Leonardo de Freitas Gomez Murta

Licenciado em Ciências de Engenharia Eletrotécnica e de Computadores

Sistema para Identificação e Acoplamento de um AGV a uma Plataforma de Transporte Móvel

Dissertação para obtenção do Grau de Mestre em

Engenharia Electrotécnica de Computadores

Orientador:	Doutor Filipe de Carvalho Moutinho, Professor Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa
Co-orientador:	Eng. André Filipe Gomes da Silva, Núcleo de Investigação, Desenvolvimento e Inovação, Introsys, SA

Júri

Presidente:	Prof. Doutor Nuno Filipe Silva Veríssimo Paulino
Arguente:	Prof. Doutor João Almeida das Rosas
Vogal:	Prof. Doutor Filipe de Carvalho Moutinho

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Novembro, 2020

Sistema para Identificação e Acoplamento de um AGV a uma Plataforma de Transporte Móvel

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Agradecimentos

A realização desta dissertação de mestrado, para conclusão de mais uma grande etapa da minha vida, se não, a maior até à data, teve altos e baixos, contudo fazendo um apanhado geral, foi mais uma grande experiência e muito levo daqui para a frente comigo.

Importante realçar que a realização desta dissertação contou com vários apoios e muitos incentivos, sem os quais teria sido muito mais difícil chegar a esta fase final ou até mesmo impossível.

Quero começar por agradecer ao Professor Doutor Filipe Moutinho, DEE- FCT-UNL, pela orientação e disponibilidades contínuas, pelo apoio, sentido prático e paciência, com que sempre acompanhou todo este processo.

Ao Co-Orientador André Silva, Introsys IDI, gostaria de manifestar o meu reconhecimento pelo seu rigor no acompanhamento dos desafios que enfrentei durante a realização desta dissertação, assim como o meu agradecimento pelas construtivas opiniões e críticas, manifestadas ao longo de todo o processo.

Não posso de forma alguma, esquecer que nada disto seria possível se não fosse a disponibilidade da Cristina Salgado, departamento de recursos humanos, Introsys, que facilitou a comunicação entre mim e a empresa, e durante fases menos boas me motivou para a conclusão deste trabalho.

À minha família, quero expressar a minha gratidão por todo o suporte durante este meu processo de crescimento e maturidade, pelo apoio neste momento menos fácil que foi a realização da dissertação, principalmente à minha Mãe que sempre me incentivou e motivou nas alturas mais difíceis e esteve sempre presente para o que fosse preciso.

Por último queria agradecer a todos os meus amigos que direta ou indiretamente, contribuíram para que conseguisse alcançar os meus objetivos académicos e outros!

Resumo

A atualidade tem revelado que o transporte de cargas na Intralogística é uma tarefa em permanente inovação. A introdução de AGV (*Automated Guided Vehicles*) no processo de transporte, veio revelar melhoria de resultados na eficiência da indústria e distribuição, libertando o homem para processos mais diferenciados na produção e distribuição.

Tendo-se verificado que já existem soluções para o transporte de cargas na Intralogística, constatou-se que as mesmas não são totalmente autónomas. Como tal, surgiu a necessidade de implementar um sistema de navegação e identificação, que permitisse fazer o acoplamento de um AGV a uma plataforma de transporte, sem que fosse necessário o recurso a mecanismos orientadores.

No âmbito de uma parceria estabelecida entre a Introsys S.A e a FCT – UNL, propõe-se nesta dissertação um sistema de identificação e navegação de uma *rack*, a qual deverá possuir 3 *tags* que a identificam.

O sistema proposto foi simulado para se poder validar e determinar critérios como luminosidade do ambiente, dimensão das *tags* e a que distância o sistema é capaz de fazer um correto acoplamento. Os resultados obtidos demonstraram que o sistema teve uma taxa de sucesso de 100%, em 40 posições diferentes, para uma luminosidade de 8 focos de luz e uma dimensão de *tag* de 10x10cm, até uma distância de 4,5 metros.

Concluiu-se que o sistema proposto é um sistema capaz de fazer um correto acoplamento a uma plataforma de transporte, desde que nesta estejam instaladas as *tags* propostas e sejam conhecidas as dimensões da plataforma de transporte.

Palavras-chave: AGV / Visão por computador / Pose de objetos / Identificação de objetos / Reconstrução 2D e 3D / *ARtag* / Simulação virtual

Abstract

Present times have revealed that the transport of loads in Intralogistics have been a task in permanent innovation. The introduction of AGV (Automated Guided Vehicles) in the transport process has shown an improvement in the efficiency of industry and distribution, giving more freedom to workers to perform more differentiated processes in production and distribution.

Having verified that there are already solutions for cargo transportation in Intralogistics, it was found that they are not completely autonomous. Therefore, the need to implement a navigation and identification system arose, which would allow the coupling of an AGV to a transport platform without requiring the use of guiding mechanisms. Within the scope of a partnership established between Introsys S.A and FCT - UNL, this dissertation proposes a rack identification and navigation system. The rack must have 3 tags that identify it.

The proposed system was simulated in order to validate and obtain criteria such as, ambient brightness, tag sizes and at which distances the system is capable of making a correct coupling. The results obtained, showed that the system had a 100% success rate in 40 different positions, for a luminosity of 8 spotlights and a 10x10cm tag dimension, up to a distance of 4.5 meters.

It was concluded that the proposed system is capable of making a correct coupling of a transport platform as long as the proposed tags are installed and the dimensions of the transport platform are known.

Keywords: AGV / Computer Vision / Object Pose / Object identification / 2D e 3D reconstruct / *ARtag* / Virtual simulation

Índice

Lista de Figuras

Lista de Tabelas

Siglas

2D Duas Dimensões

3D Três Dimensões

AGV *Automated Guided Vehicles*

AR *Augmented Reality*

CNN *Convolutional Neural Networks*

CPU *Central Processing Unit*

CV *Computer Vision*

LRF *Laser Range Finder*

MSCM *Mobile Camera Space Manipulation*

OEM *Original Equipment Manufacturing*

PCL *Point Cloud Library*

RLPF *Range and Look Palete Finder*

ROS *Robot Operating System*

SDK *Software Development Kit*

TCP *Transmission Control Protocol*

TOF *Time of Flight*

URDF *Unified Robot Description Format*

Capítulo 1

Introdução

Neste capítulo é feito um breve enquadramento a este trabalho, apresentam-se as motivações e objectivos - desenvolver um sistema de identificação e acoplamento de Automated Guided Vehicles (AGV) a uma plataforma de transporte - e por fim apresenta-se a estrutura do documento.

1.1 Enquadramento e motivação

Tem-se vindo a verificar um contínuo desenvolvimento na área da intra-logística, o que por inerência reflete uma necessidade de otimizar a automatização dos sistemas atuais existentes. Baseados na preocupação constante, com a redução de custos de produção a longo prazo e com a diminuição dos erros associados ao manuseamento pelo ser humano, a utilização de *Automated Guided Vehicles* - AGV manifesta-se condição essencial no transporte de cargas. Sendo uma componente sujeita a constante evolução tecnológica, associada à sempre ascendente tecnologia de automação, os AGV fazem hoje parte de quase todos os ramos da indústria e áreas de produção.

Na atualidade, os *Automated Guided Vehicles*, ora adiante designados como AGV, têm assumido um papel preponderante, como peças chave, tanto na definição de novos projetos, assim como, na implementação de novas unidades industriais ou de armazenamento, atendendo a que passaram a desempenhar um papel de extrema importância nas estratégias de produção e movimento de peças e ou materiais. Ou seja, quando devidamente projetados na sua instalação e operação os AGV, procedem ao transporte de cargas de um ponto A para um ponto B, como

alternativa à solução clássica de uso de empilhadoras com motoristas. [1]

De acordo com a nomenclatura atual, um *Automated Guided Vehicle* (AGV) é um robô móvel que se movimenta com base em diferentes tipos de sensores nele incorporados, segue linhas, fios marcados no chão, usa ondas radio, câmaras de visão, íman ou lasers. É um sistema flexível e inteligente capaz de manusear cargas num ambiente logístico de forma eficiente e segura [1]. No entanto, os AGV que se encontram no mercado não correspondem a um maior nível de eficiência e produtividade, já que dependem de uma série de fatores que os condicionam e guiam na sua ação. Pretende-se com isto fundamentar a necessidade de criar um sistema com um perfil mais autónomo e independente, ainda mais flexível e eficaz na sua operação, isto é, que não dependa de tarefas predefinidas como seguir linhas. Pretende-se que o AGV consiga fazer o empilhamento mesmo que as plataformas de transporte não se encontrem numa pose prevista, desde que se encontrem numa área de empilhamento com um raio predefinido.

Considerando esta premência, a Introsys S.A. [2], empresa protocolar neste projeto, com a FCT – UNL, tendo previamente desenvolvido um AGV, com o intuito de colmatar esta necessidade, disponibilizou o mesmo, para que fosse criado a partir deste um sistema de identificação e acoplamento que permitisse, de forma autónoma, proceder ao transporte de plataformas de transporte, designadas ao longo deste documento por *racks*, necessidade que fundamenta o desenvolvimento desta dissertação.

1.2 Objetivos

O objectivo principal deste trabalho é desenvolver uma solução que visa posicionar o AGV de forma autónoma e segura na posição de empilhamento de carga da plataforma de transporte (*rack*). Para tal, torna-se necessário identificar a posição do AGV em relação ao ambiente envolvente assim como, em relação à plataforma de transporte. No seguimento, o sistema de navegação deve ser capaz de movimentar e posicionar o robô, de forma segura, na posição empilhamento, tendo em conta todos os sistemas de segurança implementados. Sendo que o AGV tem uma componente de deteção de obstáculos, é importante não confundir a plataforma de transporte (e.g palete/*rack*) como um obstáculo. Para tal, primeiro deve ser identificada a plataforma de transporte e só depois entrar no modo de aproximação onde os módulos de segurança, de colisão e deteção de obstáculos do AGV, são desativados, sendo que este passa a executar movimentos mais lentos e precisos para não colidir com a plataforma de transporte. Na etapa navegação para o acoplamento é necessário identificar o local da *rack* por onde irá entrar. Serão usados os dados extraídos pela câmara que se pretende equipar que depois de serem processados e interpretados, permitirá à parte de controlo ter a informação necessária para que possa controlar os motores do AGV para proceder ao acoplamento da *rack*.

Os objectivos identificados podem-se sintetizar nas seguintes perguntas de investigação:

- Como identificar a plataforma de transporte (*rack*), bem como a sua pose?
- Como processar os dados referentes à pose da *rack* e fazer o seu acoplamento de uma forma segura e sem colisões?

1.3 Estrutura do documento

O documento apresentado está dividido em 5 capítulos:

- **Capítulo 1.** correspondente ao corrente capítulo, é introduzido o AGV enquadramento e motivação do tema desta dissertação bem como os objetivos;
- **Capítulo 2.** apresenta-se uma breve historia dos AGV, bem como os trabalhos e tecnologias utilizadas por outros autores, sensores utilizados por outros autores e ferramentas de *software* a utilizar na implementação;
- **Capitulo 3.** são descritos os componentes de hardware necessários para melhor compreensão do sistema proposto e apresentação do sistema proposto.
- **Capítulo 4.** são apresentados os testes realizados, bem como a análise detalhada aos resultados obtidos nas simulações;
- **Capítulo 5.** apresenta as conclusões ao trabalho desenvolvido no decorrer desta dissertação e trabalhos futuros.

Capítulo 2

Estado de Arte

Nas próximas secções são apresentadas algumas das soluções que outros autores propuseram para problemas relacionados com esta temática, nomeadamente identificação de objetos (paletes) e orientação/localização dos mesmos.

2.1 AGV

A área da engenharia robótica e veículos autónomos (AGV) tem-se revelado uma realidade em grande desenvolvimento. Nesta e noutras áreas, a substituição de tarefas que requerem o esforço do ser humano estão amplamente a ser substituídas por robôs.

Cronologicamente, a historia dos AGV divide-se em quatro principais eras. Podemos dizer que a primeira era teve inicio em 1950, nos Estados Unidos da América (USA), altura em que foi criado e introduzido no mercado o primeiro AGV pela, *Barrett Electronics de Northbrook*, Illinois. Nessa altura, o AGV era simplesmente um veículo de reboque que seguia uma linha no chão. Ainda na primeira era, meados de 1960, países Europeus como a Alemanha começaram também a desenvolver AGV com função de empilhar cargas. Com a evolução e reconhecimento das vantagens na utilização dos AGV, transporte de matérias-primas e produtos, rapidamente surgiu a segunda era (1970 a 1990), promovendo-se robôs cada vez mais capacitados nas suas funções e respondendo de forma mais eficiente às necessidades da indústria ou logística [3].

Com a crescente evolução da indústria automóvel e de manufactura, verificou-se a necessidade de flexibilizar e aumentar a capacidade de automação em meio fabril ou no armazenamento. Neste sentido, os AGV passaram a ser parte integrante nas soluções intra-logísticas, aumentando a produtividade e permitindo por consequência uma diminuição nos tempos de entrega. No entanto, com a recessão em 1980, a utilização dos AGV acabou por se verificar onerosa, já que não contribuíam ainda com a flexibilidade prevista, atendendo a que estavam ainda pouco desenvolvidos para as necessidades do mercado. Os principais OEM da industria automóvel, como a *Volkswagen*, *BMW* estavam de acordo, que alguma coisa tinha que ser feita para melhorar os AGV, tornando-os um sistema mais eficiente e sustentável na intra-logística. Em 1987, a (*VDI The Association of German Engineers*) [4], traz para o mercado da indústria e da logística tópicos relevantes nos AGV como *hardware*, *software* e normalização dos sistemas industriais. No final dos anos 90, novos avanços tecnológicos, permitem uma evolução ao nível do *hardware* e *software*, o que faz com que seja possível que se desenvolvam sistemas eletrónicos cada vez mais complexos, capazes de responder a um número cada vez maior de requisitos e exigências do mercado, passando-se cronologicamente para a terceira era (1990 a 2010). Nesta altura, os AGV começaram a provar que eram uma mais valia para a intra-logística, e com o surgimento de novos tipos de sensores e o recurso a uma visão por computador processada pelos novos processadores incorporados nos controladores, o avanço tecnológico torna-se evidente, desenvolvendo-se este tipo de tecnologias a uma grande escala [3].

De uma função inicial de ação guiada, os AGV num contínuo desenvolvimento tecnológico ao longo dos tempos, passam a ser um componente essencial para a intra-logística, sendo que a constante evolução a nível tecnológico, bem como, a experiência com a tecnologia de automação, vieram permitir introduzir os AGV em quase todos os ramos da indústria e áreas de produção, chegando à atualidade. Hoje, e de acordo com o previamente exposto, o AGV é um robô autónomo, que pode funcionar permanentemente, sem intervenção direta humana, através de um sistema de *software* específico. Permite fazer o transporte de cargas a nível do armazenamento, estar em linhas de produção e de distribuição.

Contudo, os AGV existentes no mercado dependem de mecanismos de orientação para movimento no espaço, que limitam o AGV na sua deslocação, já que estando dependente dos elementos orientadores (ondas de rádio, câmaras de visão, lasers, íman e até mesmo fios ou linhas marcadas no chão), quando estes por algum motivo são comprometidos, o AGV fica imobilizado não cumprindo com a sua função. A função de leitura do espaço a percorrer é de todo essencial à eficiência e capacidade de flexibilidade do sistema do AGV, aumentando por inerência o

sucesso das tarefas do mesmo, identificação da carga e transporte desta.

Perante esta necessidade, a Introsys S.A., desenvolveu um AGV que permitisse ultrapassar esta limitação, necessitando para tal que fosse criado um sistema que fizesse a identificação da pose da *rack*, (convencional *pallet*) sendo este o motivo de desenvolvimento desta dissertação.

Situando o Estado da Arte dos AGV, a tecnologia de visão por computador tornou-se sofisticada e é mais especificamente em 1995, altura em que foi criado o primeiro prototipo testado, do *robolift* [5], através de um parceria entre *Elsag Bailey Telerobot e Fiat OM Carrelli Elevatori SpA*, que torna mais visível o potencial dos AGV. Posteriormente em 1996, uma versão do mesmo (*robolift*) foi apresentado na *Hannover Fair*, *CeMat*, trazendo uma nova dinâmica a esta área. O foco base deste projeto, foi criar um sistema de Visão Computacional para um AGV de nome *robolift* com capacidade sensitiva e de se localizar num armazém industrial. A tarefa do *robolift* tinha como objetivo, fazer o transporte e empilhamento de uma palete de forma autónoma. Antes dessa data, nomeadamente em 1993, com recurso a sensores lasers, o *robolift*, já era capaz de fazer uma navegação autónoma [6]. Ainda em 1995, de modo a fazer-se o transporte de materiais em escritórios como a correspondência ou documentos, foi implementado o *telerobot* [7].

Para uma navegação segura e eficiente, um AGV pode possuir recursos tais como, linhas magnéticas, marcadores, visão ou lasers, que visam simplificar a leitura dos percursos pretendidos e localizar-se no ambiente que se encontram [1]. Técnicas baseadas em sistema de reconstrução de 3D/2D, das características do mundo envolvente e de processamento de imagem, extraídas através de vários tipos de sensores, são então fundamentais para o desenvolvimento desta dissertação.

Sendo um AGV, um robô que visa automatizar o processo de transporte de materiais ou outros objetos em fábricas ou indústria, existem vantagens e desvantagens neste processo. O autor Pathan Alamkhan G. descreve algumas destas vantagens em [8]:

- Maior produtividade no movimento robô a uma velocidade segura;
- Capacidade de prever acções e mudar de rotas de acordo com elementos que se oponham ao movimento do mesmo, flexibilidade e adaptabilidade ao ambiente da intralogística;
- Redução da força de trabalho humana;
- Evitar colisões que muitas das vezes não são previstas pelo ser humano;

- Aumento da eficácia e produtividade no transporte;
- Possibilidade de controlo instantâneo de cada mercadoria;

Referindo ainda algumas das desvantagens:

- Dificuldades em adaptarem-se a um ambiente ao ar livre;
- Manutenção continua do AGV;
- Programados para trabalhar somente em ambientes de intra-logística;
- Incompatibilidade em gerir a interação de AGV de marcas diferentes;
- No caso de fábricas com luminosidade reduzida ou por obstrução o AGV pode não reconhecer os *targets*, que muitas das vezes não são possíveis contornar sem a intervenção do ser humano.

Estes robôs, podem ser inseridos em armazéns ou ambientes industriais que movimentam grande número de cargas, sendo frequentemente utilizados em ambientes de logística ou de aplicação logística, no qual o transporte de matérias e cargas se verifique mais dificultado e menos eficiente.

Figura 2.1: Exemplo de um AGV em fábricas do grupo BMW (retirado de [9])

2.2 Identificação da plataforma de transporte

A Identificação de uma plataforma de transporte é essencial para que um AGV não processe dimensões e poses de objetos que não correspondem às necessidades. Sendo que o AGV tem uma componente de deteção de obstáculos, é importante não confundir a plataforma de transporte (e.g palete/*rack*) como um obstáculo. Para tal, primeiro deve ser identificada a plataforma de transporte.

Técnicas baseadas em deteção de objetos e processamento de imagem são pontos de partida para este primeiro objetivo, com o intuito de encontrar uma região de interesse para se fazer o acoplamento.

2.2.1 Técnicas de identificação com recurso a pontos de referência

Em [5], é proposta uma solução autónoma para substituir o sistema tradicional de empilhadores *robolift*. O reconhecimento é feito a partir de um sistema de Visão 3D, cruzando os dados extraídos pela câmara com os dados de sensores odométricos que o *robolift* possui. A eficácia do sistema de visão está fortemente relacionada com a boa calibração da câmara. Esta calibração é feita com recurso ao *pinhole camera model* [10]. A solução para identificação da plataforma de transporte, neste caso, a palete, é baseada em padrões circulares colocados na palete para um reconhecimento facilitado da mesma. Com uma boa calibração, é possível retirar dimensões e fazer um sistema de coordenadas 3D da palete.

Os autores Seelinger e Yoder em [11] apresentam uma proposta de resolução para fazer o empilhamento automático de uma palete, através de um sistema de visão que recorre a duas câmaras acopladas a um robô. A calibração do sistema de visão não foi um dos pontos cruciais neste projecto. Contudo, a identificação da palete através deste método com recurso às câmaras provou não ser tão eficaz na deteção de uma qualquer palete. Neste caso, os autores utilizaram marcas previamente colocadas na palete, para uma melhor precisão da identificação, ou seja, em vez de localizarem a palete pelas as suas características naturais, optaram por identificar as marcas que estão montadas na palete que sabiam corresponder à palete. Utilizaram um método chamado *mobile camera space manipulation* (MCSM) que, como já referido, recorre à utilização de duas câmaras, mas tem a vantagem de não necessitar de uma calibração precisa, o que muitas das vezes envolve trabalho minucioso e cuidado.

2.2.2 Técnicas de identificação com recurso às características dos objetos

O recurso ao uso de uma só câmara calibrada, tem vantagens a nível de custo e a nível de logística na montagem das mesmas, como descrito em [12], em que o autor apresenta uma sistema de visão para um empilhador automático. A câmara calibrada está montada no topo do veiculo autónomo sendo a calibração do sistema feita a partir do *coplanar Tsai calibration* [13]. Depois de uma boa calibração do sistema, processa-se a imagem extraída pela câmara. Este processo de identificação da palete é então efectuado através de segmentação de cor [14]. O sistema de visão foi capaz de identificar a palete a partir das suas próprias características naturais e não com recurso a marcas/padrões previamente instalados na palete. Resumindo o processo de segmentação, a imagem extraída pela câmara é então transformada numa imagem binária/digital. As caraterísticas da palete como a cor da mesma são então cruciais para esta reconstrução de imagem, bem como as condições do ambiente de trabalho em que o veículo está a operar, neste caso armazéns industriais, foram bastante importantes. Aspectos como a luminosidade do ambiente em redor são cruciais, pois as cores da imagem extraída pela câmara podem muitas das vezes ser alteradas com tipo de luminosidade, e induzir em erro à reconstrução da imagem para a imagem digital.

Segundo os autores de [15], o objectivo de detectar e localizar uma palete industrial era uma tarefa exigente, especialmente no caso em que a posição/área em que a palete se encontra não era conhecida a priori. Propuseram então uma solução com recurso a segmentação de cor usando uma câmara *Sonny EVI-D100P*. Para fazerem a segmentação da imagem, basearam-se no método de extracção de cores dominantes [16]. A ideia principal foi trabalhar com as características da palete e não com recurso a padrões e marcas na palete. Para tal, utilizaram um filtro chamado *morphological filtering* [17] para retirar o ruído da imagem e um filtro *Sobel* para encontrar os cantos da palete. A imagem original (figura 2.2) é transformada numa imagem digital (figura 2.3) que só apresenta as áreas da palete, sendo formada a partir das características da mesma. De seguida através do filtro *Sobel*, são delimitados e reconhecidos os cantos da palete e finalmente é criada uma nova imagem binária (figura 2.4).

Em [16], os autores propõem um sistema de identificação de uma palete, capaz de lidar com diversos tipos de oclusão, como situações de luminosidade reduzida. O sistema proposto é baseado num modelo da palete *standard*, ou seja, uma imagem do modelo da palete considerada, com graus de liberdade, tendo como propósito ser comparada com a imagem que está a ser transmitida pela câmara

Figura 2.2: Imagem Original [15]

Figura 2.3: *Morphological Filtering* [15]

Figura 2.4: Filtro *Sobel* [15]

em tempo real. Nessa imagem modelo, definem os cantos da palete comparando essa imagem com a real. Através diferentes operadores de análise de imagem, é possível verificar se a imagem da palete que está a ser transmitida em tempo real, corresponde às características predefinidas. Um estágio final classifica a imagem como palete ou não, com base num classificador *decision tree* [18]. A classificação é fornecida avaliando a presença e os atributos de algumas características e as relações entre elas.

Os resultados experimentais provaram ser funcionais a detetar a palete com sucesso. Contudo, para uma maior eficácia do sistema, a adição de características da palete a ter em conta na análise e classificação da imagem, são requisitos que poderiam melhorar a probabilidade de encontrar a palete.

2.2.3 Técnicas de identificação com recurso a câmaras *stereo*

Em [19] o autor propõe uma solução para o reconhecimento da palete com recurso a câmaras estéreo e à intensidade da imagem. A intensidade da imagem fornece informações como a localização 2D da palete, enquanto que as câmaras estéreo são usadas para obter a posição 3D e orientação da palete em relação às câmaras, que por sua vez, como as câmaras estão montadas no AGV, pode-se ter uma localização do AGV em relação à palete. Deste trabalho foi possível concluir que a identificação da palete baseada em 3D é menos confiável devido a baixa qualidade de reconstrução dos orifícios da mesma. Contudo, os requisitos relativos à localização exacta, são rigorosos, logo as paletes necessitam de ser localizadas com maior precisão. De seguida fazem uma reconstrução estéreo, fornecendo assim detalhes para possível candidato a palete. A detecção de possíveis candidatos é então analisada e classificada. A imagem seleccionada para efeito prático e real tem que obedecer a certos requisitos como: a imagem gerada pelo reconstrução estéreo não deve perder qualquer dimensão da palete; avaliar a imagem rapidamente; rejeitar apenas um certo número de candidatos para ajudar a validar etapas de

processamento adicionais, ou seja, o processo de possível candidato a palete, deve produzir uma lista de candidatos que apresentam rectângulos, pois, como visto antes, os orifícios da palete são áreas rectangulares. Para isto, são feitos alguns cálculos, descritos em [19], para detectar importantes linhas horizontais e verticais. Um histograma que acumula valores pretendidos ao longo de cada linha, é usado para encontrar máximos locais, ou seja, é inspeccionada a imagem que é lida na horizontal e vertical, pixel a pixel, para encontrar dimensões e possíveis arestas do rectângulo.

Para uma rápida classificação de imagem, *perfomance* e eficiência do mesmo, *bosted classifiers- machine learning* [20] com *soft cascading* [20] e *Histogram Intersection SVM* [21], são essenciais. Com um intuito de delimitar uma região para reconhecimento dos orifícios, a proposta em [19] visa encontrar a estrutura da palete de forma breve e garantir uma representação que é menos variável às mudanças de iluminação. Em suma, depois de alguns testes e resultados experimentais, a técnica provou ser eficaz.

2.2.4 Técnicas de identificação com recurso a *Machine Learning*

A detecção e localização de uma palete também pode ser efetuada com recurso a técnicas de *Machine Learning*. Em [22] é proposta a extracção de dados a partir de laser 2D que medem distâncias criando um sistema de coordenadas polares, obtendo uma imagem binária. A deteção da palete é feita através de uma rede neuronal que, por sua vez, depois de treinada, é capaz de reconhecer uma ou mais paletes. Está dividida em dois processos, a *Faster R-CNN* que deteta a ROI (região de interesse) na imagem, e um *CNN-based classifier* que tem como parâmetro de entrada a ROI, para classificar se é o objeto que se pretende avaliar/classificar, neste caso a palete. Podemos verificar na imagem (Figura 2.5), em que os autores de [22] apresentam o exemplo de funcionamento de uma rede neuronal treinada.

