Universidade do Minho
Escola de Engenharia

José Carlos da Costa Gonçalves

Visão Artificial em Condução Autónoma com Câmara Kinect

Visão Artificial em Condução Autónoma com Câmara Kinect
José Carlos da Costa Gonçalves
UMinho | 2013

outubro de 2013

Universidade do Minho
Escola de Engenharia

José Carlos da Costa Gonçalves

Visão Artificial em Condução Autónoma com Câmara Kinect

Tese de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de
Mestre em Engenharia Eletrónica Industrial e de Computadores

Trabalho efetuado sob a orientação do
Professor Doutor Agostinho Gil Teixeira Lopes

outubro de 2013

DECLARAÇÃO

Nome: José Carlos da Costa Gonçalves

Endereço eletrónico: a52720@alunos.uminho.pt Telefone: 919771236 / 253583190

Número do Bilhete de Identidade: 13575436

Título dissertação: Visão Artificial em Condução Autónoma com Câmara Kinect.

Orientador: Doutor Agostinho Gil Teixeira Lopes

Ano de conclusão: 2013

Designação do Mestrado: Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores – Ramo Automação, Controlo e Robótica

Nos exemplares das teses de doutoramento ou de mestrado ou de outros trabalhos entregues para prestação de provas públicas nas universidades ou outros estabelecimentos de ensino, e dos quais é obrigatoriamente enviado um exemplar para depósito legal na Biblioteca Nacional e, pelo menos outro para a biblioteca da universidade respetiva, deve constar uma das seguintes declarações:

DE ACORDO COM A LEGISLAÇÃO EM VIGOR, NÃO É PERMITIDA A REPRODUÇÃO DE QUALQUER PARTE DESTA DISSERTAÇÃO;

Universidade do Minho, ___/___/_____

Assinatura: _____

Não existem métodos fáceis para resolver problemas difíceis.

René Descartes

Agradecimentos

A realização desta dissertação envolveu vários intervenientes, desde Fevereiro até Outubro deste ano. A eles será dedicado este trabalhado e, mais propriamente, este capítulo.

Em primeiro lugar, ao docente, Professor Gil Lopes por ter aceitado esta dissertação, por ter compreendido algumas questões fora do âmbito da dissertação e, consequentemente, pela disponibilidade prestada. Ao Professor Fernando Ribeiro pela simpatia e à-vontade com que recebe os alunos no Laboratório de Automação e Robótica.

Seguidamente, impossível esquecer os colegas de curso com os quais foi partilhado o espaço de trabalho e, mais importante ainda, a troca de conhecimentos e entreajuda. A eles, Daniel, Ricardo e Aníbal.

Ao pessoal das oficinas, nomeadamente o Sr. Joel, o Sr. Carlos e a D. Ângela pelo tempo dispensado e pela sempre total disponibilidade.

Impossível esquecer, neste percurso, o meu avô que, infelizmente não conseguiu presenciar o momento final de um curso onde foi também um pilar.

À namorada, sempre compreensiva, sempre a lutar lado-a-lado. Aos amigos que, em vários momentos, foram deixados para segundo plano.

Por fim, o maior agradecimento às pessoas que possibilitaram tudo. As quase duas décadas a estudar, sempre alicerçadas no seu apoio. A eles. Os pais.

Resumo

O desenvolvimento de sistemas autónomos móveis que possam servir de base para o mercado de trabalho é cada vez mais um objeto de estudo e investigação. Os veículos autónomos poderão diminuir os custos de transportes de mercadorias, aumentar os níveis de produção e retirar a presença humana em ambientes perigosos.

Nesse sentido, é descrito, nesta dissertação, um sistema capaz de se movimentar autonomamente, através da junção do carro robótico Fórmula UM TD2, do Grupo de Controlo, Automação e Robótica da Universidade do Minho, e de uma câmara Kinect, da Microsoft.

O objetivo principal passa por, através da utilização de Técnicas de Visão por Computador, desenvolver um algoritmo capaz de reconhecer objetos, como semáforos, reconhecer lugares livres em zonas de estacionamento e determinar qual lugar ocupar e, finalmente, planear a sua trajetória, definindo à priori o caminho mais adequado. Para tal serão utilizadas várias técnicas de deteção de caraterísticas e aproveitadas as potencialidades da câmara *Kinect*.

Todos os algoritmos apresentados nesta dissertação foram desenvolvidos e simulados no software QT Creator, com base na linguagem de programação C++ e recorrendo a bibliotecas como OpenCV, Boost, PCL e OMPL e à framework OpenNI.

Palavras-chave: Visão Artificial, Processamento de Imagem, Kinect, Condução Autónoma, *OpenCV*, *Path Planning*, *SURF*

Abstract

The development of Autonomous Mobile Systems that can serve as the basis for the labor market is increasingly an object of study and research. The autonomous vehicles could reduce the cost of transport of materials, increase production levels and remove the human presence in hazardous environments.

In this sense, it is described in this dissertation, a system capable of moving autonomously through the joint robotic car Formula UM TD2, property of Control, Automation and Robotics Group at University of Minho and a Microsoft Kinect camera.

The main objective is, through the use of Computer Vision techniques, develop an algorithm capable of recognizing semaphores, identify a Parking zone and determine which place to take and, finally, plan its trajectory defining, a priori, the most appropriate path. For such things is used the capabilities of Kinect camera.

All the algorithm presented in this thesis was developed and simulated in QT Creator software, based on the programming language C++ and using libraries like OpenCV, Boost, PCL and OMPL and OpenNI framework.

Keywords: Artificial Vision, Image Processing, Kinect, Autonomous Driving, *OpenCV*, Path Planning, *SURF*

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

2D – 2 Dimensões

3D – 3 Dimensões

CMOS - Complementary Metal-Oxide Semiconductor

DARPA - Defense Advanced Research Projects Agency

FNR – Festival Nacional de Robótica

Fps – Frames per Second

IDE - Integrated Development Environment

INSECT - Information Network Social Electric City Transporter

LiPo – Lithium Polymer

LTS - Long Term Support

OMPL - Open Motion Planning Library

OS – Operative System

PC – Personal Computer

RANSAC - RANdom SAmple Consensus

RGB – Red Green Blue

RRT – Rapidly-Exploring Random Tree

SURF – Speeded-Up Robust Features

TTT - Tourism Think Thank

UM – Universidade do Minho

USB – Universal Serial Bus

VW - Volkswagen

Capítulo 1 - Introdução

Com a leitura deste capítulo é possível perceber as motivações deste trabalho de dissertação, Visão Artificial em Condução Autónoma com Câmara Kinect, e os seus objetivos. São também descritos neste primeiro capítulo, a estrutura e organização deste documento.

Condução Autónoma é a capacidade de reproduzir a complexa tarefa encarregue a um condutor no interior de um veículo, adquirindo dados sensoriais sobre o que o rodeia e tomando decisões baseadas na análise desses dados. É uma área de interesse emergente, especialmente no mundo da robótica e da indústria automóvel. Desde que surgiu o primeiro veículo autónomo no final da década de 70, a sociedade imagina como será uma viagem de carro sem a necessidade de o ser Humano conduzir o automóvel pelas estradas. Num modo geral, um sistema de condução autónoma tem como objetivo navegar de um ponto para outro sem qualquer intervenção de um operador humano, beneficiando com isso qualquer pessoa que gostaria de realizar uma viagem sem a preocupação de conduzir e também as empresas que poderiam lucrar com o facto de não ser necessária intervenção humana. Esta área de interesse é largamente explorada tanto a nível académico como industrial e está em constante expansão.

Para que os robôs operem em ambientes não estruturados têm de ser capazes de percecionar o espaço desse ambiente. Nesse sentido, o mercado providencia tecnologias de sensores 3D que podem modificar totalmente a forma como os projetos atuais estão a ser desenvolvidos. Por exemplo, o sensor Kinect da Microsoft faculta variadíssimas funcionalidades. É possível implementar algoritmos que controlam, como por exemplo, informação da profundidade, constroem nuvens de pontos em tempo real apenas com imagens provenientes da câmara associada. Pode-se esperar que a maioria dos projetos robóticos possa percecionar, futuramente, o mundo em 3D. O tema apresentado nesta dissertação pretende ser uma forma de manipular a informação capturada pela Kinect por forma a reconhecer um sistema de semáforos, estacionar o veículo autonomamente e, de acordo com o espaço onde o veículo está inserido, determinar um caminho ótimo para um local final escolhido.

Este conjunto de soluções poderá trazer benefícios para uma sociedade em constante desenvolvimento e competitividade. Os veículos autónomos poderão diminuir os custos de transportes de mercadorias, aumentar os níveis de produção e retirar a presença humana em ambientes perigosos.

1.1 Motivações

O desenvolvimento de sistemas autónomos móveis que possam servir de base para o mercado de trabalho é cada vez mais um objeto de estudo e investigação. Em termos aplicacionais, os veículos autónomos poderão diminuir os custos de logística e aumentar os níveis de produção [**1**], além da vertente competitiva, como a participação em provas e festivais robóticos. A grande utilidade, desenvolvimento e expansão futura tornam este projeto motivador e a possibilidade de utilização do mesmo em situações reais adicionam interesse ao trabalho descrito nesta dissertação.

A nível pessoal, este projeto torna-se motivador devido a três fatores essenciais, nomeadamente a programação, a utilização prática do veículo autónomo Fórmula TD2 e o desenvolvimento de uma aplicação com base em Visão por Computador. Esta dissertação permitirá fortalecer as competências adquiridas em cinco anos académicos, bem como adquirir novos conhecimentos na área de Controlo, Automação e Robótica.

1.2 Objetivos

Este projeto de dissertação tem como grande objetivo o desenvolvimento de um algoritmo com base em técnicas de visão por computador para um veículo robótico, Fórmula UM TD2, dotando-o da capacidade de se movimentar autonomamente.

Assim, com a informação de profundidade capturada pela câmara *Kinect*. será possível que o sistema planeie a sua trajetória num determinado espaço, detetando e evitando obstáculos, desde o ponto inicial ao final, definindo a trajetória mais adequada.

Além disto, é também estipulado que o sistema detete um conjunto de sinais (semáforos) e aja de acordo com o semáforo identificado. Finalmente, será implementado um algoritmo capaz de dotar o sistema da capacidade de efetuar estacionamento com ou sem obstáculos na zona pretendida.

1.3 Estrutura da Dissertação

O documento está dividido em sete capítulos. Além deste, onde é elaborada uma introdução ao projeto a desenvolver, no segundo capítulo é apresentado um Estado da Arte onde se aborda uma visão dos produtos existentes no mercado, são enumeradas algumas competições de veículos autónomos, bem como o veículo Fórmula UM TD2 e outros veículos já desenvolvidos.

Além disso é realizado o estudo/análise às tecnologias a usar no desenvolvimento do sistema, e fundamentação teórica para a implementação da dissertação.

O desenvolvimento da solução é apresentado no terceiro capítulo, onde são detalhadas as ferramentas utilizadas no desenvolvimento dos algoritmos do projeto, o protocolo de comunicação e são apresentados os próprios algoritmos de aquisição e processamento de imagem, reconhecimento de semáforos, estacionamento e planeamento da trajetória.

No quarto capítulo é demonstrado o funcionamento do sistema de condução autónoma, através de testes à aplicação. São também apresentados e analisados os resultados obtidos, sendo que no capítulo 5 são apresentadas as conclusões finais desta dissertação bem como algumas perspetivas futuras.

O último capítulo contém toda a bibliografia que serviu de base para a realização deste trabalho de dissertação, nomeadamente *websites*, artigos, livros e todos os documentos relevantes.

Capítulo 2 - Estado da Arte

Este capítulo descreve o estado da arte dos elementos que constituem esta dissertação, nomeadamente, uma visão geral sobre os sistemas já existentes e algumas competições na área da condução autónoma. É descrito o veículo autónomo a utilizar nesta dissertação e outros veículos existentes com importância neste mercado. Finalmente, são identificados os fundamentos teóricos que darão suporte à realização da dissertação

2.1 Condução Autónoma

Quando se fala em condução autónoma, surge na mente do ser humano comum uma questão. O que será o futuro dos automóveis na sociedade? Será que a tecnologia pouco ou nada desenvolvida nesta área será melhorada?

De facto, o desenvolvimento de veículos autónomos tem evoluído bastante. Os primeiros passos nesse sentido implicavam uma combinação complexa entre *hardware* e *software*, com sistemas de respostas lentas, inadaptados a ambientes desconhecidos [**2**]. No entanto, a contribuição da investigação científica nas últimas décadas é sem dúvida um enorme fator de impulso para a descoberta de soluções adequadas às necessidades tanto de pessoas como de empresas.

O primeiro veículo autónomo apareceu no Japão em 1977, há mais de trinta anos. Ainda se vivia a mudança de Regime em Portugal e este sistema era já desenvolvido no departamento de Engenharia Mecânica da *Universidade de Tsukuba* [**3**]. Como é expectável, este veículo era bastante rudimentar, conseguindo apenas seguir linhas brancas como as de uma estrada convencional a uma velocidade máxima de 30 km/h. Desde então a investigação e desenvolvimento de plataformas capazes de se orientarem autonomamente não mais parou.

Em 1980, Ernst Dickmanns e a sua equipa desenvolveram, na *Universidade Bundeswehr* em Munique uma carrinha *Mercedes-Benz* que, com técnicas de visão artificial, fez com que o veículo atingisse os 100 km/h numa rua sem trânsito [**3**]. Depois da realização desta experiência, a Comissão Europeia patrocinou o projeto *EUREKA Prometheus* com aproximadamente 800 milhões de euros, tornando-o no maior projeto de I&D feito na área [**4**].

No mesmo ano, nos EUA, a DARPA desenvolveu o *Autonomous Land Vehicle* com a capacidade de ser controlado sem intervenção humana a uma velocidade de 30 km/h, usando radares laser, visão artificial e um mecanismo robótico de controlo [**5**].

Em 1994, dois veículos autónomos desenvolvidos pela Daimler-Benz e por Ernst Dickmanns seguiram por uma autoestrada de Paris por mais de 1000kms de forma semiautónoma, com trânsito intenso, atingindo velocidade máxima de 130 km/h [**6**]. Estes veículos demonstraram ainda ser possível a condução em faixas de rodagem livres, condução em coluna, mudanças de faixa e ultrapassagens a outros veículos. Tudo isto de forma autónoma.

Um ano depois, uma *Mercedes-Benz Classe S* modificada por Ernst Dickmanns [**4**], realizou uma viagem de 1600kms entre Munique e Copenhaga (ida e volta), usando uma técnica que simula o comportamento de um olho ao procurar por pontos de interesse numa imagem, retirando informação mais detalhada desses pontos e um conjunto de microprocessadores, de forma a obter um controlo em tempo real. Durante este percurso, o veículo atingiu velocidades superiores a 175 km/h na autoestrada, conduzindo em condições normais de tráfego e executando manobras de ultrapassagem.

Ainda em 1995, o projeto *Navlab* da *Universidade Carnegie Mellon* nos Estados Unidos da América, mostrou que foi possível uma condução autónoma ao longo de um percurso com 5000kms numa viajem intitulada "*No hands across America*" [**7**]. O veículo recorria a redes neuronais para controlar automaticamente a direção e a potência, sendo que o sistema de travagem era controlado por humanos.

Em 2001, o exército dos EUA realizou um projeto denominado *DEMO III* [**8**] onde veículos autónomos evitaram obstáculos em terrenos irregulares. O sistema de controlo designado de *Real-Time Control System* foi elaborado por James Albus do *National Institute of Standards and Technology*.

No final de 2011 a Universidade de Aveiro apresentou o *Atlascar* [**9**], um veículo autónomo, desenvolvido no seu Departamento de Engenharia Mecânica, que consegue seguir o movimento de peões e prever as suas ações de forma a evitar atropelamentos.

2.2 Competições com Veículos Autónomos

2.2.1 DARPA Grand Challenge

Em 2004 ocorreu a primeira edição de uma das mais prestigiadas competições a nível mundial de veículos autónomos, o *DARPA Grand Challenge* [**10**], competição criada para incentivar a investigação e o desenvolvimento em veículos autónomos. Este certame foi criado pela organização de investigação do departamento de defesa dos Estados Unidos.

De forma a obter inscrições de equipas, o Congresso dos Estados Unidos autorizou a *DARPA* a atribuir um prémio de um milhão de dólares ao vencedor da competição. Na primeira edição nenhuma equipa conseguiu terminar o percurso de 240km no deserto de Mojave, onde o veículo melhor classificado percorreu apenas 12km.

No ano seguinte, realizou-se a segunda edição da *DARPA Grand Challenge* e os resultados superaram os do ano anterior, onde cinco veículos terminaram a prova. A equipa vencedora foi a *Stanford Racing Team* da *Universidade de Stanford* com um *VW Touareg* [**11**], na **Figura 1**, que percorreu a totalidade do percurso em 6h54min.

Figura 1 - Veículo Autónomo VW Touareg Vencedor DARPA Grand Challenge 2005 [11**]**

A terceira edição ocorreu em 2007 onde a equipa *Tartan Racing* [**12**] da *Universidade Carnegie Mellon da Pennsylvania* sagrou-se vencedora completando a prova num tempo de 4h10min, com um *Cheby Tahoe*.

Figura 2 - Veículo Autónomo Cheby Vencedor DARPA Grand Challenge 2007 [12]

2.3 Festival Nacional de Robótica

O Festival Nacional de Robótica é um festival tecnológico que reúne, desde 2001 [**13**], investigadores, docentes e estudantes portugueses na área da Robótica.

