SISTEMA DE DETEÇÃO E CONTORNO DE OBSTÁCULOS PARA ROBÓTICA MÓVEL BASEADO EM SENSOR KINECT

Ted Ferreira Cordeiro

VERSÃO FINAL

Dissertação apresentada à Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Bragança para obtenção do grau de Mestre em Engenharia Industrial, ramo de Engenharia Eletrotécnica

Orientador: Prof. Dr. José Luís Sousa de Magalhães Lima

Bragança
Fevereiro 2014

"As regards obstacles, the shortest distance between two points can be a curve."

Bertolt Brecht

Resumo

Nestes últimos anos, as aplicações dos veículos guiados automaticamente alargaram-se e já não estão limitadas ao setor industrial. Hospitais, meios de transporte públicos e particulares, museus, aeroportos, televigilância, são exemplos de tarefas que podem ser realizadas por um AGV. A crescente utilização do sensor *Kinect* para visualização tridimensional do mundo em substituição de um laser é uma solução de baixo custo. Esta tese aborda o tema de um sistema de deteção e contorno de obstáculos para robótica móvel baseada nesse sensor.

Para isso, foi necessário construir um robô móvel diferencial para validar os algoritmos de deteção e contorno de obstáculos do sistema. Um trajeto alvo será definido por uma faixa no solo. Ainda assim, é estimada e registada graficamente a posição absoluta do robô com base na odometria. O trajeto alternativo do robô em caso de obstáculo terá de ser o mais perto possível da linha e sem haver colisões. O robô terá de reconhecer as extremidades do percurso, dar meia volta e recomeçar o trajeto em sentido oposto. O sistema desenvolvido também permitirá monitorizar parâmetros do robô.

O principal objetivo é que o robô seja capaz de seguir a trajetória desejada evitando obstáculos detetados com base nas imagens de profundidade, tudo isto de forma totalmente autónoma.

Palavras-chave:
Robô móvel diferencial, AGV, RGB-D, Kinect, Deteção de obstáculos, Contorno de obstáculos, Odometria, Seguimento de faixas.

Abstract

In these last years, the applications of automated guided vehicles was expanded and are no longer limited to the industrial sector. Hospitals, public and private transports, museums, airports, tele-vigilance, are examples of tasks that can be performed by an AGV. The growing use of the Kinect sensor for three-dimensional visualization of the world instead of a laser is a low cost solution. This thesis approaches the topic of a detection system and obstacles avoidance for mobile robots based on this sensor.

For this it was necessary to construct a differential mobile robot to validate the algorithms of detection and obstacles avoidance. A target path is defined by a line on the ground. However, it is estimated and recorded graphically the absolute position of the robot based on odometry. The alternative path of the robot in case of obstacle must be as close as possible to the line and with no collisions. The robot has to recognize the signals that are on the ends of the route, turn around and start the journey in the opposite direction. The developed system also allows to monitor parameters of the robot.

The main objective is that the robot is able to follow the desired trajectory while avoiding detected obstacles based on depth images, all this autonomously.

Keywords:

Robot mobile differential, AGV, RGB-D, Kinect, Obstacle detection, Obstacles avoiding, Odometry, Line follower.

Agradecimentos

Em primeiro lugar, quero expressar o meu enorme agradecimento ao Professor Doutor José Luís Sousa de Magalhães Lima, orientador da presente tese, pelo apoio e disponibilidade constantes, pelos conhecimentos científicos que me transmitiu e acima de tudo, pelo incentivo.

À Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Bragança, e particularmente ao Laboratório de Controlo, Automação e Robótica pelas condições que proporcionou.

Gostaria ainda de agradecer à Andrea, pelo seu apoio, incentivo e sobretudo por sua enorme paciência nos momentos menos bons da elaboração deste trabalho.

Finalmente, um profundo agradecimento a meus pais pelo apoio, compreensão e dedicação demostrada tanto ao longo deste trabalho como sempre.

A todos, muito obrigado.

Índice

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

Lista de Abreviaturas

AC	*Alternating current*
AGV	*Automated guided vehicle*
CMOS	*Complementary metal-oxide-semiconductor*
DC	*Direct current*
FP	Fim de percurso
IA	Inteligência artificial
IV	Infravermelhos
LDR	*Light Dependent Resistor*
LED	*Light Emitting Diode*
RGB	*Red-Green-Blue*
RGB-D	Imagem de profundidade em tonalidades *Red-Green-Blue*
RTDB	*Real time database*
Speed	Parâmetro de velocidade linear do MD25
TTL	*Transistor-Transistor Logic*
Turn	Parâmetro de velocidade angular do MD25
USB	*Universal Serial Bus*
VGA	*Video Graphics Array*

Lista de símbolos

$\vec{d}$	Deslocamento do robô entre duas iterações
x_n	Coordenada em XX do robô num momento n [cm]
y_n	Coordenada em YY do robô num momento n [cm]
θ_n	Orientação do robô num momento n [°]
α	Ângulo entre *pixel* medido e a vertical do *Kinect*
β	Ângulo horizontal entre dois *pixels* medidos
d_g	Distância até um ponto desprezando diferenças de alturas
h_k	Altura do *Kinect*
h_{pc}	Altura do PC (teclado)
α_{min} , α_{max}	Ângulos extremos da zona relevante para o robô
d_{pc}	Distancia entre Kinect e frente do PC desprezando diferenças de alturas
ρ	Relação entre *pixels* e graus verticalmente
σ	Relação entre *pixels* e graus horizontalmente
$N_{linhas_{msc}}$	Número de linhas da máscara usado no processamento de imagens
$d_{k\_ref_{pixel}}$	Distância a que deveria estar um ponto se pertence ao solo do ponto de vista do *Kinect*
$d_{g_{pixel}}$	Distância a que está um ponto, desprezando diferenças de alturas, se pertencer ao solo
$h_{o_{pixel}}$	Altura medida num ponto relativamente ao solo
d_{g_i}	Distância mínima até obstáculo (ponto menos afastado do obstáculo) do ponto de vista do Kinect desprezando alturas
d_{g_f}	Distância máxima até um obstáculo do ponto de vista do Kinect desprezando alturas
d_{g_c}	Distância central de um obstáculo
$\beta_{obj-faixa}$	Ângulo horizontal entre faixa ponto do obstáculo mais próxima desta
d_{obj_faixa}	Distância medida entre obstáculo e faixa
d_{ff}	Distância entre faixa real e faixa imaginária, a segunda permitindo um percurso sem obstáculos

$M_{s_1}, M_{s_2}, M_{s_3}$	Margens de segurança para reduzir riscos de colisões
$d_{g_{limite}}$	Distância a partir da qual um obstáculo é considerado perigoso para o robô, isto implicará uma mudança de estado do robô
$Larg_{robô}$	Largura do robô
P_I	Ponto intermediário usado para contorno de obstáculos que cobrem a faixa
φ_1 , φ_2	Ângulos calculados e recordados pelo robô para o contorno de obstáculos
$Threshold_{Faixa}$	Valor que separa *pixels* que pertencem a faixa daqueles que não pertencem
$Faixa_{l_x}$	Posição central da faixa numa linha x da imagem RGB
$flag_{x_{esq}}$, $flag_{x_{dir}}$	*Flags* esquerda e direita da faixa de uma linha x
$erro_{l_x}$	Diferença entre posição da faixa numa linha x da imagem RGB e o centro dessa mesma linha
$Largura_{Img}$	Largura da imagem
$Altura_{Img}$	Altura da imagem
$Turn_{din}$	*Turn* dinâmico
$Speed_{din}$	*Speed* dinâmico
$k_{Speed_{Max}}$	Constante de velocidade máxima definida (usada na retas)
k_1 , k_3	Constantes de relevância de cada linha no processo de seguimento de faixa
$Larg\_Faixa_{l_x}$	Largura da faixa numa determinada linha da imagem RGB
$\Delta encoders_x$	Variação entre valores de um *encoder* x entre duas iterações
L_{Eixo}	Comprimento do eixo do robô

Capítulo 1

INTRODUÇÃO

1.1 ENQUADRAMENTO E MOTIVAÇÕES

A navegação autónoma já não é novidade em ambientes controlados como na indústria ou transportes públicos sobre carris, porém, para esta se estender a ambientes não controlados, é necessário que os AGVs tenham uma boa perceção do ambiente que os rodeia para consequentemente tomar boas decisões. As aplicações da navegação autónoma no futuro serão numerosas e diversificadas.

A evolução dos parâmetros e do preço dos sensores e da capacidade de processamento contribuem para a melhoria da modelação do mundo real. O sensor *Kinect* é uma solução de baixo custo que gera visão artificial tridimensional. Foi criado para servir de interface em videojogos. Suas dimensões e seu peso tornam possível sua incorporação num AGV de pequeno tamanho. Um único sensor que fornece uma imagem RGB e outra de profundidade (RGB-D) servindo de base ao método de navegação e a deteção de obstáculos de um robô móvel criado.

Um AGV necessita ter conhecimento de sua posição atual e passada, recorrendo para isso á odometria. Devido a falta de precisão desta sem auxílio de giroscópio, torna-se inviável

definir uma posição alvo com coordenadas. Será usado o método ótico de faixas para método de navegação, implicando uma posição alvo do robô dinâmica.

1.2 OBJETIVOS

Este estudo teve por objetivo principal a conceção de um robô móvel diferencial com um *software* de interface gráfica incluindo um sistema de deteção e contorno de obstáculos baseado no sensor *Kinect*. Um trajeto alvo será definido por um sistema de faixas no solo, quando esta acabar, o robô tem de recomeçar o trajeto no sentido oposto. A posição absoluta do robô deve ser estimada com base na odometria. Todos os parâmetros do robô devem ser monitorizados. Durante o contorno de um obstáculo, a rota alternativa escolhida pelo robô deverá ser feita pelo lado que lhe permite ficar o mais perto da linha possível, sem haver colisões.

1.3 ESTRUTURA DO DOCUMENTO

Este documento, desenvolvido no âmbito da unidade curricular de Dissertação do 2º ano do Mestrado em Engenharia Industrial, ramo de Engenharia Eletrotécnica, encontra-se dividido em 6 capítulos e pretende relatar todo o trabalho desenvolvido neste projeto.

A introdução ao documento e ao tema é realizada no capítulo 1. Aí é efetuada a caracterização detalhada do problema a tratar e são explicitados os objetivos propostos para esta dissertação.

O estado da arte preenche o capítulo 2 deste documento. Apresentam-se aqui as soluções mais comuns de veículos autónomos já existentes, tal como métodos de navegação e localização, controlo de direção, arquiteturas de *software*, sensores e visão artificial.

No capítulo 3 serão apresentadas as partes integrantes do robô móvel usado no projeto. Serão aqui apresentados detalhadamente os diferentes componentes físicos (*hardware*) do robô e suas respetivas interações.

O capítulo 4 será dedicado ao *software*, concebido no âmbito deste projeto, para monitorização e controlo do robô. Serão apresentados os algoritmos que transmitem a

autonomia necessária ao robô para realizar as tarefas requeridas como saber onde está, saber para onde quer ir e conseguir contornar os obstáculos que possam estar no seu caminho.

Todos os testes efetuados e respetivos resultados encontram-se no capítulo 5. Serão aqui discutidos os resultados em função dos objetivos iniciais.

Finalmente, no capítulo 6, é possível encontrar as conclusões finais e sugestões de trabalhos futuros.

Capítulo 2

ESTADO DA ARTE

2.1 AGVs INDUSTRIAIS

Um AGV industrial é um veículo móvel autoguiado utilizado em ambientes que requerem transporte de produtos na produção e em armazéns. É programado para transportar materiais através de rotas definidas de recolha e entrega de produtos dentro de instalações de manufatura e de distribuição. Surgem como uma alternativa à solução clássica de ter empilhadoras e motoristas de empilhadoras a transportarem as matérias-primas e produtos no *shop floor* da fábrica [1].

Em muitas referências de fabricantes surgem associados outros acrónimos relacionados com o AGV. LGV (*Laser Guided Vehicles*), SGV (*Self-Guided Vehicles*) e AGC (*Automated Guided Carts*) são alguns exemplos. A designação de AGC deve-se ao seu pequeno tamanho, baixo custo e pouco peso quando comparado com os outros AGV [1].

Os AGCs são veículos autónomos leves com a função de transportar materiais dentro de uma fábrica. Estes utilizam fita magnética e sistemas óticos para se guiarem e contêm sistemas para evitar colisões.

São usados em muitas indústrias diferentes desde a química, farmacêutica, alimentar, manufatura, transporte e armazenamento ou automóvel.

2.1.1 VANTAGENS

As vantagens do uso de sistemas compostos por AGVs são:

- Flexibilidade — Os AGVs podem ser instalados nas fábricas com poucas ou nenhumas modificações na estrutura do *shop floor*. Além disso, estes permitem adaptar-se rápida e facilmente a mudanças do ambiente de operação e também a mudanças da sequência do transporte dos materiais. Por fim, os AGVs permitem ser customizados, facilitando a adaptação destes às funções particulares de uma determinada fábrica;
- Eficiência — O recurso a sistemas de AGVs permite que as operações fiquem mais eficientes. Dependendo da tecnologia envolvida no desenvolvimento do AGV, poderá existir monitorização e controlo dos AGVs o que possibilita saber onde estão os produtos a qualquer hora e, em caso de necessidade de recolher produtos para a manufatura ou para entregas, o sistema nunca se perde. Por outro lado, nos casos onde não existe localização do AGV, este andará sempre numa rota fixa permitindo saber onde se encontra com alguma facilidade;
- Redução de custos — Os AGVs reduzem os custos operacionais, dado que podem trabalhar continuamente sem interrupções durante todo o dia com pouca ou nenhuma supervisão.

2.1.2 MÉTODOS DE NAVEGAÇÃO

2.1.2.1 TRAJETÓRIAS FIXAS

Sistema Filoguiado

O sistema filoguiado, ilustrado em 2-1 [2], consiste na definição do percurso do AGV por intermédio de condutores elétricos embutidos no chão. Estes criam um campo magnético, devido à corrente elétrica sinusoidal que os atravessa, campo esse que é detetado por uma antena colocada no AGV [1, 3, 4].

É um sistema não flexível pois não permite que as rotas possam ser alteradas facilmente. Além disso, para realizar alterações é necessário implantar, novamente, condutores no chão e tal implica maiores custos. Por isso, não é usado em indústrias que necessitem de reconfigurar o *layout* diversas vezes.

Porém, é amplamente usado devido à sua simplicidade e robustez [1, 5, 6].

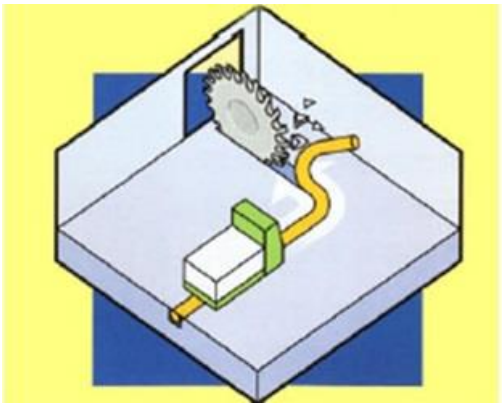

Figura 2-1 - Sistema Filoguiado.

Sistema de Faixas

Também é um método de navegação de trajetória fixa mas, em vez do uso de fios elétricos embutidos no chão, a trajetória é definida por uma fita magnética colada ao chão ou por linhas pintadas, como é representado na figura 2-2 [2]. Funciona de forma semelhante ao anterior método com um sensor apropriado a detetar as faixas.

A principal vantagem deste sistema sobre o anterior é que as rotas podem ser trocadas fácil e rapidamente, com muito menos custos associados e em menos tempo. Isto permite que seja um método mais flexível.

A principal desvantagem é que se a fita ficar danificada ou suja com a movimentação de pessoas ou objetos sobre ela, o AGV pode deixar de a identificar e, consequentemente, o AGV não conseguirá continuar o percurso [1, 5, 6, 3]. É recomendado para AGVs de baixo custo e de pequenas dimensões, também denominados de AGCs (*Automated Guided Cart*).

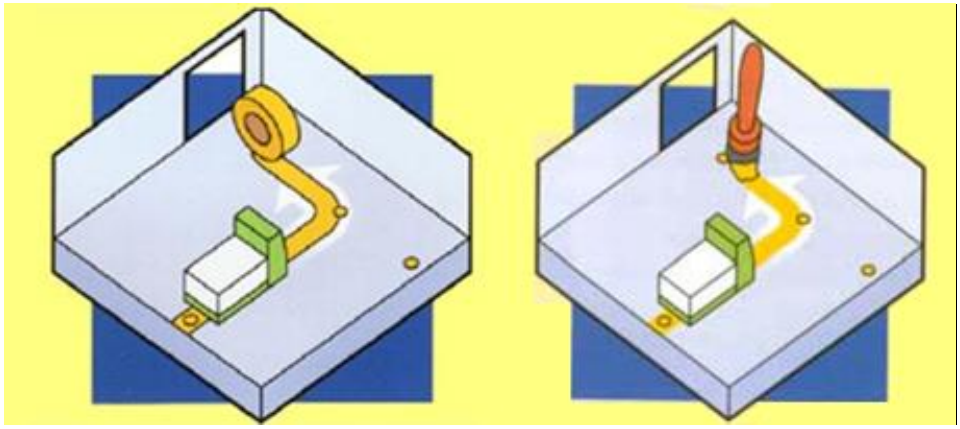

Figura 2-2 - Sistema de Faixas.

2.1.2.2 TRAJETÓRIAS DINÂMICAS

Sistema de Triangulação Laser

Neste sistema são colocados postes ou faróis refletores em colunas, paredes e noutros locais altos de fácil acesso ao laser usado pelo AGV, tal como é ilustrado na figura 2-3 [2]. O laser executa um varrimento rotativo à procura desses postes, que são usados como pontos de referência para a localização do veículo. É necessário detetar pelo menos 3 desses faróis para ser capaz de obter a sua localização. Para respeitar esta condição é necessário que haja um bom planeamento da disposição dos postes. Com esta condição cumprida, o AGV nunca se perde. A área é mapeada e guardada na memória do AGV [1].

É a forma de navegação mais fiável, segura e que garante maior precisão na posição do veículo. É também a solução com maior flexibilidade. Contudo, é também a solução mais cara [1, 5, 6, 3].

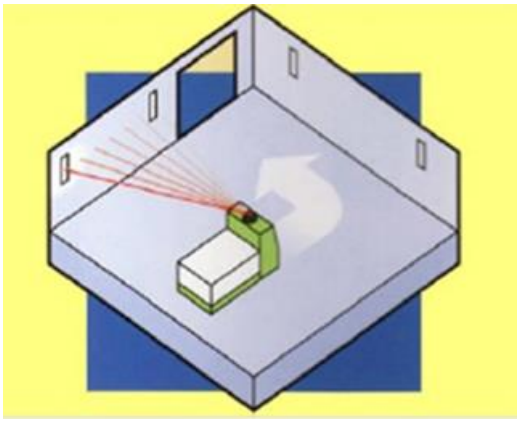

Figura 2-3 - Sistema de Triangulação Laser.

Sistema de Marcadores (ou Coordenadas)

Esta solução consiste na marcação no chão da fábrica de pequenos discos magnéticos espaçados entre si, como se pode observar na figura 2-4 [2]. No caso de marcadores magnéticos mais simples, não é possível aferir a posição absoluta do AGV. Contudo, é também possível guardar, previamente, as coordenadas dos marcadores numa base de dados fornecendo informação ao AGV da sua localização aquando da passagem deste por um marcador. Os marcadores poderão ainda informar a trajetória seguinte de forma a direcioná-lo para o próximo marcador pretendido. Caso o AGV se desvie da trajetória idealizada, por acumulação de erros, ele não irá encontrar o próximo marcador. Como consequência, o AGV ficará perdido. Por isso é que esta solução, normalmente, é usada em conjunto com um giroscópio que analisa as variações de direção.

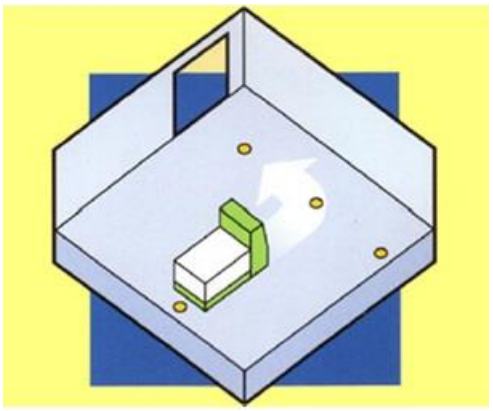

Figura 2-4 - Sistema de Marcadores.

Para tornar o sistema ainda mais flexível, é possível colocar marcadores num padrão de quadrados permitindo ao AVG conhecer sua posição absoluta como se pode observar na figura 2-5. Para aumentar a distância entre marcadores, será imperativo o uso de giroscópio [1, 2].

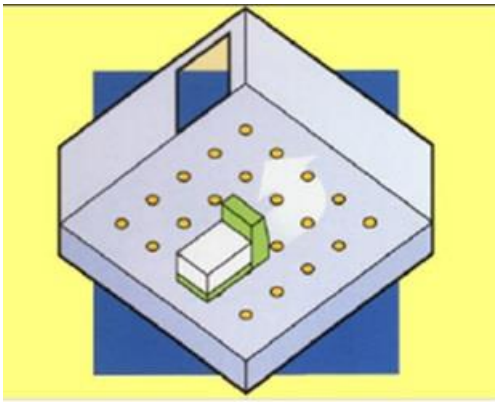

Figura 2-5 - Sistema de Coordenadas.

É, pois, uma solução bastante flexível permitindo que os caminhos sejam alterados muito fácil e rapidamente. Os custos são mais elevados do que a solução da fita magnética. Contudo, é a solução de trajetórias dinâmicas mais barata [1, 5, 6, 3].

2.1.3 CONTROLO DE DIREÇÃO

As opções de controlo de direção mais usadas em AGVs podem ser visualizadas na figura 2-6.

Na configuração do lado esquerdo, o AGV é composto por 2 rodas de tração e direção, uma à frente e outra atrás com movimento em qualquer direção (360º). Dispõe ainda de duas rodas livres de lado. Na configuração do meio é apresentada a configuração de

movimento diferencial onde as rodas estão lado a lado. A configuração do lado direito ilustra a configuração unidirecional que é composta por uma roda dianteira de tração e direção que roda sobre si própria. É chamado de controlo de tração triciclo pois contém um eixo traseiro fixo composto por duas rodas fixas [1, 5].

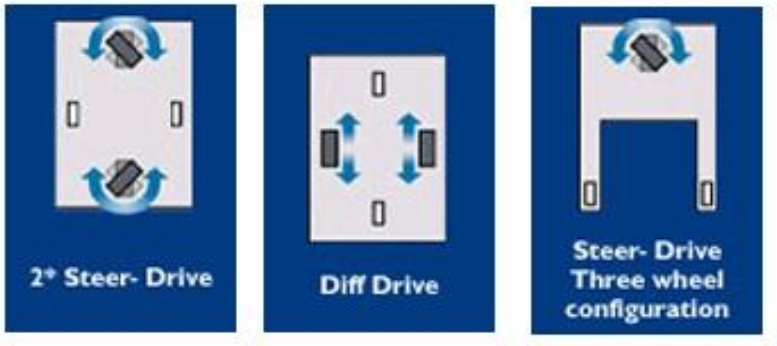

Figura 2-6 - Controlo Bidirecional [2].

Contudo existe outra configuração de robôs móveis mais recente que é composto por três (ou mais) rodas especiais todas motrizes e fixas na direção, desfasadas de 120º. O seu movimento não possui as restrições dos robôs mais convencionais mas tem por desvantagens um controlo mais complexo e fricção nas rodas [10]. A figura 2-7 [11] ilustra um robô omnidirecional.

Figura 2-7 – Robô omnidirecional.

É usado por exemplo no futebol robótico (*Medium Size League*) onde é necessário deslocações rápidas sem tempo para as manobras que teria de efetuar um robô diferencial para ajustar a sua direção.

O fato de nunca mais de uma roda estar alinhada com a direção do robô (num deslocamento linear) faz com que haja fricção nas rodas. O peso é um problema para esta configuração.

Algumas das movimentações do robô omnidirecional são apresentados na figura 2-8 [12] onde as setas brancas representam a deslocação do robô, as verdes o sentido e a velocidade de cada roda, as azuis as deslocações momentâneas das rodas, as laranjas a velocidade compensada de cada roda e finalmente o centro da rotação em amarelo.

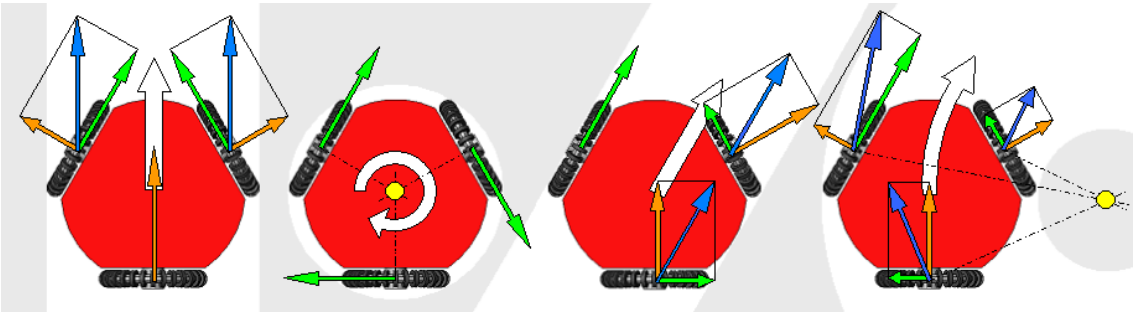

Figura 2-8 - Movimentações do robô em função do movimento de suas rodas.

2.1.4 SEGURANÇA

Para garantir a integridade do AGV e do meio envolvente (pessoas, produtos que transportam e máquinas), os AGVs dispõem de dispositivos de segurança. Os AGVs são construídos segundo a Norma Europeia para segurança de veículos autónomos (EN1525 - *Driverless industrial trucks and their systems*) [1].

- Deteção de colisões — Os AGVs possuem sensores de obstáculos (*Laser Range Finders*) que detetam objetos no seu campo de alcance e calculam a distância até estes. O AGV irá reduzir a velocidade ou até parar para prevenir colisões. Para além disso, podem usar para-choques que quando acionados garantem a paragem instantânea do AGV em caso de o sensor de obstáculos não ter atuado;

- Sinal Audio/Visual — Os AGVs são equipados com um conjunto de dispositivos de sinalização, tais como sinais de luzes piscando ou alertas sonoros que indicam o estado do AGV ou avisam os trabalhadores da sua presença;

- Controlo Manual — Os AGVs dispõem da possibilidade de controlo manual e de botões de emergência para a paragem brusca do AGV.

2.1.5 CONFIGURAÇÕES DE TRAJETOS

Existem diferentes configurações possíveis para sistemas que utilizem AGVs. As configurações são escolhidas e usadas em função do tipo de operações a realizar com estes, das condições das instalações e a localização das mesmas. A escolha de uma configuração vai influenciar na escolha do tipo de AGV. Existem configurações mais adequadas a determinados tipos de AGVs e, sendo estes modulares, também podem ser adaptados e construídos para melhor servir o trajeto delineado.

As configurações a seguir apresentadas são as mais comuns. Contudo, devido a uma das principais características do AGV, a flexibilidade, é possível adaptar as configurações e criar uma infinidade de configurações novas que se adequam às situações pretendidas [1, 7, 8].

2.1.5.1 *SINGLE LINE / BACK AND FORTH*

É o método mais simples, pois o movimento é feito apenas em linha reta. Pode ser visualizado na figura 2-7 [8]. O AGV necessita de poder andar para a frente e para trás. Só pode haver um AGV em cada linha. O espaço necessário é reduzido. A localização das estações pode ser ao longo da reta e nas extremidades da mesma [1].

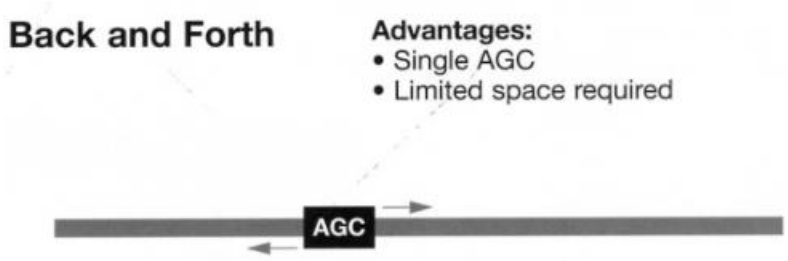

Figura 2-9 - *Single Line / Back and Forth.*

2.1.5.2 *SINGLE LOOP*

Como se pode ver na figura 2-8 [8], permite utilização de vários AGVs em simultâneo. Estes apenas andam para a frente num circuito fechado sempre à volta das estações. Elevada densidade de entrega [1].

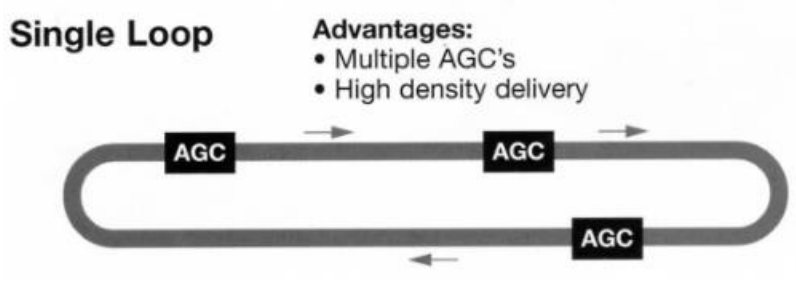

Figura 2-10 - *Single Loop.*

2.1.5.3 *DOUBLE LOOP*

Semelhante ao anterior. Entre *loops* existe um troço comum. Esta solução permite que o custo da utilização de fita magnética seja reduzido e é ideal para distâncias grandes. Porém, permite menor número de AGVs em simultâneo. Um exemplo da sua rota pode ser visto na figura 2-9 [1, 8].

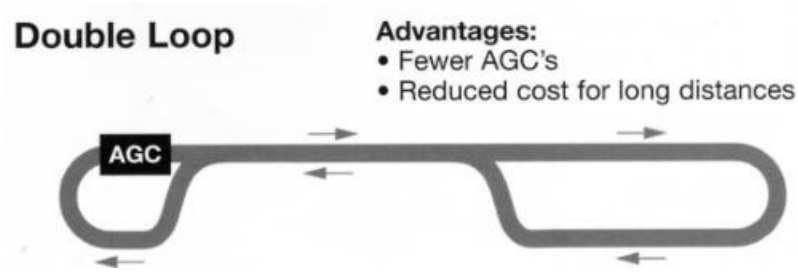

Figura 2-11 - *Double loop.*

2.1.5.4 *MULTIPLE LOOPS*

Solução composta por vários loops. Permite criar imensas bifurcações e responder a sistemas que exijam muitas estações. Requer controlo de tráfego e memorização de estações. A figura 2-10 [8] pretende ilustrar este tipo de rota [1].

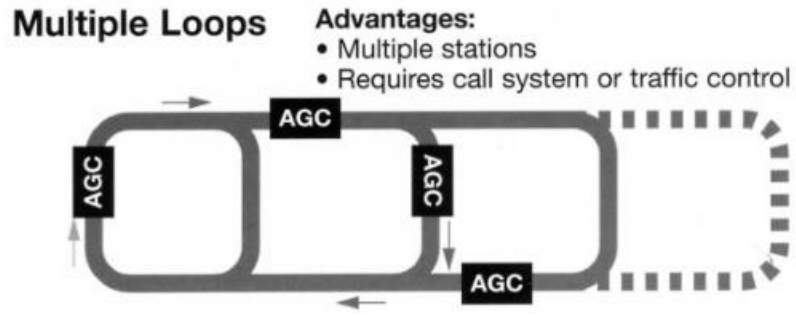

Figura 2-12 - *Multiple Loops.*

2.1.6 CONSTITUIÇÃO DE UM AGC

2.1.6.1 CONSTITUIÇÃO GERAL

Genericamente, verifica-se que um AGC deverá seguir fita magnética como método de navegação e deverá ser composto por vários módulos. Entre eles, pelo menos uma unidade de tração e de direção, uma unidade de comando/controlo, um sensor de obstáculos, um para-choques, baterias, sinalização luminosa e buzina. Uma solução para um AGC deverá englobar vários componentes como é ilustrado na figura 2-11 [1].

