ideia principal é obter candidatos através da análise do perfil de brilho da IPM. Esses candidatos são caraterizados com base em parâmetros estatísticos. Na IPM, parâmetros como a largura da linha e o raio de curvatura estão bem definidos (fig.2.10).

Figura 2.10: Imagens das câmaras do robot, convertidas para uma única imagem de IPM e deteção das linhas de marcação.

Este algoritmo foi aplicado com sucesso aos robôs de pequena escala do projeto Atlas, onde foram efetuados testes exaustivos durante as suas participações em competições de veículos autónomos, onde foram ganhas seis edições consecutivas da competição de veículos autónomos do Festival nacional de robótica, de 2006 até 2011. Demonstrando assim a capacidade de navegação deste algoritmo em espaços interiores com cenários complexos.

Capítulo 3

Algoritmos Implementados

Neste capítulo são apresentadas as metodologias utilizadas neste trabalho para a segmentação de estrada, bem como as métricas definidas para avaliar os resultados. Serão descritos todos os passos dos algoritmos utilizados e as alterações necessárias para este caso particular.

Os algoritmos utilizados foram "Lane Tracker" e "Real Time Detection of Lane Markers in Urban Streets", estes destacaram dos restantes devia á metodologia usada por estes para deteção das marcações da estrada. Estas metologias apresentavam melhores resultados face as propostas.

3.1 Real time Detection of Lane Markers in Urban Streets [10]

3.1.1 Descrição

A abordagem deste algoritmo encontra-se dividida em duas etapas. A primeira etapa é baseada na técnica da *Inverse Perspective Mapping (IPM)*, que consiste numa correção da perspetiva da imagem. Esta imagem é depois binarizada de onde são obtidos somente os valores mais altos, onde são detetadas as linhas retas usando uma transformada de *Hough* simplificada, seguida de um filtro de RANSAC para detetar as linhas curvas e, para refinar as retas encontradas anteriormente. Na segunda etapa, as linhas encontradas na imagem de perspetiva invertidas são revertidas para a imagem original onde é feita a correção e o seu prolongamento.

3.1.2 Abordagem

O primeiro passo deste algoritmo é gerar uma imagem de perspetiva invertida da estrada. Este processo tem como objetivo eliminar o efeito de perspetiva da estrada, ou seja, as linhas de marcação deixam de parecer que convergem no horizonte para estarem paralelas entre si, e ainda reduzir o tempo de processamento das imagens focando a nossa atenção somente numa região da imagem.

Para obter a imagem de perspetiva invertida da imagem original é necessário assumir que a estrada é completamente plana, e conhecer os parâmetros intrínsecos da câmara, distância focal e centro ótico, e ainda alguns parâmetros extrínsecos tais como a altura da câmera em relação ao plano da estrada, o angulo de *pitch* e *yaw*. Inicialmente foi definido o referencial mundo $\{F_w\} = \{X_w, Y_w, Z_w\}$ centrado com o centro ótico da câmara, o referencial da câmara $\{F_c\} = \{X_c, Y_c, Z_c\}$ e o referencial da imagem $\{F_i\} = \{u, v\}$ como representado na figura 3.1.

Figura 3.1: Referenciais usados (retirada de [10]).

No referencial da câmara o eixo X_c está no plano do mundo $X_w Y_w$ e ângulo de *pitch* α e de *yaw* β com o eixo ótico. Á altura da câmara ao plano do chão é dada por h. Para qualquer ponto do plano da imagem ${}^iP = \{u, v, 1, 1\}$, a sua projeção no plano da estrada é dada pela transformação homogénea 3.1.

$$
{}^g_iT = h \begin{bmatrix} -\frac{1}{f_u}c_2 & \frac{1}{f_u}s_1s_2 & \frac{1}{f_u}c_uc_2 - \frac{1}{f_u}c_vs_1s_2 - c_1s_2 & 0 \\ \frac{1}{f_u}s_2 & \frac{1}{f_v}s_1c_1 & -\frac{1}{f_u}c_us_2 - \frac{1}{f_v}c_vs_1c_2 - c_1c_2 & 0 \\ 0 & \frac{1}{f_v}c_1 & -\frac{i}{f_v}c_vc_1 + s_1 & 0 \\ 0 & -\frac{1}{hf_v}c_1 & \frac{1}{hf_v}c_vc_1 - \frac{i}{h}s_1 & 0 \end{bmatrix} \tag{3.1}
$$

