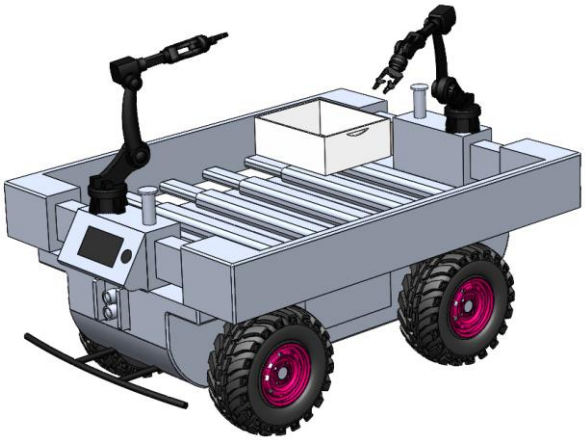
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Mecânica



Projeto de um robô móvel para tarefas de apoio à vindima

FILIPA GOMES LEITE SEMPITERNO AIRES

(Licenciada em Engenharia Mecânica)

Trabalho de Projeto para obtenção do grau de Mestre em Engenharia Mecânica

Orientadores:

Doutor Fernando Paulo Neves da Fonseca Cardoso Carreira
Doutor Mário José Gonçalves Cavaco Mendes

Júri:

Presidente: Doutor André Rui Dantas Carvalho

Vogais:

Doutor Jorge Manuel Mateus Martins
Doutor Fernando Paulo Neves da Fonseca Cardoso Carreira

Dezembro de 2023

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia Mecânica

Projeto de um robô móvel para tarefas de apoio à vindima

FILIPA GOMES LEITE SEMPITERNO AIRES

(Licenciada em Engenharia Mecânica)

Trabalho de Projeto para obtenção do grau de Mestre em Engenharia Mecânica

Orientadores:

Doutor Fernando Paulo Neves da Fonseca Cardoso Carreira
Doutor Mário José Gonçalves Cavaco Mendes

Júri:

Presidente: Doutor André Rui Dantas Carvalho

Vogais:

Doutor Jorge Manuel Mateus Martins
Doutor Fernando Paulo Neves da Fonseca Cardoso Carreira

Dezembro de 2023

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus orientadores de trabalho final de mestrado no Instituto Superior de Engenharia de Lisboa, ao Professor Fernando Carreira e Professor Mário Mendes, introduziram-me no fascinante mundo da automação e inspiraram-me a aprofundar os meus conhecimentos neste domínio. Apesar de me terem dado a liberdade para seguir o meu próprio percurso, estiveram sempre disponíveis para orientar-me na direção certa, sempre que achavam necessário.

Gostaria de deixar o meu agradecimento especial ao Tiago Almeida, ao Bogdan Lupu, ao Pedro Oliveira, ao Diogo Guerra, à Íris Esteves e à Mafalda Sousa pelo apoio que me deram durante todo o meu percurso na universidade. Assim como a todos os professores que direta ou indiretamente me deram os conhecimentos necessários para ter sucesso na minha aprendizagem e na minha vida. Um grande obrigada a todos os que me acompanharam durante esta jornada.

Por último, devo expressar a minha profunda gratidão aos meus pais e ao meu irmão por me terem dado um apoio e encorajamento contínuo ao longo dos meus anos de estudo e ao longo do processo que foi esta dissertação. A realização desta dissertação não teria sido possível sem a motivação e apoio deles.

Resumo

O setor agrícola é reconhecido pela sua elevada exigência em termos de recursos e mão-de-obra. Assim, muitos agricultores têm vindo a adotar cada vez mais tecnologia e a automação como solução para este desafio. No entanto, os robôs agrícolas existentes são frequentemente complexos, lentos e dispendiosos, o que dificulta a sua adoção generalizada. Como resultado, o setor agrícola tem ficado para trás na integração de tecnologias mais avançadas, existindo algum atraso na transformação digital possível no sector.

É importante observar que, apesar dos avanços nas pesquisas e na implementação de robôs agrícolas, a maioria das investigações e robôs existentes está focada na cultura de vinhas de uvas para produção de vinho. Poucos estudos abordaram adequadamente a colheita de uvas de mesa.

Como tal, este trabalho de investigação estudou e desenvolveu o conceito/arquitetura de um robô agrícola projetado para a colheita de uvas de mesa em campos agrícolas, bem como para a monitorização geral das culturas. Neste trabalho, o robô proposto foi construído num software de simulação robótica e foi criado um modelo de simulação semelhante a uma exploração agrícola.

A simulação robótica foi escolhida devido à sua rapidez, facilidade de implementação e capacidade de testar diferentes cenários e hipóteses de operação de forma simultânea, sem necessidade de construção de protótipos. Além disso, esta oferece liberdade e criatividade, sem preocupações com danos no hardware e custos reduzidos de desenvolvimento.

No caso de estudo da colheita de uvas, e dos vários cenários estudados, o cenário 2 foi o que representou maiores dificuldades pois os objetos eram diferentes, tanto a nível de peso como de cores. A maior complicação foi na recolha dos objetos maiores, tendo sido necessário aumentar ligeiramente o ângulo de abertura das pinças da garra. No entanto, os resultados de simulação permitiram comprovar a funcionalidade do conceito desenvolvido.

Palavras-Chave:

Robótica, Agricultura de Precisão, Colheita de uvas, Simulação

Abstract

The agricultural sector is renowned for its high demands on resources and labour. As a result, many farmers are increasingly adopting technology and automation as a solution to this challenge. However, existing agricultural robots are often complex, slow and expensive, which hinders their widespread adoption. As a result, the agricultural sector has lagged in integrating more advanced technologies, and there is some delay in the digital transformation possible in the sector.

It is important to note that, despite advances in research and the implementation of agricultural robots, most of the existing research and robots are focused on the cultivation of grape vines for wine production. Few studies have adequately addressed the harvesting of table grapes.

As such, this research work studied and developed the concept/architecture of an agricultural robot designed for harvesting table grapes in agricultural fields, as well as for general crop monitoring. In this work, the proposed robot was built using robotic simulation software and a simulation model similar to a farm was created.

Robotic simulation was chosen because of its speed, ease of implementation and ability to test different scenarios and operating hypotheses simultaneously, without the need to build prototypes. It also offers freedom and creativity, without worrying about hardware damage and reduced development costs.

In the case study of the grape harvest, and of the various scenarios studied, scenario 2 was the most difficult because the objects were different, both in terms of weight and color. The biggest complication was picking up the larger objects, which required slightly increasing the opening angle of the grapple's grippers. However, the simulation results proved the functionality of the concept developed.

Keywords:

Robotics, Precision Agriculture, Grape harvesting, Simulation

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

4WD	4 Wheel Drive (Tração às quatro Rodas)
AGV	Automated Guided Vehicle (Veículo Guiado Automático)
AMR	Autonomous Mobile Robot (Robô móvel autónomo)
AP	Agricultura de Precisão
BOM	Bill of Materials (Lista de materiais)
CROPS	Clever Robot for Crops (Robô inteligente para culturas)
DoF	Degrees of Freedom (Graus de liberdade)
FAO	Food and Agriculture Organization (Organização para a Alimentação e Agricultura)
GNSS	Global Navigation Satellite System (Sistema global de navegação por satélite)
GPS	Global Positioning System (Sistema de posicionamento Global)
GRAPE	Ground Robot for VineyArd Monitoring and ProtEction (Robô terrestre para monitorização e proteção da vinha)
GUI	Graphical User Interface (Interface gráfica do utilizador)
IMU	Inertial measurement unit (Unidade de medição por inércia)
IoT	Internet of Things (Internet das Coisas)
LIDAR	Light Detection And Ranging (Deteção de luz e alcance)
MOG	Material Other than Grapes (Outros materiais que não uvas)
NIR	Near-Infrared Spectroscopy (Espectroscopia de infravermelho próximo)
RGB	Red, Green, Blue (Vermelho, Verde, Azul)

RGB-D	Red, Green, Blue, Depth (Vermelho, Verde, Azul, Profundidade)
ROS	Robot Operating System (Sistema operativo do robô)
SAVSAR	Semi-Autonomous Vineyard Spraying Agricultural Robot (Robô agrícola semi-autónomo de pulverização de vinhas)
SIG	Sistemas de Informação Geográfica
SLAM	Simultaneous Localization And Mapping (Localização e mapeamento simultâneo)
UAV	Unmanned Aerial Vehicles (Veículos aéreos não tripulados)
UGR	Unmanned Ground Robot (Unmanned Ground Robot)
VP	Viticultura de precisão
V-REP	Virtual Robot Experimentation Platform (Plataforma de experimentação de robôs virtuais)
VRT	Variable Rate Technology (Tecnologia de Taxa Variável)

1. Introdução

Com o aumento da população, os agricultores são desafiados a mudar a forma de controlo, monitorização e gestão das suas explorações agrícolas de modo a conseguirem satisfazer a crescente procura de alimentos de alta qualidade.

Felizmente, os avanços científicos em diferentes áreas estão a transformar a forma de gerir as atividades agrícolas, reduzindo cada vez mais a intervenção humana e permitindo aumentar a produtividade com sustentabilidade [1]. Esta evolução tecnológica deu origem ao termo *Internet of Things* (IoT), que permite o controlo de segurança, eficiência operacional, redução de custos e desperdícios, sensibilização e gestão de ativos.

A agricultura de precisão (AP) ajuda a alcançar uma agricultura sustentável com a automação e o envolvimento tecnológico para controlar a utilização imprópria dos recursos. Assim, a AP pode ser considerada como uma estratégia de gestão agrícola eficaz e inteligente, que utiliza tecnologias avançadas para obter dados de campo de múltiplas fontes e tomar decisões relacionadas com a produção de culturas [2].

Tecnologias geo-espaciais, como, *Global Positioning System* (GPS), Mapeamento da Colheita, e *Variable Rate Technology* (VRT) são das vias mais competentes para garantir uma gestão de recursos de modo eficiente. A chamada "Agricultura 4.0" oferece muitas possibilidades, entre elas, drones e outras tecnologias de deteção que podem fornecer informações em tempo real, produzem imagens e capturam diferentes parâmetros agrícolas, como, o estado de fertilidade do solo, o aumento ou risco de pragas e doenças e o desenvolvimento de ervas daninhas [3].

Surge, deste modo, a aplicação da AP à viticultura, subdomínio que tomou a designação de "Viticultura de precisão" (VP), sendo a produtividade deste, extremamente importante e decisiva à qualidade do produto produzido.

A partir de sistemas tecnológicos avançados é possível a conceção de robôs para auxílio de tarefas agrícolas, tornando assim exequível a AP, tal como a criação de um robô de auxílio à vindima, como é o exemplo desta dissertação.

Na dissertação estão incluídas 52 referências no total, tendo sido retiradas de fontes referenciadas em quatro categorias principais, nomeadamente, fontes de revistas de investigação, livros, websites e artigos de investigação nacionais e internacionais. Os estudos das revistas de investigação foram obtidos de diversas revistas, tais como,

"Journal of Robotics", "Sensores", "AgriEngineering", "Biosystems Engineering", "Precision Agriculture", etc. As fontes resultaram da pesquisa em vários repositórios eletrónicos, nomeadamente "ScienceDirect", "Web of Science" e "SpringerLink". A maioria dos artigos selecionados foram publicados no período de 2016 a 2023, mas algumas ocorrências relevantes podem ser mais antigas.

Este capítulo pretende expor os principais objetivos e motivações da dissertação e apresentar a estrutura do trabalho, em capítulos, resumindo-os de uma forma muito sucinta.

1.1. Motivação

Do ponto de vista pessoal, a escolha do tema para a dissertação foi motivada pela noção da importância da agricultura na sobrevivência da humanidade e na manutenção da economia mundial, com especial foco na viticultura, visto a vinha ser uma das culturas de maior importância em Portugal.

Sabe-se hoje que o futuro da agricultura deverá estar intimamente ligado ao uso expressivo de tecnologia. Com efeito, nas últimas décadas têm sido introduzidas tecnologias que melhoram substancialmente a qualidade e eficácia da agricultura. Com a implementação de mecanismos automatizados torna-se possível fazer uma escolha mais certeira da quantidade de pesticidas e herbicidas, monitorizar o estado hídrico e sais minerais presentes no solo e na planta, fazer uma estimativa mais certeira da produção. Desta forma consegue-se melhorar o rendimento económico da atividade agrícola quer pelo aumento da produtividade e/ou qualidade, quer pela redução dos custos de produção, reduzindo também o seu impacto ambiental.

Infelizmente, as tecnologias associadas à agricultura de precisão são quase sempre demasiado complexas e dispendiosas, não estando ao alcance de muitos dos produtores de uva nacionais, com explorações de dimensão insuficiente para justificar tal investimento. Este projeto pretende criar uma ferramenta acessível e eficaz, tratando-se de um robô, que além de ajudar na árdua tarefa de vindimar as vinhas em latada, permitirá monitorizar as parcelas, estimar a produção, realizar a colheita seletiva e fazer o transporte dos cestos.

Esta dissertação foi também motivada pelo facto de ser um projeto que futuramente poderá ter continuidade noutro trabalho final de mestrado do ISEL, estando assim a

contribuir para um projeto que poderá vir a ser importante, quer para o instituto, quer para outros colegas.

1.2. Objetivos

Esta dissertação tem como objetivo principal, elaborar o projeto de um robô móvel para realizar tarefas de auxílio à vindima de vinhas em latada, sistema utilizado largamente na produção de uva de mesa, de modo a resolver alguns problemas que costumam ocorrer na viticultura tradicional, como seja a falta de mão de obra qualificada para esta árdua e sensível tarefa, aumentando a produtividade e qualidade dos produtos produzidos.

Com os avanços das novas tecnologias agrícolas consegue-se otimizar a eficiência da produção, verificar a fertilidade do solo, otimizar a qualidade, minimizar o impacto ambiental e minimizar os riscos associados à produção. Neste contexto, [2] mencionou que a VP, pode ser entendida como a gestão da variabilidade temporal e espacial das parcelas com o objetivo de melhorar o rendimento económico da atividade agrícola, quer pelo aumento da produtividade e/ou qualidade, quer pela redução dos custos de produção, reduzindo também o seu impacto ambiental e risco associado.

A colheita, transporte e armazenamento da uva de mesa, ao contrário da uva para vinho, exige enorme cuidado e delicadeza, de modo a chegar ao consumidor com boa qualidade. É preciso ter em conta numerosos fatores como o tamanho das bagas, a coloração da casca e o teor de açúcar para determinar o ponto de maturação ideal para a colheita, sendo que essas características variam com as castas.

A sua colheita manual exige cuidados de higiene e manuseio de modo a garantir a qualidade das uvas. Os cachos devem ser colhidos num corte rente ao ramo de produção, é necessário evitar o contacto dos bagos com as mãos segurando sempre o cacho pelo pedúnculo. Os cachos devem ser cuidadosamente acondicionados em caixas próprias, distribuídas ao longo das linhas de plantio, evitando o seu contato direto com o solo. O transporte tem de ser cuidadoso para evitar choques, raspões ou batidas que iriam danificar os cachos, estes devem ser mantidos à sombra até serem transportados para o local de embalagem.

Ainda não existe nenhum equipamento que esteja devidamente adaptado a este tipo de colheita, por isso, o objetivo desta dissertação é conceber um robô que efetue tarefas como monitorizar as vinhas, estimar a produção, realizar a colheita seletiva e fazer o transporte

dos cestos de vinhas em latada para produção de uva de mesa. O robô será constituído por um sistema de locomoção de *4 Wheel Drive* (4WD), sensores de mapeamento, visão computacional, sensores de localização e possuir braços robóticos para permitir a realização da vindima, ou posteriormente, adaptar outros tipos de garras para executar outras tarefas tais como a pulverização de herbicidas e a poda.

Para realizar o design e estudos estáticos do robô será utilizado o software SolidWorks. Este software oferece o ambiente necessário para o desenvolvimento do robô e a possibilidade de desenvolver estudos à sua estrutura, de modo a garantir a fiabilidade do mesmo.

Uma vez que o sistema de orientação depende do input visual, foi necessária uma simulação gráfica interativa para avaliar o seu desempenho. Foi desenvolvido um cenário em CoppeliaSim, um simulador que proporciona a capacidade de simulação precisa e eficiente de robôs em ambientes complexos interiores e exteriores.

1.3. Estrutura do documento

Esta dissertação está organizada em seis capítulos. O primeiro capítulo introduz o leitor sobre o conjunto de temas que serão abordados ao longo do documento. No capítulo seguinte é feita uma revisão bibliográfica sobre AP e mais em detalhe à VP, sendo também referidos os diversos robôs existentes para culturas diferentes.

No terceiro capítulo, são explorados os desafios da implementação da automação na AP e VP. Além disso, são discutidos os detalhes da disposição das vinhas para a produção de uvas de mesa, com destaque para a apresentação do robô desenvolvido. Este robô é analisado em detalhe, abordando os seus principais componentes, materiais utilizados e as suas propriedades específicas. O objetivo é proporcionar uma compreensão abrangente do sistema robótico, incluindo o seu funcionamento e os desafios associados à sua aplicação em ambientes de VP.

No quarto capítulo são realizados estudos estáticos ao robô, de modo mais detalhado na sua estrutura, de maneira a compreender como este se irá comportar em campo e garantir a sua fiabilidade.

No quinto capítulo, é realizada uma análise dos simuladores disponíveis no mercado, explicando a seleção e apresentando uma introdução completa ao seu funcionamento.

Além disso, é apresentada uma modelação detalhada do sistema, juntamente com a descrição do algoritmo correspondente. Em seguida, realiza-se a simulação de três cenários específicos, seguida por uma análise crítica e uma discussão profunda dos resultados obtidos.

Para finalizar, são realizadas observações finais, bem como, possíveis trabalhos futuros de modo a dar continuidade ao presente trabalho final de mestrado.

2. Estado da Arte

De acordo com a motivação e os objetivos desta dissertação, este capítulo apresenta a agricultura e viticultura de precisão, os seus benefícios e como se apresentam no mercado. Visto existirem sistemas robóticos adaptados a diversas culturas com características variadas, neste capítulo também se apresenta os resultados da investigação sobre os sistemas existentes no mercado, fazendo uma comparação entre estes, tendo em conta as suas vantagens e limitações.

2.1. Implementação da robótica na agricultura

A implementação da tecnologia robótica na agricultura pode ser rentável a longo prazo, pois esta pode aumentar a eficiência e a produtividade, reduzir os custos de mão de obra, reduzir o desperdício e melhorar a qualidade dos produtos. No entanto, a curto prazo, os custos de investimento em sistemas de automação podem ser altos e podem limitar a adoção da tecnologia por pequenos agricultores ou produtores com recursos financeiros mais limitados.

Além disso, o retorno do investimento em sistemas de automação na agricultura depende do tipo de cultivo, do tamanho da propriedade, das condições climáticas e da infraestrutura existente. Em algumas áreas, onde a mão de obra é barata e abundante, ou onde as condições climáticas são favoráveis, pode não ser economicamente viável investir em tecnologias de automação sofisticadas. Em outras áreas, onde a mão de obra é escassa ou cara, ou onde as condições climáticas são adversas, a automação pode ser mais rentável e justificável.

Outro fator que afeta a rentabilidade da automação na agricultura é a capacidade dos sistemas de automação para se adaptarem às necessidades específicas de cada propriedade ou cultivo. Os sistemas de automação bem projetados podem ser adaptados para diferentes tipos de cultivo e terrenos, permitindo que os agricultores maximizem sua eficiência e produtividade. No entanto, os sistemas de automação que são muito específicos ou que exigem muita manutenção podem ser mais difíceis de justificar em termos financeiros.

Em resumo, a implementação da tecnologia robótica na agricultura pode ser rentável, mas isso depende de vários fatores, incluindo o tipo de cultivo, o tamanho da propriedade, as

condições climáticas e a capacidade dos sistemas de automação para se adaptarem às necessidades específicas de cada propriedade ou cultivo.

2.2. Tipos de Automação

Na era da Indústria 4.0, a automação está cada vez mais presente no nosso dia a dia, principalmente para operações de fabrico, armazenamento e apoio a diversas tarefas agrícolas. Tecnologias como empilhadoras não tripuladas, máquinas de pulverização autónomas, transportes inteligentes, entre outras estão a avançar e os custos a diminuir, tornando estas opções mais acessíveis para empresas de todas as dimensões.

A chegada dos *Autonomous Mobile Robot* (AMR) que têm atraído muita atenção como alternativas mais inteligentes e flexíveis do que os *Automated Guided Vehicle* (AGV). Para se concluir qual destas opções se adequa melhor ao tipo de tarefas do caso de estudo deste trabalho, primeiro é necessário compreender a diferença entre estas duas alternativas de automação.

Embora os AGV e os AMR partilhem muitos casos de utilização, a tecnologia distingue-se pelo custo, eficiência e flexibilidade.

Os AMR são veículos que utilizam sensores de bordo para identificar e evitar obstáculos tanto em movimento como estáticos que surgem no seu caminho, enquanto navegam no espaço de trabalho. Os AMR utilizam software sofisticado de mapeamento, onde se podem estabelecer as rotas e um sistema de gestão de frota para determinar quais as melhores rotas a seguir. Por outro lado, os AGV requerem referências fixas, instaladas aquando da sua configuração inicial e definem os caminhos a seguir. Estas referências devem ser alteradas sempre que se pretende alterar um caminho ou existe alterações do espaço de trabalho.

Os AGV custam tipicamente menos do que os AMR, no entanto é necessário ter em consideração os custos de instalação, reconfiguração e operação. Os AGV requerem normalmente a instalação de guias físicas, cabos sob o pavimento, ou fita adesiva de superfície para permitir que o AGV navegue e se localize nos seus arredores [4].

O AGV funcionará num percurso definido, incapaz de se mover em torno de obstáculos inesperados que se atravessem no seu caminho. Muitas vezes, o AGV irá parar completamente até o caminho estar completamente desobstruído. Caso o seu ambiente ou

tarefa mude, incorrerá em custos adicionais para reconfigurar o seu sistema e repetir o processo de implementação.

Em contraste, um AMR não requer alterações às instalações existentes, e pode navegar autonomamente pelos espaços. Aprende o seu ambiente, "lembra-se" da sua localização e planeia dinamicamente o seu próprio caminho de um destino para outro. Se o percurso estiver bloqueado, um AMR pode redirecionar-se a si próprio sem assistência [5].

Além disso, enquanto os AGV são maioritariamente veículos de rodas, os AMR possuem diferentes tipos de locomoção. Isto inclui não só veículos com rodas, mas também veículos com lagartas, ou com locomoção por pernas.

A solução mais apropriada a ser utilizada no sistema robótico a desenvolver nesta dissertação, foi um AMR.

2.3. Agricultura de precisão

Como já foi referido no capítulo anterior, a agricultura de precisão é uma estratégia de gestão que utiliza tecnologias de informação para recolher dados úteis de fontes distintas, com o objetivo de apoiar as decisões associadas à produção de culturas. Consiste em reconhecer, localizar, quantificar e registar a variabilidade espacial e temporal de cada unidade agrícola.

Um dos potenciais benefícios da AP reside na eficiente utilização dos recursos, resultando em vantagens nos domínios económico, social e ambiental. Esta eficiência traduz-se na redução de investimentos desnecessários e na mitigação do risco de contaminação ambiental. Este processo é alcançado através da aplicação de fertilizantes apenas nos locais e momentos necessários, com base em amostras de solo e na análise dos dados de produção prevista. Adicionalmente, a gestão dos recursos hídricos é otimizada mediante a implementação de práticas de colheita automatizada. No entanto, é crucial ter em consideração os custos associados à adoção da AP, visto que algumas tecnologias requerem um investimento substancial [6].

O processo de agricultura de precisão está dividido em três fases distintas que ocorrem num padrão cíclico, demonstrado na Figura 1. A fase de análise e plano de tratamento de dados, a fase de aplicação do tratamento de precisão, e a fase de monitorização e recolha de dados. [7]

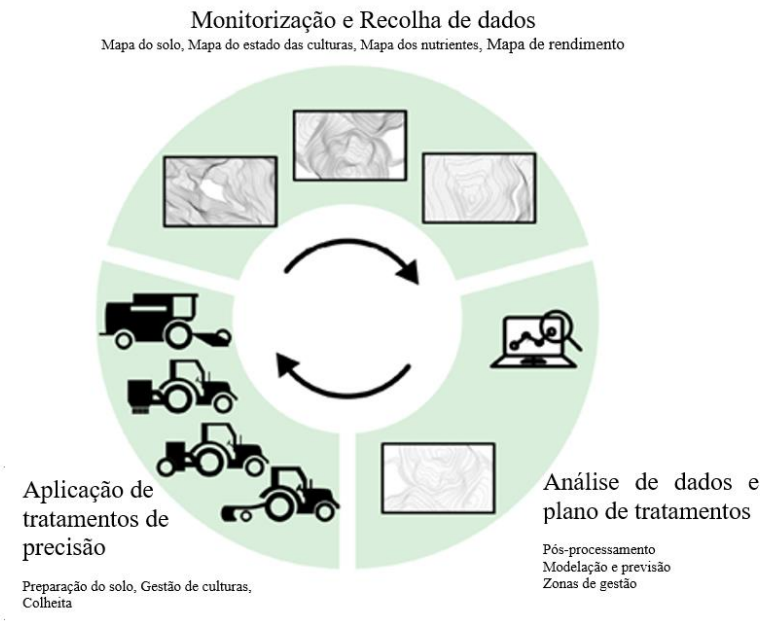


Figura 1 - Ciclo da Agricultura de Precisão e suas fases principais, adaptado de [7]

A AP envolve dados intensivos e diversas tecnologias, mas a aplicação do seu conceito e princípio não requer equipamento pesado ou campos de grandes dimensões, podendo ser personalizada e utilizada numa variedade de situações. As características essenciais exigidas pelos veículos e sistemas agrícolas autónomos que funcionam no terreno tendem a enquadrar-se em quatro categorias.

- Navegação: O veículo deve mover-se no campo autonomamente, seguindo um caminho, pontos-chave de passagem, e evitando quaisquer obstáculos ou colisões.
- Sensoriamento: O veículo deve ser capaz de detetar, medir e recolher amostras que possam ser relevantes no planeamento de operações de cultivo e gestão do solo.
- Mapeamento: As atividades de sensoriamento geram uma grande quantidade de dados, que são geralmente processados pelos Sistemas de Informação Geográfica (SIG), que são mapas que recolhem todas as características essenciais do campo.
- Ação: O mapeamento cria inúmeras zonas de gestão, e depois o tratamento necessário é realizado em cada zona. Como resultado, os veículos de AP devem estar equipados para realizar tais operações autonomamente.

A navegação de veículos autónomos depende de todas as tecnologias que permitem a um robô localizar-se numa área desconhecida e mapeá-la, um processo conhecido como *Simultaneous Localization And Mapping* (SLAM). Como resultado, a navegação autónoma requer a integração de um sistema de localização, como o sistema de *Global*

Navigation Satellite System (GNSS), com uma variedade de sensores exteriores (por exemplo, sistemas baseados em imagem, sonares, radares, *Light Detection And Ranging* (LIDAR), entre outros, para efetuar simultaneamente a localização e o mapeamento. Ao analisar os dados fornecidos por estas soluções, o mapeamento dos dados pode proporcionar aos agricultores informações valiosas sobre recursos importantes e orientação sobre a quantidade e o momento adequado para a sua utilização [7].

Este é o cerne da AP e requer um profundo entendimento dos dados sensoriais obtidos, a fim de analisá-los de forma apropriada e determinar as ações necessárias. Atualmente, estão a ser desenvolvidos sensores mais avançados, devido à complexidade desta tarefa e à crescente popularidade da AP. Geralmente, esta evolução implica uma maior automatização e a aplicação local de ações agrícolas mecanizadas tradicionais [8]. Como resultado, as aplicações autónomas costumam ser especializadas para um conjunto limitado de atividades. O objetivo da deteção na AP, independentemente da tecnologia utilizada, é avaliar o estado do solo e das culturas (como o pH do solo, teor de humidade, composição de nutrientes, estado nutricional da cultura e rendimento esperado), que podem ser realizados remotamente ou no local.

2.4. Robótica na Agricultura

Os avanços científicos e tecnológicos nas áreas da robótica móvel, visão computacional e inteligência artificial, permitem o desenvolvimento de atividades agrícolas mais precisas, económicas e ambientalistas.

Tarefas como a colheita, poda e pulverização, são áreas que têm sofrido melhorias substanciais ao tornarem-se mais autómatas pois, quando realizadas manualmente envolvem maiores custos e promovem a aplicação excessiva de produtos químicos dispendiosos, aumentando os impactos ambientais e os investimentos realizados. Monitorizar e estimar a produção das culturas, permite ainda ao produtor ter um maior controlo sobre a sua produção, identificando possíveis pragas/doenças, que são difíceis de detetar a olho nu. O investimento em sistemas robóticos agrícolas permite alcançar a monitorização de colheitas, a curto prazo, e a estimativa do rendimento a longo prazo [9].