O Festival, que decorre anualmente, promove demonstrações e competições de robots em diversas provas e categorias e, dá espaço à apresentação de trabalhos de estudantes e investigadores nacionais.

Uma das competições existentes no Festival é a Prova de Condução Autónoma [**14**]. Esta prova, que se realiza desde a primeira edição é direcionada essencialmente a equipas provenientes de universidades, onde o objetivo passa por apresentar um robô autónomo móvel que conclua com sucesso um percurso ao longo de uma pista fechada [**15**].

A pista, similar a uma estrada convencional, tem a forma de um 8 e é delimitada por duas linhas brancas, contendo ainda um par de semáforos e um túnel sobre uma das curvas, de forma a simular uma condução o mais real possível.

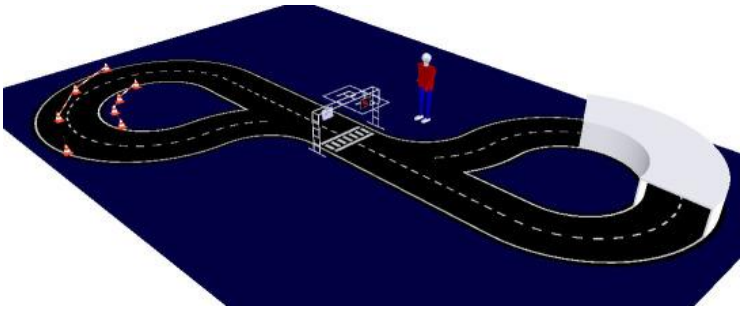

Figura 3 - Pista Fechada da Prova de Condução Autónoma [15]

Numa fase inicial, o objetivo do trabalho era a participação no Festival Nacional de Robótica na Prova atrás descrita. Com o decorrer do trabalho, em decisão conjunta com o orientador, foi tomada a decisão de não participar no Festival e abordar de forma diferente este projeto.

2.4 Veículo Autónomo - Fórmula UM TD2

Para a realização desta dissertação, foi utilizado um veículo robótico já existente no Departamento de Automação e Robótica da Universidade do Minho, o Fórmula UM TD2, ilustrado na **Figura 4**.

Figura 4 – Veículo utilizado – Fórmula UM TD2

O Fórmula UM TD2 foi desenvolvido por alunos do Mestrado em Engenharia Eletrónica em anos letivos anteriores, tendo sido já utilizado no Festival Nacional de Robótica 2012. Tem uma dimensão de 60 cm de comprimento e 36 cm de largura e um suporte para a câmara Kinect e para o computador na parte cimeira. A **Tabela 1** apresenta estes e outros dados importantes do veículo.

Tabela 1 - Dados do Fórmula UM TD2

Microcontrolador	Arduino ATMega
Comunicação	Porta Série
Velocidade Máxima	10m/s
Encoder	Sim
Alimentação	Bateria LiPo 11,1V 5000mAh
Alimentação Dedicada Kinect	Sim
Travão	Sim
Comprimento * Largura	60cm * 36cm
Altura	20cm
Altura Máxima (c/Kinect)	40cm

A comunicação entre o computador e o veículo é efetuada via porta-série através do envio de tramas ou conjunto de caracteres, detalhados em **3.2.2**. Para alimentar as placas e os motores foi utilizada uma Bateria *Zippy 20C Series* de 3 células, com 11,1V e 5000mAh que alimenta igualmente a Kinect, com um conetor dedicado à câmara. A conversão de tensão DC-DC está a cargo do Step Down *Chuangruifa Car Power DC DC*. O *encoder* encontra-se na parte traseira do veículo, junto do eixo das rodas. Os sistemas de direção e de travagem são constituídos por Servomotores DF15 Metal Gear, com rolamento duplo e tensão de operação entre 4,8V e 7,2V.

2.5 Outros Veículos Autónomos

2.5.1 Fórmula UM TD1

O Departamento de Automação e Robótica da Universidade do Minho disponibiliza também o veículo Fórmula UM TD1. Este veículo, também desenvolvido por alunos do Mestrado em Engenharia Eletrónica serviu de base a um projeto de dissertação chamado "Desenvolvimento de uma Plataforma Móvel para Condução Autónoma" tendo participado no Festival Nacional de Robótica 2012, em Guimarães.

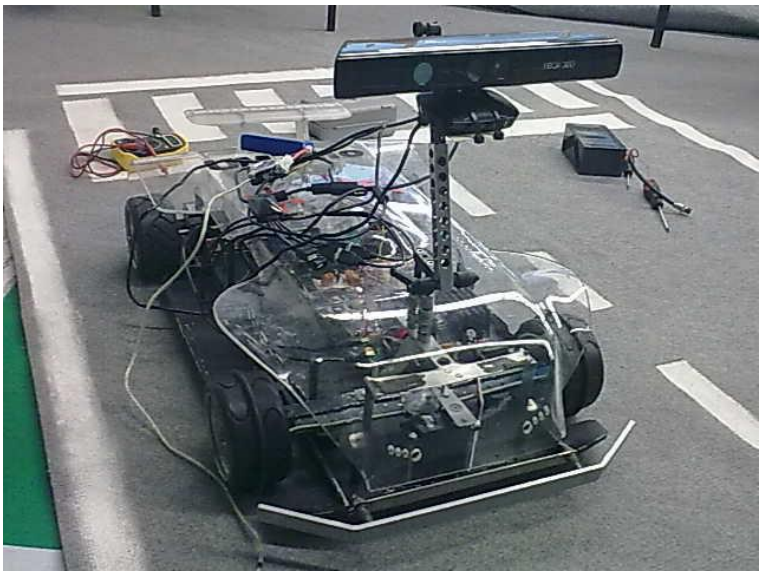

Figura 5 - Fórmula UM TD1

Tendo obtido bons resultados na competição, o Fórmula UM TD1 tem alguns dados de fábrica similares ao Fórmula UM TD2, tendo no entanto maiores dimensões. A **Tabela 2** mostra alguns dos parâmetros mais importantes do veículo.

Tabela 2 - Dados do Fórmula UM TD1

Microcontrolador	Arduino Uno
Comunicação	Porta Série
Encoder	Não
Alimentação	2 Baterias LiPo 14,8V 5000mAh
Travão	Sim
Comprimento * Largura	77cm * 50cm
Altura Máxima (c/Kinect)	60cm

2.6 Kinect

Em Novembro de 2010, a Microsoft lançou no mercado a câmara Kinect. Este dispositivo é um sensor de movimentos desenvolvido para as consolas XBOX 360 e XBOX One [**16**].

Figura 6 - Microsoft Kinect [16**]**

A câmara Kinect é constituída por uma câmara RGB, um sensor de profundidade, um microfone e um motor de inclinação. Mas o que faz da Kinect tão especial? A sua aplicabilidade.
A utilização da Kinect enquadra-se não só nos jogos para consola, como em variadíssimas aplicações em sistemas robóticos, mapeamentos 3D, aplicações orientadas à saúde e reabilitação, entre outros. O dispositivo permite uma leitura extremamente realista do ambiente onde se encontra, extraindo de forma muito precisa as caraterísticas de um objeto [**17**].

Outra particularidade é o facto de ser possível a qualquer pessoa desenvolver aplicações, através de tutoriais e *software* fornecidos pela Microsoft. Esta abertura faz com que a utilização da Kinect atinja um enorme número de utilizadores e permita uma investigação científica mais elaborada.

2.6.1 Câmara RGB

A Kinect possui, como referido atrás, duas câmaras distintas de perceção do espaço que a rodeia, uma RGB e uma por infravermelhos. A câmara RGB retorna uma matriz com três canais de cores por pixel (vermelho, verde e azul), encontrando-se separada da câmara de infravermelhos por um distância fixa, que através de transformações entre os dois padrões de informação, torna possível obter uma correspondência direta de cor e profundidade de um dado objeto. A câmara RGB apresenta uma resolução de 640 x 480 pixels e uma taxa de amostragem de 30fps.

2.6.2 Sensor de Profundidade

O sensor de profundidade da Kinect é constituído por um emissor de raios infravermelhos e um sensor CMOS monocromático. O emissor de infravermelhos projeta radiação infravermelha sob a forma de pontos em todas as superfícies que se encontram perante a câmara, criando assim uma nuvem de pontos. Ao embater numa superfície, se existir, o raio infravermelho é refletido e captado pelo sensor CMOS. O tempo entre a emissão e a receção permitirá calcular a distância ao objeto e, por conseguinte, determinar se o lugar se encontra disponível. Esse procedimento é chamado de Princípio da Triangulação e é representado na **Figura 7**.

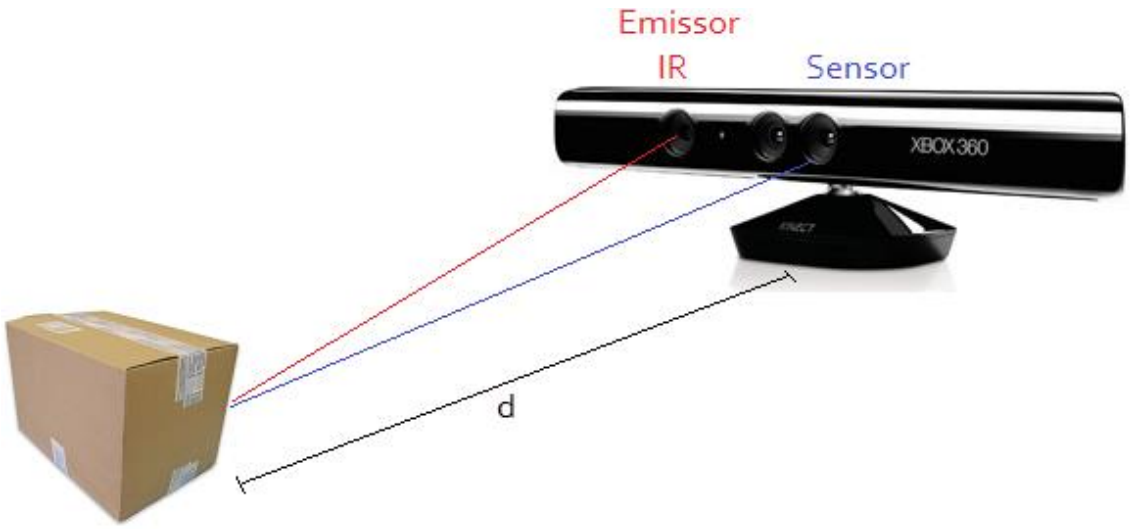

Figura 7 - Princípio da Triangulação na Kinect, adaptado de [18]

2.6.3 Microfone

Além das câmaras para reconhecimento de imagem, a Kinect tem embutido um conjunto de 4 microfones, que permite captar sons mais próximos do sensor, conseguindo diferenciar os ruídos externos. Com esta tecnologia, a Kinect abstrai-se de barulhos de fundo que, por conseguinte, não atrapalham o funcionamento da Kinect. O microfone é capaz de detetar várias pessoas diferentes, através das suas vozes, num determinado espaço.

2.6.4 Motor de Inclinação

A Kinect possui um motor de inclinação acoplado à sua base. Este elemento possibilita que a câmara execute movimentos para cima e para baixo (Pitch) entre -30° e 30° em relação ao solo.

Figura 8 - Kinect com inclinação de -30º em relação ao solo

Figura 9 - Kinect a 0º

Figura 10 - Kinect com inclinação de +30º em relação ao solo

Esta possibilidade de inclinação permite otimizar a captura e adapta a Kinect às aplicações em que está inserida. Por exemplo, nesta aplicação, se forem conjugados o algoritmo de reconhecimento de semáforos e, posteriormente, o planeamento da trajetória, a Kinect deverá inclinar-se para baixo depois de reconhecer o sinal, de forma a capturar o espaço mais abaixo possível e calcular um caminho ótimo até ao ponto desejado.

2.6.5 Outras Aplicações com Sensor Kinect

No mercado, para além da utilização da Kinect em jogos para a consola XBOX, existem inúmeras aplicações onde a câmara da Microsoft tem um papel essencial e inovador. Em **2.6.5.1**, **2.6.5.2**, **2.6.5.3** e **2.6.5.4** são descritas algumas das mais interessantes aplicações onde a Kinect está integrada.

2.6.5.1 Itinerários Interativos com Kinect e Google Street View

No âmbito do painel Tourism Think Thank (TTT) [**19**], com o tema "Tendências do TIC no Turismo: da distribuição online à inovação do produto, a GFI Multinacional desenvolveu uma aplicação orientada ao turismo, que possibilita a navegação em itinerários interativos que interagem com o corpo humano e simulam percursos reais. Esta aplicação conta ainda com um sistema de incorporação sonora do ambiente da zona, de forma a tornar ainda mais real a experiência, fazendo com que seja possível passear virtualmente pelas ruas históricas de grandes cidades e interagir com a informação dos pontos de interesse, através de vídeos e fotografias do local.

Figura 11 - Aplicação Orientada ao Turismo com Kinect [20**]**

2.6.5.2 Smart INSECT by Toyota

Em 2012, a fabricante de automóveis japonesa Toyota apresentou um protótipo do veículo INSECT (Information Network Social Electric City Transporter). A particularidade deste carro é o sistema Kinect incorporado na parte frontal acima do para-brisas, que permite ao utilizador ser reconhecido, quer por visualização facial, quer corporal. Desenvolvido para ser um automóvel inteligente na sua relação com o utilizador, o INSECT permite abrir as portas com um simples gesto e prevê uma programação que permite memorizar e antecipar as escolhas do condutor [**21**].

Figura 12 - Toyota Smart INSECT com Kinect [21**]**

2.6.5.3 Kinect usado na fisioterapia de veteranos de guerra

Num trabalho conjunto entre a Microsoft e a Força Aérea dos Estados Unidos, está a ser desenvolvido um *software* em associação com a Kinect com o objetivo de auxiliar os veteranos de guerra e os soldados feridos. Os cenários incluem funções relacionadas à fisioterapia, mas também abrangem situações de simulação de treino. Este projeto surge com o intuito de reduzir os custos com este tipo de tratamento [**22**].

2.6.5.4 IllumiRoom

O IllumiRoom é a nova tecnologia da Microsoft capaz de, através do Kinect e de um projetor vídeo, expandir o ambiente de jogo para além do ecrã de televisão. Este conceito tem como objetivo transformar a experiência de jogo recorrendo à criação de efeitos capazes de permitir combinar os mundos virtuais com o real.

Figura 13 - Illumiroom - Ambiente de jogo expandido [23**]**

A tecnologia IllumiRoom utiliza a Kinect por forma a digitalizar a geometria da divisão onde está inserido o jogador e adapta o espaço através da projeção para fora da televisão, tornando o ambiente um complemento da ação que se passa no jogo [**23**].

2.7 Revisão Bibliográfica

Depois de identificados os constituintes físicos do sistema, torna-se necessário estudar os fundamentos teóricos que darão suporte à realização da dissertação. Como já referido em nota introdutória, o sistema fará uso de técnicas de deteção de características de imagens/objetos bem como de técnicas de planeamento de trajetória. O estudo destas valências permitirá não só perceber o projeto como também iniciar o desenvolvimento dos algoritmos para aquisição e processamento de imagem.

2.7.1 Técnica de Deteção de Caraterísticas

Um dos tópicos deste trabalho consiste em dotar o sistema com a capacidade de reconhecimento de imagens e/ou objetos. Para tal, existem várias técnicas de deteção das caraterísticas das imagens a comparar. Esse processo é assente na deteção de pontos de interesse através da extração de características. O processo de procura de pontos de interesse pode ser dividido em dois passos, nomeadamente *detector* e *descriptor*. O primeiro é a deteção de um *keypoint* ou ponto-chave, que identifica uma área de interesse e o segundo é a descrição do *keypoint* que caracteriza a região onde ele está inserido. Em circunstâncias normais é identificada uma região com elevada variação de intensidade, como um canto ou uma aresta com o seu centro a ser designado de ponto-chave. A sua descrição é geralmente determinada, através da orientação dos seus pontos mais próximos, dando origem a um vetor de características que identifica um dado ponto-chave. Finalmente, poderá ser efetuada a combinação (*matcher*) entre imagens ou objetos. Este processo é baseado na distância entre os vetores de caraterísticas que identificam um ponto-chave.

Para o desenvolvimento deste projeto, foi estudado de forma aprofundada a técnica SURF (Speeded Up Robust Features), descrita de seguida.

2.7.1.1 SURF (Speeded Up Robust Features) [24]

A tarefa de encontrar pontos semelhantes entre duas imagens ou objetos está presente em várias aplicações de Visão por Computador, como por exemplo, reconhecimento de objetos, calibração de câmaras, entre outros.

O SURF é uma das técnicas mais utilizadas no que diz respeito a essas aplicações. É um método baseado numa junção de algoritmos já desenvolvidos. Entre eles, encontram-se alguns ainda da década de 80, como por exemplo o método Harris Corner Detector, que apesar de ser bastante inovador, apenas detetava objetos numa única escala. No entanto, com o avançar da tecnologia, essa deficiência foi resolvida com a utilização do determinante da matriz Hessiana e o seu Laplaciano. O método foi sendo redefinido até que Mikolajczyk e Schmid aumentaram a sua robustez, usando o determinante da matriz Hessiana para a localização do objeto, e o Laplaciano para a escala. Depois de vários melhoramentos das técnicas de deteção de caraterísticas, o SURF viu aumentar a velocidade de processamento e a precisão na busca de *keypoints*. Como descrito em **2.7.1**, a procura de *keypoints* é feita em dois passos, *detector* e *descriptor*.

