UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

William Zaniboni Silva

Desenvolvimento e análise de ambiente de simulação para robôs agrícolas

São Carlos

2021

William Zaniboni Silva

Desenvolvimento e análise de ambiente de simulação para robôs agrícolas

Monografia apresentada ao Curso de Engenharia Mecatrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Mecatrônico.

Orientador: Prof. Dr. Marcelo Becker

São Carlos

2021

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Silva, William Zaniboni
S586d Desenvolvimento e análise de ambiente de simulação para robôs agrícolas / William Zaniboni Silva; orientador Marcelo Becker. São Carlos, 2021.

Monografia (Graduação em Engenharia Mecatrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2021.

1. Simulação. 2. Robótica. 3. Agricultura. 4. TerraSentia. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE AVALIAÇÃO

Candidato: William Zaniboni Silva

Título: Desenvolvimento e análise de ambiente de simulação para robôs agrícolas

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos da Universidade de São Paulo Curso de Engenharia Mecatrônica.

BANCA EXAMINADORA

Professor Dr. Marcelo Becker - EESC - USP
(Orientador)

Nota atribuída: 10,0 (Dez)

(assinatura)

D.Sc. Andres Eduardo Baquero Velasquez - UIUC - USA

Nota atribuída: 10,0 (Dez)

(assinatura)

M.Sc. Mateus Valverde Gasparino - UIUC - USA

Nota atribuída: 10,0 (Dez)

(assinatura)

Média: 10,0 (Dez)

Resultado: APROVADO

Data: 16/12/2021.

Este trabalho tem condições de ser hospedado no Portal Digital da Biblioteca da EESC

SIM ☒ NÃO ☐ Visto do orientador

RESUMO

SILVA, W. **Desenvolvimento e análise de ambiente de simulação para robôs agrícolas**. 2021. 75p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

Este trabalho tem como objetivo primordial complementar um ambiente de simulação utilizado em aplicações robóticas na área da agricultura. O trabalho utilizado como referência principal para esta monografia viabiliza a testagem e desenvolvimento de plataformas robóticas utilizadas em campos agrícolas no software de simulação Gazebo por meio de um vasto repositório com modelos de plantas, robôs, sensores e scripts que permitem representar condições reais dos campos rurais. Esta referência considera algumas simplificações no escopo da geração das plantações (por exemplo, as plantas são consideradas estáticas, as plantações são criadas apenas para terrenos planos e as linhas de plantio são sempre linhas retas) e a partir destas, são propostas e implementadas abordagens complementares nesta monografia: adição de uma interface gráfica que permita customizar ainda mais os ambientes agrícolas e adição de animação nas plantações criadas a fim de representar interações com o vento. Uma das aplicações consideradas para o ambiente de simulação consiste em testar um algoritmo de navegação que possibilita a locomoção de forma autônoma de um robô entre linhas de plantação. Este é usado em robôs reais e foi integrado ao ambiente de simulação. Foram feitos testes para verificar o funcionamento do algoritmo e analisar o desempenho da simulação em diversos ambientes criados por meio da interface gráfica. Os resultados considerando a plantação estática foram comparados com os resultados do mapa com vento atuando. Concluiu-se que o algoritmo de navegação é impactado pelos novos parâmetros adicionados à simulação e que esta, torna-se um pouco mais complexa e dessa forma, passa a requerer mais poder computacional. Constata-se que algumas melhorias podem ser feitas em trabalhos futuros, como por exemplo, utilizar *softwares* de animação 3D para criar movimentações mais realistas das plantas.

Palavras-chave: Simulação. Robótica. Agricultura. TerraSentia.

ABSTRACT

SILVA, W. **Development and analysis of simulation environment for agricultural robots**. 2021. 75p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

The main objective of this work is to complement a simulation environment used in robotic applications in the agriculture field. The work used as the main reference for this monograph enables the testing and development of robotic platforms used in agricultural fields in the Gazebo simulator through a vast repository with plant models, robots, sensors and scripts that allow representing real conditions of rural fields. This reference considers some simplifications in the scope of plantation generation (for example, the plants are considered static, plantations are created only for flat grounds and planting lines are always straight lines) and from these, complementary approaches are proposed and implemented in this monograph: addition of a graphical interface that allows further customization of agricultural environments and addition of animation in the plantations created in order to represent interactions with wind. One of the applications considered for the simulation environment is to test a navigation algorithm that allows the autonomous navigation of a robot between plantation rows. This is used in real robots and has been integrated into the simulation environment. Tests was made to verify the functioning of the algorithm and analyze the performance of the simulation in different environments created through the graphical interface. The results considering the static plantation were compared with the results of the plantation with wind acting. It was concluded that the navigation algorithm is impacted by the new parameters added to the simulation and that the simulation becomes a little more complex and, therefore, requires more computational power. Some improvements can be made in future work, such as using 3D animation software to create more realistic plant movements.

Keywords: Simulation. Robotics. Agriculture. TerraSentia.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ROS	Robot Operating System
ARDEE	A general agricultural robotic development and evaluation environment
HIL	Hardware-in-the-Loop
SIL	Software-in-the-Loop
UDP	User Datagram Protocol
GMDB	Gazebo Model Database
URDF	Unified Robot Description Format
GPS	Global Positioning System
IMU	Inertial Measurement Unit
RGB	Red Green Blue
RGBD	Red Green Blue Depth
Lidar	Light Detection and Ranging

SUMÁRIO

1 INTRODUÇÃO

Com o desenvolvimento tecnológico, o setor agrícola vem recebendo mecanismos que permitem aumentar o rendimento da produção de diversas maneiras. Entre esses mecanismos, estão aplicações robóticas que permitem automatizar tarefas lentas, repetitivas ou perigosas, como por exemplo, colheita, poda, fenotipagem, controle de ervas daninhas e semeadura (Association for Advancing Automation (2017); Interesting Engineering (2021)).

Desenvolver uma plataforma robótica é algo complexo e atualmente, os ambientes de simulação estão sendo cada vez mais utilizados para este fim. Com os simuladores é possível compreender como os robôs devem ser projetados e controlados com o intuito de ter uma operação segura e um desempenho aprimorado, com custos menores e com maior rapidez (CHOI *et al.*, 2021).

Por meios desses *softwares* de simulação, muitas pesquisas na área da robótica se tornaram viáveis, já que com eles é factível simular a física, modelar virtualmente o robô, recriar ambientes e validar protótipos complexos (MELO *et al.*, 2019).

1.1 Objetivos

O enfoque geral deste trabalho é o desenvolvimento e análise de um ambiente de simulação para robôs agrícolas, para isto, busca-se objetivar os itens apresentados a seguir:

- Apresentar estudos sobre simulações no mundo da robótica, pontuando vantagens, desvantagens, contexto atual, *softwares* utilizados, áreas de aplicação e métricas de desempenho dos simuladores.
- Apresentar e complementar um trabalho que permite criar ambientes de simulação para robôs agrícolas. As duas complementações consideradas são: adicionar uma interface gráfica para facilitar a customização e adicionar animação aos modelos de plantas a fim de simular alguma interação com vento.
- Integrar e testar com ambiente de simulação, um algoritmo utilizado para a navegação autônoma de robôs em plantações. Não será considerado no escopo do trabalho realizar um aprofundamento em seu funcionamento.

1.2 Organização do trabalho

Este trabalho está organizado da seguinte maneira:

- **Capítulo 2 - Revisão bibliográfica:** Apresenta uma revisão da literatura acerca do contexto atual das simulações no mundo da robótica. Os simuladores mais

utilizados são apresentados e comparados com base em estudos feitos. É exposto e descrito o trabalho utilizado como referência principal para esta monografia.

- **Capítulo 3 - Desenvolvimento:** Apresenta as motivações para o que será elaborado e descreve todo o processo de desenvolvimento feito pelo autor. Os resultados obtidos são exibidos e discutidos.
- **Capítulo 4 - Conclusão:** Exibe as conclusões acerca do trabalho realizado e pontua sugestões para trabalhos futuros com base no que foi considerado como escopo do projeto.

2 REVISÃO BIBLIOGRÁFICA

O objetivo deste capítulo é apresentar uma revisão da literatura de áreas relacionadas à simulação na robótica. Dessa forma, o capítulo será estruturado da seguinte maneira: a seção 2.1 apresentará as tendências, dificuldades e contexto atual do papel das simulações no desenvolvimento robótico; a seção 2.2 identificará alguns dos *softwares* de simulação, pontuando aspectos como vantagens e desvantagens de cada um; a seção 2.3 exibirá um trabalho de desenvolvimento de ambiente de simulação agrícola, sendo este a referência principal para o desenvolvimento desta monografia; a seção 2.4 apontará ferramentas de um *software* de simulação específico, com uma análise de métricas de desempenho sendo feita na seção 2.5; e por fim, na seção 2.6 será apresentado um algoritmo de navegação que será posteriormente citado no capítulo 3.

2.1 Uma visão geral sobre robôs agrícolas e simulações na robótica

Em Bechar e Vigneault (2016) são apresentados conceitos relacionados a robôs agrícolas com operações de campo. Na maioria das operações agrícolas, os ambientes são caracterizados por mudanças rápidas de tempo e espaço, já que o terreno, a vegetação e iluminação, por exemplo, variam continuamente. Além disso, ao considerar objetos naturais como frutas e folhas, a complexidade da operação é aumentada por adicionar variáveis que não podem ser determinadas a priori, como por exemplo: orientação, posição e forma desses itens. Em relação a implementação de sistemas robóticos na agricultura, a maioria dos projetos não chega até essa fase devido a motivos como: custo excessivo para desenvolvimento e incapacidade do robô de realizar a mesma tarefa em contextos ligeiramente diferentes (BECHAR; VIGNEAULT, 2016).

O Quadro 1 apresenta uma lista de condições que, ao menos uma, segundo os autores, deve ser atendida para a implementação de uma tecnologia robótica.

Quadro 1 – Condições para a implementação de uma tecnologia robótica na agricultura

Condições (ao menos uma deve ser atendida)
O custo de utilização de robôs é menor do que o custo de qualquer método simultâneo.
O uso de robôs permite aumentar a capacidade de produção, lucro e sobrevivência da fazenda em condições competitivas de mercado.
O uso de robôs melhora a qualidade e uniformidade da produção.
O uso de robôs minimiza a incerteza e a variação nos processos de crescimento e produção.
O uso de robôs permite ao agricultor tomar decisões e atuar em maior resolução e/ou aumentar a qualidade do produto em comparação ao sistema concorrente para otimizar as etapas de cultivo e produção.
O robô é capaz de realizar tarefas específicas que são definidas como perigosas ou que não podem ser realizadas manualmente.

Fonte: Adaptado de Bechar e Vigneault (2016)

Mesmo com muitas tecnologias não chegando na etapa de implementação e comercialização, o número de pesquisas para o desenvolvimento de tecnologias robóticas vem aumentando drasticamente nos últimos anos. A Tabela 1 apresenta as áreas que envolvem robôs aplicados à agricultura com mais pesquisas nas últimas décadas. Nota-se uma elevada quantidade de trabalhos relacionados a orientação e navegação.

Tabela 1 – Esforço de pesquisa em robôs agrícolas nas últimas três décadas

Categoria	Tipo (número de artigos publicados)
Cortar	Citrus (193), maçã (363), tomate (176), pimenta (73), pepino (66), morango (51)
Tarefa agrícola	Transplante e semeadura (255), proteção das plantas e controle de ervas daninhas (326), colheita (302)
Tarefas de apoio	Orientação e navegação (446), mapeamento e localização (261), agarrar frutas e vegetais (125), interação multi-robô (60)

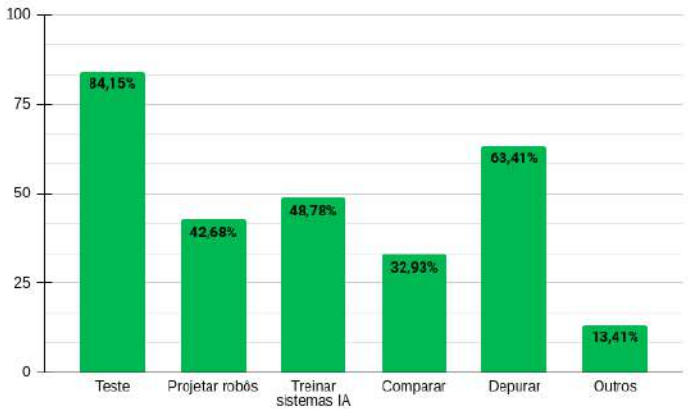
Fonte: Adaptado de Bechar e Vigneault (2016).

Nota: Uma mesmo artigo pode estar em mais de uma categoria.

Conforme apresentado em Iqbal *et al.* (2020), devido aos elevados custos de tempo e de dinheiro necessários para o desenvolvimento de plataformas robóticas, a simulação está sendo cada vez mais utilizada para testes e validação. Em relação a aplicação da simulação em robôs agrícolas, ela permite recriar os ambientes agrícolas, testar a eficiência e eficácia de diversos algoritmos e desenvolver métodos de controle (IQBAL *et al.*, 2020).

Considerando todas as áreas da robótica, não especificamente aplicação na agricultura, uma pesquisa com desenvolvedores foi realizada em Afzal *et al.* (2021) para abordar a utilização atual, as capacidades e os limites da simulação. A Figura 1 exibe as justificativas apresentadas para o uso de simulação pelas 82 pessoas entrevistadas.

Figura 1 – Razões para uso das simulações



Fonte: Retirado e adaptado de Afzal *et al.* (2021)

Nota-se que a principal razão para a utilização do ambiente de simulação é para a realização de testes, os quais podem ser categorizados como *hardware-in-the-loop* (HIL) e *software-in-the-loop* (SIL). No primeiro, o *software* controlador do robô é executado no próprio *hardware* do robô, já no segundo, ele é executado na mesma máquina que o simulador. Na mesma pesquisa observaram-se as dificuldades encontradas atualmente ao realizar testes no ambiente de simulação, elas estão apresentadas a seguir:

- **Reprodutibilidade:** Os simuladores não são determinísticos, ou seja, não é possível repetir uma simulação e obter exatamente o mesmo resultado, mesmo com as imprecisões do sistema robótico se comportando de maneira idêntica. Isso atrapalha na reprodução de falhas e consequentemente, nas suas correções.
- **Cenário e construção do ambiente:** É complexo e trabalhoso criar os cenários para as realizações de testes. Na maioria das vezes, os cenários e a própria caracterização do robô devem ser feitos manualmente.
- **Custos de recursos:** As simulações requerem altos recursos computacionais. Dessa forma, os custos de *hardware* são altos, principalmente, ao utilizar servidores na nuvem para a realização dos testes.

As barreiras atuais encontradas na simulação são apresentadas e discutidas em Choi *et al.* (2021). Elas estão presentes no Quadro 2 e algumas delas se relacionam com as dificuldades citadas em Afzal *et al.* (2021).

Quadro 2 – Algumas das barreiras atuais da simulação na robótica

Barreiras
O desenvolvimento de uma plataforma de simulação requer amplo conhecimento multidisciplinar e uma engenharia de software para integração de todos os componentes presentes.
As linguagens de modelagem existentes são imaturas e a ontologia da simulação de robótica está se desenvolvendo lentamente.
É necessário aprimorar a composição/processamento de cenários mais complexos.
As simulações não são rápidas o suficiente.
Não há um bom tratamento para as incertezas.
Caracterizar um modelo é trabalhoso.
Dificuldade em avaliar o nível necessário de complexidade de um modelo.

Fonte: Adaptado de Choi *et al.* (2021)

Um ponto interessante que foi pontuado em Choi *et al.* (2021) ao discorrer sobre as barreiras da simulação, é que muitas plataformas utilizadas para a criação de jogos para computador estão sendo utilizadas para a simulação robótica. Como vantagem, elas permitem a criação de mundos complexos e verossímeis a realidade, mas como desvantagem, elas não permitem precisão da física dos comportamentos robóticos já que não foram projetadas para isso.

A respeito dos pontos positivos dos ambientes de simulação, em Choi *et al.* (2021) também são discutidas e apresentadas oportunidades de como a simulação pode ajudar no desenvolvimento robótico. O Quadro 3 apresenta algumas das oportunidades atuais de como a simulação é ou poderia ser mais proveitosa.

Quadro 3 – Algumas das oportunidades atuais da simulação na robótica

Oportunidades
Gerar, de forma rápida e barata, grandes quantidades de dados de treinamento para aprendizado de máquina.
Acelerar o ciclo de um projeto de engenharia e reduzir os seus custos.
Disponibilizar um ambiente que permita testes seguros, controlados e acelerados.
Facilitar o desenvolvimento de robôs mais inteligentes.
Facilitar a compreensão das interações humano-robô.

Fonte: Adaptado de Choi *et al.* (2021)

Por fim, o Quadro 4 apresenta algumas recomendações para tornar a simulação mais útil na robótica.

Quadro 4 – Algumas recomendações para tornar a simulação mais útil na robótica

Recomendações
Promover o desenvolvimento e validação de plataformas de simulação de código aberto.
Estabelecer bibliotecas abertas e com curadoria de modelos validados pela comunidade.
Enfatizar o uso de arquiteturas de hardware emergentes.

Fonte: Adaptado de Choi *et al.* (2021)

Em suma, os ambientes de simulação necessitam de muito desenvolvimento, sendo que faz sentido focar em pesquisa. Para isso, terão que ser abordados conceitos multidisciplinares presentes em áreas como: engenharia mecânica, engenharia elétrica, ciência da computação, ciência cognitiva e matemática aplicada (CHOI *et al.*, 2021).

2.2 Softwares de simulação

Segundo Melo *et al.* (2019) os *softwares* de simulação mais utilizados no mercado são: Unity, Gazebo e V-Rep. A Tabela 2 exibe a quantidade de menções em trabalhos científicos de cada um deles.

Tabela 2 – *Softwares* de simulação mais utilizados

Palavra buscada	Número de resultados
Gazebo	30900
Unity3D/Unity 3D/Unity Simulator	25800
V-Rep	10400
Webots	4730
ARgOS Simulator	2100
UserSim	1650
OpenRAVE	1210
OpenHRP	877
SimSpark	425
RoboDK	103

Fonte: Retirado e adaptado de Melo *et al.* (2019)

A seguir serão expostas informações de cada um deles. A subseção 2.2.1 apresentará detalhes do *software* V-Rep, a subseção 2.2.2 exibirá especificidades do Unity, a subseção 2.2.3 apresentará o Gazebo e por fim, na subseção 2.2.4 serão reproduzidos alguns trabalhos que comparam os simuladores.

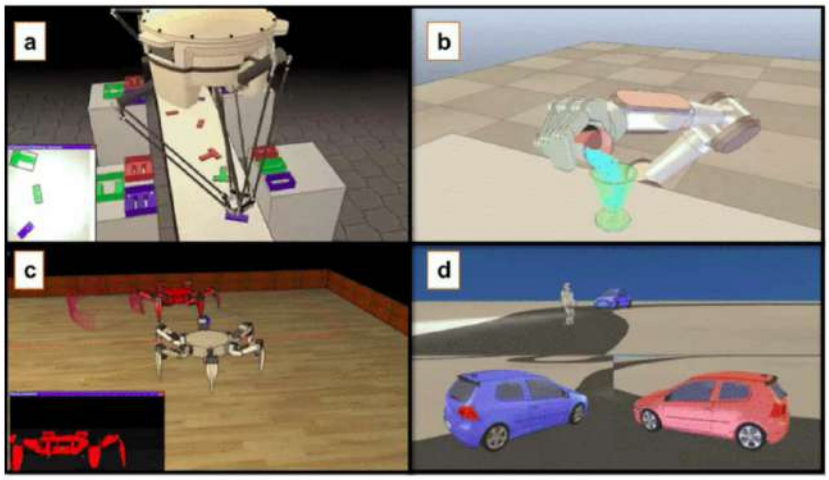
2.2.1 V-Rep (CoppeliaSim)

O simulador robótico V-Rep (*Virtual Robot Experimentation Platform*) foi desenvolvido pela *Coppelia Robots* e atualmente foi descontinuado. O novo *software* da fabricante chama-se CoppeliaSim e segundo os seus fornecedores, trata-se do V-Rep com uma mudança de nome (COPPELIA ROBOTICS, 2021). Para usar a mesma nomenclatura dos trabalhos citados, vamos manter o nome V-Rep nesta monografia.