Neste sentido, serão abordados vários tipos de robôs associados a diversas plantações agrícolas e com funcionalidades diferentes. Deste modo os robôs foram agrupados em diversas atividades agrícolas, tais como, preparação do solo antes da plantação,

sementeira, tratamento das plantas (controlo mecânico ou químico de plantas, poda, identificação de doenças, etc.), colheita e estimativa do rendimento e fenotipagem.

Uma das primeiras tarefas agrícolas a ser realizada, é a preparação do solo antes da plantação com o arado da terra e a aplicação de fertilizantes. Com o arar da terra permite-se o arejamento do solo. Já a fertilização do solo consegue fornecer os nutrientes necessários para o desenvolvimento das culturas [10].

As operações de semear e plantar são realizadas por um equipamento especializado, acoplado a um trator. No entanto, devido ao elevado peso dos tratores e à sua constante deslocação pela exploração agrícola, a compactação do solo é acentuada, resultando em diversos efeitos prejudiciais. Estes incluem o aumento da densidade do solo, a diminuição da sua porosidade e da taxa de infiltração de água, bem como a alteração das propriedades químicas do solo. Estas consequências afetam negativamente o crescimento das plantas e a biodiversidade do solo [11].

Após terminar a fase de plantação, manter os produtos agrícolas saudáveis e permitir um crescimento da plantação de forma adequada, ou seja, livre de doenças e pragas até à colheita, requer um controlo constante por parte do agricultor. É necessário cuidar da plantação para que não surjam doenças e que se espalhem por todas as culturas, tornando a colheita inviável.

Segundo a *Food and Agriculture Organization* (FAO), cerca de 20 a 40% da produção agrícola mundial é perdida devido a pragas e doenças. O crescimento de ervas daninhas pode destruir completamente as culturas e atrair pragas e pequenos animais como cobras e ratos. Os robôs podem efetuar a deteção das ervas daninhas e posterior remoção através da aplicação de herbicidas e pesticidas, de ferramentas mecânicas, ou térmicas. O tratamento de plantas é normalmente realizado através da aplicação de herbicidas e pesticidas (inseticidas e fungicidas) [3].

Este conceito pode ser generalizado dividindo o controlo de ervas daninhas de duas formas distintas: a utilização de ferramentas mecânicas e a utilização de produtos químicos (herbicidas).

A poda é outra tarefa bastante importante para um robô executar, apesar da sua complexidade. Os principais desafios deste tipo de tarefa consistem em digitalizar e medir as estruturas das plantas para conhecer o local exato de poda.

São poucos os robôs que foram construídos para fins gerais, contudo seria uma mais valia desenvolver um robô com uma grande variedade de aplicações, de modo a ser capaz de estabelecer um processo de padronização e modularização de sistemas robóticos na AP.

A atividade de colheita, além de ser uma tarefa repetitiva e requerer uma execução ágil, exige também muito esforço por parte do operário. Segundo um estudo realizado por [12], as operações de colheita representam cerca de 25% das horas de trabalho agrícola. Em 2014, na Austrália, os custos financeiros, representam entre 20 e 30% dos custos totais. Já em 2019 na China, representavam cerca de 75% dos custos de produção e a tendência era de um aumento anual. Esta diferença avassaladora, deve-se ao fato das culturas australianas terem um nível de automatização mais avançado. Devido a estes fatores, têm vindo a ser realizados vários sistemas robotizados para a realização de atividades de colheita.

A estimativa de produção é a monitorização de toda a cultura e a estimativa, através de dados precisos sobre a quantidade e qualidade do crescimento dos produtos, fornecidos por ferramentas mais sofisticadas. Variáveis como as alterações climáticas e a qualidade do solo podem interferir como desenvolvimento da planta. Como tal, ao se identificar o fenótipo das plantas, é possível identificar as condições de crescimento adequadas [13].

O fenótipo vegetal pode ser retirado de várias maneiras, tais como, monitorizando a altura, peso, biomassa, forma, cor, volume, absorção de luz e temperatura da planta. Para se conseguir estimar tais valores, é necessário que o robô possua dados sensoriais fiáveis e algoritmos eficientes de visão por computador.

A partir da Tabela 1 consegue-se observar as diferenças entre diversos robôs atualmente presentes no mercado para AP e respetivas tarefas que desempenham.

Tabela 1 - Comparação entre as diversas tarefas realizadas dos robôs existentes na AP

Robô/Tarefas	Preparação do solo	Semear e plantar	Identificação de doenças	Monda Mecânica	Monda Química	Poda	Colheita	Estimativa de produção	Fenotipagem	Cultura	REF
Greenbot	X									Horticultura, fruta e agricultura	[14]
AGRAS MG-1P					X					Arroz, soja e milho	[15]
AgBot	X	X			X		X			Algodão, soja, milho, Vinhedos	[10]
Di-Wheel		X								Horticultura no geral	[16]

Disease Robot		X							Pimentão	[17]
eAGROBOT		X				X	X		Frutas delicadas	[18]
Dino			X						Vegetais	[19]
Ted			X						Vinha	[20]
VITIROVER			X						Relva	[21]
SAVSAR				X					Vinha	[22]
AgriRobot				X					Vinha	[23]
RIPPA	X			X	X				Alface, couve-flor e brócolos	[24]
Harvey platform						X			Pimentão doce	[25]
Vegebot						X			Alface	[26]
SWEEPER						X			Morangos	[27]
Amaran						X			Coco	[28]
VINBOT							X	X	Vinha	[29]
VineRobot							X	X	Vinha	[30]
Vinoculer								X	Milho	[30]
TerraSentia		X						X	Milho	[31]

2.5. Aplicação da Robótica na Viticultura

A VP, tal como a AP, tem como objetivo melhorar o rendimento económico da atividade agrícola quer pelo aumento de produtividade e/ou qualidade quer pela redução dos custos de produção, reduzindo, por conseguinte, o impacto ambiental e risco associado.

A rápida evolução das tecnologias oferece um enorme potencial para o desenvolvimento de soluções otimizadas para a viticultura. Desenvolvimentos tecnológicos recentes permitiram a elaboração de sistemas que ajudam na monitorização e controlo de vários aspetos do crescimento da vinha.

Num contexto de concorrência nos mercados, é cada vez mais importante atingir padrões de qualidade mais elevados, o que leva a uma renovação da viticultura e técnicas agrícolas utilizadas. Consegue-se assim maximizar a qualidade e sustentabilidade através da redução e utilização mais eficiente dos fatores de produção, tais como, energia, fertilizantes e produtos químicos. Com a minimização dos custos de produção reduz-se também o impacto ambiental [32].

As tecnologias no domínio da viticultura têm sofrido um desenvolvimento rápido e uma maior aplicabilidade, devido aos seus custos reduzidos, facilidade de utilização e versatilidade. Com a utilização das soluções inovadoras consegue-se obter uma redução de custos na gestão de culturas, um aumento na qualidade e na produção, e melhoria na rastreabilidade dos processos e da sustentabilidade ambiental com uma utilização racional dos insumos químicos.

2.5.1 Exemplos de Robôs existentes

Muitos dos robôs para viticultura ainda se encontram numa fase de protótipos, mas vários projetos já foram colocados no mercado, inclusive projetos de faculdades portuguesas. Seguem-se algumas das tecnologias existentes, mais significativas, para várias tarefas.

Empresas francesas desenvolveram dois robôs, que são capazes de realizar a monda (eliminação de ervas daninhas). O “TED” foi o primeiro robô elétrico capaz de conduzir de forma autónoma e retirar as ervas daninhas da vinha enquanto areja também o solo. Espera-se que, no futuro, seja capaz de realizar outras tarefas, como a pulverização, recolha de dados e desbaste de rebentos [20]. Da mesma forma, o "Bakus" é um robô com capacidade para realizar a monda mecânica nas vinhas, e eventualmente será equipado com a capacidade de pulverização (Figura 2).

Figura 2 - Representação do robô Bakus

Existem várias entidades que têm veículos terrestres totalmente autónomos em desenvolvimento, com a função de pulverização de pesticidas e herbicidas, mas poucos são os que se encontram já disponíveis comercialmente [33].

SAVSAR, Semi-Autonomous Vineyard Spraying Agricultural Robot, desenvolvido por [22], é um robô de pulverização que tem a possibildiade de funcionar tanto em modo autónomo, como operado virtualmente ou misto. No modo misto, o robô funcionará de forma autónoma até encontrar uma situação em que a intervenção humana seja necessária (Figura 3).

Figura 3 - Representação do robô SAVSAR

Outro projeto é o GRAPE, Ground Robot for VineyArd Monitoring and ProtEction [34]. Este é um robô semiautónomo e multifuncional que inclui a função de pulverização de pesticidas e outros químicos, podendo também realizar a monitorização das plantas (Figura 4).

Figura 4 - Representação do robô GRAPE

A empresa Blue River Technology, criou uma tecnologia “See & Spray” que possui aprendizagem na identificação das plantas infestantes ao longo do tempo e visão por computador para pulverizar herbicida apenas sobre as ervas daninhas, permitindo reduzir a quantidade de pesticidas em 90% [35]

Além dos sistemas robóticos terrestres, os *Unmanned Aerial Vehicles* (UAV) também desempenham um papel relevante na automatização da monitorização de culturas, fazendo uso de câmaras e outros sensores que têm a capacidade de produzir simulações de maneira mais eficaz do que as imagens obtidas por satélite.

Existem vários veículos aéreos UAV disponíveis no mercado, que são utilizados atualmente para a pulverização de culturas. Os drones são capazes de detetar as ervas daninhas através de métodos de deteção de luz, para realizar uma pulverização mais rápida e eficiente do que um pulverizador tradicional [33]. O DroneAG e o RMAX da Yamaha são exemplos destes drones disponíveis comercialmente (Figura 5).

A pulverização por UAV para além de ser uma aplicação mais segura, rápida e eficiente também evita a compactação do solo, proporciona uma maior flexibilidade e acessibilidade aos campos, principalmente em encostas íngremes, além de evitar a exposição dos trabalhadores à pulverização de químicos, que acontece com os métodos manuais [36].

Figura 5 - Representação do RMAX da Yamaha

Quando se trata de operações vinícolas, a poda é uma das atividades que acarreta maiores despesas e, como tal, tem-se tornado uma das áreas mais visadas para a automação. A poda manual é uma tarefa repetitiva que resulta na fadiga dos trabalhadores e é bastante dispendiosa. Ainda não é possível encontrar no mercado um robô que proporcione uma poda mecânica, no entanto há vários que estão em desenvolvimento.

Em 2012, a Vision Robotics desenvolveu uma podadora que utiliza a modelação para identificar que videiras ou ramos que precisam ser aparados. Também em 2012, o robô Wall-Ye foi comercializado tendo as capacidades de realizar a poda, realizar a colheita e controlar a produção vinícola. Este robô está equipado com seis câmaras, é capaz de

efetuar um corte de 5 em 5 segundos e tem uma capacidade de poda autónoma que lhe permite funcionar continuamente entre 10 a 12 horas [37].

O processo de remoção de plantas infestantes é um dos trabalhos mais extenuantes de um viticultor. Estima-se que os trabalhadores se dobram 3000 a 4000 vezes por dia enquanto efetuam esta tarefa. Não existem muitas tecnologias que efetuem esta tarefa, mas o Wall-Ye e o TED, ambos mencionados anteriormente, têm a capacidade de extração mecânica das plantas infestantes (Figura 6).

Figura 6 - Representação dos robôs Wall-Ye e TED

A monitorização das culturas e estimativa do rendimento são dos aspetos mais procurados na viticultura de precisão, pois permite determinar se as uvas estão prontas para a vindima e calcular o rendimento em cada ano. Vários projetos robóticos estão em desenvolvimento para ajudar a melhorar a eficiência e aumentar o rendimento das vinhas.

O Vinbot, apresentado na Figura 7, é um projeto que foi desenvolvido entre 2014 e 2016 por nove organizações europeias onde se inclui o Instituto Superior de Agronomia, e é descrito como um robô móvel autónomo todo-o-terreno com um conjunto de sensores capazes de capturar e analisar imagens de vinhas e dados 3D, utiliza visão por computador para medir o rendimento das vinhas e partilhar a informação com os viticultores. Possui localizadores de gama 3D para navegar entre os campos de vinha e estimar a quantidade de folhas, uvas e outros dados sobre as vinha, permitindo aos viticultores otimizar a produção e gestão das suas vinhas [38].

Figura 7 - Representação do robô Vinbot

Em 2015, surgiu um protótipo de robô de monitorização de vinhas conhecido como VineRobot. Agora, o seu sucessor foi revelado em Portugal, com o nome de VineScout. Este projeto é análogo ao Vinbot, empregando técnicas de inteligência artificial, sensores avançados e energia solar para monitorizar o crescimento das plantas e recolher dados sobre a maturação dos frutos [39]. Adicionalmente, os sensores incorporados permitem a monitorização do estado hídrico e da composição das uvas e do solo (Figura 8).

Figura 8 - Representação do robô VineScout

Atualmente existem várias máquinas agrícolas para efetuar a vindima mecânica de uvas para vinho, como as Braud (Figura 9), que consiste em vibrar as videiras de modo a libertar as uvas maduras, no entanto, é frequente que na recolha juntamente com o fruto esteja também presente o que é conhecido como *Material other than grapes* (MOG). Visto que estas máquinas não inspecionam a qualidade, este MOG pode incluir uvas apodrecidas, folhas, caules e sementes [40].

Figura 9 - Representação da máquina agrícola Braud

Para resolver esta problemática, os robôs completamente autónomos, habilitados para a tarefa de vindima, necessitam de estar equipados com sistemas de processamento de imagem avançados, capacidades de manipulação e técnicas de aprendizagem profunda. Os sistemas de processamento de imagem devem ser capazes de discernir entre cachos de frutos maduros e não maduros, além de poderem orientar o braço manipulador para o caule adequado a ser cortado.

A realização da vindima de uvas de mesa, tem de ser efetuada com cuidado para não danificar as uvas, como tal, atualmente esse tipo de vindima é realizado manualmente, sem ajuda de mecanismos automatizados.

Clever Robot for Crops (CROPS) é um consorcio europeu de 13 empresas privadas e universidades que trabalham em conjunto com o objetivo de criarem um robô para a colheita de diversas culturas, entre elas a vinha. Utilizaram tecnologia de fluorescência para determinar a maturação dos frutos e imagem multiespectral para deteção, classificação e localização dos frutos a serem colhidos e possíveis doenças na cultura [41].

Embora ainda estejam a fazer testes na colheita em maçãs, a *Abundant Robotics*, está a criar um projeto que utiliza a visão por computador e um sistema de vácuo (em vez de pinças) para remover suavemente os frutos. A empresa espera conseguir expandir a sua tecnologia para outras culturas, inclusive a vinha [42].

Na Tabela 2 encontra-se a comparação entre as diversas tarefas realizadas dos robôs existentes para VP.

Tabela 2 - Comparação entre as diversas tarefas realizadas dos robôs existentes para viticultura

Robô/Tarefas	Monda	Pulverização	Poda	Remoção de ervas daninhas	Monitorização	Vindima
Ted	x					
Bakus	x					
SAVSAR		x				
GRAPE		x				
DroneAG		x				
RMAX		x				
Wall-Ye			x	x		
TED				x		
Vinbot					x	
VineScout					x	
Braud						x

A partir da tabela acima representada consegue-se perceber que atualmente não existe nenhum robô para VP que esteja preparado para realizar mais de 2 tarefas.

Uma das tarefas mais laboriosas durante o processo de vindima é a realização da triagem, que consiste na classificação das uvas por qualidade e maturação. Uma vez que os mecanismos de vindima atuais, não conseguem distinguir as uvas saudáveis daquelas que podem estar verdes ou podres, a triagem após colheita é essencial para assegurar a alta qualidade do produto final. Existem atualmente duas máquinas de triagem de uvas no mercado a GrapeSort Optical Sorter e Delta Oscillys.

A GrapeSort Optical Sorter, é uma tecnologia de seleção automática de uvas (Figura 10), que utiliza uma câmara de alta velocidade para determinar o teor de açúcar de cada uva, esta tecnologia permite separar MOG como folhas, ramos e insetos, das uvas e também permite aos viticultores separar as uvas de acordo com o grau de qualidade.

O Delta Oscillys faz igualmente uso de câmaras de alta velocidade, bem como de vibração e fluxo de ar preciso, para separar as uvas do MOG. O viticultor define imagens de uvas "perfeitas" e o sistema de visão por computador compara cada uva no tapete rolante com

a imagem ideal. Com um pequeno sopro de ar remove qualquer uva ou MOG que não corresponda às imagens programadas.

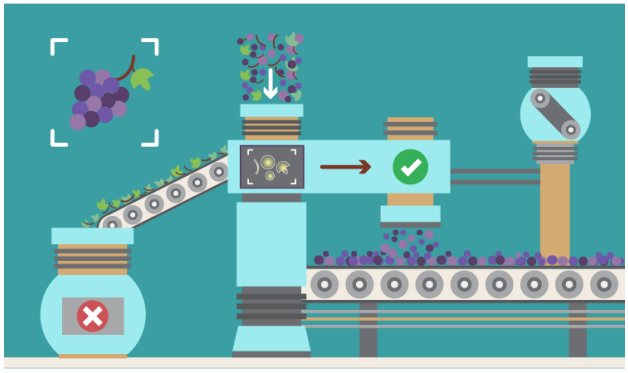


Figura 10 - Representação gráfica da tecnologia de seleção automática de uvas

2.5.2 Limitações dos equipamentos já existentes no auxílio à vindima

De uma forma geral, as principais limitações podem ser agrupadas em quatro domínios essenciais: sistemas de locomoção, sensores, algoritmos de visão computacional e tecnologias de comunicação.

Segundo Oliveira [6], a maioria dos robôs agrícolas recorre a sistemas de 4WD como ilustrado na Figura 11. No entanto, é importante salientar que o ambiente agrícola é considerado como sendo semiestruturado, o que torna os robôs 4WD suscetíveis às características locais do terreno, como rochas e ramos. Além disso, o movimento constante destes robôs por toda a exploração agrícola resulta numa elevada compactação do solo. No que diz respeito aos UAV, melhorias na autonomia do tempo de voo podem potencialmente aumentar a sua aplicação em ambientes agrícolas.

Uma alternativa para a locomoção em ambientes não estruturados consiste na adoção de robôs com membros articulados, como sugerido por [43]. Estes robôs apresentam vantagens intrínsecas, uma vez que não necessitam de um contacto contínuo com o solo para se deslocarem e podem ajustar a sua postura de acordo com a inclinação do terreno, o que lhes permite navegar em ambientes severos e de difícil acesso [44]. De acordo com a Figura 11, o sensor mais utilizado nos trabalhos analisados é a câmara *RGB* (32,23%). Apesar das câmaras *RGB-D* térmicas, hiper-espectrais e multiespectrais fornecerem mais informação (profundidade, temperatura e mais dados espectrais), têm um custo financeiro mais elevado. O compromisso entre o custo financeiro e a qualidade devem ser escolhidos de acordo com os requisitos mínimos do sistema a desenvolver, porque num ambiente

agrícola, a temperatura, humidade e incidência de poeira podem diretamente interferir com o bom funcionamento dos sensores.

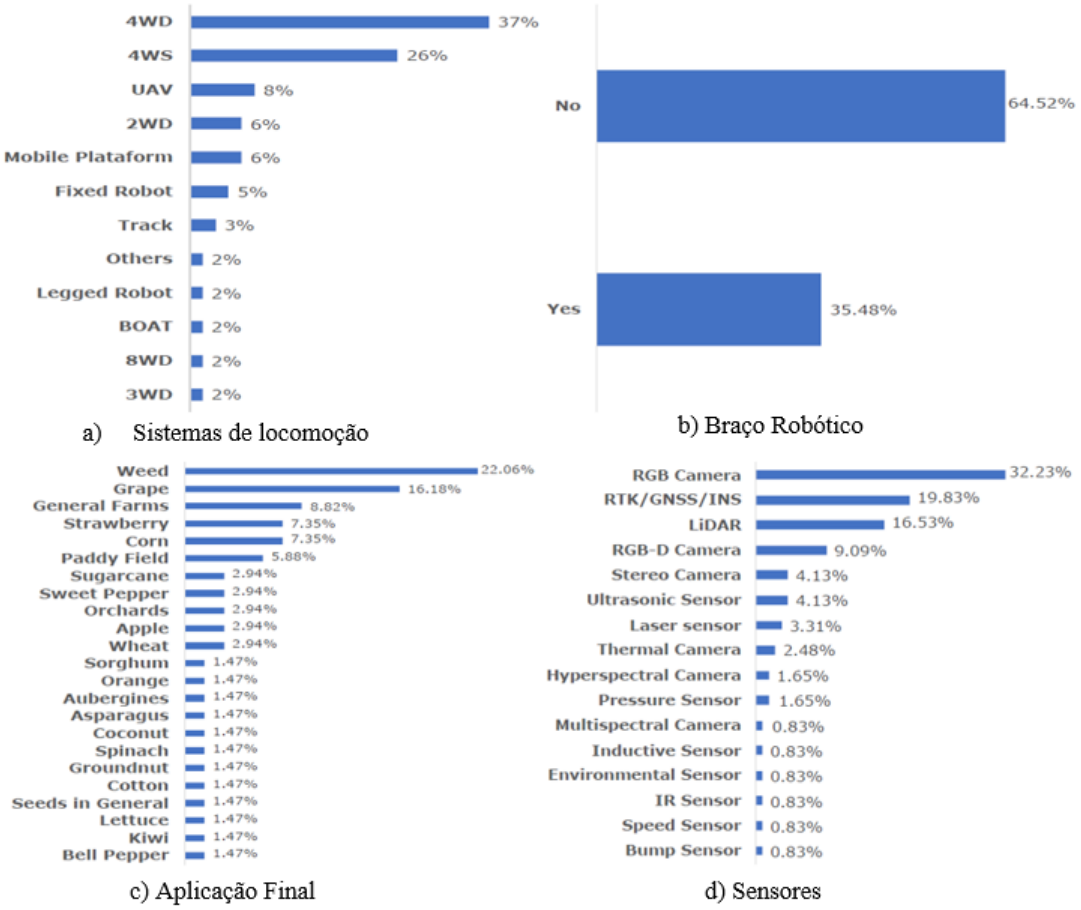


Figura 11 - Resumo de robôs agrícolas

Através da extração das características das culturas (ExG-ExR, NDVI, fluorescência à base de clorofila e imagens RGB/térmicas/Hiper espectrais/multiespectrais), a utilização de algoritmos de inteligência artificial (MLP, CNN, R-CNNN, R-YOLO e SVM) permite a identificação de doenças, deteção de ervas daninhas, aplicação seletiva de herbicida/pesticida, localização das frutas, classificação de maturação (maduro/verde) e estimativa de rendimento. Contudo, atualmente, 35,48% dos robôs existentes ainda não possuem de algoritmos de inteligência artificial Figura 12 e) [6].

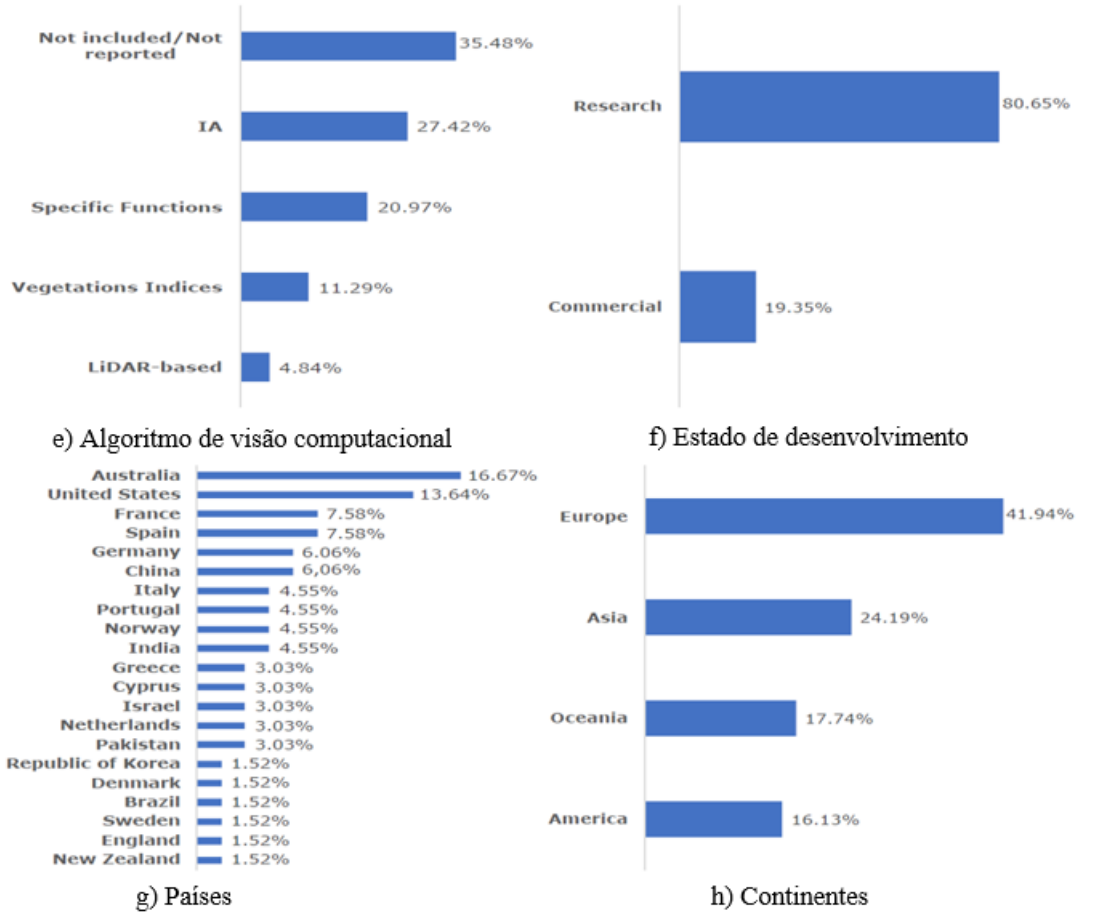


Figura 12 – Continuação do resumo de robôs agrícolas

Através de uma revisão sistemática de diversas aplicações de robôs agrícolas, bem como de uma análise progressiva da presença de robôs na agricultura de precisão e na viticultura de precisão, foi possível constatar que existem diversos robôs destinados a realizar uma ampla gama de tarefas. Estas tarefas incluem desde a preparação do terreno, sementeira, plantação, tratamento de plantas, poda e colheita.

Após uma análise cuidadosa das características que tornam esses sistemas mais adequados para tarefas específicas, levando em consideração também as suas limitações, surgiu a conceção de um robô com melhorias em sistemas cruciais. Estas melhorias abrangem os sistemas de locomoção, navegação, visão computacional e braços manipuladores, com o objetivo de otimizar a eficiência na execução das tarefas agrícolas.

2.6. Principais Componentes

A robótica é a intersecção da ciência, engenharia e tecnologia que produz máquinas, chamadas robôs, que substituem (ou replicam) as ações humanas.

Os robôs foram originalmente construídos para lidar com tarefas monótonas (como construir carros numa linha de montagem), mas desde então expandiram muito para além dos seus usos iniciais executando tarefas como o dar apoio a práticas agrícolas, limpar casas e ajudar com cirurgias. Cada robô tem um nível de autonomia diferente, desde os robôs controlados por humanos que executam tarefas que um humano tem controlo total até aos robôs totalmente autónomos, que executam tarefas sem quaisquer influências externas [45].

Existem duas categorias principais de robôs, os independentes e os dependentes. Os robôs independentes são capazes de funcionar de forma totalmente autónoma e independente do controlo do operador humano. Requerem tipicamente uma programação mais intensa, mas permitem que os robôs tomem o lugar dos humanos quando realizam tarefas perigosas, ou humanamente impossíveis, ou tarefas repetitivas, que realizadas por robôs, permitem que o humano fique disponível para realizar outras tarefas mais importantes. Os robôs independentes provaram ser perturbadores para a sociedade pois eliminam empregos, mas com isso apresentam novas possibilidades de crescimento das empresas [46].

Por outro lado, os robôs dependentes são robôs não autónomos que interagem com os humanos para melhorar e complementar as ações já existentes. Esta nova tecnologia está em constante expansão para novas aplicações, mas os exemplos mais comuns que têm sido realizados, são as próteses avançadas que são controladas pela mente humana.

A segurança é um aspeto crítico na implementação de robôs móveis para tarefas de apoio à vindima. É importante garantir que o robô possa operar com segurança no ambiente de trabalho, protegendo tanto as próprias plantas e frutas quanto o operador que pode estar presente no campo durante a operação do robô.