2.7.1.2 Deteção de Keypoints

No SURF, o processo de deteção de *keypoints* é baseado na matriz Hessiana, devido à sua elevada precisão. Basicamente, são detetados, na imagem ou objeto, pontos onde o determinante da matriz é máximo. Supondo um ponto $\mathrm{x} = (x, y)$ numa determinada imagem, a matriz Hessiana $H(\mathrm{x}, \sigma)$ será definida em x a uma escala σ da seguinte forma:

$$H(\mathrm{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathrm{x}, \sigma) & L_{xy}(\mathrm{x}, \sigma) \\ L_{xy}(\mathrm{x}, \sigma) & L_{yy}(\mathrm{x}, \sigma) \end{bmatrix}$$

Onde $L_{xx}(\mathrm{x}, \sigma)$ representa a convolução da derivada gaussiana de segunda ordem $\frac{\partial^2}{\partial x^2} g(\sigma)$ com a imagem num ponto x, e similarmente para $L_{xy}(\mathrm{x}, \sigma)$ e $L_{yy}(\mathrm{x}, \sigma)$.

Então, para decidir se um pixel da imagem é um *keypoint*, é feita uma aproximação à Matriz Hessiana, através das derivadas parciais das intensidades dos *pixels* da imagem. A base do método SURF é a deteção dos máximos e mínimos do determinante da matriz Hessiana que

definem quão robusto é cada *pixel* e se se trata ou não de um ponto de interesse. Mas surge um problema. A convolução é extremamente dispendiosa e as derivadas de Gauss de segunda ordem, necessitam de ser discretizadas antes da utilização propriamente dita. Dado que o esforço computacional aumenta com o aumento do tamanho dos filtros, o SURF simplifica o processo derivativo com uma filtragem da imagem, dividindo-a em retângulos.

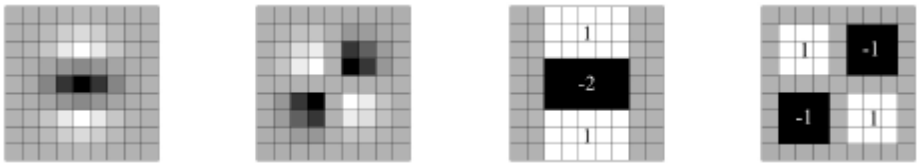

Figura 14 - Sequência de procedimentos no SURF na Deteção de Keypoints [23]

Através da **Figura 14**, é possível perceber a sequência de ações utilizadas pelo SURF. Primeiramente a máscara é discretizada e cortada da derivada parcial de segunda ordem em $\boldsymbol{y}, (\boldsymbol{L_{yy}})$ e $\boldsymbol{xy}, (\boldsymbol{L_{xy}})$, respetivamente. Depois é feita a aproximação à derivada parcial gaussiana de segunda ordem em $\boldsymbol{y}, (\boldsymbol{D_{yy}})$ e na direção $\boldsymbol{xy}, (\boldsymbol{D_{xy}})$. As regiões a cinzento correspondem a zero.

Ora, utilizando estes filtros num dado ponto $\mathrm{x} = (x, y, \sigma)$, o determinante da Matriz é igual a:

$$\det(H_{aprox}) = \boldsymbol{D_{xx}D_{yy}} - (0.9\boldsymbol{D_{xy}})^{\boldsymbol{2}}$$

$$\sigma = (\, tamanho\ do\ filtro/9) \times 1.2$$

Com isto, já há condições para o cálculo do determinante da matriz Hessiana para cada *pixel*.

O algoritmo de busca de *keypoints* deve permitir a obtenção em diferentes escalas da imagem original. A alteração de escalas dá pelo nome de espaço de escala e é implementada sob a forma de pirâmide invertida, de acordo com a **Figura 15**.

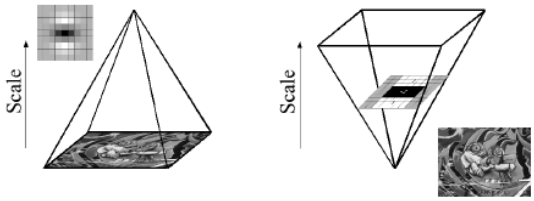

Figura 15 - Espaço de Escala em forma de pirâmide invertida [24**]**

Assim, o SURF obtém pontos de interesse para diferentes escalas através do aumento do filtro da imagem. Em vez de iterativamente reduzir o tamanho da imagem (esquerda), o uso de imagens integrais permite a expansão do filtro a um custo constante.

Outra das particularidades da deteção de keypoints está na sua localização. Para encontrar os pontos de interesse na imagem e em diferentes escalas, são determinados os máximos e mínimos no mapa de resposta, que é formado através do determinante da matriz Hessiana para cada um dos pixéis.

Finalmente, a deteção termina com a determinação da orientação dos *keypoints*. Para cada *keypoint* detetado é calculada a resposta da convolução para os dois filtros. Estes filtros têm a designação de *Haar wavelets* e através deles é possível observar as variações que estão a ocorrer nas direções x e y, como se constata pela observação da **Figura 16**.

Figura 16 - Filtros Haar Wavelet na direção x e y, respetivamente [24**]**

É calculada para cada ponto, a resposta ao filtro Haar wavelet, na direção x e y dentro de uma região circular à volta do *keypoint*. Cada ponto é ponderado, baseado na sua distância ao *keypoint*, através de uma função Gaussiana. Seguidamente é estimada a orientação dominante, somando todas as respostas em x e y. As respostas dentro de cada segmento são somadas, e formam um novo vetor sendo o vetor mais longo a orientação do *keypoint,* como ilustrado a seguir.

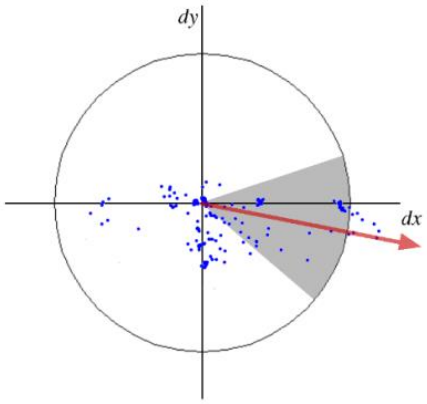

Figura 17 - Orientação dos Keypoints [24**]**

2.7.1.3 Descrição

A descrição é a fase mais importante para que a deteção de *keypoints* se processe de forma correta. Esta etapa consiste em pegar num determinado ponto de interesse e a partir do seu centro implementar um quadrado, como se verifica na **Figura 18**.

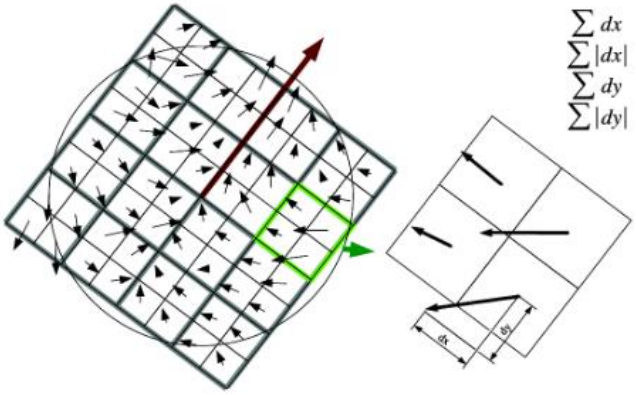

Figura 18 - Descrição dos Keypoints no SURF – Quadrado de Wavelet [24**]**

Ora, para descrever os *keypoints*, é criada uma janela à volta do mesmo e calculadas as respostas ao filtro *Haar wavelet*. As subdivisões correspondem às somas das derivadas.

2.7.2 RANSAC - Transformação entre Imagens

O método de transformação entre imagens utilizado no algoritmo de reconhecimento de semáforos será o RANSAC (RANdom SAmple Consensus). Esta técnica consiste num método de estimação de parâmetros de um modelo matemático e foi apresentado no início da década de 80 por Fischler e Bolles. O RANSAC tem como principal objetivo eliminar os *outliers* de um conjunto de dados, baseando-se na ideia de que num grupo de amostras selecionadas aleatoriamente, existem algumas que preenchem corretamente um modelo matemático enquanto outras amostras desse mesmo conjunto não obedecem ao modelo. Assim, é o papel do RANSAC determinar quais as amostras que serão consideradas ou não adequadas ao modelo.

Primeiramente é selecionado um conjunto de pontos aleatoriamente sendo calculado um modelo. Para cada um dos pontos é verificado se faz parte do modelo. A cada iteração, é verificado se o modelo é considerado válido. Se sim, esse ponto é adicionado como *inlier* e o modelo é recalculado a partir dos mais recentes *inliers*. Imaginando que o modelo atual é melhor que o modelo até então, o modelo e *inliers* atuais tornam-se nos respetivos melhores. Por fim, o algoritmo retorna os melhores *inliers* e melhor modelo, assim que terminar o número de iterações estabelecidas. No final do processo, o modelo atual será considerado como o modelo referência, mesmo que existam amostras que não correspondam ao modelo escolhido , os tais *outliers*. O ciclo reinicia e são repetidas todas as instruções até que todos os grupos tenham sido testados.

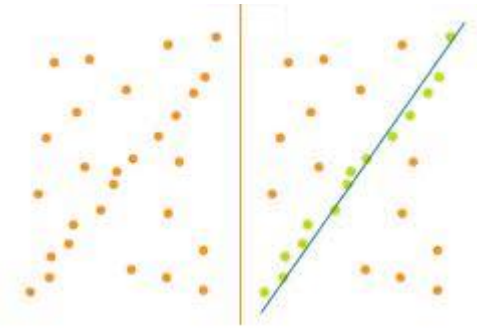

Figura 19 - Exemplo de uma iteração do método RANSAC [25**]**

2.7.3 Técnicas de Planeamento da Trajetória

Quando se fala em planear algo, fala-se na possibilidade de perceber a realidade, avaliar os caminhos. O planeamento é tão só uma construção de um referencial futuro com avaliações precisas de todo o processo associado à situação a planear. É, dessa forma, o lado racional da ação e pode dizer-se que é um processo de deliberação que escolhe e organiza ações, antecipando os resultados esperados. Esta deliberação procura alcançar, da melhor forma possível, objetivos definidos à-priori [**26**].

Para esta dissertação, deverá ser implementado um algoritmo que planeie a trajetória do veículo autónomo, que consistirá numa sequência de movimentos que levarão o robô de uma posição inicial a uma posição final. O planeamento de trajetória permite que um caminho seja planeado previamente evitando que o robô se mova em direção à posição final de forma aleatória. Essa sequência de movimentos formará, idealmente, um caminho onde o custo de cada movimento seja mínimo desde a posição inicial à posição final. Assim, um algoritmo de planeamento de trajetória é considerado completo se conseguir sempre determinar um trajeto, num tempo finito.

Em suma, foram estudados dois algoritmos de planeamento, nomeadamente o A* e o RRT. Ambos estão implementados na biblioteca OMPL [**27**], abordada no capítulo das Ferramentas Utilizadas, em **3.1.3.2**.

2.7.3.1 A* (A-Star)

O algoritmo A* é largamente usado em aplicações de planeamento de trajetória. Assumindo que se pretende ir do ponto A para ponto B, com um obstáculo entre os pontos, e com espaço limitado. O papel do A*, e de qualquer algoritmo de planeamento, passa por encontrar o melhor caminho entre dois pontos no plano. Atente-se na **Figura 20.**

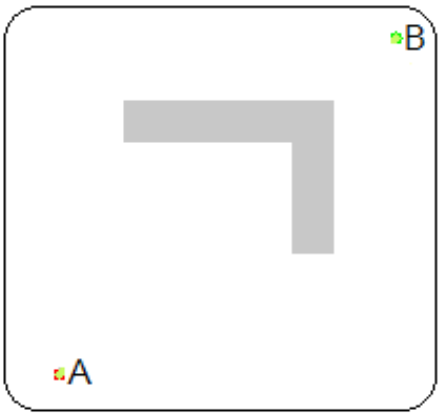

Figura 20 - Obstáculo entre Ponto Inicial A e Ponto Final B [28**]**

A busca pelo melhor caminho entre os pontos A e B é feita começando por reduzir a área de procura, dividindo o espaço onde se encontra o robô, neste caso o veículo, em nós. Depois, o algoritmo A* utiliza a mínima distância entre dois nós, obtida através de uma função heurística. Esta função retorna a menor distância entre duas posições dadas ou, neste caso, entre dois nós. Assim, o caminho ótimo será determinado encontrando quais os nós a seguir para ir do ponto A ao ponto B. Uma vez achado o caminho, o veículo mover-se-á de nó a nó até que o ponto final seja alcançado [**29**].

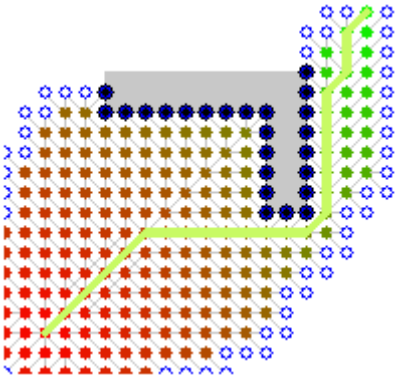

Figura 21 - Caminho determinado pelo A* [28**]**

Para determinar o caminho de forma mais eficiente, como mostrado na **Figura 21**, o A* utiliza a distância do trajeto desde a fonte mais a mínima distância heurística até ao destino para determinar a ordem de pesquisa de potenciais nós. Essa estimativa é dada pela seguinte equação:

$$f(n) = \mathrm{g(n)} + h(n)$$

Da equação acima, lê-se que a soma da (distância) do trajeto desde a fonte ao nó atual g(n) com a mínima distância heurística desde o nó atual até ao destino h(n) é igual à estimativa de custo da melhor solução f(n).

Este método apresenta como vantagens o facto de o valor de f(n) representar a menor distância possível através de um dado nó, servindo de base a uma boa estimativa da distancia total do percurso. Além disso, com a utilização da heurística, consegue-se desprezar caminhos desnecessários, comparando a menor distância possível de um trajeto através de um dado nó, com comprimentos de trajetos calculados anteriormente.

Por outro lado, o algoritmo A* demonstra não ser o ideal se o caminho por si só não seja suficiente, sendo necessários algoritmos de controlo que movimentem o robô de nó para nó. Para efetuar o trajeto encontrado, torna-se necessário um segundo algoritmo para examinar o caminho e, por conseguinte, determinar as tramas necessárias para comunicar com os atuadores do robô, dado que o A* é um método estático, onde um espaço de configuração que representa a orientação e posição de um robô é suficiente. Mesmo assim, não existem garantias de que um conjunto apropriado de entradas de controlo seja encontrado.

2.7.3.2 RRT (Rapidly-Exploring Random Tree)

O presente trabalho de dissertação envolve não só o cálculo de um caminho ótimo, mas também uma constante perceção do espaço onde o veículo se encontra. Nesse sentido, surge o algoritmo RRT ou *Rapidly Exploring Random Tree.* Este algoritmo é designado por ser uma estrutura de dados aleatórios e foi desenvolvido para uma vasta gama de aplicações cujo foco seja o planeamento da trajetória. Então, esta estrutura de dados comporta todas as vantagens das técnicas de planeamento de trajetória, mas com uma particularidade. A RRT é desenvolvida especificamente para lidar com restrições não-holonómicas. Em robótica, um sistema é considerado não-holonómico se os graus de liberdade controláveis forem menores que o número total de graus de liberdade. Ora, o veículo Fórmula UM TD2, veículo não-holonómico, pois consegue alcançar uma qualquer posição e orientação no espaço 2D, necessitando assim de

três graus de liberdade. No entanto, apenas tem dois graus de liberdade controláveis, nomeadamente, movimento para a frente e o ângulo de rotação da direção.

Uma RRT é expandida iterativamente, aplicando entradas de controlo que conduzem o sistema suavemente até pontos escolhidos aleatoriamente.

Ao contrário do A*, onde um espaço de configuração que representa a orientação e posição de um robô é suficiente, o RRT adiciona a derivada em função do tempo de cada uma das dimensões do espaço de configuração. Desse modo, o RRT estende o espaço de configuração, adicionando restrições diferenciais como a velocidade ou ângulo de direção. Isto facilita o processo de planeamento da trajetória.

Assim, o espaço estendido é conhecido como espaço de estados, representado doravante por X. Por exemplo, para um carro como o Fórmula UM TD2, o espaço de configuração pode ser dado por $X = (x,y,\theta)$ enquanto o seu espaço de estados é igual a $X = (x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$, permitindo assim planear com restrições diferenciais. Nesse sentido, este método de planeamento da trajetória torna-se mais eficaz para sistemas como o Veículo de Condução Autónoma e será, portanto, o método utilizado nesta dissertação.

Passando à prática, atente-se na **Figura 22**. É representado um objeto do tipo de um carro, a azul e é desejado determinar o caminho ótimo entre o ponto A e o ponto B.

