FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Harvesting with active perception for open-field agricultural robotics

Sandro Augusto Costa Magalhães

FOR JURY EVALUATION

Doctoral Program in Electrical and Computers Engineering

Supervisor: Professor Doctor António Paulo Gomes Mendes Moreira

Co-supervisor: Doctor Filipe Baptista Neves dos Santos

Co-supervisor: Professor Doctor Jorge Manuel Miranda Dias

June 5, 2024

Resumo

A população mundial tem vindo a experimentar um crescimento exponencial, o que faz aumentar a pressão sobre os recursos agrícolas disponíveis. Apesar disso, a extensão de terras aráveis para o cultivo de alimentos tem permanecido praticamente inalterada, resultando num desiquilíbrio entre a oferta agrícola e o crescente aumento populacional. Este desequilíbrio contribui para um aumento crítico nos índices de fome e subnutrição, que afectam actualmente $\geq$12.9 % da população global.

Perante esta realidade, torna-se imperativo que os produtores adotem medidas para optimizar a produção agrícola, em vias de alcançar níveis de eficiência e inteligência que permitam suprir as necessidades alimentares da sociedade de forma sustentável. Neste contexto, a aplicação de sistemas robóticos surge como uma solução promissora para enfrentar os desafios inerentes à agricultura. Alinhada com a agenda europeia de investigação para a robótica, os objectivos de desenvolvimento sustentável (ODS), e a Federação Internacional de Robótica (IFR), a implementação destes sistemas robóticos pode não só aumentar a produtividade agrícola, mas também mitigar obstáculos significativos do sector.

De acordo com a IFR, os robôs têm potencial para contribuir significativamente para a realização dos ODS estabelecidos pela ONU na Agenda 2030. Especificamente, os ODS 2, 8, 9 e 15, que visam erradicar a fome, promover o crescimento económico sustentável, fomentar a inovação tecnológica e proteger a biodiversidade terrestre, respectivamente, podem ser impulsionados pela adopção de tecnologias robóticas na agricultura. A automação da monitorização de culturas agrícolas, por exemplo, pode reduzir a dependência da aplicação de produtos químicos e aumentar a eficiência da colheita, contribuindo assim para a segurança alimentar global e para o desenvolvimento socioeconómico das comunidades agrícolas.

No âmbito específico da robótica aplicada à agricultura, o estado da arte tem explorado diversas abordagens para a detecção e segmentação de frutos e folhas em ambientes externos, tais como campos agrícolas, e estufas. No entanto, as soluções existentes enfrentam desafios significativos em lidar com obstáculos naturais, como a oclusão causada por folhas e outros elementos vegetais. Nesse sentido, estratégias baseadas em percepção ativa têm sido propostas como uma forma de aprimorar a capacidade dos robôs para adquirir informações do ambiente de maneira eficaz.

No âmbito desta tesem, uma revisão sistemática da literatura foi conduzida para investigar o estado da arte em percepção ativa para robótica aplicada à agrícola, focando de forma particular a colheita automatizada de frutos. Os estudos analisados destacaram a prevalência de abordagens fundamentadas em técnicas de aprendizagem profunda (*deep learning*), reconhecidas como as mais avançadas e eficientes para a detecção de objetos em ambientes complexos. Entre os modelos analisados, destacam-se aqueles baseados nas arquiteturas YOLO e SSD, incluindo a SSD MobileNet v2, a SSD Inception v2 e a SSD ResNet 50. Estes modelos demonstraram uma capacidade satisfatória na detecção de frutos, com desempenho médio de F1 em torno de 60 % e velocidades de inferência de até 25 FPS utilizando FPGAs.

Apesar dos avanços significativos alcançados, os modelos existentes ainda não permitem a percepção tridimensional de frutos, utilizando câmaras monoculares. Como resposta a essa lacuna, foram desenvolvidos dois algoritmos baseados em informações visuais para estimar a posição dos frutos no espaço tridimensional. O MonoVisual3DFilter e o Estimador de Melhores Poses de Observação (BVE) combinado com o filtro de Kalman extendido (EKF), utilizando dados obtidos pelo sistema de detecção de objetos, foram desenvolvidos para inferir a distância e a posição dos frutos no espaço de tarefas. Ambas as abordagens demonstraram resultados promissores, com erros de estimação entre 1 cm e 3 cm.

Ao longo deste estudo, foram gerados conjuntos de dados visuais e código-fonte, os quais foram disponibilizados publicamente seguindo os princípios de dados abertos da união europeia, por meio das plataformas ZENODO e GitLab do INESC TEC. Esta partilha de recursos visa promover a transparência e a colaboração na comunidade científica, facilitando a replicação e a expansão do conhecimento gerado nesta área de investigação.

Abstract

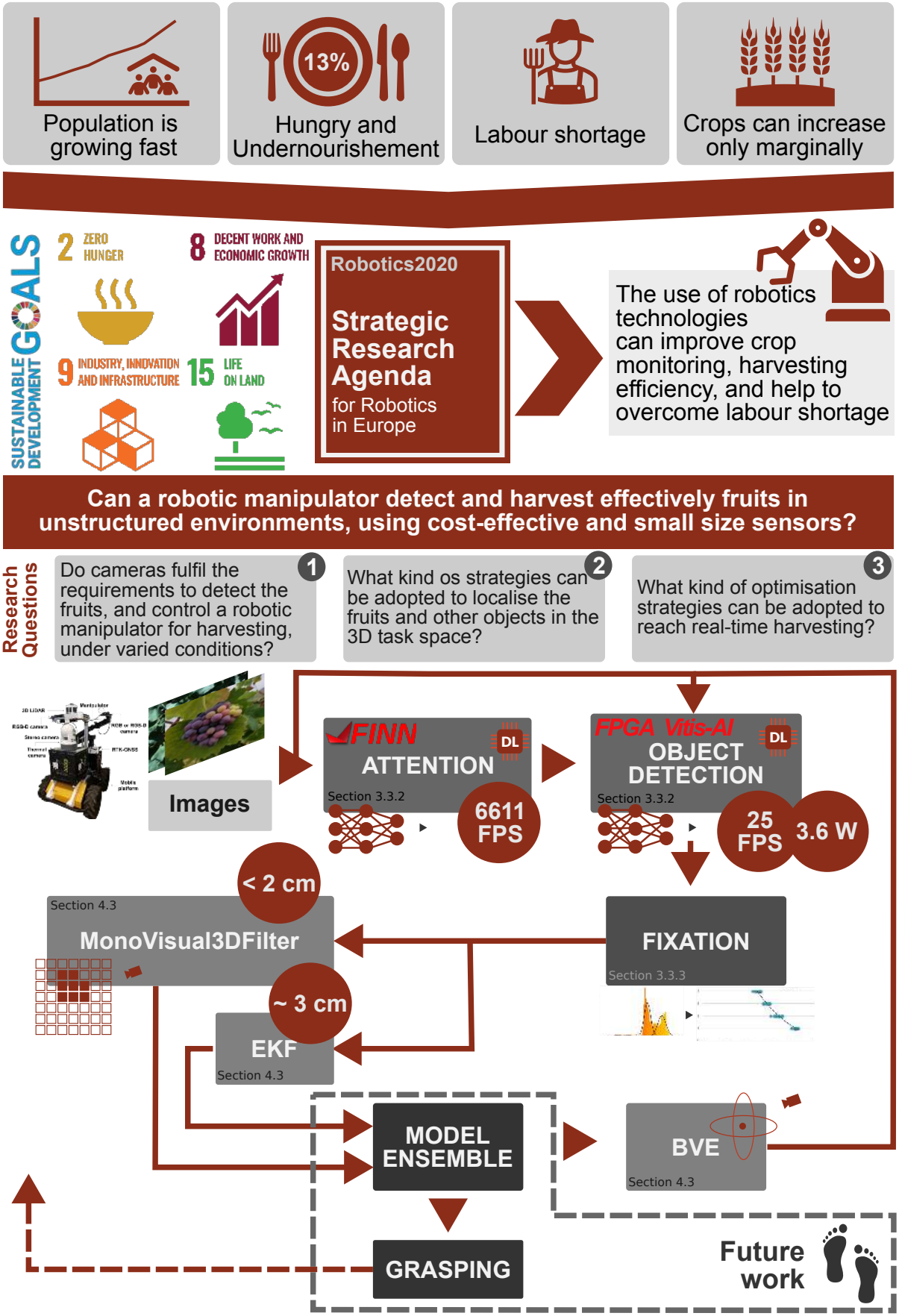
The global population is rapidly increasing, leading to significant challenges in hunger and undernourishment, now affecting ≥12.9 % of people worldwide. With agricultural land being finite and only marginally expandable, it's crucial to enhance productivity by adopting precision and intelligent farming techniques. Robotics technology emerges as a key solution to improve crop monitoring and harvesting efficiency, thus addressing this pressing societal issue, according to the Agenda for Robotics in Europe. The International Federation of Robotics (IFR) also notes that robots can play a significant role in meeting these challenges, contributing to the Sustainable Development Goals (SDGs) set by the United Nations (UN) in Agenda2030. These goals include, between others: zero hunger; decent work and economic growth; industry, innovation, and infrastructure; and life on land (goals 2, 8, 9, and 15, respectively).

In robotics for agriculture, there has been a focus on developing methods to detect and segment fruits or branches in open-field environments. Despite these efforts, many initiatives face challenges due to low visibility and occlusions. The exploration of active perception as an alternative approach to tackle these challenges has been minimal. A comprehensive review was conducted to assess the current state-of-the-art, highlighting the limitations and potential of active perception for efficient fruit detection and harvesting.

Throughout this thesis, we researched the application of advanced visual perception systems powered by deep learning for identifying fruits and other objects in agricultural scenes. We experimented with various deep learning models, including YOLO and SSD algorithms, specifically SSD MobileNet v2, SSD Inception v2, and SSD ResNet 50. These models were optimised on specialised hardware to ensure reliable, near-real-time performance. The models achieved detection F1 scores around 60 % for tomatoes and grape bunches, with acceleration techniques boosting detection speeds up to 25 FPS on FPGAs. Further experiments leveraging the FGPAs' programmable logic enabled us to achieve object detection rates at 6610.94 FPS using a MobileNet v2 classifier.

For the estimation of fruits' 3D positions using monocular cameras, we developed knowledge based algorithms, namely the MonoVisual3DFilter, and the Best Viewpoint Estimator (BVE) + Extended Kalman Filter (EKF). The MonoVisual3DFilter utilises detection data combined with histogram filters to infer fruit positions, while the BVE + EKF method iteratively corrects distance estimates to fruits based on similar data. Both approaches yielded accurate results, with estimation errors within the range of 1 cm to 3 cm.

We have made the datasets and some of the code developed during this project available to the public, adhering to the european open data principles, via ZENODO and GitLab platforms.


Population is growing fast
13%
Hungry and Undernourishement
Labour shortage
Crops can increase only marginally
SUSTAINABLE DEVELOPMENT GOALS
2 ZERO HUNGER
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
15 LIFE ON LAND
Robotics2020
Strategic Research Agenda
for Robotics in Europe
The use of robotics technologies can improve crop monitoring, harvesting efficiency, and help to overcome labour shortage
Can a robotic manipulator detect and harvest effectively fruits in unstructured environments, using cost-effective and small size sensors?
Research Questions
1 Do cameras fulfil the requirements to detect the fruits, and control a robotic manipulator for harvesting, under varied conditions?
2 What kind os strategies can be adopted to localise the fruits and other objects in the 3D task space?
3 What kind of optimisation strategies can be adopted to reach real-time harvesting?
Images
FINN
DL
ATTENTION
Section 3.3.2
6611 FPS
FPGA Vitis-AI
DL
OBJECT DETECTION
Section 3.3.2
25 FPS
3.6 W
< 2 cm
Section 4.3
MonoVisual3DFilter
~ 3 cm
EKF
Section 4.3
FIXATION
Section 3.3.3
MODEL ENSEMBLE
BVE
Section 4.3
GRASPING
Future work

Agradecimentos

Agradeço, neste momento, àqueles que de uma forma ou de outra desempenharam um papel fundamental neste percuso de doutoramento.

Aos meus orientadores, Professor Doutor António Paulo Gomes Mendes Moreira – Professor Catedrático da Universidade do Porto, Doutor Filipe Baptista Neves dos Santos – Investigador Sénior e Coordenador do Laboratório TRIBE (Laboratório de Robótica e IoT para a Agricultura e Floresta de Precisão Inteligente) e do TEC4AGRO-FOOD do INESC TEC, e Professor Doutor Jorge Manuel Miranda Dias, Professor Catedrático da Universidade de Coimbra e da Universidade Khalifa, pela sua atenção e cuidado na orientação do meu projecto de doutoramento.

Ao Doutor Pedro Baptista Machado, Senior Lecturer na Nottingham Trent University (NTU), por me ter recebido, apoiado e orientado durante o meu projecto de ERASMUS+ na NTU, parte deste projecto de doutoramento.

Aos meus colegas e amigos do Laboratório TRIBE do INESC TEC que me acompanharam nesta caminhada de doutoramento e me ajudaram a conduzir este projecto de investigação. Agradeço de forma particular àqueles que colaboraram diretamente comigo nas diferentes fases de investigação e publicações científicas produzidas, nomeadamente o Doutor Luís Santos, o Doutor André Aguiar, a Doutora Tatiana Pinho, o Mestre Germano Moreira, o Professor Doutor Mário Manuel de Miranda Furtado Campos Cunha – Professor Associado da Universidade do Porto. Agradeço ainda, de forma particular ao Doutor Filipe Santos e ao Professor Doutor António Paulo Moreira por desde cedo me terem aceite e acolhido neste laboratório, servindo como ponto de partida para esta caminhada.

Agradeço ainda ao meus amigos e jovens dos escuteiros de Oliveira do Douro e da região do Porto que serviram como ombro amigo em diversas fases desta caminhada. Eles serviram, muitas vezes, como a força que faltava para avançar com coragem e dar mais um passo, tornado possível aquilo que muitas vezes parecia impossível.

Por fim, aos meus pais, ao meu irmão e à minha família, que carinhosamente me apoiaram e me orientaram nas diferentes fases da minha caminhada. Estes foram aqueles que pacientemente me compreenderam e ajudaram a ultrapassar as minhas dificuldades. Muito obrigado pela vossa paciência e apoio.

À Fundação para a Ciência e Tecnologia (FCT), que enquadrando o Fundo Social Europeu (FSE), através do Programa Operacional Regional do Norte (NORTE2020), financiaram este projecto com uma Bolsa de Investigação para Doutoramento com a referência SFRH/BD/147117/2019 (DOI:10.54499/SFRH/BD/147117/2019).

Um agradecimento especial a todos aqueles que me acompanharam nesta caminhada, mesmo àqueles que por esquecimento não foram aqui mencionados.

Sandro Costa Magalhães

"You should be glad that bridge fell down.
I was planning to build thirteen more to that same design"

Isambard Kingdom Brunel

Contents

List of Figures

[1] Preferred reporting items for systematic reviews and meta-analyses

List of Tables

Abbreviations and Symbols

2D Two-dimensional

3D Three-Dimensional

3D-SHT 3D spherical Hough transform

3DMTS 3D move to see

AI Artificial intelligence

ANN Artificial neural network

APV Average pixels value

ASIC Application-specific integrated circuit

AUC Area under the curve

BNN Binary neural network

BVE Best viewpoint estimator

CIELAB L*a*b colour space

CNN Convolutional neural network

CNV Convolutional neural vector

COCO Common objects in context

Colab Google Collaboratory

CPU Central processing unit

CV Computer vision

CVAT Computer vision annotation tool

DaS Detection and segmentation

DaSnet Detection and segmentation artificial neural network

Deep 3DMTS Deep 3D move to see

DL Deep learning

DoF Degree of freedom

DPU Deep learning processor unit

EKF Extended Kalman filter

Faster R-CNN Faster region-based convolutional neural network

FCM Fuzzy C-Mean

FCR False colour removal

FN False negative

FP False positive

FP16 Half-precision floating-point

FP32 Single-precision floating-point

FPGA Field programmable gate array

FPN Feature pyramid network

FPS Frames per second

GNSS Global navigation satellite system

GPU Graphics processing unit

HD High-definition

HFOV Horizontal field of view

HLS High-level synthesis

HOG Histogram of oriented gradients

HSI Hue-saturation-intensity colour space

HSV Hue-saturation-value colour space

IFR International Federation of Robotics

IMRaD Introduction, materials and methods, results, and discussion

IMU Inertial movement unit

INT8 8-Bit integer

IoU Intersection over union

IP Intellectual property

IR Infrared

LiDAR Light detection and ranging

MAE Mean absolute error

mAP Mean average precision

MAPE Mean absolute percentage error

Mask R-CNN Mask region-based convolutional neural network

ML Machine learning

MobileNet Mobile neural network

MSE Mean square error

NCS Neural compute stick

NMS Non-maximum suppression

OID Open images dataset

ONNX Pen neural network exchange

PANet Path aggregation network

PE Process element

PICOC Population—intervention—comparison—outcome—context

PL Programmable logic

PLS Partial least square

PRISMA Preferred reporting items for systematic reviews and meta-analyses
PS Processing system
QNN Quantised neural network
R-CNN Region-based convolutional neural network
RAM Random-access memory
ReLU Rectified linear unit
RG2C Red grape chunk classification
RGB Red-green-blue colour space
RGB-D Red-green-blue colour and depth space
RMSE Root mean square error
RoI Region of interest
RPN Regional proposal network
RRT⋆ Probabilitically optimal rapidly exploring random tree
RTL Register-transfer level
RVM Relevance vector machine
SDG Sustainable development goal
SGD Stochastic gradient descendent
SIMD Single instruction multiple data
SoM System on-module
SOTA State-of-the-art
SPP Spatial pyramid pooling
SSD Single shot multi-box detector
SVM Support vector machine
TCP Tool centre point
TF-TRT TensorFlow TensorRT
TFLite TensorFlow lite
TN True negative
ToF Time of light
TOPS Trillion operations per second
TP True positive
TPU TensorFlow processing unit
TRT TensorRT
UN United Nations
USDA United States Department of Agriculture
VGG Visual geometry group
VRAM Video random-access memory
XAI Explainable artificial intelligence
XML Extensible markup language
YIQ Luminance of the quadrature-phase
YOLO You only look once

Chapter 1

Introduction

This research aims to contribute to the state-of-the-art (SOTA) of agricultural robots, namely harvesting robots. Innovating robots with cognitive algorithms can reduce the robots' hardware costs and improve robots' efficiency. Active perception comprehends the philosophy of algorithms that intend to make robots smarter.

This chapter is divided into six sections. Section 1.1 establishes this research's overall motivation and context. Section 1.2 defines the problem this research aims to solve. Sections 1.3 and 1.4 define the research questions and the objectives this thesis aims to achieve and solve. Section 1.5 describes the research articles published in the scope of this thesis and other main contributions, such as datasets, and the overall document structure.

1.1 Contextualisation and motivation

The strategic research agenda for robotics in Europe 2014-2020 (SRA2020) [1] highlights the importance of robotics in agriculture, especially for enhancing crop monitoring and harvesting. This is crucial to support a rapidly growing global population facing hunger and undernourishment issues, affecting ≥12.9 % of people worldwide. With limited agricultural land, which can only be marginally expanded, there's a pressing need for more efficient resource utilisation. The United Nations (UN) has issued a warning about the urgent need for a transformative change in the global food and agricultural system to meet people's needs [2].

This urgency is echoed in the sustainable development goals (SDGs) set by the UN in the 2030 Agenda [3]. Robotics in agriculture can play a significant role in achieving several SDGs, according to International Federation of Robotics (IFR), including goal 2 (zero hunger), goal 8 (decent work and economic growth), goal 9 (industry, innovation, and infrastructure), and goal 15 (life on land). Robots can help mechanise and automate labour-intensive and repetitive tasks, reduce soil compaction, and enable the precise application of fertilizers and chemicals, thus minimising usage and preventing soil acidification.

Tomatoes stand out among crops for their significance in the global production of fruits. As the second most harvested vegetable globally and a leader in greenhouse vegetable production, tomatoes are a key agricultural product [4]. Over the past decades, greenhouse

tomato production has seen a global increase due to its ability to ensure high productivity and a stable year-round supply. From 2003 to 2017, world tomato production rose annually from 124 million tonnes to over 177 million tonnes, with consumption growing at about 2.5 % annually [4]. In Almería, Spain, home to the world's largest concentration of greenhouses (over 30 000 ha), tomatoes represent 37.7 % of total production [5]. Grape production also plays a significant role globally, especially in wine production, with regions like Douro Valley in Portugal known for their unique terrains and high-quality wines.

Greenhouse tomatoes are highly valued but come with substantial costs. Manual harvesting of tomatoes is particularly labour-intensive, characterised by sporadic, tiring work that requires significant physical effort and repetition. In greenhouses, manual labour can constitute up to 50 % of total production costs, with a significant portion of this expense coming from tomato harvesting, which demands 700 h/yr/ha to 1400 h/yr/ha depending on the cropping system [6, 7]. The challenge of manual tomato harvesting is exacerbated by a global labour shortage and poor working conditions [7–9], necessitating the hire of additional workers during peak seasons [9]. Consequently, there is a growing interest among greenhouse companies in reducing labour costs through automation, including the development of harvesting robots [9, 10].

Creating robots for use in greenhouses and the steep slopes of Douro Valley vineyards presents unique challenges. These robots must navigate through unstructured environments and handle tasks with a high degree of uncertainty. Sensing mechanisms must be capable of identifying tomatoes or grape bunches amidst various disturbances such as uneven plant arrangements, differing plant sizes and shapes, and challenges like leaf coverage, sun glare, and changing light conditions [11, 12]. Additionally, the timing for harvesting climacteric fruits like tomatoes can vary. Depending on their end use, tomatoes can be picked during the physiological maturity phase when they are still green, allowing them to ripen post-harvest, or later, when they have turned red. The choice of when to harvest is influenced by how the tomatoes will be processed and distributed. For instance, tomatoes intended for the local fresh market are typically harvested when red, whereas those meant for long-distance transport are picked at an earlier maturation stage when they are still green (Figure 1.1).

(a) Green tomato (b) Reddish tomato (c) Red tomato

Figure 1.1: Tomatoes' ripeness levels: (**a**) physiological or horticultural maturation; (**b**) early phase of ripening; and (**c**) ripened tomato.

Current research in agricultural robotics is predominantly focused on in-field tasks necessary to guarantee a quality crop and harvest time, such as monitoring, soil sensing, nutrient and pesticide application, irrigation control, harvesting, and processing [13, 14]. However, developing ground robots to accomplish these tasks is a complex challenge because the robotic sensing, recognition and interpretation of the crop need to be efficient, accurate, and robust in these unstructured environments [11, 15]. In fact, current research in precision agriculture is limited and needs significant improvement. The few existing tests made with harvesting robots present a very low success rate [14]. Despite the difficulties imposed to drive and deploy an harvesting robot, there are already some prototypes developed for robotic tomato harvesting [10, 16–20]. On average, the harvesting robots described in the literature have a success rate of 85 % on fruit localisation under controlled scenarios [11], whereas this number falls to 2 % to 6 % in real-world scenarios [21]. The low success in real-world scenarios is due to the difficulties of navigation and perception in unstructured scenarios and limitations imposed by the autonomy of robots [22].

Active robot perception is a promising solution to improve the effectiveness of these robots [23–25]. Perception taken actively distinguishes from conventional perception because sensing systems can be an integral part of the manipulation and intelligence system, which improves the robot's performance significantly, in terms of object detection and manipulation [26, 27]. These sensing systems include 3D vision sensors [28] and other sensor technologies [14, 29]. This approach provides faster and more accurate real-time information about objects positioning, making the robots capable of conducting precision agricultural tasks [30]. In addition, other aspects need to be considered and further developed, such as sensors and manipulators' range, weight, safety, processing time, and scanning environment conditions [31]. These aspects are important to achieve the robotic system's feasibility through hardware and software minimisation, ease of integration, and cost-effectiveness [30]. New mechanical designs of robot grippers could also be a solution to provide higher accuracy in perception [32, 33].

This work's originality is developing cost-effective and open-field robust active perception systems for agricultural robots, enabling their operation in cultivars.

1.2 Problem statement

Harvesting, monitoring, and pruning are essential tasks in agriculture. Typical robots for these tasks are equipped with one or various manipulators, mounted on mobile platforms to efficiently perform the tasks [34]. Some studies have explored the use of manipulators on mobile platforms that navigate along trails [35]. To enhance their functionality, these robots are equipped with specialised sensors and end-effectors [36]. However, to be cost-effective, the number of sensors on these robots should be minimised. Generally, agricultural robots utilise three main types of sensors: cameras, LiDARs[1], and global navigation satellite systems

[1] Light detection and rangings

(GNSSs), which are primarily used for localisation, mapping [37], and ensuring safety. Future robots should also consider radar as a promising technology that provides relevant features from the scene [38]. Samples of features are the detection of hidden objects and cues from the scene due to their higher permeability to vegetation. Figure 1.2 showcases a mobile robot designed for agricultural tasks.

The AgRob v16, as depicted in Figure 1.2, is a prototype of the INESC TEC's TRIBE Laboratory[2]. It is built on the Clearpath Husky mobile platform[3] and features all-terrain wheels, making it suitable for both open-field and controlled agricultural environments. This robot is currently being tested in various settings, including Douro's steep slope vineyards, planar vineyards, and tomato greenhouses. It is equipped with a perception and control head that collects environmental data for navigation and mapping. Additionally, the Robotis Manipulator-H, a six degree of freedom (DoF) anthropomorphic manipulator, is installed at the robot's rear to carry out tasks such as monitoring, pruning, and harvesting. The robot incorporates various sensors in both its control head and manipulator, including multiple stereo and RGB-D cameras, an infrared (IR) camera, a 3D LiDAR, an IMU[4], and a GNSS.

Agricultural robots encounter numerous challenges in successfully harvesting and manipulating fruits. Figure 1.3 depicts a typical environment in a flat vineyard of white bunches of grapes at Quinta da Aveleda, Paredes, Portugal. In this figure, several grapes are largely hidden behind the leaves, providing natural protection against solar radiation. Additionally, the stem is challenging to observe. The vineyard consists of many aligned trees (Fig. 1.5), creating walls of vegetation. These trees, often intertwined, feature dense foliage and large leaves, with fruits typically growing beneath and behind the leaves to shield themselves from intense sunlight. Similarly, figure 1.4 shows the interior of a greenhouse used for tomato production in Barroselas, Viana do Castelo. At the greenhouse entrance, there are tomatoes displaying various colours, indicating their different stages of physiological maturity. It is rare to find fully ripe and red tomatoes in this greenhouse. Like grape production, tomatoes usually grow hidden beneath and behind leaves for protection against solar radiation. Furthermore, tomatoes can be harvested at various physiological maturity stages, adding complexity to the harvesting process as the system must assess the maturity phase before harvesting. Thus, the challenge of harvesting is encapsulated in the overarching goal:

Main Goal 1. Can a robotic manipulator detect and harvest effectively fruits in unstructured environments, using cost-effective and small size sensors?

To ensure high-quality fruit reaches consumers, there are specific harvesting techniques for different fruits. For instance, grapes must be carefully harvested by cutting the stem, while tomatoes can be either pulled or cut by the stem, although the exact method may vary based

[2] See INESC TEC. "TRIBE—Laboratory of Robotics and IoT for Smart Precision Agriculture and Forestry," INESC TEC. (2023), [Online]. Available: http://tribe.inesctec.pt (visited on 12/11/2023).

[3] See Clearpath Robotics. "Husky—unmanned ground vehicle." (Feb. 20, 2024), [Online]. Available: https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/ (visited on 05/16/2024).

[4] Inertial movement unit

(a) AgRob v16 and the installed sensors

(b) Front view

(c) Rear view

Figure 1.2: AgRob v16 — A mobile robot for agricultural purposes, developed by INESC TEC — TRIBE Laboratory, Laboratory of Robotics and IoT for Agriculture and Forestry. Designed for agricultural tasks in vineyards and other cultivars, including pruning, monitoring, and harvesting.

on the tomato species. It's crucial for harvesting robots to accurately identify the fruit, its stem, and any surrounding obstacles to harvest successfully. Additionally, these systems must be able to assess a fruit's maturity stage accurately before harvesting to ensure that properly riped fruits are picked.

To navigate through the foliage efficiently, the harvesting manipulator should be equipped with small sensors that enhance manoeuvrability. Moreover, both the robot and its sensors need to be cost-effective and reliable across various weather conditions to meet the budget and needs of farmers.

As discussed in the literature review (chapter 2), there is significant room for improvement in these systems. Enhancements in speed, visual perception efficiency, and the capabilities for manipulation and grasping are essential to match the effectiveness of human workers in harvesting tasks.

Figure 1.3: Sample image depicting bunches of white grapes hidden on a vine.

Figure 1.4: Greenhouse of tomato entrance with tomatoes in the plants and on the ground.

■

Robots designed for agricultural purposes, such as fruit harvesting, must be equipped with various sensors to gather information about their environment effectively. Frontal sensors, as in the figure 1.2, frequently utilised for navigation and localisation, can also serve to identify fruits from specific angles, activating the harvesting perception system to focus on potential regions of interest (RoIs). However, these sensors often fall short of providing the detailed information required for harvesting. To obtain precise and accurate data, at least one sensor, such as a camera or a LiDAR, should be mounted on the harvesting tool itself. This complements the data collected by the general sensors used in mobile robotics. So, the system presents challenges and procedures that need to be addressed to achieve the primary

Figure 1.5: Douro's steep slope vineyards

goal previously outlined. These procedures and challenges include:

- Utilising manipulator sensors to roughly identify regions of interest, such as fruits and stems, thereby engaging the robot's **Attention Mechanism** towards potential harvesting sites.
- Selecting a specific region of interest from the identified areas, optimising the harvesting process while avoiding conflicts between regions. This entails processing the data from that region and selecting the best approachable region of interest, a task referred to as the **Fixation Mechanism**.
- Detecting and segmenting fruits and stems for accurate mapping and localisation, mainly performed by **detection and segmentation (DaS)** and **3D Perception**.
- Enhancing environmental information through well-defined arm movements for 3D mapping, aiming to identify the optimal cutting or harvesting point. This involves **Data Augmentation** and **Viewpoint Selection**.
- Planning the most efficient path to the cutting point, ensuring visibility of the cutting point on the stem, highlighting the need for **Motion Planning** algorithms.
- Creating and controlling the robot's trajectory to the cutting point, incorporating online replanning based on the precise available information about the cutting point. This falls under **Motion Plan and Control**.

These requirements highlight the need for active perception systems, as depicted in Figure 1.6. The figure presents a framework for developing active perception systems grounded

in a literature review, in chapter 2, and the objectives of this research, serving as a guideline for the work conducted in this thesis. The proposed architecture depicts a set of mechanisms to gather information from the environment smartly for successfully performing tasks, a role of active perception algorithms. The different detailed algorithms are set into different groups for different purposes, such as attention, fixation, viewpoint selection, and data augmentation. They are also complemented by control algorithms for the robot's motion planning and control and data acquisition algorithms such as detection and segmentation.

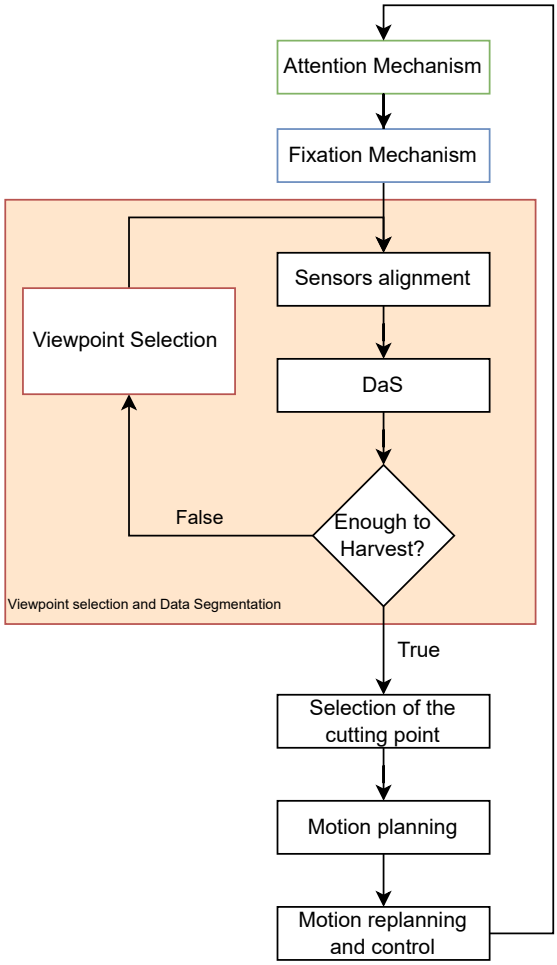


Figure 1.6: Generic framework for an active perception system. Based on the literature review in Chapter 2 and the research objectives, this framework serves as a guideline for this thesis.

Furthermore, the agricultural sector's need for robots that can match or surpass human labour capabilities is imperative. This means robots must process sensory data with high speed, precision, and efficiency, ensuring the optimisation of computational and hardware resources, as well as energy efficiency for extended operation times.

The cognitive development of harvesting robots encounters several challenges, including:

- varying lighting conditions;

- varying fruit shape, size, and colour;
- occlusions of the regions of interest;
- dynamics or unstable regions of interest;
- the need for high manoeuvrability end-effector tools, and;
- complex and computing-demanding algorithms.

Addressing these challenges is crucial in agricultural settings, such as vineyards and tomato greenhouses, where the development of efficient harvesting robots capable of picking all fruits on plants is essential. Robust and rapid active perception mechanisms can significantly enhance information gathering, management, processing, and decision-making processes in these environments.

1.3 Research questions and hypothesis

In response to the problem outlined in Section 1.2, this thesis proposes some research questions aimed at addressing and resolving the identified issue. The overarching goal, specified in the goal 1, encapsulates the primary concern into a single, broad inquiry. This primary goal is further broken down into three specific research questions, which will drive the scope of the work presented. For each of these research questions is possible to formulate hypotheses to steer the investigation towards feasible solutions.

Research Question 1. Do RGB cameras meet the necessary criteria for accurately detecting and tracking fruits and trees, as well as controlling a robotic manipulator for harvesting, under various illumination conditions?
In literature, RGB and RGB-D cameras are commonly utilised for better perception and localisation of fruits and other regions of interest. However, RGB-D cameras tend to be more expensive and larger than their RGB counterparts. The exploration of other sensors, such as LiDAR and unconventional camera types (e.g., multispectral, thermal, and infrared cameras), has also been considered. Despite their larger size and the high cost of accurate versions, their effectiveness hasn't been thoroughly clarified in the literature. Yet, sensors complementing RGB cameras are frequently employed.

The literature review in chapter 2 highlights the use of artificial intelligence (AI) algorithms for image processing and analysis. These algorithms are generally categorised into two main groups: (i) classical machine learning (ML) and computer vision (CV) techniques, and (ii) deep learning (DL) models. ML approaches often involve more traditional algorithms, such as support vector machine (SVM), combined with CV methods like region growing, skeletonisation, and colour thresholding. DL, on the other hand, is often seen as a 'black box' approach that processes images to identify the most probable regions of interest. It is applied for classification, object detection, or segmentation tasks in image analysis.

Despite the focus on identifying fruits for harvesting, recognising the surrounding environment remains crucial to refine the detection accuracy. For example, fruits on the ground should not be picked up as they might be rotten, and the system should target only ripe fruits for harvesting. Moreover, incorporating effective attention mechanisms can enhance the system's efficiency by allocating computational resources primarily to the regions of interest, such as scanning the trees for fruits and locating the stems near these fruits.

Therefore, this research question aims to evaluate the effectiveness of using RGB cameras for detecting and tracking fruits in cultivars. It will explore the most suitable strategies to achieve this goal efficiently.

Hypothesis. *To study models and detection techniques, such as DL models, to identify fruits under various conditions in cultivars more effectively. Additionally, explore other algorithms and strategies to enhance knowledge about regions of interest, facilitating an active perception system.*

Hypothesis. *Studying the perception needs of the regions of interest and determining whether they match the configurations of various sensors, like eye-in-hand coordination and peripheral cameras, among others.*

Hypothesis. *DL strategies are increasingly being recognised as effective approaches for detecting and segmenting regions of interest. The current state-of-the-art features a multitude of DL models. Yet, there is a critical need for comprehensive benchmarking and in-depth analysis to evaluate their potential for enhancements in accuracy and reliability. This is particularly vital for ensuring robustness against variations in illumination and environmental conditions. Additionally, it's important to explore potential improvements to better accommodate sequential data flow.*

■

Research Question 2. What strategies can be implemented to localise fruits and other objects in the 3D task space?

Numerous methods have been devised to convert sensor data about specific areas into useful information for tasks. The most commonly mentioned sensors in research are *RGB-D* sensors. However, literature reviews indicate that their effectiveness is compromised by natural factors like sunlight or rain. To counteract these issues, additional sensors can be integrated, although this approach increases the system's complexity and cost. Moreover, it leads to a bulkier sensory setup, complicating its integration with a robotic arm and its manoeuvrability within crops.

Hypothesis. *In geospatial analysis, triangulation methods are commonly employed. Similarly, observation estimators and statistical algorithms are frequently used in various contexts for prediction purposes. Therefore, this research will focus on studying algorithms that leverage*

these principles, among others, to collect data about specific regions of interest efficiently. This will enhance the functionality of active perception systems.

■

Research Question 3. What strategies can be implemented to achieve real-time optimisation in harvesting?
The literature review discussed in chapter 2 reveals that the majority of harvesting robots take longer to complete the harvesting process compared to human workers. This increased time consumption is primarily due to the time required for data processing and robot movement.

The need for complex and sophisticated algorithms for image processing and motion planning significantly contributes to this delay. Furthermore, the extensive data collected from sensors adds to the processing time, making the overall operation slower.

Hypothesis. *Many of the complex algorithms used for data processing and planning paths rely on loops that can run at the same time, through parallelisation. Therefore, parallelisation and other optimisation techniques, along with edge computing, can enhance the efficiency and speed of these algorithms.*

■

1.4 Aim and scope

Given the problem statement in section 1.2, alongside the research questions outlined in section 1.3, and considering the current advancements in robotic technologies for harvesting tasks as discussed in chapter 2, this research thesis aims to explore, benchmark, and advance the development in this field. Towards this scope and aims the following efforts were developed:

- Robust active perception algorithms, reliable under numerous weather conditions, to recognise fruits and other elements, in real-time, for harvesting tasks;
- Reliable and intelligent decision support systems to identify and localise the regions of interest, considering the gathered and processed data, and;
- Optimised robotic harvesting solutions that may be competitive and complimentary to human labour.

This work aimed to compare various solutions, including combinations of sensors, manipulators, and processing devices. We also focused on the optimisation concerns towards fast and near real-time algorithms. So, we tested perception and intelligent algorithms in terms of their processing speed in simulation and laboratory-simulated agricultural contexts.

1.5 Document Structure and Contributions

This section lists this PhD's contributions to the proposed research topic. Some of these publications have not yet been published in the press. However, they are relevant to this work. The section also contributes with a global overview of the following chapters of this thesis.

1.5.1 Main contributions

[C41] S. A. Magalhães, A. P. Moreira, F. N. dos Santos, and J. Dias, "Active perception fruit harvesting robots — a systematic review," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 14, May 2022. DOI: 10.1007/s10846-022-01595-3.

[C42] S. A. Magalhães *et al.*, "Evaluating the Single-Shot MultiBox detector and YOLO deep learning models for the detection of tomatoes in a greenhouse," *Sensors*, vol. 21, no. 10, p. 3569, May 2021, ISSN: 1424-8220. DOI: 10.3390/s21103569.

[C43] S. C. Magalhães, F. N. dos Santos, P. Machado, A. P. Moreira, and J. Dias, "Benchmarking edge computing devices for grape bunches and trunks detection using accelerated object detection single shot multibox deep learning models," *Engineering Applications of Artificial Intelligence*, vol. 117, p. 105 604, Jan. 2023. DOI: 10.1016/j.engappai.2022.105604.

[C44] S. C. Magalhães, F. N. dos Santos, A. P. Moreira, and J. Dias, "MonoVisual3DFilter: 3D tomatoes' localisation with monocular cameras using histogram filters," *Robotica*, 2024, Publication in Press.

[C45] S. A. Magalhães, A. P. Moreira, F. N. do Santos, and J. Dias, "BVE + EKF: A viewpoint estimator for the estimation of the object's position in the 3D task space using extended Kalman filters," in *Proceedings of the 21st International Conference on Informatics in Control, Automation and Robotics: ICINCO*, Article submitted, INSTICC, Porto, Portugal: SciTePress, 2024.

1.5.2 Complimentary contributions

[C46] T. C. Padilha, G. Moreira, S. A. Magalhães, F. N. dos Santos, M. Cunha, and M. Oliveira, "Tomato detection using deep learning for robotics application," in *Progress in Artificial Intelligence. EPIA2021*, G. Marreiros, F. S. Melo, N. Lau, H. Lopes Cardoso, and L. P. Reis, Eds., 12981 vols., ser. Lecture Notes in Computer Science, Springer International Publishing, 2021, pp. 27–38. DOI: 10.1007/978-3-030-86230-5_3.

[C47] A. S. Aguiar *et al.*, "Grape bunch detection at different growth stages using deep learning quantized models," *Agronomy*, vol. 11, no. 9, p. 1890, Sep. 2021, ISSN: 2073-4395. DOI: 10.3390/agronomy11091890.

[C48] G. Moreira, S. A. Magalhães, T. Pinho, F. N. dos Santos, and M. Cunha, "Benchmark of deep learning and a proposed HSV colour space models for the detection and classification of greenhouse tomato," *Agronomy*, vol. 12, no. 2, p. 356, Jan. 2022. DOI: 10.3390/agronomy12020356.

[C49] S. Magalhães, F. N. D. Santos, and S. Shyam, *Grape detection using Vitis AI and RetinaNet*, Online, Apr. 2022. [Online]. Available: https://www.hackster.io/452741/grape-detection-using-vitis-ai-and-retinanet-7d0d71.

1.5.3 Public Open Datasets

[C50] S. A. Magalhães, *Dataset of tomato inside greenhouses for object detection in Pascal VOC*, Pascal VOC, INESC TEC research data repository, Dataset, Last accessed on April 27th, 2023, INESC TEC, Jan. 2021. DOI: 10.25747/pc1e-nk92. [Online]. Available: https://rdm.inesctec.pt/dataset/ii-2021-001.

[C51] S. A. Magalhães, G. Moreira, F. N. dos Santos, and M. Cunha, *AgRobTomato dataset: Greenhouse tomatoes with different ripeness stages*, Pascal VOC, Dataset, INESC TEC, 2021. DOI: 10.5281/ZENODO.5596799.

[C52] G. Moreira, S. A. Magalhães, T. Padilha, F. N. dos Santos, and M. Cunha, *RpiTomato dataset: Greenhouse tomatoes with different ripeness stages*, Pascal VOC, Dataset, INESC TEC, Oct. 2021. DOI: 10.5281/ZENODO.5596363.

[C53] A. S. Aguiar and S. Magalhães, *Grape bunch and vine trunk dataset for deep learning object detection*, Pascal VOC, Dataset, INESC TEC, 2021. DOI: 10.5281/ZENODO.5139598.

[C54] M. Almeida, S. C. Magalhães, and F. N. d. Santos, *RG2C: Red grape chunk classification dataset*, 2023. DOI: 10.5281/ZENODO.8124484.

1.5.4 Document structure

The document is structured as follows.

Chapter 2 presents an extensive review of the literature pertaining to active perception and perception strategies, drawing significantly on the works of Magalhães *et al.* [C41]. This review adheres to the PRISMA[5] guidelines for conducting systematic reviews. Given the specific research goals of this thesis, we have incorporated additional iterations into the systematic review process, in alignment with the methodologies detailed in [C42, C48].

The methodology and experimental design are detailed across two chapters. Chapter 3 focuses on solving the perception challenge of detecting and assessing fruits and trunks [C42, C46, C47]. Through a series of experiments, we aim to enhance the capacity for perception and the speed of inference, thereby approximating a solution suitable for near real-time applications [C43, C47]. Subsequently, chapter 4 advances the discussion to active perception

[5] Preferred reporting items for systematic reviews and meta-analyses

strategies, examining algorithms that enable the camera to be actively positioned in strategic locations for optimal object detection [C44].

The final chapter, chapter 5, provides a comprehensive evaluation of the findings and discussions. It also outlines potential future research directions aimed at refining the proposed solution and facilitating its integration into actual prototypes.

Chapter 2

Literature review

This chapter performs a literature review to identify the main gaps in agricultural robots to perform tasks like harvesting or monitoring, considering the active perception paradigm. Here, we explore the context and the definition of active perception and frame its usability in the agricultural context. For better contextualisation, we frame our research in the problem statement previously made. The literature review mainly follows a systematic review protocol that was further improved with additional and complementary research.

We published the results of this chapter in the literature review of several research articles. However, the review article [C41] published in a scientific journal comprehends the main contributions of this literature review.

2.1 Introduction

The significance of conducting a literature review at the outset of any research project cannot be overstated, as highlighted by Baker [55]. This crucial step involves meticulously searching for and analysing existing literature within the scope of the research topic to pinpoint pivotal contributions that underscore the research's relevance and rigour. Moreover, a well-executed literature review serves a dual purpose: it not only identifies existing research gaps but also lays down a roadmap for new research directions aimed at bridging these gaps.

Applying stringent and relevant frameworks to guide the literature review process is indispensable in this context. Such frameworks ensure that significant contributions are recognised and acknowledged, thereby preventing needless duplication of known information. Furthermore, they guarantee that the literature review is thoroughly documented, which, in turn, facilitates future updates and enables the replication of research findings, a process deemed critical for peer review [56].

Literature reviews often manifest as systematic reviews [57] or meta-analyses [58]. Among the methodologies employed, the PRISMA statement [59] is widely recognised for conducting systematic reviews, especially within medical research domains. However, this method, characterised by its linear and constrained approach, could constrain some disciplines such as information systems, which are marked by rapid advancements and a proliferation of lit-

erature [56]. Given the dynamic nature of these fields and the evolving taxonomies, Brocke *et al.* [56] proposed an iterative, five-step framework specifically designed to conduct literature reviews in information systems, ensuring both the reliability and comprehensiveness of the reporting.

Despite the evident benefits associated with the methodology proposed by Brocke *et al.* [56], incorporating more systematised approaches like PRISMA could enhance the validation and replicability aspects of the literature review. Consequently, this literature review commenced with a systematic examination of the literature spanning from 2016 to September 21, 2021 [C41]. Subsequent rounds of review were conducted to update the literature base and to delve deeper into the state-of-the-art.

An initial exploration of active perception was undertaken using the Scopus and ISI Web of Science databases. This preliminary step aimed to ascertain the research interest and validate the significance of developing active perception systems for open-field robotics, particularly in agricultural tasks such as harvesting or monitoring (Figures 2.1 and 2.2). Figure 2.1 delineates the trend of publications in active perception, a field that gained traction following seminal works by Bajcsy [60] and others [61]. Despite a gradual increase in research output, the complexity and computational demands associated with intelligent systems may have impeded a rapid growth in literature. Additionally, the potential misclassification of relevant contributions under different terminologies could further skew the actual volume of research in this area.

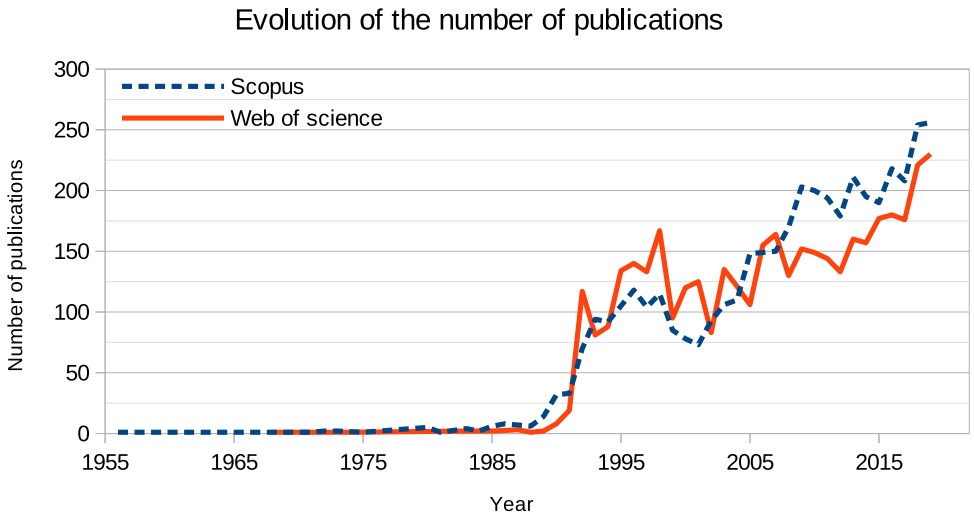


Figure 2.1: Trend of publications on active perception in the Scopus and ISI Web of Science databases using the key `("active perception" OR "active sensing" OR "active vision")`.

Open-field robots, particularly those deployed in agricultural settings, form the core subject of this research. The primary application of these robots has been in harvesting tasks, as depicted in Figure 2.2, which illustrates the growing interest in harvesting robots since the

early 21st century. It is crucial to note that while numerous studies focus on harvesting robots, their application is not limited to agriculture but extends to the medical field as well because some similar terminology in the literature and further filtration of the results is demanded.

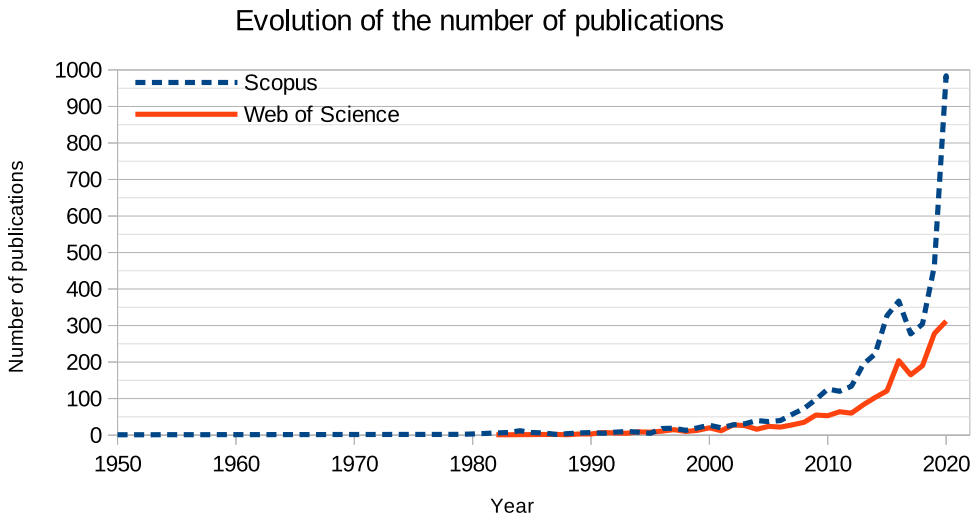


Figure 2.2: Trend of publications on harvesting robots in the Scopus and Web of Science databases, using the key `(harvest* AND robo*)`.

This review comprehensively examines the literature on harvesting robots within the agricultural domain, noting a predominant reliance on passive perception systems, even when incorporating active sensors such as LiDAR [62]. While many studies represent initial explorations and detailed strategies for detecting fruits under natural conditions [C42, 63, 64], those employing partially or fully active perception approaches [65–69] demonstrate enhanced reliability and intelligence in executing harvesting tasks.

Contrastingly, other analyses focus on detection and segmentation algorithms for fruit identification in natural settings [70–72], yet overlook the assessment of active perception systems in the agricultural sector for fruit harvesting. This review extends beyond active perception strategies to evaluate prevailing trends concerning the most utilised sensors, detection and segmentation techniques, and the acceleration of algorithms.

The methodology of this review adheres primarily to the PRISMA statement protocol [59], with an additional in-depth examination of fruit detection and algorithm acceleration.

The structure of the subsequent sections is as follows. Section 2.2 delves into the fundamentals of active perception and aims at a comprehensive systematisation of the concept. Section 2.3 discusses potential robotic architectures for agriculture and harvesting, informed by the sensor technologies reviewed in Section 2.4. Further, Sections 2.5 and 2.6 explore the nuances of active perception through the lenses of attentional and fixation mechanisms, as well as information-seeking methodologies, respectively. Section 2.7 investigates methods to expedite computationally demanding algorithms. Lastly, Section 2.8 synthesises the findings

from the literature review. The appendix A details the research protocol behind this literature review and some protocol alterations.

2.2 Active Perception

This current section endeavours to elucidate the concept of active perception, subsequently delving into its applications within the domain of fruit harvesting in agriculture.

Bajcsy [60] pioneered the formal discourse on **active perception** in 1988, proposing a broadened perspective that transcends the mere employment of active sensors (such as LiDAR, radar, or sonar). Bajcsy [60] posited that passive sensors, for instance, cameras, can also be harnessed in an active manner. Thus, she delineated active perception as a control challenge aimed at the data acquisition process, advocating for a control law that dynamically adjusts the sensing apparatus in alignment with the objective or task at hand and the current state of data interpretation. An integrative active perception framework, as per Bajcsy [60], entails the inclusion of reasoning, decision-making, and control phases.

At the inception of this discourse, contributions to active sensing were scant, prompting Bajcsy [60] to formalise this concept, foster further contributions, and delineate its distinctions from active vision [61, 73–75].

Aloimonos *et al.* [61] contended that not all implementations of active vision qualify as active perception systems. Active vision is characterised as a standalone, vision-centric system capable solely of sensor movement [61], such as pan and tilt mechanisms (gaze control mechanisms) that alter camera positions. If the camera's movement is uninformed by environmental awareness, it does not constitute an active sensing system [73]. Nonetheless, both Bajcsy [60] and Aloimonos *et al.* [61] acknowledged the potential of active vision in augmenting active perception applications.

A comprehensive review conducted in 2011 assessed the advancements in active perception over the preceding 15 years, concluding that active perception epitomises the proactive application of vision sensors for information retrieval and exploration within an environment [76]. In this era, Chen *et al.* [76] refined the definition of active perception to entail the astute determination of the visual sensor's pose and configuration, necessitating the strategic planning of multiple viewpoints to circumvent the visual sensors' limited field of view. The optimisation of viewpoint selection not only enhances scene reconstruction quality but also ameliorates task completion time [76, 77].

Gualtieri *et al.* [77] further underscored the significance of the 'where to look' dilemma within the active perception paradigm, advocating for a robotic system that intelligently selects varying viewpoints during planning to enrich scene understanding.

Vision systems are often actively manipulated to track objects or adapt to environmental changes, such as adjusting the white balance or the camera sensor's sensitivity. These systems, being reactive, tailor their responses to predefined stimuli based on preconfigured settings. Within the context of active perception, an active vision system is expected to intelli-

gently respond to stimuli, selecting the optimal sensor configuration to enhance perception. This may involve sensor repositioning for stimulus search and view alteration to augment stimulus perception. To efficiently allocate robotic resources, stimuli are typically captured and processed through attention mechanisms [78], which act as filters to prioritise sensory signal processing. Thus, in an active vision system aimed at active perception, the visual sensor is strategically moved to a purposeful visual perception pose.

The most recent formal definition of active perception was introduced by Bajcsy *et al.* [23], who revisited and refined the original concept to reflect the advancements in the state-of-the-art. They delineated the essential components requisite for any artificial agent, positing:

> "An agent is an active perceiver if it knows why it wishes to sense, and then chooses what to perceive, and determines how; when and where to achieve that perception." [23]

According to Bajcsy *et al.* [23], for an artificial agent to achieve perception, it must fulfil a comprehensive set of criteria known as the perception pentad: *why*, *what*, *how*, *when*, and *where*, as illustrated in Figure 2.3. The dimension of *why* pertains to the agent's motive – considering both its expectations and current state, the agent selects subsequent actions to engender new states, which might include remaining stationary. This decision-making process hinges on various forms of inductive reasoning. The *what* aspect focuses on identifying specific segments of the environment (e.g., an object) that the agent aims to observe, a process often termed as Scene Selection. The *how* component involves the sequences of actions leading up to the observation, which can include mechanical alignment (positioning within the appropriate sensory field), sensory alignment (adjustment of sensory mechanisms for optimal scene capture), and priming (tuning the perception system for effective interpretation of sensory data). The *when* factor addresses the timing of expectations, determining their relevance and duration. Lastly, the *where* component involves selecting the optimal viewpoint and sensory modality for each specific expectation.

In essence, an active perception system is purposive and seeks information, as discussed by [79]. It involves the dynamic control of sensory apparatus to align with specific tasks, which may include maintaining a state of inactivity.

2.3 Harvesting platform architecture

Active perception systems, as detailed in section 2.2, should have cognitive algorithms. These algorithms enable an intelligent selection of actions to control sensors and actuators actively. This control aims for efficient information-seeking and data augmentation. In light of these requirements and the essential functions for harvesting fruits, as discussed in the preceding section, figure 2.4 introduces a universal framework for a mobile harvesting manipulator. This manipulator is mounted on a mobile platform as illustrated in figure 1.2.

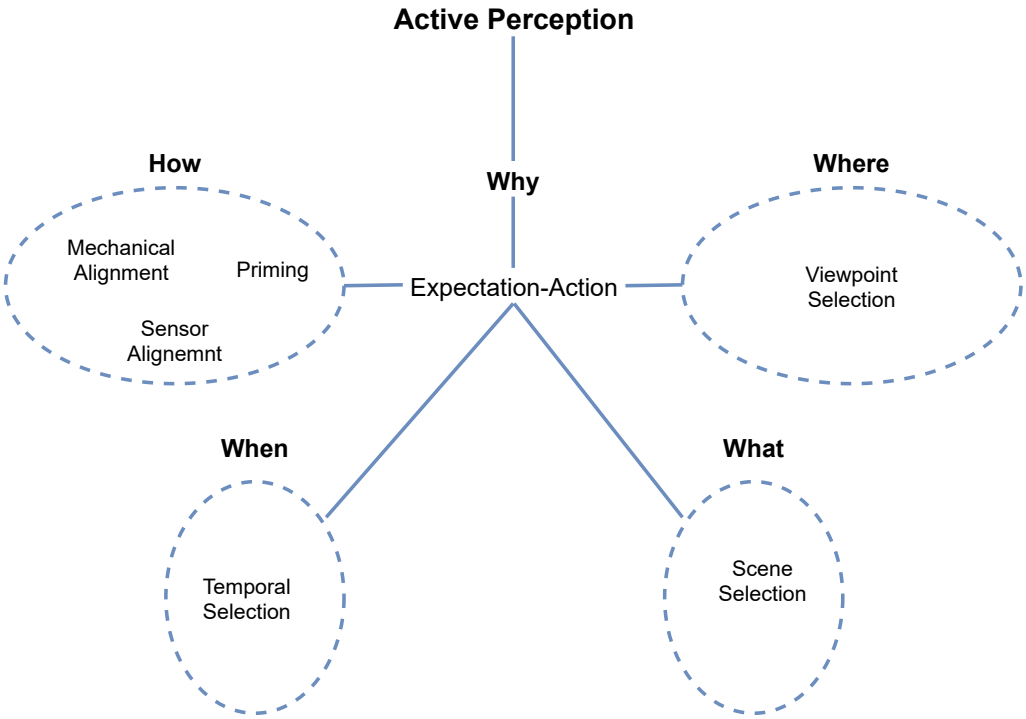


Figure 2.3: Fundamental elements of active perception systems, adapted from Bajcsy *et al.* [23].

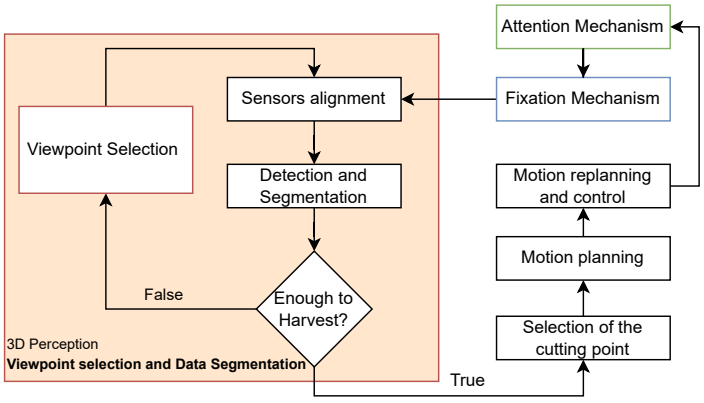


Figure 2.4: Schematic of the active perception system framework designed for harvesting grape bunches in vineyards.

