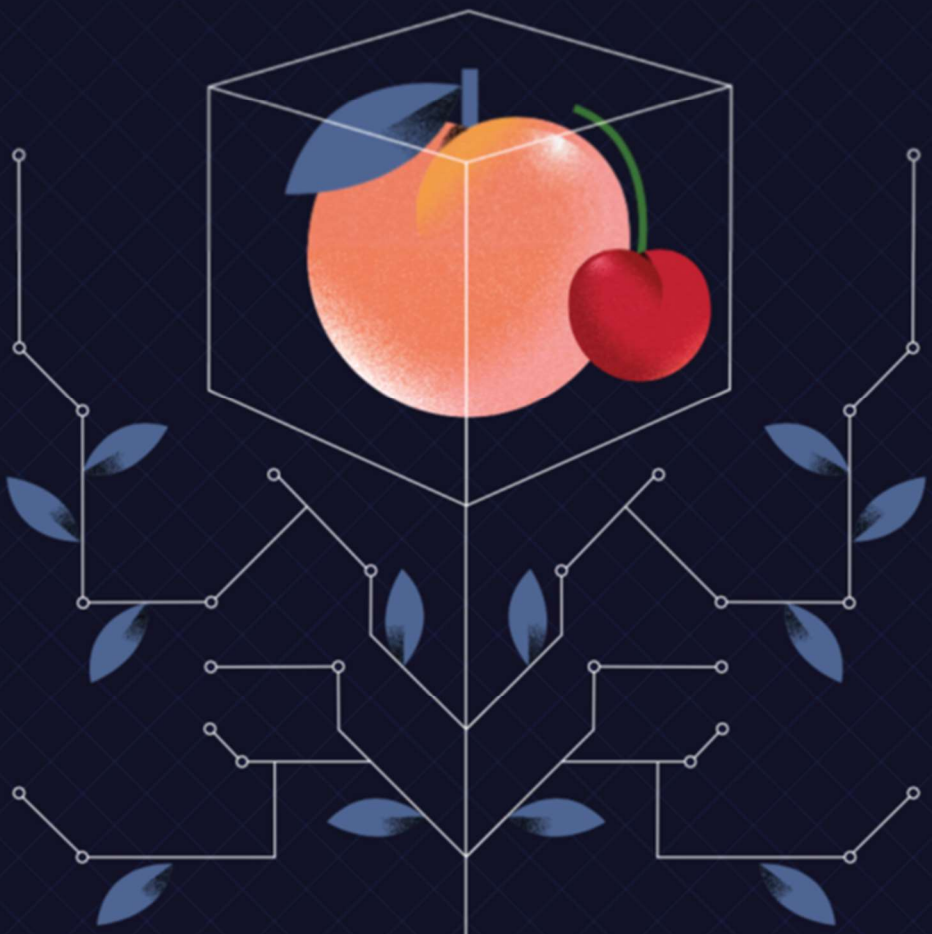
PrunusBOT

Aplicação da robótica à produção de pêssego e cereja

COTHN
CENTRO OPERATIVO E TECNOLÓGICO HORTOFRUTÍCOLA NACIONAL



PrunusBOT

PrunusBOT

Aplicação da robótica à produção de pêssego e cereja

Maria Paula Simões

(COORDENAÇÃO)

VOLUME I

CENTRO OPERATIVO E TECNOLÓGICO HORTOFRUTÍCOLA NACIONAL –
CENTRO DE COMPETÊNCIAS

Ficha Técnica

Título: PrunusBOT – Aplicação da robótica à produção de pêssego e cereja

Coordenação: Maria Paula Simões

Editor: COTHN-CC – Centro Operativo e Tecnológico Hortofrutícola Nacional – Centro de Competências

Autores e copyright:

Abel Veloso
Anabela Barateiro
André Veiros
António Ramos
Cristina Canavarro
Cristina Ramos
Dora Ferreira
Eduardo Assunção
Francisco Vieira
Hugo Fonseca
Hugo Proença
João Cunha
José Pedro Simões
Maria Paula Simões
Paulo Silvino
Pedro Dinho Silva
Pedro Dinis Gaspar
Preciosa Fragoso,
Ricardo Mesquita
Sandra Lopes

Revisão: Maria Paula Simões

Design Editorial: SUPER Brand Consultants

Tiragem: 300 exemplares

Impressão e Acabamento: Empresa Diário do Porto, Lda.