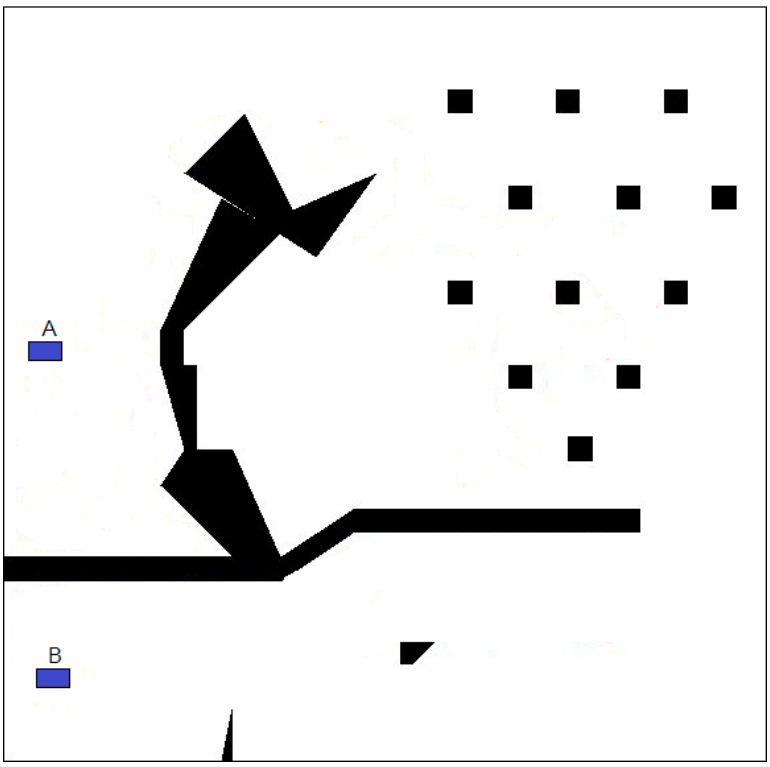

Figura 22 - Obstáculos entre Ponto Inicial A e Ponto Final B, adaptado de [30]

Assumindo uma região fixa de obstáculos, a negro, deve ser evitada, é construída uma RRT de maneira a que todos os seus vértices são estados pertencentes a um espaço livre e cada aresta da RRT corresponderá a um trajeto que se encontra totalmente no espaço.

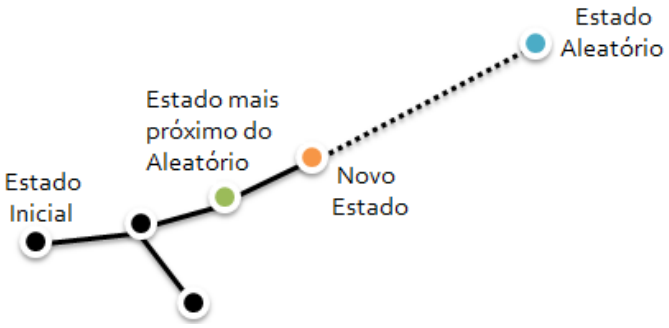

Figura 23 - Construção RRT, adaptado de [31]

A **Figura 23** é exemplificativa do processo de construção de uma RRT. Para um dado estado inicial, o primeiro vértice da RRT é exatamente esse ponto inicial. A cada iteração é gerado um Estado Aleatório. A partir do momento em que é gerado o primeiro estado aleatório, a RRT começará a expandir-se. Primeiro, é retornado o estado mais próximo do estado aleatório.

Seguidamente, serão selecionadas as entradas de controlo que conduzirão o estado mais próximo do aleatório, e integradas durante um intervalo de tempo, alcançando-se um novo estado no espaço de estados. Esta nova posição deve ser adicionada a uma determinada distância, desde o estado mais próximo do aleatório em direção ao estado aleatório. Se o trajeto até à nova posição estiver livre de obstáculos, então o novo estado é adicionado à RRT. E assim sucessivamente até ser atingido o ponto final.

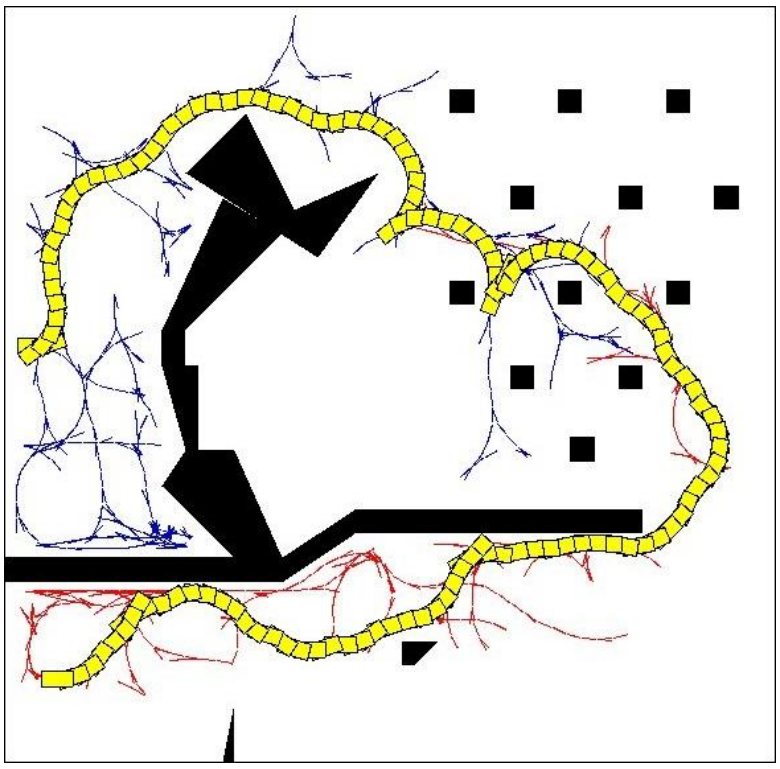

Figura 24 - RRT gerada e Caminho Ótimo Determinado [30**]**

A **Figura 24** ilustra a RRT gerada e, consequentemente o caminho ótimo a ser implementado pelo carro. Todo o processo de determinação da RRT terminará no momento em que a RRT estiver suficientemente perto do estado final, neste caso o Ponto B. Para executar o trajeto planeado deve-se recuar ao longo da árvore para identificar as arestas que conduzirão então o veículo desde o Ponto A ao Ponto B, enviando para o microcontrolador as entradas de controlo guardadas juntamente com as arestas identificadas.

Capítulo 3 - Aquisição e Processamento de Imagem

O capítulo 4 apresenta uma visão sobre os algoritmos desenvolvidos para o reconhecimento de semáforos, como seta de seguir em frente, virar à esquerda ou à direita e parar. Além disso, são explicadas as técnicas utilizadas para a manipulação e processamento das imagens provenientes da câmara bem como o planeamento da trajetória do veículo, que envolve uma transformação em polígonos dos obstáculos, um algoritmo para prever a interseção de polígonos e, com isso, a definição da trajetória propriamente dita.

3.1 Arquitetura de Software

3.1.1 Sistema Operativo

3.1.1.1 Linux Ubuntu 12.04 LTS

Todos os algoritmos desenvolvidos para o projeto de Condução Autónoma, tiveram como alicerce o sistema operativo Linux Ubuntu 12.04 LTS.

Figura 25 - Tux, mascote do Linux OS [32**]**

O Linux é um sistema operativo gratuito feito para que qualquer pessoa o possa utilizar de forma totalmente livre. Com constantes updates e upgrades, o Linux em nada fica a dever aos outros sistemas operativos, devido às suas interfaces gráficas cada vez mais amigáveis, sendo conhecido pela sua estabilidade e robustez [**26**].

Figura 26 - Logotipo Ubuntu [33**]**

O Ubuntu é hoje a distribuição Linux mais usada no mundo, atingindo o número de 20 milhões de utilizadores [**28**].

3.1.2 Plataformas de Desenvolvimento

No que diz respeito às plataformas de desenvolvimento dos algoritmos, recorreu-se a dois softwares distintos, o QT Creator e o Arduino IDE.

3.1.2.1 QT Creator

Os algoritmos de aquisição, comparação de imagens, reconhecimento de semáforos, estacionamento e planeamento da trajetória foram desenvolvidos sob a linguagem de programação C++ com recurso ao software Qt Creator 4.6 [**21**].

Figura 27 - Logotipo QT Creator [21]

O QT Creator tem a particularidade de ser um software disponível em vários sistemas operativos e ser totalmente gratuito.

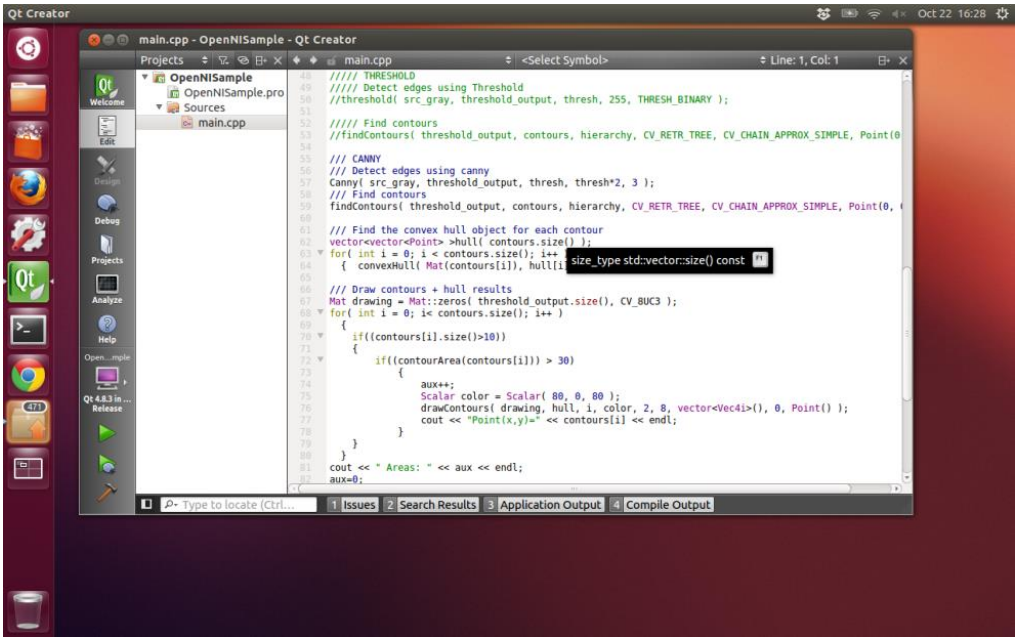

Figura 28 - Plataforma de Desenvolvimento QT

3.1.3 Bibliotecas

3.1.3.1 OpenCV

Para o desenvolvimento de aplicações na área de Visão por Computador, normalmente é necessário recorrer a uma biblioteca multiplataforma gratuita, mais concretamente, o OpenCV (Open Source Computer Vision Library).

Figura 29 - Logotipo OpenCV [31**]**

Esta biblioteca possui módulos de processamento de imagem que permitem fazer o reconhecimento de objetos, filtros de imagem e calibração de câmaras [**31**]. Foi através do OpenCV que foi implementada a técnica de deteção de objetos e o método RANSAC.

3.1.3.2 OMPL

De forma a concluir o objetivo de planear o caminho que o veículo deverá efetuar, foi utiziada a biblioteca OMPL (Open Motion Planning Library). Esta biblioteca é composta por muitos algoritmos de planeamento de movimento, entre os quais, aquele que foi utilizado nesta dissertação, o RRT.

Figura 30 - Representação OMPL [32**]**

3.1.4 Frameworks

3.1.4.1 OpenNI

De forma a interagir com a Kinect e iniciar a captura, recorreu-se à *framework* OpenNI (Open Natural Interface) que contém funções de alto nível orientadas para reconhecimento de gestos, de esqueleto humano, entre outras funções.

Figura 31 - Logotipo OpenNI [33**]**

O OpenNI em conjunto com o OpenCV, permite iniciar captura de vídeo da Kinect e, também, calibrar o sensor RGB com o sensor de infravermelhos, fazendo com que os dados RGB e de profundidade sejam coerentes e os indicados para uma correta implementação de algoritmos.

3.2 Algoritmos

3.2.1 Aquisição de Imagem

Qualquer que seja o algoritmo a desenvolver nesta dissertação, terá que ser iniciada a captura de imagem da câmara Kinect. O **Algoritmo 1** implementa a captura de imagem da câmara Kinect. Tanto para o reconhecimento de semáforos, como para o planeamento da trajetória, o algoritmo permitirá o inicio de aquisição de imagem.

Algoritmo 1 – Captura de Imagem com a câmara Kinect	
1	VARIÁVEIS: captura: VideoCapture, imagem: Mat;
2	Abrir captura (CV_CAP_OPENNI);
3	PARA (; ;) FAÇA
4	INÍCIO
5	Libertar imagem ( );
6	Reter captura ( );
7	Restaurar captura (imagem, CV_CAP_OPENNI_BGR_IMAGE);
8	FIM
9	FIM PARA;

No Algoritmo **1** as funções *Abrir Captura( )*, *Libertar Imagem( )*, *Reter Captura( )* e *Restaurar Captura( )* são parte integrante da biblioteca OpenCV, mais concretamente da categoria de Leitura e Escrita de Imagens e Vídeo. O construtor *VideoCapture* cria uma variável *captura* que, quando sujeita à função *Abrir Captura( )* abrirá um ficheiro de vídeo para captura. O parâmetro da função *Abrir Captura( )* é o ID do dispositivo de vídeo que, neste caso, terá que corresponder ao ID da Kinect. A partir da Linha 3, inicia-se um ciclo para capturar continuamente imagens através da Kinect. Declara-se uma variável *imagem* que, a cada iteração do ciclo receberá a captura através da função *Restaurar Captura( )*. Desta forma, a imagem será mostrada continuamente, como se deduz através da **Figura 32**.

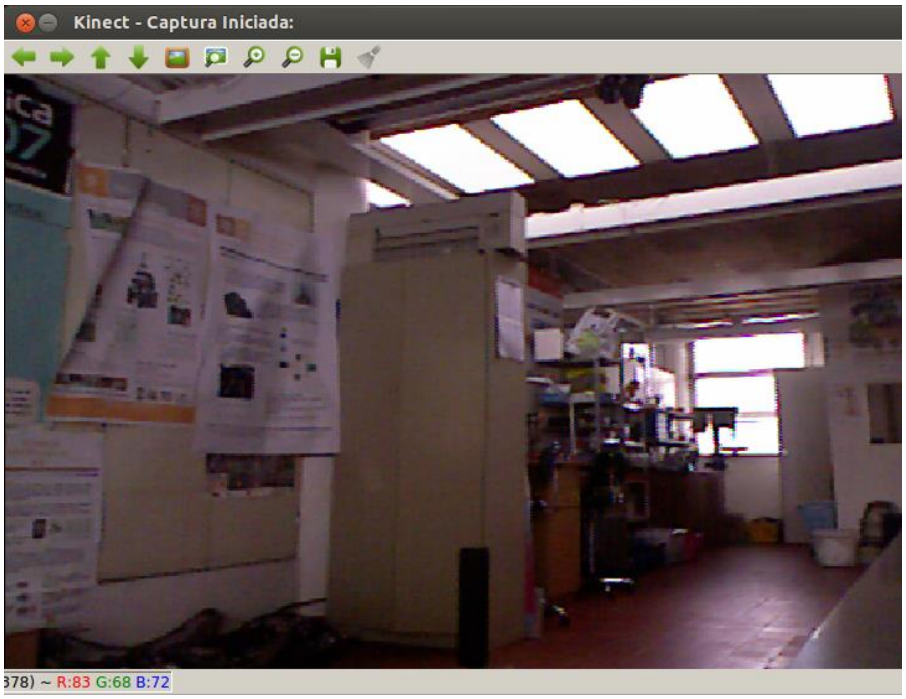

Figura 32 - Captura de Imagem Kinect

3.2.2 Comunicação PC – Veículo

A comunicação entre o computador e o veículo estabelece-se através do envio de tramas via porta-série para o microcontrolador. No caso do Fórmula UM TD2, o microcontrolador é o Arduino Mega 2560 e o envio de tramas implica o uso da biblioteca *Boost* que possibilita a comunicação via porta série. Dessa forma, os sinais serão processados pelo microcontrolador que acionará o motor da tração ou o motor da direção.

A comunicação resume-se a três tipos de tramas, como se pode ver de seguida.

Tabela 3 - Conjunto de Tramas para Comunicação

Trama	Legenda	Funcionalidade
[*]+[speedValue]	speedValue entre -100 e 100 e representa a velocidade - (unidade: decímetro/seg)	Acionar Motor de Tração
[#]+[angleValue]	angleValue entre -40 a 40 e representa o ângulo da roda (unidade: graus)	Ângulo da Direção
[*]+[0]	Bytes para Paragem	Imobiliza Motor de Tração

Ora, as tramas apresentadas na **Tabela 3** foram pensadas para o caso da utilização do veículo no Festival Nacional de Robótica, que envolve uma maior mudança de velocidade. No entanto, para a dissertação explicada neste documento, pretende-se uma velocidade baixa constante, variando apenas os ângulos da direção. Nesse sentido, foi elaborada uma alternativa que simplifica a comunicação.

Tabela 4 - Conjunto de Caracteres para Comunicação

Caracter	Legenda	Funcionalidade
'w'	Por cada 'w', incrementa a velocidade em 1m/s	Andar em Frente
's'	Por cada 's', decrementa a velocidade em 1m/s	Andar para Trás
'p'	Caracter para Paragem, velocidade 0	Imobiliza Motor de Tração
'd'	Por cada 'd', aumenta ângulo de direção À direita 5º	Ângulo da Direção
'a'	Por cada 'a', aumenta ângulo de direção À esquerda 5º	Ângulo da Direção

O envio de caracteres será processado pelo mesmo ambiente de desenvolvimento, neste caso o QT. Para tal, recorreu-se à biblioteca *Boost* que agrega um conjunto de funções que permitem o envio de tramas por porta-série, permite configuração de baud-rate e a escolha da porta à qual está conectada a placa microcontroladora.