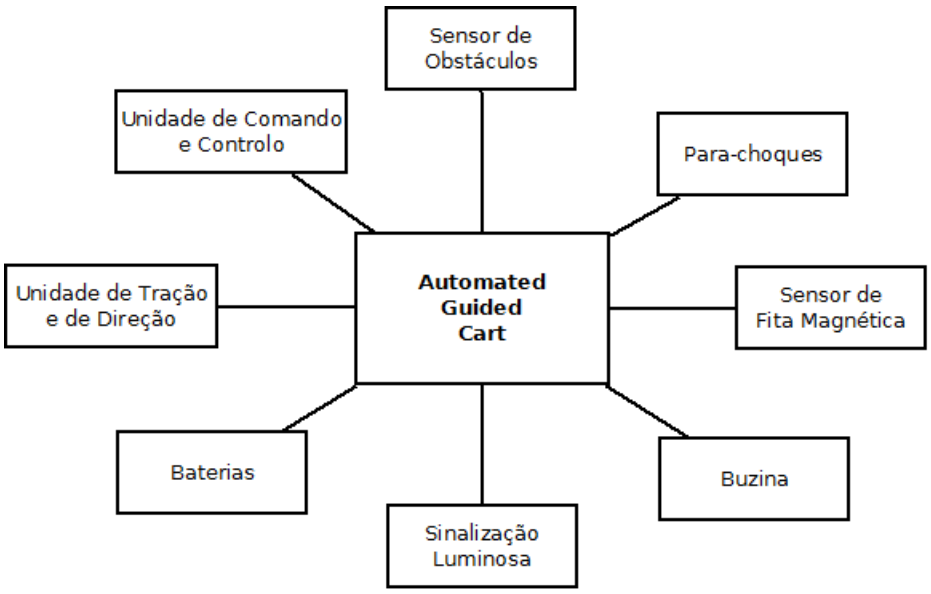

Figura 2-13 - Componentes constituintes de uma solução geral de um AGC.

2.1.6.2 SENSORES DE OBSTÁCULOS

Os sensores de obstáculos mais usados nos AGVs industriais são os *Laser Range Finders*. Estes sensores projetam um feixe de luz laser em intervalos regulares ao longo de um ângulo geralmente amplo num único plano em redor de si mesmos e aguardam o seu retorno aquando da reflexão num objeto. Uma vez que os intervalos angulares entre os feixes podem ser muito reduzidos, estes sensores são muito exatos. Para além disso, são geralmente dotados de um alcance relativamente grande. Para colmatar a limitação de que um *Laser Range Finder* apenas executa passagens num plano, estes sensores podem ser montados num motor que os faça inclinar de maneira a deslocar o seu plano de rastreio, permitindo recolher informação em três dimensões [37].

2.1.7 ESTUDO DO MERCADO

2.1.7.1 AGC

De seguida vai ser apresentada uma solução existente no mercado dos AGVs de baixo custo (ou AGCs), este é o AGV-A-32 EX, desenvolvido pela *Creform*. Esta é a solução adotada pela *Volkswagen Autoeuropa* nas suas linhas de produção devido à sua simplicidade de funcionamento e número de instruções possíveis reduzidas [1].

O A-32 navega recorrendo a fita magnética, movendo-se apenas para a frente com 2 velocidades definidas - velocidade baixa de 4 m/min e velocidade máxima de 24 m/min. Pode transportar até 350 kg de carga, sendo que para uma inclinação de até 3%, a carga total é de 200 kg. Utiliza uma bateria de 12 V.

No caso do A32 se desviar da fita, da tensão na bateria ser baixa ou da corrente ser excessiva no motor, ele dispõe de circuitos de segurança de forma a parar.

Como aparelhos de segurança dispõe de um sensor de obstáculos e um para-choques que pode acionar o estado de emergência que faz parar o AGV. Dispõe ainda de sinalização de aviso áudio e visual, nomeadamente, uma sirene e um sinal luminoso [1].

2.1.7.2 *LASER RANGE FINDERS*

Como a construção do AGC é modular, pode-se verificar individualmente cada um dos seus subsistemas. O sensor de obstáculos deverá ser capaz de detetar presença de objetos ou de pessoas de forma a evitar colisões. Após uma pesquisa de mercado, foram analisados os resultados. Estes podem ser consultados na tabela 2-1 [1, 9].

Tabela 2-1 – Tabela das características de sensores de obstáculos

Sensor	Fabricante	Tensão (V)	Alcance (m)	Peso (g)	Proteção	Tempo de Resposta (ms)	Preço (€)
PBS-03JN	*Hokuyo*	24	0,2 - 3	500	IP64	280	900
UBG-05LN	*Hokuyo*	24	0,1 - 5	260	IP64	210	1300
PX 22	*Sunx*	10 - 31	3	170	IP65	80	370
S 200	*Sick*	10 - 31	1,5	1200	IP65	80	1900

2.2 CARROS AUTÓNOMOS

O crescente avanço na utilização de veículos autônomos é consequentemente acompanhado pelo aumento de desafios para que estes se movam sem nenhuma intervenção humana em ambientes externos. Para que a locomoção autônoma em ambientes não controlados seja possível, é necessário a implementação de sistemas de localização, movimentação e perceção, entre outros [13].

2.2.1 MÉTODO DE NAVEGAÇÃO E LOCALIZAÇÃO

Para que a condução autônoma seja realizada, o veículo deve possuir um sistema de navegação eficiente. Esse sistema deve conduzir o veículo a um objetivo, através de ambientes conhecidos ou não e desviando-se de possíveis obstáculos. A navegação deve ser feita de acordo com uma arquitetura robótica, que leva em conta os sensores utilizados e como os dados são tratados. Desta forma, terá de executar três premissas fundamentais: perceber, planejar e agir [13].

Os gigantes da indústria automóvel já estão produzindo veículos equipados com sistemas semiautónomos para auxiliar na condução dos carros, e já anunciam para 2020 a produção de veículos totalmente autônomos [14].

O recente feito em destaque é da autoria da *Mercedes-Benz* em parceria com a *Nokia* que consistiu numa viajem de quase 100 km entre as cidades alemãs de Mannheim e Pforzheim quase sem intervenção humana. Durante o procedimento, um engenheiro ficou ao volante, para agir caso fosse necessária alguma correção de movimento, o que de fato ocorreu [14].

O carro está conectado ao serviço de mapeamento tridimensional da *Nokia*, o *Here*, servindo como base para condução autônoma.

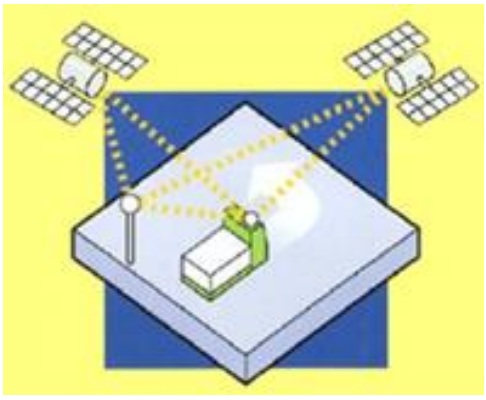

Figura 2-14 - Sistema de Localização GPS [2].

Além disso, dispõe de sensores de vários tipos que lhe permite sondar o ambiente em quase todo seu redor. Nomeadamente câmaras tridimensionais (IR), radares e sensores de ultrassom como ilustra a figura 2-15 [2] e 2-16 [14].

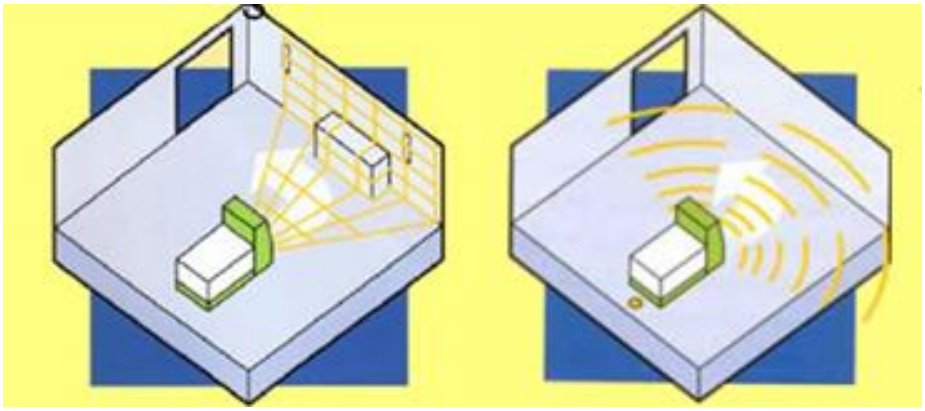

Figura 2-15 - Sistema de Visão tridimensional e de Ultrassom.

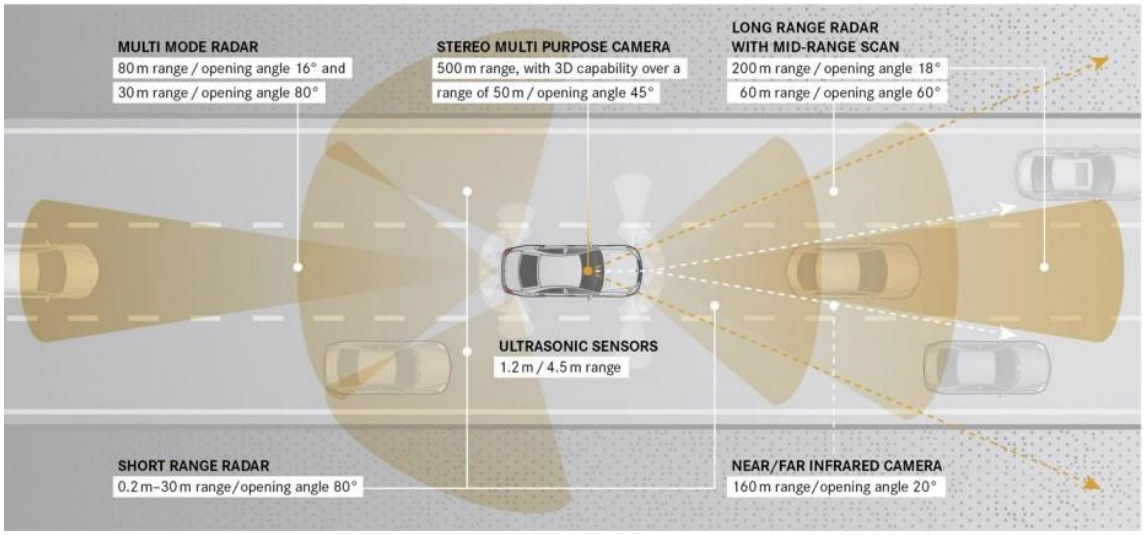

Figura 2-16 - Sensores usados e área coberta pelos Sensores / Visão Artificial.

A fusão destes sensores permitem ao carro perceber em que situação precisa se encontra para consequentemente tomar a respetiva decisão.

O processamento das imagens da câmara dianteira permitem a deteção de sinais, semáforos (e seu estado) e sinalização no solo. É possível calcular as velocidades e distâncias dos outros veículos. Os dados necessários a condução autónoma ficam assim acessíveis ao sistema. A figura 2-17 [14] apresenta um *frame* processado pelo sistema de identificação de sinalização e obstáculos.

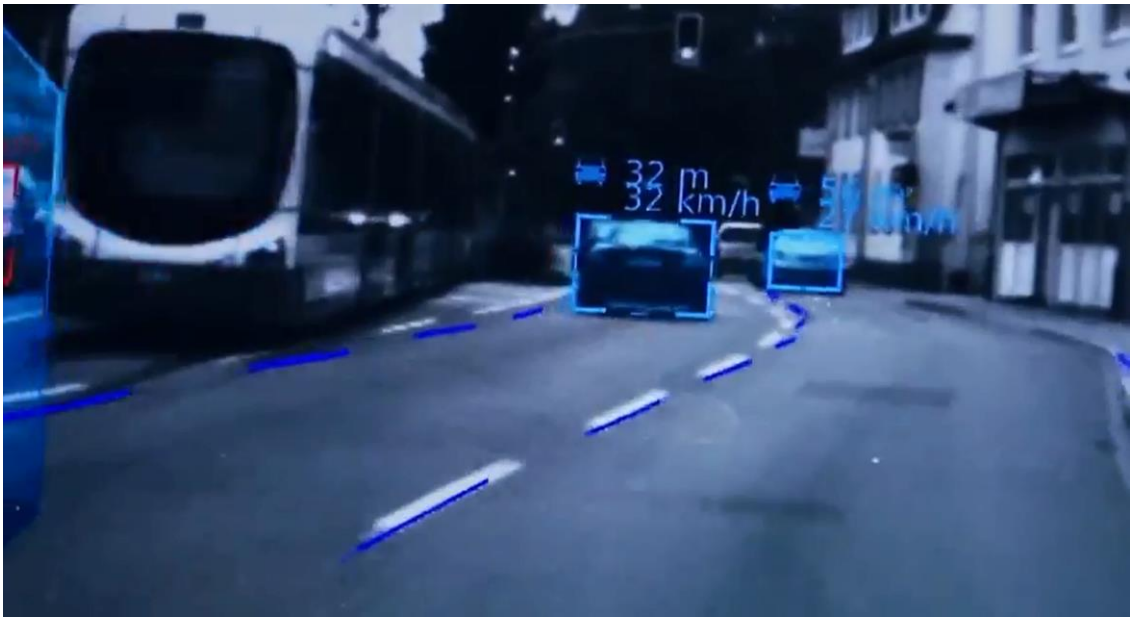

Figura 2-17 - Identificação de sinalização (solo, vertical e luminosa) e obstáculos.

Resumindo o carro sabe onde está e onde quer ir usando coordenadas GPS, e para decisões a curta distância, baseia-se nos sensores e câmaras.

2.2.2 VANTAGENS

A condução autónoma oferece diversas vantagens.

- Segurança — Suprimir erro humano, ou seja, reduzir tempo de perceção do obstáculo, reduzir tempo de reação, eliminar consequências da inatenção ou sonolência do condutor, melhorar a perceção do piso e das condições climatéricas, melhorar tempo de aviso de perigo aos veículos que nos seguem, melhorar distância de segurança, etc.
- Universal — Se o sistema não permite colidir com nada, se não permite cometer nenhuma infração e não necessita de nenhuma intervenção humana sem ser escolher o destino, então, seguramente crianças, idosos e pessoas com incapacidades motoras ou visuais poderiam usar carro. Não seria necessário carta de condução.
- Menores Consumos — Se o carro conhecer a distância que vai percorrer por inércia e sabe os metros que tem livres pela frente, não haverá acelerações fúteis compensadas por travagens. Haverá só travagens em casos inesperados, o que se vai traduzir numa descida dos consumos de combustíveis, pneus e travões. Ainda é ecológico por as mesmas razões.

- Ganho de Tempo — Deixa o condutor livre, ele é só mais um passageiro.
- Baixo Custo — Segundo a *Nissan,* o extra custará menos de 730 € [15] (obviamente não inclui o GPS e os sensores necessários).

2.3 FUTEBOL ROBÓTICO

Este subcapítulo é dedicado ao futebol robótico dado as semelhanças nos objetivos de seus robôs e do robô do presente trabalho. Ambos necessitam de métodos de localização, e métodos de deteção e contorno de obstáculos. Será apresentado mais detalhadamente o papel da visão omnidirecional.

2.3.1 MOTIVAÇÕES

A iniciativa *RoboCup* é um projeto de investigação e educação internacional com o objetivo de promover a investigação em Inteligência Artificial (Distribuída) e Robótica Inteligente [16].

O futebol serviu de inspiração para a criação do *RoboCup*, pois além de ser um jogo bastante popular em todo o mundo, o que aumenta o impacto mediático, tem um conjunto muito significativo de desafios para os investigadores, pelo facto de ser um jogo cooperativo em ambiente dinâmico [10]. O uso do futebol para problema padrão acaba por motivar os investigadores e despertar mais interesse nos observadores.

Como forma de promover a investigação na área, em 1997 foi lançado um objetivo de longo prazo: *"No ano de 2050, uma equipa de robôs autónomos humanoides, ser capaz de vencer a equipa campeã do mundo de futebol, num encontro disputado de acordo com as regras da FIFA."* [16, 18]

O fato de serem vários robôs a tentar, em conjunto, resolver um problema complexo, contribui num progresso da ciência e da tecnologia [17], nomeadamente robótica móvel, controlo de sistemas híbridos e não lineares, fusão sensorial e navegação.

2.3.2 LIGAS *ROBOCUPSOCCER*

Existem varias ligas *RoboCup*, dedicadas ao futebol (*RoboCupSoccer*):

- *Humanoid* — Na liga humanoide, robôs autónomos com aspeto humano e com sentidos semelhantes aos humanos jogam entre eles. São capazes de caminhar, correr, rematar a bola em equilíbrio, visualizar a bola, os outros jogadores, a baliza, as linhas do campo. A liga está dividida em 3 sub-ligas de acordo com o tamanho do robô: *Teen Size*, *Kid Size* e *Adult Size* [19]. Obviamente é a liga menos madura e que tem maior margem de evolução tal como os robôs humanoides.
- *Middle Size* (F2000) — Robôs da liga média não podem exceder os 50 cm de diâmetro e 80 de altura em equipas que não excedem os 6 robôs (4 em campo) com as regras da FIFA. Os robôs e o árbitro podem comunicar entre eles com redes sem fios. O campo tem dimensões aproximadas de 5 por 10 metros. Não existem sensores externos. A bola é laranja ou vermelha, o campo é verde [17, 19].
- *Small Size* (F180) — A liga dos robôs pequenos, como é conhecida, é uma das mais antigas. Seu foco está na cooperação de vários robôs inteligentes e no controlo dum ambiente altamente dinâmico com um sistema hibrido, centralizado e distribuído [19].
- *Simulation* — A liga de simulação esta focada na inteligência artificial e na estratégia coletiva. Existem duas sub-ligas: 2D e 3D [19].
- *Standart Platform* — Nesta liga os participantes usam robôs idênticos, o que permite a dedicação total dos investigadores ao *software*. Visão omnidirecional não é permitida forçando a tomada de decisões de negociar recursos de visão para a sua própria localização e para a localização da bola. O robô de base é o humanoide *Nao,* de *Aldebaran* [19].

Será dada maior atenção aos robôs da liga media por serem robôs com rodas e tendo como unidade de processamento um PC como o robô do projeto.

2.3.3 ROBÔS DE LIGA MÉDIA (*MIDDLE SIZE LEAGUE*)

2.3.3.1 CONTROLO DE DIREÇÃO

Inicialmente a maioria dos robôs desta liga eram diferenciais mas dada a importância do tempo em que o robô chega a bola, é vantajoso o uso de robôs omnidirecionais de 3 rodas desfasadas de 120º evitando manobras de acerto da orientação [10]. Esta é a

configuração típica usada quando se quer que o robô se mova em todas as direções, mas não sendo condição necessária para garantir a omnidirecionalidade pois os ângulos de esfasamento entre rodas podem ser diferentes de 120º, no entanto isso trás algumas desvantagens pois gera um desequilíbrio nos binários dos motores, o que pode levar a um esforço muito grande de um motor em relação aos outros dois. Também não é necessário que o robot seja feito com três rodas pois uma configuração com quatro rodas continua a garantir a omnidirecionalidade, permitindo obter-se mais força e velocidade ao efetuar trajetórias retilíneas com velocidade angular nula [10].

2.3.3.2 SENSORES E ATUADORES

Para poderem jogar o futebol os robôs têm de conhecer sua própria posição, a posição da bola, dos outros jogadores (da mesma equipa ou não). Para isso dispõem de sensores de proximidade laser ou ultrassom e duas câmaras, uma frontal, e outra que pode ser ou rotativa ou virada para um espelho convexo no topo para abranger todo o redor do robô (visão omnidirecional), como é possível ver na figura seguinte 2-18 [23].

Figura 2-18 - Robô *Middle Size League* da equipa *CAMBADA* da Universidade de Aveiro com visão omnidirecional.

O robô estima sua posição com recurso a odometria e giroscópio podendo atualizar sua posição conforme linhas do campo, balizas etc [17]. Eles podem também trocar informações sobre suas respetivas posições. Houve estudos sobre sistemas de localização mutua usando ultrassons aproveitando-se do facto que o guarda-redes consegue determinar facilmente sua posição absoluta dado sua posição no campo. A posição relativa entre robôs é derivada do tempo de viajem das ondas ultrassons, conseguindo depois conhecer a posição absoluta de cada robô por correlação [22].

O robô possui um mecanismo para segurar ("aspirar") a bola (mesmo efetuando uma marcha atrás) e outro para rematar [17], visíveis na figura 2-19 [20].

O árbitro transmite instruções por rede sem fios para assinalar as diferentes fases do jogo, saídas de bola etc.

Figura 2-19 - Robô (*Middle Size League*) da equipa *Tech United* da Universidade de Eindhoven.

2.3.3.3 VISÃO OMNIDIRECIONAL

A equipa *CAMBADA* da Universidade de Aveiro usa visão omnidirecional para uma quase total visualização do campo em simultâneo. A figura 2-20 [22] ilustra a imagem original da visão omnidirecional da câmara superior.

Figura 2-20 - Imagem original da visão omnidirecional do robô da equipa *CAMBADA*.

Após processar a imagem é possível reconstruir o campo de forma a ter um mapa que só seria possível ver de cima para corrigir erros de posicionamento. A figura 2-21 é a reconstrução do mapa-mundo [21].

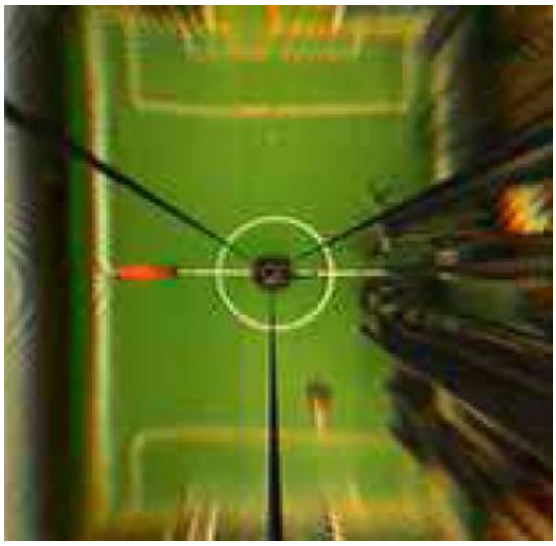

Figura 2-21 - Reconstrução do mapa-mundo.

Devido a estrutura do robô e sobretudo a do espelho, grande parte da imagem não é de nenhuma utilidade, é por isso usado uma mascara para selecionar a área da imagem a processar como é visível na figura 2-22 [21].

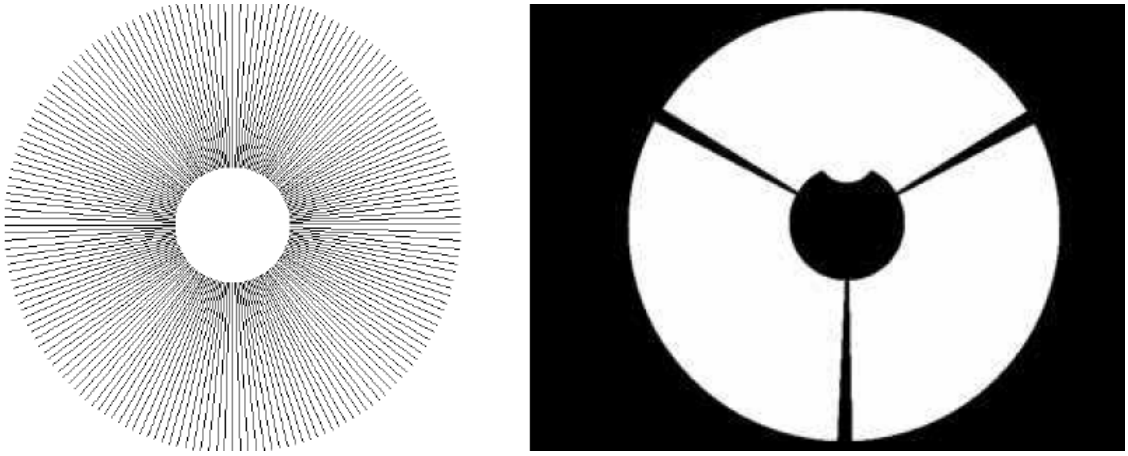

Figura 2-22 - A imagem da esquerda representa a área da imagem que abrange o espelho. Na direita a mascara usada no processamento. A área branca é a area processada.

A visão omnidirecional também é encarregue de deteção de obstáculos e da bola. É para isso necessário fazer um processamento de cor. O campo é verde, as linhas de jogo brancas, a bola laranja, e os robôs maioritariamente pretos (cor de obstáculos [17]) [21]. Os robôs são muitas vezes munidos de uma banda de cor para as equipas serem diferenciadas (*cyan* e *magenta*) [16]. A figura 2-23 ilustra a deteção e classificação de bola, obstáculos e linhas de jogo [21].

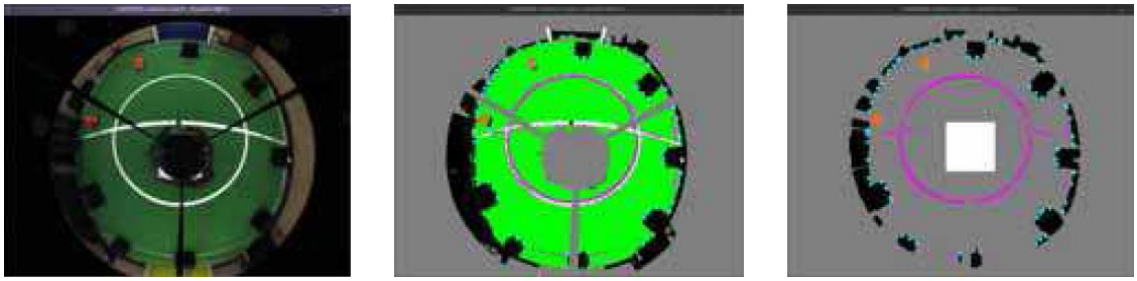

Figura 2-23 - Na esquerda a imagem original da visão omnidirecional. No centro, classificação de objetos (bola e jogadores) segundo a cor. Na direita, identificação de linhas de jogo usando mascara.

Quando a bola não está em campo é possível que só com o processamento de cor apareçam falsos positivos. Razão pela qual também é feito um processamento morfológico para deteção da bola [21]. A classificação de cada objeto será tomada em função dos dois processamentos (morfológico e de cor). A figura 2-24 [21] representa o processamento morfológico feita a visão omnidirecional.

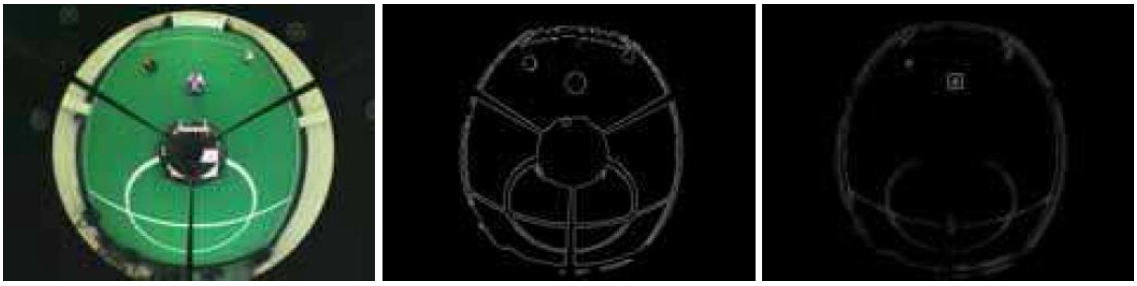

Figura 2-24 - Na esquerda a imagem original da visão omnidirecional. No centro a aplicação da transformada de Hough. Na direita: Fecho (Dilatação + Erosão) aplicada á transformada de Hough após remoção de mascara.

2.3.3.4 ARQUITETURA DO SOFTWARE

Os robôs no futebol robótico seguem basicamente um paradigma biomórfico (figura 2-25 [23]) centralizado numa unidade de processamento responsável pela camada de coordenação (alto nível) [23].

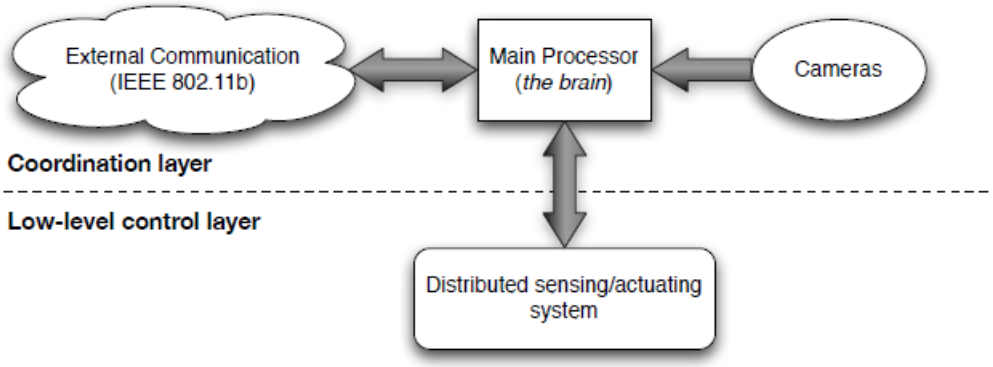

Figura 2-25 - Arquitetura biomórfica usada pelos agentes da equipa *CAMBADA*.

A unidade de processamento processa a comunicação com outros robôs e a visão com uma largura de banda elevada. A informação sensorial de baixa largura de banda e os comandos dos atuadores para controlar o robô são enviadas e recebidas, respetivamente, por meio de um sistema de deteção / acionamento de baixo nível [23].

A camada de coordenação trabalha em torno da RTDB, que contêm o estado do próprio robô, bem como uma cópia remota de um subconjunto dos estados dos outros. O protocolo usado para a rede sem fios é IEEE 802.11b [23]. A figura 2-26 representa a arquitetura do *software* dos robôs da equipa de futebol robótico *CAMBADA*.

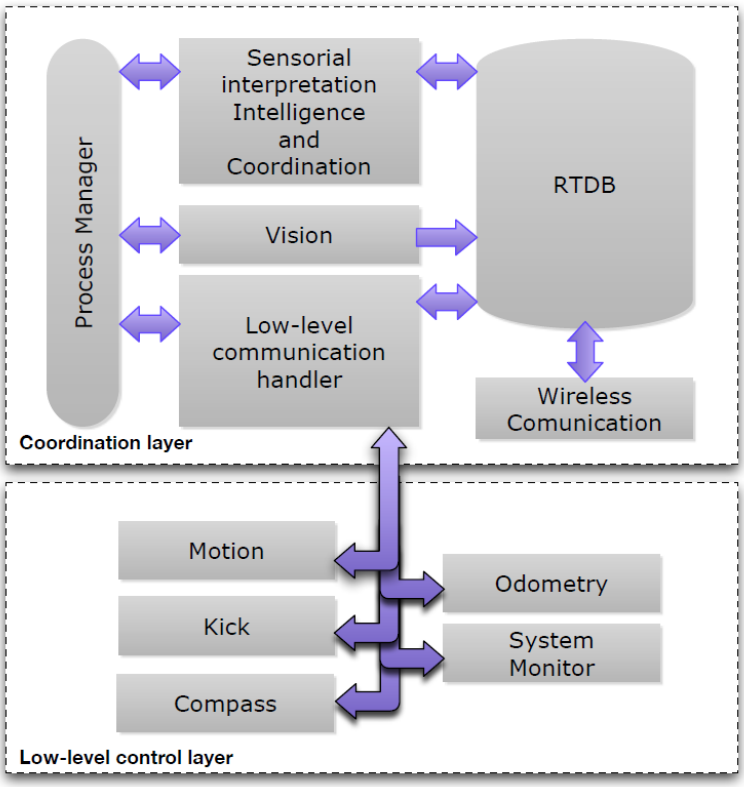

Figura 2-26 - Arquitetura do *software* em camadas dos robôs *CAMBADA*.

Capítulo 3

DESENVOLVIMENTO DO ROBÔ MÓVEL

Dados os objetivos impostos, foi necessário construir um robô móvel para testes e validação do sistema de deteção de obstáculos. Neste capítulo será apresentado o robô desenvolvido e utilizado ao longo deste estudo. O robô é um conjunto de blocos distintos ligados entre si numa estrutura móvel, estes são um computador, um comando, um *Xbox Kinect*, um conversor, uma placa de alimentação e controlo, uma bateria, e um *kit* de motores com *drive*. Após serem descritos os componentes individualmente e suas respetivas funções, serão apresentadas as ligações entre blocos e o fluxo de dados que suportam. No final o robô será testado manualmente com um comando para confirmar que tudo nele funciona como previsto.

3.1 BLOCOS DO ROBÔ MÓVEL

3.1.1 ESTRUTURA FÍSICA DO ROBÔ MÓVEL

Na fase de desenho da estrutura, foi dado prioridade á posição e orientação do sensor *Kinect*. Após uns primeiros testes verificou-se que a distância mínima que este conseguia detetar era aproximadamente 50 cm. Sendo assim, para o *Kinect* detetar o que está aos "pés" do robô, teria de estar posicionado na parte traseira da estrutura, por cima do monitor do computador. Para o resto dos componentes, optou-se por fazer um andar ajustável debaixo do andar do computador.

A estrutura física é constituída por 3 placas de acrílico cortadas e furadas a laser de 1 cm de espessura e varões roscados M8 com roscas de travão. As placas formam 3 andares ajustáveis e um subsolo, todas furadas para a passagem do varão roscado e para ligações entre os vários andares[1]. Os componentes do 1º andar foram fixados com parafusos M4 e roscas de travão. As duas rodas motrizes estão na parte traseira do robô enquanto a roda livre está a frente. No segundo andar ficaram duas mãozeiras laterais para facilitar seu deslocamento manual. As dimensões do robô são as seguintes:

Tabela 3-1 – Dimensões do Robô

	Dimensão [cm]
Altura	52
Largura	47
Comprimento	37

[1] O furo permitindo a passagem de ligações entre andares devem ser a 2,5 cm de diâmetro por causa das tomadas USB e ligações entre *drive* e motor.