Para garantir a segurança das plantas e frutas, o robô deve ser projetado de forma a minimizar o impacto nas vinhas, evitando a danificação das videiras e cachos de uvas. Isso pode ser alcançado através do uso de sensores que detetam obstáculos, como as próprias plantas, e ajustam o movimento do robô para evitar colisões.

Além disso, o robô deve ser projetado de forma a minimizar o risco de contaminação das uvas durante o processo de colheita. Isso pode ser alcançado através do uso de materiais adequados, que não deixem resíduos ou contaminantes na fruta.

Para garantir a segurança do operador, o robô deve ser projetado de forma a minimizar o risco de acidentes, incluindo a possibilidade de colisões ou impactos com outros objetos. O robô também deve ser equipado com sensores de segurança, que podem detetar a presença do operador e interromper a operação em caso de perigo.

Em resumo, a segurança é um aspeto fundamental na implementação de robôs móveis para tarefas de apoio à vindima, e deve ser cuidadosamente considerada em todas as fases do projeto, desde o design até a operação em campo.

Um robô móvel para tarefas de apoio à vindima é um sistema complexo composto por uma variedade de componentes que desempenham funções específicas para otimizar o processo de colheita de uvas. Esses componentes são projetados para oferecer eficiência, precisão e automação às tarefas envolvidas na vindima.

Um dos principais componentes é o sistema de locomoção, que permite que o robô se mova de forma autónoma pela vinha. Dependendo das características do terreno e das necessidades específicas, o robô pode ser equipado com rodas, trilhos, lagartas ou até mesmo pernas robóticas que se adaptam ao ambiente. Esse sistema de locomoção é crucial para que o robô possa navegar pelas fileiras de videiras, alcançando os cachos de uvas desejados.

As câmaras e sensores desempenham um papel importante na perceção do ambiente. Câmaras de alta resolução são utilizadas para capturar imagens dos cachos de uvas, permitindo que o robô identifique os cachos maduros e saiba qual colher. Além disso, sensores podem ser usados para medir a luminosidade, temperatura, humidade e outros parâmetros do ambiente, fornecendo informações valiosas para o sistema de tomada de decisões do robô.

Os braços manipuladores são responsáveis por executar as tarefas físicas, como a colheita dos cachos de uvas. Esses atuadores podem ser braços robóticos equipados com garras especiais projetadas para agarrar e cortar as videiras de forma precisa. Através de uma combinação de algoritmos de controle e retroalimentação dos sensores, o robô pode realizar a colheita de forma delicada, evitando danos aos cachos e às plantas.

A unidade de processamento é o "cérebro" do robô, onde ocorre o processamento dos dados provenientes das câmaras, sensores e outros componentes. Pode ser um computador ou um microcontrolador poderoso o suficiente para executar algoritmos complexos e tomar decisões em tempo real. Essa unidade de processamento é responsável

por interpretar as informações dos sensores, planear as ações do robô e coordenar o funcionamento dos diferentes componentes.

Sistemas de navegação e localização são essenciais para permitir que o robô se mova pela vinha de forma autónoma e precisa. Isso pode envolver o uso de GPS para obter as coordenadas geográficas do robô, sistemas de posicionamento visual para identificar marcos ou pontos de referência no ambiente, ou sensores de proximidade para detetar obstáculos e evitar colisões durante a operação.

Além disso, o robô móvel para tarefas de apoio à vindima requer uma fonte de energia confiável e eficiente. Isso pode ser fornecido por meio de baterias recarregáveis de alta capacidade ou até mesmo sistemas de energia solar, permitindo que o robô opere por longos períodos sem a necessidade de recarga frequente.

Todo o funcionamento do robô é controlado por meio de software especializado, que inclui algoritmos de controle e planeamento de trajetória. Os algoritmos de controle são responsáveis por tomar decisões inteligentes com base nos dados dos sensores e nas informações do ambiente. Estes algorítmos determinam a trajetória do robô, a velocidade de deslocamento, as ações dos braços manipuladores e as interações com os elementos do vinhedo.

Além disso, os robôs móveis para tarefas de apoio à vindima podem fazer uso de algoritmos de aprendizagem de máquina para melhorar suas habilidades ao longo do tempo. Esses algoritmos podem ser treinados com grandes volumes de dados para aprimorar a deteção de cachos de uvas, reconhecimento de padrões, otimização das trajetórias e tomada de decisões mais precisas. Com o tempo, o robô pode aprender a identificar e selecionar os cachos de uvas mais adequados para a colheita, tendo em consideração critérios como maturação, qualidade e integridade dos frutos.

É importante destacar que a eficácia e o desempenho dos componentes do robô móvel para tarefas de apoio à vindima podem variar de acordo com as especificações técnicas, a qualidade dos sensores e braços manipuladores utilizados, bem como o desenvolvimento e a implementação dos algoritmos de controle. Portanto, é fundamental realizar pesquisas e testes adequados para garantir que o robô atenda aos requisitos específicos da vindima e alcance resultados satisfatórios.

2.7. Características e Medidas de Desempenho nos Robôs Agrícolas

Os robôs agrícolas são projetados para ajudar na realização de diversas tarefas na agricultura, desde a sementeira e a plantação até à colheita e a pulverização. Para avaliar o desempenho desses robôs, várias características e medidas de desempenho podem ser consideradas.

Uma das medidas mais cruciais é a capacidade de carga, que se refere à quantidade de peso que um robô agrícola pode transportar, sendo fundamental para avaliar a eficácia do robô na execução de tarefas que envolvam o transporte de equipamentos e materiais agrícolas. A velocidade de operação também é de extrema importância, pois está relacionada com a rapidez com que um robô agrícola pode mover-se e executar as tarefas atribuídas, o que é crucial para avaliar a eficiência do robô em cumprir as tarefas de forma ágil e precisa.

A precisão é outra característica fundamental, referindo-se à capacidade do robô de realizar tarefas com precisão e constância. Isto é de importância primordial para garantir que as tarefas sejam realizadas de modo preciso e com alta qualidade. Além disso, a flexibilidade também deve ser considerada, uma vez que está relacionada com a capacidade do robô de se adaptar a diferentes condições de trabalho, tipos de terreno e variedades de culturas. Isto é crucial para avaliar a versatilidade do robô e a sua capacidade de executar uma variedade de tarefas agrícolas.

Finalmente, a autonomia é outra medida crucial, uma vez que diz respeito à capacidade do robô de operar sem intervenção humana por longos períodos. Isto é essencial para avaliar a eficácia do robô e a sua capacidade de reduzir a dependência de mão de obra humana na agricultura. Além destas medidas, outras características e métricas de desempenho, como a eficiência no consumo de energia, a durabilidade e a facilidade de manutenção, também podem ser tidas em consideração na avaliação do desempenho dos robôs agrícolas. Estas medidas podem variar consoante o tipo de tarefa que o robô está a desempenhar e o ambiente em que está a operar.

3. Robô proposto

Este capítulo aborda pormenorizadamente os desafios que surgem na implementação da automação tanto na AP como na VP. Além disso, são apresentados os detalhes relativos à disposição das vinhas destinadas à produção de uvas de mesa. Neste contexto, é fornecida uma visão abrangente sobre o robô desenvolvido, com uma descrição detalhada dos seus principais componentes, materiais utilizados e as respetivas propriedades específicas. O objetivo é oferecer uma compreensão completa e aprofundada do sistema robótico em questão, bem como do seu funcionamento e dos desafios associados à sua aplicação em ambientes de viticultura de precisão.

3.1. Desafios na implementação da automação na agricultura e viticultura

De acordo com Mousazadeh [47], a implementação da automação na agricultura enfrenta uma série de desafios. Um dos principais desafios é a falta de padronização na produção agrícola, o que torna difícil para os fabricantes de equipamentos agrícolas projetar máquinas que possam ser facilmente adaptadas a diferentes tipos de culturas e terrenos. Além disso, a topografia variável, os diferentes tipos de solo e as condições climáticas também apresentam desafios significativos na implementação da automação na AP.

Outro desafio importante é o alto custo dos sistemas de automação, que podem ser proibitivamente caros para os pequenos agricultores. Além disso, muitos agricultores podem não estar familiarizados com as novas tecnologias e podem ser resistentes a adotar novos sistemas.

A falta de mão de obra qualificada também é um problema, pois os sistemas de automação exigem técnicos e engenheiros treinados para instalação, manutenção e reparação. A educação e os conhecimentos em tecnologia são, portanto, fundamentais para a implementação bem-sucedida da automação na agricultura.

Mousazadeh sublinha igualmente a relevância da conectividade e da interoperabilidade na implementação da automação na agricultura. Os sistemas de automação devem ser capazes de comunicar entre si e com outros sistemas de informação, como os sistemas de monitorização do clima e de previsão do tempo, de forma a maximizar a sua eficácia.

Por último, a segurança representa uma preocupação fundamental na implementação da automação na agricultura. Os sistemas de automação devem ser concebidos para garantir

a segurança dos trabalhadores e do ambiente, bem como para serem resistentes a falhas e a ataques cibernéticos.

A aplicação da automação na VP enfrenta desafios específicos. Um dos principais obstáculos reside na diversidade de terrenos e condições de cultivo encontrados em diferentes vinhas, o que torna complexa a conceção de sistemas de automação que possam ser facilmente adaptados a variados tipos de terreno e condições climáticas. Outro desafio é o fato de que muitas áreas vinícolas serem operadas por pequenos agricultores, que podem não ter recursos financeiros ou técnicos para investir em sistemas de automação sofisticados. Além disso, muitos produtores de vinho têm uma abordagem tradicional para a produção de vinho e podem não estar dispostos a adotar novas tecnologias.

Outra questão importante na viticultura é a necessidade de garantir que os sistemas de automação sejam capazes de operar em terrenos acidentados e irregulares, e que sejam resistentes a condições climáticas adversas, como chuva, vento e temperaturas extremas. Além disso, a segurança é uma preocupação importante na viticultura, dada a inclinação acentuada de muitos dos terrenos, os sistemas de automação precisam ser projetados para garantir a segurança dos trabalhadores e do meio ambiente.

Por fim, a implementação da automação na viticultura também deve levar em consideração as necessidades específicas de cada etapa do processo de produção de vinho, desde a poda e colheita até ao processamento e envelhecimento. É importante que os sistemas de automação sejam adaptados a cada etapa do processo, de forma a maximizar sua eficácia e reduzir o desperdício.

3.2. Contextualização

Para propor novos avanços técnicos e científicos no campo da agricultura 4.0, foi primeiro necessário conhecer as principais obras existentes, expondo as suas vantagens, limitações e falhas comuns, a fim de identificar as reais necessidades de melhoria e de inovação. Após uma revisão sistemática dos sistemas robóticos agrícolas aplicados na execução da preparação do terreno antes da plantação, sementeira, tratamento das plantas, colheita, estimativa do rendimento e fenotipagem, observou-se que 37% dos sistemas robóticos são 4WD, 64,52% não têm um braço robótico, 22,06% são utilizados em tarefas de monda, 32,23% utilizam câmaras RGB, em 35,48% não estão incluídos/não são reportados algoritmos de visão por computador, 80,65% estão ainda em fase de

investigação, 16,67% são concebidos por empresas/investigadores australianos e 41,94% são desenvolvidos por países do continente europeu, como se verifica na Figura 12, atrás apresentada [42].

Com o objetivo de melhorar os atuais sistemas robóticos agrícolas, será desenvolvido neste trabalho, um sistema robótico, para realizar diversas tarefas de apoio à vindima de uvas de mesa, tais como, a estimativa de produção, a monitorização das videiras, a análise do crescimento vegetativo, a colheita seletiva e respetivo transporte dos cestos com as uvas.

Atualmente, a maior parte da investigação em curso, e robôs já em serviço têm como foco principal a cultura de vinhas de uvas para vinho, centrando-se na análise do estado hídrico das plantas, remoção de ervas daninhas, realização da poda e monitorização do crescimento das videiras, e são poucos os estudos realizados para uvas de mesa. Investigações existentes sobre o funcionamento de braços manipuladores não referem apenas a sua baixa eficiência operacional e precisão, mas também mostram que possuem uma estrutura muito complexa, com perdas de tempo, elevado peso e inercia, e elevado custo de fabrico [42].

Como tal, tendo em conta os problemas encontrados nas investigações anteriores, procurou conceber-se um robô que realiza as tarefas mais importantes de apoio à vindima, e dois braços robóticos mais leves, com baixa inércia, mais rápidos a iniciar e parar e com custos de fabrico reduzidos.

As videiras de uvas de mesa são cultivadas em treliças em latada e dispostas em linhas, como é demonstrado na Figura 13. Estas videiras possuem normalmente um espaçamento de entrelinha de 5.0m e um espaçamento na linha de 1.25 a 3.0m, são suportadas por treliças em forma de T a 1.8m de altura, construída com tubos de aço e arames de ferro. Os cachos de uvas são distribuídos ao longo de arames na parte superior da treliça. Estes, ficam geralmente a 1.8m acima do solo.

Antes da colheita, as folhas são desbastadas, as videiras após a desfolha são mostradas na Figura 14, cada videira tem normalmente cerca de 19 cachos de uvas [48].

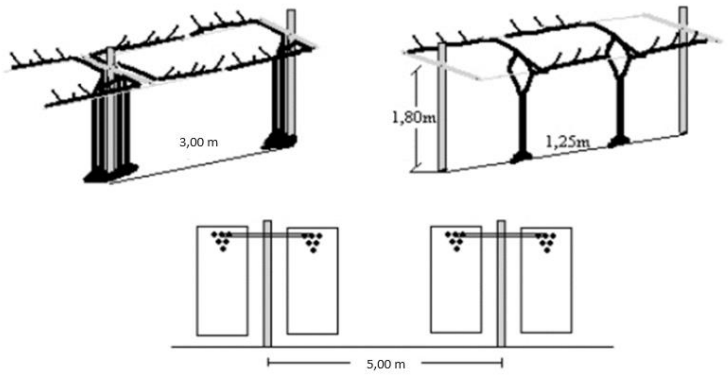


Figura 13 – Representação da estrutura de suporte das videiras

Figura 14 – Videiras em latada

A estrutura mecânica do robô proposto foi concebida através do estudo das condições de trabalho necessárias no terreno e das características desejadas do projeto, utilizando as etapas dos processos descritos na Figura 15.

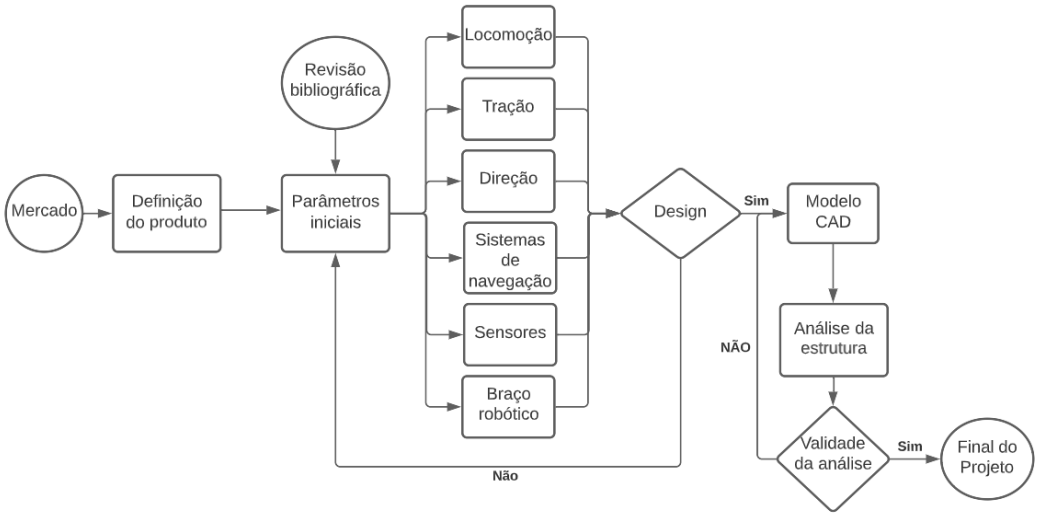


Figura 15 – Fluxograma da conceção do robô móvel agrícola

3.3. Visão Geral do robô

Nesta secção é apresentada uma visão geral do sistema desenvolvido. Este é composto por dois componentes principais de hardware, nomeadamente dois braços manipuladores e o corpo do robô, que serão descritos detalhadamente.

Cada componente de hardware acima mencionado inclui vários dispositivos especializados, a Tabela 3 inclui analiticamente a lista de dispositivos necessários no robô proposto.

Tabela 3 - Lista de dispositivos utilizados no robô

Sistema Hardware	Subsistema Hardware	Quantidade
Braços manipuladores	Garra de dois dedos	1
Corpo do robô	LIDAR Velodyne VLP-16	1
	GPS	1
	IMU RC	1
	Encoder	4
	Sensores de temperatura e humidade	2
	RGB USB 3.0	1
	NIR USB 3.0	1
	Bateria	1

O protótipo do robô é apresentado na Figura 16, com a identificação dos dispositivos e respetiva localização dos mesmos, a legenda da mesma encontra-se na Tabela 4. Na Figura 17 é mostrado um diagrama esquemático do funcionamento do robô. Os braços robóticos são montados na zona central das extremidades do corpo. São colocados nesta posição para maximizar o espaço de trabalho e conseguirem abranger uma maior área útil de colheita e alcançar toda a área do corpo do robô.

No compartimento inferior do corpo do robô encontra-se a bateria e todos os dispositivos eletrónicos necessários. Assim, os equipamentos ficam todos protegidos das condições agressivas externas, como por exemplo, o pó que se levanta com o movimento do veículo no terreno, salpicos devido a poças de água e até mesmo chuva e água de lavagem do robô.

A colocação dos braços robóticos no centro da direção transversal do veículo permite que a vindima seja realizada tanto nas videiras do lado esquerdo como do lado direito, fazendo com que seja necessário que o robô passe somente uma vez por cada entrelinha da vinha, poupando assim tempo e recursos.

Na parte frontal encontra-se uma unidade de controlo que fornece informações, tais como, percentagem de bateria, temperatura interna e externa e mapeamento da vinha, mostrando as uvas que deixou por colher, por estas ainda não estarem no ponto de maturação.

O corpo do robô é constituído por vigas deslizantes que permitem ajustar a sua largura. Isso possibilita ampliar a largura para acomodar mais cestos ou reduzi-la para adaptar-se a vinhas de vinho, que são mais estreitas. Além disso, permite moldar o robô para passar por entrelinhas com larguras variando entre 1,3 metros e 1,7 metros, tornando-o versátil para outras culturas.

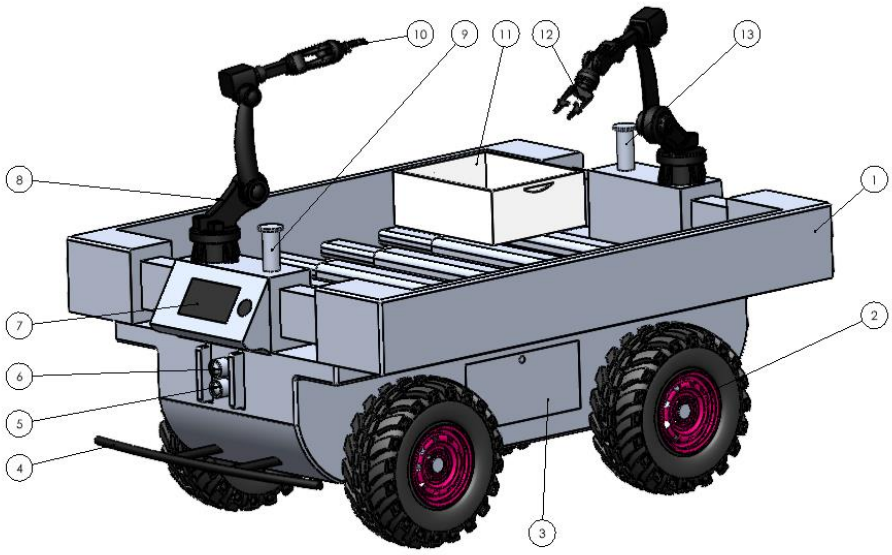


Figura 16 – Protótipo do Robô

Tabela 4 - Legenda da Figura 16

Lista de Entidades da Figura 16	
Número	**Designação**
1	Corpo do Robô
2	Encoders
3	Dispositivos eletrónicos, IMU, Bateria e Sensores de temperatura e humidade
4	Para-choques
5	Câmaras NIR e RGB
7	Unidade de controlo
8	Braço Robótico
9	Câmara LIDAR
10	Pinça de dois dedos
11	Caixa transportadora
13	Sensores ambientais e GPS

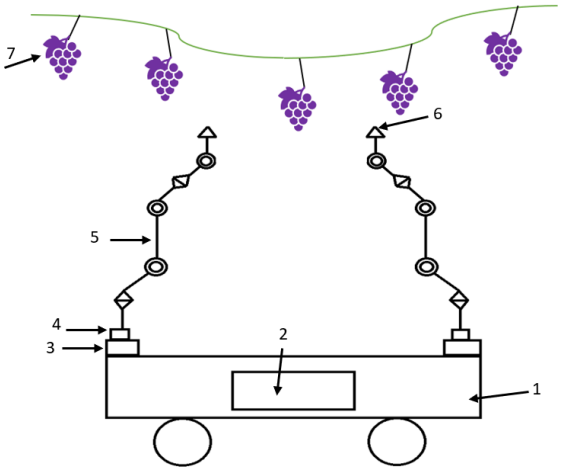


Figura 17 - Diagrama esquemático da configuração do robô. 1 - Chassis; 2 - Baterias e componentes elétricos; 3 - sistema de visão binocular; 4 - controlador principal; 5 - Braço manipulador; 6 - Garra; 7 – Uvas.

O sistema de aquisição de dados é composto pelas seguintes partes:

- Sensores de ambiente, que incluem dois sensores de temperatura e humidade, mais especificamente, um colocado dentro do compartimento dos dispositivos eletrónicos, para monitorização de avarias por sobreaquecimento, enquanto o outro é colocado externamente no robô para medição da temperatura e humidade do ambiente. [49]
- São montadas duas câmaras auxiliares no centro do robô móvel. As duas câmaras *Near-Infrared Spectroscopy* (NIR) e RGB sincronizadas são utilizadas para capturar imagens das linhas da vinha, a fim de calcular os índices de vegetação e temperatura. Os índices de vegetação são utilizados para caracterizar as áreas em termos de densidade de vegetação, permitindo ao utilizador ter uma visão geral da vinha.
- Para a odometria é realizada uma fusão de quatro *encoders* com uma *Inertial measurement unit* (IMU), e uma unidade de LIDAR para a localização do robô. A fusão dos *encoders* com a IMU permite a estimativa do estado inicial do robô. A localização é depois mais precisa com a utilização do LIDAR. É sabido que sistemas multimodais baseados numa combinação de sensores fornecem uma estimativa de estado mais precisa e robusta.

  O sistema IMU está localizado dentro da caixa com os componentes eletrónicos, cada roda tem um *encoder* e o LIDAR é colocado numa base de alumínio elevada ao lado dos braços robóticos.

- Para a operação de colheita das uvas, o robô dispõe de braços manipuladores com espaço de trabalho suficiente para realizar as operações de colheita, de acordo com a posição e orientação dos frutos. Como tal optou-se por escolher dois braços com 6 DoF, apenas com juntas de rotação, de modo a atingirem qualquer ponto do seu espaço de trabalho.

De acordo com as características de crescimento e distribuição dos cachos de uvas cultivados em latada, e as características do robô, tendo em consideração a flexibilidade e continuidade de posição das operações de vindima, o robô utiliza um modo de operação móvel como trajetória de marcha intermitente, apresentada na Figura 18. Assim sendo, o robô move-se de forma intermitente pela entrelinha das videiras. Este começa a partir da origem "O" na parte inferior esquerda, como é mostrado na Figura 18, após a recolha dos cachos que estão sobre ele estarem colhidos, continua a avançar, seguindo a trajetória

mostrada na figura. Quando o robô deteta um cacho dentro da faixa de maturidade, pára de se mover para poder realizar a colheita. Este processo é então repetido até acabar toda a trajetória ou até o robô ficar com os cestos cheios. Neste caso armazena na base de dados o último local onde fez a colheita, dirige-se até uma base onde se encontram operários que retiram os cestos cheios e colocam novos, regressando para a posição onde tinha ficado.

Os troncos de videira são utilizados como pontos de referência visuais. Este tipo de configuração permite ao robô navegar pelos corredores da vinha, virar quando chega ao fim de um corredor, regressar ao longo do próximo corredor e parar ao lado de cada videira no seu percurso. O fluxograma relativo ao movimento do robô pelas vinhas, é mostrado na Figura 19.

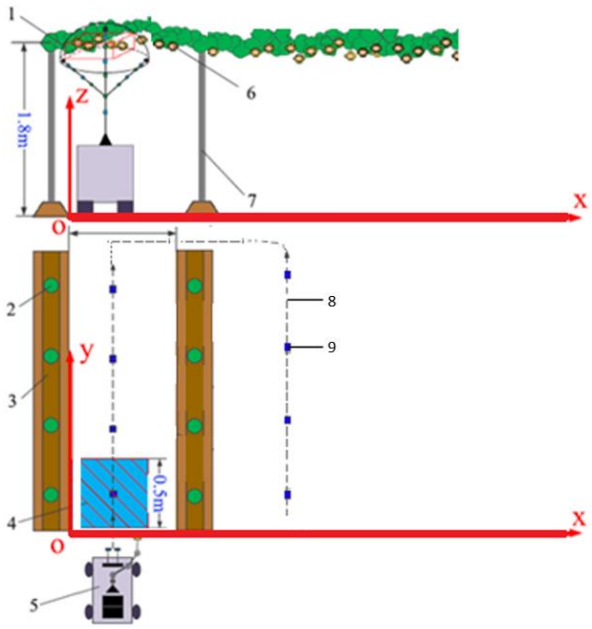


Figura 18 – Diagrama esquemático do funcionamento móvel do robô. 1- Espaço de trabalho em forma de guarda-chuva para polinização; 2- Videira; 3- Zonas de videiras distribuídas em colunas; 4- Localização para uma única colheita; 5- Robô; 6- Uvas; 7- Treliças das vinhas; 8- Trajetória do robô; 9- Pontos de paragem do robô.

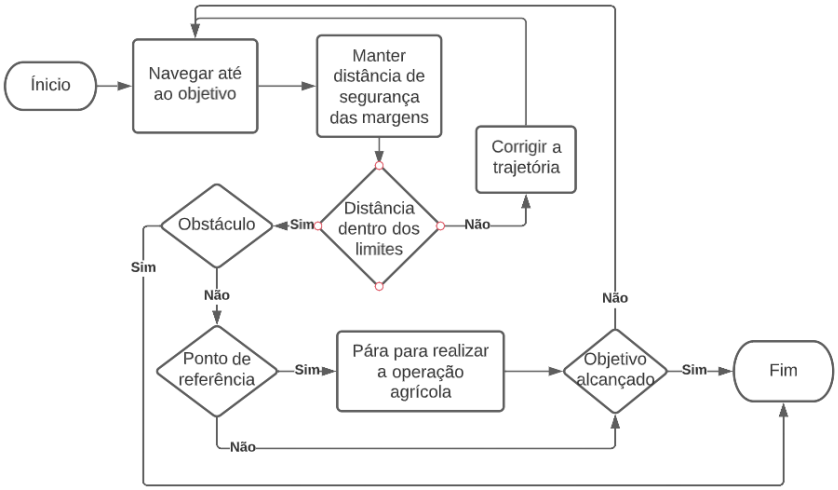


Figura 19 – Fluxograma do movimento autónomo do robô

Como anteriormente mencionado, o robô tem caminhos predeterminados, e assim consegue obter o plano operacional de navegação e mover-se relativamente da sua posição atual para o objetivo recebido. A posição atual e objetivo final são descritos por três variáveis: (x, y, θ), em que x e y descrevem o deslocamento do robô da sua posição atual nos eixos x e y, e θ descreve o ângulo de desvio.

Os braços antropomórficos do robô têm a vantagem de replicar o movimento de um braço humano através da união de múltiplas ligações com articulações rotacionais. Isto é desejado, uma vez que a maioria dos ambientes de produção atuais foram concebidos para serem trabalhados por humanos. E como tal, um robô capaz de compreender e replicar o processo completo de vindima manual (Figura 20) pode aumentar o sucesso da vindima.

Figura 20 - Exemplificação da vindima manual

Reproduzir com sucesso a vindima manual em modo automatizado tem múltiplas vantagens, como, o facto de o robô de colheita não ficar cansado após longas horas de trabalho, e poder continuar a trabalhar, desde que seja fornecida uma fonte de energia adequada. Infelizmente, a maioria dos robôs de apoio à vindima dependem atualmente das baterias como fonte primária de energia e levam bastante tempo a carregar. Este problema pode ser resolvido através da utilização de múltiplos robôs de colheita no local. Enquanto alguns estão a realizar a colheita, outros estão a carregar.

