AN AUTONOMOUS SYSTEM FOR CROP INSPECTIONS: 3D HYPERSPECTRAL RECONSTRUCTIONS, MULTI-ROBOT PLANNING AND CONTROL

BY MERRILL EDMONDS

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Mechanical and Aerospace Engineering

Written under the direction of

Dr. Jingang Yi

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

Jan, 2022

ABSTRACT OF THE DISSERTATION

An Autonomous System for Crop Inspections: 3D Hyperspectral Reconstructions, Multi-Robot Planning and Control

by Merrill Edmonds

Dissertation Director: Dr. Jingang Yi

Crop inspections are a critical part of modern agriculture. On a fundamental level, timely inspections allow farmers to identify problems with the crop and take preventative measures where necessary. More generally, inspections provide a wealth of information for downstream precision agriculture applications, which typically optimize yields using large-scale crop monitoring data captured via satellites or drones. However, the top-down images generated by satellites and drones are ill-suited for leaf-level analyses, so farmers still perform manual inspections at certain points in the growth cycle to ensure yield targets are met. Agricultural robots bridge this gap by allowing farmers to automate close-up inspections, thus capturing the benefits of both manual inspections and automated toolchains. On the other hand, aggregating the multi-modal data generated by these robots requires expert knowledge and several proprietary software packages, and the inspection routes for large robot groups are still planned by human operators. There are many similar technical challenges that prevent robotic inspections from being included in the agricultural value chain, and new systems and algorithms must be developed to overcome these challenges.

The goal of this dissertation is to develop a new autonomous system and algorithms

for crop inspections. In addition to developing a hardware and software architecture needed for high-throughput inspections, the dissertation also addresses the following technological gaps: (i) three-dimensional (3D) hyperspectral reconstruction methods necessary for efficient and low-cost inspections using robot-mounted sensors and cameras, (ii) site-specific multi-robot allocation and planning algorithms to enable high-throughput inspections, (iii) control algorithms that allow unmanned aerial vehicles (UAVs) to land on unmanned ground vehicles (UGVs) for continuous inspection missions, and (iv) the human-robot interaction (HRI) methods to provide farmers with greater control over the inspection process. When combined, the proposed methods allow the robots to autonomously select which plants to inspect as distributed UGV-UAV teams, while also giving farmers intuitive tools to preempt the inspection process. The contributions of the thesis are therefore all aimed towards creating a single, cohesive autonomous crop inspection framework.

One of the central contributions of this thesis is a novel 3D hyperspectral reconstruction method that captures the spatial and spectral properties of the plant. Unlike similar methods that use bulky, expensive, and slow hyperspectral cameras and laser scanners to capture data, the proposed method leverages the spectral and intrinsic properties of a heterogeneous set of commodity cameras to convert standard RGB images into full-spectrum data embedded onto a 3D point cloud. Since practical considerations require the robot to collect a limited number of source images per plant, a spectrally-optimal next-best-view (SONBV) problem is formulated and a method is proposed to produce the most complete 3D hyperspectral reconstruction from the source images. Novel algorithms to determine these angles from incomplete knowledge are demonstrated, and the applicability to crop inspections is further investigated by fabricating and testing a tabletop 3D hyperspectral scanner.

The proposed reconstruction method naturally leads to the multi-robot task allocation problem of selecting a representative set of scan targets, and distributing the associated scanning tasks among the team of UGVs. An equivalent optimization problem is proposed to select sampling points across a continuous metric field mapped onto

the crop field such that the cumulative posterior covariance of a Gaussian process representation of the metric field is maximized. Augmented robot-centered geodesic Voronoi graphs of the plot are then constructed to find minimum-cost hyperedges on a multi-partite graph induced by these Voronoi regions. The planning problem is proven to be NP-hard, and an approximate set of assignments are determined greedily as the minimal-cost finite-horizon combinations of scan targets. The proposed algorithm is also extended to continuous data collection by UGV/UAV groups. Simulation studies show that the proposed methods outperform state-of-the-art methods in prediction quality and average resolution, while also observing the energy constraints.

Another particularly challenging control problem for the proposed UGV/UAV inspection system is the execution of coordinated UGV/UAV landing maneuvers (e.g., when the UAV needs to land on the UGV to recharge). The UGV/UAV pair is taken in isolation, and the interactions between the UGV and the UAV are modeled as a paired model identification and trajectory optimization problem. The near-surface aerodynamic effects caused by the interaction of the multi-rotor downwash and the platform's geometry are modeled using a novel neural network architecture. Linearized disturbance models are then used to iteratively plan landing trajectories based on the on-board model predictive controller. Cooperative landing trajectories are then calculated to satisfy the multi-robot coverage problem, UAV and UGV dynamics, and any local constraints. The proposed methods are validated experimentally and by simulation.