A tabela seguinte indica qual a posição de cada bloco no robô.

Tabela 3-2 - Blocos do robô por Andar.

Andar	Blocos do robô
3º	Sensor *Xbox Kinect*
2º	PC Comando (tomada)
1º	Conversor TTL/USB *Drive* (*kit* MD25) Bateria Placa de alimentação
Subsolo	Motores (*kit* MD25) *Encoders* (*kit* MD25) 2 Rodas (*kit* MD25) 1 Roda livre

As figuras seguintes ilustram os diferentes andares do robô e os blocos residentes.

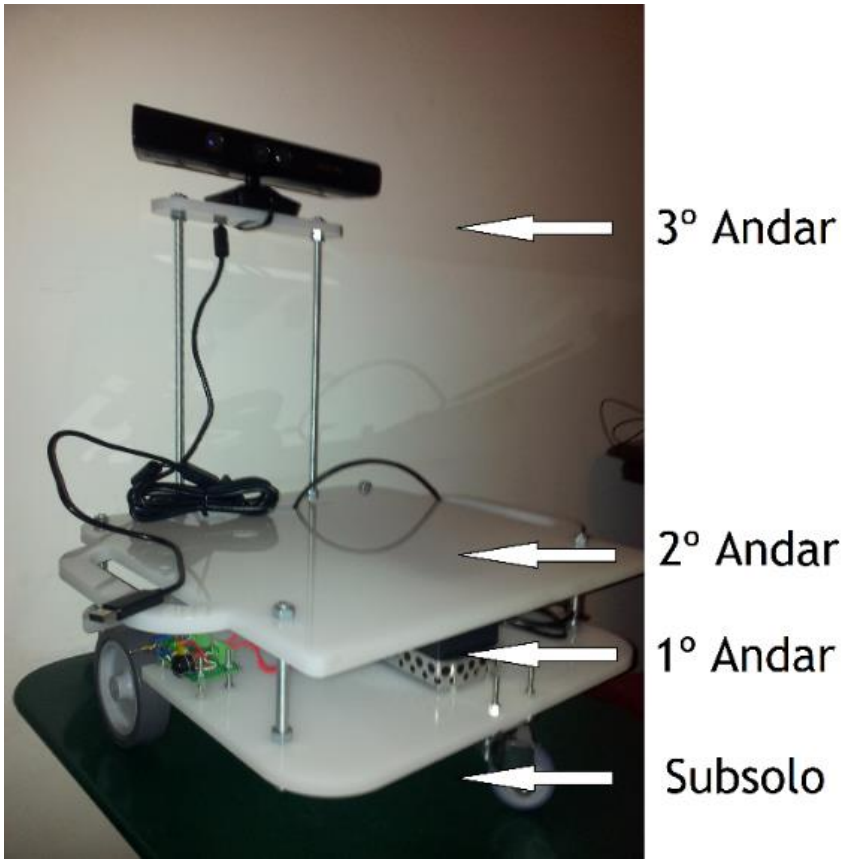

Figura 3-1 - Andares do Robô.

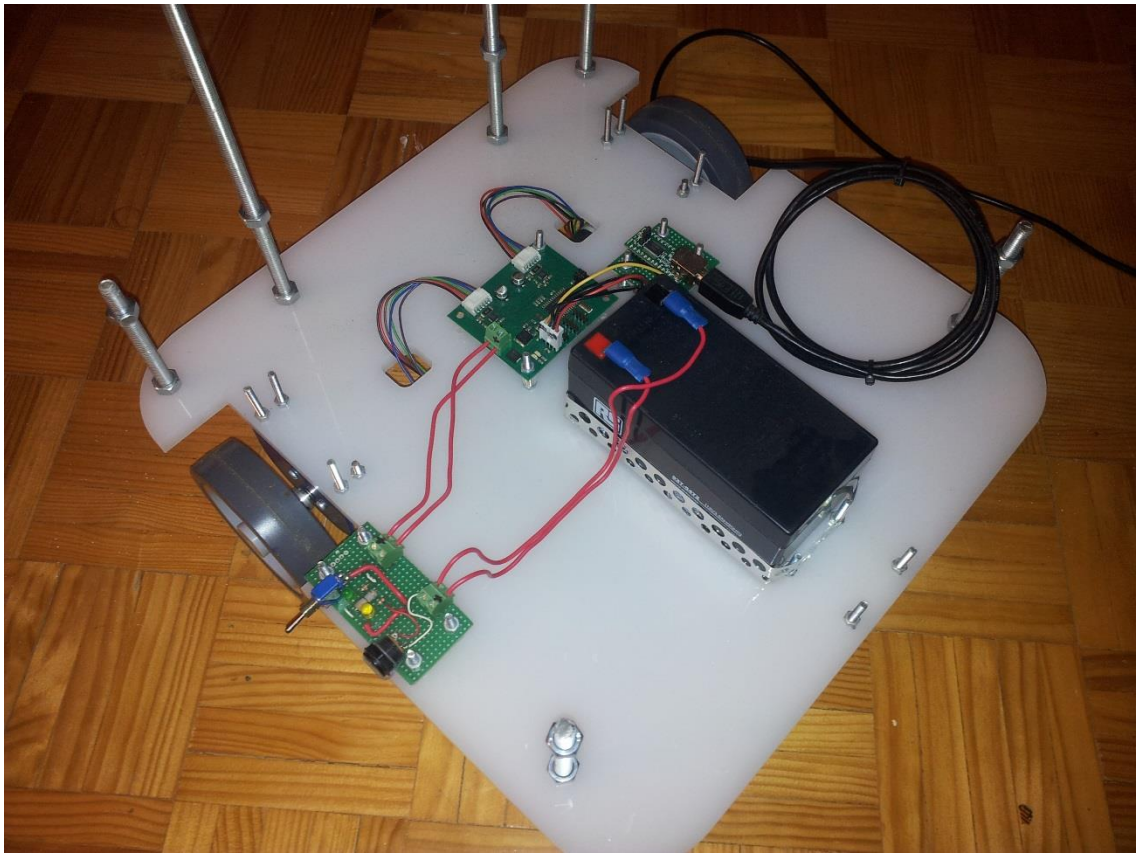

Figura 3-2 - 1º Andar do Robô.

Figura 3-3 – Subsolo do Robô.

Note-se que existem estruturas móveis comercializadas como o *Eddie Robot Platform* da *Parallax* [24].

Foto do Robô do projeto

Desenho em 3D do Robô "Eddie"

Figura 3-4 – Robô do projeto e robô "Eddie" comercializado pela *Parallax*.

3.1.2 *KIT* MD25

O *kit* MD25 é composto por um duplo *drive* de motores, dois motores DC munidos de *encoders*, duas rodas e dois suportes para estas. Embora o *drive* funcione com 5 V, ele deve ser alimentado a 12 V para a alimentação dos motores e *encoders*. O *drive* tem seu próprio regulador de 5 V, significa que se o drive estiver ligado ao PC via USB, consegue comunicar sem estar alimentado [25].

Figura 3-5 – *Kit* MD25 completo.

A figura 3-6 apresenta o drive e as respetivas ligações.

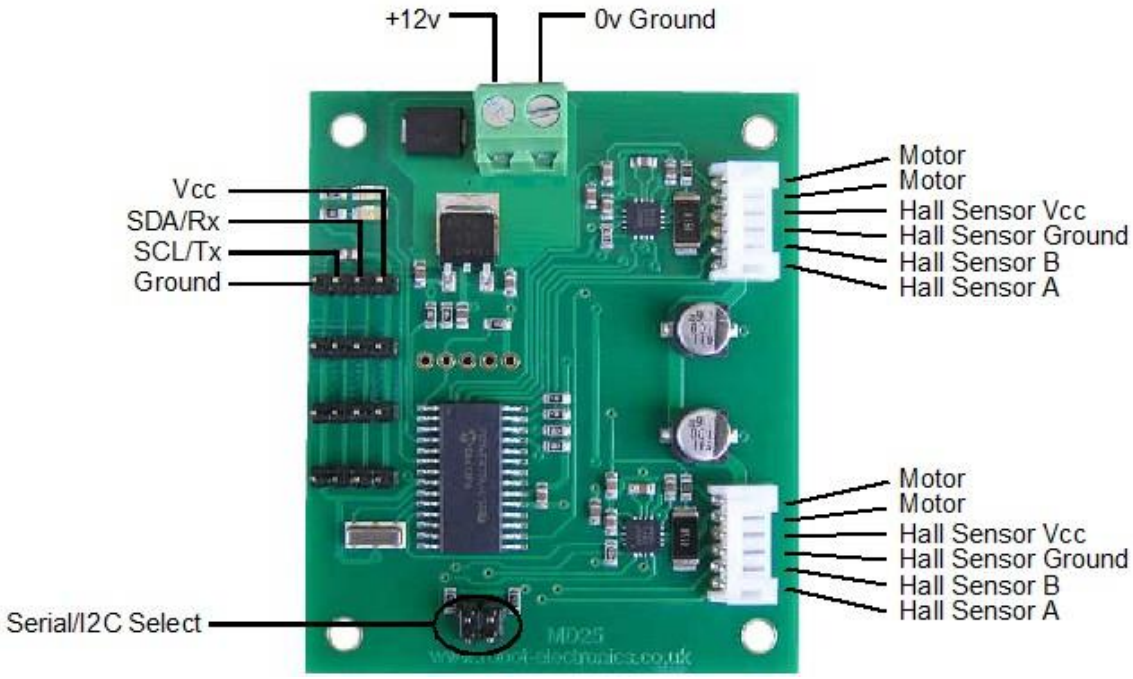

Figura 3-6 – *Pinout* do *drive* MD25.

As ligações para o motor de cada *drive* são:

- 2 para alimentação dos motores no sentido e velocidade desejada (saídas);
- 2 para tensões Vcc e GND dos sensores de efeito Hall (entradas);
- 2 dos valores dos sensores de efeito Hall A e B (entradas).

Os sensores de efeito Hall permitem saber qual foi o valor e o sentido do deslocamento de cada roda. O *drive*, encarrega-se de guardar e atualizar a distancia total que percorreu cada roda. A cada volta completa que a roda gira, o *encoder* soma 360 unidades se for para a frente, subtrai 360 se for para trás. O diâmetro da roda é de 100 mm. Os motores do *kit* são o motor DC EMG30 ilustrado na figura 3-7 [26]. Os parâmetros do motor encontram-se na tabela 3-3 [26].

Figura 3-7 – Motor EMG30 do *Kit* MD25.

Tabela 3-3 - Parâmetros do Motor EMG30

Caixa redutora	30:1
Velocidade mínima	1,5 rpm
Velocidade máxima	200 rpm
Tensão nominal	12 V
Binário nominal	1.5 kgf.cm
Velocidade nominal	170 rpm
Corrente nominal	530 mA
Velocidade sem carga	216 rpm
Corrente sem carga	150 mA
Corrente rotor bloqueado	2.5 A
Saída nominal	4.22 W
Contagens do *encoder* por volta inteira	360

Neste caso o *drive* comunica com tensões TTL, 38400 bps, 1 *start bit*, 2 *stop bits*, *no parity* (Vcc, GND, Tx e Rx), logo, será necessário converter para USB para comunicar com o PC. Esse bloco conversor não faz parte deste subcapítulo.

O *drive* para os motores quando não houver comunicação durante mais de 2 segundos por segurança. Antes de cada byte enviado para o MD25, será necessário enviar um byte de sincronização para avisar o *drive* que queremos transmitir-lhe um pedido.

A tabela seguinte apresenta a lista de comandos do *drive* MD25 [27].

Tabela 3-4 - Comandos do *drive* MD25

Comando	Nome	*Bytes* recebidos por MD25	*Bytes* enviados pelo MD25	Descrição
0x21	GET SPEED 1	2	1	Retorna a velocidade instantânea de motor1
0x22	GET SPEED 2	2	1	Retorna a velocidade instantânea de motor2
0x23	GET ENCODER 1	2	4	Contagem de *encoder* (motor1), 4 *bytes* retornados, *byte* alto primeiro
0x24	GET ENCODER 2	2	4	Contagem de *encoder* (motor2), 4 *bytes* retornados, *byte* alto primeiro
0x25	GET ENCODERS	2	8	Retorna 8 *bytes* - contagem encoder1, contagem encoder2
0x26	GET VOLTS	2	1	Retorna o nível da bateria
0x27	GET CURRENT 1	2	1	Retorna corrente puxada pelo motor1
0x28	GET CURRENT 2	2	1	Retorna corrente puxada pelo motor2
0x29	GET VERSION	2	1	Retorna a versão do *software* MD25
0x2A	GET ACCELERATION	2	1	Retorna nível de aceleração atual
0x2B	GET MODE	2	1	Retorna modo atual
0x2C	GET VI	2	3	Retorna nível da bateria, corrente motor1 e corrente motor2
0x31	SET SPEED 1	3	0	Selecionar velocidade1
0x32	SET SPEED 2 / TURN	3	0	Selecionar velocidade2/ *Turn*
0x33	SET ACCELERATION	3	0	Selecionar aceleração
0x34	SET MODE	3	0	Selecionar modo
0x35	RESET ENCODERS	2	0	Inicializar ambas contagens dos *encoders*
0x36	DISABLE REGULATOR	2	0	Saída não alterada por *feedback* dos *encoders*
0x37	ENABLE REGULATOR	2	0	Saída alterada por feedback dos encoders

0x38	DISABLE TIMEOUT	2	0	MD25 continua a entregar carga aos motores continuamente
0x39	ENABLE TIMEOUT	2	0	MD25 para os motores depois de 2 segundos sem comunicação

Por exemplo, para ler o nível da bateria [27]:

De PC para *Drive*:

0x00 – Byte de sincronização

0x26 – Comando de nível de bateria

De *Drive* para PC:

0x77 – Byte retornado, que em decimal é 119, logo é 11,9V.

Os diferentes modos do *drive* selecionam se definimos a velocidade para cada roda ou se optamos por escolher uma velocidade linear (*Speed*) e uma velocidade rotacional (*Turn*). Definem também se a largura de banda das velocidades é de -128 a 127 ou de 0 a 255. Neste caso foi usado o *drive* no modo 3. Segue-se a tabela dos modos.

Tabela 3-5 - Modos do *drive* MD25

Gama de valores	-128/127	0/255
Sem *Turn*	1	0
Com *Turn*	3	2

Logo, os movimentos do robô móvel em função do *Speed* e do *Turn* são os seguintes:

Tabela 3-6 - Movimentos do Robô em função de *Speed* e *Turn*

Speed / ***Turn***	<0	0	>0
<0	Não usado	Rotação para Esquerda	Avança, ligeiramente para Esquerda
0	Recua	Parado	Avança
>0	Não usado	Rotação para Direita	Avança, ligeiramente para Direita

O *Turn* é um fator que altera a direção do robô adicionando se a um motor e subtraindo-se a outro. O motor 1 está do lado esquerdo do robô, então se a velocidade ($velocidade1$) é de 5, e o *Turn* de 5, segundo as velocidades de cada roda [27]:

$$\begin{cases} velocidade1 = velocidade + turn \\ velocidade2 = velocidade - turn \end{cases} \quad (1)$$

$$\Leftrightarrow \begin{cases} velocidade1 = 5 + 5 = 10 \\ velocidade2 = 5 - 5 = 0 \end{cases} \quad (2)$$

A roda da direita está parada enquanto a da esquerda anda para a frente, logo o robô vira a direita. O *Turn* positivo implica uma curva para a direita. Para travar ou parar o robô, terá de se colocar a zero o valor da velocidade e do *Turn*.

A bateria alimenta o *drive*, e consequentemente os *encoders* e os motores caso o interruptor da placa de alimentação estiver ligado. O computador envia continuamente comandos ao *drive* para controlar as deslocações do robô como o *Speed* e o *Turn* enquanto este retorna, também em tempo real, parâmetros medidos tal que a velocidade medida, a distância angular percorrida, tensão da bateria, etc. para poder monitorizá-los. O *drive* recebe ordens do PC para motores e envia parâmetros medidos no andar de baixo nível para o PC. A figura 3-8 ilustra a função do drive MD25.

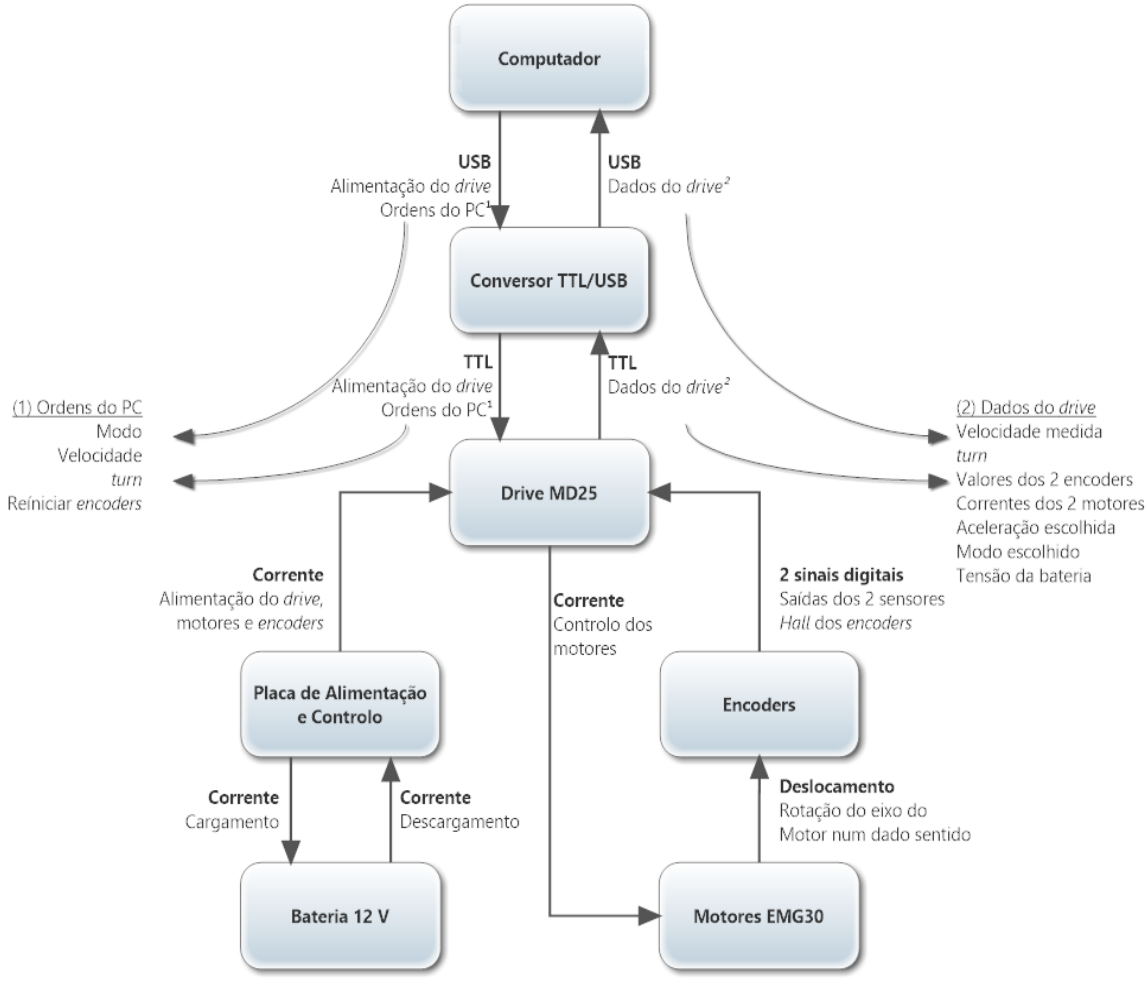

Figura 3-8 – Blocos que interagem com o *drive*.

3.1.3 PLACA DE ALIMENTAÇÃO E CONTROLO ON/OFF

A placa de alimentação e controlo possui um interruptor (*toggle,* ON/OFF), uma tomada fêmea de transformadores AC/DC, e 2 LEDs.

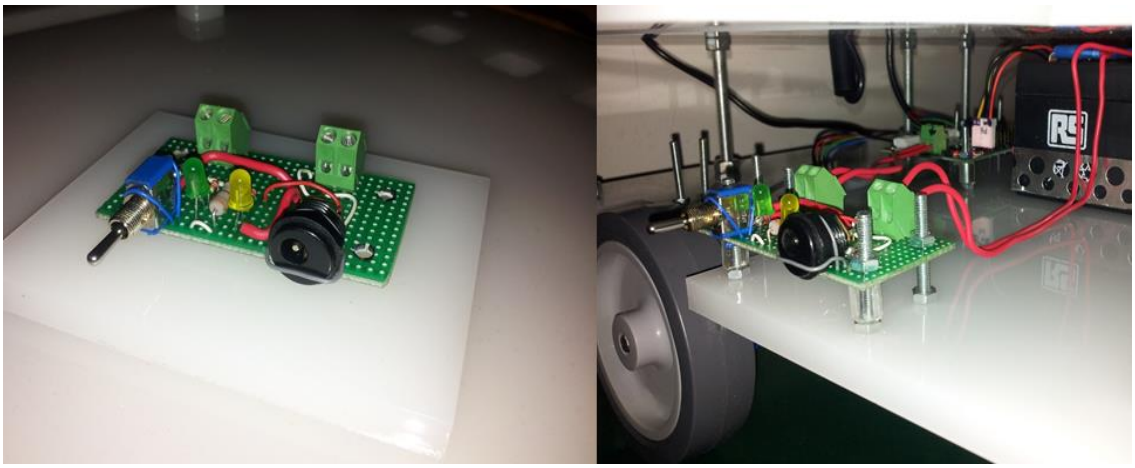

Figura 3-9 - A Placa de alimentação e Controlo.

Este bloco permite ligar ou cortar a alimentação dos motores e *encoders* dado que o PC pode estar alimentando o resto se estiver ligado. Será a maneira mais rápida de parar o deslocamento do robô em caso de emergência. O estado ON está sinalizado pelo LED verde. A tomada fêmea de transformadores AC/DC é usada para o carregamento da bateria do robô. O estado de carregamento é sinalizado pelo LED amarelo. O valor do nível da bateria será monitorizado no PC. Abaixo encontra-se o circuito da placa.

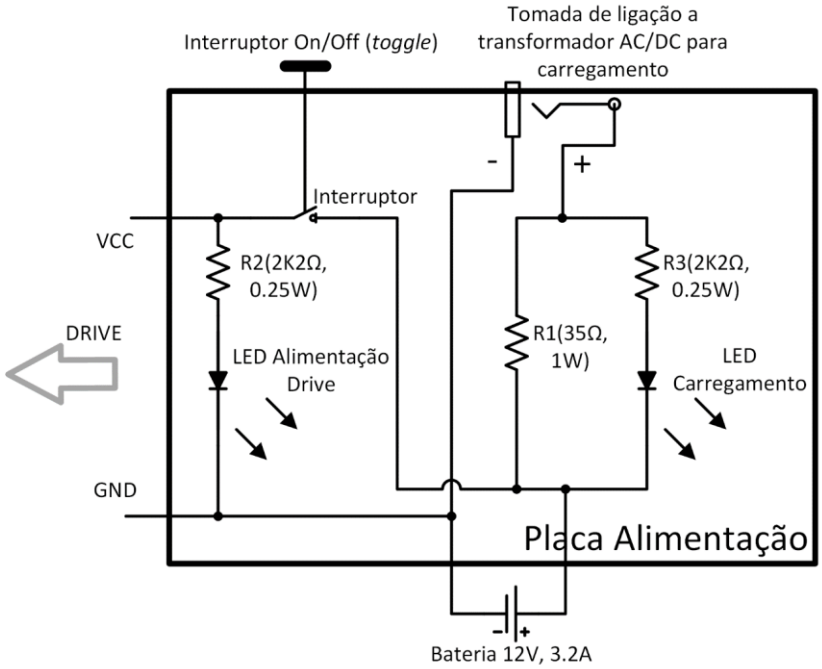

Figura 3-10 - Circuito de Placa de Alimentação e Controlo.

3.1.4 BATERIA

A fonte de energia dos motores e *encoders* é uma bateria de 12 V com as seguintes características [28]:

Tabela 3-7 - Propriedades da Bateria

Capacidade	3.2 Ah
Tipo	Ácido-chumbo
Dimensões	66 x 135 x 67 mm
Tipo Ácido-chumbo	AGM
Tensão Nominal	12V
Gama de Temperatura de Funcionamento	-20 → +60°C
Peso	1.3 kg
Aplicações Típicas	Iluminação de emergência, equipamentos de segurança

Recorda- se que o *drive* só precisa de estar em comunicação com o PC para estar ligado. É alimentado pelos 5 V do conversor que por sua vez é alimentado pela porta USB do PC.

Figura 3-11 - Bateria do Robô Móvel.

3.1.5 CONVERSOR TTL/USB

O conversor tem por função "traduzir" comunicações para o PC e o *drive* poderem comunicar. A comunicação é bidirecional. O componente usado para fazer a conversão de TTL para USB (e vice versa) é o UM232R da *FTDI chip* [29]. É no entanto necessário fazer certas ligações no conversor para configurá-lo para esse efeito. A figura seguinte apresenta o circuito e a configuração dos *jumpers*.

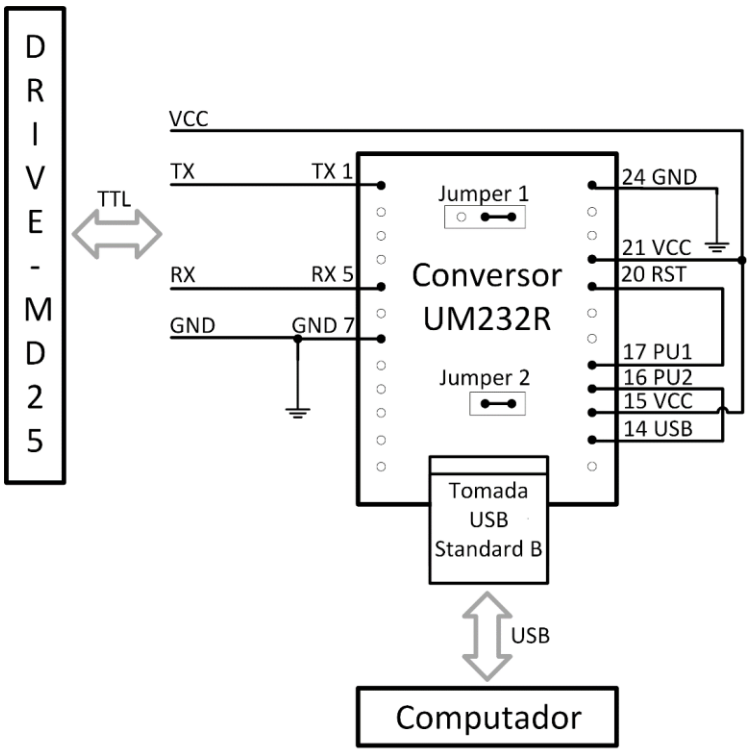

Figura 3-12 - Circuito e Configurações do Conversor UM232R.

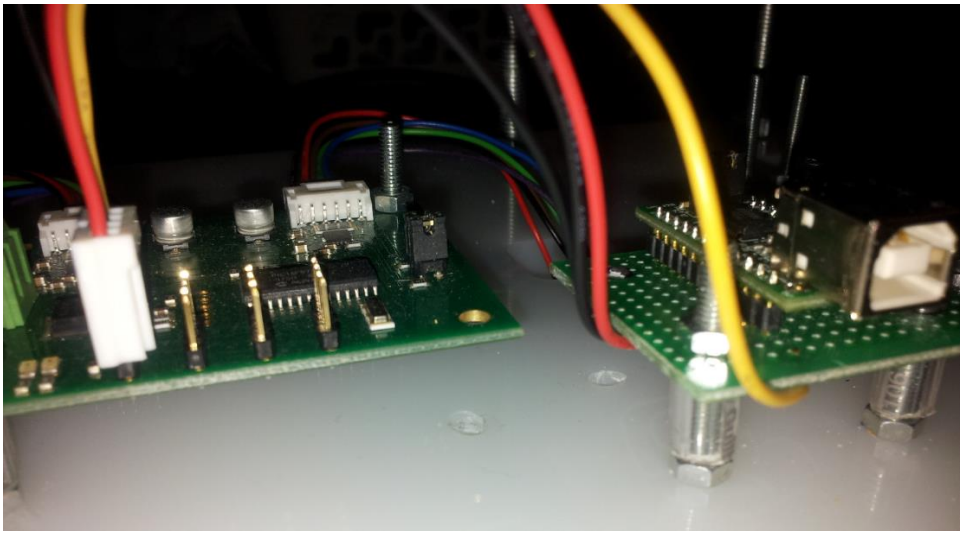

Figura 3-13 - Conversor ligado ao *drive*.

3.1.6 COMPUTADOR

Dado a necessidade do dispositivo *Kinect* ser reconhecido pelo computador, teremos de trabalhar com o sistema operativo *Linux*. Para contornar o problema, foi usado o *VMware Player* que permite simular um sistema operativo virtual dentro de um sistema operativo real. A máquina virtual usada no projeto é o *Ubuntu* 10.04 LTS (*the Lucid Lynx*) baseado em núcleo *Linux*.

O ambiente de desenvolvimento integrado escolhido para o projeto é o *Lazarus* desenvolvido para o compilador *Free Pascal*. A linguagem de programação é *Object Pascal*.

Uma grande vantagem de usar o *Lazarus* é a biblioteca *5dpo* desenvolvido pela equipa de futebol robótico do mesmo nome (FEUP, Faculdade de Engenharia da Universidade do Porto). Esta biblioteca contém vários ficheiros importantes para o desenvolvimento deste projeto, tal que a interação com o *Kinect*, a comunicação entre blocos, ou para interação do PC com um *joystick* analógico, entre outros [30].

Figura 3-14 - Robô da equipa de futebol robótico *5dpo* (*Medium Size League*), da FEUP, Faculdade de Engenharia da Universidade do Porto.

Dado que o comando usado neste projeto não possui botões analógicos, não se pode usar o ficheiro da biblioteca *5dpo* dedicado a esse propósito. Em alternativa usou-se um emulador de teclados para comandos que converte entradas no comando em entradas do teclado. A última versão gratuita disponível é *JoyToKey* (Versão 3.7.4) [31].

O bloco do computador recebe dados do comando, do *Kinect* e do *drive* enquanto envia ordens de movimentação do robô ao *drive* e comandos ao *Kinect* para ajustar sua orientação.

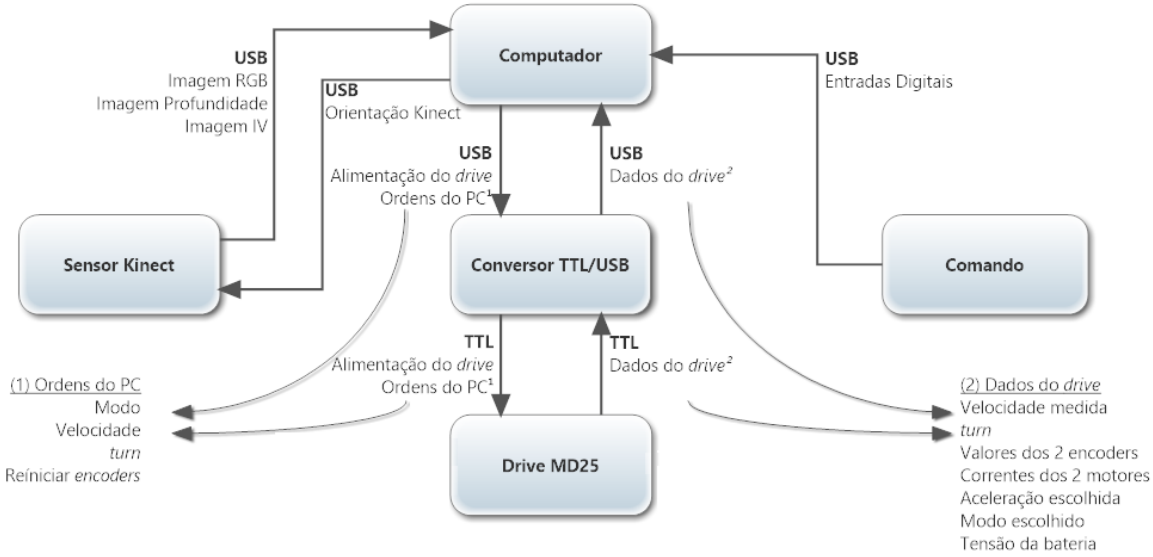

Figura 3-15 - Entradas e saídas do computador.

Estes requisitos todos cumpridos proporcionam a base para desenvolver programas para o robô. O objetivo desses programas serão de controlar o robô com o comando num primeiro tempo, para a seguir o robô tomar suas próprias decisões em função da rota ideal[2] e dos obstáculos que nela se encontram.

3.1.7 COMANDO SEM *JOYSTICKS*

O comando usado para controlo manual do robô tem 14 botões digitais e comunicação USB. Este comando não possui *joysticks*, isto é entradas analógicas. A atribuição da tecla que simula cada botão do comando é definido no programa *JoyToKey* [31]. Os botões servirão para avançar, recuar, girar sobre ele próprio em ambos lados, e aumentar e reduzir o *Speed* e o *Turn*. A vantagem neste caso de usar o comando é o comprimento do cabo dado que o PC está incorporado na estrutura móvel. Outro teclado teria servido para o efeito.