Ele é baseado em uma arquitetura distribuída que permite que cada modelo seja controlado individualmente. Dessa forma, mostra-se versátil e ideal para aplicações multi-robôs. Pode ser executado em ambientes Linux, Windows e Mac.

Sobre a sua licença, ele é gratuito apenas para fins educacionais. A Figura 2 apresenta alguns exemplos do uso do V-Rep em simulações (A - funcionamento de um *pick and place*, B - manuseio de um objeto, C - detecção de objetos, D - simulação envolvendo carros).

Figura 2 – Algumas simulações no V-Rep



Fonte: Retirado de Melo *et al.* (2019)

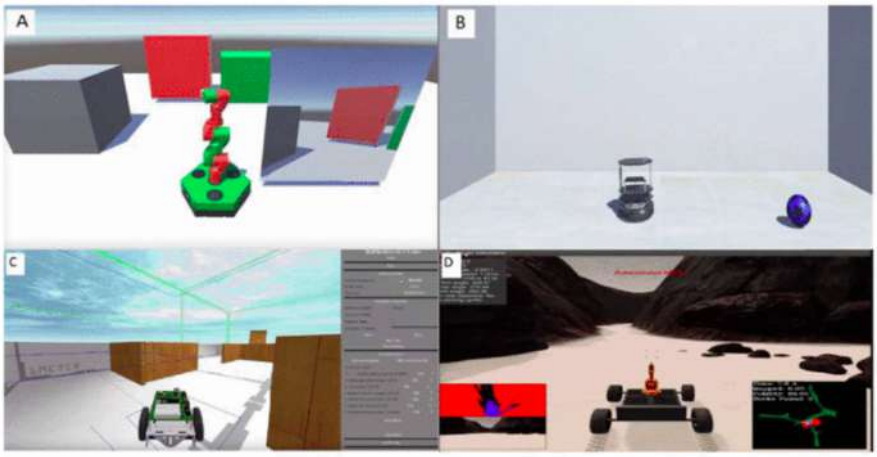
2.2.2 Unity

Unity, desenvolvida pela *Unity Technologies* em 2005, é uma das mais populares plataformas para a criação de jogos 2D e 3D. Atualmente possui recursos de realidade aumentada e realidade virtual.

Apresenta mecanismos gráficos que permitem autenticidade e realismo ao ambiente de simulação. Mostra-se um editor rico em recursos, flexível (pode ser utilizado em sistemas operacionais Windows, Linux e Mac) e que contêm uma comunidade ampla de desenvolvedores, o que permite que a prototipagem das simulações seja facilitada ao importar componentes já existentes e compartilhados por outros (HUSSEIN; GARCíA; OLAVERRI-MONREAL, 2018).

Sobre a sua licença, apresenta diversos planos, sendo que é gratuito para estudantes. A Figura 3 apresenta exemplos de simulações no Unity (A - controle das posições de juntas, B - treinamento de um robô para encontrar a bola azul, C - teste de algoritmos de navegação e D - mapeamento de terreno navegável).

Figura 3 – Algumas simulações no Unity

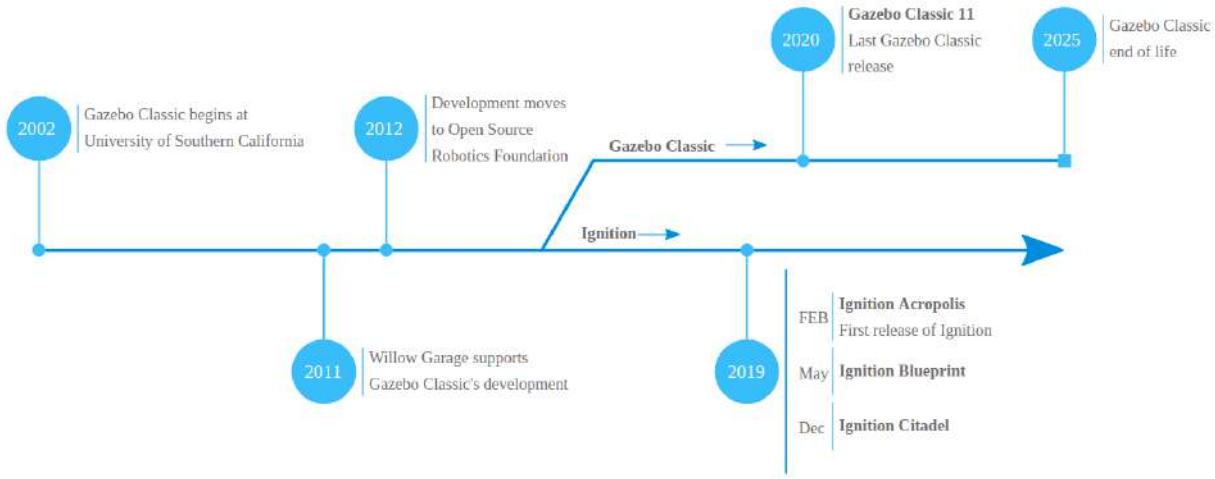


Fonte: Retirado de Melo *et al.* (2019)

2.2.3 Gazebo (Gazebo Classic)

O Gazebo (ou Gazebo Classic) teve o seu desenvolvimento iniciado em 2002 e desde 2012 é desenvolvido pela Open Source Robotics Foundation (OSRF). Após anos de desenvolvimento, ele está sendo refatorado e migrado para uma nova plataforma de simulação chamada Ignition e será descontinuado em 2025 (OPEN ROBOTICS, 2021). A Figura 4 apresenta a linha do tempo de seu desenvolvimento.

Figura 4 – Linha do tempo do desenvolvimento do Gazebo



Fonte: Retirado de Open Robotics (2021)

O Gazebo Classic (vamos chamá-lo apenas de Gazebo) apresenta uma vasta biblioteca de elementos de cena e é utilizado em campeonatos de robótica para simular os ambientes da competição. Pode ser usado em ambientes Linux (foco principal), Windows e Mac.

Sobre a sua licença, ele é *open-source* e licenciado sob Apache 2.0. A Figura 5 apresenta exemplos de simulações utilizando o Gazebo (A - manuseio de objetos, B - controle e simulação de vôo, C - simulação de robô humanoide e D - planejamento de caminho).

Figura 5 – Algumas simulações no Gazebo

Fonte: Retirado de Melo *et al.* (2019)

2.2.4 Comparações entre os simuladores

Em Melo *et al.* (2019) também é apresentada uma análise comparativa dos *softwares* de simulação mais utilizados no mercado e citados anteriormente. Em relação às conclusões obtidas a respeitos dos simuladores, para o Unity, de forma análoga ao pontuado por Choi *et al.* (2021) e presente na seção 2.1, por tratar-se de um *software* para a criação de jogos, ele otimiza os termos gráficos, mas deixa em segundo plano os termos físicos. Em relação ao Gazebo, por ser um *software* de código aberto, apresenta facilidades (menos burocracia) em converter o projeto em solução de mercado. Já o V-Rep apresenta interfaces mais amigáveis e intuitivas, tendo assim mais recursos nativos, o que pode levar na economia de tempo para desenvolvimento, mas apresenta licença gratuita apenas para fins educacionais.

Em Nogueira (2014) é tratada uma comparação entre Gazebo e V-Rep. O primeiro mostrou-se melhor na integração com ROS, já o segundo apresentou facilidade na modelagem de mundo, mostrando-se assim, mais intuitivo e fácil de usar.

Uma pesquisa presente em Ivaldi, Padois e Nori (2014) coletou *feedback* de 119

usuários acerca de ferramentas de simulação. Cerca de 13% dos entrevistados utilizavam o Gazebo como plataforma principal e cerca de 6% utilizavam o V-Rep (o Unity não foi citado). A conclusão a respeito da pesquisa feita é que o Gazebo mostra-se mais adequado para a comunidade de pesquisa e o V-Rep mais aplicável a nível industrial.

A Tabela 3 apresenta os *softwares* de simulação mais utilizados de acordo com áreas de pesquisa específicas. Veja que o Gazebo se destaca, sendo mais utilizado nas áreas de robôs móveis, robôs humanoides e robôs de serviço.

Tabela 3 – *Softwares* de simulação em áreas de pesquisa

Área de pesquisa	*Softwares* mais utilizados (menções)
Robôs humanoides	(4) ODE, (3) **Gazebo**, Robotran, OpenRave, Arboris-Python, (2) XDE, iCub-SIM
Robótica móvel	(5) **Gazebo**, ARGoS, (3) Webots, (2) V-Rep, Vortex
Robôs com pernas	(3) Webots, (2) ODE
Robôs de serviço	(4) **Gazebo**, (3) OpenRave
Drones	(2) ARGoS
Robótica de enxame	(4) ARGoS
Manipuladores industriais	(1) Bullets, Dymola, Matlab, V-Rep, XDE
Análise do movimento humano	(1) Robotran, Bullet, XDEE

Fonte: Retirado e adaptado de Ivaldi, Padois e Nori (2014)

O Gazebo é utilizado como *software* de simulação em Hunter Young (2019), referência principal para esta monografia (mais informações serão apresentadas a seguir na seção 2.3). O Quadro 5 apresenta os motivos pelos quais ele foi escolhido entre os *softwares* de simulação.

Quadro 5 – Justificativas para o uso do Gazebo

Motivos
É completamente *open-source*.
Tem um apoio mais amplo e ativo da comunidade.
É melhor mantido e documentado.
Tem uma integração com ROS mais fácil.
Permite criar e modificar mais facilmente objetos e mundos.
Melhor custo-benefício em relação a recursos disponíveis e uso de poder computacional para simulações precisas de ambientes complexos.

Fonte: Adaptado de Hunter Young (2019)

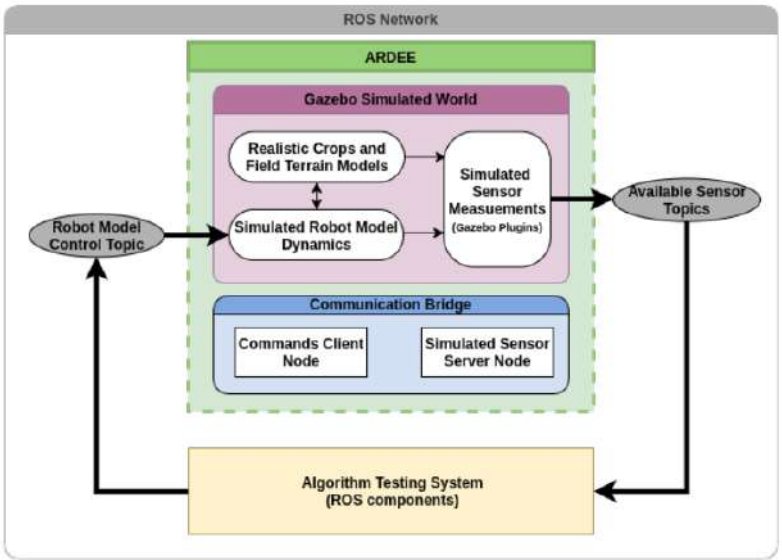
2.3 ARDEE: Desenvolvimento de um ambiente de simulação no Gazebo voltado à agricultura

Em Hunter Young (2019) é apresentado o trabalho intitulado *"ARDEE: A general agricultural robotic development and evaluation environment"*, cujo intuito é iniciar o desenvolvimento de um ambiente de simulação de código aberto que permita simular suficientemente ambientes agrícolas e que possa apoiar a comunidade de pesquisa de robótica em campo. Há três objetivos visados:

- **Disponibilizar uma coleção suficientemente realista de campos e de culturas:** isso permite que qualquer elemento essencial para o desenvolvimento e avaliação de robôs de campo como, por exemplo, sensores e dinâmica, possam ser simulados com uma relativa precisão.
- **Permitir uma fácil customização dos campos e das configurações robóticas:** dessa forma, os tipos e as posições das culturas, o solo e os sensores disponíveis no robô, por exemplo, poderiam ser configurados rapidamente.
- **Suportar a integração com o *hardware* do robô:** com isso é possível avaliar o que foi desenvolvido no próprio *hardware* do robô utilizando a simulação para gerar dados de sensores e de dinâmica, por exemplo.

Em relação ao desenvolvimento, implantação e avaliação de algoritmos, a ideia do trabalho é suportar duas abordagens:

Figura 6 – Simulação independente

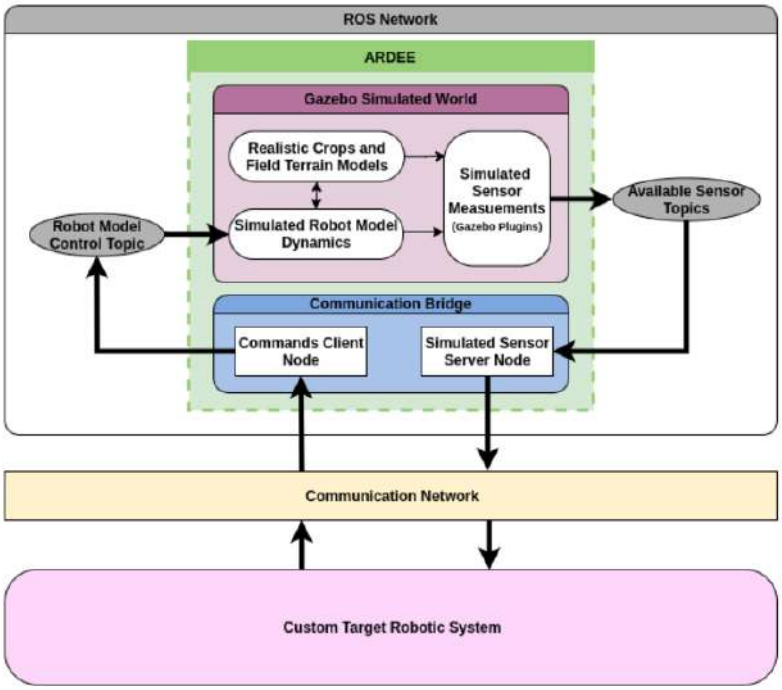


Fonte: Retirado de Hunter Young (2019)

- **Simulação independente:** esta abordagem não considera a integração com o *hardware* físico da aplicação. Nela, os algoritmos presentes no robô são executados na mesma máquina onde roda a simulação ou em uma máquina dedicada. O funcionamento considerado é ilustrado na Figura 6.

- **Simulação híbrida:** esta abordagem considera a simulação integrada com o *hardware* físico, ou seja, um teste HIL. Nela, há uma ponte UDP que faz a comunicação entre a simulação e *hardware*. O *hardware* recebe as informações da simulação (como por exemplo, dados de sensores), faz o processamento dos algoritmos e retorna para a simulação os comandos necessários. A Figura 7 ilustra o funcionamento considerado.

Figura 7 – Simulação híbrida

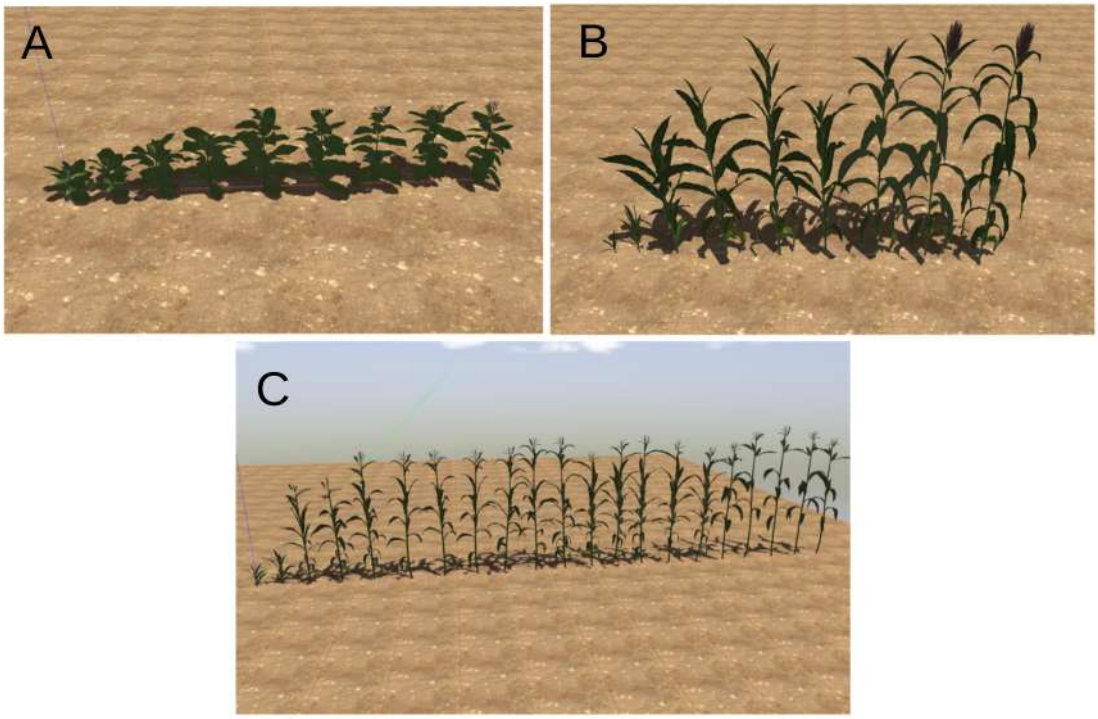


Fonte: Retirado de Hunter Young (2019)

Dessa forma, ao considerar as duas abordagens de simulação, ARDEE permite que os algoritmos sejam desenvolvidos sem que os componentes de *harware* estejam presentes, gera uma economia de tempo na integração com *hardware* nos estágios iniciais de desenvolvimento e disponibiliza meios de otimizar o tempo de testes em campo real (já que estes são escassos e não estão disponíveis durante todo o ano).

A respeito dos modelos de culturas disponíveis na simulação, há 9 variações de tabaco (Figura 8-A), 9 variações de modelos de sorgo (Figura 8-B) e 22 variações de modelos de milho (Figura 8-C). Eles não são destinados a todas as aplicações agrícolas possíveis em ambientes de simulação, como por exemplo, não é possível realizar simulações de colheita com esses modelos.

Figura 8 – Modelos disponíveis



Fonte: Retirado e adaptado de Hunter Young (2019)

A criação de modelos de alta-fidelidade é algo complexo e que demanda de conceitos específicos de modelagem gráfica 3D, dessa forma, não há uma grande quantidade de modelos realistas de plantações disponíveis. Então, o autor de ARDEE considerou a seguinte abordagem para a aquisição dos modelos presentes na Figura 8 e a consequente aplicação no Gazebo (será apresentada uma breve descrição das principais etapas realizadas):

1. **Aquisição de modelos realistas:** Os modelos foram comprados na internet em sites especializados em modelagem 3D. Alguns modelos da mesma cultura vinham juntos no mesmo arquivo.

2. **Pré-processamento:** Os modelos precisaram ser adaptados ao formato suportado pelo Gazebo. Para isso, utilizou-se de softwares de edição 3D específicos.