Por exemplo: ${}^gP = {}^g_iT{*}{}^iP$ é o ponto no plano do chão correspondente ao ponto ${}^iP$ no plano da imagem, onde $\{f_u, f_v\}$ são as distâncias focais verticais e horizontais, respetivamente, $\{c_u, c_v\}$ são as coordenadas do centro focal em que $c_1 = \cos\alpha$,$c_2 = \cos\beta$, $s_1 = \sin\alpha$ e $s_2 = \sin\beta$. Estas transformações podem ser facilmente calculadas para centenas de pontos se usadas na forma matricial. A transformação inversa pode ser facilmente encontrada usando 3.2:

$$
{}^{i}_{g}T = \begin{bmatrix} f_u c_2 + c_u c_1 s_2 & c_u c_1 c_2 - s_2 f_u & -c_u s_1 & 0 \\ s_2\left(c_u c_1 - f_v s_1\right) & c_2\left(c_v c_1 - f_v s_1\right) & -f_v c_1 - c_v s_1 & 0 \\ c_1 s_2 & c_1 c_2 & -s_1 & 0 \\ c_1 s_2 & c_1 c_2 & -s_1 & 0 \end{bmatrix} \tag{3.2}
$$

onde um ponto no plano do chão ${}^{g}P = \{x_g, y_g, -h, 1\}$ pode ser descrito no plano da imagem em pixels por ${}^{i}P = {}^{i}_{g}T^{g}P$.

Usando estas duas transformações podemos projetar uma região de interesse da imagem original para um plano no chão. Na figura 3.2 à direita podemos observar um exemplo de uma imagem de perspetiva invertida de uma região de interesse representada na imagem original a vermelho.

Figura 3.2: Exemplo de imagem de perspetiva invertida. À esquerda: a imagem original com a região de interesse a vermelho. À direita: a respetiva imagem de perspetiva invertida correspondente à janela.

Filtragem e binarização

A imagem de perspetiva invertida é depois filtrada por um filtro gaussiano a duas dimensões. A direção vertical é uma suavização gaussiana, onde σ_y é ajustado de acordo com a altura do segmento de linha a ser detetado: $f_v\left(y\right) = \exp(-\frac{1}{2\sigma_y^2}y^2)$. A direção horizontal é a segunda derivada da gaussiana, onde σ_x é ajustado de acordo com a largura esperada das linhas: $f_u\left(x\right) = \frac{1}{\sigma_x^2}\exp(-\frac{x}{2\sigma_x})\left(1 - \frac{x}{\sigma_x}\right)$. O filtro é ajustado especificamente para linhas verticais bem destacadas em fundos pretos com larguras especificas.

O uso deste filtro separado permite uma implementação mais eficiente e muito mais rápida do que usar um filtro não separado. Na figura 3.3, à esquerda, temos o elemento estruturante usado no filtro, o resultado da imagem depois de filtrada ao centro, e o resultado da imagem binarizada á direita. Como se pode ver pela imagem filtrada, existe uma resposta muito grande às linhas, onde só retemos os valores mais altos. Isto é feito selecionando q% da imagem filtrada e removendo todos os valores abaixo desse limiar. Neste passo é assumido que o veículo se encontra quase paralelo às linhas.

Figura 3.3: Imagem filtrada e binarizada. À esquerda: temos o elemento estruturante usado no filtro. No meio: a imagem depois de filtrada. À direita: a imagem depois de binarizada (retirado de [10]).

3.1.3 Deteção das linhas

Esta etapa é responsável pela deteção das linhas na imagem binarizada. São usadas duas técnicas: uma versão simplificada da transformada de *Hough* para identificar as linhas existentes na imagem, seguido de um “fitting” de linhas utilizando RANSAC para um ajuste mais apropriado. É criado um histograma com o valor do total dos pixels de cada coluna da imagem, este histograma é suavizado por um filtro gaussiano, e seguidamente são encontrados os máximos para encontrar a posição das linhas, que será usada posteriormente para obter uma precisão superior, fazendo o ajuste a uma *spline* ao máximo local e aos dois vizinhos mais próximos. Por fim, as linhas “muito próximas” são agrupadas para eliminar múltiplas deteções da mesma linha. Na figura 3.4 podemos ver o resultado desta operação.

Figura 3.4: À esquerda: histograma com o total dos pixels para cada coluna com os máximos locais a vermelho. À direita: Linhas detetadas depois de agrupadas (retirado de [10]).

O próximo passo é obter uma deteção mais precisa dos parâmetros que descrevem as linhas usando RANSAC. Para cada linha vertical detetada acima, é criada uma região de interesse à sua volta (fig. 3.5).

Figura 3.5: À esquerda: Região de interesse à volta de uma linha do passo anterior, e a vermelho a linha encontrada. À direita: Linhas resultantes deste passo (retirado de [10]).

Deteção mais precisa dos parâmetros que descrevem as linhas utilizando RANSAC

O passo anterior devolve um linha candidata na imagem, que depois é refinada neste passo. Para cada linha é criada uma região de interesse à sua volta onde é feito o ajuste das linhas. O ajuste é iniciado com as linhas da imagem anterior, pois são o ponto de partida para este passo. O ajuste usado é uma *spline* de terceira ordem. Uma *spline* é uma curva definida por dois ou mais pontos de controlo. Os pontos de controlo que ficam na curva são chamados de nós e os demais definem a tangente a curva. Neste caso são usadas *splines* de Bézier, que são curvas polinomiais expressas pela interpolação linear entre alguns pontos representativos: os pontos de controle. Por exemplo, a *spline* de Bézier representada na fig.3.6 é definida pelos pontos (A,B,C,D), é delimitada pelos nós A e D e nesses nós, a curva é tangente ao vetores AB e DC. Variando as posições dos pontos B e C, a curva apenas varia sua inclinação, mas continua passando pelos pontos A e D.