Finally, we ensure that the farmer is able to take control of the inspection process by introducing a human-robot interaction framework. This gives the human agency in an otherwise autonomous process, and allows them to preempt the selection of inspection targets via pointing gestures (e.g., to tell the robot to scan a certain plant). To achieve this, a human pose and gesture estimation module is proposed. Previous results from human-following robots are used to provide the multi-robot system with safe and robust human-aware path planning and obstacle avoidance behaviors. Pointing gestures are then used as inspection suggestions. These methods constitute one of the first human-robot interaction models for autonomous crop inspections. Experimental results are provided to show the effectiveness and intuitiveness of the approach.

Acknowledgements

First, I would like to thank my advisor Prof. Jingang Yi for his guidance throughout my studies. I am grateful for his pointed discussions during our meetings, as well as the many life lessons he has taught me. I would also like to thank my committee members Prof. Benaroya, Prof. Burlion, and Prof. Huang for their comments and insight relating to my research, which were invaluable when writing this thesis.

I would like to thank Prof. Assimina Pelegri and Prof. Alberto Cuitiño for providing me with the chance to TA for Design and Manufacturing. I would not be where I am, both academically and professionally, without that opportunity. I would also like to thank the Rutgers MAE departmental staff for their assistance in navigating the many aspects of my TAship and graduate studies: John Petrowski, Paul Pickard, Cindy Cartegna, Carmen Elsabee, Shefali Patel, Laura Kasica, Sania Sadhvani. I am especially grateful for Mr. John Petrowski's endless help and friendship during my tenure as head TA, and for his engineering and fabrication guidance throughout my studies.

I would like to thank everyone involved in the 2018 FutureMakers Challenge, and in particular: H. Sinan Bank, Rizwan Majeed, Arturo Pizano and Kurt Bettenhausen from Siemens; Dean Tom Farris, Prof. Alberto Cuitiño, Cherise Kent and Anda Cytroen from Rutgers. I would also like to extend my appreciation to our graduate director Prof. Jerry Shan, who let me be involved in the early discussions.

I am also grateful for the guidance of Naveen K. Singa and Max Wang from Siemens, whom I had a chance to work with both in a collaborative capacity when at the Rutgers RAM Lab, and in a professional capacity when at Siemens Corporate Technology in Princeton. A good portion of this thesis would not exist without their help.

I would also like to thank all my friends in the lab and at Rutgers: Tarık Yiğit, Dr. Mitja Trkov, Marko Mihalec, Dr. Kuo Chen, Dr. Kaiyan Yu, Yongbin Gong,

Kyle Hunte, Chaoke Guo, Dr. Pengcheng Wang, Dr. Siyu Chen, Semih Çetindağ, and Stephanie Rossi. I can't say I will miss going to weekly meetings with any of them, but I will definitely miss working with them. I wish them all good fortune and happiness.

I would also like to mention my great appreciation for my friends outside of Rutgers, and especially (in no particular order): Dr. Talya Yerlici, Dr. Ece Ertürk, Drs. Melis and Onur Önel, Dr. Görkem Garipler, Dr. Melis Duyar, Aslıhan Saygılı Torun, Dr. Korhan Koçak, Cem Ölçer, Levent Tüker, Sinan İlter, Natasha and Dr. Geoff Purdum, Nick Davy, Dr. Maria Frushicheva and Dr. Petr Khlyabich, Mariah McLaughlin and Dr. Emre Türköz. They made the intolerable tolerable.

Most importantly, I would like to thank my family, who have never stopped supporting me. My parents, Bike and Colin Edmonds, who immeasurably shaped my curiosity and my intellectual endeavors; my mother- and father-in-law Sürel and Osman Sezen, whose image of me I've tried to be worthy of; my sister-in-law and her husband and daughter, Merve Sezen Akbaş, Selman Akbaş, and Ada Defne Akbaş, who are a constant source of happiness; the Cennet family, who provided my wife and me a home away from home; my uncles, aunts, cousins, and their families, who always seemed interested in what I was doing, even when what I was doing was completely uninteresting; my grandparents, Ayhan and Nevzat† Kurdoğlu, and Anna† and Bill† Edmonds, for loving and supporting me like only grandparents can. Finally, I'd like to thank my wife, Dr. Melda Sezen Edmonds, for enriching my life in more ways than I know how to describe, and believing in me even when I didn't believe in myself. My PhD studies (and life in general) would be incomplete without her.

Dedication

To my unendingly patient wife.