Data de Impressão: Abril de 2022

Depósito Legal: 497599/22

ISBN: 978-972-8785-21-5

Capítulo 6

Sistema de navegação autónoma baseado em visão computacional por deteção de troncos de árvores - Aplicação a pomares de pessegueiros

José Pedro Gouveia Pires Simões[1] e Pedro Dinis Gaspar[1,2]

[1] Universidade da Beira Interior | Departamento de Engenharia Eletromecânica
[2] Center for Mechanical and Aerospace Science and Technologies (C-MAST)

6.1 Introdução

O rover robótico (R2A2) faz uso de dois sistemas de controlo, regulação e comando da locomoção. Um é baseado em sistema de posicionamento global (GPS – Global Positioning System), permitindo que o trajeto autónomo a realizar pelo rover robótico possa ser pré-programado (Menezes et al., 2022). O outro sistema de controlo, regulação e comando da locomoção autónoma fará uso de uma câmara RGB (Red-Green-Blue) disposta sobre o rover e de um algoritmo de deteção de troncos (Simões et al., 2022). Este sistema será prioritário para evitar que a plataforma robótica choque com as árvores e para garantir que esta se desloque numa linha reta sem grandes desvios, com uma margem de erro reduzida. Este capítulo tem como objetivo caracterizar este sistema de locomoção.

O algoritmo tem como entrada imagens de vídeo a tempo real captado por uma câmara montada na plataforma e, através de técnicas de Inteligência Artificial por Aprendizagem Profunda (Deep Learning), mais especificamente por Redes Neuronais Convolucionais (Convolutional Neural Networks – CNN) (Sandler et al., 2018), fará a deteção dos troncos das árvores. A estratégia de orientação é a criação de uma função linear, utilizando dois pontos dos limites de deteção de dois troncos, formando uma linha que acompanha aproximadamente a orientação da fila de árvores do pomar. Ordens serão enviadas ao robô para virar para a esquerda, direita ou prosseguir a marcha caso a caixa que delimita a deteção do tronco se encontra muito desviada da linha ou, no caso de detetar dois troncos, o declive da função linear atingir valores que indiquem que o robô segue uma trajetória irregular e potencial de choque.

Uma deslocação autónoma permite uma melhor gestão de tempo, pois não há necessidade de haver um operário a controlar a navegação do robô, logo o robô efetua uma tarefa demorada e mundana sem interrupções a qualquer hora do dia ou até mesmo da noite (Barawid Jr et al., 2007). Estas novas soluções tecnológicas aplicadas à agricultura conduzem ao aumento da eficiência dos processos e consequentemente ao aumento da produtividade, redução do desperdício, e maior resiliência aos fatores dinâmicos inerentes à atividade agrícola, como sejam as condições ambientais adversas, pragas e pestes.

6.2 Materiais e métodos

6.2.1 Software

O primeiro passo deste projeto consiste no desenvolvimento do algoritmo. A programação do algoritmo de deteção foi fundada em ambos os sistemas operativos Windows 10 e Linux na linguagem Python 3.9.5.

A biblioteca recorrida, suportada pela linguagem Python, adequada à implementação de CNN's foi Tensorflow (Dillon et al., 2017).