3. **Configuração do diretório dos modelos:** Os modelos foram configurados para o padrão GMDB (Gazebo Model Database), padrão que permite ao Gazebo reconhecê-los, para isso, foi necessário realizar uma configuração nos diretórios dos arquivos.

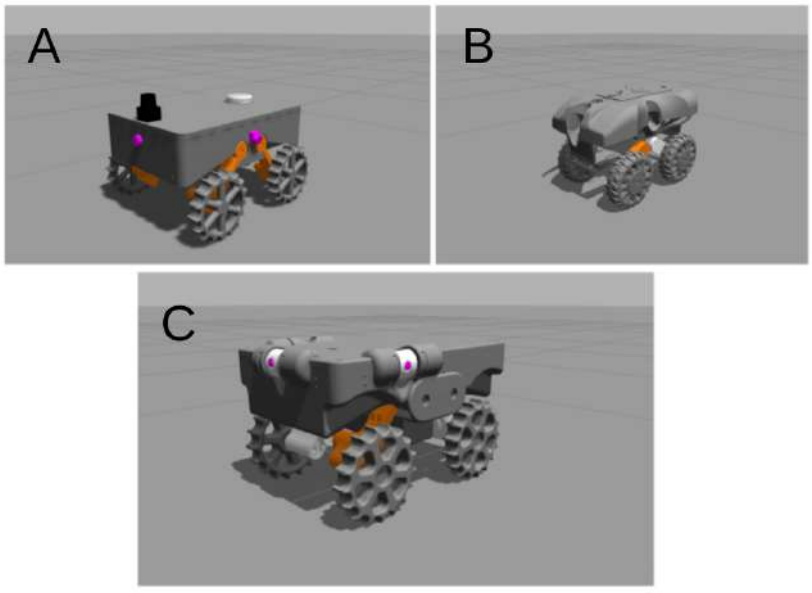
4. **Adição de elemento de colisão aos modelos:** A fim de deixar o comportamento das colisões mais real, considerou-se a adição de um cilindro no centro dos modelos. Dessa forma, um robô pode colidir e parar apenas ao ter contato com o centro da planta. A Figura 9 apresenta exemplos do elemento de colisão.

Figura 9 – Elemento de colisão nos modelos (cilindros em laranja)

Fonte: Retirado de Hunter Young (2019)

Como cada aplicação da robótica na agricultura requer uma configuração específica de robô, como por exemplo, sensores específicos, torna-se necessário que a composição dos parâmetros da plataforma robótica na simulação seja modular a fim de poupar tempo de desenvolvimento. Dessa forma, é utilizado um formato chamado Unified Robot Description Format (URDF), presente em Open Robotics (2021), e implementado uma linguagem chamada XML Macros (xacro), disponível em Open Robotics (2021), para a geração dos modelos robóticos. Por exemplo, os robôs presentes na Figura 10 foram construídos nesse formato.

Figura 10 – Robôs TerraSentia no formato URDF/xacro



Fonte: Retirado e adaptado de Hunter Young (2019)

Uma curiosidade a respeito do formato URDF é que em Choi *et al.* (2021) é dito

que ele é o primeiro passo para o estabelecimento de uma biblioteca aberta com uma curadoria de modelos validados pela comunidade (uma das recomendações presentes no Quadro 4).

Os robôs TerraSentia (EARTHSENSE, 2021), com gerações apresentadas na Figura 10, são comumente utilizados em aplicações reais de coleta automatizada de características de plantas em campo. A Figura 10-A apresenta a primeira geração, Figura 10-B exibe a versão de 2019 e a Figura 10-C mostra a versão de 2018 do robô. As três versões estão disponíveis na simulação e podem ser facilmente customizadas.

Em relação aos sensores disponíveis e as suas respectivas configurações, ARDEE apresenta uma coleção de modelos de sensores comumente utilizados em simulações no Gazebo. A Tabela 4 apresenta a lista com alguns dos sensores que podem ser utilizados e configurados de forma rápida na simulação.

Tabela 4 – Sensores disponíveis

Tipo do sensor	Nome
GPS	GPS padrão, GPS com precisão variável
IMU	Hector IMU, Hector IMU modificada
2D Lidar	Hokuyo URG-04LX, Hokuyo UST-10LX, Sick S3000, RPLidar Hokuyo UTM-30LX, Sick S300, Sick TIM 571
3D Lidar	Hokuyo 3D
Câmera RGB	Axis M5013, Axis P5512, GigE uEye CP
Câmera RGBD	Asus Xtion Pro, Kinect, Kinect V2, Fotonic E Series, OrbBec Astra
Diversos	Sonar, Bateria, Encoder rotativo

Fonte: Retirado e adaptado de Hunter Young (2019)

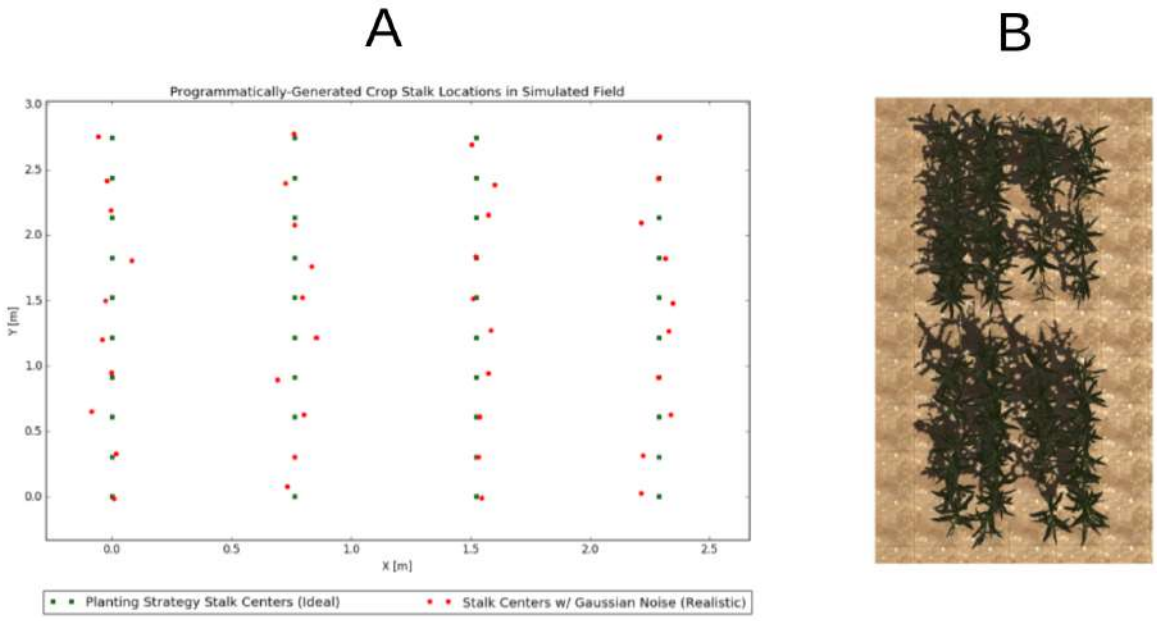
Com o objetivo de facilitar a criação de áreas de cultivo na simulação, em ARDEE também é apresentada uma ferramenta que permite configurar de forma rápida campos de plantações: consiste em um código em Python (PYTHON SOFTWARE FOUNDATION, 2021) que gera campos customizáveis e realistas.

O código gera as plantações através de pontos no plano cartesiano, sendo que cada ponto consiste no centro de uma planta (os centros sempre são dispostos idealmente em linhas retas e as plantas são sempre posicionadas em $z = 0\ m$). Para assemelhar-se com plantações reais, considerando por exemplo o crescimento natural das plantas, são feitas as seguintes considerações (a Figura 11-B apresenta um exemplo de plantação gerada):

- **A planta não cresce exatamente onde foi plantada:** É adicionado um ruído na posição de cada planta a fim de simular o deslocamento da semente (o que pode

ser causado por chuva, vento ou animais, por exemplo) de sua posição inicial de plantio. A Figura 11-A ilustra essa consideração.

Figura 11 – Geração das plantações



Fonte: Retirado e adaptado de Hunter Young (2019)

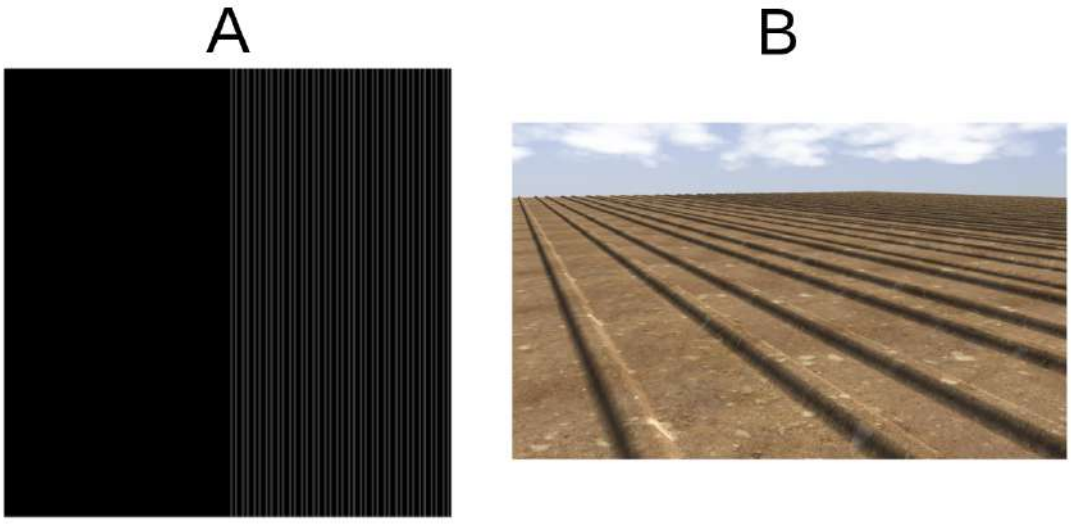
- **Nem todas as plantas crescem:** É adicionada uma probabilidade de crescimento na plantação com a finalidade de simular plantas que não cresceram durante o cultivo (o que pode ser causado por fatores como insuficiência de nutrientes no solo, por exemplo).
- **Cada planta é única:** São adicionados modelos distintos da mesma cultura e cada um é inserido na plantação com uma rotação aleatória. Isso tenta simular o fato que cada planta tem características únicas de, por exemplo, posição das folhas e largura do caule.

Sobre o solo dos campos de cultivo, o autor faz as seguintes considerações na simulação: o terreno é formado por um único corpo rígido, com propriedades dinâmicas constantes e uniformes por todo o modelo.

Para criar o terreno, o Gazebo suporta uma integração entre imagens em escala de cinza e imagens de texturas. A primeira é responsável por definir a elevação do terreno, no caso, quanto mais próximo do branco, mais alto é o terreno, já o segundo é responsável por dar as características visuais ao solo. A Figura 12-B ilustra um terreno gerado com a

combinação da imagem em escala de cinza presente na Figura 12-A e algumas texturas padrões do Gazebo.

Figura 12 – Geração do solo



Fonte: Retirado e adaptado de Hunter Young (2019)

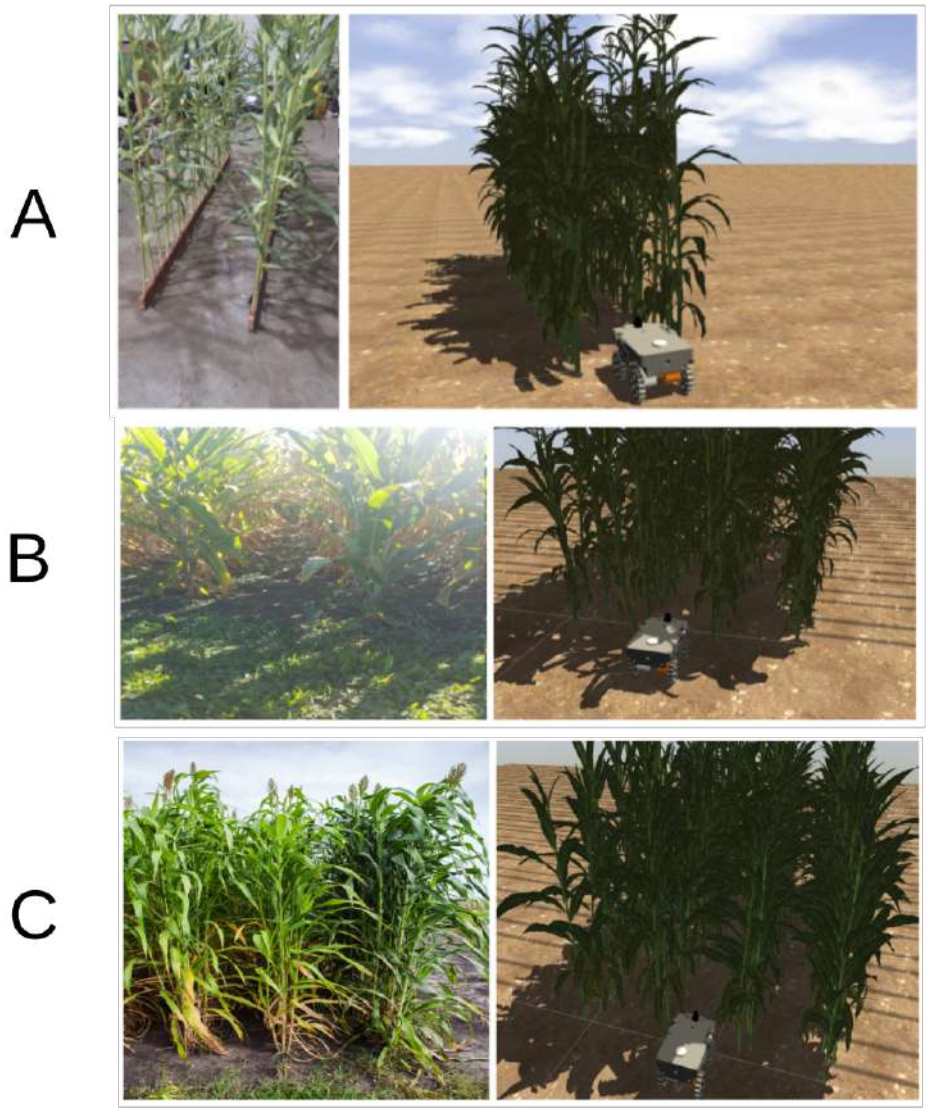
Por fim, em Hunter Young (2019) são feitos alguns experimentos envolvendo ambientes reais e ambientes simulados. O objetivo é a partir de ambientes reais, replicá-los no Gazebo (utilizando das ferramentas descritas anteriormente) e compará-los através da execução de um algoritmo de navegação baseado em sensor Lidar, presente em Higuti *et al.* (2019) e que será melhor detalhado na seção 2.6. Em todos os experimentos realizados os algoritmos são processados no próprio *hardware* do robô, ou seja, ao considerar o ambiente simulado implementa-se o esquemático da Figura 7. Foram considerados os seguintes cenários para a realização dos experimentos:

- **Campo *indoor* de milho:** Esse cenário considera a realização de um teste *indoor* com fileiras de milho. A Figura 13-A ilustra o ambiente real e a respectiva imitação no Gazebo.

- **Campo real de milho:** Esse cenário considera a realização do teste em um campo real de milho. A Figura 13-B exibe o ambiente real e a respectiva versão no Gazebo.

- **Campo real de sorgo:** Esse cenário considera a realização do teste em um campo real de sorgo. A Figura 13-C mostra o ambiente real e a respectiva adaptação no Gazebo.

Figura 13 – Experimentos realizados



Fonte: Retirado e adaptado de Hunter Young (2019)

Como conclusão acerca dos experimentos realizados, os ambientes simulados mostraram-se capazes de replicar com uma relativa precisão os dados do Lidar.

2.4 Adição de modelos animados no Gazebo

Em trabalhos como He *et al.* (2020) e Giubilato *et al.* (2020) são apresentadas simulações no Gazebo que fazem o uso de modelos animados. No primeiro, para tratar de sistemas de colaboração humano-robô, é utilizado um modelo animado de uma pessoa, já no segundo, para abordar a validação de algoritmos de mapeamento e de localização, utiliza-se de um modelo de *rover* animado com uma trajetória predefinida.

São utilizadas em ambas as publicações, He *et al.* (2020) e Giubilato *et al.* (2020), uma função do Gazebo chamada *Actor*. Com ela é possível adicionar recursos de animação

a modelos comuns, como por exemplo, os modelos presentes na Figura 8. Ao adicionar uma animação, o modelo passa a não ser afetado pela parte física da simulação, deixando assim de interagir com o resto do ambiente simulado. Dessa forma, por exemplo, uma animação não é impactada pela gravidade e não pode colidir com outros objetos.

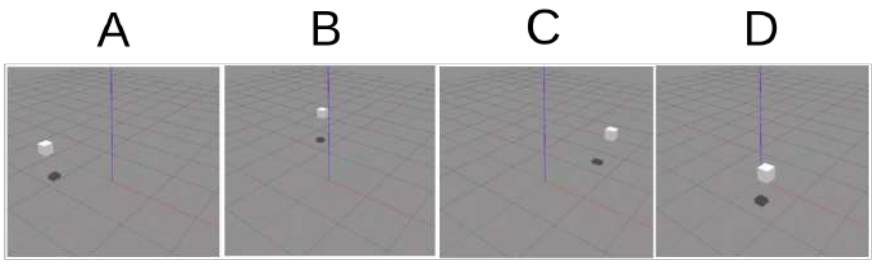
Em relação ao comportamento com sensores, uma animação pode ser visualizada por câmeras RGB e detectadas por sensores de profundidade, como por exemplo, Lidar.

Há dois tipos de animações suportadas no *Actor*/Gazebo, a primeira, chamada de animação esqueleto, consiste em uma animação que apresenta um modelo com movimento relativo entre os seus links (por exemplo, a animação de uma pessoa andando ou fazendo um movimento específico). Para criar uma animação do tipo é necessário utilizar-se de *softwares* específicos e importar o resultado para o Gazebo. Já a segunda animação suportada, resume-se em movimentar um modelo por uma trajetória definida na simulação. É possível combinar os dois tipos de animações, gerando assim, por exemplo, uma pessoa andando em uma trajetória predefinida.

Para traçar a trajetória de um modelo é necessário apresentar uma lista de *waypoints*, informando a pose e o momento específico que ela deve ser alcançada. Por exemplo, a Figura 14 exibe a movimentação de um cubo por uma trajetória em *loop* predefinida.

- **Figura 14-A:** O cubo deve alcançar a posição $[-1, -1, +1]$ em $t = 0\ s$.
- **Figura 14-B:** O cubo deve alcançar a posição $[-1, +1, +1]$ em $t = 1\ s$.
- **Figura 14-C:** O cubo deve alcançar a posição $[+1, +1, +1]$ em $t = 2\ s$.
- **Figura 14-D:** O cubo deve alcançar a posição $[+1, -1, +1]$ em $t = 3\ s$.