A streamlined algorithm for the attention mechanism facilitates identifying the main attributes of the region of interest, namely, the fruits. These attributes could include colour, such as identifying red grape bunches or ripe tomatoes, or shape, such as detecting clusters resembling circles – which are indicative of grape bunches or tomatoes. Upon locating a potential fruit, the robot's manipulator aligns with the region of interest. This alignment provides a strategic viewpoint for observing the region of interest. Subsequently, a DaS algorithm processes the acquired perspective data to confirm the region of interest's validity.

However, obstructions may limit the information available, making it challenging to determine the fruits' shape and size, pinpoint their position, and identify the stem or the cutting point on the stem. The viewpoint selection algorithm plays a crucial role in validating this information and choosing the next viewpoint to observe the fruit. This process ensures the collection of the most comprehensive and relevant data. These algorithms, essential for enhancing environmental data and swiftly selecting target areas, are fundamental for active perception systems [78, 23]. Moreover, this framework can be extended to include algorithms for tracking the region of interest up to the point of harvesting, encompassing motion planning and control algorithms. The motion planning process within this framework comprises two stages: an initial offline path planning algorithm is executed [80, 81], followed by an online planning algorithm that refines the path with newly acquired data during execution [82].

2.4 Sensors

Besides the initial expectations set by Bajcsy [60] regarding sensor use in active perception, recent research remains predominantly focused on the deployment of active vision systems within this domain (see Figure 2.5 and [C41]). Specifically, these systems emphasise the active manipulation of cameras to cognitively gather and analyse data concerning region of interest. Nonetheless, there has been a noticeable increase in the incorporation of haptic, IR sensors, and spectrometers (refer to Figure 2.5) to enrich environmental data [C41], particularly in capturing tactile information such as material roughness and other intrinsic properties. Although, it is important to note that the value of haptic feedback is primarily confined to the immediate vicinity of the region of interest. This proximity allows for the correction and verification of the regions's position, supplementing the camera's visual data with tactile details like roughness, as highlighted in studies by [83] and [84]. These enhancements are crucial for precision tasks in contexts such as pruning or harvesting.

Regarding camera technology, RGB digital cameras remain the prevalent choice, as illustrated in Fig. 2.5, mainly due to the current research focus on vegetative stages in agriculture, which often concentrates on fruit detection and segmentation. However, more sophisticated approaches for fruit detection, segmentation, and localisation are employing RGB-D cameras or stereo cameras, also known as binocular cameras. An interesting deviation is the use of IR cameras for detecting peas, as noted by Tejada *et al.* [85]. IR cameras offer robustness against various light sources but are sensitive to temperature fluctuations and require a dedicated light source for optimal operation.

The integration of RGB cameras with time of light (ToF) or LiDAR sensors enhances the capability to detect and analyse objects effectively and in the 3D space, as illustrated in Fig. 2.5. This combination is instrumental in developing cost-effective RGB-D sensors, which are pivotal for measuring distances between the sensor and the objects. Additionally, these sensor combinations enable the creation of sophisticated systems that can evaluate fruits comprehensively. Specifically, LiDAR sensors have been utilised independently to identify apples

on trees by leveraging the reflectance index to differentiate fruits from leaves and the background, as highlighted by [62].

Although haptic sensors are seldom used for fruit evaluation, emerging advancements suggest a potential for future application. On the other hand, IR sensors are increasingly employed as proximity detectors for fruits, as noted in the studies [86–88]. Moreover, spectrometers are not only useful for fruit detection but also play a crucial role in assessing fruit quality and characteristics, thereby supporting selective harvesting efforts [69, 89, 90].

Figure 2.5 summarises the sensor types applied in fruit detection and analysis. This figure is a Venn diagram reporting the number of articles using a sensor type. The number of papers is presented in an isolated manner, i.e., there are 110 articles using RGB sensors, 3 articles using both RGB and RGB-D sensor, 1 article using stereo and monocular cameras, and 6 articles using both monocular cameras and LiDAR. So, this diagram reports, for instance, a total of 120 articles using monocular cameras. From the diagram analysis, we conclude that the literature often focuses on employing either single RGB cameras or RGB-D cameras. Notably, when RGB-D sensors are utilised, there is a significant emphasis on performing 3D localisation of fruits, enhancing the precision and efficiency of the detection process.

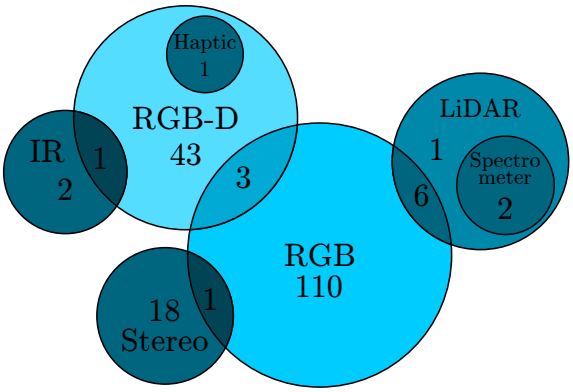


Figure 2.5: Sensors used for fruit detection, segmentation, localisation, and assessment. The

2.5 Attention and Fixation Mechanisms

In their study, Balkenius and Hulth [78] delineated the concept of attention mechanisms as pivotal for decision-making processes that filter and prioritise sensory signals for further processing. They articulated that these mechanisms efficiently reduce the processing load by focusing only on selected regions of interest. The study characterizss the attention mechanism as a critical component in managing a robot's actions, introducing three core principles for attention-driven action control:

1. Attention as action;

2. Selection-for-action; and

3. Deictic reference.

Attention as action posits that the act of directing attention towards a target is, in itself, an action rather than merely a precursor to sensory processing. This can also be described as attentional shifting, which necessitates the selection of one or more targets for action – a concept Balkenius and Hulth [78] also referred to as attentional fixation. Such fixation enables the system to maintain focus on chosen stimuli while excluding new ones, thus facilitating sustained monitoring of the attended stimuli. The identification of these stimuli is facilitated by *deictic references*, which are cues (either internal, like a target's dominant colour, or external, such as sound or sudden movement) that allow for the perception of the target without needing a comprehensive model of the object.

According to Balkenius and Hulth [78], attention mechanisms can operate under two conditions: (i) bottom-up, focusing on how a region of interest appears within its surroundings; and (ii) top-down, considering how a region of interest aligns with our objectives. An effective strategy for identifying regions of interest may involve a balanced combination of these approaches, as suggested by Rasolzadeh *et al.* [91].

Recent studies on the implementation of attentional shifting for fruit detection using visual sensor present a balanced division between the use of DL and conventional CV techniques, as indicated by Figure 2.6. This figure reports the results in a similar Venn diagram architecture to the figure 2.5, so there are a total of 85 articles using DL algorithms, 76 articles using CV algorithms, and 42 articles using ML models. While ML methods (e.g., SVM [92], partial least square (PLS) [93] [69, 68], k-means [94, 95], and wavelet [96]) are increasingly cited in the literature for fruit detection, they often complement traditional CV strategies, such as colour feature adjustments or morphological operations. It is noteworthy that ML is seldom employed in conjunction with DL models, but when it is, these algorithms aim to refine post-processing through tasks like fruit segmentation using PLS, SVM, or k-means, branch detection [97], or as part of redundancy strategies [98].

While most CV and ML algorithms are associated with segmentation techniques that nearly enable fruit harvesting, the boundary between attention mechanisms and segmentation becomes more distinct in DL approaches. DL, on high parallelisation features devices, facilitates rapid image classification, object detection, or segmentation, making it ideal for real-time applications. However, segmentation processes are generally resource-intensive and require slowing down the operation. Among the DL architectures reviewed, object detection models such as faster region-based convolutional neural network (Faster R-CNN), YOLO, and single shot multi-box detector (SSD) are prevalent, with YOLO being the most frequently used model, though Faster R-CNN is also gaining traction.

A common yet underexplored challenge in agricultural technology is the detection of green tomatoes amidst their surrounding foliage. This difficulty arises due to the close colour resemblance between the tomatoes and the plant's canopy. A comparison conducted by E Alam Siddiquee *et al.* [99] illustrates this issue well. They evaluated a ML approach known as the "cascaded object detector" against a hybrid of traditional image processing techniques

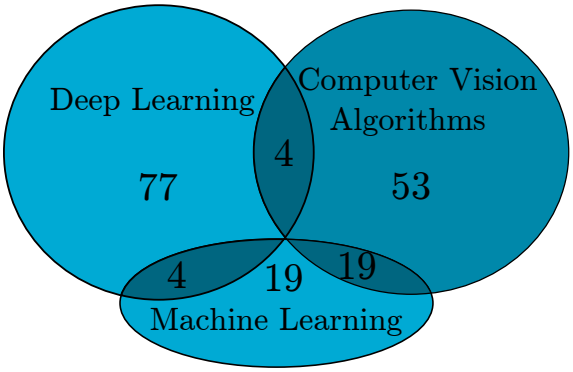


Figure 2.6: Overview of the detection on detection and segmentation algorithms on the reviewed literature

– namely 'colour transformation', 'colour segmentation', and 'circular Hough transformation' – in identifying ripe tomatoes. Their findings revealed that the ML method outperformed conventional strategies with a 95 % better accuracy rate.

For the purpose of detecting and segmenting tomatoes within a plant, researchers typically focus on the plant canopy as the primary region of interest. Within this region, other structures might obscure or overlap the fruits, complicating the detection and estimation of their locations. These challenges are particularly pronounced during the early ripening stages, where the colour similarity between the leaves and tomatoes is most significant. Despite these obstacles, most research in the field has historically focused on the later stages of tomato maturation – when the fruits are red – using colour as a key distinguishing feature [100, 101, 96, 102–106]. In attempting to differentiate fruits from their surroundings and the background, which can be pretty complex at the crop level, various colour spaces including HSI [102, 104], CIELAB [100, 101, 96], and RGB [102–105] have been utilised. Additionally, combinations of mathematical morphology [107] and ML techniques have been employed to address fruit detection in scenarios of occlusion and overlap [100–102, 104–106, 108–110, 96, 111, 112, 20].

In the context of greenhouse harvesting robots, several studies have made noteworthy contributions. Yin *et al.* [100] used K-means clustering in the CIELAB colour space to segment ripe tomatoes, achieving an average task execution time of 10.14 s. Huang *et al.* [101] applied the CIELAB colour space for segmenting and localising ripe tomatoes, employing bi-level partition fuzzy logic entropy for fruit-background discrimination, though without evaluating the algorithm's performance. Zhao *et al.* [96] developed a green-to-ripe tomato detection algorithm, extracting a* and L* components images from the L*a*b* and the luminance of the quadrature-phase (YIQ) colour spaces. These were then merged at the pixel level using wavelet transformation, with an adaptive threshold algorithm applied to distinguish the fruits from the background, achieving a 93 % detection rate.

Arefi *et al.* [102] developed an algorithm for ripe tomato recognition, leveraging a combination of RGB, HSI, and YIQ colour spaces alongside image morphological characteristics.

This approach achieved a notable accuracy of 96.36 % in greenhouse conditions with artificial lighting. Feng *et al.* [103] introduced a harvesting robot for greenhouses designed to identify and locate ripe tomatoes by transforming RGB images into the HSI colour model. This procedure was completed in 4 seconds, resulting in a harvesting success rate of 83.9 %. Similarly, Zhang [104] utilised the conversion from RGB to HSI for ripe tomato detection, employing the grey scale distribution of the H component and threshold methods for region segmentation, and the Canny operator [113] for edge detection. However, their results were not quantified.

Furthermore, Benavides *et al.* [105] designed a CV system focusing on detecting ripe tomatoes in greenhouse environments. The system primarily used the R component of RGB images and the Sobel operator [114] for fruit segmentation and edge detection, achieving a precision of 87.5 % for clustered tomatoes and 80.8 % for beef tomatoes. Malik *et al.* [106] proposed a ripe tomato detection algorithm utilizing the HSV colour space and watershed segmentation method, which effectively separated clustered fruits with a precision of 81.6 %.

Additionally, Zhu *et al.* [108] explored ripe tomato detection in greenhouses by integrating mathematical morphology with a Fuzzy C-Mean (FCM)-based method, though results were not reported. Xiang *et al.* [109] tested an algorithm for ripe cluster tomato recognition, achieving 87.5 % detection rate at 500 mm and a reduced rate of 58.4 % within a range of 300 mm to 700 mm.

In the domain of ML, Yamamoto *et al.* [110] employed various techniques to identify different ripeness stages of tomatoes, highlighting an 88 % precision in their method, which included pixel-based segmentation, blob-based correction, and X-means clustering for individual fruit detection within clusters. Zhao *et al.* [115] utilised Haar-like features and the AdaBoost classifier, complemented by colour analysis based on the average pixels value (APV), achieving a 96 % detection rate with a 10 % false negative (FN) rate and 3.5 % undetected fruits. Liu *et al.* [111] proposed a histogram of oriented gradients (HOG)-descriptor-based algorithm with a SVM classifier, integrating a coarse-to-fine scanning method and false colour removal (FCR) for false positives (FPs) elimination, culminating in a 94.41 % accuracy. Similarly, Wu *et al.* [112] developed an algorithm for a greenhouse harvesting robot, combining multiple feature analyses with an relevance vector machine (RVM) classifier and a bi-layer classification strategy, achieving a 94.90 % accuracy. Remarkably, Lili *et al.* [20] achieved a 99.3 % success rate in detecting ripe tomatoes using the Otsu segmentation algorithm [116].

In recent studies, DL has garnered significant interest, as evidenced by numerous publications [117–122]. This surge in the interest is attributed to advancements in computational capabilities, including the advent of high-performance computers and edge computing devices, such as TensorFlow processing units (TPUs), which are optimised for executing DL models. Within the realm of DL for object detection, YOLO models have emerged as a prevalent architecture [123, 124], as highlighted in several works [117, 118]. Nonetheless, other convolutional neural networks (CNNs) continue to be significant due to their precise output, despite their extended inference time, which has led to the adaptation of SSD models to mitigate these limitations [125]. These adaptations include the integration of additional layers to enhance

network resolution and modifying output layers to streamline the network for specific class detections or feature extractions.

Xu *et al.* [117] enhanced the YOLO v3 tiny method, achieving superior detection of ripe tomatoes through backbone network improvements and image enhancement techniques, resulting in an F1-score of 91.92 %, a 12 % increase over the standard YOLO v3 tiny method. Liu *et al.* [118] developed the YOLOTomato model, utilising the YOLO v3 framework augmented with a dense architecture for feature extraction and a novel bounding box, the C-box, to improve detection in moderately occluded conditions to 94.58 %, a 4 % improvement over heavily occluded conditions. Sun *et al.* [119] introduced a CNN-based detection approach utilising feature pyramid network (FPN), which surpassed traditional Faster R-CNN models by increasing detection rates from 90.7 % to 99.5 %. Mu *et al.* [120] demonstrated a model capable of detecting green tomatoes in greenhouses under occlusive conditions, leveraging a pre-trained Faster R-CNN framework with ResNet-101, fine-tuned for tomato detection on the common objects in context (COCO) dataset, achieving an accuracy of 87.83 %.

The SSD model, a variant in DL for object detection, has been recognised for its potential to significantly enhance fruit detection by efficiently performing object localisation and classification in a single step. This capability is underscored by Luna *et al.* [121], who employed both region-based convolutional neural network (R-CNN) and SSD models in a CV system designed to monitor tomato plant growth. The study found the SSD model to substantially outperform the R-CNN model, achieving a fruit detection accuracy of 95.99 % compared to the latter's 19.48 %. Yuan *et al.* [122] further validated the efficacy of the SSD model in detecting cherry tomatoes in varying ripeness stages within greenhouse environments. Through experimentation with different base networks, including visual geometry group (VGG) 16, MobileNet, and Inception v2. The Inception v2 network was identified as the most effective, reaching a detection accuracy of 98.85 %.

This synthesis of current research (table 2.1) underscores the pivotal role of DL technologies, particularly YOLO and SSD models, in advancing agricultural practices through enhanced accuracy and efficiency in fruit detection.

Table 2.1: Algorithms, methods and techniques proposed by different authors regarding tomato detection at different ripeness levels (N/A—Not Available).

Method	**Tomato Ripeness**	**Accuracy**	**Inference Time**	**Ref.**
L*a*b* colour space and K-means clustering	Ripe	N/A	10.10 s	[100]
L*a*b* colour space and bi-level partition fuzzy logic entropy	Ripe	N/A	N/A	[101]
L*a*b colour space and Threshold algorithm	Green, intermediate and ripe	93 %	N/A	[96]

RGB, HSI, and YIQ colour spaces and morphological characteristics	Ripe	96.36 %	N/A	[102]
RGB colour space images into an HSI colour model	Ripe	83.9 %	4 s	[103]
RGB colour space into an HSI colour space, threshold method and Canny operator	Ripe	N/A	N/A	[104]
R component of the RGB images and Sobel operator	Ripe	Clustered tomatoes: 87.5 % Beef tomatoes: 80.8 %	N/A	[105]
HSV colour space and watershed segmentation method	Ripe	81.6 %	N/A	[106]
Mathematical morphology and Fuzzy C-Means-based method	Ripe	N/A	N/A	[108]
Mathematical morphology, difference and iterative erosion course	Ripe	50 cm–87.5 %	N/A	[109]
Normalised colour		30 cm to 70 cm–58.4 %		
Pixel-based segmentation, blob-based segmentation and X-means clustering	Green, intermediate and ripe	88 %	N/A	[110]
Haar-like features of grey-scale image and AdaBoost classifier	Ripe	96 %	N/A	[115]
Histograms of oriented gradients and SVM	Ripe	94.41 %	N/A	[111]
Analysis and selection of multiple features, RVM and bi-layer classification strategy	Ripe	94.90 %	N/A	[112]
Otsu segmentation algorithm	Ripe	99.3 %	N/A	[20]
Improved YOLOv3-tiny method	Ripe	$F_1 = 91.92$ %	N/A	[117]
YOLOv3 detection model to create the proposed YOLOTomato model	Green, intermediate and Ripe	94.58 %	N/A	[118]
Feature pyramid network	Green, intermediate and ripe	99.5 %	N/A	[119]
Faster R-CNN structure with the deep CNN ResNet-101	Green	87.83 %	N/A	[120]
Comparison: R-CNN vs. SSD	Green, intermediate and Ripe	R-CNN: 19.48 % SSD: 95.99 %	N/A	[121]
SSD-based algorithm used to train and develop network models such as VGG16, MobileNet, Inception v2	Green, intermediate and ripe	Best performance is Inception V2 (98.85 %)	N/A	[122]

The mechanisms of attentional shifting are comprehensive algorithms that enable the simultaneous detection of multiple regions of interest. Consequently, when various regions of interest coexist, it becomes imperative to employ an additional fixation mechanism to narrow down the hypotheses under consideration [126].

In the context of strawberry greenhouses, [87, 127, 88] have applied fixation mechanisms in a bottom-up manner. Due to construction limitations, the robot prioritises harvesting strawberries from the lower regions before proceeding to those at higher levels. This strategy ensures the complete harvesting of strawberries while preventing damage to the robot or other fruits. Alternative methods include selecting the nearest fruit or opting for the region of interest with the highest confidence level of being actual fruit [78]. These approaches appear viable for harvesting various fruits, such as grapes or tomatoes. However, the prevalence of occluded fruits makes the strategies of selecting the highest confidence region of interest or the nearest fruit more effective and reliable.

Another viable fixation mechanism strategy is evaluating the fruit's maturity stage. By assessing the correct ripening stage of the target fruits, it is possible to reduce the number of region of interest. Following this assessment, strategies similar to the ones mentioned previously can be employed to focus on a single target. To address the challenge of ripening evaluation, researchers have developed methods based on colour feature analysis (Table 2.2) and DL approaches (Table 2.3).

Colour is a predominant feature used in image segmentation, particularly for differentiating ripe fruit from complex natural backgrounds, due to its distinct and consistent visual properties that are largely independent of image size. However, the effectiveness of colour segmentation may be compromised by issues such as varying lighting conditions or occlusions. To mitigate these challenges, researchers utilise various colour spaces beyond the standard RGB, including HSV, HSI, CIELAB, among others, to extract colour information from the target object, in this case, the fruit. Utilising one or multiple colour spaces, as demonstrated by Feng *et al.* [103], enhances the detection accuracy of the intended object.

2.5.1 SSD and YOLO architecture

To enhance the comprehension of the DL models discussed in the previous review and utilised throughout this PhD thesis, this section provides additional context on the DL models employed for object detection and classification. The study primarily focused on two pivotal families of one-stage detectors: the SSD and YOLO, alongside an examination of RetinaNet as an advancement of SSD, incorporating a FPN.

SSD, YOLO and RetinaNet belong to the one-step detection framework [141], characterised by a direct mapping between pixel values, bounding box coordinates, and class probabilities. This approach contrasts with the Region Proposal-Based Frameworks, such as Faster R-CNN, by offering significantly reduced inference times, enabling real-time performance.

The architecture of SSD, as illustrated in Figure 2.7, is divided into two primary components: feature extraction and object detection. The feature extraction component, or the backbone, typically utilises a state-of-the-art classification model, such as VGG 16 [142], though alternatives like ResNet [143] or MobileNet [144] are also viable options. This

Table 2.2: Results of different papers regarding tomato detection and classification through colour-based models. (N/A = Not Available).

Task	Method	No. Ripeness Classes	Results	Reference
Detection	CIELAB and K-means clustering	1 class	Inference time 10.14 s	[100]
Detection	CIELAB and Bi-level partition fuzzy logic entropy	1 class	N/A	[101]
Detection	RGB, HSV, and YIQ and Morphological characteristics	1 class	Accuracy 96.36 %	[102]
Detection	RGB colour space images into a HIS colour model	1 class	Inference time 4 s Accuracy 83.90%	[103]
Detection	RGB colour space into an HIS colour space, threshold method and Cany operator	1 class	N/A	[104]
Detection	R component of the RGB images and Sobel operator	1 class	Accuracy 87.50 % (Clustered) 80.80 % (Beef)	[105]
Detection	HSV and Watershed segmentation method	1 class	Accuracy 81.60 %	[106]
Detection	CIELAB and Threshold algorithm	3 classes	Accuracy 93 %	[96]
Classification	Aggregated percent surface area below certain Hue angles	6 classes	Accuracy 77 %	[128]
Classification	HSV colour histogram matching	5 classes	Accuracy 97.20 %	[129]
Classification	K-Nearest Neighbour based on GLCM and HSV colour space	5 classes	Accuracy 100 %	[130]
Classification	Fuzzy Rule-Based classification based on RGB	6 classes	Accuracy 94.29 %	[131]
Classification	YCbCr colour histogram	6 classes	Accuracy 98 %	[132]
Classification	Multiplication of V and Cb colour channel using Otsu thresholding	6 classes	Mean Square Error 3.14	[133]

backbone generates high-level feature maps from the input image. Additionally, SSD incorporates six extra feature maps, progressively decreasing in spatial dimensions, to facilitate the detection of objects at various scales [145, 146].

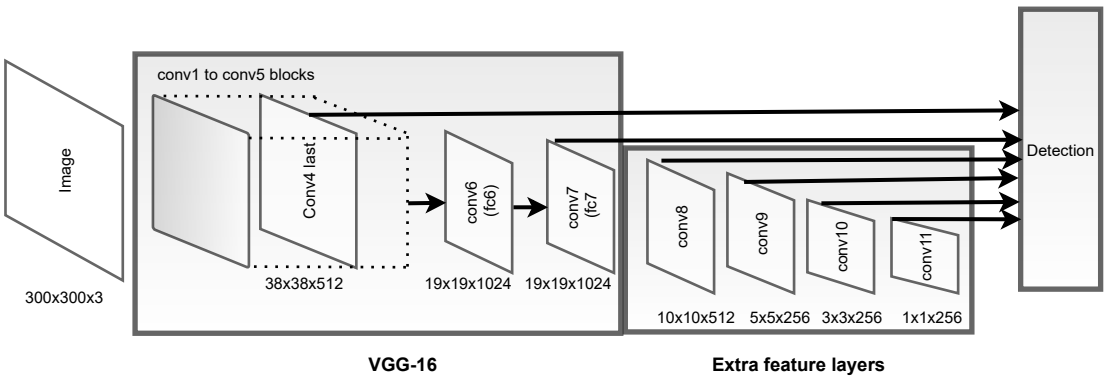


Figure 2.7: Scheme for the SSD architecture using VGG16 as the backbone. Adapted from ref. [145].

Table 2.3: Results of different papers regarding tomato detection and classification through DL one-stage detection models. (N/D = Not Described).

Task	DL Model	No. of Ripeness Classes	Results	Reference
Detection	SSD MobileNet, SSD Inception, SSD ResNet, SSD ResNet 101 and YOLO v4 Tiny	2 classes	F1-Score 66.15 % (SSD MobileNet v2)	[C42]
Detection	YOLO v4, SSD Inception v2, SSDMobileNet v2 and SSD ResNet 50	2 classes	F1-Score 61.16 % (YOLO v4)	[C46]
Detection	SSD VGG16, SSD MobileNet and SSD Inception V2	3 classes	Average Precision 98.85 % (SSD Inception v2)	[122]
Detection	Improved YOLOv3	N/D	F1-Score 94.18 %	[134]
Detection	Improved YOLOv3	1 class	Mean Average Precision 76.90 %	[135]
Detection	YOLO-Tomato	1 class	F1-Score 93.91 %	[136]
Detection	Improved YOLOv3 Tiny	1 class	F1-Score 91.92 %	[117]
Detection	YOLOv3, YOLOv3Tiny, YOLOv4, and YOLOv4 Tiny	2 classes	F1-Score 66 % (YOLOv4)	[137]
Detection	Modified YOLO-Tomato models	2 classes	F1-Score 97.90 % (YOLO-Tomato-C)	[138]
Detection	Faster-RCNN, PPN SSD MobileNet v2, RetinaNet, SSD Inception v2, YOLOv3	3 classes	Mean Average Precision 74.51 % (RetinaNet)	[139]
Classification	CCN and YOLO model	3 classes	Average Accuracy 94.67 % (YOLO)	[140]

For object detection, SSD employs a set of predefined anchor boxes (Figure 2.8) with various aspect ratios and scales, thereby constraining the potential shapes of bounding boxes [141]. A convolution layer predicts, for each anchor box and at each location on the feature maps, the location offsets and confidence scores for each class. This layer is applied to the additional feature maps and specific outputs of the backbone [145]. The fusion of predictions from multiple feature maps, each with a different resolution, enables the detection of objects of varying sizes.

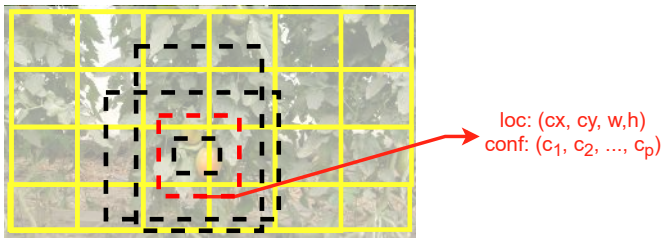


Figure 2.8: Anchor box shapes used in the SSD architecture. Adapted from ref.[145].

To refine the detection results, non-maximum suppression (NMS) algorithm is applied to retain only the highest-rated bounding boxes and reduce overlapping. Training involves a weighted sum between localisation loss (e.g., smooth L1) and confidence loss (e.g., softmax).

To enhance the performance of SSD models is recommended to adopt several strategies

tailored to the specific problem at hand. Firstly, selecting default anchors with appropriate scales and aspect ratios can significantly influence model accuracy. Additionally, data augmentation techniques should be employed to increase the diversity of training data, thereby improving the model's generalisation capability. Furthermore, implementing hard negative mining can optimise the training process by focusing on difficult examples, thus refining the model's accuracy. However, it is noteworthy that SSD models generally exhibit lower performance in detecting small objects, as these objects may not be represented across all feature maps. To address this issue, enhancements such as employing more sophisticated feature extractor backbones like ResNet, increasing input image resolution [141], or modifying the feature map layers to better accommodate the objects' sizes are recommended.

Similarly, YOLO models share a comparable architecture with SSD models, as discussed in [124]. Both architectures utilise a strategy of distributing anchor boxes (or priors) across multiple scales of the artificial neural network (ANN) for object classification. Specifically, YOLO models distribute priors across three scales, employing a logistic regression method to adjust the corners of each prior for precise localisation. Unlike SSD, which may use various backbones for feature extraction, YOLO models predominantly rely on a Darknet-based architecture for this purpose. Moreover, the loss function and backpropagation techniques employed in YOLO models differ from those used in SSD, reflecting their unique training methodologies.

2.6 Viewpoint Selection and Fruits Segmentation

Trees naturally exhibit a behaviour that allows their fruits to develop hidden behind the leaves. This adaptation protects the fruits from direct sunlight and adverse weather conditions, preventing them from becoming burnt or damaged. Such positioning of the fruits alongside the stem enhances their occlusion. To support the harvesting process, active perception methods have been developed, including viewpoint selection algorithms. These algorithms are designed to enhance the visibility of the fruit (the region of interest) and to assess the sufficiency of the gathered information for effective harvesting. Furthermore, additional segmentation strategies are essential for differentiating these regions of interest from the surrounding background, facilitating their identification and extraction.

2.6.1 Region of interest Segmentation and Assessment

Before selecting an optimal viewpoint is crucial for the system to carefully consider its target, specifically addressing the question, 'What should be perceived?"Furthermore, at each selected viewpoint, the system ought to prioritise the collection of comprehensive information from that perspective, focusing on data pertaining to fruits, peduncles, and any potential obstacles in the vicinity.

The process of gathering detailed information on fruits is predominantly facilitated by data segmentation algorithms. These algorithms, which are akin to detection algorithms, can

employ conventional image and sensory data analysis techniques, such as region growing, erosion, and dilation, alongside ML algorithms or those based on DL models. While the primary focus of this review is not to delve into the intricacies of fruit segmentation algorithms, it is pertinent to mention some notable recent works in this area for reference.

The application of CV and ML techniques plays a significant role in fruit segmentation, yielding impressive results. With the advent of state-of-the-art libraries, such as OpenCV [147], implementing these techniques has become more straightforward. Common approaches include leveraging colour features and thresholds, as well as employing corner and edge detectors like the Canny edge detector or Harris corner detection. However, these methods often fall short in 3D spaces. In classical approaches, researchers have utilised colour features and algorithms like k-means or SVM to cluster fruit pixels by colour or voxelisation based on similar principles.

The adoption of DL models represents an emerging trend in the context of segmentation, especially in 3D environments. The voluminous data and their inherent characteristics make DL an appealing choice for implementation, as it can extract more accurate and diverse features than traditional colour or shape-based methods. Popular DL models for segmentation include mask region-based convolutional neural network (Mask R-CNN) and detection and segmentation artificial neural network (DaSnet).

Among the publications reviewed, several have tackled the challenge of identifying peduncles. [148–151, 105] have made noteworthy contributions in this area. Sa *et al.* [149] developed a robotic system for harvesting peppers in greenhouses, utilising SVM for identifying the pepper's peduncle within a point cloud representation of the environment, given its proximity to the pepper. Yoshida *et al.* [150] constructed a directed acyclic graph from identified tomatoes to characterise tomato bunches, employing this graph and voxelised images to determine the precise cutting point location on the peduncle. Benavides *et al.* [105] employed a geometric and trigonometric approach to ascertain the stem's location based on the detected tomatoes' pose and size. Lehnert *et al.* [151] implemented a lightweight agricultural deep CNN to segment the stem's position within a pre-determined regions of interest.

As highlighted in section 2.4, the enhancement of environmental understanding is not limited to identifying the most informative region of interest but also extends to utilising additional sensory equipment and algorithms for assessing fruit quality. Wendel *et al.* [68], Zhao *et al.* [69], and Martins *et al.* [90] have employed spectrometry to evaluate fruit quality, paving the way for future investigations into predicting potential disease infections. Moreover, visual data can be complemented by algorithms dedicated to selective harvesting, which facilitate ripeness classification [152–154], ensuring fruits are harvested at the optimal stage of readiness.

2.6.2 Viewpoint selection

Bajcsy *et al.* [23] delineated the *where* component within the context of the pentuple (*why*, *what*, *how*, *when*, and *where*) questions, segmenting it into two distinct elements: Agent pose and sensor pose. The term 'agent pose' pertains to the strategic control and selection of the agent's (e.g., a robot or manipulator) optimal position, facilitating the acquisition of supplementary information on the targeted region of interest. Conversely, the 'sensor pose' element encompasses the proactive adjustment of the sensor's orientation (notably in devices with pan and tilt capabilities) and intrinsic settings (such as focal length, zoom, and white balance) to optimally capture the region of interest. The process of selecting a focal point, or the region of interest, is informed by the fixation mechanism (refer to section 2.5), while the management of viewpoint alterations is supported by attentional mechanisms, ensuring the object remains within tracking range without being lost.

Recent literature reviews have shown that most studies on fruit detection have utilised RGB-D sensors [C49, 155], as illustrated in Figure 2.5. Nonetheless, these studies were predominantly conducted under controlled lighting conditions to ensure the sensors' optimal performance [155]. RGB-D sensors are prone to malfunctions in open-field environments due to reflections or intense illumination [155–157]. Thus, employing auxiliary algorithms and alternative technologies is essential to mitigate the effects of lighting.

One solution to this challenge is the implementation of algorithms that enable depth perception using monocular cameras. Recent advancements have focused on CNNs to deduce this relative depth to the sensor [158–161]. Specifically, Mousavian *et al.* [161] utilized a CNN to estimate the 3D pose of objects, introducing a MultiBin loss function to optimise the model. Ma *et al.* [160, 159] developed custom CNNs, MonoPointNet and PatchNet, to generate 3D images from monocular images and detect objects. Likewise, Haq *et al.* [158] proposed a new regional proposal network (RPN) with geometric constraints for detecting 3D objects using monocular cameras, showing performance comparable to [160, 159]. Van and Do [162] adopted a chessboard background and a regression-based CNN for estimating the 3D pose of irregular objects using cuboids, although the reliance on a chessboard background limits the model's applicability in unstructured environments.

Other studies have explored the use of auxiliary sensors, like LiDARs or ToF, to create 3D scenes or for depth estimation [163]. However, high-resolution and quality LiDARs are expensive and add complexity to the operation of robotic manipulators. Therefore the observation of the objects through multiple perspectives can be an approachable solution. Perception for manipulation is often achieved through eye-in-hand techniques, with most manipulators capable of utilising either monocular or RGB-D cameras. Employing monocular cameras allows the manipulator to perceive the position of objects from multiple perspectives and estimate their pose. Approaches based on ML or statistics, such as [158–161], have been explored, though statistical approaches offer more analytical solutions with predictable outcomes.

In the realm of agricultural issues, the literature presents a scant exploration of active

viewpoint selection's contributions. Nonetheless, several studies pertinent to industrial and domestic spheres, as cited [164–166], may offer valuable insights for agricultural applications. Within the agricultural domain, notable efforts include Sa *et al.* [149], who devised a scanning strategy to enhance the visibility of peppers for harvesting purposes, enabling a clearer understanding of the fruit and peduncle positioning via a custom-designed gripper. Furthermore, Barth *et al.* [167] implemented a visual servoing strategy to maximise information acquisition about the plant through a comprehensive scan prior to initiating harvesting commands. Although this method ensures a complete plant model, it is marked by its time-intensive and computationally demanding nature. Jun *et al.* [67] employed a passive 3D perception approach for tomato harvesting, where detected tomatoes are approximated as boxes to compute their tool centre point (TCP). A hand-eye coordination scheme facilitates the gripper's navigation towards the fruit, underscoring the critical role of stem location estimation in successful real-world harvesting. The author also posits that reinforcement learning could further refine the robot's navigational accuracy. Visual servoing techniques, as employed by numerous researchers [168, 86, 169–171, 85, 167, 172, 173], primarily leverage current scene information to guide robotic actions.

Researchers such as Lehnert *et al.* [151], Arad *et al.* [35], and Wu *et al.* [174] have demonstrated the utility of employing at least two viewpoints to ensure comprehensive fruit observation. Ramon Soria *et al.* [175] developed an authentic viewpoint selection system, predicated on minimising an objective function designed to maximise the distance from the history of past poses, thereby ensuring effective convergence of the mapping process [175]. Although the approach proposed by Ramon Soria *et al.* bears similarity to the framework discussed herein (see fig. 2.4), it diverges by not accounting for the necessity of attentional systems for preliminary fruit detection, presupposing sensor alignment with the fruits. Moreover, instead of restricting focus to a singular region of interest, the methodology encompasses initial comprehensive scene modelling followed by fruit selection for harvesting, necessitating sophisticated recovery and tracking systems to monitor harvested fruits and locate all detected and modelled fruits. Implementing viewpoint selection for fruit harvesting and assessment presents more significant challenges than visual servoing, especially concerning dynamic viewpoint computations [65, 66, 176]. Nevertheless, it garners interest in scholarly literature due to its potential to capture multiple fruit perspectives, thereby facilitating the assessment of various fruit properties, including maturity classification [154]. Sarabu *et al.* [177] details a harvesting strategy employing a dual-arm robot and a multi-viewpoint approach, where one arm is tasked with fruit detection and assessment from multiple viewpoints. Upon fruit detection and selection, this arm chooses an alternate viewpoint for the same or another previously detected fruit and relays the location and harvesting strategy to the second arm for execution.

Probabilistic algorithms, capable of accurately estimating objects' poses, have also been investigated. Algorithms such as Bayesian Histogram Filters have been applied in robotics for localisation and navigation [178–180] and have shown potential in identifying and localising

various objects. For instance, Sarmento *et al.* [181] applied region histogram filters to detect people, animals, and other obstacles in ultrawideband scenes to avoid collisions and follow people. Similarly, Engin and Isler [182] utilised this algorithm for object localisation. Márton *et al.* [183] complemented a state-of-the-art position estimator with a histogram filter to accurately estimate an object's orientation.

In the domain of robotic agricultural tasks, significant advancements have been made towards enhancing the efficiency of environmental perception. Lehnert *et al.*'s [184] methodology, though not directly hinging on the concept of viewpoint selection strategy, introduces the 3D move to see (3DMTS) technique, which facilitates the continuous gathering of environmental data. This approach is characterised by constructing a bespoke matrix of cameras designed to capture the scene from multiple angles simultaneously. Each camera's output is individually assessed for quality, and based on these evaluations, the manipulator navigates through the gradient of an objective function that consolidates the ratings of all cameras.

Expanding upon this framework, Zapotezny-Anderson and Lehnert [170] unveiled an enhanced iteration of the 3DMTS algorithm, termed deep 3D move to see (Deep 3DMTS). This variant aims to augment the visibility of obscured fruits by applying DL techniques, representing a significant leap forward in the field.

In a parallel vein, Kurtser and Edan [82] delivers groundbreaking work by proposing a planning algorithm that refines both the harvest and fruit detection processes. This algorithm employs the travelling salesman problem to dynamically calculate the manipulator's path dynamically, optimising a cost function that balances data acquisition and harvesting efficiency. The algorithm continuously updates the path to include new potential harvest targets identified by the sensory system, employing an offline algorithm, like probabilitically optimal rapidly exploring random tree (RRT$\star$), to navigate between these points of interest.

Despite these innovations, the prevailing trend in the literature leans towards passive perception strategies for agricultural applications. Nevertheless, the potential for adapting these methodologies to active perception strategies is evident, as they embody the transitional phase between the two paradigms. This is exemplified by the work of Barth *et al.* [167], among others, who opted for comprehensive scene modelling despite the associated computational and temporal overheads. For instance, Kang *et al.* [185] showcase a detailed model of apple trees for harvesting purposes, employing DL for fruit detection and segmentation. The assumption that tomatoes can be approximated as spheres led to the use of the 3D spherical Hough transform (3D-SHT) for determining the apples' positions, with an octree structure providing a 3D model of both the apples and trees. Building on this, Kang *et al.* [186] utilises the PointNet model in conjunction with the octree framework to calculate the optimal grasp pose for apple harvesting.

These studies collectively underscore the ongoing shift towards more dynamic and efficient perception strategies in robotic agriculture, highlighting the interplay between innovative algorithmic approaches and practical implementation challenges.

2.7 Algorithms acceleration

The advent of AI, along with the ongoing generation of vast quantities of data, poses significant computational challenges. Traditional central processing units (CPUs) alone are insufficient for running cutting-edge AI algorithms or processing the extensive data produced by various sensors. Leading processing technology corporations, including NVIDIA, AMD, Intel, and ARM, have meticulously examined these new requirements. These companies are at the forefront, enhancing technology to provide efficient and adaptable processing solutions.

Heterogeneous computing is defined as the integration of diverse processor systems to address a specific scientific computing challenge. Such platforms consist of a variety of computational units and technologies, including multi-core CPUs, graphics processing units (GPUs), and field programmable gate arrays (FPGAs). These components offer the necessary flexibility and adaptability for a broad spectrum of application domains [187]. Utilising these computational units can significantly boost the overall system efficiency and decrease power consumption by parallelising tasks that demand extensive CPU resources over prolonged periods.

Accelerators, such as GPUs and FPGAs, are designed for massively parallel processing, facilitating the acceleration of parallelisable code segments. The integration of CPUs with GPUs and FPGAs enhances algorithm execution efficiency by allocating different computational tasks to specialised processing units. GPUs are specifically optimised for conducting matrix multiplications in parallel, a critical operation in video processing and computer graphics. However, GPUs also present certain limitations, including high power consumption and architectural constraints [188]. CNNs are inherently parallel but are not ideally suited for matrix representation, as each neuron represents a node executing a series of sequential mathematical operations. Although GPUs are highly optimised for parallel tasks, their architecture is fundamentally inspired by CPUs. Application-specific integrated circuits (ASICs) are customised designs of FPGAs, aimed at optimising and specifying operation executions. ASICs are more compact and, when designed for processing CNN algorithms, can match the speed of FPGAs. ASICs can be engineered to operate as standalone devices or be integrated with external systems.

In DL applied to visual problems, CNNs are the predominant type of ANNs. The architecture of these networks primarily comprises sequential convolutional layers trained to extract pertinent features from images. CNNs are commonly used for classification, object detection, and segmentation problems. Within the realm of object detection, the most prevalent CNN architectures include SSD [145], Faster R-CNN [189], and YOLO [190, 124]. While Faster R-CNN offers the highest precision in object detection, its two-stage processing approach results in slower inference times. Conversely, SSD and YOLO are single-shot architectures, processing the image once using feature maps to reposition object bounding boxes and perform classification. Recent studies have explored single-shot architectures for detecting fruits and other objects in open-field environments [C42, C49, 191–194], with YOLO models being par-

ticularly common due to their fast processing speeds that approach real-time performance on standard computing hardware [192], without significantly compromising performance metrics compared to other ANNs [C42]. However, challenges in detecting certain objects may necessitate larger and more capable CNN architectures. Transformers are emerging as a promising DL architecture for object detection, showing successful outcomes [193]. Despite this, many studies benchmark their findings against high-power, consumption-intensive hardware, which is not ideal for embedded or robotics applications [C42][191].

To address the limitations associated with real-time classification and power consumption, several researchers have focused on developing compact and efficient DL architectures. Notable examples include Tiny-YOLO [190, 124], YOLACT [195], and other innovative architectures [196, 144, 145]. These architectures are designed for implementation on cost-effective GPUs or even CPUs. In parallel, there is a growing interest in low-power, efficient devices capable of running parallelisable deep neural networks [197], often embodied in embedded devices. These devices span a range of types and architectures, including GPUs, FPGAs, and ASICs, with Coral TPUs and Intel neural compute sticks (NCSs) being notable examples. Another prevalent method among researchers for enhancing DL model efficiency is quantisation [198]. Typically, ANNs are trained using FP32[1] precision. However, optimisation algorithms, being iterative, tend to converge on high-resolution values that are computationally intensive and often unnecessary for classification tasks. Quantisation reduces the ANN's resolution to INT8[2] by rescaling FP32 weights, which can improve inference time and, in some cases, accuracy. Combining these strategies can yield highly efficient DL models capable of processing images at thousands of FPS.

Research has particularly emphasised embedding GPUs from the NVIDIA Jetson family, including models like NVIDIA Jetson Nano, NVIDIA Jetson TX2, and NVIDIA Jetson AGX Xavier. Zhao *et al.* [199] benchmarked two DL models, Tiny-YOLO and DNET, on NVIDIA Jetson TX2 and NVIDIA GTX Titan X, observing a minor accuracy drop (about 1 %) during the quantisation process for the NVIDIA Jetson TX2. Notably, the inference speed was approximately ten times slower on the NVIDIA Jetson TX2 (achieving 18 FPS) compared to more powerful GPUs, but it consumed 20 times less power, only about 8 W. Other studies, such as those by Suzen *et al.* [200], Chiu *et al.* [201], Rahmaniar and Hernawan [202], and Martinez *et al.* [203], have also evaluated the efficiency of DL models across NVIDIA Jetson embedded boards. The NVIDIA Jetson AGX Xavier emerged as the fastest but also the most power-intensive board. Conversely, the NVIDIA Jetson Nano was identified as less power-consuming but slower. The most frequently benchmarked DL models are from the SSD MobileNet and YOLO families, known for their smaller size, fewer convolution layers, and reduced feature retention. Martinez *et al.* [203] reported running YOLACT at 66 FPS on an NVIDIA Jetson AGX Xavier and at 16 FPS on an NVIDIA Jetson TX2, highlighting the significant hardware advancements in the latest NVIDIA Jetson board. Chiu *et al.* [201] and Rahmaniar and Hernawan [202]

[1] Single-precision floating-point
[2] 8-Bit integer

found the NVIDIA Jetson TX2 to be the fastest among the three boards when benchmarking SSD MobileNet v2, achieving 26 FPS. Suzen *et al.* [200] also included the Raspberry Pi4 in their benchmarks, but it was deemed slow and inefficient.

Despite notable improvements in power consumption, Jetson GPUs have an architecture similar to traditional NVIDIA GPUs, inheriting some limitations. As a result, researchers have begun exploring the potential of FPGAs for efficiency gains. AMD-Xilinx FPGAs, especially from the Zynq family, have been a focus. Venieris and Bouganis [204] and Chen *et al.* [205] compared a Zynq FPGA against a GPU, with Venieris and Bouganis [204] showing that the FPGA outperformed the NVIDIA Tegra X1 across multiple CNNs, achieving at least double the speed. Chen *et al.* [205] benchmarked a Xilinx ZedBoard against a NVIDIA GTX 1080Ti using the ImageNet dataset [206] and a ResNet-18 classifier. Through quantisation, they improved the network's accuracy and efficiency, running it at 20 FPS and consuming 100 times less power (only about 2.58 W). Lin *et al.* [207] compared a quantised INT8 MobileNet classifier on an FPGA's deep learning processor unit (DPU) (the main core for processing DL models) against multiple GPUs, focusing on the AMD-Xilinx ZCU104, which executed the algorithm at 376 FPS while consuming only 5 W. Zhao *et al.* [208] benchmarked an AMD-Xilinx ZCU104 against an Amazon Cloud FPGA EC2, using an YOLO INT8. Both devices achieved similar performance, with up to 13 FPS in the Penn Treebank dataset. Additionally, Jain *et al.* [209] benchmarked multiple FPGAs using a Tiny-YOLO INT8, reaching inference speeds between 12 FPS to 23 FPS on the AMD-Xilinx XC7Z035.

Researchers are actively exploring the use of specialised ASICs for executing ANNs, as these components can offer advantages in terms of cost, size, and ease of integration with other systems. Among the most notable ASICs are the Google Coral TPU and the Intel NCS. In their study, Puchtler and Peinl [197] conducted benchmarks comparing the performance of the Coral Edge TPU USB Accelerator and the Intel NCS 2 to that of an NVIDIA Jetson Nano and a Raspberry Pi 4. Their experiments utilised a SSD MobileNet v2 INT8 model, finding that the ASIC-based devices outperformed the others, with the Coral Edge TPU USB Accelerator achieving inference rates of 55 FPS and the Intel NCS 2 reaching 23 FPS. In contrast, the Raspberry Pi 4 and the Jetson Nano demonstrated significantly lower framerates of 4 FPS and 15.98 FPS, respectively. Notably, the study did not assess power consumption. Further evaluations of the Coral Edge TPU USB Accelerator's performance and efficiency have been conducted by Aguiar *et al.* [C47] and Kovács *et al.* [210].

The quest to enhance the speed and accuracy of DL models to satisfy real-time operational requirements is ongoing. However, predominant research efforts tend to concentrate on refining the architectural designs of these models rather than addressing their intrinsic characteristics, such as their capacity for high parallelisation [190, 124, 195, 196, 144, 145]. Furthermore, many studies test their algorithms on high-performance computing platforms that are not typically deployed in robotics and mobile applications [C42, C48]. While some researchers have investigated the potential of embedding these models in devices [203, 201, 202, 204, 207, 208], there remains ambiguity regarding the most appropriate device types for

specific applications.

2.8 Conclusion

This chapter conducts a comprehensive review of the scholarly literature on active perception in agricultural harvesting robots. It aggregated findings from various scientific databases and spaned a broad spectrum of topics within active perception, including fruit detection and segmentation, assessment, the selection of multiple viewpoints, visual control, and the acceleration of algorithms. The majority of the studies examined focus on fruit detection utilising visual data datasets collected in agricultural settings.

Predominantly, research on fruit detection has been conducted using detection and segmentation algorithms in 2D spaces. Nevertheless, the literature questions the efficacy of employing solely 2D sensors for successful harvesting. In practice, most studies utilising 2D sensors for harvesting purposes supplement them with depth sensors such as LiDAR or ToF [211, 212]. Regarding algorithmic approaches, the literature shows an even distribution between traditional CV strategies and DL models. Classical CV approaches, often combined with ML strategies like SVM or k-means, offer predictability through planned visual feature extraction and simpler analysis processes. However, these algorithms are slower due to their cascade implementation, which is not typically parallelised. Conversely, DL models are gaining popularity for their expressiveness, ease of training and deployment, and high feature extracting capability, albeit requiring significant computing resources (e.g., GPU, TPU, FPGA) and extensive training periods. The outcomes of DL-based models are highly data-dependent, making it challenging to discern the specific image features utilised for object prediction. Notably, most studies prioritise success rates, often overlooking the speed performance of the detection algorithms.

In terms of direct application in harvesting robots, passive perception systems predominate. However, a few studies have explored active perception systems, focusing predominantly on visual servoing. The literature acknowledges the limitations of visual servoing, particularly regarding observability and the assignment of intelligent capabilities to the harvesting system. Consequently, some researchers are adopting multiple viewpoint strategies and dynamic viewpoint planning to perceive occluded fruits and cutting points better. The integration of visual sensors and viewpoint planning not only enhances the robots' capabilities for selective harvesting but also improves maturity classification and disease detection. These advancements represent the state-of-the-art and are typically developed within controlled environments, such as laboratory testbeds or simulations.

Active perception in harvesting robots is emerging as a forward-looking direction in the literature, underscored by the advantages of intelligent viewpoint selection for fruit assessment. This approach enriches spatial information about the fruits, facilitating better harvesting outcomes and enabling the estimation of intrinsic properties, such as ripening stages.

Consequently, the next steps for implementing active perception strategies in agricultural

field robots should encompass the following:

- Developing efficient and intelligent solutions for data-seeking processes through viewpoint selection techniques;
- Integrating haptic sensors with vision (or other perceptive) sensors for enhanced effectiveness;
- Crafting smart strategies to identify the optimal grasping pose for harvesting or pruning, minimising damage to the region of interest or the plant, and;
- Implementing active searching strategies to locate fruits or other regions of interest in occluded areas, such as behind leaves, branches, or other fruits, while navigating obstacles in the most promising areas of the plants.

Furthermore, several research gaps must be addressed to ensure a robust active perception system for fruit harvesting:

- Active and dynamic viewpoint selection;
- Assessment of fruit properties;
- Detection of cutting points, and;
- Compilation of a database on fruit harvesting procedures.

This review shows that fruit detection and segmentation in the 2D space is a well-explored area. Future research should prioritise detection and segmentation in 3D spaces, identification of occluded fruits, stems, and cutting points, intelligent viewpoint selection, and fruit assessment for smart harvesting, including maturity and anomalies detection. Additionally, accelerating algorithms remains a critical focus due to their computationally intensive nature and the necessity for real-time application in robotics.

In this PhD thesis, we will explore the development and application of methodologies aimed at enhancing active perception. This exploration will involve the deployment of algorithms for detecting fruit, optimised for execution on dedicated hardware, followed by implementing information-seeking strategies through multiple viewpoint selection.

Chapter 3

Fruit perception

In this chapter, we explore various leading-edge DL object detectors for identifying fruits and other items within a cultivar environment, crucial for any robotic harvesting system. We investigate the performance of different visual system-based models and evaluate several embedded devices for running these computer vision models. Our findings demonstrate the efficiency of the selected models and confirm that diverse devices are capable of near real-time object detection, fitting for use in mobile robotic systems.

The current chapter also comprehends several studies published in scientific journals, namely [C51, C43, C48, C50, C42, C53, C47, C54, C52, C46, 213].

This chapter adheres to the standard IMRaD[1] format, delving into topics related to attentional and fixation mechanisms, paving the way for active perception.

3.1 Introduction

Effective harvesting systems depend on advanced perception technologies to accurately identify fruits and other items in agricultural environments. As discussed in chapter 1, agricultural environments are typically unstructured, containing various objects placed randomly. Furthermore, the objects of interest, namely fruits, can be at different stages of physiological maturity, complicating their detection and identification.

In chapter 2, the literature presents several methods for reliably identifying fruits and other items in these complex agricultural scenes. Initially, the focus was on traditional CV techniques that utilise colour and other features. However, due to their superior ability to recognise and categorise complex structures, DL and ML algorithms have become the preferred choice.

Despite the advantages of DL models, their use is not without challenges. Their complex, sizable architectures demand significant computing resources, necessitating powerful devices for timely image processing or risking slow performance on standard devices. In response, major tech companies have developed specialised frameworks and devices, including dedicated hardware and optimisation algorithms. These innovations aim to enable DL

[1] Introduction, materials and methods, results, and discussion

models to operate in near real-time with reduced power consumption, making them more suitable for robotic applications.

This chapter will evaluate various DL models for object detection to determine their effectiveness in fruit detection within cultivars. To support this analysis, we have created and shared datasets on the European server, Zenodo [214]. Additionally, we will explore the capabilities and limitations of different embedded heterogeneous platforms, including AMD-Xilinx FPGAs, Google Coral TPU, and NVIDIA Jetson GPUs, in processing DL models.

Because many fruits in different physiological stages can be detected simultaneously, we also study two approaches to assess their ripening stages. Colour features and DL models were benchmarked. Besides assessing their ripening stages, this proposed approach can also work as a fixation mechanism, aiding in selecting the next selectable fruit.

Subsequent sections will provide detailed descriptions of the methodologies, results of the proposed approaches, discussions, and conclusions. Additionally, the process for compiling and constructing the various datasets will be outlined.

3.2 Materials and Methods

3.2.1 Datasets generation

As we will explore later in this study, contemporary datasets like the open images dataset (OID) [215], ImageNet [206], and the COCO [216] are found to be inadequate for effectively detecting fruits and other objects within agricultural settings. To address this gap, we have compiled multiple datasets across various cultivar contexts, encompassing a diverse range of fruits and objects. In the following sections, we will outline the methodologies employed in assembling these datasets and the processes involved in their preparation.

3.2.1.1 AgRobTomato and RPiTomato datasets

To facilitate the development of a robot capable of harvesting tomatoes in greenhouses, it's critical that the robot can accurately detect the tomatoes. While there are several commonly used datasets for object detection, such as the COCO dataset [216], the OID [215], and the ImageNet dataset [206], only the OID includes images of tomatoes. However, the tomato images in these datasets do not represent the specific category we aim to identify: tomatoes in their early physiological stages on the plants in the cultivars.

To address the shortfall of relevant tomato image data in existing datasets, we created a new image dataset by collecting photographs of tomatoes from a greenhouse in Barroselas, Viana do Castelo, Portugal. The greenhouse, like others on the campus, features a layout conducive to robotic navigation: six rows of tomato plants spaced 0.9 meters apart, with the plants reaching a height of 1.10 meters (as illustrated in Figure 3.1). It's important to note that tomatoes that have fallen to the ground, indicating over-ripeness, are not suitable for harvesting by the robot.

Figure 3.1: Greenhouses' entrance.

The mobile robot AgRob v16 was employed to capture images within greenhouses, enhancing the representativeness of the collected data. This robot is outfitted with a comprehensive suite of sensors typical for robotic operations. Hence, we gathered data under the same conditions as a robot engaged in a standard harvesting operation. A human operator manually navigated the robot through the greenhouse halls, during which the robot aggregated data from its various sensors (cameras, IMU, LiDAR, etc.) into a ROSBag file. For this dataset, only RGB images were pertinent and subsequently utilised. The robot traversed along the crop row, maintaining a distance of 0.4 m to 0.6 m from the tomato plants.

Equipped with two stereo cameras, the AgRob v16 utilised the front camera primarily for localising itself within the hall. For harvesting tomatoes, we implemented an eye-in-hand strategy, enabling continuous refinement of the robotic arm's position relative to the tomatoes through active perception or gaze control mechanisms. So, the second stereo camera (ZED[2]) was mounted on an anthropomorphic manipulator at the robot's rear, which was kept in a fixed position looking sideways towards the tomato plants throughout the data acquisition phase. A Jetson Nano[3] GPU connected to the ZED camera on the robotic arm managed the image capture and processing. Subsequently, the GPU forwarded the images to the robot's onboard computer to integrate them with other collected data.

In summary, the robot recorded video images of the tomato plant wall, storing them in a ROSBag file. This phase yielded a raw data set that required processing to become a usable dataset.

DL models, categorised under supervised ML algorithms, necessitate training with an annotated dataset. In object detection, annotations typically include a bounding box detailing

[2]See Stereolabs Inc. "ZED." (2020), [Online]. Available: https://www.stereolabs.com/zed/ (visited on 11/25/2020).

[3]See NVIDIA Corporation. "Jetson Nano developer kit." (2020), [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (visited on 08/05/2022).

the object's class, size, and position. Some formats, like the Pascal VOC [219] format, also capture additional object characteristics, such as difficulty in detection, occlusion, or truncation. For this project, we employed the Pascal VOC format for its simplicity, summarising the annotations for each image in a single XML file. The additional features were omitted as the TensorFlow1 Object Detection Pipeline, the framework used for training the DL models, does not utilise these details.

Initially, we converted the continuous video of tomato images into individual sequential frames. To reduce redundancy among images in the dataset, a frame was captured every three seconds (3 FPS), ensuring an overlapping ratio of about 60 %. This approach produced a dataset comprising 297 images, each with a resolution of (1280×720) px.

All images were manually annotated using the CVAT[4] [220] or LabelImg [221] tools, which enhance the management of images and annotations and support collaborative annotation. For this dataset, we focused exclusively on the 'tomato' class, disregarding the stage of ripeness, due to the predominance of either reddish or green tomatoes in the dataset, as shown in Figure 3.2.

(a) Green tomato (b) Reddish tomato (c) Red tomato

Figure 3.2: Tomatoes' ripeness levels: (**a**) physiological or horticultural maturation; (**b**) early phase of ripening; and (**c**) ripened tomato.

We aim to leverage SSD models [145] and the tiny variants of YOLO [124] models for the prompt detection of tomato fruits online, utilising either standard GPU hardware or specialised computing devices. Our approach involves employing fine-tuning methods to obtain the trained models. However, the pre-trained models available in the TensorFlow1 Model Zoo are not capable of processing images at their full size, necessitating a preliminary step of image rescaling. For instance, the pre-trained SSD MobileNet v2 model [146] is designed to process images resized to (300×300) px by default. Consequently, we partition the original images into (300×300) px segments, according to the procedure illustrated in Figure 3.3, utilising the `pascal_voc_tools`[5] for this task. This segmentation strategy ensures a minimum overlap of 20 % between adjacent images. By dividing full-sized images of (1280×720) px into (300×300) px segments, we not only increase the size of our dataset to 5365 images but also

[4] Computer vision annotation tool

[5] See W. Tengfei. "Pascal VOC tools." (Mar. 16, 2021), [Online]. Available: https://github.com/wang-tf/pascal_voc_tools (visited on 09/08/2021).

enhance the quality of the images used for training.

Figure 3.3: Images split into (300 × 300) px images with an overlapping ratio of 20 %. The different colours are only for reference and distinguishing the different images.