As Redes Neuronais Convolucionais, no seu termo em inglês *Convolutional Neural Networks* (CNN), são frequentemente utilizadas para aplicações de processamento de imagem [23]. Na sua definição, um algoritmo treinado, que analisa uma imagem, é capaz de atribuir importâncias a várias características da imagem, definidas por quem programou a rede neuronal. O pré-processamento exigido por uma CNN é muito menor em comparação com os outros algoritmos de classificação. Enquanto nos métodos tradicionais são feitos manualmente, as CNN treinadas são capazes de o fazer de forma autónoma e por cada classificação vão sendo cada vez mais eficazes, ou seja, quanto mais são utilizadas e treinadas, maior a probabilidade de eficácia deste sistema ter sucesso [23]. Os autores de

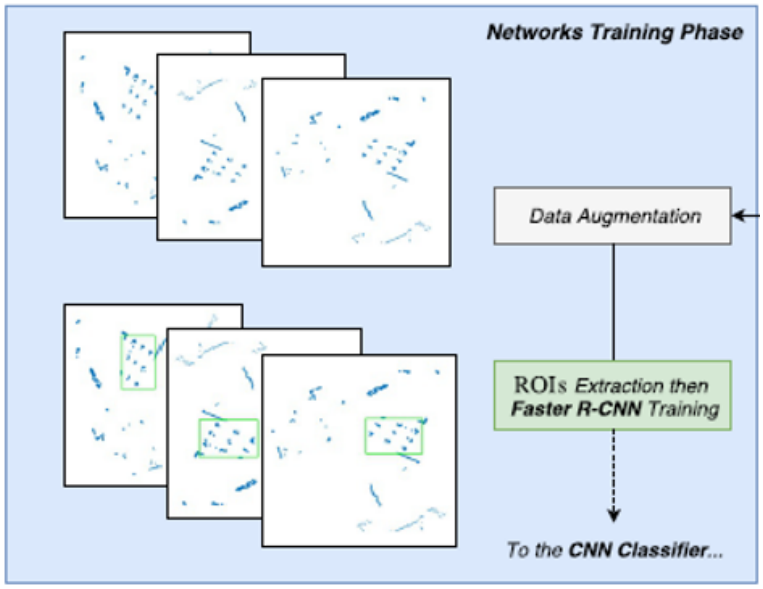

Figura 2.5: Exemplo do funcionamento de uma rede neuronal treinada (retirado de [22])

[22] concluíram que o uso de redes neuronais pode ser uma solução viável, ou seja, foi possível detectar e localizar a palete.

O recurso a redes neuronais está cada vez mais a ser utilizado em sistemas robóticos e veículos autónomos, como no caso de [24] que recorre a redes neuronais para analisar dados retirados de um 3D-LIDAR implementado num sistema de condução autónomo. O sensor 3D-LIDAR cria uma nuvem de pontos do ambiente em redor e a CNN treinada classifica essa nuvem de pontos.

2.3 Identificação da pose/orientação de uma plataforma de transporte

Como segundo objetivo desta dissertação, para desempenhar e concretizar a tarefa de colocação do AGV, em segurança por baixo da *rack* é necessário localizar e ter uma orientação da pose da *rack* em função do AGV, ou seja, confinado a uma área de trabalho reduzida e com recurso a movimentos pormenorizados e sensíveis, as medidas métricas de distância e localização do AGV em relação aos objetos, são essenciais para um seguro acoplamento do AGV ao sistema de transporte. É importante ter um especial cuidado no cálculo da pose do objeto, pois uma posição incorreta da plataforma de transporte, poderia causar acidentes no acoplamento do mesmo. Consequentemente, poderia causar danos e estragar peças que nele se encontrem. Para tal, é necessário que os dados recolhidos pelos sensores sejam precisos.

2.3.1 Técnicas de identificação de pose com recurso a reconstrução 2D/3D (*Stereo Vision e TOF*)

Uma boa reconstrução 2D / 3D, dependendo das necessidades e das condições onde irá trabalhar o AGV, são um ponto essencial para a optimização. Obter uma boa perspectiva do mundo em redor e principalmente posições certas de onde se encontra a plataforma de transporte, como num sistema de coordenadas 2D ou 3D, são essenciais para a sua eficiência e eficácia.

O empilhador *robolift*, na segunda fase do projeto [5] com recurso a um sistema de visão computacional, implementou um algoritmo de visão utilizando a câmara montada no empilhador. Este algoritmo computacional de visão foi capaz de detetar as dimensões da palete. Consistia basicamente em obter uma posição 3D da palete, projetando uma imagem geométrica do modelo esperado da palete, ou seja, era feita uma estimativa da posição da palete, num mundo 3D. Esta técnica estava pouco desenvolvida e consolidada e com o evoluir dos anos, novas técnicas de visão por computador foram desenvolvidas.

Os autores, Varga R. e Nedevschi S. em [25], para além do método de deteção da plataforma de transporte (neste caso uma palete), também propuseram um método para estimativa da posição da plataforma de transporte. Uma reconstrução estéreo, permitiu-lhes então fazer uma estimativa da palete partindo do princípio que o AGV se encontra a cerca de 2.5 metros da plataforma de transporte. Com o objetivo de fazer o empilhamento da palete, o método de reconstrução estéreo,

permitiu fazer uma reconstrução 3D, em que nesta construção 3D da palete em relação ao AGV eram obtidas distâncias e ângulos entre eles. A eficiência do método de reconstrução *stereo* está associado a uma boa calibração das câmaras.

Em [26] os autores propõem uma solução ao problema que foi a identificação e localização de uma palete, com recurso a sensores laser e uma câmara. Introduziram assim um novo algoritmo chamado RLPF (*range and look palete finder*). Os autores deste, basearam-se noutro método [27] que consistia em identificar uma determinada palete. O robô em questão sabia previamente que numa certa área poderia encontrar uma palete, mas sem localização certa e precisa da pose da palete [26]. O robô tinha então de ser capaz de determinar ângulos e medidas de distância entre o robô e palete com um erro menor que 1%. O processo de reconhecimento da palete, é então feito com recurso a um laser e uma câmara. Enquanto o LRF (*laser rande finder*) extraia informações para se criar um sistema 2D da palete, a câmara providenciava informações acerca de uma possível reconstrução 3D da palete. Através do sensor laser (LRF) encontraram os valores em coordenadas cartesianas 2D que correspondiam à parte frontal da palete. Com recurso uma técnica de nome pixelização, transformaram esses valores numa imagem binária, criando assim uma imagem que representa uma região de interesse. Depois de ter uma imagem binária da parte frontal da palete, essa imagem foi filtrada pelo método de *Hought Transform* [28] utilizado para detetar linhas geométricas com alguns ponto definidos. Fazendo uma comparação da imagem binária com imagens *standard* para classificar a palete, foi possível detetar a palete num sistema de coordenada 2D.

Em [29], primeiramente, foi feita uma identificação da palete através de uma câmara montada no empilhador. Na qual foi proposto um sistema baseado em visão para orientação e posicionamento de uma palete, através de uma técnica de nome projeção de fundo (*back-projection*). Esta técnica proposta pelos autores, consiste em retroprojeção, ou seja, é projetada uma imagem *standard* da palete refletindo assim as extremidades horizontais da parte frontal da palete (superior e inferior) num plano virtual. Assim que estas linhas horizontais ficam paralelas, a orientação das duas imagens (imagem virtual da palete e a real) será a mesma, a orientação da palete é determinada com as dimensões *standard* da palete. Com esta técnica, foi possível criar um sistema de coordenada 3D de pontos de interesse na palete. Foi atribuído um sistema de coordenadas 3D (x,y,z) no empilhador, na palete e no espaço para se poder determinar a que distâncias estão uns dos outros (figura 2.6). Depois de alguns testes, os autores, afirmaram que a técnica utilizada por estes podia ser aplicada em ambientes práticos.

Os autores de [15] depois de fazer a deteção de uma palete com recurso a

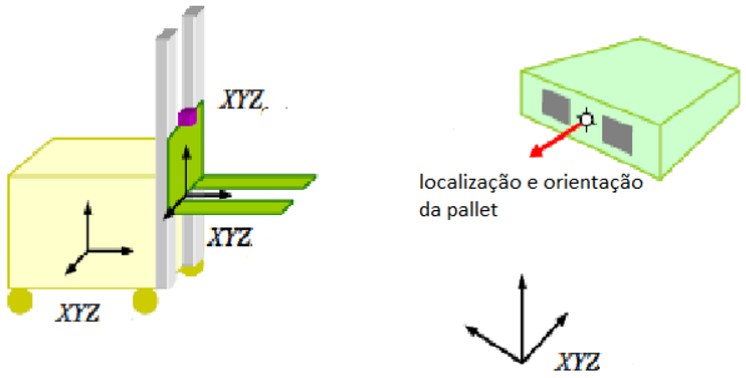

Figura 2.6: Sistema de coordenadas 3D (adaptado de [29])

segmentação por cor, conseguiram localizar os dois cantos da palete orientando-os com um sistema de coordenadas 3D (x,y,z), para extrair dimensões da palete, através do método de calibração de nome *Coplanar Tsai calibration* [30]. Representando os cantos direito (Xwcp1,Ywcp1,Zwcp1) e esquerdo (Xwcp2,Ywcp2,Zwcp2) da palete e a própria palete (Xw,Yw, Zw). Através da equação (1) foi possível localizar o centro da palete (Xwm,Ywm,Zwm).

$$\begin{cases} x_{wm} = \dfrac{x_{wcp1} + x_{wcp2}}{2} \\ y_{wm} = \dfrac{y_{wcp1} + y_{wcp2}}{2} \\ z_{wm} = \dfrac{z_{wcp1} + z_{wcp2}}{2} + \dfrac{h}{2} \end{cases} \quad (1)$$

$$\begin{cases} \overrightarrow{dp_v} = \sqrt{\left(x_{wcp1} - x_{wcp2}\right)^2 + \left(y_{wcp1} - y_{wcp2}\right)^2 + \left(z_{wcp1} - z_{wcp2}\right)^2} \\ \overrightarrow{p_v} = \dfrac{x_{wcp1} - x_{wcp2}}{\overrightarrow{dp_v}}\vec{i} + \dfrac{y_{wcp1} - y_{wcp2}}{\overrightarrow{dp_v}}\vec{j} + \dfrac{z_{wcp1} - z_{wcp2}}{\overrightarrow{dp_v}}\vec{k} \end{cases} \quad (2)$$

onde h representa uma altura *standard* de uma palete. Os vetores de direcção e orientação da palete são descobertos através da equação (2). Os métodos de identificação da palete utilizados tiveram um erro de precisão de 10.6mm no eixo X e 6.5mm no eixo y. Os testes efectuados provaram ser eficazes em 26 de 30 tentativas, o AGV encontrava-se acerca 5 metros de distância da palete.

Os autores de [31] propõem um algoritmo de reconstrução 3D para um sistema autónomo pré-industrializado de empilhadores, capaz de detetar e empilhar paletes onde a localização delas é incerta, com recurso a uma câmara calibrada. Este algoritmo é baseado em dimensões reais dos objetos (neste caso a palete) conhecidas a priori. Fazem então um *matching* dessas dimensões com a imagem a que está a ser transmitida em tempo real pela câmara. O objetivo relevante para a determinação da pose do objeto foram as extremidades da palete, pois sendo estas linhas, linhas retas, foi possível determinar os ângulos que a palete fazia com a

câmara. O recurso a câmara requer uma boa luminosidade e uma boa calibração das câmaras. Foi então criado um sistema de coordenadas 3D (x,y,z) para posicionar o objeto. A uma distância de cerca de 5 metros do objeto, verificou-se um erro nas medidas métricas de cerca de 13.16cm em X, 3.11cm em Y e 17.47cm em Z. Quanto aos erros nos ângulos da pose da palete, verificaram um erro de cerca de 6.35º em X, 3.25º em Y e 0.63º em Z. O algoritmo foi testado também mais próximo da palete, obtendo um erro aceitável quando o sistema se encontrava a cerca de 1.80m da palete.

2.3.2 Técnicas de identificação de pose com recurso a *Structured Light*

O recurso ao método de luz estruturada no seu termo em inglês- *structured light*, também é utilizado para fazer a deteção da pose de paletes. Como é o exemplo de [32] que faz uma reconstrução 3D de objetos que estejam próximos, utilizado para identificar a pose da palete. O sistema do método de *structures light* [33] é composto por dois dispositivos calibrados. Resumidamente, um emissor infravermelhos projeta um padrão de luz no objecto, de seguida uma câmara infravermelhos captura uma imagem 2D com a projecção do emissor e o cenário onde está a ser projectado o padrão, o que permite detetar assim texturas e padrões associados aos objectos através do tempo reflexão do feche de luz (TOF) que está a ser emitido. Este método pode ser melhor compreendido pela figura 2.7.

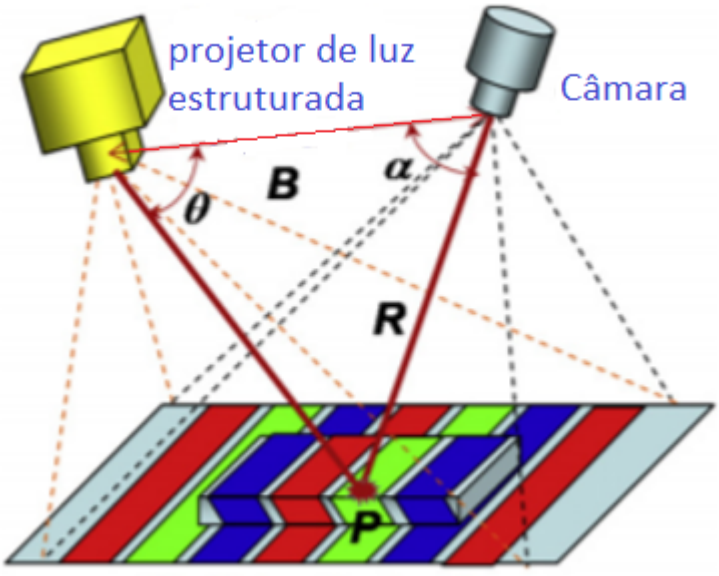

Figura 2.7: Ilustração do funcionamento de *Strutured Light* (adaptado de [33])

O processo de extracção de medidas métricas, através de luz estruturada, pode

ser melhor compreendido em [34], no qual os autores demonstram o processo de funcionamento deste sistema. O cálculo da distância representada na figura 2.7 pela letra R é descrito abaixo:

$$R = \beta \frac{sin(\Theta)}{sin(\alpha) + \Theta}$$

Podendo assim ser possível retirar dados relevantes a pose de um objecto através deste método.

2.4 Sensores utilizados em sistemas de visão

A eficácia de um sistema de visão por computador está relacionada com o desempenho e competência dos sensores disponíveis actualmente. Nos últimos anos, diferentes técnicas foram desenvolvidas para aquisição de dados como distâncias e localizações de objectos ou materiais, com recurso a sensores ópticos. Um estudo sobre tipos de sensores e capacidade de sucesso, é aprofundado em [35], onde são mencionadas, também algumas vantagens e desvantagens de tipos de sensores para reconstruções 2D e 3D. O desenvolvimento de um sistema de visão por computador capaz de reconhecer o mundo em seu redor tem sido um tema em grande desenvolvimento ao longo dos últimos anos.

O princípio de operação com recurso a sensores 3D pode ser dividido em 3 mecanismos, intensidade/ estrutura da luz, tempo de voo (TOF-*time-of-flight*) e visão *stereo*. Uma comparação destes três tipos de mecanismos, é feita em [35].

Os autores de [36] e [37] discutiram algumas vantagens e desvantagens das técnicas e tipos de sensores utilizados para detetar e orientar uma palete (a deteção de cantos da palete com recurso a lasers, que são de baixo custo, e câmaras estéreo que estão limitadas pelas condições de luz).

No caso de um sensor laser, a distância a um objeto é medida emitindo raios laser pulsados e determinando o tempo que a luz reflectida demora a voltar ao receptor (TOF-*time-of-flight*). Um Sensor Laser básico faz uma leitura simplesmente em uma dimensão (1D). O principio de funcionamento de um sensor laser básico pode ser melhor compreendido na figura 2.8.

O uso de lasers é então vantajoso, pois permite extrair dados a partir desse tipo de sensores, sem recurso a iluminação presente no ambiente de leitura, ou seja, é capaz de operar em condições de iluminação reduzida ou até mesmo sem iluminação.

Os sensores Lidar, também muito utilizados para detetar medidas, são um complemento de um sistema laser, podendo fazer leituras em 2D. Assim, pode

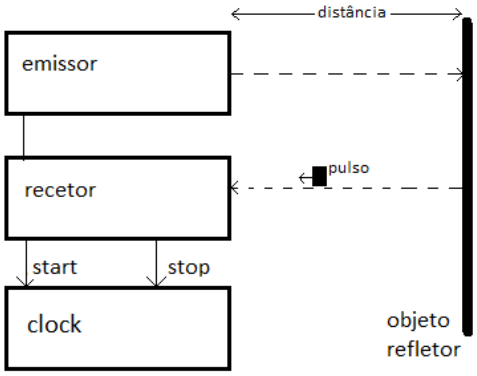

Figura 2.8: Esquema de funcionamento de um sensor laser (adaptado de[38])

então ser criado um sistema de coordenadas 2D ou 3D, permitindo que se localize e oriente um determinado robô num ambiente próximo em seu redor, nomeadamente armazéns industriais. Sendo um complemento do sensor laser básico, tem as suas vantagens, pois permite extrair dados para um possível mapeamento e localização de um robô autónomo.

A reconstrução 3D através de *Stereo Vision* do mundo em redor tem sido amplamente investigada e utilizada nos últimos anos. A técnica consiste então em utilizar duas câmaras ópticas bem calibradas e a uma distância conhecida entre si. Comparando as duas imagens extraídas pelas câmaras, pelo princípio de triangulação, é possível retirar características das imagens como distâncias, podendo assim ser criado um sistema de coordenada para orientar localizar objetos. [33] O funcionamento de um sensor laser básico pode ser melhor compreendido na figura 2.9.

Uma das câmaras utilizadas para determinação 3D de um mundo em redor é a *Kinect camera*. Esta câmara utiliza um emissor de infravermelhos, que projeta uma rede infravermelha em toda a superfície à sua volta e que se deforma formando uma rede com a forma da superfície em redor. No seu interior, existe uma câmara de profundidade que analisa a malha de infravermelhos para construir um mapa em 3D do local e das pessoas que ali se encontram. Possui também um pequeno motor eléctrico de movimento vertical que ajusta a câmara para deteção de pontos de interesse. [39]

Em [39] é feita uma abordagem e análise de sensores *Kinect*. Com uma taxa de erro métrico de 1.5cm, o autor propõe uma técnica baseada na leitura de profundidade da imagem, ou seja, é possível classificar um gesto de uma mão através de

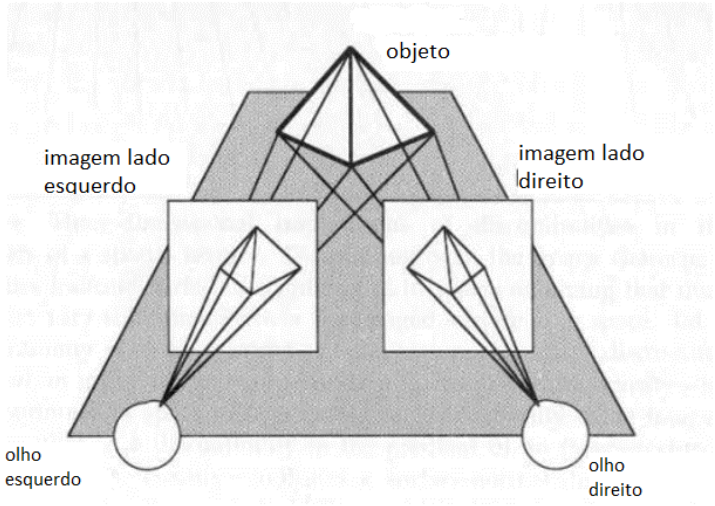

Figura 2.9: Funcionamento de uma câmara *stereo* ([33])

uma câmara *kinect*. A sua eficiência está altamente relacionada com a luminosidade do ambiente, ou seja, é determinante que exista uma boa luminosidade para a eficiência desta técnica.

2.5 Software

No decorrer desta secção, é apresentado o software a ser utilizado no desenvolvimento do sistema proposto nesta dissertação.

2.5.1 Ubuntu

O Ubuntu [40] é um sistema operativo *Linux*. Por ser um sistema operativo em código aberto e gratuito, tem uma maior procura e utilização pelos programadores na área de robótica. Ubuntu é uma das distribuições fornecidas pelo *Linux*. A versão utilizada nesta dissertação foi a 18.04.4 LTS, à data do desenvolvimento de código, foi a considerada mais adequada e mais actualizada para um bom funcionamento de todas as aplicações e programas associados à implementação. O Ubuntu, foi produzido pela Canonical [41] e pela comunidade que o utiliza, sofre actualizações constantemente para que todos os erros e vulnerabilidades encontradas sejam resolvidos e colmatados. A ideia de quem criou este sistema operativo é poder ser de fácil acesso e internacionalização, ou seja, permitir a utilização ao maior número de utilizadores. O Ubuntu oferece a maior parte das aplicações, ferramentas e programas que o sistema operativo mais conhecido (*windows*) pode fornecer. Por norma, os programas do usuário estão disponíveis com poucos acessos e não podem corromper um sistema operativo ou os ficheiros de outros usuários. Para uma maior segurança, a ferramenta *sudo* é usada para atribuir permissões para executar tarefas administrativas, o que permite que a conta administrativa permaneça bloqueada e ajuda a evitar que usuários inexperientes façam alterações indesejáveis no sistema ou fiquem sujeitos a falhas de segurança.

2.5.2 ROS

ROS de tradução estrangeira, *Robot Opertation System*, é um sistema operativo ou uma *framework* bastante utilizada na programação de robôs, é uma ferramenta que providencia diversas bibliotecas e ferramentas para o funcionamento e programação de robôs. O ROS foi desenvolvido pela *Artificial Intelligence Laboratory of Stanford University*. Sendo uma plataforma de código aberto, existe uma grande comunidade, não só de estudantes como programadores, que se entre ajudam, para resolver problemas e dúvidas recorrentes. O ROS está estruturado e organizado através de nós que tem funções específicas e tem como objetivo fazer a comunicação entre pacotes e módulos do software programado. A comunicação entre nós é feita através da troca de mensagens que podem ser assíncronas, denominadas de tópicos, ou síncronas, denominadas de serviços. A comunicação entre

os nós é feita através do protocolo TCP/IP. O ROS pode ser dividido em 3 grupos: Pacotes, bibliotecas de implementação e ferramentas de desenvolvimento [42].

Existem várias distribuições do ROS, para o desenvolvimento do sistema presente nesta dissertação foi usado o *ROS Melodic*, pois no decorrer do desenvolvimento prático da dissertação, foi a versão que se achou mais conveniente e actualizada para o pretendido. Um dos recursos mais fortes do ROS é o conjunto de ferramentas de desenvolvimento. Essas ferramentas oferecem suporte à introspecção, depuração e visualização do estado do sistema que está a ser desenvolvido. O mecanismo de publicação de tópicos subjacente permite fazer uma introspecção espontânea dos dados que fluem pelo sistema, facilitando a compreensão e a resolução de problemas à medida que ocorrem. As ferramentas do ROS tiram vantagem dessa capacidade de introspecção por meio de uma ampla colecção de utilitários gráficos e da linha de comandos, que simplificam o desenvolvimento e a resolução de problemas. Com ROS é possível ter toda uma arquitectura de processamento de dados em paralelo para controlar o robô. O ROS oferece ferramentas para criação de pacotes em C++ e Python. Cada pacote pode conter um número arbitrário de nós [43], [44]. A estrutura dos nós do ROS, implementada neste sistema proposto pode ser encontrada na figura 2.10.

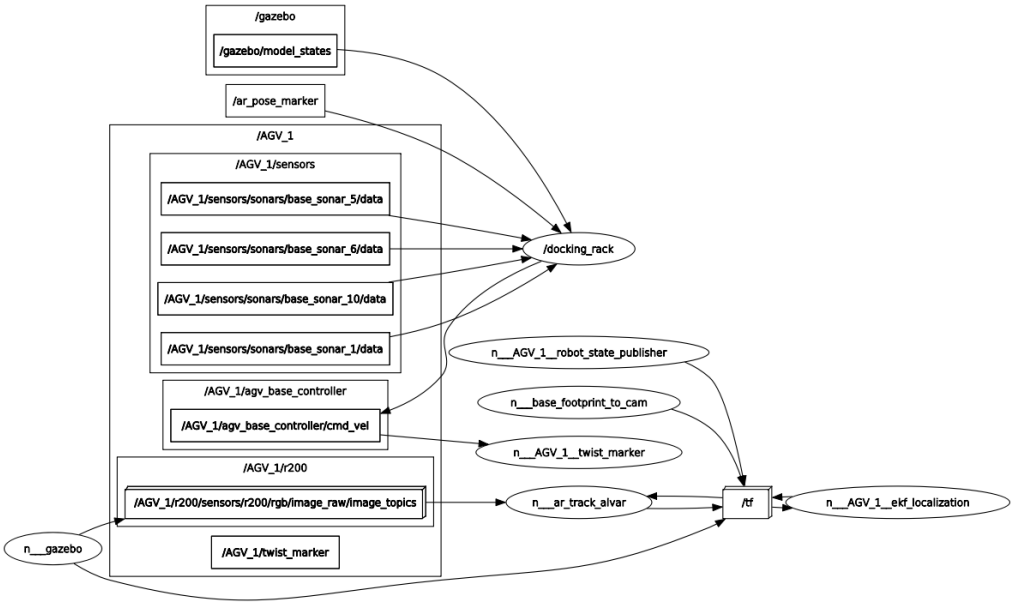

Figura 2.10: Exemplo de uma estrutura dos nós

2.5.3 Gazebo

O Gazebo, é um simulador de robôs usado para simular um ambiente físico. A simulação de robôs, é uma ferramenta essencial na construção e implementação de projetos na área da robótica. Uma boa estrutura e construção de um robô em simulador, torna possível testar algoritmos, realizar testes e monitorizar o desempenho de sensores e dados de atuadores. Com isso, o Gazebo oferece a capacidade de precisão e eficiência de simular sistemas robóticos em ambientes internos e externos complexos. É um simulador robótico baseado em ROS, que tem uma interface Cliente / Servidor, arquitetura e um modelo *Publish / Subscribe* de comunicação entre processos. Os objetos de simulação, exibidos em gráficos 3D avançados, podem ser associados a um ou mais controladores, que comandam o processo para controlar o objeto numa simulação dinâmica com vários motores de alto desempenho [45]. A possibilidade de simular um ambiente de uma forma virtual, permite desenvolver e testar um robô por completo num computador. Sendo uma ferramenta baseada em ROS, completamente gratuita, existe uma grande comunidade académica que faz uso do simulador para desenvolver projetos na área da robótica. É possível então criar um AGV e testa-lo de forma totalmente gratuita. Sem este simulador, o desenvolvimento desta dissertação seria muito mais complexo, visto que todos os componentes de um AGV são de custo elevado. Esta ferramenta providencia uma mais valia para o desenvolvimento deste projeto. Através de ficheiros URDF e SDF foi possível dimensionar o AGV. Os pacotes em questão foram fornecidos pela Introsys que tem este projeto a decorrer. Com base nestes princípios, todas as simulações e testes para a implementação do sistema de identificação da *rack*, no âmbito desta dissertação, foram efetuados com recurso a este simulador virtual .