[2] Entende-se rota ideal como a rota que se percorreria sem obstáculos.

Figura 3-16 - Comando usado no projeto.

3.1.8 SENSOR *XBOX KINECT*

O Sensor *Xbox Kinect* foi desenvolvido para a *Xbox 360* juntamente com a empresa *Prime Sense* com o objetivo de substituir os comandos e *joysticks* usados nos videojogos. Este foi lançado para o mercado em novembro de 2010. O *Kinect* é composto por uma câmara RGB (VGA 640x480, CMOS), um sensor de profundidade que permite sondar o ambiente a sua volta em 3 dimensões, microfones, um acelerómetro em 3 dimensões, e um motor na base para alterar a orientação das câmaras [32][33][34].

Figura 3-17 - Sensor *Xbox Kinect*.

O sensor de profundidade consiste numa fonte de infravermelhos e num laser que projeta um padrão de pontos que é recebido de volta por um sensor CMOS monocromático de infravermelhos [35]. Esta câmara de IV recebe o padrão refletido e converte intensidades em distâncias. A sensibilidade do Kinect não depende em nada da luminosidade do ambiente, pode trabalhar perfeitamente com um ambiente sem luz. A gama de valores medidos pelo *Kinect* não é linear com a distância. As distâncias detetáveis estão entre os 45 centímetros e

os 6 metros com uma resolução de 1 centímetro no eixo z enquanto nos restantes eixos é em milímetros. Cada *frame* gerado pelo sensor de profundidade possui resolução VGA (640 x 480 *pixels*) contendo valores de profundidade de 11 *bits* que suportam 2048 níveis de sensibilidade. O *frame rate* da saída é 30 Hz. O sistema tem seu próprio processador e *software* [35]. A figura 3-18 apresenta os componentes internos do *Kinect* e o respetivo fluxo de dados [33].

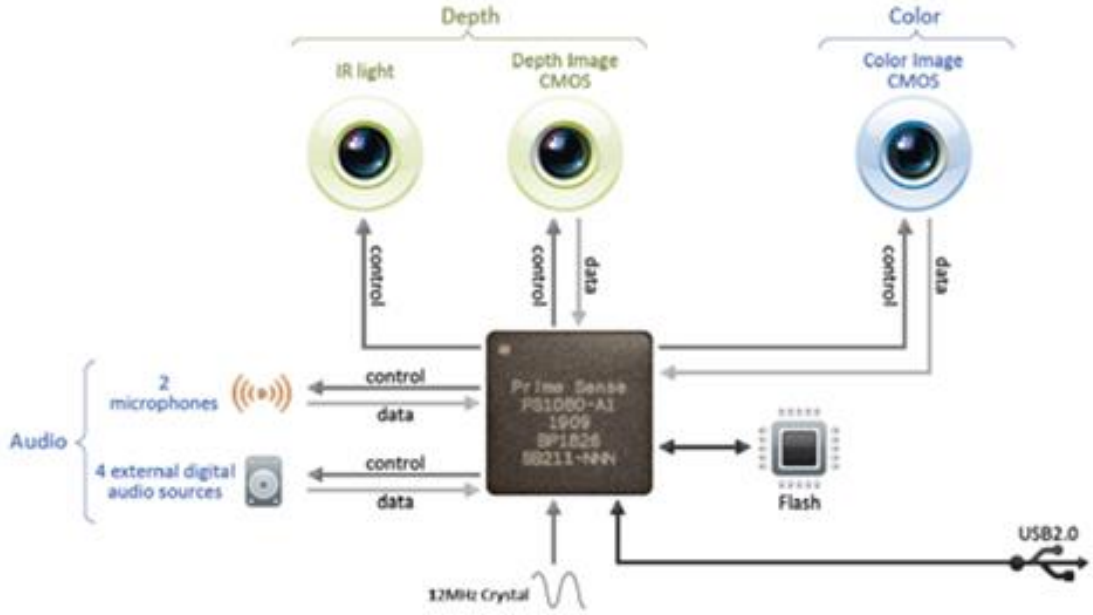

Figura 3-18 - Componentes internos do *Kinect* e fluxo de dados.

Não se dará especial importância ao *Kinect* dado que neste capítulo, o objetivo é de conceber um robô móvel controlável manualmente e não autónomo. A figura seguinte é um exemplo de uma imagem da profundidade em ambiente *Lazarus* gerada pelo *Kinect* com a interface que disponibiliza o componente *SdpoFreenect* do componente *5dpo* [36].

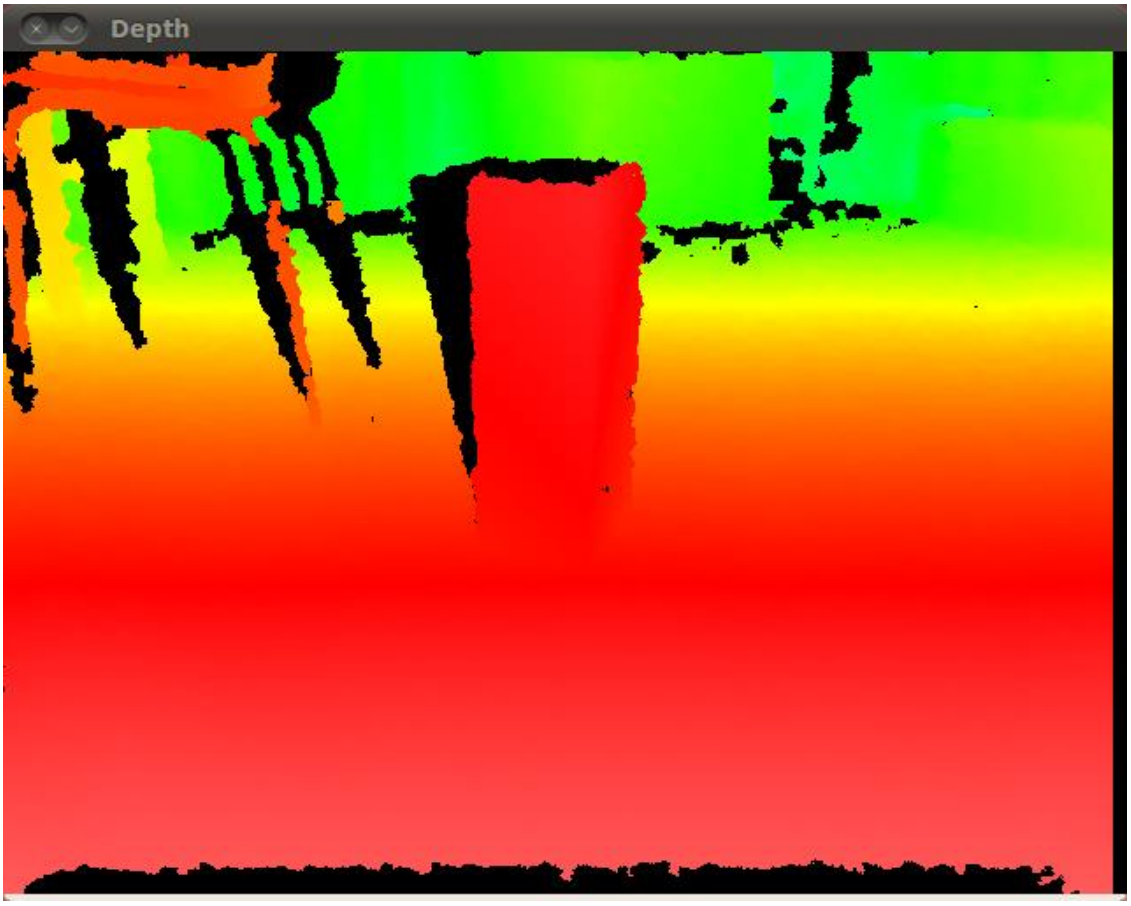

Figura 3-19 - Imagem de profundidade do Kinect usando o *package 5dpo* do *Lazarus*, da perspetiva do robô.

O componente da referida biblioteca tem incluída uma função que lineariza o valor atribuído a cada *pixel* em função da distância que facilita consideravelmente o cálculo da distância em cada ponto da imagem. O valor de um *pixel* da imagem de profundidade em função da distância (cm) encontra-se no gráfico da figura 3-20.

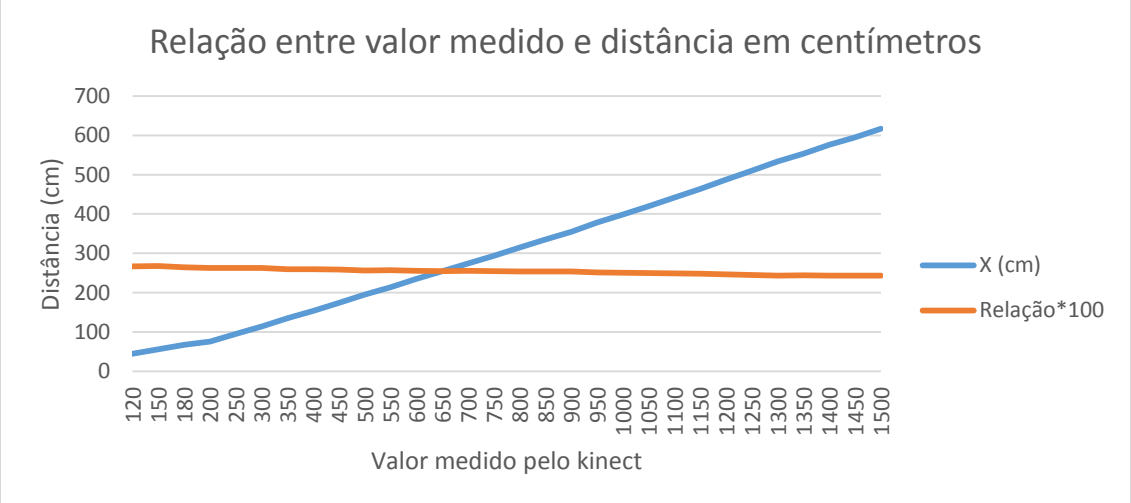

Figura 3-20 - Gráfico do valor de cada *pixel* em função da sua distância em centímetros.

O quociente do valor atribuído pelo *Kinect* a cada *pixel* por a distância varia decrescentemente entre 2,6 e 2,4. Dado que só os objetos próximos poderão ser obstáculos para o robô, optou se pela primeira razão.

Um outro facto importante é a existência de "manchas" de cor preta na imagem que podem ser consequência de dois fenómenos distintos:

- Zona de sombra de um objeto relativamente ao emissor IV, ou seja, a câmara IV tem visível (não obstruído) uma parte do ambiente analisado que ficou fora do alcance do emissor IV. A câmara não pode receber reflexão de raios IV de uma superfície que não foi alcançada por estes. Este caso é pouco relevante dado que uma sombra está sempre mais distante do que o objeto que lhe deu origem, em que este sim é detetado pelo *Kinect*.
- Zona demasiado próxima do *Kinect*. A menos de cerca de 50 cm de um objeto, o *Kinect* é incapaz de atribuir-lhe um valor de distância. O facto que este fenómeno se aplique a valores reduzidos de distância (0 – 50 cm) faz com que estas "manchas" tenham especial interesse quando se trata de evitar obstáculos.

Nos dois casos as manchas pretas são consequência da mesma causa: os raios emitidos não são recebidos. A impossibilidade de diferenciar os dois casos, sendo um relevante e outro não, coloca um problema na interpretação dos resultados medidos pelo sensor. Esta é uma das poucas desvantagens comparando com as câmaras estereoscópicas dado a independência das duas câmaras não possibilitarem sombras.

No capítulo 4 serão abordados os algoritmos aplicados a imagem de profundidade do *Kinect*.

3.2 LIGAÇÕES ENTRE BLOCOS E FLUXO DE DADOS

Agora que já foi descrita a função de cada bloco, será abordado primeiro as ligações inter-blocos e, em seguida, o fluxo de dados/potência a que dão suporte.

Foi usado cabo de 1 mm² para as ligações entre a placa de alimentação e a bateria, e entre a placa de alimentação e o *drive* por transportarem mais corrente (alimentam os motores). O resto das ligações foram feitas usado fios de *breadboard*. A seguinte figura ilustra as ligações físicas entre os componentes todos do robô.

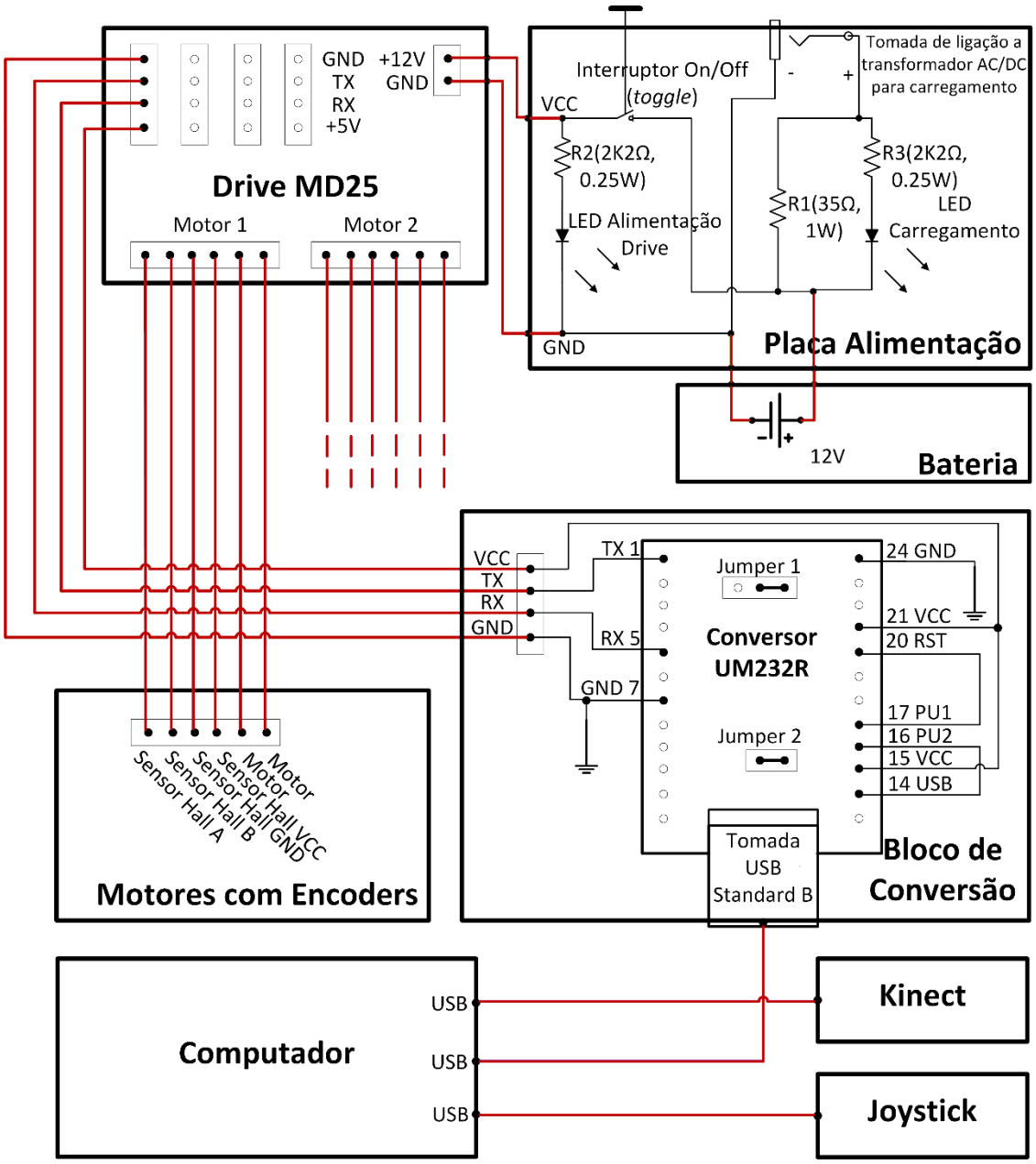

Figura 3-21 - Circuito completo do robô.

O computador só transmite informação para pedir um serviço a outro dispositivo, ou seja, fixa um estado a atingir. Pode dizer-se que é o cérebro do robô. É ele que recebe os dados relevantes dentro do robô e no exterior, que toma decisões, que encarrega outro bloco de atingir os objetivos, e que verifica que foram atingidos. Por isso é que é necessário medir os resultados e interpretá-los. Como é o caso dos *encoders* que indicam a amplitude do deslocamento, sem eles o robô não se saberia situar.

Dado o elevado número de componentes, e para uma mais fácil visualização, os dados que são trocados serão expostos com um fluxograma completo do robô que indica o sentido e o conteúdo da comunicação.

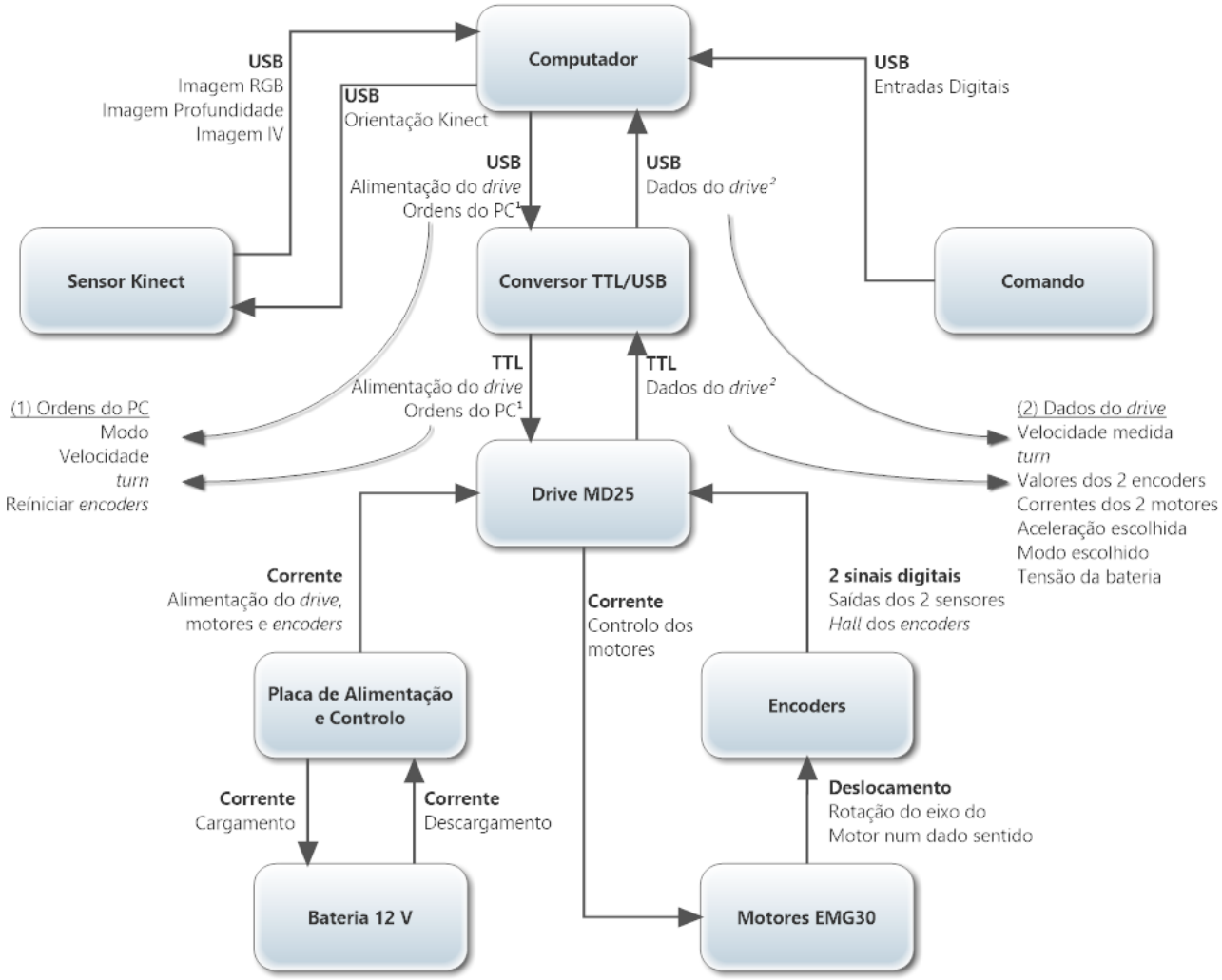

Figura 3-22 - Blocos do Robô inseridos na estrutura, respetivas ligações e sentido do fluxo de dados/potência.

O robô encontra-se concluído neste estado. O ponto seguinte relata o teste com um comando validando as ligações e configurações do conjunto.

3.3 TESTE COM COMANDO

A figura seguinte apresenta a interface gráfica do programa desenvolvido neste estudo. Note-se que esta versão sofreu entretanto melhorias fora do âmbito deste capítulo (nomeadamente por causa do *Kinect* que não faz parte do âmbito deste capítulo), daí ser mais conveniente a apresentação desta versão mais básica.

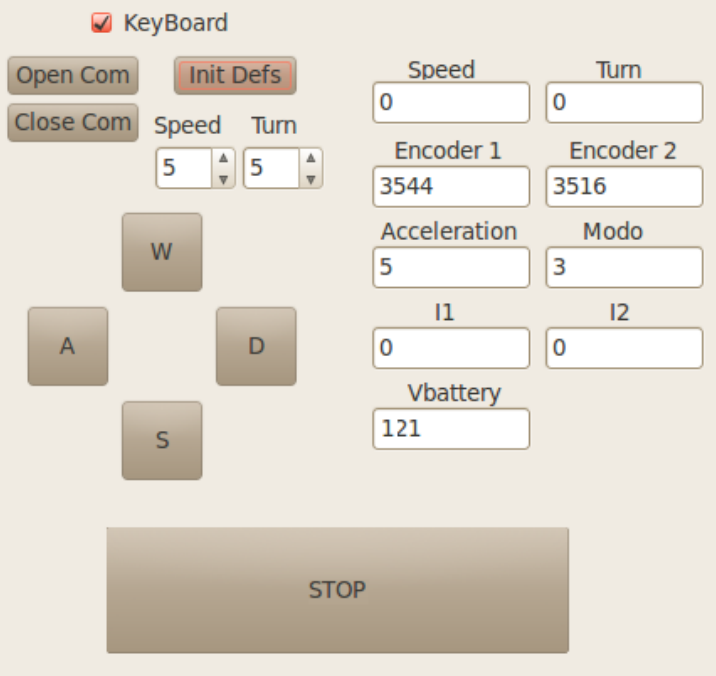

Figura 3-23 - Interface gráfica de monitorização e controlo do robô (v1).

Além de permitir controlar o robô, a interface gráfica permite o acesso aos dados recolhidos pelo *drive* como o *Speed*, *Turn*, valores do *encoders*, valor de aceleração e modo definidos, corrente em cada motor e tensão da bateria (neste caso 12,1 V).

Com as ligações todas efetuadas, as etapas para ligar e controlar remotamente o robô são as seguintes:

- Executar programa com interface gráfica de monitorização e controlo do robô desenvolvido em *Lazarus*;
- Executar o programa *JoyToKey* para emular as teclas do teclado no comando;
- Abrir a comunicação com o *drive* clicando no botão "*Open Com*";
- Fazer *reset* aos *encoders* e selecionar o modo de funcionamento do *drive* clicando no botão "*Init Defs*";

- Selecionar modo manual de controlo do robô na *checkbox* "*KeyBoard*";
- Ligar interruptor da placa de alimentação que alimenta motores e *encoders*.

O controlo do deslocamento com o comando foi bem-sucedido para a movimentação do robô, porém, dado que o comando só tem entradas digitais, a única maneira de mudar a direção do robô é a rotação sobre ele próprio. Foi escolhido a rotação sobre ele próprio por ser mais útil para dar meia volta, e não há tantos botões como curvas possíveis. Isto implica que para contornar um objeto o robô terá de alternar entre avanços em linha reta e rotações sobre ele próprio. Seu traçado seria um polígono com tantos lados como alterações de direção. Tecnicamente significa que o *Speed* e o *Turn* não podem ser não-nulos simultaneamente.

Os valores de *Speed* e *Turn* a aplicar também estão acessíveis desde o comando para encontrar valores razoáveis para estes. Pressionar um botão implica o envio da instrução correspondente ao *drive*, quando o botão é largado é enviada a instrução de colocar *Speed* e *Turn* a zero.

Com um comando analógico suportado pelo componente da biblioteca *5dpo* (*TSdpoJoystick*) [30] teria sido possível controlar o robô mais facilmente e permitir trajetórias curvadas variadas, isto é, *Speed* e *Turn* não-nulos simultaneamente. Seria então possível contornar um objeto traçando uma circunferência, neste caso, a trajetória ideal. Contudo, como o objetivo final deste estudo é o robô ser autónomo, o tipo de comando para controlo manual é irrelevante.

O carregamento da bateria foi efetuado com sucesso recorrendo a um transformador[3].

Tendo em conta todos os fatores anteriores, o integral do robô móvel está a funcionar como desejado.

No capítulo 4 serão abordados os algoritmos que permitem ao robô ganhar autonomia, isto é, saber onde ele se situa, onde quer ir, e desviar-se dos eventuais obstáculos que possam estar na sua rota.

[3] Transformador AC/DC de 18 V, 1,1 A.

Capítulo 4

SISTEMA DE CONTROLO

Este capítulo será dedicado ao sistema de controlo do robô e ao *software* desenvolvido no âmbito deste estudo. O desenvolvimento do método de navegação será abordado no primeiro ponto. A implementação da estimação da posição absoluta preenche o segundo ponto deste capítulo. A deteção e contorno de obstáculos são apresentados nos pontos 4.3 e 4.4 respetivamente. Finalmente, no último ponto, a representação global do sistema através de uma rede de Petri.

Apesar do *Kinect* conseguir produzir imagens (RGB e RGB-D) em intervalos de tempo de 30 ms, todo o resto do sistema de controlo atualiza e processa os dados a cada 100 ms devido a recolha dos dados do MD25.

Na prática, os resultados dos cálculos referentes a posições de *pixels* ou valores de variáveis informáticas inteiras deverão ser arredondados.

4.1 MÉTODO DE NAVEGAÇÃO

4.1.1 SEGUIMENTO DE FAIXA

O método de navegação usado neste estudo foi o sistema de faixas que é um método amplamente usado em robôs de pequenas dimensões destinados a locais *indoor*. A tarefa mais comum do robô será de seguir essa linha.

Existem diversas soluções distintas como fitas magnéticas [1], ou faixas de cor (normalmente preta) detetadas por os sensores apropriados. O mais usual em faixas de cor é o uso de LED's e LDR's acoplados na parte inferior do robô formando sensores capazes de medir a refletância do solo, detetando assim a faixa quando a refletância for inferior a um valor *threshold*. Para isso são necessários no mínimo 2 sensores [38][39].

Neste caso aproveitou-se a imagem RGB criada pelo *Kinect* para seguir uma faixa de cor branca no solo.

A configuração do trajeto considerada é a mais simples, *Single Line / Back and Forth*, já que o objetivo neste estudo é que o robô saia ligeiramente do seu percurso normal para seguir caminho perante um obstáculo, e não recorrer a um caminho alternativo. Dada a impossibilidade do *Kinect* alterar sua orientação no eixo horizontal e á falta de sensores de distância, o robô só poderá andar para a frente em segurança.

Nas extremidades do percurso foi colocado uma marcação indicando que é o final da linha, o que terá de implicar uma alteração na direção do robô de 180º.

A figura 4-1 ilustra um exemplo do percurso do robô.

Figura 4-1 - A configuração do trajeto considerado (*Single Line / Back and Forth).*

Para o robô conseguir seguir a faixa, é necessário recorrer á imagem RGB (640 x 480 *pixels*) gerada pelo *Kinect* para situá-la devido a sua particularidade[4]. O *Kinect* estando orientado para a frente, em sintonia com a direção do robô, o desejado será que a faixa, mais precisamente o centro da faixa, esteja no centro da imagem (figura 4-2). Qualquer desfasamento entre o centro da imagem e a faixa representa um erro que o robô tentará continuamente anular com alterações de direção para o respetivo lado.

Figura 4-2 – Imagem RGB gerada pelo *Kinect* da perspetiva do robô usada para deteção da faixa, a linha vermelha representa o centro da imagem.

Maior parte da imagem é irrelevante dentro do objetivo de deteção de linha e origina o primeiro tratamento que consiste em selecionar a parte da imagem que abrange o solo. É para isso usado a mascara da figura 4-3 (b).

[4] A faixa é branca e o solo escuro.

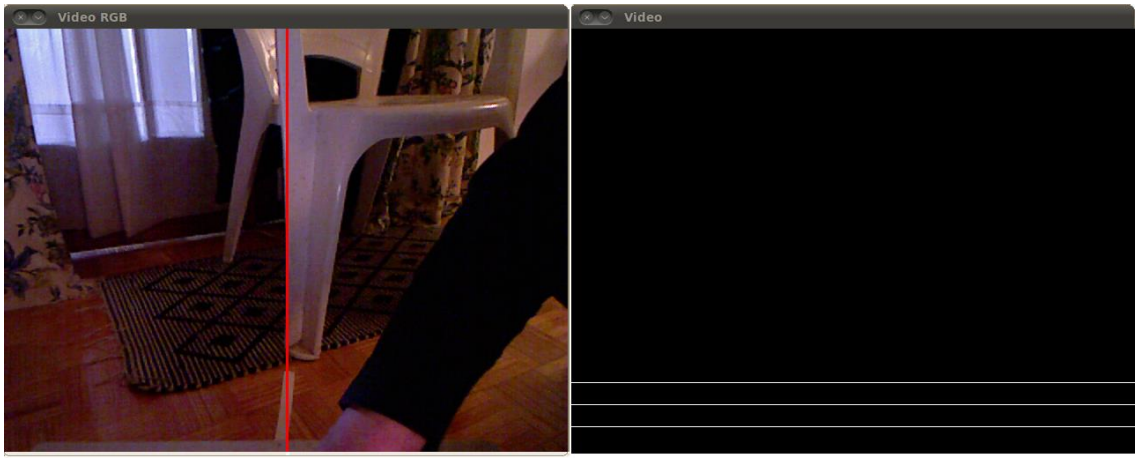

Figura 4-3 - (a) Imagem RGB; (b) Mascara usada.

O resultado são 3 linhas sendo l_3, l_2 e l_1 de cima para baixo respetivamente. Essas linhas representam cada uma 3 vetores de cada componente de cor que são convertidas para linhas a níveis de cinzento (*grayscale*) com valores de 0 a 255. A fórmula aplicada é a seguinte.

$$P_{Gray}(x;y) = 0{,}3\,.\,P_{Red}(x;y) + 0{,}6\,.\,P_{Green}(x;y) + 0{,}1\,.\,P_{Blue}(x;y) \quad (3)$$

A linha l_1, que é a mais próxima do robô, é usada para calcular certos parâmetros que poderão variar com a cor do solo ou com a luminosidade. Este calculo auxiliar tem por objetivo o robô adaptar-se ao ambiente variando o valor que separa aquilo que é considerado faixa daquilo que não é. Esse valor, $Threshold_{Faixa}$, é baseado no valor médio e no máximo[5] de l_1.

$$Threshold_{Faixa} = 0{,}2.M\acute{e}dia_{l_1} + 0{,}8.M\acute{a}ximo_{l_1} \quad (4)$$

O passo seguinte consiste na binarização das linhas, os *pixels* com valores superiores ao $Threshold_{Faixa}$ serão considerados faixa (valor lógico não nulo) e os outros não faixa (valor lógico nulo).

$$P_{Bin}(x;y) = \begin{cases} 0 \;,\; se\; P_{Gray}(x;y) \leq Threshold_{Faixa} \\ 1 \;,\; se\; P_{Gray}(x;y) > Threshold_{Faixa} \end{cases} \quad (5)$$

[5] Provavelmente a faixa já que o branco puro tem valor máximo na escala de 0 a 255.

Os 3 vetores binarizados são em seguida varridos começando pelas extremidades a procura de um flanco ascendente que será marcado com uma *flag*. Existe uma *flag* esquerda e direita por vetor como está representado na figura 4-4.

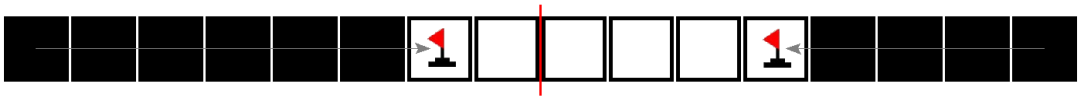

Figura 4-4 – Varrimento dos vetores a procura de flanco ascendente.