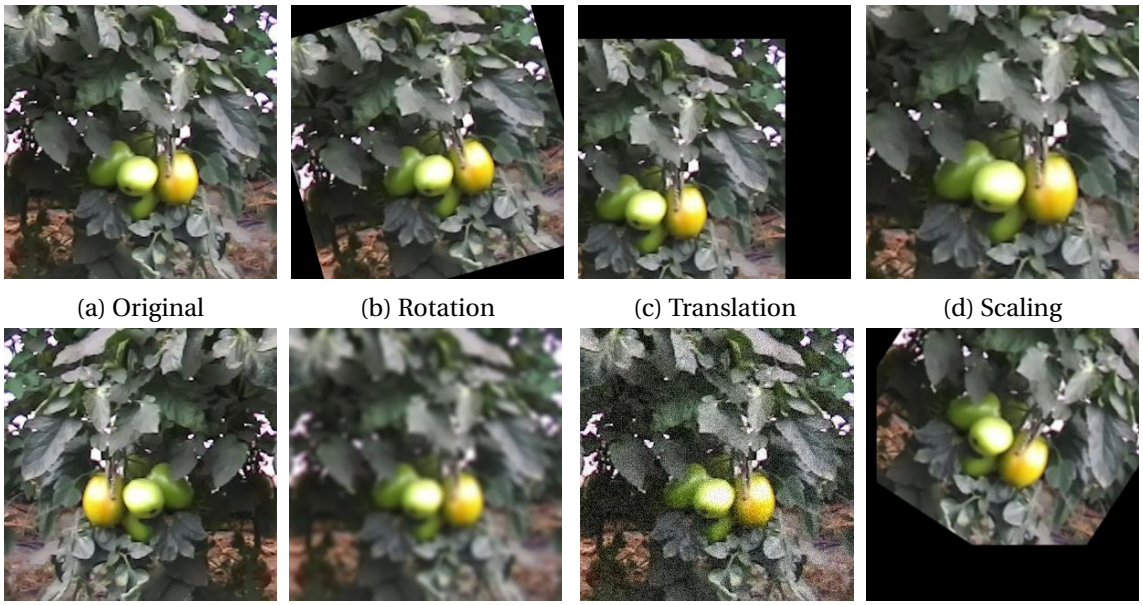
Several studies have shown that augmenting original images by applying various transformations can enhance the size and variability of datasets, thereby enriching them with new information [223, 224]. These transformations can include: (a) rotation; (b) translation; (c) scaling; (d) hue modification; (e) saturation; (f) blur; (g) noise; (h) others, even combinations of these transformations.

The diversity of transformations applied to images and their specific details are summarised in Table 3.1. All transformations were implemented randomly to ensure increased data variability. An example of how augmentation has been utilised is shown in Figure 3.4. This process resulted in a dataset comprising 23021 images, summed in 61204 descriptions of tomatoes.

For training, the dataset was split into two parts: one for training and the other for validation. The training set consisted of 18417 images with 49100 annotations, whereas the validation set included 4604 images with 12104 annotations. An external set of annotated images, acquired under similar conditions but in a different row within the tomato greenhouse, was used for model evaluation and testing. Initially, this set contained 250 full-sized images ((1080 × 720) px). However, to align with our methodology, these were not augmented but instead divided into smaller segments of (300 × 300) px, resulting in 2737 images.

The complete dataset contained 25758 images and has been made publicly available through the INESC TEC Research Data Repository [C50].

Tomatoes are a type of fruit that grows over a continuous period, leading to their ongoing harvesting based on their maturity stage.



(a) Original (b) Rotation (c) Translation (d) Scaling

(e) Flip horizontally (f) Blurring (g) Noising (h) Combination 3

Figure 3.4: Example of augmentation applied to an image. (**h**) is the random combination of 3 of the other transformations.

Table 3.1: Transformations applied to the images of the split dataset for data augmentation and the characteristics of those transformations.

Transformation	**Value**
Rotation	$-60°$ to $60°$
Scaling	50 % to 150 %
Translation	0 % to 30 % left or right
Flip	Image mirroring
Blur (Gaussian Filter)	$\mathcal{N}(0, 1 \text{ to } 3)$
Gaussian Noise	$\mathcal{N}(0, 0.03 \cdot 255 \text{ to } 0.07 \cdot 255)$
Combination3	Random combination of three of the previous transformations with random values

Because of that, an additional set of tomato images was collected to evaluate the potential of CV techniques in determining the fruits' ripeness. These images were taken from a separate greenhouse in Amorosa, Viana do Castelo, Portugal, on June 15th, 2021. A selection of 60 tomatoes at various stages of ripeness was made, and RGB images of each tomato were captured from multiple angles. The photography equipment used included a Raspberry Pi 4 Model B[6] equipped with 4 GB of random-access memory (RAM), and a Raspberry Pi High-Quality Camera[7], which features a 12.3MP sensor and a 7.9 mm diagonal image size. At-

[6]See Raspberry Pi, Ltd. "Raspberry Pi 4 model B." (2021), [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (visited on 08/15/2021).

[7]See Raspberry Pi, Ltd. "Raspberry Pi high-quality camera." (2021), [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/ (visited on 08/15/2021).

tached to this camera was a 6 mm (wide angle) CS-mount lens with a 3ṀP resolution (see Figure 3.5b). In total, 258 images were captured, forming the basis of the RpiTomato Dataset [C52]. These images were carefully framed to maximise the use of the image space, eliminating the need for image splitting.

(a) (b)

Figure 3.5: AgRob v16 (**a**) and the Raspberry Pi High Quality camera (**b**) used for image collection.

To assess the maturity levels of the fruits, our approach focuses exclusively on classification strategies rather than regression techniques. For the purpose of categorising fruits by their stages of ripeness, we have established four distinct classes. These classes are derived from the colour chart for fresh tomatoes provided by the United States Department of Agriculture (USDA) [227], as illustrated in Figure 3.6.

To enhance and expand the RPiTomato Dataset, we integrated it with the Tomato Dataset, selecting all images from the latter and excluding only the green tomato images from the RPiTomato Dataset. This strategy was implemented to maintain a balanced representation of images, given the ample number of green tomato images already present in the Tomato Dataset. As a result, a total of 632 images were chosen for inclusion.

In a manner analogous to the approach used for the Tomato Dataset, as documented in [C50], all images from both the Tomato and RPiTomato Datasets were manually annotated. This was accomplished using the open-source annotation tool CVAT, where rectangular bounding boxes were drawn to indicate the position and classify the ripeness of each tomato into one of four stages: “unripe”, “breaking stage”, “reddish”, and “ripe”, as depicted in Figure 3.6. The dataset was formatted in the Pascal VOC standard for consistency.

Due to the significant computational demands of high-resolution DL models, which are inefficient processing full-sized images, and given their requirement for square image inputs, the original images from the Tomato Dataset were segmented into smaller frames of (720×720) px. Initially, images from the RpiTomato Dataset were resized to (960×720) px and

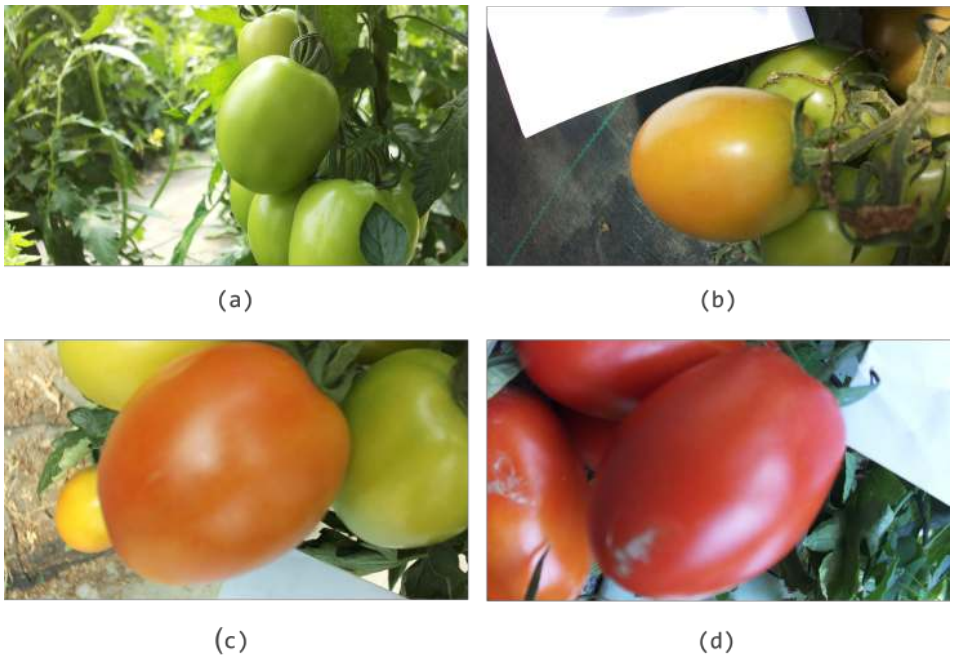


Figure 3.6: Classes defined according to the colour of tomato during ripening: Green (**a**)—more than 90% of the surface is green; Turning (**b**)—10 to 30% of the surface is yellow; Light Red (**c**)—between 60 to 90% of the surface is red; Red (**d**)—90 to 100% surface is red.

then split to achieve a uniform resolution of (720×720) px. This splitting operation doubled the number of images from the Tomato Dataset, increasing the total to 1081 images. However, after discarding images without annotations resulting from the split, the final count was adjusted to 1029 images.

The reprocessed dataset RPiTomato dataset created an updated AgRobTomato Dataset [C51]. Further, it was divided into three sets: a training set (60 %), a validation set (20 %), and a test set (20 %).

To artificially enhance the dataset and improve the model's learning efficiency and performance, data augmentation techniques were employed. These techniques were selectively applied to the training and validation sets, incorporating variations that could realistically occur during robotic harvesting operations. This approach, illustrated in Figure 3.7, aimed to introduce a diversity of data to the model, enhancing its robustness and effectiveness in real-world applications.

The data augmentation process resulted in a total of 7598 labelled images. Specifically, the training set comprised 5543 images, the validation set included 1850 images, and the test set consisted of 205 images.

Furthermore, the AgrobTomato Dataset [C51] and the RPiTomato Dataset [C52] have both been made available to the public, each being released separately.

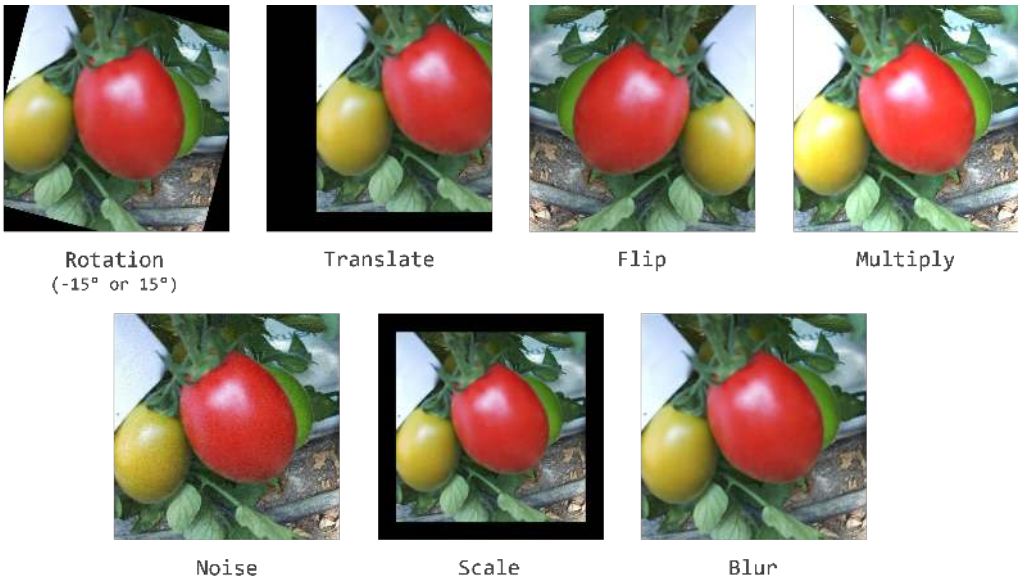
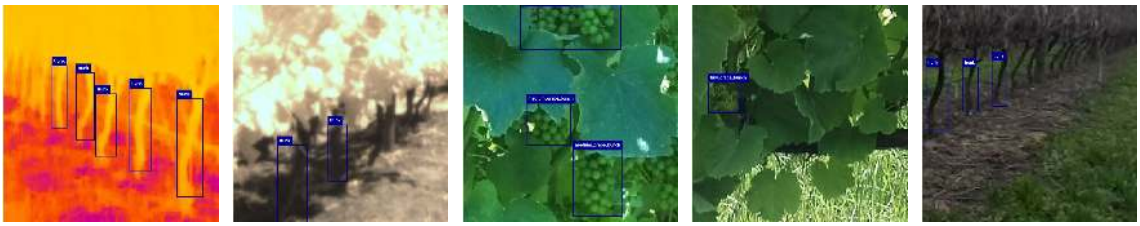


Figure 3.7: Different types of transformations applied to the images of AgRobTomato + RPiTomato Dataset.

3.2.1.2 VineSet Dataset

To enhance the research scope of this PhD Thesis and align with ongoing projects at the INESC TEC TRIBE laboratory, we developed an additional dataset named VineSet [C53]. VineSet comprises 428498 images with dimensions of (300×300) px. Each image in the dataset has been manually labelled and sourced from a variety of cameras including stereo, high-quality, and thermal imaging devices. VineSet features natural vineyard scenes categorised into three distinct classes: vine trunks, bunches of berry-corn size grapes, and bunches of berry-closed grapes. As illustrated in Figure 3.8, the dataset showcases diverse images from these categories. VineSet is a comprehensive collection that combines the VineTrunk dataset [228] – which focuses on vineyard trunks and utilises various visual data sources – with newly acquired images of grape bunches at different stages of growth.



(a) (b) (c) (d) (e)

Figure 3.8: Sample of images in the dataset [C53] with the respective ground truth bounding boxes in blue squares. (a) Thermal image of vines' trunks; (b) image of vines' trunks without infra-red filter; (c) image of bunches of medium-size grapes; (d) image of bunches of corn-size grapes; (e) image of vines' trunks.

The dataset was divided into three segments: a training set (411360 images), a validation

set (8569 images), and a test set (8569 images). To align the test set more closely with real-world data, augmentation images were removed, leaving 1125 images in the test set.

This dataset expansion seeks to enhance the VineTrunk dataset's visual diversity, focusing on detecting grape bunches at various fruit growth stages. To capture these stages, numerous data acquisition experiments were conducted with the AgRob v16 robot, documenting the grape bunches' development across different weeks. The robot traversed the same path in the vineyard (Quinta da Aveleda, Portugal) multiple times, during different times of the day and on separate days, to collect visual data that captures the fruits' growth stages and accounts for significant variations in lighting. Initially, the robot captured images of berry-corn size grapes, which are small, light green berries that appear post-bloom, measuring approximately 5 mm. Subsequent experiments focused on medium ripening stages, featuring berry-closed grapes with a regular green color and diameters around 12 mm.

The robot was equipped with two RGB monocular cameras mounted on an anthropomorphic manipulator, calibrated to ensure a consistent view of the vine canopy. The cameras used were the Raspberry Pi High-Quality Camera – Sony IMX477 and the OAK-D[8] colour camera, with its depth sensors deactivated.

The data collection was conducted in video format (H264), from which single images were extracted and stored. The videos were sampled at 1 FPS, minimising excessive overlap and redundancy between sequential images, targeting at least 60 % overlap, similar to the approach for the Tomato Dataset. This process yielded 1929 original images, showcasing various fruit growth stages.

The collected data remained unprocessed by rectification or calibration to ensure models trained on this dataset would work with raw images.

For training supervised learning models, the data underwent a labelling process for object detection, marking regions of interest with bounding boxes. This process used the CVAT software for visual annotation, identifying berry-corn size grapes as `tiny_grape_bunch` and berry-closed grapes as `medium_grape_bunch`, following the Pascal VOC challenge protocol [219]. Samples of the annotated dataset for these classes are depicted in figures 3.8c and 3.8d.

To enhance data diversity and mitigate potential overfitting, the dataset was augmented using five operations, including a double application of rotation operations. This adjustment reflects the reality of camera vibrations and the natural variability in fruit orientations, except for upside-down growth. Consequently, the augmented dataset expanded to 13503 images.

The final phase in the data generation process involved segmenting the images, in a similar strategy to the AgRobTomato dataset. Our dataset comprises full high-definition (HD) images, which exceed the input size limitations of the models available in the TensorFlow1 Model Zoo. Consequently, it is necessary to resize our images to (300×300) px. However, considering our images are full HD, resizing could result in the loss of significant features and details. To mitigate this, we employed a strategy where the augmented data was divided into

[8] See OpenCV AI and Luxonis Holding corporation. "OpenCV AI kit: OAK-D." (2023), [Online]. Available: https://store.opencv.ai/products/oak-d (visited on 09/22/2023).

Table 3.2: Description of the augmentation operations used to expand the original collection of data.

Augmentation Operation	**Description**
Rotation	Rotate the image by 30° and −30°.
Translation	Translates the image by −30 % to 30 % on the x- and y-axis.
Scale	Scale the image to a value of 50 % to 150 % of their original size.
Flipping	Mirrors the image horizontally.
Multiply	Multiplies all pixels in an image with a random value sampled once per image, which can be used to make images lighter or darker.

smaller sections, matching the network's input dimensions. Additionally, to ensure no critical information was lost between these sections, we implemented an overlap of 30 % between them, as illustrated in Figure 3.9.

Table 3.3: Number of annotated objects per class. The original dataset contains 1929 images with two different classes. To increase the dataset size, several augmentation operations were applied, increasing the number of images to 13503. Finally, the images were split, and the final dataset was composed of 302252 images.

Class	**# of Objects**	**# of Objects in Augmented Images**	**# of Objects in Split Images**
`tiny_grape_bunch`	2497	13393	25349
`medium_grape_bunch`	4292	25189	51272

The table 3.3 provides summary information about the number of labelled objects for the newly added classes, namely, `tiny_grape_bunch` and `medium_grape_bunch`, at three different stages of the dataset: original images, augmented images, and split images. After the splitting process, the total number of images we obtained with dimensions of (300×300) px was 302252.

This new dataset, after being collected and processed, has been merged with the VineTrunk dataset [228]. The VineTrunk dataset was acquired and processed using a similar protocol, so even though the visual data comes from different sources, all the images share similar features and processing.

Furthermore, the VineSet dataset [C53] has been made publicly available.

3.2.1.3 Red grape chunk classification (RG2C) dataset

Research in the literature suggests that small-sized ANN operating on dedicated hardware can significantly accelerate the inference process. To investigate this, the new dataset red grape chunk classification (RG2C) was compiled. The RG2C dataset consists of image segments depicting bunches of red grapes in steep-slope vineyards, classified in a binary manner.

Figure 3.9: Split of a collected image to a (300 × 300) px resolution. The original image with a resolution of (1920 × 1080) px generated 40 split images considering an overlapping ratio of 20 %.

These images were captured using an RGB camera at Quinta do Seixo, Valença do Douro, Portugal, in video format. Unlike previous datasets, an operator utilised a smartphone POCO F2 PRO[9] and manually navigated through the vineyards, recording video samples of the canopy with mature grape bunches. All frames were captured as RGB images, allowing for some sequential and repetitive images. At present, the dataset includes 1198 images, each with a resolution of (1920 × 1080) px.

Subsequently, each image was divided into (32 × 32) px segments, as illustrated in Figure 3.10. This segmentation process, performed without overlap, yielded a total of 10782 images.

In line with previous studies, supervised learning techniques were applied. Here, a binary classification method was used, classifying the entire image segment as either containing grapes or not, rather than identifying individual grapes within a bounding box. The dataset was organised in an Image Directory structure as shown in Figure 3.11.

The labelling process was conducted in two phases. Initially, a colour threshold technique was used to sort images into their respective class folders (`grape` or `no_grape`), taking advantage of the distinct colour of the red grape bunches against the background. However, some mislabelling occurred, necessitating a manual verification process to correct inaccurately labelled images. An image was classified as belonging to the grape category if it contained at least 25 % of a grape bunch or part thereof.

Finally, the dataset was split into three sets considering the acquisition sequence of the

[9]For details on the characteristics, see GSM Arena. “Xiaomi POCO F2 Pro.” (Apr. 24, 2024), [Online]. Available: https://www.gsmarena.com/xiaomi_poco_f2_pro-10220.php (visited on 05/16/2024).

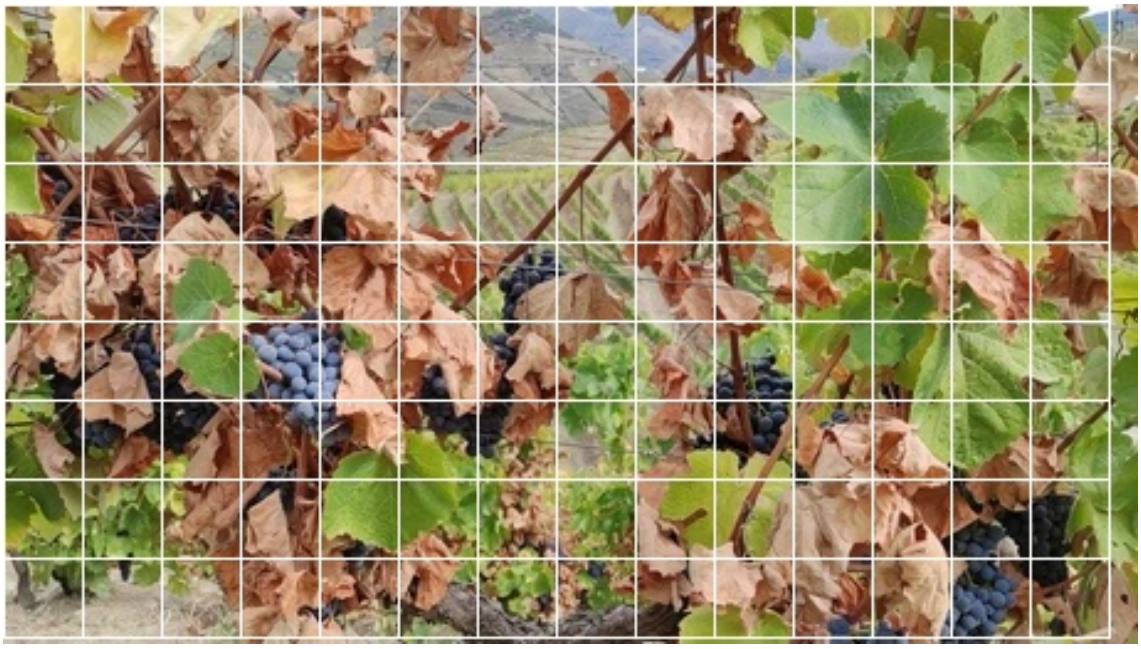


Figure 3.10: Illustration of the image segmentation scheme for the RG2C dataset.

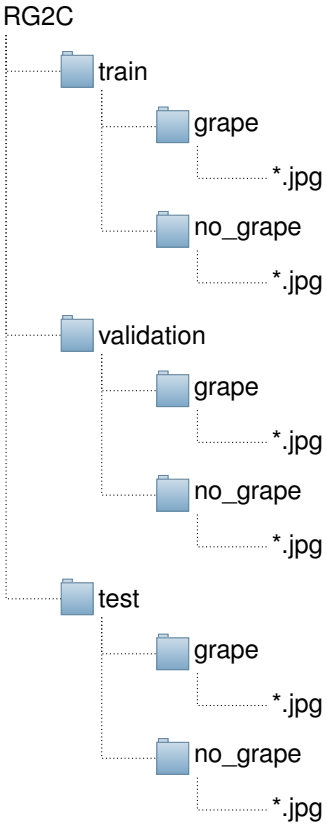


Figure 3.11: The image directory structure for the RG2C dataset.

images: train (60 %), validation (20 %), and test (20 %). The train set contains 6470 images, the validation set 2156 images, and the test set 2156 images.

The RG2C dataset [C54] was made publicly available.

3.2.2 Fruit detection models

In the chapter 2, various strategies for detecting fruits and other objects in images are reviewed. Initially, the focus was primarily on classical image analysis procedures and simpler ML algorithms. However, with the advancement of technology and scientific knowledge, DL has emerged as the state-of-the-art for image analysis and object detection. Consequently, we explored several DL models using the Tomato Dataset [C50] for the case study. Additionally, as part of a benchmark study, we compared the performance of networks trained on this dataset with others trained on the OID, which, despite containing multiple classes, including tomatoes, does not specifically focus on the agricultural context. Our goal is to evaluate their effectiveness in identifying tomatoes in cultivar settings.

The leading frameworks for DL include TensorFlow [231], Darknet [123], and PyTorch [232], with TensorFlow being notably prominent for its scalability and support for a wide range of ML algorithms, particularly DL. As an open-source platform developed by Google, TensorFlow can operate across diverse applications and devices, either in centralised or distributed systems.

During the current study's model evaluation phase, only TensorFlow 1 provided partially compatible tools for training and compiling models for use with the TPU. Therefore, we employed TensorFlow r1.15.0 for both training and inference, utilising Colab[10] notebooks[11]. These notebooks offer complimentary access to powerful GPUs and TPUs for training and inferring DL models. Although the available GPUs could vary with each Colab session initiation, the NVIDIA Tesla T4, featuring a video random-access memory (VRAM) of 12 GB and a 7.5 computation capability, consistently served as the allocated GPU throughout our sessions.

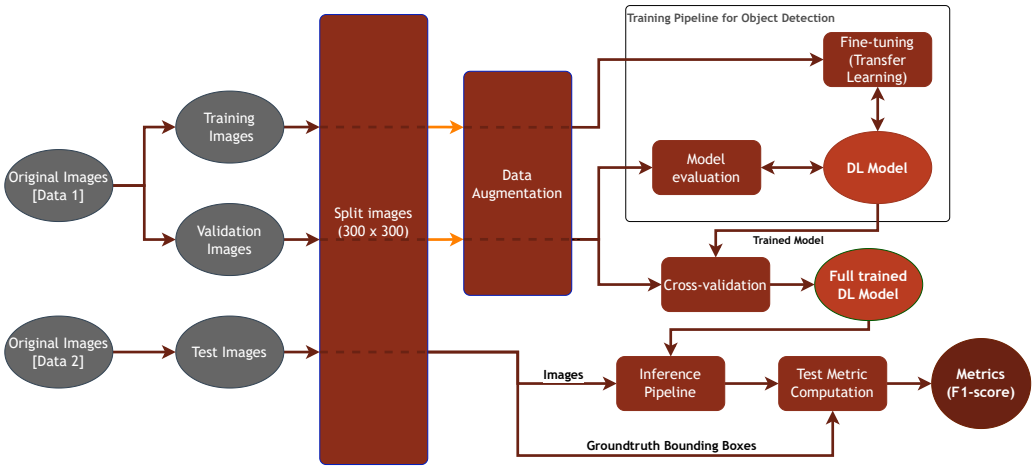


Figure 3.12: Overview of the performed methods. Training and evaluation pipeline.

[10] Google Collaboratory

[11] See Google LLC. "Google Colaboratory." (2023), [Online]. Available: http://colab.research.google.com (visited on 09/19/2023).

For benchmarking purposes, we evaluated four pre-trained SSD models from the TensorFlow database[12], as specified in Table 3.4: SSD MobileNet v2, SSD Inception v2, SSD ResNet 50, and SSD ResNet 101, in addition to the YOLOv4 Tiny model. The first three SSD models and the YOLOv4 Tiny were pre-trained on the COCO dataset, while the SSD ResNet 101 was pre-trained on the OID. We fine-tuned these pre-trained models on the Tomato Dataset [C50], over the strategy detailed in Figure 3.12, adhering to the default settings of the models' pre-training pipelines, but adjusted the batch size to suit the capabilities of the available GPU, as detailed in Table 3.5. All training sessions were conducted over 50000 steps, with evaluations every 50 steps to ensure no more than 50000 steps were necessary for convergence to the optimal solution in the solution space. In some instances, models achieved convergence after just 30000 steps. This frequent evaluation helped us monitor training progress and prevent overfitting, which was not observed in any of the trained networks.

Table 3.4: Model location in TensorFlow and Darknet databases. All SSD models are in the TensorFlow Models database at `http://download.tensorflow.org/models/object_detection/`<filename>. YOLOv4 Tiny is in the Darknet database at `https://github.com/AlexeyAB/darknet/releases/download/`<filename>.

SSD Model	**File Name**
SSD MobileNet v2	ssd_mobilenet_v2_coco_2018_03_29.tar.gz
SSD Inception v2	ssd_inception_v2_coco_2018_01_28.tar.gz
SSD ResNet 50	ssd_resnet50_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03.tar.gz
SSD ResNet 101	ssd_resnet101_v1_fpn_shared_box_predictor_oid_512x512_sync_2019_01_20.tar.gz
YOLOv4 Tiny	darknet_yolo_v4_pre/yolov4-tiny.conv.29
YOLOv4	darknet_yolo_v3_optimal/yolov4.weights

YOLO v4 and YOLO v4 Tiny are not available for the TensorFlow framework, but they are available for the Darknet framework. So, despite, at the time, they cannot be deployed for TPU, they are referenced in the literature and have aided in establishing our baseline results. The YOLO v4 Tiny model learned more quickly, requiring only 2500 steps for the training session. Darknet did not offer any available validation sessions, so it was not considered.

In this experiment, our goal was to evaluate the ability of various DL models to detect tomato fruits. We trained the models SSD MobileNet v2, SSD Inception v2, SSD ResNet 50, and YOLO v4 twice for comparison: first on a specially gathered agricultural dataset of tomatoes [C50], and second on a subset of the OID.

The experiment aimed to assess the performance of these DL models in detecting tomato fruits. However, initially, the models outputted all detected objects without considering the detection confidence. Thus, a crucial part of our training pipeline involved adjusting the con-

[12] See TensorFlow. "TensorFlow 1 detection model zoo." (Oct. 14, 2021), [Online]. Available: `https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md` (visited on 09/19/2023).

Table 3.5: Training batch size for each model.

SSD Model	Batch Size
SSD MobileNet v2	24
SSD Inception v2	32
SSD ResNet 50	8
SSD ResNet 101	8
YOLOv4 Tiny	64
YOLOv4	64

fidence rate -— a value indicating the model's certainty in its predictions. A confidence rate of 50 % means the network is 50 % sure of its detected or classified object. Higher confidence rates generally indicate more robust ANN performance, though lower rates can still yield true positives. Optimising this confidence score is vital for enhancing network efficiency. We employed a cross-validation technique to tune this hyper-parameter. On the validation set, after removing all augmentations, we computed the F1 score for confidence thresholds ranging from 0 % to 100 % in 1 % increments. A confidence threshold includes all rates greater than or equal to it. The threshold that optimised the F1 score was selected for the model's standard operation.

SSD MobileNet v1 The SSD MobileNet v1 is a popular model among the state-of-the-art models that are designed to run on low-power embedded devices. This model, introduced by Howard *et al.* [196], uses depth-wise separable convolutions, which is achieved by factorising standard convolutions into depth-wise and 1×1 convolutions that are then combined.

The input of this CNN is a tensor with shape $D_f \times D_f \times M$, where D_f represents the input channel spatial width and height, and M is the input depth. After the convolution, a feature map of shape $D_f \times D_f \times N$ is obtained, where N is the output depth.

The model contains two hyper-parameters that can be tuned to optimise the CNN performance. The first hyper-parameter has a multiplier α, which can be used to uniformly decrease the model size at each layer by a factor of α^2. This is performed by multiplying the number of both input and output channels by a constant. The second hyper-parameter, resolution multiplier ρ, is used to reduce the computational cost of the model by a factor of ρ^2 by changing the input image resolution accordingly. Both parameters can be used simultaneously to balance the performance and inference time of the model.

SSD Inception v2 The approach of Inception was first developed by Szegedy *et al.* [235]. The design of the model was based on the assumption that objects in different images can have different sizes, making it difficult to choose the right kernel size for the CNN. To overcome this, the authors created the model with three convolutional filter sizes: 1×1, 3×3, and 5×5. The results from these operations were then combined, resulting in the network's output.

To reduce the computational complexity of the original version, SSD Inception v2 [236]

was developed. This was achieved using factorisation over the convolution operations. For example, a 5×5 convolution was factorized into two 3×3 convolutions, which improved runtime performance. Similarly, a $m\times m$ convolution can be factorised into a combination of $1\times m$ and $m\times1$ convolutions.

SSD ResNet Residual Networks were introduced by He *et al.* [143]. The authors base their work on plain networks, drawing inspiration from principles like VGG [142]. Plain networks essentially consist of sequential convolution filters that process the data. The base plain network's head is composed of an average pooling layer, followed by a 1000-unit dense layer using the softmax activation function.

For the residual network, He *et al.* [143] introduced a shortcut in the plain network between each pair of convolution layers, creating its residual counterpart. The number associated with the identification of the residual network, such as ResNet 50 or ResNet 101, indicates the depth of the networks, i.e., the number of convolution layers in the network.

YOLO v4 YOLO as introduced by [190], stands as the forefront technology in one-stage deep neural networks for object detection, adopting a principle similar to SSDs. The evolution to YOLO v4 is marked by the work of [237], which retains the architectural essence of its predecessors, notably YOLO v3 as documented by [124]. It is built upon a Darknet backbone, specifically, CSPDarknet53 [238] and integrates both an spatial pyramid pooling (SPP) layer, as per [239], and a path aggregation network (PANet) structure from [240] in its neck. The head mechanism continues to mirror that of YOLO v3, maintaining a consistent approach to object detection.

More detailed information of the overall architecture of YOLO models is available in the section 2.5.1.

The present protocol aims to investigate the architecture of certain ANNs for fruit detection in natural contexts. It is worth noting that these studies were previously conducted using high-performance devices, which are not ideal for mobile robotics. While the results section shows promise, low-power systems, like mobile robots, still require dedicated hardware devices. As a result, the next phase of the protocol seeks to address this acceleration issue and explore potential strategies to overcome it.

3.2.3 Acceleration of fruit detection models

In section 2.7, we reviewed the computational and resource demands of using DL algorithms. Efforts to enhance network efficiency have led researchers to adjust the foundational architecture of ANNs. However, these modifications typically result in a trade-off, with a de-

crease in performance for target detection tasks. Notably, due to their superior parallelisation capabilities, employing dedicated hardware for running ANNs has emerged as a viable alternative.

Furthermore, in the literature review, we highlighted the advantages of utilising FPGAs for executing algorithms that may benefit from parallelisation. This parallel processing can be approached at two distinct levels: through DPU or by direct implementation on the FPGA's programmable logic (PL) layer. The limitations inherent to the PL layer of the FPGA necessitate a bifurcated approach to our research. Initially, we will conduct an experiment assessing the performance of VineSet with the RetinaNet ResNet 50 on various heterogeneous platforms. Subsequently, we will explore the deployment of simpler ANN models using the RG2C dataset on the FPGA's PL.

3.2.3.1 VineSet and RetinaNet ResNet 50 in heterogeneous platforms

In this study, our objective is to evaluate the performance of the VineSet dataset (referenced in section 3.2.1.2) using a RetinaNet model with a ResNet 50 backbone to achieve near real-time processing capabilities. The RetinaNet ResNet 50 model is an enhancement over the standard SSD ResNet 50 model, incorporating an additional FPN for improved performance. This model has been developed and processed using TensorFlow 2.8[13] and Keras[14].

To facilitate the deployment of this network across a variety of heterogeneous platforms and determine the most suitable platform for our needs, we have utilised additional frameworks to optimise the models for specific platform architectures. These include Vitis-AI 1.4[15], the Edge TPU Compiler[16], and TensorFlow TensorRT (TF-TRT)[17].

Heterogeneous Platforms This part of the research focuses on evaluating heterogeneous platforms to identify devices that offer faster inference speeds while minimising accuracy loss. It involves a comparison of three embedded GPUs – each with processing capacities of 1000 TFLOPS and 2000 TFLOPS (specifically, the Jetson Nano 2 GB[18], Jetson Nano 4 GB[19], and TX2[20]) –, DPUs (utilising FPGAs, identified as AMD-Xilinx ZCU104 and AMD-Xilinx Kria

[13] See TensorFlow. "TensorFlow." (2022), [Online]. Available: https://www.tensorflow.org/ (visited on 08/05/2022).

[14] See Keras. "Keras." (2022), [Online]. Available: https://keras.io/ (visited on 08/05/2022).

[15] See Advanced Micro Devices, Inc. "Vitis-AI." (2022), [Online]. Available: https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html (visited on 08/05/2022).

[16] See Google LLC. "Edge TPU compiler." (2020), [Online]. Available: https://coral.ai/docs/edgetpu/compiler/ (visited on 08/05/2022).

[17] See NVIDIA Corporation. "Deep learning frameworks documentation." (Oct. 27, 2022), [Online]. Available: https://docs.nvidia.com/deeplearning/frameworks/tf-trt-user-guide/index.html (visited on 11/05/2022).

[18] See NVIDIA Corporation. "Jetson Nano 2GB developer kit." (2022), [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-2gb-developer-kit (visited on 08/05/2022).

[19] See NVIDIA Corporation. "Jetson Nano developer kit." (2020), [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (visited on 08/05/2022).

[20] See NVIDIA Corporation. "Harness AI at the edge with the Jetson TX2 developer kit." (2022), [Online]. Available: https://developer.nvidia.com/embedded/jetson-tx2-developer-kit (visited on 08/05/2022).

KV260), and a TPU (found in the Coral Dev Board TPU). To ensure optimal performance, each platform required its specific compiler to enhance the efficiency of operations directly on the hardware, thereby increasing the speed of inference.

For optimisation purposes, the NVIDIA RTX3090[21] GPU was employed to train the DL model and establish a benchmark with a high-performing and efficient GPU for comparison.

In addition to the specialised hardware, all boards used in this study are equipped with a processing system (PS) to oversee task coordination and operating system management. The PS can be based on multiple architectures, but AMD64 and ARM64 are the most commonly used architectures in current state-of-the-art technology.

NVIDIA GPUs and TF-TRT Four NVIDIA GPUs were utilised in the current benchmark. The NVIDIA RTX3090 is a robust GPU that incorporates the Ampere Architecture, boasting 24 GB of VRAM. This GPU excels in training deep neural networks efficiently, handling substantial training batches with ease. Due to its power and efficiency, a direct comparison of inference speeds is not straightforward, but it serves as a valuable reference GPU for performance evaluation. However, its high power consumption, up to 350 W, makes it less suitable for embedded applications.

NVIDIA's Jetson GPUs are designed specifically for embedded systems, such as robots, requiring low power consumption. The Jetson Nano models differ in RAM capacity, offering 2 GB and 4 GB options. The NVIDIA Jetson TX2 is the evolved version of the TX1. In these boards, the available RAM is shared between the GPU and CPU.

Although all these GPUs support TensorFlow 2 and Keras models, they achieve peak performance and efficiency when the DL models are optimised for their specific architecture, utilising specialised CUDA and Tensor cores. NVIDIA's CUDA and Tensor cores are designed to optimise parallel and matrix operations, enhancing performance with CNNs. The TF-TRT, a library developed by NVIDIA, works alongside TensorFlow and TensorRT (TRT), analysing the ANN graph to determine the best transformations for speed efficiency using dedicated cores. Moreover, TF-TRT supports adjusting the network graph's resolution across FP32, FP16[22], and INT8 through quantisation. The advantage of TF-TRT over TRT lies in its compatibility with TensorFlow, allowing for a hybrid solution where certain operations are executed in TensorFlow and others in TRT.

AMD-Xilinx FPGAs, Vitis-AI and FINN field programmable gate arrays (FPGAs) are integrated circuits that can be reconfigured to meet the designer's needs. Due to its high-reconfiguration capability, FPGAs can be useful for executing parallelisable algorithms while keeping the power consumption low. These boards always have two main components processing system (PS) and programmable logic (PL). The PS is responsible for managing

[21] See NVIDIA Corporation. "GeForce RTX3090 Family." (2022), [Online]. Available: https://www.nvidia.com/en-eu/geforce/graphics-cards/30-series/rtx-3090-3090ti/ (visited on 08/05/2022).

[22] Half-precision floating-point

the operations and memory in the FPGA, while PL concerns to the reconfigurable integrated circuits. AMD-Xilinx deployed the DPU cores [249], a proprietary programable engine dedicated for CNN. This unit has a register configure module, a data controller module, and a convolution computing module. The DPU intellectual property (IP) can be integrated as a block in the PL with direct access to PS.

FPGAs are highly versatile integrated circuits designed for customisation post-manufacturing. Their adaptability makes them ideal for executing algorithms that can be parallelised, thus optimising for both performance and low-power consumption. An FPGA typically comprises two primary components: the PS and the PL. The PS oversees general operations and memory management, whereas the PL is dedicated to the reconfigurable logic that can be tailored to specific tasks. AMD-Xilinx has introduced DPU cores [249], specialised programmable engines optimised for CNNs. These cores include a register configuration module, a data controller, and a convolution computation module, and can be integrated into the PL for direct interaction with the PS.

In our evaluation, we focus on two FPGAs, identified as AMD-Xilinx ZCU104 and AMD-Xilinx Kria KV260. Both models share a similar architecture and compatibility, yet AMD-Xilinx Kria KV260 offers a more compact design suitable for robotics applications. Notably, AMD-Xilinx ZCU104 is equipped with two DPU cores, enabling it to process two neural network graphs concurrently, while AMD-Xilinx Kria KV260 incorporates only a single DPU core.

To facilitate the execution of models on the DPUs, they require quantised neural network graphs into INT8 weights, converted into a format compatible with the DPU. AMD-Xilinx provides Vitis-AI, a comprehensive Docker[23]-based environment, to streamline this process. Vitis-AI stands out as an integrated platform for AI inference development on AMD-Xilinx devices, supporting TensorFlow, PyTorch, and Caffe models. It includes specific quantisers tailored for FPGA architectures, compiling and optimising quantised models for the DPU. Moreover, Vitis-AI offers additional tools for model optimisation and debugging, such as pruning and profiling utilities.

Coral TPU and Edge TPU compiler The TPU is an AI accelerator ASIC specifically engineered by Google to enhance the performance of ANNs. This ASIC is designed to work seamlessly with TensorFlow, accommodating DL models crafted with TensorFlow lite (TFLite), TensorFlow's lighter version is tailored for edge devices.

Comparable to FPGAs, the ANNs deployed on edge computing TPUs necessitate quantisation, a process adapting the model's operations to be fully compatible with the ASIC's architecture by converting these operations into an INT8 format.

The model's design and management processes are conducted within TensorFlow and TFLite. To render a model compatible with the TPU, it is converted into a TFLite format using the Edge TPU Compiler.

[23] See Docker Inc. "Docker: Accelerated container application development." (May 8, 2024), [Online]. Available: https://www.docker.com/ (visited on 05/16/2024).

Our experiments utilised the Coral Dev Board TPU, an embedded board integrating a PS and a TPU system on-module (SoM). Like with FPGAs, operations in embedded TPUs must also be quantised to INT8. An alternative to the Coral Dev Board TPU is Google's Coral USB Accelerator[24], which connects to a host computer via USB for rapid inference. This device requires a power supply of 5 V and 500 mA, capable of delivering 4 trillion operations per second (TOPS) and achieving an efficiency of 2 TOPS/W.

RetinaNet RetinaNet, as highlighted in [252, 253], stands out in the domain of object detection as a state-of-the-art DL model, falling into the category of one-stage detectors. This model shares similarities with the SSD ANNs, detailed by Liu *et al.* [145]. It starts with an input layer, followed by a 'backbone' network that processes various feature maps to extract meaningful features from the image. Typically, this backbone is a CNN, with ResNet 50 being the preferred choice according to the seminal work by Lin *et al.* [252]. Following this, a FPN, introduced by [254], is employed. The FPN utilises a top-down architecture to refine the features processed by the CNN, aiming to boost the efficiency of box classification and regression tasks.

One of the primary enhancements RetinaNet offers over SSD DL models is the introduction of a novel custom loss function, known as focal loss [252], which focuses on improving the accurate detection and classification of objects (true positives (TPs)) over merely identifying non-objects correctly (true negatives (TNs)).

Considering the advancements provided by RetinaNet ResNet 50 over SSD networks, and given that these ANNs generally outperform YOLO architectures [C42, 255, 256], RetinaNet ResNet 50 was selected. The authors built upon a model previously implemented in TensorFlow 2 Keras by Humbarwadi [253], modifying its architecture to ensure compatibility across various platforms. The model was developed using a functional strategy[25]. However, preprocessing and post-processing layers were maintained in the sub-model format since they did not require conversion or recompilation for any specific platform, but rather were reimplemented. To maintain consistency and prevent processing speed degradation, the ResNet 50 was configured with the same pre-trained weights used by the ImageNet dataset [206].

Vitis-AI imposes certain operational constraints when compiling the DL model for FPGAs. These constraints are evident in the rectified linear unit (ReLU) operations, which must be immediately followed by another operation, such as a convolution or mathematical operation. This requirement affects the network's compilation, particularly between P6 and P7 of the FPN (Fig. 3.13), as outputs for the regression and classification layers are required at the convolution 2D P6 and convolution 2D P7. To address this, an additional convolutional 2D layer was introduced at P6, effectively duplicating the initial P6 convolution 2D layer (Figure 3.14). This adjustment allows the Vitis-AI compiler to successfully compile all core layers of

[24] See Google LLC. "USB accelerator." (2020), [Online]. Available: `https://coral.ai/products/accelerator` (visited on 09/21/2023).

[25] See TensorFlow. "The Functional API." (May 26, 2023), [Online]. Available: `https://www.tensorflow.org/guide/keras/functional` (visited on 08/05/2023).

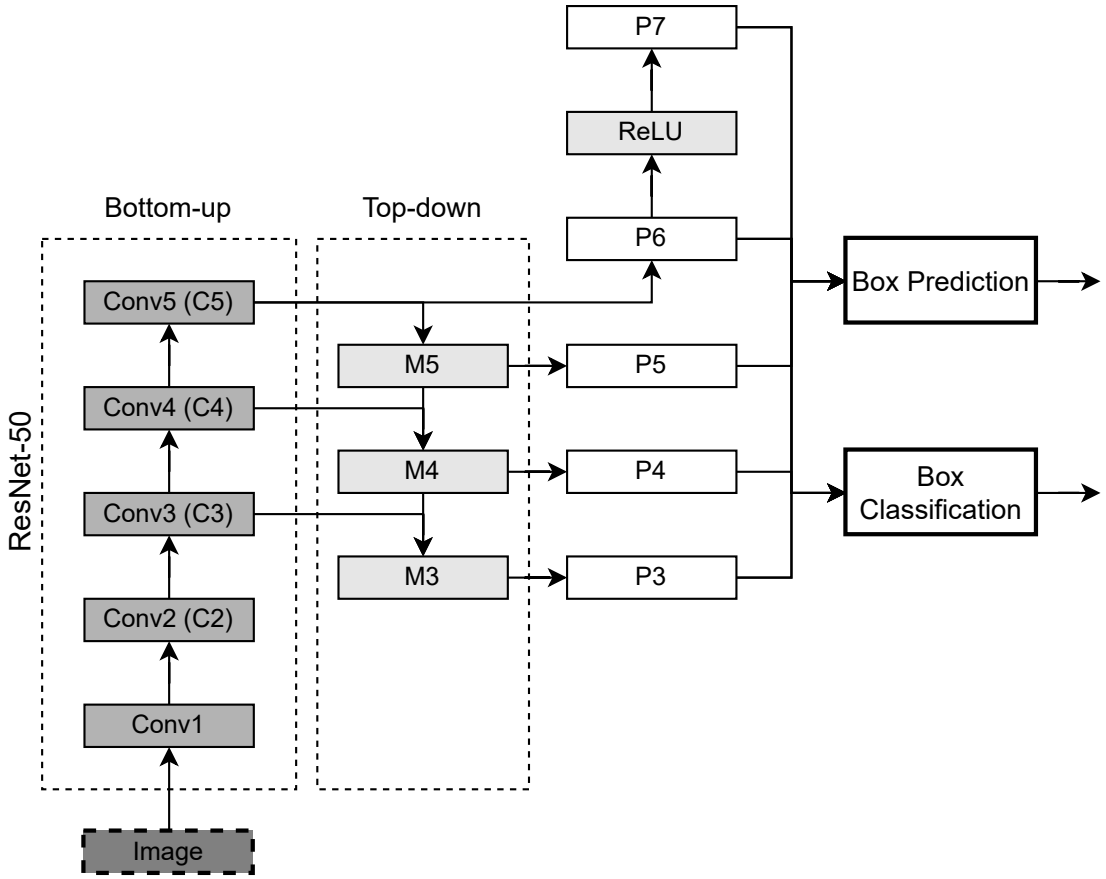


Figure 3.13: Overview of a simplified diagram of the RetinaNet ResNet 50. Conv_i are convolutional layers; M_i are intermediate layers composed by upsampling, additions and convolutions to generate FPN output layers P_i; P_i are convolution layers for the output of FPN.

the model on the DPU, preventing potential architecture splits and ensuring operations are executed on the CPU as needed.

The changed version of RetinaNet ResNet 50 (Figure 3.14) was trained by fine-tuning until the convergence of the train loss function. The training algorithm used the focal loss function and the stochastic gradient descendent (SGD) optimiser. For better adjustment of the learning rate and momentum values, the authors used the Keras Tuner library [258] with the Hyperband algorithm [259] to search for the best values that optimise the validation loss. During this stage, only two sets of the dataset are used: the train set for training the model and the validation set for evaluating the model performance in the evaluation metrics and tracking the model's overfitting. The model was trained in the GPU RTX3090.

To better understand and assess the performance and reliability of DL models across various platforms, our objective is to evaluate these models, specifically focusing on their efficacy in real-time object detection. This involves a series of steps to deploy the RetinaNet model across different heterogeneous platforms effectively.

The deployment process of RetinaNet onto each platform is generally consistent but ne-

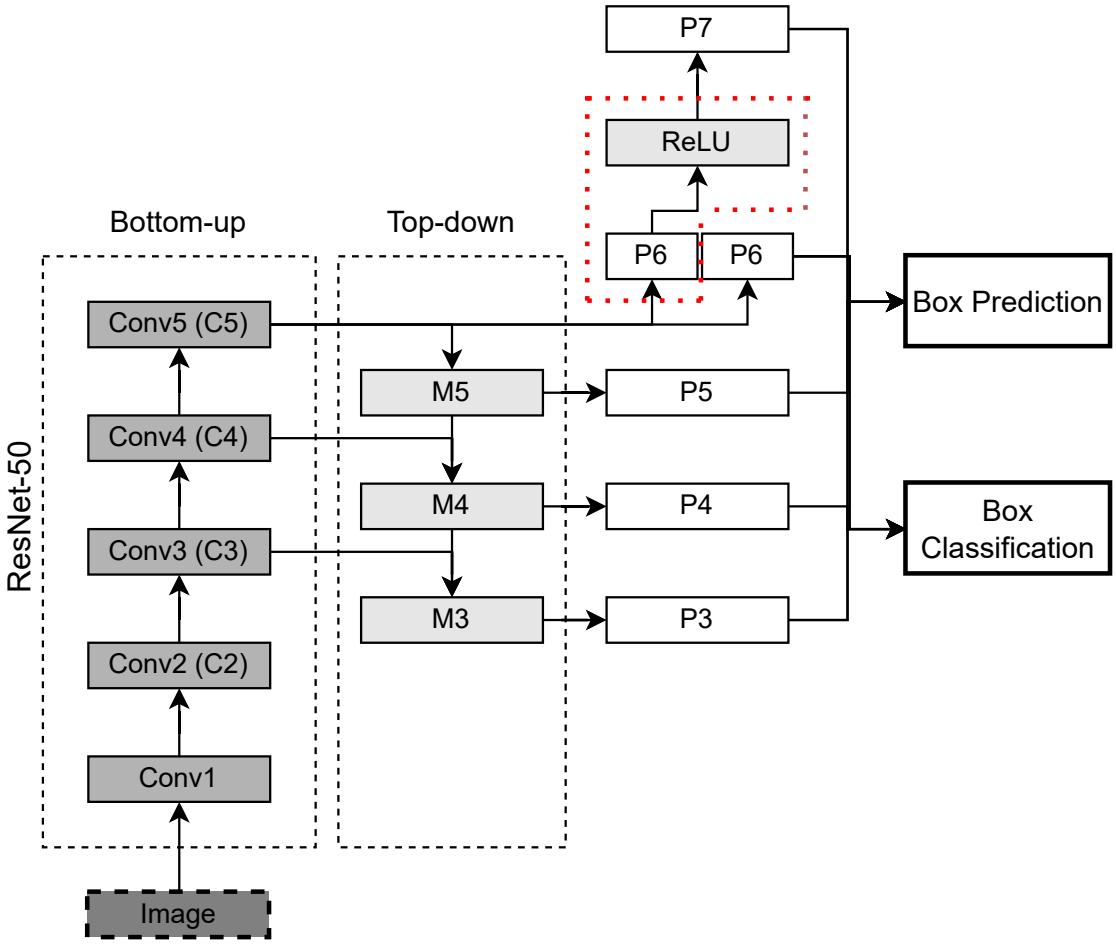


Figure 3.14: Overview of a simplified diagram of a modified RetinaNet ResNet 50 for FPGA compatibility. Conv$_i$ are convolutional layers; M$_i$ are intermediate layers composed of upsampling, additions, and convolutions to generate FPN output layers P$_i$; P$_i$ are convolution layers for the output of FPN.

cessitates the utilization of platform-specific libraries. The steps for deploying the model on each device are as follows:

1. Fine-tuning the RetinaNet with a ResNet 50 backbone on the VineSet training set;
2. Optionally quantising the model to INT8, depending on the platform's requirements, and;
3. Converting and deploying the model into a format compatible with the target platform.

The initial step involves training the ANN, which is uniformly conducted across all platforms using TensorFlow 2 on an NVIDIA GPU RTX3090. Due to some devices' inability to compile pre-processing and post-processing layers, only the ANN's core is utilised in subsequent steps. If necessary, these layers are reimplemented for execution in the PS on each device.

Post-training, the model is adjusted with platform-specific libraries. Both TPU and FPGAs necessitate model quantisation, which can either be training-aware or post-training, occurring post-model convergence. To ensure compatibility across devices, only post-training quantisation is used, given that quantisation-aware training may not be compatible with certain platforms. A calibration dataset derived from the training set is used to quantise the ANN's weights and calibrate them to the input data. Since training isn't performed at this stage, ground truth labels are not required for the calibration set. It was observed that, due to processing constraints, NVIDIA Jetson devices are unable to generate quantised models of RetinaNet ResNet 50.

The final step involves optimising the ANN nodes for the specific hardware, using proprietary compilers such as TF-TRT for GPUs, Edge TPU Compiler for TPUs, and Vitis-AI for FPGAs. A comprehensive guide for deploying the RetinaNet ResNet 50 on AMD-Xilinx FPGAs is provided by Magalhães *et al.* [C49].

Deploying ANNs on heterogeneous devices also entails reimplementing pre-processing and post-processing layers as needed. Since these layers were removed after the training phase, they are reconstructed for each device in Python using the OpenCV library, to be executed in the PS.

It is crucial to note that this study focuses solely on the core of the DL model. Optimisation efforts do not extend to pre-processing and post-processing tasks, which are executed on the device's CPUs due to certain operational limitations with the compilers.

3.2.3.2 RG2C, FINN and FPGA's PL

Given the parallelisation computing potential of the FPGA's PL, ANNs can be incredibly sped up. So, in this topic, we aim to evaluate the acceleration potential of FPGAs using the dedicated framework FINN. For this experiment, we consider the AMD-Xilinx ZCU104 and the classification dataset RG2C. Two types of classifiers will be implemented to explore the FPGAs' potential, namely the simplistic convolutional neural vector (CNV) and the MobileNet v1.

FINN and Brevitas FINN [260, 261] is a comprehensive, Docker-based environment provided by AMD-Xilinx. Unlike Vitis-AI, FINN operates directly on the PL layer of FPGA, converting ANNs into Verilog code, register-transfer level (RTL) designs, and IP blocks. It stands out as a freely available, open-source framework dedicated to constructing, executing, and deploying quantised neural networks (QNNs) on FPGA platforms. FINN offers a robust suite of tools and modules that streamline the entire QNN development process, from design to deployment, thereby enhancing the efficiency of DL models.

A workflow for deploying DL models onto FPGAs using FINN is illustrated in Figure 3.15.

A notable limitation of FINN is its exclusive compatibility with PyTorch. Consequently, all network designs must be tailored to this specification. Nonetheless, FINN provides the capability to manipulate and adjust DL models down to the bit level, thereby enabling the

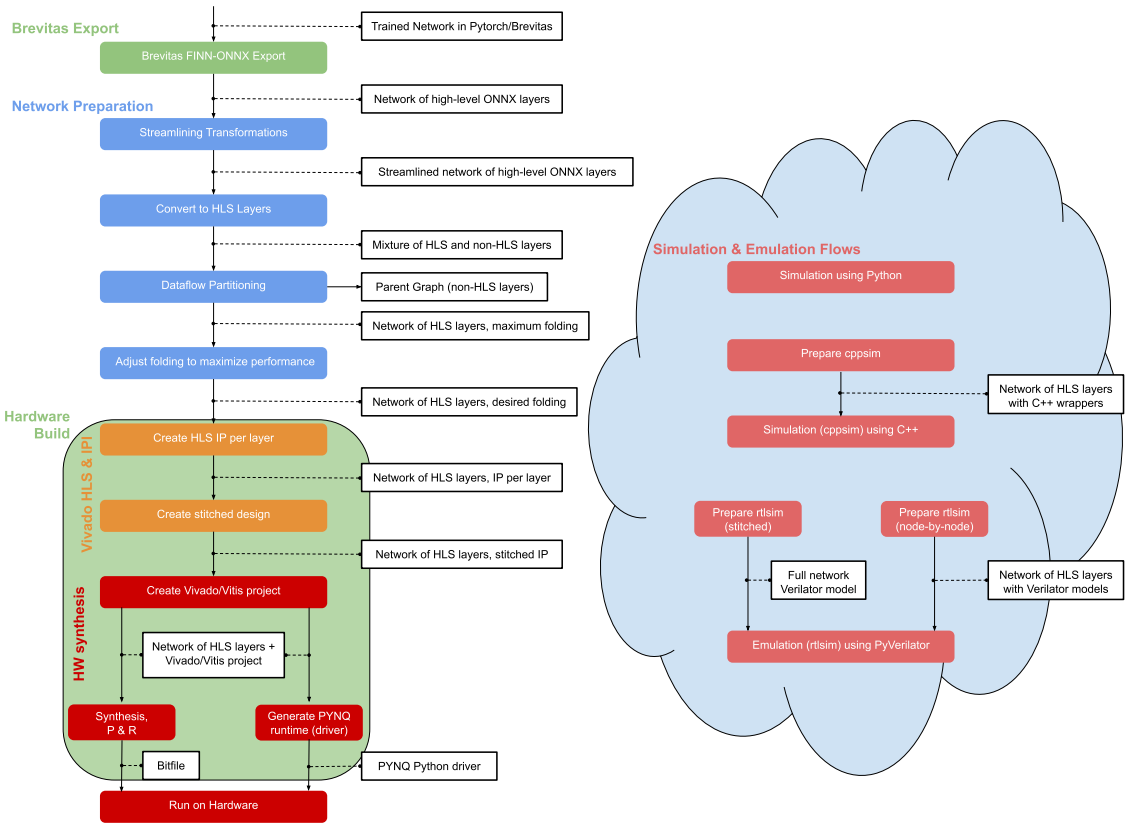


Figure 3.15: Comprehensive workflow for deploying DL models on FPGAs using FINN.

optimisation and compression of network operations to this granularity. This is facilitated by Brevitas [262], a supplementary library to PyTorch, which redefines common ANN operations to allow designers to specify the desired bit precision for computations.

In contrast to the conventional approach of employing a DPU IP core as an intermediary for DL models between the PS and PL, FINN directly deploys a dedicated IP core for each model. This strategy is primarily constrained by the finite number of configurable logic blocks available, necessitating careful adjustment of network parallelisation settings and the adoption of compact-sized ANNs for implementation.

CNV A CNV is structured with sequential layers, including two convolution layers and a maximum pooling layer. The network's prediction head is powered by three sequential dense layers. The design and structure of this proposed network are depicted in the diagram in Figure 3.16 and detailed in Table 3.6, both implementing the framework originally introduced by Umuroglu *et al.* [261] for the FINN architecture.

To facilitate the deployment of the CNV on a FPGA, we will explore two variants: one with two-bit operations in each layer and another with single-bit operations per layer. The latter is recognised as a binary neural network (BNN) [263, 264]. BNNs are designed to achieve optimal efficiency by converting all operations into binary bit-level computations, which are

more efficiently processed at the logic level.

