THÈSE POUR OBTENIR LE GRADE DE DOCTEUR DE L'UNIVERSITE DE MONTPELLIER

En Systèmes Automatiques et Microélectroniques (SYAM)

École doctorale : Information, Structures et Systèmes

Unité de recherche LIRMM

Dual-Arm Robots for Precision Fruit Harvesting in Agricultural Environments

Présentée par Ege GURSOY

Le 18 décembre 2024

Sous la direction de Andrea CHERUBINI

Devant le jury composé de

Olivier STRAUSS, Professeur des Universités, Université de Montpellier, LIRMM	Examinateur *(Président)*
Viviane CADENAT, Maître de Conférences, Université Toulouse III, LAAS	Rapporteuse
Youcef MEZOUAR, Professeur des Universités, SIGMA Clermont, Institut Pascal	Rapporteur
Marija POPOVIĆ, Assistant Professor, TU Delft (Pays-Bas)	Examinatrice
Akansel COSGUN, Senior Lecturer, Deakin University, (Australie)	Examinateur
Dana KULIĆ, Professor, Monash University (Australie)	Invitée
Andrea CHERUBINI, Professeur des Universités, Ecole Centrale Nantes, LS2N	Directeur de thèse

Titre : Robots à Deux Bras pour la Récolte de Précision des Fruits en Environnement Agricole

Résumé :

L'intérêt pour la robotique agricole a considérablement augmenté en raison de son potentiel à accroître la productivité et à réduire la main-d'œuvre dans l'agriculture. Cependant, développer des systèmes robotiques pour la récolte de fruits dans des environnements non structurés présente des défis importants, notamment en ce qui concerne la perception, la planification et la manipulation par les robots.

Pour relever ces défis, nous proposons un robot à deux bras qui intègre perception et contrôle pour la détection et la récolte des fruits. Équipé d'une caméra RGB-D et d'outils pour couper et collecter les fruits, le robot utilise une stratégie de contrôle basée sur la programmation quadratique hiérarchique (HQP) pour évoluer dans ces environnements complexes tout en respectant des contraintes telles que les limites articulaires, les collisions internes et les collisions potentielles avec les fruits et l'arbre. En combinant l'apprentissage profond avec le traitement d'images traditionnel, notre approche permet une détection et un suivi précis des fruits et des troncs d'arbres dans des environnements contrôlés. Ces méthodes sont validées par des tests en conditions réelles sur des images RGB-D et des simulations, démontrant la robustesse et la fiabilité du système.

Dans cette optique, la recherche s'étend à l'optimisation de l'efficacité et de l'adaptabilité des robots à deux bras grâce à des algorithmes de planification et de programmation globale. Pour faire face à la nature dynamique et imprévisible des environnements agricoles, nous mettons en œuvre un algorithme de planification basé sur le Problème du Voyageur de Commerce (TSP) afin de minimiser la distance nécessaire pour récolter tous les fruits visibles. La planification globale des mouvements garantit une opération sans collisions, augmentant à la fois la sécurité et l'efficacité. Cette méthodologie est validée à l'aide de deux robots distincts : Baxter, un robot accessible et largement disponible, et BAZAR, une plateforme à deux bras de haute précision développée au LIRMM. Des expériences en conditions réelles sur des pommes et des oranges illustrent l'adaptabilité de nos méthodes.

De plus, notre travail aborde les défis spécifiques liés aux occultations pendant la récolte des fruits. Ces occultations peuvent entraîner une localisation imprécise ou empêcher le robot de cueillir le fruit. Nous avons développé une méthode pour les identifier, en utilisant une combinaison d'apprentissage profond et de techniques de traitement d'images traditionnelles, puis pour les lever, en faisant pousser le bras du robot contre la branche responsable de l'occultation. Notre configuration utilise une caméra RGB-D et un seul bras robotisé pour détecter les fruits partiellement occultés et estimer l'apparence complète à l'aide d'un réseau neuronal génératif. Une extension novatrice en 3D de la transformée de Hough 2D détecte les branches dans le nuage de points, permettant ainsi au robot d'identifier quelles branches causent des occultations, et de les pousser convenablement. Cette technique a été testée sur différents types de fruits et dans diverses conditions d'éclairage, prouvant son efficacité pour augmenter la

visibilité et dégager les obstructions, et garantissant ainsi une récolte fructueuse.

En intégrant ces trois stratégies : la coordination des robots à deux bras pour une récolte robuste, l'optimisation de la planification et de la programmation pour l'efficacité, et la gestion dynamique des occultations pour la précision, notre recherche présente une solution originale aux complexités de la récolte robotisée des fruits dans des environnements non structurés.

Les résumé des travaux en français est également disponible dans l'Appendix C.

Mots-clefs : Manipulation Bi-Bras, Robotique Agricole, Récolte de Fruits, Perception Basée sur la Vision, Stratégies de Contrôle Avancées.

Discipline : Informatique, Structure et Systèmes.

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier,
UMR 5506 CNRS/Université de Montpellier,
Batiment 5 - 860 rue de St Priest,
34095 Montpellier, Cedex 5 - France.

Title: Dual-Arm Robots for Precision Fruit Harvesting in Agricultural Environments

Abstract:

Interest in agricultural robotics has significantly increased due to its potential to increase productivity and reduce labor in farming. However, developing robotic systems for fruit harvesting in unstructured environments presents considerable challenges, particularly regarding robot perception, planning, and manipulation.

To address these challenges, we propose a dual-arm robot that integrates perception and control for fruit detection and harvesting. The robot, equipped with an RGB-D camera and with tools for cutting and collecting fruit, uses a Hierarchical Quadratic Programming (HQP) control strategy to move in such complex environments while obeying constraints such as joint limits, self-collisions, and potential collisions with both the fruit and tree. By combining deep learning with traditional image processing, our approach enables accurate detection and tracking of fruits and tree trunks in controlled environments. These methods are validated through real-world testing on RGB-D images and simulations, demonstrating the robustness and reliability of the system.

Building on this, our research extends to optimize the efficiency and adaptability of dual-arm robots through global planning and scheduling algorithms. To cope with the dynamic and unpredictable nature of agricultural settings, we implement a scheduling algorithm based on the Traveling Salesperson Problem (TSP) to minimize the distance required to harvest all visible fruits. Global motion planning ensures collision-free operation, increasing both safety and efficiency. Our methodology is validated using two distinct robots: Baxter, a cost-effective and widely available robot, and BAZAR, a custom-built, high-precision dual-arm platform. Real-world experiments conducted on apples and oranges illustrate the adaptability of our methods.

Additionally, our work tackles the specific challenges caused by occlusions during fruit harvesting, occlusions which can lead to inaccurate localization or impede the robot from picking the fruit. We developed a method to identify these occlusions using a combination of deep learning and traditional image processing techniques, then clear them by having the robot arm push the branch responsible for the occlusion. Our setup uses an RGB-D camera and a single robot arm to detect partially occluded fruits and estimate their full appearance with a generative neural network. A novel 3D extension of the 2D Hough transform detects tree branches in the point cloud, allowing the robot to identify which branches cause occlusions and to strategically push them aside. This technique has been tested across different fruit types and lighting conditions, proving its effectiveness in increasing visibility and clearing obstructions, thus ensuring successful fruit harvesting.

By integrating these three strategies: coordinating dual-arm robots for robust harvesting, optimizing planning and scheduling for efficiency, and dynamically handling occlusions for accuracy, our research presents a comprehensive solution to the complexities of robotic fruit harvesting in unstructured environments.

Keywords: Dual-Arm Manipulation, Agricultural Robotics, Fruit Harvesting, Vision-Based Perception, Dual-Arm Control.

Discipline: Computer Science, Structure, and Systems.

Montpellier Laboratory of Computer Science, Robotics, and Microelectronics,
UMR 5506 CNRS/University of Montpellier,
Building 5 - 860 Rue de Saint Priest,
34095 Montpellier, Cedex 5 - France.

Acknowledgments

First things first, merci mille fois to my PhD director Andrea, who has been much more than a supervisor. You have been a mentor, an inspiration, and the backbone of my academic journey since my master's degree. Andrea, I genuinely respect you, for your brilliance and for your patience during all those nuits blanches before deadlines. Your teachings and guidance will remain my compass as I move forward in life.

To Dana Kulic and Akan Cosgun at Monash University, thank you for believing in me and welcoming me to Australia. Your support made the Monash Robotics Lab feel like a warm and welcoming place. Speaking of Monash, I want to pause and remember Tina (RIP). You were my very first friend in the lab, and your won't be forgotten.

To my jury members and rapporteurs, I am deeply grateful for your time, effort, and valuable insights. A special thanks to Youcef and Viviane, my rapporteurs, for their meticulous reviews and thoughtful feedback. Youcef, thank you for your patience as you were also in my CSI, your guidance has been indispensable. Marija, Olivier, thank you for your role as examinateurs. Olivier, having been my teacher on vision (a big section of my thesis), which inspired me to dive into this field.

Big shoutout to my colleagues (too many to name every one of them!) at LIRMM for all the shared experiments, brainstorming sessions, and laughs. My Monash colleagues, who some of them then became wonderful housemates. As myself *Edgardo* to *Adri*, *Amistoso*, *Shrayanitos*, *Vinicius*, *Sashito*, *Philipo*, and *Alexa - Kuchi Mama*, thank you for making every challenge bearable and every victory tastier with great foods. Not to forget the rest of the crew dropped by casually at our house!

In Montpellier, I've been blessed with so many amazing friends that listing them all would turn this thesis into a phone book. Thank you for being there through thick and thin. To my friends back in Turkey, who are now scattered around the globe: you've always cheered me on and supported me during hard times. Special mention to my Piri Reis crew, 23+ years of friendship and counting and Daniel's crew.

Last but never least, my family. Mom and Dad, I wouldn't be where I am today without your endless support. My brother Cem, I hope that if you want, you'll reach these achievements and surpass me.

Alors voili voilou (as Andrea would say), this thesis is as much yours as it is mine. Thank you, merci, teşekkürler!

Contents

Acknowledgments **7**

List of Figures **15**

Nomenclature **16**

1 Introduction **19**
1.1 Context . . . 19
1.2 Motivation . . . 20
1.3 Challenges . . . 22
1.4 Objectives . . . 23
1.5 Overview of the Research . . . 24
1.6 Hardware . . . 26
1.7 Contributions . . . 27
1.8 Research Output . . . 28
1.9 Thesis Structure . . . 30

2 State of the Art **33**
2.1 Perception in Robotic Harvesting . . . 33
2.2 Motion Planning in Agriculture . . . 36
2.3 Occlusions Handling in Robotic Harvesting . . . 39
2.4 Robotic Harvesting Platforms . . . 43
2.5 Synthesis . . . 45

3 Initial Development of Dual-Arm Harvesting Robot **47**
3.1 Motivation and Challenges . . . 47
3.2 Perception . . . 48
3.2.1 Fruit Localization . . . 49
3.2.2 Fruit Selection . . . 52
3.2.3 Tree Localization . . . 52
3.2.4 Synthesis . . . 53

3.3 Motion Control . . . 54
3.3.1 Dual-Arm Cooperative Control . . . 54
3.3.2 Objective Tasks . . . 55
3.3.3 Hierarchical Tasks . . . 57
3.3.4 Kinematic Constraints . . . 59
3.3.5 Collision Avoidance Constraints . . . 59
3.3.6 Optimization Problem . . . 61
3.3.7 Joint Velocity Command . . . 62
3.3.8 Tree Avoidance . . . 62
3.3.9 Detaching and Collecting . . . 63
3.3.10 Synthesis . . . 63
3.4 Experiments . . . 64
3.4.1 Platform Design . . . 65
3.4.2 Evaluation . . . 67
3.4.3 Synthesis . . . 72
3.5 Conclusion and Literature Context . . . 73

4 Towards Real-World Adaptation **77**
4.1 Motivation and Challenges . . . 77
4.2 Perception . . . 78
4.2.1 Fruit Tracking . . . 79
4.2.2 Fruit Scheduling . . . 81
4.2.3 Synthesis . . . 83
4.3 Motion Control . . . 83
4.3.1 Dual-Arm Trajectory Generation . . . 84
4.3.2 Obstacle Avoidance . . . 86
4.3.3 Detaching and Collecting . . . 87
4.3.4 Synthesis . . . 88
4.4 Experiments . . . 88
4.4.1 Platform Design . . . 89
4.4.2 Experimental Setup . . . 90
4.4.3 Fruit Detection and Tracking Results . . . 91
4.4.4 Collision Avoidance and Harvesting Results . . . 92
4.4.5 Trajectory Tracking Results . . . 92
4.4.6 Performance Metrics Evaluation . . . 94
4.4.7 Synthesis . . . 97
4.5 Conclusion and Literature Context . . . 97

5 Fruit Occlusion Handling **101**
5.1 Motivation and Challenges . . . 101
5.2 Fruit Identification . . . 102
5.2.1 Partial Fruit Segmentation . . . 103
5.2.2 Fruit Appearance Estimation . . . 104

5.2.3 Synthesis . . . 108
5.3 Occlusion Identification . . . 108
5.3.1 View Gradient . . . 108
5.3.2 Branch Line Segments . . . 110
5.3.3 Synthesis . . . 112
5.4 Occlusion Clearance . . . 113
5.4.1 Line Decision . . . 113
5.4.2 Push Action . . . 115
5.4.3 Synthesis . . . 116
5.5 Experiments . . . 116
5.5.1 Setup . . . 116
5.5.2 Perception Results . . . 117
5.5.3 Occlusion Identification Results . . . 117
5.5.4 Occlusion Clearance Results . . . 119
5.5.5 Synthesis . . . 119
5.6 Conclusion and Literature Context . . . 122

6 Conclusion **125**
6.1 Summary . . . 125
6.2 Achievements . . . 126
6.3 Discussion . . . 127
6.4 Future Work . . . 129
6.5 Final Remarks . . . 131

A Software Details **133**

B 3D Models **143**

C Résumé des travaux **147**

Bibliography **162**

List of Figures

1.1 Future of agriculture according to DALL-E 3 [Ope23]. 20
1.2 Labor-intensive manual fruit harvesting. 21
1.3 Demonstration of the initial system. 25
1.4 BAZAR robot harvesting oranges (left) and Baxter harvesting apples (right). 25
1.5 BAZAR pushes an occluding branch. 26
1.6 Cutting tools of Baxter (top-left) and BAZAR (bottom-left), along with collecting tools of Baxter (top-right) and BAZAR (bottom-right). . . . 27
1.7 Overview of the manuscript Structure, Contributions, and Limitations . 29

2.1 Examples of deep learning based fruit detection. 34
2.2 Fruit tracking with Structure from Motion by Roy et al. [RI16] 35
2.3 Examples of specially arranged orchards for automated fruit harvesting. 37
2.4 Examples of task coordination and scheduling. 38
2.5 Examples of occluded fruit detection. 40
2.6 Examples of occlusion handling. 41
2.7 Early examples robotic harvesting platforms. 43
2.8 Modern examples of dual-arm harvesting platforms. 44

3.1 Coordinate frames definitions. The Absolute Frame is linked to the fixed World Frame, whereas the Relative Frame is linked to the moving Absolute Frame. 49
3.2 *Hypothesis 1* of fruit as an ellipsoid (left) and its projection as an ellipse in the image plane (right). 50
3.3 Visualization of the tree obstacle (gray cylinder) and fruit obstacle (orange ellipsoid). 63
3.4 Dual arm robot BAZAR [CPN+19] with cutting and collecting arms. . 65
3.5 Simulation environment with the scanned tree model, detected fruits, and the BAZAR robot. 67
3.6 Tracking and detection evaluation images, with time progressing from left to right. 68

3.7 Perception images in Experiment 1.Top-left: RGB image with fruit detection. Top-right: corresponding depth image. Bottom-left: HSV mask. Bottom-right: Depth mask. 68
3.8 Perception images in Experiment 2. Top-left: RGB image with fruit detection. Top-right: corresponding depth image. Bottom-left: HSV mask. Bottom-right: Depth mask. 69
3.9 Evaluation of tracking and detection over time in Experiment 1. 70
3.10 Evaluation of tracking and detection over time in Experiment 2. 71
3.11 Results of the harvesting experiment. Top: norm of the position error in meters. Bottom: norm of the orientation error in radians. 72

4.1 BAZAR robot harvesting oranges (left) and Baxter harvesting apples (right). 78
4.2 Overview of our dual-arm harvesting framework. Purple boxes represent the differences from the methodologies in Chapter 3. 79
4.3 Cutting tools of Baxter (top-left) and BAZAR (bottom-left), along with collecting tools of Baxter (top-right) and BAZAR (bottom-right). . . . 87
4.4 Apple detection with tracking IDs usign Baxter (left) and orange detection with tracking IDs using BAZAR (right). 90
4.5 Snapshots of the autonomous harvesting experiments. 93
4.6 Evaluation of position (in m) and orientation (in deg) errors between the end-effectors and the goal poses in time. The top graphs show the norm of the position error over time, while the bottom graphs show the orientation error. 94

5.1 Camera field of view (left), with corresponding image (right). The tree branches and leaves are green/brown, and the fruit is orange. 103
5.2 Outline of the fruit estimation method. Left to right: Visible fruit pixels in masked depth image $\mathbf{D}_M$, cropped ROI depth image $\mathbf{D}'_M$ fed into the generative model (represented by a generic *autoencoder* shape), which outputs the estimated fruit depth image $\hat{\mathbf{D}}'_M$. The white circle represents the estimated fruit center $(\hat{u}'_c, \hat{v}'_c)$, and the blue circle indicates the center of the occlusion mask. The red arrow depicts the occlusion direction vector $\vec{\mathbf{O}}_f$. The cross-hatched area represents the occluded region estimated by the generative model. 104
5.3 Architecture of the Generative Neural Network for Fruit Appearance Estimation. The network consists of an encoder (left) and a decoder (right) with skip connections between corresponding layers. 106
5.4 Field of view from the camera (left) and Image (right). The figure shows: fruit centroid (white circle) $(\hat{u}_c, \hat{v}_c)$, occlusion centroid (blue circle) $(\bar{u}_c, \bar{v}_c)$, view frustum (blue cone), its center line $\mathcal{F}^z_{fruit}$ (green arrow) and view gradient $\vec{\mathbf{O}}_f$ (red arrow). 113

5.5 Line segments (grey) and their extremities (black circles). The pushing line $\mathbf{l}_{push}$ $\mathbf{l}_{push}$ is selected as the topmost according to Algorithm 4. . . . 115
5.6 Fruit estimation results. From left to right in each subfigure: segmented fruit in the depth image input to the fruit estimator, estimation result, difference between input and output, and view gradient (red vector) on the estimated fruit. 118
5.7 Line detection before (left) and after (right) removing similar lines. . . 119
5.8 From left to right: original scene, segmentation results, and line detection results. In the line detection image: detected and filtered lines are yellow, lines within the distance threshold are blue, and the selected pushing line is green. The pushing point is marked in red. 120
5.9 Robot branch pushing demonstration using BAZAR. The robot successfully clears the occlusion by pushing the selected branch, enhancing fruit visibility for harvesting. 121

B.1 Electric pruner 3D scan and its simplified mesh used in collision detection 143
B.2 BAZAR end-effector support 3D model 144
B.3 Electric pruner holder end-effector support 3D model 145

Bold symbols are for vectors.

Acronyms

CAE	Convolutional Autoencoder
CNN	Convolutional Neural Network
COCO	Common Objects in Context Dataset
DOF	Degrees Of Freedom
FPS	Frames per Second
GJK	Gilbert-Johnson-Keerthi algorithm
HQP	Hierarchical Quadratic Programming
HSV	Hue, Saturation, Value
ICP	Iterative Closest Point
KNN	K-Nearest Neighbors
LiDAR	Light Detection and Ranging
MAE	Mean Absolute Error
OSNet	Omni-Scale Network
ReID	Re-identification
RKCL	Robot Kinematics Control Library
RNN	Recurrent Neural Network
ROI	Region of Interest
SDK	Software Development Kit
SLAM	Simultaneous Localization and Mapping
SORT	Simple Online and Realtime Tracking
TAI	Tracking Accuracy of IDs
TRAC-IK	Track Inverse Kinematics Solver
TSP	Traveling Salesperson Problem
U-Net	U-Net Neural Network Architecture
URDF	Unified Robot Description Format
YAML	YAML Ain't Markup Language

List of symbols

α	Rotation angle [rad]
$\dot{\boldsymbol{\theta}}$	Joint velocity command
ϵ	Deviation
η	Efficiency ratio [%]
$\mathbf{A}$	Matrix of linear coefficients for the constraints
$\mathbf{J}$	Jacobian matrix

$\mathbf{K}$	Rotation axis vector
$\mathbf{N}$	Normal vector
$\mathbf{P}_f$	3D position [m]
$\mathbf{T}$	Time [s]
$\mathbf{\Delta}$	Average
$\dot{\mathbf{x}}$	Task space velocity
$\mathbf{b}$	Vector of inequality constraint bounds
$\mathbf{grad}$	View gradient for occlusion detection
$\mathbf{m}$	Mapping function between 2D and 3D coordinates
$\mathbf{q}$	Joint positions
$\mathbf{r}$	Robot pose
$\mathbf{x}$	Task space position
d	Distance [m]
n	Number
u	Pixel coordinate in the horizontal axis
v	Pixel coordinate in the vertical axis

CHAPTER 1

Introduction

1.1 Context

Agriculture has been a cornerstone of human civilization as it provides the essential resources needed for survival and growth. In recent decades, the agricultural industry has faced significant pressures due to a rapidly increasing global population, changing climate conditions, and the necessity for sustainable practices. Traditional farming methods, while historically effective, are struggling to meet the demands of modern society. As researchers, we are exploring ways to integrate robotics and automation into the agricultural sector. Here is a predictive image of the future of agriculture Figure 1.1.

The integration of robotics and automation into agriculture, often referred to as *precision agriculture*, promises to revolutionize the industry. Robots can perform tasks with high precision, consistency, and efficiency, reducing the reliance on manual labor. In the context of fruit harvesting, robots equipped with sensing and manipulation capabilities can identify, locate, and pick fruits without human intervention.

Research in agricultural robotics has primarily focused on developing platforms capable of performing specific tasks in controlled environments. Early robotic harvesters were typically designed with single-arm manipulators, capable of performing basic pick-and-place operations. While these systems demonstrated the feasibility of robotic harvesting, they were limited in their ability to handle the complexities of real-world agricultural settings.

The unstructured nature of orchards and fruit farms presents unique challenges for robots. Factors such as irregular tree shapes, varying fruit positions, occlusions caused by foliage, and dynamic environmental conditions require robots to possess perception, planning, and manipulation capabilities. The need for robots which can adapt to these complexities has driven research towards more sophisticated systems.



Figure 1.1: Future of agriculture according to DALL-E 3 [Ope23].

Dual-arm robots have emerged as a promising solution to increase the capabilities of robotic harvesters. Dual-arm robots can perform more complex tasks by mimicking the bimanual actions of human workers, such as simultaneously cutting and collecting fruits or removing obstacles while harvesting. The additional degrees of freedom and redundancy provided by two arms allow for greater flexibility and adaptability in unstructured environments.

1.2 Motivation

The global agricultural sector is facing issues towards the need to boost productivity and efficiency, to meet the demands of a rapidly growing population. The United

Nations projects that the world population will reach 9.7 billion by 2050 [UNSA19], intensifying the strain on agricultural systems to produce more food with limited resources. Traditional methods of fruit harvesting, heavily reliant on manual labor, are becoming unsustainable due to several factors:



Image source: https://citrusindustry.net/2020/04/02/harvesting-labor-a-concern/

Figure 1.2: Labor-intensive manual fruit harvesting.

Labor shortages in agriculture have become a pressing issue due to factors such as urban migration, aging rural workforce, and declining interest in agricultural careers among young generations. This significant reduction in available farm labor leads to higher labor costs and difficulties in adhering to harvest schedules, directly impacting crop yields and profitability. At the same time, rising production costs, particularly those associated with manual labor, are driven by economic factors and increased competition for workers. As labor costs increase, profit margins for farmers shrink, and food prices for consumers rise. Manual harvesting introduces inconsistencies, since human labor can vary in quality and efficiency. Factors such as differences in skill levels, fatigue, and human error can result in damaged produce and inconsistent harvesting rates. The physically demanding nature of fruit harvesting, which often involves long hours in difficult environmental conditions, raises health and safety concerns, further aggravating the labor shortage issue. The challenging conditions of labors are demonstrated in Figure 1.2.

Automation and robotics offer a promising solution to these challenges. By integrating robotic systems into agricultural practices, there is potential to significantly boost productivity, since robots can operate continuously without fatigue, thereby improving the overall efficiency of harvesting operations. Additionally, automation can reduce labor dependency, help mitigate the impacts of labor shortages and decrease reliance on seasonal or migrant workers. Robots can improve the quality and consistency of harvesting, offering precise and uniform performance to reduce the risk of damaging fruits. Robotic systems can enable more efficient resource management, reducing waste, and contributing to the sustainability of agricultural operations.

This thesis is motivated by the need to develop advanced robotic solutions capable of overcoming all these challenges. Specifically, it explores the potential of **dual-arm robots**n to improve the efficiency, adaptability, and reliability of fruit harvesting operations. Dual-arm robots offer several advantages over single-arm ones:

- **Dexterity and Flexibility:** Mimicking human bimanual manipulation allows dual-arm robots to perform complex tasks such as simultaneously cutting and collecting fruits.
- **Handling Occlusions:** One arm can be utilized to remove occlusions like branches or leaves, while the other arm focuses on harvesting, improving the robot's ability to access and pick occluded fruits.
- **Efficiency Through Parallelism:** Performing multiple tasks concurrently reduces overall harvesting time and increases productivity.
- **Redundancy and Reliability:** Dual-arm systems provide a level of redundancy, improving robustness in handling unexpected situations or failures.

By using the capabilities of dual-arm robots, our research aims at pushing the boundaries of agricultural robotics, contributing to the development of autonomous systems which can operate effectively in real agricultural settings.

1.3 Challenges

Despite significant advancements in robotics and automation, the effective deployment of robots for fruit harvesting in unstructured agricultural environments remains a challenging problem. The primary issues that this thesis aims to address are:

Inadequate Perception Capabilities: Current robotic systems struggle with accurately detecting and localizing fruits in environments where occlusions by leaves, branches, or other fruits are common. Traditional perception algorithms may fail under varying lighting conditions and when dealing with fruits of different sizes, shapes, and colors. There is a need for advanced perception techniques to robustly identify fruits and assess occlusions in real-time.

Limited Manipulation and Control in Complex Environments: Robots must maneuver and operate within unstructured agricultural settings. Existing single-arm robots lack the dexterity and flexibility required to perform tasks such as simultaneous cutting and collecting or manipulating occluding elements to access fruits. Developing control strategies for dual-arm robots which can coordinate multiple tasks, while avoiding collisions with the environment is a significant challenge.

Inefficient Motion Planning Under Constraints: Motion planning for robotic manipulators in cluttered environments is computationally intensive, especially when considering the kinematic constraints of dual-arm systems and the need for real-time operation. Current planning algorithms may not properly handle the trade-offs between optimality, computational efficiency, and safety, leading to suboptimal harvesting performance.

Lack of Real-World Validation and Adaptability: Many robotic harvesting platforms are tested only in simulated or highly controlled environments. There is a gap in demonstrating the effectiveness of these systems under real-world conditions, which include variable lighting, diverse fruit types, and unpredictable environmental factors. Ensuring that robotic systems can adapt to different crops and farm settings is essential for practical adoption.

Integration of Perception and Manipulation for Occlusion Handling: While some research has explored perception or manipulation independently, there is a lack of integrated approaches that combine both to handle occlusions effectively. The ability for a robot to detect occlusions and to decide and execute actions to address them (e.g., moving a branch aside) is critical for improving harvesting success rates.

Computational Constraints for Real-Time Operation: Algorithms for perception and planning can be computationally demanding, posing challenges for real-time operation on robotic platforms with limited processing capabilities. There is a need to develop methods that balance performance with computational efficiency to enable practical application.

1.4 Objectives

The main problem this thesis addresses is the development of a dual-arm platform capable of autonomously performing fruit harvesting tasks in unstructured agricultural environments, while effectively managing occlusions and environmental variability. This involves creating perception algorithms capable of robust fruit detection under occlusions, developing coordinated dual-arm manipulation strategies, and integrating these components into a system which operates efficiently in real-time. The goal is to bridge the gap between theoretical research and practical application, to develop a robot that is both effective and adaptable to the complexities of real-world agriculture. The specific objectives of this research are as follows:

Dual-Arm Robotic Platform: The goal is to design and implement a dual-arm robot that can perform coordinated actions, replicating the dexterity typically seen in

human fruit pickers. This will involve selecting appropriate robot hardware such as end-effectors and sensors, which are essential for the cutting and collecting tasks. A modular system architecture will be developed to integrate perception, planning, and control components to ensure smooth operations. Control algorithms will be implemented for synchronized dual-arm manipulation while considering kinematic constraints and collision avoidance.

Perception under Occlusions: Advanced perception methods will be developed to accurately detect and localize fruits, even when partially or fully occluded by leaves or branches. The approach will use deep learning and classic image processing techniques to enhance detection accuracy under challenging conditions. Additionally, strategies to identify and estimate the extent of occlusions will enable the robot to make informed decisions about manipulating the trees and fruits. RGB-D cameras will provide depth information, supporting precise localization and obstacle detection.

Motion Planning and Control: Motion planning algorithms will be designed to allow the robot to adapt its movements in real time, while ensuring obstacle avoidance and proper handling of occlusions. A planner will be developed to generate optimized, collision-free trajectories considering the entire environment. Strategies for task prioritization will manage multiple objectives and constraints to ensure the completion of critical tasks.

Validation in Real-World Scenarios: Our framework will undergo extensive testing to validate its practical applicability. This will involve testing across multiple robot platforms, to demonstrate the framework's adaptability. Different fruit types, such as apples and oranges, will be used to highlight versatility. Experiments conducted under various lighting and environmental conditions will demonstrate the system's robustness in mock-up agricultural settings.

1.5 Overview of the Research

To address the limitations of single-arm robots, this thesis explores the development and application of dual-arm robotic systems for fruit harvesting, advancing through three key stages of research and development. Each stage builds on the findings of the previous one, with specific improvements to address emerging challenges and optimize system performance.

Initial Development of Dual-Arm Harvesting Robot: The initial stage introduced a novel dual-arm approach designed to improve dexterity, robustness, and versatility in unstructured agricultural environments [GNC+23]. The dual-arm robot, equipped with an RGB-D camera and tools for cutting and collecting, employed a Hierarchical Quadratic Programming (HQP) control strategy to harvest, while accounting for constraints like joint limits, self-collisions, and potential collisions with fruits and tree structures. The robot setup included the dual-arm cobot BAZAR [CPN+19], equipped with two 7-degree-of-freedom Kuka LWR4 robotic arms and an Intel RealSense D435 RGB-D camera, providing real-time visual input. A local motion planner ensured

efficient inverse kinematics computation and precise, coordinated movements.

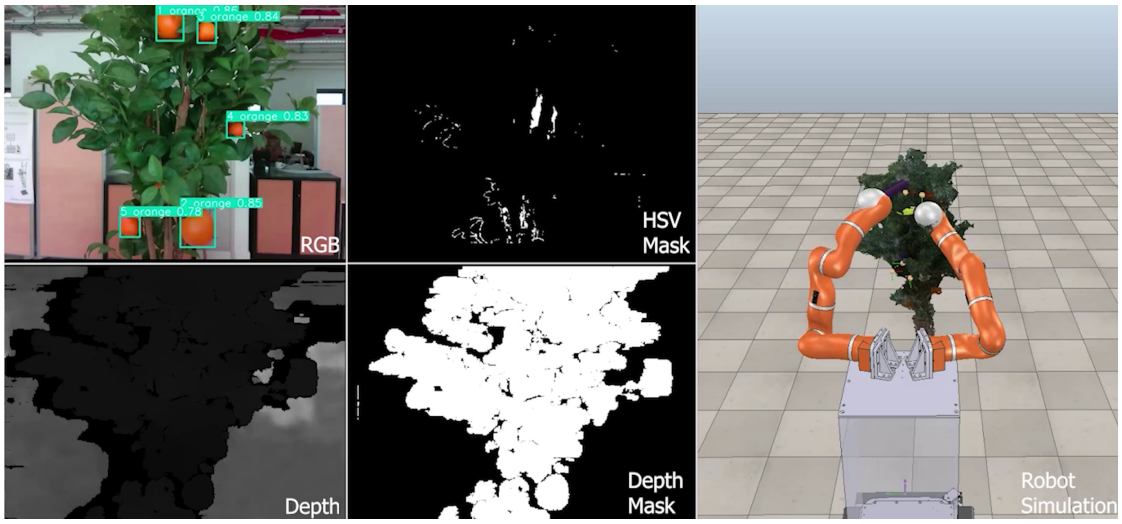


Figure 1.3: Demonstration of the initial system.

Towards Real-World Adaptation: Building on that foundational work, the second stage introduced key modifications to improve the system's real-world potential and adaptability. This phase incorporated a scheduling algorithm based on the Traveling Salesperson Problem (TSP) to optimize the harvesting sequence, minimizing travel distance and reducing harvesting time. Global motion planning was also implemented to ensure collision-free operation in more complex environments. The methodologies were validated through testing with both the Baxter and BAZAR robots, using different fruit types such as apples and oranges (Figure 4.1).



(a) BAZAR robot harvesting oranges. *(b) Baxter robot harvesting apples.*

Figure 1.4: BAZAR robot harvesting oranges (left) and Baxter harvesting apples (right).

Fruit Occlusion Handling: The third stage addressed the significant challenge of occlusions in natural environments, where fruits are often obscured by leaves, branches, or other fruits. A novel approach was developed to actively handle branch occlusions by physically clearing obstructing branches from the robot's field of view [GKC24]. The

methodology combined deep learning for fruit appearance estimation, classical image processing to determine the push direction, and an extended 3D Hough transform for branch detection. The BAZAR robot was equipped with a metal rod attached to its right arm for clearing occlusions. This approach was validated in real experiments under various lighting conditions, resulting in improved visibility and successful fruit detection (Figure 1.5).

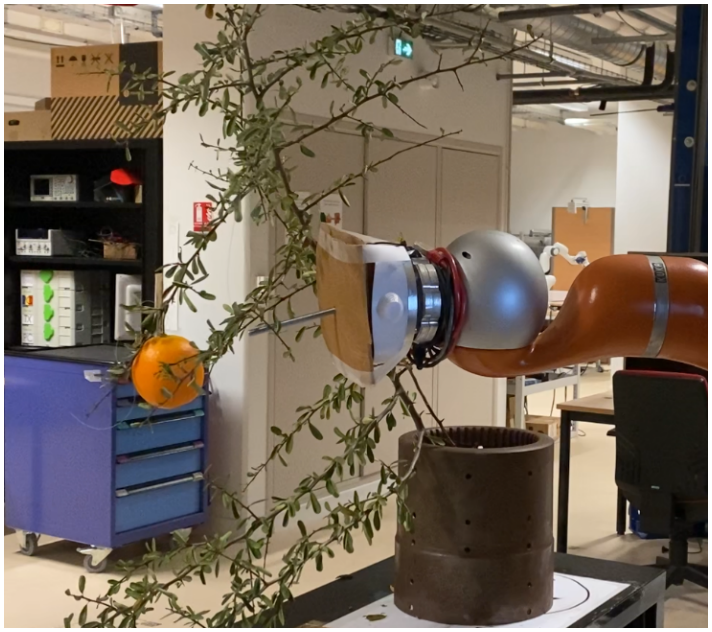


Figure 1.5: BAZAR pushes an occluding branch.

1.6 Hardware

Our research uses two robots: the BAZAR [CPN+19] and the Baxter robot by Rethink Robotics (discontinued). Both platforms are dual-arm robots, each arm offering 7 degrees of freedom (DOF). The BAZAR robot serves as a high-end, specialized platform tailored for complex tasks, while the Baxter robot represents a more accessible, lower-cost option.

Both robots are equipped with custom-designed harvesting tools. For fruit detachment, we use electric pruners and custom cutting devices mounted on the end-effectors. For fruit collection, baskets or soft bags are attached to catch the harvested fruit. For pushing operations, a slender rod is mounted on the end effector to clear obstructing branches.

To simulate realistic harvesting scenarios, we use mock-up apple and orange trees with artificial fruits. These setups allow us to conduct controlled experiments that replicate the challenges of real-world harvesting, such as occlusions and varying fruit positions.



Figure 1.6: Cutting tools of Baxter (top-left) and BAZAR (bottom-left), along with collecting tools of Baxter (top-right) and BAZAR (bottom-right).

- **Baxter Robot**:
  - Cutting Tool: A 3D-printed attachment with two blades mounted onto the robot's gripper.
  - Collecting Tool: A soft, transparent plastic bag attached to the second end-effector.
- **BAZAR Robot**:
  - Cutting Tool: An electric pruner attached to the end-effector.
  - Collecting Tool: A commercially available plastic collecting basket with rigid circular top.

1.7 Contributions

This thesis makes several key contributions to the field of agricultural robotics:

Novel Dual-Arm Robotic System for Fruit Harvesting: We introduce the development of a dual-arm robot integrating perception, planning, and control to enable efficient fruit harvesting in unstructured environments. The system supports simultaneous cutting and collecting actions through coordinated dual-arm manipulation. Additionally, it includes the design of specialized tools, such as cutting implements

and collection baskets, tailored to meet the specific requirements of fruit harvesting. A cohesive system architecture is developed, to ensure seamless interaction between the various components of the robotic platform.

Perception Methods under Occlusions: Innovative perception techniques are introduced to enhance fruit detection and localization, even when fruits are partially or fully occluded by leaves or branches. These techniques include generative modeling, to estimate the appearance of occluded fruits, coupled with an extended 3D Hough transform for detecting branches and other occluding elements. Efficient algorithms are also implemented to quantify occlusions, enabling the robot to make informed decisions during harvesting.