3.2.3 Reconhecimento de Semáforos

Um dos objetivos para a realização desta tese passa por reconhecer diferentes sinais de trânsito. Neste caso, a sinalização será a utilizada no Festival Nacional de Robótica. Os semáforos a serem detetáveis estão representados num monitor como na **Figura 33**. Existem quatro sinais a reconhecer, representados na **Figura 34**. Cada um deles terá um impacto diferente na movimentação do veículo.

Figura 33 - Sinal virar à esquerda em monitor

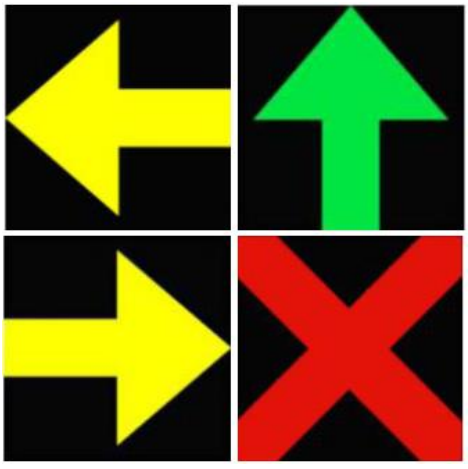

Figura 34 - Sinalização a Identificar (Setas Esquerda, Frente, Direita e Stop), adaptado de [38]

Depois de estudados o método *SURF*, em **2.7.1.1**, surge a necessidade de recorrer novamente ao *OpenCV* e às funções do *SURF* para elaborar um algoritmo capaz de detetar os *keypoints*. Dessa forma, a deteção de pontos de interesse é implementada como descrito no Algoritmo **2**.

Algoritmo 2 - SURF - Detector	
1	VARIÁVEIS: objeto: Mat, detecao: SurfFeatureDetector keypoints: vector,
2	object ← Ler Imagem("Ficheiro.jpg", 0);
3	detecao (HessianThresh);
4	Detetar detecao (objeto, keypoints);

Com a função *SurfFeatureDetector* é permitido criar uma variável *detecao* que, conjuntamente com a função *detect( )* (Linha 4) determinará os pontos-chave da imagem "Ficheiro.jpg", guardando-os no vetor *keypoints*. Na função *imread( )* do OpenCV (Linha 2), o segundo parâmetro define o tipo de cor com o qual a imagem será carregada. Se o parâmetro for maior que zero, a imagem será retornada em RGB. Quando o valor é igual a zero, é retornado uma imagem em escala de cinzentos e, finalmente, quando o parâmetro é menor que zero, a imagem é retornada exatamente com as cores de origem, com que está definida.

O parâmetro *HessianThresh* representa um *threshold* a partir do qual os pontos de maior intensidade serão considerados *keypoints*. Quanto maior o valor de *HessianThresh* menos pontos serão considerados chave, ao passo que quanto menor o valor de *HessianThresh*, mais pontos serão detetados. Normalmente o valor deste parâmetro encontra-se entre 400 e 800. O mesmo processo será realizado para deteção dos keypoints da imagem a receber através da Kinect.

Algoritmo 3 - SURF - Descriptor	
1	VARIÁVEIS: extrator: SurfDescriptorExtractor, descritor: Mat;
2	Computar extrator(objeto, keypoints, descritor);

Algoritmo 4 - SURF - Matcher	
1	VARIÁVEIS: matcher: FlannBasedMatcher descritor_Kinect: Mat;
2	matcher.knnMatch(descritor, descritor_Kinect, matches, 2);

Com os **Algoritmos 3** e **4**, fica concluído o método de deteção de caraterísticas. Primeiramente é declarada a variável *extractor* que servirá de base à extração dos *descriptors* do *objeto*. Na linha 2 do **Algoritmo 3**, a função *Computar extrator( )* tem como parâmetros a variável *objeto* que é a imagem de referência, *keypoints* que contém os *keypoints* da imagem *objeto* e, finalmente, a variável onde serão guardados os *descriptors*. É feito exatamente o mesmo processo para a imagem a adquirir da Kinect. Isso permitirá a comparação e respetiva correspondência, descrita no **Algoritmo 4**, entre as imagens (guardada e da Kinect) através da função *knnMatch( )*. Depois de comparadas a imagem inicial com a imagem recebida pela Kinect, torna-se necessário manipular a informação por forma ao veículo siga as indicações. Se estiver perante uma seta para a direita, terão que ser enviadas tramas para o veículo com ângulo de direção para a direita e, consequentemente, a velocidade para iniciar o movimento. No entanto, será necessário ter guardado as três diferentes setas? A solução seguinte simplifica esse processo. Basta uma seta como objeto de referência para conseguir identificar as setas para a esquerda e direita.

Atentando na **Figura 35**, é possível verificar que, se for usada a seta para seguir em frente como referências, as setas de virar à esquerda e à direita, não são mais do que uma seta igual à referência mas com uma rotação.

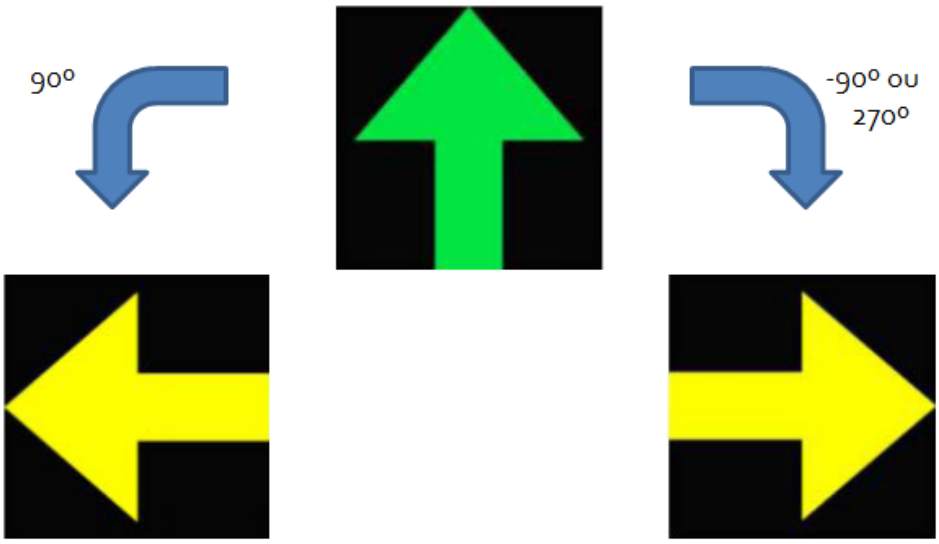

Figura 35 - Relação entre as setas

Depois da análise da **Figura 35**, depreende-se que a seta verde tem uma rotação de 0°. Para o este estudo, será tido em conta que o sentido contrário aos ponteiros do relógio. Assim, em álgebra linear, uma matriz de transformação é igual a:

$$H = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Onde θ será o ângulo de rotação. Ora, com o ângulo θ = 0, a matriz de rotação será:

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Como tal, será possível calcular as matrizes de rotação para as outras setas. Com uma rotação de 90° no sentido contrário aos ponteiros do relógio, a matriz de rotação será:

$$H = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Por outro lado, com a rotação de 270° no sentido contrário aos ponteiros do relógio a matriz de rotação será igual a:

$$H = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Depois de determinadas as matrizes, torna-se essencial implementar, através das funções do OpenCV, um algoritmo que calcule a matriz de transformação da imagem de referência, neste caso a seta em frente, e a imagem a ser captada pela Kinect. Através do Algoritmo **5** é possível identificar quais as funções utilizadas.

Algoritmo 5 – Determinação de Matriz de Transformação	
1	*VARIÁVEIS: H, objeto, imagem:Mat*
2	*H ←Encontrar Homografia(objeto, imagem, CV_RANSAC);*

É através da função *EncontrarHomografia( )* que se determina a matriz de transformação das imagens a comparar. Na Linha 2, é guardado na matriz *H* o resultado da função. Esta função compara as imagens *objeto* e *imagem* e determina a transformação entre elas. O método utilizado é o RANSAC, explicado em **2.7.2**. Depois é mostrado o valor da matriz. A **Figura 36** demonstra isso mesmo.

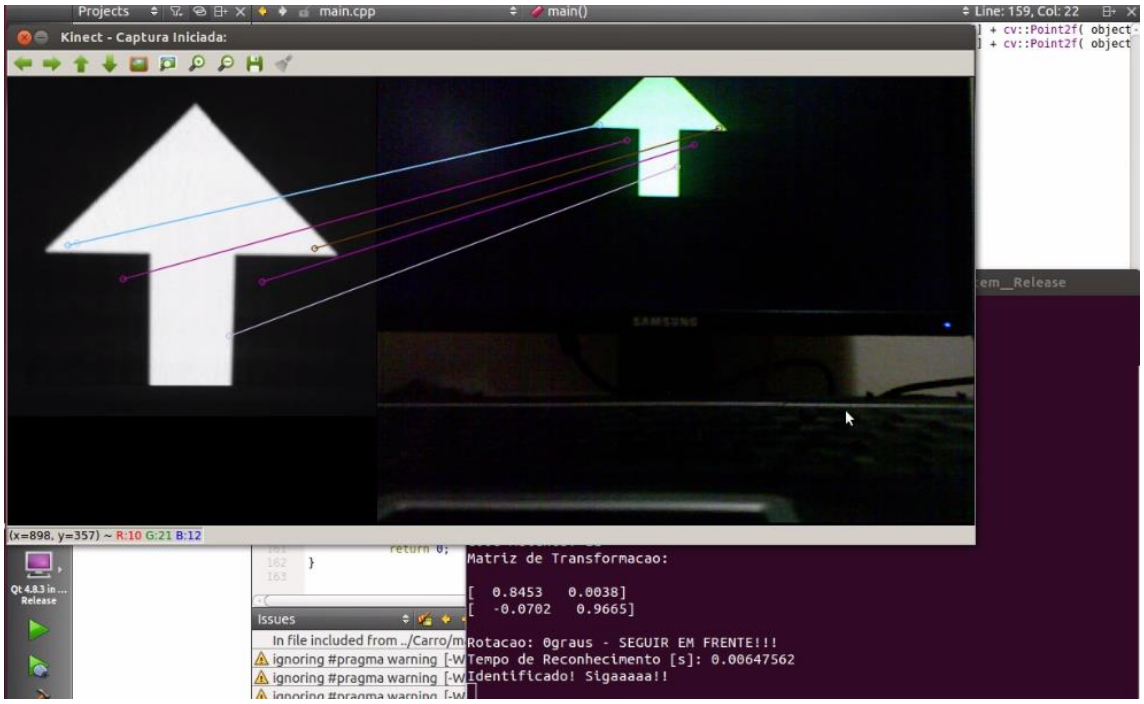

Figura 36 - Deteção Seta Seguir em Frente e Respetiva Matriz de Rotação

Ora, o caminho a seguir depende do valor desta matriz H. Nesse sentido, é necessário encontrar uma relação entre as várias possibilidades de resultado da matriz. Olhando novamente para os valores das matrizes de rotação, conclui-se que, o valor de (0,0), ou seja, linha 0, coluna 0 só é igual a 1 no caso da seta para seguir em frente. Sabe-se também que o valor presente em (1,0), isto é, linha 1, coluna 0, só é igual a 1 no caso da seta para virar à esquerda. Finalmente, o valor de (0,1), linha 0, coluna 1 apenas é igual a 1 no caso de a seta indicar a direção da direita.

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad H = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Frente Esquerda Direita

Figura 37 – Relação entre as matrizes de rotação

Assim, depois de calculada a matriz de rotação, se os valores obtidos em cada uma das matrizes destacadas acima, for superior a 0.8, conclui-se que direção deverá seguir o carro. No caso do teste apresentado na **Figura 36**, o resultado obtido para a matriz de rotação indica que o valor de H[0,0] é maior que 0,8, que foi a condição estipulada para determinar qual a orientação da seta. Nesse sentido, através da matriz a seguir percebe-se que o sinal identificado foi a seta de seguir em frente.

$$H = \begin{bmatrix} 0.8453 & 0.0038 \\ -0.0702 & 0.9665 \end{bmatrix}$$

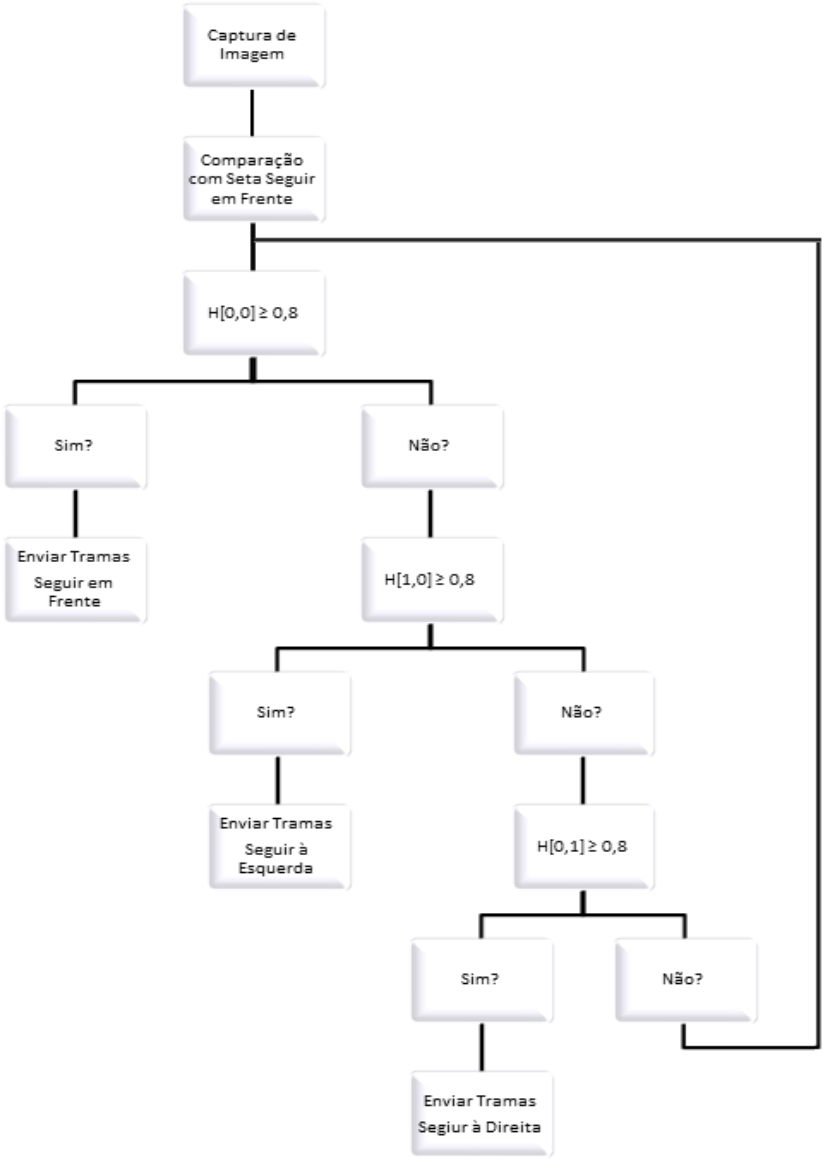

Figura 38 - Fluxograma Reconhecimento de Semáforos

Depois de solucionada toda a problemática das matrizes de rotação, bastará manipular os valores calculados pela aplicação e enviar os sinais de controlo adequados para o microcontrolador.

Algoritmo 6 – Conjunto de Caracteres a Enviar depois de Reconhecimento

```
1     SE (H[0,0] > 0.8)
2       INÍCIO
3             Escrever String Porta Serie (“w”);
4             Esperar (1 segundos);
5             Escrever String Porta Serie (“p”);
6       FIM
7
8     SE (H[1,0] > 0.8)
9       INÍCIO
10            Escrever String Porta Serie (“aa”);
11            Escrever String Porta Serie (“w”);
12            Esperar (1 segundos);
13            Escrever String Porta Serie (“p”);
14      FIM
15
16    SE (H[0,1] >0.8)
17      INÍCIO
18            Escrever String Porta Serie (“dd”);
19            Escrever String Porta Serie (“w”);
20            Esperar (1 segundos);
21            Escrever String Porta Serie (“p”);
22      FIM
```

Para a seta de seguir em frente, aciona-se o motor de tração através do envio via série do caracter “w” que fará com que a velocidade seja de 1m/s. Para o caso de o veículo estar posicionado a 1 metro do monitor, aplica-se um tempo de espera de 1 segundo que permitirá ao veículo seguir em frente. Finalmente é enviado o caracter de paragem “p”.

No que diz respeito às setas de viragem, é enviado um conjunto de dois caracteres "aa" para a esquerda ou "dd" para a direita que fará com que o ângulo de direção do veículo seja de 10º, suficiente para efetuar a viragem. Depois é enviado o caracter de andamento com um tempo de espera de acordo com a distância ao monitor. Assim que o veículo atinja a posição final, é enviado o caracter de paragem.

3.2.4 Estacionamento

Outro dos objetivos principais deste trabalho, passa por dotar o sistema da capacidade de avaliar um conjunto de lugares de estacionamento, perceber qual dos lugares se encontra livre e, finalmente, proceder ao estacionamento. O processo de estacionamento, para este trabalho, passará sempre por analisar três lugares em que dois estão ocupados. As situações possíveis estão representadas na **Figura 39**.