Figura 3.6: Spline de Bézier com nós (A, D) e pontos de controle (A, B, C, D).

Onde os pontos de controlo formam um polígono delimitador em torno da própria linha. A *spline* de terceiro grau é definida por:

$$\begin{aligned} Q(t) &= T(t)\,MP \\ &= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \end{aligned} \tag{3.3}$$

onde $t \in [0,1]$, $Q(0) = P_0$ e $Q(1) = P_3$ e os pontos P_1 e P_2 controlam a forma da linha e M o declive (fig.3.7).

Algoritmo 1 Ajuste de linhas utilizando RANSAC

```
for $i = 1$ to $numIterations$ do
  $points = getRandomSample()$
  $spline = fitSpline(points)$
  $score = computeSplineScore(spline)$
  if $score > bestScore$ then
    $bestSpline = spline$
  end if
end for
```

O Algoritmo 1 descreve o ajuste de linhas utilizando RANSAC. Existem três funções básicas dentro do ciclo principal:

1. getRamdomSample(): Esta função retira amostras dos pontos disponíveis na região de interesse que foi passada para o passo de RANSAC. É usada uma abordagem de amostragem ponderada, cujos pesos são proporcionais ao valor do pixel da imagem binarizada. Isto ajuda a escolher pontos mais relevantes.

2. fitSpline(): Esta função usa um certo número de pontos e faz o ajuste da curva de Bézier usando o método dos mínimos quadrados. Dada uma amostra de n pontos, é dado um valor $t_i \in [0, 1]$ a cada ponto $p_i = (u_i, v_i)$ na amostra, onde t_i é proporcional à soma cumulativa da distância euclidiana do ponto p_i ao primeiro ponto p_1 (3.4). Definindo o ponto $p_0 = p_1$ e obtemos :

   $$t_i = \frac{\sum_{j=1}^{i} d(p_j, p_{j-1})}{\sum_{j=1}^{n} d(p_j, p_{j-1})} \qquad t_i = 1..n \tag{3.4}$$

   onde $d(p_i, p_j) = \sqrt{(u_i - u_j) + (v_i - v_j)}$. Isto força $t_1 = 0$ e $t_n = 0$ o que corresponde ao primeiro e último ponto da linha, respetivamente. De seguida, as matrizes Q e T são definidas pela expressão (3.5)

   $$Q = \begin{bmatrix} p_1 \\ ... \\ p_n \end{bmatrix} \qquad T = \begin{bmatrix} t_1^3 & t_1^2 & t_1 & 1 \\ & ... & & \\ t_n^3 & t_n^2 & t_n & 1 \end{bmatrix} \tag{3.5}$$

   e revolve-se a matriz P usando a pseudo-inversa:

   $$P = (TM)Q \tag{3.6}$$

   obtendo pontos de controlo para a linha que minimiza a soma do erro quadrático de correspondência ao pontos de amostra.

3. computeSplineScore(): no RANSAC normal poderíamos estar interessados em calcular a distância normal de cada ponto a uma *spline* de terceiro grau para decidir o grau de confiança da linha, contudo, isto requer a resolução de uma equação de quinto grau para cada ponto. Em vez disso, seguiu-se uma abordagem mais eficiente: é calculado o grau

de confiança da linha por iteração e depois é somado os valores dos pixels pertencentes à linha. Também é tida em conta a linearidade e comprimento da linha, penalizando as mais curtas e menos retilíneas. O grau de confiança é dado por:

$$score = s(1 + k_1 l' + k_2 \theta') \tag{3.7}$$

onde s é o grau de confiança bruto da linha (a soma do valor dos pixels da linha), l' o comprimento da linha normalizado definido por $l' = \frac{l}{v} - 1$ e l é o comprimento da linha e v a altura da imagem, portanto $l' = 0$ significa que a linha é mais comprida que o definido e $l' = -1$ que a linha é mais curta. θ' é a curvatura da linha normalizada definida por $\theta' = \frac{\theta - 1}{2}$ onde θ é a média do cosseno dos ângulos entre as linhas de união entre os pontos de controlo, por exemplo, $\theta = \frac{(\cos\theta_1 + \cos\theta_2)}{2}$, e k_2 e k_1 são fatores regularizadores (fig. 3.7). Esta fórmula para o grau de confiança faz com que as linhas mais longas e menos curvas sejam menos penalizadas que as mais curtas e curvas.

Figura 3.7: Calculo do grau de confiança (retirado de [10]).