Table of Contents

Abstract ii

Acknowledgements v

Dedication vii

List of Tables xii

List of Figures xiii

1. Introduction 1

1.1. Motivation 1

1.2. Background 4

1.2.1. Agricultural Robots used in Precision Agriculture 5

Manipulators 5

Ground Robots 6

Aerial Robots 7

Multi-Robot Systems 7

1.2.2. Computer Vision for Agricultural Applications 8

Spectral Data for CV-Based Phenotyping 9

Spatial Data for CV-Based Phenotyping 11

1.2.3. Challenges for Agricultural Multi-Robot Systems 12

1.2.4. Human-Robot Interaction for Crop Management 14

1.3. Dissertation outline and contributions 14

2. Cooperative UGV-UAV Inspection Platform 20

2.1. Introduction 20

2.2. Proposed Platform 21

2.2.1. Hardware requirements . . . 21
2.2.2. On-board sensors . . . 23
2.3. Experimental Prototype . . . 24
2.4. System Dynamics . . . 26
2.4.1. UGV dynamics . . . 26
Bikebot . . . 26
Three-wheeled omni-directional UGV . . . 27
Simulated UGV . . . 28
2.4.2. UAV dynamics . . . 28
Rigid Body Dynamics . . . 29
Aerodynamics . . . 29
2.5. Software Architecture . . . 31
2.5.1. Overview of software architecture . . . 31
2.5.2. Communication and storage models . . . 34
2.5.3. Component-level details . . . 34
Mission Coordinator . . . 34
UAV Optimal Trajectory . . . 35
UGV Human-Aware Multi-Robot Planner . . . 35
Multi-Robot Task Allocation . . . 35
3D Hyperspectral Reconstruction . . . 35
Next-Best-View Planner . . . 35
2.6. Conclusion . . . 36

3. Robotic Crop Evaluations using 3D Hyperspectral Scans . . . 37
3.1. Introduction . . . 37
3.2. Multi-Camera Hyperspectral Imaging Model . . . 39
3.2.1. Lighting Model . . . 40
3.2.2. Reflectance Model . . . 41
3.2.3. Camera Model . . . 43

3.3. Camera Spectral Calibration Method 45
3.3.1. Ground Truth Curve Calculation 45
3.3.2. Field Auto-Calibration via Spectral Decomposition 46
3.4. Formalization of the Inspection Task 49
3.5. 3D Hyperspectral Reconstructions 51
3.6. Next-Best-View Planning 52
3.7. Hyperspectral Classification 55
3.8. Results 56
3.8.1. Camera and Light Calibration 56
3.8.2. Reflectance Estimation 57
3.8.3. Spectrally-Optimal NBV 59
3.8.4. Hyperspectral Classification 63
3.9. Conclusion 64

4. Multirobot Planning and Allocation for Inspection Tasks 67
4.1. Introduction 67
4.2. Multi-Robot Inspection Problem 68
4.3. Estimation of the metric field 69
4.4. Optimal Multi-Robot Task Selection and Allocation 72
4.4.1. Multi-Robot Task Allocation for Discrete Observations 72
4.4.2. Multi-Robot Task Allocation for Continuous Observations 78
4.5. Simulation results 88
4.5.1. Estimation of the metric field 88
4.5.2. MRTA for discrete observations 88
4.5.3. MRTA for continuous observations 90
4.6. Conclusion 92

5. Cooperative Landing and Takeoff for UAV-UGV Systems 94
5.1. Introduction 94
5.2. Near-surface dynamic model for multi-rotor UAVs 95

5.3. Model identification for maneuvers close to the ground 97
5.4. Trajectory optimization based on predictive controller 99
5.4.1. Trajectory optimization problem 100
5.4.2. Trailing Window Disturbance Predictions 102
5.4.3. Linear Time-Varying Model Predictive Control 103
5.4.4. Iterative LTV MPC with State-Like Disturbances 104
5.5. Experimental results 106
5.5.1. Single-Rotor Data Collection Setup 106
5.5.2. Near-Surface Effect Prediction Results 108
5.5.3. Trajectory Optimization Results 112
5.6. Conclusion 115

6. Enhanced Inspections through Human-Robot Interaction 117
6.1. Introduction 117
6.2. Gesture-Based Inspection Suggestions 118
6.3. Human-Aware Path Planning and Navigation 119
6.4. Experimental Results 124
6.4.1. Experimental Setup 124
6.4.2. State and Goal Switching 125
6.4.3. Non-Blocking Pose Selection 127
6.4.4. Multi-Robot Planning 128
6.4.5. Gesture-Based Suggestion Tracking 128
6.5. Conclusion 129

7. Conclusions and Future Work 131
7.1. Conclusions 131
7.2. Future work 133
References 136

List of Tables

2.1. Hardware requirements for proposed platform 22

2.2. Sensor requirements for proposed platform 23

3.1. Comparison of Spectrally-Optimal NBV Methods 59

3.2. Completeness for sequences of one, two and three views. 60

3.3. Comparison of approximate and actual IG metrics. 62

3.4. Model and Inference Details for HSC 65

4.1. Simulation Parameters 91

6.1. Planner comparison for large-scale multirobot simulations 128

List of Figures

2.1. The experimental prototype designed by the Rutgers Robotics, Automation and Mechatronics Lab, consisting of a custom bikebot base, a Kinova arm with a KG-2 gripper, and a 3D-printed camera mount that houses an Intel RealSense depth camera, a Point Grey industrial GigE color camera, and a Flir A65 thermal camera. 24