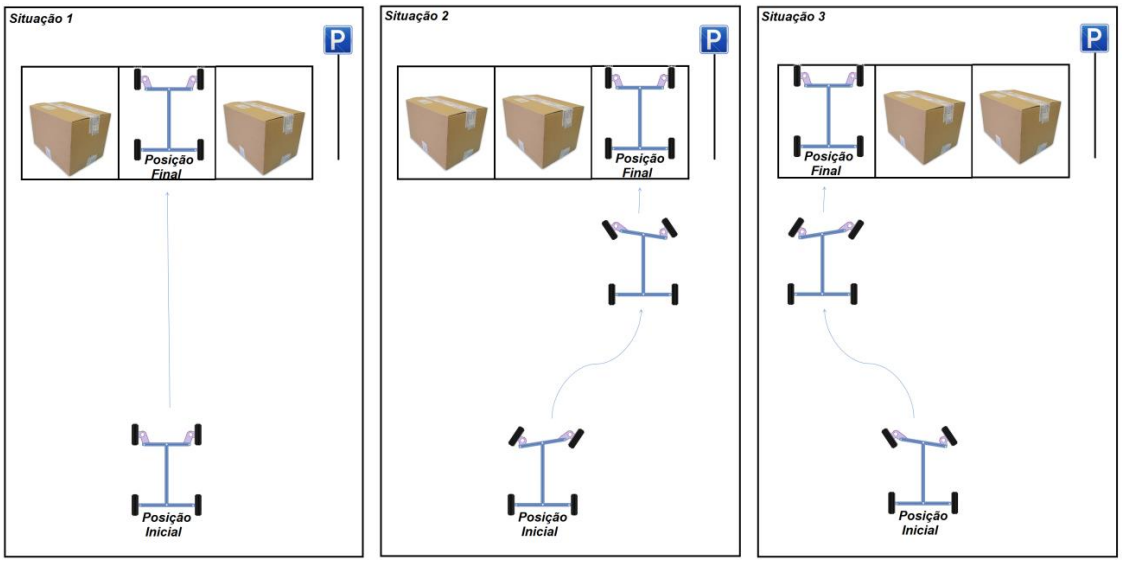

Figura 39 - Possíveis Situações para Estacionamento

Por forma a determinar qual dos lugares se encontra livre para estacionamento foi desenvolvido um algoritmo, com base nas seguintes ações.

Em primeiro lugar, depois de iniciar a captura, é dividida a zona de estacionamento na imagem capturada pela Kinect em 3 retângulos, através da inclusão de duas linhas divisórias verticais, com base na função *line( )* do OpenCV, como é demonstrado na **Figura 40**. Sabendo que a dimensão da imagem é de 640 x 480 pixels, deduz-se que a linha que separa o lugar do lado esquerdo do lugar central será a partir da coluna 215, enquanto a linha que separa o lugar do meio daquele que está mais à direita se encontra na coluna 425. Estes valores representam o pixel com o mesmo número, tendo como referência a largura da imagem. As linhas vão até ao pixel 180, que representa pouco menos de 1/3 da altura.

Figura 40 - Divisão dos Lugares de Estacionamento

Seguidamente, a estratégia passa por transformar a imagem capturada, através da aplicação de um *threshold*. Dessa forma, será manipulada uma imagem binarizada que permitirá aceder aos pixels que definem se o lugar está livre ou ocupado.

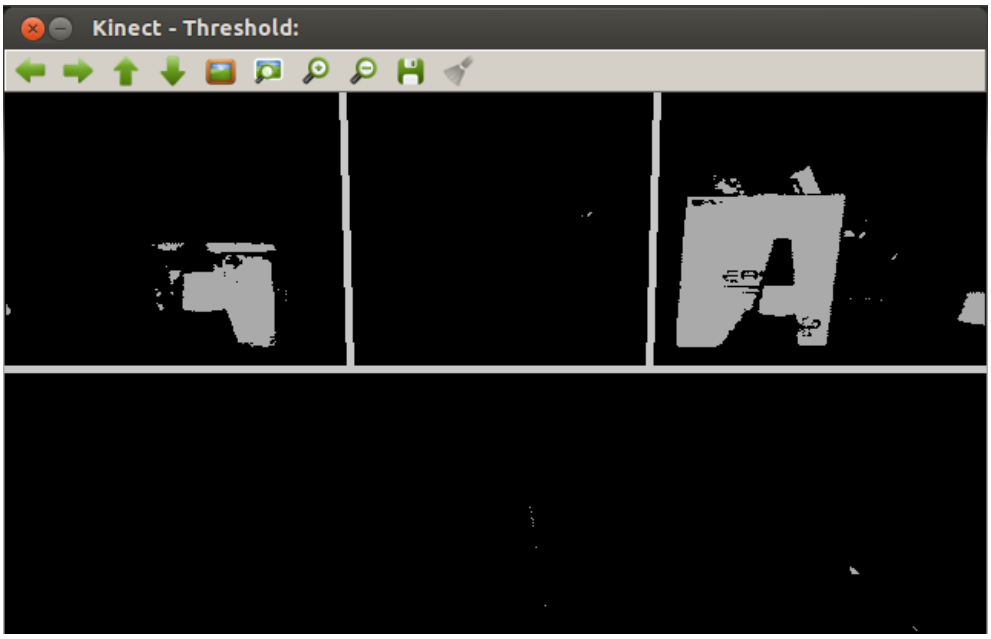

Figura 41 - Visão Binarizada sobre Zona de Estacionamento

Depois de analisar a **Figura 41**, percebe-se que os objetos que ocupam os lugares da esquerda e direita formam conjuntos de pixels brancos e que o lugar do meio que se encontra livre, não apresenta pixels brancos. Assim, foi implementado um algoritmo que permite a contagem de pixels brancos correspondentes a cada lugar de estacionamento. O lugar que apresentar o menor número de pixels brancos será o lugar livre.

Algoritmo 7 – Contagem de Pixels Brancos para determinar lugar livre

```
1   VARIÁVEIS: linha, coluna, count_frente:int,  imagem_threshold:Mat
2   PARA coluna ← 0 ATÉ 220 FAÇA
3     INÍCIO
4         PARA linha ← 0 ATÉ 180 FAÇA
5             INÍCIO
6                 SE (imagem_threshold (coluna, linha) >150) ENTÃO
7                     INÍCIO
8                         count_frente ++;
9                     FIM
10            FIM
11    FIM
```

No Algoritmo **7**, vê-se que é feito um varrimento desde o ponto (0,0) até à coluna 220 e linha 180, como explicado atrás. Nesses pixels, será determinado se o threshold ultrapassa o valor de 150, valor este que garante que o pixel representará um ponto claro, sinónimo de que está ocupado. Para a situação representada na **Figura 41**, é expectável que o output gerado indique que o lugar do meio seja o que contém menos pixels brancos. Foi exatamente o resultado obtido, como demonstrado na **Figura 42**.

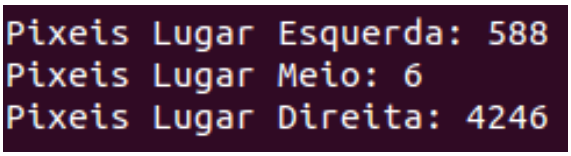

Figura 42 - Número de Pixels por Lugar de Estacionamento

Depois de conhecido o número de pixels, é efetuada a comparação entre os três lugares. Para este caso concreto, será determinado que o lugar central se encontra livre, sendo enviada a trama correspondente à ocupação desse lugar. Dado que a posição inicial é conhecida, como representado na **Figura 39** e tendo sido já determinado o lugar livre, serão enviadas as tramas correspondentes.

Para o caso de o lugar central estar livre, aciona-se o motor de tração através do envio via série do caracter "w" que fará com que a velocidade seja de 1m/s. Para o caso de o veículo estar posicionado a 3 metros da zona de estacionamento, aplica-se um tempo de espera de 3 segundos que permitirá ao veículo atingir o lugar livre. Finalmente é enviado o caracter de paragem "p".

No que diz respeito aos lugares laterais, é enviado um conjunto de 2 caracteres "aa" para a esquerda ou "dd" para a direita que fará com que o ângulo de direção do veículo seja de 10º, suficiente para atingir as posições laterais. Depois é enviado o caracter de andamento com um tempo de espera de acordo com a distância à zona de estacionamento.

Algoritmo 8 – Tramas para Estacionamento

```
1    VARIÁVEIS: count_frente, count_direita, count_esquerda:int,
2    SE (count_frente for o MENOR)
3      INÍCIO
4            Escrever String Porta Serie ("w");
5            Esperar (3 segundos);
6            Escrever String Porta Serie ("p");
7      FIM
8
9    SE (count_esquerda for o MENOR)
10     INÍCIO
11           Escrever String Porta Serie ("aa");
12           Escrever String Porta Serie ("w");
13           Esperar (3 segundos);
14           Escrever String Porta Serie ("dd");
15           Escrever String Porta Serie ("p");
16     FIM
17
18   SE (count_direita for o MENOR)
19     INÍCIO
20           Escrever String Porta Serie ("dd");
21           Escrever String Porta Serie ("w");
22           Esperar (3 segundos);
23           Escrever String Porta Serie ("aa");
24           Escrever String Porta Serie ("p");
25     FIM
```

Estes conjuntos de caracteres são bastante idênticos aos enviados no caso do reconhecimento de sinalização, à exceção do facto de que, no caso do algoritmo de estacionamento é, também, enviado o conjunto de caracteres para que a direção volte ao ângulo inicial. Se virou à esquerda inicialmente, virará à direita para finalizar o movimento. Assim que o veículo atinja a posição final, é enviado o caracter de paragem.

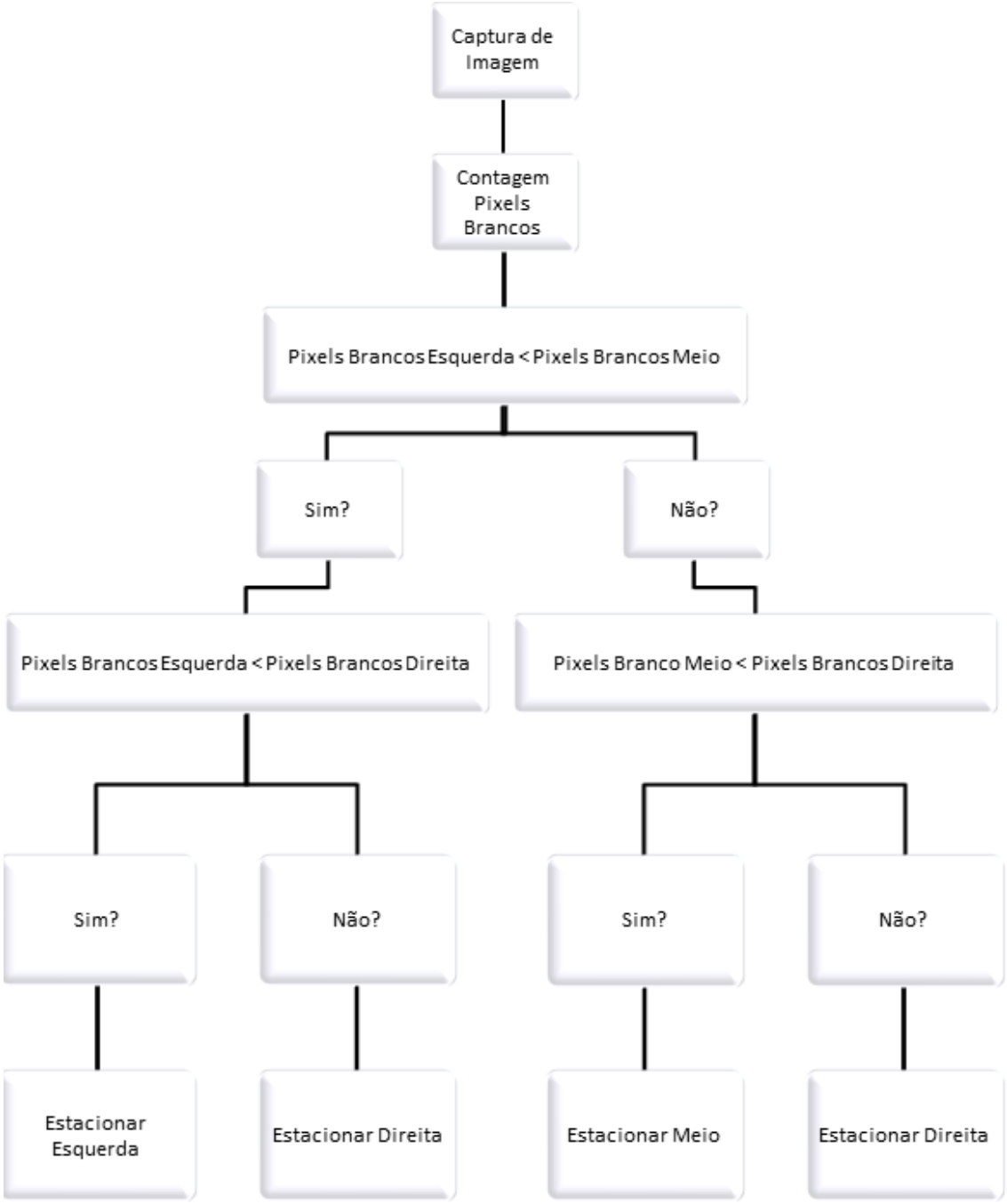

Figura 43 - Fluxograma do Processo de Estacionamento

3.2.5 Determinação do Tempo de Processamento

De forma a perceber o tempo de processamento de um determinado algoritmo, foi encontrada a solução apresentada no **Algoritmo 9**.

Algoritmo 9 – Determinação do Tempo de Processamento	
1	VARIÁVEL: t
2	t ← getTickCount( );
...	....
	CONJUNTO DE INSTRUÇÕES
...	....
n	t ← getTickCount( ) – t/getTickFrequency( );

Entre as linhas 2 e n, estará o conjunto de instruções para o qual se deseja conhecer o tempo de execução. As funções aqui detalhadas permitirão perceber de forma eficaz quanto demora o sistema a reconhecer a sinalização, bem como os lugares livres de estacionamento.

3.2.6 Planeamento da Trajetória

O último objetivo deste trabalho passa por planear a trajetória do veículo, através do conhecimento do espaço onde este se encontra inserido.

Para tal, poderá ser utilizada a imagem capturada pela Kinect para construir um mapa do local, como na ou, ainda, um ficheiro previamente existente, sempre com um tamanho de 1000x1000px, como na .

Figura 44 - Captura Kinect para Planeamento

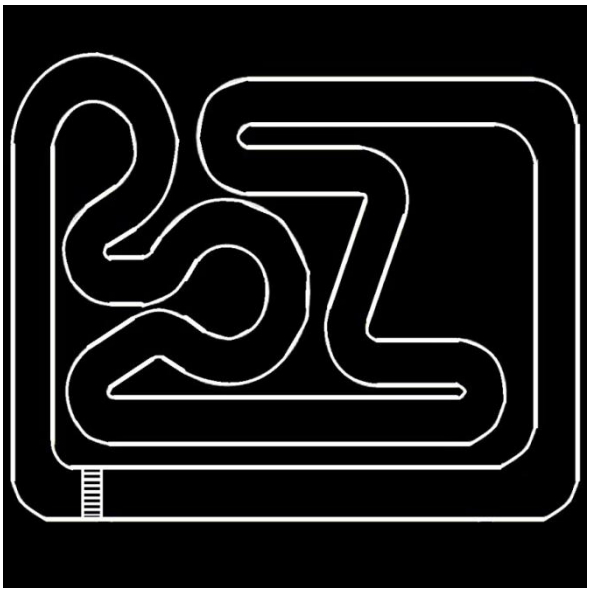

Figura 45 - Mapa 1000x1000px para Planeamento

Depois de obtido o mapa, o passo a tomar será retirar o plano do chão, de forma a que este não seja considerado obstáculo. Para tal, é aplicada uma transformação (matriz de rotação em torno do eixo transversal (pitch). Isso permitirá visualizar o que a Kinect capta numa vista de cima para baixo. Depois de obter esses dados, será utilizado um algoritmo para remover todos os pontos que se encontram abaixo da altura do carro, nomeadamente o chão, e acima da altura máxima do carro com a Kinect. Dessa forma, só serão considerados obstáculos os pontos que se mantenham entre o chão e a Kinect.

Seguidamente, terá que ser efetuada uma configuração correta do espaço onde o carro está inserido. Essa configuração consiste em aumentar o tamanho dos obstáculos e diminuir o tamanho do robô de forma a garantir um correto planeamento da trajetória e evitando colisões à quais não se esperava. Finalmente, depois da configuração, será implementado o algoritmo RRT, que calculará o planeamento entre o ponto inicial onde se encontra o veículo e o ponto final, a determinar pelo utilizador.

3.2.6.1 Aquisição de Imagem e Extração de Pontos

O algoritmo desenvolvido para planeamento da trajetória inicia-se com a captura da imagem através da Kinect, como em **3.2.1**. Seguidamente, será construída uma nuvem de pontos da

imagem recebida representante do espaço onde se encontra o carro. Esta nuvem de pontos é construída com base na biblioteca PCL (Point Cloud Library) [**39**] que disponibiliza soluções orientadas para este tipo de sistemas.

3.2.6.2 Configuração do Espaço

De forma a tornar o processo de planeamento da trajetória mais simples e eficaz, deverá transformar-se o espaço onde o carro se encontra numa representação em que o próprio carro é visto como um ponto em duas dimensões, ao passo que os obstáculos são alargados consoante a geometria do veículo. Esta representação dá pelo nome de Configuração de Espaço.

Esta configuração tem dois procedimentos essenciais. Primeiramente, é necessário escolher um ponto dentro do mapa que representa a forma do carro, para, de seguida, serem aumentados todos os pontos que são considerados obstáculos, redesenhando-os com base na estrutura do carro. Este aumento do tamanho significa uma adição do tamanho do carro a todos os vértices dos obstáculos, como representado na **Figura 46**.