Motion Planning and Control Strategies: Motion planning algorithms are developed to ensure both obstacle avoidance and adaptability in unstructured environments. Hierarchical Quadratic Programming (HQP) is used to manage constraints such as task prioritization and collision avoidance. Both global and local planning methods are integrated to optimize the robot's trajectories. Additionally, a fruit scheduling algorithm based on the Traveling Salesperson Problem is implemented to minimize travel distances and reduce harvesting times.

Real-World Validation: The system undergoes extensive validation through testing in real-world conditions using different robots and fruit types. Its adaptability is demonstrated on both the BAZAR and Baxter robots, highlighting the system's ability to operate across two hardware platforms. The research further proves the system's versatility by successfully harvesting apples and oranges, while improving detection accuracy and increasing harvesting success rates through effective occlusion handling.

Sustainable Agricultural Practices: This research contributes to the advancement of robotic harvesting technologies, promoting sustainable agricultural practices. By reducing dependency on manual labor, the system helps address labor shortages, while improving productivity. The consistent and precise harvesting capabilities of the robot optimize resource utilization, reduce waste, and provide a foundation for scaling robotic solutions in commercial agricultural operations.

1.8 Research Output

Here are the list of scientific publications that I have worked on during my PhD:
Publications:

- Ege Gursoy, Akansel Cosgun, Dana Kulić and Andrea Cherubini. PICKBOT: Dual-arm Harvesting Robot. In *2022 International Conference on Robotics and Automation (ICRA), AgriWS Workshop.*

- Ege Gursoy, Benjamin Navarro, Akansel Cosgun, Dana Kulić and Andrea Cherubini. Towards vision-based dual arm robotic fruit harvesting. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE).*



Figure 1.7: Overview of the manuscript Structure, Contributions, and Limitations

- Ege Gursoy, Dana Kulić and Andrea Cherubini. Occlusion Handling by Pushing for Enhanced Fruit Detection. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

- Ege Gursoy, Akansel Cosgun, Dana Kulić and Andrea Cherubini. Bringing Robotics to the Orchard: Dual-Arm Harvesting in the Real-World. In *IEEE Transactions on Automation Science and Engineering, (invited article, under review)*.

- Ege Gursoy, Sonny Tarbouriech and Andrea Cherubini. Can Robots Mold Soft Plastic Materials by Shaping Depth Images? *In IEEE Transactions on Robotics, 2023 (unrelated to the main PhD topic)*.

Videos:

- Robot Pushes Branches to Pick Fruit: youtu.be/hGmj7sBb9vw

- Towards Robotic Fruit Harvesting: youtu.be/ObCR5biqlOM

- Bringing Robotics to the Orchard: youtu.be/i_AllDjYyA8

- Can Robots Mold Soft Materials by Shaping Depth Images?: youtu.be/fLGf7c3cbKk

Codes and Datasets:

- Codes and datasets are available online at: seafile.lirmm.fr/d/f20f112aec264558a35a/

1.9 Thesis Structure

This thesis is organized as follows:

Chapter 1: Introduction: This chapter introduces the context, motivations, challenges, objectives, and contributions of the thesis, outlining the evolution of dual-arm robots for fruit harvesting.

Chapter 2: State of the Art: This chapter reviews existing literature related to: dual-arm robots, perception techniques under occlusions, motion planning and control strategies, and previous approaches for addressing occlusions and fruit visibility in agricultural robotics.

Chapter 3: Initial Development of Dual-Arm Harvesting Robot: This chapter details the initial design and implementation of a dual-arm robot for fruit harvesting, including our perception system using RGB-D images and deep learning, the tracking and state estimation algorithms, hierarchical motion planning, and collision avoidance strategies.

Chapter 4: Increasing Efficiency and Adaptability: This chapter explores enhancements to the initial system, focusing on improving fruit detection and tracking with advanced deep learning models, optimizing motion planning with a global planner and

adaptive obstacle modeling, implementing efficient fruit scheduling, and validating the framework with multiple robots and fruit types.

Chapter 5: Addressing Occlusions and Fruit Visibility: This chapter introduces advanced perception and manipulation strategies to handle occlusions effectively, including the use of generative modeling to estimate occluded fruits, a 3D extension of the Hough transform for branch detection, and a pushing strategy to remove occlusions, along with real-world validation.

Chapter 6: Conclusion: This chapter summarizes the thesis, discusses its contributions to the field of agricultural robotics, and outlines directions for future research.

Appendix A: Software Details: This appendix provides a technical overview of the software architecture, key libraries, modules, and parameters used throughout the experimental work presented in *Chapters 3, 4, and 5*.

Appendix B: 3D Models: This appendix includes detailed representations of the 3D models developed for the robot end-effectors, and additional tools used in the harvesting setup.

Appendix C: Résumé des Travaux: This appendix provides a summary of the thesis in French, including the research motivations, methodologies, key results, and contributions.

CHAPTER 2

State of the Art

Introduction

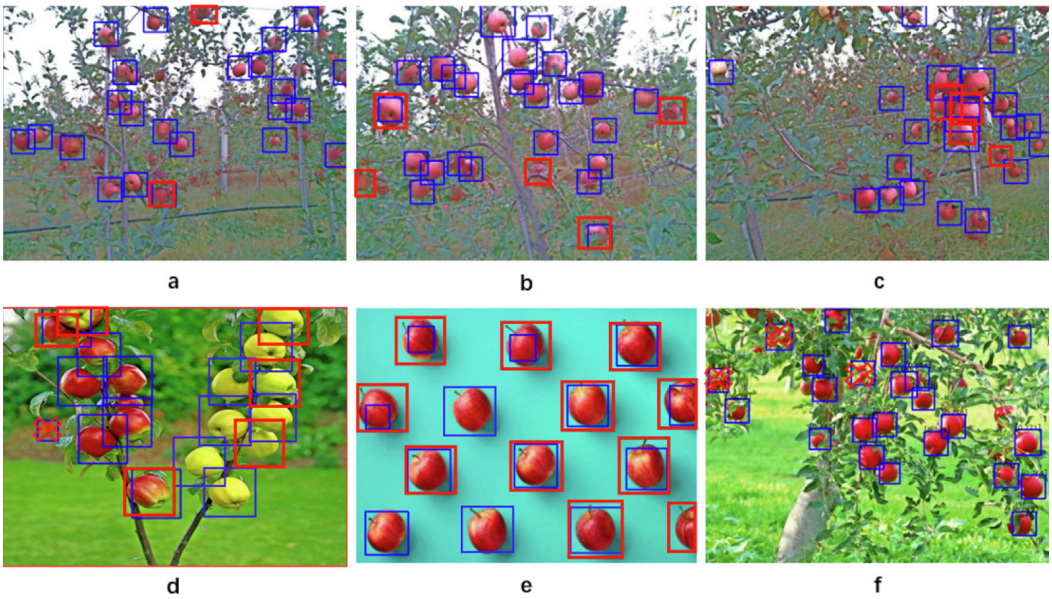
The automation of fruit harvesting has become an essential research area due to labor shortages, rising production costs, and the demand for increased efficiency and consistency in agricultural practices [BV16, BvHHE14]. Robots offer the potential to perform harvesting tasks with precision and minimal human intervention. Among these systems, dual-arm robots have emerged as a promising solution for complex harvesting scenarios, particularly when dealing with occlusions and delicate fruit handling. This chapter reviews the current state of the art in dual-arm robotic systems for fruit harvesting, focusing on their perception (Section 3.2, 4.2) , motion planning (Section 3.3, 4.3), and strategies to address occlusions and fruit visibility (Chapter 5).

2.1 Perception in Robotic Harvesting

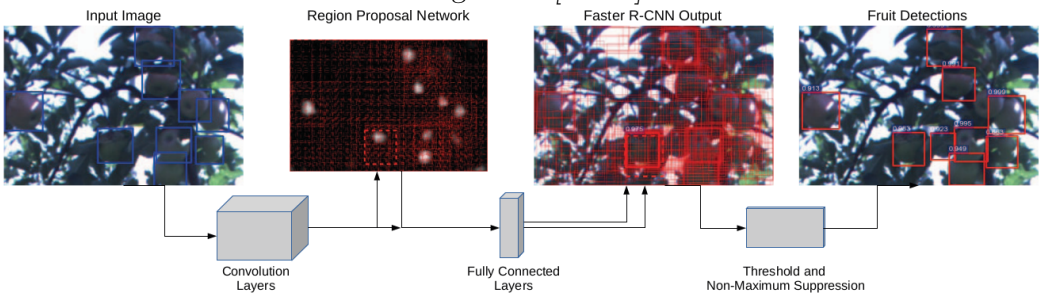
Related Work

Accurate detection and localization of fruits are essential for successful robotic harvesting, as the robot's ability to identify fruits in different environmental conditions directly impacts its success. Researchers have explored various sensors and algorithms to improve fruit detection and robustness across a range of scenarios, with a strong focus on deep learning techniques.

Many studies use deep convolutional neural networks (CNNs) for fruit detection. Sa et al. [SGD+] designed a fruit detection system based on CNNs that demonstrated high detection accuracy, showing deep learning's potential in agricultural applications. Gonzalez et al. [GAT19] applied instance segmentation using deep neural networks for blueberry detection, achieving high accuracy. Liu et al. [LYL+18] developed a similar



Kang et al. [KC20]



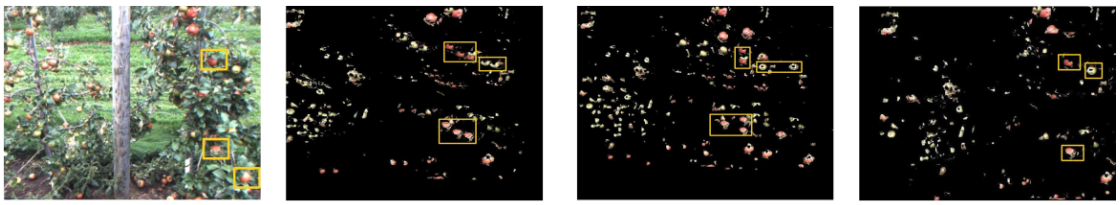
Bargoti et al. [BU17a]

Figure 2.1: Examples of deep learning based fruit detection.

visual system for citrus picking robots and they achieved higher detection accuracy. Fu et al. [WMZL21] created a real-time apple detection system based on an improved YOLOv3 network. However, deep learning approaches commonly demand high computational resources and extensive datasets for training, which limit their real-time applicability on field robots with limited processing capacity. Some examples of deep learning based fruit detection methodologies are shown in Figure 2.1.

McCool et al. [MPU17] addressed computational constraints by developing mixtures of lightweight CNNs, seeking a balance between accuracy and computational efficiency. Despite their effort, the system still required careful tuning and remained sensitive to environmental changes, potentially affecting detection in field conditions.

In another approach, Kirk et al. [KCM20] and Ganesh et al. [GVBM19] modified input data by incorporating additional color spaces to improve detection accuracy under changing lighting conditions. Kirk et al. used a six-channel input combining RGB and CIELAB color spaces, while Ganesh et al. used RGB and HSV color spaces in a Mask



(a) Reference view to which we align other views

(b) An image obtained by reprojecting 3d points produced by baseline incremental SfM

(c) An image obtained by reprojecting 3d points produced by baseline PnP based SfM

(d) Same image obtained from our pipeline

Figure 2.2: Fruit tracking with Structure from Motion by Roy et al. [RI16]

R-CNN model. Although these methods improved detection in complex scenarios, the additional input dimensions further increased computational demands.

Traditional image processing methods also contribute to fruit detection systems. Bargoti and Underwood [BU17a] used color and texture features for image segmentation to detect apples in controlled lighting. However, their method's performance suffered in dynamic outdoor environments with fluctuating illumination, limiting practical applications. Wendel and Underwood [WUW18] applied hyperspectral imaging, which provided detailed spectral data for improved accuracy but involved high costs and lengthy processing times.

Visual tracking the fruit over time (i.e., over successive image frames) is another key requirement for to develop a robust vision system for autonomous fruit harvesting. Roy et al. [RI16] use apple contours in a Structure from Motion pipeline, to generate dense matches and to reconstruct the apples in 3D (shown in Figure 4.6). Das et al. [DCQ+15] use a Support Vector Machine to detect fruits, and optical flow to associate the fruits between frames. Liu et al. [LCA+18] design a convolutional network to separate fruit and non-fruit pixels and use a Kalman Filter based Hungarian algorithm to correct a Kanade–Lucas–Tomasi (KLT) tracker.

Reviews, such as that by Koirala et al. [KWWM19], highlight challenges such as computational requirements and environmental adaptability in deep learning-based fruit detection. Large, annotated datasets and computational demands remain significant obstacles to real-time deployment on robots with limited resources.

Comparisons to Our Work

While deep learning techniques have advanced fruit detection accuracy, their high computational requirements and environmental sensitivity limit their practical use in field applications. Balancing accuracy with computational efficiency remains challenging, and traditional image processing methods, though less demanding computationally, lack robustness in unstructured environments.

Our perception method addresses the high computational demand, by using a combination of traditional image processing and machine learning for fruit detection and localization. Our method is designed to be robust under varying environmental con-

ditions. By using techniques which are less sensitive to environmental changes, we improve the reliability of fruit detection in outdoor settings.

Our approach also addresses the limitations, who relied on extensive manual annotations and large datasets for training deep learning models. By reducing the dependency on large labeled datasets, we make it easier to adapt the system to new fruit types or environments.

Methods that use additional color spaces during the training of their deep learning process, increase the complexity and computational load of the detection process. Our method operates on standard RGB data, then use the Depth data outside of the training process, which reduces computational overhead. Also, our method avoids the need for complex sensors such as hyperspectral imaging which makes our system accessible and easier to use.

In contrast to previous works that used less accessible methods for fruit tracking, we initially used an off-the-shelf tracking algorithm DeepSORT but encountered increased computational complexity and tracking inaccuracies. To maintain tracking performance, we developed a custom tracking algorithm based on spatial proximity which reduces computational demands while ensuring robust tracking under varying environmental conditions. This approach avoids reliance on complex network training, making it more accessible.

2.2 Motion Planning in Agriculture

Related Work

Motion planning ensures safe maneuvering and precise manipulation within agricultural environments. For dual-arm robots, coordinated control is necessary to perform simultaneous actions, such as grasping and cutting.

Several researchers have recognized the importance of dual-arm coordination in agricultural robotics. Ling et al. [LZG+19] highlighted the importance of coordinating both arms in a dual-arm robot to perform harvesting tasks. Their approach involved planning synchronized trajectories to avoid collisions between the arms and minimize movement time, using predefined motion patterns to optimize the harvesting process within specially arranged orchard settings as shown in Figure 2.3. However, the reliance on predefined motions limited the robot's adaptability to dynamic environments and varying fruit positions. In unstructured environments, where fruit locations and surrounding foliage can change unpredictably, such rigidity can decrease the performance.

Task coordination and scheduling are also important considerations in dual-arm systems. Davidson et al. [DHMK17] proposed a dual-arm coordination strategy to reduce harvesting cycle time by having the picking arm drop the apple into a robotic collector positioned near the detachment point. This approach minimized the time spent on transporting the fruit to a collection bin, highlighting the potential efficiency



Kondo et al. [BVD08]



Silwal et al. [SDK+17]

Figure 2.3: Examples of specially arranged orchards for automated fruit harvesting.

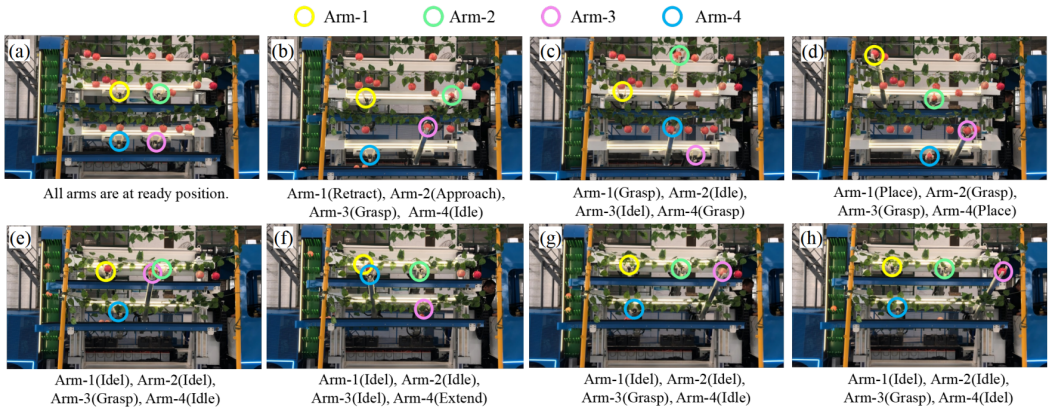
gains from dual-arm coordination. Similarly, Davidson et al. [DHMK17] explored the challenges of coordinating multiple robots operating in a shared workspace for apple harvesting. They implemented scheduling algorithms to optimize the harvesting sequence and reduce idle time among robots. While their research focused on multiple robots rather than a single dual-arm robot, the principles of task coordination and scheduling are directly applicable to dual-arm systems. Their work underscored the complexity of managing simultaneous actions and avoiding interference.

Other researchers concentrated on motion planning strategies for single-arm robots, aiming at optimal sequencing and path planning. Kurtser and Edan [KE20] also focused on planning task sequences for single-arm harvesting robots. They developed algorithms for task allocation and sequencing. Bu et al. [BHC+20] analyzed picking patterns for robotic apple harvesting and proposed strategies to minimize travel distance and time between fruits. Lehnert et al. [LSM+16] developed motion planning algorithms for sweet pepper harvesting using single-arm robots, improving efficiency by considering the robot's kinematics and environmental constraints.

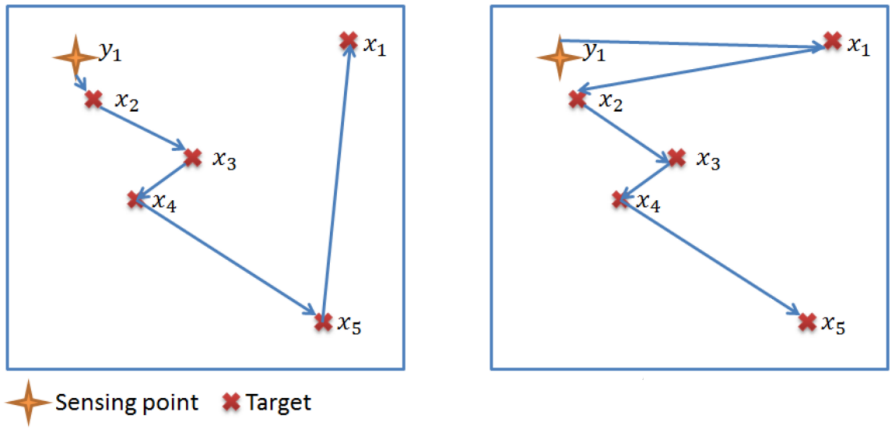
Optimization approaches like the Traveling Salesperson Problem (TSP) have been applied to harvesting sequence planning. Kurtser et al. [KE20] used a single-arm robot and applied a TSP approach to optimize the harvesting sequence, focusing on minimizing the robot arm's travel distance. Similarly, Zeeshan et al. [ZA23] explored several path planning algorithms on a mobile robot to optimize fruit harvesting routes. In recent work, Li et al. [LXQQ23] developed a task planning strategy for a multi-arm harvesting robot, using multi-agent reinforcement learning to optimize the collaboration and operational efficiency among robotic arms in complex orchard environments. Some examples of task coordination and scheduling are shown in Figure 2.4.

Comparison to Our Work

While various studies have addressed aspects of motion planning and coordination in robotic harvesting, many focus on single-arm robots or do not fully exploit the ad-



Li et al. [LXQQ23]



Kurtser et al. [KE20]

Figure 2.4: Examples of task coordination and scheduling.

vantages of dual-arm coordination. Those that consider dual-arm systems often rely on predefined motions or focus on specific tasks, limiting adaptability to unstructured environments. Comprehensive motion planning strategies that address collision avoidance between arms, synchronization of simultaneous tasks, and adaptability to varying environmental conditions remain areas requiring further development to fully realize the potential of dual-arm robotic harvesting robots.

Our motion control strategy advances the state of art by generating trajectories for both arms based on perception data. Our system adapts to the current environment and the precise positions of fruits. This is important for operating in unstructured outdoor environments where fruit locations and environmental conditions can change unpredictably.

Integrating obstacle avoidance and occlusion handling into motion planning ensures that the robot operates in complex environments, without damaging the fruits or plants. Our approach uses dual-arm coordination to perform simultaneous actions effectively. For instance, one arm can remove occluding branches or leaves, while the other arm harvests the fruit. This division of tasks improves the performance, by reducing idle

time and allows continuous operation. We address limitations observed in previous works which did not fully exploit the capabilities of dual-arm robots.

Our motion planner takes into account potential interactions between the two arms, to ensure collision avoidance and coordination. We take into account the kinematic constraints and workspace overlap of both arms, addressing the risk of interference that previous studies did not thoroughly cover.

Our framework considers environmental obstacles and include them into the motion planning. We improve safety in maneuvering and interacting with agricultural environments by continuously updating the robot's understanding of the surroundings through perception data.

2.3 Occlusions Handling in Robotic Harvesting

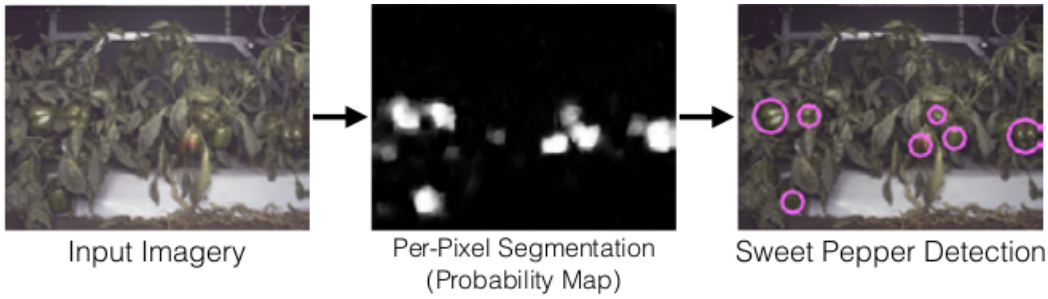
Related Work

Occlusions caused by foliage, branches, and other fruits present significant challenges in robotic fruit harvesting. Fruits are often partially or fully obscured, making accurate detection, localization, and manipulation difficult. Addressing occlusions requires both perception techniques to detect occluded fruits and manipulation strategies to handle or remove occluding objects.

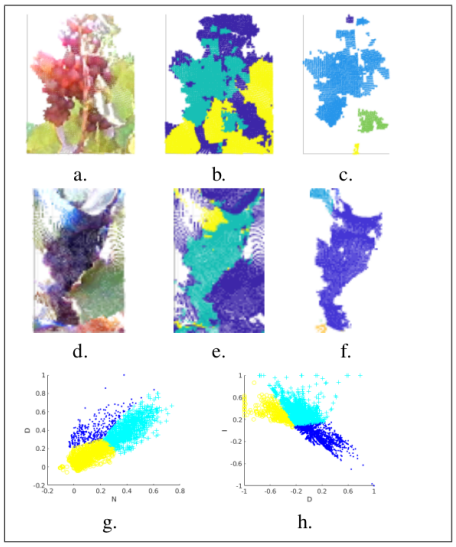
Some researchers have highlighted the persistent challenges of occlusions leading to reduced accuracy in fruit detection. Kurtulmus et al. [KLV14] applied statistical classifiers and neural networks to detect peaches but encountered significant occlusion issues that reduced the overall accuracy and reliability of the system. Similarly, Tian et al. [TYW+19] developed a detection method for apple lesions, which has implications for fruit detection under challenging conditions where environmental factors alter appearance.

Other researchers have focused on perception techniques using deep learning to detect occluded fruits. McCool et al. [MSD+16] developed methods for visual detection of occluded crops using mixtures of lightweight deep neural networks. By combining multiple neural network models, they improved detection performance under occlusions. Similarly, Gong et al. [GWWL22] introduced a high-precision deep learning method for estimating occluded targets. They used deep convolutional neural networks trained on extensive datasets to predict the locations of occluded fruits based on contextual cues, achieving high accuracy in dense foliage environments.

Kang and Chen [KC20] addressed occlusions in apple harvesting by developing vision algorithms based on deep learning. They used deep neural networks to detect apples and depth information to estimate their positions, improving detection accuracy under occlusions. Despite this improvement, they faced challenges with computational efficiency and adaptability to varying environmental conditions. The deep learning models required considerable processing power and needed training on large, diverse datasets. Kurtser et al. [KRR+20] proposed using PointNet, a deep learning architec-



McCool et al. [MSD+16]



Kurtser et al. [KRR+20]

Figure 2.5: Examples of occluded fruit detection.

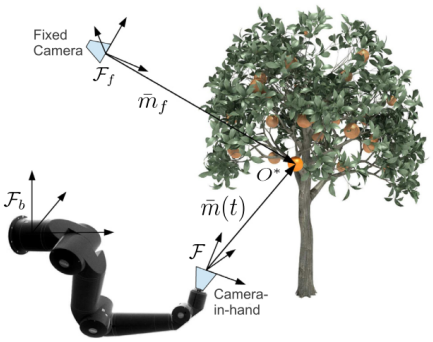
ture designed for point cloud data, along with geometric reasoning for the detection of grapevines from single-frame RGB-D data in outdoor conditions. By analyzing the three-dimensional structure and spatial relationships, they effectively addressed occlusions. However, their method required careful tuning of network parameters, was computationally intensive, and relied on large annotated datasets, which are challenging to obtain.

Multi-modal approaches have been explored to enhance detection under occlusions. Gené-Mola et al. [GMVRP+19] combined RGB and depth data for Fuji apple detection, using data fusion to improve performance in cluttered environments where fruits may be occluded.

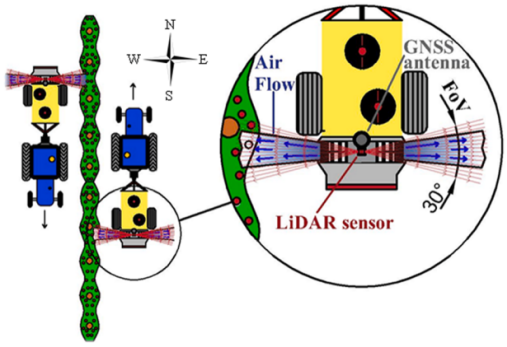
Traditional image processing techniques have also been applied to tackle occlusions. Chaivivatrakul and Dailey [CD14] used texture-based detection to identify fruits in challenging conditions. They focus on the smooth texture patterns characteristic of apples. Lv et al. [JDAWS16] developed a method using dynamic threshold segmentation to identify occluded fruits in natural environments. By adjusting the segmen-



Xiong et al. [XGGF19]



Mehta et al. [MB14]



Gené-Mola et al. [GMVRP+19]

Figure 2.6: Examples of occlusion handling.

tation thresholds based on local image characteristics, they aimed to better separate fruits from the background and occluding elements. These methods struggled in highly complex scenes with severe occlusions and dense foliage, where texture cues were significantly obscured. Luo et al. [LTL+18] used classic image processing algorithms to segment fruits in complex scenes and detect peduncle cutting points for grape harvesting. They used color segmentation, morphological operations, and edge detection to isolate grapes from the background and identify optimal cutting points. However, their method required high-resolution images to capture sufficient detail for accurate segmentation, making the system sensitive to lighting conditions.

Manipulation strategies have been explored to physically handle or remove occluding objects, improving fruit visibility and accessibility. Xiong et al. [XGGF19] designed a specialized gripper capable of physically manipulating occluding leaves to access strawberries. The gripper was equipped with mechanisms to push aside leaves and expose the fruit for harvesting. While this mechanical solution effectively improved visibility for small fruits like strawberries, it may not be suitable for larger fruits or different crop types.

Some studies investigated the use of specialized sensors to improve detection under occlusions. Mack et al. [BU17b] used LiDAR data to reconstruct grape clusters for phenotyping purposes. By scanning grapevines with laser sensors, they obtained detailed 3D models of grape clusters, allowing for accurate measurement and analysis.

Agronomic interventions have been proposed to reduce occlusions and improve fruit visibility. Gené-Mola et al. [GMGA+20] applied airflow using sprayers to diminish

occlusions in fruit trees. By directing air jets towards the foliage, they temporarily displaced leaves and branches, increasing fruit visibility for imaging and harvesting. Xiong et al. [XGGF19] used of active obstacle-separation algorithms to isolate fruits presented only a 50–75% harvest success rate. Some examples of fruit occlusion handling methodologies are shown in Fig 2.5.

Comparison to Our Work

In summary, various approaches have been proposed to address occlusions and improve fruit visibility in robotic harvesting. Deep learning techniques can improve detection under occlusions but often require high computational resources and extensive datasets. Traditional image processing methods are less computationally demanding but may struggle in complex environments with severe occlusions. Manipulation strategies and specialized sensors can improve visibility but may increase system complexity and cost. Agronomic interventions can reduce occlusions but may not be feasible on a large scale or may introduce other challenges.

Our approach addresses occlusions and fruit visibility by integrating both perception and manipulation within the same framework. We introduce an innovative use of view gradients to quantify the degree of occlusion and we identify occluding branches within the robot's field of view. By analyzing the gradients in the depth image obtained from RGB-D sensors, our framework can effectively distinguish between the surfaces of fruits and occluding elements. This allows the robot to not only detect the presence of a fruit but also assess how much of it is visible and which parts are obstructed.

Unlike methods that rely heavily on computationally intensive deep learning models, our approach balances accuracy with computational efficiency. By using RGB-D information, we avoid the need for extensive training datasets and high-end processing hardware, which are often impractical for field deployment on agricultural robots with limited computational resources.

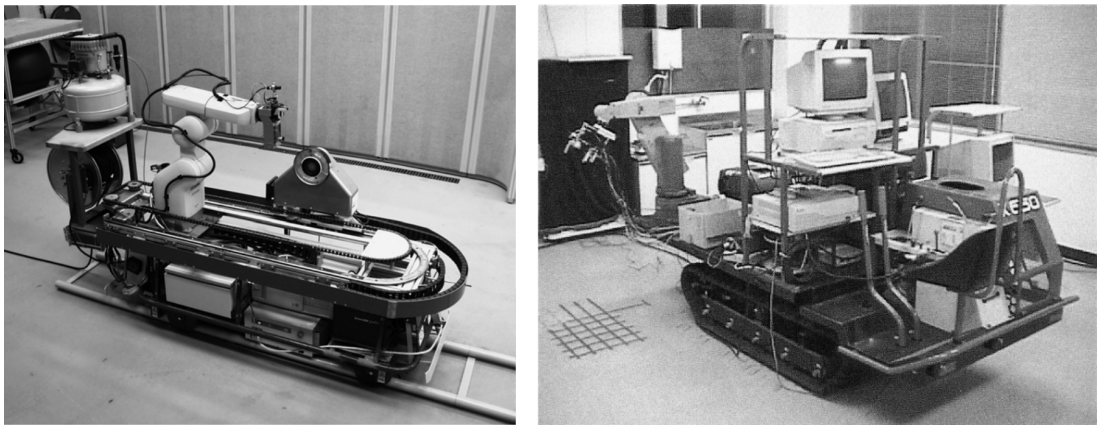
We use the dual-arm configuration to assign specific roles to each arm so that the arms work in coordination and increase harvesting efficiency. One arm is designated to remove occlusions by manipulating branches, while the other arm can focus on harvesting the fruit. This division of tasks allows the robot to handle occlusions and improves on mechanical solutions, which may be unsuitable for different crops or larger fruits.

Our method also avoids the need for agronomic interventions. While such interventions can improve fruit visibility, they introduce additional labor costs or equipment requirements.

2.4 Robotic Harvesting Platforms

Related Work

Early robotic harvesters predominantly used single-arm manipulators due to simpler control mechanisms and lower costs on different fruits such as apples [SDK+17], cherries [TFAI08], citrus [MB14], cucumbers [vHHvT+02], eggplants [HGIT02], strawberries [HSY+10] and sweet peppers [LEM+17]. However, these robots faced significant limitations in dexterity and efficiency, particularly in performing simultaneous actions such as cutting and grasping, which are essential for harvesting fruits that require precise manipulation [KYI+10, DAJJ+11, BVD08]. Some early examples of robotic platforms are shown in Figure 2.7.



Van Henten et al. [vHHvT+02] *Hayashi et al. [HGIT02]*

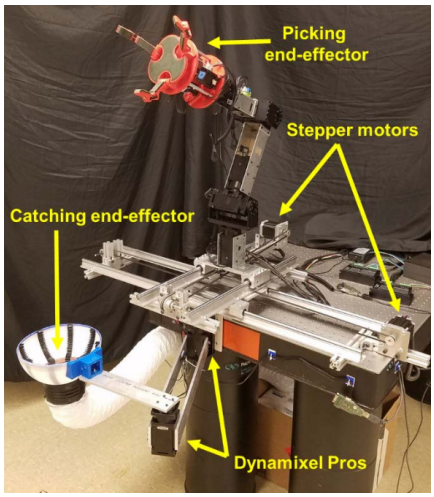
Figure 2.7: Early examples robotic harvesting platforms.

For instance, Hayashi et al. [HGIT02, HSY+10] used single-arm manipulators guided by visual servoing for strawberry harvesting. Their robots struggled with slow operation speeds and difficulties in handling occlusions. Similarly, Tanigaki et al. [TFAI08] introduced a cherry-harvesting robot using a single-arm manipulator but encountered limited success due to challenges in precise positioning and grasping of small fruits in outdoor conditions affected by wind and varying light. Xiong et al. [XGGF19] developed an autonomous strawberry-harvesting robot with a single arm equipped with a cable-driven gripper, achieving a harvest success rate of 41%, which indicates the need for improved manipulation strategies.

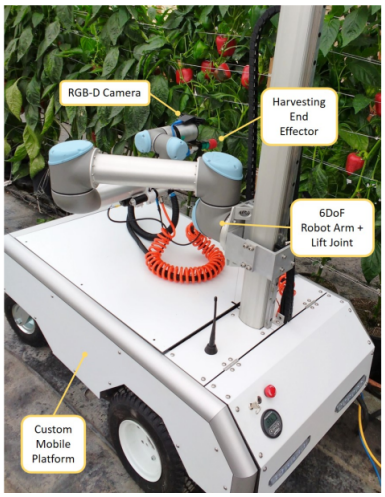
Other single-arm robots faced similar challenges. Kondo et al. [KYI+10] proposed an autonomous tomato harvesting robot with a rotational plucking gripper. Van Henten et al. [vHHvT+02] designed a single-arm robotic cucumber harvester. De-An et al. [DAJJ+11] developed a single-arm robotic system for apple harvesting using a 3D vision system for fruit detection. These approaches was limited by slow processing speeds and difficulties in handling occlusions common in orchards with dense foliage.

Silwal et al. [SDK+17] designed a robotic apple harvester with improved adaptability compared to previous systems. Their single-arm robot achieved a harvesting success rate of 84% under ideal conditions.

These limitations of single-arm robots motivated researchers to explore dual-arm configurations that could mimic the capabilities of human workers, providing greater dexterity and efficiency. Zhao et al. [ZGHL16] introduced one of the early dual-arm robotic systems designed for harvesting tomatoes in greenhouse environments. Their system had one arm equipped with a gripper to hold the fruit and another arm with a cutting device to sever the stem. The robot utilized stereo vision for fruit detection and localization. While demonstrating the potential of dual-arm robots, the system was specifically tailored for tomatoes.



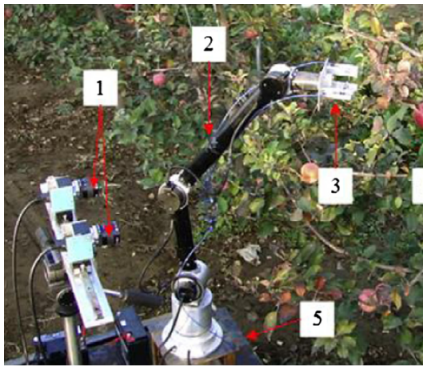
Davidson et al. [DHMK17]



Lehnert et al. [LEM+17]



Xiong et al. [XGGF19]



Zhao et al. [ZGHL16]

Figure 2.8: Modern examples of dual-arm harvesting platforms.

Building upon this concept, Ling et al. [LZG+19] propose a dual-arm system for harvesting tomatoes. The vacuum cup-type tool attached to one of the end-effectors grasps the fruit, while the other arm (equipped with a cutter) cuts the stem. Sepulveda et al. [SFN+20] introduce a human-like dual-arm aubergine harvesting robot. The planning algorithm decides the movement mode, depending on the fruit location: either

simultaneous harvesting of two fruits or harvesting a single fruit with one arm. Besides, if an aubergine is occluded, they apply a collaborative behavior of the arms to clear occlusions. Xiong et al. [XGGF19] proposed a low-cost dual-arm harvesting platform to increase the efficiency under occlusions. Davidson et al. [DHMK17] propose a dual-arm coordination strategy for fruit collection, to reduce the harvesting cycle time. Rather than displacing the fruit to a bin, the picking arm drops the apple into a robotic collector near the fruit detachment position. Some modern examples of robotic platforms are shown in Figure 2.8.

Comparison to Our Work

Our research advances the field by focusing on the development of a dual-arm robot designed for unstructured environments, and capable of handling various fruit types.

By using dual-arm coordination, we address the limitations of single-arm robots. The dual-arm configuration allows the robot to replicate harvesting actions more effectively, and perform simultaneous tasks such as grasping and cutting simultaneously.

We address occlusions with a dual-arm configuration, a significant improvement over previous systems that struggled with occlusions, required manual interventions.

Our robot is designed to handle the complexities associated with multiple fruit types, adapting methods that address variability in fruit characteristics and structural differences across crops. This versatility sets our work apart from previous studies, which were often limited to specific crop types.

2.5 Synthesis

Early robot harvesters primarily used single-arm manipulators due to their simpler control mechanisms and lower cost [SDK+17, MB14, vHHvT+02, HGIT02, HSY+10, LEM+17]. While these systems demonstrated the feasibility of robotic harvesting, they faced limitations in dexterity, and in the ability to perform synchronous actions such as cutting and grasping [TFAI08, XGGF19, KYI+10]. The inability to handle the complex tasks required for harvesting certain fruits, highlighted the need for more advanced robot configurations.