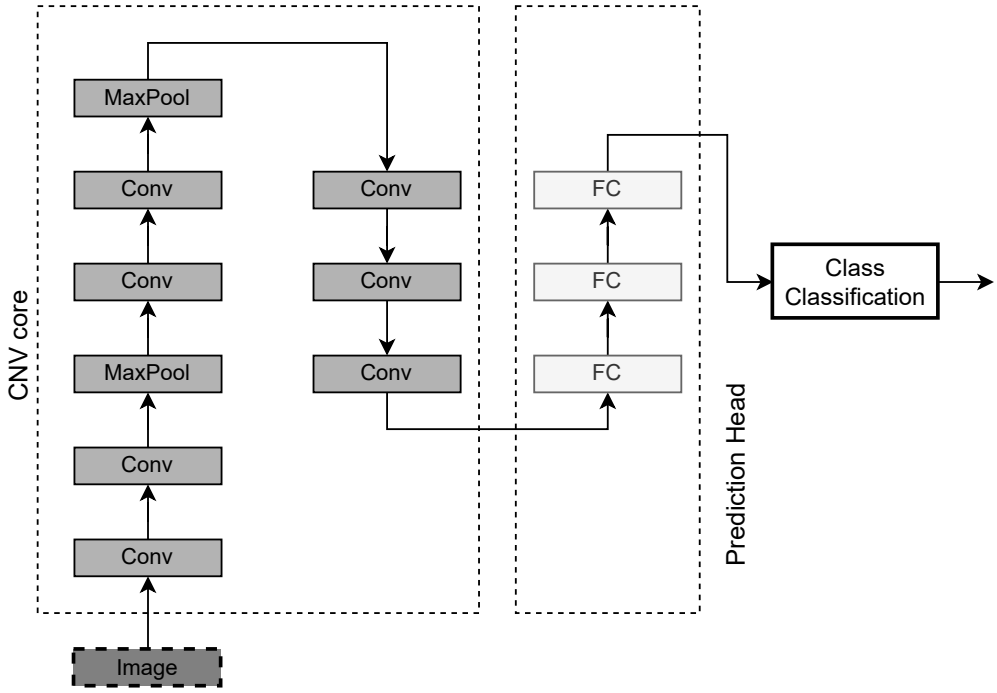


Figure 3.16: Overview of a simplified diagram of the CNV. Conv are convolution layers; MaxPool are maximum pooling layers; and FC are fully connected (or dense) layers.

Layer Type	Filter Shape	Kernel Size	Input Size
Conv	$64\times3\times3$	3×3	$1\times3\times3\times32$
Conv	$64\times64\times3\times3$	3×3	$1\times64\times30\times30$
MaxPool	Pool 2×2	2×2	$1\times64\times28\times28$
Conv	$128\times64\times3\times3$	3×3	$1\times64\times14\times14$
Conv	$128\times128\times3\times3$	3×3	$1\times128\times12\times12$
MaxPool	Pool 2×2	2×2	$1\times128\times10\times10$
Conv	$256\times128\times3\times3$	3×3	$1\times128\times5\times5$
Conv	$256\times256\times3\times3$	3×3	$1\times256\times3\times3$
Conv	–	–	$1\times256\times1\times1$
FC	256×512	–	1×256
FC	512×512	–	1×512
FC	512×1	–	1×512

Table 3.6: CNV Architecture

MobileNet v1 MobileNet v1 [196] is a DL model design for mobile and low computing devices. In its original implementation, it is designed for classification tasks.

The used version of this classifier was implemented in PyTorch as originally stated. However, some changes in the layers were made to ensure that the ANN fits in the FPGA PL and is

compatible with input data of (32×32) px. At the end of the MobileNet v1 was appended a prediction head composed of an average pooling layer, a fully connected layer and a quantised ReLU.

MobileNet v1, as described by Howard *et al.* [196], is a DL model optimised for mobile and low-resource computing devices. Primarily designed for classification tasks, this model's architecture has been adapted for efficient implementation.

In the version utilised, the MobileNet v1 model was implemented using PyTorch, in line with the original description. Modifications were made to its layers to ensure that the ANN is suitable for deployment on FPGAs' PL and to handle input data with a resolution of (32 × 32) px. To accommodate the deployment requirements, the architecture of MobileNet v1 was appended with a prediction head. This additional component consists of an average pooling layer, a fully connected layer, and a quantised ReLU.

The process of deploying FPGAs using FINN is outlined in a sequential manner, as depicted in the referenced FINN flowchart, Figure 3.15. The procedure up to the Brevitas export shares similarities with other network deployments and utilises Pytorch. A crucial step involves Brevitas, which is used to implement specific quantisation nodes. These nodes are essential for specifying the bit count for the various operations.

After training, the network is exported to a standard model format known as FINN-ONNX, which is compatible with the pen neural network exchange (ONNX)[26] format. This FINN-ONNX model undergoes several transformations aimed at optimising the network architecture, ensuring it aligns well with AMD-Xilinx IP and the FPGA's PL. Key operations include streamlining [266], which eliminates unnecessary floating-point operations, merges multiple operations into a single one when feasible, and converts some operations into multiple threshold nodes. These adjustments help reduce the number of operations and enhance parallelisation, resulting in a faster network that utilises the FPGA space effectively.

Next, the network is transitioned into high-level synthesis (HLS) layers, followed by conversion into custom IP modules or a series of IPs. This stage allows for numerous simulations to evaluate the network's optimisation level. The final step involves translating the IPs into instructions that the FPGA can understand.

Control over the network's parallelisation is achieved through two parameters: single instruction multiple data (SIMD) and process element (PE). SIMD relates to the number of data elements processed concurrently in a single computation, while PE refers to the number of parallel computations. The optimisation of these parameters must adhere to specific equations, (3.1) and (3.2), to ensure all FPGA space is efficiently used. In both equations, H and W represent the number of input and output features, respectively. These equations follow the congruence relation notation [267] and state that the remaining between the integer division of the two values is zero.

[26] See The Linux Foundation. "ONNX—Open Neural Network Exchange." (2019), [Online]. Available: https://onnx.ai/ (visited on 04/23/2024), The ONNX is a standard to interoperability of ML models between different frameworks.

$$0 \equiv H \pmod{\text{PE}} \tag{3.1}$$

$$0 \equiv W \pmod{\text{SIMD}} \tag{3.2}$$

An advantage of this deployment strategy is the direct implementation of ANNs into logic circuits (configurable logic blocks), optimising performance. Additionally, Brevitas enables bit-level control over the networks. For this analysis, three scenarios were considered:

- a MobileNet v1 with four-bit quantised weights and biases (*'mobilenet_w4a4'*);
- A CNV with two-bit quantised weights and biases (*'cnv_w2a2'*); and
- A CNV with one-bit quantised weights and biases (*'cnv_w1a1'*).

This protocol assures a probable optimisation of the deep neural networks under low-power and high-performing devices. However, they may detect multiple objects simultaneously. The following strategies aim to develop algorithms that can filter the different objects.

3.2.4 Maturity assessment

As previously mentioned, one potential strategy for identifying and selecting the next fruit to approach could be based on its stage of maturity. There are various techniques that can be applied to this end. In our essay, we aim to compare DL models and colour feature analysis against traditional statistical classifiers.

For this purpose, we will utilise the AgRobTomato and RPiTomato datasets, as detailed earlier in section 3.2.1. This combined dataset includes images of tomatoes individually categorised by their ripening stage: 'unripe', 'breaking stage', 'reddish', and 'ripe'. A total of 632 tomato images will be used in our analysis.

3.2.4.1 Deep learning for maturity classification

The DL models we considered were meticulously designed, trained, and deployed using TensorFlow version r.1.15.0 or Darknet (similarly to the problem studied in the section 3.2.2), depending on the type of model. We carried out these procedures in a Colab notebook, utilising an NVIDIA Tesla T4 GPU with 12 GB of VRAM and a computation capability that ranges from 3.5 to 7.5. Among the different DL object detection models we previously tested, we chose the SSD MobileNet v2 from the TensorFlow Model Zoo and the YOLO v4 from the Darknet database. Both models had been pre-trained on the COCO dataset, with an input size of (640×640) px for SSD MobileNet v2 and (416×416) px for YOLO v4. The decision to use these two DL models was based on the results from experiments discussed in the sections 3.2.2 and 3.3.1 and referenced in the study by Magalhães *et al.* [C42].

We fine-tuned these pre-trained models for the specific task of detecting and classifying tomatoes according to their maturity stage. The remaining protocol followed the same procedures as detailed in section 3.2.2. Figure 3.12 outlines the pipeline to reach the complete trained models.

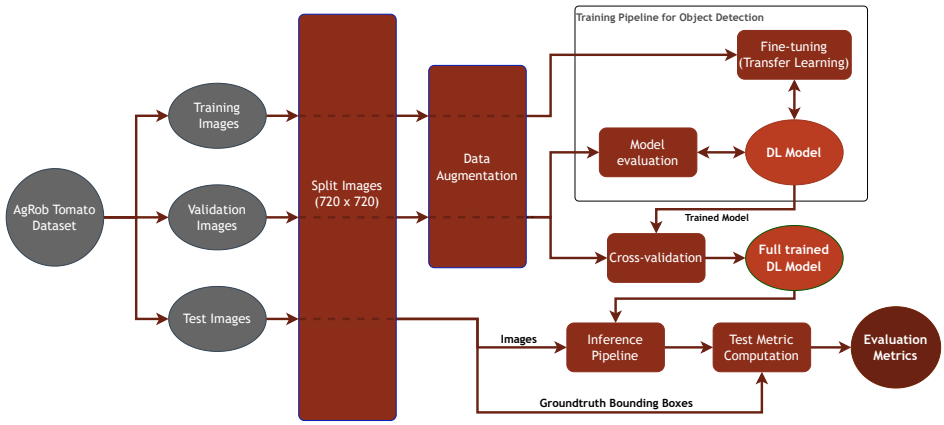


Figure 3.17: Workflow of the performed methods to reach the trained DL models.

3.2.4.2 HSV colour model space and Gaussian classification

An approach utilising histograms from the HSV colour space has been developed as an innovative alternative to DL models for the classification of tomato maturity. The code for this HSV colour space model is accessible at the GitHub repository[27]

To construct the model, images of tomatoes from different ripeness categories were selected, with ten tomatoes per category. These images were sourced from both the AgRobTomato Dataset and the RpiTomato Dataset to incorporate variety, the former providing a broader view and the latter offering closer shots of the fruits.

The first step involved extracting the regions of interest from these images, utilising the coordinates from the bounding boxes provided by the CVAT for image cropping.

Subsequently, these regions of interest images were converted from the RGB to the HSV colour space. This conversion is beneficial as the HSV colour space, particularly the Hue channel, tends to offer less noise and more robustness against lighting variations compared to the RGB colour space.

For each HSV image, a colour histogram focusing solely on the Hue channel was generated using OpenCV [147] to extract the colourimetric data. It's important to note that OpenCV represents Hue values on a scale from 0 to 179. Given that the full-colour spectrum is not necessary for accurately modelling tomatoes' colour distribution, the Hue parameter's origin was

[27] See G. Moreira and S. C. Magalhães. "Tomato maturity classification using HSV and gaussian features." (Oct. 25, 2021), [Online]. Available: https://github.com/gerfsm/HSV_Colour_Space_Model (visited on 10/25/2021).

adjusted, yielding a density h-spectrum histogram with a distribution resembling a normal curve. The entire bin range was utilised to ensure model accuracy.

Nonetheless, this segmentation method has its drawbacks, notably that the region of interest includes portions of the background, potentially causing the data to appear multimodal, i.e., displaying more than one peak. Fitting such data with a unimodal model could lead to inaccuracies.

A Gaussian mixture model was employed to address this challenge, serving as a probabilistic model for representing normally distributed subpopulations within the overall population, as illustrated in the Figure 3.18. This approach facilitates the separation of colour regions within the image by selecting the Gaussian component with the highest peak, corresponding to the region of interest, and disregarding the rest. The selection is based on the Gaussian mixture weights normalised, reflecting the law of total probability (theorem 3.2.1). These weights represent the probabilities of a data point belonging to each Gaussian component.

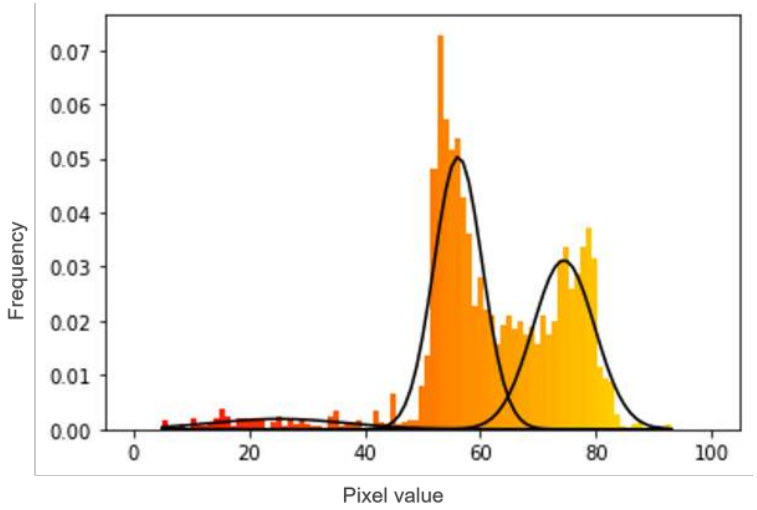


Figure 3.18: Representation of the H-spectrum and a Gaussian mixture model probability distribution.

Despite background noise, the data distribution within the region of interest is typically well-defined, suggesting a higher probability (and thus a higher weight) of being a normal distribution. This method effectively separates tomato pixels from the background pixels. For a more refined analysis, box plots for each region of interest were also produced, showcasing values within three standard deviations of the mean, corresponding to 99.7 % of the data under a Gaussian distribution.

Theorem 3.2.1 (Law of total probability)**.** *In a discrete case, if* $\{B_n : n \in \mathbb{N}^+\}$ *is a finite or countably infinite partition of a sample space and each even* B_n *is measurable, then for any event* A *of the sample space:*

$$P(A) = \sum_n P(A \cap B_n)$$

In summary, this HSV colour space approach, complemented by Gaussian mixture models, presents a compelling alternative to DL methodologies for the classification of tomato maturity, with the procedural steps and theoretical foundations graphically described in the Figure 3.19.

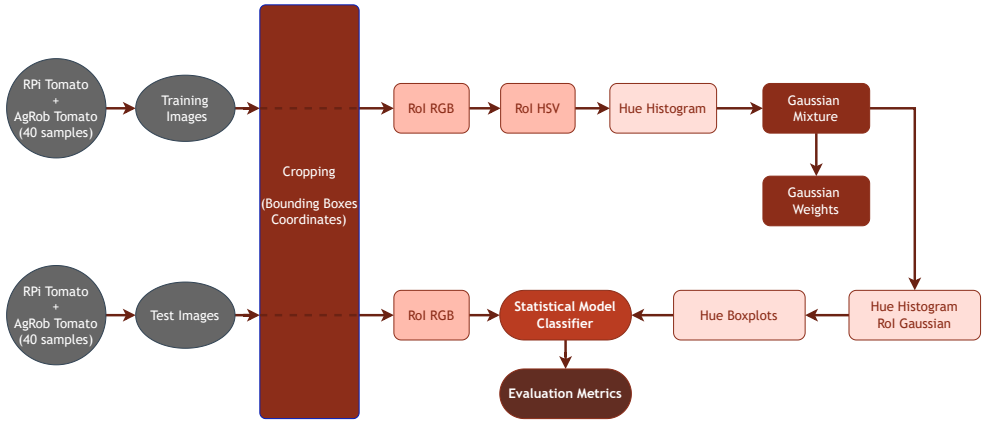


Figure 3.19: Workflow of the performed methods to reach the HSV colour space model.

3.2.5 Results evaluation

The literature highlights a variety of evaluation metrics for model performance assessment [269, 219]. Initially, we utilised the default metrics available in the pipeline during the model training phase. However, given the differences in evaluation criteria between the COCO dataset and OID, it was imperative to adopt a unified metrics pipeline for evaluation. We opted for the metrics utilised in the Pascal VOC challenge [219], specifically the precision × recall curve and the mean average precision (mAP), following the implementation by Padilla *et al.* [269] and the FiftyOne library[28]. To further enhance our evaluation, we included additional metrics: (a) total recall; (b) total precision; (c) F1 score.

Recall, as defined in equation (3.3), measures the model's capability to detect all relevant objects, essentially the proportion of detected bounding boxes within the validation set. Precision, defined in equation (3.4), assesses the model's accuracy in identifying only relevant objects, aiming to minimise FPs. The F1 score, given in equation (3.5), calculates the harmonic mean between recall and precision, providing a balance between the model's precision and recall.

In object detection, TPs signify correct detections, FPs denote objects incorrectly detected, and FNs represent missed objects. The total ground truths are computed as TP + FN, and the total detections are TP + FP. Detections are validated using the intersection over union (IoU) metric [269], only considering detections with an IoU ≥50 %.

[28] See Voxel51. "FiftyOne." (2023), [Online]. Available: https://docs.voxel51.com/ (visited on 12/14/2023).

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{3.3}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{3.4}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}} \tag{3.5}$$

For classification models, alongside the aforementioned metrics, TNs (predictions accurately identified as not belonging to a class) are also considered. Accuracy, defined in equation (3.6), measures the overall proportion of correct predictions. In scenarios with unbalanced datasets, balanced accuracy, as given in equation (3.9), is preferred for a fair performance assessment, incorporating both the sensitivity equation (3.7) – equivalent to recall – and the specificity, equation (3.8), which gauges the proportion of actual negatives correctly identified.

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{TN}+\text{FN}} \tag{3.6}$$

$$\text{Sensitivity} = \frac{\text{TN}}{\text{TN}+\text{FP}} \tag{3.7}$$

$$\text{Specificity} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{3.8}$$

$$\text{Balanced accuracy} = \frac{\text{Sensivity}+\text{Specificity}}{2} \tag{3.9}$$

This approach ensures a comprehensive evaluation of both object detection and classification models, employing a range of metrics to assess their performance accurately.

3.3 Results

During this section, we will present and explore the results of the different experiments previously made. This analysis is made concerning the evaluation metrics stated in the section 3.2.5, which are broadly accepted in the literature.

3.3.1 Fruit detection

In this section, we present and assess the findings related to the challenge of fruit detection within a cultivar setting, referring specifically to section 3.2.2. Our analysis primarily focuses on evaluating different SSD and YOLO models to determine their efficacy. Besides comparing models trained on the Tomato dataset, we also investigate the effectiveness of utilising the OID dataset for tomato (and potentially other fruits) detection in a cultivar environment. As highlighted in section 3.2.5, the performance of the trained models was gauged using metrics from the Pascal VOC challenge, along with additional metrics, which include:

- Recall × precision curve;
- mean average precision (mAP);
- Total recall;
- Total precision;
- F1 score;
- Inference time.

Prior to delving into the models' performance evaluation, it's important to identify the optimal confidence threshold. This threshold is relevant as it maximises the F1 score (illustrated in Figure 3.20), thereby ensuring a balance between precision and recall. This balance is essential for optimising the number of TPs while minimising the amount of FPs and FNs as shown in Figure 3.21. From this figure, we can discern models with superior performance, where models exhibiting flatter curves suggest higher prediction confidence and fewer FPs and FNs. Here, we determine the maximum F1 score for each model alongside its confidence threshold, as detailed in Table 3.7. These values are pivotal for the models' prediction capabilities.

SSD MobileNet v2 deserves special attention due to its notably low FPs occurrence (Figure 3.21). This characteristic is crucial to avoid mistakenly targeting non-fruits, thus preventing potential harm to the crops or the harvesting robot.

Table 3.7: Confidence threshold for each DL model that optimizes the F1 score metric, indicating their performance levels.

	Confidence	**F1-Score**
YOLOv4 tiny	≥49 %	85.92 %
SSD Inception v2	≥21 %	89.85 %
SSD MobileNet v2	≥40 %	82.22 %
SSD ResNet 50	≥46 %	90.46 %
SSD ResNet 101	≥34 %	81.75 %

The previous analysis assessed the models on a validation set. Now, we will assess the performance evaluation on the test set. The test set, an independent collection of images, helped gauge the trained DL models' generalisation ability. The study commenced with a two-pronged approach, focusing on fully characterised models and models with a ≥0 % confidence threshold. A significant insight was the advantage of imposing a confidence rate limit.

For the task of detecting tomatoes in greenhouse images, we constructed smooth precision-recall curves, revealing the trade-off between recall and precision across different confidence score thresholds, as detailed in Figure 3.22. Typically, higher confidence thresholds yield more precise but less comprehensive predictions. All models, except for SSD

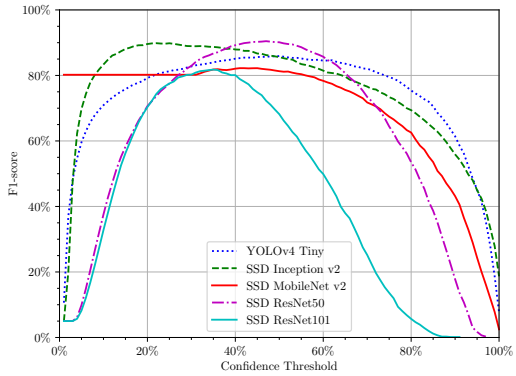
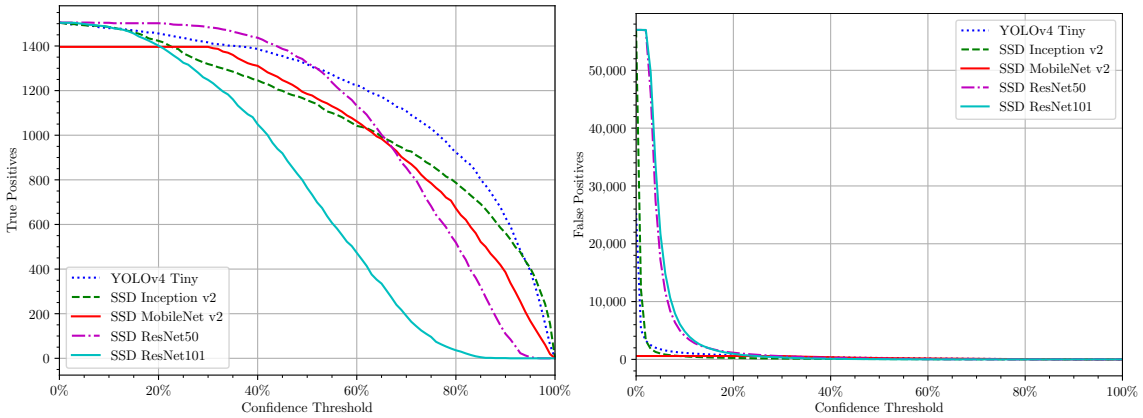
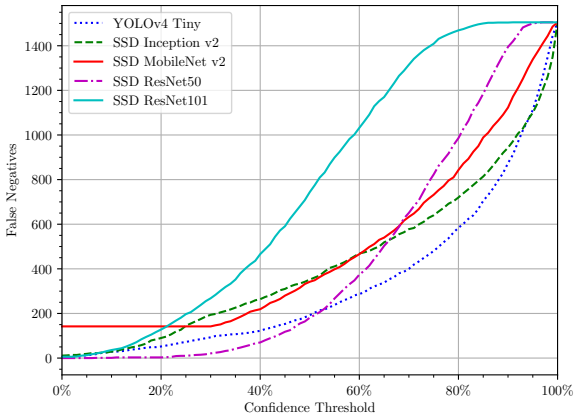


Figure 3.20: Evolution of the F1 score with the variation of the confidence threshold for all DL models in the validation set without augmentation.



(a) True Positives

(b) False Positives



(c) False Negatives

Figure 3.21: Evolution of the number of TPs, FPs, and FNs with the increase of the confidence threshold.

MobileNet v2, nearly achieved a 100 % recall rate, though, the precision hovered around 0 %. The model with the highest area under the curve (AUC), as reported by Padilla *et al.* [269], demonstrated superior performance. YOLO v4 Tiny and SSD ResNet 50 showed comparable

excellence; however, their low precision at high recall rates, coupled with reduced overall recall and F1 scores (Table 3.8), indicated substantial prediction noise and a high FP rate. Notably, SSD ResNet 50 exhibited the poorest outcomes among it and an YOLO v4 Tiny. Considering all model predictions and using the F1 score as a balanced metric of recall and precision, SSD MobileNet v2 was identified as the best-performing model.

Table 3.8: Results of the different SSD and YOLO models over many metrics, considering all the predictions and the best-computed confidence threshold.

Model	**Confidence**	**Inference Time**	**mAP**	**Precision**	**Recall**	**F1**
YOLOv4 Tiny	≥0 %	4.87 ms	77.19 %	6.38 %	97.52 %	11.98 %
SSD Inception v2	≥0 %	24.75 ms	70.39 %	3.53 %	95.82 %	6.82 %
SSD MobileNet v2	≥0 %	16.44 ms	57.99 %	78.07 %	62.44 %	69.39 %
SSD ResNet50	≥0 %	47.78 ms	75.74 %	3.6 %	97.62 %	6.94 %
SSD ResNet101	≥0 %	59.78 ms	66.88 %	3.55 %	96.32 %	6.85 %
YOLO v4 Tiny	≥49 %	4.87 ms	47.48 %	88.39 %	49.33 %	63.32 %
SSD Inception v2	≥21 %	24.75 ms	48.54 %	85.31 %	50.93 %	63.78 %
SSD MobileNet v2	≥40 %	16.44 ms	51.46 %	84.37 %	54.40 %	66.15 %
SSD ResNet50	≥46 %	47.78 ms	42.62 %	92.51 %	43.59 %	59.26 %
SSD ResNet101	≥34 %	59.78 ms	36.32 %	88.63 %	38.13 %	53.32 %

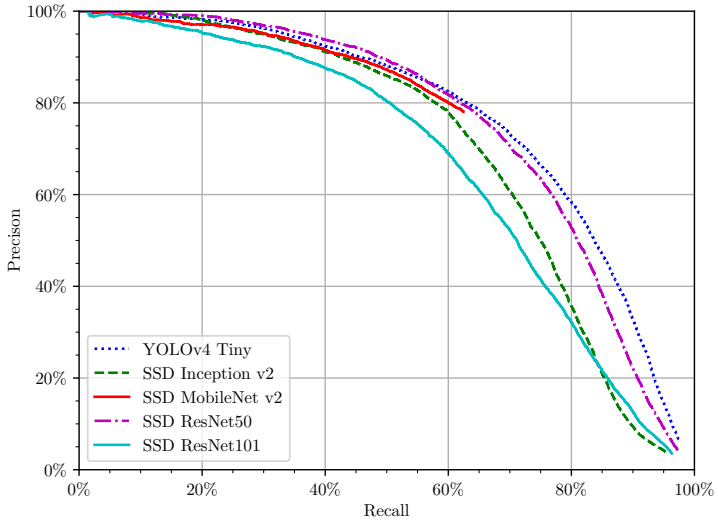


Figure 3.22: Precision × recall curve in the test set considering all the predictions.

The low precision rates were often attributed to the low confidence in predictions, as identified during the confidence threshold tuning process. By applying an additional filtering process based on the optimal confidence threshold to maximise the F1 score on the validation set (Table 3.7 and Figure 3.20), we observed an increase in precision (Table 3.8). This approach

transformed the precision × recall curve (Figure 3.23) through a truncation process, ensuring all predictions had a precision rate higher than 80 % but a recall rate lower than 60 %. Among the fully characterised models, SSD MobileNet v2 remained the top performer. Nonetheless, YOLO v4 Tiny, despite its slightly inferior performance evaluation for real-time applications, showcased notable inference time. It is essential to highlight that this model is quantised (INT8), unlike others (FP32), not warranting a fair comparison of all models. SSD ResNet 101, being overly complex for this task, overfitted the training data and performed poorly.

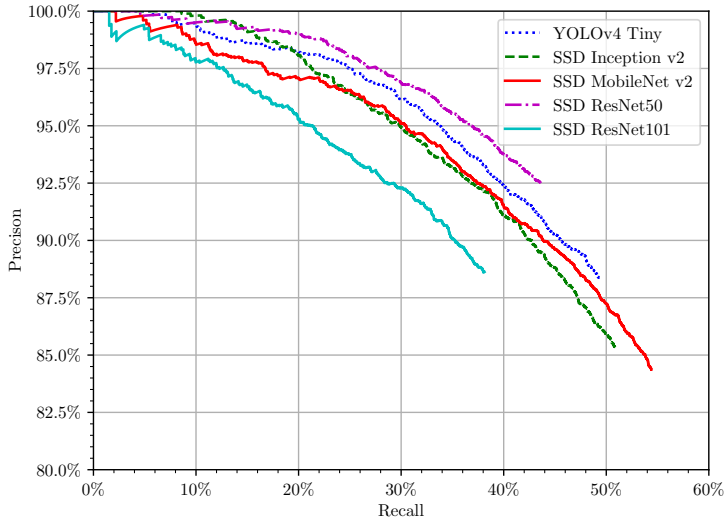


Figure 3.23: Precision × recall curve in the test set using the calibrated confidence threshold.

Quantisation maps continuous or large sets (in this context, FP32) to a confined set. In DL, typically, this process involves converting neuron weights and biases into 8-bit integers (INT8). Such quantisation suits edge devices execution for real-time operation, as it marginally impacts ANNs performance while significantly boosting their speed [271].

The models struggled to generalise effectively for tomato classification, resulting in significant performance drops from the validation to the test set. Enhancing the amount of data and the variability could mitigate this issue. Additionally, a thorough analysis of the data could be relevant, with the aim of reducing repeatability and searching for the most unique data, applying data-centric strategies.

From Figure 3.26, using unfiltered inference predictions led to numerous FPs. Filtering results using a threshold or a similar process significantly improved performance across all models, except for SSD MobileNet v2. This model demonstrated a well-balanced precision and recall with a high confidence rate in its predictions, avoiding scenarios of near 0 % precision. Moreover, it consistently achieved a precision rate above 80 %, allowing its use without further filtering.

As explained in section 3.2.2, we considered additional essays with the state-of-the-art dataset OID v6, which contains images of fruits, including tomatoes. We trained the DL mod-

els on a portion of this dataset and evaluated their ability to detect tomatoes in greenhouse plants. We have tabulated the results of using these models on the test set of the acquired dataset, with a confidence threshold of 30 %, in table 3.9. Our conclusion from these data is that no model can be used for the agricultural detection of fruits.

The training of respective models with various classes was challenging and time-consuming. The different classes considered were problematic, particularly those containing small datasets and having weak robustness. After analysing the test set, we found that using models trained on OID v6 resulted in several incorrect detections, including the labels of bounding boxes with different fruits, such as grapes, bananas, apples, lemons, among others, that are included in the 15 classes (Fig. 3.25). Among various incorrect detections, we highlight the clustering of tomatoes as a single fruit and the detection of leaves as a fruit.

The poor results of OID v6 are directly related to the tomato class dataset, with images of ripe tomatoes (red – Fig. 3.2) and with few tomatoes per image, unlike those intended to detect tomatoes. This type of fruit is uncommon in greenhouses since tomatoes must be harvested at an early stage of maturity to accommodate further processing and shipping of goods.

Table 3.9 shows the results of the different SSD and YOLO models evaluation, with a confidence threshold of 30 %. We can see that the YOLO v4 model has 0.0 % of mAP. The SSD ResNet 50 model has a mAP of 0.33 %, while the SSD Inception v2 and SSD MobileNet v2 models have mAP of 0.49 % and 1.18 %, respectively.

Model	**Dataset**	**mAP**	**Precision**	**Recall**	**F1**
YOLO v4	OID v6	0.0%	0.0%	0.0%	0.0%
SSD ResNet 50	OID v6	0.33%	95.65%	0.34%	0.68%
SSD Inception v2	OID v6	0.49%	96.97%	0.49%	0.98%
SSD MobileNet v2	OID v6	1.18%	63.91%	1.67%	3.25%

Table 3.9: Results of the different SSD and YOLO models evaluation, considering a confidence threshold of ≥30 %.

To better understand the models' capabilities, we performed additional analysis of the results by considering representative images from the dataset for specific situations in the captured dataset, such as:

1. darkened tomatoes;
2. occluded tomatoes;
3. overlapped tomatoes.

Figure 3.27 presents a representative result of darkening tomatoes, which occurs when the robot enters the greenhouse or when tomatoes are sun-protected in the shadow of other plants or leaves. In this scenario, all models yielded similar results. However, SSD MobileNet v2 outperformed slightly, detecting one additional tomato in the considered images.

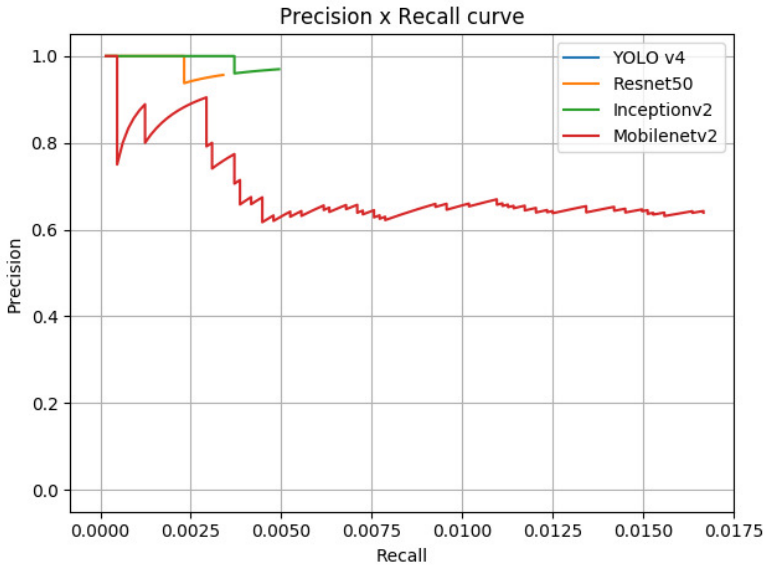
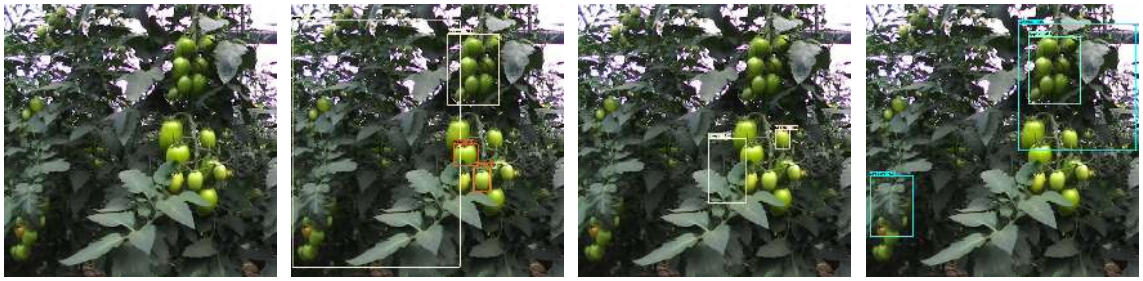


Figure 3.24: Precision × recall curve in the test set considering a confidence rate threshold of 30 %.



(a) YOLOv4 (b) SSD ResNet 50 (c) SSD Inception v2 (d) SSD MobileNet v2

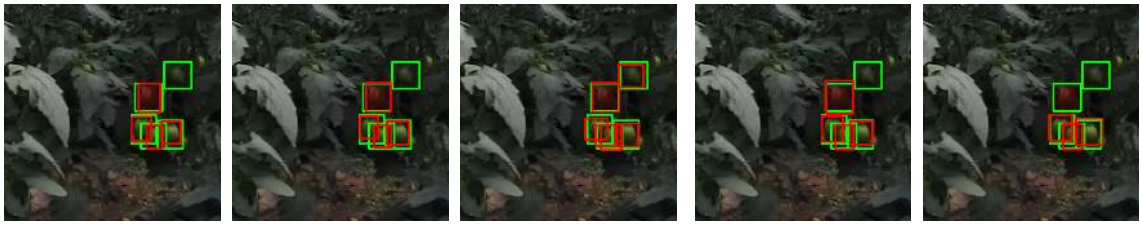
Figure 3.25: Sample image results of the different models trained on OID v6.

Occlusion refers to situations where a tomato is not fully visible, potentially occluded by branches, stems, leaves, or other tomatoes. Overlapping or clustering is a specific scenario where a tomato is occluded by other tomatoes, and the detection system should identify both tomatoes. Figure 3.28 illustrates a typical occlusion case by leaves. In this scenario, SSD MobileNet v2 demonstrated the best network generalisation, detecting tomatoes with less than 50 % of their area occluded. The other networks failed to detect the occluded tomatoes.

(a) YOLOv4 Tiny (b) SSD Inception v2 (c) SSD MobileNet v2 (d) SSD ResNet 50 (e) SSD ResNet 101

(f) YOLO v4 Tiny (g) SSD Inception v2 (h) SSD MobileNet v2 (i) SSD ResNet 50 (j) SSD ResNet 101

Figure 3.26: Comparison between using unfiltered images (**a**–**e**) and filtered images through the computed confidence threshold (**f**–**j**).



(a) YOLOv4 Tiny (b) SSD Inception v2 (c) SSD MobileNet v2 (d) SSD ResNet 50 (e) SSD ResNet 101

Figure 3.27: Result comparison for darkened images.

(a) YOLOv4 Tiny (b) SSD Inception v2 (c) SSD MobileNet v2 (d) SSD ResNet 50 (e) SSD ResNet 101

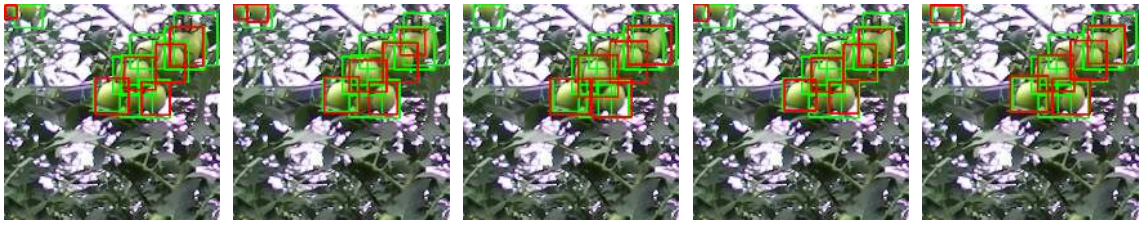
Figure 3.28: Result comparison for occluded tomatoes.

Considering the case of clusters or overlapping tomatoes (Fig. 3.29), all the DL models

performed similarly, indicating any of them could be effectively used for this situation.

In summary, SSD MobileNet v2 emerged as the best-performing model. It effectively handled all situations, minimised FPs, and was the fastest network among the SSD models, achieving inference in 16 ms with a high-performance GPU. However, the performance of YOLO v4 Tiny cannot be overlooked due to its quantised model significantly reducing processing time.

This work addressed the significant challenge of detecting tomatoes in the early ripening stage, where colour features are less distinguishable, as highlighted in the literature review. We have made this dataset public to aid further research and we have analysed the most promising ANN models for edge computing. We discovered that tuning the confidence threshold parameter could significantly enhance model performance. This research paves the way for deploying these models into real-world robots and perception systems to benchmark against human labour in terms of detection time, reliability, and accuracy.



(a) YOLOv4 Tiny (b) SSD Inception v2 (c) SSD MobileNet v2 (d) SSD ResNet 50 (e) SSD ResNet 101

Figure 3.29: Result comparison for overlapped tomatoes.

3.3.2 Acceleration of fruit detection

As an approach for the speed of inference problems performed in the previous section, here, we will cover the acceleration of DL models through the use of dedicated hardware. Our benchmarks utilised the NVIDIA RTX3090 GPU as the reference platform to assess the performance of the RetinaNet ResNet 50 model across various heterogeneous platforms, using the VineSet. It is important to note that the NVIDIA RTX3090 GPU is both high-performing and power-intensive, making the values presented here reference points rather than direct comparisons, especially in terms of inference speed. Figure 3.30 demonstrates the model's accuracy on the test set. The model was specifically compiled to optimise hardware utilization, primarily leveraging Tensor cores. Across all compiled versions of the RetinaNet ResNet 50, results were consistent, with a slight improvement observed in the default TensorFlow 2 model version. This discrepancy can be attributed to a reduction in detection confidence post-compilation, leading to the exclusion of detections below the confidence threshold.

Figure 3.31 clearly illustrates the benefits of compiling DL models for NVIDIA-specific hardware. Without altering the weight variable type, i.e., maintaining the weights in FP32,

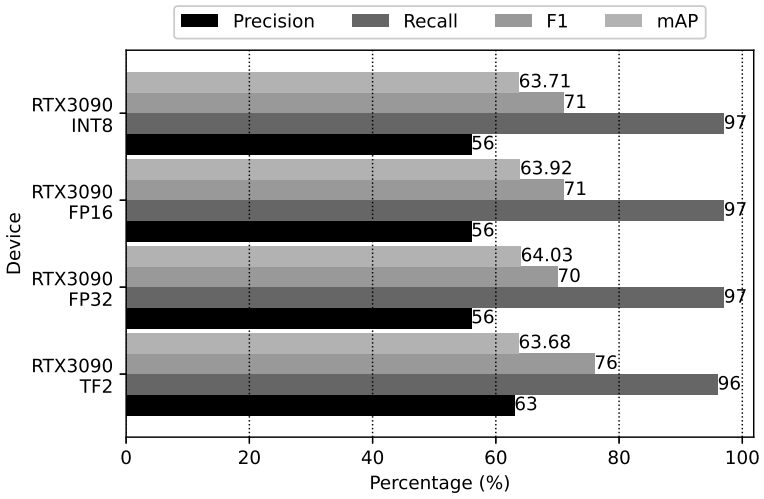


Figure 3.30: Inference performance in the evaluation metrics in the reference GPU considering RAW TensorFlow 2 and the optimised models for NVIDIA Tensor cores. INT8 report to the model's weights quantised into 8-bit integers, FP16 to weights into 16-bit floating-point, and FP32 to weights into 32-bit floating-point.

TF-TRT was able to achieve an inference speed increase of tenfold compared to TensorFlow 2. Reducing the weight resolution from FP32 to FP16 resulted in models that were 2.2 times faster than TF-TRT FP32 and 26 times faster than TensorFlow 2. Given that NVIDIA RTX3090 GPU is not optimised for integer operations, converting to INT8 proved to be ineffective.

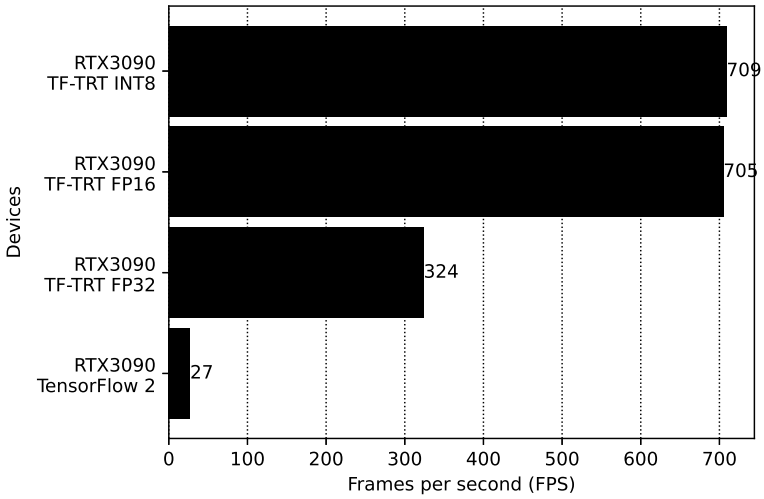


Figure 3.31: Processing framerate in the reference GPU NVIDIA RTX3090 considering RAW TensorFlow 2 and the optimised models for NVIDIA Tensor cores. INT8 report to the model's weights quantised into 8-bit integers, FP16 to weights into 16-bit floating-point, and FP32 to weights into 32-bit floating-point.

Despite these advancements, performance disparities in evaluation metrics were observed among embedded platforms (Figure 3.32). Limitations in memory and device

capacity prevented some model's evaluation on the NVIDIA Jetson Nano GPUs. The NVIDIA Jetson TX2 emerged as the top-performing device in these metrics, capable of compiling models only in FP32 and FP16 due to memory constraints that precluded INT8 quantisation. The NVIDIA Jetson TX2 achieved a commendable balance between precision and recall, maintaining a stable F1 score. Conversely, the TPU was identified as the least effective device, with quantisation significantly altering the model's weights and sacrificing resolution, which diminished both precision and recall, and consequently, the F1 score. FPGAs offered a middle ground, balancing the performance metrics by compensating for decreased recall with increased precision, or vice-versa, which helped stabilise the F1 score across devices. The mAP analysis corroborates these findings.

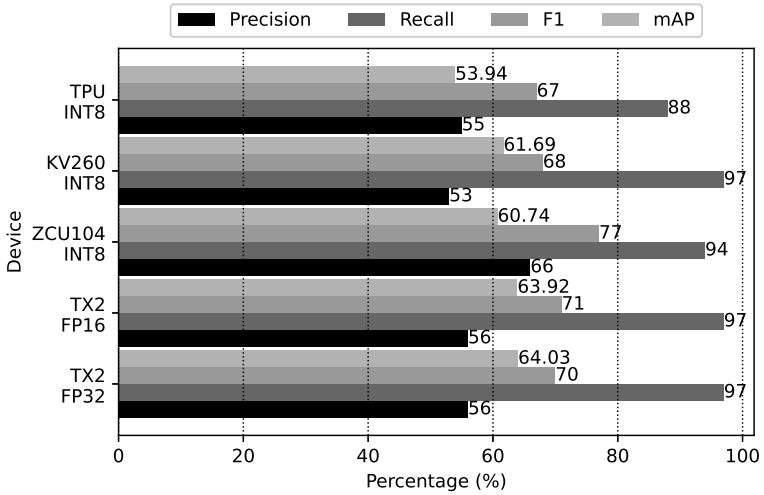


Figure 3.32: Inference performance in the evaluation metrics in the edge computing devices. INT8 report to the model's weights quantised into 8-bit integers, FP16 to weights into 16-bit floating-point, and FP32 to weights into 32-bit floating-point.

Figure 3.33 depicts the inference speeds across the various devices under study. The GPU was the slowest among the heterogeneous platforms. Using FP16 instead of FP32 resulted in a 1.6 times improvement in speed. However, the model could not be compiled and quantised to INT8. On the other hand, FPGAs demonstrated superior speed. Utilising a single DPU, these devices were 5.6 times faster than the NVIDIA Jetson TX2 FP32 and 3.4 times faster than both the NVIDIA Jetson TX2 FP16 and TPU. Employing two DPUs from ZCU104 allowed the inference speed to reach 25 FPS.

To enhance comprehension of the effects of quantisation or the impact of changing variable types, figures 3.34, 3.35, and 3.36 delineate the networks' performance across various evaluation metrics for each class. The bunches of berry-closed grapes, as depicted in figure 3.35, emerge as the most stable and predictable category. Alterations in the network's weights do not significantly influence the performance detection of evaluation metrics. Conversely, the bunches of berry-corn size grapes and trunks, exhibited in figure 3.37 and appendix B, present more challenging attributes. The berry-corn size grape bunches are notably small

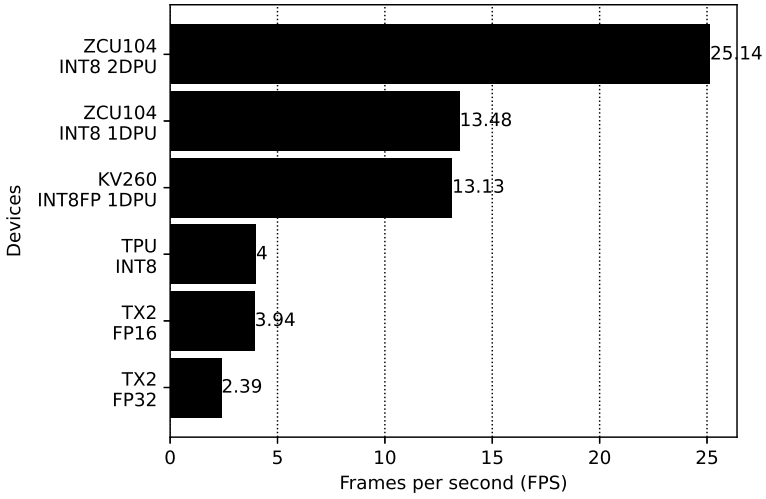


Figure 3.33: Processing framerate in the edge computing devices. INT8 report to the model's weights quantised into 8-bit integers, FP16 to weights into 16-bit floating-point, and FP32 to weights into 32-bit floating-point. FPGAs can have multiple DPU cores: 1DPU remains to the use of a single DPU and 2DPU is the simultaneous use of 2 DPU cores.

– appearing shortly after inflorescence – and bear a colour resemblance to the background. Trunks, on the other hand, display a high variability in shape and size. Moreover, the network often misidentifies many masts within vineyards as the trunks of vines.

The process of quantisation, when operating under the constrained resources of TPU, diminishes the count of detections (as shown in figures 3.34 and 3.36), which, in turn, lowers the TPU's recall (3.3). This reduction in the number of detections also decreases the incidence of FP and, consequently, impacts the TPU's precision (3.4). The marginal case, where quantisation actually mitigates the network's noise and enhances the performance detection of evaluation metrics, is unveiled in ZCU104, further supported by Gong *et al.* [272].

In mobile systems that utilise heterogeneous platforms, especially those reliant on battery power, it is crucial to manage power consumption effectively. According to the literature, these devices are prevalent in mobile applications. As such, Figure 3.38 illustrates the power consumption for various devices. It is observed that while the energy consumption for inference across all devices is relatively similar, there is a significant variation in power usage during their operating system (standby) operations.

We also aimed to assess the ability to accelerate the inference process by focusing solely on the FPGA's PL. This involves deploying ANNs as an FPGA IPs through the FINN framework. Parallel to the development with Vitis-AI, FINN is designed to translate ANN operations into logic levels. However, it's crucial to maintain a reliable metric performance for the classification problem. The performance of various networks deployed in FINN, evaluated using different metrics for the RG2C dataset, is documented in Figure 3.39. Data confirms that the networks deliver consistent and dependable performance. Thus, these networks are suitable for use in robotic applications requiring rapid attention mechanisms. Figures 3.40, 3.42, and

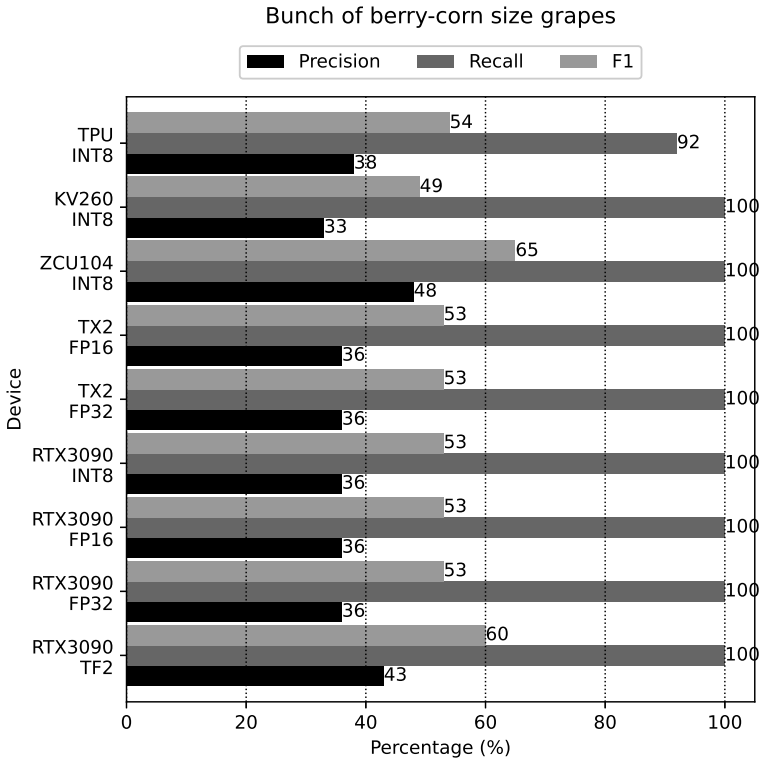


Figure 3.34: Inference performance for the evaluation metrics in the different heterogeneous devices for the class of bunches of berry-corn size grapes. INT8 report to the model's weights quantised into 8-bit integers, FP16 to weights into 16-bit floating-point, and FP32 to weights into 32-bit floating-point. FPGAs can have multiple DPU cores: 1DPU remains to the use of a single DPU and 2DPU is the simultaneous use of 2 DPU cores.

3.44 showcase some instances of FNs for the different models, whereas Figures 3.41, 3.43, and 3.45 display instances of FPs. These examples highlight certain images, specifically partials of trunks, that are frequently misidentified as grapes.

The speed of inference was further evaluated, as detailed in (3.11). Figure 3.46 illustrates the frame rate for different models deployed on the FPGA's PL using the FINN framework. This comparison clearly shows the advantage of utilising FINN models over the fastest models from the previous analysis, notably the Vitis-AI on ZCU104. MobileNet v1, with four bits for both weights and biases, achieved speeds up to 263 faster. Conversely, CNV only managed to be 32 faster. Given that CNV is significantly smaller than MobileNet v1, there is potential for further optimisation and speed improvements for CNV compared to MobileNet v1.

$$FPS_{chunk} = \frac{1}{t_{avg}} \tag{3.10}$$

$$FPS_{image} = FPS_{chunk} \times N_c \tag{3.11}$$

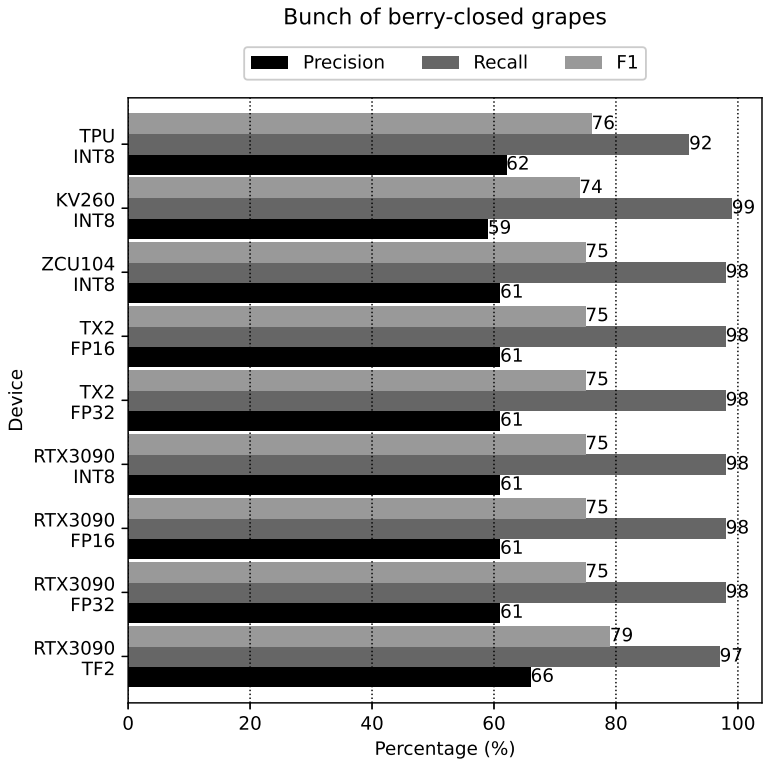


Figure 3.35: Inference performance for the evaluation metrics in the different heterogeneous devices for the class bunches of berry-closed grapes class. INT8 report to the model's weights quantised into 8-bit integers, FP16 to weights into 16-bit floating-point, and FP32 to weights into 32-bit floating-point. FPGAs can have multiple DPU cores: 1DPU remains to the use of a single DPU and 2 DPU is the simultaneous use of 2 DPU cores.

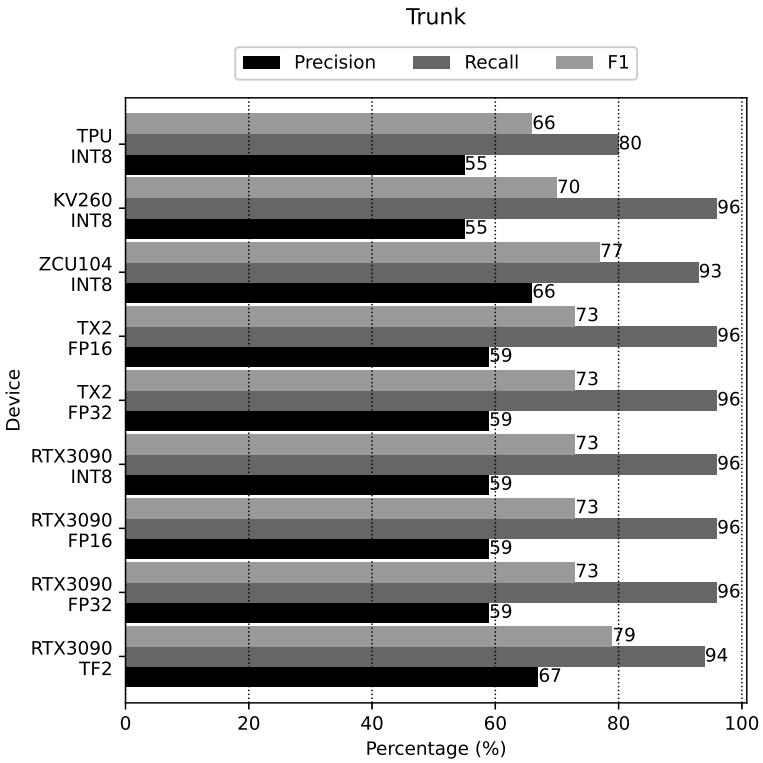


Figure 3.36: Inference performance for the evaluation metrics in the different heterogeneous devices for the class of trunks. INT8 report to the model's weights quantised into 8-bit integers, FP16 to weights into 16-bit floating-point, and FP32 to weights into 32-bit floating-point. FPGAs can have multiple DPU cores: 1DPU remains to the use of a single DPU and 2DPU is the simultaneous use of 2 DPU cores.

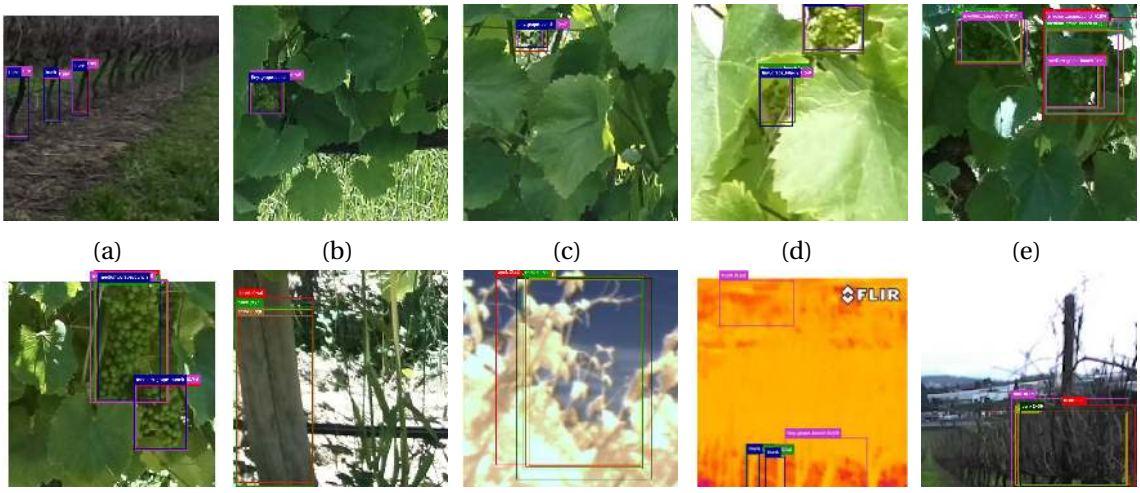


Figure 3.37: Some sample images with the inference results. Details of this figure were added to appendix B in figures B.1 to B.10. Blue – ground-truth; light green – NVIDIA RTX3090 TF2; orange – NVIDIA RTX3090 TF-TRT FP32; brown – NVIDIA RTX3090 TF-TRT FP16; dark yellow – NVIDIA RTX3090 TF-TRT INT8; red – AMD-Xilinx Kria KV260; dark green – AMD-Xilinx ZCU104; pink – Coral Dev Board TPU

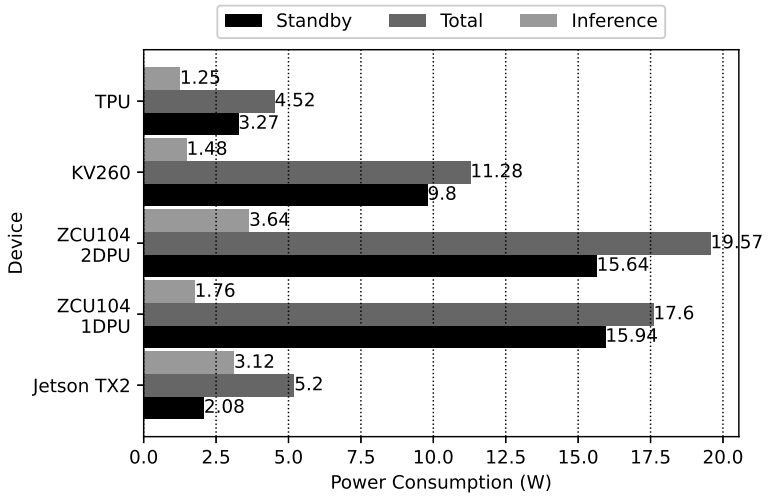


Figure 3.38: Power consumption.