Outra vantagem é que o erro na apanha de frutos pode ser drasticamente reduzido através do uso de robôs, pois estes só irão colher frutos maduros que correspondam com sucesso às características que foram programadas, enquanto operadores menos experientes podem colher incorretamente frutos verdes. Como tal, o desenvolvimento de robôs de vindima tem-se concentrado principalmente em ambientes de agricultura de precisão, onde é necessário colher produtos individualmente, pois nem todos os frutos irão amadurecer ao mesmo tempo [50].

No entanto, os robôs de vindima atuais tendem a não colher os produtos que estão totalmente ocultos. Em alguns desses casos, a replicação do processo de colheita humana exigiria muitos *Degrees of Freedom* (DoF), bem como múltiplos sensores para assegurar que o subsistema de entrega de movimento não fica preso. Contudo, para o tipo de vinha em latada, que é o objetivo principal do robô deste trabalho, teria poucos cachos ocultos pois as folhas ficam por cima dos arames e os cachos ficam pendurados, como já se verificou na Figura 14.

Com base nos fluxogramas apresentados nas Figura 21 e Figura 22, é viável discernir as discrepâncias entre a vindima manual e a vindima autónoma. A distinção predominante ocorre quando um cacho de uvas maduras fica oculto, o que impede que o braço robótico o alcance sem ficar preso. Neste cenário, o robô limita-se a registar a localização do cacho

maduro no mapa da vinha, deixando para que um operador proceda posteriormente à colheita do cacho maduro.

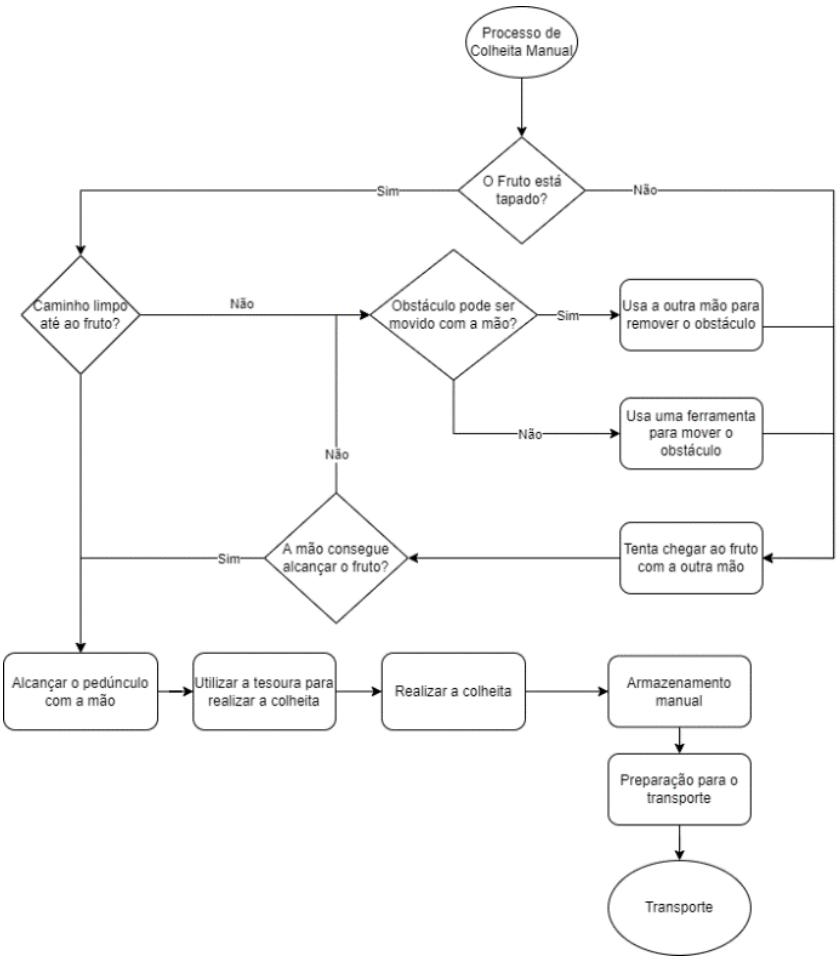


Figura 21 - Fluxograma de vindima manual

Para se colher um cacho detetado é necessário estimar a sua maturação e validá-la, pois, apenas os cachos com um grau de maturação semelhante ao desejado são colhidos. Se o cacho estiver completamente maduro, o algoritmo define o centro de massa (x0, y0, z0), converte estas coordenadas de pontos de imagem para pontos de espaço e calcula a distância relativa a partir de pontos de referência predefinidos. Estes pontos de referência são as arestas da ferramenta de corte instalada no braço manipulador como elemento final, que são sempre evidentes em cada *frame*.

Depois de cortar o cacho, o braço manipulador regressa à sua posição inicial e liberta o cacho numa caixa de colheita. Os troncos das videiras são utilizados como pontos de referência visuais para a navegação. Mais especificamente, o robô executa todas as tarefas após parar paralelamente aos troncos das videiras, de modo que a posição de origem do braço fique a meio do tronco.

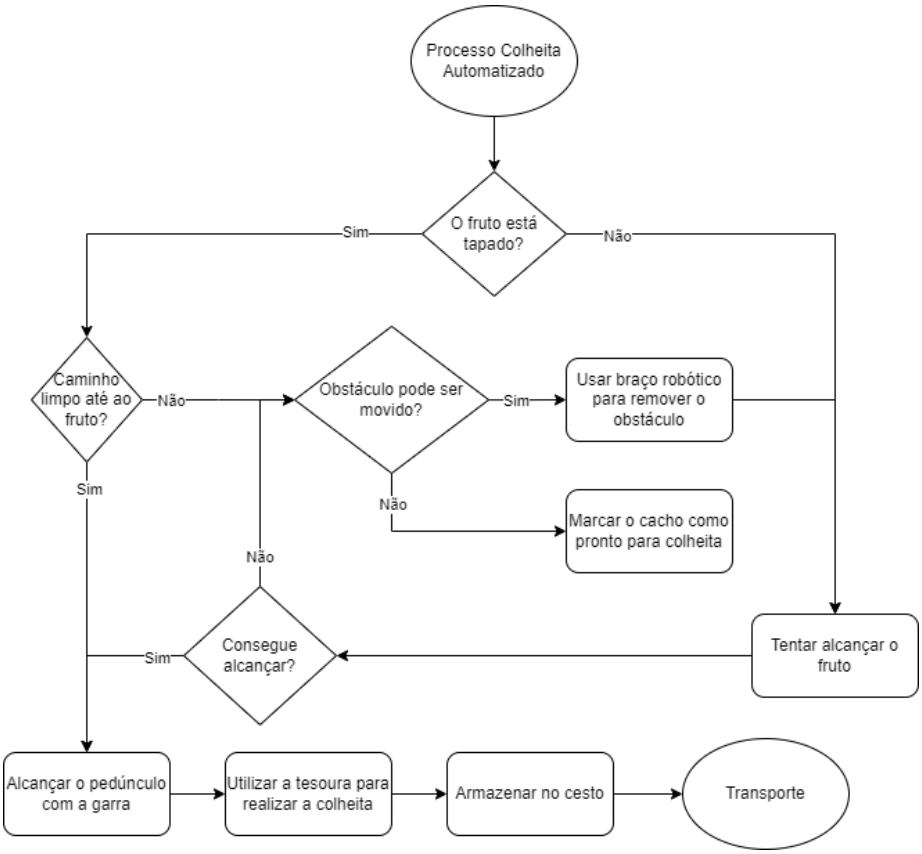


Figura 22 - Fluxograma de vindima autónoma

3.4. Componentes do robô

3.4.1 Estrutura do Robô

Quando os robôs são classificados pelo seu tamanho, surgem inúmeros padrões em termos de capacidade, design e características. A grande maioria dos *Unmanned Ground Robot* (UGR) de AP, são robôs acionados eletricamente que são usados principalmente para monitorização de culturas e/ou solo. Como são veículos de pequeno porte e capacidade limitada, precisam de um menor investimento, são mais ágeis e não têm problemas particulares de fornecimento de energia, podendo ser adaptados a uma ampla gama de culturas.

Existem também robôs com tamanho e potência de máquinas agrícolas e tratores tradicionais. Neste tipo de robôs, a tendência de projeto mais usual, é incorporar recursos autónomos nos veículos agrícolas de condução manual (por exemplo, tratores, ceifeiras-debulhadoras), em vez de se desenvolver projetos totalmente novos.

Já os robôs de tamanho médio oferecem o melhor de dois mundos, e uma ampla variedade de designs. Este grupo de robôs geralmente pode monitorizar o campo usando sensores incorporados ou remotos e realizar tarefas em campo.

Muitos robôs são altamente especializados para uma gama muito pequena de tarefas, sendo projetado de acordo com as funções específicas que irão desempenhar. Outros robôs são desenvolvidos com recurso a plataformas versáteis que podem realizar muitas tarefas, resultando em "designs" mais adaptáveis e modulares, como é o propósito do robô proposto no âmbito deste trabalho. Estes robôs têm o potencial de reduzir o investimento económico necessário e promover um novo paradigma de produção, que emprega tecnologia lucrativa e que tem vantagens ambientais e sociais significativas [7].

O corpo do robô (componente estrututral) desenvolvido é formado por um perfil de alumínio com ranhuras, o que cria uma estrutura muito resistente e leve. Essa solução foi selecionada visto que é de fácil montagem e permite a fácil fixação de diferentes peças que possam ser adicionadas após a conclusão do projeto, além do fácil reposicionamento e ajustabilidade da estrutura.

O corpo do robô possui dois braços manipuladores destinados à realização de colheita. Estes braços possuem 6 eixos e direcionam a garra que realizará a recolha dos cachos de uvas que estiverem maduros, podendo trabalhar fora da estrutura do robô para a realização de atividades próximo ao caule da cultura, tendo um alcance máximo de 1200mm.

Na Figura 23 é mostrada a perspetiva isométrica do corpo do robô, ou seja, o desenho 3D final da componente estrutural, já montado. O modelo 3D foi uma mais-valia pois possibilitou a otimização dos recursos construtivos, melhoria de componentes e restringiu possíveis erros de montagem. Com isso, foi possível realizar uma *Bill of Materials* (BOM) mais precisa (Tabela 3), que auxilia na gestão do projeto de montagem, evitando que fosse interrompido por alguma falta de componente.

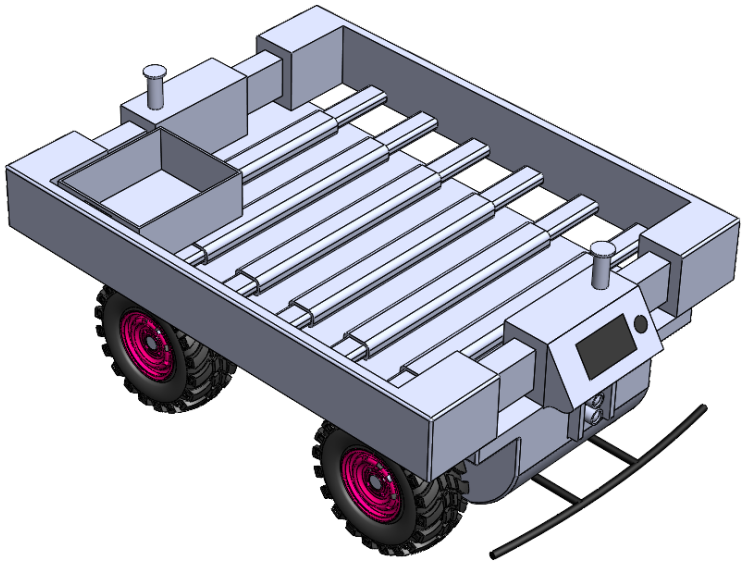


Figura 23 - Perspetiva isométrica do robô

Como tal, as dimensões do robô deste trabalho, são de 1700x1300x1035 [mm], como demonstrado na Figura 24. O material escolhido devido às suas boas características de resistência e dureza foi a liga de alumínio 6063, sendo um alumínio no estado recozido. Proporciona uma superfície de alta qualidade que pode ser facilmente anodizada. A liga 6063 oferece uma boa resistência à corrosão, principalmente se for aplicado um acabamento superficial adequado. Também oferece boa soldabilidade, brasagem e trabalhabilidade. Esta liga fornece resistência à tração moderadamente boa e maquinabilidade razoável [51]. As propriedades desta liga são representadas na Tabela 5. A estrutura tem um peso bruto de 846kg. Possui ainda um para-choques construído em alumínio para proteger a estrutura de possíveis embates que possam ocorrer.

Tabela 5 - Propriedades mecânicas da liga de alumínio 6063

Propriedades mecânicas	**Massa Específica**	**Módulo de Elasticidade**	**Resistência à tração**
Liga de alumínio 6063	2.7 g/cm^3	69 GPa	90 MPa

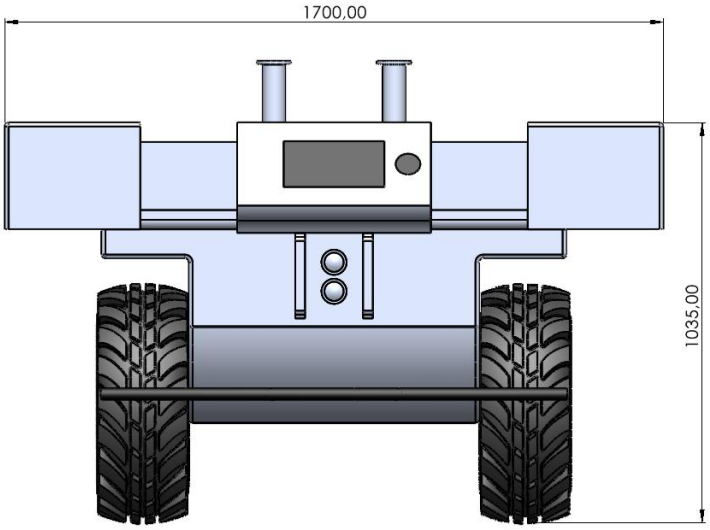


Figura 24 - Dimensões do robô

A modelação do protótipo foi desenvolvida com software CAD Solidworks. Com isso, foi realizado o planeamento de toda a construção e montagem do robô. Esta ferramenta também possibilitou a realização dos desenhos técnicos que irão auxiliar na construção da estrutura e de outros componentes. Na Figura 25 encontra-se a vista explodida do veículo realizada pelo software CAD.

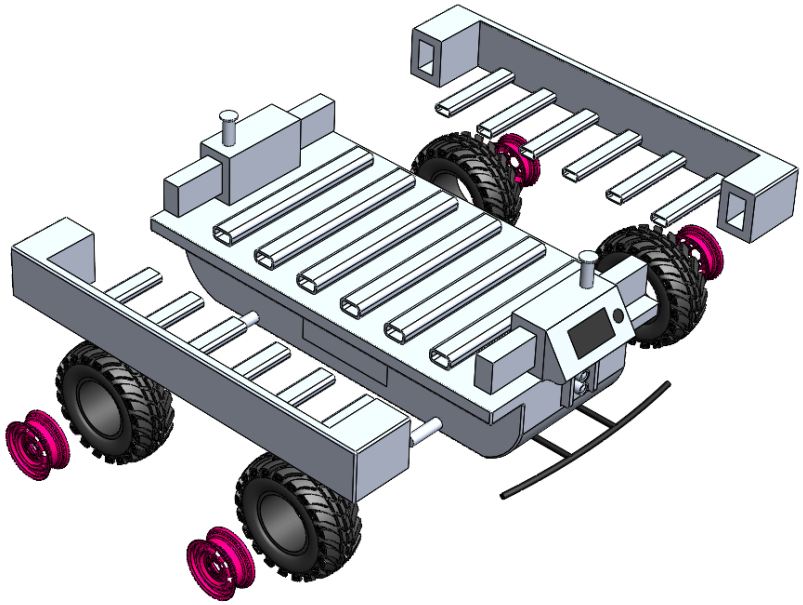


Figura 25 - Vista explodida do robô

Na Tabela 6 apresentam-se valores referentes às especificações do robô.

Tabela 6 - Especificações técnicas do Robô

Especificações do Robô	Valores
Peso Bruto	846 kg
Carga útil	60 kg
Comprimento	1300 mm
Largura	1700 mm
Altura	1035 mm

3.4.2 Braço manipulador

Com o intuito de alcançar os objetivos previamente delineados, foi concebido um robô móvel equipado com dois braços manipuladores destinados à colheita de uvas. Os parâmetros de design foram cuidadosamente selecionados de forma a cumprir os requisitos de espaço de trabalho necessários para a colheita dos cachos de uvas. Optou-se pelo nylon 6 como material para os braços robóticos, de modo a conciliar os requisitos de rigidez e resistência, reduzindo simultaneamente o peso e melhorando a durabilidade do equipamento. Os braços robóticos idealizados foram projetados para atender aos elevados padrões de precisão de movimento, visando uma colheita orientada ponto a ponto.

O braço manipulador necessita de estar equipado com elementos terminais que permitam executar as tarefas, substituindo a intervenção humana. Estes elementos incluem pinças de ação suave para a realização da colheita seletiva dos cachos, ferramentas de corte e a necessidade de manusear e armazenar os frutos colhidos com cuidado. Além disso, é possível adaptar a garra a elementos terminais que possibilitem a realização de outras tarefas agrícolas, como a monda mecânica, pulverização de precisão, bem como diversas formas de inspeção e tratamento.

Durante a operação do braço robótico durante a colheita de uvas, é imprescindível que a garra localizada na extremidade do braço seja capaz de alcançar qualquer ponto na área de trabalho. Tendo em conta que a extremidade do braço robótico possui seis graus de liberdade, permitindo que o elemento terminal seja posicionado em qualquer localização e orientação dentro do espaço de trabalho do robô (conforme descrito por Wang *et al.*,

2022), optou-se por selecionar um braço robótico articulado com esses seis graus de liberdade, como ilustrado na Figura 26.

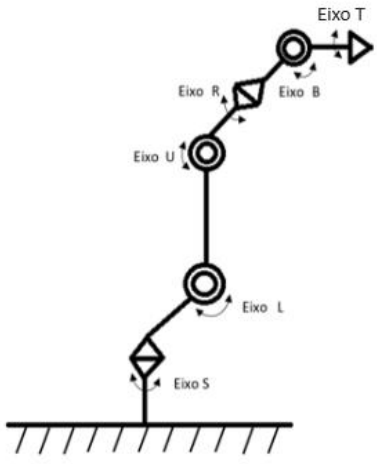


Figura 26 - Esquema do braço robótico

De modo geral, a junta de rotação no eixo S, juntamente com a junta de inclinação do antebraço no eixo L e a junta de inclinação do antebraço no eixo U controlam a posição da garra. Já a junta rotativa no eixo R, a junta de inclinação no eixo B e a junta rotativa no eixo T controlam a orientação da garra da extremidade, de modo que esta possa enfrentar diretamente o pedúnculo do cacho em qualquer orientação.

Os braços articulados podem ter 2 tipos de juntas: rotação ou translação. Podem ter combinações dos 2 tipos. Esta configuração tem 6 juntas de rotação. A forma e ordem como estão colocadas é que irão rodar ou inclinar as juntas seguintes. As primeiras 3 juntas definem a estrutura cinemática do espaço de trabalho (cartesiano, esférico, cilíndrico, etc.). Este é um robô articulado vertical. As primeiras 3 juntas posicionam o elemento no espaço de trabalho com 3 coordenadas. As outras 3 juntas adicionam graus de liberdade de orientação ao elemento terminal. A garantia que a posição cobre a área necessária da vindima tem a ver com as dimensões dos braços.

Conforme especificado nas diretrizes da Figura 27, é requerido que a extremidade do braço robótico abranja um espaço tridimensional de 500 mm × 500 mm × 150 mm. Levando em conta as margens de tolerância, a secção vertical do espaço de trabalho do braço robótico foi dimensionada para englobar uma área retangular de 750 mm × 200 mm. O espaço de trabalho da parte principal do braço robótico foi identificado, conforme ilustrado na Figura 27, onde a área sombreada representa a região de colheita. O ponto O corresponde à projeção do eixo de articulação do braço superior neste plano. A maior área

teórica de colheita, obtida através da rotação da articulação basal do braço robótico, é representada pela zona circular destacada na Figura 28.

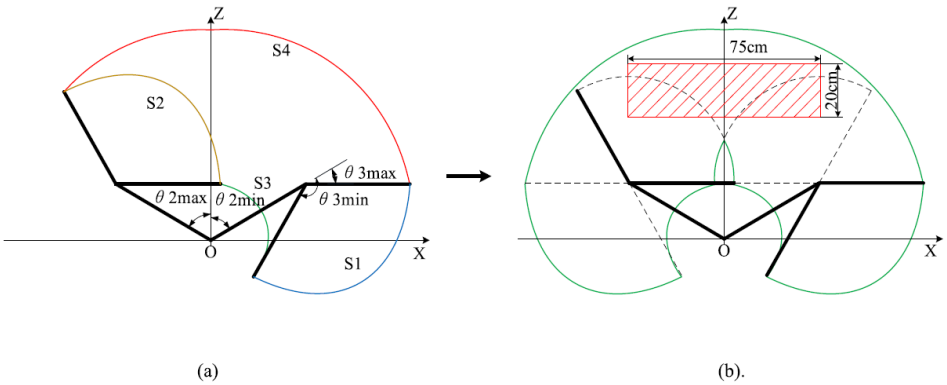


Figura 27 - Espaço de trabalho do braço robótico, adaptado de [52]

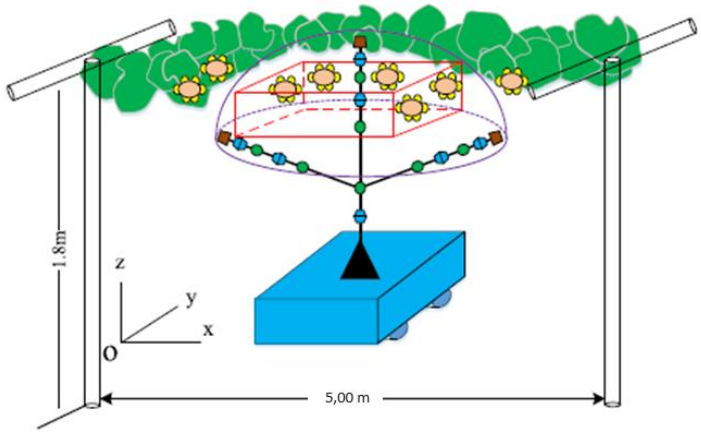


Figura 28 - Diagrama esquemático do espaço de trabalho com um apenas um braço robótico, adaptado de [52]

Para atender aos requisitos da tarefa de colheita de uvas, que envolve posicionar o terminal do braço robótico em qualquer ponto do espaço de trabalho, tornou-se necessário a utilização de dois braços robóticos nas extremidades opostas do robô móvel. O espaço de trabalho foi dimensionado para abranger as dimensões de 750 mm × 750 mm × 200 mm, o que resultou no comprimento adequado para os braços robóticos projetados. Devido à altura real dos cachos de uvas, que se aproxima dos 1800 mm, foi adotado um chassis com 1200 mm de altura para assegurar que o braço robótico seja capaz de alcançar os pedúnculos. A modelação do braço robótico foi desenvolvida com o software CAD Solidworks como demonstrado na Figura 29. Neste processo, foi realizado o planeamento de toda a construção e montagem do braço, que incluía quatro partes: base, braço superior, antebraço e articulações do pulso. Tanto a parte superior do braço como as articulações do antebraço são feitas de tubos de fibra de carbono. Com exceção das peças padrão, tais

como o motor, redutor, e parafusos, as peças chave do braço robótico foram projetadas para poderem ser impressas em 3D com nylon 6. Os parâmetros característicos do nylon 6 e dos tubos de fibra de carbono estão listados nas Tabela 7 e Tabela 8.

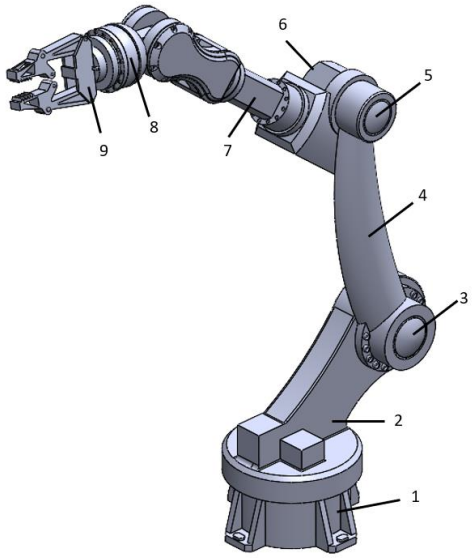


Figura 29 *- Modelo 3D do braço robótico utilizado para a vindima. 1- Base; 2- Articulação da cintura; 3- Articulação do ombro; 4- Braço superior; 5- Articulação do cotovelo; 6- Articulação do punho; 7- Antebraço; 8- Articulação de balanço do punho; 9- Garra*

Tabela 7 - Parâmetros característicos do nylon 6

Propriedades mecânicas	**Massa Específica**	**Módulo de Elasticidade**	**Resistência à tração**
Nylon 6	1.13 g/cm^3	3 400 MPa	70 – 84 MPa

Tabela 8 - Parâmetros característicos da fibra de carbono

Propriedades mecânicas	**Massa Específica**	**Módulo de Elasticidade**	**Resistência à tração**
Fibra de Carbono	1.5- 2.0 g/cm^3	240 GPa	3 400 MPa

4. Estudos estáticos da estrutura

As propriedades mecânicas do material utilizado nas análises de elementos finitos da estrutura estão apresentadas na Tabela 5, referida no ponto 3.4.1 Estrutura do Robô, configurados no software Solidworks. De modo a simplificar o modelo utilizado no cálculo, os elementos que não têm função estrutural como baterias, motores, sistema de direção e braços robóticos, foram removidos da estrutura. Apenas as estruturas principais do robô foram analisadas. Os braços robóticos foram removidos e aplicadas forças representando os esforços que a estrutura terá de suportar devido ao seu peso.

A intensidade de força que deve ser aplicada no robô para realizar tarefas de vindima dependerá de vários fatores, como o tipo de uva, o tamanho e peso do cacho, o tipo de terreno, entre outros. É importante realizar uma análise detalhada desses fatores para determinar a intensidade de força a ser aplicada no robô.

Para estabelecer a intensidade de força necessária, é viável realizar ensaios práticos em campo com o intuito de mensurar a força exigida para a colheita das uvas em variadas condições. Esta abordagem contribui para a determinação da carga de trabalho que o robô deverá suportar durante a operação de colheita.

Adicionalmente, o projeto do corpo do robô deve ser concebido tendo em consideração a capacidade de carga e a robustez da estrutura para resistir à força necessária. A aplicação de análise de elementos finitos pode ser uma ferramenta valiosa para simular as forças que atuam sobre a estrutura do robô, possibilitando a determinação da resistência necessária para suportar essas forças.

Deste modo, a intensidade da força a ser implementada no robô é multifatorial e requer uma análise minuciosa destes fatores, bem como do design do corpo do robô para ser devidamente determinada.

Determinar a magnitude apropriada da força a aplicar no robô de colheita de uvas no SolidWorks pode ser um desafio, especialmente quando não se dispõe de dados específicos. No entanto, existem algumas abordagens que podem ser consideradas:

- Análise teórica: Realizar uma análise teórica das forças envolvidas na colheita de uvas é uma opção. Isso pode implicar a consideração de fatores como a resistência dos pedúnculos, o método de colheita utilizado e a eficiência dos mecanismos de

colheita. Com base nessas considerações, é possível estimar uma magnitude razoável para a força a ser aplicada no modelo do robô.

- Consultar estudos semelhantes: Pesquisar estudos ou artigos científicos que tenham analisado a colheita de uvas por robôs agrícolas semelhantes pode ser útil. Esses estudos podem fornecer informações sobre as magnitudes de forças geralmente encontradas durante a colheita de uvas, servindo como referência para a análise.
- Conversar com especialistas: Se possível, contactar especialistas no campo da agricultura robótica ou engenheiros agrícolas é uma excelente abordagem para obter orientação e informações sobre as forças envolvidas na colheita de uvas. Estes podem oferecer insights valiosos com base nas suas experiências e conhecimentos.
- Simulações iterativas: Quando não se dispõe de informações precisas sobre as magnitudes de forças, realizar simulações iterativas no SolidWorks pode ser uma estratégia útil. Para o efeito deve-se começar com uma magnitude de força inicial e observar os resultados da análise de elementos finitos. Com base nesses resultados, ajustar a magnitude da força e repita a análise até encontrar um valor que pareça razoável e produza resultados aceitáveis em termos de tensões, deformações e fatores de segurança.

É importante salientar que essas abordagens são sugestões e podem variar dependendo do contexto específico do projeto. Sempre que possível, é recomendável buscar informações específicas sobre a colheita de uvas na região em questão ou realizar testes reais para obter dados mais precisos sobre as forças envolvidas.

Ao considerar esses fatores e analisar as necessidades específicas do robô para as tarefas de vindima, é possível definir um intervalo apropriado de intensidade de força que atenda às exigências do projeto.