Tensorflow é o framework completo da Google para desenvolvimento de aplicações que executam tarefas de Machine Learning. Estas aplicações descobrem padrões num vasto número de dados injetados no algoritmo lidando

com incertezas e probabilidades. Segue-se, nos próximos parágrafos, uma explicação do funcionamento, potencialidade e alguns elementos importantes presentes no framework. O Object Detection API (disponível no GitHub oficial de Tensorflow), é um framework desenvolvido com base em Tensorflow que auxilia a construção e o treino de um modelo de deteção de objetos em imagens (Hongkun Yu et al., 2020).

O sistema de deteção de objetos selecionado neste projeto foi, então, o SSD (Single Shot Detector). Este sistema, criado por Liu et al. (2016), é mais rápido que outros detetores do estado-da-arte, tais como o Faster R-CNN, sem sacrificar o desempenho.

Esta vantagem faz com que o SSD seja o detetor mais indicado para situações onde se pretende implementar um modelo de deteção de objetos num dispositivo móvel incapaz de acarretar mais potência. Na Figura 6.1 encontra-se uma representação gráfica da arquitetura do modelo de deteção SSD.

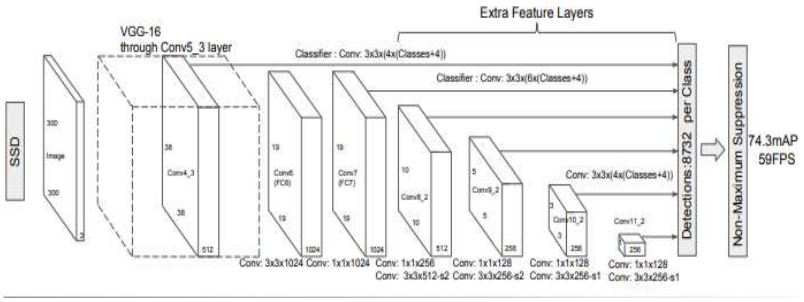


Figura 6.1 – Modelo de deteção SSD. (Liu et al., 2016)

6.2.2 Hardware

Dado que o objetivo deste projeto é o estabelecimento de um sistema que permite que a plataforma robótica se desloque autonomamente, para conciliar o algoritmo com os controladores das rodas, o algoritmo de deteção dos troncos das árvores e a câmara que capta imagem em tempo real, é essencial um

controlador com extrema mobilidade e com capacidade computacional para acarretar tantas competências.

O Raspberry Pi 4 (Foundation Raspberry_Pi, 2021) foi considerado a melhor solução para este problema. Um microcomputador de reduzidas dimensões, ideal para aplicações móveis e remotas, com uma capacidade de processamento computacional e de imagem (Lowe, 2017).

A câmara utilizada para a captação de imagem RGB do robô é a câmara módulo do Raspberry Pi versão 2 (Raspberry Pi, n.d.). Tem um sensor Sony IMX219 de 8 MP. Suporta resoluções como 1080 com 30 FPS e 720p com 60 FPS.

6.2.3 Construção da base de dados

A base de dados é composta por fotografias e vídeos obtidos num pomar de pessegueiros, localizado em Orjais, Covilhã. Estes vídeos e fotografias foram tirados na perspetiva visual que a plataforma robótica irá ter quando praticar a sua locomoção correta, ou seja, encostada o mais à esquerda possível, e em locais aleatórios do pomar. Dessas fotografias, foram aproveitadas as que tinham uma imagem mais nítida dos troncos das árvores. No total, a base de dados estava com 89 imagens e suas anotações.

A Figura 6.2 mostra três amostras das fotografias obtidas no pomar que têm conceção evidente dos troncos das árvores. Todas as imagens foram recortadas e redimensionadas para uma dimensão de 640x640 para estarem compatíveis com a configuração de input do modelo de deteção SSD que foi utilizado.

Recorrendo ao labelImg (Tzutalin, 2015) – uma ferramenta de anotação de imagens open-source – anotaram-se os troncos das imagens, criando uma área retangular com quatro coordenadas à volta dos troncos, e atribuíam-se os rótulos da classe (tronco) às caixas delimitadoras, ficando, estes dados, registados em ficheiros *.xml. Uma amostra do processo da anotação está representada na Figura 6.3.