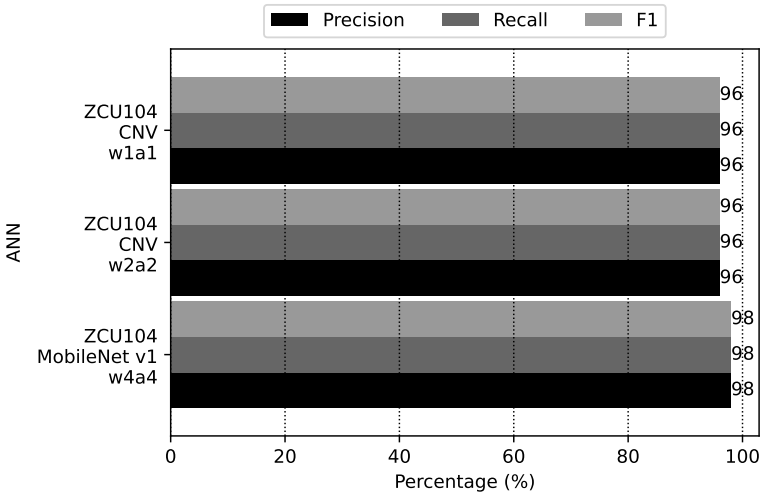


Figure 3.39: Inference performance in the evaluation metrics for FPGA FINN models considering the RG2C dataset. The i, j values in $w_i a_j$ report the number of bits being considered for the layers' weights (w) and biases (or activations, a).

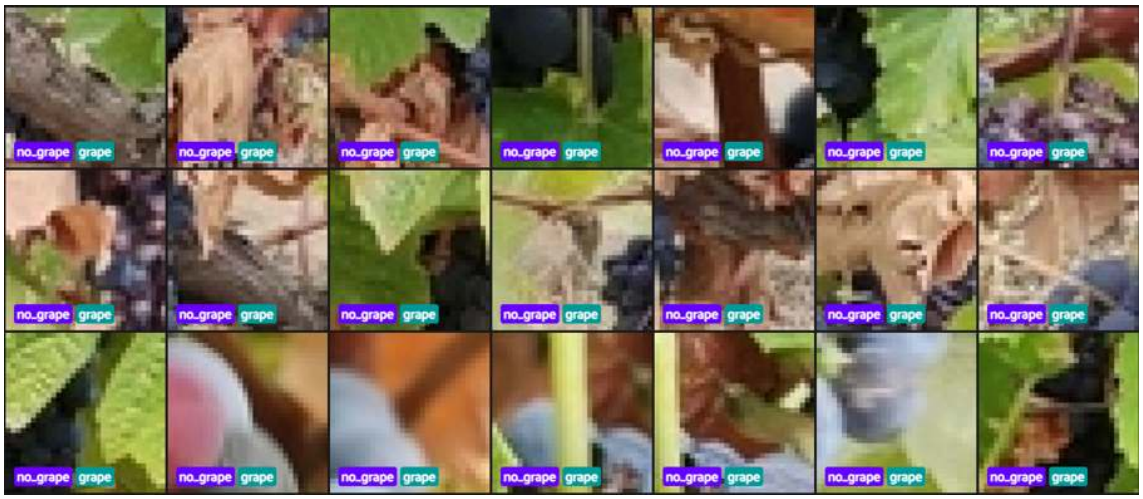


Figure 3.40: Some false negatives for FINN MobileNet v1 w4a4. Cyan labels are the ground truth and purple labels are the predictions.

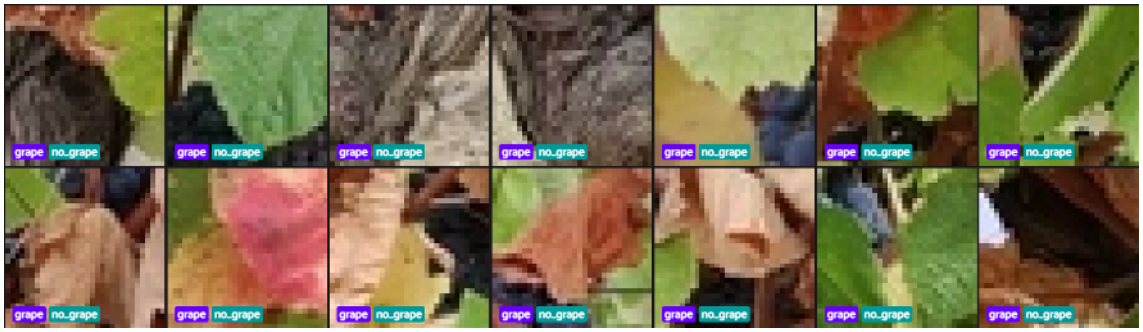


Figure 3.41: Some false positives for FINN MobileNet v1 w4a4. Cyan labels are the ground truth and purple labels are the predictions.

Figure 3.42: Some false negatives for FINN CNV w2a2. Cyan labels are the ground truth and purple labels are the predictions.

Figure 3.43: Some false positives for FINN CNV w2a2. Cyan labels are the ground truth and purple labels are the predictions.

Figure 3.44: Some false negatives for FINN CNV w1a1. Cyan labels are the ground truth and purple labels are the predictions.

Figure 3.45: Some false positives for FINN CNV w1a1. Cyan labels are the ground truth and purple labels are the predictions.

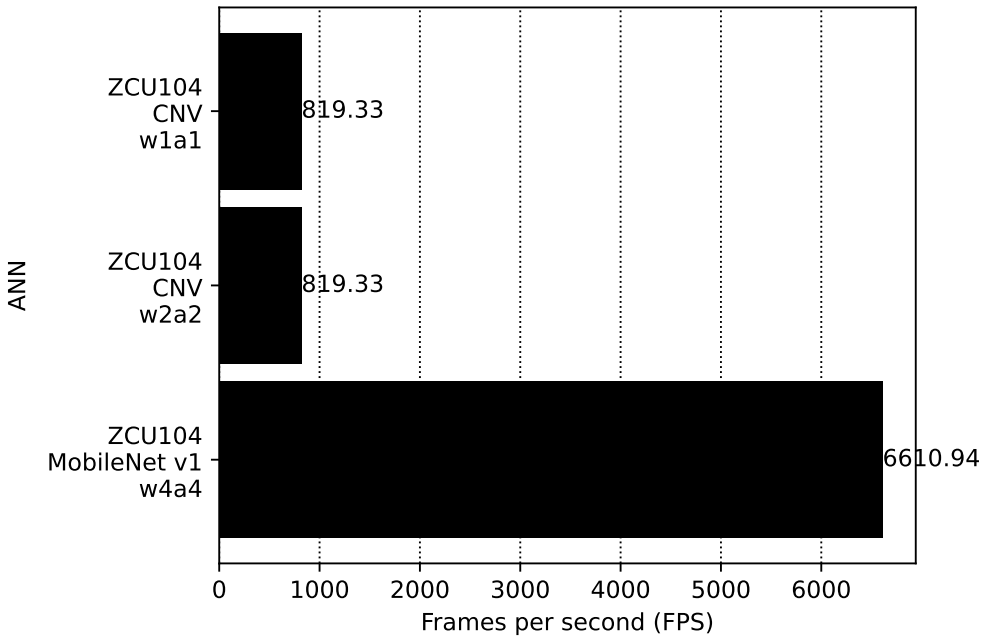


Figure 3.46: Processing frame rate for FPGA FINN models considering the RG2C dataset. The i, j values in $w_i a_j$ reports number of bits being considered for the layers' weights (w) and biases (or activations, a).

3.3.3 Maturity assessment

Two methods were used to estimate the maturity stage of the fruit: DL models and colour features with Gaussian curves, through Gaussian mixtures.

3.3.3.1 HSV Colour Space Model Classifier

The model was developed by analysing the mean of the Hue histogram for each sample and its correlation with the respective class, leading to the derivation of a quadratic function as the statistical classifier. This derivation was achieved through the application of the least squares algorithm. The correlation and the derived quadratic function are illustrated in Figure 3.47.

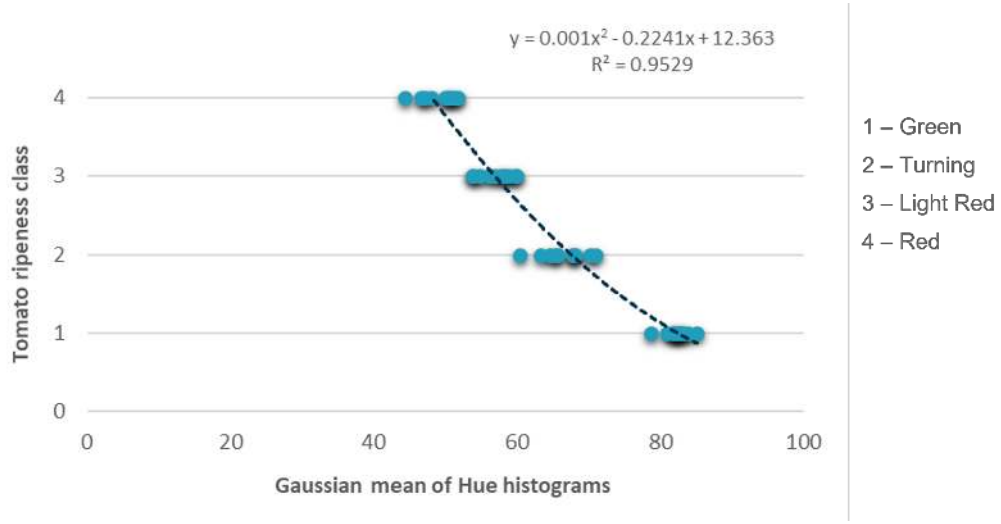


Figure 3.47: Correlation between the Gaussian mean of the Hue histogram for each sample and its respective class. This also includes the plot of the trend line, the equation, and the R^2 value for the quadratic function obtained.

To classify tomatoes into specific categories, it was necessary to establish thresholds for each class. This was achieved by adjusting the equation of the quadratic function, specifically by adding 0.25 to the independent term, as shown in the equation (3.12):

$$f(x)=\begin{cases} \text{Green}, & \text{if } 000.1x^2-0.2241x+12.613 \leq 1.5 \\ \text{Turning}, & \text{if } 1.5<000.1x^2-0.2241x+12.613 \leq 2.5 \\ \text{Light Red}, & \text{if } 2.5<000.1x^2-0.2241x+12.613 \leq 3.5 \\ \text{Red}, & \text{if } 000.1x^2-0.2241x+12.613 > 3.5 \end{cases} \tag{3.12}$$

This equation culminated in the creation of the ultimate model. For a specific image, the model processes the bounding box coordinates of the fruits to be classified (input) in a single pass. It segments the regions of interest using the HSV colour space model and extracts colourimetric information based on the Hue channel. Subsequently, the Gaussian Mixture

probabilistic model generates a histogram and calculates its mean. This mean value, through the statistical classifier, produces an output, classifying the fruit into one of the predefined categories.

3.3.4 Tomato Ripeness Classification: Deep Learning vs. HSV Colour Space Models

Regarding the classification problem, Table 3.10 presents the results for the different metrics used to evaluate two DL models and the proposed HSV colour space model. All models demonstrated a strong ability to distinguish the Red tomato class, achieving precision rates above 80 %, with the HSV colour space and YOLO v4 models notably achieving approximately 89 %. However, there was a significant challenge in classifying all relevant objects of this class, with the exception of the SSD MobileNet v2 model, which attained a recall of 84 %.

In contrast, the task of detecting and classifying green tomatoes significantly impacted the performance of the SSD MobileNet v2 model. Both the YOLO v4 and HSV colour space models delivered excellent results in terms of precision and recall, with the proposed model notably exceeding 98 % in both metrics.

A considerable challenge was observed across all models in distinguishing between the Turning and Light Red tomato classes, especially for the SSD MobileNet v2 and HSV colour space models, which failed to detect even half of the relevant objects in these two classes, resulting in poor recall rates of around 43 %.

The Macro F1 score and Balanced Accuracy metrics offer a more comprehensive understanding of the results obtained. The YOLO v4 model outperformed both the SSD MobileNet v2 and HSV colour space models with a Macro F1 score of 74.16 %. The reduced performance of the SSD MobileNet v2 model was largely due to its limited effectiveness in differentiating between Turning and Light Red tomatoes.

In terms of Balanced Accuracy, the YOLO v4 model again demonstrated superior performance in classifying the fruits, achieving a Balanced Accuracy of 68.87 %. However, the HSV colour space model also achieved a competitive result of 68.10 %, primarily due to its robust ability to classify Green and Red class tomatoes.

Figure 3.48 illustrates cases of poor classifications due to the difficulty of the models in distinguishing some tomato classes. An interesting aspect is how the DL models addressed these challenges. For some fruits, the models generated two bounding boxes of different classes. Although the tomatoes were correctly detected, at least one of the class predictions was incorrect, ultimately affecting the classification results obtained.

3.4 Discussion

For detecting fruits in their natural context within cultivars, we utilised DL models, specifically YOLO v4 Tiny, SSD MobileNet v2, SSD Inception v2, SSD ResNet 50, and SSD ResNet

Table 3.10: Classification results of the over the evaluation metrics, considering each model's best-computed confidence threshold.

DL Models	Fruit Ripeness	Precision	Recall	Macro F1-Score	Balanced Accuracy
SSD MobileNet v2	Green	77.27%	70.09%	65.93%	62.70%
	Turning	59.38%	55.88%		
	Light Red	60.61%	40.82%		
	Red	80.77%	84.00%		
YOLOv4	Green	85.38%	84.66%	74.16%	68.87%
	Turning	70.77%	67.65%		
	Light Red	76.32%	59.18%		
	Red	88.89%	64.00%		
HSV Colour Space	Green	98.24%	98.31%	70.93%	68.10%
	Turning	50.00%	63.24%		
	Light Red	58.33%	42.86%		
	Red	89.47%	68.00%		

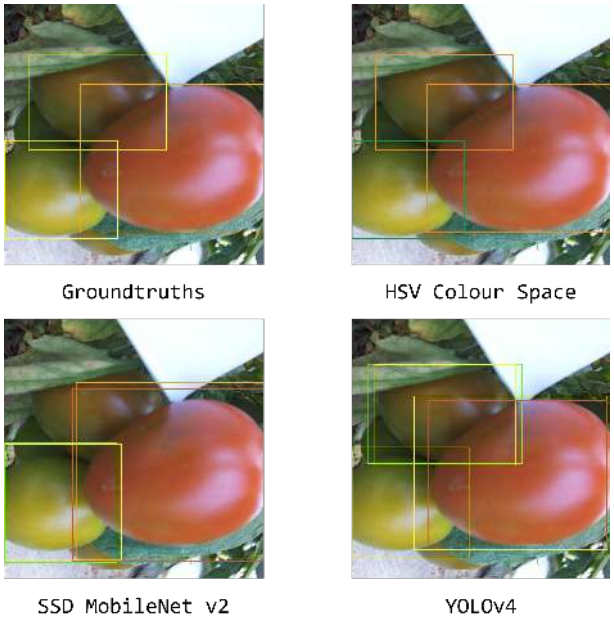


Figure 3.48: Classification results comparison between the two DL models and the HSV colour space model with the ground truth annotations. Green, Yellow, Orange and Red bounding boxes represent Green, Turning, Light Red and Red tomato predictions, respectively.

101. A relevant aspect of leveraging ANNs is their thorough characterisation, including the confidence threshold. For optimisation, we selected the value that maximises the F1 score in the valiation set, proving effective and yielding reasonable outcomes. The validation set was instrumental in monitoring the training process and optimising the confidence threshold, whereas an independent test set offered more reliable results, as shown in tables 3.7 and 3.8. An additional calibration process for confidence threshold optimisation proved beneficial, as illustrated in figure 3.20 and table 3.7, enhancing network sensitivity and improving object detection.

The incorporation of a complementary test set proved crucial. Initially, the SSD ResNet 50 model was identified as the top performer using a validation set; however, it ranked as the second-worst ANN with the test set. Thus, for this study, simpler networks, namely YOLO Tiny v4 and SSD MobileNet v2, emerged as the most adept at solving our problem. SSD Inception v2 also showed promising results, ranking second. The more complex and larger SSD ResNet 101 model was the least effective for detecting tomato fruits in greenhouse cultivars, likely due to overfitting caused by the dataset's small size and variability. To address this, increasing data variability and collecting additional data under various conditions are recommended. Additionally, a thorough review of the dataset can be beneficial by removing repeatitive data and optimising the uniqueness representative of data, using data-centric strategies.

This essay also examined the models' limitations under specific conditions such as partial occlusion and darkness. Smaller models reported fewer FPs and generally performed better. However, significant overlap observed in SSD ResNet 101 suggests a need for optimising the NMS strategy.

The literature, as reviewed in chapter 2, extensively explores perception algorithms for detecting and classifying various objects, distinguishing between classical image analysis and ML algorithms. Recent advancements focus primarily on DL models and their superior ability to identify complex objects. Our findings for detecting tomato fruits are consistent with recent literature, effectively identifying visible and accessible fruits, especially during early maturity stages. Challenges persist in detecting partially occluded and darkened fruits.

To enhance the efficacy of robotic systems, models should be executed on effective, low-power devices to achieve near real-time inference. For this purpose, several heterogeneous platforms were evaluated, including embedded GPUs, TPU, and FPGAs.

Upon comparing all benchmarked devices, it is evident that for achieving the best performance in terms of evaluation metrics and time efficiency, high-performance GPUs remain the superior choice. However, it is crucial to mention that this study did not include other high-performance devices, such as server-side FPGAs (e.g., AMD-Xilinx Alveo family), but rather focused on low-power heterogeneous devices suitable for integration into mobile systems like robots. Despite FINN networks demonstrating superior performance in evaluation metrics and faster inference, it is important to recognise that these networks were evaluated using a classification problem with a different dataset, not an object detection model. The process of compiling the network across different devices did not significantly alter the model's performance in evaluation metrics, although some resolution reduction occurred.

Within the realm of edge computing devices, even though GPUs exhibited the highest performance in evaluation metrics, FPGAs were considerably faster. It is important to note that this study only benchmarked the core model, excluding pre-processing and post-processing layers. Therefore, FPGAs, with their capabilities, could better parallelise these processes. Besides the DPU, they also incorporate the PL and an onboard GPU. While the PL was utilised through FINN models for classifying small image segments, the GPU was not considered but could potentially be employed for primarily pre-processing or post-processing tasks.

Attempts to benchmark NVIDIA Jetson Nano 2 GB and 4 GB were made, but their limited capabilities hindered the conversion and compilation of the model into TF-TRT, leading to their exclusion from this research analysis.

Figure 3.37 presents images from the test set alongside the detections made by each device and the ground truth. Extended versions of these images can be found in appendix B from figures B.1 to B.10. Generally, all devices managed to effectively detect the target objects, with most detections closely matching across different samples. Figure 3.37e illustrates a grape detected twice due to its size and partial occlusion by a leaf. From images 3.37b and 3.37c, it is evident that berry-corn size grapes pose the greatest detection challenge. This issue is highlighted in figure 3.34, where the F1 score is generally below 60 %. However, this may not significantly impact practical applications as other landmarks can aid in robot localisation. Nonetheless, detecting trunks and bunches of berry-closed size grapes is important for tasks such as monitoring or harvesting, with detection ratios for these classes ranging between 70 % to 80 % in figures 3.35 and 3.36, making them viable for practical applications. Therefore, the overall low mAP of about 60 % demonstrated in figures 3.30 and 3.32 can be attributed to the poor detection ratio of bunches of berry-corn size grapes. Figures 3.37f to 3.37j display various detection errors introduced by different model versions. Further improvements should focus on optimising the ANN's structure and parameters and closely reviewing the dataset. Hyper-parameters such as the confidence threshold could be optimised [C42]. The metric results also suggest possible misannotations of objects that are being correctly identified, indicating that some objects, like trunks, could be successfully detected by the model but were not labelled in the ground truth data.

In the literature review, no publications were found that explored the application of RetinaNet ResNet 50 or SSD ResNet 50 FPN on heterogeneous devices. Therefore, a direct comparison with state-of-the-art results is not feasible. Although the results indicate that our experiments were slightly slower than state-of-the-art findings, RetinaNet ResNet 50 is more complex than YOLO and SSD MobileNets. Considering the fast inference times with competitive evaluation performance rates, which at times are comparable to YOLO results from the literature, we conclude that this work's research is suitable for near real-time applications.

In the study conducted by Aguiar *et al.* [C47], the VineSet dataset was analysed using an SSD object detection model that incorporated two backbone feature extractors, MobileNet v1 and Inception v2, within a USB Coral Accelerator TPU. The outcomes were mAP scores of 66.96 % for MobileNet v1 and 55.78 % for Inception v2, notably achieved without considering the trunk's class. The omission of the trunk's class allows for a parallel with our results, suggesting that including it might degrade performance metrics. Thus, it's inferred that the dataset's limitations are being approached, necessitating a thorough review of labelling to pinpoint potential misannotations. Additionally, Aguiar *et al.* citeCAguiar2021 explored the optimal confidence score for metric optimisation through an inference threshold analysis, whereas a standard confidence score of 30 % was employed here.

MobileNet architectures, designed for speed and mobile application efficiency, and

Inception networks, which are simpler and faster than ResNet networks, showcased their capabilities within a TPU. Specifically, performance rates of 158.98 FPS for MobileNet and 38.36 FPS for Inception were recorded. This comparison reveals a faster performance compared to our results, although the distinction in network performance between a Google Coral USB Accelerator Edge TPU and a Google Coral Dev Board TPU remains ambiguous. An inferred average power consumption of 2.5 W for the USB stick was noted, excluding broader computer maintenance and processing power requirements.

Furthermore, FINN models demonstrated noteworthy outcomes. Figures 3.40 to 3.45 depict dataset images alongside instances of FPs and FNs. Analysis of these images indicates that false detections frequently arose from confusion with parts of the vines' trunks, while missed detections often involved grape partials potentially mistaken for shadows by low-resolution networks. For a comprehensive understanding of these errors, the application of explainable artificial intelligence (XAI) techniques, as suggested by [273], is advised. Despite visually lower performance metrics, the high inference speeds of FINN networks render them suitable for real-time object classification in high frame rates, making them intriguing for attention mechanisms within the proposed active perception architecture.

In terms of our findings, the TPU emerges as the superior choice when prioritising power reduction, despite minor performance variations across evaluation metrics and a slower inference speed. For applications balancing power consumption and inference speed, AMD-Xilinx Kria KV260 shows promise. It's crucial to acknowledge that both AMD-Xilinx Kria KV260 and AMD-Xilinx ZCU104 operate on a standard PetaLinux image from AMD-Xilinx[29], which activates all FPGAs' resources, many of which may be unnecessary. Thus, a refined analysis with an optimised PetaLinux image could offer a more accurate assessment of FPGAs' power consumption.

Finally, we explored a fixation mechanism for assessing the maturity stage of fruits, focusing on the development and evaluation of two DL models for tomato detection and classification. These models were compared against a developed model based on the HSV colour space, which classifies fruits according to their colour/ripeness through images from two datasets specifically collected for this purpose.

Overall, both DL models proved to be sufficiently generic to achieve successful tomato detection. The performance was consistent across both the validation and test sets, with the YOLO v4 model demonstrating promising outcomes and outperforming the SSD MobileNet v2 model. Notably, the application of a filtering process based on a threshold significantly benefitted only the YOLO v4 model. The SSD MobileNet v2 model displayed consistent results irrespective of the confidence threshold, suggesting its suitability for use without a filtering process. These models demonstrated the capability to detect tomatoes at various stages of ripeness, even in complex scenarios involving occlusions, overlaps, and variations in lighting conditions. Despite being one-stage detectors, which could potentially compromise accu-

[29] See Advanced Micro Devices, Inc. "PetaLinux tools." (2022), [Online]. Available: https://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html (visited on 08/05/2022).

racy, the SSD MobileNet v2 and YOLO v4 models did not show a considerable loss in accuracy, exhibiting good inference times. Combined with their precision and recall outcomes, these models are viable for integration into real-time CV systems.

When comparing our results with those of other researchers, our models showed superior performance to those presented by Yuan *et al.* [122], and Ruparelia *et al.* [137], which, despite achieving high precision rates, failed to detect all relevant objects, resulting in an overall F1 score ranging between 60 % to 66 %. Our models were only outperformed by heavily modified versions of the YOLO v3 and YOLO v3 Tiny models, as reported by Chen *et al.* [134], Liu *et al.* [136], Lawal [138], and Xu *et al.* [117], achieving an F1 score consistently above 90 %. However, it is important to note that most of these models were trained on a single class, disregarding the fruit's ripeness stage, and in some cases, were trained predominantly or exclusively on images of red tomatoes, which are naturally easier to detect due to the stark colour contrast with the background. To the best of our knowledge, the application of DL models for tomato detection in greenhouse conditions with more than three classes has not been reported in the literature. Hence, the results obtained for the SSD MobileNet v2 and YOLO v4 models, which were minimally modified and trained on four classes using images from real greenhouse environments, are notably promising, especially considering the YOLO v4 model that achieved an F1 score close to 86 %.

In terms of maturity classification capabilities, the DL models exhibited divergent behaviours. The classification results did not align with the detection outcomes, primarily due to the models' difficulties in distinguishing between intermediate classes (Turning and Light Red). The colour similarity between these two classes could potentially interfere with the classification accuracy. Moreover, the differentiation between these classes can be subjective and imprecise even to the human eye, which is reflected in the annotation process and complicates the models' learning process. Nevertheless, both models achieved impressive results in classifying green and red tomatoes.

As an alternative approach to maturity classification, an HSV colour space model was proposed. This model excelled in classifying green tomatoes but faced challenges in distinguishing between the 'Turning Red ´ and 'Light Red' classes. This issue might stem from the use of a Gaussian mixture model, an iterative algorithm that is not optimal and can yield varying results in boundary cases based on initial assumptions. Despite these challenges, the HSV colour space model outperformed the SSD MobileNet v2 model significantly and approached the performance of the YOLO v4 model, especially in terms of Balanced Accuracy. The results are particularly noteworthy considering that the DL models required training on a large dataset (7393 images), whereas the HSV colour space model was developed using only 40 images (10 from each class). The simplicity and intuitiveness of the proposed model, along with its ability to adjust the number of classes and modify confidence thresholds for each class, present distinct advantages.

Sorting fruits based on their ripeness stage is primarily associated with post-harvest processes. Consequently, most studies on fruit classification occur in structured environments,

akin to those in processing industries after harvest. The only significant paper evaluating the ability of DL models to classify tomatoes is by Mutha *et al.* [140], who achieved an average accuracy of 94.67 % using a YOLO architecture for classifying three distinct tomato classes. However, the tomato images used in this study do not represent the agricultural environment broadly, nor a greenhouse specifically, and the test set comprised only 23 images. In terms of more traditional methods, Benavides *et al.* [105] reached an accuracy of 77 % by calculating the aggregated per cent surface area below certain Hue angles for classifying tomatoes into six ripening stages. Utilising the HSV colour space to classify tomatoes into five classes, Gupta *et al.* [114] proposed a colour histogram matching method, and Malik *et al.* [106] applied a K-Nearest Neighbour approach, achieving accuracies of 97.20 % and 100 %, respectively. Despite these studies showing promising results, they were all conducted with images of tomatoes detached from the plant, in artificial settings with stable and uniform backgrounds. This highlights the novelty of our work, which was conducted with images from a real greenhouse environment, focusing on evaluating both the detection and classification of tomatoes at different ripeness stages.

In summary, while DL models excelled in the detection task, they faced challenges in classification, with the HSV colour space model surpassing the SSD MobileNet v2 model. Each model has its pros and cons: DL models are more demanding in terms of time and computation but excel in complex scenarios, a capability that significantly impacts the performance of simpler methods like the HSV colour space model. An innovative solution could be to modify and combine a DL model with the proposed HSV colour space model through ensemble modelling to develop a framework that can accurately detect and classify a larger variety of fruits without substantial accuracy loss. Efforts in this direction include the study by Ko *et al.* [275], proposing a novel method for classifying tomato ripeness through multiple streams of CNN and stochastic decision fusion. Yet, this study was also conducted in an artificial environment, highlighting an ongoing gap in applying these models in real agricultural settings. Our publicly available datasets further contribute to scientific advancement, offering resources to train and develop more precise visual perception solutions for use in greenhouse or vineyards contexts.

3.5 Conclusions

In this chapter, we addressed the challenge of detecting and identifying fruits in complex scenarios within cultivar environments. To effectively explore this topic, we developed various datasets of agricultural objects, grouped by their agricultural proximity. These include the AgRobTomato and RPiTomato datasets, which comprise RGB images of tomatoes at different stages of ripeness, and the VineSet and RG2C datasets, featuring RGB images of grape bunches and vine trunks.

For these datasets, we experimented with different DL models, all based on the same SSD and YOLO architectures. We tested various backbones for these architectures, each with dif-

fering levels of complexity. The lighter backbones, MobileNet v2 and Inception v2, demonstrated superior performance, although ResNet 50 model also showed promise. The YOLO v4 Tiny model matched the performance benchmarks reported in the literature, offering excellent results and rapid inference speeds. This study underscores the necessity of developing and maintaining specialised datasets for agricultural applications due to the specific demands of the environment and the inadequacy of general-purpose datasets, such as the OID, in detecting fruits and other objects in cultivar settings.

In addition to having specialised models for detecting fruits and other objects in cultivars, it is essential to have efficient and dedicated hardware to process these complex models in robotic systems. We evaluated a range of heterogeneous devices. TPUs are noted for their low power consumption, small size, and efficiency but have limitations in compatibility with certain operations. As a result, FPGAs emerge as a viable alternative, offering a fully programmable system across different levels. Furthermore, in contexts of classification, FPGAs can outperform other devices with very small models, capable of inferring thousands of images per second. Additionally, in inference scenarios, FPGAs are the most power-efficient devices, with their standby power consumption optimisable by disabling certain features and peripherals.

Given the simultaneous detection of multiple objects, fixation mechanisms are crucial in selecting the most accessible fruits. Algorithms for assessing ripeness assist in identifying fruits ready for harvest. In this evaluation, both DL models and colour feature algorithms proved equally effective. However, the use of colour feature algorithms with a mixture of Gaussians required fewer images and offered more predictable modelling and easier calibration adjustments.

With a comprehensive system capable of detecting fruits in a robotic framework, the subsequent step involves integrating this system with a harvesting mechanism. This integration aims to precisely detect the 3D positions of fruits and guide a manipulator harvester to efficiently harvest them. That can be achieve through active perception strategies.

Chapter 4

Towards active perception

In agriculture, tasks such as fruit monitoring or harvesting necessitate the accurate perception of objects' spatial positions. However, RGB-D cameras face limitations in open-field environments due to lighting interferences. In this study, we explored the application of Histogram Filters (Bayesian Discrete Filters) for estimating the position of fruits' in the tree. Additionally, we developed an observability optimiser to enhance the autonomy of the MonoVisual3DFilter, making it more active and aware of its environment. An enhancement of the observability optimiser, incorporating Kalman filters, granted it autonomous perception capabilities for estimating the fruit positions. The algorithms and solutions were primarily evaluated in a simulation environment, although some laboratory tests were also conducted.

The current chapter also comprehends studies in press for scientific journals, namely [C44], and a submitted conference paper [C45].

Similar to the previous chapter, this chapter adheres to the IMRaD protocol.

4.1 Introduction

Until now, our focus has been primarily on detecting fruits, specifically tomatoes and bunches of grapes, within agricultural settings. However, for effective harvesting, it is crucial to estimate their 3D positions accurately.

In the section 2.6.2, a thorough review of the literature is performed towards assessing valid solutions to estimate the position of the objects in the 3D space. The most technologically advanced works use RGB-D sensors that provide all the information directly. However, these sensors usually malfunction in agricultural scenes because of natural interferences. Therefore, using monocular sensors equipped with predictable algorithms could be an approachable solution.

As reviewed in the same section, Bayesian Histogram Filters are suitable for identifying the 3D position of objects without requiring detailed models. Therefore, we applied the histogram filter to identify the 3D position of tomatoes in a testbed using a monocular camera mounted on a robotic arm, using an eye-in-hand model, in a solution we termed MonoVisual3DFilter. Initially, the arm used fixed multi-viewpoints to observe the tomatoes from various perspectives.

Given the dependence on fixed viewpoints, we developed the best viewpoint estimator (BVE) in the second stage to optimise fruit observation for a better assessment of their real 3D position. Additionally, the BVE was enhanced with an extended Kalman filter (EKF) that iteratively refines the fruit's position for the optimisation of the next viewpoint.

This work explores the use of Bayesian Histogram Filters for estimating the 3D position of objects within the reach of a robotic manipulator using a monocular camera. The implemented algorithm underwent evaluation in both simulated and testbed conditions in the laboratory, focusing on fruit detection. Subsequently, efforts were made to enhance the MonoVisual3DFilter by incorporating an active perception strategy that enables autonomous target control for the manipulator using the BVE, which, when supported by an EKF, was capable of singularly estimating the fruit position. The contributions of this current work include:

- Introducing and evaluating the MonoVisual3DFilter;
- Applying histogram filters to detect the centre position of objects accurately;
- Utilising the MonoVisual3DFilter for real-world applications, such as detecting fruits within the canopy of plants;
- Investigating the impact and benefits of various kernel types;
- Expanding the MonoVisual3DFilter to autonomously select optimal viewpoints; and
- Implementing an additional estimator for the 3D position of fruits based on the EKF.

4.2 Materials and Methods

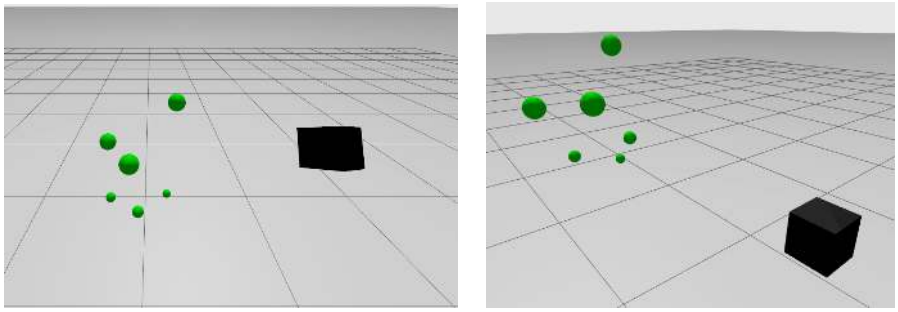
4.2.1 MonoVisual3DFilter

4.2.1.1 Real data and simulation

The development and testing of the MonoVisual3DFilter algorithm took place in two distinct settings: a simulated environment and a laboratory testbed, aiming to mimic near real-world conditions. Both settings utilised a monocular camera with fixed viewpoints.

A basic simulation environment was created using the Ignition Gazebo Simulator[1]. This setup included six spheres, with diameters of 5 cm and 10 cm, to evaluate the algorithm's performance and facilitate its implementation (Fig. 4.1). A bounding box camera was incorporated into the scene to detect the objects and assist the position estimation algorithm. During the MonoVisual3DFilter's execution, the camera was strategically moved to predetermined viewpoints to test and validate the estimation process. The camera employed object detection algorithms to identify the visible portions of the objects using bounding boxes.

[1]See Open Robotics. "Gazebo." (2023), [Online]. Available: https://gazebosim.org/ (visited on 05/12/2023).



(a) (b)

Figure 4.1: Simulated environment to validate the histogram filter effectiveness. Green spheres are the objects being detected, representing the tomatoes, and the black box is the bounding box camera looking at the spheres.

For the laboratory testbed, designed to replicate near real-world greenhouse conditions, plastic leaves and tomatoes were used (Figures. 4.2a and 4.2b). An OAK-1 camera[2] (Figure 4.2d) served as the bounding box camera, attached to the 6 DoF Robotis Manipulator-H (Figure 4.2c). This camera processed an object detection model, specifically trained to recognise tomatoes using the YOLO v8 tiny, based on a dataset by Magalhaes et al. [C42, C50] and section 3.2.1.1, and including samples of the plastic tomatoes from various angles. The manipulator moved to fixed viewpoints to ensure the visibility of the tomatoes, mounted on the mobile platform AgRob v16 from INESC TEC (Fig. 1.2). However, the 3D positioning of the tomatoes was calculated relative to the base frame of the manipulator.

The OAK-1 is a comprehensive system for bounding box cameras, designed by Luxonis Holding Corporation, featuring a single 12 MP RGB camera module. For on-system object detection, the camera module connects to the OAK-SoM[3]. The entire camera system utilises USB-C for device connectivity. The OAK-SoM is a SoM engineered for integration into both high-level and low-power systems, capable of processing ANNs. This camera module was integrated into the AgRob v16 robot (Fig. 1.2).

(a) Testbed with the tomatoes

(b) Plastic tomato

(c) Manipulator-H

(d) OAK-1 Camera

Figure 4.2: Simulated testbed in the laboratory to essay the histogram filter algorithm

[2]Luxonis Holding Corporation. "OAK-1." (2023), [Online]. Available: https://docs.luxonis.com/projects/hardwae/en/latest/pages/BW1093/ (visited on 09/26/2023).

[3]See Luxonis Holding Corporation. "OAK-SoM." (2023), [Online]. Available: https://docs.luxonis.com/projects/hardware/en/latest/pages/BW1099/ (visited on 09/26/2023).

This approach ensures that the MonoVisual3DFilter algorithm is rigorously evaluated under controlled conditions, facilitating robustness and reliability in object detection tasks.

4.2.1.2 Histogram Filter

Histogram filters have been extensively utilised in literature for the self-localisation and navigation of mobile robots [179, 180]. In this study, we plan to adapt histogram filters for determining the 3D position of tomatoes relative to the manipulator's base frame, an approach we refer to as MonoVisual3DFilter.

The histogram filter is a computationally demanding algorithm capable of estimating the relative positions of objects by calculating probabilities across a discretised space. Subsequently, it combines the probabilities from different viewpoints to deduce the localisation and the extent of the regions of interest (Figure 4.3).

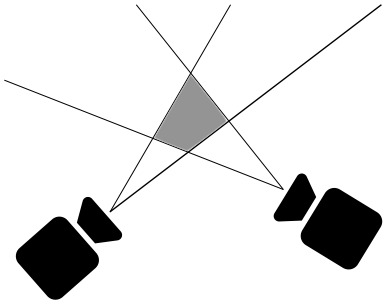


Figure 4.3: Intersection between multiple viewpoints in 2D plane

Histogram filters are essentially discrete Bayes filters applied to a continuous state space, as described by Thrun *et al.* [178].

The continuous state space is decomposed into a finite set of regions. Equation 4.1 outlines the discretisation of the state space, where X_t represents the random variable for the state of detected objects at time t. The term $\text{dom}(X_t)$ defines the state space, encompassing all possible values of X_t. A simple method to discretise a continuous state space is by using a multidimensional grid, with $\boldsymbol{x_{k,t}}$ denoting each grid cell.

$$\text{dom}(X_t) = \boldsymbol{x_{1,t}} \cup \boldsymbol{x_{2,t}} \cup \cdots \cup \boldsymbol{x_{K,t}} \tag{4.1}$$

To minimise computational efforts, only a portion of the state space is discretised, particularly considering the manipulator's reachability. Upon detecting a region of interest with the manipulator's camera from the initial viewpoint, the space behind the camera is segmented using a grid scheme. The probable space around the object is considered to be twice the manipulator's reach. This reachability of the manipulator provides sufficient margin to identify and assess some fruits within its range. The 3D decomposition is centred on the camera in the yOz plane, i.e., at $(0,0)$ in the camera's frame, and extends outward by the manipulator's reachability radius in Ox. Once segmented, the discrete state space is fixed, and only $\boldsymbol{P}_{\text{dom}(\boldsymbol{X_t})}$ undergoes updates.

Operating the histogram filter requires shifting the camera to various strategic viewpoints and refreshing the probabilities grid. At the onset of space decomposition, a probabilities matrix associated with each cell is initialised to one, implying that initially, the object of interest could be anywhere within the decomposed space, i.e., $\mathrm{dom}([X_0]) = [1 \ldots 1]$.

To estimate the positions of objects, $\mathrm{dom}(\boldsymbol{X_t})$ is updated at each new viewpoint. Each cell, $x_{i,t}$, within the decomposed state space, is converted from the manipulator's base frame to the image's frame. The probability of an object being within a given cell, $x_{i,t}$, considering the viewpoint, is calculated as shown in (4.2). The revised probability of an object's presence within cell $x_{i,t}$ is determined by (4.3).

$$p(\boldsymbol{x_{i,t}}|\boldsymbol{viewpoint_k}) = \frac{1}{N}\sum_{j}^{N} p(\boldsymbol{x_{i,t}}|\boldsymbol{bbox_j}, \boldsymbol{viewpoint_k}) \tag{4.2}$$

$$p(\boldsymbol{x_{i,t}}) = p(\boldsymbol{x_{i,t}}|\boldsymbol{viewpoint_k}) \cdot p(\boldsymbol{x_{i,t-1}}) \tag{4.3}$$

For $p(\boldsymbol{x_{i,t}}|\boldsymbol{bbox_j}, \boldsymbol{viewpoint_k})$ in (4.2), a kernel function was devised to localise objects within the state space. We experimented with two kernel functions: square and Gaussian. The square kernel, when applied to each bounding box, assigns a probability of one if the transformed point falls inside the bounding box, and zero otherwise, as shown in (4.4). This binary approach asserts that a point inside a bounding box likely indicates an object's presence; if not, then the object is absent.

$$p(\boldsymbol{x_{i,t}}|\boldsymbol{bbox_j}, \boldsymbol{viewpoint_k}) = \begin{cases} 1 & \text{if inside bounding box} \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

To temper the aggressive behaviour of the square function, we explored the bidimensional Gaussian function, as shown in equation (4.5). This function provides a smoother effect on the borders of the bounding box and affects some cells $\boldsymbol{x_{i,t}}$ outside the bounding boxes. Thus, a Gaussian function is more suitable for handling irregular objects and noise. In equation (4.5), (x_0, y_0) represents the centre of bounding box j in the sensor's frame, and (x, y) denotes the position of each cell $\boldsymbol{x_{i,t}}$ in the sensor's frame. The coordinates in the image's frame are projected using a projection model described in section 4.2.1.3. The standard deviation values (σ_x, σ_y) are set to half the size of the bounding box j. These values were determined experimentally and yielded reasonable results. Utilising the Gaussian kernel ((4.5)) for object position estimation involves equation (4.2), which is akin to a mixture of Gaussians, considering the camera detects multiple objects. This mixture of Gaussians creates a function that becomes smoother with more Gaussians in the mixture. To counteract this effect, a normalised version of the mixture of Gaussians, as detailed in equation (4.6), is employed to accentuate the detected objects. Furthermore, the updated state space $\dim(X_t)$ is also normalised at the conclusion of each histogram filter iteration.

Equations (4.5) and (4.6) are crucial for understanding the process. Equation (4.5) cal-

culates the probability of a cell $\boldsymbol{x_{i,t}}$ based on its relation to the bounding box $\boldsymbol{bbox_j}$ and the viewpoint $\boldsymbol{viewpoint_k}$. Equation (4.6) then normalises this probability across different viewpoints to ensure distinct objects are highlighted effectively.

$$p(\boldsymbol{x_{i,t}}|\boldsymbol{bbox_j},\boldsymbol{viewpoint_k}) = \exp\left(-\frac{(x-x_0)^2}{2\sigma_x^2} - \frac{(y-y_0)^2}{2\sigma_y^2}\right) \tag{4.5}$$

$$p(\boldsymbol{x_{i,t}}|\boldsymbol{viewpoint_k}) = \frac{p(\boldsymbol{x_{i,t}}|\boldsymbol{viewpoint_k})}{\max(p(\boldsymbol{x_{i,t}}|\boldsymbol{viewpoint_k}))} \tag{4.6}$$

Algorithm 1 outlines the steps for updating the weights of cells during the histogram filtering process. It involves iterating through each viewpoint, transforming cell coordinates from the mainframe to the camera's and then to the sensor's frame, and calculating the probability of each cell being within a bounding box. These probabilities are then normalised to adjust the overall probabilities matrix, ensuring an accurate representation of detected objects from various viewpoints.

Data: $decomposition\_grid, probabilities\_matrix, bounding\_boxes$
Result: $probabilities\_matrix$
for *each viewpoint* **do**
  for $\boldsymbol{x_{i,t}}, p(\boldsymbol{x_{i,t}})$ *in* $decomposition\_grid, probabilities\_matrix$ **do**
    $cell\_camera \leftarrow$ transforms cell from the mainframe to camera's frame;
    $cell\_sensor \leftarrow cell\_camera$ in the sensor's frame;
    $(u,v) \leftarrow cell\_sensor$ in the image's frame;
    $p(x_{i,t}|\text{viewpoint}_k) \leftarrow 0$;
    for $\boldsymbol{bbox}$ *in* $bounding\_boxes$ **do**
      $p(\boldsymbol{x_{i,t}}|\boldsymbol{viewpoint_k}) \leftarrow p(\boldsymbol{x_{i,t}}|\boldsymbol{viewpoint_k}) + \frac{1}{N} \times p(\boldsymbol{x_{i,t}}|\boldsymbol{bbox})$;
    end
  end
  $p(\boldsymbol{x_t}|\boldsymbol{viewpoint_k}) \leftarrow \text{normalise}(p(\boldsymbol{x_t}|\boldsymbol{viewpoint_k}))$;
  $p(\boldsymbol{x_t}) \leftarrow p(\boldsymbol{x_t}) \times p(\boldsymbol{x_t}|\boldsymbol{viewpoint_k})$;
  $p(x_t) \leftarrow \text{normalise}(p(\boldsymbol{x_t}))$;
end

Algorithm 1: Histogram filter – updating weights

4.2.1.3 Camera Projection Model

When applying the histogram filter, we divided the state space into a finite grid. To accurately estimate the object's 3D position, we manoeuvred the detection camera around the object and the segmented state space, enabling visualisation from multiple angles. Figure 4.4 illustrates a state space's decomposition. The camera detects an object, and then the MonoVisual3DFilter decomposes the state space around it. The points in the decomposed are converted to the camera's frame. If they fall outside the detected object, they are discarded, remaining the points converted to inside the object's detection.

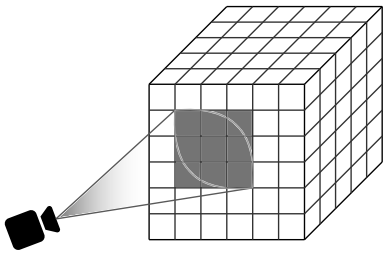


Figure 4.4: Decomposition of the state space and the intersection of the points in this space and the camera's detection.

Object detection with the detection camera necessitates a reliable projection model to predict the object's 3D location by converting between the 3D space coordinates and the image frame. We utilised the Pinhole model to transform the 3D space coordinates into the image frame.

Acknowledging the 3D space points in the camera's frame, we must translate them into the sensor's frame before projecting them onto the image frame. Although they share the same origin, their orientations differ. The sensor's frame orientation, as shown in Fig. 4.5, can be described using Euler angles, specifically Euler(YZX) $=(y,z,x)=(90°,-90°,0°)$, which equates to the quaternion $\boldsymbol{q}=(x,y,z,w)=(-0.5,0.5,-0.5,0.5)$.

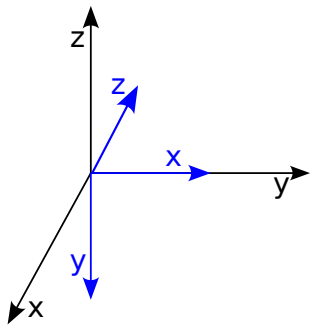


Figure 4.5: Conversion between the camera's and sensor's frames (blue – sensor's frame; black – camera's frame).

The transformation from the sensor's frame to the image's coordinates utilises the intrinsic parameters matrix, as defined in equation 4.7. This matrix accounts for the image's width (w) and height (h), along with the camera's focal length (f). The focal length, influenced by the camera's horizontal field of view (HFOV) and the image's width, can be determined by equation 4.8.

$$(u,v,1)=\begin{bmatrix} f & 0 & \frac{w}{2} \\ 0 & f & \frac{h}{2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix} \tag{4.7}$$

$$f=\frac{0.5\times w}{\tan\left(0.5\times HFOV\times\frac{\pi}{180}\right)} \tag{4.8}$$

However, this modelling approach primarily applies to ideal scenarios, such as simulations. For practical applications with the OAK-1 camera, an additional calibration step was required to estimate the intrinsic parameters accurately. For this purpose, we used the Kalibr software [279].

4.2.1.4 Objects positioning

After executing the histogram filter from multiple viewpoints, the state space $\text{dom}(X_t)$ will exhibit several clusters of points. Given the number of objects in the scene, as determined by the detection camera, we can employ the k-means algorithm to group these points and calculate the centre of each cluster.

The k-means algorithm aims to cluster the various points within the discrete state space by minimising the geometric distance between points and the cluster's centre, as defined in the equation (4.9). This equation focuses on minimising the Euclidean distance between points and $\boldsymbol{\mu_j}$, the centre point of each cluster in A.

$$\sum_{i=0}^{n} \min_{\boldsymbol{\mu_j} \in A} (||\boldsymbol{x_{i,t}} - \boldsymbol{\mu_j}||^2) \tag{4.9}$$

Following the k-means clustering, the state space should contain as many point clouds as objects detected by the detection camera. The computation of the centres of these clouds, to determine the positions of the detected objects, can be approached in two ways:

1. computation of the geometric centre of the cloud; or
2. computation of the weighted centre of the cloud.

The geometric centre of each point cloud is the Euclidean centre $\boldsymbol{\mu_j}$ minimised during the k-means algorithm for equation 4.9. In addition to the geometric centre, the k-means algorithm also returns the points, $\boldsymbol{x_{i,t}}$, that belong to each cloud, S_j. Given the state of each element of the state space $\text{dom}(X_t)$ at the conclusion of the histogram filter, each element $\boldsymbol{x_{i,t}}$ is assigned a weight, w_i. Thus, the weighted centre is the weighted average, as shown in equation (4.10), of the coordinates of $\boldsymbol{x_{i,t}}$ that belong to the set S_j.

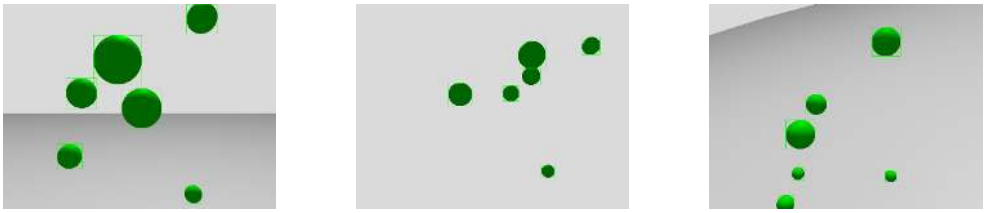
$$\hat{\boldsymbol{\mu}}_{\boldsymbol{j}} = \left[\frac{\sum_i^N w_i \cdot x_{i,t_1}}{N} \quad \frac{\sum_i^N w_i \cdot x_{i,t_2}}{N} \quad \frac{\sum_i^N w_i \cdot x_{i,t_3}}{N} \right]^T \qquad \forall \boldsymbol{x_{i,t}} \in S_j \tag{4.10}$$

4.2.1.5 Experiments

Three essays were performed in different environments to validate the effectiveness of the MonoVisual3DFilter.

In the first scenario, we utilised the Gazebo simulator to craft a scene populated with multiple spheres to estimate their positions (see Figure 4.1). We employed a bounding box camera devoid of noise, facilitating the validation of the filter's actual performance free from external

artefacts or noise. Additionally, we executed another essay where we introduced Gaussian noise. This noise randomly altered the position and size of the bounding box for detected objects, as well as the success rate of object detection. Since we did not equip the simulator with a manipulator, the bounding box camera had increased flexibility in setting its pose. Therefore, we adjusted the camera's pose to ensure the visibility of the spheres. Figure 4.6 showcases the camera's view at each pose. Initially, the camera is positioned straight towards the spheres (Figure 4.6a). Subsequently, the camera is moved down and left, tilting upwards (Figure 4.6b), and finally, it is moved up and right, angling downwards back to the initial pose (Figure 4.6c). This camera setup was maintained for both simulation experiments.



(a) (b) (c)

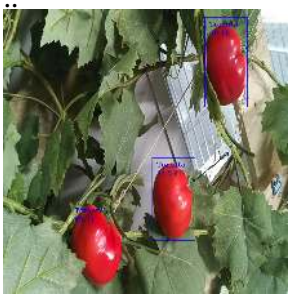
Figure 4.6: View of the spheres by the bounding box camera at each fixed viewpoint. The green square boxes around the spheres are the bounding boxes of the detected spheres by the bounding box camera.

In the third essay, we deployed a realistic testbed in a laboratory to test the algorithm under near-real-world conditions, as illustrated in Figure 4.2). The testbed was composed of artificial leaves and plastic tomatoes, simulating a realistic environment. The tomatoes were suspended among the leaves. To determine the baseline position of the tomatoes within the testbed, we relied on the manipulator's kinematics. For each tomato, the manipulator end-effector was moved to the fruit, and the end-effector's position was recorded. This position relative to the manipulator's base frame served as the baseline for the tomato's location. Like the simulation essays, the bounding box camera was adjusted to three fixed poses to ensure tomato visibility. Unlike the simulation essays, the testbed involved conducting several experiments, ranging from localising one to three tomatoes simultaneously, totalling ten tomatoes and sixty measurements. Figure 4.7 illustrates the visibility of the tomatoes through the OAK camera at each selected pose for the various experiments.

To better evaluate the histogram filter's performance in estimating object positions, we extracted several error metrics, namely the mean absolute error (MAE) (4.11), the mean square error (MSE) (4.12), the root mean square error (RMSE) or standard deviation (4.13), and the mean absolute percentage error (MAPE) (4.14). In these equations, μ_j represents the actual centre of the object for cluster S_j, and $\hat{\mu}_j$ is the estimated centre using the aforementioned methods for cluster j up to the maximum number of clusters M.

(a)

(b)



(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

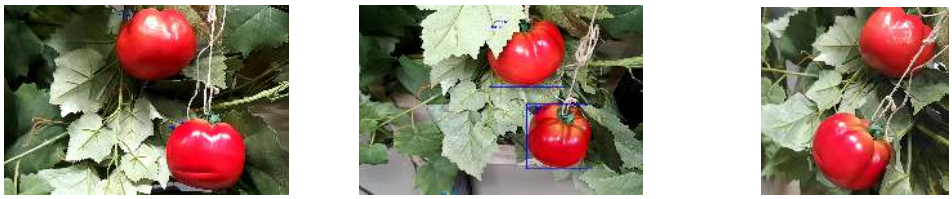
(k)

(l)

(m)

(n)

(o)



(p) (q) (r)

Figure 4.7: View of the tomatoes in the testbed at each pose of the OAK-1 camera. The blue squares around the tomatoes are the detected tomatoes by the bounding box camera OAK-1 using a custom-trained YOLO v8 tiny detector. Inside each bounding box are the detected class (tomato) and the detection confidence. Each row is an experiment, in a total of six experiments, and each figure contains the number of tomatoes being detected.

$$\text{MAE}\,(\boldsymbol{\mu_j},\hat{\boldsymbol{\mu}}_{\boldsymbol{j}})=\frac{1}{N\cdot M}\sum_{i}^{N}\sum_{j}^{M}|\boldsymbol{\mu_{ij}}-\hat{\boldsymbol{\mu}}_{\boldsymbol{ij}}| \qquad \forall\, j\in\mathbb{N}:\{1..M\} \tag{4.11}$$

$$\text{MSE}\,(\boldsymbol{\mu_j},\hat{\boldsymbol{\mu}}_{\boldsymbol{j}})=\frac{1}{N\cdot M}\sum_{i}^{N}\sum_{j}^{M}(\boldsymbol{\mu_{ij}}-\hat{\boldsymbol{\mu}}_{\boldsymbol{ij}})^2 \qquad \forall\, j\in\mathbb{N}:\{1..M\} \tag{4.12}$$

$$\text{RMSE}\,(\boldsymbol{\mu_j},\hat{\boldsymbol{\mu}}_{\boldsymbol{j}})=\sqrt{\frac{1}{N\cdot M}\sum_{i}^{N}\sum_{j}^{M}(\mu_{ij}-\hat{\mu}_{ij})^2} \qquad \forall\, j\in\mathbb{N}:\{1..M\} \tag{4.13}$$

$$\text{MAPE}\,(\boldsymbol{\mu_j},\hat{\boldsymbol{\mu}}_{\boldsymbol{j}})=\frac{1}{N\cdot M}\sum_{i}^{N}\sum_{j}^{M}\left|\frac{\boldsymbol{\mu_{ij}}-\hat{\boldsymbol{\mu}}_{\boldsymbol{ij}}}{\boldsymbol{\mu_{ij}}}\right|\times 100 \qquad \forall\, j\in\mathbb{N}:\{1..M\} \tag{4.14}$$

4.2.2 Best viewpoint estimator (BVE)

The position estimator MonoVisual3DFilter has a limitation in selecting the next best viewpoint to perceive the fruits from a different perspective. This limitation affects the optimal estimation of the fruits' positions. Probabilities theory proves that two orthogonal random variables are uncorrelated [280, ch. 6]. Therefore, observing the fruits from orthogonal positions to previous ones delivers a completely new perspective on the object and aids in accurately estimating the position. To address this limitation, we intend to design an algorithm called the best viewpoint estimator (BVE) that selects the next best pose and increases the observability of the fruit.

To achieve this, we can set it as a statistical optimisation problem. We start with an initial Gaussian distribution resulting from observing the fruit in the first position. Next, we select the next viewpoint whose observation of normal distribution intersects with the current one and minimises the resulting covariance of the Gaussian distribution. We use the product of Gaussians (4.15) to establish a minimisation objective. In equation (4.15), $N_i(\boldsymbol{\mu_i},\boldsymbol{\Sigma_i})$ is a Gaussian distribution with the index $i\in\mathbb{N}$ of each observation viewpoint to the fruit.

$$N(\boldsymbol{\mu_p},\boldsymbol{\Sigma_p})=N_1(\boldsymbol{\mu},\boldsymbol{\Sigma_1})\cdot\ldots\cdot N_n(\boldsymbol{\mu},\boldsymbol{\Sigma_n}) \tag{4.15}$$

According to Petersen and Pedersen [281], we can divide the product of Gaussian distributions (4.15) into two separate equations for the mean values and the covariance, as shown in equations (4.16) and (4.17), respectively.

$$\boldsymbol{\mu_p} = \boldsymbol{\Sigma_p} \cdot (\boldsymbol{\Sigma_1}^{-1} \cdot \boldsymbol{\mu_1} + \ldots + \boldsymbol{\Sigma_n}^{-1} \cdot \boldsymbol{\mu_n}) \tag{4.16}$$

$$\boldsymbol{\Sigma_p} = (\boldsymbol{\Sigma_1}^{-1} + \ldots + \boldsymbol{\Sigma_n}^{-1})^{-1} \tag{4.17}$$

The fruit remains stopped in all positions, hanged on the tree. Therefore, the position of the tomato, $\boldsymbol{k}$, is always constant, $\boldsymbol{\mu_i} = \boldsymbol{k}$. Based on this consideration, we can define to optimise a function related to equation (4.17).

The camera's covariance, $\boldsymbol{\Sigma_c}$, is mostly a characteristic of the camera and is always constant in the camera frame C. Thus, we can consider it as the camera's observability noise in its frame, as shown in equation (4.18), where $\Sigma_{ij} : i, j \in \{x, y, z\}$ are the elements of the matrix and the variance or covariance for each axis.

$$\boldsymbol{\Sigma_c} = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} & \Sigma_{xz} \\ \Sigma_{xy} & \Sigma_{yy} & \Sigma_{yz} \\ \Sigma_{xz} & \Sigma_{yz} & \Sigma_{zz} \end{bmatrix} \tag{4.18}$$

To relate the different observability noises between different poses, we need to state the camera's covariance matrix in the main's frame W through equation (4.19). In this equation, the matrix $\boldsymbol{R}_C^W$ is a rotation matrix that relates the camera frame C to the main frame W.

$$\boldsymbol{\Sigma_n} = \boldsymbol{R}_C^W \, \boldsymbol{\Sigma_c} \, \boldsymbol{R}_C^{W\intercal} \tag{4.19}$$

The covariance matrix changes at each iteration of the algorithm, resulting from the computation of equations (4.17) and (4.19). To generalise the system's initial conditions, we consider a generic covariance matrix, as shown in equation (4.20), as the result of the previous states of the system. This generic covariance matrix is the result of all the covariance matrices until $k-1$.

$$\boldsymbol{\Sigma_o} = \begin{bmatrix} \Sigma_{o,xx} & \Sigma_{o,xy} & \Sigma_{o,xz} \\ \Sigma_{o,xy} & \Sigma_{o,yy} & \Sigma_{o,yz} \\ \Sigma_{o,xz} & \Sigma_{o,yz} & \Sigma_{o,zz} \end{bmatrix} \tag{4.20}$$

Rotation matrix As previously considered, the known covariance matrix, $\boldsymbol{\Sigma_c}$, represents the observability noise for the fruit in the camera frame. This frame is not static; it moves with the camera's movement relative to the fixed main frame. Therefore, it is necessary to establish a rotation matrix that articulates the relationship between the camera frame and the main frame. It is important to note that the camera's translation is disregarded because it does not affect the covariance.