2.2. 3D-printed camera mount installed onto the KG-2 gripper. Several cameras can be mounted onto the gripper to provide phenotyping capabilities to the mobile manipulator. 25

2.3. 3D-printed ground probe used to collect environmental and ground measurements. 25

2.4. (a) Single-track two-wheel steering bikebot developed in the Rutgers RAM Lab. (b) Schematic for the bikebot. (c) Back view of the rolling motion. (d) Top view. 26

2.5. A three-wheel omni-directional robot used for in-lab experiments. The robot is fitted with color stereo and RGB-D cameras. Vicon motion tracking cameras are used to validate the experimental result. 27

2.6. System architecture for a distributed multi-robot inspection system. The business logic for the system is handled by an on-premise industrial PC, while low-level controls and data collection is handled by on-board computers installed onto each robot. 32

2.7. Node diagram for the ROS implementation of the proposed software architecture. 33

3.1. (a) The Phong reflection model. Direction vectors $\hat{\boldsymbol{L}}$ and $\hat{\boldsymbol{V}}$ are dependent on the position of the light and the viewer, respectively, while $\hat{\boldsymbol{N}}$ depends on the point of reflection. (b) Example of specular reflections on a pepper plant, where the plant surface looks whiter due to direct reflections. 42

3.2. A typical scene for the NBV planning task. Cameras are fixed to a mounting system and pointed toward the object, capturing the images on the left. Illumination is provided by lights l_1 and l_2. 43

3.3. (a) Field setup for a hyperspectral camera. (b) Color checker calibration tile used to determine camera sensitivity and light spectral power distribution curves. 45

3.4. An illustration of the NBV procedure. (a) The space of all possible views is calculated as V. (b) A graph $\mathcal{G}$ of neighboring views is calculated by sampling V. (c) $\mathcal{G}$ is pruned to generate candidate views V_C. (d) An approximate NBV sequence is selected from V_C as $\mathcal{V}^*$. 53

3.5. The proposed healthy/unhealthy hyperspectral classifiers based on a relatively shallow deep neural network. HSC-Full (left) takes high-resolution hyperspectral data as input, whereas HSC-Coeff takes reflectance coefficients as input. 55

3.6. (a) Estimation errors for each camera's sensitivity curve, normalized across all wavelengths for which data was available. The performance of the estimator decreases around the fringes for the selected basis, but is within acceptable limits. (b) Estimation errors for each light's spectral power distribution, normalized across all wavelengths for which data was available. 56

3.7. Camera and light coefficients converge after a few iterations of the alternating least squares estimator. Here, the optimizer alternates between minimizing costs for κ (white background) and for ς (light red background). The first cost terms (normed $\tilde{\mathbf{P}}$ errors) are plotted in blue, and the second cost terms (Tikhonov regularization cost) are plotted in orange. Cost coefficients are set to $\alpha_1 = \alpha_2 = \alpha_3 = 1$. 57

3.8. Effect of Tikhonov regularization on estimation results. (a) Estimation error cost generally decreases with increasing α_2 and α_3. (b) Increasing the Tikhonov cost weighting reduces sharp rises and falls in the final estimate. (c) Basis functions derived from principal components tend to have higher Tikhonov cost as the basis number increases. 58

3.9. (a) Estimated (solid) and ground truth (dotted) reflectances for the primary color calibration tiles. (b) Average reflectance errors for all tiles. Reflectances are calculated by assuming diffuse-only reflections, and solving for ($\boldsymbol{\rho}$) like any other reflectance. Both κ and σ are fixed after camera and light calibration, so any errors in those coefficients carry over to the reflectance estimation, which can skew results. 58

3.10. Reflectance coefficients ($\boldsymbol{\rho}$) are calculated by fixing all other variables, calculating the residual error term e_R and minimizing Eq. (3.23). Estimates for the coefficients converge rapidly (within a couple of optimization steps), but require several iterations of the three-step optimization process for quality estimates. 59

3.11. (a) A representative hyperspectral point cloud. Each point in the cloud contains both position and normal data as well as spectral data such as mean reflectances and their variances, etc. (b) Flowchart of hyperspectral point cloud generation from camera images. Raw sensor data $p_i(\boldsymbol{m})$ is used in conjunction with known spectral power distribution and camera sensitivity curves for the lights and the sensors, respectively, to generate an estimate of the reflectance curve. The reflectance curve is calculated and stored as a linear combination of basis functions, which allows for smaller storage and faster retrieval. 60

3.12. NBV refinement using Algorithm 4 for $N_v = 3$. Each dot represents the position of the central camera in the corresponding view candidate. The optimal view sequence moves along the utility gradient in the neighborhood of $\mathcal{V}$ at each step of the optimization. Cameras and the lines connecting them are the same color within a sequence. 61