Figura 14 – Exemplo de modelo animado no Gazebo



Fonte: Retirado e adaptado de Open Source Robotics Foundation (2021)

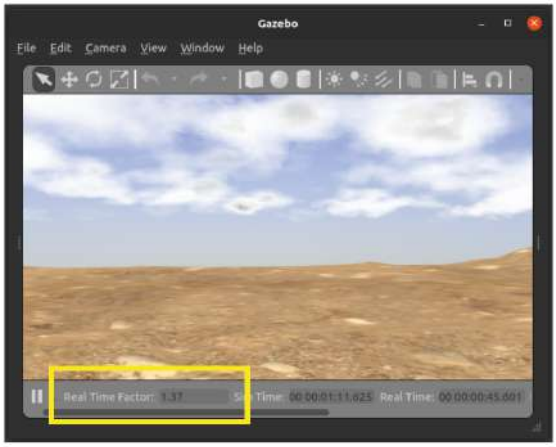
2.5 Métricas de desempenho de simuladores

Em Pitonakova *et al.* (2018) é feito um estudo comparando o desempenho de alguns simuladores presentes no mercado. Por exemplo, são citados e comparados *softwares* como

Gazebo e V-Rep. Para isso, são utilizados os seguintes parâmetros de desempenho:

- ***Real-time factor (R):*** Consiste na razão entre o tempo da simulação e o tempo real. Ele mensura o quanto a simulação executada está mais rápida ou mais lenta em comparação com o tempo real. Essa relação depende de quanta computação está sendo necessária para executar a simulação. A Figura 15 apresenta a interface gráfica do Gazebo e em destaque, o indicador do *real-time factor*.

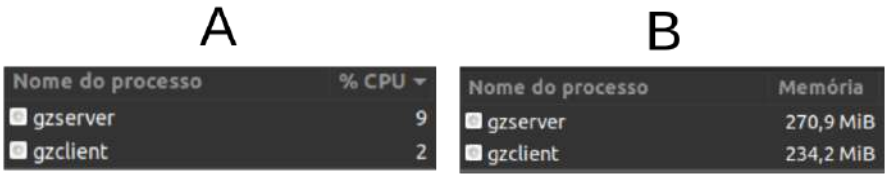
Figura 15 – *Real-time factor* na interface gráfica do Gazebo



Fonte: Elaborado pelo autor

- **Memória (M) e CPU (C):** Indicam, respectivamente, a quantidade de memória RAM e a quantidade de CPU utilizadas durante a simulação. A Figura 16 exibe a utilização dos recursos computacionais por parte dos processos executados pelo Gazebo, sendo que:
  - **gzclient:** Consiste na interface gráfica. Nela é possível visualizar e controlar algumas propriedades da simulação.
  - **gzserver:** Executa as atualizações físicas e a geração dos dados dos sensores na simulação. É o *core* do Gazebo e pode ser executado de forma independente (sem a interface gráfica).

Figura 16 – Memória e CPU utilizadas durante um exemplo de simulação no Gazebo



Fonte: Elaborado pelo autor

Já em Saglam e Papelis (2020), além de usar o *real-time factor*, os autores utilizam como métrica o tempo real decorrido para simular uma quantidade de tempo pré-definida, no caso, eles utilizam como base o tempo gasto para completar 120 segundos no simulador.

2.6 Algoritmos de navegação

Conforme comentado na seção 2.3, em Higuti *et al.* (2019) é apresentado um algoritmo de navegação baseado em um único sensor Lidar. Com ele, um robô móvel terrestre consegue navegar em faixas estreitas e desordenadas de campos agrícolas com alta cobertura foliar, com oclusão frequente de sensor e com linhas vizinhas visíveis. As seguintes considerações são feitas para aplicação do algoritmo em campo: as faixas apresentam largura conhecida, as linhas da plantação são paralelas (não necessariamente precisam ser retas, podem ser um pouco curvas) e não há obstáculos nas faixas percorridas pelo robô. Considerando uma breve descrição do seu funcionamento, através dos dados enviados pelo sensor Lidar, o algoritmo é responsável por estimar as linhas da plantação à esquerda e à direita do robô, validar essa estimativa e enviar os comandos de controle para o robô permanecer no centro da faixa.

A respeito de limitações e sugestões para trabalhos futuros, os autores comentam sobre a aplicação do algoritmo em faixas com curvas grandes e sobre a integração do algoritmo com mais sensores, por exemplo. Este último, foi apresentado recentemente em Velasquez *et al.* (2021) e consiste na integração do algoritmo apresentado anteriormente com *encoders* e IMU.

3 DESENVOLVIMENTO

O objetivo deste capítulo é exibir o que foi desenvolvido com base na análise de pontos apresentados no capítulo 2. O capítulo será organizado da seguinte maneira: a seção 3.1 apresentará a integração entre o ambiente de simulação presente em Hunter Young (2019) e o código de navegação desenvolvido em Velasquez *et al.* (2021), e as seções 3.2 e 3.3 exibirão abordagens complementares ao que foi exposto na seção 2.3.

Todas as implementações e testes foram feitos em um computador Dell G3 3579, com 16 GB de memória, processador Intel Core i7 de 8ª geração, placa de vídeo Nvidia GeForce GTX 1050 Ti e Ubuntu 20.04.2 LTS. A versão utilizada do Gazebo é a 11.5.1 e do ROS é a noetic.

3.1 Integração entre ambiente de simulação e código de navegação

A fim de permitir a realização de testes no ambiente de simulação do algoritmo de navegação presente em Velasquez *et al.* (2021), foi feita a integração entre o trabalho descrito na seção 2.3 e o código de navegação. É considerado apenas testes SIL (diagrama presente na Figura 6), ou seja, não é necessário ter acesso ao *hardware* utilizado no robô real para realizar os testes.

Tanto o código de Hunter Young (2019) quanto o de uma das versões do algoritmo apresentado em Velasquez *et al.* (2021) foram disponibilizados pelo Distributed Autonomous Systems Laboratory (DASLAB TEAM, 2021).

Antes de realizar qualquer alteração nos códigos, foi necessário entender a estruturação dos repositórios disponibilizados e o fluxo das informações ao executar os processos. Após algumas tentativas de realizar a comunicação entre o ambiente de simulação e o algoritmo implementado em robôs reais foi observada a necessidade de efetuar algumas adaptações no código de Hunter Young (2019). Elas estão expostas a seguir:

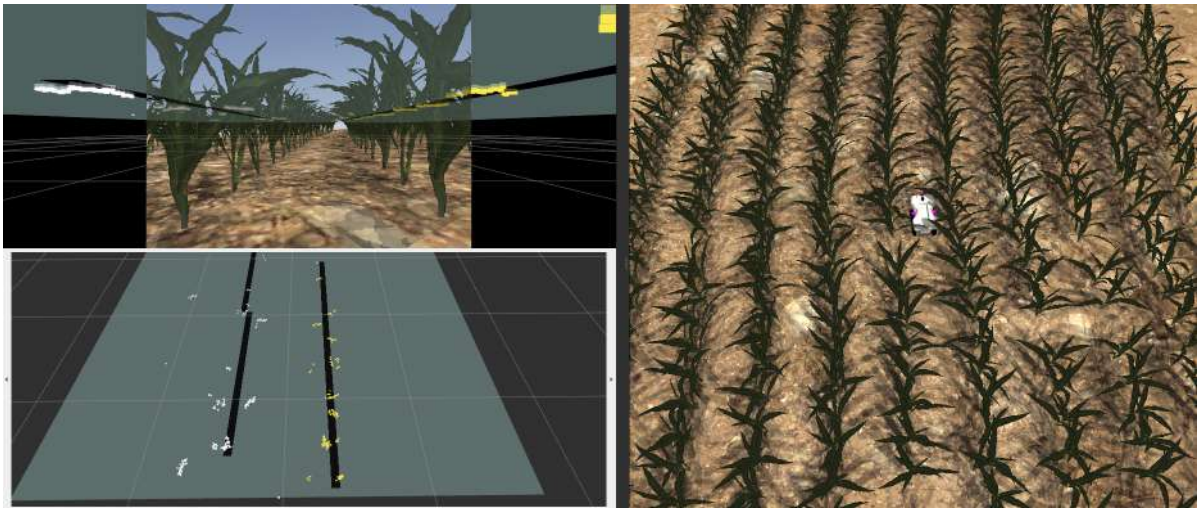
- **Correções no ambiente de simulação devido ao versionamento do Gazebo:** Devido ao fato do trabalho de Hunter Young (2019) ter sido feito em uma versão diferente do Gazebo, houve alguns problemas ao utilizá-lo na versão mais recente do simulador. Por exemplo, os modelos de plantas não estavam aparecendo corretamente e os sensores Lidar disponíveis não estavam funcionando. O primeiro erro foi corrigido adicionando a tag **<name>** em cada modelo e o segundo, era um erro de notação da linguagem xacro.

- **Alteração nos tópicos dos sensores:** As principais alterações feitas foram nos tópicos responsáveis por exibir os dados dos sensores. O tópico responsável por

transmitir as informações do Lidar teve que ser renomeado. Os dados dos *encoders* e do GPS tiveram que ser alterados para os padrões de mensagens suportados pelo algoritmo de navegação.

A Figura 17 ilustra o algoritmo de navegação integrado com o ambiente de simulação. À direita é apresentado o TerraSentia percorrendo uma plantação de milho e à esquerda a visualização de alguns dos parâmetros calculados pelo algoritmo (em amarelo pontos à direita do robô e em branco pontos à esquerda).

Figura 17 – Integração entre simulação e código de navegação



Fonte: Elaborado pelo autor

Durante alguns testes com o algoritmo foi observado um erro no controle de navegação do robô. Ele não estava seguindo a faixa da plantação como esperado. Depois de algum tempo foi observado que o erro ocorria devido a um controlador PID implementado. Ao ser removido, o TerraSentia passou a percorrer a plantação como esperado. Este erro já havia sido corrigido pelos autores na versão mais recente do algoritmo.

3.2 Interface gráfica para gerar plantações na simulação

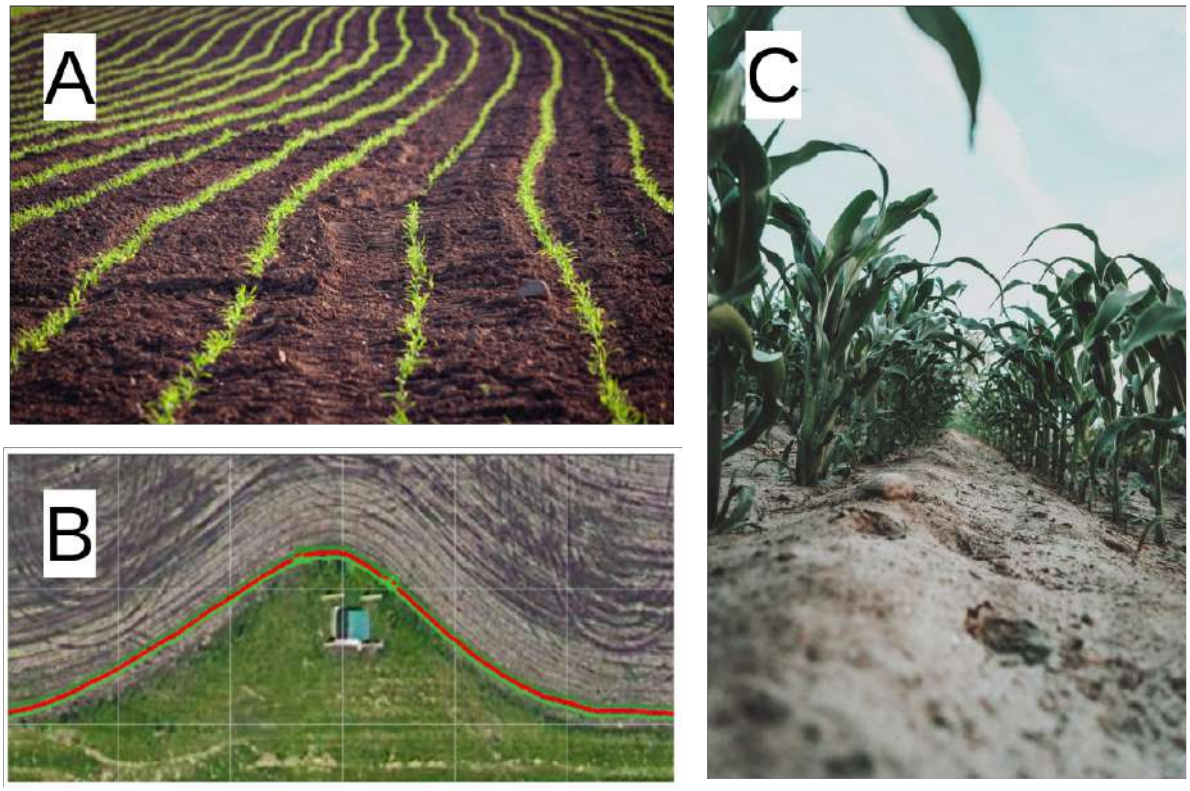
Conforme visto na seção 2.3, ao gerar os mapas da simulação de forma automatizada, são feitas algumas considerações:

- **As linhas da plantação são idealmente retas:** É considerado que a plantação apresenta faixas formadas apenas por linhas retas, conforme pode ser visualizado na Figura 11.
- **Não há integração entre a geração do solo e da plantação:** Ao gerar a plantação, não é levado em conta a elevação do solo. Com isso, só é possível criar plantações em terrenos majoritariamente planos, conforme exemplificado na Figura 13.

- **O solo apresenta propriedades constantes:** É considerado que todo o terreno é um modelo único com propriedades dinâmicas constantes e uniformes.

Dessa forma, ao levar em conta as duas primeiras considerações, plantações como as presentes na Figura 18 não podem ser criadas de forma automatizada e reproduzidas em sequência na simulação. As duas primeiras imagens, Figura 18-A e Figura 18-B, apresentam plantações cujas faixas não são compostas por linhas retas. Já a Figura 18-C exibe uma plantação em um terreno acidentado.

Figura 18 – Exemplos de plantações reais



Fonte: Retirado de UNSPLASH (2021), Velasquez *et al.* (2021) e UNSPLASH (2021), respectivamente

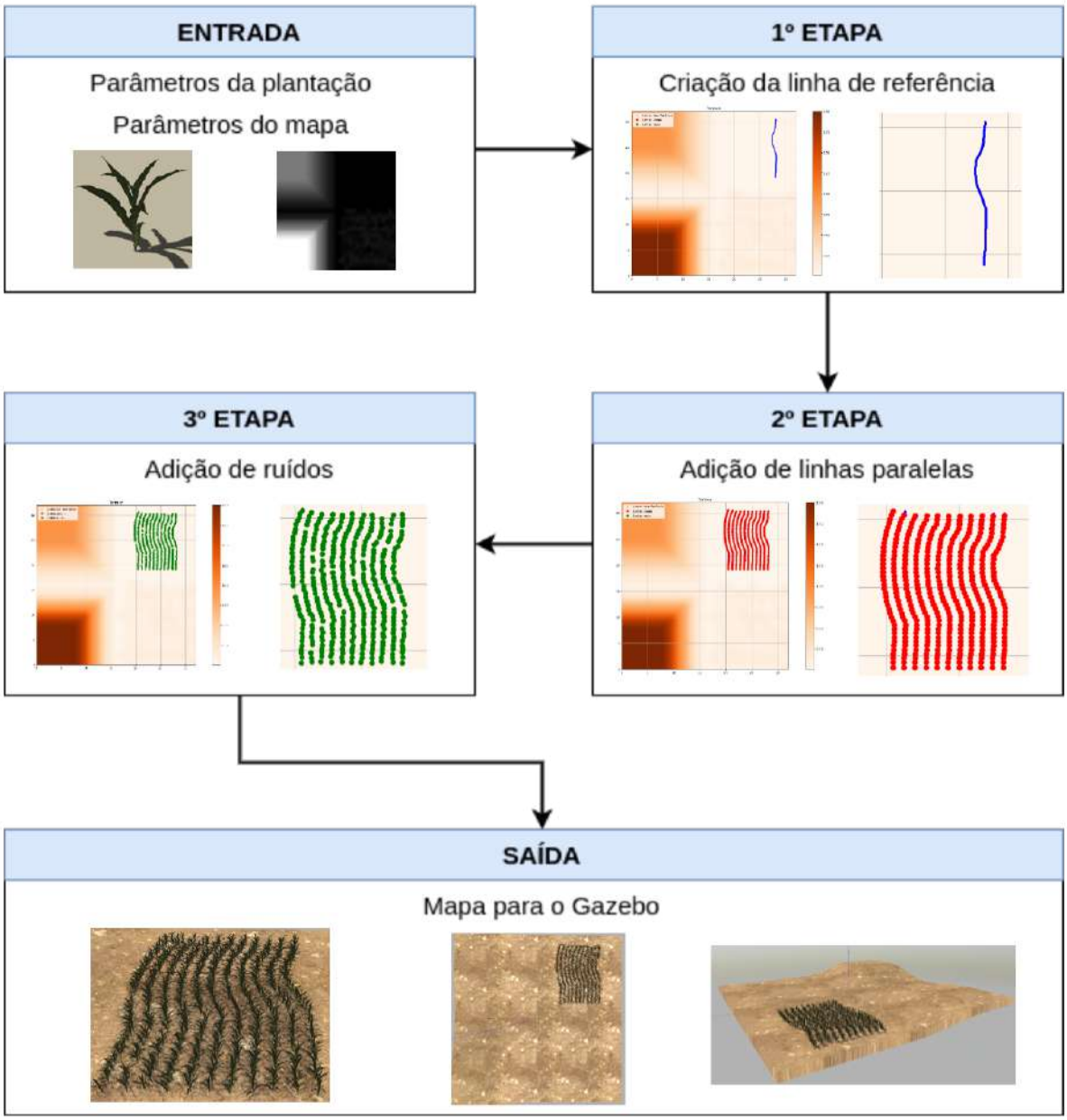
A respeito da terceira consideração, ela remove a possibilidade de deixar a dinâmica do solo mais próxima da realidade. Como por exemplo, com ela não é possível ter um terreno com áreas com propriedades dinâmicas diferentes.

A ideia então é complementar a ferramenta desenvolvida por Hunter Young (2019) de forma a permitir a criação de qualquer tipo de plantação, como por exemplo, as apresentadas na Figura 18. A subseção 3.2.1 apresentará as etapas da ferramenta desenvolvida e a subseção 3.2.2 exibirá alguns resultados.

3.2.1 Funcionamento e implementação

A ferramenta foi desenvolvida em linguagem Python. A Figura 19 apresenta o diagrama do seu funcionamento.

Figura 19 – Etapas para a criação da plantação



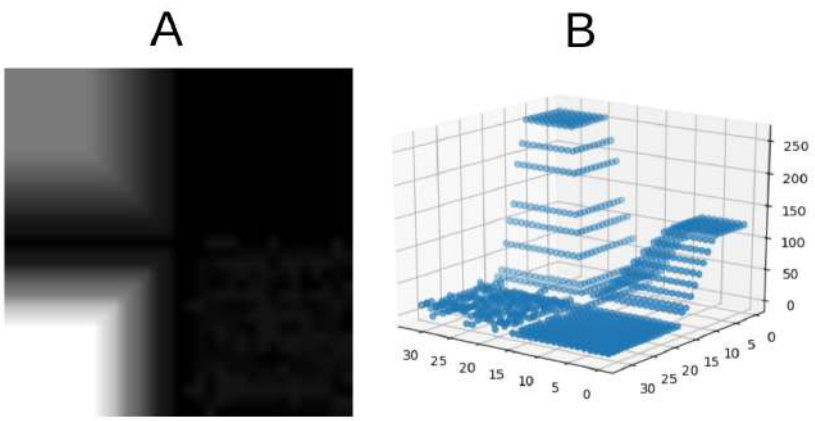
Fonte: Elaborado pelo autor

A seguir serão descritas cada umas das etapas:

- **Entrada:** Além dos parâmetros da plantação já utilizados em Hunter Young (2019), como por exemplo, distância entre linhas paralelas, distância entre as plantas,

probabilidade de uma planta crescer, distância máxima de uma semente do seu lugar de plantio e modelos 3D disponíveis da cultura, adicionou-se a necessidade de fornecer os parâmetros do mapa: dimensões, escala em tons de cinza de sua elevação e a sua elevação máxima. Na Figura 20-A é apresentado o modelo padrão utilizado para gerar o terreno: consiste em uma imagem de 33×33 *pixels* com tons de cinza que variam de 0 (preto) até 255 (branco). A Figura 20-B ilustra a respectiva escala de cinza da imagem.