2.5.4 Rviz

O Rviz (ROS visualization) é uma ferramenta de visualização 3D para aplicativos ROS, que fornece uma visão do modelo de robôs, captura informações dos sensores do robô e reproduz os dados capturados, podendo exibir dados de câmaras, lasers, de dispositivos 3D e 2D, incluindo imagens e nuvens de pontos (PCL). Este software tem ainda marcadores de visualização (visualization markers), que permitem ao programador enviar mostradores como cubos, setas e linhas coloridas, como o usuário pretender e programar. A combinação de dados do sensor e visualização personalizada de mostradores, tornam o Rviz uma ferramenta poderosa para o desenvolvimento das capacidades do robô. Este, é utilizado nesta dissertação em concreto, para se verificar o que está a ser lido pela câmara Real Sense

R200. Os dados são transmitidos do simulador gazebo para o Rviz em tempo real. De uma forma resumida, consiste em interpretar os tópicos publicados pelo simulador Gazebo e apresentá-los de uma forma organizada e com uma interface para melhor entendimento e facilidade na interpretação dos dados extraídos [46].

2.5.5 Python

Phyton é uma linguagem de programação de fácil aprendizagem e versátil na sua utilização, isto é, pode ter uma aplicação simples, mas efetiva de programação orientada a objetos. Uma das suas principais características é facilitar a leitura de códigos-fonte, exigindo poucas linhas de código. Conjuga uma organização da linguagem clara e concisa, com os recursos provenientes de uma variada coleção de subprogramas (biblioteca), e de ficheiros com código que facultam as capacidades das linguagens de programação com uma funcionalidade genérica [47]. Estas características conjuntamente com a sua natureza interpretativa, tornam esta linguagem ideal para algoritmos e desenvolvimento de software, nomeadamente na área da robótica. Este princípio justifica o recurso a esta linguagem, com o objetivo de facilitar o desenvolvimento de um conjunto de funções que facultam um processo de desenvolvimento de código mais simplificado.

2.5.6 OpenCV

OpenCV (*Open Computer Vision*), que na sua tradução significa visão por computador aberta, é uma biblioteca de código aberto desenvolvida para processamento de imagem, visão por computador, *machine learning* e, na sua última actualização, é capaz de acelerar a unidade de processamento gráfico (GPU) o que reduz o tempo das operações em tempo real [48]. Sob a licença de distribuição de software Berkeley (BSD), a biblioteca OpenCV é gratuita para uso comercial e de pesquisa, portanto, fácil de usar e modificar conforme a finalidade desejada. Focando em eficiência computacional voltada para aplicações em tempo real, fornece um bom suporte para expandir a percepção da máquina em tempo real e o desenvolvimento de uma ampla variedade de aplicativos. tais como [49]:

- Extrair modelos de objetos em 3D,
- Detetar e identificar *tags*,
- Comparar imagens,
- Detetar movimentos,

Através desta biblioteca foi possível analisar a imagem que estava a ser lida pela a câmara e com isto detectar e identificar as *tags* pretendidas. Esta biblioteca é utilizada pelo pacote de software recorrido para implementação do sistema proposto nesta dissertação, que será mencionado posteriormente.

2.5.7 PCL

A biblioteca de nuvem de pontos, do estrangeirismo *Point Cloud Library* comummente conhecida como PCL, é uma biblioteca extremamente útil quando o objectivo principal é processar nuvens de pontos 2D ou 3D. Lançado sob a licença BSD, esta biblioteca de código aberto, desenvolvida em C ++ e Python é compatível com Linux, Windows, MacOS X, Android e iOS. Inclui algoritmos de filtragem, necessários para processar quaisquer dados brutos da nuvem antes de aplicar operações de pedido (ou seja, remoção de pontos ruidosos), algoritmos de segmentação, que permitem analisar apenas partes relevantes da nuvem, bem como estimativa de recursos, ajuste de modelo e reconstrução de superfície, algoritmos importantes para extrair pontos-chave, reconhecer e analisar objetos independentes. A maioria dos seus cálculos matemáticos usa Eigen [50], outra biblioteca de código aberto direccionada para resolver álgebra linear operações com matrizes e vetores [51]. Uma vez que a capacidade processamento é uma grande preocupação de eficiência, o PCL pode ser dividido em bibliotecas de código menos "pesado", que podem ser compiladas de forma independente, melhorando o desempenho do sistema usando apenas os recursos necessários para a finalidade do sistema. Assim, o PCL demonstra mecanismos importantes para ajudar na parte prática da dissertação. Foi necessário o recurso a esta biblioteca através do pacote de software que será mencionado posteriormente, com o intuito de descobrir a pose de uma *tag* que por sua vez dita a pose da *rack* em que a *tag* está montada.

2.6 Considerações finais

Neste levantamento do estado de arte verificou-se que já existem algumas técnicas e soluções para o problema em questão nesta dissertação. Contudo, a informação encontrada e disponibilizada nas plataformas de pesquisa, foca-se no reconhecimento de paletes e em empilhadores, enquanto que esta dissertação tem como propósito a identificação, orientação e localização do AGV em relação a uma *rack*. Deste levantamento, foi possível identificar algumas técnicas e ideias para o desenvolvimento e estudo da temática em questão nesta dissertação, sem esquecer que têm que ser adaptados para o propósito em questão, AGV e *rack*. As dimensões standard das *rack* e do AGV serão então algo que irá diferenciar este trabalho, dos que foram revistos nos estado de arte, pois as dimensões e características do objeto (*rack* vs palete) não são iguais. A diferença é, que o AGV irá para baixo da *rack*, ou seja, o orifício da *rack* será maior para que o AGV possa lá entrar, à diferença dos empilhadores que tem dois garfos como ferramenta de empilhamento, e das paletes que tem dois orifícios menores. Verificou-se que os requisitos para maior sucesso do sistema de visão estão maioritariamente relacionados com: mudanças de iluminação, diferentes tipos de marcas no piso das fábricas, pontos de referência degradados ou até mesmo desgastados, ou seja, o desempenho deste sistema está dependente da visualização dos pontos de referência. Concluindo, podem então utilizar-se algumas técnicas já utilizadas noutros sistemas. A adaptação destas técnicas para o objectivo desta dissertação será então um desafio. Pretende-se melhorar, aperfeiçoar e adaptar algumas das técnicas referidas neste estado de arte, ou recriar novas formas de abordar tópicos como identificação de objetos e navegação do AGV.

Capítulo 3

Sistema Proposto

Neste capítulo é apresentado o sistema proposto para identificar e localizar racks, bem como o sistema de navegação para o acoplamento do AGV. Sendo que, na primeira secção deste capítulo é apresentado o equipamento utilizado, AVG, rack e as tags utilizadas para identificar as racks.

3.1 Equipamento

Para melhor entendimento do sistema proposto, justifica-se fazer uma breve introdução ao equipamento que foi utilizado para o desenvolvimento do sistema. Será descrita, nesta secção, a arquitetura do AGV, bem como a câmara que nele se propõe instalar, e ainda a estrutura do sistema de transporte denominado RACK, no qual estão instaladas *tags* (também apresentadas nesta dissertação) para um acoplamento efetivo da RACK.

3.1.1 Arquitetura do AGV

O modelo de AGV (figura 3.1), já abordado no primeiro capitulo é um veiculo autónomo. Como tal, para poder fazer uma navegação autónoma o AGV necessita de ter componentes de *hardware* instalados. O AGV projetado pela Introsys e utilizado para esta dissertação é composto por um chassi e carcaça que são a sua estrutura, motor que permite a navegação, sensores para extrair dados relevantes e micro-processadores para controlar todas as unidades. De modo a que fosse simulado o funcionamento do AGV no simulador foi necessário criar os

ficheiros/pacotes para fazer um *launch* do robô, tendo estes ficheiros sido fornecidos pela Introsys S.A. De forma sucinta, pode-se dizer que a estrutura do AGV é importante ter em consideração para o desenvolvimento do sistema proposto, contudo não é um fator determinante, desde que os ficheiros/pacotes do robô sejam compatíveis com os do sistema, pode incorporar-se ao mesmo. Pode-se então dividir a arquitectura do sistema/AGV no simulador Gazebo (descrito na secção 2.5.3) em 3 partes: sensores de aquisição de dados, processamento dos dados e navegação. Para além dos sensores de aquisição de dados, o AGV tem também instalado sensores sonar, que programados são capazes de evitar colisões (estes não serão abordados nesta dissertação).

Outro elemento essencial e alvo de estudo nesta dissertação é a instalação de uma câmara com capacidade de determinar a profundidade de uma imagem (secção 3.1.2), sendo esta a câmara utilizada para o sistema proposto nesta dissertação. O processamento dos dados é feito através do processador da máquina onde está a ser simulado, capaz de comunicar com todos os componentes do AGV através do ROS. O AGV também tem implementado outros módulos, como o deteção de obstáculos, movimentos para certas posições, etc, que não são relevantes para o processo de acoplamento. Contudo para segurança do equipamento e das linhas de montagem, o sistema de deteção de obstáculos (não incluído nesta dissertação) deve correr em paralelo, para que não ocorram colisões. Quanto à navegação, é feita através de dois motores elétricos que possibilitam o AGV fazer movimentos retilíneos (deslocar-se para a frente e para trás) tal como girar em torno de si próprio 360º. O AGV é um veiculo autónomo com agilidade suficiente para se efetuar as tarefas necessárias em intra-logística.

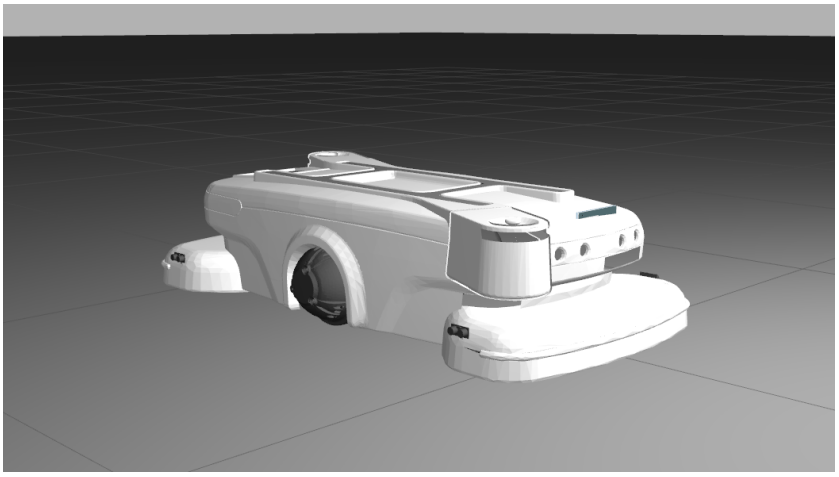

Figura 3.1: AGV utilizado, cortesia da Introsys S.A [2]

3.1.2 Câmara

Para o desenvolvimento do sistema proposto foi necessário a utilização de uma câmara com recurso a *depht image*, para extrair os dados relativos ao espaço e objetos no ambiente em que o AGV se encontra. A câmara utilizada foi uma câmara produzida pela Intel [52] de nome *Real Sense R200* (figura 3.2). A escolha desta câmara, baseou-se na capacidade específica na reconstrução 3D e mapeamento de ambientes em seu redor, apresentando um índice de erro associado menor do que as outras câmaras [53]. Esta câmara produzida pela Intel, é capaz de extrair dados relativos à imagem, ter acesso a dimensões de profundidade dos objetos e do ambiente que a câmara está a ler e ainda produzir vídeo infravermelho em tempo real. O vídeo em tempo real com *depht image*, consiste em caracterizar a cada pixel do *frame* de imagem, uma distancia, ou seja, a cada pixel da imagem, corresponde uma distancia de uma nuvem de pontos gerada pelos sensores infravermelhos. A *depht image*, acontece devido à existência de dois sensores infravermelho na câmara. A projeção do sensor laser infravermelho, permite caracterizar um determinado pixel quanto á sua cor e distancia a que se encontra da câmara. O vídeo em tempo real com profundidade é então gerado através da junção de vídeo real e os dois sensores infravermelhos, resultando numa reconstrução em 3D do campo de visão da câmara (figura 3.3).

Sendo a câmara frequentemente utilizada na área de robótica, esta contém vários pacotes para simulação no simulador Gazebo, em código aberto, permitindo assim a utilização desta para implementação do sistema proposto nesta dissertação.

Figura 3.2: Real Sense R200 (cortesia da Intel [52])

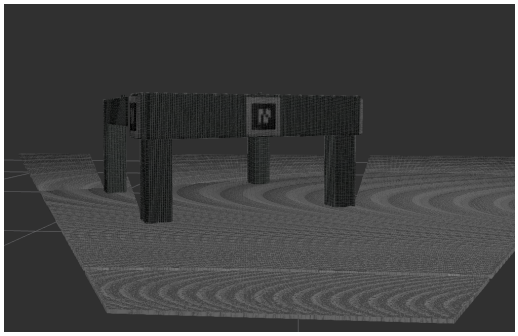

Figura 3.3: Exemplo do campo de visão da câmara com uma nuvem de pontos associada

3.1.3 Plataforma de transporte

Uma *rack* é denominada pela intra-logística como uma plataforma de transporte.É um componente fundamental na industria pois é com este sistema de transporte que é feito o transporte de materiais. É por norma uma estrutura de metal reforçado capaz de suportar o transporte de vários tipos de materiais ou cargas. No contexto desta dissertação e para fins de implementação é importante saber medidas referentes ao modelo da *rack*, tendo esta uma dimensão de 1x1metros e uma altura de cerca de 50cm. Para efeitos de simulação e teste, foi feito um modelo inicial com as mesmas características de uma *rack* convencional, 4 pernas de apoio ao solo e uma base onde possam ser inseridas as cargas. Atendendo a que o intuito desta dissertação não é o desenho da *rack*, foram apenas retiradas as características necessárias para a implementação do processo de acoplamento. O modelo da *rack* utilizado foi construído utilizando a aplicação *blender* e [54] pode ser melhor compreendido na figura 3.4.

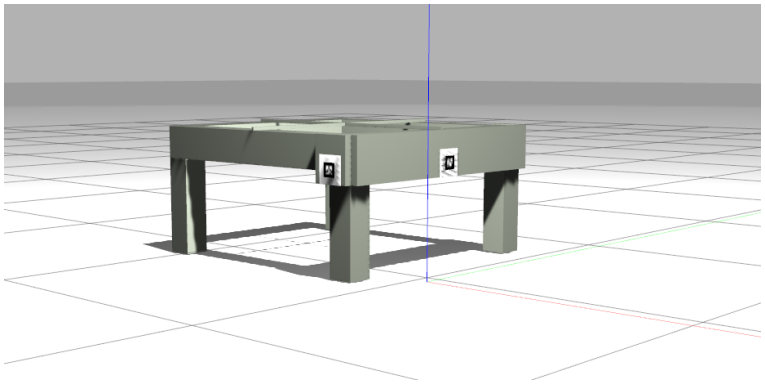

Figura 3.4: Modelo de *rack* utilizado

Neste trabalho propõe-se a utilização de *tags* para suportar a identificação das racks. O modelo de *rack* proposto é composto por 3 *tags* diferentes. Duas *tags* identificam o lado (uma identifica o lado esquerdo, e outra o lado direito) enquanto a terceira identifica a face de entrada do AGV na *rack*, para fazer o acoplamento. A *tag* da face de entrada é a *tag* principal. Sendo por exemplo a *tag* principal a de ID 2 (figura 3.6), a do lado esquerdo é 2-1= 1 (figura 3.5) enquanto a do lado direito é a 2+1=3 (figura 3.7), ou seja, a *tag* correspondente à face do lado esquerdo é de ID 1. As *tags* laterais foram colocas mais próximas dos cantos da face de entrada do AGV pois presume-se que na intra-logística os sistemas de transporte estão encostados a superfícies (parede ou linha de montagem) e podem ter outras *racks* a seu lado, aumentando assim a visibilidade e probabilidade do sistema encontrar a *tags* pretendida. Não se entendeu relevante colocar uma *tag* na parte traseira da *rack* (também devido este motivo), contudo o algoritmo implementado está

preparado para reconhecer, caso se pretenda, uma *tag* traseira e facilmente ser interpretada pelo algoritmo como correspondente à face traseira. Um requisito essencial para a identificação da *tag* pelo algoritmo, é que *tag* tenha um plano de fundo onde está montada. Sabendo onde estão montadas as *tags*, pela identificação da pose das mesmas, é possível saber a pose da *rack*, pois conhece-se a localização das *tags* e as dimensões da *rack*.

3.1.4 *Tags*

Como já referido, de modo a fazer-se a identificação da *rack*, são instaladas 3 *tags*. O recurso à utilização de *tags*, permite que através do ID da *tag*, seja criada uma base de dados que permita o armazenamento de informação, em relação a que material (a ser transportado) está na *rack*, possibilitando ao AGV identificar especificamente a *rack* a transportar.

Por exemplo, numa fábrica automóvel, existem várias linhas de montagem que podem produzir peças de diferentes tipos. Sendo o transporte feito pelas *racks* é necessário informar o AGV, que *rack* se pretende transportar, caso contrário, se as *racks* não fossem identificadas pela *tag*, teriam de ter um qualquer outro tipo de identificação para que o AGV pudesse identificar e diferenciar a *rack*. Através das *tags* é também possível identificar a pose e orientação da mesma ou do plano em que está montada, permitindo assim alcançar um dos objetivos desta dissertação.

As *Tags* utilizadas neste trabalho foram apresentadas por *Mark Fiala* em 2004 e foram inspiradas numa biblioteca chamada de ARtoolkit(1999) [55]. São marcas que consistem num quadrado que está dividido em quadrados pretos e brancos, representa um código binário "1"ou "0"respectivamente. Tendo uma moldura de quadrados pretos que delimita a *tag*, o código correspondente à identificação da mesma encontra-se dentro dessa moldura quadrada, como um código de barras por exemplo. Existem várias dimensões e tipos de *tags*, sendo que quando é criada a *tag* tem que ser introduzida a quantidade dos píxeis a qual corresponde o código binário. Neste caso foram criadas *tags* de 9x9 quadrados (figura 3.6), na qual a moldura que delimita a *tag* está a ocupar todo o contorno da *tag*, faltando assim 5x5 para o código binário que contem o ID da *tag*. O quadrado 5x5 contém o código bem como a orientação da *tag*.

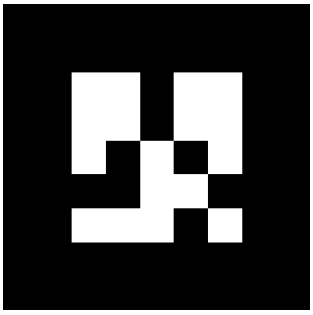

Figura 3.5: *Tag* ID=1

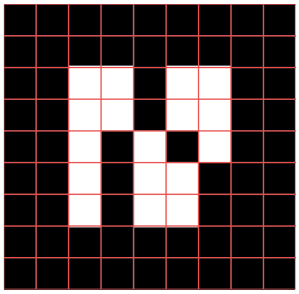

Figura 3.6: *Tag* ID=2

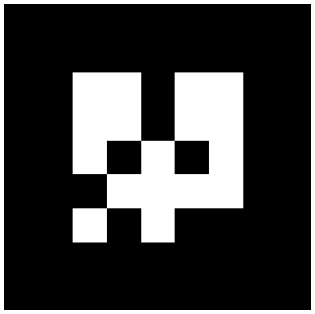

Figura 3.7: *Tag* ID=3

3.2 Sistema de identificação e acoplamento do AGV à *rack*

De modo a fazer uma correta aproximação e acoplamento do AGV à plataforma de transporte (*rack*) é necessário, localizar o AGV em relação à *rack*, o que implica a utilização de um pacote de software providenciado em código aberto de nome *ar track alvar* [56]. Este pacote de software, é utilizado como sendo um sistema de "caixa preta", isto é, o seu funcionamento interno não é tido em conta no âmbito desta dissertação, sendo apenas utilizado o resultado obtido através da sua utilização. Contudo, para que se compreendam as implicações da utilização deste sistema, apresenta-se de seguida uma breve descrição do seu modo de funcionamento.

Depois da identificação da *rack*, através da *tag* instalada para o efeito, e da obtenção da posição desta, conhecendo-se as dimensões da *rack*, é processado o acoplamento pelo AGV

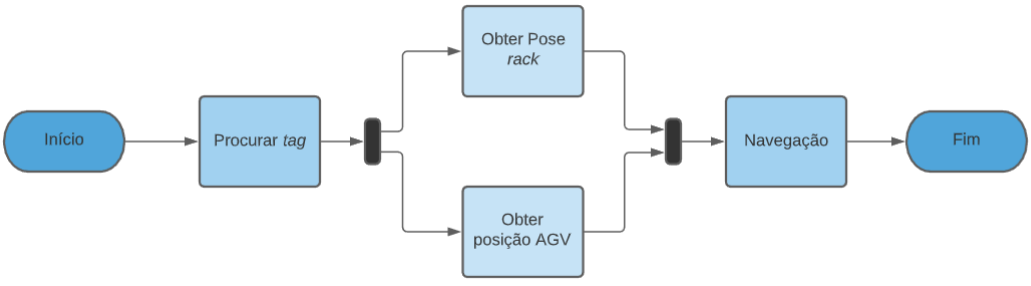

Figura 3.8: Diagrama de actividade do sistema proposto

Pelo diagrama de atividades (figura 3.8) é possível compreender melhor o sistema proposto. Inicialmente, procede-se à procura da *tag*, quando esta é identificada, é extraída a sua pose. De seguida, os dados relativos á pose da *tag*, assim

como, do AGV são processados em paralelo, e é então efectuado o processo de navegação para o acoplamento do AGV à *rack*. A posição do AGV em relação ao espaço, é necessária para que este se movimente, ou seja, quando se pretende proceder ao movimento, acionam-se os motores e verifica-se qual a variação da pose do AGV até estar de acordo com as distâncias ou ângulos inseridos para a movimentação. Finalizando, presume-se que o AGV esteja na posição final de acoplamento, ou seja, prevê-se que esteja posicionado por baixo da *rack* para se efectuar o processo de empilhamento da *rack*, o qual não é tido em conta no âmbito desta dissertação.

3.2.1 Identificação e localização da pose da *tag*

3.2.1.1 Pacote de software utilizado

De forma a identificar a *rack* e a sua pose, começa-se por identificar a *tag* montada na mesma. Para tal, é utilizado o pacote de software de *ar track alvar* [56], de realidade aumentada (AR), fornecido pelo autor *Scott Niekum*, este pacote faz parte de um conjunto de ferramentas de realidade aumentada ALVAR [57], como é o caso dos autores de [58], que o utilizaram para localizar um determinado robô em relação a uma *tag*. Estes propuseram um sistema de aterragem de um *drone quadcopter* de uma forma autónoma, numa região identificada pela *tag*, tendo este sido aprovado pelos autores, um bom sistema para identificação e localização de um robô. O pacote *ar track alvar* foi adotado para o desenvolvimento do sistema proposto nesta dissertação pois é um pacote de software atualizado e em que a compatibilidade com a câmara utilizada e com os pacotes de software do AGV é adequada. Existem outros pacotes de software para identificação de *tags* (ar sys, ar pose, visp auto tracker, and ar track alvar) [59]. Contudo, no inicio da pesquisa de identificação de *tags* optou-se pelo recurso ao uso do pacote *ar track alvar*, pois os outros pacotes de software para identificação de *tags* ou não são compatíveis com o pacotes do AGV ou não tem uma taxa de sucesso aceitável.

O pacote de software *ar track alvar* providencia 3 principais funcionalidades importantes para o processo de identificação da *tag*:

- Gerar *tags* com os parâmetros pretendidos, tais como, tamanho, resolução, podendo ainda serem guardados dados tais como ID da *tag*;

- Identificar a pose *tag* gerada, podendo ser só identificada através do tratamento de imagem ou também com a introdução de imagem de profundidade (*pointcloud*, *depht image*);

- Providencia como resultado, distancias até á *tag*, posição e orientação da mesma;

Para uma identificação das *tags* em questão é necessário introduzir alguns parâmetros ao pacote de software para que reconheça as *tags* pretendidas. Estes parâmetros são:

- Tamanho lateral da *tag* a ser identificada (*marker_size*);
- Erro associado à deteção de uma *tag* (*max_track_error*);

Adicionalmente, é necessário introduzir tópicos de entrada ao software como:

- Tópico associado à imagem fornecida pela câmara, pode ser apenas a imagem ou a imagem com profundidade (*camera_image*);
- Nome do tópico publicado pela câmara, que providencie informações para calibrar a câmara (*camera_info*);
- Nome do tópico que publica as coordenadas do AGV (*output_frame*);

3.2.1.2 Identificação da *tag*

O processo de identificação consiste em procurar candidatos correspondestes a *tags*, ou seja, pela análise da imagem, o sistema procura quadrados que possam corresponder a candidatos a *tags*. Um filtro é aplicado extraindo as formas que estão na imagem dada como *input*, que não correspondem a forma geométricas quadradas, ou seja, através de um *threshould* adaptado, os contornos correspondentes a formas quadradas são selecionados como candidatos, enquanto que os contornos que não correspondem a quadrados são extraídos. Esta primeira etapa consiste basicamente em fazer uma filtragem de possíveis candidatos a *tag*.

Após ter um ou mais possíveis candidatos a *tag*, esses candidatos são analisados de maneira a descodificar e verificar o código binário inserido na *tag*, com o intuito de saber se essa *tag* corresponde à *tag* pretendida (figura 3.9). Para tal, para cada quadrado preto ou branco é colocado um novo quadro de menor dimensão dentro do correspondente para se ter certeza da cor a que corresponde (preto ou branco) (figura 3.10).