Pós-processamento

O passo final deste algoritmo é o pós-processamento dos resultados obtidos nos passos anteriores para melhorar a localização da linha e prolongar-la na imagem (fig. 3.8). Isto é feito na imagem de perspetiva após a projeção das linhas encontrada, onde são realizados os seguintes passos:

1. Localização: São retirados pontos equidistantes de amostragem da spline, através dos quais são traçados segmentos de reta normais a direção tangente da *spline* nos pontos. Seguidamente são obtidos os perfis de tons de cinza para estes segmentos de reta, onde são localizados os máximos locais. Estes devolvem uma localização mais correta dos pontos da *spline* e permitem uma deteção mais precisa das linhas da estrada.

2. Extensão: É feita uma extensão das *splines* para as ajustar ainda mais às linhas da estrada. Isto é feito prolongando a *spline* a partir pontos extremos ao longo da direção da sua tangente. Nestes novos segmentos de reta é efetuado novamente o passo de localização. Os novos pontos só são aceites caso se encontrem dentro de limites estabelecidos para os valores máximos de escala de tons de cinza e variação de orientação da *spline* dominante.

3. Verificação geométrica: Após cada um dos passos anteriores é feita uma verificação geométrica das *splines*; caso estas excedam os limites de curvatura e comprimento, são substituídas pelas *splines* do passo anterior.

Figura 3.8: Localização e extensão da linha (retirado de [10]).

Figura 3.9: Esquerda: Linhas antes do pós-processamento a verde. Direita: Linhas após o pós-processamento a azul.

3.1.4 Implementação

Para ser possível implementar este algoritmo no AtlasCar foram necessárias algumas alterações, nomeadamente ao nível da leitura das definições da câmara, da maneira como as imagens são lidas pelo algoritmo e processamento das mesmas. Para fazer o processamento das imagens foi utilizada a biblioteca OpenCV (Open Source Computer Vision Library) [1].

Assim, foi criado um módulo ROS que subscreve as mensagens de imagem e de configuração da câmara a usar, e gera um ficheiro com todas as configurações necessárias para o algoritmo. Essas configurações são calculadas por este módulo com base nos parâmetros intrínsecos e extrínsecos da câmara, tais como a distância focal e a transformação entre o referencial do carro e o da câmara.

Quando o algoritmo é iniciado, o primeiro passo é ler os ficheiros de configuração da câmara e das linhas, de onde são retiradas as configurações para criar a imagem de perspetiva invertida, filtros de RANSAC e parâmetros de binarização.

Inicialmente, a leitura das imagens era feita a partir de "frames" individuais guardados previamente em disco, o que impossibilitava o uso de imagens de tempo real. Para resolver esta questão, as imagens são recebidas em mensagens de imagem ROS; porém, como as menssagens ROS não usam o mesmo formato que o *OpenCV*, é necessário o uso da biblioteca *CvBridge* que faz a conversão das imagens para poderem ser usadas no *OpenCV* (fig. 3.10).

Figura 3.10: Converter mensagens de imagens ROS em imagens do OpenCV.

A principal alteração feita no processamento deste algoritmo foi a introdução de um *TopHat* antes do processamento da imagem de perspetiva invertida, para facilitar a extração de linhas em condições de luminosidade adversas como, por exemplo, sombras. O *TopHat* é uma operação que extrai pequenos elementos e detalhes da imagem, e, é definido como a diferença entre a imagem original e a sua abertura morfológica.

Depois de encontradas e refinadas as linhas, são escolhidos os pontos extremos para gerar um polígono representativo do espaço navegável da estrada. Os pontos pertencentes a esse poligono são publicados numa mensagem ROS.

Figura 3.11: Polígono representativo do espaço navegável criado a partir das linhas da estrada.

3.2 Algoritmo “Lane Tracker”

Este algoritmo teve como base um exemplo da “toolbox” de visão artificial do *Matlab* . Este exemplo deteta as linhas numa sequência de vídeo e notifica o condutor caso este mude de faixa de rodagem. Este exemplo demonstra a deteção de linhas com base na transformada de *Hough*.

Abordagem

Neste algoritmo, a imagem é dividida em duas regiões, a superior e a inferior, e esta separação é feita por um valor fixo pré-determinado. Somente a parte inferior é usada para fazer a procura de linhas. Essa imagem é filtrada e seguidamente binarizada onde é aplicada a transformada de Hough para detetar as linhas da estrada. As linhas encontradas são guardadas e serão usadas para comparação no ”frame”seguinte.