The figure 4.8 illustrates a potential relationship between the frames. $O\,x_C\,y_C\,z_C$ is the camera's frame, centred on the sensor, with the axis $\boldsymbol{x_C}$ indicating the direction the camera faces. For simplification, we assume that $\boldsymbol{y_C}$ is always parallel to the plane defined by ${}_{x_W}O_{y_W}$. This simplification is feasible because the covariance matrix is ideally symmetrical along the axis $\boldsymbol{x_C}$, and the orientation of the other axes is not critical.

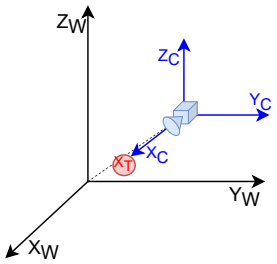


Figure 4.8: Definition of the camera's and the main's frame.

Considering the estimated position of the tomato in the main frame, $\hat{\boldsymbol{k}}$, the unit vector of the axis $\boldsymbol{x_C}$, denoted as $\boldsymbol{e_{x_C}} = \boldsymbol{e_{x_C^W}}$, is calculated by (4.21), where $\boldsymbol{c}$ represents the position of the camera in the main frame. With $\boldsymbol{e_{x_C^W}}$ known, we can define each axis of the camera frame in the main frame ($\boldsymbol{x_C^W}$, $\boldsymbol{y_C^W}$, and $\boldsymbol{z_C^W}$) through (4.21), (4.22), and (4.23). In the equation (4.23), $e_{x_{c,i}^W}$ is the i^{th} element of the vector $\boldsymbol{e_{x_c^W}}$. Each vector must be normalised for the rotation matrix definition to yield the unit vectors. The rotation matrix $\boldsymbol{R_C^W}$, which relates the camera frame to the main frame, is given by (4.24). In the equation (4.24), $\boldsymbol{e_{x_C^W}}$, $\boldsymbol{e_{y_C^W}}$, and $\boldsymbol{e_{z_C^W}}$ are the unit vectors of $\boldsymbol{x_C^W}$, $\boldsymbol{y_C^W}$, and $\boldsymbol{z_C^W}$, and $\boldsymbol{x_W}$, $\boldsymbol{y_W}$, and $\boldsymbol{z_W}$, respectively.

$$\boldsymbol{e_{x_c^W}} = \frac{\hat{\boldsymbol{k}} - \boldsymbol{c}}{||\hat{\boldsymbol{k}} - \boldsymbol{c}||} \tag{4.21}$$

$$\boldsymbol{y_C^W} = \begin{bmatrix} -e_{x_{c,2}^W} & e_{x_{c,1}^W} & 0 \end{bmatrix}^{\mathsf{T}} \tag{4.22}$$

$$\boldsymbol{z_C^W} = \boldsymbol{x_C^W} \times \boldsymbol{y_C^W} = \begin{bmatrix} -e_{x_{c,1}^W} \cdot e_{x_{c,3}^W} \\ -e_{x_{c,2}^W} \cdot e_{x_{c,3}^W} \\ (e_{x_{c,1}^W})^2 + (e_{x_{c,2}^W})^2 \end{bmatrix} \tag{4.23}$$

$$\boldsymbol{R_C^W} = \begin{bmatrix} \boldsymbol{e_{x_C^W}} & \boldsymbol{e_{y_C^W}} & \boldsymbol{e_{z_C^W}} \end{bmatrix} \tag{4.24}$$

Objective function We have designed a loss function to minimise the intersection of the covariance matrices in the main frame. This loss function is based on the application of the Gaussian intersection. The objective of the loss function is to compute a scalar value, which can be achieved through two types of loss function: the first one is a dependency of dispersion, and the other is the maximum absolute covariance.

For each observation pose, the intersection between two covariance matrices is given by the formula in equation (4.25). To compute this equation, we need to compute three inverse matrices, which is computationally demanding. To simplify this operation, we opted to use the precision matrix (4.26), which is the inverse of the covariance matrix. The precision can be simplified by a scalar characterising the whole matrix, the concentration, which is the determinant of the precision (4.27). Through the properties of the determinant of matrices, we know that the inverse of the determinant of the covariance matrix is equal to the determinant of the inverse of the covariance matrix, $\det(\boldsymbol{A}^{-1}) = \det(\boldsymbol{A})^{-1}$. Hence, we can define the objective function as the dispersion, i.e., the inverse of concentration. Because the proposed objective function has a very low magnitude, we scaled it to the logarithm scale (4.28). In equation (4.28), $\boldsymbol{P}$ and $\boldsymbol{\Sigma_n}$ are dependent on $\hat{\boldsymbol{c}}$, the next estimated position for the camera, which we aim to optimise. Through the proposed strategy, we can reduce the computation of the most complex inverse matrix, instead computing the inverse of a scalar.

$$\boldsymbol{\Sigma_u} = (\boldsymbol{\Sigma_o}^{-1} + \boldsymbol{\Sigma_n}^{-1})^{-1} \tag{4.25}$$

$$\boldsymbol{P} = \boldsymbol{\Sigma_o}^{-1} + \boldsymbol{\Sigma_n}^{-1} \tag{4.26}$$

$$c = \det(\boldsymbol{P}) \tag{4.27}$$

$$f(\hat{\boldsymbol{c}}) = \ln\left|\frac{1}{\det \boldsymbol{P}}\right| = \ln\left|\frac{1}{\det(\boldsymbol{\Sigma_n}^{-1} + \boldsymbol{\Sigma_o}^{-1})}\right| = -\ln(\det(\boldsymbol{\Sigma_n}^{-1} + \boldsymbol{\Sigma_o}^{-1})) \tag{4.28}$$

Alternatively, we can minimise the absolute maximum eigen value of the covariance matrix if we have enough computing power to compute (4.25). While using this loss function, we should remember that it is highly non-linear and whose derivative function varies at each step because of the maximum function.

$$f(\hat{\boldsymbol{c}}) = \max(|\text{eig}(\boldsymbol{\Sigma_u})|) \tag{4.29}$$

We can use optimisation algorithms that can operate with non-linear restrictions and loss functions to optimise both functions. For the current analysis, we opted to use an interior-point algorithm [282] that has already been implemented in Matlab[4].

In the following sections, we will observe that both loss functions can effectively estimate the best observations for the objects. However, the loss function that minimises the absolute maximum covariance (4.29) tends to deliver slightly better results and is faster to compute.

We also intend to effectively drive the camera to the objects to perform tasks while estimating the fruit's position. For that, we added an additional component to the loss function, as illustrated in equation (4.31). The $\text{act}(i,a,b)$ is an activation function that, in this case, is the sigmoid function (Fig. 4.30). This function activates the additional component, forcing the camera to approximate the object. In this function, a and b are control hyper-parameters

[4]See The MathWorks, Inc. "MATLAB 9.14.0.2206163 (R2023a)." (2024), [Online]. Available: https://www.mathworks.com (visited on 01/17/2024).

that set the aggressiveness of the activation function and the set point for this function (i.e., the value of the function for 0.5), respectively. The figure 4.30 illustrated the effect of the sigmoid activation function with varied a and a fixed $b=20$ – higher a values make the penalisation of the distance from the camera to the fruit faster. i is the procedure's iteration number. α, β are scale factors to adjust the intensity of the loss and activation functions, respectively. Through this strategy, we can activate gradually the Euclidean error to the fruit according to the evolution of the estimation procedure.

$$\text{act}(i,a,b)=\frac{1}{1+e^{-a\cdot(i-b)}} \tag{4.30}$$

$$F(\hat{\boldsymbol{c}})=\alpha\times f(\hat{\boldsymbol{c}})+\beta\cdot\text{act}(i,a,b)\cdot||\hat{\boldsymbol{k}}-\hat{\boldsymbol{c}}|| \tag{4.31}$$

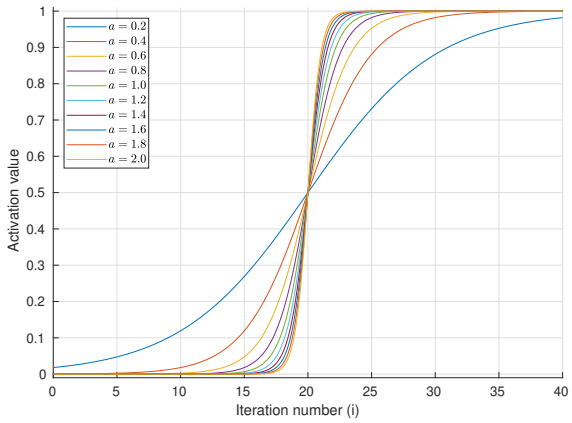


Figure 4.9: Effect of the sigmoid activation function with $b=20$ and $a=[0.2;2]$ in steps of 0.2

Restrictions The designed algorithm effectively estimates optimal camera positions to maximise the visibility of the target fruit. However, it is essential to incorporate practical limitations to reflect the real-world scenario and the constraints of the environment. Specifically, the camera's placement must consider the physical space and the robotic arm's manoeuvrability.

Although theoretically, the camera could be positioned anywhere within the task space, in practice, the robot manipulator's working space limits its location. To address this, we have decided that acceptable camera positions must fall within the robot's operational area. For simplicity, we have modelled this area as a sphere, as illustrated in the figure 4.10a, centred at $\boldsymbol{m}$ with a radius of r_m meters, thus the camera's position $\hat{\boldsymbol{c}}$ must satisfy the condition in equation (4.32).

$$((\hat{\boldsymbol{c}}-\boldsymbol{m})\cdot(\hat{\boldsymbol{c}}-\boldsymbol{m})^{\intercal})-r_m^2\leq 0 \tag{4.32}$$

Additionally, the camera must not intrude into the space the fruit occupies. Given that

we are estimating the fruit's central position but not its exact volume, we will use an average fruit radius r_k, centred at $\hat{\boldsymbol{k}}$ (Fig. 4.10b). Therefore, the camera's position must also adhere to equation (4.33).

$$-((\hat{\boldsymbol{c}}-\hat{\boldsymbol{k}})\cdot(\hat{\boldsymbol{c}}-\hat{\boldsymbol{k}})^{\intercal})+r_k^2 \leq 0 \tag{4.33}$$

Optimising the camera's position also requires considering its orientation; it must be aimed towards the fruit. Considering the camera's conical field of view, we introduce a further constraint ensuring the fruit is within this field. This is detailed in figure 4.10c and equation (4.37), where $HFOV$ represents the camera's horizontal field of view. In this restriction, we aim to constrain the angle formed by the vectors $\boldsymbol{e_c}$ and $\boldsymbol{x_{\text{lim}}}-\hat{\boldsymbol{c}}$ to be inferior to $HFOV/2$.

$$\boldsymbol{e_c} = \frac{\hat{\boldsymbol{k}}-\hat{\boldsymbol{c}}}{||\hat{\boldsymbol{k}}-\hat{\boldsymbol{c}}||} \tag{4.34}$$

$$\boldsymbol{e_{c_\perp}} = \begin{bmatrix} -e_{c,2} & e_{c,1} & e_{c,3} \end{bmatrix}^{\intercal} \tag{4.35}$$

$$\boldsymbol{x_{\text{lim}}} = \hat{\boldsymbol{k}} + r_k \cdot \boldsymbol{e_{c_\perp}} \tag{4.36}$$

$$0 \geq \frac{\boldsymbol{x_{\text{lim}}}-\hat{\boldsymbol{c}}}{||\boldsymbol{x_{\text{lim}}}-\hat{\boldsymbol{c}}||} \cdot \boldsymbol{e_c} - \cos\left(\frac{HFOV}{2}\right) \tag{4.37}$$

In a tomato greenhouse, where plants are aligned in rows, the robot must avoid crossing these rows to prevent damage. This is managed by defining a restriction in equation (4.40), modelling the plant rows as a planar boundary to keep the robot on one side, set at a distance d meters from the fruit. The plane's orientation is determined by the normal vector $\boldsymbol{e_{n_{\text{plane}}}}$, that represents the normalised vector of $\boldsymbol{n_{\text{plane}}}$. Because we only have a point in the plant tree, we approximate the trees' plane (Fig. 4.10d) by its normal vector, between the fruit and the origin, placed in the manipulator's base. With this approximation, we consider that the robot is parallel to the fruits' plants and aligned with the fruit. This is a rough approximation that serves its purpose on a small scale, such as the one we are working on.

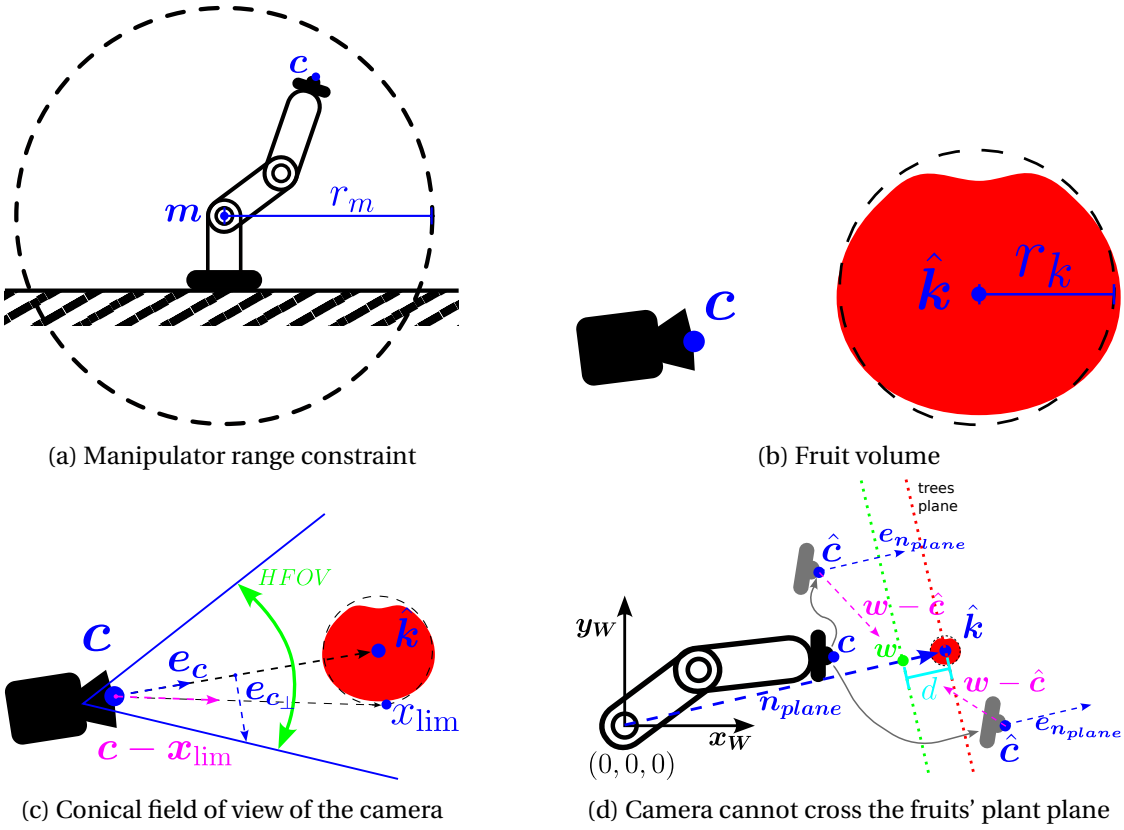
$$\boldsymbol{n_{\text{plane}}} = \begin{bmatrix} \hat{k}_0 & \hat{k}_1 & 0 \end{bmatrix}^{\intercal} \tag{4.38}$$

$$\boldsymbol{w} = \hat{\boldsymbol{k}} - d \cdot \boldsymbol{e_{n_{\text{plane}}}} \tag{4.39}$$

$$0 \geq \boldsymbol{e_{n_{\text{plane}}}} \cdot (\hat{\boldsymbol{c}} - \boldsymbol{w}) \tag{4.40}$$

Further constraints were introduced to accommodate the specific features of the manipulator used, ensuring the chosen positions are feasible and the kinematics can be computed.

In addition to these constraints, further experiments were conducted using simplified forms of constraints, focusing solely on the distance between the camera and the fruit. This was defined by a range $l \in [l_{min}, l_{max}]$, as specified in equations (4.41) and (4.42).



(a) Manipulator range constraint

(b) Fruit volume

(c) Conical field of view of the camera

(d) Camera cannot cross the fruits' plant plane

Figure 4.10: Graphical explanation of restrictions

$$||\hat{\boldsymbol{c}} - \hat{\boldsymbol{k}}|| - l_{min} < 0 \tag{4.41}$$

$$-||\hat{\boldsymbol{c}} - \hat{\boldsymbol{k}}|| - l_{max} < 0 \tag{4.42}$$

4.2.2.1 Fruit pose estimation using EKF

In our current approach, we approximate the initial location of fruit using a basic estimation method that relies on their average dimensions. However, this method only provides a rough estimation of the fruit that we may not rely on. Therefore, we propose a more robust alternative that leverages an EKF to refine this estimation, working in conjunction with the BVE for enhanced accuracy. We initialised the EKF with a rough estimation of the fruit's position, and then used the EKF to refine and improve the estimation of the fruit's position. The EKF method offers a progressive improvement in pinpointing the fruit's coordinates within a 3D task environment.

For optimal performance of the EKF, it is crucial that the camera manoeuvres smoothly to maintain a constant observation angle on the fruit. This continuous adjustment is essential for iteratively refining the fruit's positional estimate. To facilitate this, we enforce a constraint in BVE, ensuring the camera's movement towards the most advantageous subsequent view-

point is within a specified radius, r_d metres, as formalised in the equation (4.43).

$$||\hat{\boldsymbol{c}}_{\boldsymbol{k+1}} - \boldsymbol{c}_{\boldsymbol{k}}|| - r_d < 0 \tag{4.43}$$

The EKF is structured around two principal phases, as illustrated in Figure 4.11: prediction and correction. The prediction phase anticipates the fruit's location by considering both its dynamics and expected movements. Subsequently, in the correction phase, the fruit's position is adjusted based on real-time observations made by a dedicated sensor, thereby aligning the estimated coordinates with actual measurements.

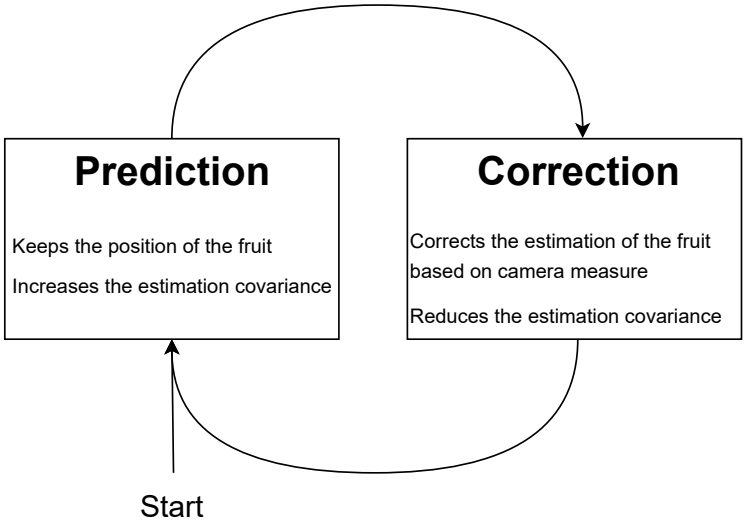


Figure 4.11: Diagram of the EKF applied.

Prediction During this phase, we will be estimating the position of the fruit based on its dynamics. However, since the fruit is not expected to have any dynamics, it should maintain its pose. Therefore, the predicted position of the fruit will be the same as its previous position, as stated in equation (4.44). Additionally, we must propagate the covariance estimation error, as outlined in equation (4.45). In this equation, $\boldsymbol{Q}_{\boldsymbol{k}}$ is the covariance of the prediction noise associated to the estimation variable.

$$\hat{\boldsymbol{x}}_{\boldsymbol{k}|\boldsymbol{k-1}} = f(\hat{\boldsymbol{x}}_{\boldsymbol{k-1}|\boldsymbol{k-1}}, \boldsymbol{u}_{\boldsymbol{k-1}}) = \boldsymbol{I} \cdot \hat{\boldsymbol{x}}_{\boldsymbol{k-1}} \tag{4.44}$$

$$\boldsymbol{P}_{\boldsymbol{k}|\boldsymbol{k-1}} = \boldsymbol{F}_{\boldsymbol{k}} \cdot \boldsymbol{P}_{\boldsymbol{k}|\boldsymbol{k-1}} \cdot \boldsymbol{F}_{\boldsymbol{k}}^{\mathsf{T}} + \boldsymbol{Q}_{\boldsymbol{k}} = \boldsymbol{P}_{\boldsymbol{k}|\boldsymbol{k-1}} + \boldsymbol{Q}_{\boldsymbol{k}} \tag{4.45}$$

$$\boldsymbol{F}_{\boldsymbol{k}} = \frac{\partial f}{\partial \boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}_{\boldsymbol{k-1}|\boldsymbol{k-1}}, \boldsymbol{u}_{\boldsymbol{k}}} = 1 \tag{4.46}$$

Correction In this scenario, it is assumed that the camera is observing the fruit. After a prediction phase, a correction phase follows where the camera pose is adjusted (4.54), based on

measures obtained from the camera sensor (4.47). This correction also corrects the covariance propagation error (4.52). The estimated position of the fruit for each instance is represented by $\hat{\boldsymbol{x}}$, and a random noise variable ε is added to simulate environmental noise (in real-world conditions, this value is realistically measured). $\boldsymbol{R_k}$ is the covariance associated to the observation noise.

$$h(\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}) = ||\hat{\boldsymbol{x}}_{\boldsymbol{k-1}} - \boldsymbol{c}|| \tag{4.47}$$

$$\boldsymbol{z_k} = ||\boldsymbol{k} - \boldsymbol{c}|| + \varepsilon \cdot \sqrt{\sigma_{xx}} \tag{4.48}$$

$$\boldsymbol{H_k} = \nabla h(\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}) = \frac{\hat{\boldsymbol{x}}_{\boldsymbol{k-1}} - \boldsymbol{c}}{||\hat{\boldsymbol{x}}_{\boldsymbol{k-1}} - \boldsymbol{c}||} \tag{4.49}$$

$$\boldsymbol{K_k} = \boldsymbol{P_{k|k-1}} \cdot \boldsymbol{H_k}^\intercal \cdot (\boldsymbol{H_k} \cdot \boldsymbol{P_{k|k-1}} \cdot \boldsymbol{H_k}^\intercal + R_k)^{-1} \tag{4.50}$$

$$R_k = \sigma_{xx} \tag{4.51}$$

$$\boldsymbol{P_{k|k}} = (\boldsymbol{I} - \boldsymbol{K_k} \cdot \boldsymbol{H_k}) \cdot \boldsymbol{P_{k|k-1}} \tag{4.52}$$

$$\tilde{\boldsymbol{y}}_{\boldsymbol{k}} = \boldsymbol{z_k} - h(\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}) \tag{4.53}$$

$$\hat{\boldsymbol{x}}_{\boldsymbol{k|k}} = \hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}} + \boldsymbol{K_k} \cdot \tilde{\boldsymbol{y}}_{\boldsymbol{k}} \tag{4.54}$$

$$\tag{4.55}$$

4.2.2.2 Experiments

In our study, we conducted several simulations in Matlab to validate our designed algorithms. These simulations were structured around the goal of optimising camera positioning to minimise the observation covariance, denoted as $\boldsymbol{\Sigma}$.

Initially, we explored scenarios where the camera positions that minimise $\boldsymbol{\Sigma}$ are orthogonal, considering only the BVE without any constraints.

Subsequently, we expanded our simulations to include simplified constraints, specifically focusing on maintaining the camera within a defined distance from the target object (the fruit). These constraints are mathematically represented as (4.42) and (4.41). For example, we stipulated that the camera must be positioned within (1.0 ± 0.1) m of the fruit. Regarding the hyper-parameters of the other constraints, $d = 0.1$ m; the working space was centred in the first joint of the Robotis Manipulator-H, $\boldsymbol{m} = \begin{bmatrix} 0 & 0 & 0.159 \end{bmatrix}^\intercal$ m and radius of $r_m = 0.645$ m.

The amount of restrictions was sequentially increased in our model until those that ensured the robot's kinematics were valid. We specified geometric constraints related to the camera and manipulator's operational space, and accounted for realistic observation covariances through a diagonal covariance matrix with increased covariance along the x-axis.

To further refine our observations, we integrated an EKF with the BVE, considering a maximum moving step of 0.02 m, enabling continuous adjustment of the fruit's estimated position in the 3D space. This integration was tested under various conditions, including different loss functions designed to optimise the camera's positioning.

To systematise the different experiments, we define the following protocol:

E2.1 For this experiment, we used the dispersion-based loss function (4.28) to the BVE. To restrict the BVE's behaviour, we limited the position of the camera between (1.0 ± 0.1) m to the fruit, (4.41) and (4.42);

E2.2 In this essay, we repeated the previous essays, but we also considered the restriction (4.32) that assures that the camera is inside the manipulator's working space. So, besides the (4.28) loss function, we consider the restrictions (4.41), (4.42), and (4.32);

E2.3 In this experiment, we consider the dispersion-based loss function (4.28) and the restrictions (4.32), (4.33), and (4.37). Restriction (4.33) ensures that the camera's position is outside of the area used by the tomato, and the restriction (4.37) ensures that the camera is always looking to the tomato and it fits in its visible area;

E2.4 This experiment considers the restrictions and the loss function of E2.3 and adds the restriction (4.40), which ensures that the camera sensor never crosses the wall defined by the different aligned plants;

E2.5 This experiment repeats the previous experiment, adding the kinematics constraints, ensuring that the camera's pose is always a valid pose for the manipulator;

E2.6 Repeats the experiment E2.1, considering the loss function (4.29), based on the minimisation of the maximum covariance, instead of the dispersion-based loss function (4.28);

E2.7 Repeats the experiment E2.2, considering the loss function (4.29);

E2.8 Repeats the experiment E2.3, considering the loss function (4.29);

E2.9 Repeats the experiment E2.4, considering the loss function (4.29); and

E2.10 Repeats the experiment E2.5, considering the loss function (4.29).

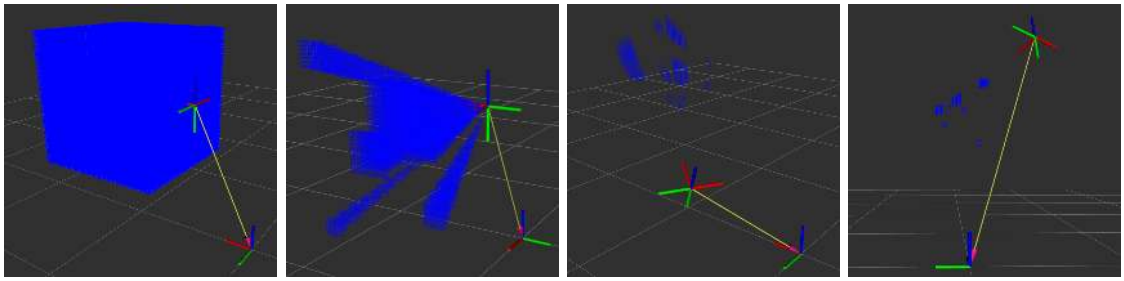
To evaluate our algorithm's performance, we compared it against the MonoVisual3DFilter using metrics such as MAPE, MAE, RMSE, and MSE.

4.3 Results

In the preceding discussion, we set three distinct experiments to evaluate the MonoVisual3DFilter. Two of these experiments were conducted in a virtual environment, while the third occurred within a controlled laboratory setting designed to simulate real-world conditions. The evaluation of the MonoVisual3DFilter's performance leveraged various error metrics, including (4.11), (4.12), (4.13), and (4.14). For each trial, the camera and manipulator were positioned strategically to ensure constant visibility of the fruits, and we systematically explored all positional permutations. This approach yielded six distinct estimations for each

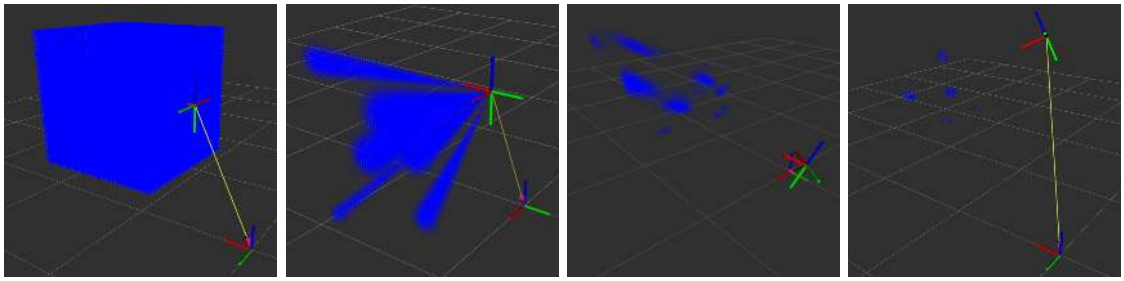
tomato within every experiment, with the error calculated based on the deviation from the actual position of each tomato.

The first experiment utilised a simulation environment to ascertain the 3D locations of green spheres, as depicted in Figure 4.1, without introducing any noise factors. Throughout this process, the bounding box camera adopted various positions to capture and integrate these locations. Each camera position involved adjusting the state space to either consolidate or refine the presence of objects, guided by the chosen kernel. We compared two kernel types: a square kernel (Figure 4.12) and a Gaussian kernel (Figure 4.13). Although visually similar in performance, the Gaussian kernel offers two adjustable hyper-parameters that provide greater control over the kernel's dimensions and the precision of object positioning, thereby enabling more nuanced filtering. For this specific experiment, we deployed a 2D Gaussian Filter characterised by $\mathcal{N}(0, \text{size}/2)$, where 'size' denotes the object's width or height. The results, illustrated in Figure 4.14, highlight the comparative error rates using square versus Gaussian kernels, alongside the geometric centres determined through the k-means algorithm and the weighted centres derived from the histogram filter's output. The findings suggest that employing a Gaussian kernel with weighted centre estimation yields superior accuracy.



(a) (b) (c) (d)

Figure 4.12: Progression of the Histogram filter in simulation detecting six spheres using a square kernel. (a) Initial state space decomposition; (b)-(d) Detection outcomes from successive viewpoints.



(a) (b) (c) (d)

Figure 4.13: Histogram filter iterations in simulation for six sphere detection using a Gaussian kernel, $\mathcal{N}(0, 0.2)$. (a) Initial state space decomposition; (b)-(d) Detection results from consecutive viewpoints.

In our experiments, we introduced variability into the simulation by altering the bound-

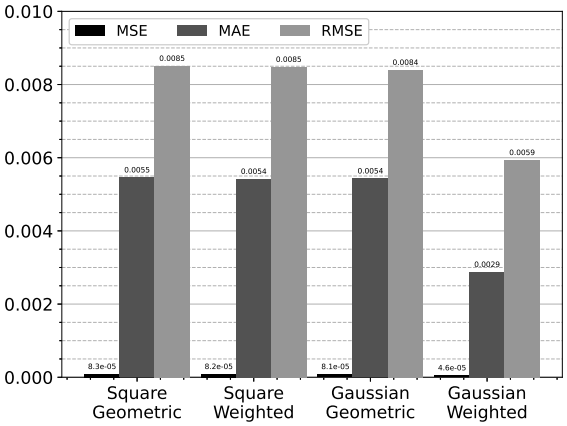


Figure 4.14: Simulation-based sphere position estimation error without noise.

ing box's centre and dimensions, as well as incorporating random deletions. The outcomes of these manipulations are illustrated in Figure 4.15. To modify the bounding box's centre and size, we applied a Gaussian distribution noise with $\mathcal{N}(0, 0.05)$. Additionally, we applied a failure detection rate of 2 % for the bounding box. The Gaussian kernel, particularly when using a weighted estimation for the spheres' centre, exhibited superior performance.

Moreover, this kernel proved more robust and accurate than the square kernel. While the histogram filter's behaviour in each pose mirrored that of our prior study, an increase in noise was noted. To mitigate this, we adjusted the Gaussian kernel to $\mathcal{N}(0, \text{size}/3)$, ensuring a more stable filter. This adjustment was the minimum required to prevent any loss of fruits and avoid overly sparse point clouds that could intersect with those of other fruits.

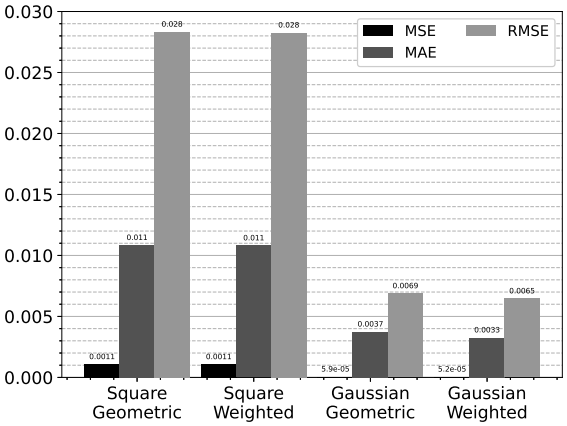


Figure 4.15: Error in estimating the position of the spheres in simulation with added noise.

The series of experiments demonstrated a notable success, with an MAE under 1 cm. Consequently, we proceeded to test the algorithm in a laboratory setting. Figure 4.16 showcases the box plot error of the MonoVisual3DFilter in this environment, utilising the Robotis Manipulator-H and the OAK-1 camera. An error analysis is detailed in the table 4.1, incorporating the average Euclidean error distance and the MAPE for a comprehensive evaluation of

the MonoVisual3DFilter's feasibility. Throughout the camera's various positions, the visibility of all tomatoes was maintained (Figure 4.7). The laboratory test comprised six experiments designed to simultaneously estimate the positions of one to three tomatoes (as depicted in Figure 4.7), totalling sixty position estimates for the tomatoes. During these experiments, Gaussian kernels underperformed compared to square kernels, with no significant difference observed between the results of weighted and geometric centre estimations despite the former exhibiting a lower error for the same standard deviation. That could be due to real noise that seriously affects the behaviour of the Gaussian kernel. Further essays should study softer Gaussian kernels, making the MonoVisual3DFilter more robust to noise.

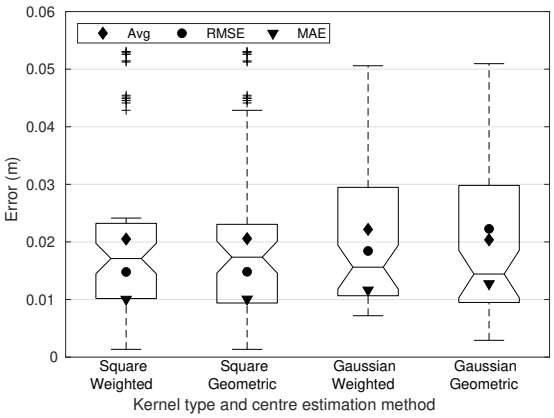


Figure 4.16: Error in estimating the position of tomatoes in the laboratory testbed.

Table 4.1: Error analysis for the laboratory testbed experiments, comparing different kernels and centre estimation methods.

	MAPE (%)	**MAE** (m)	**RMSE** (m)	**MSE** (m^2)	**Euclidean Error (Avg)** (m)
Square kernel and Weighted centre	63.52	10.0×10^{-3}	14.8×10^{-3}	219×10^{-6}	20.5×10^{-3}
Square kernel and Geometric centre	63.51	10.1×10^{-3}	14.8×10^{-3}	220×10^{-6}	20.6×10^{-3}
Gaussian kernel and Weighted centre	57.35	11.6×10^{-3}	18.4×10^{-3}	340×10^{-6}	22.2×10^{-3}
Gaussian kernel and Weighted centre	74.15	12.7×10^{-3}	22.3×10^{-3}	496×10^{-6}	20.4×10^{-3}

The BVE, when combined with the EKF, presents a strategy that either complements or competes with the MonoVisual3DFilter. This approach can estimate the position of fruits in the 3D task space using a monocular camera based on the perception of the fruit. We conducted ten experiments, as detailed in section 4.2.2.2, employing various optimisation functions and constraints. For each experiment, one hundred simulations were performed, and

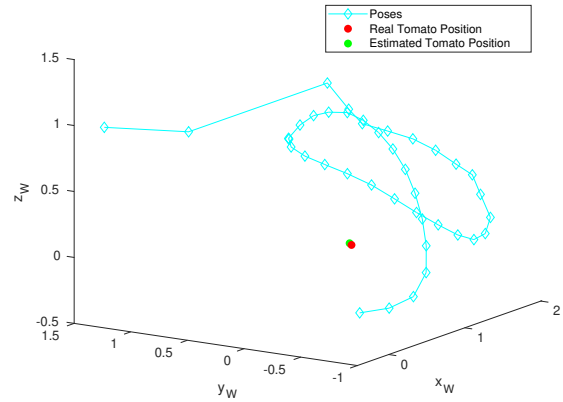
the average errors are reported in the table 4.2. Additionally, samples of the motion generated in each experiment are depicted through the plots in Figure 4.17.

Table 4.2: Error computations for the centre estimation using the BVE and the EKF

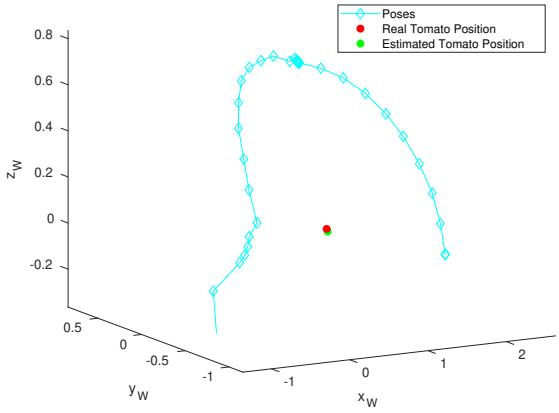
	MAPE (%)	**MAE** (m)	**RMSE** (m)	**MSE** (m^2)
E2.1	5.124	37.1×10^{-3}	15.5×10^{-3}	242×10^{-6}
E2.2	12.86	52.5×10^{-3}	25.5×10^{-3}	650×10^{-6}
E2.3	8.639	53.8×10^{-3}	28.1×10^{-3}	791×10^{-6}
E2.4	11.65	57.2×10^{-3}	29.1×10^{-3}	848×10^{-6}
E2.5	10.62	60.9×10^{-3}	31.2×10^{-3}	971×10^{-6}
E2.6	32.38	53.5×10^{-3}	26.3×10^{-3}	690×10^{-6}
E2.7	14.79	53.1×10^{-3}	24.7×10^{-3}	612×10^{-6}
E2.8	12.62	48.0×10^{-3}	21.2×10^{-3}	448×10^{-6}
E2.9	10.44	53.0×10^{-3}	23.4×10^{-3}	548×10^{-6}
E2.10	20.06	62.3×10^{-3}	31.3×10^{-3}	983×10^{-6}

Overall, our analysis suggests that using a differentiable loss function (as seen in experiments E2.1 to E2.5), such as the dispersion (4.28), offers advantages over a non-differentiable loss function (experiments E2.6 to E2.10), like (4.29), which relies on a maximum operation. Empirical analysis comparing the performance of both loss functions under identical conditions indicates that the dispersion loss function is not only more efficient in terms of computation – owing to one fewer matrix inversion – but also results in lower error rates in fruit position estimation. This is because the camera has greater freedom to navigate around the region of interest. In both strategies, the BVE tends to plan an approximated circular path, from scenarios where the algorithm freely designs its path to those with more restrictions, as illustrated in the Figures 4.17. These circular paths do not always happen in the same plan but in various plans, even transversal plans, as expected.

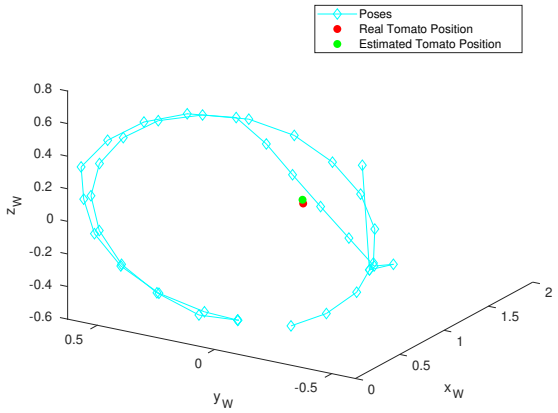
The initial analysis provided insights into the performance of the two models but left the exploration of their limitations and recovery capabilities untouched. Therefore, we expanded our investigation to include a recoverability analysis for both loss functions. This was accomplished by conducting multiple tests aimed at accurately estimating the positions of fruits in the 3D task space, starting with an initial estimation error ranging from 0 cm to 50 cm, in 1 cm increments. For each level of initial error, we carried out ten simulations, averaging the results to maintain consistency across the benchmark. Figures 4.18 and 4.19 depict the variance in average errors under different initial conditions. To simplify, we focused on extreme scenarios only, namely tests E2.1, E2.5, E2.6, and E2.10. These tests were chosen for their straightforward conditions – an average object distance of 1 m and full restrictions, including manoeuvrability constraints. The findings suggest that both loss functions perform comparably under stringent conditions. However, the dispersion minimisation loss function (4.28) allows for greater initial estimation errors, is more straightforward to compute, and facilitates quicker and easier next viewpoint estimation.



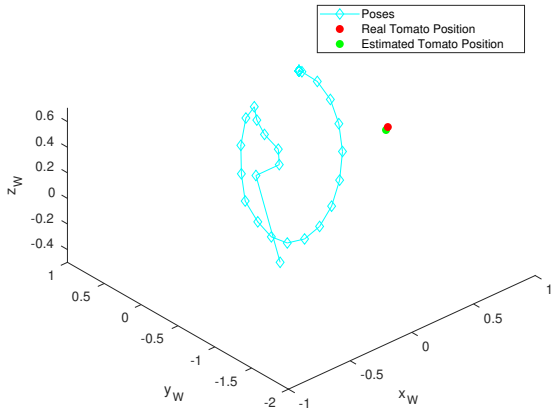
(a) E2.1



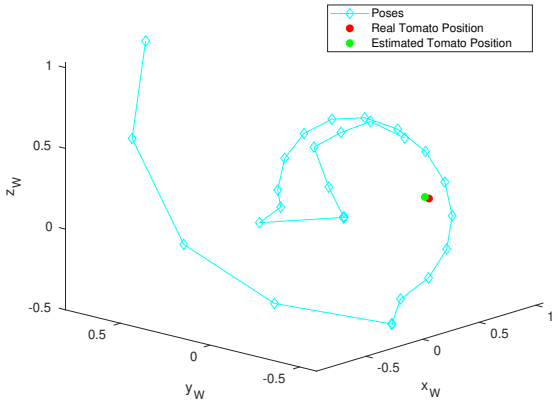
(b) E2.6



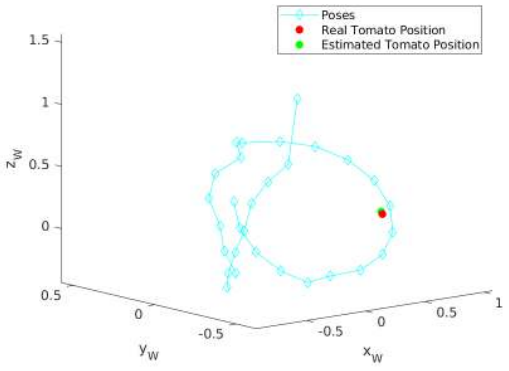
(c) E2.2



(d) E2.7

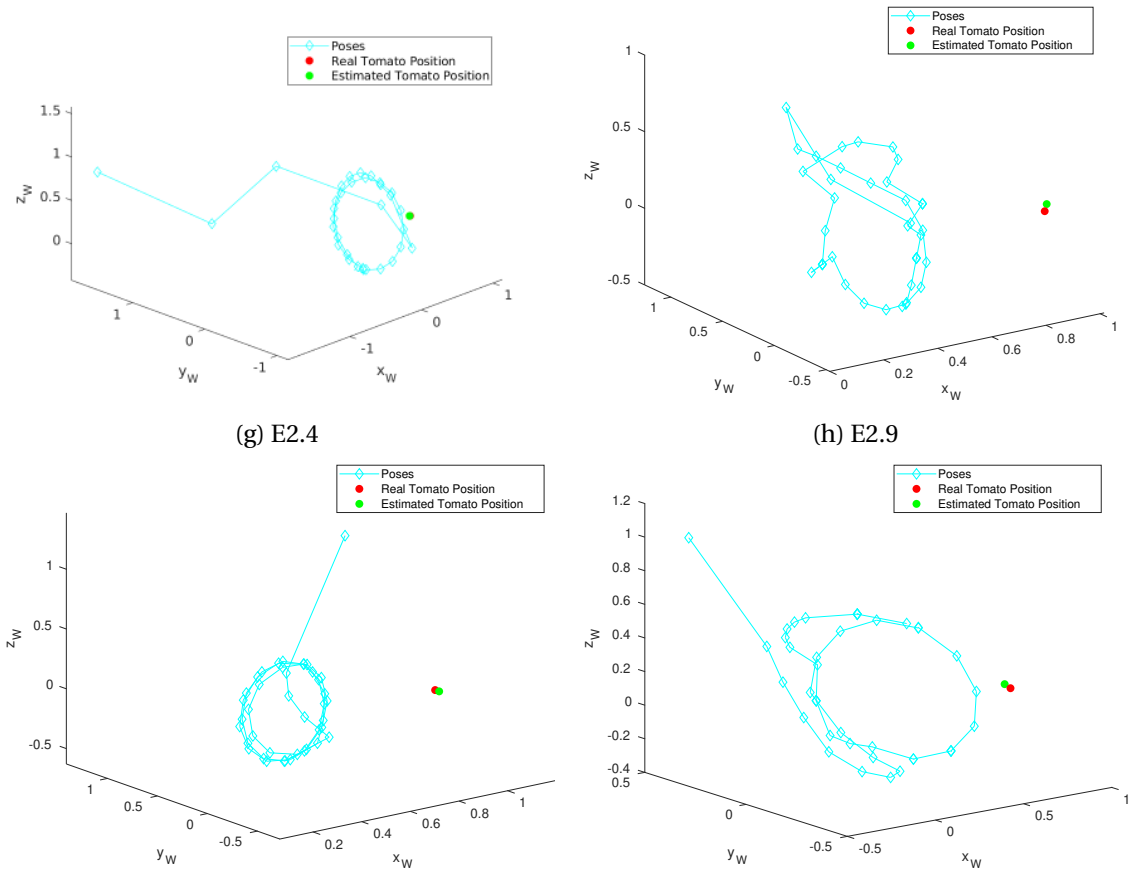


(e) E2.3



(f) E2.8

While effective in identifying the best viewpoint for estimating fruit positions, it is critical to ensure that the sensory apparatus can also move towards the object for additional tasks. This involves first positioning the fruit and then approaching it. Employing a well-designed loss function, such as (4.31), allows the BVE + EKF algorithms to iteratively refine the fruit's position while advancing towards it. Figure 4.20 demonstrates a potential sensor path from the starting pose to the object under constraint E2.5, showcasing a circular trajectory as the algorithm corrects the fruit's position.



(g) E2.4 (h) E2.9

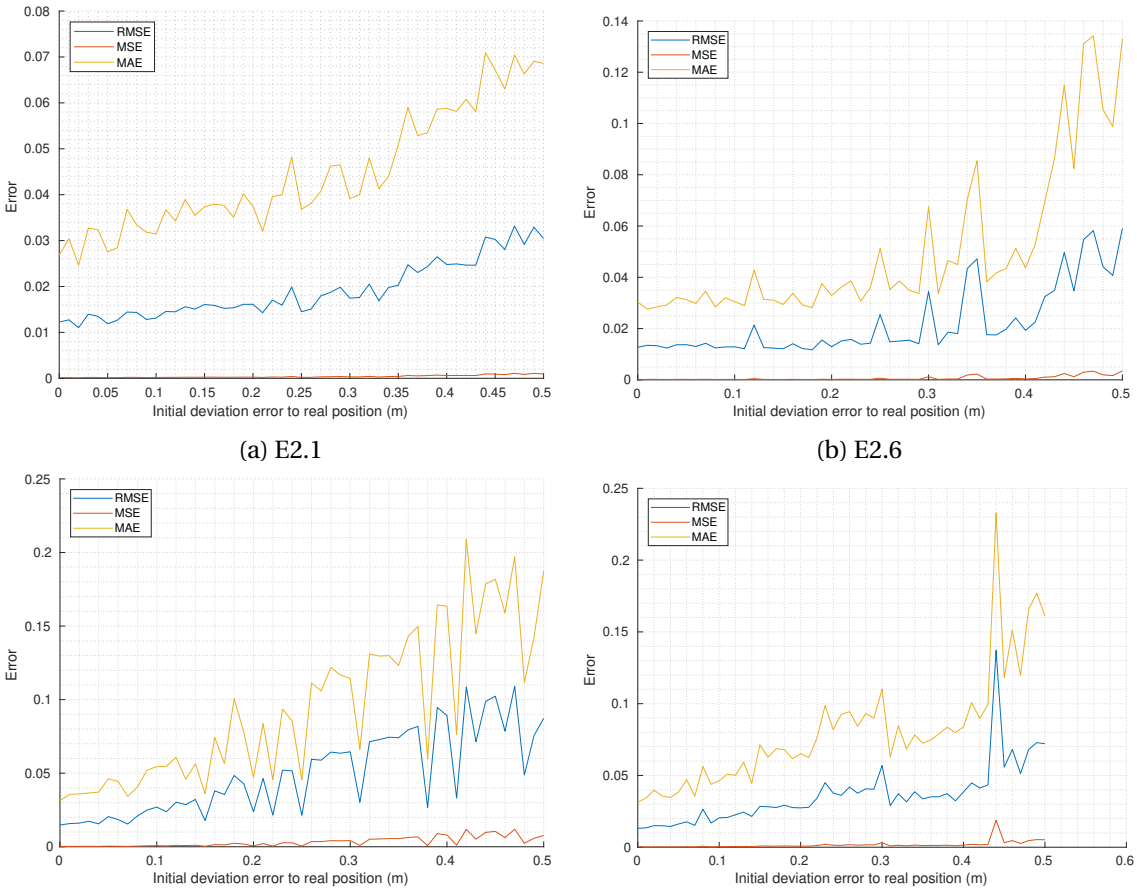
(i) E2.5 (j) E2.10

Figure 4.17: Sample paths generated by the different experiments to assess the fruit's position.

Similarly, we also analysed the recoverability of the algorithm employing the loss function (4.31). The results, depicted in figure 4.21, indicate that this function leads to poorer performance and greater error accumulation. The algorithm remains reliable with an initial estimation error up to about 15 cm. Beyond this threshold, the resulting final estimation error becomes excessively large.

4.4 Discussion

The MonoVisual3DFilter demonstrates effectiveness in estimating the 3D position of tomato fruits with promising results. When tested in a simulated environment, the system consistently achieved a maximum error of less than 10 mm. Enhancing the resolution of the discretised state space could potentially improve the system's accuracy, albeit at the cost of increased processing time and memory consumption. Introducing noise into the system, as done in the second experiment, highlights the significance of employing smooth kernels. Unlike the square kernel, which, due to its binary aggressive behaviour, may reject some state space positions irrecoverably, the Gaussian filter exhibits a smooth approach,



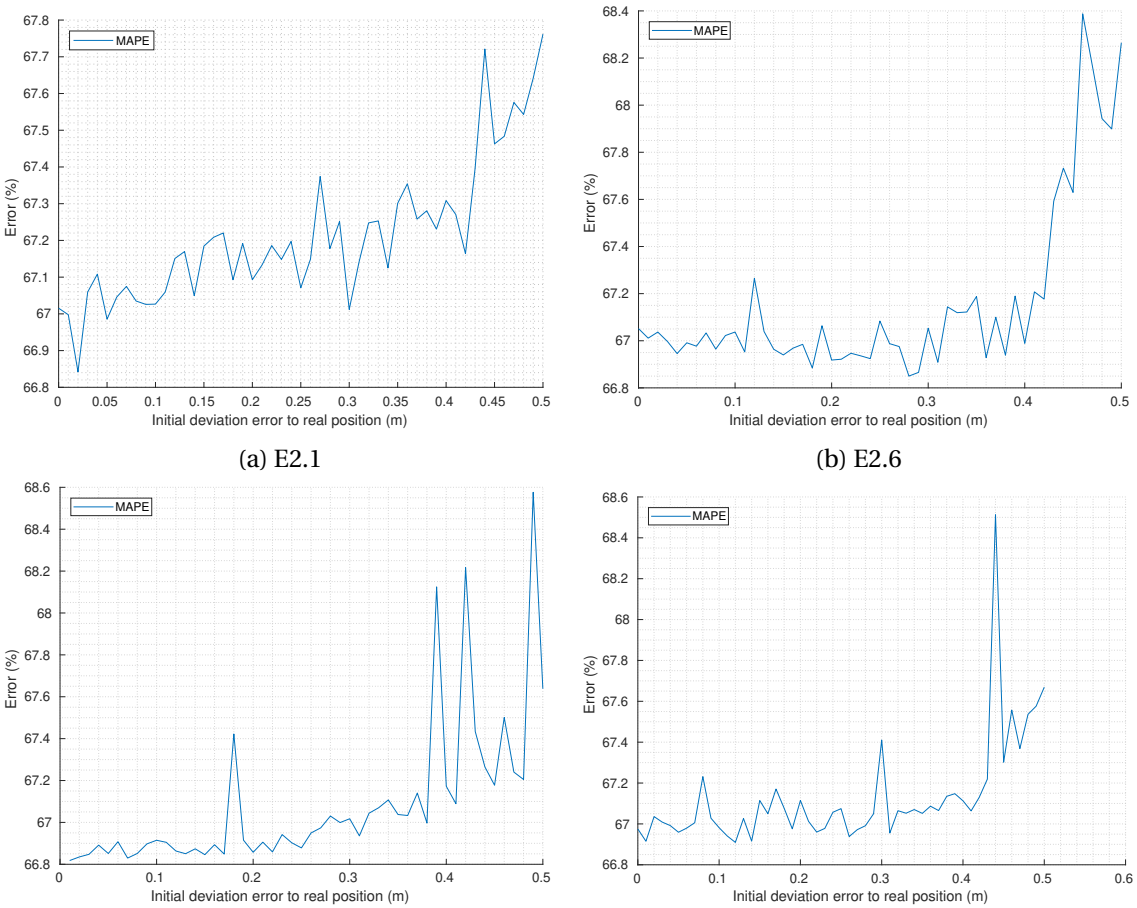
(a) E2.1

(b) E2.6

(c) E2.5

(d) E2.10

Figure 4.18: Average error for the recoverability of the loss functions for the BVE + EKF considering different initial estimation errors.



(a) E2.1

(b) E2.6

(c) E2.5

(d) E2.10

Figure 4.19: Average MAPE for the recoverability of the loss functions for the BVE + EKF considering different initial estimation errors.

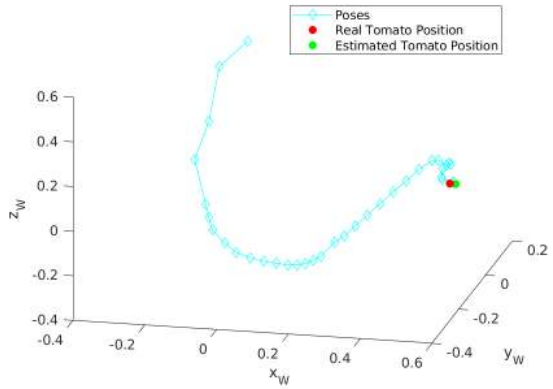


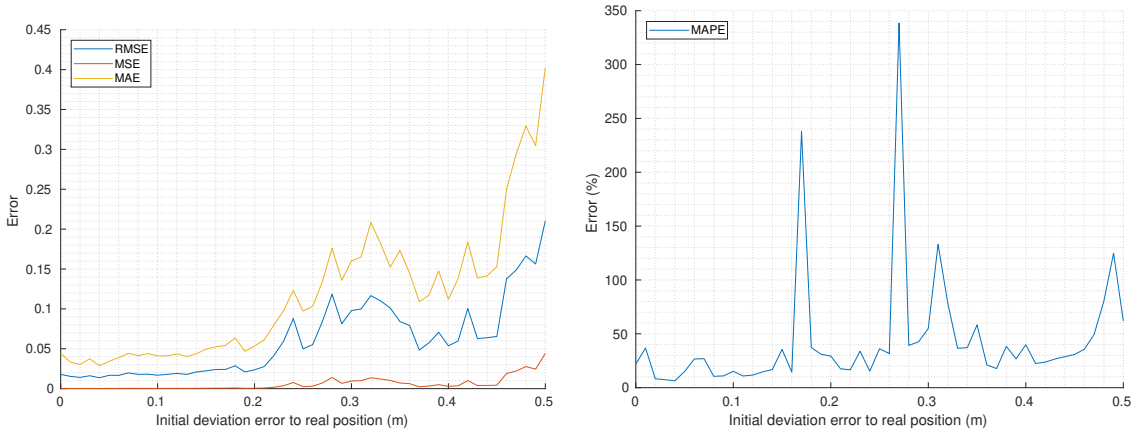
Figure 4.20: Sensor approximation's path using the loss function (4.31) and the restrictions such as in E2.5

iteratively eliminating positions based on their proximity to the filter's centre. This allows for the recovery of some points in the state space, making smooth kernels advantageous. Furthermore, the Gaussian filter, despite being more prone to failures in detecting fruits compared to the square kernel, enables a more accurate estimation of the tomato's centre by prioritising positions nearer to it and reducing deviations caused by sparse clouds.

However, when applying the algorithm to a real robot and camera set-up in a testbed, the performance deviated from expectations. Surprisingly, the Gaussian kernels were less effective, and square kernels yielded higher accuracy. In this real-world application, the choice between geometric or weighted centre estimations became irrelevant. The system reported a MAE of about 20 mm, which could increase to nearly 60 mm, potentially limiting the algorithm's applicability for more precision-demanding tasks. Despite these findings, it's crucial to further investigate the source of the error, whether it stems from the MonoVisual3DFilter or inaccuracies in the baseline used for ground truth, which could be poorly aligned with the tomato's centre. It was noted that positioning the camera closer to the fruits could potentially diminish the error margin. The observed error levels might not be critical for applications involving soft-grippers for harvesting or when used alongside complementary algorithms. For monitoring purposes, the impact of such errors could be even less significant.

Additional experiments shed light on the importance of selecting appropriate viewpoints. Co-linear viewpoints failed to effectively estimate the positions of objects, whereas viewpoints arranged perpendicularly to one another enhanced the filter's view intersection, leading to more accurate estimations. This consideration is particularly relevant in real-world and testbed scenarios, where manipulators often face positional and orientational constraints.

The feasibility of employing the MonoVisual3DFilter for partially occluded fruits was also evaluated. While completely occluded objects remain beyond the detection capabilities of the algorithm, the MonoVisual3DFilter could handle partial occlusions effectively, estimating the positions of tomatoes with reasonable accuracy. This suggests that the MonoVisual3DFilter is relatively unaffected by occlusions in terms of object position estimation.



(a) Error (b) MAPE

Figure 4.21: Average Error and MAPE for the recoverability of the loss functions for the BVE + EKF considering different initial estimation errors for the loss function (4.31) and the restrictions such as in E2.5.

Additionally, both the BVE and EKF algorithms showed promise in simulating the position estimation of fruits, warranting further investigation in laboratory and real-world conditions to fully validate the algorithms. At this stage, it appears that the MonoVisual3DFilter slightly outperforms these algorithms in a simulated environment. Employing both BVE and EKF alongside the MonoVisual3DFilter could complement their results, enhancing the accuracy of region of interest positioning. An ensemble strategy that combines both solutions could refine the object's position estimation, as illustrated in Figure 4.22, based on the confidence delivered by each algorithm. Such an ensemble model [284], carefully designed and benchmarked against state-of-the-art methodologies, could offer a more effective solution for precise object positioning.