The exploration of dual-arm robots provided a promising solution to these limitations. Researchers like Zhao et al. [ZGHL16] and Ling et al. [LZG+19] introduced dual-arm robots capable of mimicking human harvesting actions. However, these systems were often confined to controlled environments, and lacked adaptability to different crops.

Perception techniques have evolved significantly, with the introduction of deep learning methods improving fruit detection accuracy [SGD+, BU17a, MPU17] and tracking [RI16, LCA+18]. Yet, the reliance on computationally intensive algorithms and the need for large datasets limit their practicality.

Motion planning and control strategies are important for the effective operation of robotic harvesters. While studies have addressed dual-arm coordination and obstacle avoidance [LZG+19, DHMK17, MB14], many rely on predefined motions or lack integration of online perception, limiting their adaptability. In agricultural environments, integrating obstacle avoidance within motion planning also presents challenges for safe maneuvering. However, most existing methods primarily target obstacle avoidance and often miss the additional challenges of manipulation tasks during harvesting.

Handling occlusions remains a challenge, since fruits are often partially or fully obscured by foliage or by other fruits. Methods based on deep learning or specialized sensors have been proposed [MSD+16, KRR+20], but these method often come with increased computational cost. Other innovative approaches including mechanical solutions [XGGF19], vision algorithms [KC20], and agronomic interventions [GMGA+20]. However, these methods either lack adaptability to various crops, require expensive equipment, or involve additional labors, which are not ideal for practical implementation.

The current advancements in robotic fruit harvesting have established a strong foundation and show the potential of robots to handle agricultural tasks. Yet, significant challenges remain in terms of adaptability, efficiency, handling occlusions.

Our research contributes to the field by addressing these limitations through the development of a dual-arm robotic platform, designed for unstructured environments and adaptable to various fruit types, such as apples and oranges. We focus on integrating perception techniques, to localize and tracking fruits and introduce innovative methods for occlusion handling.

Furthermore, our motion planning and control strategies uses perception data to generate trajectories for both arms, using obstacle avoidance and occlusion handling directly in the motion planning process. This approach allows the robot to adapt to changing environments, to perform simultaneous actions, and to maneuver around complex agricultural settings safely.

By building upon and extending the current state of the art, our work aims to advance the capabilities of dual-arm robotic fruit harvesting robots, bringing them closer to practical. We address the key challenges identified in the literature to develop a platform which meets the demands of modern farming practices.

CHAPTER 3

Initial Development of Dual-Arm Harvesting Robot

3.1 Motivation and Challenges

Autonomous fruit harvesting robots have the potential to improve efficiency and reduce reliance on manual labor in agriculture. However, single-arm robotic systems often lack the dexterity and redundancy needed for complex tasks in unstructured environments. A dual-arm configuration offers greater capabilities, allowing simultaneous cutting and collecting of fruits while avoiding obstacles such as branches and other fruits.

Developing such a system presents several key challenges: robust perception and localization in dynamic, cluttered environments; tracking and state estimation of fruits; hierarchical motion planning and control; and collision avoidance in complex agricultural settings.

Firstly, fruits in unstructured agricultural environments are often partially obscured by foliage and vary widely in appearance due to differences in size, shape, color, and lighting conditions. A robust perception system that accurately detects and localizes fruits is essential. RGB-D images combined with deep learning techniques, specifically Convolutional Neural Networks (CNNs), improve detection accuracy under varying conditions. Depth information provides spatial context, allowing precise estimation of the three-dimensional positions of fruits within the workspace.

Secondly, accurate tracking of detected fruits across multiple frames is crucial for precise harvesting. A tracking algorithm capable of handling occlusions and changing environmental conditions ensures continuous state updates for each fruit. Techniques such as data association and predictive models like Kalman filters estimate fruit positions in real time and provide reliable inputs for motion planning.

Thirdly, the dual-arm configuration introduces complexity in planning and executing coordinated movements. A hierarchical motion planning approach enables the robot to navigate efficiently around obstacles and position both arms accurately for simultaneous cutting and collecting. In a cooperative task space representation, tasks are defined in terms of both absolute positions and relative configurations between the arms. Prioritizing tasks hierarchically ensures that critical objectives, such as collision avoidance, are satisfied before optimizing for efficiency. Hierarchical Quadratic Programming (HQP) handles multiple objectives and constraints, adjusting movements in response to environmental uncertainties and task requirements.

Lastly, collision avoidance in cluttered environments is a significant challenge. The robot must prevent damage to itself and the plants by adapting to changes in the environment and avoiding potential collisions with branches, fruits, or the tree trunk. Sensor data from the RGB-D camera creates a dynamic representation of the environment. Motion planning algorithms generate collision-free paths that consider the robot's kinematics and the spatial distribution of obstacles. Methods like velocity damping help find safe trajectories.

Addressing these challenges advances the development of dual-arm robots capable of autonomously harvesting fruits in diverse agricultural environments. This research combines robust perception, tracking algorithms, hierarchical motion planning, and collision avoidance to handle the complexities of agricultural settings effectively. The methods have been validated within a laboratory setting, demonstrating potential for practical application.

In developing our method, we make the following assumptions to simplify the problem and focus on the core challenges:

- *Hypothesis 1:* Fruits are assumed to have a perfect ellipsoidal shape, to simplify the modeling.
- *Hypothesis 2:* Fruit localization is based on the pixel positions of the detected bounding box of the fruit to calculate the fruit centroid. Without accurate pixel data, precise localization is challenging.
- *Hypothesis 3:* Fruit localization uses depth data from the camera to project the centroid into 3D space. In practice, depth data at the centroid can sometimes be noisy or occluded. This may results in 3D localization errors.

3.2 Perception

Here, we present our perception system for detecting, tracking and localizing fruits, as well as the fruit selection strategy, and the tree location estimation methodology.

In our work, we consider fruits as ellipsoids (see Figure 3.2), which appear in a 2D image as ellipses with axes aligned with the image height and width. The ellipsoid surrounding the fruit has a circular symmetry i.e. it has two equal semi-diameters

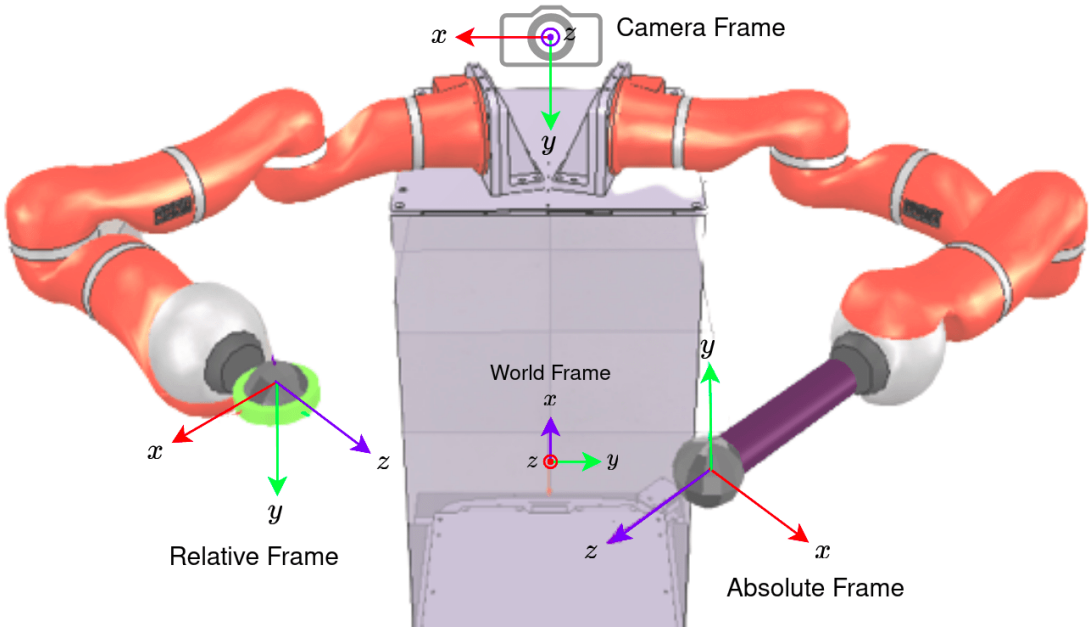


Figure 3.1: Coordinate frames definitions. The Absolute Frame is linked to the fixed World Frame, whereas the Relative Frame is linked to the moving Absolute Frame.

along the x and z axes in the camera frame. Since most round fruits (e.g., apples, oranges) have this property, we can assume that our approach is valid in those cases.

Coordinate Frames

Figure 3.1 illustrates the coordinate frames used in our system. The camera frame $\mathcal{F}_{cam}$ is the reference frame in which the fruit positions are computed. The robot uses these coordinates for motion planning and control.

3.2.1 Fruit Localization

In this subsection, we detail the process of fruit localization, which involves detecting fruits in images, assigning unique identifiers to them through tracking, and calculating their 3D spatial positions in the camera frame, as illustrated in Figure 3.1.

Fruit Detection

The first step in fruit localization is fruit detection, which involves identifying fruits in the image and determining the pixel coordinates of the corners of the bounding rectangles which enclose them. We use a deep learning object detection algorithm, YOLOv5 [Joc20], due to its balance of speed and accuracy, making it suitable for real-time applications.

To adapt YOLOv5 to our specific task of detecting fruits, we perform domain adaptation and transfer learning. We initialize the network with weights pre-trained

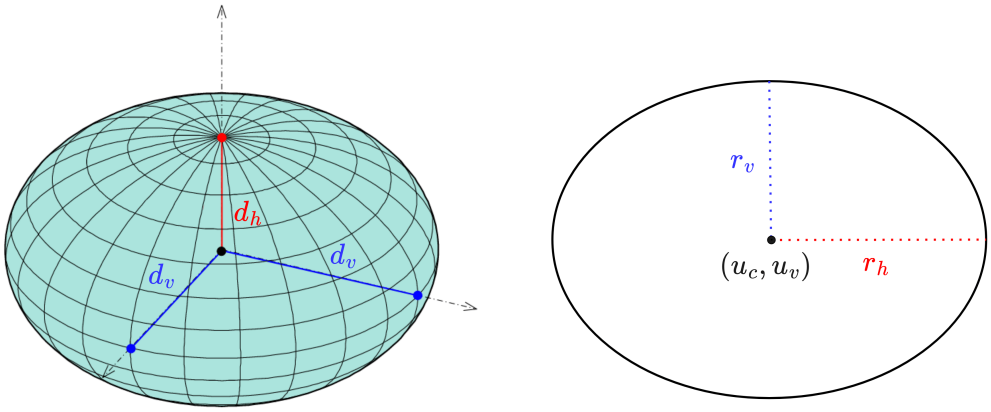


Figure 3.2: Hypothesis 1 of fruit as an ellipsoid (left) and its projection as an ellipse in the image plane (right).

on the COCO dataset [LMB+14], which allows us to leverage existing feature extractors. We then fine-tune the network using 300 images of oranges from the AgriPlant Dataset [POS+17]. We split the dataset into 70% for training and 30% for validation to assess the performance of the model.

Fruit Tracking

Once the fruits are detected in individual frames, we need to track them across multiple frames, to maintain consistent identification. This is essential for correlating fruit data over time and for determining whether newly detected fruits correspond to existing ones. We use the DeepSORT algorithm [WBP17] for this purpose. DeepSORT is an improved version of SORT [BGO+16], which enhances frame-to-frame association using a deep association metric based on appearance features.

DeepSORT combines motion and appearance information to associate detections across frames. It uses a Kalman filter to predict the position of each fruit in the next frame and a deep neural network to extract appearance features for each detected fruit. The algorithm then matches detections to existing tracks by minimizing a combined cost function based on the Mahalanobis distance [Mah18] of the motion model and on the cosine distance of the appearance features.

We use the OSNet ReID model [ZYCX21] as the deep appearance descriptor due to its lightweight architecture and accuracy. OSNet is pre-trained on the ImageNet dataset [DDS+09], to extract features from various objects, which helps in distinguishing fruits, based on their appearance.

3D Fruit Localization

With the detected fruits and their assigned IDs, we proceed to compute their 3D positions in the camera frame by incorporating depth information from the RGB-D camera. Let (u_{TL}, v_{TL}) and (u_{BR}, v_{BR}) denote the pixel coordinates of the top-left and

bottom-right corners of the bounding box for each detected fruit.

The pixel coordinates of the fruit center are calculated as:

$$(u_C, v_C) = \left(\frac{u_{TL} + u_{BR}}{2}, \frac{v_{TL} + v_{BR}}{2} \right). \tag{3.1}$$

Given the fruit center, we compute the horizontal and vertical semi-axes of the ellipse surrounding the fruit as:

$$r_h = u_{BR} - u_C, \quad r_v = v_{BR} - v_C. \tag{3.2}$$

These semi-axes represent the distances from the center of the fruit to its edges along the horizontal and vertical directions in the image plane.

To project a pixel (u, v) from the image plane to a 3D point in the camera frame, we use the function proj:

$$(x, y, z) = \text{proj}(u, v), \tag{3.3}$$

which utilizes the camera's intrinsic parameters and the depth information to compute the 3D coordinates.

Under *Hypothesis 1* that fruits are ellipsoidal and have circular symmetry, the point visible at the center pixel (u_C, v_C) is the point on the fruit's surface closest to the camera's optical center. We denote its depth as $z_{\min}$. Yet, the actual center of the fruit is located further along the line of sight. To estimate the depth of the fruit center, z_f, we use the horizontal semi-axis and compute the distance from the fruit surface to its center in 3D space.

In summary, the depth of the fruit center is calculated by:

$$z_f = z_{\min} + \text{dist}\left(\text{proj}(u_C, v_C), \text{proj}(u_C + r_h, v_C)\right), \tag{3.4}$$

where $\text{dist}(\cdot, \cdot)$ denotes the Euclidean distance between two points in 3D space.

This calculation assumes that the fruit has a circular symmetry along the horizontal axis, and the distance from the surface point at (u_C, v_C) to the center is equal to the 3D distance corresponding to the horizontal semi-axis.

The 3D coordinates of the fruit center are then given by:

$$(x_f, y_f, z_f) = (x_{\min}, y_{\min}, z_f), \tag{3.5}$$

where $(x_{\min}, y_{\min}, z_{\min}) = \text{proj}(u_C, v_C)$.

Similarly, the distance from the fruit center to the peduncle (the stem attaching the fruit to the tree) is equal to the 3D transformation of the vertical semi-axis:

$$d_v = \text{dist}\left(\text{proj}(u_C, v_C), \text{proj}(u_C, v_C + r_v)\right). \tag{3.6}$$

This provides an estimate of the distance from the fruit center to its top along the vertical axis, which is useful for determining the cutting point.

3.2.2 Fruit Selection

Fruit selection involves determining the best candidate fruit for harvesting. For each detected fruit, we compute the 3D distance associated with its tracking IDs. The distance from the fruit to the optical center of the camera, denoted as d_{ID}, is calculated using the color and depth information acquired as detailed in Section 3.2.1.

For each image frame, we maintain a list of fruits $\mathbf{F}$ with their associated IDs and distances to the camera. To ensure consistent tracking, we update the list only if the position error between the current and previous state of a fruit ID is smaller than a predefined threshold d_{update}. If the position error exceeds d_{update}, we assume that the fruit's ID might have been incorrectly assigned due to occlusions or detection errors, and we retain the previous state. This approach adds robustness to temporary ID mismatches.

To manage the list efficiently, we remove fruit IDs if they are not detected in the scene for a specified number of consecutive frames defined by $n_{consecutive}$. This prevents the list from being cluttered with obsolete entries corresponding to fruits that are no longer visible or have already been harvested.

The parameters used in fruit selection are:

- d_{update}: the maximum allowable position error between the current and previous state of a fruit ID for updating.
- $n_{consecutive}$: the number of consecutive frames a fruit must be absent from the scene, before its ID is removed.

After updating the list, we select – for harvesting – the fruit with the smallest distance from the camera. This strategy prioritizes fruits which are closer to the robot, potentially reducing the movement required and increasing the harvesting efficiency.

The selected fruit, which is closer to the camera $F_{closest}$ is defined by:

$$F_{closest} = \arg \min_{(d_{ID}) \in \mathbf{F}} d_{ID}. \tag{3.7}$$

3.2.3 Tree Localization

Accurate estimation of the tree's position is important to avoid collisions between the robot and the tree, which could damage the plant or the robot during the harvesting process. Unlike fruit detection, which requires machine learning, due to the variations in fruit appearance and high levels of occlusion, we can use a simpler and more computationally efficient method for tree localization, based on color and depth information.

We generate a segmentation mask in the HSV color space, combined with a depth mask to separate the tree branches and trunk from the rest of the scene. The HSV color space is chosen since it represents color with a single hue value, making it more robust to lighting variations than RGB color space.

First, we convert the RGB image from the camera into the HSV color space. We then apply predefined thresholds for Hue, Saturation, and Value to create a binary mask that highlights the pixels corresponding to the tree. These thresholds are determined experimentally by observing the tree under different lighting conditions to ensure that the mask is robust.

Next, we create a depth mask by applying a depth threshold to the depth image obtained from the RGB-D camera. This depth mask filters out pixels which are beyond a certain distance from the camera, ensuring that only nearby objects (potentially part of the tree) are considered.

We combine the HSV mask and the depth mask using a logical AND operation, resulting in a final mask which isolates the tree trunk and branches, while excluding other objects and background elements. This combined mask effectively highlights the tree in the scene.

To estimate the position of the tree, we compute the mean of the pixel coordinates (u, v) and the depth values z of the pixels that remain, after applying the combined mask. The mean depth z_t is calculated as:

$$z_t = \frac{1}{N} \sum_{i=1}^{N} z_i, \tag{3.8}$$

where N is the number of pixels in the combined mask, and z_i is the depth value of the i-th pixel.

The mean pixel coordinates (u_t, v_t) are similarly computed as:

$$u_t = \frac{1}{N} \sum_{i=1}^{N} u_i, \quad v_t = \frac{1}{N} \sum_{i=1}^{N} v_i. \tag{3.9}$$

We then project these mean pixel coordinates and the mean depth into 3D coordinates in the camera frame, using the projection function defined in (3.3):

$$(x_t, y_t, z_t) = \text{proj}(u_t, v_t). \tag{3.10}$$

This gives us the estimated 3D position of the tree $\mathbf{P}_t = (x_t, y_t, z_t)$ in the camera frame.

3.2.4 Synthesis

In this chapter, we presented a comprehensive perception system, which integrates deep learning and image processing techniques to address the challenges of fruit detection, tracking, localization, and tree estimation in agricultural environments.

For **fruit detection**, we used the YOLOv5 object algorithm due to its balance of speed and accuracy, which makes it suitable for real-time applications. By performing domain adaptation and transfer learning, we fine-tuned the network with a dataset specific to our task, which allowed the model to effectively detect fruits under varying lighting conditions and partial occlusions.

For **fruit tracking**, we applied the DeepSORT algorithm to maintain consistent identification of each fruit across multiple frames. DeepSORT combines motion and appearance information, using a Kalman filter for motion prediction and a deep neural network for extracting appearance features. This approach ensures robust tracking even when fruits experience occlusions or changes in viewpoint. This robustness is crucial for accurate state estimation over time.

For **fruit localization**, we used the depth information from the RGB-D camera, to calculate precise 3D positions of the fruits in the camera frame. By approximating fruits as ellipsoids with circular symmetry, we developed a method to estimate the depth of the fruit centers and the distance to the peduncle. This information is vital for the robot to perform precise motion planning and to execute harvesting actions accurately.

In the **fruit selection** process, we determined the optimal candidate for harvesting by computing the distances of detected fruits to the camera's optical center. We maintained a list of fruits with their associated IDs and updated it based on the position error threshold, to ensure robustness against temporary ID mismatches, caused by occlusions or detection errors. By selecting the fruit closest to the robot, we aimed to improve harvesting efficiency and to reduce unnecessary movements.

For **tree localization**, we implemented a computationally efficient method using HSV color space segmentation, combined with depth masking. By converting RGB images to the HSV color space and applying carefully chosen thresholds, we isolated the pixels corresponding to the tree trunk and branches. The depth mask further refined this, by excluding pixels beyond a certain distance. Calculating the mean position of these pixels provided an estimate of the tree's location in the camera frame. This estimation is essential for collision avoidance, allowing the robot to move safely near the tree and to prevent damage to both the plant and robot.

The perception system effectively addresses the complexities of unstructured agricultural environments by providing necessary information for autonomous fruit harvesting. The system enables precise and safe robot navigation and manipulation. These capabilities form a solid foundation for the subsequent motion control stage.

3.3 Motion Control

For a dual-arm fruit harvesting robot, both manipulator arms need to approach fruits located at different positions with high accuracy and flexibility, while safely manipulating the objects. To this end, we design a dual-arm control strategy based on cooperative task representation and Hierarchical Quadratic Programming (HQP).

3.3.1 Dual-Arm Cooperative Control

We model our fruit-picking task using the cooperative task representation framework proposed by Chiacchio et al. [CCS96], which fully characterizes the operational

space for dual-arm control. This representation allows us to specify the robot's operations in terms of an *absolute task*, with pose attached to a fixed world frame, and a *relative task*, attached to the absolute task, as shown in Figure 3.1.

The main advantages of using this representation are:

- It allows the adaptation of control algorithms (e.g., quadratic programming approaches) originally designed for single manipulators, to dual-arm robots;
- It facilitates safe dual-arm object manipulation, without colliding with obstacles or damaging the fruits.

Our tasks include a cutting phase, to detach the fruit from its stem and a collecting phase, to catch the falling fruit. We consider this representation suitable for our tasks, since the cutting operation is best described in a fixed frame (the absolute task), while the collecting action is more appropriately described relative to the cutting arm (the relative task).

We define fruit cutting as the *absolute task*, which controls the motion of the robot in the workspace, and fruit collecting as the *relative task*, which regulates the relative motion between the two arms.

Let us formally define these absolute and relative tasks. We define a desired pose of the absolute task $\mathbf{x}_a = [x_a, y_a, z_a, \phi_a, \theta_a, \psi_a]^\top$ for the cutting (left) arm in a fixed frame (in our case, the camera frame $\mathcal{F}_{\text{cam}}$), and a desired pose of the relative task $\mathbf{x}_r = [x_r, y_r, z_r, \phi_r, \theta_r, \psi_r]^\top$ for the collecting (right) arm with respect to the frame attached to the cutting (left) arm. We use the fixed-angle (Tait–Bryan) convention for angle representation, where ϕ, θ, and ψ correspond to roll, pitch, and yaw angles, respectively.

3.3.2 Objective Tasks

To ensure safe and efficient harvesting, we guide the robot by defining some intermediate tasks: *Approach*, *Harvest*, and *Leave*.

- *Approach*: The robot moves to a position near the target fruit, maintaining a safe distance to avoid any potential collisions with the fruit or branches. This task prepares the robot for the precise movements required during harvesting.
- *Harvest*: The robot executes the harvesting operation, moving its end-effectors to the precise locations necessary to detach and collect the fruit. This task ensures the cutting tool is correctly positioned at the fruit's peduncle, while the collecting tool is placed underneath, to catch the falling fruit.
- *Leave*: After harvesting, the robot moves away from the harvested fruit to a safe distance, preventing any damage to the tree or to itself, while it prepares for the following operations.

Once we select the fruit to harvest, as described in Section 4.2.2, both arms should move towards that fruit. During harvesting, the tip of the cutting tool should be positioned at the peduncle, to detach the fruit, while the center of the collecting tool should be slightly below the fruit to catch it, when it falls. We compute the peduncle location using the vertical semi-axis length d_v and the fruit position $\mathbf{P}_f$, as explained in Section 3.2.1.

As for the orientations, we do not impose any hard constraint on θ_a, to maximize the reachable points in the workspace, since cutting only requires the tip of the tool to reach the peduncle. However, θ_r is constrained to $\pi/2$ radians, resulting in a $\pi/2$ radian angle between the two arms at all times, to prevent collisions between the arms. Other orientations are set to remain unchanged from the robot's default pose with arms positioned in front.

We define the *Harvest* poses to reach a fruit relative to the camera frame by:

$$\mathbf{x}_a^{\text{Harvest}} = \begin{bmatrix} x_f \\ y_f - d_v - d_s \\ z_f \\ 0 \\ * \\ \pi \end{bmatrix} \quad \mathbf{x}_r^{\text{Harvest}} = \begin{bmatrix} 0 \\ -d_v - d_s \\ 0 \\ 0 \\ \pi/2 \\ \pi \end{bmatrix} \tag{3.11}$$

where:

- (x_f, y_f, z_f) is the fruit position in the camera frame $\mathcal{F}_{\text{cam}}$;
- d_v is the vertical semi-axis length of the fruit ellipsoid, representing the distance from the fruit center to the peduncle, see Section 3.2.1;
- d_s is a safety distance, preventing the cutting tool from damaging the fruit during the approach;
- $*$ indicates a free task component.

For the *Approach* and *Leave* tasks, we adjust the robot's position to ensure safe movement toward and away from the fruit. The desired poses for these tasks are similar to those in equation (3.11), with modifications to the z-coordinate to control the distance along the approach axis.

During the *Approach* task, the robot moves to a position at a distance d_z away from the fruit along the z-axis. The desired poses are:

$$\mathbf{x}_a^{\text{Approach}} = \begin{bmatrix} x_f \\ y_f - d_v - d_s \\ z_f + d_z \\ 0 \\ * \\ \pi \end{bmatrix}, \quad \mathbf{x}_r^{\text{Approach}} = \begin{bmatrix} 0 \\ -d_v - d_s \\ d_z \\ 0 \\ \pi 2 \\ \pi \end{bmatrix} \tag{3.12}$$

where d_z is a positive parameter defining the distance from the fruit before initiating the harvesting phase.

After the fruit is harvested, the robot executes the *Leave* task by moving away from the fruit along the z-axis to the same distance d_z to ensure clearance from the tree and preparing for the next operation. The desired poses are:

$$\mathbf{x}_a^{\text{Leave}} = \begin{bmatrix} x_f \\ y_f - d_v - d_s \\ z_f + d_z \\ 0 \\ * \\ \pi \end{bmatrix}, \quad \mathbf{x}_r^{\text{Leave}} = \begin{bmatrix} 0 \\ -d_v - d_s \\ d_z \\ 0 \\ \pi 2 \\ \pi \end{bmatrix} \tag{3.13}$$

Note that the *Approach* and *Leave* poses are identical in this formulation.

3.3.3 Hierarchical Tasks

In the cooperative dual-arm task representation, both the relative and absolute tasks must be executed simultaneously. Yet, it is important to prioritize the absolute task over the relative one. This is because fulfilling the absolute task is crucial; the cutting arm's position must be more precise than that of the collecting arm, to ensure successful detachment of the fruit.

To manage this task prioritization, we use Hierarchical Quadratic Programming (HQP), as described in [TNF+22]. HQP involves solving a sequence of quadratic programming problems, where the solution space of a given task is restricted to the null space of higher-priority tasks. This ensures that lower-priority tasks do not interfere with the execution of higher-priority ones.

The relationship between the joint velocities $\dot{\mathbf{q}}$ and the task space velocities $\dot{\mathbf{x}}$ is given by:

$$\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}, \tag{3.14}$$

where:

- $\dot{\mathbf{x}}$ is the task space velocity vector,
- $\mathbf{J}$ is the Jacobian matrix of the robot,
- $\dot{\mathbf{q}}$ is the joint velocity vector.

The inverse kinematics module based on HQP, outputs the joint velocity vector $\dot{\mathbf{q}}$ to be sent to the robot actuators. The HQP formulation includes a set of hard constraints to be fulfilled at all times, defined both at the joint space level (e.g., joint limits, velocity limits) and at the task space level (e.g., collision avoidance).

We define the set of constraints in the standard form of linear inequality equations:

$$\mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}, \tag{3.15}$$

where:

- $\mathbf{A}$ is the matrix of linear coefficients representing the constraints.
- $\mathbf{b}$ is a column vector in the inequality constraints.

Considering the cooperative dual-arm task space and the constraints, we solve the highest-priority (absolute) and lower-priority (relative) tasks through the following optimization problems:

For the absolute task, we solve the optimization problem:

$$\dot{\mathbf{q}}_a^* = \arg\min_{\dot{\mathbf{q}}} \|\mathbf{J}_a\dot{\mathbf{q}} - \dot{\mathbf{x}}_a\|_2^2, \tag{3.16}$$

subject to:

$$\mathbf{A_a}\dot{\mathbf{q}} \leq \mathbf{b_a}, \tag{3.17}$$

and For the relative task, we solve:

$$\dot{\mathbf{q}}_r^* = \arg\min_{\dot{\mathbf{q}}} \|\mathbf{J}_r\dot{\mathbf{q}} - \dot{\mathbf{x}}_r\|_2^2, \tag{3.18}$$

subject to:

$$\mathbf{A_r}\dot{\mathbf{q}} \leq \mathbf{b_r}, \tag{3.19}$$
$$\mathbf{J}_a\dot{\mathbf{q}} = \mathbf{J}_a\dot{\mathbf{q}}_a^*, \tag{3.20}$$

where:

- $\mathbf{q}$, $\mathbf{q_a}$, $\mathbf{q_r}$ are all the joints associated with the robot, joints related to the absolute task, joints related to the relative task. $*$ denotes the desired joints.
- $\mathbf{J}_a$, $\mathbf{J}_r$ are the Jacobian matrices corresponding to the absolute, relative tasks.
- $\dot{\mathbf{x}}_a$, $\dot{\mathbf{x}}_r$ are the desired velocity of the absolute, relative tasks.
- $\mathbf{A_a}$, $\mathbf{A_r}$ and $\mathbf{b}_a$, $\mathbf{b}_r$ represent the inequality constraints of absolute, relative tasks.

The constraint $\mathbf{J}_a\dot{\mathbf{q}} = \mathbf{J}_a\dot{\mathbf{q}}_a^*$ ensures that executing the relative task does not interfere with the higher-priority absolute task.

3.3.4 Kinematic Constraints

The inequality constraints $\mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}$ represent the physical and safety limitations of the robotic system. These constraints include:

Joint Velocity Limits: Each joint velocity $\dot{q}_i$ must satisfy:

$$\dot{q}_{i,\min} \leq \dot{q}_i \leq \dot{q}_{i,\max}, \tag{3.21}$$

where $\dot{q}_{i,\min}$ and $\dot{q}_{i,\max}$ are the minimum and maximum allowable velocities for joint i.

Joint Position Limits: To prevent joints from exceeding their position limits, we convert position constraints into velocity constraints:

$$\frac{q_{i,\min} - q_i}{\Delta t} \leq \dot{q}_i \leq \frac{q_{i,\max} - q_i}{\Delta t}, \tag{3.22}$$

where $q_{i,\min}$ and $q_{i,\max}$ are the position limits, q_i is the current joint position, and Δt is the control loop sampling time.

Task Space Velocity Limits: To maintain safe operation, we limit the end-effector velocities:

$$\dot{\mathbf{x}}_{\min} \leq \mathbf{J}_{\text{ee}}\dot{\mathbf{q}} \leq \dot{\mathbf{x}}_{\max}, \tag{3.23}$$

where $\mathbf{J}_{\text{ee}}$ is the end-effector Jacobian, and $\dot{\mathbf{x}}_{\min}$, $\dot{\mathbf{x}}_{\max}$ are the minimum and maximum allowable end-effector velocities.

3.3.5 Collision Avoidance Constraints

To prevent collisions with obstacles and between the robot's arms, we formulate collision avoidance constraints for two types of potential collisions:

- **Collisions with External Objects:** These occur when parts of the robot approach obstacles in the environment.
- **Self-Collisions:** These happen when different parts of the robot come too close to each other during operation.

For both collision types, we use Velocity Damping to avoid unsafe regions. The obstacle avoidance strategy is expressed as linear constraints using the Velocity Damper proposed by Faverjon and Tournassoud [FT87]. The Velocity Damper defines a permissible rate of approach between the robot and obstacle to ensure that the robot slows down as it gets closer to the obstacle.

The Velocity Damper is defined by:

$$\dot{d}^* = -\xi \frac{d - d_s}{d_i - d_s}, \quad d \leq d_i, \tag{3.24}$$

where:

- $\dot{d}^*$ is the desired rate of change of the distance d between the robot and the obstacle.

- d_i is the activation distance, beyond which the collision avoidance constraint is inactive.

- d_s is the safety distance, below which the robot must not pass.

- ξ is a parameter that determines how quickly the robot decelerates as it approaches the obstacle.

The derived inequality constraint is given by:

$$\dot{d} \geq \dot{d}^*, \tag{3.25}$$

which ensures that the actual rate of approach $\dot{d}$ does not exceed the permissible rate $\dot{d}^*$. When two bodies are separated by a distance smaller than d_i, their relative speed becomes bounded, and the robot's motion is slowed, until safety distance d_s is reached. As soon as d becomes larger than d_i, the constraint is inactive, and the robot can move freely.

Collision with External Objects

Let d be the distance between a point on the robot and an external obstacle, measured along the vector $\mathbf{n}$ pointing from the obstacle to the robot. When $d \leq d_i$, where d_i is the influence distance, we define the constraint:

$$\mathbf{n}^\top \mathbf{J}_{\text{obj}} \dot{\mathbf{q}} \geq -\xi \left(\frac{d - d_s}{d_i - d_s} \right), \tag{3.26}$$

where:

- $\mathbf{J}_{\text{obj}}$ is the Jacobian mapping joint velocities to the velocity of the point on the robot.

- d_s is the safety distance.

- ξ is a positive gain controlling the deceleration rate as the robot approaches the obstacle.

This constraint slows and stops the robot's motion toward the obstacle as it nears the safety distance d_s, preventing a collision.

Self-Collisions

For self-collisions, considering two points on the robot with distance d and direction $\mathbf{n}$, we define:

$$\mathbf{n}^\top(\mathbf{J}_1 - \mathbf{J}_2)\dot{\mathbf{q}} \geq -\xi\left(\frac{d - d_s}{d_i - d_s}\right), \tag{3.27}$$

where:

- $\mathbf{J}_1$ and $\mathbf{J}_2$ are the Jacobians for the two points.
- The term $(\mathbf{J}_1 - \mathbf{J}_2)$ represents their relative motion.

This constraint prevents different parts of the robot from coming closer than d_s, avoiding self-collisions.

3.3.6 Optimization Problem

These optimization problems aim to find joint velocities $\dot{\mathbf{q}}$ that minimize the error between the desired task velocities and the actual velocities while satisfying all constraints. By assembling all the constraints presented in Section 3.3.4 and 3.3.5, we construct the overall constraint matrices $\mathbf{A}$ and $\mathbf{b}$.

For the absolute task, we solve the quadratic program in equation (3.16) subject to $\mathbf{A}\dot{\mathbf{q}} \leq \mathbf{b}$ to obtain $\dot{\mathbf{q}}_a^*$. This solution achieves the absolute task while respecting the constraints.

For the relative task, we solve the quadratic program in equation (3.18) subject to the same constraints and the additional equality constraint $\mathbf{J}_a\dot{\mathbf{q}} = \mathbf{J}_a\dot{\mathbf{q}}_a^*$. This ensures that the relative task does not interfere with the absolute task.

In practice, we solve the optimization problems sequentially using quadratic programming techniques:

- **First Level (Absolute Task):** Solve the optimization problem for the absolute task to obtain $\dot{\mathbf{q}}_a^*$.
- **Second Level (Relative Task):** Solve the optimization problem for the relative task, using the solution from the first level as a constraint, to obtain the final joint velocities $\dot{\mathbf{q}}_{a,r}^*$.

This hierarchical approach ensures that the absolute task has the highest priority, and the relative task is optimized without affecting the absolute task. All physical and safety constraints are respected throughout the motion.

3.3.7 Joint Velocity Command

After solving the two hierarchical optimization problems, we obtain the joint velocity command $\dot{\mathbf{q}}^*_{a,r}$ which realizes the tasks while fulfilling the constraints. This joint velocity command is then integrated over time to obtain the joint positions, which are sent to the robot actuators, to move the arms accordingly.

3.3.8 Tree Avoidance

In real-world scenarios, tree branches and trunks have highly complex shapes, making precise modeling computationally intensive. To simplify the problem while maintaining safety, we represent the trunk with its branches as an infinite-height cylindrical collision object, surrounding the trunk. This cylinder has a variable radius which depends on the position of the selected fruit, preventing the robot from entering the area between the fruit and the trunk while guaranteeing that the robot can reach the fruit. This approach reduces computational complexity while providing an effective means of obstacle avoidance.

Let r_{obs} denote the radius of the cylinder, and let $\mathbf{P}_t$ represent the position of the tree trunk, as estimated in Section 3.2.3. The cylinder is centered at $\mathbf{P}_t$, and its height extends along the y-axis in the camera frame $\mathcal{F}_{\text{cam}}$. The radius is defined as:

$$r_{\text{obs}} = \text{dist}(\mathbf{P}_f, \mathbf{P}_t) - d_h, \tag{3.28}$$

where:

- $\text{dist}(\mathbf{P}_f, \mathbf{P}_t)$ is the Euclidean distance between the fruit position $\mathbf{P}_f$ and the tree position $\mathbf{P}_t$.
- d_h is the horizontal semi-axis length of the fruit ellipsoid, representing the fruit's radius in the horizontal plane (see Sect. 3.2.1).

By updating r_{obs} for each newly detected fruit, we ensure that the obstacle model accurately reflects the current harvesting scenario. This variable radius provides a safe approaching distance towards the fruit, preventing the robot from moving behind the fruit, where it might collide with branches and trunk.

Since we model each fruit as an ellipsoid with a circular section in the xz-plane, the fruit is characterized by its position $\mathbf{P}_f$ and semi-axis lengths d_h and d_v. Figure 3.3 illustrates the tree and fruit obstacles, showing the cylinder representing the tree and the ellipsoidal model of the fruit.

Since the fruit and tree positions are calculated in real-time, the locations and sizes of the obstacles are also updated dynamically. This allows the robot to adapt to changes in the environment, such as moving branches or fruits swaying in the wind.