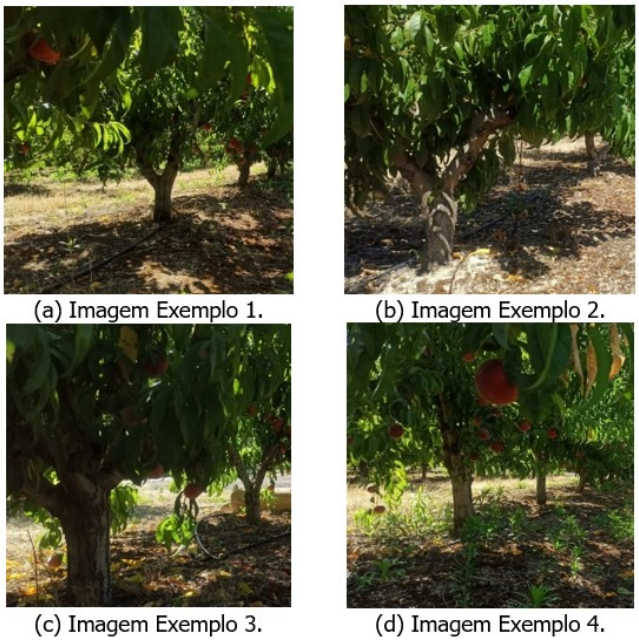


Figura 6.2 - Exemplo de quatro fotografias consideradas adequadas para o treino do modelo de deteção, com a perspetiva pretendida, cortadas e redimensionadas para o treino.

Figura 6.3 – Processo de anotação da classe dos troncos, nas imagens.

6.2.4 Estratégia de orientação

Com o modelo já treinado e a detetar os troncos corretamente, é necessário formar uma estratégia para indicar à plataforma robótica quando, e em que condições, terá que ajustar a sua trajetória. A câmara é montada na frente do robô e a visão que dispõe tem um panorama mais próximo da fila de árvores esquerda que da direita, como mostra a Figura 6.4.

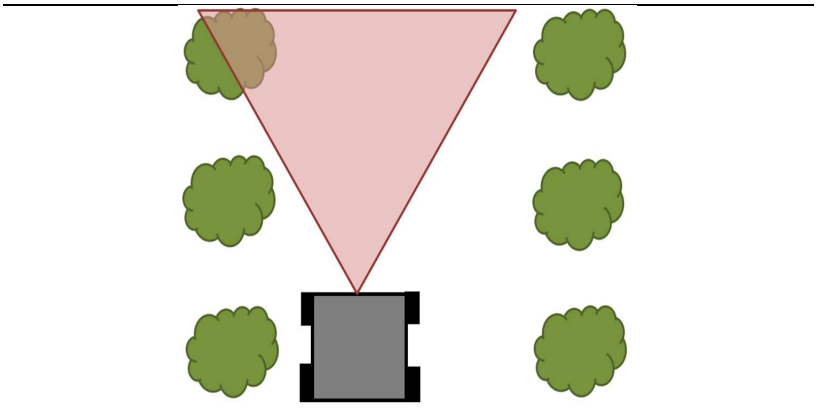


Figura 6.4 – Visão panorâmica da plataforma robótica (perspetiva topográfica).

Observando a perspetiva da plataforma robótica, nota-se uma particularidade: as filas das árvores, tanto da direita como da esquerda, formam uma linha que converge num ponto comum, sendo este o final do corredor do pomar, como está representado na Figura 6.5. Optou-se, então, que, a tendência desta linha formada no lado esquerdo ter uma trajetória correta sem obstruções para o ponto de chegada da plataforma robótica seria o melhor método de orientação. Para abordar a técnica de orientação, recorreu-se à obtenção de coordenadas de dois pontos de duas caixas delimitadoras de troncos detetados, e, a partir destes, forma-se uma função linear que atualiza a cada quadro de vídeo que contém uma deteção da mesma configuração.