3.13. Comparison of multiple sequences containing the same initial view. (Left) Points captured by the sequence are shown on top of the full mesh. Points covered by the 1-sequence are colored red, points covered by the 2-sequence are colored green, points covered by the entire 3-sequence are colored blue, and points that were not covered by the sequence are colored black. (Center Left, Center Right, Right) Variances are shown in shades of gray. Lower variance data are darker. 62

3.14. Comparison of a reconstructed experimental hyperspectral point cloud to the ground truth hyperspectral point cloud. Point intensity denotes relative reflectance error, darker is less error. 62

3.15. Comparison of the imec Classifier and HSC-full. (a) Color image of original hyperspectral data, where pure white indicates spectral oversaturation, (b) Classification results obtained using the imec Classifier, (c) Classification results obtained using HSC-full, (d) Difference between the two classification results, where darker is better. 63

3.16. Classification results for stressed/healthy fescue grass samples raised in a growth chamber. Spectral curves are colored according to the predicted class, where green is healthy and red is stressed. Higher saturation indicates higher concentration of one class over the other. 64

4.1. Schematic of kernel estimation network $K_\theta^{(t)}$. (a) Data flow diagram for the network. Time series data is resampled into pair-wise vectors and fed through the deep neural network (DNN). Resulting kernel estimates are used to construct custom mean squared error and temporal coherence losses as defined in (4.5). Training is performed using an Adam optimizer. (b) Overall DNN architecture, implemented in Tensorflow-Keras as a sequential network of dense layers (fc) with regularization and dropout. 71

4.2. The difference in Voronoi partitions when the diagram is generated directly on a plot $\mathcal{P}$ (left) and when it is generated using G (right). 75

4.3. Unmanned aerial/ground crop inspection missions are often used to capture high-resolution crop imagery for phenotyping or yield estimation applications. The spatial (pixel) resolution of the plant image is inversely proportional to the distance. 81

4.4. Energy constraints limit robots to subsets of their reachable set for long missions where charge depletion is undesirable. The reduced reachable set depends on the selection of a starting charging station, and is bounded by $\overline{d}_j$ afterwards. Here, the UGV (green circle) selects between the subset defined by stations A-B-C (dotted blue) or the subset defined by stations D-E (dotted pink). The subset defined by stations F-G (dotted red) is unreachable. 84

4.5. Unlike UGVs, UAVs can hop between regions outside their own range limit by landing on cooperative UGVs. Here, the UAV (orange star) can use the UGVs (blue circles) to hop between their reachable sets (dotted blue) by combining UAV-only moves (orange arrows), UGV only moves (blue arrows) and free moves (black arrows). The resultant reachable set for the UAV is shown as a dotted orange outline. 87

4.6. The approximate reachable set for a UAV r_i (orange star) is calculated by iteratively expanding an initial approximate reachable set $\tilde{R}_i^{(1)}$ (solid light orange) to first include a set of points $\tilde{R}_i^{(2)}$ (solid orange) reachable with the help of an adjacent UGV r_j with reachable set $\tilde{R}_j$ (dotted blue), then repeating the expansion to include a set of points $\tilde{R}_i^{(3)}$ (solid dark orange) reachable using adjacent charging stations. 88

4.7. Comparison of metric fields for our simulated model (a) using anisotropic Gaussian RBF, (b) using kernel estimation, and (c) ground truth data. We sample only 30 randomly selected locations for each t. Our learned kernel produces better long-term predictions compared to the RBF-based kernel under identical sampling constraints. 89

4.8. (a) Retopography times and (b) Pathfinding solution times for various $|\boldsymbol{R}|$ using Python's `networkx` and `pathfinding` libraries. Targets are selected either as random points in $\mathcal{P}$ that are not necessarily reachable from the robot's position, or as known reachable points. Retopography in this sense constitutes any restructuring of the underlying graph or map. 90

4.9. Comparison of (a) total cost and (b) total travel distance for the same plot using Voronoi diagrams that use graph distance (red) vs. L2 distance (blue). Both use our greedy allocation method, but with different d_i. Our method travels shorter distances for comparable cost. 91

4.10. Comparison of planning metrics for DARP without energy constraints, DARP with energy constraints, and ECMP. 92

5.1. A multirotor landing on a mobile platform. Thrust and torques depend on the robot's proximity to any hard surfaces such as the ground. 95

5.2. Data flow for (a) our near-surface estimation network and (b) our multi-rotor prediction network. Pose and velocity information is concatenated with latent-space representation of nearby surfaces and passed through a DNN-based prediction network. Single-rotor predictions are then used to predict corrected thrusts. 97