Em estudos e projetos de robôs agrícolas, as forças de transporte podem variar dependendo do tipo de cultura, do peso médio dos produtos a serem transportados e do design do robô em si. Em média, as forças de transporte utilizadas em estudos anteriores para o transporte de colheitas variam de 50 N a 500 N ou mais, dependendo do peso das colheitas e do design do robô.

Esses intervalos de força são apenas exemplos gerais e podem variar significativamente com base nas especificidades de cada projeto e nos requisitos de transporte.

É importante lembrar que o intervalo de intensidade de força foi escolhido tendo em conta as características específicas do projeto, como o design do robô, as condições do terreno e as características das videiras. Além disso, o intervalo de intensidade de força deve ser validado por meio de testes em laboratório e simulações numéricas para garantir a segurança e eficiência do robô durante a operação.

Tendo em conta, as forças resultantes explicadas acima, foram então introduzidas no software de simulação como demonstradas na Figura 30.

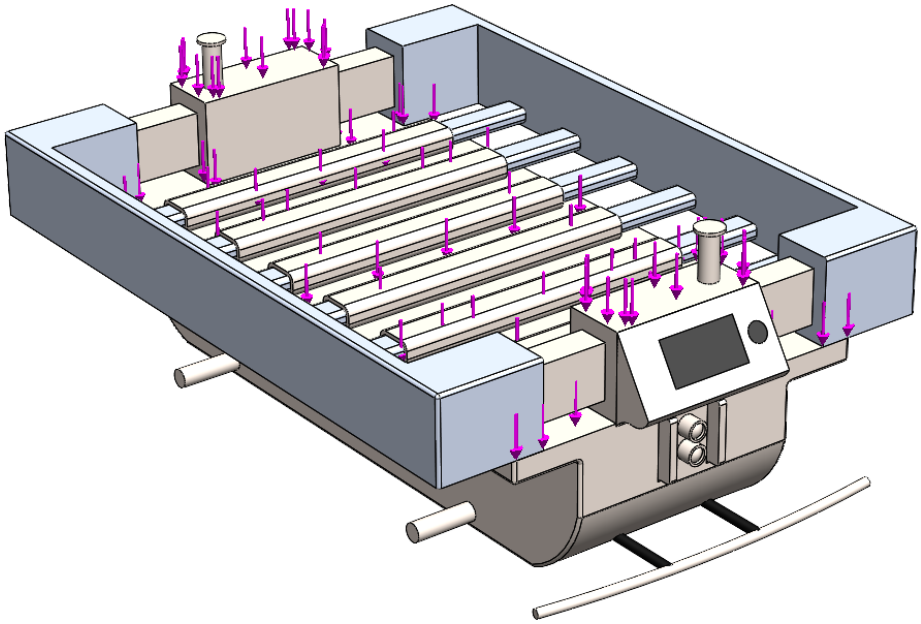


Figura 30 - Forças resultantes aplicadas no robô

4.1 Casos de Estudo

A simulação foi realizada considerando a estrutura em condições de tensão dinâmica em duas condições severas de utilização: através de obstáculo, trajeto com inclinação lateral. Os detalhes da estrutura com as forças aplicadas são apresentados na Figura 31.

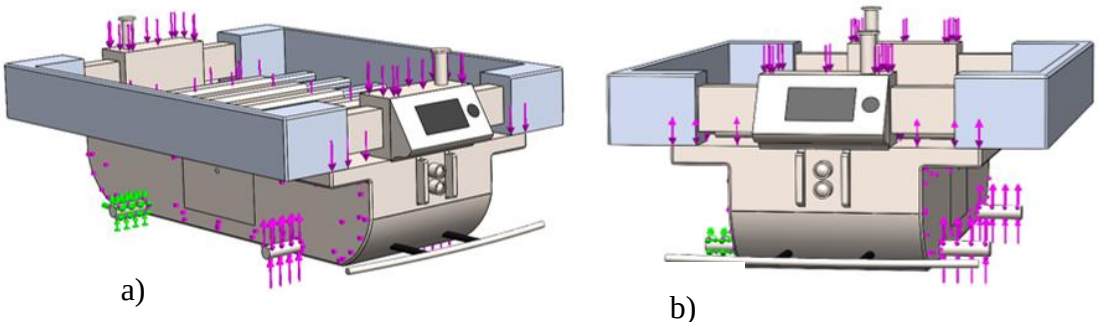


Figura 31 - Forças aplicadas e condições de fronteira aplicadas: a) através de um obstáculo, b) com inclinação lateral direita

Quando um robô agrícola está a atravessar um obstáculo, como terrenos irregulares, inclinações acentuadas ou superfícies acidentadas, ou pequenos troncos, as forças de movimentação podem variar significativamente devido à necessidade de superar a resistência adicional do obstáculo.

A determinação das forças de movimentação em condições de atravessar obstáculos depende de vários fatores, incluindo o tamanho e a geometria do obstáculo, a capacidade de tração do robô, a distribuição de peso, a velocidade de deslocamento e o tipo de sistema de locomoção utilizado.

Em estudos e projetos que envolvem a travessia de obstáculos por robôs agrícolas, foram relatadas forças de movimentação variadas. Algumas referências indicam que as forças podem variar de 500 N a 1000 N, dependendo do tamanho e tipo de obstáculo, da inclinação do terreno e das características do robô.

Tendo em conta projetos anteriores já realizados, considerou-se para a primeira situação, que o robô é operado com apenas as duas rodas traseiras apoiadas no solo, e as rodas dianteiras simulam o aparecimento de um obstáculo com uma força vertical de 1000N, em cada apoio das rodas dianteiras.

Na última simulação, é considerada uma inclinação lateral direita fazendo com que apenas as duas rodas do lado esquerdo estejam apoiadas no solo, e as rodas da direita são submetidas a uma força de 1000 N, em cada apoio destas.

Os resultados obtidos das simulações, contendo os dados de deformação total e tensão equivalente (von-Mises) são apresentados nas Figura 32, Figura 33, Figura 34 e Figura 35. É necessário salientar que a escala de deformação utilizada nas figuras é aumentada para facilitar a visualização.

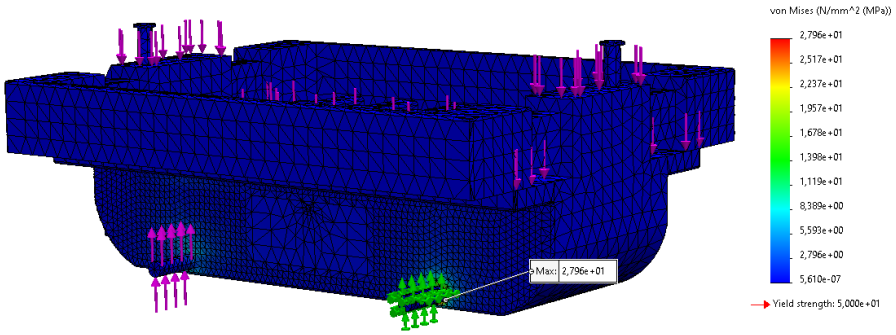


Figura 32 - Tensão total na estrutura na primeira situação

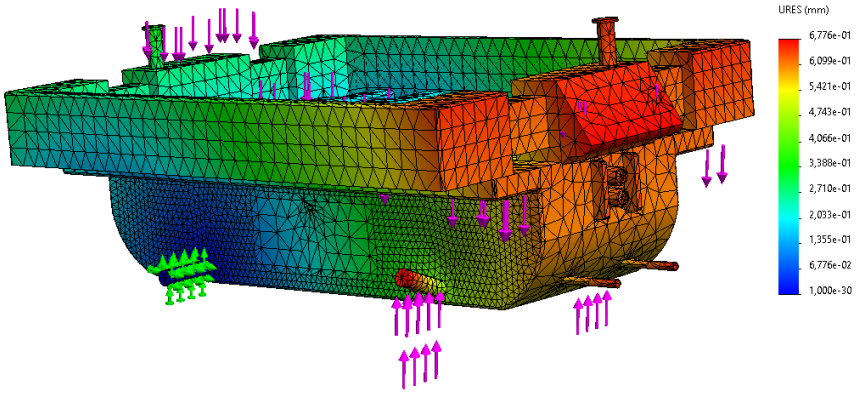


Figura 33 - Deformação total na estrutura na primeira situação

As Figura 32 e Figura 33 mostram o resultado da simulação do sistema na primeira situação (através de obstáculo). É possível observar que os pontos de acumulação de tensão (27,96 MPa) ocorreram na junção entre a estrutura e os apoios das rodas traseiras, mas abaixo da tensão de rotura (197 MPa). Relativamente à deformação é de notar que a maior (0,677 mm) parte da deformação ocorreu na zona dianteira da estrutura, mas sem comprometer o desempenho do conjunto.

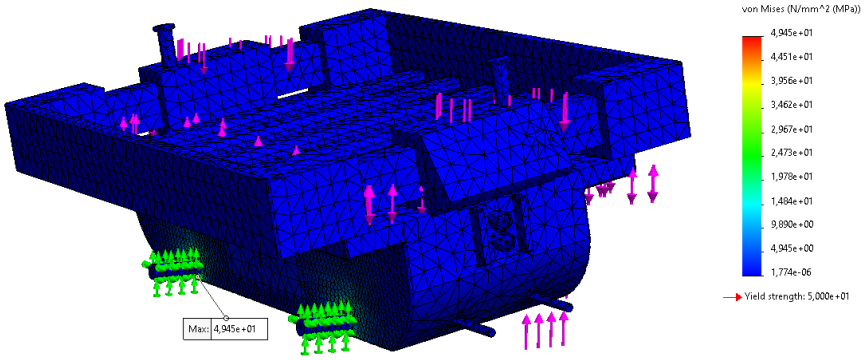


Figura 34 - Tensão total na estrutura na segunda situação

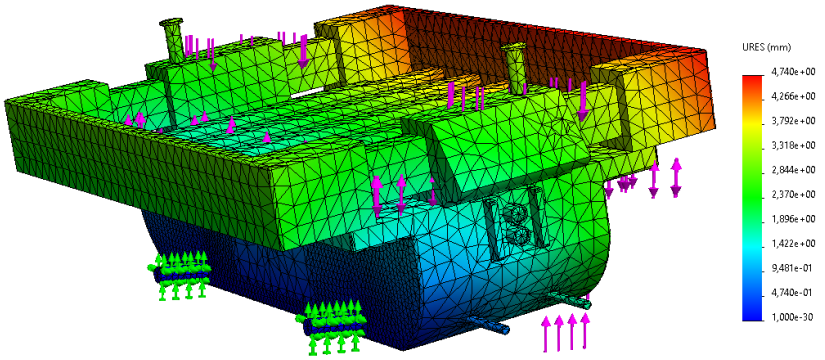


Figura 35 - Deformação total na estrutura na segunda situação

As Figura 34 e Figura 35 mostram o resultado da simulação da segunda situação (trajeto com inclinação lateral). Em comparação com a primeira situação, a deformação é superior (49,45 MPa), tendendo a torcer a estrutura. Em relação ao stress, aumentou devido à torção da estrutura (4,74 mm), concentrando-se nos mesmos pontos da primeira situação, mas neste caso nas rodas do lado esquerdo. Em nenhum dos casos, a estrutura foi comprometida, visto as tensões de Von Mises obtidas foram inferiores à resistência máxima da liga de alumínio 6063 (50 MPa).

5. Simulação de navegação do robô móvel na vinha

Neste capítulo é apresentada uma análise abrangente dos simuladores disponíveis no mercado, explicando detalhadamente o processo de seleção e proporcionando uma introdução completa ao seu funcionamento. A compreensão do raciocínio subjacente à programação de um robô é essencial, o que justifica a apresentação de uma modelação detalhada do sistema, bem como a descrição do algoritmo correspondente

Por fim, é realizada a simulação de três cenários específicos, seguida por uma análise crítica e uma discussão profunda dos resultados alcançados, com o objetivo de extrair conclusões e conhecimentos relevantes para o desenvolvimento do trabalho.

5.1. Simuladores de Robótica

Os simuladores de robótica são programas de computador que desempenham a função de modelar, simular e representar visualmente o comportamento de sistemas robóticos complexos em um ambiente virtual. Essas ferramentas são amplamente empregadas em diversos domínios, incluindo pesquisa, desenvolvimento e educação em robótica.

Alguns dos simuladores de robótica mais renomados são:

- CoppeliaSim (anteriormente conhecido como V-REP (Virtual Robot Experimentation Platform)): Esta plataforma de simulação versátil possibilita a modelação, simulação e visualização de sistemas dinâmicos complexos, abrangendo robôs, máquinas, veículos, sensores e atuadores. Reconhecido pela sua interface gráfica de fácil utilização, recursos avançados e comunidade ativa de utilizadores e programadores, o CoppeliaSim é amplamente utilizado nos campos da robótica e engenharia.
- Gazebo: Este simulador de robótica de código aberto é amplamente adotado para simulações em ambientes tridimensionais, incluindo robôs. Oferece uma gama extensa de funcionalidades, como simulação de física em tempo real, deteção de colisões e modelação de sensores e atuadores.
- Webots: permite simular em três dimensões robôs, sensores, atuadores e ambientes. É usado em diversas áreas, desde investigação até desenvolvimento e educação em robótica, e destaca-se por recursos avançados como simulação em

tempo real, deteção de colisões e compatibilidade com o *Robot Operating System* (ROS).

- RobotStudio: Especializado na simulação e programação de robôs ABB, o RobotStudio permite representar o comportamento desses robôs em ambientes virtuais. É amplamente empregado na programação de robôs industriais.
- ROS: é uma plataforma de código aberto bastante difundida no desenvolvimento robótico. Oferece uma vasta gama de ferramentas e bibliotecas para controlo de robôs, navegação, processamento de imagem e outras aplicações. Além disso, o ROS integra vários simuladores de robótica, incluindo o Gazebo.

Em resumo, os simuladores de robótica são recursos de software preciosos para a modelação, simulação e visualização de sistemas robóticos complexos em ambientes virtuais. A escolha do simulador depende das necessidades específicas de cada utilizador, uma vez que cada um destes apresenta vantagens e desvantagens distintas.

5.2. Escolha do Simulador

Para a realização da simulação do sistema em análise, após testes realizados com várias ferramentas disponíveis, a decisão recaiu sobre o uso do CoppeliaSim. Esta escolha fundamentou-se em diversas vantagens que esta aplicação oferece em comparação com outros programas de simulação. Estas incluem uma interface gráfica intuitiva e de fácil utilização, que permite aos utilizadores criar modelos tridimensionais e controlar a simulação em tempo real.

O CoppeliaSim é uma plataforma versátil, capaz de simular não só robôs, mas também sistemas dinâmicos complexos em várias áreas, como engenharia mecânica, controlo de sistemas e inteligência artificial. Além disso, proporciona flexibilidade ao suportar uma vasta gama de linguagens de programação, incluindo Python, Lua, C++, MATLAB e ROS, permitindo aos utilizadores escolher a linguagem com a qual se sintam mais confortáveis.

Destacam-se ainda os recursos avançados oferecidos pelo CoppeliaSim, como física em tempo real, deteção de colisões, geração automática de código e integração com outros programas e hardware. Estas características tornam-no uma escolha sólida para a simulação do sistema em questão.

5.3 Introdução ao CoppeliaSim

CoppeliaSim é um software de simulação multiuso que permite modelar, simular e visualizar sistemas dinâmicos complexos, como robôs, máquinas, veículos, sensores e atuadores. Foi desenvolvido pela Coppelia Robotics e é também conhecido *como* V-REP.

CoppeliaSim é amplamente utilizado em pesquisa, desenvolvimento e educação em áreas como robótica, engenharia mecânica, controle de sistemas e inteligência artificial. O *software* possui recursos avançados, como física em tempo real, deteção de colisão, geração de código automático e integração com outros softwares e hardware. Esses recursos tornam a plataforma uma ferramenta valiosa para simular e testar sistemas antes de implementá-los na prática.

Em resumo, CoppeliaSim é uma plataforma de simulação poderosa e flexível que oferece uma ampla gama de recursos e é amplamente utilizada em diferentes áreas da robótica e engenharia.

Lua é uma linguagem de programação versátil, reconhecida pela sua simplicidade, eficiência e flexibilidade. A sua sintaxe clara e concisa facilita a leitura e a manutenção do código. A sua notável velocidade torna-a uma escolha popular no desenvolvimento de jogos e em outras aplicações que exigem um alto desempenho. Além disso, a sua portabilidade e extensa documentação fazem com que seja amplamente utilizada em diversos projetos de software em várias áreas.

Para além disso, a linguagem Lua é altamente flexível e pode ser facilmente integrada a outras linguagens de programação, suportando a incorporação de bibliotecas em outras linguagens, como C e C++, o que a torna uma opção atraente para projetos que exigem integração com outras linguagens. É altamente portátil e pode ser facilmente executada numa ampla variedade de plataformas, incluindo Windows, Linux, macOS, Android e iOS. Isso torna a linguagem Lua uma escolha popular para desenvolvimento de aplicativos móveis e jogos. Tem uma boa documentação, com um manual abrangente, bem organizado e atualizado. Apesar disso, existem poucos recursos online, como fóruns e tutoriais, que auxiliem a comunidade a obter ajuda e compartilhar informações.

5.3.1 Interface Gráfica do Utilizador

A interface gráfica do CoppeliaSim é uma das características mais marcantes do software, pois permite uma interação intuitiva com o ambiente de simulação e seus objetos. A

interface é dividida em várias janelas que mostram diferentes informações e opções de configuração do cenário de simulação. A seguir, descreve-se as principais janelas da interface do CoppeliaSim:

Janela de cenário: Esta é a janela principal do CoppeliaSim, onde o utilizador pode visualizar o ambiente de simulação em 3D e interagir com os objetos e robôs. É possível girar, mover e ampliar a visualização da cena, além de selecionar e arrastar objetos para posicioná-los.

Janela de hierarquia de objetos: Esta janela mostra uma lista hierárquica de todos os objetos presentes na cena, permitindo ao utilizador selecionar objetos específicos e visualizar suas propriedades e componentes. É possível alterar as propriedades dos objetos, como as suas dimensões, massa, posição e orientação.

Janela de scripts: Esta janela permite ao utilizador criar e editar scripts Lua que controlam o comportamento dos objetos e robôs na cena. É possível escrever códigos para mover os robôs, ativar sensores e controlar os atuadores. O CoppeliaSim possui uma ampla variedade de funções integradas para tornar a programação mais fácil e eficiente.

Janela de ferramentas: Esta janela contém várias ferramentas úteis para a edição de objetos, como ferramentas de desenho, ferramentas de seleção e ferramentas de criação de objetos. Além disso, também é possível aceder às opções de configuração do software, como as configurações de física e renderização.

Janela de informações: Esta janela exibe informações detalhadas sobre os objetos selecionados na cena, incluindo suas propriedades, componentes e scripts Lua associados.

Janela de simulação: Esta janela exibe informações em tempo real sobre a simulação em execução, incluindo o tempo decorrido, o número de ciclos de simulação por segundo e o estado dos objetos e componentes.

Considerando a Figura 36, que apresenta a representação gráfica da interface do software CoppeliaSim, é possível descrever as características de cada um dos elementos identificados.

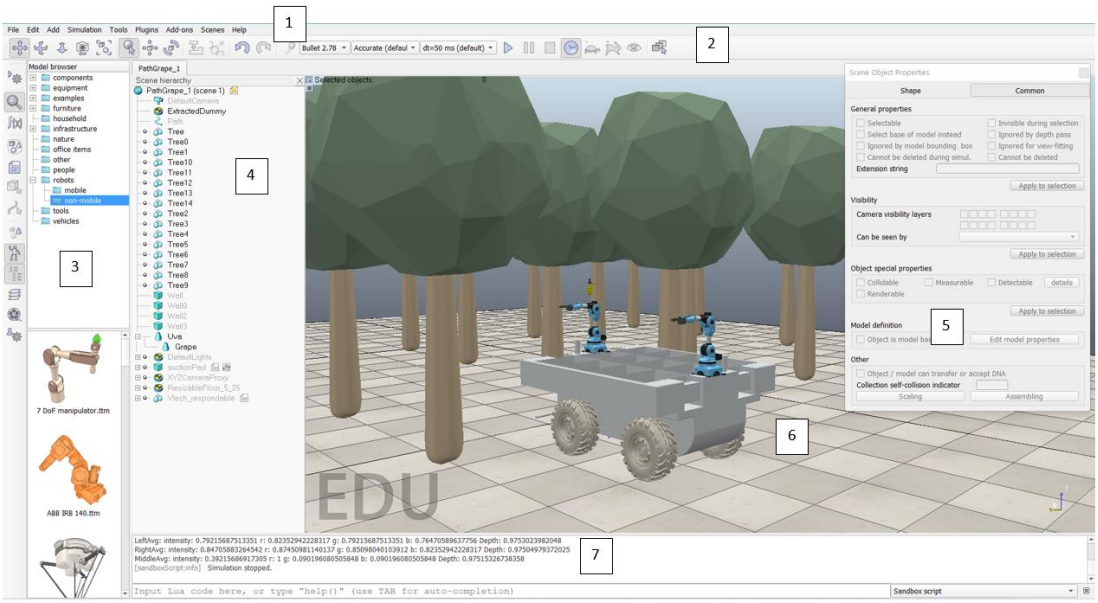


Figura 36 - Interface Gráfica do Simulador

1. Barra de Menu: Esta seção permite o acesso às funcionalidades do simulador que, ao contrário das mais frequentemente utilizadas, não podem ser acedidas por meio da interação com os modelos, menus pop-up e barras de ferramentas.
2. Barras de Ferramentas: Esses componentes, visando a conveniência do utilizador, incorporam as funções mais amplamente utilizadas e essenciais para interagir com o simulador.
3. Browser dos Modelos: A parte superior exibe as pastas de modelos disponíveis no CoppeliaSim, enquanto a parte inferior apresenta as miniaturas dos modelos que podem ser inseridos na cena através da funcionalidade de "arrastar e soltar" suportada pelo simulador.
4. Hierarquia da Cena: Aqui, é possível analisar todo o conteúdo de uma cena, ou seja, todos os objetos que a compõem. Dado que cada objeto é construído de forma hierárquica, esta organização é representada por meio de uma árvore hierárquica, conforme exemplificado na Figura 37. Com um duplo clique no nome de cada objeto, o utilizador consegue aceder à "Interface Personalizada" que permite efetuar alterações. Esta funcionalidade do simulador, aliada ao recurso de "arrastar e soltar," é utilizada para estabelecer relações parentais entre os objetos, onde os objetos filhos são incorporados na estrutura do objeto pai.
5. Interface Personalizada pelo Utilizador: Através desta janela, é possível configurar rapidamente todos os componentes inseridos na cena, sendo ativada sempre que necessário para cada um dos objetos.

6. Cena: Este elemento representa a componente gráfica da simulação, isto é, o o ambiente tridimensional com todos os objetos que foi criado e programado.
7. Barra de Estados: Este elemento exibe informações sobre as operações em curso, comandos e mensagens de erro. Além disso, os utilizadores também podem utilizá-la para imprimir *strings* a partir de um *script*.

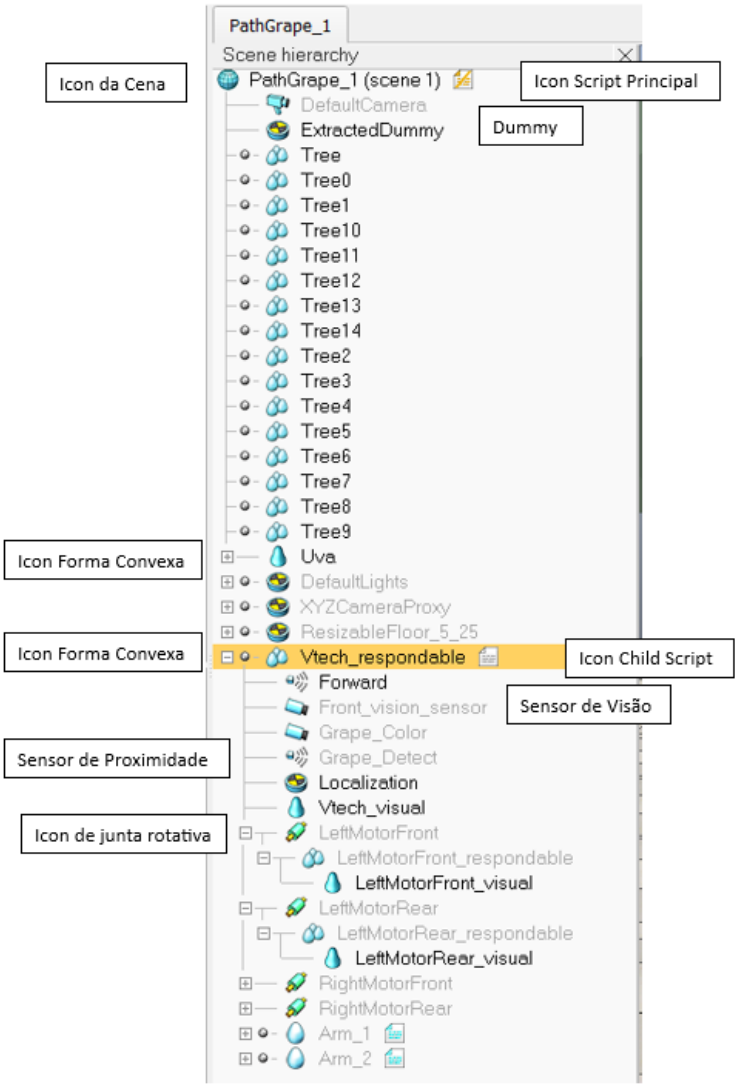


Figura 37 - Exemplo de uma hierarquia criada composta por objetos, juntas, formas, dummies, sensores de proximidade e sensores de visão

5.3.2 Mecanismos de Controlo

Para construir uma cena complexa que envolva movimento, cálculos ou interação, é necessário realizar tarefas de configuração e programação, além de utilizar objetos de cena e dos módulos de cálculo básicos. Para controlar todos os aspetos de uma simulação, existem diferentes abordagens disponíveis, cada uma com suas próprias vantagens. Ao executar uma simulação, o código pode ser executado de três modos distintos:

- Utilizando diferentes máquinas (computadores ou robôs) conectados ao computador onde a simulação está a ser realizada.
- Executando o código numa máquina separada do processo principal da simulação.
- Executando o código e a simulação no mesmo computador.

No CoppeliaSim, há quatro abordagens para implementar o código e programar um modelo:

1. *Scripts* Embutidos: Estes *scripts* são o principal mecanismo de controlo e estão entre as ferramentas mais poderosas do *CoppeliaSim*. Permitem que o utilizador programe diretamente dentro do modelo, eliminando a necessidade de *software* externo. A linguagem de codificação utilizada aqui é o Lua.
2. *Plug-ins*: Os plug-ins são usados para personalizar ainda mais uma cena, expandir funcionalidades ou adicionar novos comandos de script. Alguns recursos, como a interface OMPL ou ROS, são incorporados através de plug-ins que utilizam uma interface C/C++.
3. *Add-ons*: Os *add-ons* podem ser funções independentes ou utilizados em conjunto com o código em execução. Esta abordagem também utiliza a linguagem Lua e permite personalizar o simulador.
4. API Remota: A API remota facilita a interação com *software* externo ou máquinas, como robôs reais e o *CoppeliaSim*. Essa integração pode ser realizada através de cinco linguagens de codificação diferentes.

Estas diversas abordagens e métodos de implementação proporcionam flexibilidade e capacidade ao programar modelos no *CoppeliaSim*, permitindo a criação de cenários complexos e a interação com diversos sistemas e dispositivos externos.

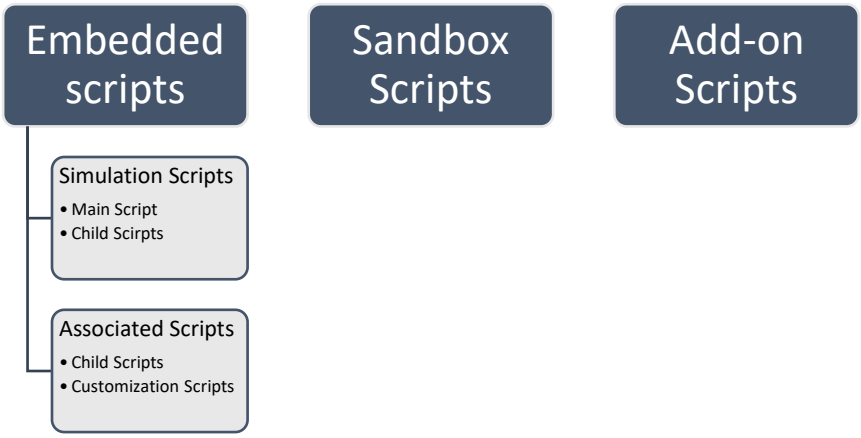


Figura 38 - Tipos de Scripts suportados pelo simulador

Os *scripts* embutidos desempenham um papel fundamental no controle de simulações realizadas no ambiente do CoppeliaSim. Esse método oferece uma integração fácil, escalabilidade inerente e evita conflitos com diferentes versões do software. Além disso, sua utilização demanda menos esforço na criação, modificação e manutenção dos modelos de simulação, eliminando problemas de sincronização. Os scripts embutidos permitem aos utilizadores aproveitar as mais de quinhentas funções disponíveis na biblioteca de API do CoppeliaSim, abrangendo praticamente todas as possibilidades de programação. Esses scripts atuam como uma conexão central para todos os outros mecanismos de controle, desempenhando um papel essencial em qualquer cena simulada.