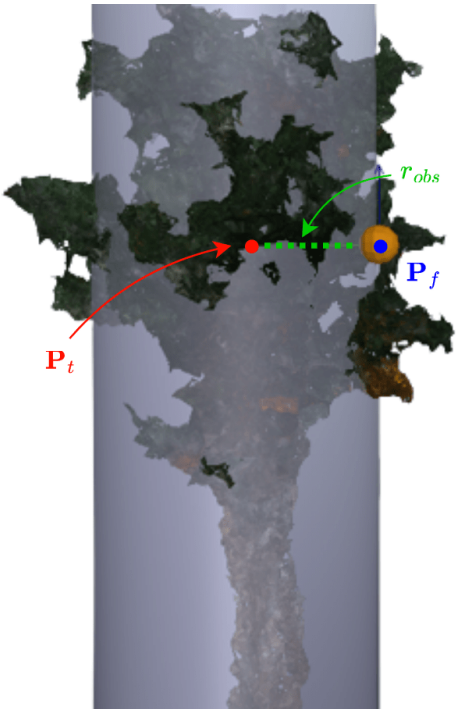


Figure 3.3: Visualization of the tree obstacle (gray cylinder) and fruit obstacle (orange ellipsoid).

3.3.9 Detaching and Collecting

Once both arms have reached their desired poses, the robot proceeds with the detaching and collecting phases. The cutting tool, attached to the end-effector of the cutting arm, is precisely positioned at the peduncle of the fruit. The collecting tool, attached to the collecting arm, is positioned slightly below the fruit to catch it upon detachment.

After the fruit is successfully collected, the robot updates its fruit list $\mathbf{F}$ by removing the harvested fruit and re-evaluates the remaining fruits using the selection method described in Section 4.2.2. The process is repeated for the next target fruit and so on, until all the detected fruits have been collected. If a fruit is unreachable, it is skipped.

While specific tools are not the focus of this study, they can be envisioned as mechanical shears for cutting, and as a basket or net for collecting. The integration of these tools on the robot's end-effectors is essential for practical implementation.

3.3.10 Synthesis

In this Section, we presented a motion control strategy designed for autonomous fruit harvesting with a dual-arm robot. The strategy includes a cooperative control framework and Hierarchical Quadratic Programming (HQP), to achieve the coordinated movements needed for this complex task.

By adopting the cooperative task representation, we defined the fruit harvesting

operation in terms of an absolute task and a relative task. The absolute task governs the motion of the cutting arm in the workspace, focusing on positioning the cutting tool precisely at the fruit's peduncle for detachment. The relative task manages the motion of the collecting arm relative to the cutting arm, ensuring that the collecting tool is positioned to catch the fruit once it is detached. This hierarchical structure allows the system to prioritize critical actions, with the cutting operation taking precedence to ensure accuracy, while the collecting action adapts accordingly, without interfering with the primary task.

The use of HQP enables the simultaneous consideration of multiple tasks and constraints within the motion planning process. By formulating the control problem as a sequence of quadratic programming optimizations, we guaranteed that lower-priority tasks do not compromise the fulfillment of higher-priority ones. This framework also allowed us to define hard constraints related to joint limits, velocity limits, and collision avoidance directly into the control algorithm.

We addressed obstacle avoidance by modeling the tree trunk and branches as an infinite-height cylinder with a variable radius, dynamically adjusted based on the position of the target fruit. This simplified – albeit effective – representation reduces computational complexity, while providing a practical means for preventing collisions. We modeled each fruit as an ellipsoid, allowing for accurate detection and avoidance of nearby fruits during the robot's movements.

The collision avoidance strategy was integrated into the HQP framework, using the Velocity Damper method. We defined the constraints based on the relative distances between the robot and obstacles, to ensure that the robot slows down, as it approaches potential collisions and it maintains a safe distance at all times. These constraints were updated in real-time, so that the robot adapts to changes in the environment.

During the detaching and collecting phases, the robot precisely executes the cutting action, while positioning the collecting tool to catch the falling fruit. The coordination between the two arms ensures that both tasks are performed efficiently, without compromising safety and accuracy. After each harvesting operation, the system updates the list of remaining fruits and re-initiates the process.

3.4 Experiments

We conducted a case study to validate our methodology using artificial orange trees set up in our laboratory. The primary objectives were: to test the fruit detection and tracking strategies with real images from the robot camera and to validate the motion control strategy within a simulated environment that closely resembles the real-world setup.

The fruit detection and tracking algorithms were evaluated on images captured by the RGBD camera mounted on the robot. This allowed us to verify the real-time performance of our perception system under conditions that mimic the complexities of an actual orchard, such as variable lighting and occlusions, caused by leaves and



Figure 3.4: Dual arm robot BAZAR [CPN+19] with cutting and collecting arms.

branches.

3.4.1 Platform Design

The experimental setup consists of the dual-arm mobile cobot BAZAR [CPN+19], shown in Figure 3.4. BAZAR is equipped with two 7-degree-of-freedom Kuka LWR4 robotic arms, providing the necessary dexterity and range of motion for complex manipulation tasks like fruit harvesting. An Intel RealSense D435 RGB-D camera is mounted on the robot to supply real-time visual input for the perception system.

The robot controller is implemented using the Robot Kinematic Control Library (RKCL), which allows efficient computation of inverse kinematics and motion control for the dual-arm system, enabling precise and coordinated movements. This library facilitates the implementation of our hierarchical motion planning framework, to ensure that the robot can move within complex environments while adhering to joint limits

and avoiding collisions.

For the perception component, we used the YOLOv5 object detection algorithm, due to its balance of speed and accuracy, which makes it suitable for real-time applications. The hyperparameters for YOLOv5 were kept at their default values, which gave satisfactory results.

For tracking, we used the DeepSORT algorithm, which enhances tracking performance by combining appearance information with motion cues. The specific parameters for DeepSORT were configured as follows:

- *Model type*: OSNet x0.25 [ZYCX19]
- *Cosine distance threshold*: 0.2
- *Overlap threshold*: 0.5
- *Maximum age*: 300 frames
- *Minimum detections before initialization*: 10
- *Nearest neighbor budget*: 30

These parameters were chosen based on empirical testing, to optimize the balance between tracking accuracy and computational efficiency.

The Intel RealSense SDK was utilized for image acquisition and camera parameter retrieval, which are essential for projecting 2D image coordinates into 3D space -— a critical step for accurate fruit localization. The SDK provided robust functions for handling both color and depth data streams from the RGB-D camera.

OpenCV[1] was used for image processing operations, including color space conversions, filtering, and masking. For the segmentation mask used in tree localization, we converted RGB images to the HSV color space, which is more effective for color-based segmentation under varying lighting conditions. The HSV thresholds were set as follows, based on OpenCV's conventions:

- *Hue (H)*: 10 to 20
- *Saturation (S)*: 70 to 130
- *Value (V)*: 80 to 140

These values were determined through experimentation, to effectively isolate the tree structures from the background in our laboratory setting.

In our fruit selection algorithm, we set the parameters to:

- $d_{update} = 2\,\text{cm}$:

[1] https://opencv.org

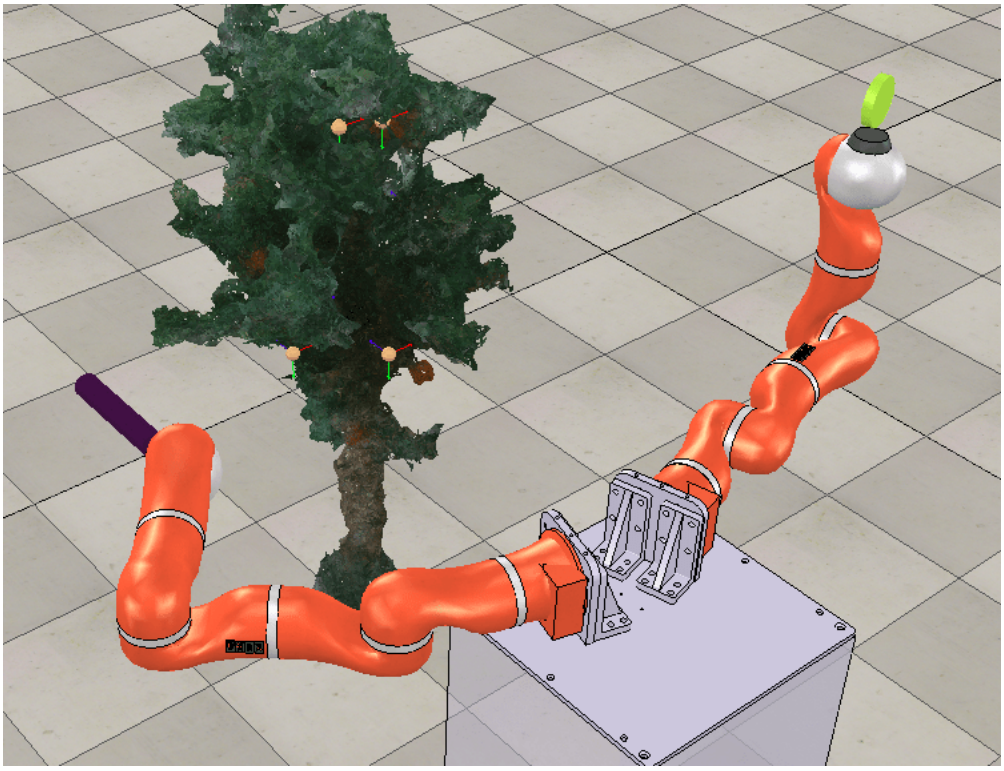


Figure 3.5: Simulation environment with the scanned tree model, detected fruits, and the BAZAR robot.

- $n_{consecutive} = 200$:

To simulate the robot's actions and validate the motion control strategy, we used CoppeliaSim[2]. The simulation included the dual-arm robot model and a 3D representation of the artificial orange tree. The virtual scene is shown in Figure 3.5

The fruit detection and tracking codes run on an Ubuntu 20.04 system equipped with an Intel Xeon W-10885M CPU, Nvidia Quadro RTX 5000 GPU, and 64 GB of RAM.

The remaining software implementation details and parameters are described in Appendix A.

3.4.2 Evaluation

The fruit detection and tracking processes were carried out using the RGB-D camera mounted on BAZAR. We validated our detection and tracking methods on a dataset of 1,935 images of artificial oranges captured in our lab, where we deliberately introduced occlusions for testing purposes. The fruit ID assignment was correct in 83% of the 266 images where fruits were occluded, demonstrating the system's resilience to partial occlusions and its ability to maintain consistent tracking. In images without occlusions,

[2] https://www.coppeliarobotics.com

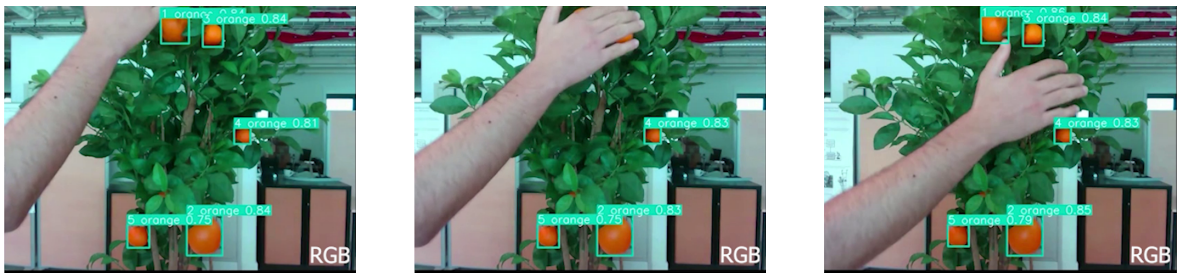


Figure 3.6: Tracking and detection evaluation images, with time progressing from left to right.

the the IDs remain constant all the times, indicating reliable performance under normal conditions. The real-time detection operates at 15 frames per second (FPS), while tracking runs at 2 FPS on average. Example images from tracking and detection images are shown in Figure 3.6.

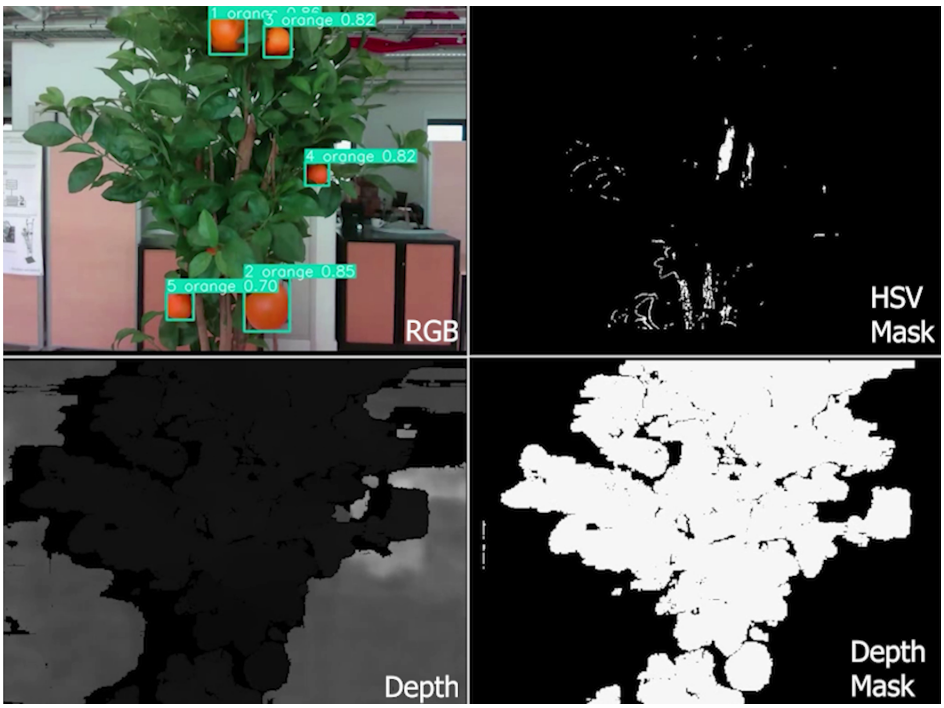


Figure 3.7: Perception images in Experiment 1.Top-left: RGB image with fruit detection. Top-right: corresponding depth image. Bottom-left: HSV mask. Bottom-right: Depth mask.

The oranges detected in our laboratory using the RGB-D camera were replicated in the simulation based on their assigned IDs, positions, and dimensions relative to the camera frame. The data was updated at a sampling frequency of 15 Hz to reflect real-time changes. These virtual fruits served as the targets for the robot's harvesting tasks.

For the simulation of the harvesting tools, we used a black cylinder to represent the cutting tool and a yellow disk to represent the collecting basket, both integrated

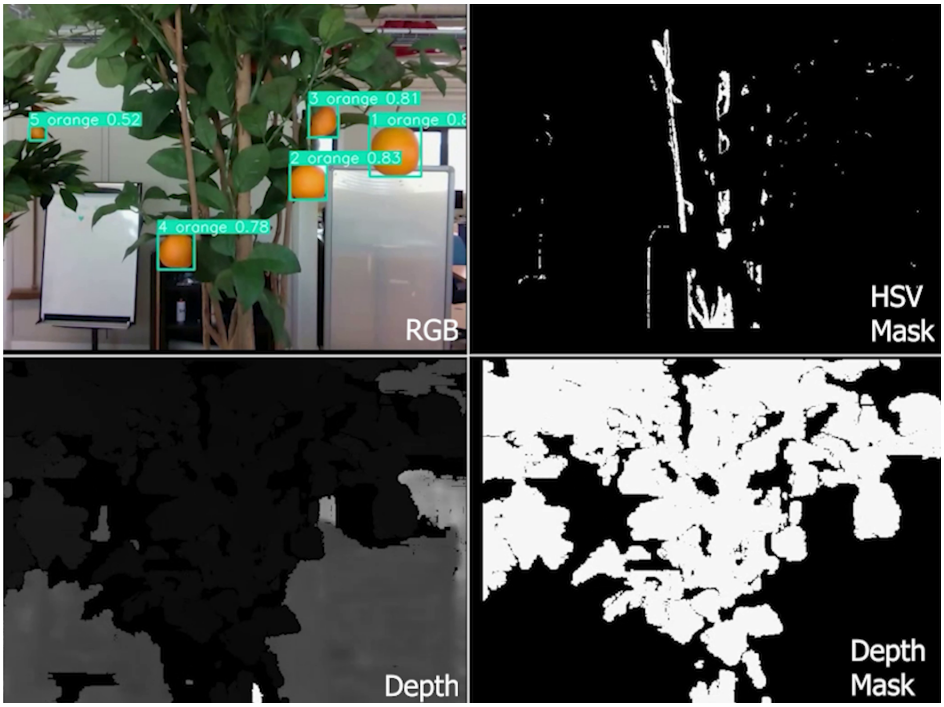


Figure 3.8: Perception images in Experiment 2. Top-left: RGB image with fruit detection. Top-right: corresponding depth image. Bottom-left: HSV mask. Bottom-right: Depth mask.

in the CoppeliaSim environment. The criteria for a successful task is defined as both end-effectors reaching their respective goal poses for the target fruit, with small error tolerances. If the norm of the position error between the robot's end-effector and the goal pose was below a threshold ϵ_{xyz}, the position was considered successfully reached. The same criterion applied to orientation, using $\epsilon_{\phi\theta\psi}$ as the threshold for orientation accuracy. We set $\epsilon_{xyz} = 0.01$m and $\epsilon_{\phi\theta\psi} = 0.01$rad during the experiments.

The cutting arm needed to position the tool at the fruit's peduncle, while the collecting arm positioned the basket beneath the fruit to catch it upon release, which is discussed in Section 3.3.2. We set $d_s = 0.02$m.

In the first experiment, the system detected five oranges, as illustrated in Figure 3.8. One additional orange was not detected due to heavy occlusion by leaves. To further evaluate the tracking performance, we intentionally introduced obstacles and manually moved fruits within the scene. This created dynamic scenarios that mimicked real-world conditions where fruits may sway due to wind or branch movement. Perception and motion planning images from experiment 1 are shown in Figure 3.7 and 3.9.

The results, depicted in Figure 3.11, show the norms of the position errors for both the absolute (cutting arm) and relative (collecting arm) tasks throughout the harvesting process. The position errors remained within acceptable thresholds, confirming that the robot accurately followed the planned trajectories. All detected fruits were successfully harvested without any external intervention. The simulation experiment was completed in 32 seconds, highlighting the system's efficiency and the potential for

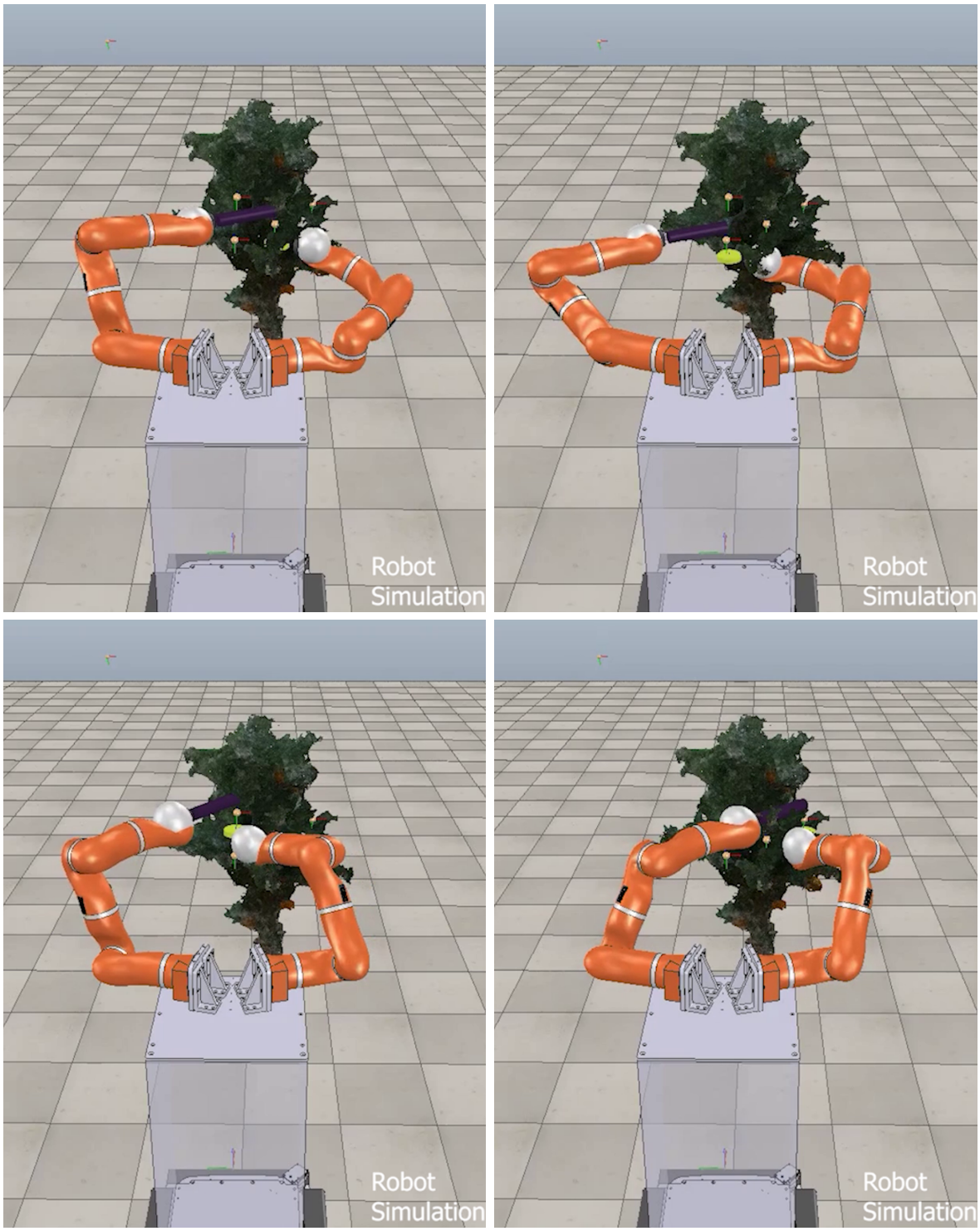


Figure 3.9: Evaluation of tracking and detection over time in Experiment 1.

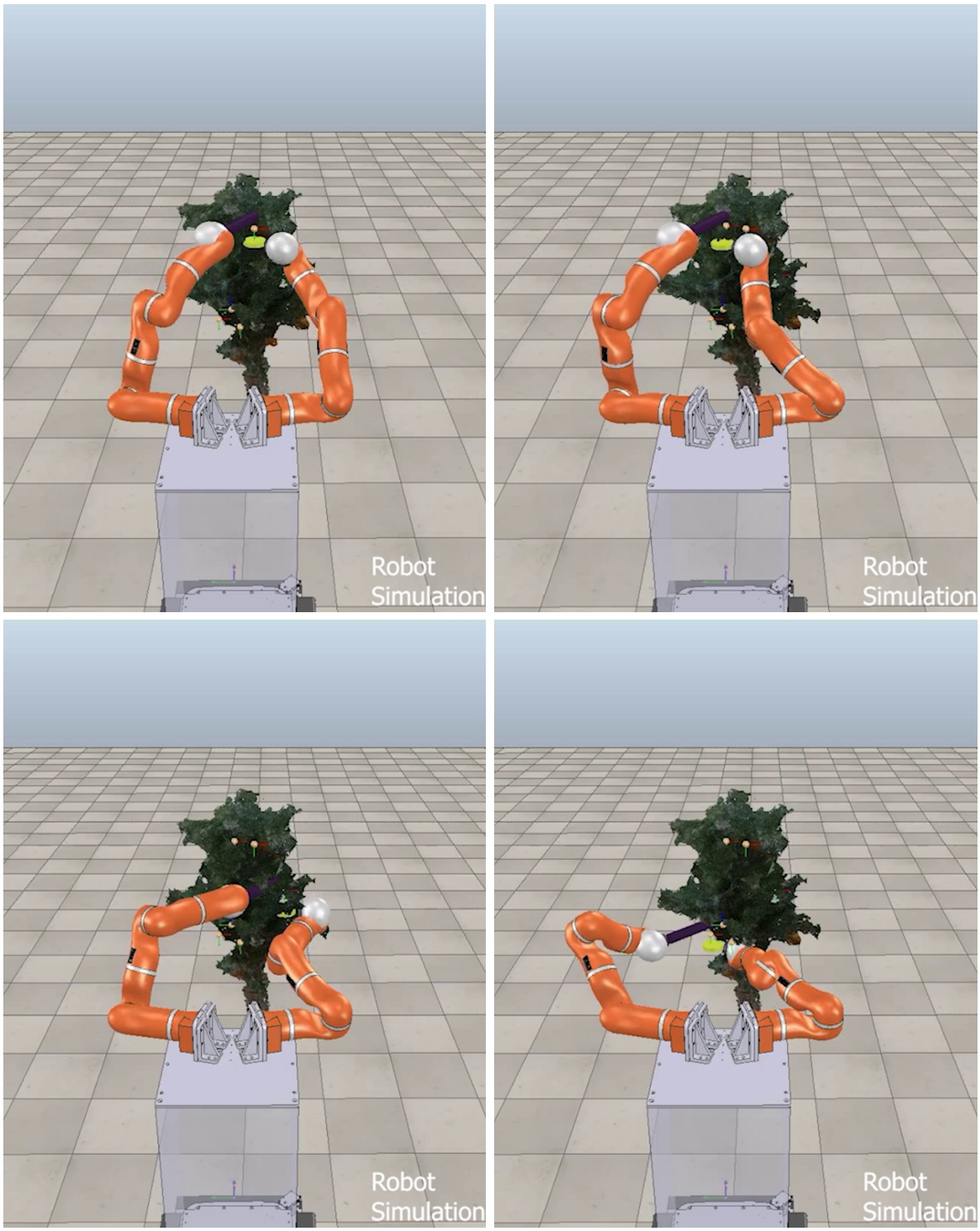


Figure 3.10: Evaluation of tracking and detection over time in Experiment 2.

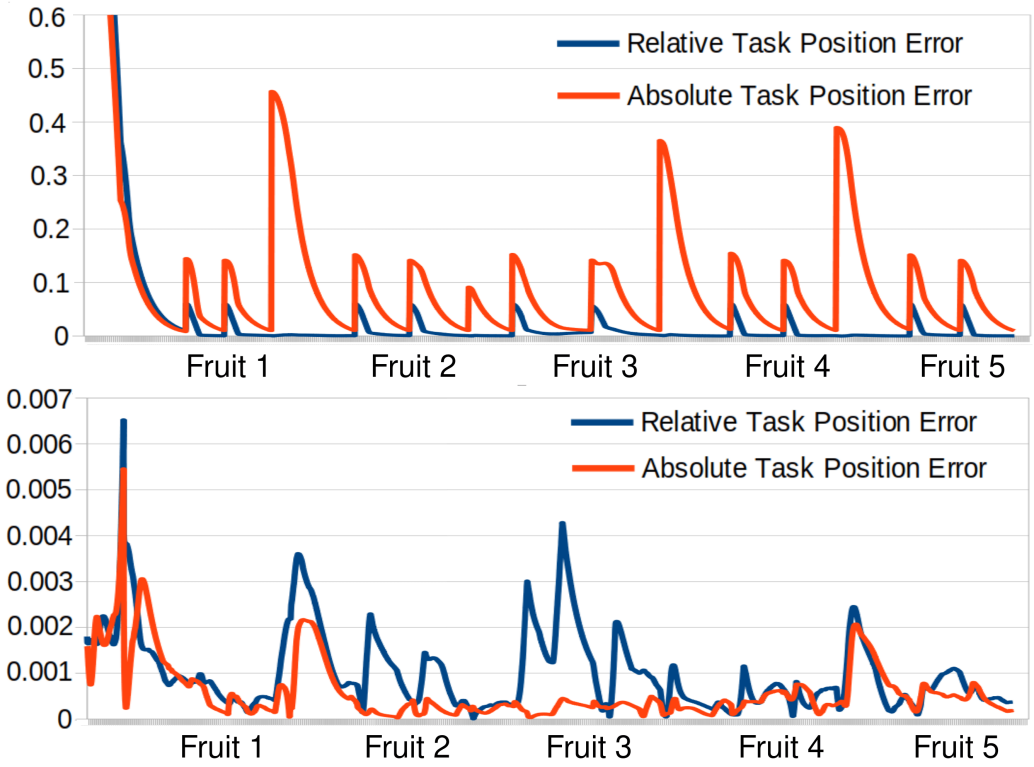


Figure 3.11: Results of the harvesting experiment. Top: norm of the position error in meters. Bottom: norm of the orientation error in radians.

real-time operation.

In the second experiment, five oranges are placed at different locations than the first experiment. The camera detected all five oranges in the scene and maintained accurate tracking at all times. Yet, the robot failed to harvest one of the oranges, which was positioned outside the robot's reachable workspace. This outcome underscores the importance of considering the robot's kinematic constraints during task planning. Despite this limitation, the robot successfully harvested the remaining four oranges without any issues. Perception and motion planning images from experiment 2 are shown in Figure 3.8 and 3.10.

Throughout both simuulation experiments, there were no observed collisions between the robot and the tree trunk or the fruits. The collision avoidance strategies implemented in the motion control framework effectively prevented any unintended contact. The use of simplified obstacle models proved sufficient for real-time collision avoidance, without requiring excessive computational power.

3.4.3 Synthesis

The experiments conducted in this study validate the effectiveness of our dual-arm robot for autonomous fruit harvesting in a simulated environment. The system success-

fully demonstrated the integrated operation of perception, motion control components and achieved accurate fruit detection, tracking, and harvesting tasks. The results indicate that the proposed methods are capable of handling dynamic conditions and occlusions commonly encountered in agricultural settings.

During the experiments, the perception system accurately detected fruits using the YOLOv5 object detection algorithm, even when these were partially occluded by leaves or branches. The DeepSORT tracking algorithm maintained consistent identification of fruits across multiple frames, demonstrating reliable tracking. The combination of depth information from the RGB-D camera allowed for accurate 3D fruit localization.

The motion control strategy, based on the cooperative dual-arm framework and on Hierarchical Quadratic Programming (HQP), enabled the robot to perform coordinated movements required for fruit cutting and collecting. The system prioritized the absolute task of positioning the cutting arm at the fruit's peduncle while adjusting the collecting arm's position relative to the cutting arm. Obstacle avoidance was effectively managed by modeling the tree trunk and branches as a variable-radius cylinder. The implementation of collision avoidance constraints maintained safe operation without any collisions occurring during the experiments.

The experiments demonstrated the robot's ability to adapt to changes in the environment, such as moving fruits or branches, by updating obstacle models and online movements. The system's performance in the simulated environment suggests a strong potential for practical application in real-world agricultural scenarios.

However, while we initialized real-robot test, self-occlusions presented a challenge not accounted for in the simulation. The robot's arms or end-effectors blocked the camera's view, causing loss of visual tracking of the fruit and errors in scene interpretation. The motion planning posed a significant challenge, as the robot frequently became stuck in local minima near obstacles, preventing it from escaping with the current control approach.

While the simulated experiments provide valuable insights on the system's capabilities, further testing with different types of fruits and under real-world conditions is necessary, to fully assess the robustness and genericity of the system. Since the motion planning parts were conducted in simulation, real-world experiments with a physical robot are needed to validate the motion control strategies and to ensure that the system functions correctly outside the simulation. Varying lighting conditions, weather effects, diverse fruit shapes and sizes, and the presence of multiple trees will introduce additional complexities not fully replicated in the simulation.

3.5 Conclusion and Literature Context

In this chapter, we presented the initial development of a dual-arm robotic system for autonomous fruit harvesting. Our motivation stemmed from the limitations of single-arm robotic harvesters in handling the complexities of unstructured agricultural environments, where dexterity and adaptability are crucial due to variable fruit appear-

ances, occlusions, and delicate plant structures. By introducing a dual-arm approach, we aimed to improve the efficiency, precision and versatility (e.g., wrt different fruits) of robotic fruit harvesting.

Our perception system combined deep learning and image processing techniques to achieve robust fruit detection, tracking, and localization. We used YOLOv5 for real-time fruit detection and DeepSORT for tracking. These choices allowed us to maintain consistent identification of fruits even in the presence of occlusions or motion. We used depth information from an RGB-D camera and assumed that fruits were ellipsoids, to accurately calculate their 3D positions in the camera's frame. For tree localization, we applied HSV color space segmentation and depth masking to estimate the position of the tree trunk and branches. This allowed the robot to avoid collisions during motion planning.

For motion control, we used a dual-arm cooperative control framework based on Hierarchical Quadratic Programming (HQP). This method allowed us to set absolute and relative tasks for the two arms, with priority given to the cutting operation while coordinating the collecting action. The obstacle avoidance strategy included dynamic obstacle modeling and collision avoidance constraints in the HQP framework, improving the robot's ability to safely move around complex structures like tree trunks and branches.

Our simulations validated the effectiveness of the proposed system. The robot successfully detected, tracked, and harvested fruits, handling dynamic conditions and occlusions. The motion control strategy helped with the coordinated movements without collisions, demonstrating the system's potential for real-world applications.

Compared to current research, most studies have focused on single-arm robotic systems for fruit harvesting, which often lack the flexibility needed for complex tasks in unstructured environments. For example, Davidson et al. [DHMK17] and Hayashi et al. [HSY+10] explored single-arm solutions, and faced issues with adaptability and efficiency. Our dual-arm approach builds on these studies by introducing a cooperative control framework that improves the robot's ability to handle tasks.

Our use of deep learning combined with depth information improves upon the work of Kang and Chen [KC20] and Koirala et al. [KWWM19], offering a more reliable solution for detecting and tracking fruits. Additionally, we use HSV color segmentation for tree localization, which is a simpler and faster alternative to more complex models.

Our motion control strategy differs from the others, by combining dual-arm control with HQP, allowing the robot to handle multiple tasks and constraints at the same time. This is more flexible than the motion strategies used in earlier single-arm systems [SDK+17, TFAI08, MB14, vHHvT+02, HGIT02, HSY+10, LEM+17, SDK+17, TFAI08, XGGF19, KYI+10], and aligns with the dual-arm control approach proposed by Adorno et al. [CCS96].

In summary, our work addresses important challenges in perception and motion control for dual-arm fruit harvesting robots. By demonstrating that our approach functions in a hybrid simulated environment using real images, we have laid the groundwork for further development.

Despite these contributions, the current system has limitations. It has not been thoroughly tested in real-world conditions. Simulations, while informative, do not capture the full complexity of agricultural environments. In the next chapter (4), we address these limitations by adapting the system towards real-world applications. We focus on improving its robustness and performance in actual agricultural environments and validate its effectiveness through experiments beyond simulation. This progression is necessary to move closer to practical robotic harvesting.

CHAPTER 4

Towards Real-World Adaptation

4.1 Motivation and Challenges

Building upon the foundational work of developing an initial dual-arm robotic system for fruit harvesting, challenges remain in adapting the system for effective operation in unstructured and dynamic agricultural environments. The initial system faced limitations in handling complex occlusions, maintaining tracking accuracy, and optimizing motion paths, which hindered its efficiency and practicality for real-world applications.

Firstly, fruits are often partially or fully occluded by leaves, branches, or other fruits, making detection and localization difficult. The initial approach struggled under such conditions, leading to missed harvesting opportunities and reduced efficiency. To overcome this, a robust perception system combining advanced deep learning models with geometric reasoning is developed. This system uses RGB-D images to accurately detect and locate fruits, even amidst complex occlusions and dynamic environmental changes.

Secondly, consistent identification and localization of fruits across multiple frames are crucial for effective harvesting. Previous tracking algorithms had difficulties maintaining accuracy over time, especially when dealing with moving branches or varying lighting conditions. A spatial-displacement based tracking solution utilizing 3D displacement data from object detection and depth information is introduced. This approach increases tracking stability and reduces errors caused by occlusions and environmental variations.

Thirdly, the initial motion planning relied on local planners, which are limited in navigating cluttered spaces without collisions. This often resulted in suboptimal paths, increasing the time and energy required for harvesting and raising the risk of damaging fruits or tree structures. To address this limitation, a global motion planner integrated with an adaptive obstacle model is implemented. This planner considers the entire



Figure 4.1: BAZAR robot harvesting oranges (left) and Baxter harvesting apples (right).

environment and task constraints to generate optimized, collision-free trajectories, ensuring safe and efficient maneuvering through cluttered environments.

Additionally, determining the most efficient sequence for harvesting fruits is vital for reducing total travel distance and time. Without proper scheduling, harvest time may increase due to inefficient movement between fruits. By formulating the harvesting sequence as a Traveling Salesperson Problem (TSP) and developing a scheduling algorithm, the optimal harvesting sequence is computed in real time, improving operational efficiency.

Finally, to validate the system's adaptability and effectiveness under varied conditions, experiments are conducted using two dual-arm robots, Baxter and BAZAR, each equipped with distinct end-effectors for detaching and catching fruits. The robots are tested on different fruit types, such as apples and oranges, which present diverse challenges in size, shape, texture, and attachment. Robot setups are shown in Figure 4.1.

Addressing these challenges advances the development of a dual-arm robot capable of autonomously harvesting fruits in real-world agricultural settings. By focusing on robust fruit tracking, optimized motion planning and control, efficient fruit scheduling, and real-world validation, the system demonstrates improved robustness and adaptability without compromising safety or efficiency. An overview of the methodologies used in this chapter is shown in Figure 4.2 with the differences with the Chapter 3.

In this chapter, we make the same assumptions described in Section 3.1 to focus on the other important aspects of the dual-arm harvesting robots.