Uma função linear é, tipicamente, representada pela Equação (1), onde m é o declive da reta e b a ordenada na origem.

$$y = mx + b \tag{1}$$

Cada caixa delimitadora possui quatro valores: uma abcissa mínima (xmin) e máxima (xmax), e uma ordenada mínima (ymin) e máxima (ymax). Qualquer combinação das abcissas com as ordenadas resulta num vértice da caixa e, conhecendo a orientação dos eixos numa imagem, optou-se por se escolher o vértice com a abcissa máxima e ordenada máxima (xmax; ymax). Neste projeto, assumiu-se os dois pontos, A e B, como sendo vértices de duas caixas delimitadoras de dois troncos detetados com as abcissas e ordenadas máximas consideradas, estando um exemplo desta tática na Figura 6.6.

Figura 6.5 – Linhas imaginárias para a estratégia de orientação.

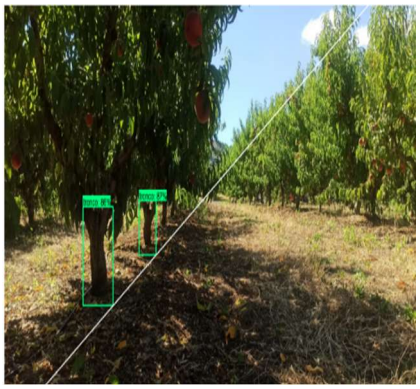


Figura 6.6 – Função linear gerada pela deteção de dois troncos de pessegueiros.

Após a primeira criação da função linear com dois troncos detetados, esta é guardada e atualizada posteriormente caso haja novamente deteção dupla. Esta função linear será o ponto base de orientação e terá duas condições que fazem com que a plataforma robótica se ajuste á trajetória.

A primeira condição é o desvio de um só tronco detetado. Haverá, inevitavelmente, devido a obstruções visuais, troncos de formas ambíguas e à própria imprecisão do modelo, algumas frações de tempo em que só é detetado um tronco. Com a função linear criada, o algoritmo recolhe a ordenada máxima (ymax) do único tronco detetado e calcula uma abcissa pertencente à função (xref). De seguida, esse valor xref é comparado com a abcissa máxima (xmax) da caixa delimitadora e, dada uma tolerância segundo o eixo x de 150 pixéis, o algoritmo avalia se é necessário efetuar uma das três decisões:

- Se xmax>xref+150, a plataforma vira à direita para deslocar a linha da função linear para ficar mais próxima do vértice da caixa em avaliação.
- Se xmax<xref-150, a plataforma vira à esquerda para deslocar a linha da função linear para ficar mais próxima do vértice da caixa em avaliação.
- Se xref-150<xmax<xref+150, o algoritmo assume que a plataforma está com uma boa orientação e prossegue a marcha.

Esta condição é essencial, mas não suficiente, porque o robô pode desviar-se demasiado da trajetória sem deteções e assim a estratégia de deteção do tronco relativamente à função linear pode dar indicações incorretas. Desse modo, para que a plataforma não tenha uma tendência, ou de colisão com a fila de árvores, ou de afastamento desta, é feita também uma avaliação do declive da função linear criada. Uma reta com uma inclinação muito acentuada pode significar que as árvores estão afastadas do panorama comum, ou, no caso contrário, se a reta estiver muito abatida, com um aspeto mais horizontal, significa que o robô tem uma tendência de deslocamento perpendicular relativamente à fila de árvores.

Para evitar esta situação, quando o declive é calculado, verifica-se se este está dentro de um intervalo aceitável de valores. As indicações são as seguintes:

- Se m<-1, significa que a reta criada pela função está a ficar mais acentuada, assim sendo a afastar-se, logo o robô ajusta a trajetória para a esquerda (imagem da direita da Erro! A origem da referência não foi encontrada.).
- Se m>-0,2, significa que a reta criada pela função está a ficar mais abatida, assim sendo em colisão com a fila, logo o robô ajusta a trajetória para a

direita (imagem da esquerda da Erro! A origem da referência não foi encontrada.).