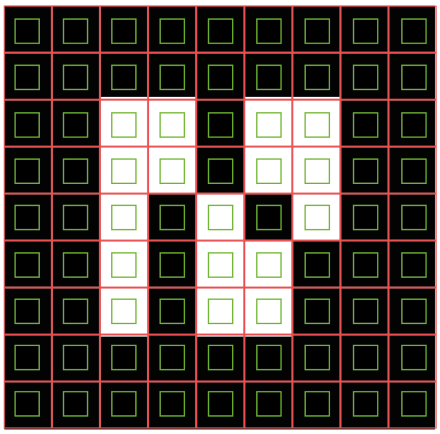

Figura 3.9: Marker ID=2 com a divisão dos píxeis

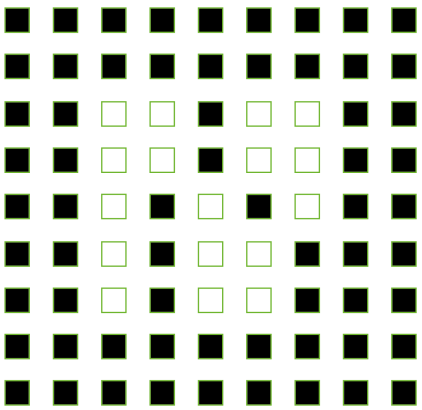

Figura 3.10: Resultado da extração do código binário

A partir daqui, para a descodificação do código correspondente aos píxeis 9x9 só se utilizam estes quadrados verdes (figura 3.10) em que o quadrado preto corresponde a 1 e o quadrado branco corresponde a 0. Seguidamente é criada uma matriz correspondente ao código dos 0 e 1 da *tag*, possibilitando então que seja feita a descodificação e correspondência do código do ID, como se fosse um código de barras e as *tags* são identificadas como correspondentes ou não. Todo este processo é efetuado pelo pacote de software *ar track alvar*.

3.2.1.3 Localização da *tag*

Aquando da identificação de uma *tag*, é efetuado o processo que permite a identificação da pose da mesma. Este processo consiste em identificar dois pontos distintos, que correspondem à superfície plana onde se encontra a *tag*, de modo a definir a pose desse mesmo objecto, neste caso a *rack*. Este processo é efectuado pela biblioteca PCL e OpenCV em simultâneo, através de filtros, mediante os quais é possível selecionar a região plana onde a *tag* está montada (figura 3.11). Determinada a região a que se pretende obter a pose, encontram-se dois pontos desse plano para se retirar a pose, através de algumas contas, é possível descobrir o ângulo que a face de entrada da *rack* faz com o AGV.

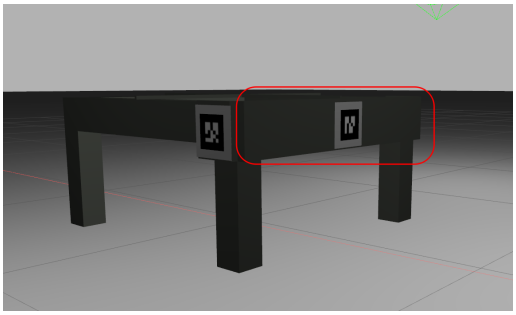

Figura 3.11: Plano que se pretende a pose, representado pela região a vermelho

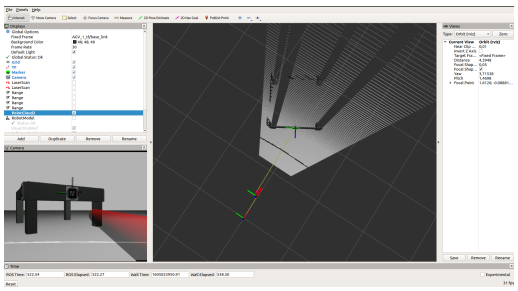

Figura 3.12: Resultado do software Rviz referente ás poses do AGV e *tag* identificada

O pacote de software *ar track alvar* disponibiliza todo este procedimento dando um resultado da posição e localização da *tag* inserida em relação a uma posição que neste caso é a do AGV. A deteção da *tag* pode ser melhor compreendida através da figura 3.12 que é um resultado do software *Rviz* que interpreta os dados gerados pelo pacote de software utilizado. No sistema de coordenadas apresentado é possível verificar as posições relativas à pose do AGV e da *tag* identificada e correspondente, bem como a imagem que está no campo de visão do AGV.

3.2.1.4 Interpretação dos dados

É essencial, para que o processo de acoplamento aconteça, que os dados da pose e ID da *tag*, obtidos através do pacote de *software*, sejam interpretados da melhor forma. Então, de modo a orientar-se o AGV em relação ao espaço em que se encontra, é criado um sistema de coordenadas (X,Y,Z) da sua posição inicial, bem como o sistema de coordenadas da localização da *tag*:

- Sistema de coordenadas do AGV em relação ao mundo (X,Y,Z);
- Sistema de coordenadas do AGV em relação à *tag* (X,Y,Z)

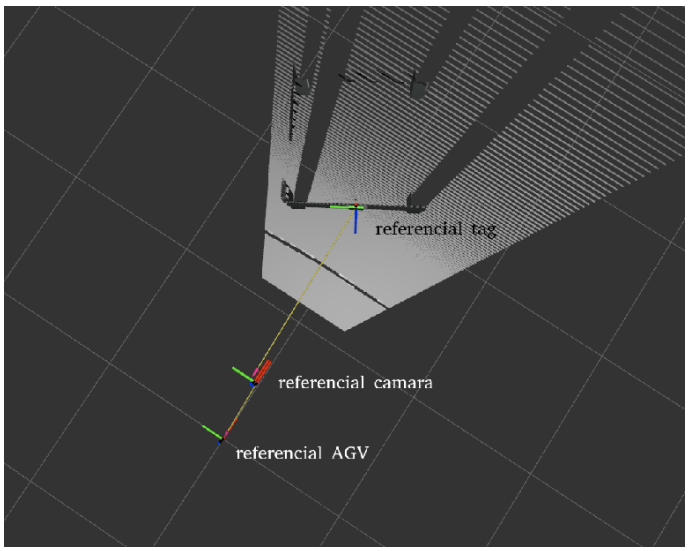

Figura 3.13: Referenciais AGV e *tag*

Pela figura 3.13, pode verificar-se através do visualizador Rviz a representação da pose da *rack* em relação ao AGV.

O sistema de coordenadas do AGV em relação ao mundo justifica-se quando é detetada a pose de uma *tag*, de forma a que o AGV consiga localizar-se em relação ao espaço em que se encontra, assim como, quando é feito o processo de acoplamento. O sistema de navegação consiste em verificar a variação em relação á pose inicial, ou seja, quando são acionados os motores, verifica-se a diferença entre a pose actual do AGV e a sua pose inicial, determinando-se com base nesta diferença o ângulo ou distância que se pretende variar. Quando esta diferença é igual à instruída ao sistema, os motores são parados.

O sistema de coordenadas do AGV em relação à *tag*, é o que dita que distâncias e ângulos devem ser instruídos aos tópicos dos motores para que se proceda ao acoplamento. Para se obter este sistema de coordenadas, visto que os dados são

retirados pela câmara, é feita uma translação para que se obtenha os dados de pose relativamente ao centro do AGV e não à câmara. O resultado gerado pelo o pacote de software 3.14 é divido em duas partes, localização e orientação:

- Sistema de coordenadas da localização da *tag* (X,Y,Z)
- Sistema de coordenadas da orientação da *tag* (x,y,z, w)

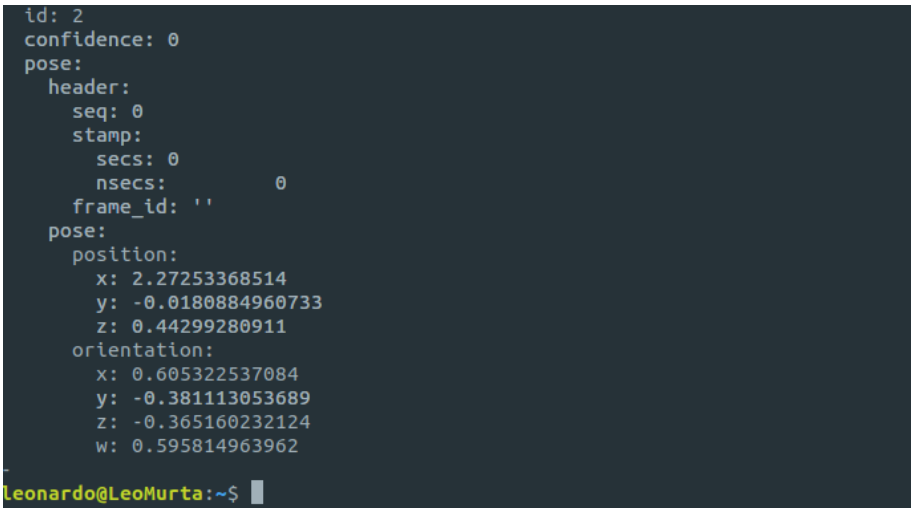

Figura 3.14: Resultado obtido da pose de uma *tag*

Como se pode verificar pelo *output* na figura 3.14 (apresentada como exemplo), os dados referentes á localização e orientação da *tag* são apresentados em 3 dimensões, (X,Y,Z). Contudo para o processo de acoplamento, apenas são necessários os dados em duas dimensões (x,y). Isto deve-se ao AGV navegar num plano em apenas duas dimensões. A figura 3.15 exemplifica o AGV alinhado com a *tag*, sendo o *output* do *software* utilizado o da figura 3.14

Analisando os dados obtidos pelo sistema de identificação (figura 3.14, verifica-se que a *tag* se encontra a uma distancia estimada de 2.27cm, (na figura 3.15 representada pela letra B), e um ângulo com o eixo x de 0.60rad (na figura 3.15 representada por β). Pode-se verificar também, que o ID da *tag* é 2. Com estes dados sabe-se a que distancia o AGV está da *rack*, e que ângulo a face frontal da *rack* faz com o AGV. Torna-se então possível calcular o movimento do AGV para inicio do acoplamento. O cálculo do ângulo (na figura 3.15 representada por α) e da distância (na figura 3.15 representada por A) necessários para o movimento do AGV estão descritos pelas equações seguintes:

$$B = \sin((1)) \div A$$

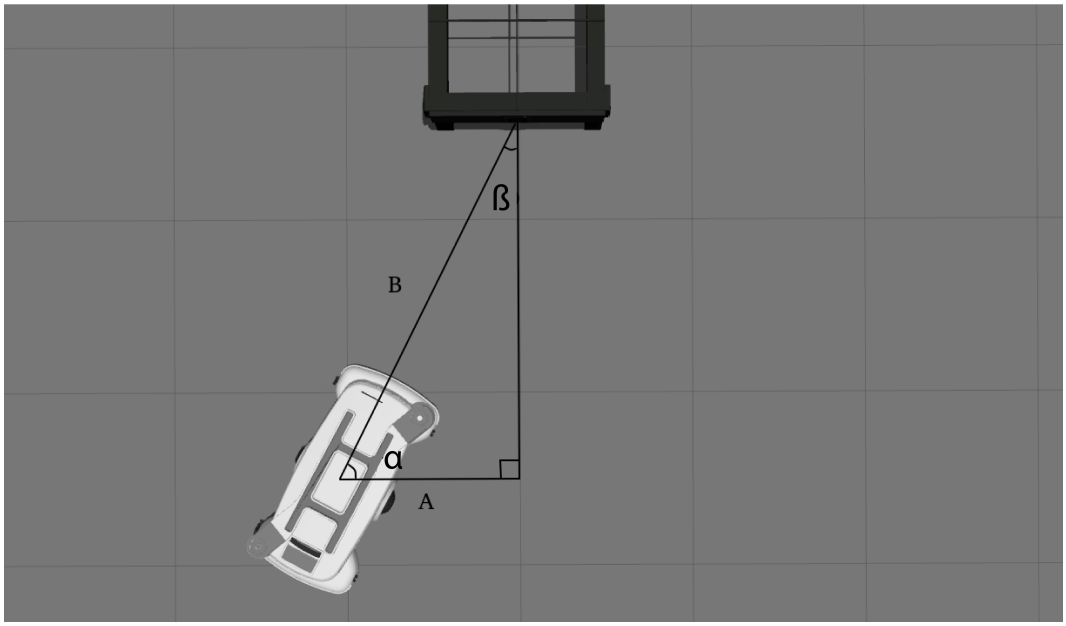

Figura 3.15: Interpretação da pose do AGV em relação à *rack*

$$\alpha = 3,14 - (3,14 \div 2) - A$$

De forma generalizada, este processo acontece quando é detetada uma *tag* e se pretende efetuar movimentos no AGV no sentido de navegação e acoplamento deste, o qual será descrito secção seguinte.

3.2.2 Navegação - aproximação e acoplamento do AGV à *rack*

O processo de navegação para o acoplamento do AGV à *rack*, é possível depois de uma validação do pacote de *software ar track alvar*, o qual providencias os dados relativos à pose da *tag*, que por sua vez correspondem à pose da *rack*. O sistema de navegação, está em constante comunicação com o pacote de software *ar track alvar*, ou seja, a partir da pose e orientação da *tag*, disponibilizada pelo *software*, são publicados os valores nos tópicos de movimento dos motores do AGV. Na sequência deste, o sistema comunica continuamente com o pacote *ar track alvar* durante as 5 diferentes etapas (discriminadas abaixo) do processo de navegação para o acoplamento. Isto significa que quando se detecta uma *tag* correspondente à que se pretende, o algoritmo permite um ajustamento ao percurso para acoplamento de acordo com a pose da *rack*, reduzindo os erros associados. Estes erros podem ocorrer, quando as medidas extraídas não correspondem (localização da *rack*), devido à menor precisão nos movimentos do AGV (variação de ângulos) e também a possíveis deslocações inesperadas da *rack* (a *rack* mudar de posição). No entanto, o sistema aqui proposto pretende anular a possibilidade de um acoplamento indevido, corrigindo constantemente os erros detetados.

O processo de acoplamento do AGV ao *rack* está dividido nas seguintes etapas:

- 1. Rodar em torno de si próprio de forma a procurar uma *tag* correspondente;
- 2. Aproximar-se da *rack*;
- 3. Colocar-se em frente á face de entrada;
- 4. Aproximar-se da face de entrada;
- 5. Seguir para baixo da *rack*;

O comportamento do sistema de navegação e acoplamento do AGV ao *rack* é descrito através do fluxograma apresentado na figura 3.16.

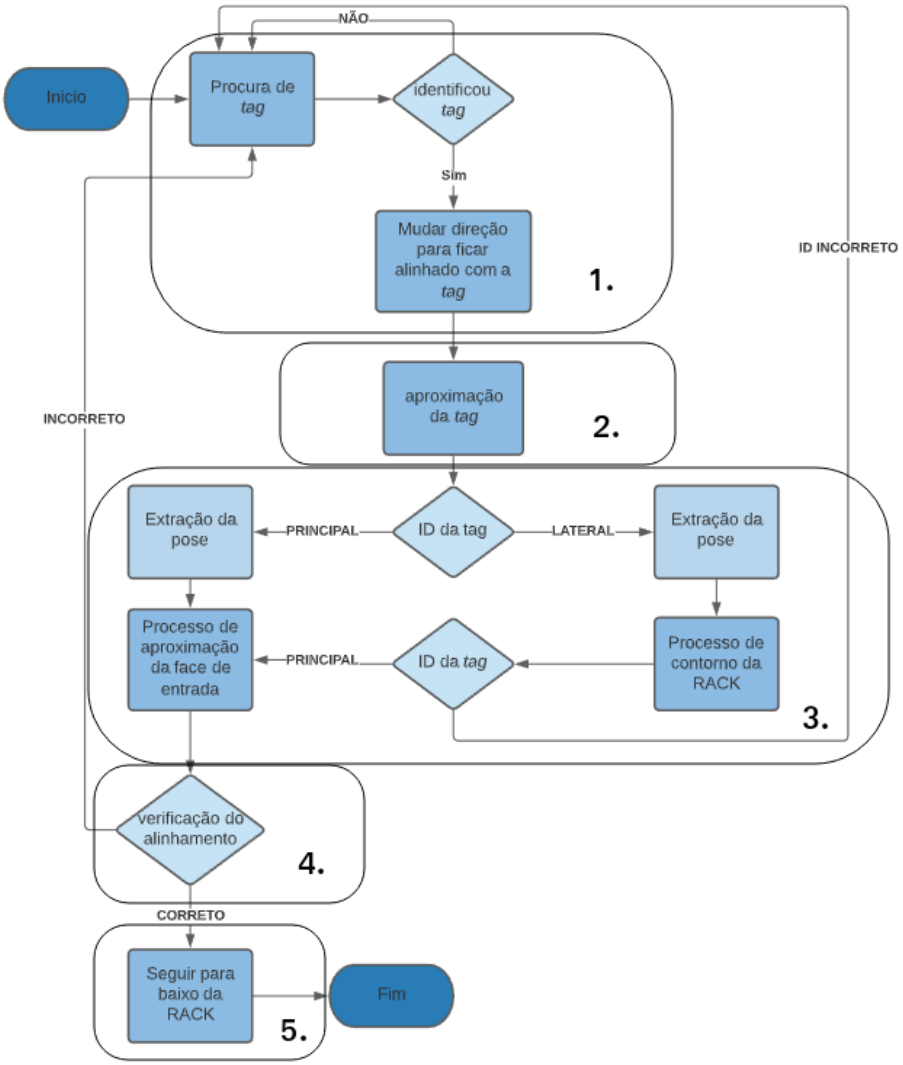

Figura 3.16: Fluxograma do sistema de navegação para acoplamento

Referente ao fluxograma apresentado na figura 3.16, segue-se a informação detalhada das etapas apresentadas:

1. Primeiramente o AGV roda em torno de si próprio para procurar uma *tag* numa área de 360º. assim que é detectada uma *tag*, são accionados os motores para que o AGV fique exactamente de frente para a *tag*. Essa representação pode ser melhor entendida pela figura 3.17. A partir dai são extraídos todos os dados necessários referentes a distâncias e ângulos para o cálculo da execução da primeira trajectória de aproximação.

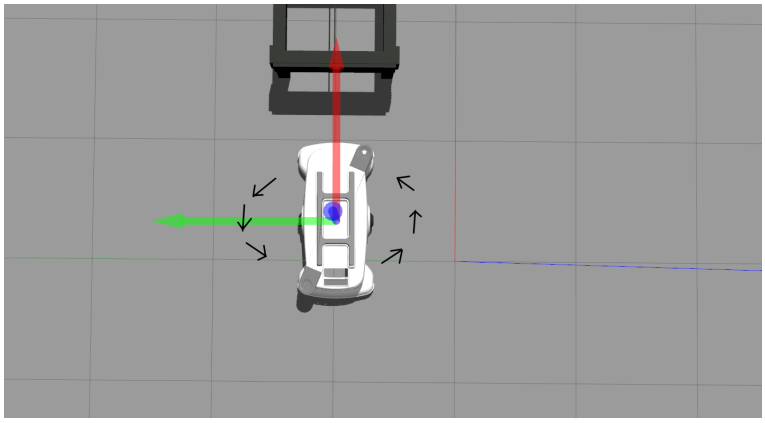

Figura 3.17: Procura de *tag*

2. Caso o AGV se encontre a uma distância de mais de 1.20 metros da *tag*, procede a uma aproximação para que quando o algoritmo extrair os dados relevantes para os cálculos, estes sejam mais precisos e com um menor erro.

3. Depois de se obter a pose e orientação da *tag* através do *package* referido na secção anterior e sabendo onde está montada a *tag* na *rack*, pode-se assumir que já se sabe a localização da *rack* em relação ao AGV. Tendo assim como input ao algoritmo de navegação os dados referentes à pose e orientação da *tag* e conhecendo as dimensões da *rack* existem agora dois casos possíveis, ou o sistema identificou uma *tag* correspondente à face de entrada do AGV (figura 3.18) ou identificou uma *tag* correspondente à face lateral (figura 3.19).

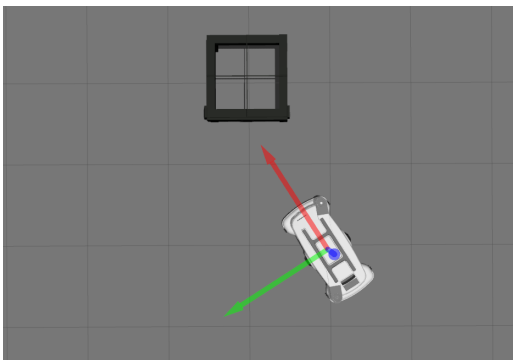

Figura 3.18: Pose do AGV alinhado com a *tag* de entrada

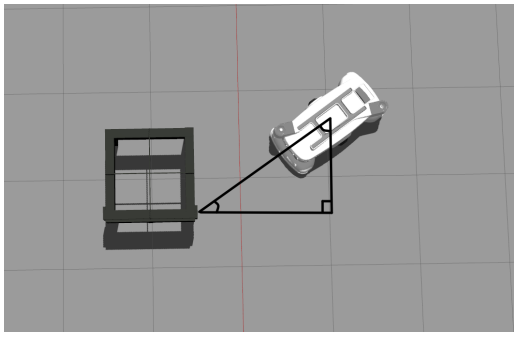

Figura 3.19: Pose do AGV alinhado com a *tag* lateral

Identificada a *tag* que corresponde ao ID de entrada do AGV na *rack*, é necessário saber os ângulos e distâncias do AGV à *tag*. Estes dados são providenciados pelo pacote de *software ar_track_alvar*. De uma forma mais simples são necessárias a distância do centro do AGV até à *tag*, e ângulo que a *tag* faz com o eixo de origem do AGV (representado na figura 3.20 por β).

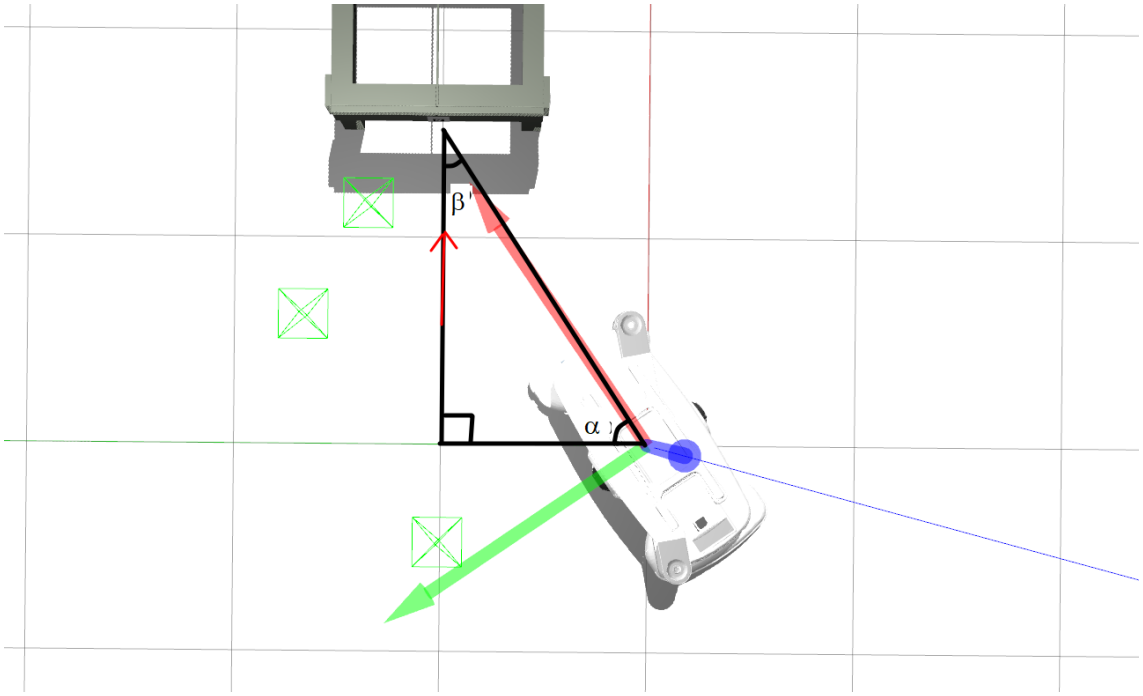

Figura 3.20: Enquadramento do AGV em relação à *rack*/*tag*

Nesta etapa, calcula-se a distância que o AGV irá deslocar-se para se posicionar rigorosamente de frente para a *tag*, ou seja, diretamente em frente à face de entrada, em que o ângulo β representado na figura 3.20, será zero aquando o AGV estiver nessa posição, bem como o ângulo α representado na figura 3.20, que será necessário para o cálculo do cateto menor. Ao obter o ângulo β e ao saber que o triângulo, é um triângulo recto, calcula-se o ângulo α pelo teorema de Pitágoras. Esse ângulo será o ângulo que o AGV terá que se mover para prosseguir ao avanço rectilíneo.

Neste caso, o algoritmo publica nos comandos do AGV, a informação necessária para que fique em paralelo com a face de entrada da *rack* (figura 3.21), de seguida o AGV avança até á posição em que fica exatamente em frente á face de entrada, para finalizar esta trajectória o AGV vira 90º em direção á face de entrada (figura 3.22).

Caso tenha sido identificada uma *tag* lateral o sistema sabe que essa *tag* corresponde a uma parte lateral do AGV, que pode ser ainda do lado esquerdo ou do direito. Visto que as *tags* tem ID diferentes correspondentes aos lado da *tag*, o AGV procede a uma trajectória para se deslocar para a face de entrada. Este processo utiliza os mesmos ângulos e distancia referidos anteriormente (distância do AGV à *tag* e o ângulo número β e α), como neste caso se encontram na face lateral, depois de avançar até estar em frente da *tag* em questão, ainda em paralelo, avança mais 1.4 metros, ou seja, em vez de virar os 90º para ficar de frente para *tag*, avança mais 1.4 metros depois de passar a *tag*, com isto o AGV irá aproximar-se

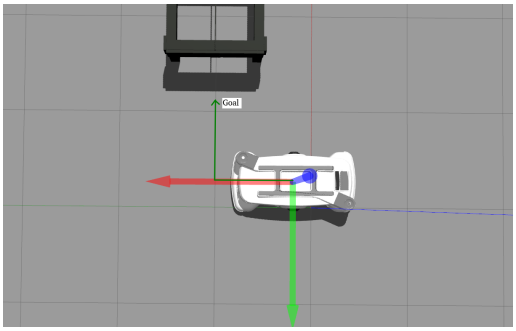

Figura 3.21: AGV paralelo com a face de entrada na *rack*

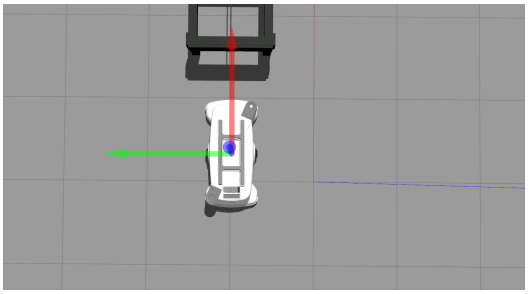

Figura 3.22: Pose de entrada do AGV na *rack*

da face de entrada. seguidamente depois de avançar esses 1.4 metros, vira então á direita ou esquerda, dependendo da face lateral em que encontra e avança 1.3 metros. este processo faz com que o AGV contorne a *rack* e fique em posição para conseguir visualizar a *tag* principal (entrada na *rack*).