4.2 Perception

In this chapter, we improve the perception module of our dual-arm robot, to address the challenges identified in the previous implementation, particularly in maintaining tracking accuracy under complex occlusions and in different environmental conditions. While the fruit detection and tree localization methods remain unchanged from those described in the previous chapter, we introduce significant improvements in the fruit

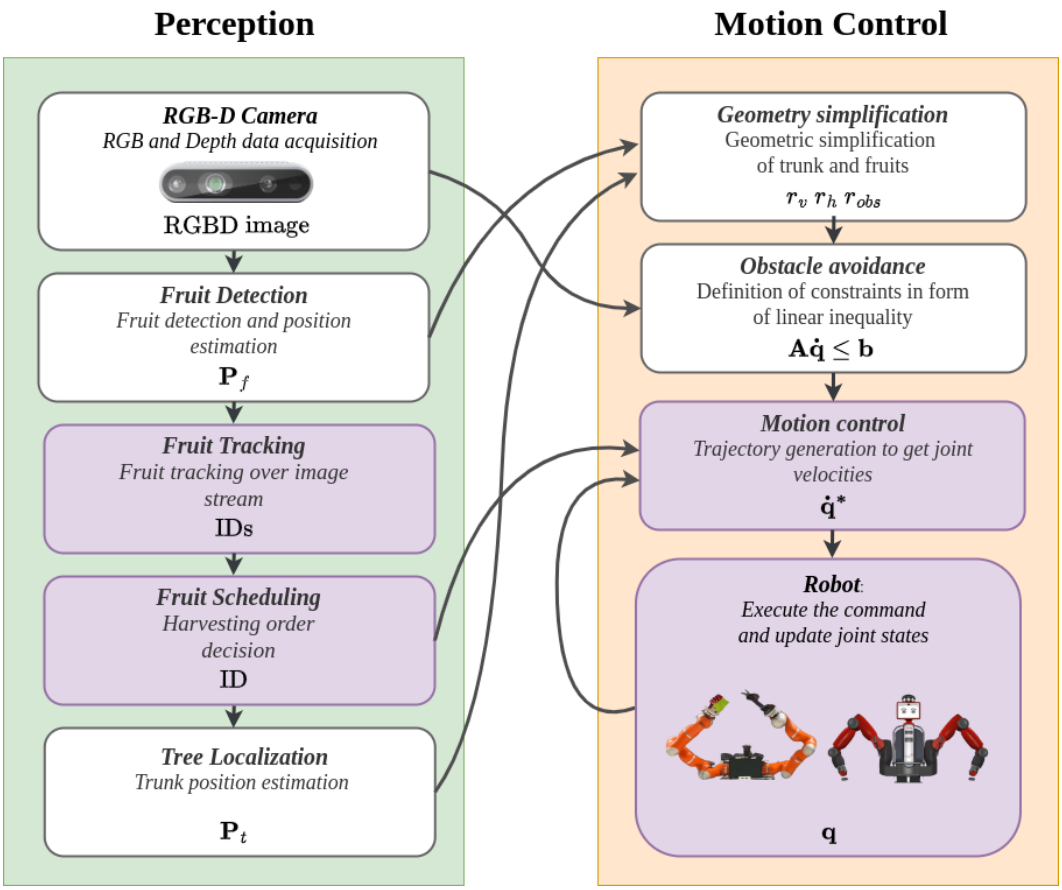


Figure 4.2: Overview of our dual-arm harvesting framework. Purple boxes represent the differences from the methodologies in Chapter 3.

tracking and scheduling components. By enhancing the tracking algorithm and integrating an efficient fruit scheduling strategy within the perception module, we aim to optimize the overall harvesting process, ensuring that fruits are not only detected and tracked accurately but also harvested in an optimal sequence.

4.2.1 Fruit Tracking

In our previous work [GNC+23], we used DeepSORT [WBP17], an off-the-shelf tracking algorithm primarily designed for human tracking using a Kalman filter and a re-identification model. While DeepSORT provided a baseline level of performance, it did not meet the accuracy requirements for our application, particularly in handling complex occlusions and in maintaining consistent fruit identification across frames.

To address these limitations, we developed a spatial-displacement based fruit tracking algorithm based on the 3D displacement of detected fruits. Our algorithm uses the spatial information obtained from the RGB-D camera to track fruits more reliably, especially in dynamic and cluttered environments.

Spatial-Displacement Based Tracking

The core idea of our spatial-displacement based tracking algorithm is to use the 3D positions of detected fruits to associate detections across frames based on their spatial proximity. By operating in 3D space rather than in 2D image space, we reduce the ambiguity caused by overlapping detections and occlusions which commonly occur in agricultural environments. The tracking methodology is detailed in Algorithm. 1.

Algorithm 1 Spatial-Displacement Based Tracking Algorithm

Input: Detection matrix **D**
Output: Tracking matrix **T**
Initialize $\mathbf{T} \leftarrow \emptyset$, $\mathbf{C} \leftarrow \emptyset$
for each detection f^D in **D do**
    for each track f^T in **T do**
        if $\left|\left|f^D - f^T\right|\right| < d_t$ **then**
            Update f^T position with f^D position
        else
            f^T.unseen $\leftarrow f^T$.unseen $+ 1$
            if f^T.unseen $> n_{\text{unseen}}$ **then**
                Remove f^T from **T**
            end if
        end if
    end for
    for each candidate f^C in **C do**
        if $\left|\left|f^D - f^C\right|\right| < d_t$ **then**
            Update f^C position with f^D position
            f^C.counter $\leftarrow f^C$.counter $+ 1$
            if f^C.counter $> n_{\text{add}}$ **then**
                Add f^C to **T**
            end if
        else
            Add f^D to **C** with a new ID
        end if
    end for
end for

The Algorithm 1 operates on three data structures:

- **Detection Matrix D**: Contains the 3D positions of fruits detected in the current frame.
- **Tracking Matrix T**: Maintains active fruit IDs over time. Each track includes a unique ID, the fruit's position, and a counter for the number of consecutive frames during which the fruit has not been detected.

- **Candidate Matrix C**: Stores potential new fruit IDs that are not yet considered active. Each candidate includes the fruit's position and a counter for the number of consecutive frames during which the fruit has been detected.

We define the following key parameters:

- **Threshold Distance** d_t: Determines whether a detection can be associated with an existing track, based on the Euclidean distance between their 3D positions.
- **Maximum Absence Frames** $n_{\textbf{unseen}}$: Specifies the maximum number of consecutive frames a fruit can remain undetected before its track is removed from the tracking matrix.
- **Confirmation Frames** $n_{\textbf{add}}$: Specifies the number of consecutive frames a new detection must be observed before it is promoted from a candidate to an active track.

We initialize the tracking matrix **T** and the candidate matrix **C** as empty sets (line 1). The tracking matrix **T** maintains active fruit ID, and the candidate matrix **C** holds potential new fruit ID not yet confirmed.

For each detection f^D in **D**, we attempt to associate it with an existing track in **T** (lines 2–11). If a match is found within the threshold distance d_t, we update the track's position, reset its unseen counter, and set *matched* to **True**.

If no match is found, we try to match the detection with a candidate in **C** (lines 12–22). If a match is found, we update the candidate's position, increment its seen counter, and if the seen counter exceeds n_{add}, we promote it to an active track in **T**.

If the detection does not match any candidate, we add it as a new candidate with a seen counter of 1 (lines 23–24).

After processing all detections, we increment the unseen counters of IDs not updated in the current frame (lines 25–30). IDs with unseen counters exceeding n_{unseen} are removed from **T**.

4.2.2 Fruit Scheduling

Efficient harvesting requires not only accurate detection and tracking, but also an optimized sequence of actions to minimize the total operation time. In our work, we introduce an enhanced fruit scheduling procedure which determines the optimal sequence of fruits to harvest by solving a Traveling Salesperson Problem (TSP).

Formulating the Traveling Salesperson Problem

The Traveling Salesperson Problem seeks the shortest possible route that visits a set of locations once. In our case, the locations are the positions of the tracked fruits, and the starting point is the robot's current position.

By solving the TSP, we aim to:

- **Minimize Total Travel Distance**: Reduce the distance the robot's arms need to move between fruits improves efficiency and reduces operation time.
- **Optimize Energy Consumption**: Shorter paths result in less energy usage, which is crucial for a future battery-powered robot.
- **Improve Productivity**: Efficient scheduling allows the robot to harvest more fruits in less time, increasing productivity.

4.2.2.1 Algorithm Implementation

We implement the fruit scheduling procedure using Algorithm 2.

Algorithm 2 Fruit Scheduling Algorithm

Require: Tracking matrix $\mathbf{T}$, Robot position $\mathbf{P}_{\text{robot}}$
Ensure: Optimal harvesting sequence $PATH$
Initialize $d_{\min} \leftarrow \infty$, $PATH \leftarrow \emptyset$
Extract fruit positions $\mathbf{F} = \{\mathbf{P}_{f^T} \mid f^T \in \mathbf{T}\}$
Generate permutations $\mathcal{P}$ of $\mathbf{F}$
for each permutation Q in $\mathcal{P}$ **do**
    Compute total distance d starting from $\mathbf{P}_{\text{robot}}$
    $d \leftarrow |\mathbf{P}_{\text{robot}} - Q[1]|$
    for $i = 1$ to $|Q| - 1$ **do**
        $d \leftarrow d + |Q[i] - Q[i+1]|$
    end for
    if $d < d_{\min}$ **then**
        $d_{\min} \leftarrow d$
        $PATH \leftarrow Q$
    end if
end for

The Algorithm 2 operates on the following data structures:

- **Tracking Matrix $\mathbf{T}$**: Contains the 3D positions of the fruits currently being tracked.
- **Fruit Position Set $\mathbf{F}$**: The set of fruit positions extracted from $\mathbf{T}$.
- **Permutation Set $\mathcal{P}$**: All possible permutations of the fruit positions in $\mathbf{F}$.

We define the following key parameters:

- **Robot Position $\mathbf{P}_{\text{robot}}$**: The current 3D position of the robot.
- **Minimum Distance $d_{\min}$**: Initialized to infinity, represents the shortest total travel distance found so far.

We initialize the minimum distance $d_{\min}$ to infinity and the optimal path $PATH$ to empty (line 1). From the tracking matrix $\mathbf{T}$, we extract the 3D positions of the tracked fruits to form the set $\mathbf{F}$ (line 2). We generate all possible permutations $\mathcal{P}$ of the fruit positions (line 3).

For each permutation Q, we calculate the total travel distance d starting from the robot's position (lines 4–7). We compute the distance from the robot to the first fruit (line 5) and sum the distances between consecutive fruits in the sequence (lines 6–7).

If the calculated distance d is less than the current minimum $d_{\min}$, we update $d_{\min}$ and set $PATH$ to Q (lines 8–10).

In practice, some fruits may be unreachable due to constraints or obstacles. If a fruit is found to be unreachable during motion planning, we remove it from $\mathbf{F}$ and recompute the optimal path by rerunning the algorithm starting from line 2.

The TSP is NP-hard, and the number of permutations grows factorially with the number of fruits. For small numbers of fruits (up to 10), it is computationally feasible to compute all permutations.

The fruit scheduling algorithm is integrated into the robot's decision-making process. Once the optimal path is determined, the robot plans its motions accordingly, following the sequence of fruits specified by $PATH$.

4.2.3 Synthesis

By developing a spatial-displacement based fruit tracking algorithm and implementing an optimized fruit scheduling procedure, we have improved the perception module of our dual-arm robotic system. These improvements address the limitations identified in the previous chapter, particularly in maintaining high tracking accuracy under complex occlusions and optimizing motion paths in dynamic environments.

The spatial-displacement based tracking algorithm uses 3D spatial information to improve tracking stability and reliability, while the scheduling algorithm ensures efficient operation by minimizing total travel distance through effective fruit scheduling. Together, these enhancements contribute significantly to the system's overall robustness and adaptability, enabling it to perform effectively in challenging agricultural settings.

4.3 Motion Control

In this chapter, we improve the motion control strategies of our dual-arm robot, to address the limitations identified in the previous chapter. There, our approach relied on a cooperative task representation [CCS96] with velocity damping for collision avoidance, controlling both manipulator arms simultaneously towards the fruit position. While that method offered certain advantages, its reliability in real-world scenarios was not always guaranteed. This was primarily due to its dependence on local planning and to the challenges associated with maintaining absolute-relative poses for dual arms moving in complex environments.

To overcome these limitations, we adopt separate control strategies for each manipulator arm and rely on a global motion planner (MoveIt[1]), to generate optimal, collision-free trajectories. By considering the entire environment and the complete task, the global planner accounts for all obstacles and constraints, leading to better overall performance and effectiveness in task completion.

We use the MoveIt software platform for global motion planning, replacing its default inverse kinematics solver by TRAC-IK [BA15]. TRAC-IK provides fast convergence to solutions and handles kinematic redundancy effectively, making it well-suited for the complex and dynamic tasks involved in our application.

Tree avoidance is implemented as described in Section 4.3.2 of the previous chapter, i.e., we model the tree trunk and branches using simplified geometric shapes, to efficiently prevent collisions without incurring high computational costs.

In the following sections, we detail the dual-arm trajectory generation process and the detaching and collecting mechanisms, highlighting the improvements and adjustments made over our previous implementation.

4.3.1 Dual-Arm Trajectory Generation

To ensure safe and efficient harvesting, it is essential to prevent collisions with fruits and branches. We achieve this by defining robot's actions into three distinct tasks: *Approach*, *Harvest*, and *Leave*. To ensure safe and efficient harvesting, it is essential to prevent collisions with fruits and branches. We achieve this by defining three distinct robot tasks: *Approach*, *Harvest*, and *Leave*.

- *Approach*: The robot moves to a position near the target fruit, maintaining a safe distance to avoid any potential collisions with the fruit or branches. This task prepares the robot for the precise movements required during harvesting.

- *Harvest*: The robot executes the harvesting operation, moving its end-effectors to the precise locations necessary to detach and collect the fruit. This task ensures the cutting tool is correctly positioned at the fruit's peduncle, while the collecting tool is placed underneath, to catch the falling fruit.

- *Leave*: After harvesting, the robot moves away from the harvested fruit to a safe distance, preventing any damage to the tree or to itself, while it prepares for the following operations.

In contrast to the cooperative task representation used previously, where both arms were controlled in a coupled manner, we now treat each arm independently within the global motion planning framework. This approach allows for greater flexibility and adaptability in generating collision-free trajectories, since the planner can optimize the path of each arm individually while considering the other arm as a dynamic obstacle.

[1] https://moveit.ros.org/

Trajectory Planning Process

For each task (*Approach*, *Harvest*, *Leave*), we generate trajectories for both the cutting and collecting arms using the following process:

1. **Pose Definition**: We define the desired end-effector pose of each arm during each task. The poses are specified in the camera frame $\mathcal{F}_{\text{cam}}$, using the fruit's position $\mathbf{P}_f$ and dimensions calculated from the perception module.

2. **Constraint Specification**: We set constraints on the orientations and positions, to prevent collisions and to ensure the tools are correctly aligned for harvesting. For instance, we constrain the angle θ between the arms, to maintain a separation which avoids interference.

3. **Path Generation**: Using the global motion planner in MoveIt with the TRAC-IK solver, we generate feasible paths to move the end-effectors from their current poses to the desired poses for each task, while respecting the specified constraints and avoiding obstacles.

4. **Trajectory Validation**: Each path is validated, to ensure it is collision-free and it respects all kinematic and dynamic constraints. If a valid solution exists for all tasks, we concatenate the path into a single executable trajectory with time parameterization.

5. **Execution or Replanning**: If valid trajectories are found, the combined path is executed on the robot. If not, the fruit is considered unreachable, and the system proceeds to the next fruit in the scheduled sequence.

Pose Definitions

We define the goal poses for cutting and collecting arms during each task as follows:

- **Approach Task**:

$$\begin{aligned} \mathbf{x}_{\text{cut}}^{A} &= [x_f - d_a,\ y_f - d_v - d_s,\ z_f - d_a,\ 0,\ *,\ \pi], \\ \mathbf{x}_{\text{col}}^{A} &= [x_f + d_a,\ y_f + d_v + d_s,\ z_f - d_a,\ 0,\ *,\ \pi], \end{aligned} \tag{4.1}$$

- **Goal Task**:

$$\begin{aligned} \mathbf{x}_{\text{cut}}^{G} &= [x_f,\ y_f - d_v - d_s,\ z_f,\ 0,\ *,\ \pi], \\ \mathbf{x}_{\text{col}}^{G} &= [x_f,\ y_f + d_v + d_s,\ z_f,\ 0,\ *,\ \pi], \end{aligned} \tag{4.2}$$

- **Leave Task**:

$$\begin{aligned} \mathbf{x}_{\text{cut}}^{L} &= [x_f - d_c,\ y_f - d_v - d_s,\ z_f - d_c,\ 0,\ *,\ \pi], \\ \mathbf{x}_{\text{col}}^{L} &= [x_f + d_c,\ y_f + d_v + d_s,\ z_f - d_c,\ 0,\ *,\ \pi], \end{aligned} \tag{4.3}$$

Where:

- (x_f, y_f, z_f) are the fruit's 3D coordinates.
- d_a is the *Approach* distance.
- d_c is the *Leave* distance.
- d_s is a safety margin to avoid contact with the fruit during cutting.
- d_v is the vertical semi-axis length of the fruit (from the ellipsoid model).

The pitch angle θ is unconstrained (indicated by $*$) to maximize the reachable workspace, while the yaw angle is set to π to align the end-effectors appropriately.

Advantages of the Global Planner Approach

By employing a global motion planner, we achieve several benefits over the previous local planning approach:

- **Comprehensive Obstacle Avoidance**: The planner takes into account all known obstacles. These include the tree trunk, branches, fruits, as well as the other arm, to ensure collision-free paths.
- **Optimized Trajectories**: The planner searches for the most efficient paths satisfying all constraints. This reduces unnecessary movements and saves time.
- **Improved Reliability**: Global planning reduces the likelihood of the robot getting stuck in local minima or encountering unforeseen obstacles. This improves the system's robustness in complex environments.
- **Independent Arm Control**: Treating each arm separately allows for greater flexibility in managing their movements. This is especially beneficial when dealing with kinematic redundancy and workspace limitations.

4.3.2 Obstacle Avoidance

As explained in Section , we model the tree trunk and branches using simplified geometric shapes like cylinders and ellipsoids to efficiently represent obstacles.

The global motion planner integrates these obstacle models into its planning process, continuously updating the obstacle representations based on perception data.

The planner also treats the other arm as a dynamic obstacle and ensures that the trajectories of both arms do not interfere with one another. This is especially important during the *Harvest* task, when both arms are in close proximity.

We also modeled the tools as dynamic obstacles and integrated them into the robot's geometry. The tools are discussed in detail in Section 4.3.3. While most of the tools



Figure 4.3: Cutting tools of Baxter (top-left) and BAZAR (bottom-left), along with collecting tools of Baxter (top-right) and BAZAR (bottom-right).

were modeled with minor deviations using basic geometric shapes, the BAZAR cutting tool required special attention due to its more complex shape. To address this, we 3D scanned the pruning tool, simplified the mesh to reduce computational load, and included it into the robot's geometry. A 3D rendering of the scan and the simplified mesh is provided in Appendix B.

4.3.3 Detaching and Collecting

Once the robot reaches the desired poses during the *Harvest* task, the harvesting operation is executed. The cutting tool is activated to detach the fruit, and the collecting tool is positioned to catch the fruit as it falls. Figure 4.3 demonstrates the tools used on each robot:

- **Baxter Robot**:
  - Cutting Tool: A 3D-printed attachment with two blades mounted onto the robot's gripper.
  - Collecting Tool: A soft, transparent plastic bag attached to the second end-effector.
- **BAZAR Robot**:

- Cutting Tool: An electric pruner attached to the end-effector.
- Collecting Tool: A commercially available plastic collecting basket with rigid circular top.

These tools are designed to be effective for the specific fruit types and harvesting actions required. The cutting tools cleanly detach the fruit without harming the tree, and the collecting tools safely catch the fruit to avoid bruising or loss.

After the fruit is successfully harvested, the robot transitions to the *Leave* task, moving both arms away from the tree to a safe position. The system then selects the next fruit to harvest, as determined by the fruit scheduling procedure, and the process repeats.

4.3.4 Synthesis

The motion control strategies developed in this chapter improve the effectiveness of the dual-arm robotic system for fruit harvesting. By adopting a global motion planning approach and treating each arm independently, we overcome the limitations of the previous local planner, which was based on the cooperative task representation.

This methodology allows for precise and safe manipulation of both arms, minimizing the risk of collisions with fruits, branches, and between the arms themselves. The global planner creates optimized, collision-free paths, which consider the whole environment and task limits. This improves the reliability and efficiency of the harvesting process.

These advancements in motion control, combined with the perception and tracking methods described earlier, contribute to the overall goal of developing an autonomous dual-arm robot capable of operating effectively in challenging agricultural settings.

4.4 Experiments

To validate the effectiveness and versatility of our dual-arm robotic harvesting framework, we conducted two comprehensive case studies using different robotic platforms and fruit types.

The first case study involved the BAZAR robot [CPN+19] harvesting artificial oranges in a laboratory environment. The second experiment required the Baxter robot to harvest artificial apples under different conditions. Conducting experiments in two distinct laboratories, one in Montpellier, France and the other at Monash University, Australia allowed us to evaluate the robustness of our methodologies in different controlled environments.

We used two different robots and fruit types, to show that our framework is both effective and adaptable to various agricultural robotics tasks. The experiments were set up to evaluate the system's accuracy in perception, efficiency in motion planning, and reliability.

In the following sections, we first detail the hardware and software configurations of the experimental platforms. We then describe the experimental procedures and present the results, highlighting key observations and performance metrics.

4.4.1 Platform Design

To fully test our dual-arm harvesting system, we used two different robots, each equipped with specialized tools tailored for the specific requirements of the fruit harvesting task:

BAZAR Setup

Our first experimental platform is BAZAR [CPN+19]. The robot setup is the same as described in previous chapter, with two KUKA LWR4 arms and an Intel RealSense D435 RGB-D camera mounted on the robot 'head'. The KUKA LWR lightweight robot arms are known for their high precision and sensitivity, making them suitable for delicate tasks.

For the end-effectors, we designed custom tools to perform the cutting and collecting tasks. The left arm is equipped with an electric pruner for precise fruit stem cutting. The right arm carries a rigid plastic basket with a wide opening, for collecting the fruits as they are harvested. These tools are securely mounted to the robot's wrists. Support component that enables us to attach tools to the end effectors. We created 3D renderings of the support components to attach tools to the end effectors which can be found in Appendix B.

Baxter Setup

The second experimental platform involves the Rethink Robotics Baxter robot (see Figure 4.4, right). Baxter is a widely used research robot, featuring two 7-DOF arms and designed for safe human-robot interaction. Each arm comes with an electric parallel gripper, which we modified to accommodate our custom harvesting tools. As in the BAZAR setup, an Intel RealSense D435 RGB-D camera is mounted on Baxter's upper body.

For the end-effectors, we used Baxter's grippers as they already have a control library avaible to the public. The tool features two angled blades designed to cut fruit stems effectively, when the gripper closes. On the right gripper, we attached a soft, transparent plastic bag to collect the harvested apples. We created 3D renderings of the support components to attach tools to the end effectors which can be found in Appendix B.

Software and Implementation Details

Both robots use the same software architecture to maintain consistency in performance evaluation. The perception system is written in Python and uses PyTorch to



Figure 4.4: Apple detection with tracking IDs usign Baxter (left) and orange detection with tracking IDs using BAZAR (right).

run the YOLO object detection algorithm.

The Intel RealSense SDK connects to the RGB-D camera, providing real-time access to both color and depth streams, along with camera calibration parameters.

For image processing tasks like color space conversions and morphological operations, we used OpenCV[2], an open-source computer vision library popular in the robotics community.

The robots are controlled using the Robot Operating System (ROS) Noetic[3], which provides communication tools, hardware abstraction, and device drivers, for integrating perception, planning, and control modules.

Motion planning is managed by MoveIt[4], an advanced framework that works with ROS. We used the TRAC-IK inverse kinematics solver [BA15] within MoveIt for its speed and ability to find solutions in constrained environments.

The fruit detection and tracking codes run on an Ubuntu 20.04 system equipped with an Intel Xeon W-10885M CPU, Nvidia Quadro RTX 5000 GPU, and 64 GB of RAM.

The remaining software implementation details and parameters are described in Appendix A.

4.4.2 Experimental Setup

The experiments were conducted in controlled indoor laboratory environments, designed to simulate orchards. The artificial orange and apple trees were arranged to present challenges such as varying fruit heights, occlusions from leaves and branches, and differing levels of ambient lighting. In the Montpellier laboratory, the BAZAR robot operated in an environment with artificial orange trees. At Monash University,

[2] https://opencv.org
[3] http://wiki.ros.org/noetic
[4] http://moveit.ros.org

the Baxter robot was tested in a similar setting with artificial apple trees.

For each experiment, the robot was positioned in front of an artificial tree bearing either apples or oranges, depending on the platform. The RGB-D camera mounted on the robot's head provided real-time visual data. The laboratory environments were set up to imitate real orchards, with changing lighting, leaves and branches causing occlusions, and fruits positioned at various depths and angles.

Operational Parameters and Safety Considerations

To prioritize safety during the experiments, especially given the close proximity of the robotic arms to the fruits and trees, we adjusted the operational speeds of both robots. For Baxter, the velocity scaling factor was set to 0.8, balancing efficiency with the need to prevent sudden movements that could cause damage. For BAZAR, we opted for a more conservative scaling factor of 0.1, due to its higher payload capacity and the to delicacy required when handling fragile fruits like oranges.

Parameter Settings for Methodology

To maintain consistency across experiments and ensure repeatability, we standardized the parameters used in our methodology:

- *Tracking Parameters:* The threshold distance for associating detections across frames was set to $d_t = 0.5$ meters. The unseen frame counter, which determines how long a fruit can be unseen before being considered lost, was set to $n_{unseen} = 200$ frames. The seen frame counter, dictating how many consecutive frames a new detection must appear in, before being confirmed, was set to $n_{add} = 50$ frames.

- *Trajectory Variables:* For the *Approach* and *Leave* phases, we set the distances to $d_a = d_c = 0.1$ meters for BAZAR and $d_a = d_c = 0.2$ meters for Baxter. These values define the buffer zones to prevent the robot from getting too close to the fruit during non-harvesting motions. The security distance while cutting, $d_s = 0.02$ meters, was set to ensure that the cutting tool operates effectively without applying excessive force that could damage the fruit or the tree.

- *Motion Planner and Communication Parameters:* All other variables related to the motion planner and to ROS communication were kept at their default settings, as provided by the respective software packages.

4.4.3 Fruit Detection and Tracking Results

In the experiment with the Baxter robot, the perception system successfully detected six apples within the robot's workspace, as shown in Figure 4.5 (top). Each

apple was accurately identified and assigned a unique tracking ID. The detection algorithm demonstrated robustness, by effectively handling partial occlusions and overlapping fruits, and by maintaining consistent tracking IDs across frames. The real-time detection operates at 15 frames per second (FPS), while tracking runs at 8 FPS on average.

For the BAZAR robot experiment, the system detected five oranges, depicted in Figure 4.5 (bottom). All oranges within the camera's field of view were correctly identified and tracked without any omission. The tracking algorithm maintained high accuracy even when the fruits were partially obscured by foliage or branches.

4.4.4 Collision Avoidance and Harvesting Results

Throughout both experiments, the robots executed the harvesting tasks without any collisions with tree trunks, branches, or other fruits. The motion planning and obstacle avoidance strategies effectively prevented potential damage to both the robots and the artificial trees. The global motion planner generated collision-free trajectories for both the cutting and collecting arms, taking into account the environment and the positions of all detected fruits.

Each fruit harvesting sequence involved three main tasks:

- *Approach*: The robot moved its arms to predefined positions near the target fruit, maintaining safe distances to avoid early contact.
- *Harvest*: The arms moved to precise positions for harvesting. The cutting tool was aligned with the fruit's stem, and the collecting tool was positioned beneath the fruit to catch it upon detachment.
- *Leave*: After successfully harvesting the fruit, the robot retracted its arms to prepare for the next target.

During the Baxter experiment, we encountered an issue with one of the apples, since the cutting tool did not fully sever the stem. This resulted in the apple remaining attached momentarily before eventually falling into the collecting bag after a slight delay. This incident highlights the importance of precise control and feedback mechanisms in the cutting process.

Snapshots from various stages of the experiments are presented in Figure 4.5, to show the robots in action during the harvesting tasks.

4.4.5 Trajectory Tracking Results

Figure 4.6 illustrates the position and orientation errors over time for both experiments. The graphs show that robots have achieved the goal poses throughout the harvesting process. Differences in error patterns between the two robots are due to variations in operational parameters, such as the d_a and d_c distances and motion profiles generated by the global planner.

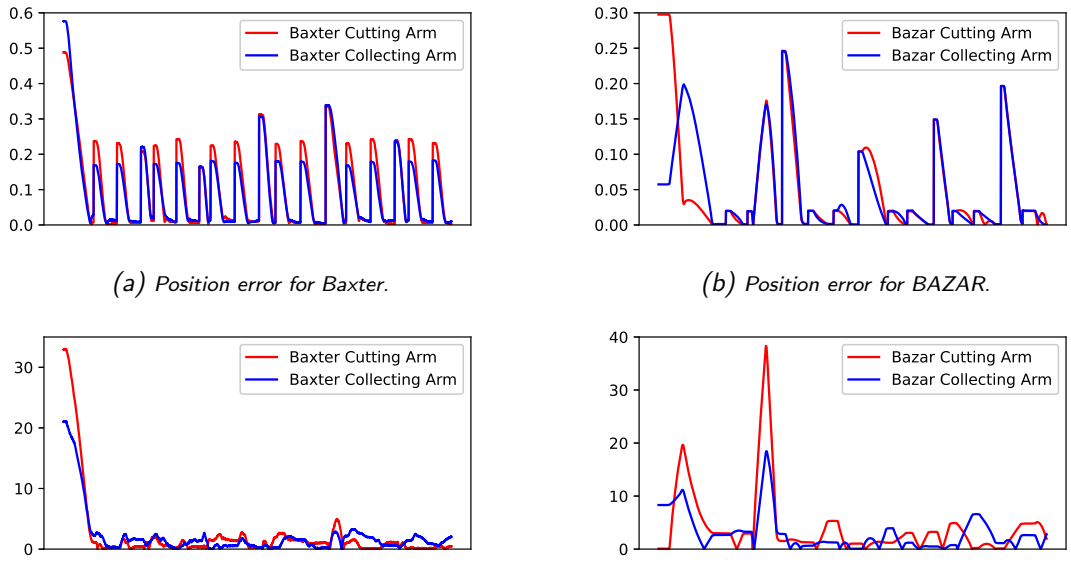


(a) BAZAR robot harvesting oranges.



(b) Baxter robot harvesting apples.

Figure 4.5: Snapshots of the autonomous harvesting experiments.



(a) Position error for Baxter. *(b) Position error for BAZAR.*

(c) Orientation error for Baxter. *(d) Orientation error for BAZAR.*

Figure 4.6: Evaluation of position (in m) and orientation (in deg) errors between the end-effectors and the goal poses in time. The top graphs show the norm of the position error over time, while the bottom graphs show the orientation error.

4.4.6 Performance Metrics Evaluation

To comprehensively assess the system's performance, we defined several metrics on perception accuracy, operational speed, and motion control effectiveness. We detail such metrics below.

Perception Metrics

We evaluated the perception methods using two metrics: the Mean Absolute Error (MAE) of localization and the Tracking Accuracy of IDs (TAI).

For the perception aspects, we define two metrics: *Mean Absolute Error* of localization and *Tracking Accuracy of IDs*: *Mean Absolute Error* (**MAE**) of localization is used to measure the average magnitude of errors between predicted and actual positions of fruits in the images. This metric is used to quantify the accuracy of fruit detection systems, with lower (**MAE**) values indicating better performance. It is calculated by taking the mean of the absolute differences between the detected positions of the fruits in the images and their corresponding real positions, calculated by (4.4).

$$\mathbf{MAE} = \frac{1}{N_d} \sum_{i=1}^{N_d} \left| \mathbf{P}_{f,i} - \mathbf{P}^*_{f,i} \right|, \tag{4.4}$$

where N_d is the total number of detected fruits, $\mathbf{P}_{f,i}$ is the detected position of fruit i, and $\mathbf{P}^*_{f,i}$ is the ground truth position of fruit i. Ground truths are obtained through

empirical analysis of each image, with fruit positions manually determined.

Tracking Accuracy of IDs (**TAI**) measures the effectiveness of a fruit tracking system, by calculating the ratio of correctly tracked fruits to total number of tracking results in the frames. This metric indicates how accurately the system can maintain the correct identification of each fruit throughout the tracking process. Higher values of **TAI** signify more reliable and precise tracking performance. It is calculated as in (4.5).

$$\mathbf{TAI} = \frac{\sum_{k=1}^{N\times M} C_k}{\sum_{k=1}^{N\times M} T_k}, \tag{4.5}$$

where $N \times M$: Total number of tracking instances across all frames and fruit IDs, C_k: Indicator variable (1 if a fruit is correctly tracked in a given frame; 0 otherwise). T_k: Indicator variable (1 if a fruit should be tracked in a given frame; 0 otherwise).

The results are summarized in Table 4.1.

Table 4.1: Perception Metrics Evaluation

Experiment	**MAE (pixels)**	**TAI (%)**
BAZAR harvesting 5 oranges	5.90	89
Baxter harvesting 6 apples	16.03	95

In Table 4.1 The Mean Absolute Error **MAE** and Tracking Accuracy of IDs **TAI** metrics demonstrate the effectiveness of our perception system. For BAZAR harvesting 5 oranges, the **MAE** was 5.90 px, and the **TAI** was 89 %, while Baxter harvesting 6 apples resulted in an **MAE** of 16.03 px and a **TAI** of 0.95. The decreased performance of the **MAE** observed in Baxter's experiment can be attributed to the larger size of the apples in pixels due to the camera angle, leading to increased errors in pixel position within the images. **TAI** results demonstrate a good tracking accuracy in both setup. These metrics suggest that our system can accurately detect and track fruits with small errors.

Operational Speed Metrics

To evaluate the speed of the entire framework, we define three metrics: *Total Harvesting Time* **T** measures the overall time taken to complete the harvesting task, from start to end. Similar to **T**, *Average Harvest Time per Fruit* $\mathbf{\Delta t}$ calculates the mean time required to harvest each fruit. The theoretical minimum harvest time per fruit calculated by dividing **T** to the velocity factor of the robots (described in Section 4.4.3) as $\mathbf{\Delta t}^*$, calculated by (4.6).

$$\mathbf{\Delta t}^* = \frac{\mathbf{T}}{\alpha_v} \tag{4.6}$$

with α_v as the velocity scaling factor of the robot.

Table 4.2: Operational Speed Evaluation

Experiment	**T** (s)	**Δt** (s)	**Δt*** (s)
BAZAR harvesting 5 oranges	103	20.60	2.06
Baxter harvesting 6 apples	49	8.17	6.53

In terms of speed, the Table 4.2 shows that the Total Harvesting Time **T** and Average Harvest Time per Fruit **Δt** highlight the performance differences between the two robotic systems. BAZAR took 103 seconds to harvest 5 oranges, averaging 20.60 seconds per fruit, while Baxter took 49 seconds to harvest 6 apples, averaging 8.17 seconds per fruit. Bazar's reduced performance is due to a lower velocity scaling factor, implemented for safety, given its more delicate construction and the complexity involved in repairs. The theoretical minimum harvest time per fruit $\mathbf{\Delta t}^*$ is calculated as 2.06 seconds for BAZAR and 6.53 seconds for Baxter.

Motion Control Metrics

We define three metrics to evaluate our motion control methods: *Path Efficiency Ratio* for the cutting η_{cut} and collecting η_{col} arms are defined as the ratio of the actual distance traveled by the robot to the shortest possible distance needed to visit all the fruits, as determined by the Traveling Salesperson Problem solution in Algorithm 2. This metric evaluates the efficiency and optimality of the robot's trajectory, with a value closer to 1 indicating a more efficient path. *Maximum Path Deviation* $\epsilon_x, \epsilon_\theta$ evaluates the accuracy of the robot's movement by measuring the largest deviation of the actual end-effector pose (position for ϵ_x and orientation for ϵ_θ) from the planned trajectory. A low value of these metrics ensures the robot follows the intended path closely.

Table 4.3: Motion Control Metrics Evaluation

Experiment	η_{cut} (%)	η_{col} (%)	ϵ_x (cm)	ϵ_θ (deg)
BAZAR harvesting 5 oranges	33	37	2.58	9.18
Baxter harvesting 6 apples	24	26	2.56	7.40

The motion control metrics in Table 4.3, including Path Efficiency Ratios η_{cut} and η_{cut} and Maximum Path Deviation from Planned Trajectory (ϵ_x and ϵ_θ), further illustrate the effectiveness of our approach. Both robots showed relatively low path efficiency ratios, with BAZAR at 0.33 for cutting and 0.37 for collecting, and Baxter at 0.24 for cutting and 0.26 for collecting. While these values are lower than ideal, they fall within an acceptable range given the complexity of the task. The reduced efficiency is primarily due to the additional phases in the harvesting process. The robot follows a path from the initial pose to the first fruit and from the last fruit to the final pose. We define *Approach*, *Goal*, and *Leave* tasks for each fruit, all of which contribute to the increased distance traveled by the robot. The path deviations (ϵ_x and ϵ_θ) were small, indicating that the robots closely followed the planned trajectories.

4.4.7 Synthesis

The experiments with our dual-arm robotic harvesting framework, using the BAZAR and Baxter platforms have clearly shown the capability to perform autonomous fruit harvesting tasks with high precision and reliability. By successfully harvesting both artificial oranges and apples in laboratory environments, we demonstrated that the framework can adapt to different conditions and hardware setups, proving its versatility, robustness, and scalability.

The perception module detected and tracked fruits accurately, even in difficult conditions like occlusions and variable lighting. The Mean Absolute Error (MAE) values showed that the system could localize fruits within a few pixels of their actual positions, ensuring accurate 3D positioning for manipulation. Specifically, BAZAR had an MAE of 5.90 pixels and a Tracking Accuracy of IDs (TAI) of 89%, while Baxter achieved an MAE of 16.03 pixels and a TAI of 95%. These high TAI values confirmed the reliability of the tracking algorithm in maintaining consistent fruit identification despite environmental changes.

The motion control strategies prevented collisions and ensured precise end-effector positioning. The global motion planner, integrated with adaptive obstacle modeling, allowed the robots to avoid collisions with tree trunks, branches, and fruits, reducing the risk of damage. Low maximum path deviations in both experiments showed that the robots followed the planned paths closely, confirming the accuracy in motion planning and motion control. Path Efficiency Ratios further demonstrated the system's effectiveness in handling complex environments.