- Se -1<m<-0,2, o algoritmo assume que a trajetória da plataforma é aceitável e prossegue a marcha.

É importante dizer que, no início da marcha do robô, enquanto não houver uma deteção dupla de troncos, o robô continua o seu movimento a direito até adquirir informação para que possa fazer a avaliação do redor.

Esta estratégia é possível graças a um algoritmo criado que faz a deteção com o modelo treinado que, consequentemente, dará a ordem de movimentos segundo comandos de output e estruturas de seleção.

6.3 Resultados e discussão

Após ter sido verificado que a estratégia de orientação estava a funcionar corretamente, prosseguiu-se ao teste do modelo convertido para uint8 e compilado para suporte com Edge TPU. O intuito desta simulação é verificar se o desempenho foi afetado e o modelo está apto para orientar a plataforma robótica a tempo real.

No Capítulo 7 referente à deteção dos frutos, está representada a avaliação aritmética realizada ao modelo convertido, mostrando que se mantém semelhante ao original, sem perdas relevantes de desempenho. Realizou-se uma avaliação do tempo entre inferências para determinar se este é adequado para praticar a deteção simultaneamente dando ordens de orientação sem atraso de informação. Verificou-se que o início de execução levou aproximadamente 0,6 segundos e, daí em diante, as inferências tiveram um intervalo de tempo entre si de cerca de 0,37 segundos.

Na Figura são apresentadas imagens da simulação do modelo no Raspberry Pi 4 com a inferência e com a ordem de orientação.

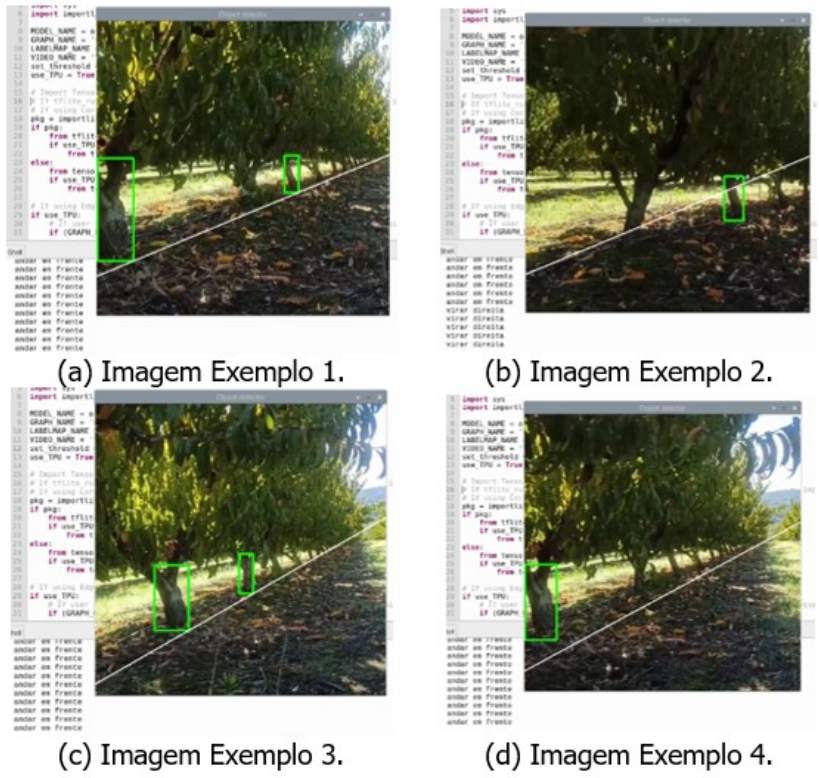


Figura 6.7 - Imagens da simulação feita no Raspberry Pi 4 num vídeo do modelo convertido para TFLite e compilado para Edge TPU.