Assume-se agora que o AGV se encontra em frente da face de entrada, é então feito um reconhecimento novamente, rodando em torno de si próprio, voltamos agora a fase número β, em que o AGV procura uma *tag* a qual corresponde a *rack*. O processo a partir de agora volta ao inicio, reduzindo assim os erros associados, caso o AGV fizesse o processo de acoplamento directamente sem retirar novas distancias e ângulos, o erro acumulado poderia ser muito maior, assim pode reduzir-se esse erro. O previsto nesta etapa é que o AGV veja a *tag* correspondente á face de entrada e irá fazer o processo de aproximação para fazer a acoplamento correctamente. A trajectória prevista no caso de encontrar uma *tag* lateral da *rack* pode ser melhor entendido na figura 3.23

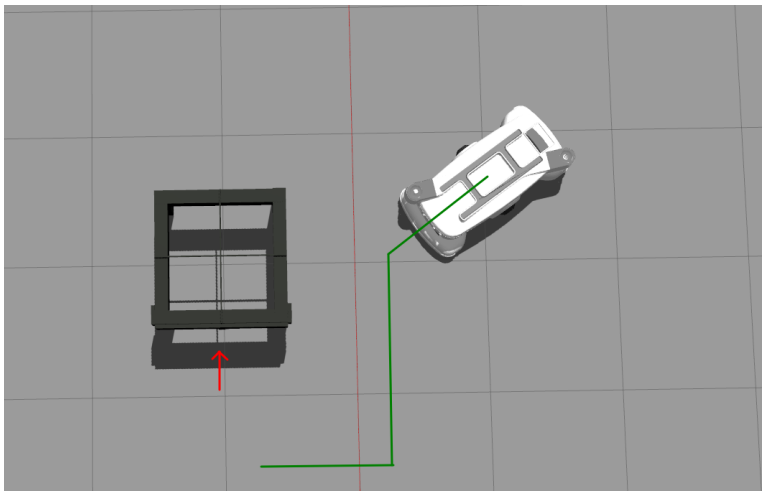

Figura 3.23: Trajeto (linha verde) do AGV para navegação até à entrada da *rack* (representado com a seta vermelha)

Existe ainda a hipótese do AGV ver as duas *tags* ao mesmo tempo, ou seja, num *frame* estarem duas *tags* correspondentes à mesma *rack* (figura 3.24) neste caso o AGV tem preferência pela *tag* de entrada, à excepção de quando o AGV não tem a capacidade de fazer o acoplamento direto e então escolhe a *tag* lateral e faz o trajeto de contorno da *rack*, representado na figura 3.25.

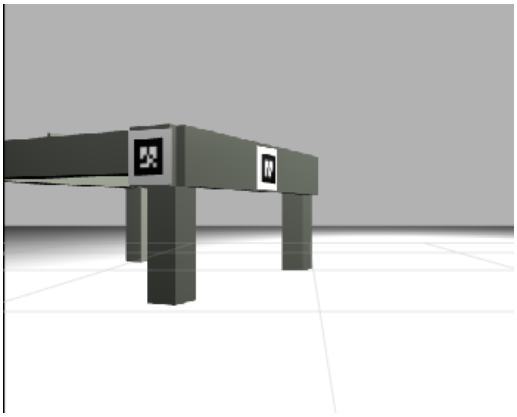

Figura 3.24: Perspectiva da câmara do AGV obtida pelo Rviz

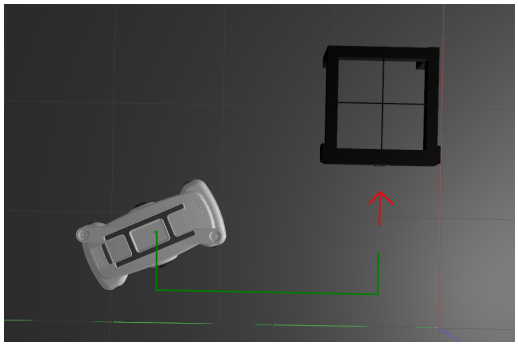

Figura 3.25: Trajeto previsto do AGV representado a linha verde para fazer contorno da *rack*

4. Encontrando-se o AGV de frente para a face de entrada existem duas possibilidades, ou o AGV está corretamente posicionado para entrar para por baixo da *rack* ou por motivos não previstos, pode a *rack* ter-se deslocado ou ter ocorrido um erro, para tal, é revisto se a *tag* está de frente para o AGV, caso não esteja existe um *loop* que neste momento irá voltar a corrigir a posição do AGV e se necessário volta a executar o processo desde a etapa número 1. Caso esteja tudo correto e o ângulo β seja igual ou esteja perto de 0º o AGV desloca-se em direcção a *tag* de entrada de forma rectilínea, corrigindo a toda constantemente o trajecto, ou seja, caso a *rack* sofra alterações na posição o AGV corrige a sua pose para efetuar corretamente o acoplamento da *rack*.

5. Para finalizar, visto que a câmara encontra-se montada na parte da frente do AGV, esta deixa de ver a *tag*. Nesta etapa prevê-se que o AGV encontre-se corretamente alinhado de modo a entrar por baixo da *rack*. Este processo é feito acionando os motores para andarem certa distância em frente até o AGV estar por baixo da *rack*.

Conclui-se que, no final desta etapa, o AGV encontre-se por baixo da *rack* (figura 3.26), o que permitirá, proceder às próximas tarefas que não fazem parte do âmbito desta dissertação.

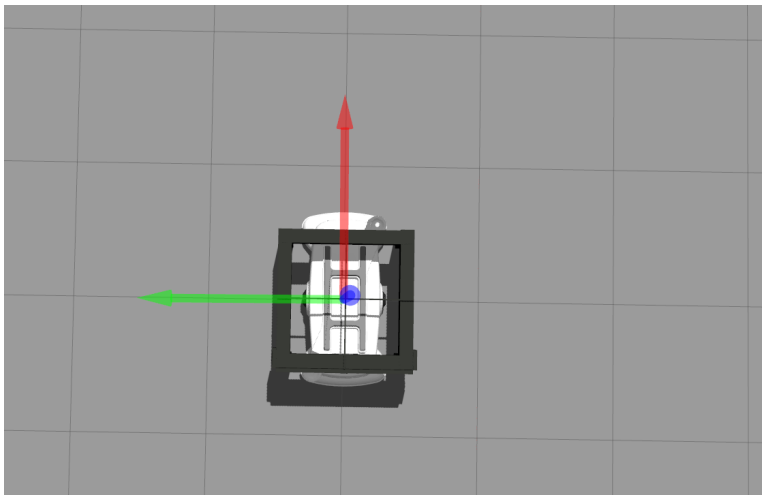

Figura 3.26: Estado final do posicionamento do AGV por baixo da *rack*

Capítulo 4

Testes e Análise de resultados

Neste capitulo serão apresentados as parametrizações para os testes efetuados, procedimento para as simulações, bem como a análise dos resultados obtidos nas diferentes simulações

Com o objetivo de se obter uma validação do sistema implementado nesta dissertação, foram efetuadas diversas simulações. Estas permitiram verificar o comportamento do AGV em diferentes poses iniciais, fazendo variar a luminosidade e dimensão das *tags* e avaliar os resultados obtidos, bem como, as situações em que o AGV desempenha ou não as tarefas pretendidas.

Atendendo à situação atual (contingências limitantes decorrentes da pandemia), não foi possível testar o algoritmo implementado no protótipo AGV, nas instalações físicas da Introsys. Com o objetivo de ultrapassar esta limitação, os testes foram efetuados no simulador Gazebo, o qual simula o ambiente ("world") num sistema virtual. Decorrem possíveis erros que estão associados ao ambiente físico/real, contudo o simulador Gazebo também simula alguns desses erros, como por exemplo colisões e erros associados ao *delay* dos motores.

Contudo numa fase de protótipo e desenvolvimento, os resultados obtidos através do simulador são bastante relevantes para o progresso do desenvolvimento do AGV.

O objectivo principal da validação do sistema, neste caso, foi fazer um estudo relativamente aos diferentes tipos de parâmetros a ter em conta quando se pretende instalar um AGV na intra-logística e ao impacto desses no comportamento do sistema proposto.

4.1 Parametrização do sistema

Pretende-se validar o sistema proposto avaliando o mesmo de uma forma quantitativa e qualitativa, fazendo variar os parâmetros de:

- Pose do AGV;
- Luminosidade do ambiente;
- Dimensão das *tags* montadas na *rack*;

Variando estes 3 parâmetros é possível obter-se diferentes resultados e verificar o comportamento dos mesmo consoante a sua variação.

Quanto às poses do AGV, foram selecionadas diferentes posições do AGV em relação à *rack*, para se avaliar as distâncias obtidas pela câmara comparativamente ao previsto. Nas simulações iniciais, as poses não foram escolhidas ao acaso, a área da variação das poses do AGV foi escolhida em função dos requisitos, ou seja, sabe-se que o AGV irá trabalhar a uma certa distância da *rack* (entre os 0.5 e os 4.5 metros). Assim, foram escolhidas as poses próximas da área onde o AGV irá trabalhar, para que se possa analisar o comportamento do algoritmo implementado, fazendo variar os 3 parâmetros, luminosidade, dimensão das *tags* e pose do AGV. As poses escolhidas como amostra de estudo são as representadas na figura 4.1.

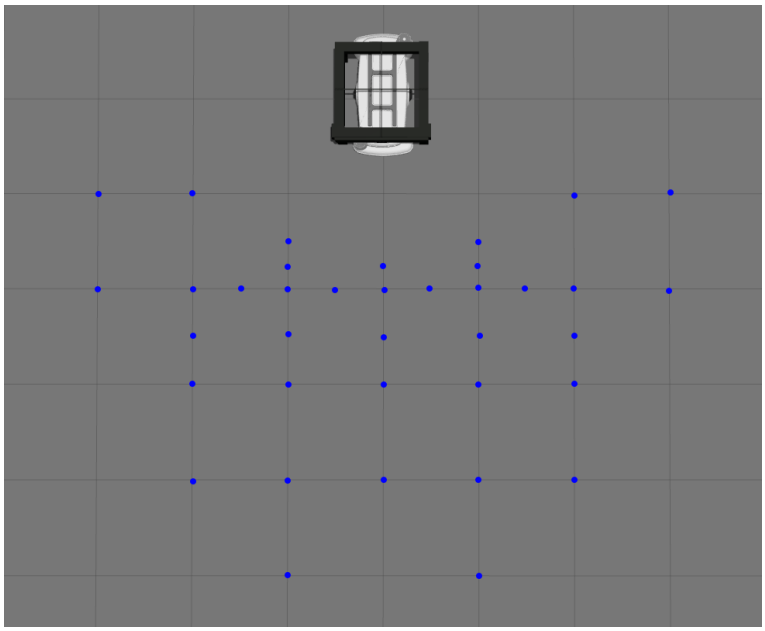

Figura 4.1: Poses de amostragem AGV

Cada ponto representa as diferentes poses em que o AGV foi distribuído para efeitos de teste, de modo a estudar comportamento do AGV consoante as variações já referidas.

Quanto à luminosidade foram selecionados 4 tipos diferentes de luminosidade respetivamente:

- Luminosidade L1 - luminosidade baixa (um ponto de luz)
- Luminosidade L2 - luminosidade maior (três pontos de luz)
- Luminosidade L3 - luminosidade normal (oito pontos de luz)
- Luminosidade L4 - luminosidade excessiva (treze pontos de luz)

Quanto à variação da dimensão das *tags*, foram testadas 3 dimensões:

- 10x10cm
- 7.5x7.5cm
- 5x5cm

Visto que o pacote de *software ar track alvar*, dispõe da funcionalidade de identificar a pose de uma *tag* sem o recurso a uma imagem de profundidade (sem nuvem de pontos associada à imagem), o sistema proposto foi também testado sem este tipo de recurso. Neste sentido, comparou-se os resultados obtidos com os resultados registados em situações ideais (luminosidade e dimensão das *tags*) com recurso a imagem de profundidade.

Para finalizar a fase de testes, foram ainda executados dois testes de validação do sistema, que pretenderam avalia-lo qualitativamente. Um teste em que foram geradas posições aleatórias do AGV (situações em que o AGV pode estar em posições menos frequentes ou de pouca probabilidade). Com este teste foi possível avaliar também a variação do tempo em função da distância da *rack*. Num segundo teste, existindo vários ID's de identificação disponíveis, pertencentes a diferentes *racks*, o AGV teve que fazer o acoplamento à *rack* de acordo com o ID aleatórias gerado.

4.2 Procedimento

Para testar todos estes parâmetros foi necessário recorrer a um computador capaz de suportar todo o processamento dos dados. Para tal foi utilizado um portátil ASUS de processador Intel® Core™ i7-6700HQ CPU @ 2.60GHz × 8, com placa gráfica Nvidia GeForce GTX 950M/PCIe/SSE2 nele instalada, a correr o sistema operativo Linux Ubuntu 18.04.4 LTS e com uma memória RAM de 16GB.

De forma a automatizar as simulações pretendidas foi criado um *script* que fizesse os lançamentos dos comandos na linha de comandos para efetuar cada simulação, enquanto outro *script* estava a correr em simultâneo para terminar os comandos lançados com um intervalo de 90 segundos. Foram escolhidos os 90 segundos pois numa fase inicial de testes, verificou-se que o sistema demorava menos de 90 segundos a fazer o acoplamento. Para gerar as linhas de comando com diferentes parâmetros (Luminosidade, dimensão das *tags* e pose do AGV) elaborou-se, em *LibreOffice Calc*, uma pequena macro que através da variação dos parâmetros introduzidos gera um comando. Optou-se por uma abordagem de forma automatizada, já que demoraria significativamente mais tempo a efetuar os testes na totalidade, correndo-se o risco de haver erros associados à introdução e ao lançamento dos parâmetros, caso se optasse por fazer as simulações de forma manual. Por fim, os dados gerados, em relação às simulações efetuadas, foram guardados automaticamente num ficheiro (.txt) para, posteriormente poderem ser analisados.

4.3 Testes e análise de resultados

4.3.1 Variação de luminosidade

Como já referido, a luminosidade do ambiente onde se pretende instalar o sistema foi testada fazendo variar a luminosidade em 4 diferentes tipos de intensidade (L1, L2, L3, L4) tendo como amostra de avaliação as 40 posições diferentes, selecionadas. A luminosidade L1 consiste em ter apenas um ponto de luz de intensidade normal na *tag* de entrada do AGV e 1 ponto em cada *tag* lateral (figura 4.2), L2 corresponde a ter 3 pontos de luz de intensidade normal figura na *tag* de entrada do AGV e 2 pontos em cada *tag* lateral (figura 4.3), L3 corresponde a ter uma luminosidade considerada teoricamente ideal, correspondendo a 8 pontos de luz incidentes na *tag* de entrada e 6 pontos nas em cada *tag* lateral (figura 4.4), por fim, L4 correspondente a um excesso de luminosidade com 13 pontos de luz incidentes na *tag* de entrada do AGV e 9 pontos com maior intensidade em cada *tag* lateral (figura 4.5).

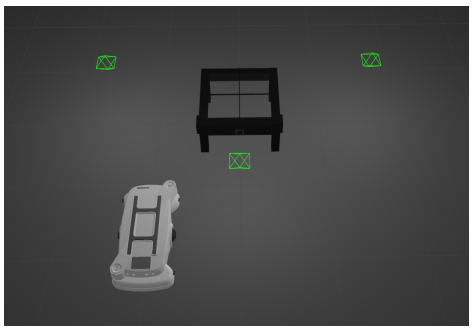

Figura 4.2: Luminosidade L1

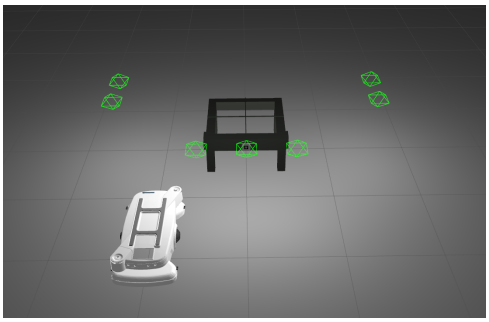

Figura 4.3: Luminosidade L2

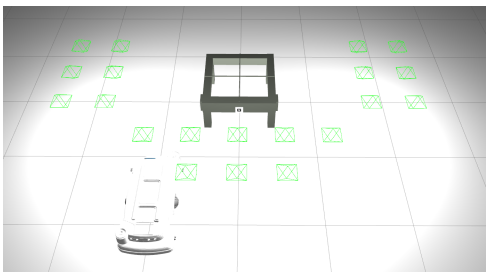

Figura 4.4: Luminosidade L3

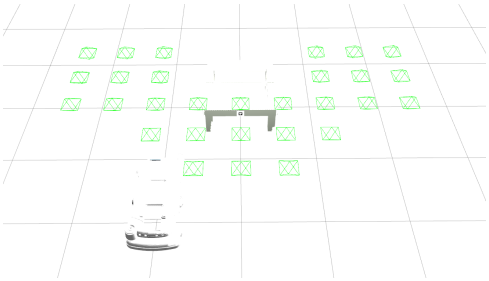

Figura 4.5: Luminosidade L4

Começou-se por variar a luminosidade para uma dimensão das *tags*, de 10x10cm tendo sido feitos 40 x 4 simulações para o teste da luminosidade num total de 160 simulações para o estudo da variação da luminosidade. Seguidamente executou-se o *script* programado para o lançamento das simulações que demorou cerca de 4 horas para obter os resultados da variação da luminosidade. Os dados referentes aos resultados das simulações foram gravados automaticamente num ficheiro

(.txt) para poderem ser avaliados posteriormente.

Os resultados obtidos para uma luminosidade de L1, L2, L3 e L4 com uma dimensão da *tag* de 10x10cm podem ser encontrados nos anexos I, respetivamente.

Na tabela 4.1 estão representados as falhas relativas ao testes obtidos pela análise dos resultados.

Tabela 4.1: Número falhas para diferentes luminosidades

	10x10cm
L1	28 falhas
L2	11 falhas
L3	0 falhas
L4	8 falhas

Ao analisar detalhadamente os casos de falha para uma luminosidade L1 ocorrem 28 falhas num total de 40 simulações (anexo I.1), obtendo assim uma taxa de sucesso de 30%. O número de falhas que ocorrem, deve-se à baixa luminosidade (praticamente nula) em que torna difícil o AGV detetar uma *tag*, pois a câmara não extrai uma imagem nítida e clara, neste caso o AGV é colocado numa situação extrema que não se prevê acontecer no ambiente em que o AGV se pretende instalar. Contudo existem casos de sucesso em que o AGV consegue desempenhar a tarefa de acoplamento como por exemplo, as simulações finais em que o AGV se encontra numa posição mais próxima das *tags*.

Quanto aos casos de falha apresentados no anexo I.2, para uma luminosidade de L2, detetaram-se 11 casos de falha, optendo-se uma taxa de sucesso de 72,5%. Sendo este um melhor resultado comparado com a luminosidade L1. Como já verificado no teste da luminosidade L1, quando o AGV se encontra a uma maior distância da *tag* que identifica a *rack*, o número de casos de falha é maior, este deve-se ao sistema não conseguir identificar a *tag* devido á pouca luminosidade, contudo ocorrem situações como a da simulação numero 32.1 em que o AGV deteta a *tag* mas a pose da mesma não é a correta, fazendo com que o AGV não desempenhe a tarefa de acoplamento da forma correta.

Para uma luminosidade de L3, verificou-se uma taxa de sucesso de 100%. O sistema conseguiu identificar a *tag* ou as *tags* e efetuar o acoplamento da *rack* em todas as 40 posições simuladas (anexo I.3).

O sistema foi ainda testado para uma luminosidade L4, onde se obtiveram 8 falhas (anexo I.4) correspondentes a uma taxa de sucesso de 80%, ou seja, o sistema também fica comprometido quando submetido a situações de excesso luminosidade. Tal como nos outros testes, existe maior número de falhas quando o AGV se encontra a maiores distâncias da *rack*, contudo, nos casos de falha de

número de simulação 4.1, 16.1, 31.1 e 40.1, o AGV conseguiu identificar a *tag* mas a pose não foi extraída corretamente, consequentemente não executa a tarefa de acoplamento corretamente (anexo I.4).

Não se considerou proeminente efetuar mais testes para diferentes tipos de luminosidade, pois foi possível retirar resultados conclusivos em relação à diminuição da luminosidade, bem como ao excesso da luminosidade também afeta o sistema e não seria uma hipótese a considerar, pois o aumento da luminosidade implica custos relativos ao consumo de energia, e por norma não seria uma boa solução a implementar.

Em suma, com uma dimensão da *tag* de 10x10cm para uma luminosidade de L1 a taxa de sucesso é de 30%, para uma luminosidade de L2 a taxa de sucesso é de 72,5%, para L3 a taxa de sucesso é de 100% e por fim, L4 apresentou uma taxa de sucesso de 80%. Os casos de falha em função da luminosidade podem ser melhor compreendidos através do gráfico na figura 4.6. Com estes resultados é possível concluir que a variação da luminosidade afeta significativamente o desempenho do sistema de acoplamento, sendo que a luminosidade L3 apresentou os melhores resultados.

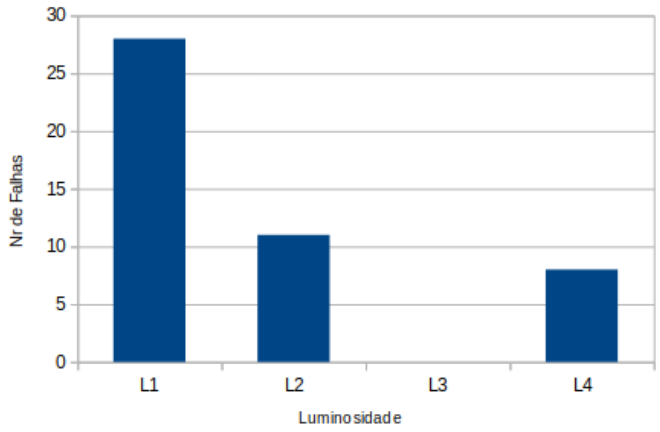

Figura 4.6: Número de falhas em função da luminosidade

4.3.2 Variação da dimensão das *tags*

Após se ter concluído que a luminosidade ideal, consoante os testes efetuados e no conjunto de testes selecionados, é de Luminosidade L3 (8 pontos de luz), foram efetuados os testes relativos à dimensão das *tags* com esta luminosidade (L3). Neste conjunto de testes pretende-se analisar o comportamento do AGV ao variar a dimensão das *tags*. Como referido anteriormente, utilizou-se três dimensões diferentes das *tags* (10x10cm, 7.5x7.5cm e 5x5cm), respetivamente representadas nas figuras 4.7, 4.8 e 4.9 para uma luminosidade de L3.

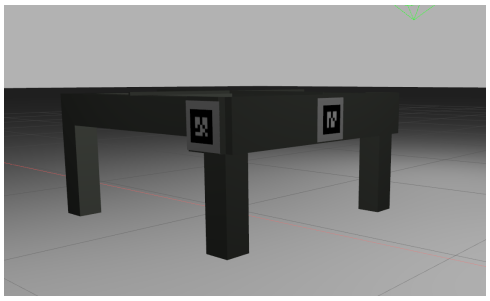

Figura 4.7: *Tag* 10x10cm

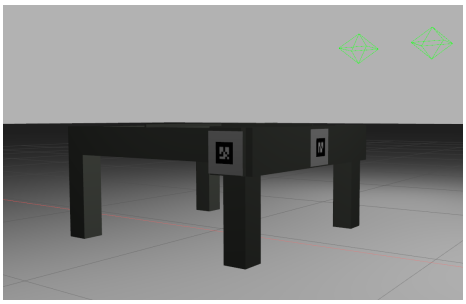

Figura 4.8: *Tag* 7.5x7.5cm

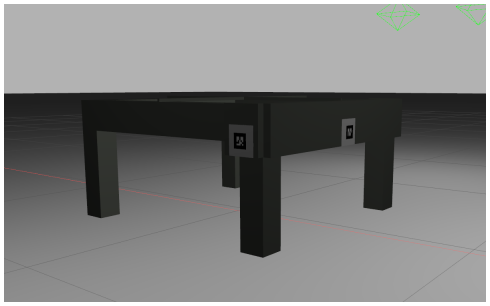

Figura 4.9: *Tag* 5x5cm

Neste conjunto de testes foram efetuadas duas 80 simulações (40x2) para verificar o comportamento do sistema quando submetido a variação da dimensão das *tags* instaladas na *rack*. De notar que os testes para uma dimensão de 10x10cm já foram simulados na subsecção anterior, tendo sido feitas nesta bancada de testes as simulações para 7.5x7.5cm e 5x5cm. Os resultados obtidos estão anexados em II, de onde foram extraídos os casos de falha para a tabela 4.2.

Tabela 4.2: Número falhas para diferentes dimensões de *tags*

	10x10cm	**7.5x7.5cm**	**5x5cm**
L3	0 falhas	10 falhas	16 falhas

Os resultados obtidos para uma dimensão de 10x10cm resultaram numa taxa de sucesso de 100%, como analisados na subsecção anterior, sendo este um bom resultado.

Ao analisar os resultados obtidos para uma dimensão de *tag* de 7.5x7.5cm anexados em III.1, verificou-se que ocorreram 10 casos de falha num total de 40 simulações, obtendo uma taxa de sucesso 75%. Através de uma análise mais detalhada dos resultados correspondentes a cada simulação verificou-se que quando a distância do AGV é maior, ou seja, nas primeiras simulações, o sistema não deteta a *tag* que identifica a *rack*, e com isto não consegue proceder ao processo de acoplamento como esperado. Nos casos de falha relativos ás simulações 4.1, 9.1, 13.1, 16.1 e 32.1 verificou-se através de uma análise mais detalhada, que o AGV conseguiu detetar a *tag* que identifica a *rack*, contudo não detetou a pose da mesma de uma forma correta, com isto não conseguiu proceder ao processo de acoplamento.

Quanto aos resultados gerados na simulação para uma *tag* de dimensão 5x5cm anexados em III.1 foram obtidos 16 casos de falha num total de 40 simulações, correspondendo a uma taxa de sucesso de 60%. Desses 16 casos de falha, tal como como nas simulações de 7.5x7.5cm verificou-se que nas primeira simulações (1.1 até 9.1, 12.1, 13.1) em que a distância entre o AGV e *rack* é maior, o sistema não detetou as *tags* correspondentes à *rack*, e com isto não foi possível fazer o acoplamento do AGV à *rack*. Nas simulações 9.1, 14.1 e 32.1, o sistema detetou a *tag* mas não conseguiu desempenhar a tarefa de acoplamento corretamente devido à dimensão da *tag* não ser a ideal.

Não se considerou relevante efetuar mais simulações para dimensões diferentes, pois foram obtidos resultados satisfatórios para o dimensionamento da *tag* e da luminosidade do ambiente em que o AGV se propõe instalar.

Podemos concluir, que a variação da dimensão das *tags* tem uma relevância acrescida no processo de acoplamento do AGV à *rack*, verificando-se que a diminuição das *tags* está associada a um pior desempenho do sistema de acoplamento (grafico 4.10).