O CoppeliaSim suporta cinco tipos diferentes de scripts, cuja relação pode ser analisada na imagem 19 para fins de utilização. O *script* principal, conhecido como *Main script*, está sempre presente em todas as cenas simuladas, uma vez que contém o código necessário para a execução da simulação. É importante ressaltar que esse *script* não deve ser modificado, pois a sua função é invocar os scripts secundários. Além do *Main script*, existem os chamados *Child scripts*, que estão associados a objetos ou modelos presentes na cena e são utilizados para implementar algoritmos de controle específicos. Esses scripts também podem ser aplicados em outros contextos. Os *Child scripts* são classificados em dois tipos: *Non-threaded* e *Threaded*. Os scripts *Non-threaded* consistem em funções que são chamadas a cada etapa da simulação, realizando tarefas específicas (*callbacks*) e, em seguida, o script que os invocou retoma o controlo. É essencial que isso ocorra, pois, caso contrário, a simulação será interrompida e não poderá continuar. O *script Non-threaded* é composto por quatro funções principais:

- *sysCall init*: Responsável pela inicialização do *script* e é a única função obrigatória. Esta função é executada apenas uma vez durante a simulação e contém o código de inicialização necessário.
- *sysCall actuation*: Executada a cada etapa da simulação durante a fase de atuação. Geralmente, esta função contém o código relacionado às ações dos componentes do modelo.
- *sysCall sensing*: Executada a cada etapa da simulação durante a fase de sensoriamento. Esta função é responsável pela aquisição de dados por meio dos sensores do modelo.
- *sysCall cleanup*: Função de encerramento que é executada uma vez no final da simulação ou quando o script é removido.

Por outro lado, os scripts do tipo *Threaded* são executados em uma *thread* separada, paralela à simulação. Isso significa que eles não estão sincronizados com as etapas da simulação, embora seja possível alcançar essa sincronização utilizando um conjunto específico de funções. No entanto, a simulação não é interrompida durante a execução desses scripts, sendo que a execução do próprio script é interrompida no final de uma etapa e retomada na etapa seguinte.

5.3.3 Configuração das Propriedades Dinâmicas

Para realizar a simulação dinâmica de modelos no CoppeliaSim, é imperativo efetuar a configuração apropriada das propriedades das formas utilizadas. Cada forma pode ser categorizada com base nas seguintes características:

- **Dinâmica ou Estática**: As formas dinâmicas estão sujeitas à influência de forças externas, podendo deslocar-se ou sofrer alterações em resposta a essas forças. Por outro lado, as formas estáticas mantêm-se fixas numa posição específica ou seguem o movimento do objeto pai na hierarquia da cena.
- ***Respondable* ou *Non-respondable***: As formas *respondable* originam uma reação quando colidem com outras formas *respondable* durante a simulação. Por contrapartida, as formas *non-respondable* não geram qualquer reação quando ocorre uma colisão com outras formas.

A escolha das propriedades adequadas para cada forma depende do objetivo pretendido na simulação. Estas configurações podem ser ajustadas na caixa de diálogo que abrange as características dinâmicas das formas, conforme ilustrado na Figura 39. Para além das propriedades previamente referidas, esta caixa de diálogo permite a modificação de outros parâmetros de relevância, tais como a massa e os momentos de inércia dos objetos.

Ao ajustar devidamente estas características, torna-se viável alcançar uma simulação mais precisa e realista, levando em consideração a interação das formas com as forças externas, colisões e quedas. Estas configurações desempenham um papel fundamental na representação adequada do comportamento dos objetos simulados, assegurando resultados coerentes e fiáveis.

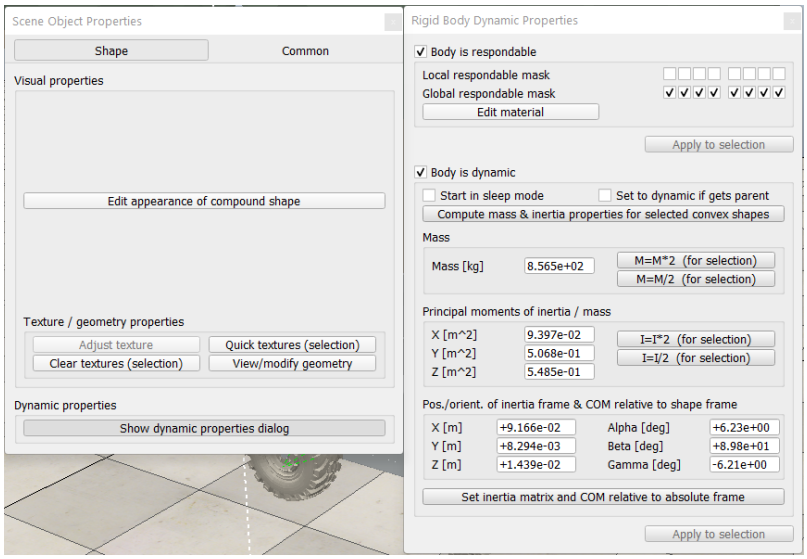


Figura 39 - Propriedades dinâmicas de uma Forma

5.3.4 Hierarquia do sistema

Para realizar a simulação do robô no CoppeliaSim, é essencial criar a estrutura hierárquica correspondente. A hierarquia deve ser estabelecida a partir da base do modelo, cuja identificação varia de acordo com o tipo de robô. A construção dessa hierarquia pode variar de modelo para modelo, mas deve seguir a hierarquia de movimentos relativos dos objetos. A hierarquia utlizada neste modelo está apresentada Figura 40.

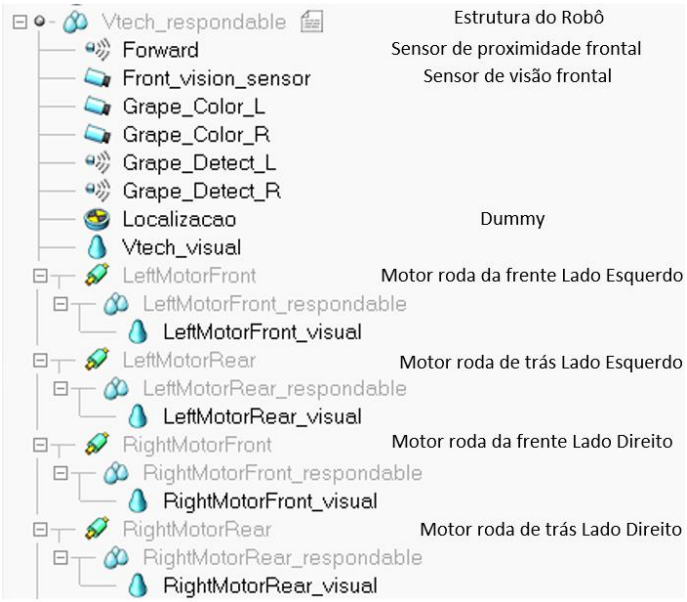


Figura 40 - Hierarquia criada para o robô

Uma vez definida a hierarquia dos diversos componentes que compõem o robô, é possível dar início à simulação. No entanto, as formas importadas a partir do desenho inicial não serão utilizadas diretamente. Com o objetivo de aprimorar a representação visual da

simulação, esses modelos 3D importados serão "anexados" às formas primitivas dos vários componentes. Essa ação permite que, durante as simulações, os modelos 3D importados sigam o movimento das formas primitivas.

É importante destacar que essa nova estrutura também inclui componentes que não faziam parte do robô original, mas que são necessários para a implementação do algoritmo proposto. Um exemplo disso é um "dummy" utilizado para rastrear a localização do robô. Esses componentes foram anexados à forma primitiva "EstruturaRobo" do robô, uma vez que, na realidade, seriam instalados nessa mesma posição.

Ao adotar essa abordagem, é possível aprimorar a visualização da simulação, permitindo uma representação mais precisa do movimento e interação dos componentes do robô. Essa técnica de anexar os desenhos 3D importados às formas primitivas contribui para uma simulação mais imersiva e realista, facilitando a análise e avaliação do desempenho do robô durante as simulações realizadas no CoppeliaSim.

5.4 Modelação do Sistema

5.4.1 Criação do modelo

Após a apresentação do software utilizado nesta experiência e das diversas funcionalidades disponíveis, descreve-se agora o processo completo da transição do modelo 3D para o CoppeliaSim.

Conforme mencionado anteriormente, inicialmente foi realizada a modelação do robô no SolidWorks e, em seguida, feita a importação da mesma para o simulador CoppeliaSim, a representação final do robô encontra-se ilustrada na Figura 41.

Vale ressaltar que, durante a importação dos diversos componentes do robô para o simulador, foram ignorados objetos irrelevantes para fins de simulação que apenas aumentariam a carga computacional, como parafusos e acessórios de fixação.

No caso do compartimento que contém as baterias e outros componentes de controlo, apenas foi considerado o peso total da caixa, incluindo as baterias e seus acessórios correspondentes, para avaliar eventuais problemas de tração. Completada essa etapa, dispõe-se do design original e das formas primitivas que serão utilizadas para a simulação, ambas visíveis na Figura 41.

Ao seguir esse processo de transição, o modelo 3D do robô é adequadamente integrado ao ambiente de simulação do CoppeliaSim, proporcionando uma representação fiel do comportamento do robô durante a execução das simulações.

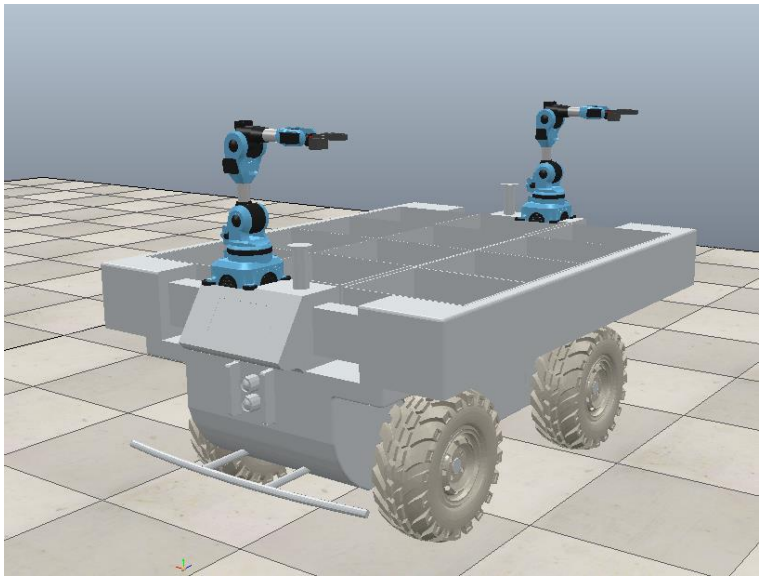


Figura 41 - Layout do robô no simulador CoppeliaSim

Após a importação do robô previamente configurado no software CoppeliaSim, procedeu-se à criação de um ambiente simulado com semelhanças aos campos de cultivo, onde o referido robô será operado.

Nesse contexto, tornou-se necessário desenvolver elementos que representassem fielmente os cachos de uvas a serem colhidos. Para determinar suas dimensões e pesos, considerou-se a classificação das massas dos bagos e cachos de uvas, cujos detalhes podem ser consultados na Tabela 9.

Dado que os cachos podem apresentar variações consideráveis em termos de tamanho e peso, optou-se por incorporar na simulação elementos com diversos pesos, abrangendo pesos que variam de 200 a 450 g. Todos esses objetos foram criados utilizando a ferramenta SolidWorks e posteriormente importados para o software CoppeliaSim, com o intuito de replicar a aparência de um cacho de uvas Figura 42.

Tabela 9 - Pesos médios por casta

Casta	Média de peso por baga (g)	Média de peso por cacho (g)
Touriga Nacional	3,0	200
Cardinal	9,0	450
Red Globe	8,0	400
Crimson seedless	5,0	367,4

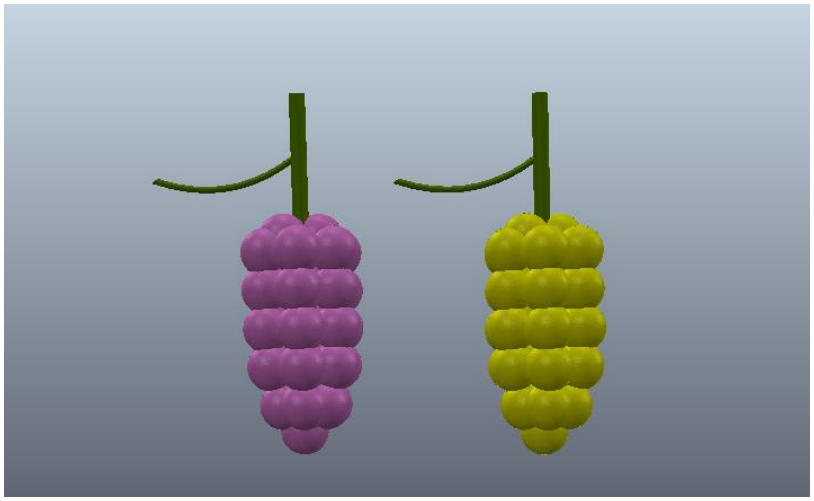


Figura 42- Representação de cachos de uvas no software CoppeliaSim

5.5 Criação do Algoritmo

5.5.1 Algoritmo Proposto

Para o robô realizar a tarefa de recolha de objetos foi desenvolvido o fluxograma representado na Figura 43, que cumpre as várias etapas e condições lógicas necessárias para o correto funcionamento do algoritmo.

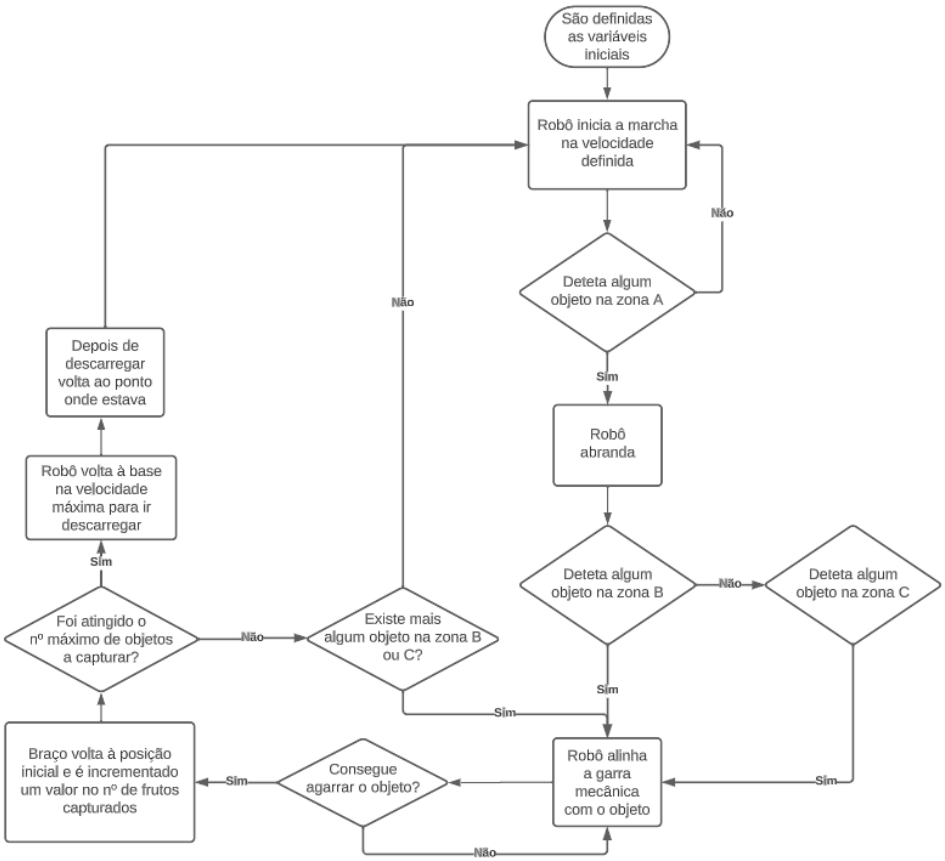


Figura 43 - Fluxograma do controlo do robô

5.5.2 Explicação das condições Iniciais

O presente trabalho tem como desígnio a conceção e implementação de um algoritmo de controlo destinado a um robô, fazendo uso do simulador CoppeliaSim. O controlo preciso e eficiente de robôs móveis assume uma relevância primordial no domínio da robótica, uma vez que possibilita o aperfeiçoamento da autonomia e eficácia desses sistemas em ambientes de complexidade acentuada. Nesse contexto, reveste-se de importância crucial a seleção criteriosa do tipo de *script* a ser empregue no simulador, de modo a possibilitar a vinculação ao objeto em análise e assegurar um desempenho otimizado.

Como referido no capítulo 5.3 Introdução ao CoppeliaSim, este simulador oferece diversas alternativas para a inclusão de scripts que permitem controlar diversos aspetos de um modelo robótico. Tendo em conta a natureza do algoritmo a ser desenvolvido para controlar um robô numa simulação, a escolha recaiu sobre a utilização de Child scripts. Isto deve-se ao facto de os Child scripts permitirem a associação a um objeto específico no contexto da simulação, o que é particularmente relevante, uma vez que o algoritmo se destina ao controlo de um robô, considerado um objeto singular no ambiente simulado.

No que diz respeito à categoria *Child scripts*, é importante decidir entre os tipos *Non-threaded* e *Threaded*. Após a realização de várias simulações e uma análise comparativa, concluiu-se que o *Child script* do tipo *Non-threaded* é o mais adequado para o projeto em questão, pois demonstrou ser mais simples de implementar, tornando o processo de desenvolvimento mais eficiente e reduzindo a probabilidade de erros no código e verificou-se que o *Child script Non-threaded* proporciona uma simulação mais fluída e estável, resultando num comportamento mais consistente do robô durante a execução do algoritmo.

Esta conclusão é apoiada pelas recomendações dos desenvolvedores do simulador CoppeliaSim, que sugerem a preferência por scripts *Non-threaded* em situações semelhantes. Tal deve-se ao facto de os scripts do tipo *Threaded* poderem apresentar aspetos inerentes que podem afetar negativamente o desempenho durante a simulação. A Figura 44 no texto ilustra a inserção e a ligação do *Child script Non-threaded* ao robô no simulador.

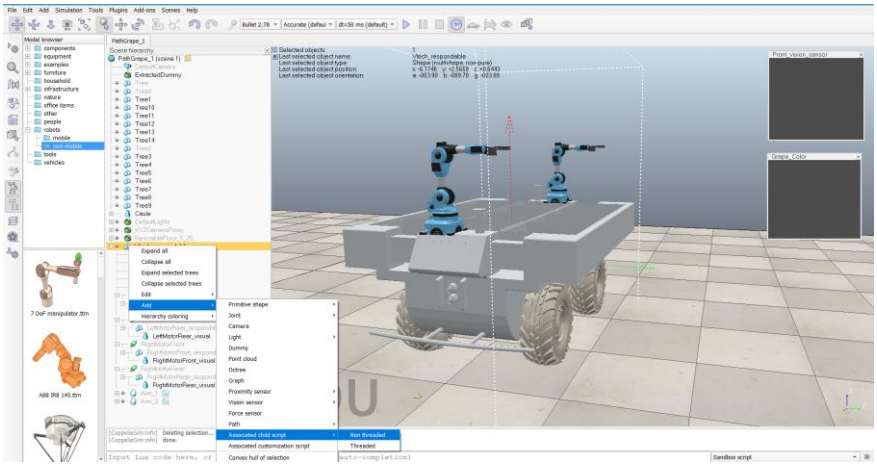


Figura 44 – Representação da inserção de um script

Após a associação do script ao robô, procedeu-se ao início da programação do algoritmo de controlo propriamente dito. Para evitar um subcapítulo extenso, serão apresentados apenas os passos mais relevantes adotados na criação do algoritmo. No entanto, o algoritmo completo pode ser consultado nos anexos deste documento (Anexo 1 – Programação do Primeiro Caso de Estudo, Anexo 2 – Programação do Segundo Caso de Estudo e Anexo 3 - Programação do Terceiro Caso de Estudo).

A fase inicial do algoritmo envolveu a declaração dos vários objetos a serem controlados durante a experiência. Para isso, utilizou-se a função "sim.getObjectHandle" da biblioteca de funções do API, a qual permite atribuir uma variável para o controlo de cada objeto

específico no simulador. A Figura 45 exibe a declaração dos objetos relevantes para a simulação do robô.

```
function sysCall_init()
    -- do some initialization here
    lfmotor=sim.getObjectHandle("LeftMotorFront")
    rfmotor=sim.getObjectHandle("RightMotorFront")
    lrmotor=sim.getObjectHandle("LeftMotorRear")
    rrmotor=sim.getObjectHandle("RightMotorRear")
    LeftFront=sim.getObjectHandle('LeftFront')
    LeftRear=sim.getObjectHandle('LeftRear')
    RightFront=sim.getObjectHandle('RightFront')
    RightRear=sim.getObjectHandle('RightRear')
    Forward=sim.getObjectHandle('Forward')
```

Figura 45 - Código para a declaração dos vários objetos

Adicionalmente, foram definidos os valores de algumas variáveis utilizadas no algoritmo, muitas vezes servindo apenas como valores iniciais, passíveis de serem alterados durante a simulação.

É relevante destacar que todo o código apresentado até agora foi inserido na função sysCall_init(), conforme recomendado pelo manual do utilizador do CoppeliaSim, onde se estipula que todos os objetos a serem usados na simulação devem ser declarados e definidos nesta função. Isso ocorre devido ao facto de os Child scripts do tipo Non-threaded serem compostos por quatro funções, como já foi mencionado no capítulo 5.3.2 Mecanismos de Controlo. Após a etapa inicial, todo o código subsequente foi inserido na função "sysCall actuation()", a qual é responsável por conter o código relacionado à atuação dos vários componentes do modelo do robô. Na fase inicial desta função, começou-se por criar um conjunto de variáveis que terá por intuito “medir” a posição ou velocidade de determinados objetos, que importam saber a sua exata posição no decorrer da simulação. Para isso foram utilizadas algumas funções específicas da biblioteca API, observadas na Figura 46.

```
--GPS ACTUATION
positionX = sim.getFloatSignal('gpsX')
positionY = sim.getFloatSignal('gpsY')
positionZ = sim.getFloatSignal('gpsZ')

 --ACELEROMETER ACTUATION
xAccel=sim.getFloatSignal('accelerometerX')
yAccel=sim.getFloatSignal('accelerometerY')
zAccel=sim.getFloatSignal('accelerometerZ')

print ('Position X: ', positionX)
print ('Aceleration X: ', xAccel)

print ('Position Y: ', positionY)
print ('Aceleration Y: ', yAccel)

print ('Position Z: ', positionZ)
print ('Aceleration Z: ', zAccel)

-- Call to set the left and right motor velocities

    --INITIAL VELOCITY OF ROBOT
    sim.setJointTargetVelocity(lfmotor,velStraight)
    sim.setJointTargetVelocity(rfmotor,velStraight)
    sim.setJointTargetVelocity(lrmotor,velStraight)
    sim.setJointTargetVelocity(rrmotor,velStraight)
```

Figura 46 - Código criado para a definição das varáveis que irão medir posições

Com o algoritmo devidamente desenvolvido e implementado, o controlo do robô é otimizado, permitindo uma maior eficiência e precisão nas suas operações dentro de ambientes complexos.

5.5.3 Criação da Graphical User Interface (GUI)

Foi introduzida uma nova funcionalidade denominada "simUI.create", com o propósito de possibilitar a criação de uma GUI durante a simulação. Essa interface oferece informações instantâneas sobre alguns parâmetros em processo de simulação e pode inclusive permitir a inclusão de novos dados para a simulação, constituindo-se como uma ferramenta de suporte ao utilizador.

Determinadas variáveis serão empregadas na GUI, cuja representação da programação está ilustrada na Figura 47 e a janela que aparece está representada na Figura 48, exibindo as informações fornecidas. Além disso, a GUI proporciona a capacidade ao operador de requerer o retorno do robô à base, em caso de ser necessário devido a alguma eventualidade.

```
ui = '<ui enabled="true" modal="false" title="Comando Virtual" closeable="true" layout="vbox" placement="relative" position="50,50">'..
'<label text="Num. Frutas ate ir a base: " id="1"/>'..
'<label text="Distancia da base: " id="2"/>'..
'<label text="Velocidade: " id="3"/>'..
'<button text="Ir a Base" on-click="DevolveBase" checkable="true" id="4"/>'..
'</ui>'

simUI.create(ui)
```

Figura 47 - Código para a criação da GUI

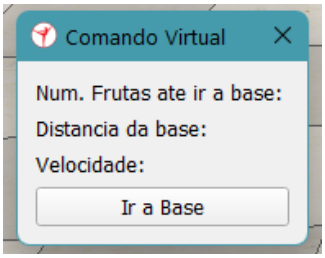


Figura 48 - Comando criado para a simulação

5.5.4 Verificação das Zonas A, B e C da câmara de Video

Posteriormente, a etapa subsequente compreendeu a formulação de um código para verificar a presença de objetos que exigem deteção nas áreas A, B e C da câmara de vídeo, bem como definir as instruções a serem seguidas caso essa situação ocorra. Nesse contexto, uma das instruções é determinar a velocidade operacional do robô, para a qual foram propostas três opções de velocidade: "mínima", "média" e "máxima."

Essa implementação tem como objetivo garantir o cumprimento dos requisitos lógicos estabelecidos na fase inicial do fluxograma, conforme ilustrado na Figura 43. Importante destacar que esse código é executado no início da simulação e sempre que houver a deteção de um objeto.

A avaliação mencionada é conduzida através da análise de segmentos da imagem capturada pela câmara de vídeo. Para determinar o tamanho ideal da imagem a ser analisada, foram realizados vários testes, concluindo-se que um tamanho de 30x30 pixels é o mais apropriado. Esta escolha visa evitar a deteção de objetos não relevantes, como ramos e troncos, em imagens mais pequenas, e também reduzir a carga computacional, uma vez que imagens excessivamente grandes abrangeriam uma área maior do que a necessária para a recolha dos objetos.

Assim, ao capturar uma imagem parcial de 30x30 pixels, são realizadas duas verificações cruciais:

1. O valor médio dos pixels capturados deve corresponder a uma gama de cores previamente definida;
2. O valor médio dos pixels capturados deve estar a uma distância mínima, de acordo com os padrões preestabelecidos.

Se a imagem analisada não cumprir esses requisitos, o ciclo "for" repetirá o procedimento para os pixels subsequentes, prosseguindo dessa forma até analisar todos os pixels da

imagem, como ilustrado na Figura 49. Caso, em algum momento, a imagem parcial cumpra os requisitos, a condição "Verdadeiro" será estabelecida, permitindo que a simulação prossiga para a próxima etapa.

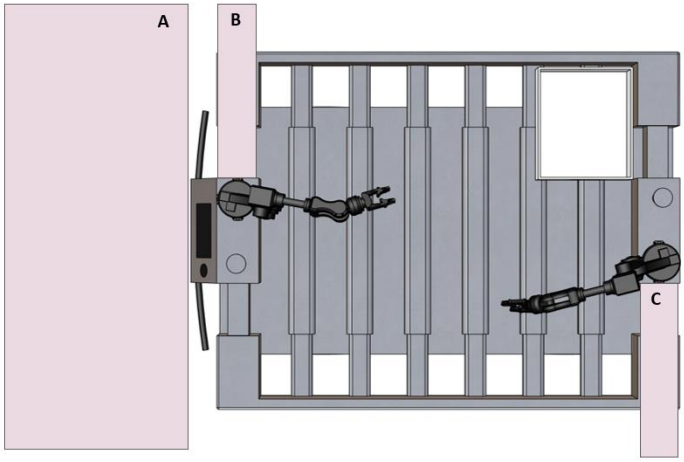


Figura 49 - Representação gráfica das zonas de visão

5.5.5 Recolha de objetos

Para a realização da tarefa de recolha de objetos, foi definido um conjunto específico de movimentos que o robô deve executar. Esses movimentos estão descritos de forma detalhada no fluxograma apresentado na Figura 50.