5.3. Planar quadcopter model with state-like disturbances. 102

5.4. Experimental setup for thrust measurements. A 5kg load cell attached to a weighted cantilever arm is used to measure rotor thrust without impeding airflow. Polycarbonate sheets are positioned under the rotor to simulate a UGV-mounted landing pad and to change flow conditions. 106
5.5. Experimental thrusts for different configurations. 108
5.6. (a) Single-rotor thrust predictions. Predicted thrusts (light blue cross marks) follow the ground truth thrust distribution (mean at solid dark blue line), and (b) Single-rotor thrust errors corresponding to the per-sample thrusts shown in (a). Error distributions remain roughly constant across test heights. 109
5.7. Ground truth (left) and reconstructed (right) landing pads. Note the direction and height differences between pads (a) and (b). 109
5.8. Normalized prediction errors for the trained single-rotor network. 110
5.9. Heatmap of normalized experimental errors for various platform configurations. 111
5.10. Time complexity for the prediction network with varying N_o. 111
5.11. (a) Corrected thrust predictions for multi-rotor estimation network. (b) Prediction errors for the multirotor thrust network. 112
5.12. Comparison of the trajectories obtained by using a Cascaded PID$_{ts}$ tuned to minimize settling time (green), a Cascaded PID$_{os}$ tuned to minimize overshoot (blue), and our LTV MPC with SLD method (red). The quadcopter and the two inputs are plotted at 15 equitemporal points along each trajectory, with the trajectories encompassing roughly 20, 40 and 6 seconds, respectively. The length of the protruding segments correspond to the thrusts for each motor. 113
5.13. Comparison of state and input trajectories for PID$_{os}$ (top), PID$_{ts}$ (center), and LTV MPC w/ SLD (bottom). Note that the timescale for each set of plots is different. 114

5.14. Iteration process for trajectory correction. At each step, an MPC solution is used to perform a rollout. The rollout is then used to constrain and initialize then next iteration. Corrections are made to not only the $\boldsymbol{x}$ trajectory shown here, but also to the $\boldsymbol{u}$ and $\boldsymbol{\nu}$ trajectories (not shown). 115

6.1. (a) Point cloud data is used to estimate user pose and gait. (b) Human pose estimates determine robot pose suggestions by casting a ray from the extended arm onto the environment and selecting the closest feasible pose in $\mathcal{X}_i$. Optimal trajectories are calculated via (6.4) to a pose within a polytope $\mathcal{G}_i$ around the pose suggestion. 119

6.2. The robot's operational mode and its goal pose are determined by a state machine with three states: *Following (F)*, where the robot keeps within δ_f of the human, *Waiting (W)*, where the robot parks itself nearby, and *Approaching (A)*, where the robot approaches to within δ_a. 121

6.3. The robot transitions from its *Following* state to its *Waiting* state when the robot reaches the following-to-waiting transition radius $R_{f\to w}$, and resumes following by transitioning from its *Waiting* state to its *Following* state when the user reaches the waiting-to-following transition radius $R_{w\to f}$. Switching to the robot's *Approaching* state pre-empts this transition, and instead triggers an approaching-to-following or approaching-to-waiting transition. 121

6.4. Data flows in a similar manner for both the simulation and the experimental platforms. Sensor data is used to perform skeleton tracking, trajectory prediction, localization and local obstacle mapping. Skeleton data is used to determine user gestures, which are then used to determine positioning suggestions. Gesture, trajectory and localization data is used to determine each robot's state as shown in Fig. 6.2. The goal pose is used to plan a global and local optimal trajectory. 124

6.5. An environment is simulated in Gazebo, and includes up to 40 autonomous robots and simulated humans. Several humans can be actively controlled by the user to test pointing and walking behavior, and the rest are simulated using crowd dynamics. 125

6.6. (a) Plot of the distance between the robot and its selected goal. (b) Comparison of the goal position and the human position. The goal pose initially aligns with the human's pose due to the absence of a trailing path, and is subsequently switched to an optimal waiting pose as the robot crosses $R_{f\to w}$. Following behavior resumes when the human starts moving away, with a goal pose that trails the human's trajectory by δ_f. The robot selects a waiting pose based on user suggestions once it reaches $R_{f\to w}$ again. (c) Comparison of robot and human positions as influenced by distance-dependent state switching. 126

6.7. Results from a user following test, where the robot was asked to follow the user at a distance of $R_{f\to w} = 1$ m. The robot initially follows the user until it reaches $R_{f\to w}$, and then moves to the suggested waiting pose (purple). The robot resumes following once the user moves further than $R_{w\to f}$. 126

6.8. Optimal non-blocking poses selected for various scenarios. (a-c) User suggestion is varied along the projected path. (d-g) User approaches and passes robot. 127

6.9. (a) Combined global trajectories for 50 people, using data from all trials. (b) Trajectories from a single trial (no robots) colored by person, showing high-activity areas and bottlenecks. (c) Trajectory data from a trial with robots using the multirobot planner. Map resolution is 5cm/px. 128

6.10. (a) Experimental gesture-based suggestion tracking results for a single trial. (b) Histogram of normed tracking errors across all trials, with mean error (dashed red) median error (solid black). 129