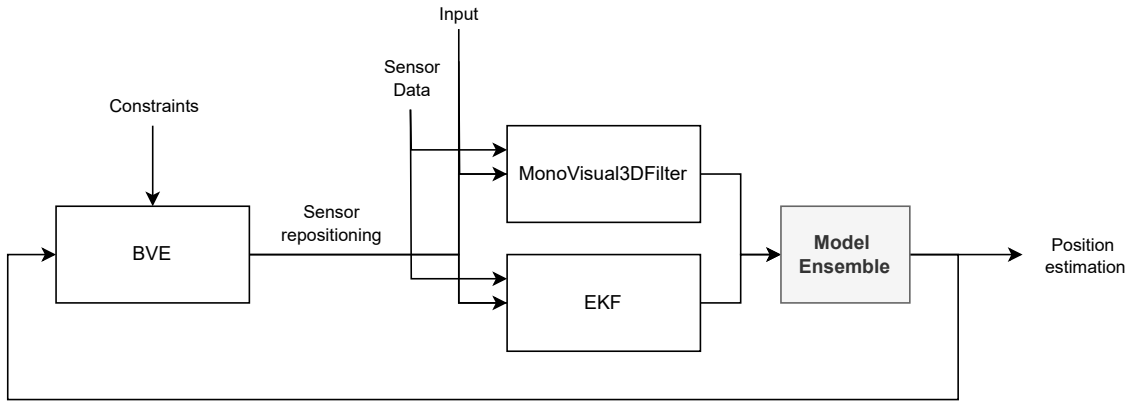


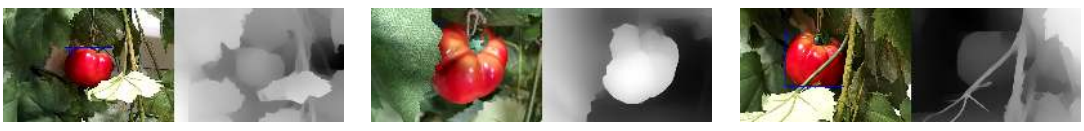
Figure 4.22: Diagram of a proposal algorithm to ensemble the results of multiple algorithms.

In the reviewed literature, it is evident that DL solutions are frequently applied to deduce depth information from monocular RGB cameras, as highlighted in section 2.6.2. A notable example of such a solution is the MiDaS network, which is designed to efficiently estimate

depth from images [285, 286]. For the purpose of comparison with the MonoVisual3DFilter, the MiDaS v3.1 DPT SWIN2 Large 384 model was utilised on the images from our experiment, as shown in Figures 4.7g, 4.7h, and 4.7i. The results, depicted in Figure 4.23, reveal the outputs of the MiDaS CNN for the selected images. Since the network generates a relative pose, calibration is necessary to accurately estimate the objects' real depth from the camera. According to [285], the absolute depth can be determined using a linear regression curve. Following this approach, a preliminary calibration was conducted, the results of which are illustrated in Figure 4.24a. Despite these efforts, as can be inferred from Figure 4.23, the depth images appear flat, making it challenging to discern the depth of the fruit.

Consequently, the network struggles to accurately determine the fruit's position, with errors reaching up to 10 cm as shown in Fig. 4.24b. However, further depth assessments and calibration processes are necessary to definitively ascertain the limitations of MiDaS in-depth estimation. Specifically, MiDaS may require a complete RGB-D system to calibrate the RGB sensor and network properly. Despite these shortcomings, it is important to note that DL-based solutions generally demand significant computational resources and are not straightforward, complicating efforts to enhance results and identify error sources. Moreover, they necessitate extensive training and reliable data. Conversely, the MonoVisual3DFilter is independent of data, offering more predictable performance.

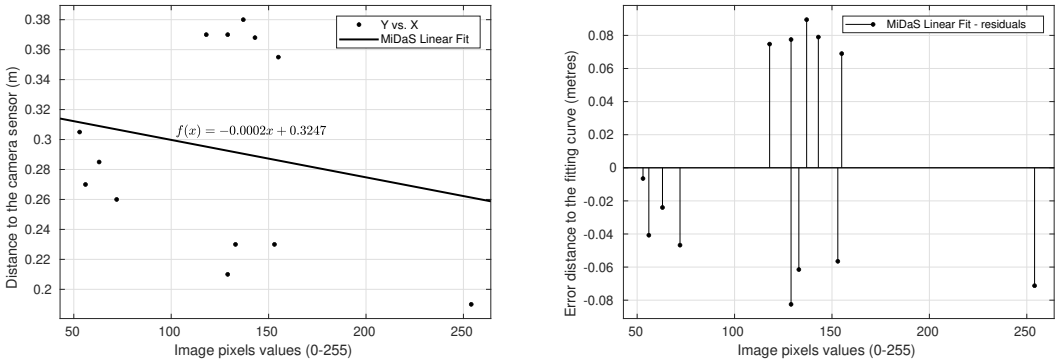
At the current development phase, comparisons with state-of-the-art methodologies like the BVE and EKF are preliminary. A comprehensive comparison with the state-of-the-art necessitates experiments under robotic simulation conditions or real-world scenarios, either controlled or uncontrolled. Nonetheless, a comparative analysis of the results presented in table 4.2 suggests certain conclusions. Despite achieving superior outcomes, the operational capabilities of manipulators may be hindered by less stringent constraints due to their inherent limitations. Therefore, not all camera poses selected may be feasible for the manipulators. Opting for intermediate solutions that consider both the features of the manipulator and the operational area, including the camera's positioning relative to the fruit, can yield satisfactory results with minimal estimation errors. Nonetheless, the feasibility of selected poses remains a critical challenge.



(a) (b) (c)

Figure 4.23: RGB and Depth images from the MiDaS v3.1 DPT SWIN2 Large 384 for estimating the tomatoes' distance to the camera's sensor.

The MonoVisual3DFilter can estimate objects' position in 3D space, particularly those that are circular. This algorithm is computationally intensive, requiring significant resources and time to compute a solution. In our testing, it took approximately one minute per pose to compute the decomposed state space on an Intel Core i7 with 8 GB of RAM. However, the



(a) (b)

Figure 4.24: Calibration curve to estimate the absolute depth in metres to the camera sensor for MiDaS CNN

algorithm is highly parallelisable since all positions are independent and can be computed simultaneously, allowing for potential speed improvements.

Implementing the MonoVisual3DFilter in parallel on a CPU, GPU, or especially an FPGA could significantly enhance inference speeds. Additionally, optimising the code, particularly by utilising more efficient programming languages like C or C++ instead of the less efficient Python, could further improve performance.

To assess the benefits of parallelising the MonoVisual3DFilter, we applied Amdahl's law [287], represented by the equation 4.56. In this equation, $\sigma(n)$ denotes the inherently sequential computations, $\varphi(n)$ represents the potentially parallel computations, and p is the number of processors. It is important to note that Amdahl's law does not account for communication between processes, thus only calculating the maximum theoretical speedup, $\Psi(n,p)$.

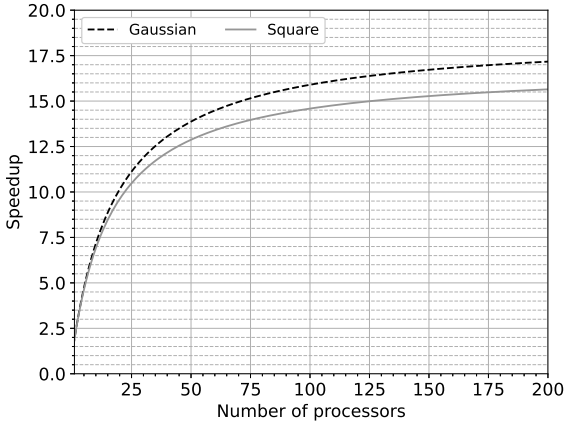


Figure 4.25: Analysis of maximum speedup for parallelisation according to Amdahl's Law for the Gaussian and Square kernels.

$$\Psi(n,p) \leqslant \frac{\sigma(n)+\varphi(n)}{\sigma(n)+\frac{\varphi(n)}{p}} \tag{4.56}$$

Our study illustrated the maximum achievable speedup through parallelisation in the plot 4.25. We observed that using the Gaussian filter, a significant speedup of about 17.5 was achievable. However, the square kernel, being a more straightforward operation, produced a lower speedup, limited by the inherently sequential operations that cannot be optimised. In simulations on a computer with an Intel Core i7 and 8 GB of RAM, the histogram filter's performance was around 2 s per pose with the Gaussian Filter and the square kernel without parallelisation. The number of available cores also influences the speedup potential of the histogram filter. Given its high parallelisability, the algorithm benefits from utilising many cores, making CPUs less appealing compared to GPUs or FPGAs, which are not as limited in the number of cores.

4.5 Conclusion

In this experiment, we developed a histogram filter-based algorithm named MonoVisual3DFilter to determine the 3D positions of tomatoes within the canopy of the tomato plant using monocular cameras. The algorithm showed promising results, achieving an overall error margin of approximately 20 mm in conditions controlled within a laboratory environment. To complement the MonoVisual3DFilter, we also implemented the BVE and EKF algorithms. The BVE algorithm aims to minimise the covariance observation error across regions of interest by estimating new viewpoints. On the other hand, the EKF uses known characteristics of the objects to iteratively estimate the positions of the fruits. In mathematical simulations, this combined approach reported an error ranging from 15 mm to 32 mm.

While the MonoVisual3DFilter proved effective in estimating the positions of tomatoes, it requires further improvements. Future efforts should focus on optimising the selection of observation poses, allowing these poses to be adjustable and variable depending on the fruit being analysed, and enhancing observability through intelligent algorithms like the BVE. Integrating the BVE with the MonoVisual3DFilter could significantly enhance the system, moving it towards an active perception system. Additionally, we could improve the execution time by optimising the existing code and employing parallelisation strategies.

Further testing of the BVE and EKF algorithms in both robotics simulation and controlled laboratory conditions is essential to fully validate their effectiveness for operational active perception robotics.

Once both algorithms are thoroughly validated, investigating ensemble strategies to combine the outcomes provided by the MonoVisual3DFilter and the BVE + EKF should be considered. This approach could help in reducing errors and biases, thereby yielding more accurate estimations for the objects.

Therefore, employing histogram filters, specifically the MonoVisual3DFilter, for object position estimation is feasible and appropriate for use in controlled field conditions. Future studies should explore real-world applications as well as the implementation of active perception strategies. Moreover, incorporating the BVE represents a promising direction for enhancing the MonoVisual3DFilter. Integrating Kalman filters with the BVE is also seen as a viable solution for achieving our objectives.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis investigates active perception techniques for enhancing autonomous harvesting in open-field environments with manipulation and mobile robots. The study unfolds in two principal phases: the initial focus on perception strategies for fruit detection under varying weather conditions, followed by an examination of sensor placement for improved region monitoring.

A pivotal goal, as outlined at the beginning (goal 1), queries whether a robotic manipulator can effectively detect and harvest fruits in unstructured environments using cost-effective and compact sensors. This overarching inquiry was subdivided into three detailed research questions:

1. The exploration of RGB cameras' effectiveness in fruit and tree detection forms the first research question. Despite inherent limitations, these cameras offer cost-efficiency and versatility, easily adapting to different scenarios. The emergence of DL has revolutionised image analysis, enabling precise identification of diverse objects under various conditions. This study concentrated on rapid-processing object detection algorithms, such as YOLOs and SSDs, leveraging different backbones to recognise various fruits. Given the unpredictable nature of DL models, additional image analysis techniques based on colour features were also applied to refine detection accuracy.

2. The second question investigates the capability to detect fruits within the 3D task space rather than merely in the image space. Although RGB-D sensors are widely referenced in literature, their effectiveness is compromised in open-field environments due to uncontrollable natural and weather disturbances. The research introduced two novel algorithms to the field: the MonoVisual3DFilter and the BVE + EKF. Both algorithms employ triangulation for task space analysis from different perspectives. The MonoVisual3DFilter relies on histogram filters and object detection algorithms, whereas the BVE + EKF maximises observation orthogonality and utilises an EKF for iterative position

estimation of the regions of interest. These algorithms demonstrated potential but necessitated further validation under real conditions.

3. The third question delves into the acceleration and optimisation possibilities for the proposed algorithms, focusing on the computing demands that pose challenges for practical robot implementation. Early development stages of the MonoVisual3DFilter and the BVE + EKF preclude extensive optimisation studies; hence, the emphasis shifted to enhancing object detection algorithms. Investigations centred on small, fast DL models like YOLO, SSD MobileNet v2, and RetinaNet ResNet 50, with the latter showing less efficiency. Improvement efforts involved acceleration devices like TPUs, GPUs, and FPGAs, with FPGAs emerging as highly configurable options. Using the RetinaNet ResNet 50 on an AMD-Xilinx ZCU104, inference speeds of up to 25 FPS were achieved.

This thesis also contributes to the agricultural domain by releasing specific datasets in open access, including the TRIBE AGROset [213], a comprehensive dataset featuring 22 classes of trees, fruits, and vegetables.

Our research is competitive with the current literature at different levels. In the area of object detection, we use state-of-the-art DL models and also provide remarkable datasets specifically designed for agricultural contexts. We have improved these models by using dedicated hardware, and provide a complete performance analysis to show that object detectors can now be accelerated to meet the requirements of near real-time detection robots.

For 3D localisation of objects, we take a more classical approach based on statistical analysis instead of using state-of-the-art ML and DL models. This approach results in a more predictable solution and makes it more reliable. As a result, this solution is easier to certify for operational robots.

In summary, this research advances the state-of-the-art in object detection through DL models and significantly enriches the agricultural community with valuable datasets. Besides, this research also contributes to algorithms for object localisation in the 3D space using monocular cameras.

5.2 Future Work

Several objectives can be established for future work based on the available solutions and their respective limitations. However, the ultimate goal should be developing a dependable robotic system that can effectively operate in the field autonomously.

To achieve this, it is crucial to incorporate an accelerated visual perception system into a mobile manipulator to identify regions of interest. Additionally, the system should be equipped with other essential features, including the ability to identify cutting points and relevant monitoring points in agricultural contexts.

Furthermore, after thorough validation, the MonoVisual3DFilter and the BVE + EKF should be combined to enhance the accuracy of their respective results and provide a more

precise position of the fruit. In case of failure, these solutions should also be able to substitute for each other.

Once all components are integrated, the robotic system should evolve to gather more comprehensive information about the scene and register it for improved operation. The active perception system should also have the capacity to utilise the robot's complementary sensors to augment the provided information and validate its results.

Appendix A

The Literature Review Protocol

A.1 Introduction

The literature review presented in chapter 2 is underpinned by a comprehensive systematic review conducted in accordance with the PRISMA protocol, elaborately detailed in the study by Magalhães *et al.* [C41]. Adhering to the PRISMA protocol ensures a rigorous and replicable methodology, facilitating the re-examination of the review's conclusions with precision. Consequently, the primary goal of this appendix is to meticulously outline the protocol employed during the literature review process detailed in Chapter 2.

The *rationale* methodology guiding the literature review is outlined in section 2.1, emphasizing the exploration of active perception applications within the agricultural sector.

A.1.0.1 Objectives

The systematic review critically evaluates existing literature on deploying active perception in harvesting robots. The study's primary objectives are to elucidate:

- the main strategies used by the authors to detect the fruits under natural conditions;
- strategies devised to address challenges posed by occluded fruits and stems;
- leading approaches for harvesting fruits;
- sensor technologies utilised for fruit and tree detection;
- mechanisms through which robots enhance their understanding of the surrounding environment.

The scope of this research is delineated within the PICOC[1] framework as outlined by Wohlin et al. [288]:

Population: harvesting robots

[1] Population—intervention—comparison—outcome—context

Intervention: fruit detection, segmentation and harvesting using active perception

Comparison: *not applicable for the current study*

Outcome: A set of different, and cascade algorithms able to perceive the environment and control the robot with the goal of acquiring information about the environment and actively detecting the most important key points for successfully harvesting the ripe fruits.

Context: For this research, is mainly considered the primary publications of robots in the agricultural context.

It's important to note that this review specifically targets harvesting strategies and the componentry of robotic harvesters, explicitly excluding an analysis of the harvesting tools themselves.

A.1.1 Literature review methodology

A.1.1.1 Search strategy and selection and data collection processes

Following a comprehensive investigation of scientific databases, it is imperative to report thousands of articles for review. Establishing inclusion and exclusion criteria facilitates a fair evaluation of these publications. For the systematic review process, we employed the online tool Parsifal [289], which aids in structuring the entire research methodology – including protocol definition, elimination of duplicates, screening, quality assessment, and data extraction.

In this systematic review, our focus was exclusively on primary indexed publications concerning fruit harvesting robots from 2016 up to September 21, 2021. After removing duplicated entries, the remaining publications were screened based on their titles and abstracts. They were excluded if they met any of the following criteria:

- The publication does not refer to a fruit harvesting robot in agriculture.
- The publication is not in English.
- The publication dates back to before 2016.
- The publication is not a primary manuscript[2].

Subsequently, we conducted a thorough reading of the chosen manuscripts. Each manuscript underwent a quality assessment to ascertain its alignment with the objectives of the current study. The assessment evaluated each manuscript on three parameters: Yes (1.0), Partially (0.5), and No (0.0). Publications scoring less than 3.0 were excluded from further data extraction. The quality assessment criteria included:

[2]Here, a primary manuscript is defined as works that present an experimental study, a benchmark, or introduce a new algorithm or method within the agricultural domain specifically for fruit harvesting.

- Does the paper refer the sensory system?
- Does the work incorporate active perception?
- Did the authors implement a scene information-enhancement algorithm?
- Did the authors employ detection, segmentation, or both?
- Is the research applicable to agriculture and harvesting robots?
- Was the work tested in real-world scenarios[3]?

In the data extraction phase, we compiled information on:

- The fruits targeted by the studies;
- The sensory systems utilised;
- Strategies for fruit detection and segmentation; and,
- Any additional active perception algorithms.

As the review progressed, it became evident that most publications primarily focused on detection and segmentation algorithms. Consequently, the review had to be partially concentrated on this aspect.

The literature review was limited to publications sourced from scientific databases: ACM Digital Library [290], EI Compendex [291], IEEE Digital Library [292], ISI Web of Science [293], and Scopus [294]. The search utilised the following base string, with necessary adaptations for each database's specific requirements:

```
(robot*) AND (agricultur* OR harvest* OR open-field OR prun*) AND ("active
    perception" OR "active sensing" OR "active vision" OR "viewpoint" OR
   detect OR segment* OR "visual servoing") AND (fruit)
```

The term `robot*` aims to capture all publications mentioning robot, robots, robotic, or robotics. With interest in harvesting robots, the terms `agricultur* OR harvest* OR open-field OR prun*` target publications related to agricultural robots, harvesting, pruning, or those designed for open-field environments. Lastly, a specific focus on robots employing active perception for fruit harvesting is encapsulated by `("active perception" OR "active sensing" OR "active vision" OR viewpoint OR detect* OR segment* OR "visual servoing") AND (fruit)`, thereby including manuscripts on detection, segmentation, and manuscripts discussing active sensing or perception as synonyms. The term "viewpoint" (also known as viewpoint selection) is considered a component of active perception within this context.

[3] Real-world scenarios include open-field farms, agricultural environments, and greenhouses, as opposed to simulations or laboratory testbeds.

A.1.1.2 Search results

The search conducted with the specified base key string across various databases yielded a total of 1034 publications within the period from 2016 to 2021. Figure A.1 illustrates the distribution of publications among these databases. It was observed that all databases contributed an even number of publications, with the exception of the IEEE Digital Library, which accounted for a smaller share. In contrast, the Scopus database appeared more generic, encompassing a broader range of manuscripts from various publishers. Furthermore, while the ISI Web of Science and EI Compendex databases are not limited to specific publishers, they implement more stringent indexing criteria and tend to be more focused on particular fields (mainly, EI Compendex – Engineering Village).

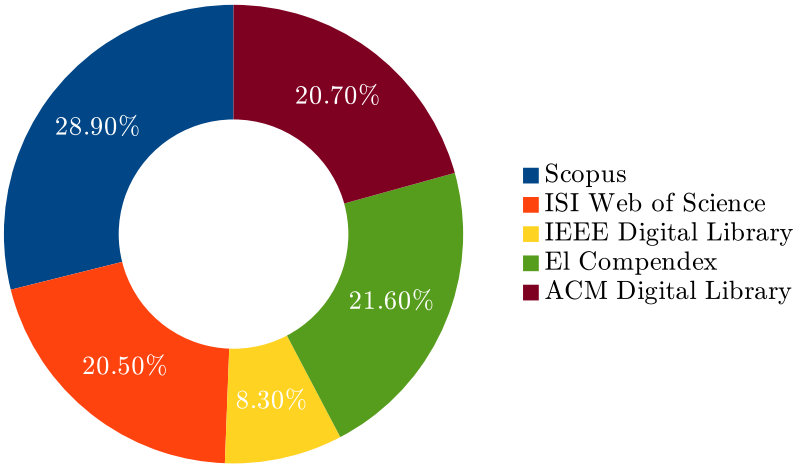


Figure A.1: Distribution of publications across databases based on the search key utilised

An annual reorganisation of the collected data revealed a significant trend within the scope of the current review, as depicted in Figure A.2. Since 2016, there has been a notable increase in publication volume, with 2021 experiencing a slight decline. However, it is important to mention that the analysis for 2021 only extends up to September 21st.

Figure A.3 outlines the publication selection process for this systematic review, adhering to the PRISMA standards as described by [295]. Initially, all potential publications were identified, followed by the removal of duplicates. The screening phase involved examining the titles, abstracts, and figures of articles to determine relevance. Publications not meeting the predefined exclusion criteria were subsequently eliminated. A thorough review of the remaining publications assessed their quality and relevance to the research questions. Out of the 1034 identified publications, only 195 were ultimately deemed suitable for inclusion in the literature review.

This structured approach ensures a comprehensive and methodical examination of the literature within the defined scope, thereby enhancing the reliability and depth of the review findings.

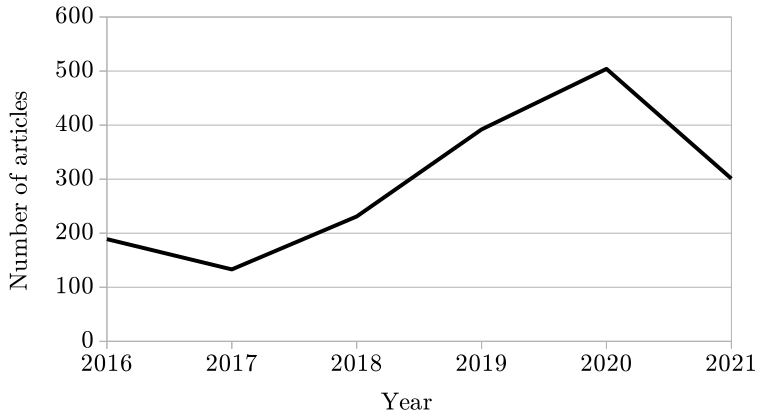


Figure A.2: Yearly evolution of publication volume for the investigated search key.

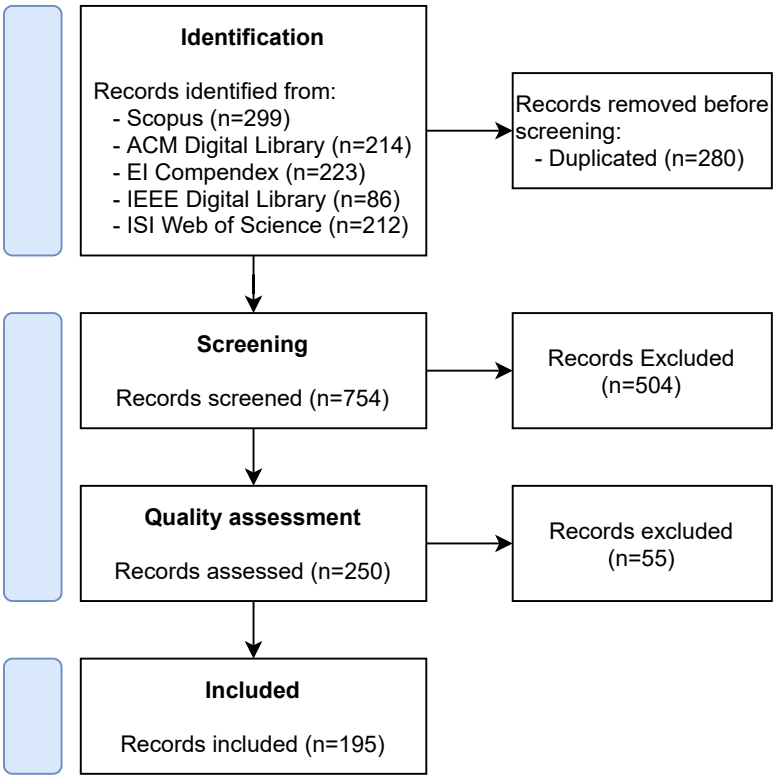


Figure A.3: PRISMA flow diagram illustrating the publication selection process for the systematic review

A.2 Conclusion

The protocol outlined so far presents a rigorous approach to conducting a systematic review. This strict methodology, however, poses challenges for incorporating subsequent developments in the state-of-the-art that may emerge during the evolution of the current PhD thesis.

To accommodate these advancements, the literature review protocol was slightly adapted

beyond its initial parameters, yet its fundamental nature was preserved. This adaptation primarily facilitated the inclusion of new research focusing on fruit detection methodologies and acceleration strategies.

For readers specifically interested in adhering to the original systematic review framework, a detailed description can be found in [C41].

Appendix B

Sample images of the assessment of VineSet and ResNet 50 in heterogeneous platforms

In the chapter 3, we perform multiple essays searching for effective and fast solutions for object detection. Suitable solutions come from deploying one-shot object detectors into high-performing devices such as embedded GPUs, FPGAs, or ASICs – like TPUs, OAK-SoM, or NCSs. These devices, besides being low-power consumption, also provide dedicated resources to execute ANNs.

The cost behind deploying DL models into dedicated hardware is to quantise their layers and use dedicated software for that purpose. Sometimes, that implies losing some accuracy.

Figure 3.37 in section 3.3.2 provides a general analysis of these symptoms, derived from the experiments detailed in section 3.2.3. However, that figure illustrates an overlapping of the multiple tested heterogeneous platforms, difficulting the analysis of the illustrated loss of accuracy is difficult. Therefore, Figures B.1 to B.10 reference a more detailed view of the previous study and allow a deeper observation of the reported conclusions.

Generically, all devices performed similarly and can be used for object detection according to the desired requirements. Sometimes, the TPU looks to perform better, eventually due to the quantisation process that created a more strict network. In other situations, that may mean also a loss of accuracy.

(a) Ground truth (b) RTX3090 TF2 (c) TF-TRT FP32 (d) TF-TRT FP16

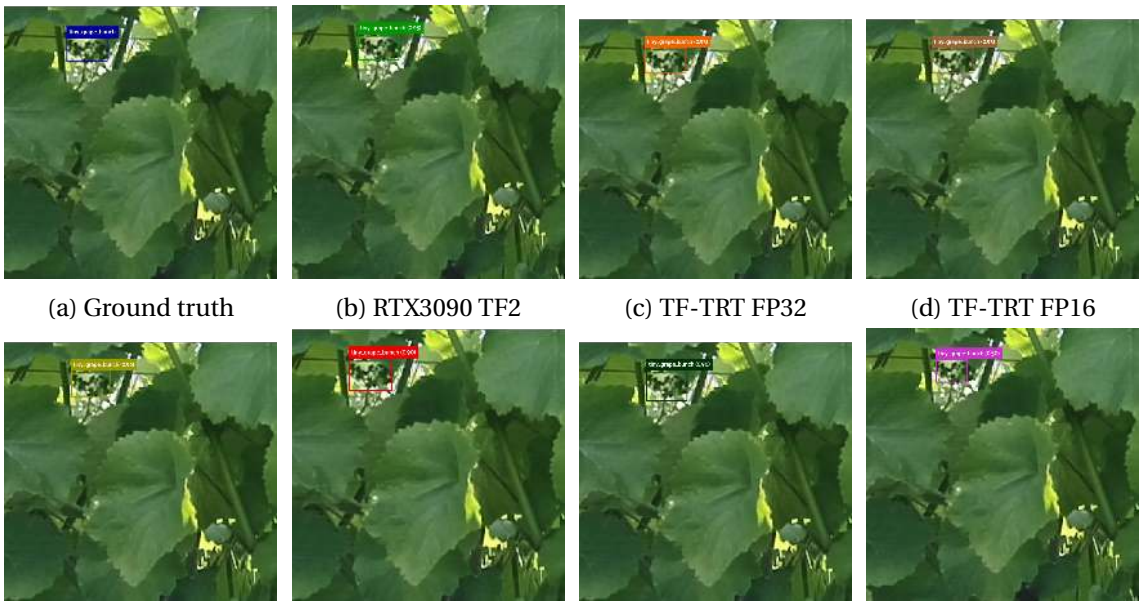
(e) TF-TRT INT8 (f) KV260 (g) ZCU104 (h) TPU

Figure B.1: Detailed sample image 3.37a from figure 3.37 from Blue – ground-truth; light green – NVIDIA RTX3080 TF2; orange – NVIDIA RTX3090 TF-TRT FP32; brown – NVIDIA RTX3090 TF-TRT FP16; dark yellow – NVIDIA RTX3090 TF-TRT INT8; red – AMD-Xilinx Kria KV260; dark green – AMD-Xilinx ZCU104; pink – Coral Dev Board TPU

(a) Ground truth (b) RTX3090 TF2 (c) TF-TRT FP32 (d) TF-TRT FP16

(e) TF-TRT INT8 (f) KV260 (g) ZCU104 (h) TPU

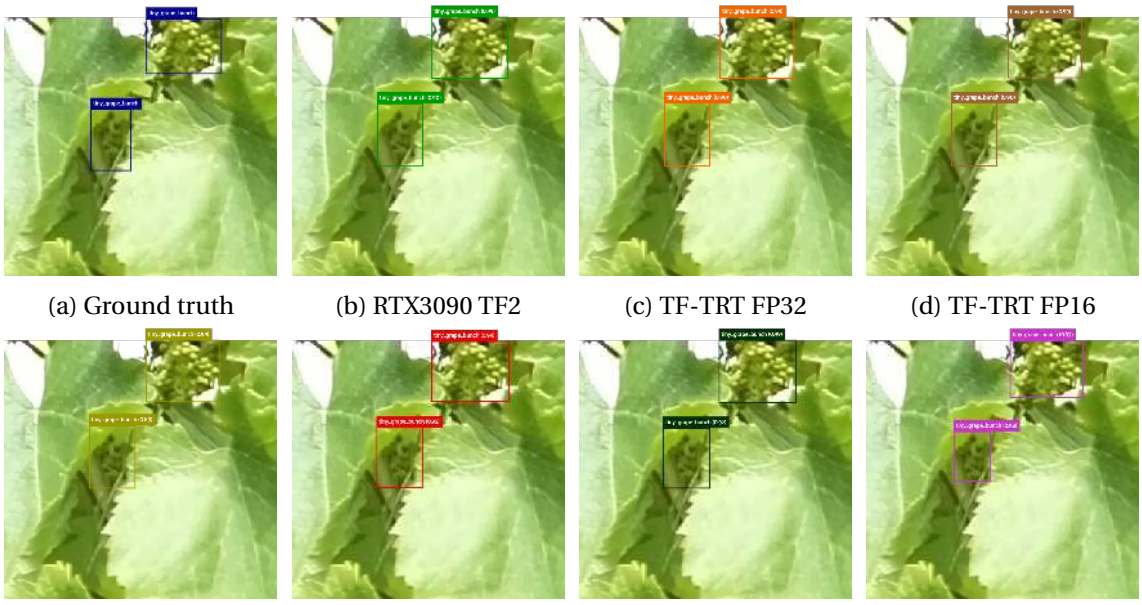
Figure B.2: Detailed sample image 3.37b from figure 3.37 from Blue – ground-truth; light green – NVIDIA RTX3080 TF2; orange – NVIDIA RTX3090 TF-TRT FP32; brown – NVIDIA RTX3090 TF-TRT FP16; dark yellow – NVIDIA RTX3090 TF-TRT INT8; red – AMD-Xilinx Kria KV260; dark green – AMD-Xilinx ZCU104; pink – Coral Dev Board TPU



(a) Ground truth (b) RTX3090 TF2 (c) TF-TRT FP32 (d) TF-TRT FP16

(e) TF-TRT INT8 (f) KV260 (g) ZCU104 (h) TPU

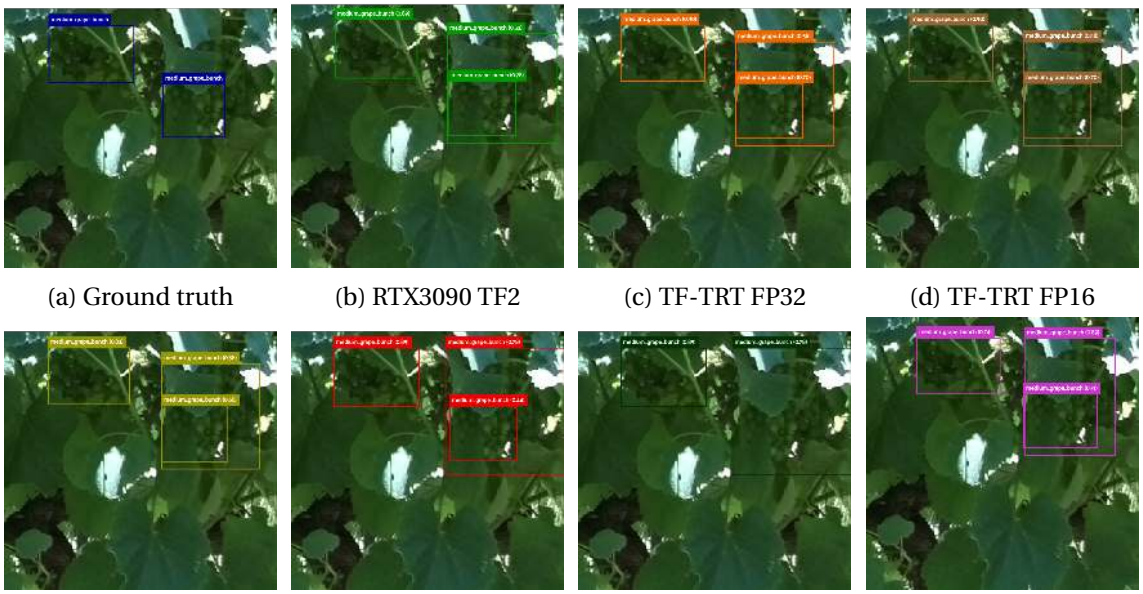
Figure B.3: Detailed sample image 3.37c from figure 3.37 from Blue – ground-truth; light green – NVIDIA RTX3080 TF2; orange – NVIDIA RTX3090 TF-TRT FP32; brown – NVIDIA RTX3090 TF-TRT FP16; dark yellow – NVIDIA RTX3090 TF-TRT INT8; red – AMD-Xilinx Kria KV260; dark green – AMD-Xilinx ZCU104; pink – Coral Dev Board TPU



(a) Ground truth (b) RTX3090 TF2 (c) TF-TRT FP32 (d) TF-TRT FP16

(e) TF-TRT INT8 (f) KV260 (g) ZCU104 (h) TPU

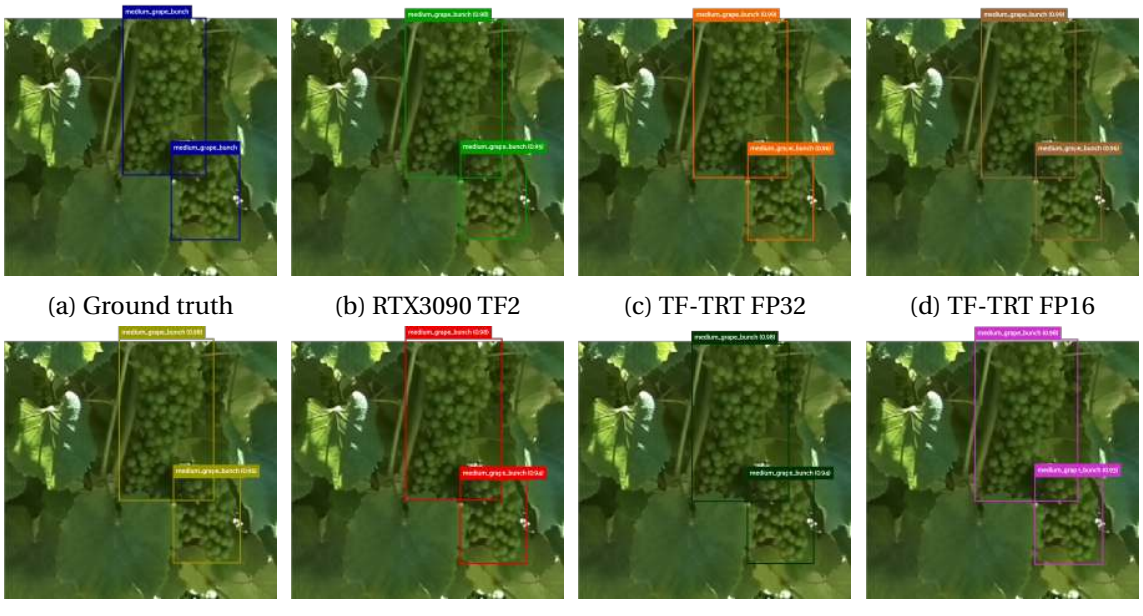
Figure B.4: Detailed sample image 3.37d from figure 3.37 from Blue – ground-truth; light green – NVIDIA RTX3080 TF2; orange – NVIDIA RTX3090 TF-TRT FP32; brown – NVIDIA RTX3090 TF-TRT FP16; dark yellow – NVIDIA RTX3090 TF-TRT INT8; red – AMD-Xilinx Kria KV260; dark green – AMD-Xilinx ZCU104; pink – Coral Dev Board TPU



(a) Ground truth (b) RTX3090 TF2 (c) TF-TRT FP32 (d) TF-TRT FP16

(e) TF-TRT INT8 (f) KV260 (g) ZCU104 (h) TPU

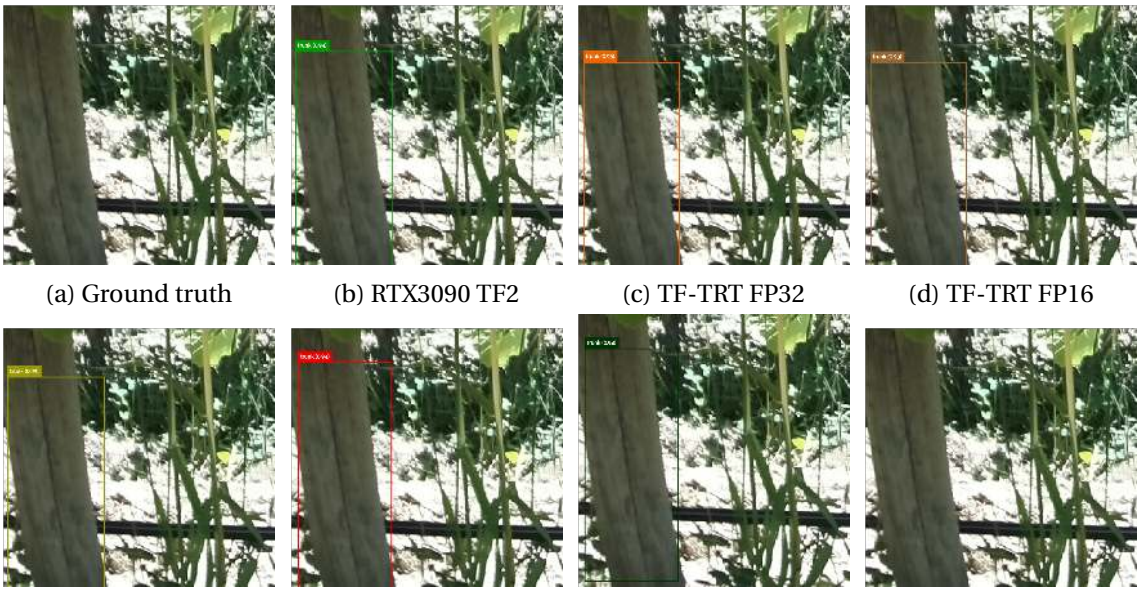
Figure B.5: Detailed sample image 3.37e from figure 3.37 from Blue – ground-truth; light green – NVIDIA RTX3080 TF2; orange – NVIDIA RTX3090 TF-TRT FP32; brown – NVIDIA RTX3090 TF-TRT FP16; dark yellow – NVIDIA RTX3090 TF-TRT INT8; red – AMD-Xilinx Kria KV260; dark green – AMD-Xilinx ZCU104; pink – Coral Dev Board TPU



(a) Ground truth (b) RTX3090 TF2 (c) TF-TRT FP32 (d) TF-TRT FP16

(e) TF-TRT INT8 (f) KV260 (g) ZCU104 (h) TPU

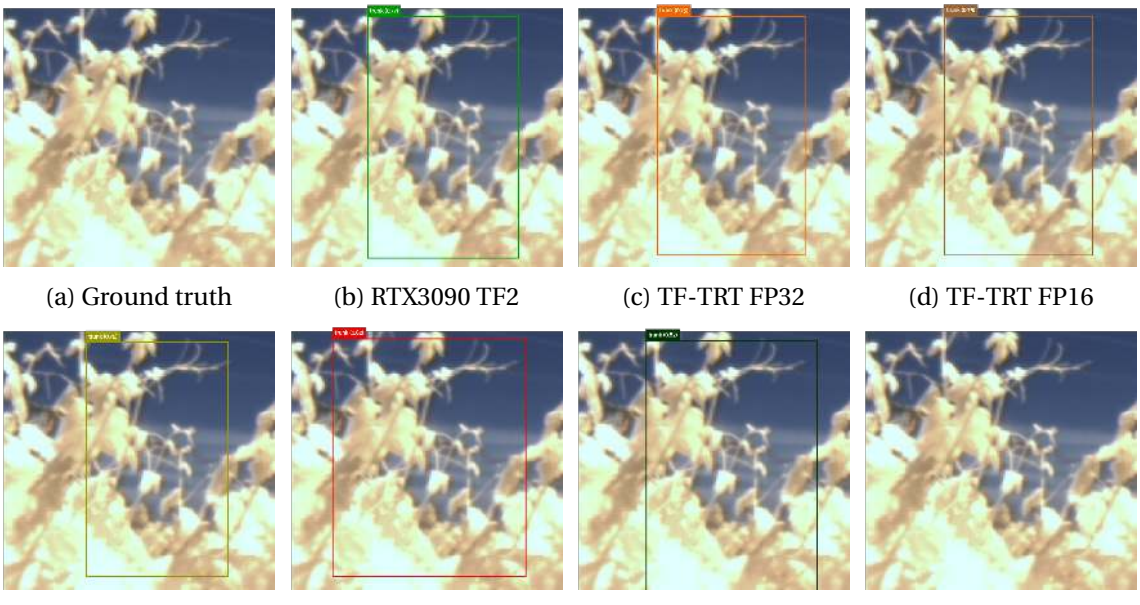
Figure B.6: Detailed sample image 3.37f from figure 3.37 from Blue – ground-truth; light green – NVIDIA RTX3080 TF2; orange – NVIDIA RTX3090 TF-TRT FP32; brown – NVIDIA RTX3090 TF-TRT FP16; dark yellow – NVIDIA RTX3090 TF-TRT INT8; red – AMD-Xilinx Kria KV260; dark green – AMD-Xilinx ZCU104; pink – Coral Dev Board TPU



(a) Ground truth (b) RTX3090 TF2 (c) TF-TRT FP32 (d) TF-TRT FP16

(e) TF-TRT INT8 (f) KV260 (g) ZCU104 (h) TPU

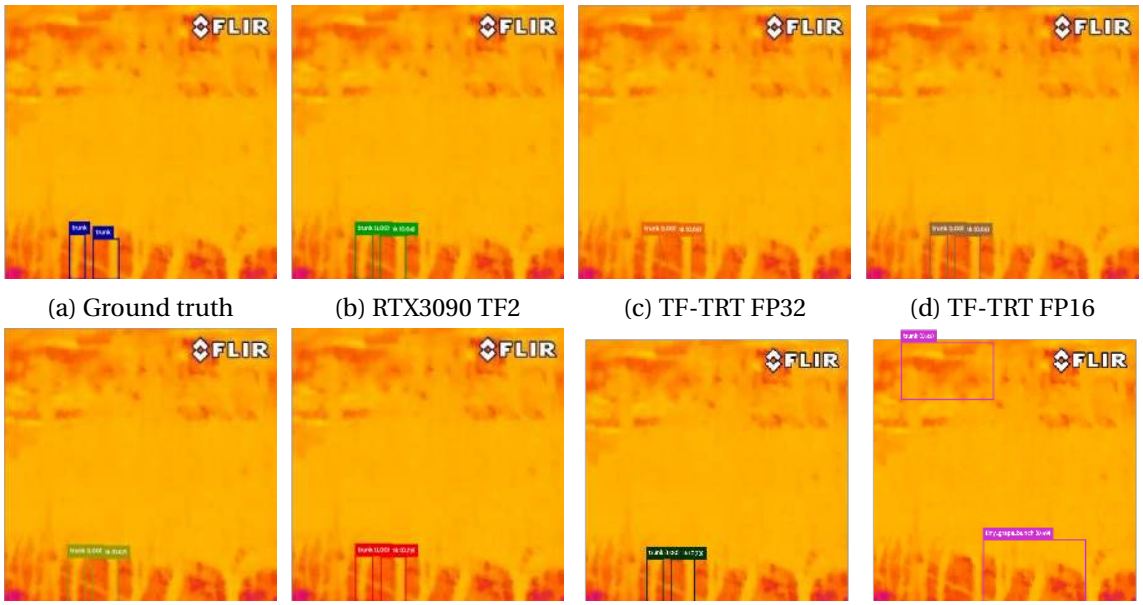
Figure B.7: Detailed sample image 3.37g from figure 3.37 from Blue – ground-truth; light green – NVIDIA RTX3080 TF2; orange – NVIDIA RTX3090 TF-TRT FP32; brown – NVIDIA RTX3090 TF-TRT FP16; dark yellow – NVIDIA RTX3090 TF-TRT INT8; red – AMD-Xilinx Kria KV260; dark green – AMD-Xilinx ZCU104; pink – Coral Dev Board TPU



(a) Ground truth (b) RTX3090 TF2 (c) TF-TRT FP32 (d) TF-TRT FP16

(e) TF-TRT INT8 (f) KV260 (g) ZCU104 (h) TPU

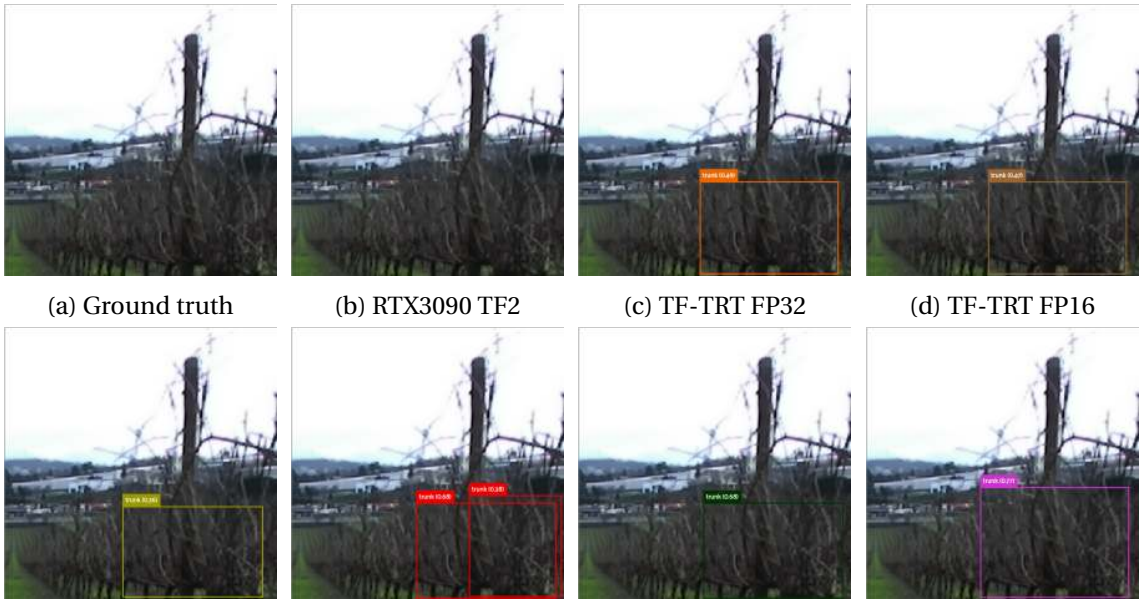
Figure B.8: Detailed sample image 3.37h from figure 3.37 from Blue – ground-truth; light green – NVIDIA RTX3080 TF2; orange – NVIDIA RTX3090 TF-TRT FP32; brown – NVIDIA RTX3090 TF-TRT FP16; dark yellow – NVIDIA RTX3090 TF-TRT INT8; red – AMD-Xilinx Kria KV260; dark green – AMD-Xilinx ZCU104; pink – Coral Dev Board TPU



(a) Ground truth (b) RTX3090 TF2 (c) TF-TRT FP32 (d) TF-TRT FP16

(e) TF-TRT INT8 (f) KV260 (g) ZCU104 (h) TPU

Figure B.9: Detailed sample image 3.37i from figure 3.37 from Blue – ground-truth; light green – NVIDIA RTX3080 TF2; orange – NVIDIA RTX3090 TF-TRT FP32; brown – NVIDIA RTX3090 TF-TRT FP16; dark yellow – NVIDIA RTX3090 TF-TRT INT8; red – AMD-Xilinx Kria KV260; dark green – AMD-Xilinx ZCU104; pink – Coral Dev Board TPU



(a) Ground truth (b) RTX3090 TF2 (c) TF-TRT FP32 (d) TF-TRT FP16

(e) TF-TRT INT8 (f) KV260 (g) ZCU104 (h) TPU

Figure B.10: Detailed sample image 3.37j from figure 3.37 from Blue – ground-truth; light green – NVIDIA RTX3080 TF2; orange – NVIDIA RTX3090 TF-TRT FP32; brown – NVIDIA RTX3090 TF-TRT FP16; dark yellow – NVIDIA RTX3090 TF-TRT INT8; red – AMD-Xilinx Kria KV260; dark green – AMD-Xilinx ZCU104; pink – Coral Dev Board TPU

References

[1] SPARC—The Partnership for Robotics in Europe, "Robotics 2020 multi-annual roadmap - realease B," euRobotics, Tech. Rep., Feb. 2015, p. 308. [Online]. Available: https://old.eu-robotics.net/cms/upload/topic_groups/H2020_Robotics_Multi-Annual_Roadmap_ICT-2017%7BB%7D.pdf.

[2] United Nations, "Extreme poverty in developing countries inextricably linked to global food insecurity crisis, senior officials tell second committee, Delegates Warn Agriculture Sector Underdeveloped, Underfunded, Beset by Crises," United Nations, Meeting Coverage GA/EF/3590, Oct. 2023. [Online]. Available: https://press.un.org/en/2023/gaef3590.doc.htm.

[3] General Assembly, "Transforming our world: The 2030 Agenda for sustainable development," United Nations, Resolution A/RES/70/1, Oct. 2015. [Online]. Available: https://sdgs.un.org/2030agenda.

[4] Food and Agriculture Organization of the United Nations, *FAOSTAT*, License: CC BY-NC-SA 3.0 IGO, Accessed on March 21st, 2023, 2022. [Online]. Available: http://www.fao.org/faostat/.

[5] D. Valera, L. Belmonte, F. Molina-Aiz, A. López, and F. Camacho-Ferre, "The greenhouses of Almería, Spain: Technological analysis and profitability," *Acta Horticulturae*, no. 1170, pp. 219–226, Jul. 2017. DOI: 10.17660/actahortic.2017.1170.25.

[6] V. Sousa Ferreira, "A cultura do tomate em estufa avaliação das condições climáticas em dois tipos de estufa e sua influência na produtividade e nos custos de produção do tomate, na região do oeste," M.S. thesis, Universidade de Lisboa, Lisboa, 2017. [Online]. Available: http://hdl.handle.net/10400.5/13897.

[7] J. M. Cámara-Zapata, J. M. Brotons-Martínez, S. Simón-Grao, J. J. Martinez-Nicolás, and F. García-Sánchez, "Cost–benefit analysis of tomato in soilless culture systems with saline water under greenhouse conditions," *Journal of the Science of Food and Agriculture*, vol. 99, no. 13, pp. 5842–5851, 2019. DOI: 10.1002/jsfa.9857. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jsfa.9857.

[8] S. Fongmul, "Effects of shortage of agricultural workers on food security in Chiang Mai," *Journal of Agricultural Research and Extension*, vol. 37, no. 1, pp. 118–125, 2020.

[9] C. Mitaritonna and L. Ragot, "After Covid-19, will seasonal migrant agricultural workers in Europe be replaced by robots?" CEPII research center, CEPII Policy Brief, 2020. [Online]. Available: https://EconPapers.repec.org/RePEc:cii:cepipb:2020-33.

[10] Z. Li, J. Liu, P. Li, and W. Li, "Analysis of workspace and kinematics for a tomato harvesting robot," in *2008 International Conference on Intelligent Computation Technology and Automation (ICICTA)*, Changsha, China: IEEE, Oct. 2008, pp. 823–827. DOI: 10.1109/icicta.2008.138.

[11] C. W. Bac, E. J. van Henten, J. Hemming, and Y. Edan, "Harvesting robots for high-value crops: State-of-the-art review and challenges ahead," *Journal of Field Robotics*, vol. 31, no. 6, pp. 888–911, Jul. 2014, ISSN: 1556-4967. DOI: 10.1002/rob.21525.

[12] K. Kapach, E. Barnea, R. Mairon, Y. Edan, and O. Shahar, "Computer vision for fruit harvesting robots — state of the art and challenges ahead," *International Journal of Computational Vision and Robotics*, vol. 3, no. 1/2, pp. 4–34, Apr. 9, 2012, ISSN: 1752-9131. DOI: 10.1504/ijcvr.2012.046419.

[13] M. Bergerman, J. Billingsley, J. Reid, and E. Van Henten, "Robotics in agriculture and forestry," in *Springer Handbook of Robotics*, vol. 56, Springer International Publishing, Jan. 2016, pp. 1463–1492, ISBN: 9783319325521. DOI: 10.1007/978-3-319-32552-1_56.

[14] A. Zujevs, V. Osadcuks, and P. Ahrendt, "Trends in robotic sensor technologies for fruit harvesting: 2010-2015," in *Procedia Computer Science*, vol. 77, Elsevier B.V., Jan. 2015, pp. 227–233. DOI: 10.1016/j.procs.2015.12.378.

[15] T. Conceição, F. N. dos Santos, P. Costa, and A. P. Moreira, "Robot localization system in a hard outdoor environment," in *Advances in Intelligent Systems and Computing*, A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, Eds., vol. 1, Springer Verlag, Nov. 2017, ch. ROBOT 2017, pp. 215–227. DOI: 10.1007/978-3-319-70833-1_18.

[16] C. Ji, J. Zhang, T. Yuan, and W. Li, "Research on key technology of truss tomato harvesting robot in greenhouse," *Applied Mechanics and Materials*, vol. 442, pp. 480–486, Oct. 2013. DOI: 10.4028/www.scientific.net/amm.442.480.

[17] Y. Zhao, L. Gong, C. Liu, and Y. Huang, "Dual-arm robot design and testing for harvesting tomato in greenhouse," *IFAC-PapersOnLine*, vol. 49, no. 16, pp. 161–165, 2016. DOI: 10.1016/j.ifacol.2016.10.030.

[18] S. Yasukawa, B. Li, T. Sonoda, and K. Ishii, "Development of a tomato harvesting robot," *Proceedings of International Conference on Artificial Life and Robotics*, vol. 22, pp. 408–411, Jan. 2017. DOI: 10.5954/icarob.2017.os22-1.

[19] F. Taqi, F. Al-Langawi, H. Abdulraheem, and M. El-Abd, "A cherry-tomato harvesting robot," in *2017 18th International Conference on Advanced Robotics (ICAR)*, Hong Kong, China: IEEE, Jul. 2017, pp. 463–468. DOI: 10.1109/icar.2017.8023650.

[20] W. Lili *et al.*, "Development of a tomato harvesting robot used in greenhouse," *International Journal of Agricultural and Biological Engineering*, vol. 10, no. 4, pp. 140–149, Jul. 2017, ISSN: 1934-6344. DOI: 10.25165/j.ijabe.20171004.3204.

[21] C. W. Bac, J. Hemming, B. A. J. van Tuijl, R. Barth, E. Wais, and E. J. van Henten, "Performance evaluation of a harvesting robot for sweet pepper," *Journal of Field Robotics*, vol. 34, no. 6, pp. 1123–1139, Sep. 2017, ISSN: 1556-4959. DOI: 10.1002/rob.21709.

[22] J. J. Roldán *et al.*, "Robots in agriculture: State of art and practical experiences," in *Service Robots*, InTech, Jan. 2018. DOI: 10.5772/intechopen.69874.

[23] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 177–196, Feb. 2018. DOI: 10.1007/s10514-017-9615-3.

[24] G. Best, J. Faigl, and R. Fitch, "Online planning for multi-robot active perception with self-organising maps," *Autonomous Robots*, vol. 42, no. 4, pp. 715–738, Apr. 2018, ISSN: 1573-7527. DOI: 10.1007/s10514-017-9691-4.

[25] B. Charrow, "Information-theoretic active perception for multi-robot teams," Ph.D. dissertation, University of Pennsylvania, Jan. 2015. [Online]. Available: https://repository.upenn.edu/dissertations/AAI3721239.

[26] D. Holz *et al.*, "Active recognition and manipulation for mobile robot bin picking," in *Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe:*, F. Röhrbein, G. Veiga, and C. Natale, Eds., Cham: Springer International Publishing, 2014, pp. 133–153, ISBN: 978-3-319-03838-4. DOI: 10.1007/978-3-319-03838-4_7.

[27] R. Fernández, C. Salinas, H. Montes, and J. Sarria, "Multisensory system for fruit harvesting robots. experimental testing in natural scenarios and with different kinds of crops," *Sensors*, vol. 14, no. 12, pp. 23 885–23 904, Dec. 2014, ISSN: 1424-8220. DOI: 10.3390/s141223885.

[28] D. Kragic and K. Daniilidis, "3-D vision for navigation and grasping," in *Springer Handbook of Robotics*, Springer International Publishing, Jan. 2016, pp. 811–824, ISBN: 9783319325521. DOI: 10.1007/978-3-319-32552-1_32.

[29] M. R. Cutkosky, R. D. Howe, and W. R. Provancher, "Force and tactile sensors," in *Springer Handbook of Robotics*, Springer Berlin Heidelberg, 2008, pp. 455–476. DOI: 10.1007/978-3-540-30301-5_20.

[30] L. Emmi, M. Gonzalez-De-Soto, G. Pajares, and P. Gonzalez-De-Santos, "New trends in robotics for agriculture: Integration and assessment of a real fleet of robots," *The Scientific World Journal*, vol. 2014, 2014, ISSN: 1537-744X. DOI: 10.1155/2014/404059.

[31] L. Pérez, Í. Rodríguez, N. Rodríguez, R. Usamentiaga, and D. García, "Robot guidance using machine vision techniques in industrial environments: A comparative review," *Sensors*, vol. 16, no. 3, p. 335, Mar. 2016, ISSN: 1424-8220. DOI: 10.3390/s16030335.

[32] M. Ceccarelli, G. Figliolini, E. Ottaviano, A. S. Mata, and E. J. Criado, "Designing a robotic gripper for harvesting horticulture products," *Robotica*, vol. 18, no. 1, pp. 105–111, 2000, ISSN: 0263-5747. DOI: 10.1017/S026357479900226X.

[33] C. Parisses and E. Marinis, "Design of an autonomous robotic vehicle and development of a suitable gripper for harvesting sensitive agricultural products," in *Proceedings of the International Conference on Information and Communication Technologies*, M. Salampasis and A. Matopoulos, Eds., Skiathos, Sep. 2011, pp. 339–349.

[34] S. Fountas, N. Mylonas, I. Malounas, E. Rodias, C. H. Santos, and E. Pekkeriet, "Agricultural robotics for field operations," *Sensors*, vol. 20, no. 9, p. 2672, May 2020. DOI: 10.3390/s20092672.

[35] B. Arad *et al.*, "Development of a sweet pepper harvesting robot," *Journal of Field Robotics*, vol. 37, no. 6, pp. 1027–1039, Sep. 2020, ISSN: 1556-4967. DOI: 10.1002/rob.21937.