Fazendo a média das duas *flags* (esquerda e direita, exemplo para l_1: $flag_{1_{esq}}$ e $flag_{1_{dir}}$) obtemos a posição média da faixa em cada linha da imagem $Faixa_{l_x}$. Os vetores têm comprimento igual á largura da imagem ($Largura_{Img}$, 640 *pixels*), começam na posição 0 e terminam na posição $Largura_{Img} - 1$. A posição da faixa em cada linha pode assumir qualquer valor nesse intervalo e calcula-se como é descrito abaixo.

$$Faixa_{l_x} = \frac{(flag_{x_{esq}} + flag_{x_{dir}})}{2}\ [pixels] \tag{6}$$

O erro por linha é a distância entre a posição da faixa na linha e o centro do vetor, é por isso necessário mudar o referencial obtendo valores concretos de $-\frac{Largura_{Img}}{2}$ a $\frac{Largura_{Img}}{2}$:

$$erro_{l_x} = Faixa_{l_x} - \frac{Largura_{Img}}{2}\ [pixels] \tag{7}$$

Assim, o módulo do erro indicará a grandeza do desalinhamento entre a faixa e o centro da imagem, e o seu sinal indicará em que metade da imagem se encontra a faixa.

Além disso, o modo usado no MD25 também usa o 0 como um valor neutro e limiar, nomeadamente no *Turn* que está diretamente envolvido. Quando a faixa se encontra na metade positiva da imagem produz um *Turn* positivo, e vice-versa anulando assim o erro.

Para seguimento da linha, o *Turn* dinâmico do robô dependerá dos erros por linha conforme a equação seguinte:

$$Turn_{din} = k_1.erro_{l_1} + k_3.erro_{l_3} \tag{8}$$

Após testes $\frac{1}{50}$ e $\frac{3}{50}$ revelaram-se valores adequados para k_1 e k_3 respetivamente.

A velocidade linear ou *Speed* dinâmico dependerá das curvas tal como o *Turn* dinâmico mas inversamente. A velocidade linear dinâmica aumenta nas retas e diminui das curvas onde será necessário acertar a direção e é definida pelas seguintes equações:

$$Speed_{din} = \begin{cases} \frac{k_{Speed_{Max}}}{|Turn_{din}|} & , \ se\ Turn_{din} \neq 0 \\ k_{Speed_{Max}} & , \ se\ Turn_{din} = 0 \end{cases} \quad (9)$$

As curvas da trajetória não podem ser demasiadamente apertadas, caso contrário pode acontecer que o robô não consiga acompanhar a linha e consequentemente se perca. Usando sensores de refletância é possível o robô acompanhar curvas mais apertadas [38][39] por não ter ângulos mortos como aqui usando a câmara.

As extremidades do percurso são as únicas estações do AGV. Após serem detetadas, o robô terá de chegar a uma extremidade do percurso, dar meia volta e recomeçar a seguir a linha no sentido oposto.

A marcação desses terminais é formada por uma faixa suplementar sobreposta perpendicularmente perto do final das extremidades do percurso como se pôde ver na figura 4-1. O algoritmo para detetar a marcação baseia-se nas 3 linhas binarizadas, mais precisamente sobre a distância entre *flags* que representa a largura da faixa nessa linha $Larg\_Faixa_{l_x}$.

$$Larg\_Faixa_{l_x} = flag_{x_{dir}} - flag_{x_{esq}}\ [pixels] \quad (10)$$

Quando a largura da faixa na linha do meio l_2 for 4 vezes superior á média das outras duas, o marcador foi detetado e implica que só resta cerca de 50 centímetros até ao final do percurso (FP):

$$FP = \begin{cases} 1 \ , \ se\ Larg\_Faixa_{l_2} \geq 4.\left(\frac{Larg\_Faixa_{l_1} + Larg\_Faixa_{l_3}}{2}\right) \\ 0 \ , \ se\ Larg\_Faixa_{l_2} < 4.\left(\frac{Larg\_Faixa_{l_1} + Larg\_Faixa_{l_3}}{2}\right) \end{cases} \quad (11)$$

O momento exato onde o marcador é detetado é quando a faixa perpendicular se sobrepõe a l_2 como está esquematizado na figura 4-5.

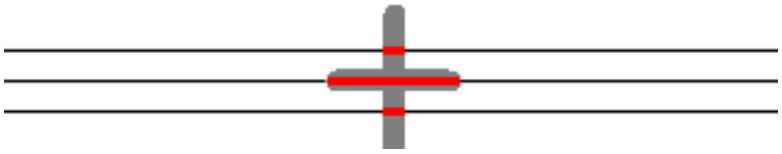

Figura 4-5 – Marcador detetado.

A partir do momento em que o marcador foi detetado o robô deixa de seguir a linha para entrar num estado que tem por objetivo a inversão da marcha como é possível ver na rede de Petri da figura 4-6.

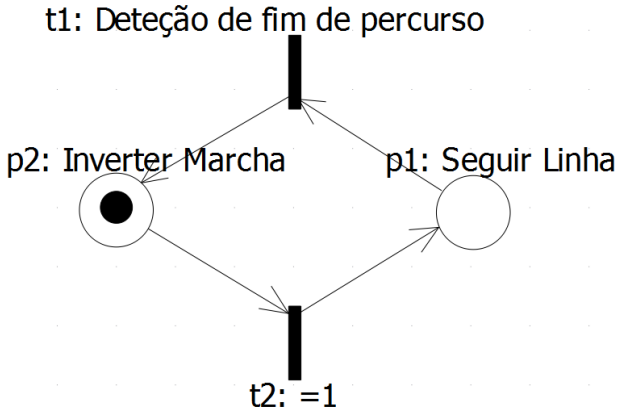

Figura 4-6 - Rede de Petri Parcial do Sistema.

A transição *t2* está sempre ativa, para o *token* voltar para o estado de seguir a linha bastará que o estado de inversão de marcha chegue ao fim do seu processo. Porém, existe neste momento um problema quanto a validação do fim do estado de inversão de marcha porque o robô só dispõe de informação quanto ao seu destino e não sobre sua posição absoluta e orientação.

Razão pela qual se recorreu a odometria para obter informações sobre a posição absoluta e orientação do robô atual e passada. Esse tema é abordado em 4.2.

4.1.2 ALGORITMO DE PERDA DE FAIXA

Para que o robô seja consciente que se perdeu quando não vê a faixa num momento em que era suposto vê-la, foram implementados algoritmos para alertar o robô para esse facto. É usado um valor *threshold* para a cor dos elementos de uma linha analisada da imagem

com o objetivo de encontrar a faixa, que deverá ter uma certa largura e provavelmente estará situada perto do centro horizontal da imagem.

Também será necessário usar este algoritmo para averiguar se o obstáculo está sobre a faixa ou ao lado em 4.4.1, no Teste 1, só que neste caso, as linhas da imagem analisadas são aquelas que estão ao nível do obstáculo e não as linhas usadas para navegação. A figura 4-7 representa a rede de Petri atualizada com a deteção de perda de faixa tornando o sistema mais fiável.

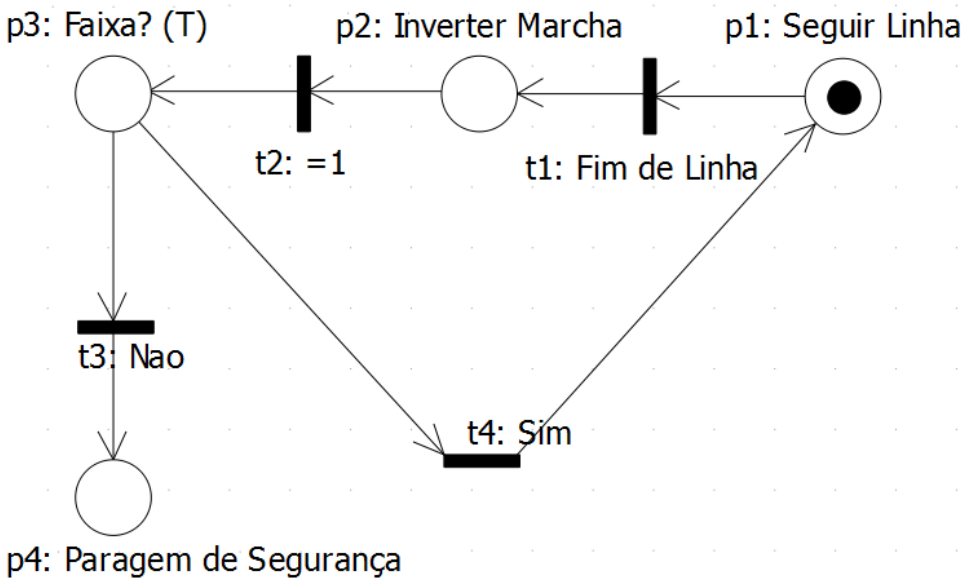

Figura 4-7 - Rede de Petri Atualizada com deteção de perda de faixa.

4.2 ODOMETRIA

4.2.1 CÁLCULO DO DESLOCAMENTO

A odometria é um método amplamente usado para estimar a posição dos robôs sendo baseada na distância percorrida, na velocidade do veículo ou na sua aceleração dependendo dos métodos e sensores usados.

O sensor *Kinect* tem incorporado um acelerómetro em 3 dimensões mas devido a necessidade de integrar duas vezes os seus valores a cada iteração, o erro seria ampliado e apresenta-se desvantajoso em relação aos *encoders* que não têm erro cumulativo e têm uma boa resolução, 360 incrementações por volta completa da roda. O diâmetro de cada roda é de 10 centímetros. Pode-se deduzir que:

$$1\,Volta = 360\,[inc] = \pi.10[cm] \tag{12}$$

Então:

$$1[cm] = \frac{360}{10\pi}[inc] \quad \Leftrightarrow \quad 1\,[inc\,] = \frac{10\pi}{360}\,[cm] \tag{13}$$

Enquanto o robô andar em linha reta os *encoders* de ambos lados variam similarmente e a distância percorrida pelo robô é igual á percorrida por cada uma das rodas. A equação para calcular o módulo do vetor deslocamento a cada iteração é a seguinte:

$$\left|\vec{d}\right| = \frac{\Delta encoders_1 + \Delta encoders_2}{2}.\frac{10\pi}{360}\,[cm] \tag{14}$$

Onde a variação do $encoders_x$ é definida da seguinte forma:

$$\Delta encoders_x = encoders_{x_{Atual}} - encoders_{x_{Anterior}} \tag{15}$$

Quanto às alterações na orientação do robô são criadas pelas diferenças entre as variações dos *encoders*. O comprimento do eixo L_{Eixo} do robô também é parâmetro necessário para poder calcular a rotação no deslocamento, neste caso o comprimento do eixo é de 34 cm. O valor de $2\pi L_{Eixo}$ em centímetros é a diferença entre distâncias percorridas por as rodas requerida para alterar a orientação do robô diferencial em 360°. O fato de esse valor ser constante implica que o valor dos *encoders* são suficientes para conhecer a orientação do robô $\angle\,\vec{d}$ mas insuficientes para estimar a posição por si só. A variação da orientação a cada iteração calcula-se da maneira seguinte:

$$\angle\,\vec{d} = \frac{\Delta encoders_2 - \Delta encoders_1\,[\cancel{inc}]}{\frac{2\pi L_{Eixo}[\cancel{cm}]}{\cancel{360}\,[°]}.\frac{\cancel{360}\,[\cancel{inc}]}{10\pi.\cancel{1}[\cancel{cm}]}} = \frac{5(\Delta encoders_2 - \Delta encoders_1)}{L_{Eixo}}\,[°] \tag{16}$$

O cálculo da variação de orientação do robô $\angle\,\vec{d}$ pela equação anterior aproxima a trajetória a uma sequência de arcos de circunferência de cumprimento $\left|\vec{d}\right|$. Sua precisão requer que curvatura da trajetória, dada por $\angle\,\vec{d}/\left|\vec{d}\right|$, seja constante no intervalo entre amostras. A frequência de amostragem terá então de ser tão alta quão possível (neste caso 100 Hz) para minimizar o efeito de desprezar que a curvatura pode mudar dentro de um mesmo intervalo.

Em [40] é descrita a implementação de um filtro de Kalman que pretende anular erros gaussianos de media 0 baseando-se em várias amostras.

A odometria baseada em *encoders* sem giroscópio tende a acumular erros de orientação que irão amplificar-se nas estimações de posição em trajetos mais longos. A imprecisão deste método sem giroscópios fez com que a odometria não pudesse servir de sistema de navegação para este estudo. Contudo, estes algoritmos permitem ao robô situar-se em curtas distâncias e fazer tarefas como andar uma distância concreta e saber que chegou a seu destino como será necessário no contorno de obstáculos, saber parar de girar quando alcançados os 180° da alteração da orientação do robô como é necessário nos finais de percurso.

4.2.2 ATUALIZAÇÃO DAS COORDENADAS

Conhecendo a posição inicial e os deslocamentos do robô, é possível estimar a posição e orientação do robô ao longo do tempo. Este faz um *reset* aos *encoders* no início de cada jornada, colocando-se na posição $(x = 0\,;\, y = 0)$ do mapa-mundo com ângulo de orientação de 0° (θ_n).

As coordenadas atuais são estimadas e atualizadas a cada iteração com base nos parâmetros do momento imediatamente anterior e do deslocamento atual medido da seguinte forma:

$$\begin{cases} x_n = x_{n-1} + \left|\vec{d}\right|.\cos(\theta_n) \\ y_n = y_{n-1} + \left|\vec{d}\right|.\sin(\theta_n) \\ \theta_n = \theta_{n-1} + \angle\vec{d} \end{cases} \tag{17}$$

O uso de marcadores em complementaridade a odometria permite a atualização de coordenadas absolutas fiáveis [40], contudo é necessário ter em memória o *template* de cada marcador. Como neste caso só existem 2 estações, possuem o mesmo marcador e estão alternadas no ciclo do percurso do robô, um *bit* indicando qual foi a última paragem do robô é suficiente para ele saber qual é a seguinte. Sempre que o robô acabar de dar meia volta para voltar a seguir a linha na estação inicial, as coordenas serão repostas para valores iniciais ($(0;0),\ \theta_n = 0°$) para anular erros acumulados na orientação e posição do robô.

Além de permitir estimar a posição atual, a odometria também permite criar um registro gráfico do percurso do robô, tema abordado no seguinte subcapítulo.

4.2.3 DESENHO DE MAPA-MUNDO

O mapa foi desenhado recorrendo às funções *Canvas* do *Lazarus*. Este contem um referencial com réguas em metros para facilitar a visualização da posição do robô, e no canto inferior direito, apresenta as coordenadas atuais do robô assim como seu ângulo de orientação θ_n. O trajeto do robô será atualizado a cada iteração desenhando um segmento da posição anterior até a posição atual. A figura 4-8 ilustra o mapa-mundo desenhado contendo a vermelho o trajeto do robô.

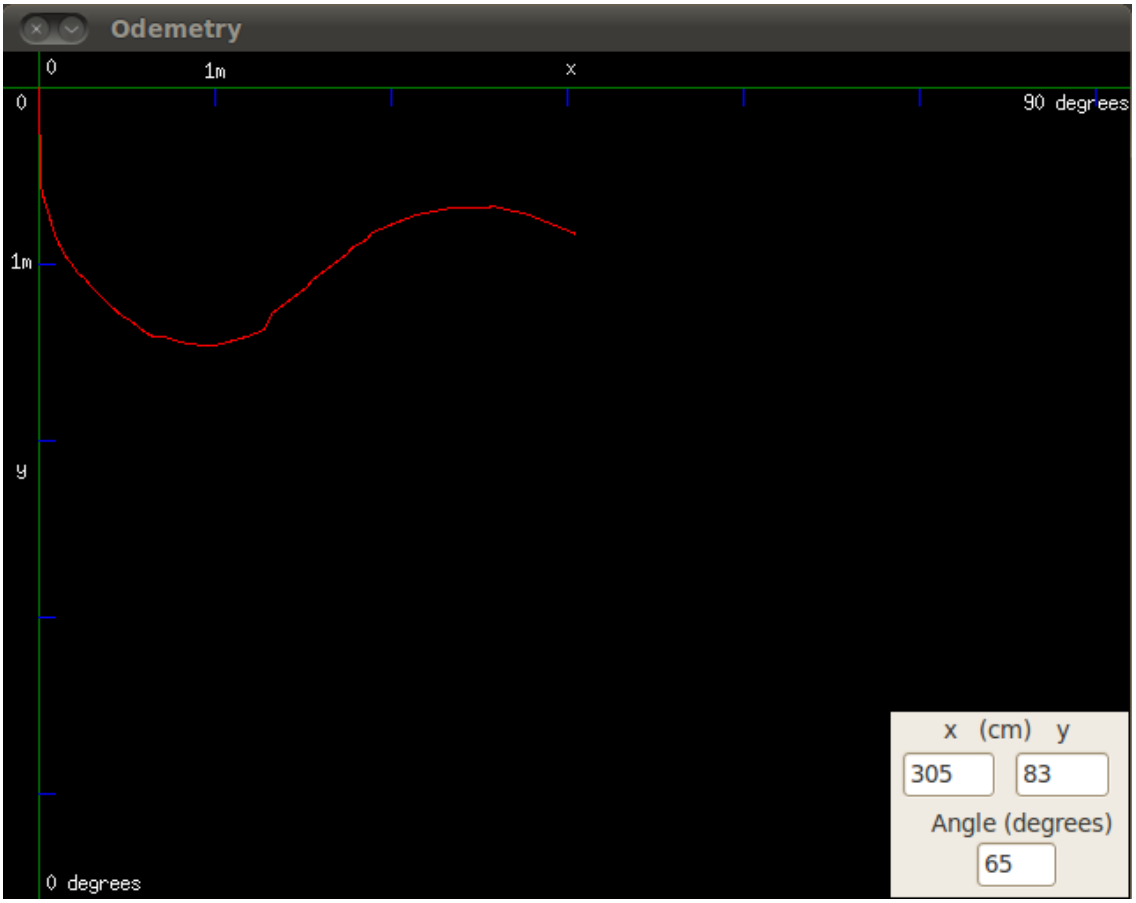

Figura 4-8 - Mapa-mundo com registro de passagem do robô. No centro inferior direito estão as coordenadas atuais o ângulo da orientação.

4.3 DETEÇÃO DE OBSTÁCULOS

A deteção de obstáculos em tempo real tem sido amplamente objeto de estudo nos últimos anos seguindo o progresso dos sensores na procura de um método que seja fiável, simples e económico para guiar os AGVs até ao destino em ambientes *indoor* não controlados.

Imagens a cor são de pouco interesse para a deteção de obstáculos já que em ambientes não controlados um obstáculo pode ser de qualquer cor e até não ter cor alguma como vidros ou um degrau descendente de umas escadas. Torna-se por isso necessário obter um mapa tridimensional relativo ao robô ou dito de outra forma uma imagem de profundidade. Para tal existem diversos métodos como a estereoscopia [41] e a medição da deformação de luz estruturada [42].

4.3.1 ESTUDO DE OUTRAS ABORDAGENS

4.3.1.1 ESTEREOSCOPIA

O *Kinect* baseia-se na deformação de um padrão de raios IV. Tem por defeito principal a criação de zonas ambíguas na imagem de profundidade comentadas em 3.1.8. que podem ser de interesse ou não. A estereoscopia não tem esse problema de perda de dados dado que é baseada em duas imagens RGB de duas câmaras distintas, no entanto esta técnica acarreta outros problemas dado que se baseia na cor para diferenciar obstáculos do solo. A figura 4-9 representa as duas imagens captadas por câmaras estereoscópicas [41].

Figura 4-9 - Imagens esquerda e direita de câmaras estereoscópicas.

Em [41] a cor que está no fundo da imagem (zona em frente do robô) é usada como referência considerando que os obstáculos serão de uma cor diferente. Definidos os

obstáculos em cada imagem e seus contornos (figura 4-10 [41]), é feita a correspondência entre contornos verticais[6] de cada objeto nas duas imagens aproveitando os fatos de os pontos de referência estarem na mesma posição vertical, terem a mesma cor, e estar na mesma ordem em cada imagem.

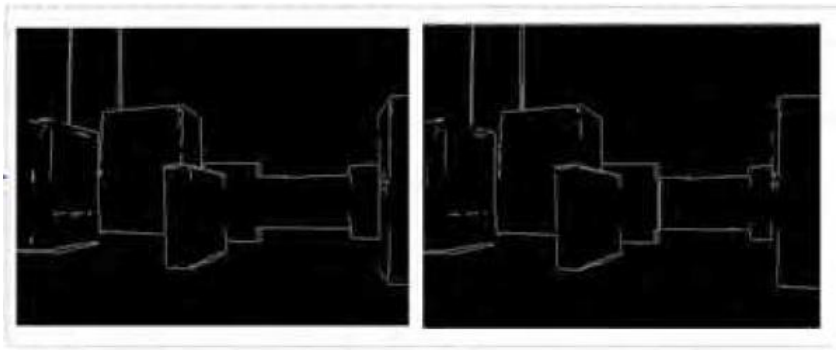

Figura 4-10 - Contornos de objetos de imagens estereoscópicas.

A distância de cada contorno vertical ao robô é calculada em função da distância que os separa em cada imagem. A distância do robô aos contornos horizontais do objeto (que nas imagens podem ser oblíquos dependendo do ponto de vista) varia linearmente entre valores definidos por os contornos verticais que estão nas suas extremidades. Na figura 4-11 pode observar-se uma imagem de profundidade dos contornos de obstáculos baseada em estereoscopia [41].

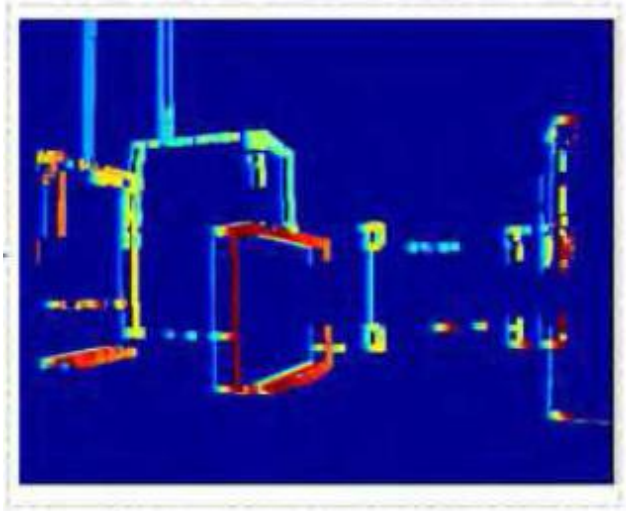

Figura 4-11 - Imagem de profundidade dos contornos baseado em estereoscopia.

[6] A correspondência é feita com os contornos contrários a orientação do alinhamento das câmaras.

O método usando câmaras estereoscópicas revelou-se satisfatório, contudo considera que o solo não tem padrões, que os obstáculos são de cor distinta e em forma de prismas regulares. É provável que o ponto do objeto mais próximo do robô não esteja no contorno se a forma do objeto for diferente (esferas, e toda forma convexa).

4.3.1.2 DEFORMAÇÃO DE LUZ ESTRUTURADA LINEAR

O artigo [42] referido neste subcapítulo é de particular interesse porque retrata, mesmo que de maneira mais simples, o que é necessário fazer com o *Kinect* para detetar obstáculos. Isto é, detetar relevos positivos ou negativos no solo.

Para tal, o robô móvel em questão mede com uma câmara[7] a deformação de um feixe de luz IV com o padrão de uma linha projetado à frente do robô. A figura 4-12 ilustra a configuração física do robô [42].

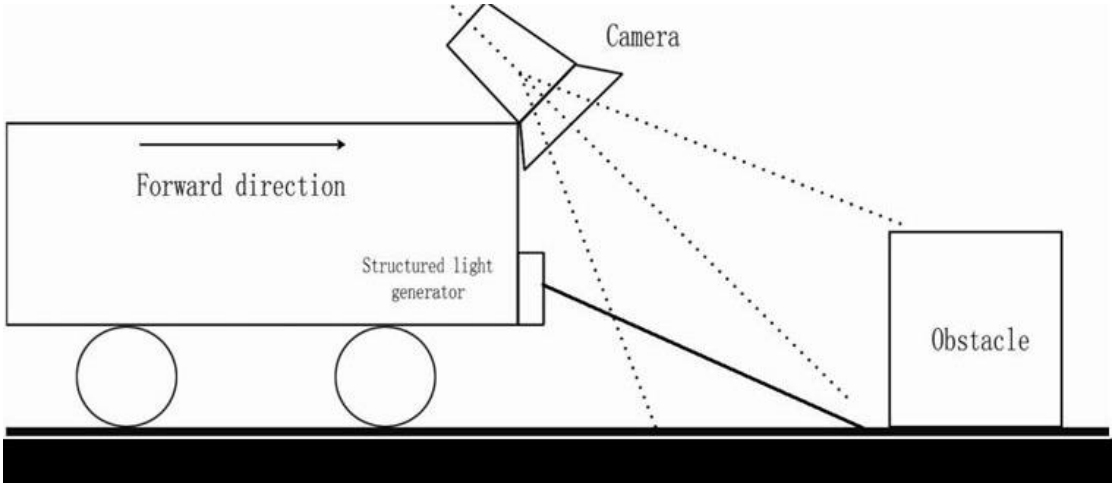

Figura 4-12 - Configuração física do robô [42].

Se não houver obstáculos, a linha projetada estará no centro da imagem. Perante um obstáculo, a linha desce na imagem em função da altura da projeção relativamente ao solo. A linha só poderia subir na imagem com um obstáculo negativo como um degrau descendente. A figura 4-13 ilustra a deformação da linha projetada perante um objeto e a binarização da imagem isolando a linha [42]. Note-se que as imagens binarizadas ilustradas também sofreram processamentos de adelgaçamento.

[7] A lente da câmara foi coberta por um filtro de 650nm para filtrar ondas visíveis e favorizar a entrada a raios IV.

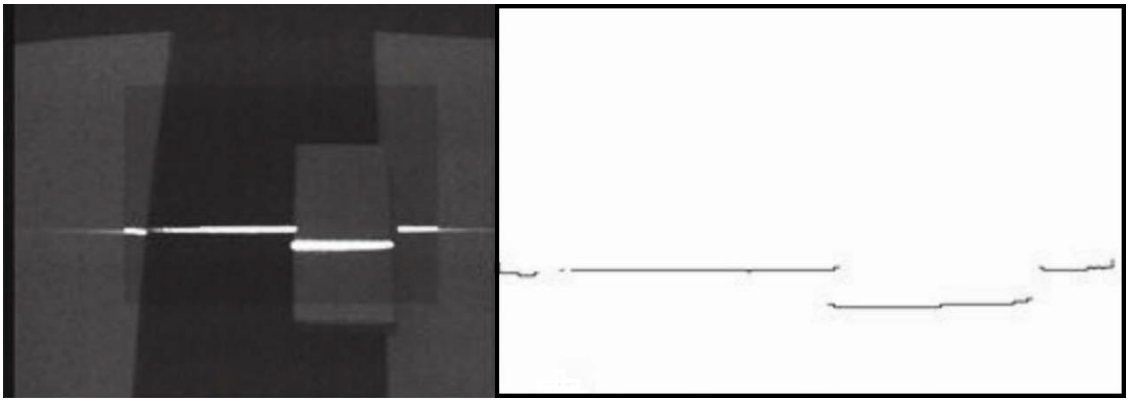

Figura 4-13 - A esquerda a imagem filtrada, à direita a binarizaçao da imagem.

As imagens binarizadas são em seguida comparadas a uma imagem binarizada de referência sem obstáculos que terá de ser guardada em memória pelo robô. Subtraindo as linhas da imagem de referência (solo) à imagem medida (tudo), ficamos com os obstáculos como se pode ver na figura 4-14 [42].

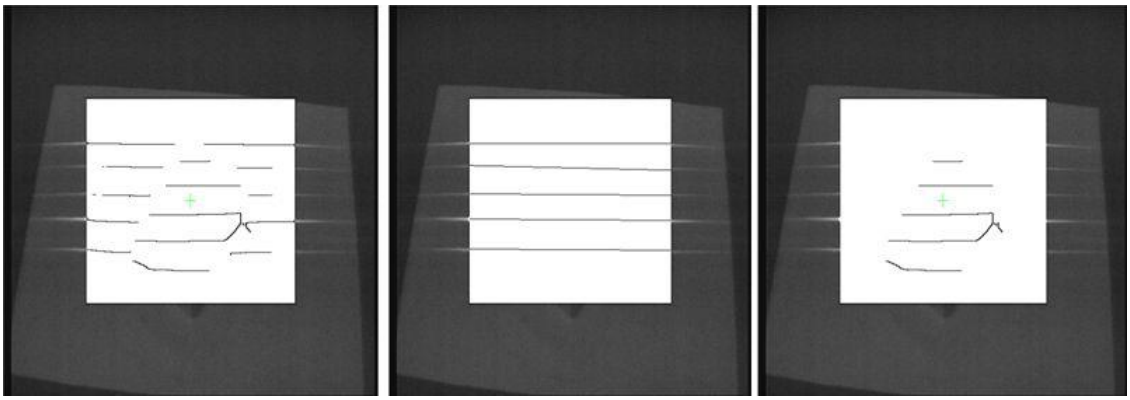

Figura 4-14 - Na esquerda a imagem capturada, no centro a imagem de referência e na direita os obstáculos.

Uma linha é suficiente para conhecer a altura, aumentando o número de linhas no padrão, conseguimos aumentar a resolução das outras dimensões dos obstáculos. Os valores das estimações da altura e da largura dos obstáculos são aceitáveis.

Este padrão às riscas horizontais é suficiente para a deteção de obstáculos, no entanto pode existir um fenómeno de *aliasing* entre as linhas para certas alturas de obstáculos que poderia tornar o obstáculo invisível á deteção de obstáculos. É por isso que o *Kinect* usa um padrão especial memorizado que não permite este tipo de fenómenos.

4.3.2 ALGORITMOS DE DETEÇÃO DE OBSTÁCULOS

Para o robô poder atingir o destino de forma segura, sem colisões nem quedas, é necessário diferenciar as zonas navegáveis do espaço das não-navegáveis. Neste subcapítulo serão abordados os algoritmos de deteção de obstáculos desenvolvidos neste estudo. Estes permitem ainda o cálculo das suas dimensões físicas e posições relativamente ao robô.

4.3.2.1 PRINCÍPIO

O princípio em que estão baseados os algoritmos é que uma zona é navegável se for coincidente com o plano em que está assente o robô, o solo. Vendo o problema de outra maneira, qualquer ponto do espaço que tenha um relevo (ou altura) não nulo em relação ao solo é um obstáculo ou um precipício, apresentando relevo positivo e negativo respetivamente. O robô assume portanto que o solo é plano e liso.

4.3.2.2 CAMPO DE VISÃO

O sensor *Kinect* está a uma altura h_k recuado de d_{pc} relativamente á parte frontal do robô. Nesta estrutura móvel, aumentando o ângulo vertical de visão do *Kinect* (α), o PC é a ultima parte visível do robô situado a uma altura h_{pc}. O parâmetro d_k representa a distância medida pelo *Kinect* até um ponto e d_g a distância pelo solo até ele como se pode ver na figura 4-15.

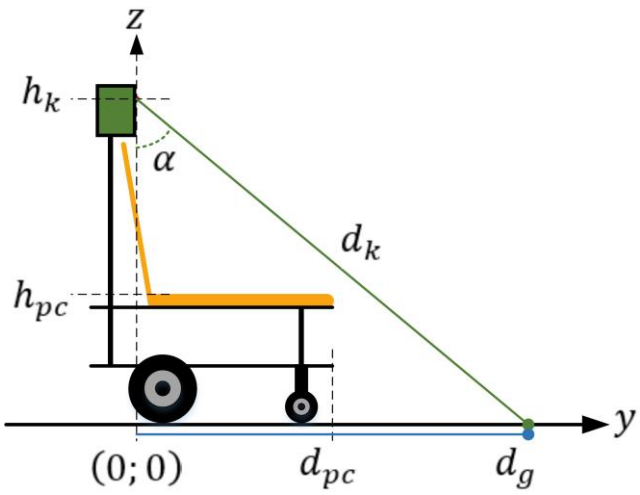

Figura 4-15 - Modelo do robô e respetivos parâmetros.

O ângulo de visão vertical do *Kinect* é bastante superior ao necessário, sobretudo quando não se pode garantir que a trajetória do robô será retilínea. Foi por isso delimitada

um ângulo de visão que cobre o solo entre a zona do solo visível mais próxima do robô e 70 cm á frente do robô. Estes ângulos verticais mínimos e máximos são calculados da seguinte forma:

$$\alpha_{min} = \tan^{-1}\frac{h_k - h_{pc}}{d_{pc}} = 45\ [°] \tag{18}$$

$$\alpha_{max} = \tan^{-1}\frac{d_{g_{max}}}{h_k} = 63{,}4349\ [°] \tag{19}$$

Um corte lateral ilustra o robô e sua zona do espaço analisada na figura 4-16.