Chapter 1

Introduction

1.1 Motivation

In 2009, the United Nations General Assembly held a panel discussion as part of the World Summit on Food Security to determine long-term strategies to address the effects and causes of the 2008 global food crisis [1]. The summit produced many actionable food security goals, including the oft-cited goal of doubling global production by 2050 [2], which has been one of the main drivers for agricultural research in the past decade. Unfortunately, recent global projections still fall short of the 2050 goals despite the major technological progress made since 2009 [3–5]. In many ways, a Fourth Agricultural Revolution and a decades-long boost in yields are required to meet the 2050 goals and respond to emerging demographic and environmental challenges [6]. While there are many non-engineering solutions to this yield problem [7–9], these solutions by themselves are not enough. By all accounts, non-engineering solutions must be paired with better agricultural automation practices based on cross-domain technologies such as machine learning [10], computer vision [11], artificial intelligence [12], and agricultural robotics [13]. These technologies act as enablers for better crop management, and form the core of precision agriculture (PA) methodology. These technologies focus on maximizing yields by optimizing growing conditions and providing growers with smarter farming tools [14]. The primary motivation for this dissertation is therefore the expansion of PA to include autonomous robotic inspection platforms that utilize the aforementioned cross-domain technologies.

The technical challenges that prevent the immediate integration of these new technologies are varied. First and foremost, it is difficult to design and build agricultural inspection systems that handle both the robotics aspects and the sensor and data

management aspects required for high-throughput inspections. The lack of a unified framework and architecture therefore limits the applicability of recent advances, especially since a lot of the work is proprietary. Similarly, the different types of agricultural data require separate software and analysis pipelines, which are often difficult to integrate with larger scale decision systems. For example, non-intrusive inspections utilize different data modalities (e.g., point clouds to capture morphology and color and near-infrared images to capture spectral data) that all require separate sensors connected to a custom program that uses proprietary drivers for each device. A more canonical representation could be used to store both morphological data and spectral data in a combined hyperspectral point cloud (HSPC), which could then be used to recover these data modalities for specific analysis pipelines. There are many active research questions surrounding the use of HSPCs, starting with the lack of efficient methods to generate HSPCs from common sensors such as color cameras.

Furthermore, it is unclear how we can acquire inspection data for not just a single plant but for entire fields. One approach is to use single-plant data from a sample of plants to predict field-wide conditions, similar to the state-of-the-art in environmental monitoring. This is a natural extension of how we approach phenotyping today, and would enable us to perform the same type of phenotyping operations (and thus retain the same pipelines) while also generating complete data about the field as a whole. However, this would require new methods to determine which subset of plants among millions to select for single-plant scans, and additional new methods to determine an estimation model for the entire field based on this sparse sample. There are also unsolved robotics challenges associated with this sampling approach, mostly due to the scale of the inspection problem. First, how do we allocate these scanning tasks to the numerous robots that will be needed to cover the entire field in a timely manner? Second, how do we ensure that this allocation makes physical sense so that the robots do not block each other while navigating between scan targets? Third, how do we connect this allocation method back into the field-wide estimation model to minimize the number of scans necessary to make an accurate assessment? These are all complex questions without definitive answers in the state-of-the-art developments, and they are all parts

of the larger multi-robot crop inspection problem. However, they are questions that must be answered in the immediate future, since accurate and timely data collection is a critical part of what enables PA to increase yields so effectively.

In addition to the autonomy challenges, human-centric challenges must also be addressed. Despite the yield increases already provided by current PA methods, the adoption of PA systems has been lackluster [15]. Farmers know that these technologies provide an edge to the larger producers that are able to absorb the upfront costs, and they are cognizant that they will need to adopt PA methods to stay competitive, yet they neither trust nor want the technology. According to USDA, the main barrier to entry is the technology itself, and specifically that it is costly and complex with uncertain economic returns [15]. While USDA focuses heavily on education as the solution to this problem, it is equally valid to consider it as a technology problem: Adoption can be made more enticing by improving the autonomy and interactivity of the systems. In the short term, technical barriers can be lowered by introducing intuitive ways to tell the robots what to scan or where to go, rather than relying on the end-user to spend the resources to learn and unilaterally integrate complex and unintuitive engineering tools. These types of human-robot interactions are already extensively studied in other domains (e.g., manufacturing), and in many cases, the methods developed in other domains are applicable to agriculture as well. If improving human-robot interactions can improve the adoption rates for robotic crop inspection systems, it is absolutely worth the additional research effort.

For these reasons, this dissertation addresses the technical challenges and the adoption barriers concurrently by introducing a design and methodology for user-guided, autonomous multi-robot crop inspections. The requirements for such a system are varied and complex, and the multi-disciplinary work needed to design commercial robots that perform inspection tasks traditionally carried out by experts exceeds what can be covered in a single dissertation. This dissertation therefore focuses on a subset of questions relating to robotic phenotyping, multi-robot planning and control, and human-robot interactions. Specifically, the following challenges are addressed:

(C1) The evaluation of common metrics (e.g., crop health) using on-board sensor data,

(C2) The selection of a representative subset of plants such that an entire field can be monitored without inspecting each plant,

(C3) The development of control and planning algorithms for multi-robot inspections, and

(C4) The use of on-board sensors to enable intuitive, human-guided inspections.