Neste algoritmo existem três funções principais, que tem como input as coordenadas dos pontos extremos das linhas:

Algoritmo 2 Localização das linhas

```
for $i = 1$ to $EndFrame$ do
    $lanematching()$
    $departuredetect()$
    $colorandtype()$
end for
```

1. lanematching(): Esta função localiza as linhas da estrada. Nesta função são calculadas as distâncias entre as linhas encontradas no “frame” atual e às guardadas, e são escolhidas as que melhor correspondem ás guardadas. Caso a linha guardada corresponda a linha encontrada, esta é adicionada ao repositório e o número de linhas guardadas é incrementado.
2. departuredetect(): Esta função deteta a mudança de faixa. Para tal é procurada a linha que interseta o limite inferior da imagem da esquerda para direita e aquela que se encontra mais próxima do centro do limite inferior da imagem. Seguidamente, é calculada a distância da linha ao centro do limite inferior da imagem caso essa distância seja inferior a um valor pré-estabelecido.
3. colorandtype(): Esta função deteta a cor da linha amarelo/branco e se é tracejada/continua. São encontrados os pontos extremos das linhas com os quais é gerada uma região de interesse onde são procuradas as cores amarela e branca, usando o espaço de cor YCbCr.

As linhas encontradas são desenhadas na imagem com a indicação da cor e do tipo de linha. Caso exista uma mudança de faixa esta é indicada ao utilizador assim como o lado para que esta mudança ocorre.

3.2.1 Implementação

Na implementação deste algoritmo foram usadas duas bibliotecas: OpenCV e Eigen, onde a Eigen é uma biblioteca de álgebra linear.

Para que a procura das linhas seja o mais exata e autónoma possível, foi usada uma região de procura dinâmica. Para que a região de procura das linhas seja dinâmica, foi usado um algoritmo de crescimento de regiões já implementado no OpenCV para separar a imagem em estrada e o ambiente em redor: o *watershed*. Para este procedimento é necessário definir as sementes das regiões de procura (fig. 3.12). A separação das regiões é feita com base na média em y dos pontos pertencentes a linha de separação das regiões (fig. 3.13).

Figura 3.12: Segmentação de regiões usando o *watershed*.

Figura 3.13: Separação das regiões obtidas.

Depois de encontradas as linhas, é criado um polígono com os respetivos pontos extremos; esse polígono representa o espaço navegável da estrada (fig. 3.14). Estes postos são posteriormente publicados numa mensagem ROS.

Figura 3.14: Poligono representativo do espaço navegável da estrada.

3.2.2 Avaliação do desempenho

Para ser possível avaliar os resultados obtidos, foi necessário gerar o “groud-truth” do “dataset” utilizado, para tal foi criada uma ferramenta em *Matlab* capaz de registar o “groud-truth” gerado manualmente para cada ”frame”individual. Este é representado por um polígono definido manualmente (fig. 4.10).

Figura 3.15: Esquerda: Trajeto percorrido para geração do “groud-truth”. Direita: Um exemplo do “groud-truth” gerado.

A avaliação quantitativa dos métodos utilizados é feita num dataset de 300 ”frames”com base no método desenvolvido por J. M. Alvarez [9]. Para tal, foram classificadas as diferentes regiões das imagem (fig. 3.16).

Para se obterem as regiões descritas anteriormente são avaliados todos os pixels individualmente da imagem gerada pelo algoritmo e comparados com a imagem do ground-truth. Desta avaliação resultam então as 4 regiões:

- True Negative: Caso o pixel seja classificado como não pertencente a estrada e não pertença a esta.

- False Negative: Caso o pixel seja classificado como não pertencente a estrada mas pertença a esta.

		Ground truth	
		Não estrada	Estrada
Resultado do algoritmo sobre cada pixel	Não estrada	True Negative (TN)	False Negative (FN)
	Estrada	False Positive (FP)	True positive (TP)

Figura 3.16: Medidas usadas para o calculo do desempenho. Exemplo de caso concreto (frame nº145).

- False Positive: Caso o pixel seja classificado como estrada e não pertença a esta.
- True Positive: Caso o pixel seja classificado como estrada e pertença a esta.

Com base neste método foram calculadas 3 medidas de desempenho: qualidade $\hat{g}$, rácio de deteção DR, precisão da deteção DA (table 3.1).

Medida ponderada do pixel	**Definição**
Qualidade	$\hat{g} = \frac{TP}{TP+FP+FN}$
Rácio de deteção	$DR = \frac{TP}{TP+FP}$
Precisão da deteção	$DA = \frac{TP}{TP+FN}$

Tabela 3.1: Medidas de desempenho usadas para avaliar a deteção no frame (adaptado de [9]).

Para definir a validade dos resultados foi usado o VRI (Valid Road Index); quando uma dada percentagem (K) da estrada é corretamente classificada, o resultado é valido (VRI=1), caso contrário é invalido (VRI=0). Este método é utilizado para reduzir o erro inerente á segmentação manual das imagens para gerar o “groud-truth”.

$$VRI = \begin{cases} 0 & K < \frac{TP}{Ground-truth} \\ 1 & K > \frac{TP}{Ground-thruth} \end{cases} \tag{3.8}$$

3.2.3 Estimativa da distância

O uso de uma única câmara não proporciona um método direto de medição de profundidade, porque devido ao efeito de perspetiva não existe uma linearidade entre a posição de um objeto na imagem e a sua posição no plano da estrada. Para compensar este efeito recorreu-se à imagem de perspetiva invertida, onde o efeito de perspetiva é corrigido. Desta forma, é possível fazer o mapeamento de todos os pixels da imagem de perspetiva invertida para o plano da estrada de uma forma linear (fig. 3.17).