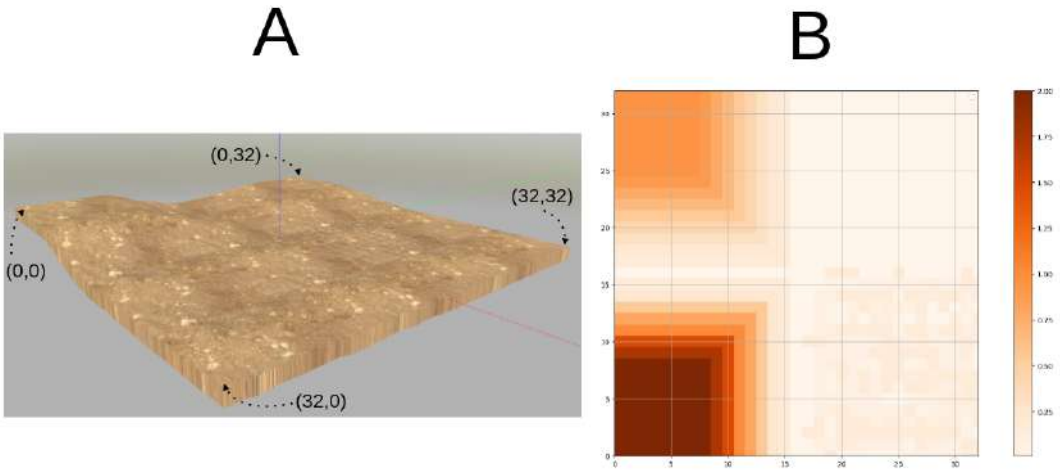
Figura 20 – Exemplo de imagem em escala de cinza



Fonte: Elaborado pelo autor

A Figura 21-A ilustra o terreno gerado no Gazebo utilizando essa imagem em tons de cinza (o terreno tem 33 m de largura por 33 m de comprimento). No caso, o pixel branco (escala 255 de cinza) corresponde a uma altura de 2 metros (Figura 21-B).

Figura 21 – Exemplo de solo gerado

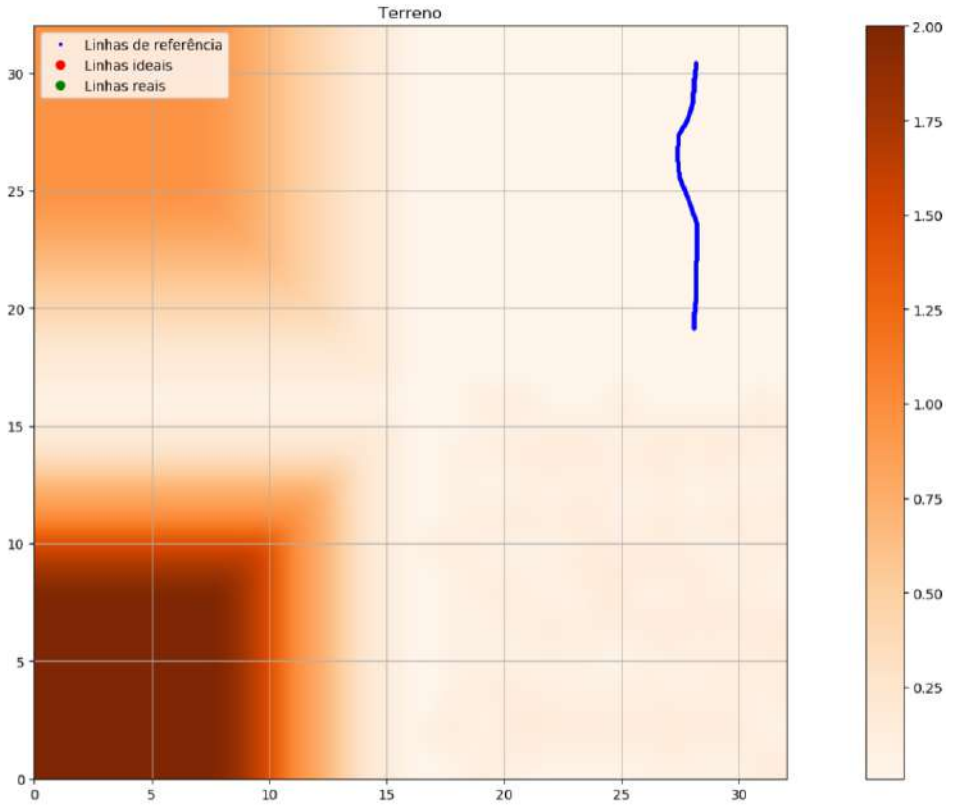


Fonte: Elaborado pelo autor

A ideia de considerar Figura 20-A como imagem-padrão na geração do terreno é que com ela é possível deixar o mapa o mais genérico e abrangível possível: o terreno é completamente plano no canto superior direito, é irregular no canto inferior direito, tem um desnível pouco acentuado no canto superior esquerdo e muito acentuado no canto inferior esquerdo. Com isso, é possível utilizar o mesmo mapa para uma vasta gama de aplicações na simulação.

- **1ª Etapa:** A primeira etapa consiste em criar a linha de referência da plantação. Através do movimento do *mouse* na interface gráfica é criado o perfil desejado e a partir dele será possível gerar curvas paralelas. A Figura 22 ilustra a interface gráfica com uma linha de referência já criada (em azul).

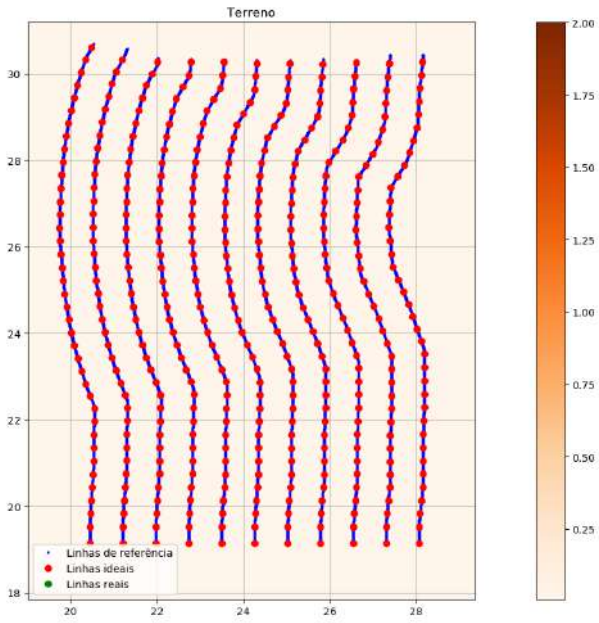
Figura 22 – Linha de referência na interface gráfica



Fonte: Elaborado pelo autor

- **2ª Etapa:** Consiste na criação das curvas paralelas ao perfil criado na primeira etapa. Ao criar uma curva paralela, automaticamente já são posicionadas, considerando um comportamento ideal, as respectivas plantas. No exemplo em questão, a plantação é de milho com linhas distantes em 0.762 m e plantas posicionadas a cada 0.3 m. Para a criação das linhas paralelas é utilizado o pacote Shapely (GILLIES *et al.*, 2007–). A Figura 23 apresenta as curvas paralelas criadas com base na linha em azul mais à direita e em vermelho, a localização de cada planta.

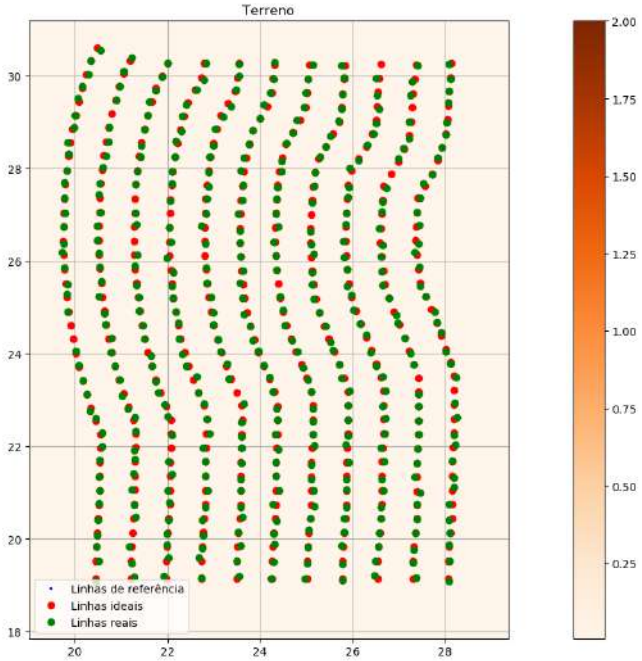
Figura 23 – Linhas paralelas e posicionamento ideal das plantas



Fonte: Elaborado pelo autor

- **3ª Etapa:** Consiste em adicionar aleatoriedades na plantação caso necessário. No caso exemplificado, a probabilidade considerada de uma planta crescer é de 95% e a máxima distância de deslocamento em relação ao local de plantio é de 7 *cm* para cada coordenada.

Figura 24 – Adição de aleatoriedades na plantação



Fonte: Elaborado pelo autor

Em verde estão localizadas as plantas após a adição dessas aleatoriedades e em vermelho, o posicionamento original e idealizado do plantio. É possível adicionar ou remover plantas, conforme o interesse, clicando sobre o terreno ou sobre a respectiva planta que deve ser removida.

Ao final desta etapa, temos uma lista da posição (x, y) de cada planta representada por meio do ponto verde.

- **Saída:** Por fim, é gerado o mundo em formato reconhecido pelo Gazebo (Figura 25).

Figura 25 – Plantação resultante no Gazebo

Fonte: Elaborado pelo autor

Para a criação do mundo, é necessário ter a posição de cada uma das plantas e conforme apresentado na 3ª Etapa, a posição no eixo x e no eixo y já estão determinadas, resta então, calcular a posição no eixo z.

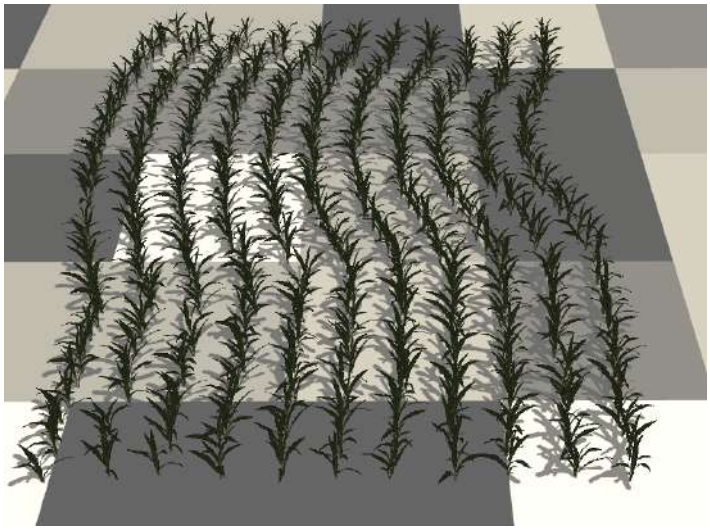
O Gazebo não apresenta nenhum método prático para fornecer a posição em z do terreno dado um ponto no plano xy. Dessa forma, é necessário realizar algumas considerações: por meio da imagem em escala de cinza (Figura 20), altura máxima do terreno e utilizando o pacote *SciPy* (VIRTANEN *et al.*, 2020), é feita uma interpolação linear em 25000000 (valor ajustado durante testes) pontos do terreno. Assim é possível ter um valor aproximado da altura do terreno no Gazebo em um ponto (x, y).

Uma outra opção de geração de mapa também está disponível e relaciona-se com a terceira consideração apresentada no início da seção 3.2 ("o solo apresenta propriedades constantes"). Ela considera a geração de um terreno plano e segmentado, com áreas que apresentam as mesmas propriedades sendo de mesma cor [1] . É necessário informar as

[1] Implementação sugerida por Mateus Valverde Gasparino (DASLAB TEAM, 2021).

características permitidas para o terreno e as dimensões dos blocos que o constituem. A Figura 26 mostra o exemplo de um terreno formado por blocos de 3 $m \times 3$ m, considerando apenas alterações no coeficiente de atrito.

Figura 26 – Terreno com áreas de propriedades distintas

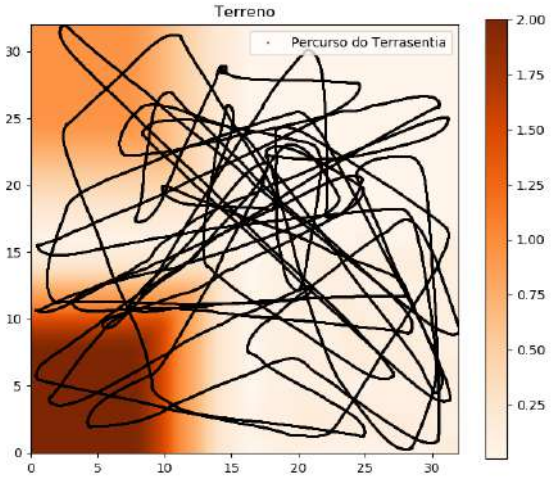


Fonte: Elaborado pelo autor

3.2.2 Resultados obtidos

Foi feito o seguinte experimento a fim de analisar o impacto da integração com o terreno no desempenho da simulação: no terreno da Figura 21-A, sem nenhuma planta, adiciona-se o TerraSentia e por meio de um algoritmo de exploração aleatória, ele percorre todo o mapa. Durante a simulação são coletados os valores do *real-time factor*. A Figura 27 ilustra o percurso do robô durante uma das iterações feitas.

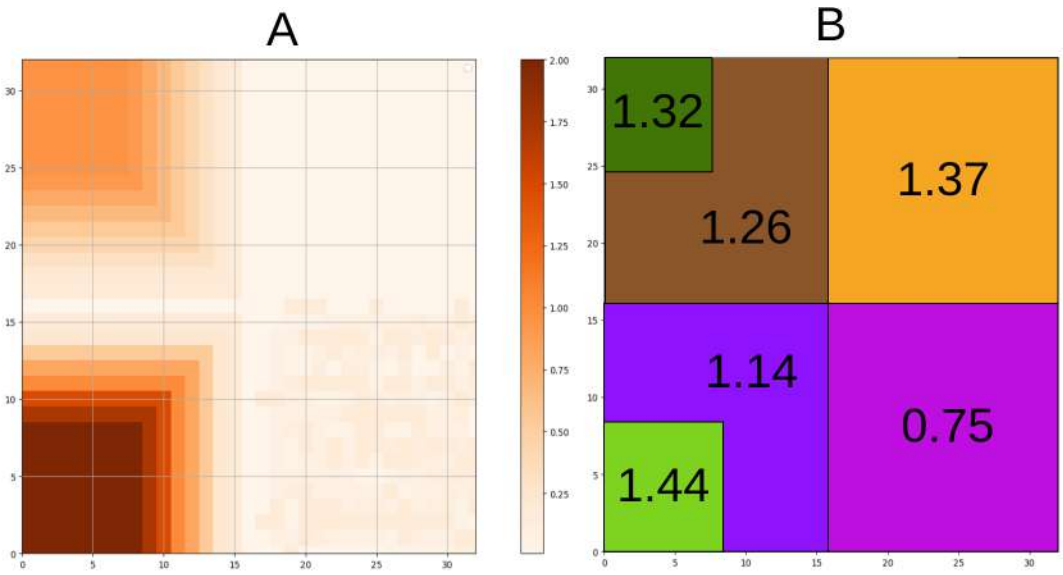
Figura 27 – Percurso do TerraSentia no terreno



Fonte: Elaborado pelo autor

A Figura 28-B apresenta o *real-time factor* médio da respectiva área do mapa. Os valores foram obtidos por meio da média de 5 iterações do algoritmo de exploração aleatória, com cada um sendo executado por cerca de 25 *min*.

Figura 28 – Média do *real-time factor* no mapa



Fonte: Elaborado pelo autor

Nota-se que as áreas planas do mapa apresentam um RF maior. Quando o terreno é irregular, conforme canto inferior direito da Figura 28-A, o desempenho da simulação é bastante prejudicado. Esses resultados levam em conta apenas o deslocamento do TerraSentia pelo mapa.

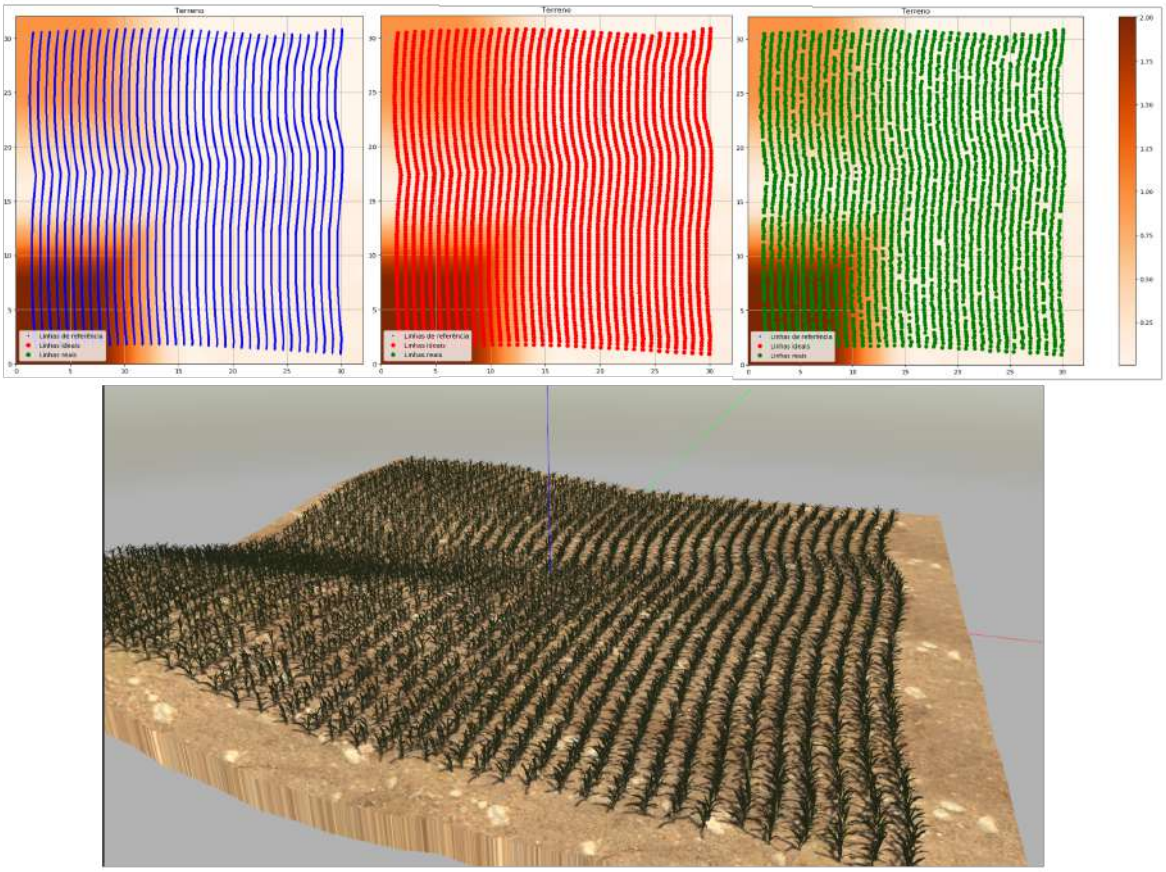
Para analisar o poder computacional necessário para preencher completamente o terreno com uma plantação, gerou-se o mapa da Figura 29. A Tabela 5 exibe as demandas computacionais de cada processo ao iniciar o Gazebo. Veja que a exigência computacional é muito alta (principalmente em relação ao uso da memória), fato que implica na necessidade de gerar mapas mais simples para a realização de experimentos.