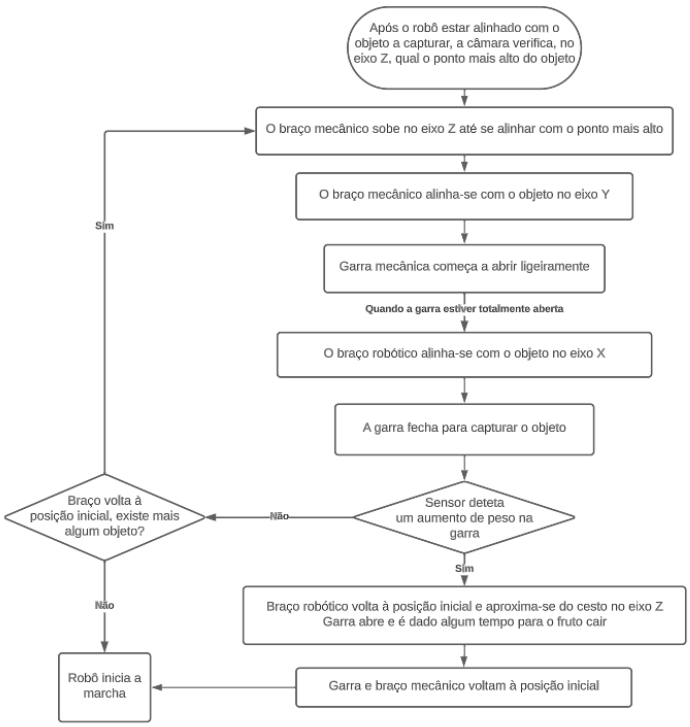


Figura 50 - Fluxograma para recolha de objetos

Entre os movimentos preestabelecidos, é de suma importância a justificação de determinadas escolhas. Por exemplo, no início do fluxograma, merece destaque a ação da câmara que consiste na verificação do ponto mais elevado ao longo do eixo Z. Esta ação é fundamentada no pressuposto que o robô já se encontra devidamente alinhado com o objeto no eixo X, contudo, é imprescindível assegurar que o braço manipulador (responsável pela movimentação no eixo Y) esteja corretamente alinhado com a garra mecânica em relação ao objeto em questão.

Após a identificação do ponto mais elevado, a garra mecânica inicia sua ascensão ao longo do eixo Y e, posteriormente, desloca-se na sua direção ao longo do eixo X, até que esteja totalmente alinhado para a apanha do objeto (cacho).

Outro aspeto importante a ser esclarecido no fluxograma é a transição do primeiro passo para o segundo. Verificou-se a possibilidade do braço, ao subir, entrar em contato com o objeto, tornando-se necessário elevar primeiro a garra mecânica a fim de o ultrapassar o e somente, em seguida, abrir as pinças. Essa abordagem visa encontrar a melhor posição e orientação do elemento terminal, para a de forma a abrir as pinças da garra e capturar apenas o caule do cacho sem danificar as uvas.

5.5.6 Retornar à Base

Foi elaborada uma função com o objetivo de permitir que o robô retorne à base em duas situações distintas: quando o mesmo atingir a capacidade máxima de objetos recolhidos ou quando o operador emitir a ordem de retorno. Após regressar à base, os operadores procedem à descarga de todos os objetos coletados e, em seguida, o robô voltará à sua posição anterior com a velocidade ajustada para o valor máximo.

5.5.7 Diferenciação das Cores

A paleta de cores empregue no CoppeliaSim corresponde ao sistema RGB, o qual constitui um sistema cromático aditivo que combina as componentes de vermelho (Red), verde (Green) e azul (Blue) em diferentes níveis de intensidade, a fim de reproduzir um amplo espectro de cores.

A intensidade de cada cor varia de 0 a 255, onde 0 representa a total ausência desse componente e 255 corresponde à sua intensidade máxima. No entanto, no CoppeliaSim, a escala de cores opera num intervalo de 0 a 1, representando a proporção da respetiva intensidade na gama 0 a 255.

Assim, para realizar a deteção de objetos, tornou-se necessário analisar a escala de cores associada aos objetos utilizados em cada uma das tarefas. Os valores obtidos pelo sensor de visão para cada um desses objetos encontram-se detalhadamente descritos na Tabela 10.

Tabela 10 - Código de cor para os cachos maduros

Objeto	**Código da Cor**		
	Vermelho	Verde	Azul
Cacho maduro	0.71	0.00	0.66

5.6 Simulação de cenários de navegação

5.6.1 Robô autónomo a mover-se pela vinha

Neste caso de estudo foi estudada a capacidade de o robô conseguir mover-se de forma autónoma pela vinha. Para isso, foi colocado num ambiente com linhas de vinha e movido num trajeto de 10 metros, mantendo-se no centro dessa entrelinha representado na Figura 51.

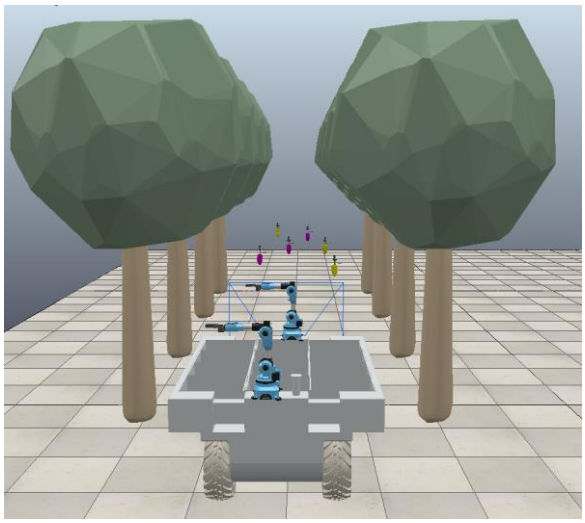


Figura 51 - Cenário da simulação do robô a mover-se pela vinha

5.6.2 Colher as uvas

No presente caso de estudo, foi testada a capacidade do algoritmo para recolher objetos, representando os cachos de uvas. Como tal foi colocado um conjunto de 5 objetos a

recolher, espalhados aleatoriamente ao longo de um trajeto de 10 metros que o robô irá percorrer.

De salientar que foi definida a cor que representa o cacho maduro, para o robô conseguir distinguir os cachos maduros dos não maduros e apenas colher os que estão prontos para a colheita.

Para o presente caso de estudo, foram adicionados dois possíveis cenários:

- **Cenário 1:** Objetos com pesos iguais e cores diferentes
- **Cenário 2:** Objetos com pesos diferentes e cores diferentes

De salientar ainda que, para cada um dos cenários foram realizados três testes diferentes, em que cada um do teste os objetos eram colocados em posições aleatórias tal como o parâmetro que estaria em análise.

5.6.2.1 Cenário 1

Na realização deste primeiro cenário foram colocados os 5 objetos com um peso de 350g. A escolha deste peso é justificada por ser o valor médio que os cacho de uvas podem pesar, tal como descrito no capítulo 5.4.1 Criação do modelo.

A gama de cores utilizada para representação dos cachos maduros foi a demonstrada na Tabela 10, e as posições que foram dadas aos objetos para cada um dos testes é a representada na Figura 52.

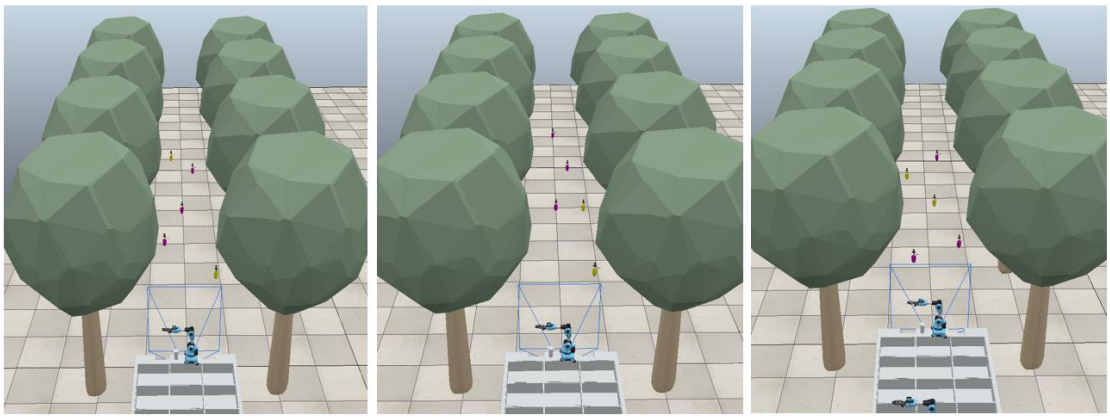


Figura 52 - Testes para o cenário 1

5.6.2.2 Cenário 2

Neste segundo cenário foram atribuídos pesos que variam entre os 200 e 450g aos 5 objetos. A gama de cores utilizada para representação dos cachos maduros foi a

apresentada na Tabela 10 e as posições dos objetos em cada teste é a representada na Figura 53.

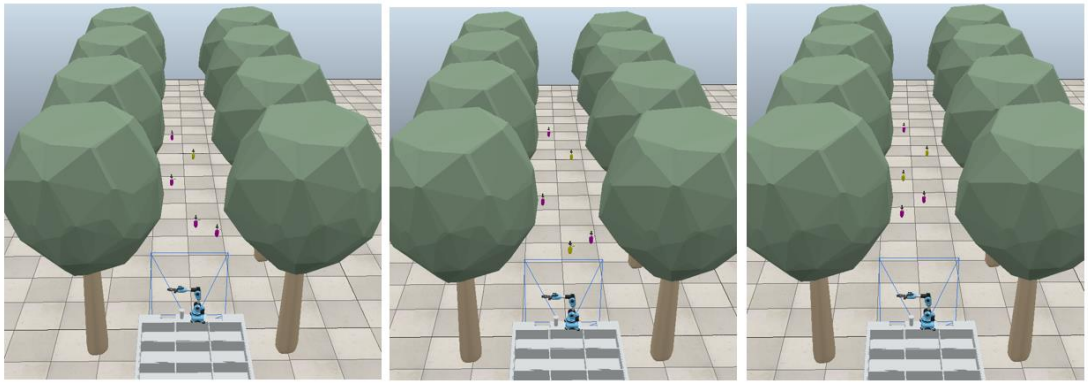


Figura 53 - Testes para o cenário 2

5.6.3 Deteção e contorno de obstáculos

Este teste foi realizado num ambiente novo, sendo que foi definido o ponto inicial e final do percurso do robô, mas considerado a existência de objetos no caminho do mesmo, para demonstrar a capacidade deste se desviar para evitar colisões e regressar ao caminho que estava a percorrer, representado na Figura 54.

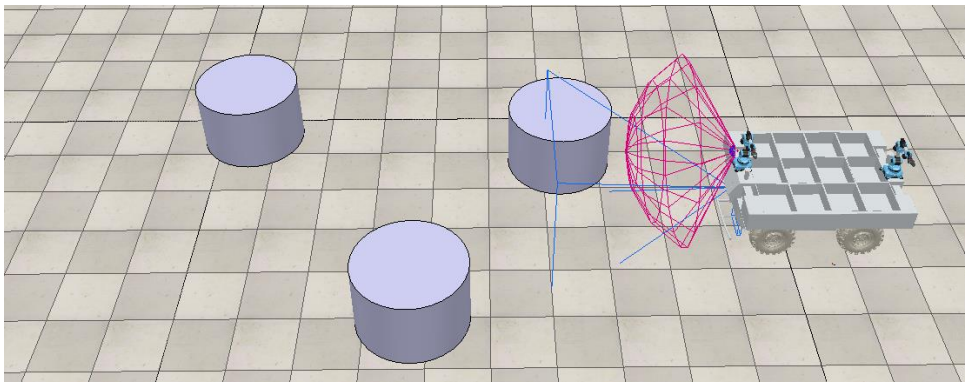


Figura 54 - Cenário para a simulação de deteção e contorno de obstáculos

5.7 Análise e Discussão de Resultados

5.7.1 Resultados dos Casos de Estudo

Após a exposição das principais problemáticas emergentes ao longo das várias simulações conduzidas e a apresentação das simulações efetuadas para os dois cenários de estudo, é cabível inferir que o algoritmo proposto obteve um desempenho notável. Em ambas as instâncias de estudo, o algoritmo demonstrou consistentemente a capacidade de detetar os objetos visados para a colheita, bem como de executar com sucesso as etapas subsequentes à deteção dos referidos objetos.

5.7.2 Robô autónomo a mover-se pela vinha

A pesquisa conduzida proporcionou uma análise detalhada da simulação de um robô autónomo em movimento dentro do contexto de uma vinha. Os resultados deste estudo, baseado numa observação cuidadosa e aprofundada, permitem-nos afirmar que a simulação do referido robô autónomo não se deparou com desafios substanciais ou complexidades agravadas.

Especificamente, a tarefa de programação do robô para simular o seu percurso entre as linhas da vinha revelou-se uma tarefa acessível. O sistema de controlo desenvolvido demonstrou com sucesso a capacidade de manter o robô consistentemente posicionado no centro da entrelinha da vinha, cumprindo o objetivo predefinido para esta simulação.

Este êxito na simulação do deslocamento do robô pela vinha reflete a eficácia do sistema de controlo em manter o alinhamento desejado, garantindo que o robô percorra as entrelinhas de forma precisa e eficaz. Além disso, destaca a competência da programação adotada, que conseguiu replicar com sucesso o comportamento pretendido para o robô autónomo no ambiente de simulação.

Foi possível determinar o tempo de operação necessário para que o robô alcance o final da linha de vinhas. Realizaram-se 3 testes com o intuito de obter resultados mais precisos. Os tempos em questão encontram-se identificados na Tabela 11.

Tabela 11 - Tempos decorridos na realização do caso de estudo 1

Caso de estudo 1	**Tempo (s)**
Teste 1	30
Teste 2	28
Teste 3	31

No entanto, é fundamental ressalvar que esta análise se concentra na fase de simulação e programação do robô autónomo, sem considerar necessariamente as complexidades que podem surgir num cenário real, como variações no terreno ou a presença de obstáculos imprevistos, entre outros fatores. Portanto, embora a simulação tenha decorrido de forma satisfatória, é importante reconhecer que a implementação prática deste sistema pode implicar desafios adicionais que exigem considerações e adaptações específicas.

5.7.3 Colher as uvas

Neste caso de estudo a deteção dos objetos foi conseguida nos diferentes cenários que se propunham. Incluindo no cenário 2, que seria o que representa maior dificuldade, uma vez que todos os objetos eram diferentes em cor e dimensão. Referir que o robô apenas apresentou dificuldade identificada na recolha dos objetos maiores, especialmente com pesos superiores a 400g. Para solucionar essa dificuldade foi necessário aumentar ligeiramente o ângulo de abertura das juntas da garra mecânica.

Nas diversas simulações efetuadas em cada cenário, determinaram-se os tempos de operação necessários para que o robô concluísse a tarefa de recolha dos objetos. Para cada um dos cenários foram realizados três testes diferentes, representados anteriormente. Estes tempos estão devidamente identificados na Tabela 12.

Tabela 12 - Tempos decorridos na realização do caso de estudo 2

Caso de estudo 2		**Tempo**
Cenário 1	Teste 1	6m20s
	Teste 2	6m15s
	Teste 3	6m18s

Cenário 2	Teste 1	6m21s
	Teste 2	6m28s
	Teste 3	6m30s

5.7.4 Deteção e contorno de obstáculos

A simulação relativa à deteção e contorno de obstáculos revelou-se insuficiente em termos de robustez para atender às exigências e necessidades específicas deste estudo em questão. Este desafio decorre, em grande parte, da escassez de informações disponíveis, particularmente quando se trata de aplicar a deteção e contorno de obstáculos em ambientes agrícolas reais, onde a variedade e complexidade dos obstáculos são significativas.

Embora tenha sido possível desenvolver uma simulação, esta não consegue reproduzir integralmente as complexas condições do ambiente real. A dificuldade em obter dados e parâmetros precisos sobre a deteção e contorno de obstáculos em ambientes agrícolas limitou a capacidade da simulação em replicar fielmente os desafios que um robô autónomo poderá enfrentar no terreno.

Portanto, é evidente a necessidade de aprofundar a investigação nesta área, explorando soluções mais avançadas e robustas que possam lidar de forma eficaz com a deteção e contorno de obstáculos em ambientes agrícolas diversificados e desafiadores. Este é um aspeto crítico para a implementação bem-sucedida de sistemas robóticos autónomos na agricultura, dada a importância de evitar colisões e otimizar a navegação em espaços agrícolas complexos.

6 Conclusões e perspetivas de desenvolvimentos futuros

No presente capítulo, serão apresentadas as principais conclusões decorrentes da pesquisa realizada, destacando as principais dificuldades encontradas ao longo do desenvolvimento do estudo e delineadas algumas sugestões para investigações futuras.

6.1 Conclusões

As simulações realizadas, do robô autónomo a mover-se pela vinha, da colheita das uvas testada em dois possíveis cenários e da deteção e contorno de obstáculos, foram conseguidas com maior ou menor nível de realismo.

A principal complexidade enfrentada durante a realização deste estudo centrou-se na criação dos modelos virtuais do sistema. Apesar de parecer uma das fases mais simples do trabalho, o seu desenvolvimento foi bastante desafiante. A maior dificuldade emergiu, sobretudo, devido à necessidade de compreender a implementação correta das cadeias dinâmicas dos modelos no ambiente de simulação. A seleção adequada dos valores de inércia e massa das formas exerceu uma influência substancial no comportamento das referidas cadeias. Por vezes, a disparidade nos valores de inércia e massa entre as formas resultava em comportamentos indesejados no sistema, como vibrações. Adicionalmente, ao criar uma ligação rígida para simular a ação de agarrar objetos pelas garras, verificou-se a produção de considerável ruído nas leituras do sensor de força. Para solucionar este problema, foi necessário atribuir valores às propriedades dos elementos envolvidos na simulação que não representam a realidade, mas que foram escolhidos de forma a garantir a estabilidade dos modelos. Esta abordagem implicou a necessidade de ajustes manuais demorados e poderá constituir um obstáculo na criação de modelos que se aproximem o mais possível da realidade.

O estudo efetuado revelou que a simulação do robô autónomo a mover-se pela vinha, não apresentou grandes problemas. A sua programação não foi demasiado complexa tendo-se conseguido a simulação do percurso entre as linhas de vinha, mantendo-se o robô sempre no centro dessa entrelinha.

Relativamente à simulação de colheita das uvas, encontram-se algumas dificuldades tendo-se testado dois possíveis cenários. No cenário 1 consideraram-se objetos com pesos

iguais e cores diferentes e no cenário 2 consideraram-se objetos com pesos e cores diferentes.

No cenário 1 a maior dificuldade foi a deteção apenas dos cachos que estavam prontos para a colheita, evitando a deteção de objetos não relevantes. Isso implicou a análise de segmentos da imagem capturada pela câmara de vídeo de modo a se conseguir o tamanho ideal da imagem a ser analisada, tendo-se concluído que o tamanho de 30x30 pixels é o mais apropriado.

No cenário 2, tendo-se considerado objetos de pesos diferentes, além da cor, a maior dificuldade encontrada foi na recolha dos objetos maiores, tendo sido necessário aumentar ligeiramente o ângulo de abertura das pinças da garra.

A simulação de deteção e contorno de obstáculos realizada não apresenta a robustez necessária e desejada para o presente caso de estudo. Devido à dificuldade em encontrar informação disponível sobre esta temática, principalmente aplicada em situação de campo onde os obstáculos podem ser muito diversificados. Conseguiu-se realizar uma simulação, mas que não representa por completo as condições reais.

Após a pesquisa nos fóruns da Coppelia Robotics em busca de uma solução, constatou-se que o motor de física Vortex Studio lida de forma mais eficaz com essas questões e é considerado o mais realista entre as opções disponíveis no simulador. No entanto, vale ressaltar que este é o único motor que requer a aquisição de uma licença para uso.

6.2 Trabalhos Futuros

Conforme mencionado anteriormente, uma das etapas cruciais para a continuação deste projeto consiste na integração dos algoritmos previamente desenvolvidos no protótipo do robô. Este processo implica ajustes substanciais, uma vez que os algoritmos dependem de variáveis que monitorização em tempo real as posições das juntas prismáticas, da garra mecânica, do sensor de força no eixo Z e até mesmo das medições de velocidade das rodas. Portanto, pode ser necessário incorporar sensores adicionais no protótipo do robô.

Da mesma forma, é essencial abordar a obtenção precisa da localização em tempo real do robô. No contexto da simulação, *dummies* foram utilizados para determinar a sua posição, assim como a localização da base (onde a simulação é iniciada e os objetos são descarregados) e o ponto limite que o robô pode alcançar. Isso permitiu calcular as

distâncias máximas a serem percorridas, bem como determinar a sua localização e velocidade. Entretanto, em uma aplicação do mundo real, para explorar essas funcionalidades, seria necessário equipar o robô com um GPS.

Ao replicar esses algoritmos no cenário real, seria possível analisar as discrepâncias entre os resultados obtidos no simulador e os resultados reais, especialmente no que se refere aos processos de deteção de objetos. Na recolha de objetos, seria viável avaliar o desempenho real da garra mecânica.

Adicionalmente, no que se refere à recolha de objetos, seria pertinente incorporar um sensor de peso/força entre o robô e os cestos de recolha. Com base nessas informações e na tarefa em questão, seria possível emitir comandos para que o robô retorne à base ou acione um alerta ao operador. Essas implementações seriam relativamente simples, uma vez que o algoritmo já inclui uma sub-rotina para quando o número máximo de objetos a serem coletados é atingido, sendo apenas necessário adicionar essa nova condição.

Uma área de investigação adicional com potencial seria a conceção de um algoritmo, no qual o utilizador só precisasse definir a área total do campo em que o robô iria operar, para recolha objetos ou para outras tarefas, enquanto evita os obstáculos. Simultaneamente, esse algoritmo poderia realizar autonomamente o mapeamento de toda a área de atuação, permitindo que o utilizador não tivesse de se preocupar constantemente com o progresso do trabalho.

Por fim, uma melhoria adicional poderia consistir na implementação de um sistema mais robusto para a deteção de objetos a serem recolhidos (cachos maduros).

Bibliografia

[1] N. M. Hackenhaar, C. Hackenhaar, e Y. Vieira De Abreu, «Robótica na agricultura Robotics in agriculture», pp. 119–129, 2014, [Em linha]. Disponível em: http://dx.doi.org/10.1590/1518-70122015110.

[2] R. Braga, *Viticultura de Precisão*, 1º. Lisboa, 2009.

[3] S. S. Valle e J. Kienzle, «Agriculture 4.0», vol. 24, p. 40, 2020.

[4] V. Gupta, «Difference between AMR and AGV», 2020. https://www.geeksforgeeks.org/difference-between-amr-and-agv/ (acedido Abr. 20, 2022).

[5] J. Walker, «AMR vs AGV: A Clear Choice for Flexible Material Handling», 2021. https://locusrobotics.com/amr-vs-agv/ (acedido Abr. 20, 2022).

[6] L. F. P. Oliveira, A. P. Moreira, e M. F. Silva, «Advances in agriculture robotics: A state-of-the-art review and challenges ahead», *Robotics*, vol. 10, n. 2, pp. 1–31, 2021, doi: 10.3390/robotics10020052.

[7] A. Botta, P. Cavallone, L. Baglieri, G. Colucci, e L. Tagliavini, «A Review of Robots , Perception , and Tasks in Precision Agriculture», pp. 830–854, 2022.

[8] I. Cisternas, I. Velásquez, A. Caro, e A. Rodríguez, «Systematic literature review of implementations of precision agriculture», *Comput. Electron. Agric.*, vol. 176, n. July, p. 105626, 2020, doi: 10.1016/j.compag.2020.105626.

[9] Robotics Online Marketing Team, «Robotics in Agriculture: Types and Applications», 2017. https://www.automate.org/blogs/robotics-in-agriculture-types-and-applications (acedido Abr. 21, 2022).

[10] N. Khan, G. Medlock, e S. Graves, «GPS Guided Autonomous Navigation of a Small Agricultural Robot with Automated Fertilizing System», pp. 1–7, 2018, doi: 10.4271/2018-01-0031.Abstract.

[11] M. U. Hassan, M. Ullah, e J. Iqbal, «Towards Autonomy in Agriculture : Design and Prototyping of a Robotic Vehicle with Seed Selector», 2016.

[12] S. Hayashi *et al.*, «Field operation of a movable strawberry-harvesting robot using a travel platform», *Japan Agric. Res. Q.*, vol. 48, n. 3, pp. 307–316, 2014, doi: 10.6090/jarq.48.307.

[13] N. Noguchi, J. F. Reid, K. Ishii, e H. Terao, «Multi-Spectrum Image Sensor for Detecting Crop Status by Robot Tractor», *IFAC Proc. Vol.*, vol. 34, n. 19, pp. 111–115, 2001, doi: 10.1016/S1474-6670(17)33122-1.

[14] P. Makers, «greenbot @ precisionmakers.com», 2015. https://precisionmakers.com/en/greenbot.

[15] DJI, «mg-1p @ www.dji.com», 2016. https://www.dji.com/pt/mg-1p.

[16] S. Sukkarieh, «Mobile on-farm digital technology for smallholder farmers», n. 2059-2018–203, p. 9, 2017.

[17] L. Haibo, D. Shuliang, L. Zunmin, e Y. Chuijie, «Study and Experiment on a Wheat Precision Seeding Robot», *J. Robot.*, vol. 2015, 2015, doi: 10.1155/2015/696301.

[18] S. K. Pilli, B. Nallathambi, S. J. George, e V. Diwanji, «2nd International Conference on Electronics and Communication Systems, ICECS 2015», *2nd Int. Conf. Electron. Commun. Syst. ICECS 2015*, n. Icecs, pp. 1684–1689, 2015.

[19] M. Preudhomme, «Autonomous Mechanical Weeding Robot», *naio Technol.*, 2019.

[20] N. Technologies, «Multifunctional straddling vineyard robot», 2020.

[21] VITIROVER Solutions, «VITIROVER—A Revolution in Soil Grassing Management. 2020.», 2020. https://www.vitirover.fr/en-home.

[22] G. Adamides *et al.*, «Design and development of a semi-autonomous agricultural vineyard sprayer: Human–robot interaction aspects», *J. F. Robot.*, vol. 34, n. 8, pp. 1407–1426, 2017, doi: 10.1002/rob.21721.

[23] R. Berenstein e Y. Edan, «Automatic Adjustable Spraying Device for Site-Specific Agricultural Application», *IEEE Trans. Autom. Sci. Eng.*, vol. 15, n. 2, pp. 641–650, 2018, doi: 10.1109/TASE.2017.2656143.

[24] R. Bogue, «Robots poised to revolutionise agriculture», *Ind. Rob.*, vol. 43, n. 5, pp. 450–456, 2016, doi: 10.1108/IR-05-2016-0142.

[25] C. Lehnert, C. McCool, I. Sa, e T. Perez, «Performance improvements of a sweet pepper harvesting robot in protected cropping environments», *J. F. Robot.*, vol. 37, n. 7, pp. 1197–1223, 2020, doi: 10.1002/rob.21973.

[26] S. Birrell, J. Hughes, J. Y. Cai, e F. Iida, «A field-tested robotic harvesting system for iceberg lettuce», *J. F. Robot.*, vol. 37, n. 2, pp. 225–245, 2020, doi: 10.1002/rob.21888.

[27] B. Arad *et al.*, «Development of a sweet pepper harvesting robot», *J. F. Robot.*, vol. 37, n. 6, pp. 1027–1039, 2020, doi: 10.1002/rob.21937.

[28] R. K. Megalingam *et al.*, «Amaran:An Unmanned Robotic Coconut Tree Climber and Harvester», *IEEE/ASME Trans. Mechatronics*, vol. 26, n. 1, pp. 288–299, 2021, doi: 10.1109/TMECH.2020.3014293.

[29] C. M. Lopes *et al.*, «Vineyard yield estimation by VINBOT robot - preliminary results with the white variety Viosinho», *Proc. 11th Int. Terroir Congr. Jones, G. Doran, N.(eds.), pp. 458-463. South. Oregon Univ. Ashland, USA.*, n. July, 2016.

[30] A. Shafiekhani, S. Kadam, F. B. Fritschi, e G. N. Desouza, «Vinobot and vinoculer: Two robotic platforms for high-throughput field phenotyping», *Sensors (Switzerland)*, vol. 17, n. 1, pp. 1–23, 2017, doi: 10.3390/s17010214.

[31] V. A. H. Higuti, A. E. B. Velasquez, D. V. Magalhaes, M. Becker, e G. Chowdhary, «Under canopy light detection and ranging-based autonomous navigation», *J. F. Robot.*, vol. 36, n. 3, pp. 547–567, 2019, doi: 10.1002/rob.21852.

[32] A. Matese e S. F. Di Gennaro, «Technology in precision viticulture: A state of the art review», *Int. J. Wine Res.*, vol. 7, n. 1, pp. 69–81, 2015, doi: 10.2147/IJWR.S69405.