6.4 Considerações finais

Reunindo os resultados das simulações e os valores das métricas dos vários modelos, obtiveram-se as seguintes conclusões:

Relativamente aos valores das métricas, matematicamente e estatisticamente, o modelo original tem um desempenho excelente com ambas uma precisão e revocação de 94,4%. No entanto, foi crucial a conversão e compilação de um modelo 32 float para 8 uint com suporte de Edge TPU. Só desta maneira se

conseguiu garantir uma maior velocidade de execução da inferência e um peso computacional muito mais reduzido para execução num Raspberry Pi 4, tornando, assim, o modelo ideal para tarefas móveis.

Feita a avaliação à inferência nas mesmas imagens de teste com o modelo quantiado e compilador, o valor da precisão mantém-se praticamente inalterado, provando que quase não existem deteções residuais. Porém, o modelo ficou com uma revocação mais baixa, podendo não identificar troncos por que passe. Todavia, após realizadas as simulações em vídeo, assume-se que esta baixa de valores é aceitável e o modelo está apto para ser testado no terreno.

A execução da inferência no Raspberry Pi 4, com a conversão e compilação do modelo original, é relativamente rápida com intervalos de tempo aproximadamente de 0,37 segundos. entre inferências dos quadros de vídeos, dando bastante tempo à plataforma robótica para se ajustar com uma informação do ambiente redor praticamente exata.

O comportamento do algoritmo mostrou-se adequado e a cumprir os objetivos propostos. As simulações em imagens e vídeos, tanto do modelo original como do convertido, mostraram resultados positivos, com poucas falhas de deteção. O algoritmo inferia a deteção dupla para criação da função linear de referência relativamente rápida e mesmo com deteções individuais, as ordens dadas eram as corretas.

Agradecimentos

Este trabalho foi desenvolvido no âmbito do projeto PrunusBOT – Sistema robótico aéreo autónomo de pulverização controlada e previsão de produção frutícola, Operação n.º PDR2020-101-031358 (Líder), Parceria nº 340 / Iniciativa nº 140, promovida pelo PDR2020 e cofinanciada pelo FEADER no âmbito do Portugal 2020.

Referências bibliográficas

Barawid Jr, O.C., Mizushima, A., Ishii, K., & Noguchi, N. 2007. Development of an autonomous navigation system using a two-dimensional laser scanner in an orchard application. Biosystems Engineering, 96 (2), 139–149.

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., & Saurous, R. A. 2017. Tensorflow distributions. *ArXiv* Preprint ArXiv:1711.10604.

Foundation Raspberry_Pi. 2021. Raspberry Pi 4. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

Hongkun Yu, Chen Chen, Xianzhi Du, Yeqing Li, Abdullah Rashwan, le Hou, Pengchong Jin, Fan Yang, Frederick Liu, Jaeyoun Kim, & Jing Li. 2020. TensorFlow Model Garden. https://github.com/tensorflow/models.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. 2016. Ssd: Single shot multibox detector. European Conference on Computer Vision, 21–37.

Lowe, D. 2017. Electronics All-in-One For Dummies (2nd Edition).

Menezes, G., Gaspar, P.D:, Mesquita, R., Assunção, E. e Simões, M.P. 2022. GPS Based-autonomous navigation system for multitask robotic rover for agricultural activities with augmentes reality web application for supervision support – tests in peach orchards (*in press).*

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., e Chen, L.C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings*.

Simões, J., Gaspar, P.D., Assunção, E., Mesquita, R., & Simões, M.P. 2022. Navigation System of Autonomous Multitask Robotic Rover for Agricultural Activities based on Computer Vision through Tree Trunk Detection - Application to Peach Orchards.

Tzutalin. (2015). Labellmg. https://github.com/tzutalin/labellmg.