The difference in operational speeds between BAZAR and Baxter highlighted how hardware configurations, robot dynamics, and safety measures affect system performance. BAZAR moved more slowly but safely, taking 103 seconds to harvest five oranges (20.60 seconds per fruit), while Baxter harvested six apples in 49 seconds (8.17 seconds per fruit). However, the theoretical speed for the harvest without safety limitations are 2.06s for BAZAR and 6.63s for Baxter per fruit in average. These times are reasonable, given the need for safe and precise movements in cluttered environments.

A minor issue during Baxter's experiment with an incomplete cut suggests the need for improvements in the cutting mechanism or in the use of force control. Adding feedback, to verify successful cuts could improve reliability and performance.

Overall, the experiments validate the strength and adaptability of our dual-arm robotic framework in realistic conditions. The system combined enhanced perception and motion control, to handle complex tasks in dynamic environments. Performance metrics showed strong results in fruit detection, tracking, motion planning, and execution.

4.5 Conclusion and Literature Context

In this chapter, we have presented and validated our novel dual-arm robotic framework for autonomous fruit harvesting through two comprehensive case studies using the

BAZAR and Baxter robots in different laboratory settings. The experiments focused on harvesting oranges with BAZAR and apples with Baxter, conducted in Montpellier and Monash laboratories under varying lighting conditions. These experiments demonstrate the versatility, robustness, and scalability of our approach across various scenarios, robot architectures, and fruit types.

Our approach differs from existing literature through dual-arm coordination, efficient fruit scheduling, and a spatial-displacement based tracking algorithm specifically tailored for agricultural settings. Unlike previous works that primarily focus on single-arm robots or use simpler detection models, our system offers greater flexibility and precision in handling diverse fruit types, robots, and challenging environments. The dual-arm configuration allows simultaneous execution of complementary tasks—cutting and collecting, which increases operational efficiency and mimics the dexterity of human laborers.

In the existing literature, most studies have focused on single-arm robots for fruit harvesting; these often struggle with dexterity and adaptability in complex environments. For example, Davidson et al.[DHMK17] and Hayashi et al.[HSY+10] explored single-arm solutions but faced challenges in handling multiple tasks simultaneously and maintaining efficiency. Our dual-arm approach addresses these issues by enabling coordinated movements and improving harvesting precision and speed.

In the field of perception, earlier methods using traditional image processing techniques were less effective under variable lighting and occlusion conditions. Our integration of YOLOv5 for real-time fruit detection and the introduction of a spatial-displacement-based tracking algorithm based on 3D displacement builds upon the work of Kang and Chen [KC20] and Koirala et al. [KWWM19], providing a robust solution for accurate fruit localization and tracking in dynamic environments. This combination addresses challenges like occlusions and environmental changes, maintaining consistent fruit identification and localization over multiple frames.

Regarding motion control, our use of global motion planning with MoveIt and TRAC-IK offers a clear improvement over the local planning in our previous approach. In real-world harvesting, environments are heavily cluttered with obstacles like branches and leaves. Global planning evaluates the entire workspace, avoiding the risk of getting stuck in local minima and ensuring better path optimization. It also allows for replanning when the robot encounters unexpected obstacles or changes in the environment, resulting in more reliable and efficient movement.

The experimental validation on two different robots (BAZAR and Baxter) and two different fruits (apple and orange) demonstrates the adaptability and scalability of our approach. By successfully harvesting oranges and apples in distinct laboratory settings with varying lighting conditions, we have shown that our system can be generalized across various hardware configurations and operational environments. This cross-platform capability is a significant advancement toward practical deployment in real-world agricultural settings.

In summary, our work advances the state of the art in agricultural robotics by introducing a dual-arm framework that combines advanced perception and motion

control techniques. The successful experiments with both BAZAR and Baxter robots demonstrate the system's effectiveness in accurately detecting, tracking, and harvesting fruits in complex environments.

Despite these achievements, the current system has limitations. The experiments were conducted in controlled laboratory settings, which do not fully replicate the complexities of real-world agricultural environments. Factors such as varying weather conditions, natural lighting variations, uneven terrain, and the presence of multiple trees introduce additional challenges that were not addressed in our experiments. Moreover, one of the most significant challenges in real-world applications is fruit occlusion, where fruits are partially or entirely occluded by branches or leaves. Our system needs to be tested and adapted to handle such occlusions effectively.

In the next chapter, we will explore the impact of fruit occlusions and discuss strategies to overcome this obstacle. Addressing these limitations is another step towards real-world deployment of our platform.

CHAPTER 5

Fruit Occlusion Handling

5.1 Motivation and Challenges

Autonomous fruit harvesting robots encounter significant challenges in real-world agricultural environments due to occlusions from branches, leaves, and other obstacles. These occlusions hinder accurate detection and localization of fruits, leading to misidentification, reduced detection accuracy, and slower harvesting processes. Previous work demonstrated effective fruit detection and tracking methods but did not specifically address the complexities introduced by occlusions. Accurate fruit localization is essential for successful harvesting and collision-free path planning. Traditional approaches relying on unobstructed views are inadequate in these settings, failing to account for the natural complexities of orchards and resulting in reduced reliability and performance.

This chapter focuses on developing perception and manipulation strategies that allow robots to operate effectively in environments with significant occlusions during fruit harvesting. The goal is to ensure the robot can detect fruits despite occlusions and physically interact with the environment to clear obstructed views. This involves perception capabilities to detect and localize partially obscured fruits and creating methods for the robot to identify and manipulate occluding elements, primarily branches, to clear the line of sight without causing damage to the plant or the robot.

To achieve that, first, generative modeling estimates the complete appearance of occluded fruits by predicting the full shape and features based on partially visible data and prior knowledge of the fruit's typical characteristics.

Second, a three-dimensional extension of the Hough transform detects branch directions in 3D space which allows the robot to identify branches contributing to occlusions and model them accurately for manipulation planning. Third, a pushing strategy determines the optimal points and directions to apply force to occluding branches, based

on the areas of the fruit that remain covered. This strategy enables the robot to physically push branches out of the way safely and efficiently, clearing its line of sight and creating accessible paths for harvesting.

By confronting the challenges posed by occlusions, this work bridges the gap between controlled laboratory settings and real-world agricultural scenarios, facilitating the practical deployment of autonomous harvesting robots. By enhancing fruit visibility and accessibility, we ultimately increase detection accuracy and reachability of the fruit in a harvesting scenario. The research aims to develop a robust system capable of perceiving and localizing fruits despite occlusions and actively manipulating the environment for successful harvests. The proposed methods are validated through real-world experiments on various fruits, such as apples, lemons, and oranges, under different environmental conditions, demonstrating their potential for practical application.

In developing our method, we make the following assumptions to simplify the problem and focus on the core challenges:

- *Hypothesis 1:* The properties of the real (occluded) fruit are identical to those estimated by the generative model. That is, the estimated position, dimensions and centroid coordinates accurately represent the actual fruit. This assumption is justified when the generative model effectively reconstructs the occluded regions, providing reliable estimations for manipulation planning.

- *Hypothesis 2:* Fruits are considered to have a perfect ellipsoidal shape, which helps in modeling the view frustum and estimating the fruit's dimensions.

- *Hypothesis 3:* There is only one primary occlusion region per fruit that significantly affects the robot's ability to detect and reach the fruit. While multiple minor occlusions may exist, we focus on the most significant one for the purpose of occlusion clearance.

- *Hypothesis 4:* Branches causing occlusions can be approximated as straight line segments in 3D space. This simplification allows us to use line detection algorithms and simplifies the computation for manipulation planning.

- *Hypothesis 5*: Undetected pixels of the fruit are in the camera field of view, reducing the image dimensions.

5.2 Fruit Identification

In agricultural environments, fruits are often partially occluded by branches, leaves, or other fruits, making accurate detection and localization challenging. Figure 5.1 demonstrates an occluded fruit that we will use as a reference during this chapter. To enable effective harvesting, it is essential to identify fruits even when they are not fully visible. This section presents our method for fruit identification in the presence

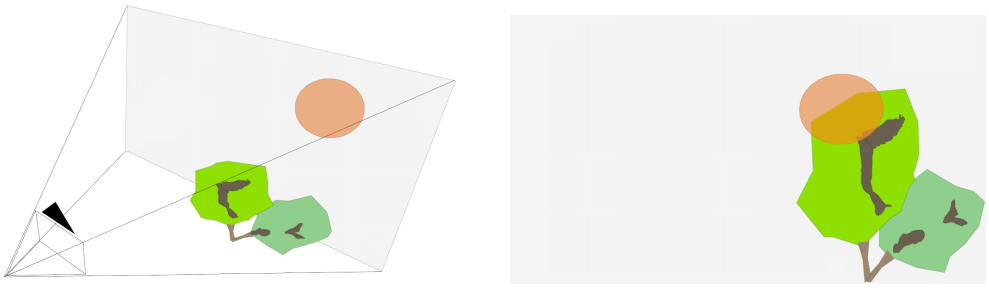


Figure 5.1: Camera field of view (left), with corresponding image (right). The tree branches and leaves are green/brown, and the fruit is orange.

of occlusions, which involves two main steps: partial fruit segmentation and complete fruit appearance estimation by a generative neural network.

5.2.1 Partial Fruit Segmentation

The first step in identifying occluded fruits is to segment the visible portions of the fruit from the rest of the image. This process relies on analyzing the color information in the image and extracting pixels that correspond to the fruit based on their color characteristics.

Let $\mathcal{I} = \{1, \ldots, w\} \times \{1, \ldots, h\} \subset \mathbb{N}^2$ denote the set of pixel coordinates in the image captured by the camera, where w and h are the image width and height, respectively. Each pixel $(u, v) \in \mathcal{I}$ corresponds to a point in the image plane.

We define a mapping $\mathbf{m}$ that associates each pixel (u, v), along with its corresponding depth value $z_{uv} \in \mathbb{R}^+$ (obtained from the depth image), to a 3D point $\mathbf{P} = (x, y, z)^\top$ in the camera frame $\mathcal{F}_{\text{cam}}$. Assuming an undistorted perspective projection (i.e., a pinhole camera model), this mapping is given by:

$$\begin{array}{rccc} \mathbf{m}: & \mathcal{I} \times \mathbb{R}^+ & \to & \mathcal{F}_{\text{cam}} \\ & \begin{pmatrix} u \\ v \\ z_{uv} \end{pmatrix} & \mapsto & \begin{pmatrix} x = \frac{(u-u_0)}{f} z_{uv} \\ y = \frac{(v-v_0)}{f} z_{uv} \\ z = z_{uv} \end{pmatrix} \end{array} \tag{5.1}$$

Here, $(u_0, v_0) \in \mathbb{R}^2$ are the coordinates of the camera's principal point in the image plane, and $f \in \mathbb{R}^+$ is the focal length of the camera. This mapping allows us to project pixel coordinates and depth information into 3D space within the camera frame.

The mapping $\mathbf{m}$ is bijective, meaning it has an inverse mapping $\mathbf{m}^{-1}$ that projects 3D points in the camera frame back to pixel coordinates in the image:

$$\begin{array}{rccc} \mathbf{m}^{-1}: & \mathcal{F}_{\text{cam}} & \to & \mathcal{I} \times \mathbb{R}^+ \\ & \begin{pmatrix} x \\ y \\ z \end{pmatrix} & \mapsto & \begin{pmatrix} u = \frac{fx}{z} + u_0 \\ v = \frac{fy}{z} + v_0 \\ z = z \end{pmatrix} \end{array} \tag{5.2}$$

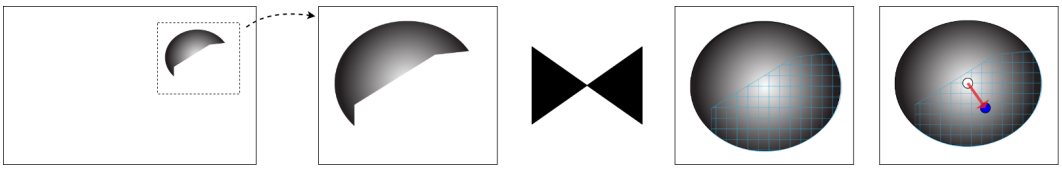


Figure 5.2: Outline of the fruit estimation method. Left to right: Visible fruit pixels in masked depth image $\mathbf{D}_M$, cropped ROI depth image $\mathbf{D}'_M$ fed into the generative model (represented by a generic autoencoder shape), which outputs the estimated fruit depth image $\hat{\mathbf{D}}'_M$. The white circle represents the estimated fruit center $(\hat{u}'_c, \hat{v}'_c)$, and the blue circle indicates the center of the occlusion mask. The red arrow depicts the occlusion direction vector $\vec{\mathbf{O}}_f$. The cross-hatched area represents the occluded region estimated by the generative model.

To perform fruit segmentation, we analyze the color image and examine the Hue (H), Saturation (S), and Value (V) components of each pixel. Fruits often have distinctive color characteristics which can be used for segmentation. By determining the ranges of H, S, and V values that correspond to the fruit's color, we create a binary mask $\mathbf{M} \in \mathcal{I} \times \{0, 1\}$, where:

$$\mathbf{M}(u, v) = \begin{cases} 1, & \text{if } (H(u, v), S(u, v), V(u, v)) \text{ matches fruit color} \\ 0, & \text{otherwise} \end{cases} \tag{5.3}$$

Using this mask, we isolate the pixels that are likely to belong to the fruit. Once we have the segmentation mask $\mathbf{M}$, we retrieve the corresponding depth values z_{uv} from the depth image for all pixels classified as fruit. This allows us to construct the depth image of the segmented fruit, denoted as $\mathbf{D}_f$, where:

$$\mathbf{D}_f(u, v) = \begin{cases} z_{uv}, & \text{if } \mathbf{M}(u, v) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{5.4}$$

The depth image $\mathbf{D}_f$ contains depth information only for the pixels corresponding to the fruit, with all other pixels set to zero. This image serves as the basis for further processing to estimate the fruit's complete appearance.

5.2.2 Fruit Appearance Estimation

To handle occlusions and to estimate the occluded parts of the fruit, we use a generative neural network based on an encoder-decoder architecture. The network is designed to reconstruct the complete depth image of the fruit from the partially visible depth image $\mathbf{D}_f$.

Figure 5.2 illustrates the steps involved in the fruit appearance estimation process. We hereby detail these steps.

Masking and Region of Interest (ROI) Extraction

First, we apply the binary mask $\mathbf{M}$ to the depth image $\mathbf{D}_f$ to obtain a masked depth image $\mathbf{D}_M$:

$$\mathbf{D}_M(u,v) = \mathbf{M}(u,v) \cdot \mathbf{D}_f(u,v), \tag{5.5}$$

where $\cdot$ denotes element-wise multiplication (Hadamard product). In $\mathbf{D}_M$, only the depth values corresponding to the segmented fruit pixels are retained.

Under *Hypothesis 5* in Section 5.1 that the complete fruit should appear within a certain region of interest (ROI), we crop $\mathbf{D}_M$ to focus on this area. The ROI is defined based on the size and location of the segmented fruit. We compute the bounding box of the segmented fruit and define the ROI coordinates accordingly.

Let (u_c, v_c) be the centroid of the segmented fruit in the image, calculated as:

$$u_c = \frac{1}{N_f} \sum_{(u,v)\in\mathcal{F}} u, \quad v_c = \frac{1}{N_f} \sum_{(u,v)\in\mathcal{F}} v, \tag{5.6}$$

where $\mathcal{F} = \{(u,v) \in \mathcal{I} \mid \mathbf{M}(u,v) = 1\}$ is the set of fruit pixels, and N_f is the total number of fruit pixels.

We define the ROI as a square region centered at (u_c, v_c) with a size of d_{ROI} pixels. The top-left and bottom-right coordinates of the ROI are:

$$\begin{aligned} u_{TL} &= u_c - \frac{d_{\text{ROI}}}{2}, \\ v_{TL} &= v_c - \frac{d_{\text{ROI}}}{2}, \\ u_{BR} &= u_c + \frac{d_{\text{ROI}}}{2}, \\ v_{BR} &= v_c + \frac{d_{\text{ROI}}}{2}. \end{aligned} \tag{5.7}$$

We set $d_{\text{ROI}} = 56$ in the rest of the work. We crop $\mathbf{D}_M$ to obtain the ROI depth image $\mathbf{D}'_M$:

$$\mathbf{D}'_M(u,v) = \mathbf{D}_M(u + u_{TL}, v + v_{TL}), \quad \forall u, v \in [0, d_{\text{ROI}}]. \tag{5.8}$$

This cropping operation isolates the fruit region and reduces computational complexity for the fruit estimation.

Generative Neural Network Architecture

The cropped depth image $\mathbf{D}'_M$ is fed to the generative neural network, which estimates the occluded parts of the fruit. The network is based on a modified U-Net architecture [RFB15] from our previous work [GTC23], designed for image segmentation and filling in missing parts of images.

The network architecture is illustrated in Figure 5.3 and consists of an encoder and a decoder with skip connections:

Encoder:

- Input layer: size 56×56 (size of the ROI).

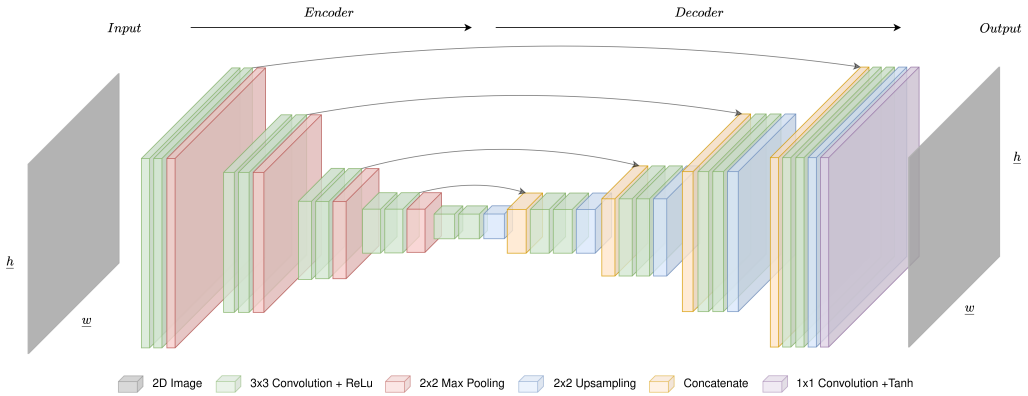


Figure 5.3: Architecture of the Generative Neural Network for Fruit Appearance Estimation. The network consists of an encoder (left) and a decoder (right) with skip connections between corresponding layers.

- Four downscaling layers, each containing two 3×3 convolutional layers followed by a 2×2 max pooling layer.
- Convolutional layers have filter sizes of 32, 64, 128, and 256, respectively.
- Activation function: Rectified Linear Unit (ReLU).

Decoder:

- Five upscaling layers, each containing two 3×3 convolutional layers followed by a 2×2 upsampling layer.
- Convolutional layers have filter sizes of 512, 256, 128, 64, and 32, respectively.
- Skip connections are used by concatenating outputs from corresponding encoder layers.
- Activation function: ReLU.

Output Layer:
A 1×1 convolutional layer with a sigmoid activation function, producing the final reconstructed depth image $\hat{\mathbf{D}}'_M$.

All convolutional layers use a stride of 1. The network is trained using the binary cross-entropy loss function, comparing the generated image with the ground truth non-occluded image.

Training the Network

The network is trained on a dataset of synthetic fruit images with simulated occlusions. Non-occluded fruit images are generated using synthetic ellipses colored to resemble the target fruits (e.g., oranges). Occlusions are simulated by randomly cropping parts of the fruit images.

The dataset comprises 100 images, split into 70 training images and 30 validation images. During training, the network learns to reconstruct the full fruit appearance from the occluded input images.

Estimating Fruit Properties

After training, the network can generate the complete estimated depth image $\hat{\mathbf{D}}'_M$ from an input occluded depth image $\mathbf{D}'_M$. From the reconstructed depth image, we compute various properties of the fruit essential for manipulation:

- **Centroid of the Estimated Fruit:** The centroid in the ROI image coordinates is calculated as:

$$\hat{u}'_c = \frac{1}{N_{\neq 0}} \sum_{(u,v)\in\mathcal{H}} u, \quad \hat{v}'_c = \frac{1}{N_{\neq 0}} \sum_{(u,v)\in\mathcal{H}} v, \tag{5.9}$$

  where $\mathcal{H} = \{(u,v) \in [0, d_{\text{ROI}}] \times [0, d_{\text{ROI}}] \mid \hat{\mathbf{D}}'_M(u,v) \neq 0\}$ is the set of pixels with non-zero depth values in the reconstructed image, and $N_{\neq 0}$ is the total number of such pixels.

- **Dimensions of the Estimated Fruit:** The width $\hat{w}_f$ and height $\hat{h}_f$ are determined by:

$$\begin{aligned} \hat{w}_f &= \hat{u}_{\max} - \hat{u}_{\min}, \\ \hat{h}_f &= \hat{v}_{\max} - \hat{v}_{\min}, \end{aligned} \tag{5.10}$$

  where:

$$\begin{aligned} \hat{u}_{\max} &= \max_{(u,v)\in\mathcal{H}} u, \quad \hat{u}_{\min} = \min_{(u,v)\in\mathcal{H}} u, \\ \hat{v}_{\max} &= \max_{(u,v)\in\mathcal{H}} v, \quad \hat{v}_{\min} = \min_{(u,v)\in\mathcal{H}} v. \end{aligned} \tag{5.11}$$

- **Mapping Centroid Back to Original Image Coordinates:** Using the function $\mathbf{r}$ defined in eq. (5.8), we map the centroid coordinates back to the original image:

$$\hat{u}_c = \hat{u}'_c + u_{TL}, \quad \hat{v}_c = \hat{v}'_c + v_{TL}. \tag{5.12}$$

- **Projecting Centroid to 3D Space:** The 3D position of the fruit's center in the camera frame $\mathcal{F}_{\text{cam}}$ is obtained by:

$$\hat{\mathbf{p}}_c = \begin{pmatrix} \hat{x}_c \\ \hat{y}_c \\ \hat{z}_c \end{pmatrix} = \mathbf{m} \begin{pmatrix} \hat{u}_c \\ \hat{v}_c \\ \hat{\mathbf{D}}_M(\hat{u}_c, \hat{v}_c) \end{pmatrix}, \tag{5.13}$$

  where $\hat{\mathbf{D}}_M(\hat{u}_c, \hat{v}_c)$ is the depth value of the centroid in the masked depth image.

Under *Hypothesis 1* described in Section 5.1, in the following sections, we assume that the properties of the real (occluded) fruit are identical to those of the fruit estimated from the generative model. That is, the estimated position $\hat{\mathbf{p}}_c$, dimensions $\hat{w}_f$, $\hat{h}_f$, and centroid coordinates $(\hat{u}_c, \hat{v}_c)$ accurately represent the actual fruit. This assumption is justified when the generative model performs well in reconstructing the occluded regions.

5.2.3 Synthesis

The fruit identification process is a critical component of our approach to handling occlusions in agricultural environments. By segmenting the visible portions of the fruit and estimating its complete appearance with a generative neural network, we effectively address the challenges posed by occlusions.

This method increases the robot's perception capabilities, allowing it to accurately estimate the fruit's position and dimensions even when the fruit is only partially visible. These estimations are essential for planning precise manipulation and harvesting actions, and ultimately improving the robot's effectiveness in complex, unstructured environments.

In the following section, we build upon this foundation to develop strategies for detecting occluding branches and planning manipulation actions to clear occlusions, enabling the robot to access and harvest fruits that would otherwise be unreachable.

5.3 Occlusion Identification

This section details our method for occlusion identification, which involves: calculating the view gradient representing the presence and direction of occlusions, and utilizing a 3D extension of the Hough transform to detect branch line segments in space.

5.3.1 View Gradient

The view gradient is an important componant, which we define in our work [GKC24] to determine the presence and direction of occlusions obstructing the fruit. It is computed by comparing the estimated complete fruit appearance with the actual observed (and potentially occluded) fruit.

First, we generate two binary mask images from the depth images obtained in the fruit identification process:

- $\mathbf{M}' \in \mathcal{I} \times \{0,1\}$: a binary mask derived from the observed masked depth image $\mathbf{D}'_M$.
- $\hat{\mathbf{M}}' \in \mathcal{I} \times \{0,1\}$: a binary mask derived from the estimated complete fruit depth image $\hat{\mathbf{D}}'_M$ obtained from the generative neural network.

These masks are created by setting pixel values to 1 if their corresponding depth values are greater than zero (indicating the presence of fruit) and to 0 otherwise:

$$\mathbf{M}'(u,v) = \begin{cases} 1, & \text{if } \mathbf{D}'_M(u,v) > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{5.14}$$

$$\hat{\mathbf{M}}'(u,v) = \begin{cases} 1, & \text{if } \hat{\mathbf{D}}'_M(u,v) > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5.15}$$

We compute the difference between these two masks, to identify the occluded regions:

$$\bar{\mathbf{M}}' = \left| \mathbf{M}' - \hat{\mathbf{M}}' \right|, \tag{5.16}$$

where $\bar{\mathbf{M}}' \in \mathcal{I} \times \{0,1\}$ represents the pixels with discrepancy between the observed and estimated fruit masks (i.e., the occluded areas).

To determine whether the fruit is occluded, we calculate the ratio of the area of the occluded regions to the area of the observed fruit:

$$r_o = \frac{\text{area}(\bar{\mathbf{M}}')}{\text{area}(\mathbf{M}')}. \tag{5.17}$$

Here, area$(\cdot)$ denotes the number of pixels with value 1 in the corresponding mask, $(\cdot)$.

If the ratio r_o exceeds a predefined threshold (e.g., $r_o > 0.1$), we consider the fruit to be occluded. This threshold can be adjusted based on empirical observations to balance sensitivity and specificity in occlusion detection.

When an occlusion is detected, we compute the centroid of the occluded regions in the ROI image coordinates, $(\bar{u}'_c, \bar{v}'_c)$, with a calculation similar to ((5.6)):

$$\bar{u}'_c = \frac{1}{N_o} \sum_{(u,v)\in\mathcal{O}} u, \quad \bar{v}'_c = \frac{1}{N_o} \sum_{(u,v)\in\mathcal{O}} v. \tag{5.18}$$

Here, $\mathcal{O} = \{(u,v) \in \mathcal{I} \mid \bar{\mathbf{M}}'(u,v) = 1\}$ is the set of occluded pixels, and N_o is the total number of occluded pixels.

Next, we compute the occlusion direction vector $\vec{\mathbf{O}}_f \in \mathbb{R}^2$, referred to as the *view gradient*. It indicates the direction from the fruit's center to the occluded region:

$$\vec{\mathbf{O}}_f = \frac{(\bar{u}'_c - \hat{u}'_c, \bar{v}'_c - \hat{v}'_c)^\top}{|\bar{u}'_c - \hat{u}'_c, \bar{v}'_c - \hat{v}'_c|}, \tag{5.19}$$

where $(\hat{u}'_c, \hat{v}'_c)$ is the centroid of the estimated fruit from Equation (5.6), and $|\cdot|$ denotes the Euclidean norm.

The view gradient $\vec{\mathbf{O}}_f$ is a unit vector pointing from the fruit's estimated center to the center of the occluded area within the ROI. This vector will serve, to plan the direction in which the robot should push the occluding object to clear the fruit's view.

An illustration of the view gradient is given in Figures 5.2 and 5.4, where the red arrow represents $\vec{\mathbf{O}}_f$.

5.3.2 Branch Line Segments

To identify the branches causing occlusions, we develop a 3D extension of the Hough transform, for detecting line segments representing branches in three-dimensional space. The Algorithm 3 is used in this section to identify the branches.

Branch Segmentation: First, we segment the branch pixels from the image, using color information. As for fruit segmentation, we analyze the Hue (H), Saturation (S), and Value (V) components, to distinguish branches from the background. Branches typically have color values which characterize them from other elements in the scene.

Binary Image Formation: We create a binary image where pixels corresponding to branches are set to 1, and all other pixels are set to 0. Additionally, we perform depth thresholding by setting pixels with valid depth values (i.e., $z > 0$) to 1 to ensure that only pixels with valid 3D information are considered.

2D Hough Transform: We apply the standard 2D Hough transform [DH72] to the binary image, to detect straight line segments in the image plane. The Hough transform is particularly effective in detecting lines in images, since it transforms points in the image space to a parameter space where collinear points in the image space correspond to intersecting curves in the parameter space.

The detected lines $\mathbf{l}_{2D}$ are defined by their end points (u_{l1}, v_{l1}), (u_{l2}, v_{l2}) in the image plane as shown in Figure 5.5.

$$\mathbf{l}_{2D} = \left((u_{l1}, v_{l1}) , (u_{l2}, v_{l2}) \right) . \tag{5.20}$$

Similar Lines Filtering: To avoid redundancy, we define a distance threshold ρ_{th} and an angle threshold θ_{th} to remove similar lines. To this end, we define a distance threshold ρ_{th} and an angle threshold θ_{th}. We compare all detected lines and remove similar lines if their Hough distances wrt ρ and θ are smaller than ρ_{th} and θ_{th}.

3D Extension of the Hough Transform: To extend the detected 2D lines into 3D space, we perform the following steps:

1. **Plane Definition:** For each detected 2D line segment $\mathbf{l}_{2D}$, we define a plane $\mathbf{\Pi}$ in the camera frame $\mathcal{F}_{\text{cam}}$ that passes through the camera center (optical center) and the line segment in the image plane. This plane can be determined by projecting the line segment's end points in 3D space and computing the normal vector.

   The end points are projected using the mapping $\mathbf{m}$ from Equation (5.1), with the z coordinate set to the focal length f, to place the points at a consistent depth:

   $$(x_l, y_l, z_l)^\top = \mathbf{m} \left(u_l, v_l, f \right) . \tag{5.21}$$

   The plane $\mathbf{\Pi}$ is then defined by its normal vector $\mathbf{N}_\Pi = (A, B, C)^\top$ as:

   $$\mathbf{N}_\Pi = (x_{l1}, y_{l1}, z_{l1}) \times (x_{l2}, y_{l2}, z_{l2}) . \tag{5.22}$$

   The plane equation is given by:

$$Ax + By + Cz = 0. \tag{5.23}$$

2. **Point Cloud Filtering:** We project the segmented branch pixels in 3D to form a point cloud $\mathbf{P}_b$:

$$\mathbf{P}_b = \{\mathbf{m}(u, v, z_{uv}) \mid \mathbf{M}_b(u, v) = 1\}, \tag{5.24}$$

where $\mathbf{M}_b$ is the binary mask of branch pixels. We then reduce the points in $\mathbf{P}_b$ by selecting only those which are within a distance threshold d_Π from the plane $\mathbf{\Pi}$:

$$\mathbf{P}_b^\Pi = \left\{ \mathbf{p} \in \mathbf{P}_b \mid \frac{|Ax_p + By_p + Cz_p|}{|\mathbf{N}_\Pi|} \leq d_\Pi \right\}, \tag{5.25}$$

where $\mathbf{p} = (x_p, y_p, z_p)^\top$.

3. **Plane Alignment:** We rotate the points in $\mathbf{P}_b^\Pi$ to align the plane $\mathbf{\Pi}$ with the camera's optical axis $\mathcal{F}_{\text{cam}}^z$. This simplifies the problem to detecting lines in a plane parallel to the image plane. The rotation matrix $\mathbf{R}$ is calculated using Rodrigues' rotation formula:

$$\begin{aligned} \mathbf{K} &= \mathcal{F}_{\text{cam}}^z \times \mathbf{N}_\Pi, \\ \cos(\alpha) &= \frac{\mathcal{F}_{\text{cam}}^z \cdot \mathbf{N}_\Pi}{|\mathcal{F}_{\text{cam}}^z|\,|\mathbf{N}_\Pi|}, \\ \mathbf{R} &= \mathbf{I} + \sin(\alpha)[\mathbf{K}]_\times + (1 - \cos(\alpha))[\mathbf{K}]_\times^2, \end{aligned} \tag{5.26}$$

with $[\mathbf{K}]_\times$ the skew-symmetric matrix of $\mathbf{K}$, and $\mathbf{I}$ the 3-dimensional identity matrix.

The rotated points are:

$$\mathbf{P}_b^{\Pi\prime} = \mathbf{R}\mathbf{P}_b^\Pi. \tag{5.27}$$

4. **2D Hough Transform on Rotated Points:** We form a new 2D image from $\mathbf{P}_b^{\Pi\prime}$ and reapply the Hough transform to detect straight line segments, removing similar lines to get $\mathbf{l}'_{2D}$.

5. **Mapping Back to 3D:** The detected 2D lines are mapped back to 3D space, applying (5.13) in Equation (5.28).

$$\mathbf{l}'_{3D} = \mathbf{m}\left(u'_l, v'_l, z'_l\right). \tag{5.28}$$

6. **Inverse Rotation:** We apply the inverse rotation to the 3D lines to obtain the final 3D line segments in the original camera frame:

$$\mathbf{l}_{3D} = \mathbf{R}^{-1}\mathbf{l}'_{3D}. \tag{5.29}$$

We compile all the detected 3D line segments into a set $\mathbf{l}_b$, representing the branches in the scene:

$$\mathbf{l}_b = \left\{ \mathbf{l}_{3D}^{(i)} \mid i = 1, \ldots, N_l \right\}, \tag{5.30}$$

where N_l is the number of detected lines.

Algorithm 3 Line finding algorithm

Require: Color and Depth information
Ensure: 3D branch line segments $\mathbf{l}_b$ which contains all the $\mathbf{l}_{3D}$
1: Branch segmentation in HSV
2: Form an image from segmented pixels
3: Apply 2D Hough line transform on the image: $\mathbf{l}_{2D}$
4: **for** $\mathbf{l} \in \mathbf{l}_{2D}$ **do**
5: Compute the plane $\mathbf{\Pi}$
6: Filter $\mathbf{P}_b$ by distance to Π: $\mathbf{P}_b^{\Pi} \leftarrow \mathbf{P}_b < d_{\Pi}$
7: Rotate the points $\mathbf{P}_b^{\Pi}$: $'\mathbf{P}_b^{\Pi} \leftarrow \mathbf{R}(\mathbf{K}, \alpha) \cdot \mathbf{P}_b^{\Pi}$
8: Form a new image from $'\mathbf{P}_b^{\Pi}$
9: Apply 2D Hough transform on the new image: $\mathbf{l}'_{2D}$
10: Map $\mathbf{l}'_{2D}$ back to 3D: $\mathbf{l}'_{3D} \leftarrow \mathbf{m}(\mathbf{l}'_{2D})$
11: Apply inverse rotation to $\mathbf{l}'_{3D}$: $\mathbf{l}_{3D} \leftarrow \mathbf{R}(\mathbf{K}, -\alpha) \cdot \mathbf{l}'_{3D}$
12: **end for**

All line finding steps are described in Algorithm 3, where the image processing and line segment post-processing steps are shortened by *2D Hough line transform*.

5.3.3 Synthesis

In this section, we introduced a method for branch occlusion identification and characterization. The method calculates a *view gradient* pointing from the centroid of the occluded fruit from the centroid of the missing part of the fruit and detects branch line segments in 3D space. The view gradient provides a quantitative measure of the occlusion's direction wrt the fruit, setting the ground for the robot manipulation actions. The detection of branches that cause occlusions, will also provide essential information for planning how to interact with the said branches.

By combining image processing techniques with 3D geometric transformations, we can effectively identify occlusions and the branches responsible for them. This approach ensures that the robot can selectively manipulate occluding elements to clear its line of sight, improving fruit visibility and accessibility for harvesting.

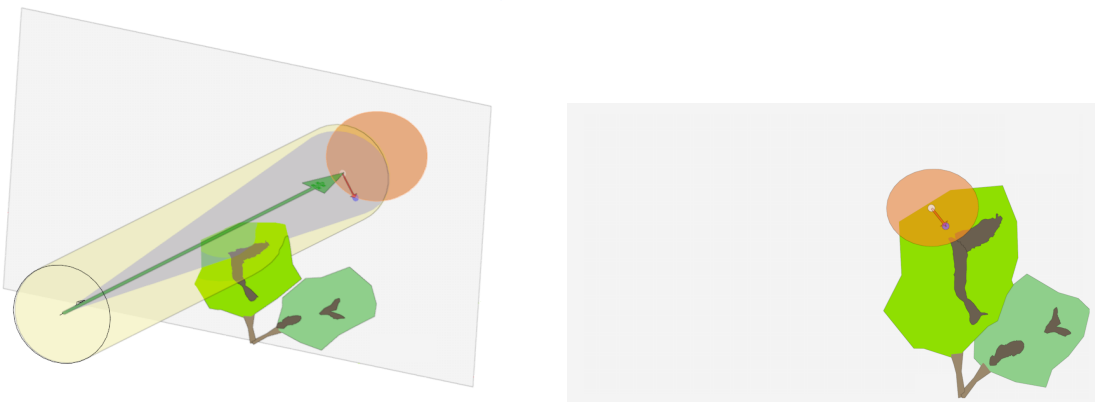


Figure 5.4: Field of view from the camera (left) and Image (right). The figure shows: fruit centroid (white circle) $(\hat{u}_c, \hat{v}_c)$*, occlusion centroid (blue circle)* $(\bar{u}_c, \bar{v}_c)$ *, view frustum (blue cone), its center line* $\mathcal{F}^z_{fruit}$ *(green arrow) and view gradient* $\vec{\mathbf{O}}_f$ *(red arrow).*

5.4 Occlusion Clearance

After identifying the occluding branches and estimating the fruit's appearance, we aim to physically clear the occlusion by pushing the appropriate branch. This section details the method for selecting the branch to push and planning the push action to effectively remove the occlusion without causing damage to the fruit or the robot.

5.4.1 Line Decision

The first step in clearing the occlusion is to determine which branch to push. Algorithm 4 summarizes the steps involved in line decision. We define the fruit frame $\mathcal{F}_{\text{fruit}}$, as illustrated in Figure 5.4. In this frame, the z-axis points from the camera towards the fruit. The transformation from the camera frame $\mathcal{F}_{\text{cam}}$ to the fruit frame $\mathcal{F}_{\text{fruit}}$ is given by:

$$\begin{array}{rccc} \mathbf{f}: & \mathcal{F}_{\text{cam}} & \rightarrow & \mathcal{F}_{\text{fruit}} \\ & \begin{pmatrix} x \\ y \\ z \end{pmatrix} & \mapsto & \mathbf{R}^{\mathbf{p}_c} \begin{pmatrix} x^\dagger \\ y^\dagger \\ z^\dagger \end{pmatrix} \end{array} \tag{5.31}$$

where $\mathbf{R}^{\mathbf{p}_c}$ is the rotation matrix aligning the camera's z-axis $\mathcal{F}^z_{\text{cam}}$ with the vector pointing to the fruit's centroid $\mathbf{p}_c$. This rotation matrix can be calculated using Rodrigues' rotation formula, as in Equation (5.26).