Tabela 5 – Demanda computacional de um mapa completo

Processo	Memória (%)	CPU (%)
gzclient	23,1	8
gzserver	44,3	8

Fonte: Elaborado pelo autor

Figura 29 – Mapa totalmente preenchido com cerca de 3500 modelos de milho



Fonte: Elaborado pelo autor

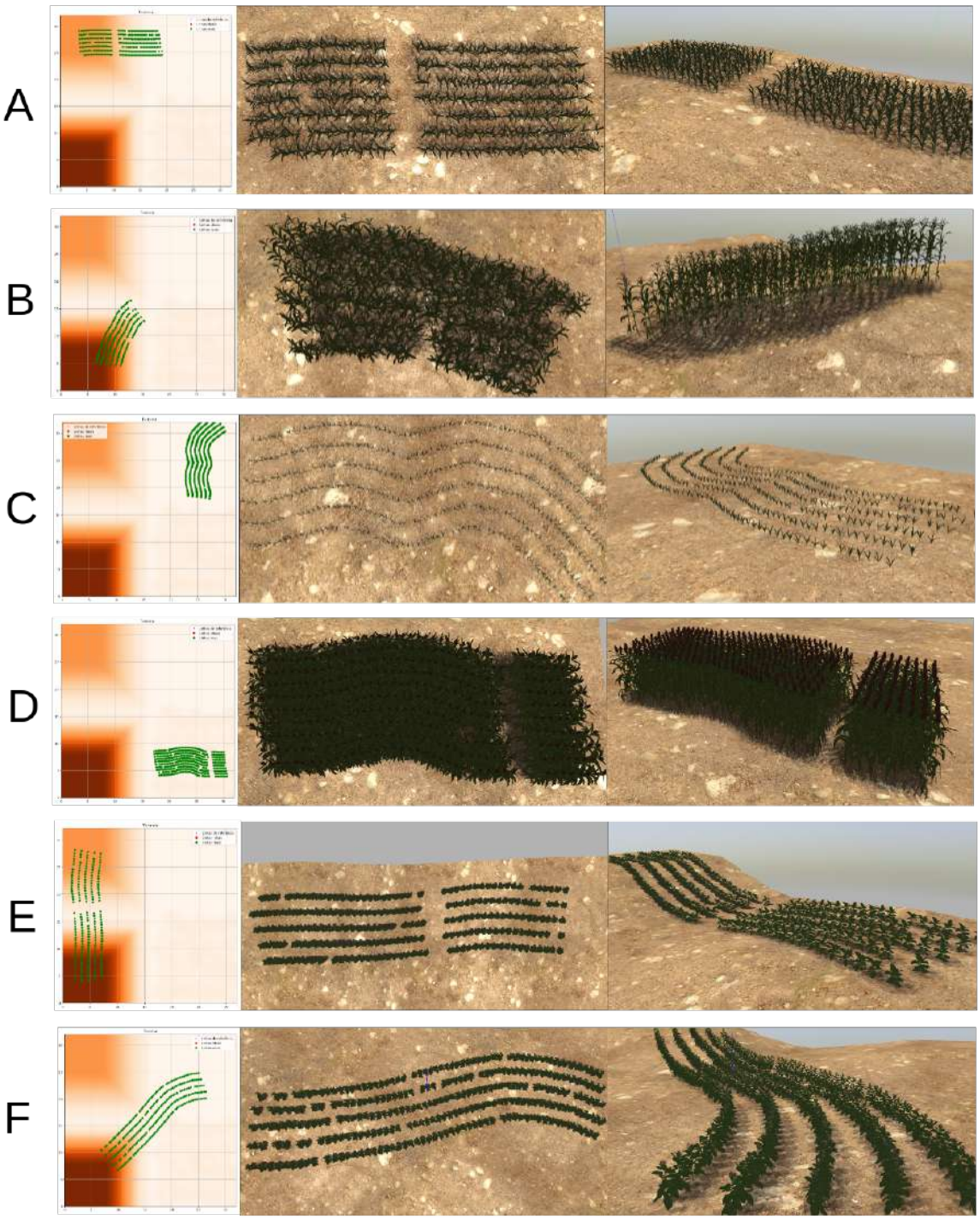
Por fim, a Figura 30 ilustra exemplos de mapas criados com a ferramenta descrita anteriormente. Os parâmetros da plantação estão presentes na Tabela 6.

Tabela 6 – Exemplos de mapas criados

Figura	Cultura	Distância entre as linhas (m)	Distância entre as plantas (m)
Figura 30-A	Milho	0.76	0.3
Figura 30-B	Milho	0.9	0.4
Figura 30-C	Sorgo	0.8	0.2
Figura 30-D	Sorgo	0.5	0.3
Figura 30-E	Tabaco	1.2	0.5
Figura 30-F	Tabaco	1.2	0.4

Fonte: Elaborado pelo autor

Figura 30 – Diversos mapas gerados com a ferramenta



Fonte: Elaborado pelo autor

3.3 Inserção de movimentos nas plantas

Uma outra consideração feita em Hunter Young (2019) é que toda a plantação é estática. Dessa forma, a ideia é complementar a abordagem e adicionar a opção das plantas terem alguma animação. Com isso, seria possível simular algumas interações com vento, por exemplo. A Figura 31 ilustra plantações reais sofrendo influência do vento.

Figura 31 – Interações da plantação com o vento

Fonte: Retirado de Thomas County Ag (2021)

A subseção 3.3.1 irá apresentar as considerações feitas e o processo de implementação. A subseção 3.3.2 exibirá os impactos dessa abordagem nos parâmetros e em alguns resultados da simulação.

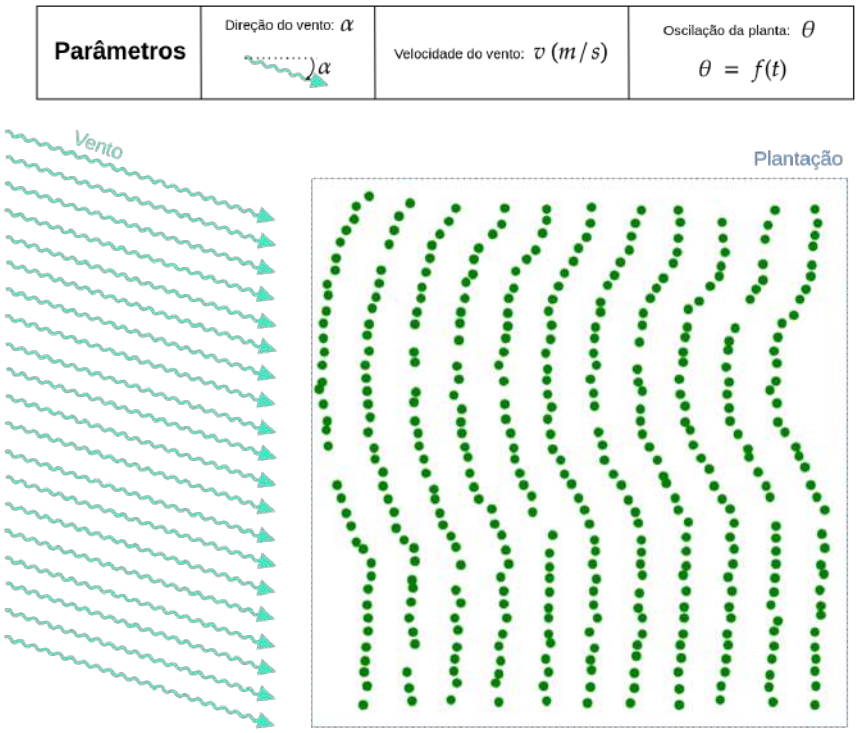
3.3.1 Considerações feitas e implementação

Conforme discutido na seção 2.4, o Gazebo permite a adição de animações utilizando a ferramenta *Actor*. Entre as duas opções permitidas, animação esqueleto ou animação de trajetória de um modelo rígido, escolheu-se por dar enfoque na segunda. A primeira permite criar animações com movimentos mais realistas, mas para isso, necessita que elas sejam criadas previamente em *softwares* de animação 3D. Outro ponto considerado é que os modelos de culturas disponíveis no repositório (Figura 8) já sofreram um pré-processamento (conforme comentado na seção 2.3) para o formato do Gazebo, fato que dificulta a importação e a manipulação em outros programas.

Dessa forma, para uma abordagem inicial, faz sentido considerar as plantas como modelos rígidos com a animação sendo causada por movimentos de rotação e translação. Estes podem ser gerados de forma rápida já que não necessitam da realização de um pré-processamento nos arquivos do repositório.

A implementação foi feita em Python e de forma integrada com a ferramenta de geração de mapas descrita na seção 3.2. A Figura 32 apresenta um diagrama ilustrando a ideia que será explicada a seguir.

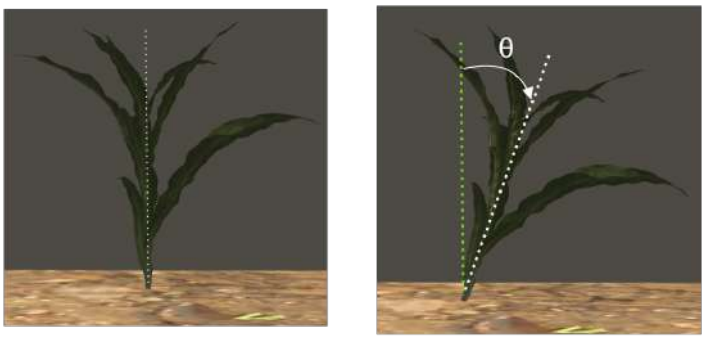
Figura 32 – Considerações feitas para a implementação do vento



Fonte: Elaborado pelo autor

A abordagem considera que o vento atua de forma uniforme e sobre toda a plantação, sendo descrito pelos seguintes parâmetros: seu sentido de atuação (ângulo α), sua velocidade (v $[m/s]$) e o comportamento oscilatório das plantas ao serem influenciadas por ele ($\theta = f(t)$). A Figura 33 ilustra a consideração feita para o ângulo θ.

Figura 33 – Oscilação causada pelo vento

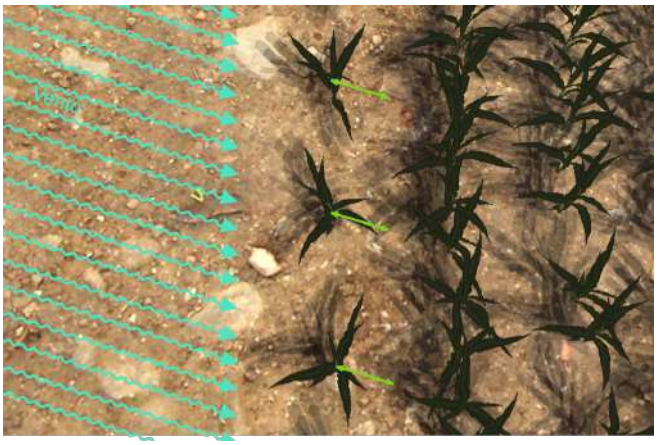


Fonte: Elaborado pelo autor

Ao sofrer influência, a planta oscilará no sentido de atuação do vento (por exemplo, conforme destacado em verde claro na Figura 34) e de acordo com uma função pré-definida. Para exemplificar, estamos considerando a função $\theta(t)$ presente na Equação 3.1.

$$\theta(t) = \frac{A}{2} \cdot sen(2 \cdot \pi \cdot f \cdot t - 0,5 \cdot \pi) + \frac{A}{2} \tag{3.1}$$

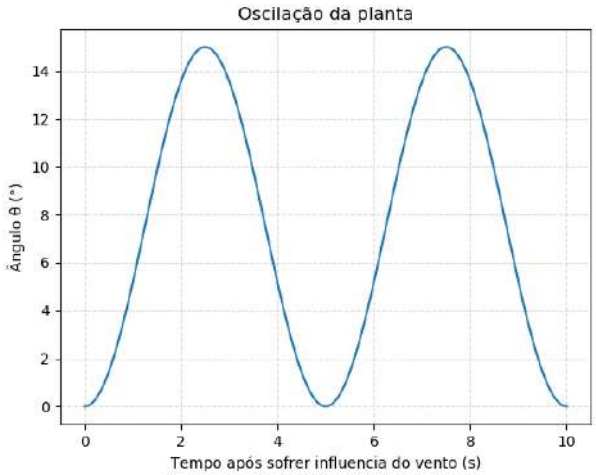
Figura 34 – Sentido da oscilação da planta



Fonte: Elaborado pelo autor

Com $A = 15°$ e $f = 0,2\ Hz$ (Figura 35). O tempo t é considerado como o tempo que se passou desde a interação com o vento, sendo $t = 0$ o momento de interação daquela planta em específico. Essa função foi escolhida aleatoriamente e pode ser facilmente alterada e customizada de acordo com o interesse do desenvolvedor.

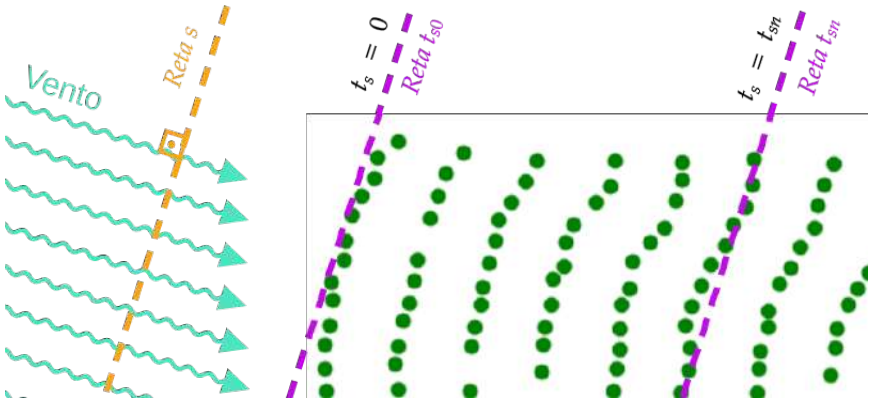
Figura 35 – Função de oscilação



Fonte: Elaborado pelo autor

Em uma plantação, algumas plantas interagem com o vento antes das demais. Por exemplo, se o vento atua da esquerda para direita, as plantas mais à esquerda interagem primeiro com o vento. A Figura 36 ilustra esse comportamento na simulação para o caso exemplificado (considere t_s como sendo o tempo da simulação): ao iniciar a simulação, em $t_s = 0$, as plantas pertencentes a reta t_{s0} sofrerão impacto do vento primeiro, posteriormente com o decorrer da tempo, em $t_s = t_{sn}$, as plantas influenciadas serão as pertencentes a reta t_{sn}.

Figura 36 – Atuação do vento por toda a plantação



Fonte: Elaborado pelo autor

A fim de simular esse comportamento, por meio do parâmetro α e das dimensões do mapa é possível estimar uma reta s, destacada em amarelo na Figura 36 e descrita pela Equação 3.2, que não cruza a plantação.

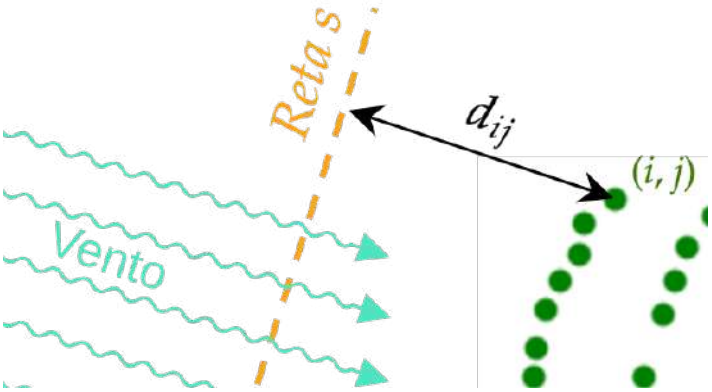
$$a \cdot x + b \cdot y + c = 0 \tag{3.2}$$

Calcula-se a distância d_{ij} (ilustrada na Figura 37 e equacionada pela Equação 3.3) entre cada uma das plantas e essa reta s. Por meio da velocidade do vento v é determinado o tempo em que aquela respectiva planta sofrerá impacto considerando que a origem do vento se dê na reta s.

Adotando que em $t_s = 0$ ocorra a oscilação da primeira planta, é calculado, por meio da Equação 3.4, o tempo de simulação t_{s-ij} que uma planta localizada na posição (i, j) do terreno irá começar a oscilar, sendo $min\left(\frac{d_{i'j'}}{v}\right)$ o menor tempo calculado por toda a plantação.

$$d_{ij} = \frac{|a \cdot i + b \cdot j + c|}{\sqrt{a^2 + b^2}} \tag{3.3}$$

Figura 37 – Interações da plantação com o vento

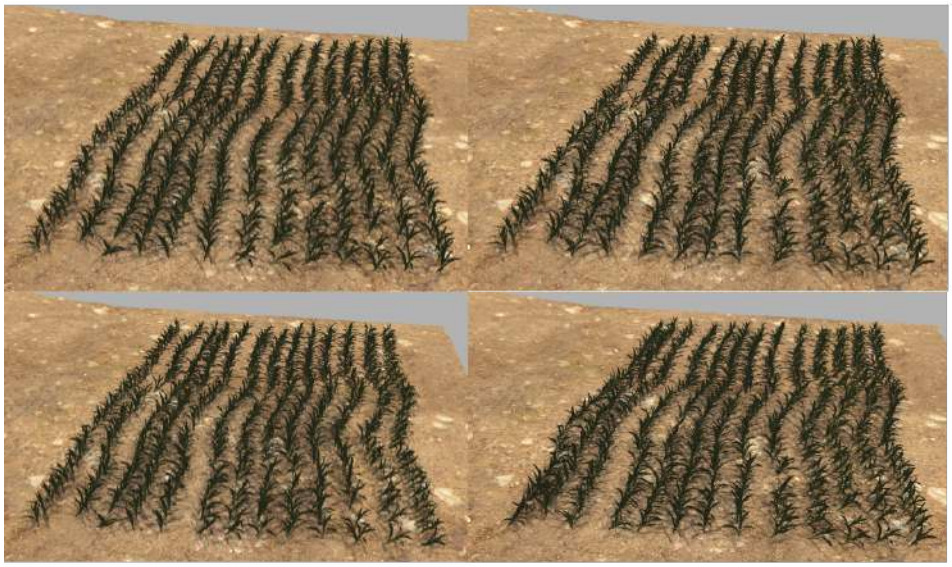


Fonte: Elaborado pelo autor

$$t_{s-ij} = \frac{d_{ij}}{v} - min\left(\frac{d_{i'j'}}{v}\right) \tag{3.4}$$

A Figura 38 ilustra um exemplo do resultado dessa abordagem na plantação durante a simulação. São considerados quatro momentos distintos com $\alpha = 0°$ (vento atua da esquerda para a direita), $v = 2\ m/s$, $f = 0,2\ Hz$ e $A = 45°$ para melhor visualização na imagem.