O mapeamento pode ser expresso por:

$$(\mu, \nu, 1)^t = KTR(x, y, z, 1)^t \tag{3.9}$$

onde R é a matriz rotação:

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

T a matriz translação

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & -\frac{h}{\sin\theta} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

e K a matriz com os parâmetros da câmara:

$$K = \begin{pmatrix} f * k_\mu & 0 & \mu_0 & 0 \\ 0 & f * k_\nu & \nu_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.12}$$

onde f é a distancia focal da câmara e $(k_\mu * k_\nu)$ o rácio de pixels.

A eq. 3.9 também pode ser expressa por:

$$\begin{pmatrix} \mu \\ \nu \\ 1 \end{pmatrix} = \begin{pmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} \\ p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} \tag{3.13}$$

Como só estamos interessados no plano do chão ($Y_w = 0$):

$$\begin{pmatrix} \mu \\ \nu \\ 1 \end{pmatrix} = \begin{pmatrix} p_{1,1} & p_{1,3} & p_{1,4} \\ p_{2,1} & p_{2,3} & p_{2,4} \\ p_{3,1} & p_{3,3} & p_{3,4} \end{pmatrix} \begin{pmatrix} X_w \\ Z_w \\ 1 \end{pmatrix} \tag{3.14}$$

Utilizando esta equação é possível fazer o mapeamento de todos os pixel da imagem de perspetiva invertida.

Figura 3.17: Referenciais usados para o cálculo da distância.

Da transformação de coordenadas da imagem para coordenadas mundo, obtemos a distância relativa da estrada á câmara. É então feita uma transformação para o referencial do carro como se pode observar na fig.3.18.

Figura 3.18: Grafo de transformações utilizadas neste método.

Capítulo 4

Resultados e Discussão

Para avaliar o desempenho do software implementado serão descritas neste capítulo medições e comparações entre estes, bem como enumerar algumas situações de falhas observadas. Pretende-se também estabelecer qual a fiabilidade dos métodos e quais as suas limitações. Para tal foi efetuada uma análise quantitativa aos dois modelos, e realizaram-se algumas experiências ao longo de um percurso na cidade de Aveiro (fig. 4.1).

Figura 4.1: Trajeto percorrido para geração do "groud-truth".

Para se poder recriar as mesmas condições usou-se uma ferramenta disponibilizada pelo ROS para acumulação de mensagens em disco ao longo do tempo, podendo assim efetuar uma análise comparativa entre os métodos utilizados. Desta forma, o sistema operativo ROS apresenta um papel fundamental na obtenção de resultados, já que permite a avaliação dos métodos abordados exatamente nas mesmas condições.

Para cada algoritmo implementado foram variados os valores de alguns parâmetros que possuem significado aparente mais relevante. No próximo conjunto de gráficos é possível observar a variação destes parâmetros e o seu impacto no comportamento dos algoritmos. O valor ótimo do parâmetro será aquele que maximiza a deteção da estrada, para tal foi usado o VRI(Valid Road Index), ou seja, o resultado é útil (VRI=1) quando pelo menos 80% dos pixels da estrada são identificados corretamente, caso contrário os resultados são descartados (VRI=0).

4.1 Algoritmo “Real Time Detection of Lane Markers in Urban Streets”

Para este algoritmo foram testados vários parâmetros, vários dos quais não apresentaram resultados interessantes, sendo que alguns destes são parâmetros próprios do tipo de imagem (ex: tamanho da janela de análise). Os parâmetros escolhidos foram, ”DetectionThreshold”, ”GroupThreshold”, ”KernelWith”, ”KernelHeight”, ”LineWith”, ”lineHeight”and ”LineGap”.

- *DetectionThreshold*: Este parâmetro define o *threshold* utilizado para a binarização da imagem de perspetiva invertida. Binarização é um processo de segmentação de imagens baseado na diferença dos níveis de cinza que compõe diferentes objetos de uma imagem. A partir de um limiar estabelecido de acordo com as características dos objetos a isolar, a imagem pode ser segmentada em dois grupos, com níveis de cinza abaixo e acima do limiar.

Parâmetros

DetectionThreshold	Variável
GroupThreshold	10
KernelWith	2
KernelHeight	2
LineWith	2000
lineHeight	300

Figura 4.2: Variação do parâmetro ”detectionthresold”para diferentes definições de VRI.

- *GroupThreshold*: Este parâmetro define o número de linhas a agrupar na imagem de perspetiva invertida.

Parâmetros	
DetectionThreshold	30
GroupThreshold	Variável
KernelWith	2
KernelHeight	2
LineWith	2000
lineHeight	300

Figura 4.3: Variação do parâmetro "groupThreshold"para diferentes definições de VRI

- *KernelWith* and *KernelHeight*: Estes parâmetro definem respetivamente a largura e altura dos valores usado no filtro gaussiano da imagem de perspetiva invertida.