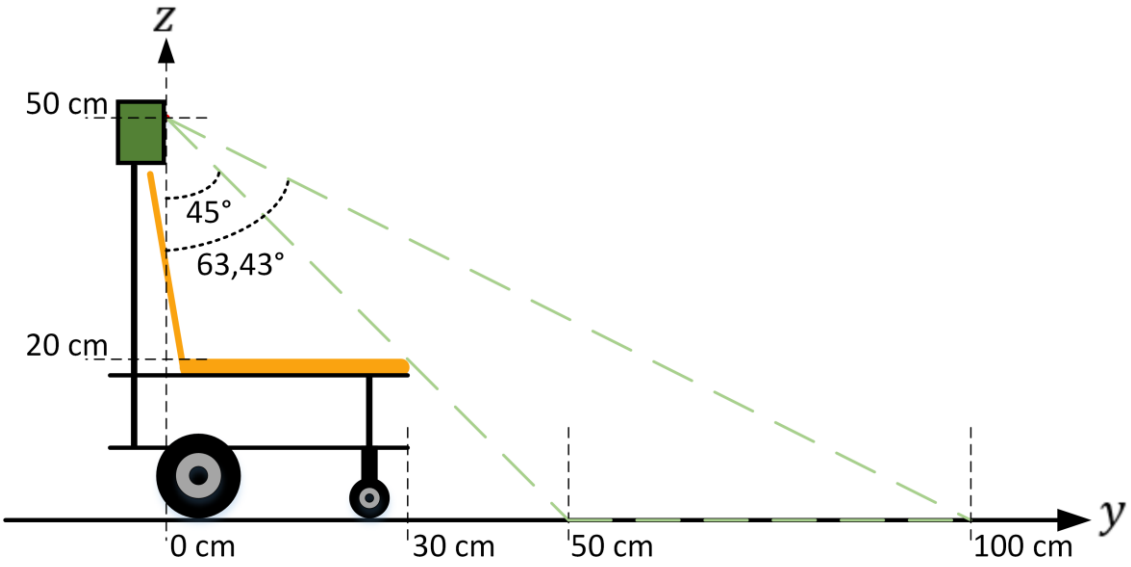

Figura 4-16 - Corte lateral com dimensões do robô e campo de visao desejado.

Note-se que a área à frente do robô que não está visível do ponto de vista do *Kinect* é responsável por a impossibilidade do robô seguir a faixa em curvas demasiadamente apertadas.

4.3.2.3 Máscara do Processamento de Imagem

A máscara usada no processamento da imagem de profundidade (640 x 480 *pixels*) consiste em delimitar na imagem a zona de interesse. O ângulo de visão vertical e horizontal do sensor *Kinect* é de 43º e 57º respetivamente [35]. Consideraremos que o ângulo vertical (α) e horizontal (β) variam linearmente em função da linha ou coluna da imagem em que está o ponto do espaço analisado. Existe por isso uma relação entre ângulos em graus e distâncias em *pixels*:

$$\rho = \frac{Altura_{Img}}{\hat{A}ngulo\_visual\_vertical} = \frac{480}{43} = 11{,}1628\ [^{pixels}/_{\circ}] \quad (20)$$

$$\sigma = \frac{Largura_{Img}}{\hat{A}ngulo\_visual\_horizontal} = \frac{640}{57} = 11{,}2281\ [^{pixels}/_{\circ}] \quad (21)$$

É possível determinar a altura da janela da máscara partindo da primeira linha de *pixels* que deteta solo (que está a 45º).

$$N_{linhas_{msc}} = \rho.(\alpha_{max} - \alpha_{min}) = 205{,}7856 \Rightarrow 206\ [pixels] \quad (22)$$

No fundo das imagens RGB e RGB-D, o número de linhas em *pixels* que são abrangidas por o PC é dado por $N_{linhas_{base}}$ (que neste caso é igual a 17), linhas que serão descartadas do processamento. No caso da imagem RGB-D, a zona abrangida pelo PC aparece como uma mancha preta (valores não-medidos) devido a sua proximidade (figura 3-19).

A janela da máscara terá a forma te um trapézio devido a perspetiva gráfica. O ângulo das partes laterais faz 60º com a base da imagem, isto para obstáculos que não estejam na trajetória do robô não causarem alterações na sua rota (paragem e/ou contorno do obstáculo). A figura 4-17 ilustra a máscara usada na deteção de obstáculos.

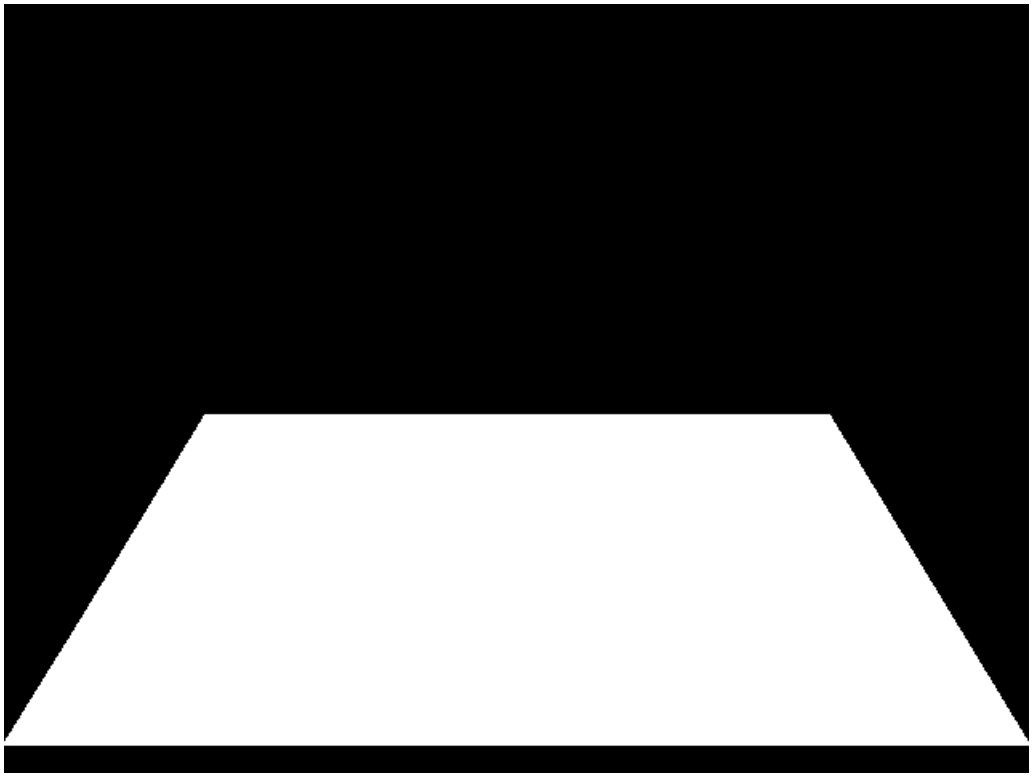

Figura 4-17 – Máscara de Deteção de obstáculos.

4.3.2.4 DETEÇÃO DE OBSTÁCULOS USANDO O KINECT

O ângulo α do ponto medido pelo *Kinect* é parâmetro imprescindível para averiguar se este se encontra no plano do solo ou não. Este é derivado da posição vertical do *pixel* na imagem RGB-D. O ângulo em graus de cada *pixel* da imagem em função da linha em que se encontra (l_{img}) é dado por:

$$\alpha(l_{img}) = \frac{(Altura_{img} - 1) - (N_{linhas_{base}}) - l_{img}}{\rho} + \alpha_{min}\ , \quad Sendo: 256 < l_{img} < 462 \tag{23}$$

Conhecendo o ângulo em graus a partir da linha da imagem em que se encontra o ponto na imagem RGB-D, é possível determinar a que distância (no eixo YY) se encontra o dito ponto no espaço considerando que não há obstáculos:

$$d_{g_{pixel}} = h_k . \tan \alpha_{pixel}\ [cm] \tag{24}$$

Ainda sem obstáculos, é possível determinar a que distância estarão do *Kinect* todos os pontos da imagem. Note-se que a semelhança do próprio *Kinect*, consideramos que todos os pontos de uma linha horizontal estão à mesma distância (tanto para d_k como para d_g). A distância entre o solo e o *Kinect* em função do ângulo é dado por:

$$d_{k\_ref_{pixel}} = \frac{h_k}{\cos \alpha_{pixel}}\ [cm] \tag{25}$$

As equações (24) e (25) só são verdadeiras se não houver obstáculos. No caso da equação (25), ela define a distância de referência para cada ponto, isto é, a distância se os pontos pertencem ao solo. Logo, se o valor do *Kinect* para um *pixel* (K_{pixel}) é sensivelmente igual a $d_{k_{pixel}}$, significa que o *pixel* representa um ponto do espaço que está no solo, que consequentemente, pertence a uma área navegável para o robô. A diferença entre o parâmetro medido e o de referência aumenta com a altura do obstáculo ou com a profundidade do precipício.

Na figura 4-18, pode-se observar que para ângulos diferentes existem valores de d_g e d_{k_ref}[8] diferentes. Contrariamente aos outros dois casos, para α_3 o valor medido pelo *Kinect,*

[8] Na figura 4-17, d_{k_ref} para α_3 é igual a d_{k_3} mais a parte tracejada.

d_{k_3}, diverge do valor de referência. O *pixel* em questão representa um obstáculo e será definido como tal.

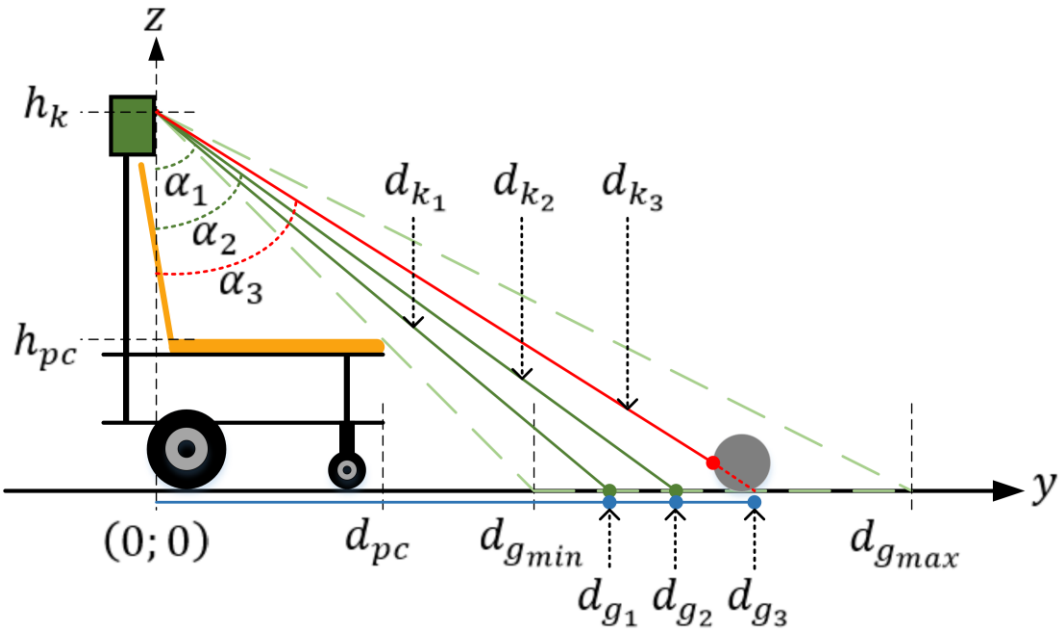

Figura 4-18 – Corte lateral do robô e seu campo de visão. Procura de obstáculos para 3 ângulos diferentes. Existe um obstáculo para α_3 dado que o valor medido pelo *Kinect* diverge da distância de referência (real).

A figura 4-19 mostra um exemplo de deteção de obstáculos baseado na imagem RGB-D usando os algoritmos anteriores.

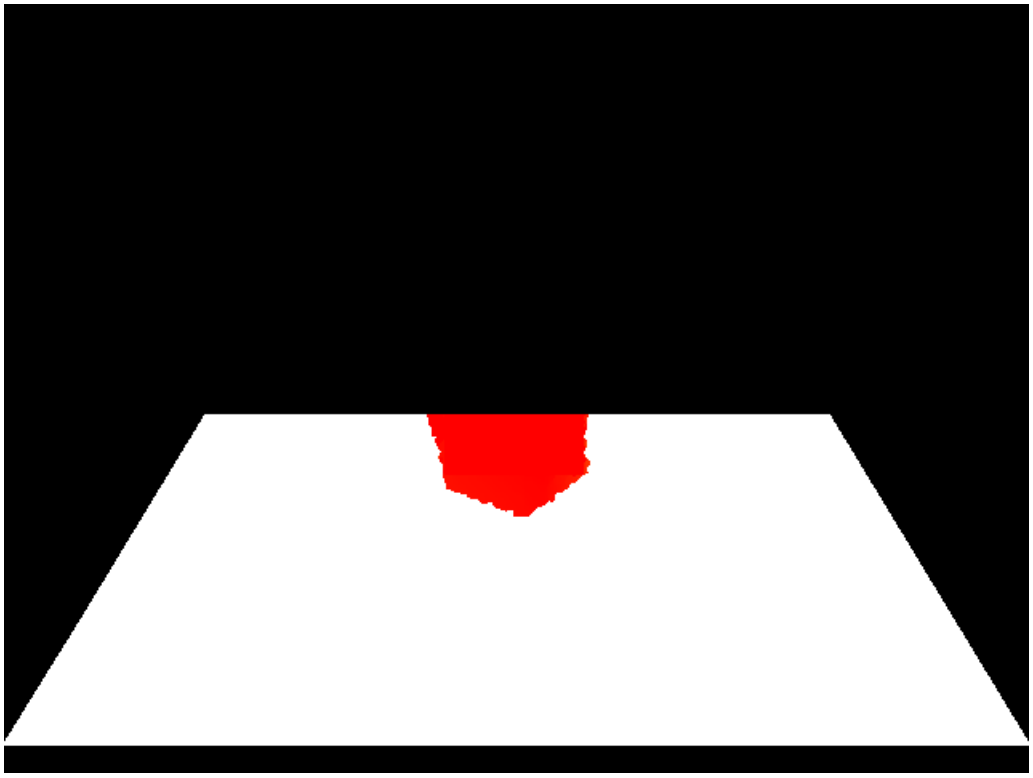

Figura 4-19 – Resultado de deteção de obstáculos baseado na imagem RGB-D. A imagem foi filtrada com a máscara e foi processada com os algoritmos de deteção de obstáculos.

Outra maneira de obter os valores de referência de profundidade em função do ângulo seria de fazer como [42], guardar uma imagem de referência, sem obstáculos, e compará-la com a imagem atual. A diferença das duas imagens são os obstáculos.

Para evitar que alterações na orientação do *Kinect* ou vibrações tirem o solo do plano de referência, é aconselhado autocalibrar o ângulo forçando os valores do fundo da janela da máscara a serem valores negativos. Por outras palavras assume-se que esses pontos pertencem ao solo, em seguida é corrigido o ângulo. Essa calibração e feita a cada iteração.

No Subcapítulo seguinte trata-se de caracterizar os obstáculos.

4.3.2.5 CARACTERIZAÇÃO DOS OBSTÁCULOS

Número de Obstáculos

O algoritmo de contagem de obstáculos não foi implementado no sistema. Contudo, este é relativamente simples pois é resultado do somatório das manchas na imagem binarizada de deteção de obstáculos.

Altura de um Ponto do Obstáculo

Um ponto pertencente a um obstáculo tem uma determinada altura relativamente ao solo. A distância real entre o *Kinect* e esse ponto é dado por a seguinte equação:

$$d_{k\_ref_{pixel}} = \frac{h_k - h_{o_{pixel}}}{\cos \alpha_{pixel}} \ [cm] \tag{26}$$

$$\Leftrightarrow h_{o_{pixel}} = h_k - d_{k\_ref_{pixel}}.\cos \alpha_{pixel} \ [cm] \tag{27}$$

O valor $d_{k\_ref_{pixel}}$ não é conhecido na prática, mas temos o valor do *Kinect* para esse ponto $d_{k_{pixel}}$. Assim podemos estimar a altura de um obstáculo num ponto:

$$h_{o_{pixel}} = h_k - d_{k_{pixel}}.\cos \alpha_{pixel} \ [cm] \tag{28}$$

O $h_{o_{pixel}}$ dos *pixels* que não são obstáculos será evidentemente sensivelmente 0. O *threshold* da altura que separa zonas navegáveis de obstáculos pode aqui ser definido. Neste estudo considerou-se solo aquilo que tinha altura fora do intervalo [-1:1] cm até para corrigir a imprecisão do *Kinect*. Para robôs exteriores com rodas muito maiores, não faria sentido considerar relevos da ordem de 1 cm como obstáculos pelo que o threshold a usar aqui depende do robô e do tipo de ambiente onde vai movimentar-se.

Para conhecer a altura máxima do obstáculo terá de se encontrar o *pixel* que maximiza a equação anterior na respetiva mancha.

Distância Mínima e Máxima de um Obstáculo

Para calcular a distância (no eixo YY) até um ponto pertencente a um obstáculo depende da altura desse ponto relativamente ao solo e dado pela equação seguinte:

$$d_{g_{pixel}} = (h_k - h_{o_{pixel}}).\tan \alpha_{pixel} \ [cm] \tag{29}$$

Substituindo com (28), $d_{g_{pixel}}$ passa a depender somente do valor medido do *Kinect* $d_{k_{pixel}}$:

$$d_{g_{pixel}} = d_{k_{pixel}}.\cos \alpha_{pixel} \,.\tan \alpha_{pixel} \ [cm] \tag{30}$$

Para conhecer a distância a um objeto, precisamos de saber qual é o ponto do objeto que está mais perto do robô. Considerando um *pixel,* isto é com ângulo fixo, igualando as equações (24) e (25), expressando a em ordem a $d_{g_{pixel}}$, e substituir valor real $d_{k\_ref_{pixel}}$ por medição $d_{k_{pixel}}$, obtemos:

$$d_{g_{pixel}} = h_k . \tan\cos^{-1}\frac{h_k}{d_{k_{pixel}}} \quad (31)$$

Como a equação anterior é crescente em função de valores de $d_{k_{pixel}}$, os valores extremos de $d_{g_{pixel}}$ ocorrerão para valores extremos de $d_{k_{pixel}}$. O valor de distância mínimo d_{g_i} e d_{g_f} máximo no eixo YY serão determinados a partir de valores mínimos e máximos de $d_{k_{pixel}}$ da respetiva mancha. A distância até ao centro do comprimento (eixo YY) do objeto é a média da distância inicial com a final. As equações da distância inicial, final e central são as seguintes respetivamente:

$$d_{g_i} = d_{k_i} . \cos\alpha_i . \tan\alpha_i \ [cm] \quad (32)$$

$$d_{g_f} = d_{k_f} . \cos\alpha_f . \tan\alpha_f \ [cm] \quad (33)$$

$$d_{g_c} = \frac{d_{g_i} + d_{g_f}}{2} \ [cm] \quad (34)$$

A figura 4-20 ilustra num corte lateral os diferentes parâmetros estimados neste e nos pontos anteriores.

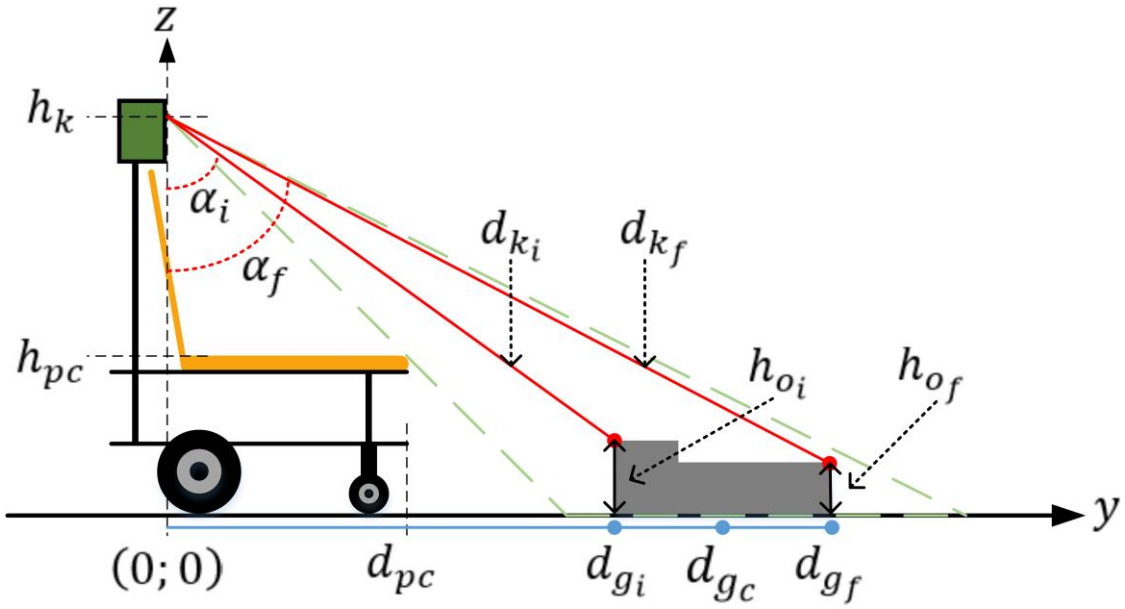

Figura 4-20 – Corte lateral do robô e seu campo de visão. Caracterização do obstáculo (dimensões e distâncias).

Distância Obstáculo-Faixa

Este ponto refere-se exclusivamente ao caso específico do obstáculo estar ao lado da faixa permitindo que esta seja vista entre as distâncias d_{g_f} e d_{g_f}, comprimento do objeto. O ângulo horizontal que existe entre a faixa e o ponto do obstáculo mais perto da faixa é dado por a diferença entre a abcissa desse ponto e a posição da faixa nessa mesma linha da imagem:

$$\beta_{obj-faixa} = \frac{pixel_{obj\_faixa}(x) - Faixa_{l_{obj\_faixa}}}{\sigma} \ [°] \tag{35}$$

O *pixel* mais próximo da faixa é o *pixel* mais a esquerda da mancha se o obstáculo está a direita da faixa e vice-versa. A distância da faixa ao ponto mais próximo dela do obstáculo tem por parâmetros o ângulo que os separa do ponto de vista do robô $\beta_{obj-faixa}$ e a distância medida pelo *Kinect* $d_{k_{faixa}}$ na faixa. A equação seguinte é no entanto uma aproximação pressupondo que a trajetória da faixa é retilínea e que a diferença entre $d_{k_{faixa}}$ e $d_{k_{obstáculo}}$ seja mínima já que pertencem a mesma linha da imagem.

$$d_{obj\_faixa} = \left| 2.\, d_{k_{faixa}} . \tan\frac{\beta}{2} \right| \tag{36}$$

Este cálculo da distância d_{obj_faixa} permite eliminar falsos positivos que apesar de apresentarem relevo relativamente ao solo, não são obstáculos por a distância mínima até a faixa ser superior a metade da largura robô. Por outro lado também permite ao robô aproveitar melhor a área navegável de maneira a se distanciar o menos possível da faixa.

Os diferentes algoritmos referentes ao contorno de obstáculos serão apresentados no ponto 4.4.

4.4 CONTORNO DE OBSTÁCULOS

Neste subcapítulo serão descritos os algoritmos de contorno de obstáculos relativamente a trajetórias fixas. No caso de trajetórias dinâmicas, a posição final do percurso tem influência sobre o lado que o robô escolhe para efetuar o contorno. Em trajetórias fixas, esse lado será definido exclusivamente em função da posição do obstáculo relativamente ao percurso normal (a faixa neste caso). Isto porque o desejável é que o robô se afaste o menos possível da faixa para evitar que o robô se perca resultado de uma elevada acumulação de erros de posição estimados por odometria. Por outro lado, se o obstáculo obstruir a

visualização da faixa, não se pode ter nenhuma certeza quanto a posição da linha após o objeto daí a traseira do obstáculo se tornar a posição objetivo temporariamente.

Perante um obstáculo, o robô sai do estado de seguimento de faixa, para, e toma uma decisão baseada na análise da situação. Essa situação será classificada e atribuída a uma das 3 situações típicas distintas, sendo estes casos:

- Caso 1: Obstáculo perto da faixa. O robô consegue visualizar a faixa desde d_{g_i} até d_{g_f} do obstáculo;
- Caso 2: Obstáculo sobre a faixa. O obstáculo não permite a visualização da faixa;
- Caso 3: Sem caminhos alternativas. É importante o robô reconhecer situações em que não pode, pelo menos de forma segura e garantida, continuar sua tarefa principal que é deslocar-se em direção ao seu objetivo. É por isso necessário implementar algoritmos que façam "desistir" o robô em caso situações demasiadamente complexas.

Essa tomada de decisão acontece exatamente no momento em que um obstáculo atinge a zona de risco delimitada por $d_{g_{min}}$ (50 cm) e $d_{g_{limite}}$ (60 cm) como se pode ver na figura 4-21. O resto da área analisada pelo *Kinect* é usada para efetuar as caracterizações do obstáculo referidas no ponto 4.3.2.5. Os pontos seguintes tratam de caracterizar detalhadamente cada situação de alteração de rota do robô devido a existência de obstáculos no percurso.

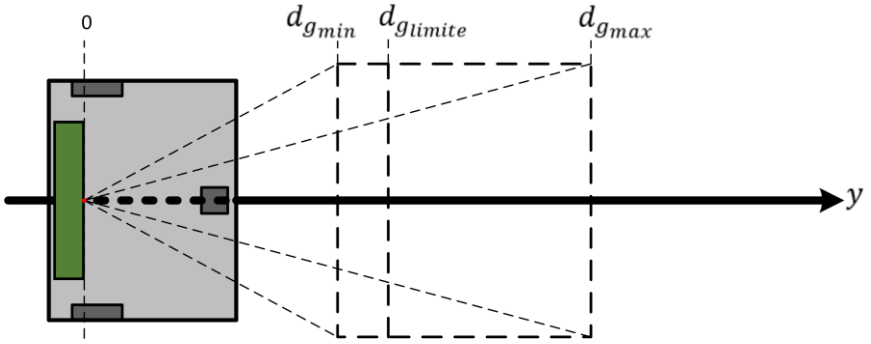

Figura 4-21 – Zona analisada pelo *Kinect* de um ponto de vista aéreo. Caso haja um obstáculo no retângulo delimitado por a distância mínima e a distância limite (60 cm), o robô para e toma uma decisão. A faixa coincide com o eixo YY.

4.4.1 CASO 1: OBSTÁCULO PERTO DA FAIXA

Um exemplo de um caso onde um obstáculo influencia a rota do robô sem se sobrepor a faixa é ilustrado em 4-22. A vantagem neste caso é que o robô dificilmente se pode perder dado que vê a faixa na quase totalidade da manobra. O robô para seu deslocamento na primeira iteração onde forem detetados obstáculos na zona de risco que se situa entre 20 a 30 cm da parte frontal deste.

Averiguar se é possível visualizar a faixa ao longo do comprimento do objeto (de d_{g_i} a d_{g_f}) é o primeiro teste a ser feito (teste 1). Caso o resultado desse teste seja positivo, a situação em questão pertence ao primeiro caso (Caso 1).

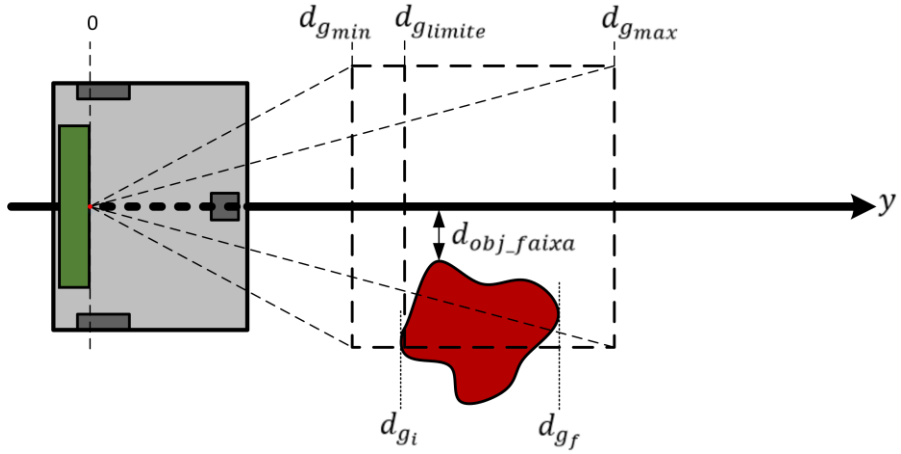

Figura 4-22 – Situação pertencente a primeira categoria onde o obstáculo não impede a visualização da faixa ao longo do seu comprimento.

A primeira etapa do caso 1 é aceder aos dados referentes ao obstáculo e sua posição relativamente a faixa. Isto é d_{g_i}, d_{g_f}, d_{obj_faixa} e em que lado da faixa está o obstáculo. Estes dados são todos eles adquiridos no momento em que o robô parou devido a proximidade do obstáculo (obstáculo na zona de risco).

O segundo passo é a etapa de execução, dado que o robô já sabe quanto vai ter de se deslocar para o lado livre para passar, assim como sabe que distância falta para passar o obstáculo. O princípio do contorno neste primeiro caso é simular uma faixa imaginária semelhante à real mas deslocada de tal forma que o obstáculo já não esteja na trajetória do robô. Após ter passado o obstáculo, o robô volta a seguir a faixa real. Voltar a seguir a faixa real não causa muitos problemas dado que o campo de visão do robô abrange uma largura suficiente para qualquer distância d_{obj_faixa} relevante. Um obstáculo só ficará na trajetória do robô (obstáculo relevante) se d_{obj_faixa} for inferior a metade da largura do robô. É aconselhado adicionar uma margem suplementar por segurança, M_{s_1}, dado que não existem

nenhuns sensores laterais no robô. A distância entre faixa real e imaginária no eixo XX, d_{ff}, é dada pela seguinte equação:

$$d_{ff} = \left(\frac{Larg_{robô}}{2} + M_{s_1}\right) - d_{obj\_faixa} \tag{37}$$

Se o regresso à faixa real é direto por esta estar na imagem RGB, o mesmo não se pode dizer da ida para a faixa imaginária que não permite deixar escapar a faixa real da imagem dado que serve de referência à faixa imaginária. É por isso necessário deslocar o robô perpendicularmente a faixa (eixo XX) mas mantendo a mesma orientação. Só um robô omnidirecional pode fazer esse deslocamento num só passo. Com um robô diferencial é necessário 2 rotações (90º) separadas por uma translação (d_{ff} cm). Durante esta translação, o aparecimento de qualquer novo obstáculo na zona de risco terá por consequência a paragem definitiva do robô. A figura 4-23 ilustra a manobra de afastamento da faixa no eixo XX mantendo a orientação inicial. Considerou-se que o objetivo do *Kinect* era coincidente com o centro do eixo do robô.

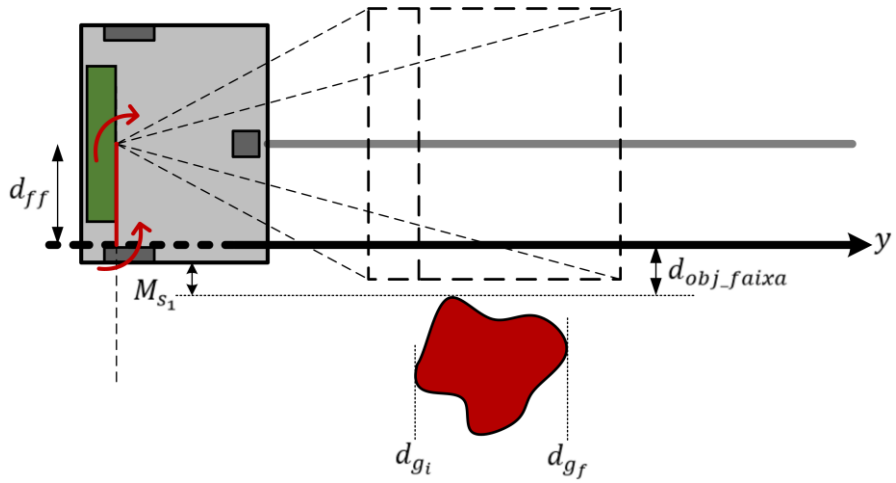

Figura 4-23 – Manobra de afastamento da faixa no eixo XX mantendo orientação.

Se o robô conseguir concluir a manobra de afastamento da faixa com o caminho livre, começará a seguir a faixa imaginaria, que tem um deslocamento conhecido da faixa real, até ao fim do obstáculo. Essa distância é igual a d_{g_f} podendo ser acrescentada de uma margem de segurança, M_{s_2}, devido a falta de sensores laterais. Neste caso foi usado $M_{s_2} = 0$. Mais uma vez, qualquer obstáculo na zona de risco provocará uma paragem definitiva.

Quando a translação atingir a distância pré-determinada, o robô passará a seguir a faixa real que está visível na imagem dado que tem continuado de servir de referência.

A figura 4-24 ilustra a trajetória (pode ser uma trajetória curvada) que faz passar o robô a uma distância M_{s_1} do obstáculo e o retorno ao estado de seguimento de faixa (real).

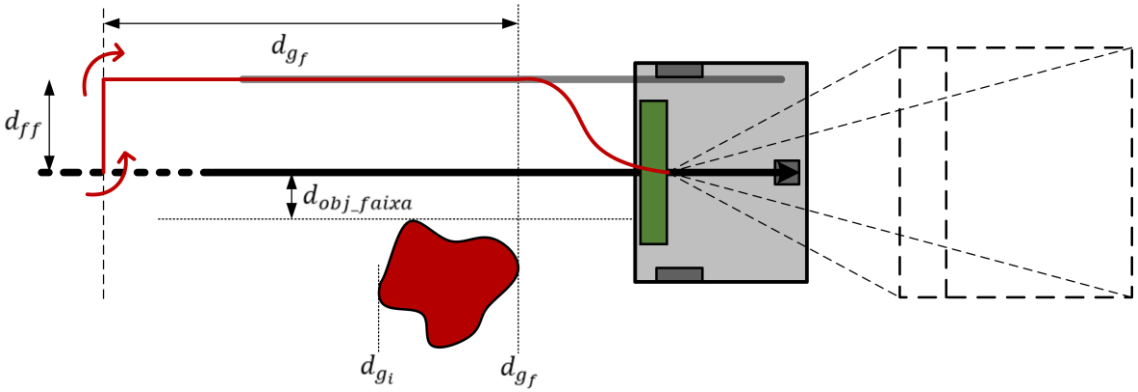

Figura 4-24 – Seguimento de faixa imaginária que permite uma passagem do robô a M_{s_1} do obstáculo seguido do regresso ao estado normal de seguimento de faixa real.