View Frustum Definition

Under **Hypothesis 2** described in Section 5.1 (fruits are considered perfectly elliptical in shape), the view frustum connecting the camera and the fruit is modeled as a

cone with an elliptical cross-section. This cone, depicted in Figure 5.4, is defined by the equation:

$$\frac{x^{\dagger 2}}{\mathbf{m}(w_f)^2} + \frac{y^{\dagger 2}}{\mathbf{m}(h_f)^2} = \frac{z^{\dagger 2}}{|\mathbf{p}_c|^2}, \quad \text{subject to } |z^\dagger| \leq |\mathbf{p}_c| \tag{5.32}$$

where $\mathbf{m}(w_f)$ and $\mathbf{m}(h_f)$ are the width and height of the fruit in meters, obtained through the mapping $\mathbf{m}$.

Selecting the Branch to Push

We aim to select the branch that, when pushed, will effectively clear the occlusion. We consider the set of detected branch lines $\mathbf{l}_{\text{branch}}$ and filter them to obtain the subset $\mathbf{l}_{\text{branch}}^{\text{view}}$ consisting of lines that satisfy:

- Their distance to the view frustum V is less than a predefined threshold d_V.
- Their depth z_{branch} is less than the fruit centroid depth z_c (i.e., they are between the camera and the fruit).

Each line $\mathbf{l}_{3D}$ in $\mathbf{l}_{\text{branch}}^{\text{view}}$ is projected onto the image plane via the inverse mapping $\mathbf{m}^{-1}$ to obtain $\mathbf{l}_{2D}$. We compute the angle α between $\mathbf{l}_{2D}$ and the view gradient $\vec{\mathbf{O}}_f$, which indicates the direction of the occlusion:

$$\alpha = \arccos\left(\frac{\mathbf{l}_{2D} \cdot \vec{\mathbf{O}}_f}{|\mathbf{l}_{2D}| \cdot |\vec{\mathbf{O}}_f|}\right) \tag{5.33}$$

We define γ as the deviation of α from $\frac{\pi}{2}$:

$$\gamma = \left|\alpha - \frac{\pi}{2}\right| \tag{5.34}$$

The line $\mathbf{l}_{\text{push}}$ with the smallest γ is the one most perpendicular to the view gradient, making it the optimal candidate for pushing to clear the occlusion.

Algorithm 4 Line Decision Algorithm

Input: Line segments $\mathbf{l}_{\text{branch}}$, Occlusion direction $\vec{\mathbf{O}}_f$
Output: Line to push $\mathbf{l}_{\text{push}} \leftarrow \mathbf{l}_{3\text{D}}$ with the smallest γ

1: Filter branch segments: $\mathbf{l}_{\text{branch}}^{\text{view}} \leftarrow \{\mathbf{l} \in \mathbf{l}_{\text{branch}} \mid \text{dist}(\mathbf{l}, V) < d_V \text{ and } z_{\text{branch}} < z_c\}$
2: **for** $\mathbf{l}_{3\text{D}} \in \mathbf{l}_{\text{branch}}^{\text{view}}$ **do**
3: Project $\mathbf{l}$ onto the image plane: $\mathbf{l}_{2\text{D}} \leftarrow \mathbf{m}^{-1}(\mathbf{l}_{3\text{D}})$
4: Compute the angle between $\mathbf{l}_{2\text{D}}$ and $\vec{\mathbf{O}}_f$: $\alpha \leftarrow \arccos\left(\frac{\mathbf{l}_{2\text{D}} \cdot \vec{\mathbf{O}}_f}{\|\mathbf{l}_{2\text{D}}\| \|\vec{\mathbf{O}}_f\|}\right)$
5: Check how far α is from $\frac{\pi}{2}$: $\gamma \leftarrow \left|\alpha - \frac{\pi}{2}\right|$
6: **end for**

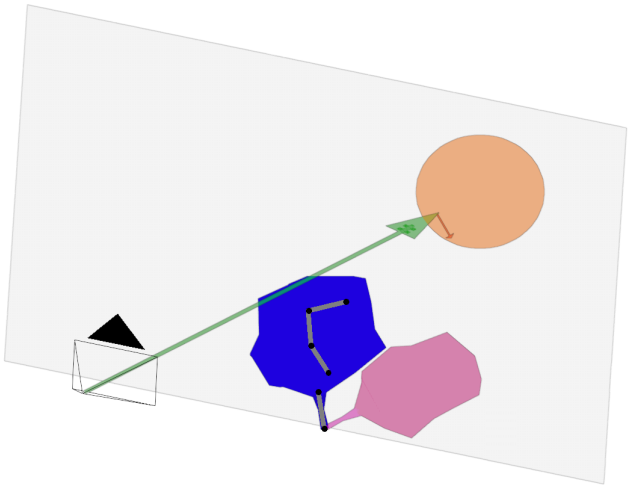


Figure 5.5: Line segments (grey) and their extremities (black circles). The pushing line $\mathbf{l}_{push}$ $\mathbf{l}_{push}$ is selected as the topmost according to Algorithm 4.

5.4.2 Push Action

With the pushing line $\mathbf{l}_{\text{push}}$ identified, we proceed to determine the initial pushing point and execute the push action.

Determining the Pushing Point

If $\mathbf{l}_{\text{push}}$ intersects the view frustum V, we select the intersection point closest to the robot to minimize the risk of self-occlusion:

$$\mathbf{p}_{\text{push}} = (\mathbf{l}_{\text{push}} \cap V)_{\text{closest to robot}} \tag{5.35}$$

If there is no intersection, we select the point on $\mathbf{l}_{\text{push}}$ closest to the view frustum:

$$\mathbf{p}_{\text{push}} = \arg \min_{\mathbf{p} \in \mathbf{l}_{\text{push}}} \text{dist}(\mathbf{p}, V) \tag{5.36}$$

Figure 5.5 demonstrates the p_{push} and l_{push}

Determining the Push Magnitude

The magnitude of the push d_{push} is set to the largest distance between any two points in the estimated occlusion mask $\mathbf{M}'$:

$$d_{\text{push}} = \max_{\mathbf{p}_i, \mathbf{p}_j \in \mathbf{M}'} |\mathbf{p}_i - \mathbf{p}_j| \tag{5.37}$$

Executing the Push Action

The robot end-effector moves from $\mathbf{p}_{\text{push}}$ along the view gradient direction $\vec{\mathbf{O}}_f$ for a distance of d_{push}. The end-effector orientation is maintained at an angle of $\frac{\pi}{4}$ radians relative to the camera's principal axis, directed away from the image plane to prevent self-occlusions.

5.4.3 Synthesis

By selecting the optimal branch to push and carefully planning the push action, the robot can effectively clear occlusions and improve fruit visibility for harvesting. The method ensures that the push action is executed safely, minimizing the risk of damage to the fruit or the robot, and takes into account the geometric relationships between the camera, the fruit, and the occluding branches.

5.5 Experiments

To validate our proposed methods for fruit identification, occlusion detection, and occlusion clearance, we conducted a series of experiments using different fruits (apples, lemons, and oranges), under varying lighting conditions. Additionally, we performed a real-world experiment using the BAZAR robot, to demonstrate the practical applicability of our approach in handling occlusions during fruit harvesting.

5.5.1 Setup

We conducted four experiments across three distinct scenes, designed to simulate realistic agricultural environments with varying degrees of occlusions and lighting conditions. The experiments aimed at testing the perception algorithms and our strategy for handling occlusions. We used artificial fruits, providing a controlled setting to assess the performance of our system without the variability introduced by natural fruits.

For the real robot experiment, we used the right arm of the BAZAR robot [CPN+19], i.e., a 7-DOF KUKA LWR manipulator. An Intel RealSense D435 RGB-D camera was mounted on the head of BAZAR. To perform the push action required to clear the occlusion, we attached a metal rod to the center of the last arm link. This rod servs as a simple yet effective end-effector.

The software framework was built using the Robot Operating System (ROS) Noetic for robot control and communication. We used the MoveIt motion planning framework for trajectory planning and execution. The Intel RealSense SDK provided camera acquisition functionalities, while image processing tasks including segmentation, filtering, and feature extraction, were implemented using OpenCV.

The remaining software implementation details and parameters are described in Appendix A.

5.5.2 Perception Results

Fruit Appearance Estimation

We tested the fruit appearance estimation method on images of apples, lemons, and oranges under different lighting conditions and occlusion scenarios. Figure 5.6 presents the results of the fruit estimation process for various fruits. In each row, from left to right, we display:

1. The segmented fruit in the depth image, serving as the input to the fruit estimation model.
2. The estimation result obtained from the generative neural network.
3. The difference between the input and the output, highlighting the occluded regions.
4. The calculated view gradient $\vec{\mathbf{O}}_f$ (red arrow) overlaid on the estimated fruit.

The results demonstrate that the generative model successfully predicts the complete shape of the fruit, even when significant portions are occluded. Although the depth estimation is not accurate, due to the occluded regions, the predicted shapes are still precise enough for the following processing steps. The binary masks derived from these estimations effectively highlight occluded areas, to calculate the view gradient $\vec{\mathbf{O}}_f$.

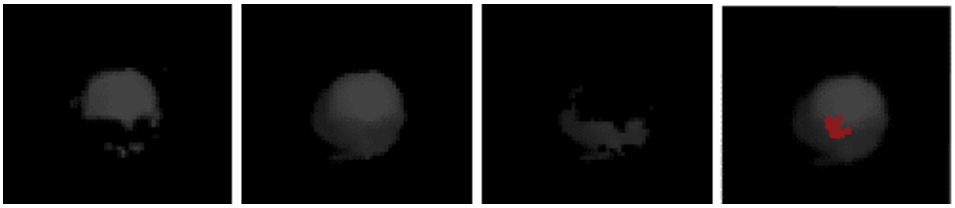
Occlusion Detection and View Gradient Calculation

By comparing the estimated fruit with the segmented input, we calculated the occlusion mask $\bar{\mathbf{M}}'$ and determined the presence of occlusions. The view gradient $\vec{\mathbf{O}}_f$, shown as a red arrow in Figure 5.6, indicates the direction of the occlusion relative to the fruit's centroid. This vector will be used for planning the direction in which the robot should manipulate the occluding object to clear the fruit's view.

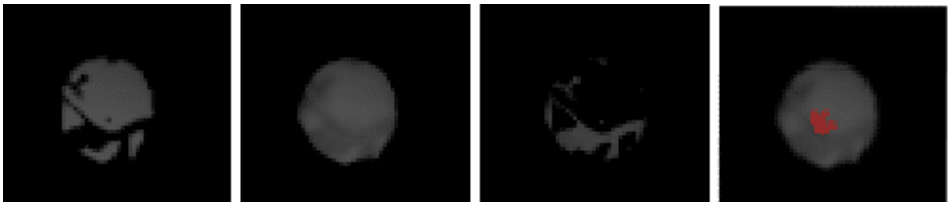
5.5.3 Occlusion Identification Results

Branch Line Detection

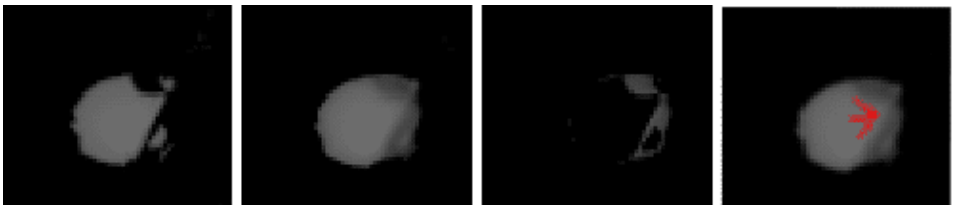
We applied the 3D extension of the Hough transform to detect branch line segments in the scenes. Figure 5.7 illustrates the result of the line detection results before and after the removal of similar lines. In the left image, the initial set of detected lines includes redundant lines due to minor variations in detection parameters. By filtering with thresholds $\rho_{th} = 25$ pixels and $\theta_{th} = \frac{\pi}{6}$ radians, we eliminate similar lines, resulting in a cleaner representation of the branch structures in the right image.



(a) Apple under standard lighting conditions with partial occlusion by leaves.



(b) Orange with significant occlusion due to branches and leaves.



(c) Lemon with complex occlusion patterns and variable lighting.



(d) Orange (Top View), with different occlusion angles.

Figure 5.6: Fruit estimation results. From left to right in each subfigure: segmented fruit in the depth image input to the fruit estimator, estimation result, difference between input and output, and view gradient (red vector) on the estimated fruit.

Line Selection for Pushing

We identified candidate branches for pushing, based on their proximity to the view frustum and their alignment with the view gradient. Figure 5.8 shows different scenes with the branch and fruit segmentation results, including detected branch lines $\mathbf{l}_{\text{branch}}$, the selected pushing line $\mathbf{l}_{\text{push}}$, and the pushing point $\mathbf{p}_{\text{push}}$. Line detection results are shown in Figure 5.8.

The outcomes indicate that branches are accurately predicted, and the selected pushing lines and points are reasonable from an empirical standpoint. This demonstrates the effectiveness of our line selection algorithm, in identifying the optimal branch for manipulation.

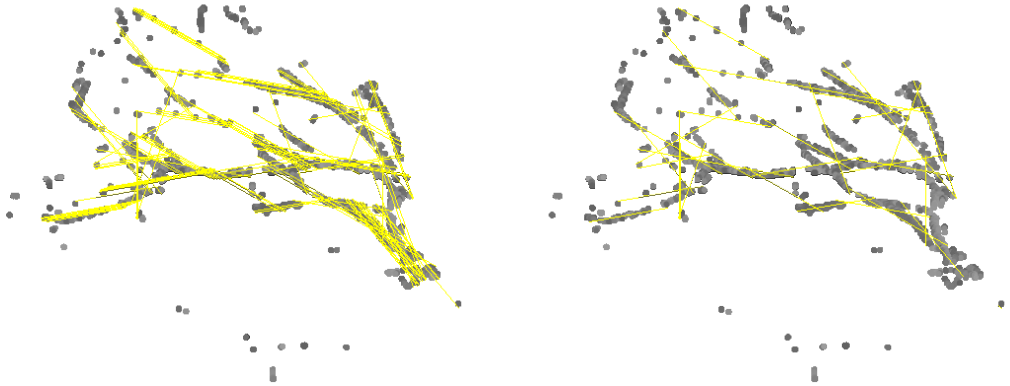


Figure 5.7: Line detection before (left) and after (right) removing similar lines.

5.5.4 Occlusion Clearance Results

Robot Experiment

We conducted a real robot experiment using BAZAR, to validate the effectiveness of our occlusion clearance strategy in a practical setting. The experiment involved the robot performing a push action to clear an occluding branch and improve the visibility of an artificial orange.

Figure 5.9 shows the robot in action during the branch pushing task. The sequence illustrates the robot approaching the pushing point and executing the push action along the calculated direction. The robot successfully moved the occluding branch out of the way, resulting in a clear view of the fruit.

5.5.5 Synthesis

The experiments conducted demonstrate the robustness and effectiveness of our proposed methods across different fruit types, scenes, and environmental conditions. Key observations include:

- **Fruit Appearance Estimation**: The generative neural network effectively estimated the complete appearance of partially occluded fruits. The reconstructed shapes closely matched the actual fruit shapes, providing reliable data for occlusion detection.

- **Occlusion Detection**: The view gradient calculation accurately identified the direction of occlusions, enabling precise planning for occlusion clearance actions.

- **Branch Line Detection**: The 3D extension of the Hough transform successfully detected branch line segments, even in complex scenes with multiple branches and varying lighting conditions.



Figure 5.8: From left to right: original scene, segmentation results, and line detection results. In the line detection image: detected and filtered lines are yellow, lines within the distance threshold are blue, and the selected pushing line is green. The pushing point is marked in red.



Figure 5.9: Robot branch pushing demonstration using BAZAR. The robot successfully clears the occlusion by pushing the selected branch, enhancing fruit visibility for harvesting.

- **Pushing Line Selection**: The algorithms for selecting the optimal branch to push performed effectively, with the selected lines leading to successful occlusion clearance.

- **Robot Execution**: The real robot experiment validated the practical applicability of our approach. The robot was able to execute the planned push action accurately, clearing the occlusion and enhancing fruit visibility for subsequent harvesting tasks.

The experimental results validate our proposed methods for perception and occlusion handling in agricultural environments. Across different scenes, fruit types, and conditions, our approach demonstrated fruit detection and occlusion identification. The successful execution of the push action by BAZAR demonstrates the practical application of our system in real-world scenarios.

We have developed a framework that integrates perception methods with manipulation and enables the harvesting robot to handle occlusions commonly found in agricultural environments. The experiments highlight the potential of our dual-arm robotic framework to improve automated fruit harvesting and lead to a reliable robotic harvesting in real-world.

5.6 Conclusion and Literature Context

In this chapter, we have presented a novel approach for improving fruit visibility and reachability by pushing branches to improve fruit detection and harvesting accuracy, by addressing the challenges posed by occlusion. Our method integrates perception techniques with manipulation strategies, enabling the robot to detect partially occluded fruits and to actively clear occlusions to help harvesting.

We used generative neural networks within an encoder-decoder architecture, to estimate the complete appearance of partially occluded fruits based on their visible portions. This approach allows for accurate fruit localization even when significant parts of the fruit are hidden behind branches or leaves. The use of RGB-D camera data and depth image segmentation provided the necessary 3D information to support this estimation.

We developed a 3D extension of the Hough transform to detect branch directions in 3D space, enabling the identification of branches that contribute to occlusions. By calculating the view gradient, we determined the direction of the occlusions relative to the fruit, which guided the robot push actions to clear the occluding branches.

Experimental results demonstrated the robustness and effectiveness of our framework across various conditions, including different fruit types (apples, lemons, and oranges) and environmental settings. The robot accurately detected and estimated occluded fruits, identified occluding branches, and performed manipulation actions to clear them, which improved fruit visibility and accessibility for harvesting.

Our work stands out in several key ways compared to existing research. Many previous studies have focused solely on fruit detection using deep learning algorithms [SdSdSA20, KC20, KWWM19]. While effective under certain conditions, these methods often struggle when fruits are partially or fully hidden by branches or leaves and do not offer solutions for handling such occlusions. Some studies have tried to estimate occlusions using heuristic methods or statistical models [FWWL15, KLV14], but these approaches may not work well in complex, real-world environments. Our approach uniquely combines perception with manipulation strategies. This integration allows the robot to detect fruits in the presence of occlusions and to physically interact with the environment to clear obstructed views, to improve effectiveness in complex settings.

We use a generative neural network to predict the complete shapes of fruits that are partially hidden. This is a significant improvement over traditional methods which relied on simple geometric models, without accounting for the variability of real-world scenarios [GFZ+20, FFM+18]. Our deep learning approach accurately reconstructs the hidden portions of the fruit, increasing detection accuracy even when significant occlusions are present.

While branch detection has been recognized as important for occlusion handling [CCT+15, GWWL22], prior methods have not fully integrated this with manipulation strategies. We extended the Hough transform to 3D for detecting the directions of branches that might be covering the fruits. Our method identifies and manipulates occluding branches, based on their 3D spatial relationships with the fruit and robot. This allows the robot to physically clear obstructions, directly improving fruit visibility and accessibility for harvesting.

Alternative approaches, such as manually removing leaves or using airflow to reduce occlusions [LSL+22], are labor-intensive and not always practical. Our method offers an automated solution that adapts to various conditions, without requiring significant manual intervention. Instead of relying solely on estimations under occlusions, our robot actively clears occlusions to reveal the fruits, reducing uncertainty and increasing the reliability of the harvesting process.

By combining these elements, our work addresses the challenges posed by occlusions in fruit harvesting. We provide a system which accurately detects and estimates fruits in the presence of occlusions and actively clears obstructions to help harvesting. However, our approach has several limitations. First, manipulating branches assumes they are flexible and won't damage the tree or robot, which may not hold for all vegetation types. Second, the generative model accuracy relies on diverse training data, insufficient variation can reduce reliability. Additionally, the method does not account for dynamic factors like wind, which could affect perception and manipulation. Addressing these issues is in the future will be necessary to deploy the robot into real orchards.

CHAPTER 6

Conclusion

6.1 Summary

This thesis has addressed the challenge of automating fruit harvesting in unstructured agricultural environments by using the advantages of a dual-arm robotic platform. Our research was motivated by the limitations of existing single-arm robotic harvesters, which struggle with dexterity, efficiency, and adaptability, particularly in complex tasks such as simultaneous cutting and collecting, and when operating under significant occlusions caused by foliage.

We proposed an approach to increase these aspects, via the design and implementation of perception, planning, and control components. The research progressed through three significant stages:

Initial Development of the Dual-Arm Harvesting Robot: We introduced a dual-arm robot equipped with an RGB-D camera and specialized end-effectors for cutting and collecting fruits. A cooperative control framework based on **Hierarchical Quadratic Programming (HQP)** was developed, to enable coordinated movements between the two arms, which allow simultaneous cutting and collecting. The perception system combined deep learning, specifically **YOLOv5** for fruit detection and **DeepSORT** for tracking, with depth information from RGB-D sensors to detect, track, and localize fruits and obstacles in the environment. The robot's motion planner takes into account kinematic constraints and collision avoidance to ensure safety.

Towards Real-World Adaptation: Building upon the initial framework, we upgraded the perception module, by developing fruit tracking algorithm which uses 3D spatial information to improve tracking stability under complex occlusions. An optimized fruit scheduling procedure based on the **Traveling Salesperson Problem (TSP)** was implemented to minimize travel distance and reduce harvesting time. Motion control was improved by adopting a global motion planning approach, treating each arm

independently to generate optimized, collision-free trajectories. Experiments with different robots (**BAZAR** and **Baxter**) and fruit types (apples and oranges) demonstrated the system's versatility, robustness, and adaptability to various hardware configurations and operational environments.

Fruit Occlusion Handling: Recognizing occlusions as a critical challenge in agricultural environments, we developed perception and manipulation strategies to handle them effectively. We used **generative neural networks** within an encoder-decoder architecture to estimate the complete appearance of partially occluded fruits, utilizing **Convolutional Autoencoders (CAEs)** to reconstruct the full fruit shape from partial observations. A **3D extension of the Hough transform** was developed to detect branch directions in three-dimensional space, allowing the robot to accurately identify occluding branches. By integrating these perception techniques with manipulation strategies, the robot could physically clear occlusions, by using one arm to push aside obstructing branches to increase fruit visibility and accessibility for harvesting by the other arm.

6.2 Achievements

Looking back at the initial objectives of this thesis, we have successfully achieved the main goals:

Dual-Arm Harvesting Platform: We designed and implemented a dual-arm platform that effectively demonstrates dexterity in fruit harvesting tasks. Through hardware integration of robotic arms, custom end-effectors, sensors, along with a modular system architecture, the robot demonstrated coordinated dual-arm manipulation. Control strategies based on HQP were implemented, to manage kinematic constraints, self-collisions, and collision avoidance, and to perform the actions needed for simultaneous cutting and collecting.

Perception under Occlusions : Novel perception methods were developed using deep learning and image processing, to detect and localize fruits under challenging conditions, i.e., in particular significant occlusions. Generative neural networks were used to estimate the full appearance of partially occluded fruits, which increased detection accuracy and localization. Depth sensing provided precise 3D spatial information for obstacle detection and handling of occlusions. The extended 3D Hough transform detected occluding branches, allowing the robot to plan and execute actions for clearing these obstructions.

Motion Planning and Control Methodologies : We designed motion planning algorithms to adapt the coordinated robot movements while considering obstacle avoidance and occlusion handling. Global motion planning methods, hierarchical task prioritization through HQP, and adaptive obstacle modeling were applied to achieve optimized, collision-free trajectories. A TSP-based scheduling algorithm was developed to optimize the sequence of fruit harvesting, minimizing travel distance and operational time. We developed simplified and efficient computation methods for obstacle avoidance to improve the performance.

Validate the System in Real-World Environments: The methodologies were tested using different robots (BAZAR and Baxter) and fruit types (apples and oranges). Experiments were conducted under varying lighting and environmental conditions, to demonstrate the system's robustness and adaptability. The successful execution of harvesting tasks in real-world scenarios, including handling complex occlusions and dynamic environments, confirmed the practical applicability of the proposed methods.

6.3 Discussion

Implications for Agricultural Robotics

The research presented in this thesis has significant implications for the field of agricultural robotics:

Efficiency and Productivity: Dual-arm robots, capable of performing simultaneous tasks with improved dexterity, can increase harvesting speed and reduce labor costs. This addresses labor shortages and efficiency challenges in the agricultural sector.

Adaptability to Unstructured Environments: Perception and motion planning strategies allow robots to operate effectively in unstructured outdoor environments. They can then handle variability in fruit appearance, occlusions, and changing environmental conditions.

Handling of Occlusions: The integrated approach, combining perception with manipulation strategies, provides a new direction for robotic harvesting platforms. Physically clearing occlusions allows for higher detection accuracy and reduces reliance on estimations under uncertainty, hence improving harvesting success rates.

Scalability and Versatility: The system's adaptability to different robots and fruit types shows its potential for broad application across various crops and agricultural settings. Its modular design improves scalability and integration with existing agricultural systems.

Contribution to Sustainable Agriculture: Automating the harvesting process advances sustainable agricultural practices by reducing waste, improving resource usage, and supporting precision agriculture. The system addresses labor shortages and increases efficiency, contributing to both the sustainability and profitability of agricultural operations.

Addressing the Challenges

Our research effectively tackled the challenges identified in the introduction:

Perception in Unstructured Environments: We developed perception methodologies which combine deep learning, image processing with depth sensing and generative modeling. The use of RGB-D cameras provided essential depth information, and the integration of CAEs enabled estimation of occluded fruits. The extended 3D Hough

transform allowed for detection of occluding branches, for planning occlusion handling actions.

Manipulation and Control: The dual-arm control strategies allowed the robot to perform cutting and collecting tasks simultaneously. We tested two control methodologies. HQP-based local motion planning allowed for hierarchical task prioritization, managing multiple objectives and constraints, including kinematic limits, self-collisions, and obstacle avoidance. Another control approach based on global motion planning is used because of its robustness.

Computational Constraints: Efforts were made to balance computational efficiency with performance. The perception algorithms operated in online through the use of efficient deep learning models and classical image processing techniques.

Robustness and Reliability:The system demonstrated robustness, through successful experiments under various conditions. Safety considerations were part of the design, with collision avoidance strategies and self-collision modeling. The system demonstrated reliability and adaptability to different robots and environments, and potential for practical deployment.

Limitations of the Current Work

While our research has made significant advancements, several limitations remain:

Controlled Environment Testing: Most experiments were conducted in laboratory settings with artificial trees and fruits. Although efforts were made to simulate challenging conditions, real-world agricultural environments present additional complexities, such as varying weather, natural lighting changes, uneven terrain, and biological variability in plants.

Hardware Constraints: The robotic platforms used in the experiments had limitations in terms of speed and precision. Safety considerations required operating at reduced speeds, which may not reflect the performance potential in commercial applications. Improvements in hardware could improve operational efficiency.

Simplified Environmental Models: Modeling fruits as ellipsoids and trees as rigid structures simplifies the perception and planning, but may not capture the full variability found in real orchards. Fruits and branches often have irregular shapes, and plants may move due to wind.

Limited Crop Types and Variability: The experiments were conducted on apples and oranges, with lemons used for testing occlusions. Further testing with a broader range of fruit types and varieties, particularly those with shapes other than ellipsoids (since we assume that the fruits are ellipsoidal), is needed. Such testing will verify the system's effectiveness across crops, with different physical characteristics.

6.4 Future Work

Building upon the findings and addressing the limitations identified in this research, several avenues for future work are proposed to improve the capabilities and applicability of dual-arm robotic systems for precision fruit harvesting.

Improving Perception under Severe Occlusions

The developed perception system detects and localizes fruits under moderate occlusions, but more severe occlusions remain a challenge. Future work can focus on state-of-the-art deep learning architectures, such as transformers or attention-based models, which may increase the robustness of detection by capturing global contextual information. This would allow the system to detect the presence of fruits even with limited visibility.

The use of multiple cameras from different viewpoints could offer complementary perspectives, helping reconstruct occluded regions by comparing images taken from different angles. Temporal information from consecutive frames could further improve this process, as the robot moves, providing new views of the environment.

Active vision strategies can be explored, where the robot adjusts its sensors or manipulates its environment to improve visibility of occluded fruits. This could involve moving the robot's arms to reposition obstructing foliage, rather than relying solely on passive perception.

Additional sensors like hyperspectral cameras or LiDAR may be integrated, to overcome the limitations of visual occlusions. For instance, hyperspectral imaging can detect fruits based on their spectral signatures, and LiDAR can capture depth information, providing useful data even when visual cues are absent.

Advanced Manipulation Techniques

Force-torque sensors and tactile feedback mechanisms could be embedded in the robot's end-effectors to allow for compliant manipulation. This would enable the robot to sense the forces during grasping and cutting and adjust its movements accordingly, to ensure gentle handling of delicate fruits.

Adaptive end-effectors could be designed to conform to the size and shape of different fruits. Mechanisms like underactuated grippers or soft robotic technologies would enable secure, albeit delicate grasping of fruits of various shapes and sizes.

Learning-based manipulation policies, including reinforcement learning or imitation learning, could enable the robot to refine its manipulation strategies. By learning from human demonstrations or through trial and error, the robot could adapt to complex or unpredictable tasks.

Improving dual-arm coordination could allow the robot to perform more complex agricultural tasks. Algorithms that enable synchronized movement of both arms in

dynamic environments, such as pruning or canopy management, would improve the robot's versatility.

Scaling to Different Crops and Environments

The robotic system needs to be adaptable to different types of crops and environmental conditions. Research in the specific characteristics of various fruits, such as clustering patterns, stem strength, and optimal harvesting, could inform the development of crop-specific end-effectors and manipulation strategies. Handling clusters of grapes or berries, for instance, requires different approaches compared to handling apples or oranges.

To ensure that the system operates reliably under a variety of environmental conditions, algorithms which can adapt to changes in lighting, temperature, and weather will be necessary. Color constancy techniques or the use of polarization imaging could reduce the effects of variable lighting conditions.

Mobility solutions that enable the robot to navigate uneven terrains, such as those found in orchards or fields, could improve its range of operation. Terrain mapping and path planning algorithms, combined with robust locomotion systems, would allow the robot to handle more challenging agricultural environments.

Hardware Enhancements

End-effectors designed for specific tasks or fruit types would increase the system's versatility. Modular designs, created using additive manufacturing, could allow for rapid prototyping and customization to meet specific agricultural needs.

To ensure durability in agricultural environments, the system could be engineered to withstand dust, moisture, and temperature extremes. Appropriate materials should be selected, and components sealed against contaminants, while also considering ease of maintenance.

Field Testing in Real Agricultural Settings

Field testing is essential to evaluate the system's performance in real-world conditions. Pilot studies in commercial orchards and farms could reveal operational challenges, including integration with existing farm machinery, and handling unexpected environmental factors.

Gathering user feedback during field testing could help refine the design and functionality of the robot. An iterative development process, informed by real-world use cases, would ensure that the system meets the practical needs of its users.

Autonomous Navigation and Mobility

Autonomous navigation will enable the robot to operate independently across agricultural environments. Implementing Simultaneous Localization and Mapping (SLAM) algorithms would allow the robot to map its surroundings and track its position accurately. Techniques such as LiDAR-based SLAM, visual SLAM, or combining multiple sensor modalities could provide robust localization, especially in environments where GPS signals are weak or unavailable.

Path planning algorithms that take into account the layout of the farm, including rows of trees and obstacles like irrigation systems, would allow the robot to navigate the environment.

Robust mobility platforms that can adapt to different terrains and conditions could improve the robot's range of operation. Designing platforms with suspension systems for wheeled robots or exploring legged robots for navigating uneven terrain could increase the robot's operational capabilities.

Collaboration with Agronomists and Farmers

Collaborating with agronomists, horticulturists, and agricultural engineers will be essential to ensure that the system meets the practical requirements of the agricultural industry. Insights into crop characteristics and farming practices could inform the system design and functionality.

6.5 Final Remarks

We believe that this thesis has made important contributions to the advancement of dual-arm robotic systems for precision fruit harvesting in unstructured agricultural environments. By addressing critical challenges in perception, manipulation, and motion planning, the research has demonstrated the feasibility and potential impact of integrating advanced robotics into agriculture.

The development of a dual-arm robotic platform capable of coordinated manipulation tasks, perception under occlusions, and coordinated motion planning lays a strong foundation for future innovations. The experimental validations across different platforms and conditions underscore the system's robustness and adaptability.

Yet, the journey towards fully autonomous and practical robotic harvesting systems is ongoing. The proposed future work highlights the diverse aspects of this effort, which include technical improvements, interdisciplinary collaboration, and real-world implementation.

Since global agricultural challenges persist, including labor shortages and the need for sustainable practices, the integration of robotics offers a promising solution. Continuous research, development, and collaboration among engineers, scientists, and agricultural stakeholders are essential to realize the full potential of robotic systems in transforming agriculture.

The advancements achieved in this thesis contribute to the field of robotics, but also to the broader goals of food security, resource optimization, and sustainable development. It is hoped that this work inspires further exploration and innovation, driving progress towards a future where robotics and automation to have a hand in meeting the world's needs.

APPENDIX A

Software Details

Introduction

This appendix provides a technical overview of the software execution steps, modules and parameters used on this work. Parameters do not change over different subsections unless specified otherwise.

RKCL Robot Control

Execution Steps

1. **State Acquisition**: The system gathers current joint states and sensor data, updating the robot's internal representation.

2. **Task Evaluation**: Based on task objectives and priorities, the system evaluates which tasks are active, with higher-priority tasks given precedence.

3. **Control Computation**: Joint commands are calculated using RKCL's kinematic control loop to achieve target positions and minimize error.

4. **Trajectory Generation**: Using Reflexxes, the system generates jerk-limited trajectories to ensure smooth motion between calculated points.

5. **Collision Checking**: sch-core checks for potential collisions, and, if detected, the system modifies trajectories in real-time to avoid obstacles.

6. **Command Execution**: Commands are sent to the robot's actuators, completing the control cycle and readying the system for the next update.

Modules

- **RBDyn (Rigid Body Dynamics Library)**: Provides kinematic and dynamic modeling tools, including calculations for forward kinematics.
- **RKCL (Robot Kinematic Control Library)**: Implements the kinematic control loop.
- **Reflexxes Motion Libraries**: Responsible for generating online trajectories based on set constraints. .
- **sch-core**: Manages collision detection using the Gilbert-Johnson-Keerthi (GJK) algorithm. It calculates minimum distances between convex shapes and supports real-time trajectory modifications to prevent collisions.
- **ROS (Robot Operating System)**: Serves as the communication framework with the sensors.

RKCL Structure

The BAZAR robot's kinematic structure is defined in a YAML description file, which includes:

- **Kinematic Properties**: Describes link lengths, joint limits, and the overall robot topology.
- **Dynamic Properties**: Specifies the mass and inertia tensors for each link, critical for accurate dynamic calculations.
- **Joint Groups**: Organizes joints into functional units with set priorities to support coordinated, multi-joint control. RBDyn provides forward kinematics computations, and RKCL organizes joints with priorities, ensuring that primary tasks are addressed by higher-priority groups, while lower-priority groups assist as needed.

RKCL structures data to manage the robot state and control objectives as:

- **Robot Object**: The central data structure containing the robot's kinematic model and joint groups.
- **Control Points**: Each control point is associated with a task variable (e.g., end-effector position) and holds the current state, target, and control commands.
- **Observation Points**: Used to monitor frames or objects within the environment without directly controlling them.

Each control point follows a specific process:

- **State Update**: Obtains the current state via feedback.
- **Target Definition**: Sets the desired state based on task objectives.
- **Command Computation**: Calculates control commands to minimize the difference between the current state and the target.

Parameters

Trajectory Planning Parameters:

- **Maximum Velocity** ($v_{\max}$): Limits the joint speed to prevent overshooting or excessive wear.
  $v_{\max} = 0.17\,\mathrm{m/s}$
- **Maximum Acceleration** ($a_{\max}$): Controls how quickly joints can change velocity.
  $a_{\max} = 1.5\,\mathrm{m/s^2}$
- **Maximum Jerk** ($j_{\max}$): Sets smoothness by limiting acceleration change rate.
  $j_{\max} \approx 100\,\mathrm{m/s^3}$
- **Control Cycle Time** (Δt): Interval between trajectory updates.
  $\Delta t = 0.001\,\mathrm{s}$

Collision Detection Parameters:

- **Proximity Threshold** (`proximity_threshold`): Sets the minimum distance for proximity queries to trigger collision detection.
  `proximity_threshold` = 0.02 m

Kinematic Control Parameters:

- **Joint Limits**: Defines allowable joint range for safe operation.

Table A.1: KUKA LWR Joint Limits

Joint	Minimum (Degrees)	Maximum (Degrees)
Joint 1	-170°	+170°
Joint 2	-120°	+120°
Joint 3	-170°	+170°
Joint 4	-120°	+120°
Joint 5	-170°	+170°
Joint 6	-120°	+120°
Joint 7	-175°	+175°

MoveIt Robot Control

Execution Steps

1. **State Acquisition**: The system gathers real-time joint states and sensor data from the robot hardware through ROS topics. This data forms the basis for current position tracking.

2. **Motion Planning**: MoveIt's planning pipeline generates a collision-free trajectory based on the end-effector's target pose or joint configuration. It integrates joint constraints and includes collision information.

3. **Kinematic Solving**: The TRAC-IK solver computes the necessary joint positions to achieve the target end-effector pose. It optimizes for joint limits and manages system redundancy.