Parâmetros	
DetectionThreshold	30
GroupThreshold	10
KernelWith	Variável
KernelHeight	Variável
LineWith	2000
lineHeight	300

Figura 4.4: Variação do parâmetro ”KernelWith”e ”KernelHeight”para diferentes definições de VRI

- *LineHeight* and *LineWith*: Estes parâmetros definem respetivamente a largura e altura máxima das linhas na imagem de perspetiva invertida.

Parâmetros	
DetectionThreshold	30
GroupThreshold	10
KernelWith	2
KernelHeight	2
LineWith	Variável
lineHeight	Variável

Figura 4.5: Variação do parâmetro "lineheight"e "linewidth"para diferentes definições de VRI

4.2 Algoritmo “Lane Tracker”

Para este algoritmo foram testados vários parâmetros dos quais ”LineGap”, ”MinVote”e ”TrackThreshold”foram os que apresentaram resultados mais interessantes.

- *MaxLineGap*: Este parâmetro define o intervalo máximo entre dois pontos para estes poderem ser considerados pertencentes a mesma linha.

Parâmetros	
LineGap	Variável
MinVote	30
TrackThreshold	100

Figura 4.6: Variação do parâmetro ”MaxLineGap”para diferentes definições de VRI.

- *MinVote*: Este parâmetro define o número mínimo de votos para definir uma linha.

Parâmetros	
LineGap	170
MinVote	Variável
TrackThreshold	100

Figura 4.7: Variação do parâmetro "MinVote"para diferentes definições de VRI.

- *TrackThreshold*: Este parâmetro define o limiar de binarização da imagem das linhas da estrada.

Parâmetros	
LineGap	170
MinVote	30
TrackThreshold	Variável

Figura 4.8: Variação do parâmetro ”Threshold”para diferentes definições de VRI.

4.3 Comparação entre os algoritmos

Com base nos gráficos apresentados, foi determinado um conjunto de valores dos parâmetros, sendo aqueles que maximizam o desempenho médio, usando assim o método anteriormente descrito para o quantificar [7],[9].

Tabela 4.1: Valores dos parâmetros usados para efetuar os testes aos algoritmos.

Parâmetros	
DetectionThreshold	30
GroupThreshold	10
KernelWith	2
KernelHeight	2
LineWith	500
lineHeight	150

(a) Parâmetros usados no algoritmo Real Time Detection of Lane Markers in Urban Streets

Parâmetros	
LineGap	170
MinVote	20
TrackThreshold	100

(b) Parâmetros usados no algoritmo Lane Tracker

Foram ainda testadas combinações de valores maximizantes para os parâmetros definidos, para verificar a existência de dependência entre parâmetros. Contudo estes revelaram-se independentes.

Os resultados apresentados na tabela 4.2 demonstram o desempenho dos algoritmos para a deteção das linhas de marcação da estrada para os parâmetros definidos na tabela 4.1.

Tabela 4.2: Desempenho dos diferentes algoritmos para um mesmo dataset.

	Dataset		
	$\hat{g}$	DA	DR
Real Time Detection of Lane Markers in Urban Streets	0.892 ± 0.07	0.899 ± 0.08	0.992 ± 0.01
Lane Tracker	0.868 ± 0.08	0.886 ± 0.03	0.979 ± 0.03
Lane Tracker[a]	0.835 ± 0.07	0.982 ± 0.03	0.848 ± 0.03

---

[a] Utilizando a segmentação de regiões ”watershed”

Nas figuras seguintes são apresentados os resultados dos algoritmos implementados para cada frame individual assim como a estimativa da distância do ponto mais distante da linha encontrada ao carro.

E possível observar que em ambos os casos existem casos em que a estrada não é bem identificada e como, por exemplo, o carro que segue em sentido contrário (frame 149) não é bem identificado. Contudo em casos de curva é visível que o algoritmo Real Time Detection of Lane Markers in Urban Streets é mais preciso (Frame 35 e 260).

Nos casos em que o algoritmo não seja capaz de identificar qualquer tipo de linha será publicado o resultado do frame anterior, pois em ambos os casos a distância de deteção das linhas é superior á distância entre eixos do veiculo (fig. 4.11)

Figura 4.11: Distância entre eixos AtlasCar.

Uma vez que o algoritmo trabalha a uma frequência de 50hz, e o carro viaja a uma velocidade máxima de 50km/h em estradas urbanas, o carro percorre cerca de 0.2m entre cada análise.