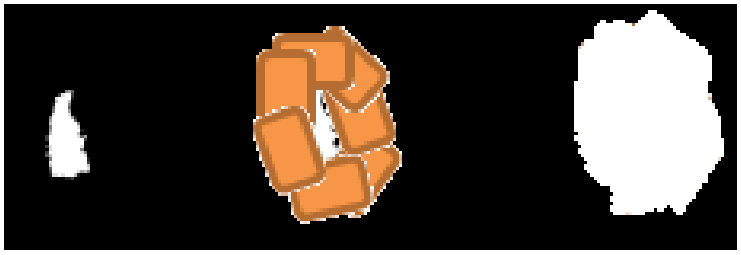

Figura 46 - Aumento da Área do Obstáculo com base no carro

Através do OpenCV, são determinados os contornos do mapa projetado no plano. Os contornos definidos são compostos por vértices que possibilitam a criação do espaço de configuração do robô. A projeção do veículo Fórmula UMTD2 sobre o plano xz, pode ser aproximada a um retângulo com 60cm por 36cm. Assim sendo, é possível formar a configuração do espaço de qualquer mapa selecionando o centro do retângulo que representa o carro conduzindo-o à volta dos vértices que constituem os obstáculos, para obter um espaço de configuração similar ao lado direito da **Figura 46**.

3.2.6.3 Deteção de Colisões

A biblioteca OMPL não congrega um método de deteção de colisões. Este facto advém de que todos os problemas de robótica são únicos e não é possível generalizar ou idealizar um método de deteção universal. Nesse sentido, para o algoritmo de planeamento, o utilizador deve selecionar uma representação computacional para o robô e providenciar um método explícito de validação/colisão. Foi então definido que o carro seria representado por um ponto, e o espaço de configuração criado permite averiguar a validade de um estado. Se um determinado estado do veículo, representado por um ponto, estiver dentro de uma região definida como obstáculo, isto é, um ponto branco, então o estado é considerado não-válido. Por outro lado, se o ponto representativo do veículo estiver numa região livre, ponto preto, o estado é considerado válido.

3.2.6.4 Construção da RRT

O passo final para o planeamento da trajetória está na construção da RRT. Esta tarefa congrega os seguintes procedimentos:

- Definir ponto aleatório no mapa e perceber qual o ponto mais próximo
- Definir um angulo aleatório de direção do carro (p. ex. 30º)
- Definir uma velocidade constante (p. ex. 1m/s);
- Integrar esse movimento durante uma constante de tempo (p. ex. 1seg).
- Quanto maior a constante de tempo, menores as mudanças de direção, mas mais tempo a determinar o caminho.

3.2.7 Outros Algoritmos

Durante o período de realização desta dissertação, foram implementadas várias soluções que, não se enquadram na solução final. Apesar desse fato, alguns dos algoritmos são de interesse elevado para projetos similares a este e são aqui apresentados.

3.2.7.1 Inclinação do Motor da Câmara Kinect

Por forma a utilizar a capacidade de inclinação do motor da câmara Kinect, foi implementada uma classe que possibilita uma rotação nos dois sentidos. Num ficheiro KinectControl.h são implementadas funções que acedem aos registos do motor na Kinect.

Primeiramente é declarada uma função *create ( )* que, quando chamada, cria uma variável de estado, *res*, que iniciará a comunicação por USB. Seguidamente, é efetuada uma comparação por forma a verificar se a comunicação USB se encontra inicializada. Se sim é retornada uma flag de confirmação. Isto permitirá que, na função de movimento, *Move( )*, se envie o ângulo pretendido, por forma a efetuar a rotação da Kinect, através da função *USBSendControl( ).*

No ficheiro main.cpp, será declarada uma variável xKinectMotor que permitirá enviar o ângulo pretendido. Como se verifica em **2.6.4**, a Kinect poderá manter ângulos entre -30º e 30º em relação ao solo.

3.2.7.2 Visão Binarizada para Reconhecimento de Pista

Incialmente, esta dissertação tinha como objetivo a participação no FNR 2013. Uma das provas é a corrida numa pista igual à da **Figura 3**. De forma a facilitar o reconhecimento das linhas da pista, foi desenvolvido um algoritmo que captura a imagem com um determinado threshold e permite capturar a imagem da pista apenas focado nas linhas delimitadoras.

Algoritmo 10 – Visão Binarizada

```
1   VARIÁVEIS: captura: VideoCapture, imagem, imagem_thresh: Mat;
2                  threshmin, threshmax: Int
3   Abrir captura (CV_CAP_OPENNI);
4   PARA ( ; ; ) FAÇA
5   INÍCIO
6         Libertar imagem ( );
7         Reter captura ( );
8         Restaurar captura (imagem, CV_CAP_OPENNI_GRAY_IMAGE);
9         threshold (imagem, imagem_thresh, threshmin, threshmax, THRESH_BINARY);
10        Mostrar imagem (image_thresh)
11  FIM
12  FIM PARA;
```

De acordo com o **Algoritmo 10**, é iniciada a aquisição de imagem, tal e qual como em **3.2.1**. No entanto, a função *retrieve ( )* (Linha 8) devolverá a imagem em tons de cinzento, através do parâmetro *CV_CAP_OPENNI_GRAY_IMAGE.* Na Linha 9 é aplicada a função *threshold( )*, com os parâmetros imagem de entrada *imagem*, imagem de saída *imagem_thresh*, os valores limites de *threshold*, para o caso entre 180 e 255 e finalmente o filtro de imagem *THRESH_BINARY*. O resultado é mostrado na figura seguinte.

Figura 47 - Visão Binarizada para Reconhecimento de Pista

Capítulo 4 - Testes e Resultados

4.1 Resultados

4.1.1 Reconhecimento de Semáforos

O reconhecimento de sinalização foi testado para os três diferentes sinais, nomeadamente o de seguir em frente, o de virar à direita e o de virar à esquerda. O veículo foi colocado num ponto inicial às distâncias de 50 cm, 1m, 2m e 3m de um monitor. Para cada distância houve três tipos de luminosidade ambiente. O tempo de reconhecimento e atuação foi medido por software, através do algoritmo explicado em **3.2.5**. Quando o tempo de reconhecimento ultrapassou os 30 segundos, o teste foi interrompido e concluído que para a situação em causa, o sistema não permite a identificação da seta. A Kinect manteve-se, para todos os testes, com um ângulo de +30º em relação ao veículo.

Figura 48 - Testes ao Reconhecimento de Semáforos

Tabela 5 – Resultados dos Testes ao Reconhecimento de Sinalização – Seguir em Frente

Distância do Carro ao Monitor	Luminosidade	Reconheceu Seta?	Nº de Keypoints	Tempo de Reconhecimento
50 cm	Alta	Sim	16	< 1seg
50 cm	Média	Sim	11	< 1seg
50 cm	Baixa	Sim	13	1,13seg
1m	Alta	Sim	16	< 1seg
1m	Média	Sim	9	2,98seg
1m	Baixa	Sim	9	2,51seg
2m	Alta	Sim	17	3,02seg
2m	Média	Sim	13	14,18seg
2m	Baixa	Sim	8	19,34seg
3m	Alta	Sim	14	4,12seg
3m	Média	Sim	12	2,55seg
3m	Baixa	Não	3	n.a.

Tabela 6 - Resultados dos Testes ao Reconhecimento de Sinalização – Virar à Direita

Distância do Carro ao Monitor	Luminosidade	Reconheceu Seta?	Nº de Keypoints	Tempo de Reconhecimento
50 cm	Alta	Sim	13	2,14seg
50 cm	Média	Sim	13	2,77seg
50 cm	Baixa	Sim	7	28,08seg
1m	Alta	Sim	10	8,12seg
1m	Média	Sim	7	11,16seg
1m	Baixa	Não	3	n.a.
2m	Alta	Sim	6	4,88seg
2m	Média	Não	4	n.a.
2m	Baixa	Não	2	n.a.
3m	Alta	Não	4	n.a.
3m	Média	Não	3	n.a.
3m	Baixa	Não	2	n.a.

Tabela 7 - Resultados dos Testes ao Reconhecimento de Sinalização – Virar à Esquerda

Distância do Carro ao Monitor	Luminosidade	Reconheceu Seta?	Nº de Keypoints	Tempo de Reconhecimento
50 cm	Alta	Sim	16	< 1seg
50 cm	Média	Sim	13	4,00seg
50 cm	Baixa	Sim	13	13,12seg
1m	Alta	Sim	17	1,85seg
1m	Média	Sim	10	8,43seg
1m	Baixa	Não	4	n.a.
2m	Alta	Sim	9	4,79seg
2m	Média	Sim	7	22,87seg
2m	Baixa	Não	2	n.a.
3m	Alta	Não	5	n.a.
3m	Média	Não	4	n.a.
3m	Baixa	Não	4	n.a.

4.1.2 Análise de Resultados – Reconhecimento de Semáforos

O algoritmo de Reconhecimento de Semáforos demonstrou maior eficácia no reconhecimento da seta de seguir em frente. Este resultado, teoricamente, é previsível, pois a imagem com que é comparada a imagem capturada pela Kinect é, exatamente, a seta de seguir em frente. Relativamente aos parâmetros testados, tanto a distância, como a luminosidade influenciam bastante os resultados. Para distâncias mais curtas do veículo em relação ao monitor, o sinal foi reconhecido em 90% das vezes. A partir dos 2 metros de distância a dificuldade no reconhecimento aumentou, à exceção da seta de seguir em frente que foi reconhecida facilmente. Em termos de luminosidade, o sistema comporta-se melhor com maiores luminosidades. Conclui-se, ainda, que qualquer que seja o semáforo, o sistema tende a ter um demonstrar maior dificuldade de reconhecimento por volta dos 2,3 metros de distância, onde apenas em duas situações foi reconhecida a seta de andar em frente. Não foram efetuados testes a mais de 3 metros de distância, nem com luminosidade nula. Para alguns testes em que não foi reconhecida a seta em menos de 30 segundos, foi repetida a experiência, tendo-se obtido, um melhor resultado.

4.1.3 Estacionamento

O processo de estacionamento foi testado sempre com dois lugares ocupados em três. O carro foi colocado num ponto inicial às distâncias de 50 cm, 1m, 2m e 3m. À semelhança dos testes ao reconhecimento de sinalização, para cada distância houve três tipos de luminosidade ambiente. O tempo de reconhecimento e atuação foi medido por software. A Kinect manteve-se, para todos os testes, com um ângulo de -30°, como se pode verificar na **Figura 49**.

Figura 49 - Testes ao Processo de Estacionamento

Tabela 8 - Resultados dos Testes ao Estacionamento – Lugar Livre Frente

Distância do Carro à Zona de Estacionamento	Luminosidade	Determinou Lugar Livre?	Tempo de Reconhecimento
50 cm	Alta	Sim	7,12seg
50 cm	Média	Sim	6,99seg
50 cm	Baixa	Sim	7,75seg
1m	Alta	Sim	7,22seg
1m	Média	Sim	7,51seg
1m	Baixa	Sim	7,08seg
2m	Alta	Sim	8,33seg
2m	Média	Sim	9.36seg
2m	Baixa	Sim	9,46seg
3m	Alta	Sim	9.33seg
3m	Média	Sim	9.47seg
3m	Baixa	Sim	9,31seg

Tabela 9 - Resultados dos Testes ao Estacionamento – Lugar Livre Esquerda

Distância do Carro à Zona de Estacionamento	Luminosidade	Determinou Lugar Livre?	Tempo de Reconhecimento
50 cm	Alta	Sim	7,88seg
50 cm	Média	Sim	7,23seg
50 cm	Baixa	Sim	6,14seg
1m	Alta	Sim	8,01seg
1m	Média	Sim	7,91seg
1m	Baixa	Sim	8,32seg
2m	Alta	Sim	8,33seg
2m	Média	Sim	7.99seg
2m	Baixa	Sim	8,90seg
3m	Alta	Sim	12,76seg
3m	Média	Sim	9,34seg
3m	Baixa	Sim	10,55seg

Tabela 10 - Resultados dos Testes ao Estacionamento – Lugar Livre Direita

Distância do Carro à Zona de Estacionamento	Luminosidade	Determinou Lugar Livre?	Tempo de Reconhecimento
50 cm	Alta	Sim	7,79seg
50 cm	Média	Sim	7,87seg
50 cm	Baixa	Sim	7,42seg
1m	Alta	Sim	8,07seg
1m	Média	Sim	8,18seg
1m	Baixa	Sim	8,06seg
2m	Alta	Sim	8,33seg
2m	Média	Sim	9,44seg
2m	Baixa	Sim	9,69seg
3m	Alta	Sim	9,38seg
3m	Média	Sim	9,50seg
3m	Baixa	Sim	8,94seg

4.1.4 Análise de Resultados – Estacionamento

O algoritmo desenvolvido para proceder ao estacionamento revelou-se 100% eficaz para os testes efetuados. Comparativamente com os resultados do Reconhecimento de Sinalização, nota-se que a luminosidade não tem grande interferência no reconhecimento dos lugares ocupados e, consequentemente, do lugar livre. A distância apenas faz aumentar o tempo no que concerne à demora do veículo a chegar ao local de estacionamento, não aumentando significativamente o tempo de reconhecimento. Isto prende-se com o facto de não haver um processamento pesado para o sistema. O algoritmo baseia-se no cálculo de pixels brancos. Independentemente da distância ou luminosidade, os pontos brancos serão reconhecidos. Conclui-se que o parâmetro que poderá influenciar nos resultados será a área de cálculo dos pixels brancos. As linhas e colunas que delimitarão os lugares à vista da Kinect variam com a distância e, nesse sentido, uma correta escolha dos limites de cada lugar de estacionamento é essencial. Não foram efetuados testes a mais de 3 metros de distância.

4.1.5 Planeamento da Trajetória

O processo de planeamento da trajetória foi testado através da captura do espaço, onde foram colocados, propositadamente, alguns obstáculos. O carro foi colocado num ponto inicial conhecido e a Kinect manteve-se sempre a 40 cm do solo, com uma inclinação de -10º.

Figura 50 - Espaço capturado nº1, com obstáculos

Como se pode ver na **Figura 50**, foram colocados 6 pinos, 3 de cada lado e um outro obstáculo. Além disso, mais afastado do carro, está um número considerável de obstáculos. Então, como explicado em **3.2.6**, depois da captura da imagem, determinar-se-á um conjunto de pontos. Os representados a branco são considerados obstáculos, enquanto que os representados a preto simbolizam uma posição livre.

Figura 51 - Mapa de Obstáculos

Como se pode depreender depois da análise da **Figura 51**, os tanto os "reais" obstáculos, como o próprio chão são considerados obstáculos. Nesse sentido é necessário aplicar o algoritmo de remoção do chão, ou seja, remover todos os pontos abaixo da altura do robô e acima da altura da Kinect. O resultado é o seguinte:

Figura 52 - Espaço Resultante depois da Remoção do Chão

A **Figura 52** demonstra o resultado obtido depois da remoção dos pontos acima mencionados. Neste momento, o passo seguinte será implementar a configuração do espaço, isto é, aumentar o tamanho dos obstáculos com base no tamanho do robô.

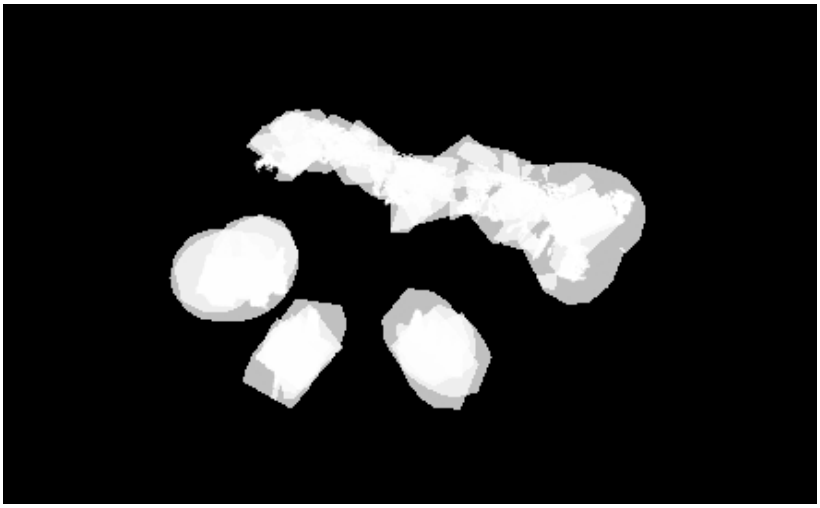

Figura 53 - Configuração do Espaço do Mapa Obtido

Com a obtenção da configuração do espaço resta planear a trajetória do ponto inicial onde o carro se encontra até ao ponto final, a determinar pelo utilizador. Esse ponto corresponderá a uma coordenada escolhida imediatamente antes do planeamento.