4. **Trajectory Generation**: The generated trajectory is time-parameterized to achieve smooth, jerk-limited movements. It adheres to joint velocity and acceleration constraints.

5. **Collision Checking**: MoveIt! uses the Flexible Collision Library (FCL) to continuously check for collisions along the trajectory and adjust the path if necessary to avoid potential obstacles.

6. **Command Execution**: The planned trajectory is sent to the robot's controllers via ROS action servers or topics to execute joint movements.

7. **Feedback Monitoring**: During movement, joint states are continually monitored to verify conformity with the planned path, with commands dynamically adjusted if deviations are detected.

Modules

- **Robot Operating System (ROS)**: Provides essential communication between various modules, acting as a middleware to manage data flow.

- **MoveIt**: Handles motion planning, kinematics, and collision detection, using the Unified Robot Description Format (URDF) and the Semantic Robot Description Format (SRDF) to define the robot's physical and planning structures. URDF describes joint configurations, link dimensions, mass properties, and limits, while SRDF contains information on joint groups, end-effectors, and default collision parameters.

- **TRAC-IK (Topologically Robust Analytical and Continuous Inverse Kinematics)**: Serves as an inverse kinematics solver, enhancing the standard KDL solver

by incorporating joint limits and optimizing for redundancy. This allows for more efficient path planning in constrained spaces.

- **Flexible Collision Library (FCL)**: Detects potential collisions and provides continuous collision monitoring along the trajectory. FCL evaluates both static and dynamic models to confirm that paths remain clear of obstacles and allows for real-time collision avoidance.

BAZAR Additions

- **Custom Controllers**: Controllers are developed to manage individual joint actuations to achieve synchronized movements of dual-arm setup.
- **System Initialization**: URDF and SRDF files provide MoveIt with kinematic and semantic details, including link lengths, joint limits, and priority-based grouping of joints for dual-arm tasks.

Baxter Additions

- **Codebase Refactoring**: Baxter's original control stack was refactored for Python 3 compatibility, as its native setup was developed for Python 2 under ROS Indigo. Adjustments to dependencies, CMakeLists, and package configurations were necessary to bridge communication with Baxter's hardware in ROS Noetic.
- **Control Mode**: Baxter was operated in velocity control mode to allow smoother trajectory execution.

Parameters

Key parameters govern the performance and safety of the control system:

TRAC-IK:

- **Search Resolution**: The resolution for inverse kinematics search. `kinematics_solver_search_resolution` = 0.001
- **Timeout**: Maximum time allowed for IK computation. `kinematics_solver_timeout` = 0.05s,

Collision Detection (Flexible Collision Library - FCL):

- **Proximity Threshold**: Detects model overlap, which triggers trajectory modifications when potential collisions are identified. `collision_distance_field` = 0.01m,

RealSense:

- **Depth Range**: Filters for objects within this depth range.
  `DEPTH_MIN` = 0.15 m
  `DEPTH_MAX` = 3.0 m,
- **Frame Rate**: Set to video stream frames per second (FPS).
  `frame_rate` = 30 m,
- **Depth Range**: Filters for objects within this depth range.
  `DEPTH_MIN` = 0.15 m
  `DEPTH_MAX` = 3.0 m,
- **Frame Rate**: Set to video stream frames per second (FPS).
  `frame_rate` = 30 m,
- **Hole Filling Mode**: Fills missing depth values from the left neighboring pixels to smooth the depth map.
  `hole_filling_mode` = left,
- **Depth Visualization Range**: Displays depth values between 0.5 m and 1.5 m for clearer visualization.
  `visualization_min_depth` = 0.5 m,
  `visualization_max_depth` = 1.5 m,

Fruit Detection and Tracking

Execution Steps

1. **Environment Setup**: The system configures RealSense depth cameras and defines parameters.
2. **Image Preprocessing**: Input images are resized to the expected YOLOv5 input size, with pixel values normalized to the range [0, 1], and converted to PyTorch tensors.
3. **Model Inference**: The YOLOv5 model processes preprocessed images to detect fruits.
4. **Tracking Implementation**: Using DeepSORT for initial tracking, the system later transitions to a custom tracking algorithm, optimized for spatial association between frames.
5. **3D Localization**: RealSense provides depth data for 3D positioning for the fruit localization.

Modules

- **YOLOv5**: Provides real-time detection of fruits within each frame, generates bounding boxes, class labels, and confidence scores.

- **Intel RealSense SDK**: Manages communication with the camera to obtain RGB and depth data. Depth data is processed with noise filters and hole filling through spatial and temporal filters so that it aligns with RGB frames for pixel-wise correspondence.
- **DeepSORT**: Tracks detected fruits across frames by associating YOLO detections with track positions.
- **Custom Tracking Algorithm**: Developed to associate detected fruits in consecutive frames using spatial proximity and Euclidean distance between bounding box centroids.
- **OpenCV**:

Parameters

Detection (YOLOv5):

- **Confidence Threshold**: Filters out low-confidence detections.
  `confidence_threshold` = 0.45,
- **IoU Threshold**: Used for Non-Maximum Suppression.
  `iou_threshold` = 0.45,
- **Max Detections per Image**: Limits the number of detections to avoid overhead.
  `max_detections` = 1000,

Branch Detection

Execution Steps

1. **Data Loading**: Import three-dimensional point cloud data from PLY files which includes spatial coordinates and RGB color data.
2. **Preprocessing**: Convert RGB data to HSV color space and apply HSV-based filtering to isolate branches based on specific color characteristics.
3. **Transformation to Detect Visible Branches**: Align the point cloud with the robot coordinate frame through rotation around the X, Y, and Z axes.
4. **Edge Detection**: Project the transformed point cloud to a 2D plane and apply edge detection to highlight potential branch edges.
5. **Line Extraction**: Use the Probabilistic Hough Transform on detected edges to extract line segments representing branches.

6. **Line Filtering**: Remove redundant lines based on slope and distance thresholds, refining the detected branch lines.
7. **Plane Estimation**: Calculate plane equations for each potential branch.
8. **Transformation to Detect Occluded Branches**: Align the point cloud with the plane through rotation around the X, Y, and Z axes.
9. **Occluded Branch Detection**: Repeat steps *4-5-6*.
10. **Visualization and Validation**: Overlay detected branches on the original point cloud for visual verification.

Modules

- **Similarity Filtering**: Filters redundant lines.
- **Branch Plane Estimation**: Calculates the plane equations.
- **Intel RealSense SDK**: Manages data acquisition from the Intel RealSense D435 camera, capturing synchronized RGB and depth information to generate 3D point clouds.
- **OpenCV**: Used for image processing tasks such as the conversion of RGB data to HSV, Canny edge detection `Canny()`, projection of 3D data onto a 2D plane and 2D Hough line detection `HoughLinesP()`.
- **SciPy**: Used for the spatial transformation.

Parameters

The branch detection system uses several key parameters to fine-tune each processing step, ensuring reliable and precise branch detection:

Edge Detection and Line Detection (OpenCV):

- **Canny Thresholds**: Control the sensitivity of edge detection on the 2D projection of branches.
  $\texttt{threshold}_1 = 100$,
  $\texttt{threshold}_2 = 200$,
- **Hough Threshold**: Minimum number of intersections in Hough space required to detect a line.
  $\texttt{HOUGH_THRESHOLD} = 50$,
- **Hough Min Line Length**: Minimum length for a line to be considered valid.
  $\texttt{HOUGH_MIN_LINE_LENGTH} = 50\,\text{px}$,

- **Hough Max Line Gap**: Maximum gap allowed between line segments to connect them into a single line.
  `HOUGH_MAX_LINE_GAP` = 10 px,

APPENDIX B

3D Models

Introduction

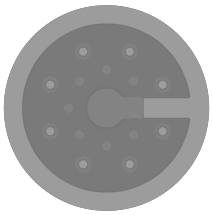
In this appendix we present the 3D Models designed and used during this work.

Renderings

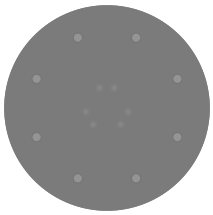


(a) Scanned mesh *(b) Collision mesh*

Figure B.1: Electric pruner 3D scan and its simplified mesh used in collision detection



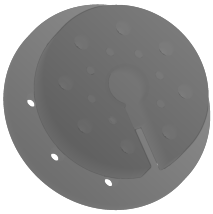
(a) Top view



(b) Bottom view

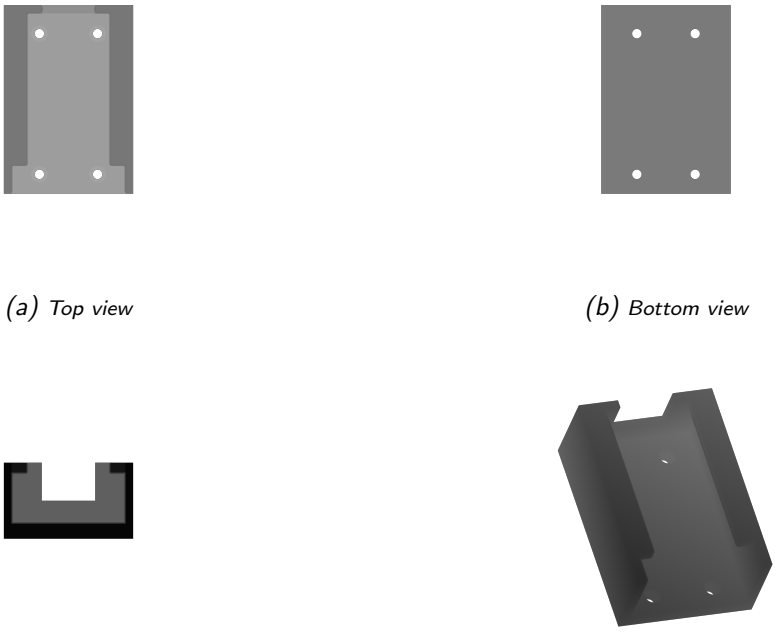


(c) Side view



(d) Isometric view

Figure B.2: BAZAR end-effector support 3D model



(a) Top view *(b) Bottom view*

(c) Side view *(d) Isometric view*

Figure B.3: Electric pruner holder end-effector support 3D model

APPENDIX C

Résumé des travaux

Dans cette thèse intitulée "Robots à deux bras pour la récolte précise de fruits dans des environnements agricoles non structurés", nous avons exploré le développement et la mise en œuvre de systèmes robotiques à deux bras visant à automatiser la récolte de fruits dans des contextes agricoles complexes. Face à la croissance démographique mondiale et à la diminution progressive de la main-d'œuvre agricole, il est impératif de développer des solutions innovantes pour accroître la productivité et l'efficacité du secteur agricole. La récolte manuelle des fruits est non seulement laborieuse et coûteuse, mais elle souffre également d'incohérences en termes de qualité et d'efficacité. L'intégration de la robotique et de l'automatisation offre des perspectives prometteuses pour relever ces défis et transformer les pratiques agricoles actuelles.

Motivation et Problématique

La motivation principale de cette recherche réside dans la nécessité d'améliorer l'efficacité, la flexibilité et la fiabilité des robots récolteurs de fruits. Les robots à un seul bras existants présentent des limitations significatives en termes de dextérité et d'adaptabilité, particulièrement lorsqu'il s'agit d'effectuer des tâches complexes telles que la coupe et la collecte simultanées des fruits, ou de gérer les occlusions causées par le feuillage dense des arbres fruitiers. Les environnements agricoles non structurés sont caractérisés par une grande variabilité dans l'apparence des fruits, des occlusions fréquentes dues aux branches et aux feuilles, ainsi que des conditions d'éclairage changeantes, ce qui complique considérablement les tâches de perception et de manipulation pour les robots.

Les principaux défis identifiés sont les suivants :

- **Perception robuste sous occlusions :** Les fruits peuvent être partiellement ou to-

talement cachés par le feuillage, rendant leur détection et leur localisation précises difficiles avec des techniques de vision traditionnelles.

- **Manipulation complexe :** La nécessité de manipuler délicatement les fruits sans les endommager, tout en évitant les collisions avec l'environnement, requiert une coordination avancée et une dextérité élevée.

- **Planification et contrôle en temps réel :** L'adaptation des mouvements du robot en temps réel pour gérer les obstacles imprévus et les variations de l'environnement est essentielle pour une opération efficace.

- **Validation dans des conditions réelles :** La démonstration de l'applicabilité pratique du système nécessite des tests approfondis dans des environnements variés et avec différents types de fruits.

Objectifs

Pour répondre à ces défis, la thèse vise à développer un système robotique à deux bras capable de :

- **Manipulation coordonnée à deux bras :** Reproduire la dextérité humaine pour effectuer des tâches telles que la coupe et la collecte simultanées des fruits, en synchronisant les mouvements des deux bras pour une efficacité optimale.

- **Amélioration de la perception en présence d'occlusions :** Développer des méthodes avancées de vision par ordinateur pour détecter et localiser avec précision les fruits, même lorsqu'ils sont partiellement ou totalement cachés par le feuillage ou les branches.

- **Intégration de stratégies avancées de planification et de contrôle des mouvements :** Concevoir des algorithmes permettant au robot d'adapter ses mouvements en temps réel, en tenant compte de l'évitement des obstacles, de la gestion des occlusions et de la priorisation des tâches.

- **Validation expérimentale en environnements réels :** Tester le système avec différentes plateformes robotiques et types de fruits sous diverses conditions environnementales pour démontrer sa robustesse et son applicabilité pratique.

Méthodologie

La recherche a été structurée en trois phases principales, chacune apportant des améliorations techniques spécifiques :

1. Développement initial du système de récolte à deux bras

Nous avons introduit une approche novatrice en utilisant un robot à deux bras équipé d'une caméra RGB-D et d'outils spécialisés pour la coupe et la collecte. Une stratégie de contrôle basée sur la **Programmation Quadratique Hiérarchique (HQP)** a été mise en place pour permettre au robot de naviguer dans des environnements complexes tout en respectant des contraintes strictes, telles que les limites articulaires, l'évitement des collisions internes (auto-collisions) et externes (avec les fruits et l'arbre). Le HQP permet de hiérarchiser les tâches du robot, en donnant la priorité aux actions critiques tout en respectant les contraintes de sécurité.

2. Amélioration de l'efficacité et de l'adaptabilité

Pour optimiser les performances du système, nous avons mis en œuvre un **algorithme d'ordonnancement basé sur le problème du voyageur de commerce (TSP)**. Cet algorithme détermine la séquence optimale de récolte des fruits en minimisant la distance parcourue et le temps total de récolte. De plus, une planification de mouvement globale a été intégrée en utilisant des méthodes de planification basées sur des échantillons, telles que **RRT*** et **PRM**, pour générer des trajectoires sans collision dans des environnements plus complexes et dynamiques. Des expériences ont été menées avec deux plateformes robotiques distinctes—**BAZAR** et **Baxter**—pour récolter différents types de fruits (pommes et oranges), démontrant la polyvalence et la robustesse du système.

3. Gestion des occlusions et amélioration de la perception

La dernière phase s'est concentrée sur la résolution du problème des occlusions. Nous avons développé des méthodes de perception avancées en combinant :

- **Apprentissage profond pour l'estimation de l'apparence des fruits :** Utilisation de **réseaux de neurones génératifs** (tels que les Auto-Encodeurs Variations, VAE) pour reconstruire l'apparence complète des fruits à partir de données partiellement visibles.

- **Extension 3D de la transformée de Hough pour la détection des branches :** Adaptation de la transformée de Hough pour détecter les branches en trois dimensions, permettant ainsi une modélisation précise de l'environnement et l'identification des obstacles potentiels.

- **Stratégie de manipulation pour dégager les occlusions :** Introduction d'une stratégie de poussée où le robot utilise l'un de ses bras pour déplacer physiquement les branches obstructives, améliorant ainsi la visibilité et l'accessibilité des fruits pour le bras de récolte.

Contributions

La thèse apporte plusieurs contributions significatives au domaine de la robotique agricole :

- **Développement d'un système robotique à deux bras innovant :** Intégration réussie de la perception avancée, de la planification globale des mouvements et du contrôle hiérarchique pour permettre des actions coordonnées à deux bras dans des environnements non structurés.
- **Avancées en perception sous occlusions :** Élaboration de méthodes innovantes combinant l'apprentissage profond et la vision par ordinateur pour la détection et la localisation précises des fruits en présence d'occlusions importantes, améliorant la fiabilité du système dans des conditions réelles.
- **Planification dynamique des mouvements et contrôle adaptatif :** Création d'algorithmes permettant une adaptation en temps réel des mouvements du robot, intégrant l'évitement proactif des obstacles, la gestion des contraintes dynamiques et la priorisation des tâches essentielles.
- **Validation expérimentale sur différentes plateformes et types de fruits :** Réalisation de tests approfondis démontrant la robustesse, la flexibilité et l'efficacité du système sur les robots BAZAR et Baxter, et sur des fruits variés tels que les pommes et les oranges.
- **Contribution à la durabilité agricole :** En proposant une solution automatisée pour la récolte des fruits, la recherche contribue à réduire la dépendance à la main-d'œuvre, à augmenter l'efficacité opérationnelle et à promouvoir une utilisation plus durable des ressources agricoles.

Résultats et Analyse

Les résultats expérimentaux confirment l'efficacité et la fiabilité du système proposé :

Précision de la perception

Les techniques de perception développées ont permis une **détection et un suivi précis des fruits** même dans des conditions d'occlusion sévère et sous des conditions d'éclairage variables. Les réseaux neuronaux génératifs ont reconstruit avec succès l'apparence complète des fruits, améliorant la localisation en trois dimensions. L'utilisation de capteurs RGB-D a fourni des informations de profondeur cruciales pour une modélisation précise de l'environnement.

Efficacité du contrôle des mouvements

Les stratégies de planification de mouvement, combinant planification globale et contrôle hiérarchique, ont assuré une **opération sans collision** et un positionnement précis des effecteurs terminaux. Les robots ont pu naviguer efficacement autour des obstacles dynamiques, tels que les branches mobiles, en adaptant leurs trajectoires en temps réel.

Performance opérationnelle

Le système à deux bras a démontré une **augmentation significative de l'efficacité opérationnelle**, en réduisant le temps de récolte grâce à des actions coordonnées et simultanées. Les tests ont montré que le robot pouvait gérer efficacement différentes variétés de fruits et s'adapter à des configurations environnementales variées.

Gestion proactive des occlusions

La stratégie de manipulation des occlusions a permis au robot de **dégager physiquement les obstacles**, améliorant la visibilité des fruits et augmentant le taux de réussite de la récolte. Cette approche réduit la dépendance à l'estimation sous incertitude et améliore la fiabilité du système dans des environnements complexes.

Discussion et Positionnement par Rapport à l'État de l'Art

Comparé aux travaux existants, notre système se distingue par :

- **L'intégration de la perception et de la manipulation pour gérer les occlusions**, alors que de nombreux travaux se concentrent uniquement sur la détection ou la manipulation, sans les combiner de manière cohérente.
- **L'utilisation de techniques avancées d'apprentissage profond** pour estimer l'apparence des fruits sous occlusions, surpassant les méthodes traditionnelles basées sur des modèles géométriques simples.
- **La planification de mouvement dynamique en temps réel**, intégrant des algorithmes de planification globale et des contrôles adaptatifs pour gérer les contraintes complexes des environnements non structurés.
- **La validation expérimentale approfondie sur plusieurs plateformes robotiques et types de fruits**, démontrant la polyvalence et l'applicabilité pratique du système dans des scénarios réels.

Perspectives et Travaux Futurs

Bien que les résultats soient prometteurs, plusieurs axes d'amélioration et de recherche future sont identifiés :

Améliorations matérielles

Le développement de **nouvelles plateformes robotiques** avec une meilleure vitesse, une précision accrue et une capacité de charge utile plus élevée pourrait améliorer l'efficacité opérationnelle. La conception d'**effecteurs terminaux sur mesure**, adaptés aux caractéristiques spécifiques des différents fruits, pourrait également augmenter les taux de réussite de la récolte.

Tests sur le terrain en environnements réels

La conduite d'expériences dans de véritables exploitations agricoles permettrait d'évaluer les performances du système dans des conditions encore plus variées, incluant des facteurs tels que les conditions météorologiques, le terrain irrégulier et la présence d'autres éléments perturbateurs.

Perception avancée et apprentissage adaptatif

L'intégration de **capteurs supplémentaires** (par exemple, capteurs hyperspectraux, tactiles) et l'utilisation de **techniques d'apprentissage automatique en ligne** pourraient améliorer la robustesse du système face à des variations inattendues et permettre une adaptation continue aux nouvelles situations.

Optimisation des algorithmes pour le temps réel

L'optimisation des algorithmes de perception et de planification pour réduire les charges computationnelles est essentielle pour une opération en temps réel sur des plateformes aux ressources limitées. L'utilisation de **matériel spécialisé** (comme les GPU embarqués ou les FPGA) pourrait accélérer les calculs.

Navigation autonome et mobilité

Le développement de capacités de **navigation autonome** permettrait au robot de se déplacer entre les arbres ou les rangées sans intervention humaine, augmentant ainsi l'autonomie opérationnelle. Cela nécessiterait l'intégration de systèmes de localisation et de cartographie simultanées (SLAM) robustes.

Conclusion

Cette thèse a démontré que les robots à deux bras peuvent révolutionner la récolte des fruits en offrant une solution automatisée capable de fonctionner efficacement dans des environnements agricoles non structurés. En combinant des avancées en perception, planification et contrôle, le système proposé surmonte les limitations des approches précédentes et ouvre la voie à une adoption plus large de la robotique dans le secteur agricole.

Les contributions apportées ont le potentiel de transformer les pratiques agricoles, en augmentant la productivité, en réduisant les coûts liés à la main-d'œuvre et en promouvant une agriculture plus durable. En poursuivant les travaux dans les directions identifiées, il est possible d'atteindre un niveau d'autonomie et d'efficacité encore supérieur, rapprochant ainsi l'objectif d'une automatisation complète de la récolte des fruits.

Bibliography

[BA15] Patrick Beeson and Barrett Ames. Trac-ik: An open-source library for improved solving of generic inverse kinematics. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 928–935. IEEE, 2015. 84, 90

[BGO+16] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016. 50

[BHC+20] Lingxin Bu, Guangrui Hu, Chengkun Chen, Adilet Sugirbay, and Jun Chen. Experimental and simulation analysis of optimum picking patterns for robotic apple harvesting. *Scientia Horticulturae*, 261:108937, 2020. 37

[BU17a] S. Bargoti and J. P. Underwood. Deep fruit detection in orchards. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3626–3633, 2017. 34, 35, 45

[BU17b] Suchet Bargoti and James Underwood. Deep fruit detection in orchards, 2017. 41

[BV16] A. Bechar and C. Vigneault. Agricultural robots for field operations: Concepts and components. *Biosystems Engineering*, 149:94–111, 2016. 33

[BVD08] John Billingsley, Arto Visala, and Mark Dunn. *Robotics in Agriculture and Forestry*, pages 1065–1077. 01 2008. 37, 43

[BvHHE14] C. Wouter Bac, Eldert J. van Henten, Jochen Hemming, and Yael Edan. Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, 31(6):888–911, 2014. 33

[CCS96] P. Chiacchio, S. Chiaverini, and B. Siciliano. Direct and inverse kinematics for coordinated motion tasks of a two-manipulator system. *Journal of Dynamic Systems, Measurement, and Control*, 118(4):691–697, 12 1996. 54, 74, 83

[CCT+15] Xiangyu Chen, Krishneel Chaudhary, Yoshimaru Tanaka, Kotaro Nagahama, Hiroaki Yaguchi, Kei Okada, and Masayuki Inaba. Reasoning-based vision recognition for agricultural humanoid robot toward tomato harvesting. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 6487–6494. IEEE, 2015. 123

[CD14] Supawadee Chaivivatrakul and Matthew Dailey. Texture-based fruit detection. volume 15, 12 2014. 40

[CPN+19] Andrea Cherubini, Robin Passama, Benjamin Navarro, Mohamed Sorour, Abdellah Khelloufi, Osama Mazhar, Sonny Tarbouriech, Jihong Zhu, Olivier Tempier, André Crosnier, et al. A collaborative robot for the factory of the future: Bazar. *The International Journal of Advanced Manufacturing Technology*, 105(9):3643–3659, 2019. 13, 24, 26, 65, 88, 89, 116

[DAJJ+11] Zhao De-An, Lv Jidong, Wei Ji, Zhang Ying, and Chen Yu. Design and control of an apple harvesting robot. *Biosystems Engineering - BIOSYST ENG*, 110:112–122, 10 2011. 43

[DCQ+15] Jnaneshwar Das, Gareth Cross, Chao Qu, Anurag Makineni, Pratap Tokekar, Yash Mulgaonkar, and Vijay Kumar. Devices, systems, and methods for automated monitoring enabling precision agriculture. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 462–469. IEEE, 2015. 35

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 50

[DH72] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972. 110

[DHMK17] Joseph R Davidson, Cameron J Hohimer, Changki Mo, and Manoj Karkee. Dual robot coordination for apple harvesting. In *2017 ASABE annual international meeting*, page 1. American Society of Agricultural and Biological Engineers, 2017. 36, 37, 44, 45, 46, 74, 98

[FFM+18] Longsheng Fu, Yali Feng, Yaqoob Majeed, Xin Zhang, Jing Zhang, Manoj Karkee, and Qin Zhang. Kiwifruit detection in field images using faster r-cnn with zfnet. *IFAC-PapersOnLine*, 51(17):45–50, 2018. 123

[FT87] Bernard Faverjon and Pierre Tournassoud. A local based approach for path planning of manipulators with a high number of degrees of freedom. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 1152–1159. IEEE, 1987. 59

[FWWL15] Qingchun Feng, Xiaonan Wang, Guohua Wang, and Zhen Li. Design and test of tomatoes harvesting robot. In *2015 IEEE international conference on information and automation*, pages 949–952. IEEE, 2015. 123

[GAT19] Sebastian Gonzalez, Claudia Arellano, and Juan E Tapia. Deepblueberry: Quantification of blueberries in the wild using instance segmentation, 2019. 33

[GFZ+20] Fangfang Gao, Longsheng Fu, Xin Zhang, Yaqoob Majeed, Rui Li, Manoj Karkee, and Qin Zhang. Multi-class fruit-on-plant detection for apple in snap system using faster r-cnn. *Computers and Electronics in Agriculture*, 176:105634, 2020. 123

[GKC24] Ege Gursoy, Dana Kulić, and Andrea Cherubini. Occlusion handling by pushing for enhanced fruit detection. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 76–81, 2024. 25, 108

[GMGA+20] Jordi Gené-Mola, Eduard Gregorio, Fernando Auat Cheein, Javier Guevara, Jordi Llorens, Ricardo Sanz-Cortiella, Alexandre Escolà, and Joan R. Rosell-Polo. Fruit detection, yield prediction and canopy geometric characterization using lidar with forced air flow. *Computers and Electronics in Agriculture*, 168:105121, 2020. 41, 46

[GMVRP+19] Jordi Gené-Mola, Verónica Vilaplana, Joan R Rosell-Polo, Josep-Ramon Morros, Javier Ruiz-Hidalgo, and Eduard Gregorio. Multi-modal deep learning for fuji apple detection using rgb-d cameras and their radiometric capabilities. *Computers and Electronics in Agriculture*, 162:689–698, 2019. 40, 41

[GNC+23] Ege Gursoy, Benjamin Navarro, Akansel Cosgun, Dana KuliÄ‡, and Andrea Cherubini. Towards vision-based dual arm robotic fruit harvesting. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–6, 2023. 24, 79

[GTC23] Ege Gursoy, Sonny Tarbouriech, and Andrea Cherubini. Can robots mold soft plastic materials by shaping depth images? *IEEE Transactions on Robotics*, 39(5):3620–3635, 2023. 105

[GVBM19] P. Ganesh, K. Volle, T.F. Burks, and S.S. Mehta. Deep orange: Mask r-cnn based orange detection and segmentation, 2019. 6th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture AGRICONTROL 2019. 34

[GWWL22] Liang Gong, Wenjie Wang, Tao Wang, and Chengliang Liu. Robotic harvesting of the occluded fruits with a precise shape and position reconstruction approach. *Journal of Field Robotics*, 39(1):69–84, 2022. 39, 123

[HGIT02] Shigehiko Hayashi, Katsunobu Ganno, Yukitsugu Ishii, and Itsuo Tanaka. Robotic harvesting system for eggplants. *Japan Agricultural Research Quarterly: JARQ*, 36:163–168, 07 2002. 43, 45, 74

[HSY+10] Shigehiko Hayashi, Kenta Shigematsu, Satoshi Yamamoto, Ken Kobayashi, Yasushi Kohno, Junzo Kamata, and Mitsutaka Kurita. Evaluation of a strawberry-harvesting robot in a field test. *Biosystems engineering*, 105(2):160–171, 2010. 43, 45, 74, 98

[JDAWS16] Lv Jidong, Zhao De-An, Ji Wei, and Ding Shihong. Recognition of apple fruit in natural environment. *Optik*, 127(3):1354–1362, 2016. 40

[Joc20] Glenn Jocher. Ultralytics yolov5, 2020. 49

[KC20] Hanwen Kang and Chao Chen. Fast implementation of real-time fruit detection in apple orchards using deep learning. *Computers and Electronics in Agriculture*, 168:105108, 2020. 34, 39, 46, 74, 98, 123

[KCM20] Raymond Kirk, Grzegorz Cielniak, and Michael Mangan. L* a* b* fruits: A rapid and robust outdoor fruit detection system combining bio-inspired features with one-stage deep learning networks, 2020. 34

[KE20] Polina Kurtser and Yael Edan. Planning the sequence of tasks for harvesting robots. *Robotics and Autonomous Systems*, 131:103591, 2020. 37, 38

[KLV14] Ferhat Kurtulmus, Won Suk Lee, and Ali Vardar. Immature peach detection in colour images acquired in natural illumination conditions using statistical classifiers and neural network. *Precision agriculture*, 15:57–79, 2014. 39, 123

[KRR+20] Polina Kurtser, Ola Ringdahl, Nati Rotstein, Ron Berenstein, and Yael Edan. In-field grape cluster size assessment for vine yield estimation using a mobile robot and a consumer level rgb-d camera. *IEEE Robotics and Automation Letters*, 5(2):2031–2038, 2020. 39, 40, 46

[KWWM19] Anand Koirala, KB Walsh, Zhenglin Wang, and C McCarthy. Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of â€˜mangoyoloâ€™. *Precision Agriculture*, 20(6):1107–1135, 2019. 35, 74, 98, 123

[KYI+10] Naoshi Kondo, Koki Yata, Michihisa Iida, Tomoo Shiigi, Mitsuji Monta, Mitsutaka Kurita, and Hiromi Omori. Development of an end-effector for a tomato cluster harvesting robot. *Engineering in Agriculture, Environment and Food*, 3(1):20–24, 2010. 43, 45, 74

[LCA+18] Xu Liu, Steven W. Chen, Shreyas Aditya, Nivedha Sivakumar, Sandeep Dcunha, Chao Qu, Camillo J. Taylor, Jnaneshwar Das, and Vijay Kumar. Robust fruit counting: Combining deep learning, tracking, and structure from motion. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1045–1052, 2018. 35, 45

[LEM+17] Christopher Lehnert, Andrew English, Christopher McCool, Adam W. Tow, and Tristan Perez. Autonomous sweet pepper harvesting for protected cropping systems. *IEEE Robotics and Automation Letters*, 2(2):872–879, 2017. 43, 44, 45, 74

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 50

[LSL+22] Denghui Li, Xiaoxuan Sun, Shengping Lv, Hamza Elkhouchlaa, Yuhang Jia, Zhongwei Yao, Peiyi Lin, Haobo Zhou, Zhengqi Zhou, Jiaxing Shen, et al. A novel approach for the 3d localization of branch picking points based on deep learning applied to longan harvesting uavs. *Computers and Electronics in Agriculture*, 199:107191, 2022. 123

[LSM+16] C. Lehnert, I. Sa, C. McCool, B. Upcroft, and T. Perez. Sweet pepper pose detection and grasping for automated crop harvesting. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2428–2434, 2016. 37

[LTL+18] Lufeng Luo, Yunchao Tang, Qinghua Lu, Xiong Chen, Po Zhang, and Xiangjun Zou. A vision methodology for harvesting robot to detect cutting points on peduncles of double overlapping grape clusters in a vineyard. *Computers in industry*, 99:130–139, 2018. 41

[LXQQ23] Tao Li, Feng Xie, Quan Qiu, and Feng Qingchun. Multi-arm robot task planning for fruit harvesting using multi-agent reinforcement learning. 08 2023. 37, 38

[LYL+18] Yan-Ping Liu, Chang-Hui Yang, Huang Ling, Shingo Mabu, and Takashi Kuremoto. A visual system of citrus picking robot using convolutional neural networks. pages 344–349, 2018. 33

[LZG+19] Xiao Ling, Yuanshen Zhao, Liang Gong, Chengliang Liu, and Tao Wang. Dual-arm cooperation and implementing for robotic harvesting tomato using binocular vision. *Robotics Auton. Syst.*, 114:134–143, 2019. 36, 44, 45, 46

[Mah18] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. *Sankhyā: The Indian Journal of Statistics, Series A (2008-)*, 80:S1–S7, 2018. 50

[MB14] SS Mehta and TF Burks. Vision-based control of robotic manipulator for citrus harvesting. *Computers and Electronics in Agriculture*, 102:146–158, 2014. 41, 43, 45, 46, 74

[MPU17] Chris McCool, Tristan Perez, and Ben Upcroft. Mixtures of lightweight deep convolutional neural networks: Applied to agricultural robotics. *IEEE Robotics and Automation Letters*, 2(3):1344–1351, 2017. 34, 45

[MSD+16] Christopher McCool, Inkyu Sa, Feras Dayoub, Christopher Lehnert, Tristan Perez, and Ben Upcroft. Visual detection of occluded crop: For automated harvesting. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2506–2512, 2016. 39, 40, 46

[Ope23] OpenAI. Dall-e 3. https://openai.com/index/dall-e-3/, 2023. [Accessed: 2024-10-01]. 13, 20

[POS+17] Pornntiwa Pawara, Emmanuel Okafor, Olarik Surinta, Lambert Schomaker, and Marco Wiering. Comparing local descriptors and bags of visual words to deep convolutional neural networks for plant recognition. In *International Conference on Pattern Recognition Applications and Methods*, volume 2, pages 479–486. SciTePress, 2017. 50

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 105

[RI16] Pravakar Roy and Volkan Isler. Surveying apple orchards with a monocular vision system. In *2016 IEEE international conference on automation science and engineering (CASE)*, pages 916–921. IEEE, 2016. 13, 35, 45

[SDK+17] Abhisesh Silwal, Joseph Davidson, Manoj Karkee, Changki Mo, Qin Zhang, and Karen Lewis. Design, integration, and field evaluation of a robotic apple harvester. *Journal of Field Robotics*, 34, 03 2017. 37, 43, 44, 45, 74

[SdSdSA20] Thiago T Santos, Leonardo L de Souza, Andreza A dos Santos, and Sandra Avila. Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Computers and Electronics in Agriculture*, 170:105247, 2020. 123

[SFN+20] Delia SepúLveda, Roemi Fernández, Eduardo Navas, Manuel Armada, and Pablo González-De-Santos. Robotic aubergine harvesting using dual-arm manipulation. *IEEE Access*, 8:121889–121904, 2020. 44

[SGD+] Inkyu Sa, Zongyuan Ge, Feras Dayoub, Ben Upcroft, Tristan Perez, and Chris McCool. Deepfruits: A fruit detection system using deep neural networks. *Sensors*, 16(8):1222. 33, 45

[TFAI08] K. Tanigaki, T. Fujiura, A. Akase, and J. Imagawa. Cherry-harvesting robot. *Computers and Electronics in Agriculture*, 63(1):65–72, 2008. 43, 45, 74

[TNF+22] Sonny Tarbouriech, Benjamin Navarro, Philippe Fraisse, André Crosnier, Andrea Cherubini, and Damien Sallé. An admittance based hierarchical control framework for dual-arm cobots. *Mechatronics*, 86:102814, 2022. 57

[TYW+19] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang. Apple detection during different growth stages in orchards using the improved yolo-v3 model. *Computers and Electronics in Agriculture*, 157:417–426, 2019. 39

[UNSA19] Department of Economic United Nations and Population Division Social Affairs. World population prospects 2019: Highlights, 2019. Accessed: 2024-10-01. 21

[vHHvT+02] E. J. van Henten, J. Hemming, B. A. van Tuijl, J. G. Kornet, J. Meuleman, J. Bontsema, and E. A. van Os. An autonomous robot for harvesting cucumbers in greenhouses. *Autonomous Robots*, 13(3):241–258, 2002. 43, 45, 74

[WBP17] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017. 50, 79

[WMZL21] Lin Wu, Jie Ma, Yuehua Zhao, and Hong Liu. Apple detection in complex scene using the improved yolov4 model. *Agronomy*, 11(3), 2021. 34

[WUW18] Alexander Wendel, James Underwood, and Kerry Walsh. Maturity estimation of mangoes using hyperspectral imaging from a ground based mobile platform. *Computers and Electronics in Agriculture*, 155:298–313, 2018. 35

[XGGF19] Ya Xiong, Yuanyue Ge, Lars Grimstad, and Pål From. An autonomous strawberry-harvesting robot: Design, development, integration, and field evaluation. *Journal of Field Robotics*, 37, 08 2019. 41, 42, 43, 44, 45, 46, 74

[ZA23] Sadaf Zeeshan and Tauseef Aized. Performance analysis of path planning algorithms for fruit harvesting robot. *Journal of Biosystems Engineering*, 48(2):178–197, 2023. 37

[ZGHL16] Yuanshen Zhao, Liang Gong, Yixiang Huang, and Chengliang Liu. A review of key techniques of vision-based control for harvesting robot. *Computers and Electronics in Agriculture*, 127:311–323, 2016. 44, 45

[ZYCX19] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Omni-scale feature learning for person re-identification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3702–3712, 2019. 66

[ZYCX21] Kaiyang Zhou, Yongxin Yang, Andrea Cavallaro, and Tao Xiang. Learning generalisable omni-scale representations for person re-identification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 50