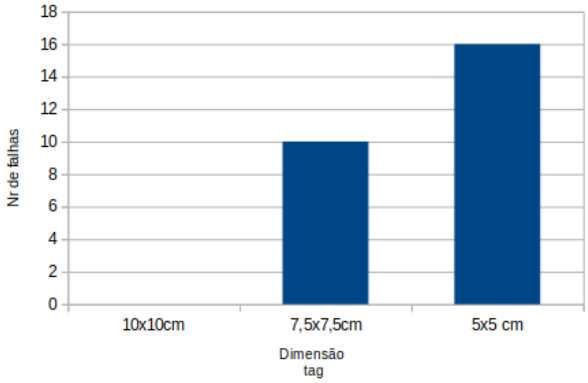

Figura 4.10: Número de falhas em função da dimensão das *tags*

4.3.3 Comparação entre Imagem e *depth image*

A eficiência do sistema, está também dependente da componente de *depth image* (nuvem de pontos correspondente à imagem), que é introduzida no *software ar track alvar*, quando é necessário localizar a *tag*. Pretende-se que esta componente do sistema, reduza a margem de erro, na extração de medidas que calculam a pose da *tag*. No entanto, nem sempre é possível fazer recurso de câmaras com *depth image*, pelo que se entendeu necessário comparar resultados provenientes da utilização de *depth image* e de câmara sem esta capacidade.

Atendendo aos testes efetuados e anteriormente descritos, concluiu-se que a dimensão da *tag* com maior taxa de sucesso é a de 10x10cm, com uma luminosidade do ambiente de L3 (8 pontos de luz), fatores estes utilizados no conjunto de testes a descrever.

Os resultados dos testes correspondentes às simulações do sistema de acoplamento, sem recurso a *depth image*, podem ser encontrados no anexo III.1. Na tabela anexada em III.2 estão apresentados os erros associados a estes testes.

Ao analisar a tabela de erros gerada através dos resultados, é possível verificar que a média de erro na posição final do AGV, depois do processo final de acoplamento em X, é de 5.2 centímetros sem recurso a *depth image* e 1 centimetro com recurso a *depth image*, em Y é de 1.7 centímetros sem recurso a *depth image* e 1.3 centímetros com recurso a *depth image*.

Pode-se concluir que a pose final do AGV com recurso a *depth image*, revela um erro aproximado de 1.3 centímetros em relação à posição final prevista para o acoplamento, enquanto que sem recurso a *depth image* existe um erro de aproximadamente 5.2 centímetros, o qual significa que o AGV se encontra 5.2 centímetros fora da posição prevista, embora tenha conseguido fazer o acoplamento. Relativamente ao erro associado às distâncias extraídas quando o AGV deteta uma *tag*, verificou-se um erro com recurso a *depth image* de cerca de 0.8 centímetros e 4.5 centímetros sem recurso a *depth image*, em relação à distância prevista. Nos testes efetuados (40 amostras) (anexo III.1), sem recurso a *depth image*, ocorreram 2 falhas, o que revela uma taxa de sucesso de 95%, enquanto a taxa de sucesso com recurso a *depth image* foi de 100%. Quanto às falhas sem recurso a *depth image*, ocorridas nas simulações número 35.1 e 36.1, pela análise detalhada dos resultados, é possível verificar que o AGV deteta a *tag*, contudo, como se encontra relativamente muito próximo da *rack*, não tem espaço para fazer movimentos e não consegue executar o acoplamento corretamente. Nas simulações 31.1, 32.1, 37.1, 38.1, 39.1 e 40.1, o AGV efetuou a tarefa de acoplamento, mas revela um

erro associado relativamente alto, o que pode comprometer a operação de acoplamento.

Verificou-se que com este tipo de imagem (sem mapa de profundidade), também é possível determinar a pose do AGV em relação à *rack*.

4.3.4 Posições aleatórias

Para completar e consolidar os resultados obtidos quanto à luminosidade e dimensão das *tags* foi ainda feito um teste em que foram geradas, 50 posições diferentes do AGV, aleatórias. Em 30 dessas posições, o AGV encontra-se numa área de 3x6m em frente à *rack*. Em 10 posições, numa área de 2x2m do lado esquerdo da *rack* e outras 10 numa área de 2x2m do lado direito da *rack*. Essas áreas estão representadas a azul na figura 4.11. Pretende-se com este teste colocar o AGV em situações não esperadas, ou seja, em posições aleatórias. As condições de luminosidade parametrizadas são as correspondentes a L3 e com *tags* de dimensão 10x10cm.

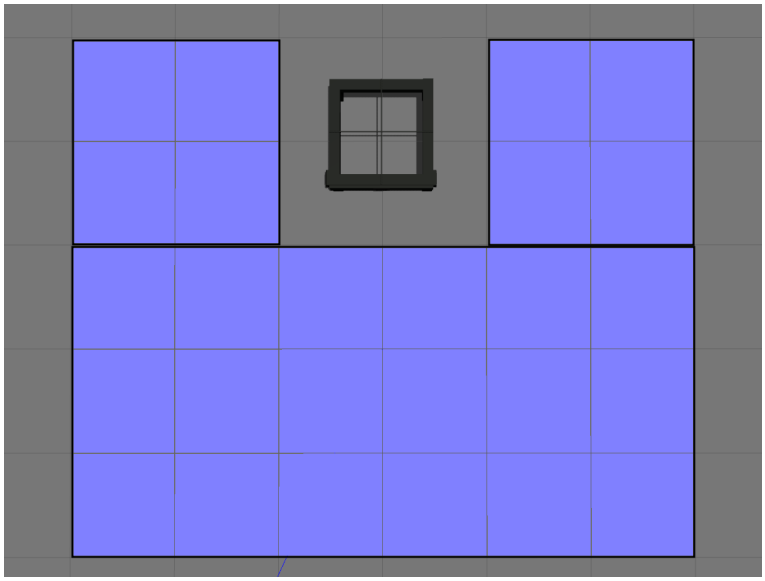

Figura 4.11: Áreas da poses aleatórias do AGV

Os resultados das simulações efetuadas podem ser encontrados no anexo III.3. Através dos resultados, é possível verificar que ocorreram 7 casos de falha no acoplamento num total de 50 simulações efetuadas, ou seja, uma taxa de sucesso de 86%.

De forma a analisar os casos de falha ocorridos de uma forma detalhada, verificou-se que: nas primeiras 30 simulações (situação em que o AGV se encontra em frente da face de entrada) o sistema falhou 4 vezes, das quais 3 estão associadas à distancia a que o AGV se encontra da *rack*, não tendo sido detetada nenhuma *tag*; uma falha, na qual foi possível detetar uma *tag* correspondente, contudo não foi possível determinar corretamente a pose da *tag*, o que resultou num mal acoplamento; nas seguintes simulações (30.1 a 40.1) em que o AGV se encontra do lado esquerdo da *rack*, não ocorreram falhas e o acoplamento foi feito de uma forma correta; nas simulações finais (40.1 a 50.1) em que o AGV se encontra do lado direito da *rack* ocorreram duas falhas, uma (46.1) devido ao AGV ter

identificado a *tag* principal enquanto deveria ter detetado a lateral, consequentemente, não fez o contorno da *rack*, de maneira a fazer o acoplamento em vez disso, foi diretamente para a face de entrada e não teve capacidade de fazer a manobra, resultando num caso de insucesso. O ultimo caso de falha (49.1) ocorreu devido ao AGV se encontrar muito próximo da *rack*. Com a câmara montada na frente do AGV, a *tag* não se encontra no campo de visão da câmara, o que não permitiu a deteção da *tag*.

Através dos resultados obtidos, apresentados no anexo III.3 foi também possível gerar um gráfico 4.12 (tempo em função da distância) para uma melhor compreensão que, quanto maior for a distância a que o AGV se encontra da *rack* maior será o tempo do processo de acoplamento.

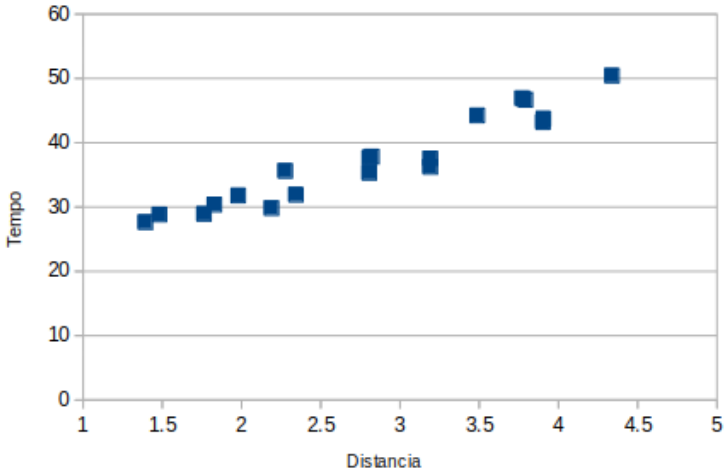

Figura 4.12: Gráfico do tempo em função da distância

Os dados inquiridos para o gráfico são relativos a uma deteção da *tag* correspondente à face de entrada (ID 2), de forma a analisar o pressuposto de uma forma correta. Caso fossem incluídas as distâncias correspondentes às *tags* laterais, não seria possível analisar corretamente o tempo em função da distância, pois para fazer o acoplamento o sistema executa o contorno da *rack* podendo assim estar mais perto da mesma, aumentando o tempo de acoplamento, o que resulta numa discrepância da distância em relação ao tempo para dados estatísticos.

4.3.5 Várias *racks*

Com o objetivo de analisar o desempenho do AGV quanto ao seu sucesso, simulou-se a situação em que o AGV tenha disponivel diferentes *racks* para o seu acoplamento, ou seja, *racks* com diferentes IDs. Estes ID foram gerados aleatoriamente para que se possa verificar se o AGV faz o acoplamento de uma forma correta ou incorreta.

Quanto à luminosidade e dimensão da *tag* para este teste, utilizou-se, a luminosidade L3 e as dimensões de 10x10cm, pois pelo testes já efetuados, verificaram-se ser o parâmetros com maior sucesso e com melhores resultados quando testados quantitativamente.

Neste conjunto de testes as *racks* encontram-se em posições ilustradas pela figura 4.13. Pretende-se que o AGV faça o acoplamento de uma forma autónoma da *rack* selecionada.

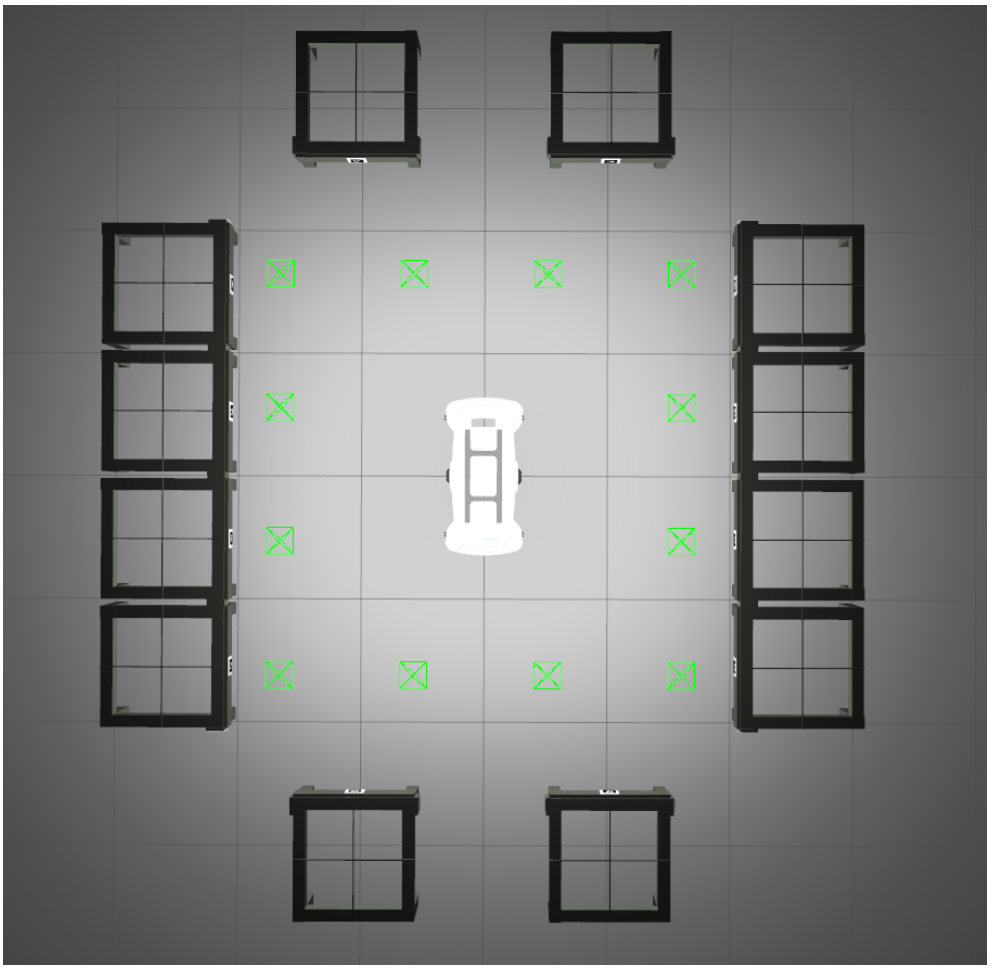

Figura 4.13: Exemplo de uma simulação efetuada em que o AGV tem disponível várias *racks* para fazer o acoplamento

Foram efetuadas 30 simulações para *rack* de ID's diferentes, os dados foram guardados num ficheiro .txt, tal como nas outras simulações efetuadas anteriormente. Os resultados podem ser encontrados no anexo III.4. Como esperado, nas 30 simulações efetuadas, não ocorreram falhas. Com estes resultados foi possível avaliar a capacidade que o sistema tem para fazer o acoplamento a uma *rack* previamente selcionada. Visto que as condições de luminosidade e dimensão da *tag* predefinidas foram as com melhores resultados nas simulações anteriores, quanto à eficiência do sistema verificou-se uma taxa de sucesso de 100% neste conjunto de testes, consolidando assim os resultados obtidos nas simulações anteriores.

Capítulo 5

Conclusão

No âmbito das testes e análises efetuadas, face à necessidade de desenvolver um sistema capaz de fazer o acoplamento de um AGV a uma plataforma de transporte (*rack*), é possível inferir que o sistema proposto de identificação e navegação, contribui positivamente para a implementação do AGV, que está a ser desenvolvido pela Introsys S.A. Considerando as diferentes condições de operação do AGV, foi possível verificar que o comportamento do sistema está dependente da variação de luminosidade e dimensão das *tags*, bem como, das situações de pose do AGV em relação à plataforma de transporte.

Mais especificamente, com base nas simulações efetuadas concluiu-se, que a dimensão adequada, para um correto funcionamento que do AGV entre uma distância de 0.5 metros e 4.50 metros da *rack*, é de 10x10cm, com uma luminosidade de L3 (8 pontos de luz incidentes na *tag* de entrada e 6 pontos nas em cada *tag* lateral). Pode ainda concluir-se que, quando o sistema é submetido a situações de variação de luminosidade, nomeadamente uma luminosidade reduzida, a sua taxa de sucesso é menor, assim como, quando submetido a situações de luminosidade extrema. Foi obtida uma taxa de sucesso, de 100% para uma luminosidade considerada normal (L3) no respetivo conjunto de testes utilizados.

Quanto à dimensão das *tags*, foi possível concluir que a dimensão das mesmas a utilizar numa plataforma de transporte, com o objetivo de obter os melhores resultado, foi a de 10x10cm. Contudo, o sistema também demonstrou ser eficaz quando a dimensão das *tags* é diminuída (7.5x7.5cm ou 5x5cm), limitando assim a distância a que o sistema deteta concretamente, as tags.

A partir dos resultados obtidos na comparação feita entre imagem e *depth*

image, conclui-se que, quando não existe recurso a *depth image*, alcança-se uma taxa de sucesso equivalente a 95%, comparativamente a uma situação de 100%, quando ocorre recurso à *depth image*. De referir, que quando sem recurso a *depth image*, os resultados revelaram um erro na pose final do AGV, em média de 4.5cm no acoplamento. No entanto, pode-se dizer que o sistema de navegação sem recurso a *depth image*, tem uma boa taxa de sucesso (95%), o que permite ser implementado em situações em que o erro da pose final do AGV possa ser inferior a 5cm, e quando não exista a possibilidade de montar uma câmara capaz de extrair imagem com mapa de profundidade associado (*depth image*).

Os resultados para poses aleatórias efetuados, provaram que o sistema é capaz de desempenhar a tarefa de acoplamento nas condições ideais de luminosidade (L3) e com uma dimensão de *tag* de 10x10cm, mesmo quando sujeito a algumas condicionantes, como por exemplo, os casos em que o AGV se encontra a uma distancia superior a 4.5 metros da *rack*, bem como, as limitações em que a proximidade excessiva da plataforma de transporte afeta o campo de visão do AGV, por consequência o sistema não deteta uma *tag* e não desempenha com sucesso a tarefa de acoplamento.

Ainda neste contexto, foi possível validar o sistema perante a existência de diferentes *racks*, com ID diferentes à disposição, isto é, quando o AGV tem à sua disposição diferentes *racks*, verificou-se que este é capaz de identificar a *rack* selecionada e proceder ao acoplamento.

A demonstração de resultados, vem comprovar que o recurso às *tags* para identificação e acoplamento da plataforma de transporte, é um sistema que apresenta uma excelente taxa de sucesso (100%), desde que, estejam reunidas as condições previamente determinadas, ou seja, a luminosidade, distância e a dimensão das *tags* na plataforma de transporte.

Como tal, pode afirmar-se que a solução proposta para o problema de fazer a identificação e acoplamento de um AGV a uma *rack*, apresentado ao longo desta dissertação é um sistema passível de implementar num AGV ou num robô que se equipare ao AGV.

Para trabalho futuro, entende-se necessário a realização de testes no AGV "real"(protótipo), de modo a consolidar os resultados obtidos em simulador, condição que como previamente referido, não foi possível cumprir. Perspetiva-se ainda, criar uma base de dados, que possua uma interface capaz de associar e guardar características referentes ao ID das *tags*, justificando deste modo, o recurso a *tags* para identificar diferentes *racks*, o que facilitará a utilização do sistema por parte dos utilizadores.

Conclui-se que este sistema – Sistema de Identificação e Acoplamento de um

AGV a uma Plataforma de Transporte, facultará uma flexibilização na organização do espaço e do tempo para operação, sendo que, o processo de transporte na intra-logística beneficiará de um fluxo, em que as atividades serão mais rápidas e eficientes, reduzindo, a longo prazo, custos operacionais e aumentando a eficiência da indústria e distribuição.

Bibliografia

[1] B. Y. Qi, Q. L. Yang e Y. Y. Zhou. "Application of AGV in intelligent logistics system". Em: *IET Conf. Publ.* 2015.CP676 (2015). DOI: 10.1049/cp.2015.1527.

[2] IntrosysSA. *INTROSYS – Global Control System Designers*. URL: https://www.introsys.eu/.

[3] G. Ullrich. "The History of Automated Guided Vehicle Systems". Em: *Autom. Guid. Veh. Syst. A Prim. with Pract. Appl.* (215). DOI: 10.1007/978-3-662-44814-4.

[4] *VDI*. URL: https://www.vdi.de/fts.

[5] G. Garibotto, S. Masciangelo, M. Ilic e P. Bassino. "ROBOLIFT: a vision guided autonomous fork-lift for pallet handling". Em: *IEEE Int. Conf. Intell. Robot. Syst.* 2 (1996), pp. 656–663. DOI: 10.1109/iros.1996.571028.

[6] E. Freund e F. Dierks. "Laser Scanner Based Free Navigation of Autonomous Vehicles". Em: *IFAC Proc. Vol.* 26.1 (1993), pp. 229–234. ISSN: 14746670. DOI: 10.1016/s1474-6670(17)49304-9.

[7] G. Garibotto, M. Ilic e S. Masciangelo. "Vision-based navigation in service robotics". Em: *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)* 974 (1995), pp. 313–318. ISSN: 16113349. DOI: 10.1007/3-540-60298-4_276.

[8] *Automated guided vehicle*. URL: https://pt.slideshare.net/alamkhan111/automated-guided-vehicle-71159283.

[9] *BMW Group introduces self-driving robots in Supply Logistics*. URL: https://www.press.bmwgroup.com/global/article/detail/T0257786EN/bmw-group-introduces-self-driving-robots-in-supply-logistics?language=en.

[10] H. A. Martins, J. R. Birk e R. B. Kelley. "Camera models based on data from two calibration planes". Em: *Comput. Graph. Image Process.* 17.2 (1981), pp. 173–180. ISSN: 0146664X. DOI: 10.1016/0146-664X(81)90024-1.

[11] M. Seelinger e J. D. Yoder. "Automatic pallet engagment by a vision guided forklift". Em: *Proc. - IEEE Int. Conf. Robot. Autom.* 2005.April (2005), pp. 4068–4073. ISSN: 10504729. DOI: 10.1109/ROBOT.2005.1570744.

[12] J. Pagès, X. Armangué, J. Salvi, J. Freixenet e J. Martí. "A Computer Vision System for Autonomous Forklift Vehicles in Industrial Environments 1 ." Em: *Proc. 9th Mediterr. Conf. Control Autom. MEDS* (2001), pp. 1–6.

[13] R. Y. Tsai. *Calibration - Tsai.pdf*.

[14] É. D. O. Nunes e A. Conci. "Segmentação por textura e localização do contorno de regiões em imagens multibandas". Em: *IEEE Lat. Am. Trans.* 5.3 (2007), pp. 185–192. ISSN: 15480992. DOI: 10.1109/TLA.2007.4378503.

[15] G. Z. Cui, L. S. Lu, Z. D. He, L. N. Yao, C. X. Yang, B. Y. Huang e Z. H. Hu. "A robust autonomous mobile forklift pallet recognition". Em: *CAR 2010 - 2010 2nd Int. Asia Conf. Informatics Control. Autom. Robot.* 3 (2010), pp. 286–290. DOI: 10.1109/CAR.2010.5456688.

[16] R. Cucchiara, M. Piccardi e A. Prati. "Focus based Feature Extraction for Pallets Recognition". Em: January 2014 (2013), pp. 70.1–70.10. DOI: 10.5244/c.14.70.

[17] P. Maragos. "Morphological Filtering for Image". Em: *Processing* 2 (1999), pp. 1–26.

[18] C. C. Yang, S. O. Prasher, P. Enright, C. Madramootoo, M. Burgess, P. K. Goel e I. Callum. "Application of decision tree technology for image classification using remote sensing data". Em: *Agric. Syst.* 76.3 (jun. de 2003), pp. 1101–1117. ISSN: 0308521X. DOI: 10.1016/S0308-521X(02)00051-3.

[19] R. Varga e S. Nedevschi. "Robust Pallet Detection for Automated Logistics Operations". Em: 4.Visigrapp (2016), pp. 470–477. DOI: 10.5220/0005674704700477.

[20] R. E. Schapire. *The Strength of Weak Learnability*. Rel. téc. 1990, pp. 197–227.

[21] S. Maji, A. C. Berg e J. Maliks. "Classification using intersection kernel support vector machines is efficient". Em: *26th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR* (2008). DOI: 10.1109/CVPR.2008.4587630.

[22] I. S. Mohamed, A. Capitanelli, F. Mastrogiovanni, S. Rovetta e R. Zaccaria. "Detection, localisation and tracking of pallets using machine learning techniques and 2D range data". Em: *Neural Comput. Appl.* May 2019 (2019). ISSN: 14333058. DOI: 10.1007/s00521-019-04352-0. arXiv: 1803.11254.

[23] Y. Lecun, Y. Bengio e G. Hinton. "Deep learning". Em: *Nature* 521.7553 (2015), pp. 436–444. ISSN: 14764687. DOI: 10.1038/nature14539.

[24] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto e U. J. Nunes. "DepthCN Vehicle detection using 3D LIDAR and ConvNet". Em: (2017).

[25] R. Varga e S. Nedevschi. "Vision-based autonomous load handling for automated guided vehicles". Em: *Proc. - 2014 IEEE 10th Int. Conf. Intell. Comput. Commun. Process. ICCP 2014* (2014), pp. 239–244. DOI: 10.1109/ICCP.2014.6937003.

[26] L. Baglivo, N. Biasi, F. Biral, N. Bellomo, E. Bertolazzi, M. Da Lio e M. De Cecco. "Autonomous pallet localization and picking for industrial forklifts: A robust range and look method". Em: *Meas. Sci. Technol.* 22.8 (2011). ISSN: 13616501. DOI: 10.1088/0957-0233/22/8/085502.

[27] R. Bostelman, T. Hong e T. Chang. "Visualization of pallets". Em: *Intell. Robot. Comput. Vis. XXIV Algorithms, Tech. Act. Vis.* 6384 (2006), p. 638408. ISSN: 0277786X. DOI: 10.1117/12.684677.

[28] J. Ji, G. Chen e L. Sun. "A novel Hough transform method for line detection by enhancing accumulator array". Em: *Pattern Recognit. Lett.* 32.11 (ago. de 2011), pp. 1503–1510. ISSN: 01678655. DOI: 10.1016/j.patrec.2011.04.011.

[29] S. Byun e M. Kim. "Real-time positioning and orienting of pallets based on monocular vision". Em: *Proc. - Int. Conf. Tools with Artif. Intell. ICTAI* 2 (2008), pp. 505–508. ISSN: 10823409. DOI: 10.1109/ICTAI.2008.124.

[30] H. B. Zhang, K. Yuan e J. D. Liu. "A fast and robust vision system for autonomous mobile robots". Em: *RISSP 2003*. Vol. 2003-October. Institute of Electrical e Electronics Engineers Inc., 2003, pp. 60–65. DOI: 10.1109/RISSP.2003.1285549.

[31] W. S. Kim, D. Helmick e A. Kelly. "Model-Based Object Pose Refinement for Terrestrial and Space Autonomy Won S. Kim, Daniel Helmick". Em: ().

[32] J. Nygårds, T. Högström e Wernersson. "Docking to pallets with feedback from a sheet-of-light range camera". Em: *IEEE Int. Conf. Intell. Robot. Syst.* 3 (2000), pp. 1853–1859. ISSN: 21530866. DOI: 10.1109/IROS.2000.895241.

[33] J. Geng. "Structured-light 3D surface imaging: a tutorial". Em: *Adv. Opt. Photonics* 3.2 (2011), p. 128. ISSN: 1943-8206. DOI: 10.1364/aop.3.000128.

[34] L. Li. "Time-of-Flight Camera–An Introduction". Em: *Texas Instruments - Tech. White Pap.* January (2014), p. 10.

[35] F. Weichert, D. Bachmann, B. Rudak e D. Fisseler. "Analysis of the accuracy and robustness of the Leap Motion Controller". Em: *Sensors (Switzerland)* 13.5 (2013), pp. 6380–6393. ISSN: 14248220. DOI: 10.3390/s130506380.

[36] F. Weichert, S. Skibinski, J. Stenzel, C. Prasse, A. Kamagaew, B. Rudak e M. Ten Hompel. "Automated detection of euro pallet loads by interpreting PMD camera depth images". Em: *Logist. Res.* 6.2-3 (jun. de 2013), pp. 99–118. ISSN: 1865035X. DOI: 10.1007/s12159-012-0095-8.