[33] D. Sarri, L. Martelloni, M. Rimediotti, R. Lisci, S. Lombardo, e M. Vieri, «Testing a multi-rotor unmanned aerial vehicle for spray application in high slope terraced vineyard on co m m er ci al u se on ly on al», vol. L, 2019, doi: 10.4081/jae.2019.853.

[34] F. Roure, M. Soler, D. Serrano, e P. Astolfi, «GRAPE : Ground Robot for vineyArd Monitoring and ProtEction GRAPE : Ground Robot for vineyArd», n. February, 2018, doi: 10.1007/978-3-319-70833-1.

[35] E. Vrochidou *et al.*, «An Autonomous Grape-Harvester Robot: Integrated System Architecture», *Electronics*, vol. 10, n. 9, p. 1056, 2021, doi: 10.3390/electronics10091056.

[36] J. Kim, S. Kim, C. Ju, H. I. L. Son, e S. Member, «Unmanned Aerial Vehicles in Agriculture : A Review of Perspective of Platform , Control , and Applications», vol. 7, 2019.

[37] N. Baiju, «Top 12 Agricultural Robots For Vineyard Applications», 2019. https://roboticsbiz.com/top-12-agricultural-robots-for-vineyard-applications/.

[38] C. M. Lopes, «VINEYARD YIELD ESTIMATION BY VINBOT ROBOT - PRELIMINARY RESULTS WITH THE WHITE VARIETY VIOSINHO», n. July, 2016, doi: 10.13140/RG.2.1.3912.0886.

[39] J. Tardaguila, «A wheeled robot to monitor grape growth», 2017. https://cordis.europa.eu/article/id/122526-a-wheeled-robot-to-monitor-grape-growth.

[40] N. H. Agriculture, «Nova GAMA COMPACTA Gama compacta Braud . vindimas inteligentes .», p. 28, 2015.

[41] S. Best, O. Ringdahl, e R. Oberti, «CROPS : Clever Robots for Crops», n. January, 2015, doi: 10.1049/etr.2015.0015.

[42] L. F. P. Oliveira, A. P. Moreira, e M. F. Silva, «Agricultural robotics: a state of the art survey», n. August, pp. 24–26, 2020.

[43] J. G. Mooney e E. N. Johnson, «A Comparison of Automatic Nap-of-the-earth Guidance Strategies for Helicopters», *J. F. Robot.*, vol. 33, n. 1, pp. 1–17, 2014, doi: 10.1002/rob.

[44] M. F. Silva e J. A. T. MacHado, «A literature review on the optimization of legged robots», *JVC/Journal Vib. Control*, vol. 18, n. 12, pp. 1753–1767, 2012, doi: 10.1177/1077546311403180.

[45] Robotics Technology, «ROBOTICS OVERVIEW», 2020. https://builtin.com/robotics (acedido Abr. 21, 2022).

[46] K. Legun e K. Burch, «Robot-ready : How apple producers are assembling in anticipation of new AI robotics», *J. Rural Stud.*, vol. 82, pp. 380–390, 2021, doi: 10.1016/j.jrurstud.2021.01.032.

[47] H. Mousazadeh, «A technical review on navigation systems of agricultural autonomous off-road vehicles», *J. Terramechanics*, vol. 50, n. 3, pp. 211–232, 2013, doi: 10.1016/j.jterra.2013.03.004.

[48] A. Lopes, «Universidade Técnica de Lisboa Universidade do Porto Previsão quantitativa de vindimas António José de Oliveira Lopes Mestrado em Viticultura e Enologia Lisboa», 2009.

[49] Last minute engineers, «How DHT11 DHT22 Sensors Work & Interface With Arduino», 2020. https://lastminuteengineers.com/dht11-dht22-arduino-tutorial/.

[50] L. E. Montoya-Cavero, R. Díaz de León Torres, A. Gómez-Espinosa, e J. A. Escobedo Cabello, «Vision systems for harvesting robots: Produce detection and localization», *Comput. Electron. Agric.*, vol. 192, n. November 2021, 2022, doi: 10.1016/j.compag.2021.106562.

[51] Q. Wang *et al.*, «Interactive effects of porosity and microstructure on strength of 6063 aluminum alloy .pdf». 2022, doi: https://doi.org/10.1016/S1003-6326(22)65834-5.

[52] K. Li, Y. Huo, Y. Liu, Y. Shi, Z. He, e Y. Cui, «Design of a lightweight robotic arm for kiwifruit pollination», *Comput. Electron. Agric.*, vol. 198, n. June, p.

107114, 2022, doi: 10.1016/j.compag.2022.107114.

Anexo 1 – Programação do Primeiro Caso de Estudo

```
function sysCall_init()
    -- do some initialization here
    lfmotor=sim.getObjectHandle("LeftMotorFront")
    rfmotor=sim.getObjectHandle("RightMotorFront")
    lrmotor=sim.getObjectHandle("LeftMotorRear")
    rrmotor=sim.getObjectHandle("RightMotorRear")
    LeftFront=sim.getObjectHandle('LeftFront')
    LeftRear=sim.getObjectHandle('LeftRear')
    RightFront=sim.getObjectHandle('RightFront')
    RightRear=sim.getObjectHandle('RightRear')
    Forward=sim.getObjectHandle('Forward')

    leftSensor = sim.getObjectHandle('LeftSensor')
    rightSensor = sim.getObjectHandle('RightSensor')
    middleSensor = sim.getObjectHandle('MiddleSensor')
    print(leftSensor)

    --Velocity of the robot
    velStraight = 0.8
    velLeft = 1.2
    velRight = 1.2
    velSharp = 0.965
    vel_turn = 1.4

    avg_default = 0.15
    fwd_default = 100
    distLeftFront = avg_default
    distRightFront = avg_default
    distLeftRear = avg_default
    distRightRear = avg_default
    fwd = fwd_default
    diffLeft = 0

    -- Grape Parameters
    goodColor = {0.71,0.0,0.66}
    badColor = {0.79,0.79,0.0}

    -- Initialize auxiliary variables
    T_last_inserted = 0
    deltaTime = 0
    hasStopped = false
    boxList = {}
    boxDummyList = {}
    boolList = {}
    uva=0
    vel=0
    base={0}
    robot={0}
```

```
    -- Initialize handles, set Vtech Velocity

    proximity = sim.getObjectHandle('Grape_Detect')
    ColorDetect = sim.getObjectHandle('Grape_Color')
    OS_default = 0.15
    objectSensor=OS_default

    ui = '<ui enabled="true" modal="false" title="Comando Virtual" closeable="true" layout="vbox" placement="relative" position="50,50">'..
    '<label text="Num. Frutas ate ir a base: " id="1"/>'..
    '<label text="Distancia da base: " id="2"/>'..
    '<label text="Velocidade: " id="3"/>'..
    '<button text="Ir a Base" on-click="DevolveBase" checkable="true" id="4"/>'..
    '</ui>'

    simUI.create(ui)


end
function sysCall_actuation()

    -- put your actuation code here
       simUI.setLabelText(ui,1,string.format("Num. Frutas ate ir a base: %.2f"),-1*(uva-5))
       simUI.setLabelText(ui,2,string.format("Distancia da base [m]: %.2f",(robot[1]-base[1])))
       simUI.setLabelText(ui,3,string.format("Velocidade[m/s]: %.2f",velStraight))

    -- Call to set the left and right motor velocities

        --INITIAL VELOCITY OF ROBOT
        sim.setJointTargetVelocity(lfmotor,velStraight)
        sim.setJointTargetVelocity(rfmotor,velStraight)
        sim.setJointTargetVelocity(lrmotor,velStraight)
        sim.setJointTargetVelocity(rrmotor,velStraight)

```

```
       --DETECT IF ITS IN THE MIDDLE

       if (data_Left ~= nil) then
           intensity_left = data_Left [12]
           intensity_right = data_Right [12]
           intensity_middle = data_Middle [12]


           if (intensity_right >= 1 and intensity_left ~= 1) then
               print ('Turn Left')
               sim.setJointTargetVelocity(lfmotor,velLeft)
               sim.setJointTargetVelocity(lrmotor,velLeft)
               sim.setJointTargetVelocity(rfmotor,velRight - vel_turn)
               sim.setJointTargetVelocity(rrmotor,velRight - vel_turn)

           elseif (intensity_left >= 1 and intensity_right ~= 1) then
               print ('Turn Right')
               sim.setJointTargetVelocity(lfmotor,velLeft - vel_turn)
               sim.setJointTargetVelocity(lrmotor,velLeft - vel_turn)
               sim.setJointTargetVelocity(rfmotor,velRight)
               sim.setJointTargetVelocity(rrmotor,velRight)
           end

       end

end
function sysCall_sensing()
    -- Read Proximity sensor (0= nothing detected, 1 = object detected)

    result, data_Left = sim.readVisionSensor(leftSensor)
    result1, data_Right = sim.readVisionSensor(rightSensor)
    result2, data_Middle = sim.readVisionSensor(middleSensor)

    print ('LeftAvg: intensity: ' ..data_Left[11].. ' r: ' .. data_Left [12].. ' g: ' ..data_Left [13].. ' b: ' ..data_Left [14].. ' Depth: ' ..data_Left [15])
    print ('RightAvg: intensity: ' ..data_Right[11].. ' r: ' .. data_Right [12].. ' g: ' ..data_Right [13].. ' b: ' ..data_Right [14].. ' Depth: ' ..data_Right [15])
    print ('MiddleAvg: intensity: ' ..data_Middle[11].. ' r: ' .. data_Middle [12].. ' g: ' ..data_Middle [13].. ' b: ' ..data_Middle [14].. ' Depth: ' ..data_Middle [15])


end
function sysCall_cleanup()
    -- do some clean-up here
end
```

Anexo 2 – Programação do Segundo Caso de Estudo

Programação na Estrutura do Robô

```
function sysCall_init()
    -- do some initialization here
    lfmotor=sim.getObjectHandle("LeftMotorFront")
    rfmotor=sim.getObjectHandle("RightMotorFront")
    lrmotor=sim.getObjectHandle("LeftMotorRear")
    rrmotor=sim.getObjectHandle("RightMotorRear")
    LeftFront=sim.getObjectHandle('LeftFront')
    LeftRear=sim.getObjectHandle('LeftRear')
    RightFront=sim.getObjectHandle('RightFront')
    RightRear=sim.getObjectHandle('RightRear')
    Forward=sim.getObjectHandle('Forward')

    leftSensor = sim.getObjectHandle('LeftSensor')
    rightSensor = sim.getObjectHandle('RightSensor')
    middleSensor = sim.getObjectHandle('MiddleSensor')
    print(leftSensor)

    --Velocity of the robot
    velStraight = 0.8
    velLeft = 1.2
    velRight = 1.2
    velSharp = 0.965
    vel_turn = 1.4

    --Distance From wall
    avg_default = 0.15
    fwd_default = 100
    distLeftFront = avg_default
    distRightFront = avg_default
    distLeftRear = avg_default
    distRightRear = avg_default
    fwd = fwd_default
    diffLeft = 0

    -- Grape Parameters
    --insertCoordinate = {-1.75,1.175,0.575}
    --goodPercentage = 0.1
    --goodColor = {0.71,0.0,0.66}
    --badColor = {0.25,0.55,0.18}

    -- Initialize auxiliary variables
    T_last_inserted = 0
    deltaTime = 0
    hasStopped = false
    boxList = {}
    boxDummyList = {}
    boolList = {}

```

```
    -- Initialize handles, set Vtech Velocity

    proximityR = sim.getObjectHandle('Grape_Detect_R')
    ColorDetectR = sim.getObjectHandle('Grape_Color_R')
    proximityL = sim.getObjectHandle('Grape_Detect_L')
    ColorDetectL = sim.getObjectHandle('Grape_Color_L')
    OS_default = 0.15
    objectSensor=OS_default

    jointsHandles={-1,-1,-1,-1,-1,-1}
    for i=1,5,1 do
        jointsHandles[i]=sim.getObjectHandle('Cacho'..i)
    end
end
function sysCall_actuation()


    -- Call to set the left and right motor velocities

    --INITIAL VELOCITY OF ROBOT
    sim.setJointTargetVelocity(lfmotor,velStraight)
    sim.setJointTargetVelocity(rfmotor,velStraight)
    sim.setJointTargetVelocity(lrmotor,velStraight)
    sim.setJointTargetVelocity(rrmotor,velStraight)

    --ACTIVATE PROXIMITY SENSOR
    result,distanceR,detectedPoint,detectedObjectHandle,detectedSurfaceNormalVector =sim.readProximitySensor(proximityR)
    result,distanceL,detectedPoint,detectedObjectHandle,detectedSurfaceNormalVector =sim.readProximitySensor(proximityL)


    --print(distance)

    --ACTIVATE VISION SENSOR
    result1,dataVision = sim.readVisionSensor(ColorDetectR)
    result2,dataVision1 = sim.readVisionSensor(ColorDetectL)
```

```
    --DETECT GRAPES Right Side
    if (distanceR ~= nil and distanceL == nil) then

    --RGB INTENSITY
    print ("RED = " .. dataVision[7], "GREEN = " .. dataVision[8], "BLUE = " .. dataVision[9])

        --IDENTITY IF RED OR GREEN GRAPE
        if(( 0.5 <= dataVision[7] and dataVision[7] < 1) and (0 <= dataVision[8] and dataVision[8] < 1.0) and
            (0.5 <= dataVision[9] and dataVision[9] < 1) and (MovimentoBracos == false))
            or dataVision[7] == dataVision[8] and dataVision[8] == dataVision[9])then

            sim.setJointTargetVelocity(lfmotor,0)
            sim.setJointTargetVelocity(rfmotor,0)
            sim.setJointTargetVelocity(lrmotor,0)
            sim.setJointTargetVelocity(rrmotor,0)
            print("RED")
            print("STOP")
            Parado = true

            local otherObjectHandle=sim.getObjectHandle('NiryoOne#0')
            local otherScriptHandle=sim.getScriptAssociatedWithObject(otherObjectHandle)

            MovimentoBracos = true

        else

            sim.setJointTargetVelocity(lfmotor,velStraight)
            sim.setJointTargetVelocity(rfmotor,velStraight)
            sim.setJointTargetVelocity(lrmotor,velStraight)
            sim.setJointTargetVelocity(rrmotor,velStraight)
            print("GREEN")
            print("KEEP GOING")

        end
```

```
    --DETECT GRAPES Left Side
    if (distanceL ~= nil and distanceR == nil) then

    --RGB INTENSITY
    print ("RED = " .. dataVision_1[7], "GREEN = " .. dataVision_1[8], "BLUE = " .. dataVision_1[9])

        --IDENTITY IF RED OR GREEN GRAPE
        if(( 0.50 <= dataVision_1[7] and dataVision_1[7] < 1.0) and (0 <= dataVision_1[8] and dataVision_1[8] < 1.0) and
            (0.4 <= dataVision_1[9] and dataVision_1[9] < 1.0) and (MovimentoBracos == false))
            or dataVision_1[7] == dataVision_1[8] and dataVision_1[8] == dataVision_1[9])then

            sim.setJointTargetVelocity(lfmotor,0)
            sim.setJointTargetVelocity(rfmotor,0)
            sim.setJointTargetVelocity(lrmotor,0)
            sim.setJointTargetVelocity(rrmotor,0)
            print("RED")
            print("STOP")
            Parado = true

            local otherObjectHandle=sim.getObjectHandle('NiryoOne')
            local otherScriptHandle=sim.getScriptAssociatedWithObject(otherObjectHandle)

            MovimentoBracos = true

        else

            sim.setJointTargetVelocity(lfmotor,velStraight)
            sim.setJointTargetVelocity(rfmotor,velStraight)
            sim.setJointTargetVelocity(lrmotor,velStraight)
            sim.setJointTargetVelocity(rrmotor,velStraight)
            print("GREEN")
            print("KEEP GOING")

        end
```

```
        --DETECT IF ITS IN THE MIDDLE
         if (distLeftFront ~= nil and distRightFront ~= nil and Parado == false) then

            if (distLeftFront > distRightFront) then
                print ('Going away from the wall, turn left')
                sim.setJointTargetVelocity(lmotor,velLeft - vel_turn)
                sim.setJointTargetVelocity(rmotor,velRight)

            elseif (distLeftFront < distRightFront) then
                print ('Going toward the wall, turn right')
                sim.setJointTargetVelocity(lmotor,velLeft)
                sim.setJointTargetVelocity(rmotor,velRight - vel_turn)

            end
         end

    end
    end

end
```

```
function sysCall_sensing()

    -- Read Proximity sensor (0= nothing detected, 1 = object detected)
    flag6,objectSensor=sim.readProximitySensor(proximityR)
    flag6,objectSensor=sim.readProximitySensor(proximityL)

    --print('flag6 =' .. flag6 .. ' ' .. 'OS =' .. objectSensor)-

    if (flag6 == 1 ) then
        Object_sensor = objectSensor
    else
        Object_sensor = OS_default
    end
    --print('Grape Distance = ' .. Object_sensor)

    -- put your sensing code here
    flag1,distLeftFront=sim.readProximitySensor(LeftFront)
    flag2,distLeftRear=sim.readProximitySensor(LeftRear)
    flag3,distRightFront=sim.readProximitySensor(RightFront)
    flag4,distRightRear=sim.readProximitySensor(RightRear)
    flag5,distForward = sim.readProximitySensor(Forward)

    --print distance between sensors
    --print('flag1 =' .. flag1 .. ' ' .. 'LF =' .. distLeftFront)
    --print('flag2 =' .. flag2 .. ' ' .. 'LR =' .. distLeftRear)
    --print('flag3 =' .. flag3 .. ' ' .. 'RF =' .. distRightFront)
    --print('flag4 =' .. flag4 .. ' ' .. 'RR =' .. distRightRear)
    --print('flag5 =' .. flag5 .. ' ' .. 'F =' .. distForward)

    --print average distance and the difference LeftSide
    if (flag1 == 0 and flag2 == 1) then
        avgLeft = distLeftRear
        diffLeft = 0
    elseif (flag1 == 1 and flag2 == 0) then
        avgLeft = distLeftFront
        diffLeft = 0
    elseif (flag1 == 1 and flag2 == 1) then
        avgLeft = 0.5*(distLeftFront + distLeftRear)
        diffLeft = distLeftFront - distLeftRear
    else
        avgLeft = avg_default
        diffLeft = 0
    end
    --print('avg_Left = ' .. avgLeft .. ' ' .. 'diff_Left = ' .. diffLeft)
```

```
    --print average distance and the difference RightSide
    if (flag3 == 0 and flag4 == 1) then
        avgRight = distRightRear
        diffRight = 0
    elseif (flag3 == 1 and flag4 == 0) then
        avgRight = distRightFront
        diffRight = 0
    elseif (flag3 == 1 and flag4 == 1) then
        avgRight = 0.5*(distRightFront + distRightRear)
        diffRight = distRightRear - distRightFront
    else
        avgRight = avg_default
        diffRight = 0
    end
    --print('avg_Right = ' .. avgRight .. ' ' .. 'diff_Right = ' .. diffRight)

    if (flag5 == 1 ) then
        fwd = distForward
        else
        fwd = fwd_default
    end
    --print('avg_Foward = ' .. fwd)

end
function sysCall_cleanup()
    -- do some clean-up here
end
```

Programação nos braços robóticos

```
function sysCall_threadmain()
    jointHandles={-1,-1,-1,-1,-1,-1}
    for i=1,6,1 do
        jointHandles[i]=sim.getObjectHandle('NiryoOneJoint'..i)
    end

    connection=sim.getObjectHandle('NiryoOne_connection')
    gripper=sim.getObjectChild(connection,0)
    gripperName="NiryoNoGripper"
    if gripper~=-1 then
        gripperName=sim.getObjectName(gripper)
    end

    -- Set-up some of the RML vectors:
    vel=20
    accel=40
    jerk=80
    currentVel={0,0,0,0,0,0,0}
    currentAccel={0,0,0,0,0,0,0}
    maxVel={vel*math.pi/180,vel*math.pi/180,vel*math.pi/180,vel*math.pi/180,vel*math.pi/180,vel*math.pi/180}
    maxAccel={accel*math.pi/180,accel*math.pi/180,accel*math.pi/180,accel*math.pi/180,accel*math.pi/180,accel*math.pi/180}
    maxJerk={jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180}
    targetVel={0,0,0,0,0,0}

    sim.wait(30)
    targetPos1={0*math.pi/180,90*math.pi/180,0*math.pi/180,0*math.pi/180,0*math.pi/180,0*math.pi/180}
    sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,targetPos1,targetVel)
    sim.wait(1)

```

```
    targetPos4={90*math.pi/180,90*math.pi/180,10*math.pi/180,0*math.pi/180,-45*math.pi/180,0*math.pi/180}
    sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,targetPos4,targetVel)
    sim.setIntegerSignal(gripperName..'_close',1)
    sim.wait(4)

    targetPos3={0,0,0,0,0,0}
    sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,targetPos3,targetVel)

    targetPos2={0*math.pi/180,-54*math.pi/180,0*math.pi/180,0*math.pi/180,-36*math.pi/180,-90*math.pi/180}
    sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,targetPos2,targetVel)
    sim.clearIntegerSignal(gripperName..'_close')
    sim.wait(2)

    targetVel={0,0,0,0,0,0}

    sim.wait(0)
    targetPos1={0*math.pi/180,90*math.pi/180,0*math.pi/180,0*math.pi/180,0*math.pi/180,0*math.pi/180}
    sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,targetPos1,targetVel)
    sim.wait(1)

```

```
    targetPos4={90*math.pi/180,90*math.pi/180,10*math.pi/180,0*math.pi/180,-45*math.pi/180,0*math.pi/180}
    sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,targetPos4,targetVel)
    sim.setIntegerSignal(gripperName..'_close',1)
    sim.wait(4)

    targetPos3={0,0,0,0,0,0}
    sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,targetPos3,targetVel)

    targetPos2={0*math.pi/180,-54*math.pi/180,0*math.pi/180,0*math.pi/180,-36*math.pi/180,-90*math.pi/180}
    sim.rmlMoveToJointPositions(jointHandles,-1,currentVel,currentAccel,maxVel,maxAccel,maxJerk,targetPos2,targetVel)
    sim.clearIntegerSignal(gripperName..'_close')
    sim.wait(2)


end
function sysCall_cleanup()
    -- Put some clean-up code here
end
```

Anexo 3 - Programação do Terceiro Caso de Estudo

```
getLightSensors=function()
    data=sim.receiveData(0,'lightSensor')
    if (data) then
        lightSens=sim.unpackFloatTable(data)
    end
    return lightSens
end
function sysCall_threadmain()
    -- Initialization:
    sim.setThreadSwitchTiming(200) -- We will manually switch in the main loop

    bodyElements=sim.getObjectHandle("Vtech_respondable")
    lfmotor=sim.getObjectHandle("LeftMotorFront")
    rfmotor=sim.getObjectHandle("RightMotorFront")
    lrmotor=sim.getObjectHandle("LeftMotorRear")
    rrmotor=sim.getObjectHandle("RightMotorRear")
    proxSens={-1,-1,-1,-1,-1,-1}
    for i=1,6,1 do
        proxSens[i]=sim.getObjectHandle("proxSensor"..i)
    end

    maxVel = 1.1
    vel_turn = 0.8
    ledColors={{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0},{0,0,0}}

    -- Braitenberg weights for the 4 front prox sensors (avoidance):
    braitFrontSens_leftMotor={-1,-2,2,1}
    -- Braitenberg weights for the 2 side prox sensors (following):
    braitSideSens_leftMotor={-3,0}
    -- Braitenberg weights for the 4 sensors (following):
    braitAllSensFollow_leftMotor={-3,-1.5,-0.5,0.8}
    braitAllSensFollow_rightMotor={0,1,0.8,-0.5,-1.5}
    braitAllSensAvoid_leftMotor={0,0.5,1,-1}
    braitAllSensAvoid_rightMotor={-0.5,-0.5,-1,1}

    while sim.getSimulationState()~=sim.simulation_advancing_abouttostop do
        st=sim.getSimulationTime()
        velLeft=0
        velRight=0
        opMode=0--sim.getScriptSimulationParameter(sim.handle_self,'opMode')
        lightSens=getLightSensors()
        s=sim.getObjectSizeFactor(bodyElements) -- make sure that if we scale the robot during simulation, other values are scaled too!
        noDetectionDistance=0.6*s
        proxSensDist={noDetectionDistance,noDetectionDistance,noDetectionDistance,noDetectionDistance,noDetectionDistance,noDetectionDistance}
```

```

        for i=1,6,1 do
            res,dist=sim.readProximitySensor(proxSens[i])

            proxSensDist[i]=0
            if (dist and dist<noDetectionDistance) then
                proxSensDist[i]=dist
            end

        end

        if (opMode==0) then -- We wanna follow the line
            if (math.mod(st,2)>1.5) then
                intensity=1
            else
                intensity=0
            end
            for i=1,9,1 do
                ledColors[i]={intensity,0,0} -- red
            end
            velRight=maxVel
            velLeft=maxVel

            if (proxSensDist[1]+proxSensDist[2]+proxSensDist[3]+proxSensDist[4] / 4 ~= 0) then
                -- Obstacle in front. Use Braitenberg to avoid it
                    for i=1,4,1 do
                        velLeft=velLeft+maxVel*braitFrontSens_leftMotor[i]*(1-(proxSensDist[i]/noDetectionDistance))
                        velRight=velRight+maxVel*braitFrontSens_leftMotor[5-i]*(1-(proxSensDist[i]/noDetectionDistance))
                    end
            else

                    print ("proxSensDists ", proxSensDist)
                if (proxSensDist[5]+proxSensDist[6] / 2~=0) then

                    print ("noDetectionDistance x 4 ", noDetectionDistance*4)
                    -- Nothing in front. Maybe we have an obstacle on the side, in which case we wanna keep a constant distance with it:
                    if (proxSensDist[6]<0.1*noDetectionDistance) then
                        velLeft=velLeft+maxVel*braitSideSens_leftMotor[1]*(1-(proxSensDist[6]/noDetectionDistance))
                        velRight=velRight+maxVel*braitSideSens_leftMotor[2]*(1-(proxSensDist[6]/noDetectionDistance))
                        print ("Sensor Esquerda")
                    end
                    if (proxSensDist[5]<0.1*noDetectionDistance) then
                        velLeft=velLeft+maxVel*braitSideSens_leftMotor[2]*(1-(proxSensDist[5]/noDetectionDistance))
                        velRight=velRight+maxVel*braitSideSens_leftMotor[1]*(1-(proxSensDist[5]/noDetectionDistance))
                        print ("Sensor Direita")
                    end
                else
```

```
                -- Now make sure the light sensors have been read, we have a line and the 4 front prox. sensors didn't detect anything:
                    if lightSens and((lightSens[1]<0.5)or(lightSens[2]<0.5)or(lightSens[3]<0.5)) then

                        if (lightSens[1]<=0.5) then
                            velLeft=maxVel
                        else
                            velLeft = maxVel - vel_turn
                            velRight = maxVel

                        end
                        if (lightSens[3]<=0.5) then
                            velRight=maxVel
                        else

                            velRight = maxVel - vel_turn
                            velLeft = maxVel

                        end
                    end
                end
            end
        end
        if (opMode==1) then -- We wanna follow something!
            index=math.floor(1+math.mod(st*0.9))
            for i=1,9,1 do
                if (index==i) then
                    ledColors[i]={0,0.5,1} -- light blue
                else
                    ledColors[i]={0,0,0} -- off
                end
            end
            velRightFollow=maxVel
            velLeftFollow=maxVel
            minDist=1000
            for i=1,6,1 do
                velLeftFollow=velLeftFollow+maxVel*braitAllSensFollow_leftMotor[i]*(1-(proxSensDist[i]/noDetectionDistance))
                velRightFollow=velRightFollow+maxVel*braitAllSensFollow_rightMotor[i]*(1-(proxSensDist[i]/noDetectionDistance))
                if (proxSensDist[i]<minDist) then
                    minDist=proxSensDist[i]
                end
            end
```

```
              velRightAvoid=0
              velLeftAvoid=0
              for i=1,6,1 do
                  velLeftAvoid=velLeftAvoid+maxVel*braitAllSensAvoid_leftMotor[i]*(1-(proxSensDist[i]/noDetectionDistance))
                  velRightAvoid=velRightAvoid+maxVel*braitAllSensAvoid_rightMotor[i]*(1-(proxSensDist[i]/noDetectionDistance))
              end
              if (minDist>0.025*s) then minDist=0.025*s end
              t=minDist/(0.025*s)
              velLeft=velLeftFollow*t+velLeftAvoid*(1-t)
              velRight=velRightFollow*t+velRightAvoid*(1-t)
          end
          sim.setJointTargetVelocity(lfmotor,velLeft)
          sim.setJointTargetVelocity(rfmotor,velRight)
          sim.setJointTargetVelocity(lrmotor,velLeft)
          sim.setJointTargetVelocity(rrmotor,velRight)

          sim.switchThread()
      end
end

function sysCall_cleanup()
    -- Clean-up:

end
```