Figura 38 – Exemplo de plantação durante quatro momentos da simulação



Fonte: Elaborado pelo autor

Por fim, a respeito de como as informações apresentadas anteriormente são adicionadas no Gazebo, faz-se o uso de *waypoints* apresentados na seção 2.4. Como estamos

considerando movimentos em *loop* para cada planta, iniciados a partir do início da simulação, usamos apenas o primeiro período da função $\theta(t)$ para gerá-los. Cada ciclo de uma planta localizada na posição (i, j, z) do mapa será composta pelos seguintes *waypoints*:

- **1º *waypoint***: Em $t_l = t_{s-ij}$, a posição deve ser (i, j, z) e a rotação deve ser $(0, 0, yaw_{(i,j)})$
- **2º *waypoint***: Em $t_l = t_{s-ij} + \dfrac{1}{2 \cdot f}$, a posição deve ser (i, j, z) e a rotação deve ser $(f_{roll(i,j)}\left(t = \frac{1}{2 \cdot f}\right), f_{pitch(i,j)}\left(t = \frac{1}{2 \cdot f}\right), yaw_{(i,j)})$
- **3º *waypoint***: Em $t_l = t_{s-ij} + \dfrac{1}{f}$, a posição deve ser (i, j, z) e a rotação deve ser $(0, 0, yaw_{(i,j)})$

Sendo que é t_l o tempo do *waypoint* dentro do *loop* (a cada iteração esse valor é reiniciado), $yaw_{(i,j)}$ é o ângulo em que o modelo já foi rotacionado na geração da plantação (discutido na seção 2.3 e pode ser visualizado na Figura 34 com as três plantas em destaque apresentando rotações distintas) e com $f_{roll(i,j)}(t)$ e $f_{pitch(i,j)}(t)$ descritas pela Equação 3.5 e Equação 3.6, respectivamente.

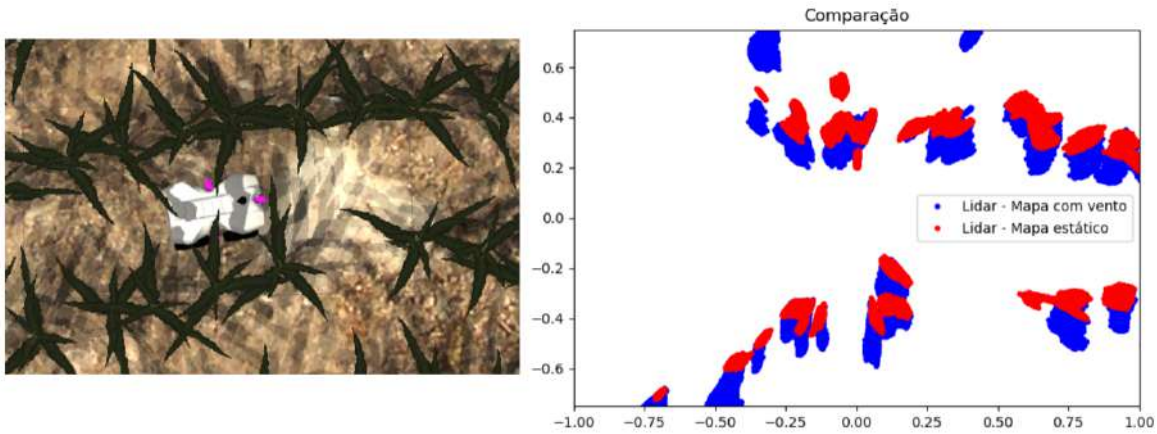
$$f_{roll(i,j)}(t) = \theta(t) \cdot (cos(\alpha) \cdot sen(yaw_{(i,j)}) + sen(\alpha) \cdot cos(yaw_{(i,j)})) \tag{3.5}$$

$$f_{pitch(i,j)}(t) = \theta(t) \cdot (cos(\alpha) \cdot cos(yaw_{(i,j)}) - sen(\alpha) \cdot sen(yaw_{(i,j)})) \tag{3.6}$$

3.3.2 Resultados obtidos

A Figura 39 apresenta um exemplo de comparação acerca do impacto da abordagem considerada nas leituras do sensor Lidar.

Figura 39 – Interferência no Lidar

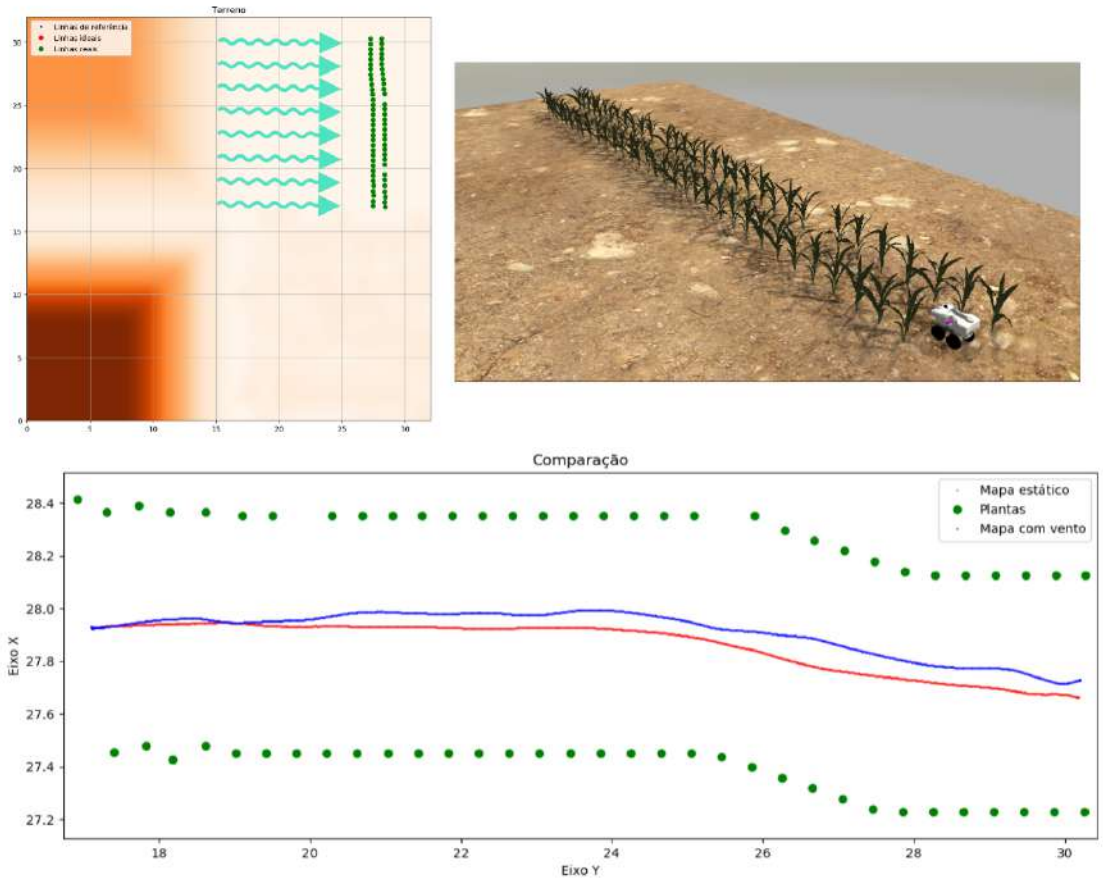


Fonte: Elaborado pelo autor

No exemplo dado, o robô TerraSentia permaneceu estático e os dados foram coletados através de 10 s de simulação. O vento atua de cima para baixo (utilizando como referência a imagem à esquerda), com $A = 15°$, $f = 0,2\ Hz$ e $v = 2\ m/s$. Como pode ser visualizado no gráfico à direita (com referência ao eixo do sensor), ao adicionar movimento nas plantas há a leitura de novos pontos e estes estão deslocados em relação aos dados do Lidar no mapa estático na direção e no sentido da atuação do vento.

A respeito do impacto do vento no funcionamento do algoritmo de navegação (seção 3.1), a Figura 40 ilustra um dos testes feitos. Foi considerada uma faixa de plantação aproximadamente reta (linhas a 90 cm de distância), com as plantas idealmente em suas posições de plantio (plantas posicionadas a cada 40 cm). O vento atua da esquerda para direita ($\alpha = 0°$), com $A = 15°$, $f = 0,2\ Hz$ e $v = 2\ m/s$. Foi utilizada a parte totalmente plana do mapa, e realizou-se a simulação com mapa estático e a simulação com o vento. Durante as simulações foram coletadas as trajetórias do robô.

Figura 40 – Impacto do vento na trajetória



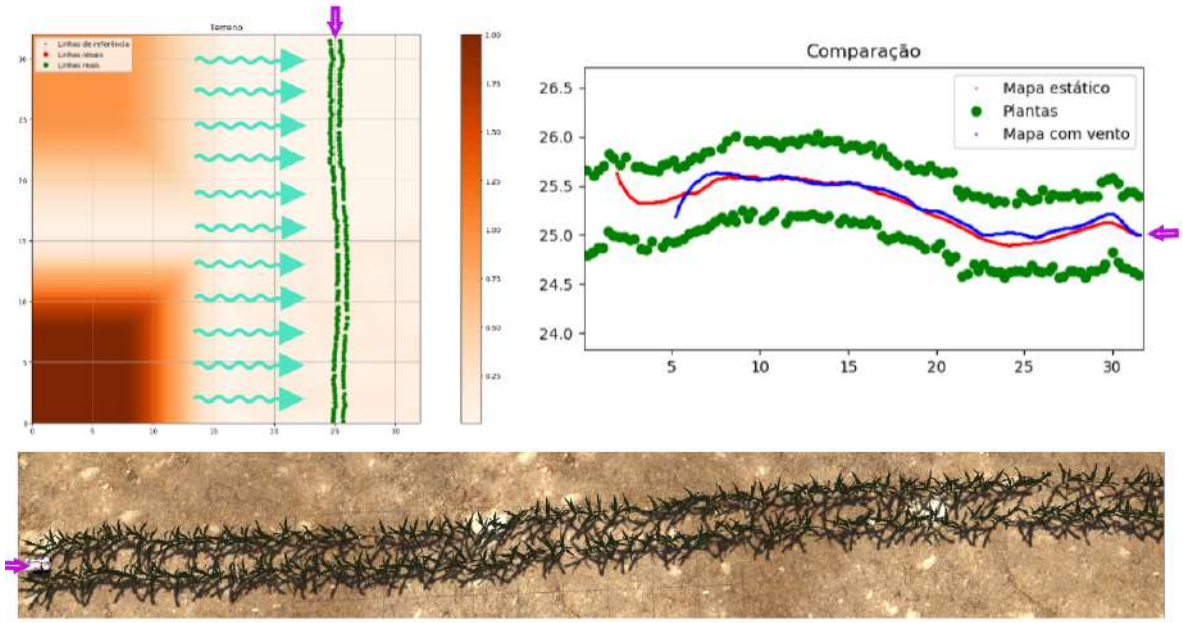
Fonte: Elaborado pelo autor

Observa-se que a trajetória do TerraSentia ao animar a plantação (linha azul), comparado com o mapa estático (linha vermelha), oscila mais. Como o vento atua na

direção perpendicular ao trajeto do robô e de forma oscilatória, ele interfere na detecção da localização das linhas da plantação pelo algoritmo ao afetar a leitura do Lidar conforme Figura 39. Dessa forma, a trajetória tende a variar mais dentro da faixa.

A Figura 41 exibe um outro teste feito. Neste houve apenas alteração na plantação: estamos considerando uma faixa maior, mais irregular, com plantas a cada 30 *cm* e linhas distantes a 76 *cm*, há ruídos na posição das plantas, e o mapa abrange a área plana e a parte irregular do terreno. As setas em roxo estão indicando a referência em relação ao mapa de cada uma das sub-imagens.

Figura 41 – Impacto em uma plantação e terrenos irregulares

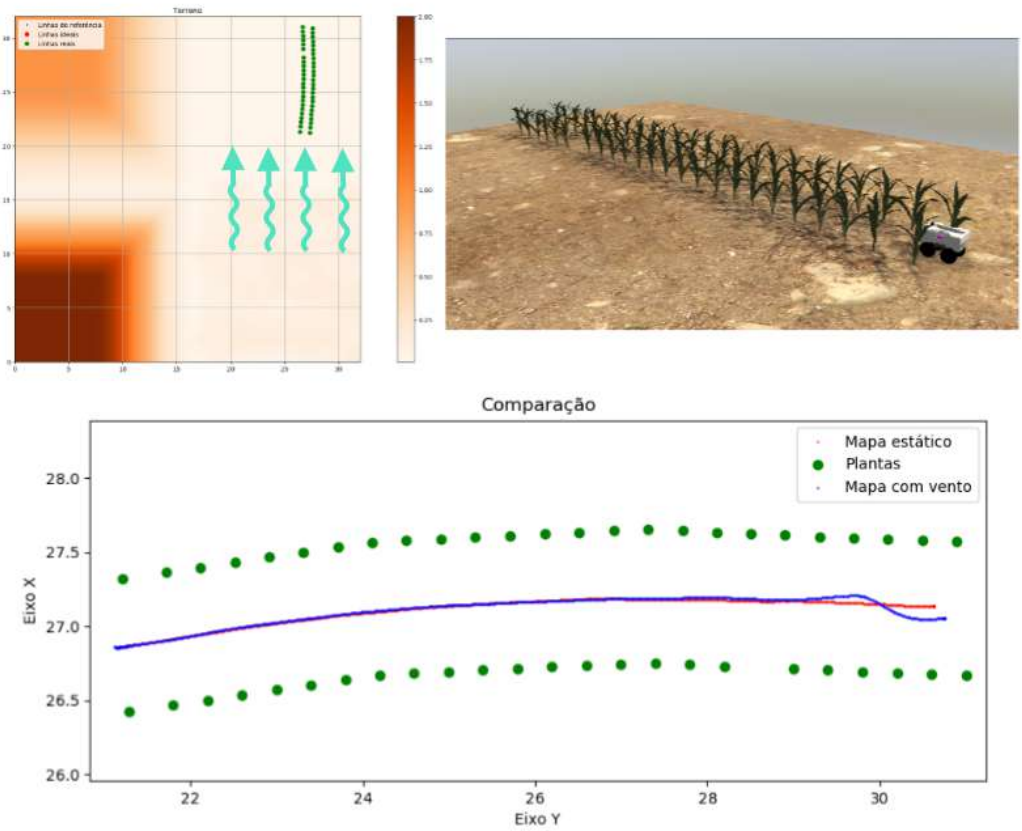


Fonte: Elaborado pelo autor

De forma similar ao caso anterior, a trajetória com vento oscila mais dentro da faixa. Para este caso, tanto na simulação estática quanto na simulação com vento, o robô perdeu o controle na parte irregular do mapa e se chocou com a plantação.

Por fim, a Figura 42 ilustra um teste feito com o vento paralelo às linhas da plantação ($\alpha = 270°$). Os parâmetros do vento são os mesmos dos casos anteriores. As linhas estão distantes em 90 *cm* e as plantas estão posicionadas a cada 40 *cm*. Percebe-se que a trajetória do robô foi muito próxima para ambas as abordagens, não houve uma oscilação tão visível quando dos casos anteriores. Como a oscilação das plantas ocorre de forma paralela à trajetória do TerraSentia, não há tanta alteração na detecção das linhas laterais pelo algoritmo.

Figura 42 – Trajetória paralela à direção do vento



Fonte: Elaborado pelo autor

Em relação ao impacto das animações no desempenho da simulação, foram feitos testes nos mapas da Figura 43-A e da Figura 43-B. No primeiro, um terreno plano, e no segundo, um terreno irregular, foram criadas plantações de milho com linhas distantes em 0.9 m e plantas posicionadas a cada 0.3 m. O vento atua com ângulo $\alpha = 45°$ e os outros parâmetros são os mesmos dos testes anteriores.

O TerraSentia foi colocado entre cada uma das linhas adjacentes de cada plantação e o algoritmo de navegação foi executado considerando tanto a plantação estática quanto a plantação com vento. Dessa forma, para cada mapa foram feitas 22 execuções do algoritmo (11 considerando os modelos estáticos e 11 para os modelos animados).

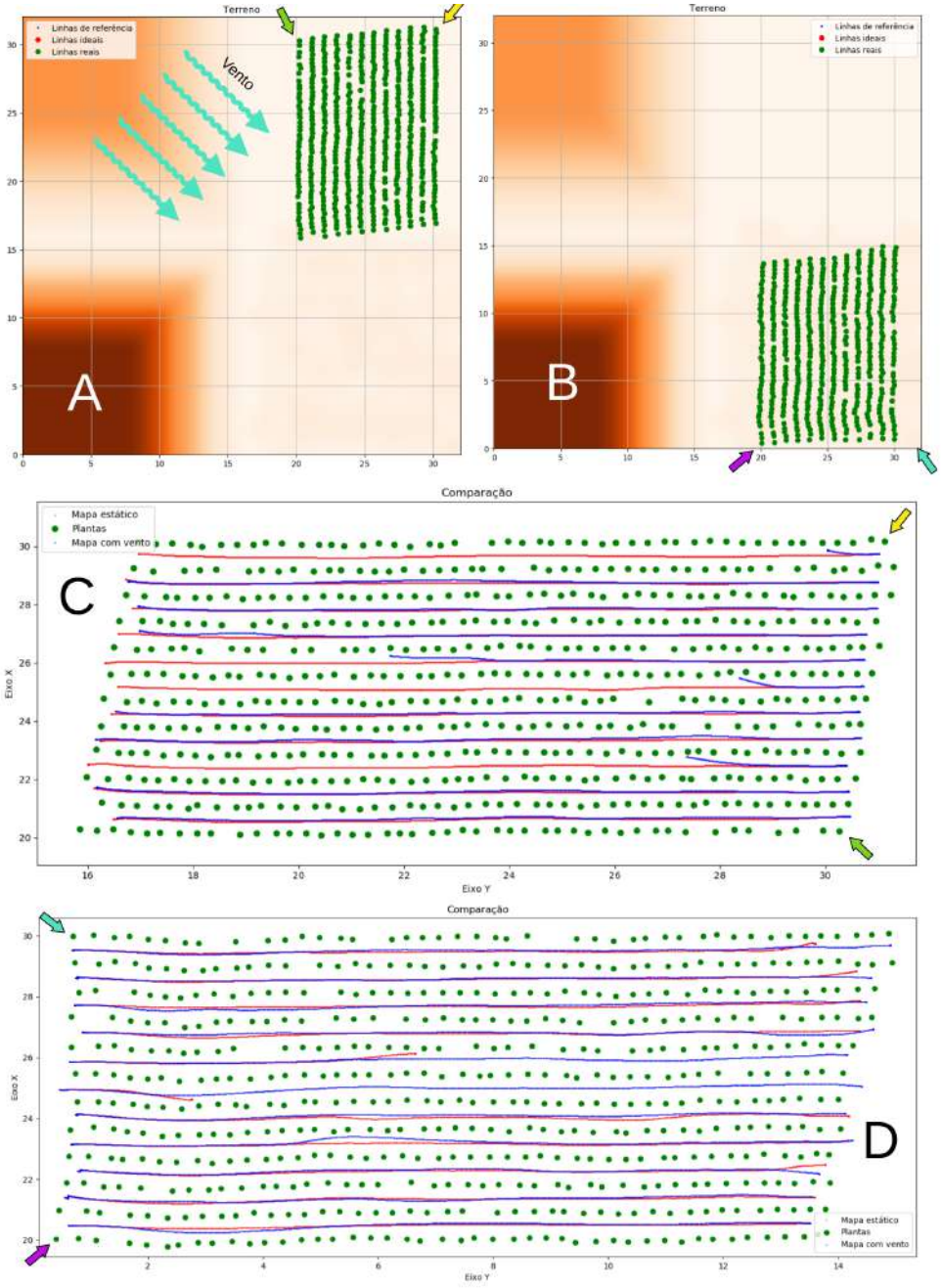
Durante o trajeto feito pelo robô foram coletadas as métricas de desempenho da simulação. A Tabela 7 apresenta os valores coletados durante o teste, sendo que RF_m é a média dos valores mínimos do *Real-time-factor* de cada iteração, RAM_m é a média de memória utilizada e CPU_m é a média do uso da CPU. Estas duas últimas estão divididas entre os dois processos do Gazebo: gzserver e gzclient.