[37] C. Prasse, S. Skibinski, F. Weichert, J. Stenzel, H. Müller e M. Ten Hompel. "Concept of automated load detection for de-palletizing using depth images and RFID data". Em: *Proc. - 2011 IEEE Int. Conf. Control Syst. Comput. Eng. ICCSCE 2011* (2011), pp. 249–254. DOI: 10.1109/ICCSCE.2011.6190531.

[38] T. Kahlmann. "Range imaging metrology: investigation, calibration and development". Em: 17392 (2007), p. 143. DOI: 10.3929/ethz-a-005465562. URL: http://en.scientificcommons.org/24859411.

[39] K. Khoshelham e S. O. Elberink. "Accuracy and resolution of kinect depth data for indoor mapping applications". Em: *Sensors* 12.2 (2012), pp. 1437–1454. ISSN: 14248220. DOI: 10.3390/s120201437.

[40] CANONICAL. *Community | Ubuntu*. URL: https://ubuntu.com/community%20https://www.ubuntu.com/community.

[41] Ubuntu Wiki. *RootSudo - Community Help Wiki*. 2019. URL: https://help.ubuntu.com/community/RootSudo.

[42] *ROS.org | About ROS*. URL: https://www.ros.org/about-ros/.

[43] *Key considerations when choosing a robot's operating system | Ubuntu*. URL: https://ubuntu.com/whitepapers/beta/robotics/introduction.

[44] *Why should we learn ROS? - Mastering ROS for Robotics Programming - Second Edition*. URL: https://subscription.packtpub.com/book/hardware%7B%5C_%7Dand%7B%5C_%7Dcreative/9781788478953/1/ch01lvl1sec10/why-should-we-learn-ros.

[45] *Gazebo*. URL: http://gazebosim.org/.

[46] *ROS - Data display with Rviz | Stereolabs*. URL: https://www.stereolabs.com/docs/ros/rviz/.

[47] *Welcome to Python.org*. URL: https://www.python.org/ (acedido em 29/11/2020).

[48] *ABOUT | OpenCV*. URL: https://opencv.org/about/%20http://opencv.org/about.html.

[49] *OpenCV | NVIDIA Developer*. 2017. URL: https://developer.nvidia.com/opencv.

[50] *Eigen*. URL: http://eigen.tuxfamily.org/index.php?title=Main%7B%5C_%7DPage (acedido em 30/11/2020).

[51] R. B. Rusu e S. Cousins. *3D is here: Point Cloud Library (PCL)*. Rel. téc. URL: http://pointclouds.org.

[52] *Câmera 3D Intel® RealSense™ R200 Product Specifications*. URL: https://ark.intel.com/content/www/br/pt/ark/products/92256/intel-realsense-camera-r200.html.

[53] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen e A. Bhowmik. *Intel R RealSense TM Stereoscopic Depth Cameras*. Rel. téc.

[54] *blender.org - Home of the Blender project - Free and Open 3D Creation Software*. URL: https://www.blender.org/.

[55] M. Fiala. "Comparing ARTag and ARToolkit plus fiducial marker systems". Em: *HAVE 2005: IEEE International Workshop on Haptic Audio Visual Environments and their Applications* 2005 (2005), pp. 148–153. DOI: 10.1109/HAVE.2005.1545669.

[56] Scott Niekum. *ar_track_alvar - ROS Wiki*. 2016. URL: http://wiki.ros.org/ar%7B%5C_%7Dtrack%7B%5C_%7Dalvar.

[57] VIT. *Augmented Reality / 3D Tracking*. 2016. URL: http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/desktop/index.html.

[58] P. Benavidez, J. Lambert, A. Jaimes e M. Jamshidi. *LANDING OF AN ARDRONE 2.0 QUADCOPTER ON A MOBILE BASE USING FUZZY LOGIC*. Rel. téc. 2. 2013, pp. 5–25.

[59] *GitHub - ablarry91/ros-tag-tracking: A repository for tracking binary matrices in ROS*. URL: https://github.com/ablarry91/ros-tag-tracking.

ANEXO I

Resultados luminosidade

Descrição dos elementos das tabelas

A descrição dos parâmetros apresentados nas tabelas seguintes são apresentados seguidamente:

- Nr Sim - Número correspondente à simulação efetuada
- Dim - Dimensão da *tag* no decorrer da simulação
- Lum - Luminosidade do ambiente no decorrer da simulação
- ID - ID da *tag* identificada pelo sistema
- X - Posição inicial do AGV em centímetros em X
- Y - Posição inicial do AGV em centímetros em Y
- DISTANCIA - Distância a que foi detetada a *tag*
- DURAÇÃO - Duração total do processo de desde inicio da procura até o acoplamento
- X_FINAL - Posição final do AGV em centímetros em X
- Y_FINAL - Posição final do AGV em centímetros em Y
- REUSLT - Resultado da simulação (Sucesso corresponde a "Done", Insucesso corresponde a "Problem")

Tabela I.1: Resultados L1

Nr sim:	Dim	Lum	ID	X	Y	DISTANCIA	DURAÇÃO	X_FINAL	Y_FINAL	RESULT
1.1	10	L1	0	-3	-1	0	90.0241110325	-2.99998559809	-0.999999853323	Problem
2.1	10	L1	0	-3	1	0	90.057649851	-2.99997604373	0.999999786415	Problem
3.1	10	L1	0	-2	-2	0	90.1158981323	-1.99998288977	-1.99999986858	Problem
4.1	10	L1	0	-2	-1	0	90.1049411297	-1.99998247682	-0.999996206667	Problem
5.1	10	L1	0	-2	0	0	90.055925131	-1.99997674425	3.2804065701E-08	Problem
6.1	10	L1	0	-2	1	0	90.037667036	-1.99997669804	0.999999785481	Problem
7.1	10	L1	0	-2	2	0	90.051284075	-1.99997766133	2.0000000315	Problem
8.1	10	L1	0	-1	-2	0	90.061819077	-0.999982636973	-1.99999629936	Problem
9.1	10	L1	0	-1	-1	0	90.037575007	-0.999982312369	-0.999996298858	Problem
10.1	10	L1	0	-1	0	0	90.039636135	-0.999977182185	3.21790889735E-08	Problem
11.1	10	L1	0	-1	1	0	90.065893889	-0.999977527603	1.00000004076	Problem
12.1	10	L1	0	-1	2	0	90.069903135	-0.999979900026	1.99996637081	Problem
13.1	10	L1	0	-0.5	-2	0	90.017910004	-0.499981837919	-1.99999620568	Problem
14.1	10	L1	0	-0.5	-1.5	0	90.006097078	-0.499981661251	-1.49999986741	Problem
15.1	10	L1	0	-0.5	-1	0	90.040133953	-0.499983641623	-0.999999849985	Problem
16.1	10	L1	0	-0.5	-0.5	0	90.069530964	-0.49998187762	-0.499999867038	Problem
17.1	10	L1	0	-0.5	0	0	90.0511069298	-0.499940955452	-5.1220336519E-06	Problem
18.1	10	L1	0	-0.5	0.5	0	90.0786211491	-0.499974791766	0.499999788211	Problem
19.1	10	L1	0	-0.5	1	0	90.016241074	-0.499976270418	0.999999786091	Problem
20.1	10	L1	0	-0.5	1.5	0	90.02046299	-0.49997866988	1.50000024007	Problem
21.1	10	L1	0	0	-3	0	90.052139044	1.648356313E-05	-2.99999986359	Problem
22.1	10	L1	0	0	-2	0	90.072498083	1.6736032625E-05	-1.99999986318	Problem
23.1	10	L1	0	0	-1.5	0	90.023572922	2.0795932893E-05	-1.49997876208	Problem
24.1	10	L1	2	0	-1	1.7960181375	37.302876949	2.01965155748	0.0186350122292	Done
25.1	10	L1	2	0	-0.5	1.572408924	34.247158051	2.01618029032	0.0188819525753	Done
26.1	10	L1	2	0	0	1.4899270798	26.061839104	2.01560769878	0.0160761067353	Done
27.1	10	L1	2	0	0.5	1.5722519592	30.800860882	2.00133629338	0.00896089339013	Done
28.1	10	L1	2	0	1	1.7960478324	34.209002018	2.00550814543	0.0103066063772	Done
29.1	10	L1	0	0	1.5	0	90.042353153	2.2613530222E-05	1.49999950054	Problem
30.1	10	L1	0	0	2	0	90.041842938	2.0136868605E-05	1.99996677957	Problem
31.1	10	L1	0	0	3	0	90.03403616	2.2869176598E-05	2.99999978486	Problem
32.1	10	L1	2	0.25	1	1.5954245585	32.996181965	2.00703329017	0.00663008462272	Done
33.1	10	L1	2	0.25	0	1.2399145967	24.7714080811	2.00947698625	0.0153508632711	Done
34.1	10	L1	2	0.25	-1	1.5960390103	34.096980095	2.01613426602	0.0173346310992	Done
35.1	10	L1	1	0.5	2	1.8490452419	66.773874044	1.99963414191	0.0183934738546	Done
36.1	10	L1	3	0.5	-2	1.8537474818	66.552407026	2.02088532564	0.00969439519847	Done
37.1	10	L1	0	1	-3	0	90.0028111935	1.00001637021	-2.99999986377	Problem
38.1	10	L1	3	1	-2	1.613049571	71.285784006	1.99830137056	0.0119948143984	Done
39.1	10	L1	1	1	2	1.6080471504	67.639734983	1.99992585718	0.0188861359194	Done
40.1	10	L1	0	1	3	0	90.094098806	1.00002270692	2.9999997873	Problem

Tabela I.2: Resultados L2

Nr sim:	Dim	Lum	ID	X	Y	DISTANCIA	DURAÇÃO	X_FINAL	Y_FINAL	RESULT
1.1	10	L2	0	-3	-1	0	90.066704035	-2.99998602054	-0.999999867648	Problem
2.1	10	L2	0	-3	1	0	90.028891802	-2.99997445688	0.999999788693	Problem
3.1	10	L2	2	-2	-2	4.092869718	51.34075594	2.00323749521	0.0181494597751	Done
4.1	10	L2	0	-2	-1	0	90.04767108	-1.99996375431	-0.999960683168	Problem
5.1	10	L2	0	-2	0	0	90.081818819	-1.99997715658	-2.1517170881E-07	Problem
6.1	10	L2	0	-2	1	0	90.042235136	-2.00008680329	0.999964580606	Problem
7.1	10	L2	2	-2	2	4.088910698	45.509943962	2.01954202294	0.0113164525942	Done
8.1	10	L2	0	-1	-2	0	90.0034871101	-0.999759355647	-1.99980828418	Problem
9.1	10	L2	2	-1	-1	2.6842307197	41.397083998	2.01165161879	0.0158486582481	Done
10.1	10	L2	2	-1	0	2.4899281505	31.7791109085	1.9919634364	0.0195196266796	Done
11.1	10	L2	2	-1	1	2.6845429362	38.730257988	2.00627310012	0.00796232623026	Done
12.1	10	L2	0	-1	2	0	90.055242062	-0.999985213893	1.99987761843	Problem
13.1	10	L2	2	-0.5	-2	2.8256763947	44.496413946	2.01895853496	0.00889063947976	Done
14.1	10	L2	2	-0.5	-1.5	2.4944106449	39.372327089	2.01413124974	0.0183376937332	Done
15.1	10	L2	2	-0.5	-1	2.2291529686	38.03934598	2.00127937639	0.00848602585361	Done
16.1	10	L2	2	-0.5	-0.5	2.0524362273	34.933059931	2.01381246861	0.0156354179802	Done
17.1	10	L2	2	-0.5	0	1.9899274779	28.973073006	2.01145461507	0.015375070674	Done
18.1	10	L2	2	-0.5	0.5	2.0522760983	33.7145299911	2.00569422678	0.0136043696813	Done
19.1	10	L2	2	-0.5	1	2.2282823786	36.252974987	2.01415063538	0.00782122385644	Done
20.1	10	L2	2	-0.5	1.5	2.4940208821	39.2709980011	2.01757572245	0.00752115085145	Done
21.1	10	L2	0	0	-3	0	90.080537081	3.5634603313E-05	-3.00002214438	Problem
22.1	10	L2	2	0	-2	2.4993816546	40.886326075	1.99979092807	0.0214382384087	Done
23.1	10	L2	2	0	-1.5	2.11770752094	37.833350897	2.01998198827	0.016489210148	Done
24.1	10	L2	2	0	-1	1.7958299678	39.04962492	2.00396377669	0.0194902585255	Done
25.1	10	L2	2	0	-0.5	1.5724526135	31.202939987	2.00235340074	0.0179779744598	Done
26.1	10	L2	2	0	0	1.4899285022	28.381228924	1.99936935837	0.0168261056824	Done
27.1	10	L2	2	0	0.5	1.5720170374	30.825763941	2.01818119902	0.0144553970943	Done
28.1	10	L2	2	0	1	1.7962786878	35.872057915	2.01821120681	0.00770509643519	Done
29.1	10	L2	2	0	1.5	2.1172848519	37.8255229	2.00344904284	0.00785323378962	Done
30.1	10	L2	2	0	2	2.4992722485	40.459293842	2.00578522405	0.010358244591	Done
31.1	10	L2	0	0	3	0	90.049743176	3.8063014271E-05	2.99999134328	Problem
32.1	10	L2	1	0.25	1	1.4254166379	90.033559084	-6.31557886738	-3.54644913023	Problem
33.1	10	L2	2	0.25	0	1.2399162494	24.764424801	2.00931834419	0.0153935661272	Done
34.1	10	L2	2	0.25	-1	1.5958068318	35.575147152	2.01301846843	0.0169150920024	Done
35.1	10	L2	2	0.5	2	2.2373350557	40.065582991	2.052560293	-0.00036972996564	Done
36.1	10	L2	2	0.5	-2	2.2379652078	39.545125961	2.05648857815	0.0151580869271	Done
37.1	10	L2	3	1	-3	2.5713429486	83.320166826	2.00805854536	0.0107466957036	Done
38.1	10	L2	3	1	-2	1.6133464345	72.68145299	2.01351552598	0.0110141605413	Done
39.1	10	L2	1	1	2	1.6080053624	70.141685009	2.01597314429	0.0203729479444	Done
40.1	10	L2	1	1	3	2.5660557694	69.850855827	2.0063841601	0.0201233779302	Done

Tabela I.3: Resultados L3

Nr sim:	Dim	Lum	ID	X	Y	DISTANCIA	DURAÇÃO	X_FINAL	Y_FINAL	RESULT
1.1	10	L3	2	-3	-1	4.600729316	50.844724894	2.01314056288	0.012793796367	Done
2.1	10	L3	2	-3	1	4.6017078682	47.662399054	1.99435385508	0.00712712853796	Done
3.1	10	L3	2	-2	-2	4.0244565752	53.559044123	2.00181999905	0.0112481347664	Done
4.1	10	L3	2	-2	-1	3.6313937482	47.280779123	2.01535259474	0.0141714976266	Done
5.1	10	L3	2	-2	0	3.4899282314	38.268874884	2.01408342964	0.0174789996302	Done
6.1	10	L3	2	-2	1	3.63112052341	43.905475855	2.00714725559	0.0120355594806	Done
7.1	10	L3	2	-2	2	4.0245027371	47.849421024	2.00477760511	0.00605105442434	Done
8.1	10	L3	2	-1	-2	3.1961545391	52.104382038	1.99411890261	0.0103372015189	Done
9.1	10	L3	2	-1	-1	2.6845602289	41.418296099	2.01144932343	0.01589020075	Done
10.1	10	L3	2	-1	0	2.489928617	31.92749691	2.00731122945	0.0144951594064	Done
11.1	10	L3	2	-1	1	2.6841536045	38.653106928	2.00666408218	0.0127793120778	Done
12.1	10	L3	2	-1	2	3.1972480209	43.775060892	2.00712352147	0.00780449102828	Done
13.1	10	L3	2	-0.5	-2	2.8258151572	44.690440893	2.00980426858	0.0132429052683	Done
14.1	10	L3	2	-0.5	-1.5	2.4936470255	41.692483902	1.99860781106	0.010392935931	Done
15.1	10	L3	2	-0.5	-1	2.2282363962	38.657413006	2.01027026739	0.0146547168885	Done
16.1	10	L3	2	-0.5	-0.5	2.0522590308	34.633324862	2.02128151969	0.0102387240777	Done
17.1	10	L3	2	-0.5	0	1.9899280416	28.877722979	2.01192537026	0.0158168902852	Done
18.1	10	L3	2	-0.5	0.5	2.0518703202	33.820652008	2.02195269412	0.0142629516267	Done
19.1	10	L3	2	-0.5	1	2.2284500252	36.134593964	1.99803849777	0.0116743439259	Done
20.1	10	L3	2	-0.5	1.5	2.4942862773	39.237756968	2.0178478788	0.00743213076251	Done
21.1	10	L3	3	0	-3	2.9613360482	76.836776972	2.01461706694	0.012180199806	Done
22.1	10	L3	2	0	-2	2.49821121369	40.85039711	2.01372473085	0.0176504675925	Done
23.1	10	L3	2	0	-1.5	2.11801667478	37.970319986	2.01940645367	0.0189022998967	Done
24.1	10	L3	2	0	-1	1.7958780769	35.644377947	2.00212449659	0.018049283324	Done
25.1	10	L3	2	0	-0.5	1.5723758648	31.1961150169	2.00189990194	0.0177336551506	Done
26.1	10	L3	2	0	0	1.48992252114	26.18241787	2.01544430868	0.01559211297	Done
27.1	10	L3	2	0	0.5	1.5719502096	30.723023891	2.01768636558	0.0119941286681	Done
28.1	10	L3	2	0	1	1.7958425378	35.547204971	2.00244448881	0.0111871947178	Done
29.1	10	L3	2	0	1.5	2.11728661207	37.929954052	2.01931718232	0.00989841090131	Done
30.1	10	L3	2	0	2	2.4974020976	40.641945124	2.02102759395	0.0116385257378	Done
31.1	10	L3	1	0	3	2.9573337832	81.630740166	2.01218834695	0.0233696650763	Done
32.1	10	L3	1	0.25	1	1.42520411353	64.879559994	2.00913766932	0.0147519299844	Done
33.1	10	L3	2	0.25	0	1.2399203673	24.563014984	2.00960383274	0.0129487449245	Done
34.1	10	L3	2	0.25	-1	1.595589447	35.605004072	1.99672426918	0.0224077980119	Done
35.1	10	L3	2	0.5	2	2.2370714265	39.740505934	2.02211192372	0.00087382601403	Done
36.1	10	L3	2	0.5	-2	2.2417914978	38.619715929	2.02524468912	0.0150379536983	Done
37.1	10	L3	3	1	-3	2.5712252346	76.6706779	2.00806135689	0.0105281701246	Done
38.1	10	L3	3	1	-2	1.6133825991	69.422897816	1.99939625271	0.0115258176636	Done
39.1	10	L3	1	1	2	1.6080095377	67.987557888	2.0105876317	0.0199663103252	Done
40.1	10	L3	1	1	3	2.566073536	69.8451118469	2.00065278632	0.0177669106587	Done

Tabela I.4: Resultados L4

Nr sim:	Dim	Lum	ID	X	Y	DISTANCIA	DURAÇÃO	X_FINAL	Y_FINAL	RESULT
1.1	10	L4	0	-3	-1	0	90.024935961	-2.99996307126	-0.999994234746	Problem
2.1	10	L4	0	-3	1	0	90.070885897	-2.99996514629	0.999997421682	Problem
3.1	10	L4	0	-2	-2	0	90.044307947	-1.99979137861	-1.99988339193	Problem
4.1	10	L4	2	-2	-1	3.6316304214	90.04469204	0.829709024459	-0.303142667803	Problem
5.1	10	L4	2	-2	0	3.4898900757	41.863013983	1.99964720585	0.0142819343741	Done
6.1	10	L4	2	-2	1	3.63111505167	42.905275106	2.00754499001	0.0125206609611	Done
7.1	10	L4	0	-2	2	0	90.092571974	-1.99967420009	1.99982382444	Problem
8.1	10	L4	2	-1	-2	3.1967713987	46.081743002	2.00033556814	0.00788464799651	Done
9.1	10	L4	2	-1	-1	2.6843603842	41.370863915	2.01145451195	0.0142941538418	Done
10.1	10	L4	2	-1	0	2.4899313074	31.885062933	2.00765345708	0.0139172387475	Done
11.1	10	L4	2	-1	1	2.6838885774	38.699829102	1.99184125844	0.00501745240972	Done
12.1	10	L4	2	-1	2	3.19825115098	43.37242794	1.99933592733	0.00625141260905	Done
13.1	10	L4	2	-0.5	-2	2.827866137	41.603461027	2.02489096449	0.0180542063672	Done
14.1	10	L4	2	-0.5	-1.5	2.4949270877	39.394579887	2.0143577943	0.0219673792122	Done
15.1	10	L4	2	-0.5	-1	2.2295103198	38.214478016	2.01640759984	0.0115281896366	Done
16.1	10	L4	2	-0.5	-0.5	2.0525290502	90.064426184	0.979601217772	-0.00951822822344	Problem
17.1	10	L4	2	-0.5	0	1.9899297562	28.858847857	2.01149501449	0.01653525943	Done
18.1	10	L4	2	-0.5	0.5	2.0519213492	33.774274826	1.98992833129	0.00606777071849	Done
19.1	10	L4	2	-0.5	1	2.2281881563	36.099251986	1.99924863076	0.0052583785395	Done
20.1	10	L4	2	-0.5	1.5	2.494470703	39.233725071	2.00499683202	0.00512465895516	Done
21.1	10	L4	3	0	-3	2.9621785695	75.659203053	2.00692291366	0.0115317191053	Done
22.1	10	L4	2	0	-2	2.5030632865	39.817740917	2.01049533739	0.0209499663671	Done
23.1	10	L4	2	0	-1.5	2.11852693481	37.812135935	2.0212834382	0.0178913105779	Done
24.1	10	L4	2	0	-1	1.7962480776	35.749152899	2.00350214862	0.0234620856462	Done
25.1	10	L4	2	0	-0.5	1.5723190168	31.485093832	2.00162195892	0.0203209567531	Done
26.1	10	L4	2	0	0	1.489926935	26.045704126	2.01554389412	0.0157218881022	Done
27.1	10	L4	2	0	0.5	1.5719967425	31.418731928	2.00139360417	0.00837811425064	Done
28.1	10	L4	2	0	1	1.796210456	35.721926928	2.02119402757	0.0117245488682	Done
29.1	10	L4	2	0	1.5	2.11856545707	37.979079962	1.9981437923	0.00019125830368	Done
30.1	10	L4	2	0	2	2.5039737717	39.9111530781	2.00507194611	0.00089862785172	Done
31.1	10	L4	1	0	3	2.9572387367	90.018657923	-1.01859801002	2.16253593115	Problem
32.1	10	L4	1	0.25	1	1.4251324829	64.685774088	2.00789583507	0.0176868110942	Done
33.1	10	L4	2	0.25	0	1.2399240165	24.388479948	1.99383644503	0.0123359075547	Done
34.1	10	L4	2	0.25	-1	1.596088366	35.271743059	2.0093053239	0.0209058525089	Done
35.1	10	L4	1	0.5	2	1.8490645098	64.623152018	1.99934112272	0.0195376150492	Done
36.1	10	L4	3	0.5	-2	1.853886839	67.606948137	2.00428849912	0.0133082026724	Done
37.1	10	L4	3	1	-3	2.57113481522	77.375464916	2.01242904454	0.00976388758822	Done
38.1	10	L4	3	1	-2	1.6133246488	69.391870022	1.99735955428	0.0116883969388	Done
39.1	10	L4	1	1	2	1.6082086703	69.21834898	2.01655339582	0.020352181889	Done
40.1	10	L4	1	1	3	2.5659710132	90.040086031	-0.233866846077	2.53592778098	Problem

ANEXO II

Resultados dimensão *tag*

A descrição dos parâmetros apresentados nas tabelas seguintes são apresentados seguidamente:

- Nr Sim - Número correspondente à simulação efetuada
- Dim - Dimensão da *tag* no decorrer da simulação
- Lum - Luminosidade do ambiente no decorrer da simulação
- ID - ID da *tag* identificada pelo sistema
- X - Posição inicial do AGV em centímetros em X
- Y - Posição inicial do AGV em centímetros em Y
- DISTANCIA - Distância a que foi detetada a *tag*
- DURAÇÃO - Duração total do processo de desde inicio da procura até o acoplamento
- X_FINAL - Posição final do AGV em centímetros em X
- Y_FINAL - Posição final do AGV em centímetros em Y
- REUSLT - Resultado da simulação (Sucesso corresponde a "Done", Insucesso corresponde a "Problem")