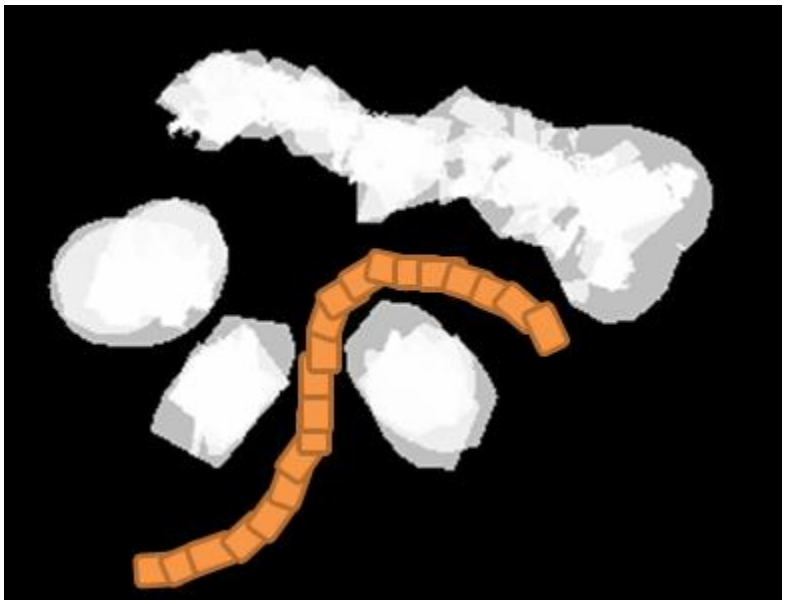

Figura 54 - Caminho Ótimo determinado

Depois de realizado o teste atrás descrito, foi construído um novo ambiente onde o veículo se encontrava, definindo o ponto para o qual se desejava construir a trajetória. O ambiente é demonstrado na seguinte figura.

Figura 55 - Ambiente nº2

Aplicando os algoritmos de remoção do plano do chão e da configuração do espaço, o resultado obtido está demonstrado na **Figura 56**.

Figura 56 - Plano do Chão Removido e Configuração do Espaço

Finalmente, foi calculada a RRT e, consequentemente a trajetória. O resultado obtido é apresentado na **Figura 57**.

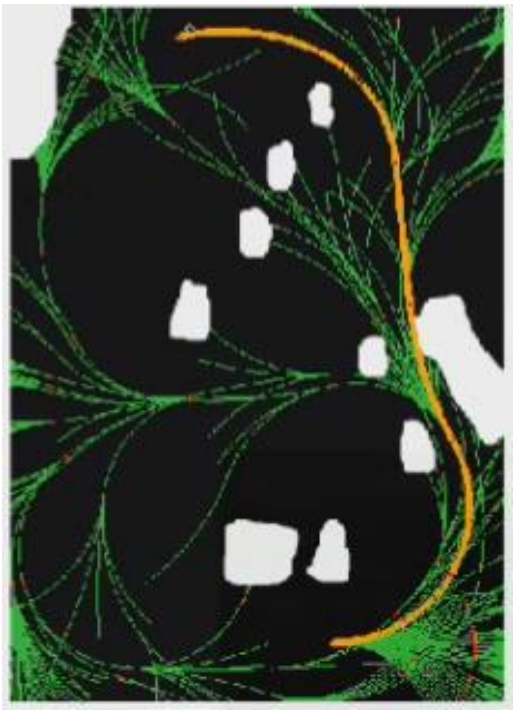

Figura 57 - RRT Gerada para Ambiente nº2

4.1.6 Análise de Resultados – Planeamento da Trajetória para Captura de Kinect

Sem sombra de dúvida, o planeamento da trajetória é o objetivo mais complexo desta dissertação. Envolveu um estudo aprofundado de várias bibliotecas e técnicas de manipulação de dados e mapeamento. Os testes demonstrados atrás demonstram que o algoritmo se adapta aos vários ambientes.

Como explicado anteriormente, o planeamento pode ser efetuado também tendo em conta um mapa previamente construído / desenhado, desde que os obstáculos / delimitações estejam representados a branco e o tamanho do ficheiro de imagem seja de 1000x1000px.

Analisando a **Figura 58**, percebe-se que os pontos a brancos são obstáculos e os pequenos pontos a laranja são os pontos inicial (mais abaixo) e final (mais acima). A trajetória calculada é apresentada na **Figura 59**.

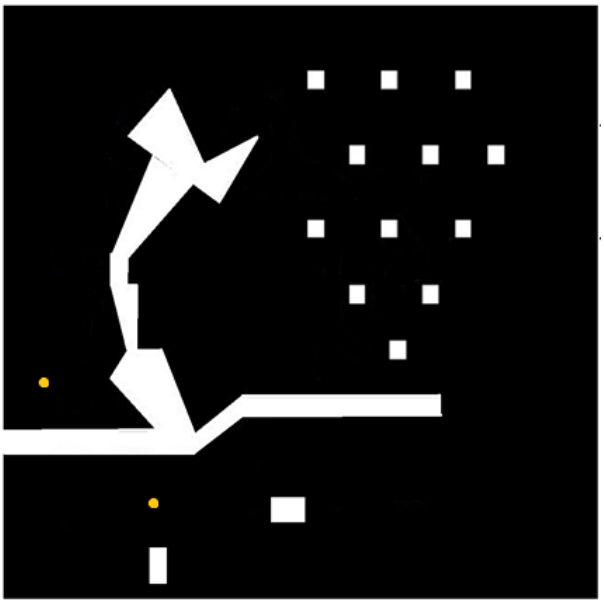

Figura 58 - Mapa nº1

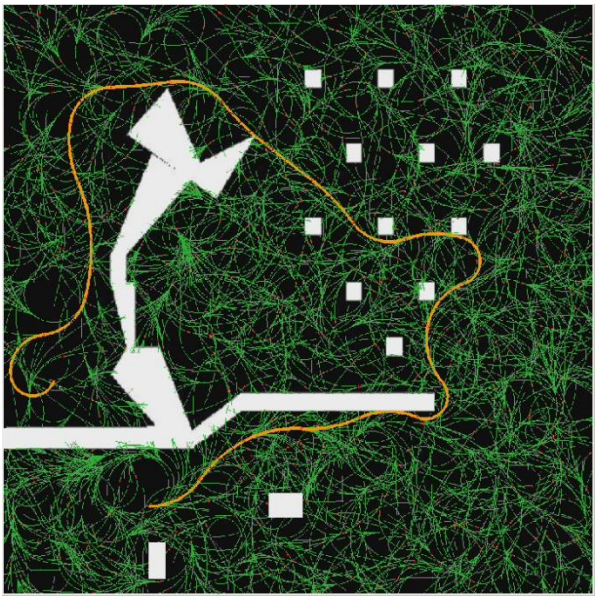

Figura 59 - Trajetória obtida para Mapa nº1

Outro exemplo é demonstrado de seguida. Desenhando uma pista e definindo os pontos iniciais e finais, é calculada a RRT para que o veículo percorra o percurso dentro das delimitações e de forma eficaz. A pista apresentada na **Figura 60** será percorrida com a trajetória gerada, demonstrada na **Figura 61**.

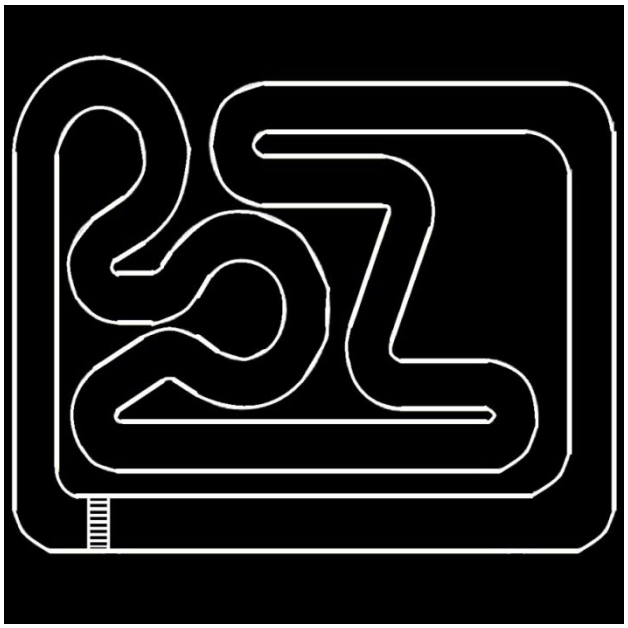

Figura 60 - Mapa nº2

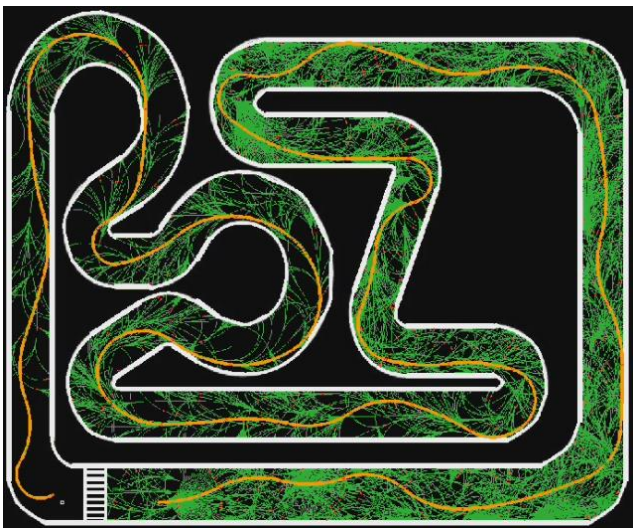

Figura 61 - RRT gerada para Mapa nº2

4.1.7 Análise de Resultados – Planeamento da Trajetória para Mapas

Foram desenhados mapas de forma a testar o algoritmo de planeamento da trajetória. Para os mapas testados, os resultados foram satisfatórios, como se demonstra neste capítulo. É da opinião do aluno que o algoritmo é extremamente robusto e adaptável a qualquer mapa, desde que reúna as condições explicadas.

Capítulo 5 - Conclusões e Perspetivas Futuras

O capítulo 5 apresenta as conclusões do trabalho realizado nesta dissertação. Além disso, são feitos alguns comentários à performance do sistema, tendo em conta os objetivos cumpridos. É dado, também a perceber, ideias de trabalho futuro nesta área de Condução Autónoma.

5.1 Conclusão

Neste trabalho de dissertação foi desenvolvido um conjunto de algoritmos que dotam o carro da capacidade de realizar atividades de forma autónoma, nomeadamente Reconhecimento de Sinalização, Estacionamento e Planeamento de Trajetória. Para cumprir o estipulado utilizou-se o veículo autónomo Fórmula UM TD2 aliado à tecnologia mediática oferecida pela Câmara Kinect da Microsoft. Com a Kinect é possível identificar objetos e sinalização, bem como construir mapas 2D e 3D.

Os algoritmos desenvolvidos em C++ permitiram consolidar os conhecimentos adquiridos ao longo do curso. No que diz respeito ao Reconhecimento de Sinalização, demonstrou-se a eficácia do sistema em perceber qual a decisão a tomar. Apenas em casos de pouca luminosidade e distâncias superiores a 2 metros, o sistema apresentou dificuldades. No entanto, para distâncias mais curtas do veículo em relação ao monitor, o sinal foi reconhecido em 90% das vezes.
No que diz respeito ao processo de estacionamento, revelou-se uma eficácia de 100% nos testes efetuados. Para a determinação dos lugares ocupados, pensa-se que foi encontrada uma solução extremamente eficaz. O facto de serem utilizados os pixels brancos ao invés da informação de profundidade, torna mais coeso e robusto o algoritmo.
Finalmente, para o planeamento da trajetória, foi desenvolvido um conjunto de soluções que congrega não só informação capturada pela Kinect, bem como mapas desenhados manualmente. Os resultados foram satisfatórios e provou-se que o algoritmo se adapta a qualquer mapa que reúna as condições, como o tamanho do ficheiro de imagem e os pixels brancos representem obstáculos.

Relativamente ao hardware, o veículo Fórmula UM TD2, construído por alunos do Grupo de Controlo, Automação e Robótica, demonstrou alguns problemas de controlo, mas, para os algoritmos desenvolvidos, foi possível utilizá-lo e obter os resultados esperados. O computador utilizado na realização desta dissertação foi um HP Pavilion Dv6, com 4GB de memória RAM e um processador Intel Core 2 Duo de 2.26GHz. A velocidade de processamento e implementação considera-se aceitável e não se verificaram problemas na fase de testes.

Em suma, pode-se dizer que a dissertação Visão Artificial em Condução Autónoma com Câmara Kinect teve um desfecho positivo, tendo sido um desafio interessante e com grau de dificuldade elevado. Os objetivos foram cumpridos e, prova disso, são os resultados obtidos.

5.2 Perspetivas Futuras

Em termos futuros, seria relevante continuar a melhorar o Veículo Fórmula UM TD2, nomeadamente os problemas de controlo. Os circuitos funcionam, no entanto, seria um desafio à altura construir um manual de utilizador, com detalhe no processo de controlo, conversor de tensão, PWM, e todas essas caraterísticas elétricas. Durante os testes. o motor de tração era, de quando em vez, acionado sem razão aparentemente lógica para tal. Nesse sentido, será importante, no futuro, melhorar os circuitos e aumentar a robustez do material.

Além disso, acredita-se que o planeamento da trajetória seja uma solução extraordinária para o sistema. O facto de mover um carro para um ponto desejado, tendo apenas que se conhecer um mapa do local, poderá ser uma enorme vantagem para o mercado. Assim, deverá ser criado um sistema de localização mais robusto com planeamento em tempo real.
Propõe-se também a identificação de um conjunto mais alargado de sinalização para, por exemplo, participar em competições do género do FNR.
Finalmente, seria interessante testar todos os algoritmos em ambientes dinâmicos e hostis para perceber o comportamento do sistema.

Capítulo 6 - Bibliografia

Neste capítulo é apresentada a lista de todos os websites, livros e documentos consultados durante a realização desta dissertação estão referenciados neste oitavo e penúltimo capítulo.

6.1 Referências

[1] Autonomous Systems. [Online]. https://sites.google.com/site/autonomoussystemsmw/

[2] José Júlio Areal Ferreira, "Demonstrador de Condução Autónoma," 2012.

[3] R.I.D.E. [Online]. http://www.mikechiafulio.com/RIDE/history.htm

[4] EUREKA Prometheus Project. [Online]. http://en.wikipedia.org/wiki/EUREKA_Prometheus_Project

[5] Fast Company. [Online]. http://www.fastcompany.com/pics/here-come-autonomous-cars?slide=2#2

[6] VaMP. [Online]. http://en.wikipedia.org/wiki/VaMP

[7] NO HANDS ACROSS AMERICA. [Online]. http://www.cs.cmu.edu/afs/cs/usr/tjochem/www/nhaa/nhaa_home_page.html

[8] DEMO III. [Online]. http://www.globalsecurity.org/military/systems/ground/xuv.htm

[9] ATLAS Project. [Online]. http://atlas.web.ua.pt/media.html

[10] DARPA Grand Challenge (2004). [Online]. http://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2004)

[11] World Car Fans. [Online]. http://content.worldcarfans.co

[12] Tartan Racing. [Online]. http://www.tartanracing.org

[13] Festival Nacional de Robótica. [Online]. http://pt.wikipedia.org/wiki/Festival_Nacional_de_Robótica

[14] Condução Autónoma (CA). [Online]. http://pt.wikipedia.org/wiki/Condução_Autónoma

[15] FNR - Condução Autónoma. [Online]. http://www.spr.ua.pt/fnr/

[16] Kinect. [Online]. http://pt.wikipedia.org/wiki/Kinect

[17] Kinect for Windows. [Online]. http://www.microsoft.com/en-us/kinectforwindows/

[18] Kinect Sensor, Engadget. [Online]. www.engadget.com/media/2010/06/kinect-pr-top-1.jpg

[19] TTT -Tourism Think Thank. [Online]. http://www.tttourism.com/

[20] GFI cria aplicação para turismo baseado no Kinect. [Online]. http://wintech.pt/14-noticias/11373-

[21] Toyota Smart Insect Concept with Kinect. [Online]. http://www.engadget.com/2012/10/02/toyotas-smart-insect-concept-ev-kinect/

[22] Wintech. [Online]. http://wintech.pt/14-noticias/13496-

[23] Illumiroom - Microsoft Research. [Online]. http://research.microsoft.com/en-us/projects/illumiroom/

[24] Andreas Ess, Tinne Tuytelaars e Luc Van Gool Herbert Bay, "Speeded-Up Robust Features (SURF)," ETH Zurich, Zurich, 2008.

[25] (2011, Janeiro) A quick peek behind the curtain: Position detection. [Online]. http://www.3dcalifornia.com/2011/01/

[26] Planeamento. [Online]. http://pt.wikipedia.org/wiki/Planejamento

[27] OMPL Documentation. [Online]. http://ompl.kavrakilab.org/

[28] A* Search Algorithm. [Online]. http://en.wikipedia.org/wiki/A*_search_algorithm

[29] A * Pathfinding para Iniciantes. [Online]. http://www.policyalmanac.org/games/aStarTutorial_port.htm

[30] RRT Page. [Online]. http://msl.cs.uiuc.edu/rrt/gallery_car.html

[31] Improving Optimality of RRT. [Online]. http://joonlecture.blogspot.pt/2011/02/improving-optimality-of-rrt-rrt.html

[32] Linux. [Online]. http://pt.wikipedia.org/wiki/LINUX

[33] Ubuntu. [Online]. http://www.ubuntu.com/

[34] Ask Ubuntu. [Online]. http://askubuntu.com/questions/80379/how-many-ubuntu-users-are-there-worldwide

[35] QT Creator | Qt Project. [Online]. http://qt-project.org/search/tag/qt~creator

[36] OpenCV. [Online]. http://opencv.org/

[37] OpenNI. [Online]. http://www.openni.org/

[38] Regulamento Competição Condução Autónoma. [Online]. http://www.robotica2012.org/downloads/rules/FNR2012_Autonomous_Driving_Competition_en.pdf

[39] Point Clouds. [Online]. http://pointclouds.org/documentation/

[40] Arduino - Software. [Online]. http://arduino.cc/en/main/software