4.4.2 Caso 2: Obstáculo Sobre a Faixa

Numa situação onde um obstáculo corta a continuidade da faixa devido a sua sobreposição, é necessário que o robô consiga chegar á parte da faixa que está após o obstáculo com recurso a odometria, características e posição do obstáculo.

Considerou-se que o obstáculo está numa parte da faixa retilínea já que é impossível conhecer a real posição da faixa após o obstáculo em trajetórias aleatórias. Se o início da curva fosse ainda visível, poderia estimar-se a posição baseando-se na curvatura da faixa mas esta pode não ser constante ao longo da curva. Por outro lado, a visualização da faixa por cima do obstáculo permitiria uma melhor estimação mas só funcionaria com obstáculos com pouca altura.

Com sensores laterais seria possível usar um método parecido com o do Caso 1 já que a trajetória é retilínea. Ao passar o obstáculo teria de procurar a faixa porque estaria fora do alcance do *Kinect*. Para isso, o valor d_{g_f} teria de ser fiável. Recorda-se d_{g_f} é calculado em função do ponto do objeto mais afastado do *Kinect*, ora maior parte do conjunto de pontos que satisfazem esse requisitos estão inacessíveis do ponto de vista do robô.

Optou-se então por encontrar uma posição fora da faixa onde o robô possa determinar onde a faixa está de novo livre. Essa posição encontra-se no eixo do centro do comprimento do objeto no eixo YY de forma a que o obstáculo seja visto de um ponto de vista privilegiado. A distância dessa posição ao ponto do obstáculo mais próximo (extremidade lateral do obstáculo no momento de perceção do obstáculo no lado menos obstruído) deve ser no

mínimo 38 cm (acrescentada de uma margem de segurança $M_{s_3} = 5$ cm) dado que essa é a distância entre o centro de rotação do robô e suas esquinas mais afastadas para evitar posteriormente colisões na mudanças de direção devido a falta de deteção de obstáculos justo após o robô (um intervalo de 20 cm).

Ao aparecer um obstáculo na zona de risco, o robô estando no estado normal de seguimento de faixa para, e averigua se a faixa é visível ao nível do obstáculo (Teste 1). O resultado desse teste sendo negativo, será necessário determinar se existe um caminho alternativo para o robô (Teste 2) para a situação pertencer ao caso 2, caso contraio pertencerá ao caso 3.

A figura 4-25 ilustra o momento em que o robô para, recolhe informação e define qual o caso em questão numa situação onde o obstáculo está sobreposto a faixa. Sendo atribuída a situação atual ao caso 2, o robô terá de decidir por que lado contornar o obstáculo para minimizar o afastamento da faixa. O obstáculo terá necessariamente uma extremidade lateral em cada metade (horizontal) da imagem resultado da deteção de obstáculos. O menor afastamento (d_{obj_centro}) das extremidades laterais do objeto ao centro da imagem definirá o percurso menos obstruído. A posição no eixo XX desse ponto também é necessária por servir de referência a posição intermediária, P_I, que ocupará o robô fora da faixa. A posição em YY desse ponto é coincidente com o eixo do centro do objeto calculado inicialmente.

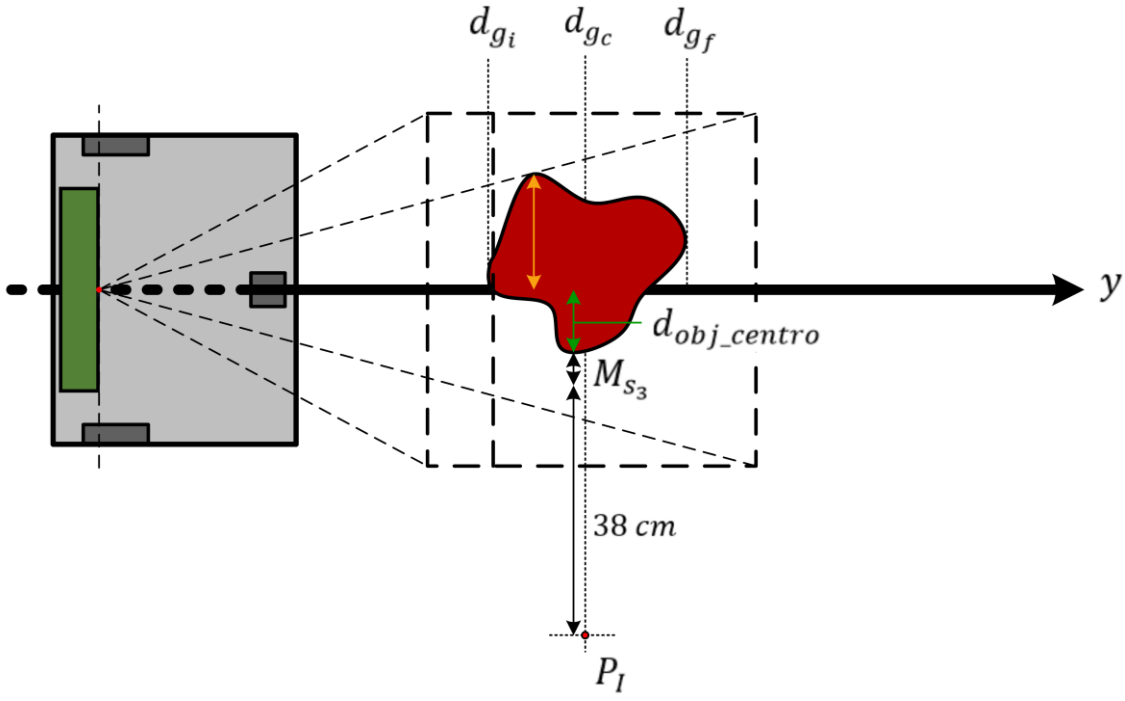

Figura 4-25 – Paragem devido a obstáculo caracterizada como sendo do caso 2. Cálculo da posição intermedia.

A distância d_{obj_centro} é calculada em função das equações (35) e (36) substituindo a posição da faixa pelo centro da imagem já que a faixa provavelmente não está visível nessa linha da imagem. Considerando mais uma vez o centro de rotação do robô coincidente com o objetivo do *Kinect* e o pixel medido aquele usado para calcular d_{obj_centro}, o ângulo (φ_1) que terá de rodar o robô para o lado menos obstruído é dado por:

$$\varphi_1 = \tan^{-1}\frac{(d_{obj_{centro}} + M_{s_3} + 38)}{d_{k_{pixel}}} \tag{38}$$

No caso 2 é mais fácil usar um novo referencial coincidente com a figura 4-25 do que o referencial global do mapa-mundo. Após girar φ_1 para o lado menos obstruído, o robô iniciará uma translação até alcançar ou o valor de d_{g_c} no eixo YY, ou o valor de $(d_{obj_{centro}} + M_{s_3} + 38)$ no eixo XX que por erros relacionados com odometria poderão não serem simultâneos. Ao atingir qualquer uma das duas condições anteriores, o robô terá de ajustar sua orientação conforme a inicial girando φ_1 para o lado mais obstruído. A figura 4-26 ilustra a evolução da trajetória do robô.

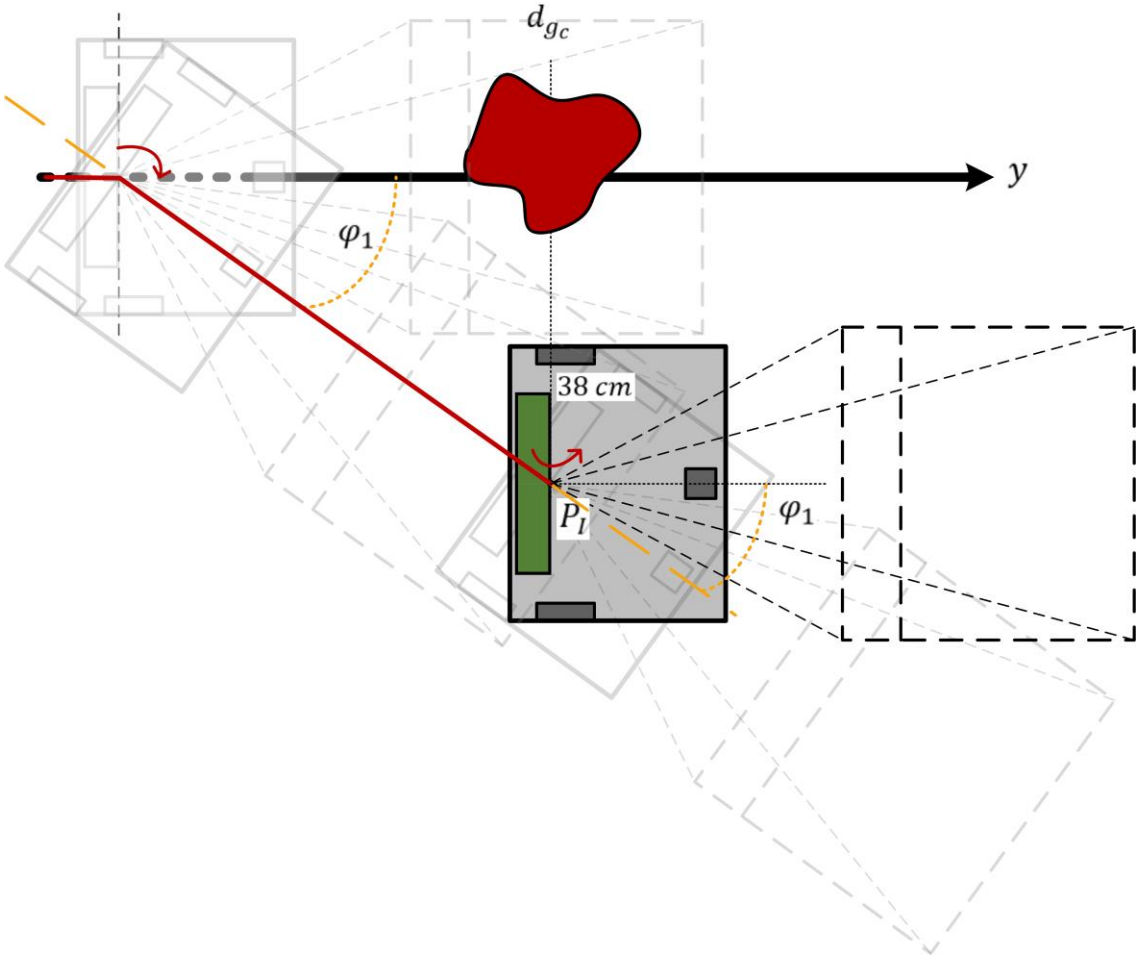

Figura 4-26 – Evolução da trajetória do robô até ao ponto intermedio.

A manobra seguinte consiste em definir um ângulo de retorno a faixa, φ_2, de forma a que o robô possa voltar a sobrepor-se a trajetória original (sem obstáculos) sem colisões. O robô gira no sentido mais obstruído até aparecer o obstáculo na zona de perigo ou na zona de analise de obstáculos, em seguida gira ligeiramente para o lado oposto até ele voltar a sair da zona de visão do robô de forma a ter um caminho livre de obstáculos. A orientação do robô nesse momento define φ_2 como se pode ver na figura 4-27.

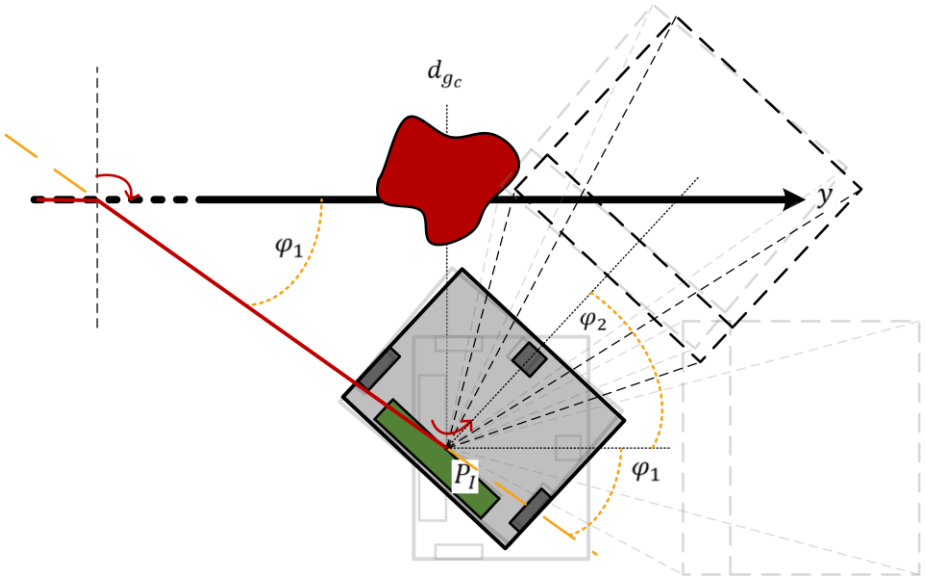

Figura 4-27 - Manobra para definir ângulo de regresso à faixa φ_2.

O robô orientado no ângulo conhecido φ_2, terá de andar em linha reta até alcançar a posição inicial no eixo XX e em seguida girar φ_2 para o lado menos obstruído para ficar com a mesma orientação do que no momento inicial (perceção do obstáculo) e da faixa supostamente (figura 4-28). Antes de retornar ao estado de seguimento normal da linha, o robô terá de confirmar que a faixa está na imagem, caso contrário o robô passará para o estado de paragem permanente. Também em caso de aparecimento de um novo obstáculo na zona de risco durante as translações, o robô passará para o estado de paragem.

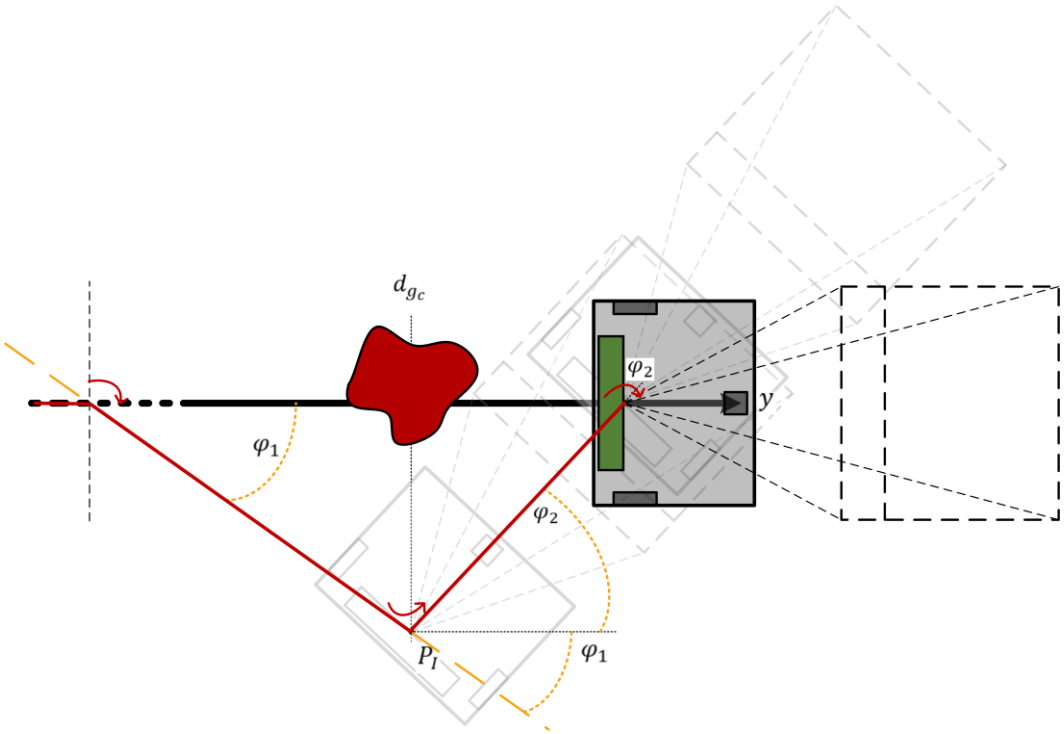

Figura 4-28 - Conclusão de contorno de um obstáculo do segundo caso.

4.4.3 Caso 3: Sem Caminhos Alternativos

Por fim, nos casos em que o robô é incapaz de escolher um caminho alternativo devido ao tamanho do obstáculo será preferível parar permanentemente devido ao grande afastamento da faixa que implicaria. O fato de não ser possível medir nenhuma das duas distâncias entre extremos laterais do objeto e a faixa (devido ao objeto ser demasiado grande para caber na imagem) implica que o resultado do Teste 2 é negativo e consequentemente, a situação em questão pertence ao caso 3.

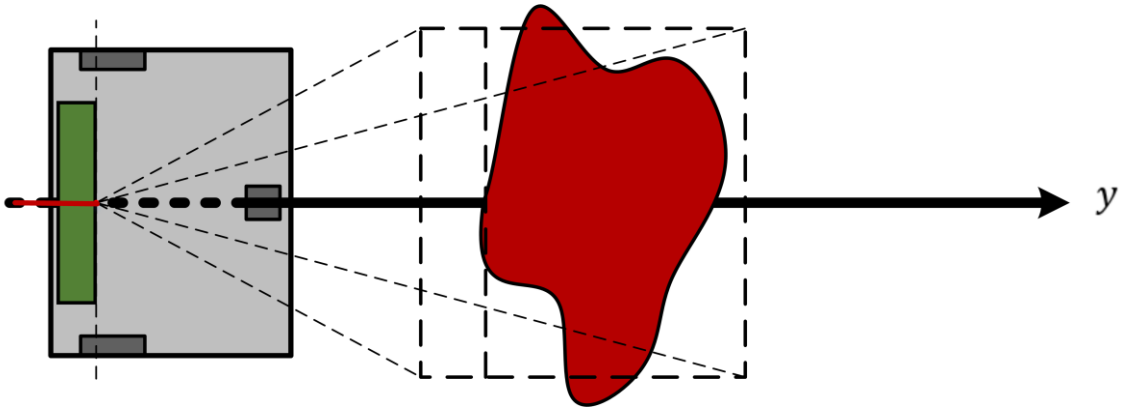

Figura 4-29 – Situação pertencente ao terceiro caso onde o objeto é demasiado grande para ser medido, daí originando a paragem permanente do robô.

Com um robô omnidirecional, as etapas do caso 1 e 2 teriam sido facilitadas. No presente caso, para o robô poder determinar o lado menos obstruído, teria de girar para ambos os lados para medir as larguras do objeto de cada lado da faixa. Já não bastaria analisar uma vez o objeto num referencial mas sim em três vezes em três referenciais diferentes complicando bastante o problema.

No subcapítulo seguinte será apresentada a rede de Petri global do sistema de controlo.

4.5 REDE DE PETRI DO SISTEMA

Com o objetivo de simplificar o sistema de controlo, foi desenhada uma rede de Petri do sistema com os diferentes estados que o robô pode assumir, assim como as transições que permitem uma mudança de estado (figura 4-30). A rede seguinte só possui um *token* e começa no estado de verificação que a faixa está na imagem. Os casos 1, 2 e 3 estão referenciados em 4.4.1, 4.4.2 e 4.4.3 respetivamente. O Teste 1 averigua se há faixa a nível do obstáculo e o Teste 2 averigua se existe um caminho alternativo para o robô.

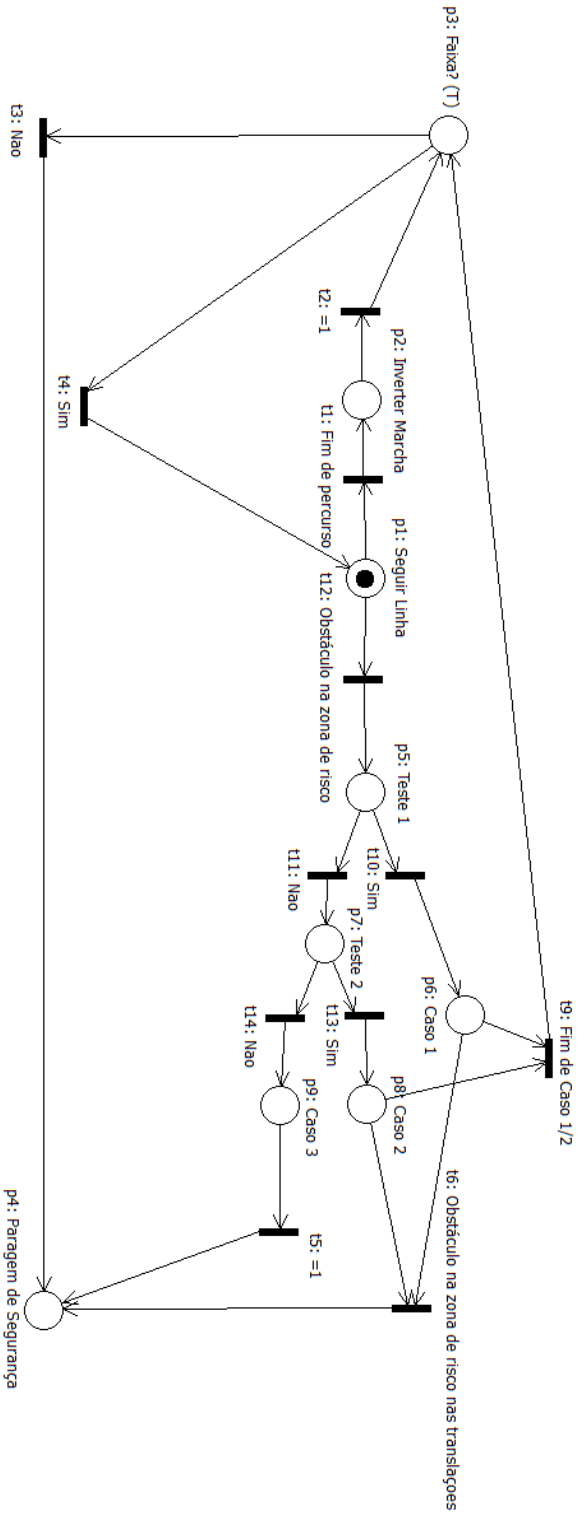

Figura 4-30 – Rede de Petri do Sistema de Controlo do Robô[9].

[9] A rede de Petri foi desenhada com recurso ao *software* PndK – Petri Nets Development Toolkit – v1.0.0-r237, ESTiG.

Capítulo 5

TESTES E RESULTADOS

Neste capítulo são apresentados os resultados dos diferentes testes a que foi submetido o robô. Esses testes consistem em seguir a faixa do percurso, evitar obstáculos, e registar o percurso do robô com base na odometria.

5.1 TESTE DE SEGUIMENTO DE FAIXA

O robô segue a faixa de cor branca enquanto esta não tenha curvas demasiadamente apertadas. Essa limitação tem por causa a zona a frente do robô não abrangida pelo *Kinect* (20 cm). Pode observar-se nas figuras de 5-2 a 5-4 que o robô saiu de cima da faixa devido a esse mesmo facto mas também à elevada curvatura da trajetória. As figuras seguintes são *frames* retirados de um vídeo onde o robô segue a faixa sem haver qualquer obstáculo no percurso. Note-se que neste teste desprezou-se a inversão de marcha nas extremidades do percurso.

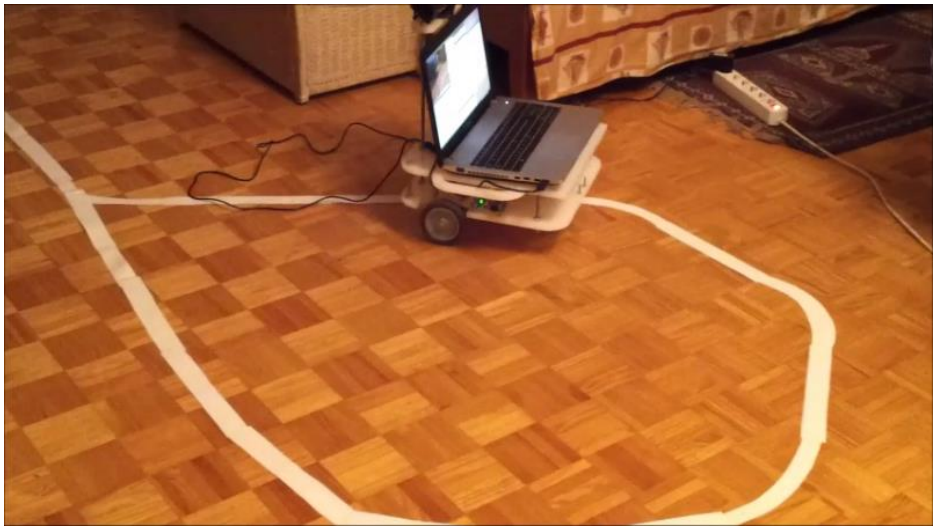

Figura 5-1 – Seguimento de Faixa, *frame* nº 1.

Figura 5-2 - Seguimento de Faixa, *frame* nº 2.

Figura 5-3 - Seguimento de Faixa, *frame* nº 3.

Figura 5-4 - Seguimento de Faixa, *frame* nº 4.

Figura 5-5 - Seguimento de Faixa, *frame* nº 5.

Figura 5-6 - Seguimento de Faixa, *frame* nº 6.

Figura 5-7 - Seguimento de Faixa, *frame* nº 7.

5.2 TESTE DE CONTORNO DE OBSTÁCULO

Neste teste foi colocado um obstáculo sobre o percurso reto da faixa. O obstáculo, um garrafão de água, está sobre a faixa e pertence a um caso do segundo tipo, este está ligeiramente mais a direita da faixa favorizando assim o contorno pela esquerda. As figuras seguintes representam diversos *frames* de um vídeo onde o robô contorna o obstáculo pelo lado mais favorável.

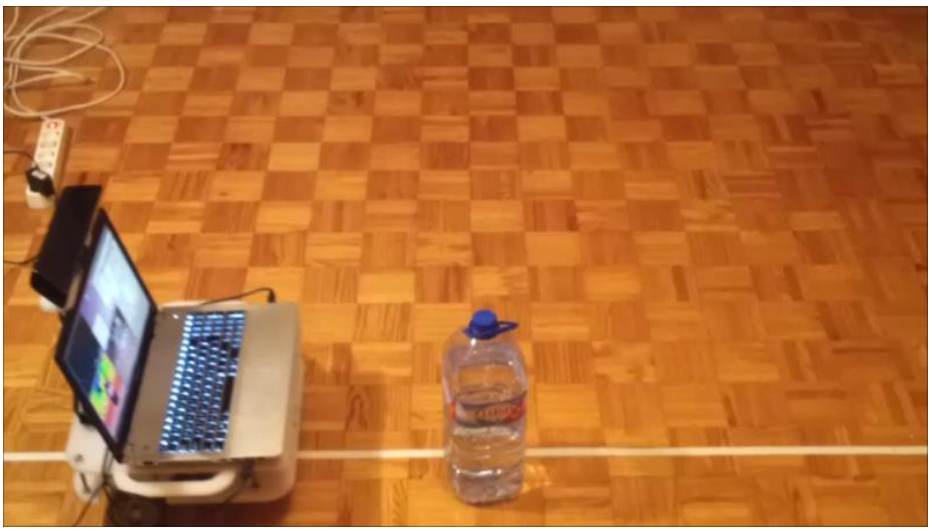

Figura 5-8 - Contorno de obstáculo (do Caso 2), *frame* nº 1.

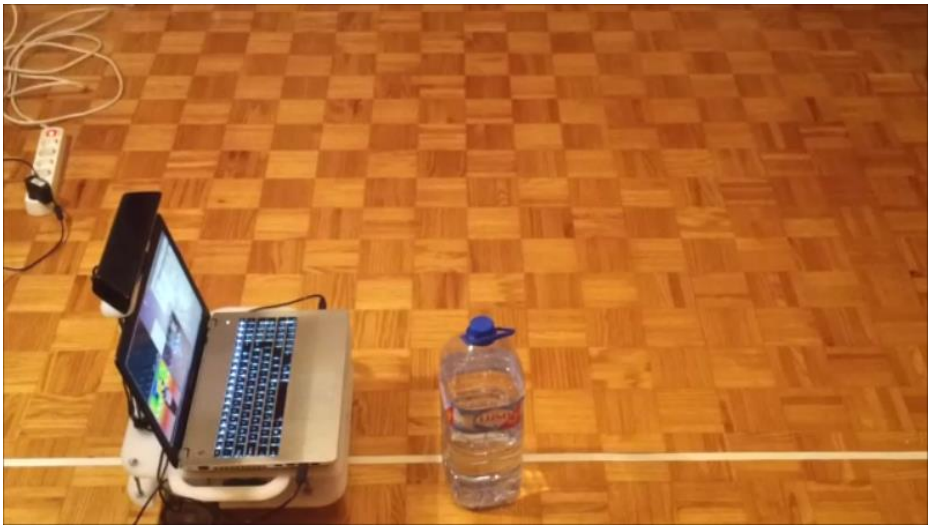

Figura 5-9 - Contorno de obstáculo (do Caso 2), *frame* nº 2.

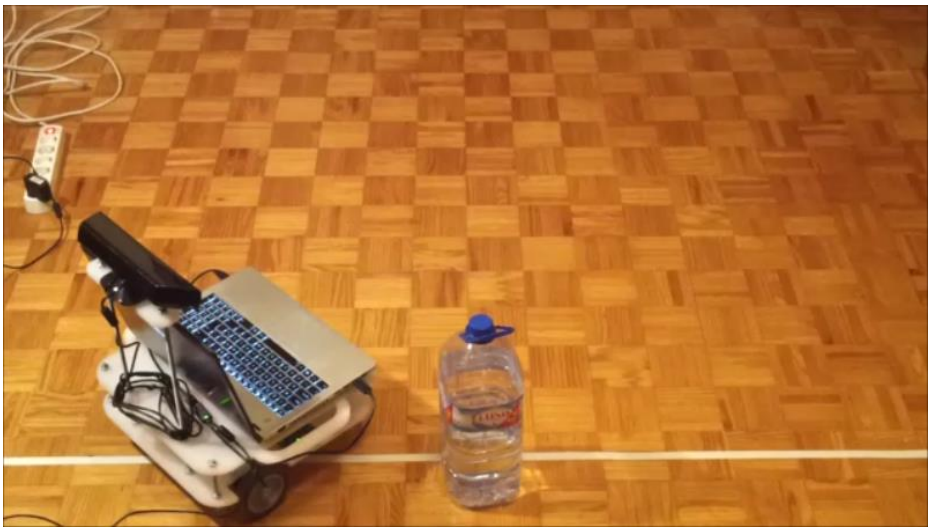

Figura 5-10 - Contorno de obstáculo (do Caso 2), *frame* nº 3.

Figura 5-11 - Contorno de obstáculo (do Caso 2), *frame* nº 4.

Figura 5-12 - Contorno de obstáculo (do Caso 2), *frame* nº 5.

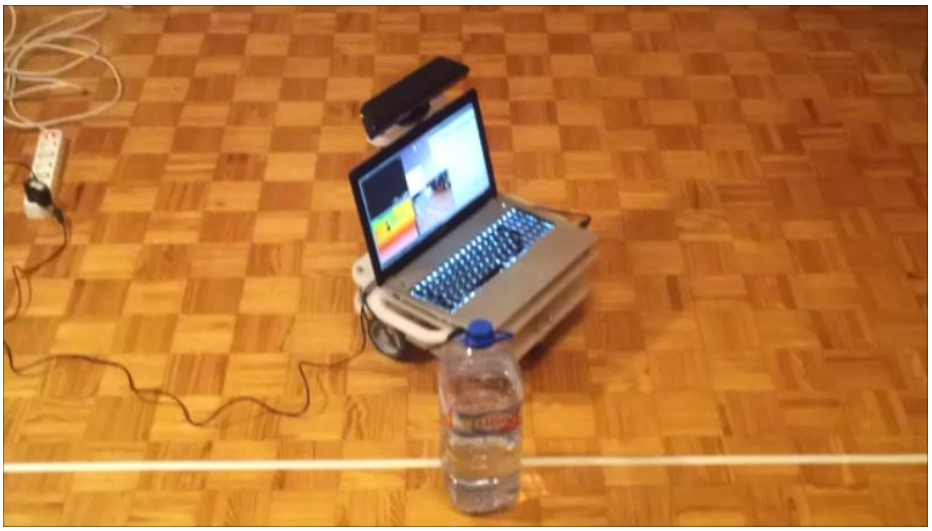

Figura 5-13 - Contorno de obstáculo (do Caso 2), *frame* nº 6.