Tabela II.1: Resultados 7.5x7.5cm

Nr sim:	Dim	Lum	ID	X	Y	DISTANCIA	DURAÇÃO	X_FINAL	Y_FINAL	RESULT
1.1	7.5	L3	0	-3	-1	0	90.014719963	-2.99998466562	-0.999999376447	Problem
2.1	7.5	L3	0	-3	1	0	90.093734026	-2.99999501609	0.999962145331	Problem
3.1	7.5	L3	0	-2	-2	0	90.058382034	-1.99989884896	-1.99994570682	Problem
4.1	7.5	L3	2	-2	-1	3.631644229	90.027499914	0.984440589018	-0.0173005184007	Problem
5.1	7.5	L3	2	-2	0	3.4899142501	41.375623941	2.01539285572	0.0185299747756	Done
6.1	7.5	L3	2	-2	1	3.6316044248	42.258548021	2.00667573878	0.0111004892986	Done
7.1	7.5	L3	0	-2	2	0	90.022172928	-1.99993376472	1.99997527163	Problem
8.1	7.5	L3	0	-1	-2	0	90.030802965	-0.999662867056	-1.99975011637	Problem
9.1	7.5	L3	2	-1	-1	2.6838723707	90.071043015	0.859442604904	-0.691616102262	Problem
10.1	7.5	L3	2	-1	0	2.4899619147	31.800640822	1.99132577116	0.0129663761489	Done
11.1	7.5	L3	2	-1	1	2.6842540405	38.752691031	2.00761941509	0.0106395552991	Done
12.1	7.5	L3	2	-1	2	3.1932426417	48.59610796	2.00976971102	0.0069979096444	Done
13.1	7.5	L3	2	-0.5	-2	2.8196030729	90.041316032	-0.167913657309	0.0331498706568	Problem
14.1	7.5	L3	2	-0.5	-1.5	2.4958742171	41.618134022	2.01124792803	0.0116291339	Done
15.1	7.5	L3	2	-0.5	-1	2.22881140297	38.12668395	2.01099936505	0.0146535678486	Done
16.1	7.5	L3	2	-0.5	-0.5	2.052482401	90.0112540722	0.83644378974	-0.284424238166	Problem
17.1	7.5	L3	2	-0.5	0	1.9899629125	28.913382053	2.01244051867	0.0164150318929	Done
18.1	7.5	L3	2	-0.5	0.5	2.052272494	33.741163969	2.00548530298	0.0131137250089	Done
19.1	7.5	L3	2	-0.5	1	2.2282041538	36.333370209	2.01449371566	0.00730925420812	Done
20.1	7.5	L3	2	-0.5	1.5	2.49394311051	39.296252012	2.00417438362	0.0095819284227	Done
21.1	7.5	L3	3	0	-3	2.9622717263	77.64310813	2.01505343068	0.0121582731989	Done
22.1	7.5	L3	2	0	-2	2.5008491681	52.502550125	2.00171215442	0.0193294474051	Done
23.1	7.5	L3	2	0	-1.5	2.11720362475	37.863057852	2.00383830256	0.0202186846335	Done
24.1	7.5	L3	2	0	-1	1.7972336248	39.059106827	2.00819212996	0.0187276592864	Done
25.1	7.5	L3	2	0	-0.5	1.5722820279	31.416707993	2.00021927493	0.0196672197845	Done
26.1	7.5	L3	2	0	0	1.48996177117	26.071722031	2.01542162393	0.0164947943781	Done
27.1	7.5	L3	2	0	0.5	1.5720580866	31.227159977	2.00097467077	0.0100638685669	Done
28.1	7.5	L3	2	0	1	1.7966300049	35.494040012	2.00412489841	0.00943276158671	Done
29.1	7.5	L3	2	0	1.5	2.11749889291	37.63776803	1.9973322977	0.00600495005958	Done
30.1	7.5	L3	2	0	2	2.498565149	39.477030993	1.98865827377	0.00764636867669	Done
31.1	7.5	L3	1	0	3	2.9573197459	64.837955952	2.01445401782	0.0244917150793	Done
32.1	7.5	L3	1	0.25	1	1.4242203786	90.092630148	0.198057155721	-0.313778769628	Problem
33.1	7.5	L3	2	0.25	0	1.2399525391	24.759488106	2.00916566185	0.0141976174632	Done
34.1	7.5	L3	2	0.25	-1	1.5959035164	35.209942102	2.01361652934	0.0164386290619	Done
35.1	7.5	L3	1	0.5	2	1.8495505098	64.9356110096	1.99998791125	0.0192527574404	Done
36.1	7.5	L3	3	0.5	-2	1.8530684715	65.929809094	1.99748309244	0.011213569554	Done
37.1	7.5	L3	3	1	-3	2.57135614112	79.617942095	1.99244438025	0.0106719518229	Done
38.1	7.5	L3	3	1	-2	1.6133184512	69.851991892	2.01392902061	0.0105118624142	Done
39.1	7.5	L3	1	1	2	1.6081699345	69.27096796	2.0006736221	0.0182405284356	Done
40.1	7.5	L3	1	1	3	2.5659407074	69.459783077	2.00142872613	0.019319670507	Done

Tabela II.2: Resultados 5x5cm

Nr sim:	Dim	Lum	ID	X	Y	DISTANCIA	DURAÇÃO	X_FINAL	Y_FINAL	RESULT
1.1	5	L3	0	-3	-1	0	90.083139896	-2.9999834786	-0.999999870369	Problem
2.1	5	L3	0	-3	1	0	90.000441074	-2.99997749921	1.00000003854	Problem
3.1	5	L3	0	-2	-2	0	90.060562134	-1.99998325335	-1.99999986997	Problem
4.1	5	L3	0	-2	-1	0	90.072366953	-1.99998400271	-0.999999871252	Problem
5.1	5	L3	0	-2	0	0	90.050987005	-1.99997446203	-2.1131397938E-07	Problem
6.1	5	L3	0	-2	1	0	90.031800032	-1.99997692428	1.00000001882	Problem
7.1	5	L3	0	-2	2	0	90.058655977	-1.99997684764	2.00000004624	Problem
8.1	5	L3	0	-1	-2	0	90.0771451	-0.999982843786	-1.99999620634	Problem
9.1	5	L3	2	-1	-1	2.6846505424	90.03481698	0.85681431338	-0.332143786992	Problem
10.1	5	L3	2	-1	0	2.4899606416	35.483598948	2.02316725524	0.0186823397173	Done
11.1	5	L3	2	-1	1	2.682458416	41.0858531	2.00806026954	0.0104151749713	Done
12.1	5	L3	0	-1	2	0	90.017307997	-0.999963672546	1.99998605122	Problem
13.1	5	L3	0	-0.5	-2	0	90.051076889	-0.499983789516	-1.99999986403	Problem
14.1	5	L3	2	-0.5	-1.5	2.4913112402	90.081962109	1.13033868332	-0.137749263094	Problem
15.1	5	L3	2	-0.5	-1	2.2290393135	37.64903903	1.99896573748	0.0127278495842	Done
16.1	5	L3	2	-0.5	-0.5	2.0524560705	34.851657867	1.99489066965	0.0169016924607	Done
17.1	5	L3	2	-0.5	0	1.9899789101	28.800424814	2.01174929705	0.0151240932489	Done
18.1	5	L3	2	-0.5	0.5	2.05188115055	33.539782047	2.00629455916	0.0152223908449	Done
19.1	5	L3	2	-0.5	1	2.22811388292	36.027734995	2.01594949749	0.00943768236358	Done
20.1	5	L3	2	-0.5	1.5	2.4959210929	37.66368103	2.0229791252	0.0104974587333	Done
21.1	5	L3	0	0	-3	0	90.015040159	6.5792838583E-05	-2.99991487051	Problem
22.1	5	L3	3	0	-2	2.1814591846	62.45802784	2.00161163312	0.014377737209	Done
23.1	5	L3	2	0	-1.5	2.11976395036	36.978434086	1.99652031303	0.0219122701357	Done
24.1	5	L3	2	0	-1	1.7964576091	34.840651989	2.01839947583	0.0196204076166	Done
25.1	5	L3	2	0	-0.5	1.5725189704	90.076183081	1.02795149021	-0.049895055676	Problem
26.1	5	L3	2	0	0	1.4899740218	26.10759902	2.01512161745	0.0208908816767	Done
27.1	5	L3	2	0	0.5	1.5721683155	31.196182013	2.00263590747	0.0156279781504	Done
28.1	5	L3	2	0	1	1.7965270846	34.754769087	2.00417195646	0.00664985749231	Done
29.1	5	L3	2	0	1.5	2.11842994984	36.715622902	2.00143193768	0.00963945315636	Done
30.1	5	L3	1	0	2	2.17977301191	62.243074179	2.00451996795	0.0196908502013	Done
31.1	5	L3	0	0	3	0	90.050504923	5.6148518735E-05	2.99996314233	Problem
32.1	5	L3	1	0.25	1	1.4241995008	90.100076199	-6.2164459225	-3.7752749057	Problem
33.1	5	L3	2	0.25	0	1.2399674491	24.553224087	1.99297686529	0.0133698926012	Done
34.1	5	L3	2	0.25	-1	1.596048784	35.134155035	2.01091112267	0.0183264958868	Done
35.1	5	L3	1	0.5	2	1.8496209374	64.730108976	2.01565267696	0.0213659166791	Done
36.1	5	L3	3	0.5	-2	1.8539260529	65.173877955	2.01257378183	0.0140955852329	Done
37.1	5	L3	3	1	-3	2.5712978581	78.3418011665	2.00808900274	0.0137519261368	Done
38.1	5	L3	3	1	-2	1.6131732632	70.025630951	2.00687334695	0.0143048767833	Done
39.1	5	L3	1	1	2	1.6081543975	70.763679028	1.99305009511	0.0240776520706	Done
40.1	5	L3	1	1	3	2.5659997793	68.271329165	2.01796355259	0.0220940280728	Done

ANEXO III

Outros resultados

A descrição dos parâmetros apresentados nas tabelas seguintes são apresentados seguidamente:

- Nr Sim - Número correspondente à simulação efetuada
- Dim - Dimensão da *tag* no decorrer da simulação
- Lum - Luminosidade do ambiente no decorrer da simulação
- ID - ID da *tag* identificada pelo sistema
- X - Posição inicial do AGV em centímetros em X
- Y - Posição inicial do AGV em centímetros em Y
- DISTANCIA - Distância a que foi detetada a *tag*
- DURAÇÃO - Duração total do processo de desde inicio da procura até o acoplamento
- X_FINAL - Posição final do AGV em centímetros em X
- Y_FINAL - Posição final do AGV em centímetros em Y
- REUSLT - Resultado da simulação (Sucesso corresponde a "Done", Insucesso corresponde a "Problem")

Tabela III.1: Resultados sem recurso a *depth image*

Nr sim:	Dim	Lur	ID	X	Y		DISTANCIA	DURAÇÃO	X_FINAL	Y_FINAL	RESULT
1.1	10	L3	2	-3	-1	0	4.77349765156	52.8819351196	1.95872264521	0.023578235526	Done
2.1	10	L3	2	-3	1	0	4.77349759386	48.7097890377	1.96025904932	-0.00200510608781	Done
3.1	10	L3	2	-2	-2	0	4.20921532167	46.4289851189	1.94336505108	0.0313542247687	Done
5.1	10	L3	2	-2	0	0	3.77032171033	38.5586230755	1.95246991628	0.0117252685312	Done
4.1	10	L3	2	-2	-1	0	3.77481461993	44.2246689796	1.94136729489	0.0319377426078	Done
6.1	10	L3	2	-2	1	0	3.77479415805	44.1661601067	1.94751218838	0.00668424531628	Done
7.1	10	L3	2	-2	2	0	4.20919291435	46.715280056	1.96535497493	-0.00196802313857	Done
8.1	10	L3	2	-1	-2	0	3.28884363936	43.9933459759	1.92760005392	0.0392475833683	Done
9.1	10	L3	2	-1	-1	0	2.76002267136	39.046391964	1.94503891186	0.0320768234359	Done
10.1	10	L3	2	-1	0	0	2.56279515231	31.6043231487	1.9433409376	0.00821862946687	Done
11.1	10	L3	2	-1	1	0	2.75369521913	38.9926819801	1.96382425243	-0.0106401321292	Done
12.1	10	L3	2	-1	2	0	3.26938459339	44.1115291119	1.96092754311	-0.0166412455025	Done
13.1	10	L3	2	-0.5	-2	0	2.9017795722	42.3869888783	1.90814799157	0.0479672791996	Done
14.1	10	L3	2	-0.5	-1.5	0	2.54355357122	39.5949759483	1.95176620894	0.0268142289576	Done
15.1	10	L3	2	-0.5	-1	0	2.27531277801	36.4143149853	1.94074341637	0.0334271125442	Done
16.1	10	L3	2	-0.5	-0.5	0	2.07547666847	34.1526999474	1.95901477305	0.0174444469212	Done
17.1	10	L3	2	-0.5	0	0	2.00907765742	28.5855998993	1.96326926507	0.0173770474376	Done
18.1	10	L3	2	-0.5	0.5	0	2.07687433892	33.9071381092	1.95822767146	-0.0073788367854	Done
19.1	10	L3	2	-0.5	1	0	2.27249867914	36.6998450756	1.95986821619	-0.0026459111989	Done
20.1	10	L3	2	-0.5	1.5	0	2.5392359921	39.5584671497	1.9518613823	-0.0128019159187	Done
21.1	10	L3	3	0	-3	0	3.03473356679	78.0575289726	1.89736456894	0.0157840036747	Done
22.1	10	L3	2	0	-2	0	2.55867477384	41.0131890774	1.95747211563	0.0429442805681	Done
23.1	10	L3	2	0	-1.5	0	2.1529626901	38.2399890423	1.94791398186	0.0337726764459	Done
24.1	10	L3	2	0	-1	0	1.81920554175	35.589111805	1.92027120917	0.0390156538971	Done
25.1	10	L3	2	0	-0.5	0	1.58932265601	31.6914749146	1.93745702262	0.0268049066221	Done
26.1	10	L3	2	0	0	0	1.51547854431	25.798635006	1.96862182873	0.015060662657	Done
27.1	10	L3	2	0	0.5	0	1.58769745737	31.6005268097	1.9705645526	-0.00952309796128	Done
28.1	10	L3	2	0	1	0	1.81489533381	35.9899280071	1.95903871768	0.00289675016811	Done
29.1	10	L3	2	0	1.5	0	2.14572016155	38.1473760605	1.96882746433	-0.011336500576	Done
30.1	10	L3	2	0	2	0	2.54593317099	41.0946550369	1.95917782579	-0.0321236468132	Done
31.1	10	L3	1	0	3	0	3.02751303555	75.4902451038	1.89383914347	0.00243495952381	Done
32.1	10	L3	1	0.25	1	0	1.44202639688	62.0113799572	1.87727632507	0.0140206802836	Done
33.1	10	L3	2	0.25	0	0	1.24900625411	23.7900879383	1.94554499103	0.00430819067499	Done
34.1	10	L3	2	0.25	-1	0	1.61419322578	36.2319991589	1.97537671505	0.0246755978834	Done
35.1	10	L3	2	0.5	2	0	2.27208470498	90.0108270645	0.917768280372	0.0898462337047	Problem
36.1	10	L3	2	0.5	-2	0	2.28212340895	90.0315599442	0.923869110545	-0.0802366407608	Problem
37.1	10	L3	3	1	-3	0	2.64816853315	78.5419421196	1.89021818643	0.0195004048202	Done
38.1	10	L3	3	1	-2	0	1.63060528114	70.3733708858	1.88980439279	0.022637713148	Done
39.1	10	L3	1	1	2	0	1.63258349833	68.1089000702	1.89353456487	-0.00628440360818	Done
40.1	10	L3	1	1	3	0	2.62696671189	77.2767589092	1.88946006834	0.000537392153906	Done

Tabela III.2: Erros associados com e sem recurso a *depth image* nas posições finais e distâncias do AGV à *tag*

	ERROS IMAGEM PROFUNDIDADE(cm)				ERROS IMAGEM(cm)			
Nr. SIM	**ERRO EM X**	**ERRO EM Y**	**ERRO DIST**	**RESULT**	**ERRO EM X**	**ERRO EM Y**	**ERRO DIST**	**RESULT**
1.1	0.013	0.013	0.009	Done	0.041	0.024	0.164	Done
2.1	0.006	0.007	0.008	Done	0.040	0.002	0.164	Done
3.1	0.002	0.011	0.007	Done	0.057	0.031	0.178	Done
4.1	0.015	0.014	0.009	Done	0.048	0.012	0.270	Done
5.1	0.014	0.017	0.010	Done	0.059	0.032	0.135	Done
6.1	0.007	0.012	0.009	Done	0.052	0.007	0.135	Done
7.1	0.005	0.006	0.007	Done	0.035	0.002	0.178	Done
8.1	0.006	0.010	0.005	Done	0.072	0.039	0.087	Done
9.1	0.011	0.016	0.008	Done	0.055	0.032	0.067	Done
10.1	0.007	0.014	0.010	Done	0.057	0.008	0.063	Done
11.1	0.007	0.013	0.008	Done	0.036	0.011	0.061	Done
12.1	0.007	0.008	0.004	Done	0.039	0.017	0.068	Done
13.1	0.010	0.013	0.003	Done	0.092	0.048	0.073	Done
14.1	0.001	0.010	0.006	Done	0.048	0.027	0.044	Done
15.1	0.010	0.015	0.008	Done	0.059	0.033	0.039	Done
16.1	0.021	0.010	0.009	Done	0.041	0.017	0.014	Done
17.1	0.012	0.016	0.010	Done	0.037	0.017	0.009	Done
18.1	0.022	0.014	0.010	Done	0.042	0.007	0.015	Done
19.1	0.002	0.012	0.008	Done	0.040	0.003	0.036	Done
20.1	0.018	0.007	0.006	Done	0.048	0.013	0.039	Done
21.1	0.015	0.012	0.012	Done	0.103	0.016	0.086	Done
22.1	0.014	0.018	0.002	Done	0.043	0.043	0.059	Done
23.1	0.019	0.019	0.003	Done	0.052	0.034	0.032	Done
24.1	0.002	0.018	0.007	Done	0.080	0.039	0.016	Done
25.1	0.002	0.018	0.009	Done	0.063	0.027	0.008	Done
26.1	0.015	0.016	0.010	Done	0.031	0.015	0.015	Done
27.1	0.018	0.012	0.009	Done	0.029	0.010	0.007	Done
28.1	0.002	0.011	0.007	Done	0.041	0.003	0.012	Done
29.1	0.019	0.010	0.004	Done	0.031	0.011	0.024	Done
30.1	0.021	0.012	0.003	Done	0.041	0.032	0.046	Done
31.1	0.012	0.023	0.008	Done	0.106	0.002	0.079	Done
32.1	0.009	0.015	0.008	Done	0.123	0.014	0.025	Done
33.1	0.010	0.013	0.010	Done	0.054	0.004	0.001	Done
34.1	0.003	0.022	0.005	Done	0.025	0.025	0.013	Done
35.1	0.022	0.001	0.001	Done	1.082	0.090	0.036	Problem
36.1	0.025	0.015	0.006	Done	1.076	0.080	0.046	Problem
37.1	0.008	0.011	0.015	Done	0.110	0.020	0.092	Done
38.1	0.001	0.012	0.014	Done	0.110	0.023	0.032	Done
39.1	0.011	0.020	0.009	Done	0.106	0.006	0.034	Done
40.1	0.001	0.018	0.009	Done	0.111	0.001	0.070	Done
MEDIA	**0.010**	**0.013**	**0.008**	**0**	**0.052**	**0.017**	**0.045**	**2**

Tabela III.3: Resultados poses aleatórias

Nr sim:	Dim	Lum	ID	X	Y	DISTANCIA	DURAÇÃO	X_FINAL	Y_FINAL	RESULT
1.1	10	L3	1	0.3	2.9	2.7198531546	77.979673147	2.00386281458	0.0241448807446	Done
2.1	10	L3	2	-0.1	-0.9	1.8286245009	35.347862005	2.0157492081	0.0200964330996	Done
3.1	10	L3	0	-1.5	2.9	0	110.06321907	-1.49965232654	2.89968637561	Problem
4.1	10	L3	2	-0.7	0	2.1899293786	30.021208048	2.00357955336	0.0156024028088	Done
5.1	10	L3	2	-0.6	0.9	2.27641169448	39.602627039	2.01122216783	0.00831260187263	Done
6.1	10	L3	0	-0.2	-2.8	0	110.088035107	-0.199852016168	-2.79980116356	Problem
7.1	10	L3	2	-1.4	2.6	3.9051919079	54.533345938	2.00178321463	0.00527218023338	Done
8.1	10	L3	2	-1.7	0.1	3.19118974015	37.658252955	1.99619289407	0.00872901472104	Done
9.1	10	L3	2	-1.3	-0.3	2.8061827001	37.932428122	2.01450768806	0.0168217250696	Done
10.1	10	L3	2	0.5	-1.1	1.4820363749	42.548138142	2.09495325775	0.0335216694287	Done
11.1	10	L3	0	-1	3	0	110.085222006	-0.999682336656	2.99967863956	Problem
12.1	10	L3	3	0.8	-2.8	2.4340787912	82.419674873	2.01770282256	0.0115186730105	Done
13.1	10	L3	2	-0.2	-0.5	1.7630937563	32.448029995	2.00368129667	0.0224197736018	Done
14.1	10	L3	2	0	-1.3	1.9793201913	45.90100193	2.00071460083	0.0227750982738	Done
15.1	10	L3	2	-1.7	-2.9	4.3378446816	71.9762020111	2.011823716	0.0069995267198	Done
16.1	10	L3	2	-0.8	0.5	2.3440058695	35.145334005	2.00919906016	0.0119319706502	Done
17.1	10	L3	2	-1.7	-1.4	3.4877250025	47.664226055	2.00764477484	0.0102910741385	Done
18.1	10	L3	2	-1.5	-2.3	3.771698508	61.35755682	2.00375950639	0.0111701069587	Done
19.1	10	L3	2	-1.2	-0.8	2.8070749949	41.308084965	2.00451484656	0.0151864169612	Done
20.1	10	L3	2	-1.8	2.1	3.9033309626	49.358559132	1.98935986512	0.00623122163675	Done
21.1	10	L3	3	0.3	-0.9	1.3465747384	79.279473066	2.0088405392	0.0157303981327	Done
22.1	10	L3	3	-0.9	-3	3.5289648499	81.7211520672	2.00672605444	0.0110150995633	Done
23.1	10	L3	2	0.1	0.1	1.3932584215	29.97340107	2.00297706398	0.0115802519975	Done
24.1	10	L3	2	-1.9	-1.7	3.7919856862	61.090291977	2.00927013625	0.0117451758735	Done
25.1	10	L3	1	0.9	2.1	1.7385518808	68.229290962	2.00517501936	0.0191740525825	Done
26.1	10	L3	0	0.9	0.2	0	110.018762827	0.837320010463	0.213515500524	Problem
27.1	10	L3	2	0	-2.4	2.8229988011	50.447553873	2.00067406848	0.0212641891754	Done
28.1	10	L3	2	-1.6	0.8	3.1919413562	40.989545822	2.02121370729	0.0130617608637	Done
29.1	10	L3	1	0.9	2.3	1.9240639423	73.382203102	2.0134480065	0.0203380351664	Done
30.1	10	L3	2	-0.6	-2.4	3.1808357393	110.066334963	1.22472950631	-0.179825503561	Problem
31.1	10	L3	1	1.6	2.3	1.7997289373	75.443700075	1.9981166834	0.0180173589324	Done
32.1	10	L3	1	1.1	1.7	1.292493361	66.120291948	1.99467447426	0.018966732631	Done
33.1	10	L3	1	1.9	1.5	1.050379547	69.445155144	1.99177332634	0.0181270562773	Done
34.1	10	L3	1	3	2.6	2.5364668671	89.83059001	1.99783402762	0.0190049679959	Done
35.1	10	L3	1	2.7	1.2	1.3259569568	76.766736984	2.00770340049	0.0159415304208	Done
36.1	10	L3	1	2.8	1.6	1.6468893503	76.219649077	1.99840131726	0.0190043816033	Done
37.1	10	L3	1	1.4	2.9	2.4062892559	71.690676928	1.99528109759	0.0187714363374	Done
38.1	10	L3	1	1.8	1.7	1.2198637863	72.307836056	2.01062123508	0.020335762288	Done
39.1	10	L3	1	2.4	2.1	1.7991300799	76.502649069	2.00725515358	0.021664794473	Done
40.1	10	L3	1	2.1	2.2	1.7782386578	76.253184795	2.01360959403	0.0196818459532	Done
41.1	10	L3	3	1.2	-1.9	1.4557137445	77.908910036	1.98550317473	0.0133949080328	Done
42.1	10	L3	3	2.3	-1.3	1.0817436112	76.941629887	2.00663819957	0.0150308622271	Done
43.1	10	L3	3	1.8	-1.9	1.4218670071	78.332229137	2.00999871119	0.0118406242193	Done
44.1	10	L3	3	1.9	-1.8	1.3434512107	78.363424063	2.00503769756	0.0112737305712	Done
45.1	10	L3	2	1.1	-1.5	1.5528785733	110.019775152	1.09861926393	-0.111487872021	Problem
46.1	10	L3	3	1.4	-1.8	1.3172236305	76.12748313	2.00872308736	0.0125563489653	Done
47.1	10	L3	3	1.3	-1.8	1.3346782172	77.0715110302	2.00215538904	0.0105214967691	Done
48.1	10	L3	3	1.3	-2.4	1.9255491488	84.978899002	2.01087274479	0.0120596047156	Done
49.1	10	L3	0	1.3	-1.3	0	110.010637045	1.30000753371	-1.29999215248	Problem
50.1	10	L3	3	1	-2.3	1.8962978485	78.525457144	2.00414914191	0.0109355220431	Done

Tabela III.4: Resultados várias *racks*

Nr sim:	Dim	Lum	ID	DISTANCIA	DURAÇAO	X_FINAL	Y_FINAL	RESULT
1.1	10	L3	32	2.05186724439	36.7614810467	-0.491947581551	-2.50525482239	Done
2.1	10	L3	23	2.49230866591	45.7219910622	-1.52137041923	2.51517709228	Done
3.1	10	L3	29	2.05167188686	37.0370118618	0.50774994785	-2.51397334393	Done
4.1	10	L3	35	2.49486079257	42.0852599144	-1.4944214456	-2.50465464839	Done
5.1	10	L3	32	2.05220841696	36.7173140049	-0.491739359813	-2.50480417462	Done
6.1	10	L3	20	2.05141202827	43.1704230309	-0.507066684422	2.50702429213	Done
7.1	10	L3	29	2.05182597737	37.0620980263	0.507555231219	-2.51484593733	Done
8.1	10	L3	26	2.49360104914	39.3374550343	1.51995073331	-2.50019221765	Done
9.1	10	L3	35	2.49473125328	42.8297250271	-1.49083090249	-2.52130618705	Done
10.1	10	L3	26	2.49332239816	39.5318601131	1.51998991639	-2.50072869819	Done
11.1	10	L3	35	2.49423978134	41.8894610405	-1.49439680908	-2.50508232048	Done
12.1	10	L3	41	2.68348039003	46.5187978745	-3.01156154968	-1.00650753871	Done
13.1	10	L3	20	2.0515798823	42.2442600727	-0.523469813066	2.50261599903	Done
14.1	10	L3	2	2.68447768023	41.2104427814	2.99507182914	1.01426972547	Done
15.1	10	L3	35	2.49482850746	42.2275419235	-1.49492474249	-2.50390022511	Done
16.1	10	L3	20	2.05137022701	43.4503340721	-0.504867853144	2.51359202067	Done
17.1	10	L3	5	2.68434999089	38.6887259483	2.99091944903	-0.99514549675	Done
18.1	10	L3	29	2.05152506516	35.7375409603	0.52220976413	-2.50169635644	Done
19.1	10	L3	17	2.05230676168	42.9898989201	0.491061181842	2.50630720177	Done
20.1	10	L3	2	2.6845584482	41.2744669914	3.01171137742	1.01422036223	Done
21.1	10	L3	14	2.4943869327	48.4179430008	1.49361411441	2.50534890686	Done
22.1	10	L3	2	2.6844766824	41.7283909321	3.00795282874	1.01376299943	Done
23.1	10	L3	5	2.68410807845	38.5615429878	3.02193742549	-0.988800598335	Done
24.1	10	L3	26	2.49334249449	41.6823620796	1.50828556335	-2.49876399146	Done
25.1	10	L3	29	2.0516265417	36.8218250275	0.506319674218	-2.4974719238	Done
26.1	10	L3	23	2.49268529515	45.7057681084	-1.52037101428	2.49969129928	Done
27.1	10	L3	41	2.68360131829	45.9615879059	-2.99391876481	-1.00864146851	Done
28.1	10	L3	35	2.49492366443	42.1728680134	-1.48950578044	-2.52355827385	Done
29.1	10	L3	23	2.49317304953	48.3008899689	-1.50715402468	2.51202587386	Done
30.1	10	L3	26	2.49310888844	39.2156739235	1.51966707235	-2.5002069075	Done