Capítulo 5

Conclusões e trabalho futuro

5.1 Conclusões

O processo de separação da imagem em duas regiões permitiu melhorar a qualidade dos resultados obtidos pelo algoritmo “Lane Tracker”. Assim, foi bem sucedida a implementação de uma metodologia de crescimento de regiões que se demonstrou uma forma interessante de proceder à separação da imagem em regiões. Contudo o algoritmo que apresenta melhor desempenho é o “Real Time Detection in Urban Streets”, este apresenta uma descrição mais exata das linhas e uma taxa de sucesso superior relativamente aos métodos apresentados. A escolha dos parâmetros e a calibração da câmara revelaram-se de fato um passo importante para a deteção das marcações da estrada melhorando significativamente a qualidade dos resultados obtidos. Assim, foi bem sucedida a implementação de dois algoritmos de deteção de delimitadores de estrada que se demonstraram uma forma interessante de proceder à segmentação, especialmente em ambientes urbanos.

O trabalho efetuado nesta dissertação de Mestrado não se encontra isolado do enquadramento de aplicações em sistemas de apoio à condução. Com a aplicação conjunta de algoritmos de seguimento de obstáculos aplicados a lasers, é possível proceder ao planeamento de uma trajetória. Desta forma, é importante o estudo e a continuação do desenvolvimento de sistemas de visão artificial, não só para o uso de sistemas de apoio á condução, mas também para aplicações industriais. A aplicação atual de sistemas de visão enfrenta o problema da fiabilidade total. Infelizmente, apesar dos resultados obtidos serem bastante satisfatórios, a aplicação de metodologias de apoio a condução não são totalmente fiáveis, sobretudo em condições muito variáveis, é ainda um problema por resolver. Assim sendo, o trabalho aqui desenvolvido não apresenta uma conclusão final, mas demonstra aquilo que deve ser continuado a nível de sistemas de visão, as áreas e situações que devem ser melhoradas.

5.2 Trabalho futuro

Com a realização deste trabalho pode-se concluir que muito se tem efetuado na área da visão artificial. Esta é uma área muito vasta, com inúmeras aplicações e soluções.

Alguns dos caminhos que podem ser explorados no seguimento deste trabalho serão a deteção das faixas de rodagem, a deteção da topologia das marcações de estrada e com esta informação seria possível saber o sentido de rodagem das faixas.

Um dos maiores desafios da deteção de estrada usando visão, são as condições de luz variáveis em ambientes de estrada, como por exemplo durante a noite onde não existem condições de luz suficientes para que esta solução seja viável. Para ultrapassar este problema poderia ser usada uma câmera térmica, pois durante a noite a estrada continua a ter uma assinatura térmica distinta do ambiente que a rodeia.

Referências

[1] *OpenCV documentation http://docs.opencv.org.* Accessed: jul,2014.

[2] *Projeto atlas http://atlas.web.ua.pt*, (2014).

[3] N. Ahmed Salim, X. Cheng, and X. Degui, *A robust approach for road detection with shadow detection removal technique*, Information Technology Journal, (2014).

[4] C.-L. Chen and C.-L. Tai, *Adaptive fuzzy color segmentation with neural network for road detections*, Engineering Applications of Artificial Intelligence, 23 (2010), pp. 400–410.

[5] D. C.Moore, J. Leonard, and S. Teller, *Perception and navigation for the DARPA urban challenge*, CSAIL, (2007).

[6] C.Rasmussen, *Grouping dominant orientations for ill-structured road following.*, IEEE, (2004).

[7] H. Ding, B. Zou, K. Guo, and C. Chen, *Comparison of several lane marking line recognition methods*, in 2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP), June 2013, pp. 53–58.

[8] H.Kong, *General road detection from a single image*, IEEE Transactions on Image Processing, VOL. 19, (8, AUGUST 2010).

[9] J. M. Alvarez, T. Gevers, and A. M. Lopez, *3d scene priors for road detection*, IEEE, (2010).

[10] A. Mohamed, *Real time detection of lane markers in urban streets*, IEEE Intelligent Vehicles Symposium, (2008).

[11] M.Oliveira and V.Santos, *Real time extraction of road border lines using simple statistical descriptors*, 2nd Workshop on Planning,Perception and Navigation for Intelligent Vehicles, (2008).

[12] M. Oliveira, V. Santos, and A. D. Sappa, *Short term path planning using a multiple hypothesis evaluation approach for an automous driving competition*, IEEE 4th Workshop on Planning,Perception and Navigation for Intelligent Vehicle, (2011).

[13] M. Oliveira, V. Santos, and A. Sappa, *Multimodal inverse perspective mapping.*, 2014.

[14] M. Oliveira, P. Stein, J. Almeida, and V. Santos, *Modular scalable architecture for the navigation of the atlas autonomous robots.*, 9th Conference on Autonomous Robot Systems and Competitions, (2009).

[15] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, and R. Wheeler, *Ros: an open-source robot operating system*, International Conference on Robotics and Automation (ICRA), (2009).

[16] V. Santos, J. Almeida, E. Ávila, D. Gameiro, M. Oliveira, R. Pascoal, R. Sabino, and P. Stein, *Atlascar - technologies for a computer assisted driving system on board a common automobile.*, IEEE, (2010).

[17] S. Surkutlawar and R. Kulkarni, *Shadow supression using rgb and hsv color space in moving object detection.*, J.Adv. Comput. Sci., (2013).