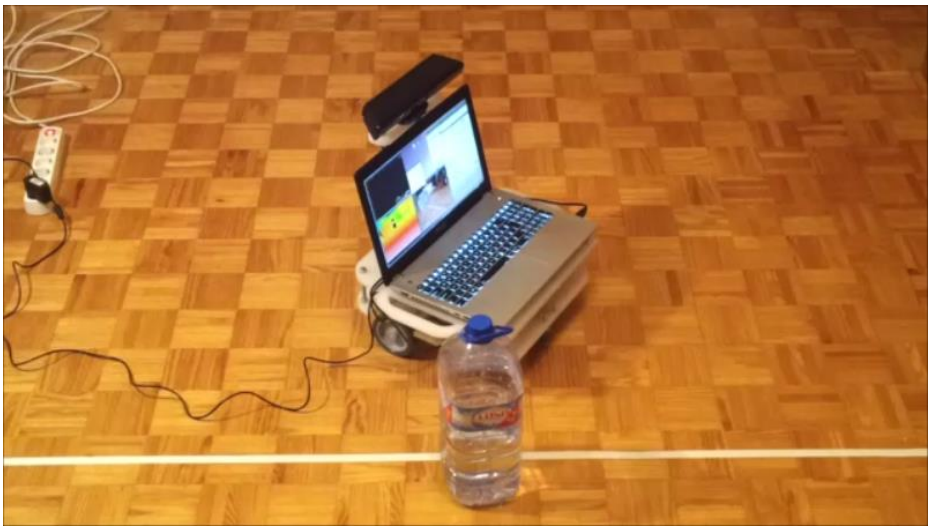

Figura 5-14 - Contorno de obstáculo (do Caso 2), *frame* nº 7.

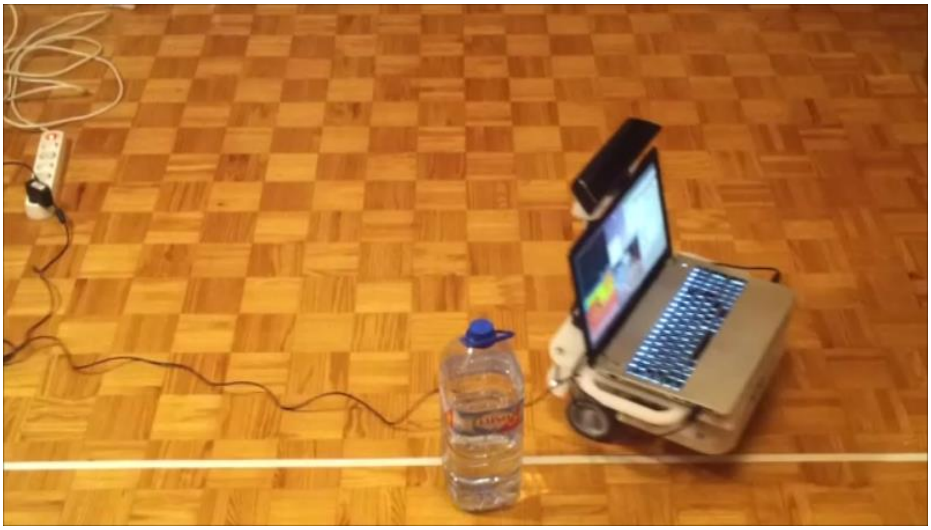

Figura 5-15 - Contorno de obstáculo (do Caso 2), *frame* nº 8.

Figura 5-16 - Contorno de obstáculo (do Caso 2), *frame* nº 9.

Figura 5-17 - Contorno de obstáculo (do Caso 2), *frame* nº 10.

5.3 ESTIMAÇÕES DE PERCURSOS BASEADAS EM ODOMETRIA

O robô colocado numa pista desta vez semelhante a da figura 5-18, foi registado graficamente o percurso estimado por odometria do robô no mapa-mundo. Na figura 5-19 pode observar-se três percursos estimados por odometria de 3 viagens distintas do robô.

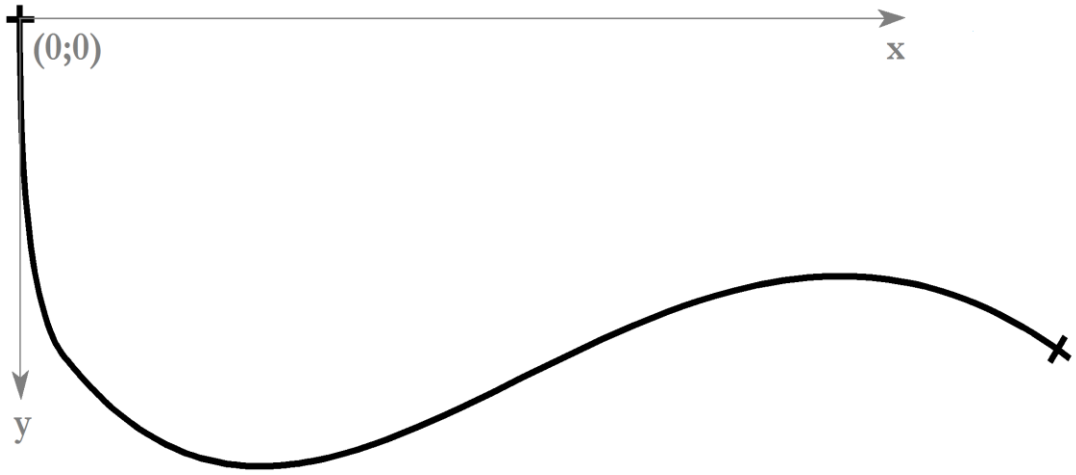

Figura 5-18 - A configuração do trajeto considerado.

Nas figuras 5-20 e 5-21 são representados os percursos estimados por odometria durante contornos de obstáculos similares ao mencionado em 5.2, um obstáculo sobre uma trajetória retilínea. Na esquina inferior direita são monitorizadas a orientação e coordenadas atuais assim como o estado em que se encontra o robô[10]. Tudo igual para a figura 5-22 à diferença que o obstáculo desta vez está a obstruir mais o lado esquerdo.

[10] Os diferentes estados que o robô pode assumir foram mencionados na rede de Petri do subcapítulo 4.5.

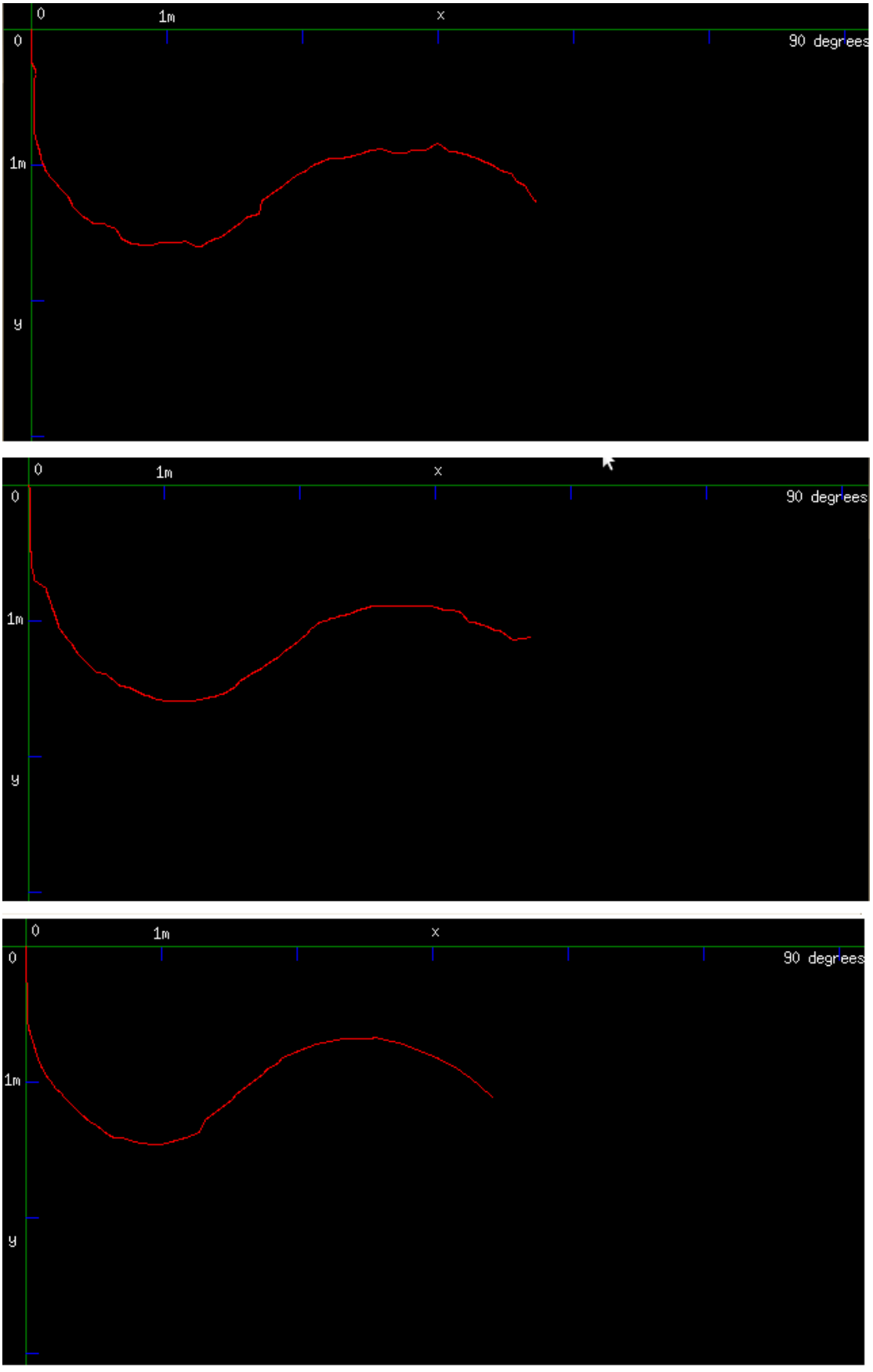

Figura 5-19 - Percursos estimados por odometria ao longo da pista representada na figura 5-18.

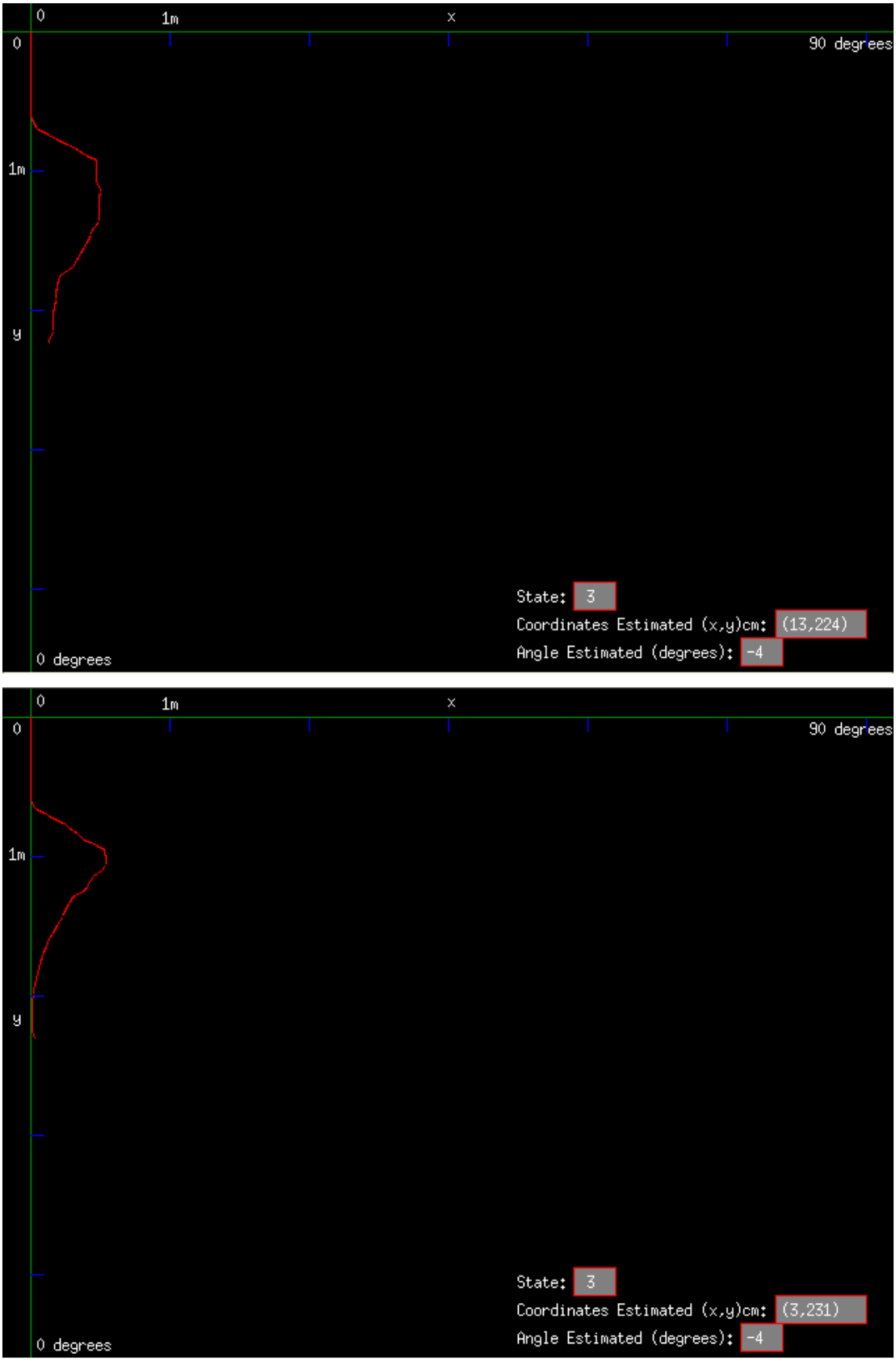

Figura 5-20 - Percursos estimados por odometria ao longo do contorno de obstáculo pela esquerda (1).

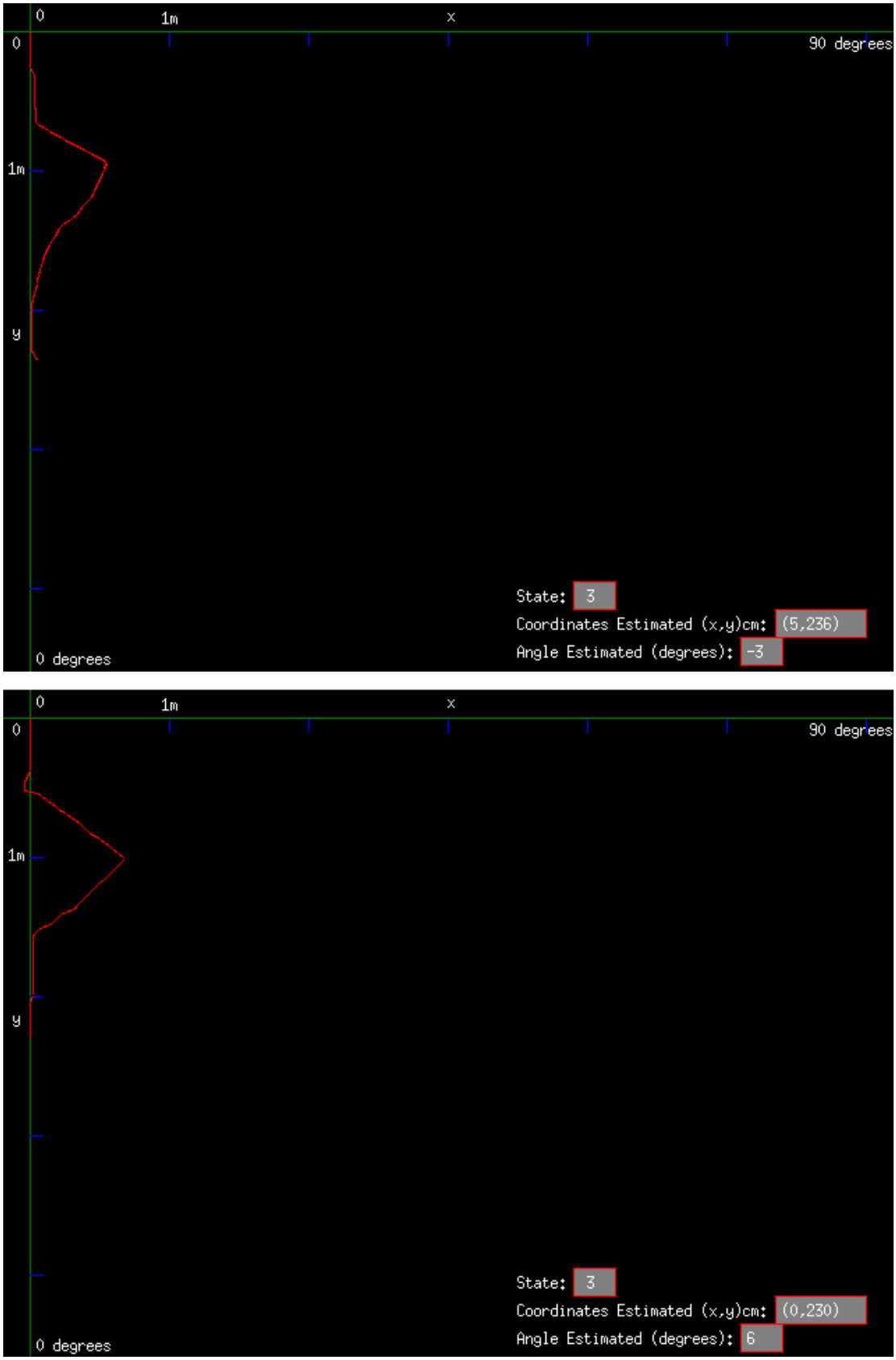

Figura 5-21 - Percursos estimados por odometria ao longo do contorno de obstáculo pela esquerda (2).

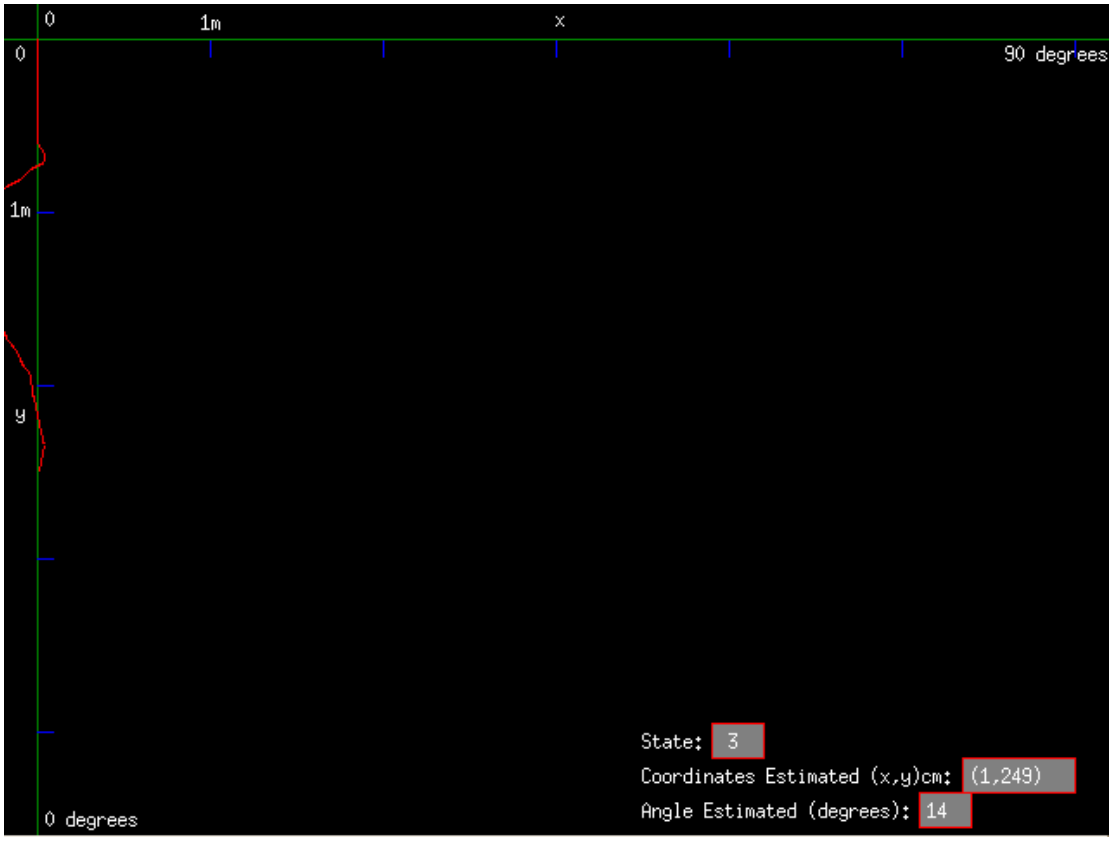

Figura 5-22 - Percursos estimados por odometria ao longo do contorno de obstáculo pela direita.

No capítulo seguinte serão apresentadas as conclusões deste estudo assim como propostas de melhorias e de trabalhos futuros.

Capítulo 6

CONCLUSÕES

6.1 SÍNTESE E CONCLUSÕES

O objetivo principal do trabalho era a conceção de um robô móvel diferencial que validasse os algoritmos de deteção de obstáculos baseados no sensor *Kinect*. Após a conceção e teste manual do robô, foi necessário definir um método de navegação para definir o percurso normal tendo sido escolhido o seguimento de faixas. O método de localização implementado foi a estimação por odometria baseada nos *encoders* dos motores, sendo memorizado e representado num mapa-mundo o percurso passado do robô. Invertendo a marcha e atualizando a posição nas extremidades do percurso, sem obstáculos o ciclo de trabalho do robô não tem fim. Para evitar colisões devido a existência de obstáculos na trajetória do robô sem comprometer totalmente as hipóteses de realizar sua tarefa, foram implementados algoritmos de deteção e contorno de obstáculos de qualquer forma e cor[11]. A inversão de marcha e contorno de obstáculos são efetuados em função da posição e dimensões do objeto e com o auxílio da odometria. Foi implementada uma máquina de

[11] Devido ao método de navegação ser o seguimento de faixa (faixa branca), será preferível que os obstáculos não sejam da mesma cor.

estados virtual para definir os diferentes estados do robô. O *Kinect* revelou-se uma boa solução para deteção de obstáculos para robótica móvel dado que é sensível a cor e à profundidade à semelhança do olho humano, tendo uma resolução suficiente para este efeito e com um *frame rate* de 30 ms. Contudo, a escolha do seu posicionamento e orientação depende do tipo de robô, do seu propósito e do ambiente em que está inserido. Aquilo que se quer dos sensores num robô móvel é que abrangem a máxima área ao redor do robô, por isso o sensor ideal seria uma câmara de profundidade (RGB-D) com ângulo de visão de 360º, à semelhança da visão omnidirecional RGB usada no futebol robótico.

6.2 MELHORIAS E TRABALHOS FUTUROS

6.2.1 ESTRUTURA MÓVEL

Este ponto é dedicado às modificações que seriam efetuadas se a estrutura móvel fosse concebida no final deste trabalho, tais como:

- O auxílio de giroscópios no robô para estimar a orientação dado que é o ponto fraco da odometria baseada nos *encoders*.
- Para a atualização da localização poderia recorrer-se ao uso de *landmarks* já que o *Kinect* possui uma câmara RGB.
- Sendo a estrutura usada um robô diferencial, o uso de sensores infravermelhos nas laterias do robô seria vantajoso na medida em que reduziria as etapas do contorno de obstáculos, mas também auxiliaria o *Kinect* na deteção já que este não abrange as zonas que estão lado a lado com o robô.
- Uma das mais importantes modificações seria aumentar a altura do *Kinect* na estrutura. A primeira vantagem é que o *Kinect* aproveita melhor o seu alcance angular, o que aumentaria a zona analisada. A zona de ângulo morto a frente do robô seria reduzida. Suficientemente alto até abrangeria as laterais e traseira do robô para ter uma visão omnidirecional. Finalmente, as manchas pretas na imagem de profundidade provocadas por objetos demasiadamente próximos deixariam de criar uma ambiguidade pois estas não poderiam constituir assim nenhum obstáculo.

- Continuando com o posicionamento do *Kinect*, este deveria ter o recetor IV e a câmara RGB centradas no robô de um ponto de vista frontal, o que é impossível devido à estrutura desta versão do *Kinect* por estes estarem separados. Também, convém que ele esteja na mesma linha vertical que o centro de rotação do robô diferencial para facilitar cálculos de distância e ângulos de rotação.
- Neste projeto, somente o *Kinect* necessitou alimentação da rede, o uso de um adaptador para este ser alimentado por a bateria seria necessário para o robô ser energeticamente autónomo.
- Finalmente, se o robô fosse omnidirecional, poderia fazer uma translação mudando a sua orientação e a do *Kinect*, e assim aumentar a zona do espaço analisada. Também nos contornos de obstáculos o robô omnidirecional teria necessitado menos etapas intermédias.

6.2.2 SISTEMA DE CONTROLO

Neste ponto serão apresentadas as melhorias que deveriam ser feitas respondendo a lacunas ou falhas do sistema de controlo, tais como:

- A deteção de obstáculo pelo método desenvolvido é computacionalmente mais leve que o uso de *Depth Point Cloud* pois só é uma parte deste último, a única útil para este trabalho. As *Depth Point Cloud* também reconhecem o solo no objetivo de não o incluir na reconstituição virtual em 3D.
- O robô poderia trabalhar completamente às escuras se fosse alterado o sistema de navegação que necessita de luz para identificação da faixa.
- Contar os obstáculos visíveis num dado momento, referenciá-los e guardar suas localizações em memória no mapa-mundo criando um tipo de mapeamento.
- O mapa-mundo deveria ter um *zoom* automático que não permita que o robô saia da janela de visualização.
- A monitorização remota da totalidade dos dados do robô para vigilar a distância o estado do robô. Daí, seria vantajoso um tipo de rede de Petri dinâmica que indicaria o estado atual e passado do robô.

Referências

[1] Eduardo António da Silva Santos. Logística baseada em AGVs. Tese de mestrado, Faculdade de Engenharia da Universidade do Porto, 2013.

[2] Paulo Leitão. Tecnologias de Sistemas de Automação Industrial - Sistemas de Transporte Automático. *Slides* de Sistemas de Automação, IPB, 2010.

[3] David Da Silva Lima. Localização absoluta de robôs móveis em ambientes industriais. Tese de mestrado, Faculdade de Engenharia da Universidade do Porto, 2010.

[4] Rui Paulo Pinto da Rocha. Desenvolvimento de um sistema de gestão de AGVs. Tese de mestrado, Faculdade de Engenharia da Universidade do Porto, 1998.

[5] Rui Paulo Rocha. Estado da arte da robótica móvel em Portugal. Tese de mestrado, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, 2001.

[6] Roberto José Magalhães da Silva. Localização de AGVs industriais baseada em marcadores. Tese de mestrado, Faculdade de Engenharia da Universidade do Porto, 2011.

[7] T. Ganesharajah, N.G. Hall, e C. Sriskandarajah. Design and operational issues in agv-served manufacturing systems. *Annals of Operations Research*, 76:109 – 54, 1998. URL: http://dx.doi.org/10.1023/A:1018936219150.

[8] Creform. Creform fts. Documentação de apresentação do AGV da Creform. URL: http://www.creform.de/en/produkte/agv/general/.

[9] Sick Group. Sensores de obstáculos a Laser. Visitado a 18/10/13. URL: http://www.sick.com/group/EN/home/products/product_portfolio/optoelectronic_protective_devices/Pages/safetylaserscanners.aspx

[10] José Alexandre de Carvalho Gonçalves. Controlo de *robots* omnidireccionais. Tese de mestrado, Universidade do Porto, 2005.

[11] CSE. Robô omnidirecional. Visitado a 18/10/13. URL: http://www.cse.iitb.ac.in/~erts/robotics.html

[12] Wikimedia. Movimentos de um robô omnidirecional. Visitado a 18/10/13. URL: http://commons.wikimedia.org/wiki/File:Robot_omnidirectional_drive.PNG

[13] João Henrique Rodrigues Costa. Implementação de Fusão Sensorial para Localização de um Veículo Autônomo. Tese de Mestrado, Universidade Federal de Minas Gerais.

[14] Mercedes World News. Visitado a 23/10/13. URL: http://www2.mercedes-benz.com.au/content/australia/mpc/mpc_australia__website/en/home_mpc/passengercars/home/passenger_cars_world/news/september_2013/s-class-intelligent-drive.0001.html

[15] Computerstories. Visitado a 23/10/13. URL: http://computerstories.net/nissan-wants-self-driving-car-by-2020/

[16] Luís Paulo Reis. Coordenação em Sistemas Multi-Agente. Tese de doutoramento, Faculdade de Engenharia da Universidade do Porto, 2003.

[17] José Almeida e Alfredo Martins. Futebol Robótico. Laboratório de Sistemas Autónomos, Instituto Superior de Engenharia do Porto, 2003.

[18] Kitano, H.; Tambe, M.; Stone, P.; Veloso, M.; Noda, I.; Osawa, E.; Asada, M., *The RoboCup Synthetic Agents' Challenge*. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1997.

[19] RoboCup. Visitado a 30/10/2013. URL: http://www.robocup.org/

[20] Steinbuch. A novel ball handling mechanism for the RoboCup middle size league, Visitado a 30/10/13. URL: http://steinbuch.wordpress.com/2012/04/11/a-novel-ball-handling-mechanism-for-the-robocup-middle-size-league/

[21] Antonio J. R. Neves, Jose Luis Azevedo, Bernardo Cunha, Nuno Lau, Joao Silva, Frederico Santos, Gustavo Corrente, Daniel A. Martins, Nuno Figueiredo, Artur Pereira, Luis Almeida, Luis Seabra Lopes, Armando J. Pinho, João Rodrigues and Paulo Pedreiras (2010). *CAMBADA* Soccer Team: from Robot Architecture to Multiagent Coordination, Robot Soccer, Vladan Papi (Ed.), ISBN: 978-953-307-036-0, InTech, Available from: http://www.intechopen.com/books/robot-soccer/cambada-soccer-team-from-robot-architecture-to-multiagentcoordination

[22] José M. N. Vieira, Sérgio I. Lopes, Carlos C. Bastos e Pedro N. Fonseca. Sistema de Localização Mútua para Robots Utilizando Ultrassons. Universidade de Aveiro, 2005.

[23] Eurico Farinha Pedrosa. Ambiente de Simulação para agentes em futebol robótico. Tese de Mestrado, F Universidade de Aveiro, 2010.

[24] Eddie Robot Platform da Parallax. Disponivel em http://www.parallax.com/product/28992. Acesso em 16/09/2013.

[25] MD25 Technical Specifications. Visitado a 16/09/2013. URL: http://www.robot-electronics.co.uk/htm/md25tech.htm

[26] EMG30. Visitado a 16/09/2013. URL: http://www.robot-electronics.co.uk/htm/emg30.htm

[27] MD25 Serial Documentation. Visitado a 16/09/2013. URL: http://www.robot-electronics.co.uk/htm/md25ser.htm

[28] RS. Bateria. Visitado a 30/10/2013. URL: http://uk.rs-online.com/web/p/lead-acid-rechargeable-batteries/5375472/

[29] Conversor UM232R. Visitado a 17/09/2013. URL: http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_UM232R.pdf

[30] 5dpo. Component of Lazarus. Visitado a 17/09/2013. URL: http://wiki.freepascal.org/5dpo

[31] JoyToKey (Ver3.7.4). Emulador de teclados para *gamepads*. Visitado a 17/09/2013. URL: http://www.electracode.com/4/joy2key/JoyToKey%20English%20Version.htm

[32] Wikipedia. Kinect. Visitado a 17/09/2013. URL: http://pt.wikipedia.org/wiki/Kinect

[33] Prime sense reference Design (Kinect IR Laser). Visitado a 17/09/2013. URL: http://www.memonic.com/user/silvan/folder/interactive/id/1pfeg

[34] Prime sense http://www.primesense.com/solutions/technology/

[35] Andrew Davison, "Kinect Open Source Programming Secrets: Hacking the Kinect with OpenNI, NITE, and Java", 2012.

[36] Wikipedia. AGV. Visitado a 17/09/2013. URL: http://fr.wikipedia.org/wiki/V%C3%A9hicule_%C3%A0_guidage_automatique

[37] Pedro André Henriques Polónio. Cambada@home. Tese de Mestrado. Universidade do Aveiro, 2010.

[38] M. Pakdaman and M. Sanaatiyan, 'Design and implementation of line follower robot', *IEEE Second International Conference on Computer and Electrical Engineering*, vol 2, pp. 585--590, 2009.

[39] K. Hasan, A. Al-Nahid and A. Al Mamun, 'Implementation of autonomous line follower robot', *IEEE Informatics, Electronics & Vision (ICIEV), 2012 International Conference*, pp. 865--869, 2012.

[40] F. Chenavier and J. Crowley, 'Position estimation for a mobile robot using vision and odometry', *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference*, pp. 2588--2593, 1992.

[41] O. Montiel, A. González and R. Sepúlveda, 'Vision Based Obstacle Detection Module for a Wheeled Mobile Robot', *IEEE Mobile Robots Navigation*, 2010.

[42] W. Boyu, G. Junyao, K. Li, Y. Fan, G. Xueshan and B. Gao, 'Indoor mobile robot obstacle detection based on linear structured light vision system', *Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics*, pp. 834--839, 2009.