[36] B. Zhang, Y. Xie, J. Zhou, K. Wang, and Z. Zhang, "State-of-the-art robotic grippers, grasping and control strategies, as well as their applications in agricultural robots: A review," *Computers and Electronics in Agriculture*, vol. 177, p. 105 694, Oct. 2020, ISSN: 0168-1699. DOI: 10.1016/j.compag.2020.105694.

[37] J. M. Mendes, F. N. dos Santos, N. A. Ferraz, P. M. do Couto, and R. M. dos Santos, "Localization based on natural features detector for steep slope vineyards," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 3-4, pp. 433–446, Mar. 2018, ISSN: 1573-0409. DOI: 10.1007/s10846-017-0770-8.

[38] D. Tiozzo Fasiolo, L. Scalera, E. Maset, and A. Gasparetto, "Towards autonomous mapping in agriculture: A review of supportive technologies for ground robotics," *Robotics and Autonomous Systems*, vol. 169, p. 104 514, Nov. 2023, ISSN: 0921-8890. DOI: 10.1016/j.robot.2023.104514.

[39] INESC TEC. "TRIBE—Laboratory of Robotics and IoT for Smart Precision Agriculture and Forestry," INESC TEC. (2023), [Online]. Available: http://tribe.inesctec.pt (visited on 12/11/2023).

[40] Clearpath Robotics. "Husky—unmanned ground vehicle." (Feb. 20, 2024), [Online]. Available: https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/ (visited on 05/16/2024).

[C41] S. A. Magalhães, A. P. Moreira, F. N. dos Santos, and J. Dias, "Active perception fruit harvesting robots — a systematic review," *Journal of Intelligent & Robotic Systems*, vol. 105, no. 14, May 2022. DOI: 10.1007/s10846-022-01595-3.

[C42] S. A. Magalhães *et al.*, "Evaluating the Single-Shot MultiBox detector and YOLO deep learning models for the detection of tomatoes in a greenhouse," *Sensors*, vol. 21, no. 10, p. 3569, May 2021, ISSN: 1424-8220. DOI: 10.3390/s21103569.

[C43] S. C. Magalhães, F. N. dos Santos, P. Machado, A. P. Moreira, and J. Dias, "Benchmarking edge computing devices for grape bunches and trunks detection using accelerated object detection single shot multibox deep learning models," *Engineering Applications of Artificial Intelligence*, vol. 117, p. 105 604, Jan. 2023. DOI: 10.1016/j.engappai.2022.105604.

[C44] S. C. Magalhães, F. N. dos Santos, A. P. Moreira, and J. Dias, "MonoVisual3DFilter: 3D tomatoes' localisation with monocular cameras using histogram filters," *Robotica*, 2024, Publication in Press.

[C45] S. A. Magalhães, A. P. Moreira, F. N. do Santos, and J. Dias, "BVE + EKF: A viewpoint estimator for the estimation of the object's position in the 3D task space using extended Kalman filters," in *Proceedings of the 21st International Conference on Informatics in Control, Automation and Robotics: ICINCO*, Article submitted, INSTICC, Porto, Portugal: SciTePress, 2024.

[C46] T. C. Padilha, G. Moreira, S. A. Magalhães, F. N. dos Santos, M. Cunha, and M. Oliveira, "Tomato detection using deep learning for robotics application," in *Progress in Artificial Intelligence. EPIA2021*, G. Marreiros, F. S. Melo, N. Lau, H. Lopes Cardoso, and L. P. Reis, Eds., 12981 vols., ser. Lecture Notes in Computer Science, Springer International Publishing, 2021, pp. 27–38. DOI: 10.1007/978-3-030-86230-5_3.

[C47] A. S. Aguiar *et al.*, "Grape bunch detection at different growth stages using deep learning quantized models," *Agronomy*, vol. 11, no. 9, p. 1890, Sep. 2021, ISSN: 2073-4395. DOI: 10.3390/agronomy11091890.

[C48] G. Moreira, S. A. Magalhães, T. Pinho, F. N. dos Santos, and M. Cunha, "Benchmark of deep learning and a proposed HSV colour space models for the detection and classification of greenhouse tomato," *Agronomy*, vol. 12, no. 2, p. 356, Jan. 2022. DOI: 10.3390/agronomy12020356.

[C49] S. Magalhães, F. N. D. Santos, and S. Shyam, *Grape detection using Vitis AI and RetinaNet*, Online, Apr. 2022. [Online]. Available: https://www.hackster.io/452741/grape-detection-using-vitis-ai-and-retinanet-7d0d71.

[C50] S. A. Magalhães, *Dataset of tomato inside greenhouses for object detection in Pascal VOC*, Pascal VOC, INESC TEC research data repository, Dataset, Last accessed on April 27th, 2023, INESC TEC, Jan. 2021. DOI: 10.25747/pc1e-nk92. [Online]. Available: https://rdm.inesctec.pt/dataset/ii-2021-001.

[C51] S. A. Magalhães, G. Moreira, F. N. dos Santos, and M. Cunha, *AgRobTomato dataset: Greenhouse tomatoes with different ripeness stages*, Pascal VOC, Dataset, INESC TEC, 2021. DOI: 10.5281/ZENODO.5596799.

[C52] G. Moreira, S. A. Magalhães, T. Padilha, F. N. dos Santos, and M. Cunha, *RpiTomato dataset: Greenhouse tomatoes with different ripeness stages*, Pascal VOC, Dataset, INESC TEC, Oct. 2021. DOI: 10.5281/ZENODO.5596363.

[C53] A. S. Aguiar and S. Magalhães, *Grape bunch and vine trunk dataset for deep learning object detection*, Pascal VOC, Dataset, INESC TEC, 2021. DOI: 10.5281/ZENODO.5139598.

[C54] M. Almeida, S. C. Magalhães, and F. N. d. Santos, *RG2C: Red grape chunk classification dataset*, 2023. DOI: 10.5281/ZENODO.8124484.

[55] M. J. Baker, "Writing a literature review," *The Marketing Review*, vol. 1, no. 2, pp. 219–247, Nov. 2004, ISSN: 1469-347X. DOI: 10.1362/1469347002529189.

[56] J. Brocke *et al.*, "Reconstructing the giant: On the importance of rigour in documenting the literature search process," *ECIS 2009 Proceedings*, Jan. 2009.

[57] J. Nunn and S. Chang, "What are systematic reviews?" *WikiJournal of Medicine*, vol. 7, no. 1, p. 5, 2020, ISSN: 2002-4436. DOI: 10.15347/WJM/2020.005.

[58] M. Egger and G. D. Smith, "Meta-analysis: Potentials and promise," *BMJ*, vol. 315, no. 7119, pp. 1371–1374, Nov. 1997, ISSN: 1468-5833. DOI: 10.1136/bmj.315.7119.1371.

[59] D. Moher, A. Liberati, J. Tetzlaff, and D. G. Altman, "Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement," *PLoS Medicine*, vol. 6, no. 7, e1000097, Jul. 2009, ISSN: 1549-1676. DOI: 10.1371/journal.pmed.1000097.

[60] R. Bajcsy, "Active perception," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 966–1005, 1988, ISSN: 1558-2256. DOI: 10.1109/5.5968.

[61] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 333–356, Jan. 1988, ISSN: 0920-5691. DOI: 10.1007/BF00133571.

[62] J. Gené-Mola *et al.*, "Fruit detection in an apple orchard using a mobile terrestrial laser scanner," *Biosystems Engineering*, vol. 187, pp. 171–184, Nov. 2019. DOI: 10.1016/j.biosystemseng.2019.08.017.

[63] Y. He, F. Pan, B. Wang, Z. Teng, and J. Wu, "Transfer learning based fruits image segmentation for fruit-picking robots," in *2020 IEEE 3rd International Conference on Computer and Communication Engineering Technology (CCET)*, IEEE, Aug. 2020. DOI: 10.1109/ccet50901.2020.9213127.

[64] H. Cecotti, A. Rivera, M. Farhadloo, and M. A. Pedroza, "Grape detection with convolutional neural networks," *Expert Systems with Applications*, vol. 159, p. 113 588, Nov. 2020. DOI: 10.1016/j.eswa.2020.113588.

[65] H. U. Rehman and J. Miura, "Viewpoint planning for automated fruit harvesting using deep learning," in *2021 IEEE/SICE International Symposium on System Integration (SII)*, IEEE, Jan. 2021. DOI: 10.1109/ieeeconf49454.2021.9382628.

[66] P. Kurtser and Y. Edan, "The use of dynamic sensing strategies to improve detection for a pepper harvesting robot," in *IEEE International Conference on Intelligent Robots and Systems*, Institute of Electrical and Electronics Engineers Inc., Dec. 2018, pp. 8286–8293, ISBN: 9781538680940. DOI: 10.1109/IROS.2018.8593746.

[67] J. Jun, J. Kim, J. Seol, J. Kim, and H. I. Son, "Towards an efficient tomato harvesting robot: 3D perception, manipulation, and end-effector," *IEEE Access*, vol. 9, pp. 17 631–17 640, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3052240.

[68] A. Wendel, J. Underwood, and K. Walsh, "Maturity estimation of mangoes using hyperspectral imaging from a ground based mobile platform," *Computers and Electronics in Agriculture*, vol. 155, pp. 298–313, Dec. 2018. DOI: 10.1016/j.compag.2018.10.021.

[69] M. Zhao, Y. Peng, L. Li, and X. Qiao, "Detection and classification manipulator system for apple based on machine vision and optical technology," in *2020 ASABE Annual International Virtual Meeting, July 13-15, 2020*, American Society of Agricultural and Biological Engineers, 2020. DOI: 10.13031/aim.202000498.

[70] D. K. Barbole, P. M. Jadhav, and S. B. Patil, "A review on fruit detection and segmentation techniques in agricultural field," in *Second International Conference on Image Processing and Capsule Networks*, J. I.-Z. Chen, J. M. R. S. Tavares, A. M. Iliyasu, and K.-L. Du, Eds., Cham: Springer International Publishing, 2022, pp. 269–288, ISBN: 978-3-030-84760-9.

[71] L. Fu, F. Gao, J. Wu, R. Li, M. Karkee, and Q. Zhang, "Application of consumer RGB-D cameras for fruit detection and localization in field: A critical review," *Computers and Electronics in Agriculture*, vol. 177, p. 105 687, 2020, ISSN: 0168-1699. DOI: 10.1016/j.compag.2020.105687.

[72] J. Naranjo-Torres, M. Mora, R. Hernández-García, R. J. Barrientos, C. Fredes, and A. Valenzuela, "A review of convolutional neural network applied to fruit image processing," *Applied Sciences*, vol. 10, no. 10, p. 3443, May 2020, ISSN: 2076-3417. DOI: 10.3390/app10103443.

[73] D. H. Ballard, "Animate vision," *Artificial Intelligence*, vol. 48, no. 1, pp. 57–86, Feb. 1991, ISSN: 0004-3702. DOI: 10.1016/0004-3702(91)90080-4.

[74] E. Rivlin and H. Rotstein, "Control of a camera for active vision: Foveal vision, smooth tracking and saccade," *International Journal of Computer Vision*, vol. 39, no. 2, pp. 81–96, 2000. DOI: 10.1023/a:1008166825510.

[75] D. Ognibene and G. Baldassare, "Ecological active vision: Four bioinspired principles to integrate bottom–up and adaptive top–down attention tested with a simple camera-arm robot," *IEEE Transactions on Autonomous Mental Development*, vol. 7, no. 1, pp. 3–25, Mar. 2015. DOI: 10.1109/tamd.2014.2341351.

[76] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *International Journal of Robotics Research*, vol. 30, no. 11, pp. 1343–1377, Sep. 2011, ISSN: 0278-3649. DOI: 10.1177/0278364911410755.

[77] M. Gualtieri, A. T. Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, Institute of Electrical and Electronics Engineers Inc., Nov. 2016, pp. 598–605, ISBN: 9781509037629. DOI: 10.1109/IROS.2016.7759114. arXiv: 1603.01564.

[78] C. Balkenius and N. Hulth, "Attention as selection-for-action: A scheme for active perception," in *1999 3rd European Workshop on Advanced Mobile Robots, Eurobot 1999 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 1999, pp. 113–119, ISBN: 0780356721. DOI: 10.1109/EURBOT.1999.827629.

[79] T. J. Prescott, M. E. Diamond, and A. M. Wing, "Active touch sensing," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 366, no. 1581, pp. 2989–2995, 2011, ISSN: 1471-2970. DOI: 10.1098/rstb.2011.0167.

[80] S. A. Magalhães, F. N. dos Santos, R. C. Martins, L. F. Rocha, and J. Brito, "Path planning algorithms benchmarking for grapevines pruning and monitoring," in *Lecture Notes in Computer Science*, Springer International Publishing, 2019, pp. 295–306, ISBN: 9783030302443. DOI: 10.1007/978-3-030-30244-3_25.

[81] S. Paulin, T. Botterill, J. Lin, X. Chen, and R. Green, "A comparison of sampling-based path planners for a grape vine pruning robot arm," in *ICARA 2015 - Proceedings of the 2015 6th International Conference on Automation, Robotics and Applications*, Institute of Electrical and Electronics Engineers Inc., Apr. 2015, pp. 98–103, ISBN: 9781479964666. DOI: 10.1109/ICARA.2015.7081131.

[82] P. Kurtser and Y. Edan, "Planning the sequence of tasks for harvesting robots," *Robotics and Autonomous Systems*, vol. 131, p. 103 591, Sep. 2020, ISSN: 0921-8890. DOI: 10.1016/j.robot.2020.103591.

[83] W. Xu *et al.*, "Shadow detection and removal in apple image segmentation under natural light conditions using an ultrametric contour map," *Biosystems Engineering*, vol. 184, pp. 142–154, Aug. 2019. DOI: 10.1016/j.biosystemseng.2019.06.016.

[84] H. Liu, Y. Yu, F. Sun, and J. Gu, "Visual–tactile fusion for object recognition," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 996–1008, Apr. 2017, ISSN: 1545-5955. DOI: 10.1109/TASE.2016.2549552.

[85] V. F. Tejada, M. F. Stoelen, K. Kusnierek, N. Heiberg, and A. Korsaeth, "Proof-of-concept robot platform for exploring automated harvesting of sugar snap peas," *Precision Agriculture*, vol. 18, no. 6, pp. 952–972, Aug. 2017. DOI: 10.1007/s11119-017-9538-1.

[86] K. Nilay *et al.*, "A proposal of FPGA-based low cost and power efficient autonomous fruit harvester," in *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, IEEE, Apr. 2020. DOI: 10.1109/iccar49639.2020.9108079.

[87] Y. Xiong, C. Peng, L. Grimstad, P. J. From, and V. Isler, "Development and field evaluation of a strawberry harvesting robot with a cable-driven gripper," *Computers and Electronics in Agriculture*, vol. 157, pp. 392–402, Feb. 2019. DOI: 10.1016/j.compag.2019.01.009.

[88] Y. Xiong, Y. Ge, and P. J. From, "Push and drag: An active obstacle separation method for fruit harvesting robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2020. DOI: 10.1109/icra40945.2020.9197469.

[89] I. Pérez-Borrero, D. Marín-Santos, M. E. Gegúndez-Arias, and E. Cortés-Ancos, "A fast and accurate deep learning method for strawberry instance segmentation," *Computers and Electronics in Agriculture*, vol. 178, p. 105 736, Nov. 2020. DOI: 10.1016/j.compag.2020.105736.

[90] R. C. Martins, S. Magalhães, P. Jorge, T. Barroso, and F. Santos, "Metbots: Metabolomics robots for precision viticulture," in *Progress in Artificial Intelligence*, Springer International Publishing, 2019, pp. 156–166. DOI: 10.1007/978-3-030-30241-2_14.

[91] B. Rasolzadeh, M. Björkman, K. Huebner, and D. Kragic, "An active vision system for detecting, fixating and manipulating objects in the real world," *The International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 133–154, Feb. 2010, ISSN: 0278-3649. DOI: 10.1177/0278364909346069.

[92] X. Xue *et al.*, "Detection of young green apples in orchard environment using adaptive ratio chromatic aberration and HOG-SVM," in *Computer and Computing Technologies in Agriculture XI*, Springer International Publishing, 2019, pp. 253–268. DOI: 10.1007/978-3-030-06137-1_24.

[93] C. Warren, "Global path planning using artificial potential fields," in *1989 IEEE International Conference on Robotics and Automation*, Los Alamitos, CA, USA: IEEE Computer Society, May 1989, pp. 316, 317, 318, 319, 320, 321. DOI: 10.1109/ROBOT.1989.100007.

[94] E. Badeka *et al.*, "Grapes visual segmentation for harvesting robots using local texture descriptors," in *Lecture Notes in Computer Science*, Springer International Publishing, 2019, pp. 98–109. DOI: 10.1007/978-3-030-34995-0_9.

[95] P. Fan *et al.*, "A method of segmenting apples based on gray-centered RGB color space," *Remote Sensing*, vol. 13, no. 6, p. 1211, Mar. 2021. DOI: 10.3390/rs13061211.

[96] Y. Zhao, L. Gong, Y. Huang, and C. Liu, "Robust tomato recognition for robotic harvesting using feature images fusion," *Sensors*, vol. 16, no. 2, p. 173, Jan. 2016, ISSN: 1424-8220. DOI: 10.3390/s16020173.

[97] J. Li, Y. Tang, X. Zou, G. Lin, and H. Wang, "Detection of fruit-bearing branches and localization of litchi clusters for vision-based harvesting robots," *IEEE Access*, vol. 8, pp. 117 746–117 758, 2020. DOI: 10.1109/access.2020.3005386.

[98] S. Mao, Y. Li, Y. Ma, B. Zhang, J. Zhou, and K. Wang, "Automatic cucumber recognition algorithm for harvesting robots in the natural environment using deep learning and multi-feature fusion," *Computers and Electronics in Agriculture*, vol. 170, p. 105 254, Mar. 2020. DOI: 10.1016/j.compag.2020.105254.

[99] K. N. E Alam Siddiquee, M. S. Islam, M. Y. U. Dowla, K. M. Rezaul, and V. Grout, "Detection, quantification and classification of ripened tomatoes: A comparative analysis of image processing and machine learning," *IET Image Processing*, vol. 14, no. 11, pp. 2442–2456, Sep. 2020, ISSN: 1751-9659. DOI: 10.1049/iet-ipr.2019.0738.

[100] H. Yin, Y. Chai, S. X. Yang, and G. S. Mittal, "Ripe tomato recognition and localization for a tomato harvesting robotic system," in *2009 International Conference of Soft Computing and Pattern Recognition*, Malacca, Malaysia: IEEE, 2009, pp. 557–562, ISBN: 9780769538792. DOI: 10.1109/socpar.2009.111.

[101] L. Huang, S. X. Yang, and D. He, "Abscission point extraction for ripe tomato harvesting robots," *Intelligent Automation & Soft Computing*, vol. 18, no. 6, pp. 751–763, Jan. 2012, ISSN: 1079-8587. DOI: 10.1080/10798587.2012.10643285.

[102] A. Arefi, A. Motlagh, K. Mollazade, and R. Teimourlou, "Recognition and localization of ripen tomato based on machine vision," *Australian Journal of Crop Science*, vol. 5, no. 10, pp. 1144–1149, Sep. 2011, ISSN: 1835-2693.

[103] Q. Feng, X. Wang, G. Wang, and Z. Li, "Design and test of tomatoes harvesting robot," in *2015 IEEE International Conference on Information and Automation*, IEEE, Aug. 2015, pp. 949–952, ISBN: 9781467391047. DOI: 10.1109/icinfa.2015.7279423.

[104] F. Zhang, "Ripe tomato recognition with computer vision," in *Proceedings of the 2015 International Industrial Informatics and Computer Engineering Conference*, Paris, France: Atlantis Press, Mar. 2015, pp. 466–469, ISBN: 978-94-62520-54-7. DOI: 10.2991/iiicec-15.2015.107.

[105] M. Benavides, M. Cantón-Garbín, J. A. Sánchez-Molina, and F. Rodríguez, "Automatic tomato and peduncle location system based on computer vision for use in robotized harvesting," *Applied Sciences*, vol. 10, no. 17, p. 5887, Aug. 2020, ISSN: 2076-3417. DOI: 10.3390/app10175887.

[106] M. H. Malik, T. Zhang, H. Li, M. Zhang, S. Shabbir, and A. Saeed, "Mature tomato fruit detection algorithm based on improved HSV and watershed algorithm," *IFAC-PapersOnLine*, vol. 51, no. 17, pp. 431–436, 2018, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2018.08.183.

[107] E. R. Dougherty, "Optimal binary morphological bandpass filters induced by granulometric spectral representation," *Journal of Mathematical Imaging and Vision*, vol. 7, no. 2, pp. 175–192, 1997, ISSN: 0924-9907. DOI: 10.1023/A:1008209706862.

[108] A. Zhu, L. Yang, and Y. Chen, "An FCM-based method to recognize and extract ripe tomato for harvesting robotic system," in *IEEE International Conference on Automation and Logistics, ICAL*, 2012, pp. 533–538, ISBN: 9781467303620. DOI: 10.1109/ICAL.2012.6308135.

[109] R. Xiang, Y. Ying, and H. Jiang, "Tests of a recognition algorithm for clustered tomatoes based on mathematical morphology," in *Proceedings of the 2013 6th International Congress on Image and Signal Processing, CISP 2013*, vol. 1, 2013, pp. 464–468, ISBN: 9781479927647. DOI: 10.1109/CISP.2013.6744040.

[110] K. Yamamoto, W. Guo, Y. Yoshioka, and S. Ninomiya, "On plant detection of intact tomato fruits using image analysis and machine learning methods," *Sensors*, vol. 14, no. 7, pp. 12 191–12 206, Jul. 2014, ISSN: 1424-8220. DOI: 10.3390/s140712191.

[111] G. Liu, S. Mao, and J. H. Kim, "A mature-tomato detection algorithm using machine learning and color analysis," *Sensors*, vol. 19, no. 9, p. 2023, Apr. 2019, ISSN: 1424-8220. DOI: 10.3390/s19092023.

[112] G. Wu, Q. Zhu, M. Huang, Y. Guo, and J. Qin, "Automatic recognition of juicy peaches on trees based on 3D contour features and colour data," *Biosystems Engineering*, vol. 188, pp. 1–13, Dec. 2019. DOI: 10.1016/j.biosystemseng.2019.10.002.

[113] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986, ISSN: 0162-8828. DOI: 10.1109/tpami.1986.4767851.

[114] S. Gupta, S. G. Mazumdar, and M. T. Student, "Sobel edge detection algorithm," *International Journal of Computer Science and Management Research*, vol. 2, pp. 1578–1583, 2013.

[115] Y. Zhao, L. Gong, B. Zhou, Y. Huang, and C. Liu, "Detecting tomatoes in greenhouse scenes by combining AdaBoost classifier and colour analysis," *Biosystems Engineering*, vol. 148, pp. 127–137, Aug. 2016, ISSN: 1537-5110. DOI: 10.1016/j.biosystemseng.2016.05.001.

[116] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans Syst Man Cybern*, vol. SMC-9, no. 1, pp. 62–66, 1979, ISSN: 0018-9472. DOI: 10.1109/tsmc.1979.4310076.

[117] Z.-F. Xu, R.-S. Jia, Y.-B. Liu, C.-Y. Zhao, and H.-M. Sun, "Fast Method of Detecting Tomatoes in a Complex Scene for Picking Robots," *IEEE Access*, vol. 8, pp. 55 289–55 299, 2020, ISSN: 2169-3536. DOI: 10.1109/access.2020.2981823.

[118] J. Liu, Y. Peng, and M. Faheem, "Experimental and theoretical analysis of fruit plucking patterns for robotic tomato harvesting," *Computers and Electronics in Agriculture*, vol. 173, Jun. 2020, ISSN: 0168-1699. DOI: 10.1016/j.compag.2020.105330.

[119] J. Sun, X. He, M. Wu, X. Wu, J. Shen, and B. Lu, "Detection of tomato organs based on convolutional neural network under the overlap and occlusion backgrounds," *Machine Vision and Applications*, vol. 31, no. 5, pp. 1–13, Jul. 2020, ISSN: 1432-1769. DOI: 10.1007/s00138-020-01081-6.

[120] Y. Mu, T.-S. Chen, S. Ninomiya, and W. Guo, "Intact detection of highly occluded immature tomatoes on plants using deep learning techniques," *Sensors*, vol. 20, no. 10, p. 2984, May 2020, ISSN: 1424-8220. DOI: 10.3390/s20102984.

[121] R. G. de Luna, E. P. Dadios, A. A. Bandala, and R. R. P. Vicerra, "Tomato growth stage monitoring for smart farm using deep transfer learning with machine learning-based maturity grading," *AGRIVITA Journal of Agricultural Science*, vol. 42, no. 1, pp. 24–36, 2020, ISSN: 2477-8516. DOI: 10.17503/agrivita.v42i1.2499.

[122] T. Yuan *et al.*, "Robust cherry tomatoes detection algorithm in greenhouse scene based on SSD," *Agriculture*, vol. 10, no. 5, p. 160, May 2020, ISSN: 2077-0472. DOI: 10.3390/agriculture10050160.

[123] J. Redmon. "Darknet: Open source neural networks in C." (2016), [Online]. Available: http://pjreddie.com/darknet/.

[124] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," Apr. 8, 2018. DOI: 10.48550/ARXIV.1804.02767. arXiv: 1804.02767 [cs.CV].

[125] X. Liu, D. Zhao, W. Jia, C. Ruan, S. Tang, and T. Shen, "A method of segmenting apples at night based on color and position information," *Computers and Electronics in Agriculture*, vol. 122, pp. 118–123, Mar. 2016. DOI: 10.1016/j.compag.2016.01.023.

[126] J. K. TsoTsos, "A framework for visual motion understanding," PhD Dissertation, University of Toronto, 1980.

[127] Y. Xiong, Y. Ge, and P. J. From, "An obstacle separation method for robotic picking of fruits in clusters," *Computers and Electronics in Agriculture*, vol. 175, p. 105 397, Aug. 2020, ISSN: 0168-1699. DOI: 10.1016/j.compag.2020.105397.

[128] K.-H. Choi, G.-H. Lee, Y. J. Han, and J. M. Bunn, "Tomato maturity evaluation using color image analysis," *Transactions of the ASAE*, vol. 38, no. 1, pp. 171–176, 1995. DOI: 10.13031/2013.27827.

[129] C. Li, Q. Cao, and F. Guo, "A method for color classification of fruits based on machine vision," *WSEAS TRANSACTIONS on SYSTEMS*, vol. 8, no. 2, pp. 312–321, Feb. 2009.

[130] O. R. Indriani, E. J. Kusuma, C. A. Sari, E. H. Rachmawanto, and D. R. I. M. Setiadi, "Tomatoes classification using K-NN based on GLCM and HSV color space," in *2017 International Conference on Innovative and Creative Information Technology (ICITech)*, Salatiga, Indonesia: IEEE, Nov. 2017, pp. 1–6. DOI: 10.1109/innocit.2017.8319133.

[131] N. Goel and P. Sehgal, "Fuzzy classification of pre-harvest tomatoes for ripeness estimation — An approach based on automatic rule learning using decision tree," *Applied Soft Computing*, vol. 36, pp. 45–56, Nov. 2015. DOI: 10.1016/j.asoc.2015.07.009.

[132] S. R. Rupanagudi, B. S. Ranjani, P. Nagaraj, and V. G. Bhat, "A cost effective tomato maturity grading system using image processing for farmers," in *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, Mysore, India: IEEE, Nov. 2014. DOI: 10.1109/ic3i.2014.7019591.

[133] Y. A. Sari, S. Adinugroho, P. P. Adikara, and A. Izzah, "Multiplication of V and Cb color channel using Otsu thresholding for tomato maturity clustering," in *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, Malang, Indonesia: IEEE, Nov. 2017, pp. 209–214. DOI: 10.1109/siet.2017.8304136.

[134] J. Chen *et al.*, "An improved YOLOv3 based on dual path network for cherry tomatoes detection," *Journal of Food Process Engineering*, vol. 44, no. 10, Jul. 2021. DOI: 10.1111/jfpe.13803.

[135] W. Zhang, K. Chen, J. Wang, Y. Shi, and W. Guo, "Easy domain adaptation method for filling the species gap in deep learning-based fruit detection," *Horticulture Research*, vol. 8, no. 1, Jun. 2021. DOI: 10.1038/s41438-021-00553-8.

[136] G. Liu, J. C. Nouaze, P. L. T. Mbouembe, and J. H. Kim, "YOLO-Tomato: A robust algorithm for tomato detection based on YOLOv3," *Sensors*, vol. 20, no. 7, p. 2145, Apr. 2020, ISSN: 1424-8220. DOI: 10.3390/s20072145.

[137] S. Ruparelia, M. Jethva, and R. Gajjar, "Real-time tomato detection, classification, and counting system using deep learning and embedded systems," in *Advances in Intelligent Systems and Computing*, Springer Singapore, Aug. 2021, pp. 511–522, ISBN: 978-981-16-2122-2. DOI: 10.1007/978-981-16-2123-9_39.

[138] M. O. Lawal, "Tomato detection based on modified YOLOv3 framework," *Scientific Reports*, vol. 11, no. 1, Jan. 2021. DOI: 10.1038/s41598-021-81216-5.

[139] V. Tsironis, S. Bourou, and C. Stentoumis, "TOMATOD: Evaluation of object detection algorithms on a new real-world tomato dataset," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIII-B3-2020, pp. 1077–1084, Aug. 2020. DOI: 10.5194/isprs-archives-xliii-b3-2020-1077-2020.

[140] S. A. Mutha, A. M. Shah, and M. Z. Ahmed, "Maturity detection of tomatoes using deep learning," *SN Computer Science*, vol. 2, no. 6, Sep. 2021. DOI: 10.1007/s42979-021-00837-9.

[141] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[142] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. arXiv: 1409.1556.

[143] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016. DOI: 10.1109/cvpr.2016.90.

[144] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA: IEEE, Jun. 2018. DOI: 10.1109/cvpr.2018.00474.

[145] W. Liu *et al.*, "SSD: Single shot MultiBox detector," in *Computer Vision — ECCV 2016*, ser. Lecture Notes in Computer Science, vol. 9905, 9905 vols., Springer International Publishing, 2016, pp. 21–37, ISBN: 9783319464480. DOI: 10.1007/978-3-319-46448-0_2.

[146] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA: IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.351.

[147] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.

[148] L. Luo, Y. Tang, Q. Lu, X. Chen, P. Zhang, and X. Zou, "A vision methodology for harvesting robot to detect cutting points on peduncles of double overlapping grape clusters in a vineyard," *Computers in Industry*, vol. 99, pp. 130–139, Aug. 2018, ISSN: 0166-3615. DOI: 10.1016/j.compind.2018.03.017.

[149] I. Sa *et al.*, "Peduncle detection of sweet pepper for autonomous crop harvesting-combined color and 3-D information," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 765–772, Apr. 2017, ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2651952. arXiv: 1701.08608.

[150] T. Yoshida, T. Fukao, and T. Hasegawa, "Cutting point detection using a robot with point clouds for tomato harvesting," *Journal of Robotics and Mechatronics*, vol. 32, no. 2, pp. 437–444, Apr. 2020, ISSN: 1883-8049. DOI: 10.20965/jrm.2020.p0437.

[151] C. Lehnert, C. McCool, I. Sa, and T. Perez, "Performance improvements of a sweet pepper harvesting robot in protected cropping environments," *Journal of Field Robotics*, vol. 37, no. 7, pp. 1197–1223, Oct. 2020, ISSN: 1556-4967. DOI: 10.1002/rob.21973.

[152] O. M. Ogorodnikova and W. Ali, "Method of ripe tomato detecting for a harvesting robot," in *PHYSICS, TECHNOLOGIES AND INNOVATION (PTI-2019): Proceedings of the VI International Young Researchers' Conference*, AIP Publishing, 2019. DOI: 10.1063/1.5134297.

[153] S. Kaur, S. Randhawa, and A. Malhi, "An efficient ANFIS based pre-harvest ripeness estimation technique for fruits," *Multimedia Tools and Applications*, vol. 80, no. 13, pp. 19 459–19 489, Feb. 2021. DOI: 10.1007/s11042-021-10741-2.

[154] B. Harel, R. van Essen, Y. Parmet, and Y. Edan, "Viewpoint analysis for maturity classification of sweet peppers," *Sensors*, vol. 20, no. 13, pp. 1–22, Jul. 2020, ISSN: 1424-8220. DOI: 10.3390/s20133783.

[155] M. S. Kumar and S. Mohan, "Selective fruit harvesting: Research, trends and developments towards fruit detection and localization – a review," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 237, no. 6, pp. 1405–1444, Oct. 2022. DOI: 10.1177/09544062221128443.

[156] O. Ringdahl, P. Kurtser, and Y. Edan, "Performance of RGB-D camera for different object types in greenhouse conditions," in *2019 European Conference on Mobile Robots (ECMR)*, Prague, Czech Republic: IEEE, Sep. 2019, pp. 1–6. DOI: 10.1109/ECMR.2019.8870935.

[157] J. Gené-Mola *et al.*, "Assessing the performance of RGB-D sensors for 3D fruit crop canopy characterization under different operating and lighting conditions," *Sensors*, vol. 20, no. 24, p. 7072, Dec. 2020, ISSN: 1424-8220. DOI: 10.3390/s20247072.

[158] Q. M. ul Haq, M. A. Haq, S.-J. Ruan, P.-J. Liang, and D.-Q. Gao, "3D object detection based on proposal generation network utilizing monocular images," *IEEE Consumer Electronics Magazine*, vol. 11, no. 5, pp. 47–53, Sep. 2022. DOI: 10.1109/mce.2021.3059565.

[159] X. Ma, S. Liu, Z. Xia, H. Zhang, X. Zeng, and W. Ouyang, "Rethinking pseudo-LiDAR representation," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J. +Frahm, Eds., ser. Lecture Notes in Computer Science, vol. 12358, Online: Springer International Publishing, Aug. 2020, pp. 311–327. DOI: 10.1007/978-3-030-58601-0_19.

[160] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan, "Accurate monocular 3D object detection via color-embedded 3D reconstruction for autonomous driving," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019, pp. 6850–6859. DOI: 10.1109/iccv.2019.00695.

[161] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3D bounding box estimation using deep learning and geometry," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.597.

[162] X. H. Van and N. Do, "An efficient regression method for 3D object localization in machine vision systems," *IAES International Journal of Robotics and Automation (IJRA)*, vol. 11, no. 2, pp. 111–121, Jun. 2022. DOI: 10.11591/ijra.v11i2.pp111-121.

[163] G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna, "CalibNet: Geometrically supervised extrinsic calibration using 3D spatial transformer networks," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain: IEEE, Oct. 2018. DOI: 10.1109/iros.2018.8593693.

[164] X. Fu, Y. Liu, and Z. Wang, "Active Learning-Based Grasp for Accurate Industrial Manipulation," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1610–1618, Oct. 2019, ISSN: 1558-3783. DOI: 10.1109/TASE.2019.2897791.

[165] G. von Wichert *et al.*, "The robotic bar – an integrated demonstration of man-robot interaction in a service scenario," in *Proceedings. 11th IEEE International Workshop on Robot and Human Interactive Communication*, IEEE, 2002, pp. 374–379, ISBN: 0-7803-7545-9. DOI: 10.1109/ROMAN.2002.1045651.

[166] B. Calli, W. Caarls, M. Wisse, and P. P. Jonker, "Active vision via extremum seeking for robots in unstructured environments: Applications in object recognition and manipulation," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 4, pp. 1810–1822, Oct. 2018, ISSN: 1545-5955. DOI: 10.1109/TASE.2018.2807787.

[167] R. Barth, J. Hemming, and E. J. van Henten, "Design of an eye-in-hand sensing and servo control framework for harvesting robotics in dense vegetation," *Biosystems Engineering*, vol. 146, pp. 71–84, Jun. 2016, ISSN: 1537-5110. DOI: 10.1016/j.biosystemseng.2015.12.001.

[168] S. S. Mehta, C. Ton, M. Rysz, P. Ganesh, Z. Kan, and T. F. Burks, "On achieving bounded harvest times in robotic fruit harvesting: A finite-time visual servo control approach," *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 114–119, 2019. DOI: 10.1016/j.ifacol.2019.12.507.

[169] S. S. Mehta, W. MacKunis, and T. F. Burks, "Robust visual servo control in the presence of fruit motion for robotic citrus harvesting," *Computers and Electronics in Agriculture*, vol. 123, pp. 362–375, Apr. 2016. DOI: 10.1016/j.compag.2016.03.007.

[170] P. Zapotezny-Anderson and C. Lehnert, "Towards active robotic vision in agriculture: A deep learning approach to visual servoing in occluded and unstructured protected cropping environments," *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 120–125, 2019. DOI: 10.1016/j.ifacol.2019.12.508.

[171] N. Hani and V. Isler, "Visual servoing in orchard settings," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2016. DOI: 10.1109/iros.2016.7759456.

[172] A. Ostovar, O. Ringdahl, and T. Hellström, "Adaptive image thresholding of yellow peppers for a harvesting robot," *Robotics*, vol. 7, no. 1, p. 11, Feb. 2018. DOI: 10.3390/robotics7010011.

[173] B. Lee, D. Kam, B. Min, J. Hwa, and S. Oh, "A vision servo system for automated harvest of sweet pepper in korean greenhouse environment," *Applied Sciences*, vol. 9, no. 12, p. 2395, Jun. 2019. DOI: 10.3390/app9122395.

[174] G. Wu, B. Li, Q. Zhu, M. Huang, and Y. Guo, "Using color and 3D geometry features to segment fruit point cloud and improve fruit recognition accuracy," *Computers and Electronics in Agriculture*, vol. 174, p. 105 475, Jul. 2020, ISSN: 0168-1699. DOI: 10.1016/j.compag.2020.105475.

[175] P. Ramon Soria, F. Sukkar, W. Martens, B. C. Arrue, and R. Fitch, "Multi-view probabilistic segmentation of pome fruit with a low-cost RGB-D camera," in *Advances in Intelligent Systems and Computing*, vol. 694, Springer Verlag, Nov. 2018, pp. 320–331, ISBN: 9783319708355. DOI: 10.1007/978-3-319-70836-2_27.

[176] P. Kurtser and Y. Edan, "Statistical models for fruit detectability: Spatial and temporal analyses of sweet peppers," *Biosystems Engineering*, vol. 171, pp. 272–289, Jul. 2018. DOI: 10.1016/j.biosystemseng.2018.04.017.

[177] H. Sarabu, K. Ahlin, and A.-P. Hu, "Leveraging deep learning and RGB-D cameras for cooperative apple-picking robot arms," in *2019 Boston, Massachusetts July 7-July 10, 2019*, American Society of Agricultural and Biological Engineers, 2019. DOI: 10.13031/aim.201901125.

[178] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics* (Intelligent Robotics and Autonomous Agents series). MIT Press, Aug. 2005, ISBN: 0262201623.

[179] N. Boyko and Y. Hladun, "Histogram filter for robot localization," in *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, LVIV, Ukraine: IEEE, Sep. 2021, pp. 38–43. DOI: 10.1109/csit52700.2021.9648585.

[180] A. Moscowsky, "Subdefinite computations for reducing the search space in mobile robot localization task," in *Artificial Intelligence. RCAI 2021*, ser. Lecture Notes in Computer Science, vol. 12948, Springer International Publishing, 2021, pp. 180–196. DOI: 10.1007/978-3-030-86855-0_13.

[181] J. Sarmento, F. N. D. Santos, A. S. Aguiar, H. Sobreira, C. V. Regueiro, and A. Valente, "FollowMe—a pedestrian following algorithm for agricultural logistic robots," in *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Santa Maria da Feira, Portugal: IEEE, Apr. 2022, pp. 179–185. DOI: 10.1109/icarsc55462.2022.9784791.

[182] S. Engin and V. Isler, "Active Localization of Multiple Targets from Noisy Relative Measurements," in *Algorithmic Foundations of Robotics XIV. WAFR 2020*, S. M. LaValle, M. Lin, T. Ojala, D. Shell, and J. Yu, Eds., ser. Springer Proceedings in Advanced Robotics, vol. 17, Springer International Publishing, 2021, pp. 398–413. DOI: 10.1007/978-3-030-66723-8_24.

[183] Z. C. Márton *et al.*, "Improving object orientation estimates by considering multiple viewpoints," *Autonomous Robots*, vol. 42, no. 2, pp. 423–442, Apr. 2017. DOI: 10.1007/s10514-017-9633-1.

[184] C. Lehnert, D. Tsai, A. Eriksson, and C. McCool, "3D Move to See: Multi-perspective visual servoing for improving object views with semantic segmentation," *arXiv e-prints*, Sep. 2018. DOI: 10.48550/ARXIV.1809.07896. arXiv: 1809.07896 [cs.RO].

[185] H. Kang, H. Zhou, and C. Chen, "Visual perception and modeling for autonomous apple harvesting," *IEEE Access*, vol. 8, pp. 62 151–62 163, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2984556.

[186] H. Kang, H. Zhou, X. Wang, and C. Chen, "Real-time fruit recognition and grasping estimation for robotic apple harvesting," *Sensors*, vol. 20, no. 19, pp. 1–15, Oct. 2020, ISSN: 1424-8220. DOI: 10.3390/s20195670. arXiv: 2003.13298.

[187] H. S. de Andrade, "Software concerns for execution on heterogeneous platforms," Ph.D. dissertation, Chalmers University of Technology and University of Gothenburg, 2018.

[188] Intel. "What is a GPU? graphics processing units defined." (Oct. 16, 2023), [Online]. Available: https://www.intel.co.uk/content/www/uk/en/products/docs/processors/what-is-a-gpu.html (visited on 05/16/2024).

[189] C. Wang and Z. Peng, "Design and implementation of an object detection system using Faster R-CNN," in *2019 International Conference on Robots & Intelligent System (ICRIS)*, Haikou, China: IEEE, Jun. 2019. DOI: 10.1109/icris.2019.00060.

[190] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016. DOI: 10.1109/cvpr.2016.91.

[191] M. Sozzi, S. Cantalamessa, A. Cogato, A. Kayad, and F. Marinello, "Automatic bunch detection in white grape varieties using YOLOv3, YOLOv4, and YOLOv5 deep learning algorithms," *Agronomy*, vol. 12, no. 2, p. 319, Jan. 2022. DOI: 10.3390/agronomy12020319.

[192] H. Zhao *et al.*, "Autonomous live working robot navigation with real-time detection and motion planning system on distribution line," *High Voltage*, vol. 7, no. 6, pp. 1204–1216, Jun. 2022. DOI: 10.1049/hve2.12221.

[193] A. G. Olenskyj *et al.*, "End-to-end deep learning for directly estimating grape yield from ground-based imagery," *Computers and Electronics in Agriculture*, vol. 198, p. 107 081, Jul. 2022. DOI: 10.1016/j.compag.2022.107081.

[194] F. Terra, L. Rodrigues, S. Magalhães, F. Santos, P. Moura, and M. Cunha, "PixelCropRobot, a cartesian multitask platform for microfarms automation," in *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*, IEEE, Sep. 2021, pp. 382–387. DOI: 10.1109/iria53009.2021.9588786.

[195] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South): IEEE, Oct. 2019. DOI: 10.1109/iccv.2019.00925.

[196] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," Apr. 17, 2017. DOI: 10.48550/ARXIV.1704.04861. arXiv: 1704.04861 [cs.CV].

[197] P. Puchtler and R. Peinl, "Evaluation of Deep Learning Accelerators for Object Detection at the Edge," in *Lecture Notes in Computer Science*, Springer International Publishing, 2020, pp. 320–326. DOI: 10.1007/978-3-030-58285-2_29.

[198] J. Yang *et al.*, "Quantization networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, Jun. 2019. DOI: 10.1109/cvpr.2019.00748.

[199] H. Zhao, W. Zhang, H. Sun, and B. Xue, "Embedded deep learning for ship detection and recognition," *Future Internet*, vol. 11, no. 2, p. 53, Feb. 2019. DOI: 10.3390/fi11020053.

[200] A. A. Suzen, B. Duman, and B. Sen, "Benchmark analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN," in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Ankara, Turkey: IEEE, Jun. 2020. DOI: 10.1109/hora49412.2020.9152915.

[201] Y.-C. Chiu, C.-Y. Tsai, M.-D. Ruan, G.-Y. Shen, and T.-T. Lee, "MobileNet-SSDv2: An improved object detection model for embedded systems," in *2020 International Conference on System Science and Engineering (ICSSE)*, Kagawa, Japan: IEEE, Aug. 2020. DOI: 10.1109/icsse50014.2020.9219319.

[202] W. Rahmaniar and A. Hernawan, "Real-time human detection using deep learning on embedded platforms: A review," *Journal of Robotics and Control (JRC)*, vol. 2, no. 6, 2021. DOI: 10.18196/jrc.26123.

[203] R. P. Martinez, I. Schiopu, B. Cornelis, and A. Munteanu, "Real-time instance segmentation of traffic videos for embedded devices," *Sensors*, vol. 21, no. 1, p. 275, Jan. 2021. DOI: 10.3390/s21010275.

[204] S. I. Venieris and C.-S. Bouganis, "fpgaConvNet: A toolflow for mapping diverse convolutional neural networks on embedded FPGAs," Nov. 23, 2017. DOI: 10.48550/ARXIV.1711.08740. arXiv: 1711.08740 [cs.CV].

[205] Y. Chen *et al.*, "T-DLA: An open-source deep learning accelerator for ternarized DNN models on embedded FPGA," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Miami, FL, USA: IEEE, Jul. 2019. DOI: 10.1109/isvlsi.2019.00012.

[206] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Apr. 2015. DOI: 10.1007/s11263-015-0816-y.

[207] G.-Z. Lin, H. M. Nguyen, C.-C. Sun, P.-Y. Kuo, and M.-H. Sheu, "A novel bird detection and identification based on DPU processor on PYNQ FPGA," in *2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, Penghu, Taiwan: IEEE, Sep. 2021. DOI: 10.1109/icce-tw52618.2021.9603066.

[208] X. Zhao, X. Zhang, F. Yang, P. Xu, W. Li, and F. Chen, "Research on machine learning optimization algorithm of CNN for FPGA architecture," *Journal of Physics: Conference Series*, vol. 2006, no. 1, p. 012 012, Aug. 2021. DOI: 10.1088/1742-6596/2006/1/012012.

[209] V. Jain, N. Jadhav, and M. Verhelst, "Enabling real-time object detection on low cost FPGAs," *Journal of Real-Time Image Processing*, vol. 19, no. 1, pp. 217–229, Oct. 2021. DOI: 10.1007/s11554-021-01177-w.

[210] B. Kovács, A. D. Henriksen, J. D. Stets, and L. Nalpantidis, "Object detection on TPU accelerated embedded devices," in *Lecture Notes in Computer Science*, Springer International Publishing, 2021, pp. 82–92. DOI: 10.1007/978-3-030-87156-7_7.

[211] J. R. Davidson, A. Silwal, C. J. Hohimer, M. Karkee, C. Mo, and Q. Zhang, "Proof-of-concept of a robotic apple harvester," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2016. DOI: 10.1109/iros.2016.7759119.

[212] E. Vitzrabin and Y. Edan, "Adaptive thresholding with fusion using a RGBD sensor for red sweet-pepper detection," *Biosystems Engineering*, vol. 146, pp. 45–56, Jun. 2016. DOI: 10.1016/j.biosystemseng.2015.12.002.

[213] B. B. Pinto, I. Pinheiro, and S. C. Magalhães, *TRIBE AGROset—an agricola visual dataset for object detection*, 2023. DOI: 10.5281/ZENODO.8121089.

[214] European Organization For Nuclear Research and OpenAIRE, *Zenodo*, en, 2013. DOI: 10.25495/7GXK-RD71. [Online]. Available: https://www.zenodo.org/.

[215] A. Kuznetsova, T. Maleva, and V. Soloviev, "Using YOLOv3 algorithm with pre- and post-processing for apple detection in fruit-harvesting robot," *Agronomy*, vol. 10, no. 7, p. 1016, Jul. 2020, ISSN: 2073-4395. DOI: 10.3390/agronomy10071016.

[216] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693, Springer Verlag, 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48. arXiv: 1405.0312.

[217] Stereolabs Inc. "ZED." (2020), [Online]. Available: https://www.stereolabs.com/zed/ (visited on 11/25/2020).

[218] NVIDIA Corporation. "Jetson Nano developer kit." (2020), [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (visited on 08/05/2022).

[219] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010, ISSN: 0920-5691. DOI: 10.1007/s11263-009-0275-4.

[220] B. Sekachev *et al.*, *Opencv/cvat: V1.1.0*, 2020. DOI: 10.5281/ZENODO.4009388.

[221] Tzutalin. "LabelImg." (Sep. 22, 2022), [Online]. Available: https://github.com/tzutalin/labelImg (visited on 05/16/2024).

[222] W. Tengfei. "Pascal VOC tools." (Mar. 16, 2021), [Online]. Available: https://github.com/wang-tf/pascal_voc_tools (visited on 09/08/2021).

[223] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, Jul. 2019, ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0.

[224] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, Apr. 2018, ISSN: 0168-1699. DOI: 10.1016/j.compag.2018.02.016.

[225] Raspberry Pi, Ltd. "Raspberry Pi 4 model B." (2021), [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (visited on 08/15/2021).

[226] Raspberry Pi, Ltd. "Raspberry Pi high-quality camera." (2021), [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/ (visited on 08/15/2021).

[227] United States Department of Agriculture, *United States standards for grades of fresh tomatoes*, en, A. M. Service, Ed., Standard, United States of America: Fruit and VegetableDivision, Oct. 1, 1991. [Online]. Available: https://www.ams.usda.gov/grades-standards/tomato-grades-and-standards (visited on 07/11/2023).

[228] L. Santos, A. Aguiar, and F. N. d. Santos, *Vine trunk image/annotation dataset*, 2021. DOI: 10.5281/ZENODO.5362354.

[229] OpenCV AI and Luxonis Holding corporation. "OpenCV AI kit: OAK-D." (2023), [Online]. Available: https://store.opencv.ai/products/oak-d (visited on 09/22/2023).

[230] GSM Arena. "Xiaomi POCO F2 Pro." (Apr. 24, 2024), [Online]. Available: https://www.gsmarena.com/xiaomi_poco_f2_pro-10220.php (visited on 05/16/2024).

[231] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, Savannah, GA: USENIX Association, 2016, pp. 265–283, ISBN: 978-1-931971-33-1.

[232] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[233] Google LLC. "Google Colaboratory." (2023), [Online]. Available: http://colab.research.google.com (visited on 09/19/2023).

[234] TensorFlow. "TensorFlow 1 detection model zoo." (Oct. 14, 2021), [Online]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md (visited on 09/19/2023).

[235] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016. DOI: 10.1109/cvpr.2016.308.

[236] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, 2015, pp. 448–456. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.

[237] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," Apr. 23, 2020. DOI: 10.48550/ARXIV.2004.10934. arXiv: 2004.10934 [cs.CV].

[238] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, Jun. 2020. DOI: 10.1109/cvprw50498.2020.00203.

[239] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Lecture Notes in Computer Science*. Springer International Publishing, 2014, pp. 346–361, ISBN: 9783319105789. DOI: 10.1007/978-3-319-10578-9_23.

[240] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018. DOI: 10.1109/cvpr.2018.00913.

[241] TensorFlow. "TensorFlow." (2022), [Online]. Available: https://www.tensorflow.org/ (visited on 08/05/2022).

[242] Keras. "Keras." (2022), [Online]. Available: https://keras.io/ (visited on 08/05/2022).

[243] Advanced Micro Devices, Inc. "Vitis-AI." (2022), [Online]. Available: https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html (visited on 08/05/2022).

[244] Google LLC. "Edge TPU compiler." (2020), [Online]. Available: https://coral.ai/docs/edgetpu/compiler/ (visited on 08/05/2022).

[245] NVIDIA Corporation. "Deep learning frameworks documentation." (Oct. 27, 2022), [Online]. Available: https://docs.nvidia.com/deeplearning/frameworks/tf-trt-user-guide/index.html (visited on 11/05/2022).

[246] NVIDIA Corporation. "Jetson Nano 2GB developer kit." (2022), [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-2gb-developer-kit (visited on 08/05/2022).

[247] NVIDIA Corporation. "Harness AI at the edge with the Jetson TX2 developer kit." (2022), [Online]. Available: https://developer.nvidia.com/embedded/jetson-tx2-developer-kit (visited on 08/05/2022).

[248] NVIDIA Corporation. "GeForce RTX3090 Family." (2022), [Online]. Available: https://www.nvidia.com/en-eu/geforce/graphics-cards/30-series/rtx-3090-3090ti/ (visited on 08/05/2022).

[249] Advanced Micro Devices, Inc. "DPU for convolutional neural network." Last accessed on 27/07/2022. (2022), [Online]. Available: https://www.xilinx.com/products/intellectual-property/dpu.html.

[250] Docker Inc. "Docker: Accelerated container application development." (May 8, 2024), [Online]. Available: https://www.docker.com/ (visited on 05/16/2024).

[251] Google LLC. "USB accelerator." (2020), [Online]. Available: https://coral.ai/products/accelerator (visited on 09/21/2023).

[252] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, Feb. 2020. DOI: 10.1109/tpami.2018.2858826.

[253] S. Humbarwadi. "Object detection with RetinaNet." (Jul. 10, 2023), [Online]. Available: https://github.com/keras-team/keras-io/blob/master/examples/vision/retinanet.py (visited on 05/16/2024).

[254] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA: IEEE, Jul. 2017. DOI: 10.1109/cvpr.2017.106.

[255] L. Tan, T. Huangfu, L. Wu, and W. Chen, "Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification," *BMC Medical Informatics and Decision Making*, vol. 21, no. 1, Nov. 2021. DOI: 10.1186/s12911-021-01691-8.

[256] Á. Morera, Á. Sánchez, A. B. Moreno, Á. D. Sappa, and J. F. Vélez, "SSD vs. YOLO for detection of outdoor urban advertising panels under multiple variabilities," *Sensors*, vol. 20, no. 16, p. 4587, Aug. 2020, ISSN: 1424-8220. DOI: 10.3390/s20164587.

[257] TensorFlow. "The Functional API." (May 26, 2023), [Online]. Available: https://www.tensorflow.org/guide/keras/functional (visited on 08/05/2023).

[258] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, *et al.* "KerasTuner." Last accessed on 28/07/2022. (2019), [Online]. Available: https://github.com/keras-team/keras-tuner.

[259] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, no. 185, pp. 1–52, 2018.

[260] M. Blott *et al.*, "FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 3, pp. 1–23, 2018.

[261] Y. Umuroglu *et al.*, "FINN: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17, ACM, 2017, pp. 65–74.

[262] A. Pappalardo, *Xilinx/Brevitas*, 2023. DOI: 10.5281/zenodo.3333552.

[263] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognition*, vol. 105, p. 107 281, Sep. 2020, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2020.107281.

[264] C. Yuan and S. S. Agaian, "A comprehensive review of binary neural network," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 12 949–13 013, Mar. 2023, ISSN: 1573-7462. DOI: 10.1007/s10462-023-10464-w.

[265] The Linux Foundation. "ONNX—Open Neural Network Exchange." (2019), [Online]. Available: https://onnx.ai/ (visited on 04/23/2024).

[266] Y. Umuroglu and M. Jahre, "Streamlined deployment for quantized neural networks," May 2018. DOI: 10.48550/ARXIV.1709.04060.

[267] M. Insall and E. W. Weisstein. "Modular arithemetic." MathWorld—A Wolfram Web Resource, Ed. (May 11, 2024), [Online]. Available: https://mathworld.wolfram.com/ModularArithmetic.html (visited on 05/14/2024).

[268] G. Moreira and S. C. Magalhães. "Tomato maturity classification using HSV and gaussian features." (Oct. 25, 2021), [Online]. Available: https://github.com/gerfsm/HSV_Colour_Space_Model (visited on 10/25/2021).

[269] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, IEEE, 2020, pp. 237–242. DOI: 10.1109/iwssip48289.2020.9145130.

[270] Voxel51. "FiftyOne." (2023), [Online]. Available: https://docs.voxel51.com/ (visited on 12/14/2023).

[271] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," 2020. DOI: 10.48550/ARXIV.2004.09602. arXiv: 2004.09602 [cs.LG].

[272] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," Dec. 18, 2014. DOI: 10.48550/ARXIV.1412.6115. arXiv: 1412.6115 [cs.CV].

[273] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (XAI): Toward medical XAI," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, pp. 4793–4813, Nov. 2021, ISSN: 2162-2388. DOI: 10.1109/tnnls.2020.3027314.

[274] Advanced Micro Devices, Inc. "PetaLinux tools." (2022), [Online]. Available: https://www.xilinx.com/products/design-tools/embedded-software/petalinux-sdk.html (visited on 08/05/2022).

[275] K. Ko, I. Jang, J. H. Choi, J. H. Lim, and D. U. Lee, "Stochastic decision fusion of convolutional neural networks for tomato ripeness detection in agricultural sorting systems," *Sensors*, vol. 21, no. 3, p. 917, Jan. 2021, ISSN: 1424-8220. DOI: 10.3390/s21030917.

[276] Open Robotics. "Gazebo." (2023), [Online]. Available: https://gazebosim.org/ (visited on 05/12/2023).

[277] Luxonis Holding Corporation. "OAK-1." (2023), [Online]. Available: https://docs.luxonis.com/projects/hardwae/en/latest/pages/BW1093/ (visited on 09/26/2023).

[278] Luxonis Holding Corporation. "OAK-SoM." (2023), [Online]. Available: https://docs.luxonis.com/projects/hardware/en/latest/pages/BW1099/ (visited on 09/26/2023).

[279] J. Maye, P. Furgale, and R. Siegwart, "Self-supervised calibration for robotic systems," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, Gold Coast, QLD, Australia: IEEE, Jun. 2013, pp. 473–480. DOI: 10.1109/ivs.2013.6629513.

[280] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes* (McGraw-Hill series in electrical and computer engineering), 4th ed. Boston [u.a.]: McGraw-Hill, 2002, 852 pp., Hier auch später erschienene, unveränderte Nachdrucke, ISBN: 9780072817256.

[281] K. B. Petersen and M. S. Pedersen, *The matrix cookbook*, Version 20121115, Technical University of Denmark, Nov. 2012. [Online]. Available: http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html.

[282] J. Nocedal, F. Öztoprak, and R. A. Waltz, "An interior point method for nonlinear programming with infeasibility detection capabilities," *Optimization Methods and Software*, vol. 29, no. 4, pp. 837–854, Feb. 2014, ISSN: 1029-4937. DOI: 10.1080/10556788.2013.858156.

[283] The MathWorks, Inc. "MATLAB 9.14.0.2206163 (R2023a)." (2024), [Online]. Available: https://www.mathworks.com (visited on 01/17/2024).

[284] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 241–258, Aug. 2019, ISSN: 2095-2236. DOI: 10.1007/s11704-019-8208-z.

[285] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 1623–1637, Mar. 2022, ISSN: 1939-3539. DOI: 10.1109/tpami.2020.3019967.

[286] R. Birkl, D. Wofk, and M. Müller, "MiDaS v3.1 – a model zoo for robust monocular relative depth estimation," Jul. 26, 2023. DOI: 10.48550/ARXIV.2307.14460. eprint: 2307.14460.

[287] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *AFIPS '67 (Spring): Proceedings of the April 18-20, 1967, spring joint computer conference*, Atlantic City New Jersey: ACM Press, Apr. 1967, pp. 483–485. DOI: 10.1145/1465482.1465560.

[288] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer Berlin Heidelberg, 2012. DOI: 10.1007/978-3-642-29044-2.

[289] V. Freitas, *Parsifal*, Last accessed on 16/12/2021, 2021. [Online]. Available: https://parsif.al/.

[290] A. C. M., Inc., *ACM digital library*, Last Accessed on 16/12/2021, 2021. [Online]. Available: http://portal.acm.org.

[291] Elsevier, *Engineering village*, Last accessed on 16/12/2021, 2021. [Online]. Available: http://www.engineeringvillage.com.

[292] IEEE, *IEEE xplore*, Last accessed on 16 /12/2021, 2021. [Online]. Available: http://ieeexplore.ieee.org.

[293] Clarivate, *Web of science*, Last accessed on 20/10/2021, 2021. [Online]. Available: https://www.webofscience.com/wos/woscc/basic-search.

[294] B. V. Elsevier, *Scopus*, Last accessed on 20/10/2021, 2021. [Online]. Available: https://www.scopus.com/.

[295] M. J. Page *et al.*, "PRISMA 2020 explanation and elaboration: Updated guidance and exemplars for reporting systematic reviews," *BMJ*, vol. 372, n160, 2021. DOI: 10.1136/bmj.n160.