Tabela 7 – Métricas de desempenho ao adicionar animação no mapa

Mapa	Considerações	RF_m	RAM_m (gzserver-gzclient)	CPU_m (gzserver-gzclient)
Figura 43-A	Sem vento	1,23	$11,5\% - 5\%$	$14,5\% - 8,9\%$
	Com vento	1,07	$15\% - 5,5\%$	$19\% - 9,3\%$
Figura 43-B	Sem vento	0,64	$10,6\% - 4,8\%$	$12\% - 9\%$
	Com vento	0,51	$14,4\% - 5,25\%$	$16,5\% - 9\%$

Fonte: Elaborado pelo autor

Figura 43 – Testes do impacto da animação no desempenho da simulação



Fonte: Elaborado pelo autor

Nota-se que ao adicionar animação, o valor do RF_m é reduzido, e o valor de RAM_m e CPU_m são aumentados, ou seja, a simulação se torna mais complexa.

Uma curiosidades a respeito dos testes feitos é que no mapa da Figura 43-A, com a plantação estática, o TerraSentia conseguiu percorrer com sucesso as 11 faixas, mas ao adicionar o vento, em 4 delas ele perdeu o controle e colidiu com a plantação. A Figura 43-C ilustra o percurso destes testes. Já para o mapa da Figura 43-B, com o vento, ele foi capaz de percorrer completamente as 11 faixas, mas sem, ele perdeu o controle em 3 delas. Estes casos podem ser visualizados na Figura 43-D. As setas coloridas indicam a referência de posicionamento entre os gráficos e os mapas.

4 CONCLUSÃO

É possível concluir que as abordagens complementares ao trabalho de Hunter Young (2019) consideradas nesta monografia permitem ampliar o nível de customização do ambiente de simulação para robôs agrícolas. Com a interface gráfica e com a integração da plantação com a elevação do terreno, o ambiente simulado pode passar a ser criado e utilizado, de forma rápida e prática, para casos específicos de testes. Ao considerar a opção de adicionar vento à plantação, uma nova perspectiva para a realização de testes é incorporada aos desenvolvedores.

Além disso, as adições feitas tornaram as simulações um pouco mais complexas. Para as configurações do computador utilizado nos testes, elas continuaram se mostrando viáveis.

A respeito do algoritmo de navegação presente Velasquez *et al.* (2021), foi possível integrá-lo e testá-lo nos ambientes criados. Por meio dos testes feitos foi observado que o seu funcionamento é impactado por parâmetros como o vento e como o solo. Como foram realizados poucos experimentos e o foco do trabalho não foi a compreensão detalhada e testagem do algoritmo, não é possível obter resultados conclusivos acerca do assunto.

4.1 Trabalhos futuros

A seguir estão descritas sugestões de trabalhos futuros. Eles levam em conta pontos que foram desconsiderados neste trabalho e ideias que permitam complementar o que foi apresentado.

- **Deixar mais real a animação das plantas ao serem influenciadas pelo vento:** Conforme discutido na subseção 3.3.1, a abordagem considerada para animar as plantas pode ser melhorada. Para isso, torna-se necessário o estudo de *softwares* de animação 3D a fim de deixar a movimentação das folhas e do caule o mais natural possível.

- **Levantar estudos e analisar a viabilidade da aplicação de equacionamentos mais precisos na oscilação das plantas e no comportamento do vento:** Para simular o comportamento oscilatório das plantas, o *software* desenvolvido permite o uso de qualquer equacionamento, no caso, foi utilizada a Equação 3.1 para a realização de testes. Além disso, o vento foi descrito como atuando periodicamente e em linha reta pela plantação. Dessa forma, faz sentido verificar se há equações na literatura que descrevem melhor esses comportamentos e analisar se estas têm impacto relevante na simulação.

- **Realizar uma bateria de testes mais completa, entender detalhadamente o algoritmo de navegação e compreender os impactos do ambiente de simulação em seu funcionamento:** Conforme discutido anteriormente, o algoritmo de navegação de Velasquez *et al.* (2021) teve o seu funcionamento impactado por parâmetros da simulação, como por exemplo, terreno irregular e vento. Com isso, é interessante entender se esses erros são pertinentes e se caso forem, os motivos pelos quais o robô perde o controle e se choca contra a plantação.
- **Entender o novo Ignition Gazebo e realizar a migração do código:** Conforme apresentado na subseção 2.2.3 o Gazebo será descontinuado em 2025. Dessa forma, é apropriado compreender o funcionamento da nova plataforma (Ignition Gazebo) e realizar a refatoração do que foi desenvolvido em Hunter Young (2019) e complementado nesta monografia.

REFERÊNCIAS

9 Robots That Are Invading The Agriculture Industry. 2021. Disponível em: <https://interestingengineering.com/9-robots-that-are-invading-the-agriculture-industry>. Acesso em: 08 nov. 2021.

ACTOR. 2021. Disponível em: http://gazebosim.org/tutorials?tut=actor&cat=build_robot. Acesso em: 08 nov. 2021.

AFZAL, A. *et al.* Simulation for robotics test automation: Developer perspectives. *In*: **2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)**. [*S.l.: s.n.*], 2021. p. 263–274.

ARDEE: A general agricultural robotic development and evaluation environment. 2019. Disponível em: http://hdl.handle.net/2142/104729. Acesso em: 08 nov. 2021.

BECHAR, A.; VIGNEAULT, C. Agricultural robots for field operations: Concepts and components. **Biosystems Engineering**, v. 149, p. 94–111, 2016. ISSN 1537-5110. Disponível em: https://www.sciencedirect.com/science/article/pii/S1537511015301914.

CHOI, H. *et al.* On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 118, n. 1, 2021. ISSN 0027-8424. Disponível em: https://www.pnas.org/content/118/1/e1907856118.

COPPELIASIM. 2021. Disponível em: https://www.coppeliarobotics.com/coppeliaSim. Acesso em: 08 nov. 2021.

DASLAB. 2021. Disponível em: http://daslab.illinois.edu/index.html. Acesso em: 08 nov. 2021.

GILLIES, S. *et al.* **Shapely: manipulation and analysis of geometric objects**. 2007–. Disponível em: https://github.com/Toblerity/Shapely.

GIUBILATO, R. *et al.* Simulation framework for mobile robots in planetary-like environments. *In*: **2020 IEEE 7th International Workshop on Metrology for AeroSpace (MetroAeroSpace)**. [*S.l.: s.n.*], 2020. p. 594–599.

HE, L. *et al.* Method to integrate human simulation into gazebo for human-robot collaboration. **IOP Conference Series: Materials Science and Engineering**, IOP Publishing, v. 825, p. 012006, may 2020. Disponível em: https://doi.org/10.1088/1757-899x/825/1/012006.

HIGUTI, V. A. H. *et al.* Under canopy light detection and ranging-based autonomous navigation. **Journal of Field Robotics**, v. 36, n. 3, p. 547–567, 2019. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21852.

HUNTER, J. D. Matplotlib: A 2d graphics environment. **Computing in Science & Engineering**, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

HUSSEIN, A.; GARCíA, F.; OLAVERRI-MONREAL, C. Ros and unity based framework for intelligent vehicles control and simulation. *In*: **2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)**. [*S.l.: s.n.*], 2018. p. 1–6.

IGNITION. 2021. Disponível em: https://ignitionrobotics.org/about. Acesso em: 08 nov. 2021.

IQBAL, J. *et al.* Simulation of an autonomous mobile robot for lidar-based in-field phenotyping and navigation. **Robotics**, v. 9, n. 2, 2020. ISSN 2218-6581. Disponível em: https://www.mdpi.com/2218-6581/9/2/46.

IVALDI, S.; PADOIS, V.; NORI, F. Tools for dynamics simulation of robots: a survey based on user feedback. **arXiv preprint arXiv:1402.7050**, 2014.

MELO, M. Santos Pessoa de *et al.* Analysis and comparison of robotics 3d simulators. *In*: **2019 21st Symposium on Virtual and Augmented Reality (SVR)**. [*S.l.: s.n.*], 2019. p. 242–251.

NOGUEIRA, L. Comparative analysis between gazebo and v-rep robotic simulators. **Seminario Interno de Cognicao Artificial-SICA**, v. 2014, n. 5, 2014.

PITONAKOVA, L. *et al.* Feature and performance comparison of the v-rep, gazebo and argos robot simulators. *In*: . [*S.l.: s.n.*], 2018. ISBN 978-3-319-96727-1.

PYTHON. 2021. Disponível em: https://www.python.org/. Acesso em: 08 nov. 2021.

ROBOTICS in Agriculture: Types and Applications. 2017. Disponível em: https://www.automate.org/blogs/robotics-in-agriculture-types-and-applications. Acesso em: 08 nov. 2021.

SAGLAM, A.; PAPELIS, Y. Scalability of sensor simulation in ros-gazebo platform with and without using gpu. *In*: **2020 Spring Simulation Conference (SpringSim)**. [*S.l.: s.n.*], 2020. p. 1–11.

TERRASENTIA. 2021. Disponível em: https://www.earthsense.co/. Acesso em: 08 nov. 2021.

UNSPLASH. 2021. Disponível em: https://unsplash.com/. Acesso em: 08 nov. 2021.

URDF format. 2021. Disponível em: http://wiki.ros.org/urdf. Acesso em: 08 nov. 2021.

VELASQUEZ, A. E. B. *et al.* Multi-sensor fusion based robust row following for compact agricultural robots. **arXiv preprint arXiv:2106.15029**, 2021.

VIRTANEN, P. *et al.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. **Nature Methods**, v. 17, p. 261–272, 2020.

WIND On Corn. 2021. Disponível em: https://thomascountyag.com/2015/04/21/wind-on-corn/. Acesso em: 08 nov. 2021.

XACRO. 2021. Disponível em: http://wiki.ros.org/xacro. Acesso em: 08 nov. 2021.

APÊNDICES

APÊNDICE A – EXEMPLO DE ACTOR NO GAZEBO

A Figura 44 ilustra um exemplo da função *Actor* do Gazebo. Nela um cubo percorre o mapa em movimentos cíclicos (Figura 14).

Figura 44 – Movimentação em *loop* de um cubo

```
<?xml version="1.0" ?>
<sdf version="1.6">
   <world name="default">
      <include>
         <uri>model://ground_plane</uri>
      </include>
      <include>
         <uri>model://sun</uri>
      </include>
      <actor name="animated_box">
        <link name="box_link">
          <visual name="visual">
            <geometry>
              <box>
                <size>.2 .2 .2</size>
              </box>
            </geometry>
          </visual>
        </link>
        <script>
          <loop>true</loop>
          <auto_start>true</auto_start>
          <trajectory id="0" type="square">
             <waypoint>
                <time>0.0</time>
                <pose>-1 -1 1 0 0 0</pose>
             </waypoint>
             <waypoint>
                <time>1.0</time>
                <pose>-1 1 1 0 0 0</pose>
             </waypoint>
             <waypoint>
                <time>2.0</time>
                <pose>1 1 1 0 0 0</pose>
             </waypoint>
             <waypoint>
                <time>3.0</time>
                <pose>1 -1 1 0 0 0</pose>
             </waypoint>
             <waypoint>
                <time>4.0</time>
                <pose>-1 -1 1 0 0 0</pose>
             </waypoint>
          </trajectory>
        </script>
      </actor>
   </world>
</sdf>
```

Fonte: Open Source Robotics Foundation (2021)

Dentro do *script*, a *tag* **<loop>** é responsável por controlar se o movimento pela trajetória ocorrerá em *loop* ou não, a *tag* **<auto-start>** é incumbida de controlar se o *script* se iniciará com o início da simulação no Gazebo e a *tag* **<trajectory>** apresenta a lista de posições e tempos da animação.

A Figura 45 apresenta o código de um único modelo de milho animado (animação em *loop* e que se inicia junto com a simulação). Este deve ser repetido para cada uma das plantas presentes na plantação. O calculo dos *waypoints* é feito de acordo com a seção 3.3.1 e *model://corn-variant-1* é a localização do arquivo que contem o modelo de milho.

Figura 45 – *Actor* de uma planta

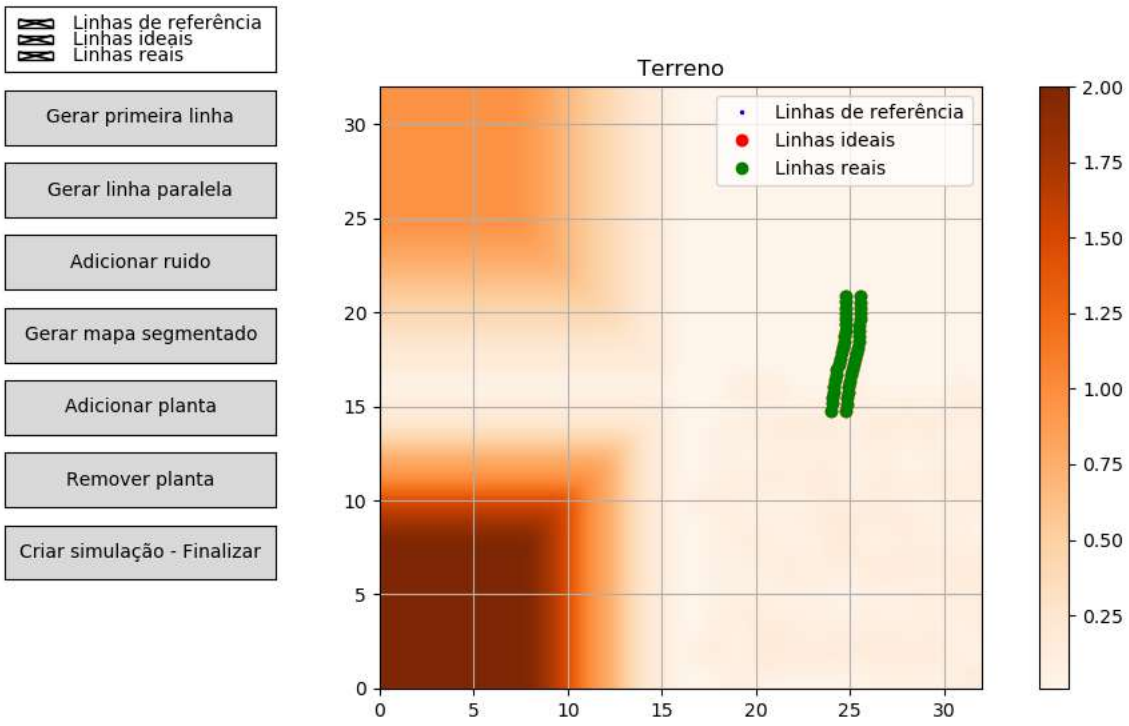
```
<actor name="model1">
        <include>
            <uri>model://corn_variant_1</uri>
            <name>1</name>
        </include>
        <script>
            <loop>true</loop>
            <auto_start>true</auto_start>
            <trajectory id="1" type="corn">
                <waypoint>
                    <time>0.2534924179476512</time>
                    <pose>11.364821354383054 5.720113211466554 0.00784313725490196 0.0 0.0 2.4085543677521746</pose>
                </waypoint>
                <waypoint>
                    <time>2.753492417947651</time>
                    <pose>11.364821354383054 5.720113211466554 0.00784313725490196 0.1945548603560874 0.17517798310242277 2.5152872291607826</pose>
                </waypoint>
                <waypoint>
                    <time>5.253492417947651</time>
                    <pose>11.364821354383054 5.720113211466554 0.00784313725490196 0.0 0.0 2.4085543677521746</pose>
                </waypoint>
            </trajectory>
        </script>
    </actor>
```

Fonte: Elaborado pelo autor

APÊNDICE B – INTERFACE GRÁFICA PARA GERAR AS PLANTAÇÕES

A Figura 46 apresenta a interface gráfica completa da ferramenta desenvolvida para gerar as plantações. Ela foi desenvolvida por meio da biblioteca Matplotlib (HUNTER, 2007). À esquerda estão os botões que permitem gerar as ações para criar a plantação e à direita o terreno com área que interage com o *mouse*. Os parâmetros da plantação como, por exemplo, a distância entre as linhas e o modelo da planta devem ser adicionados dentro do próprio código.

Figura 46 – Interface gráfica

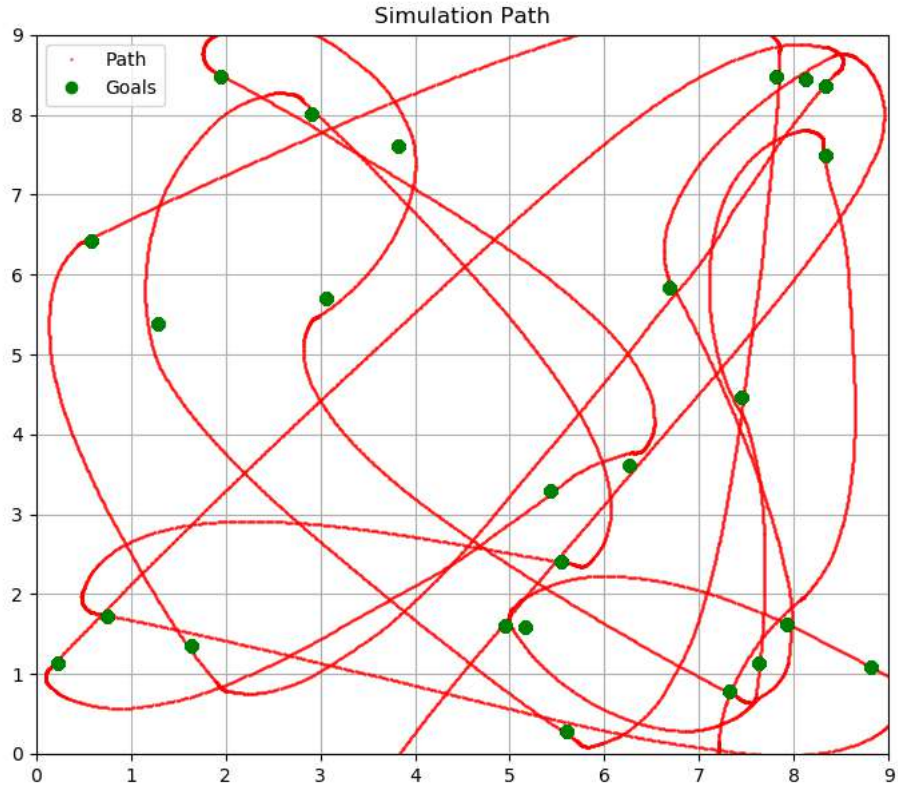


Fonte: Elaborado pelo autor

APÊNDICE C – ALGORITMO DE EXPLORAÇÃO ALEATÓRIA

O algoritmo de exploração aleatória [1] foi implementado em Python. Consiste em gerar ciclicamente um ponto aleatório pelo mapa e fazer o robô se aproximar dele. Quando estiver próximo o suficiente do ponto, um novo ponto objetivo é gerado. A Figura 47 exibe uma das execuções do algoritmo em um mapa plano de 9 $m \times 9$ m: em verde estão as metas para o algoritmo e em vermelho o percurso feito pelo TerraSentia.

Figura 47 – Exploração aleatória - Objetivos e percurso



Fonte: Elaborado pelo autor

[1] Implementação sugerida e posteriormente adaptada por Mateus Valverde Gasparino (DASLAB TEAM, 2021).