Each chapter of this dissertation tackles a portion of these challenges, starting with the computer vision problem described by (C1), leading into the the multi-robot planning and control problems described by (C2) and (C3), and concluding with the human-robot interaction problem described by (C4). The next section provides the technical background necessary to discuss these problems.

1.2 Background

Key agricultural tasks such as seeding, picking, weeding, disease detection, monitoring, and phenotyping are now being roboticized by combining vision algorithms with artificial intelligence and mobile robotics. Such roboticization is critical to increasing yields, and enables growers to manage larger farms without additional personnel requirements. Naturally, the methods and robots developed for these applications are as varied as the applications themselves, and the hardware is often tailored for certain crops. In contrast, this dissertation focuses on the development of a general multi-robot crop inspection platform that can handle multiple agricultural use cases. The proposed hardware and software architectures contribute to a modular and extensible platform, and the developed computer vision and multi-robot control and planning methods are critical to ensuring a group of these platforms can perform high-throughput inspections. Similar to the problem it is meant to solve, the platform is comprised of many distinct but connected parts.

The review presented in this section therefore covers a wide array of topics such as agricultural robots for precision agriculture, the vision systems that enable robotic phenotyping, the planning and coordination challenges associated with multi-robot systems, and current human-robot interaction use cases for crop management. Research

results outside the agricultural domain are also included due to their potential applications in the agricultural domain [16]. Note that the work in this dissertation also has applications in other domains such as mining, ecological surveying, and industrial inspections. For a comprehensive analysis, readers are referred to [13] for an overview of agricultural robotics and [17] for a recent survey of the agricultural applications of robotic inspections.

1.2.1 Agricultural Robots used in Precision Agriculture

Precision agriculture methods require the collection of large amounts of data for a large number of plants. As such, the use of robotics (and in particular, mobile robotics [18]) is a strong requirement for effective PA. Robots used for PA can be divided into roughly three categories corresponding to their capabilities: manipulators, ground robots, and aerial robots [16]. Their respective and combined uses for agricultural inspections are detailed below.

Manipulators

Robot manipulators are standalone robot arms that consist of linkages and joints, which form open chains such that the base of the robot is fixed, and the other end (i.e., end effector) is free [16]. Manipulators used in agriculture have anywhere from 2 to 9 degrees of freedom (DOFs) corresponding to the number of joints, with 5 or 6 DOFs being the most common [17]. The wide range of possible joint configurations and the customizability of the end-effectors and grippers make manipulators versatile tools for picking, spraying, watering, and weeding tasks. The manipulator can also be used to position cameras for vision applications by attaching cameras and sensors to the end effector (see Section 1.2.2 for more detail on computer vision in agriculture). Compared to industrial arms, modern manipulators are often lightweight and portable (i.e., operated with batteries), and are thus suited for applications that require moving from plant to plant. A review of manipulator and gripper technologies can be found in [19].

In this work, we consider the combination of any number of manipulators and a single ground robot to be a *mobile manipulator* [20], and treat it as a single robot

when defining multi-robot systems in later sections. The assessment of the capabilities and manipulation performance of these combined systems is outside the scope of this dissertation, and readers are instead directed to several related surveys [21, 22].

Ground Robots

Ground robots are mobile robots that move primarily on land, and can be categorized based on their mode of locomotion as wheeled, tracked, hybrid, or legged [18]. Classifications within these categories can also be used to further distinguish robots based on their motion modalities (e.g., omni-directional or skid-steer). Most agricultural robots nowadays use wheeled locomotion to navigate crop field, but legged or tracked robots are more suitable for certain applications [23]. The mobility provided by ground robots is essential to systems that need to position manipulators and sensors to accomplish manipulation task such as crop inspections.

The use of ground robots in agriculture is a century-old concept [24], of course, but the use of self-steering and unmanned ground vehicles (UGVs) is a relatively new trend that was pioneered in the early 2000s by John Deere [25]. It is now estimated that around a third of all cropland in the United States is plowed by self-guided robots such as driverless tractors [25]. A large number of ground robots used commercially (e.g., for surveillance, agriculture and military) still require teleoperation [26] (i.e., require a pilot to issue movement commands), but a wide variety of map-based techniques such as GPS navigation and simultaneous localization and mapping (SLAM) [27], and mapless methods such as optical flow [28] are also available to support autonomous navigation functions [29–34].

Since the differences between UGVs used for agriculture and other purposes are application but not technology-based, many recent developments in other application domains carry over to agricultural UGVs as well. Readers are directed to [35–39] for recent reviews.

Aerial Robots

Aerial robots used in PA are typically unmanned aerial vehicles (UAVs) such as fixed-wing planes (e.g., gliders) or multi-rotor copters (e.g., quadcopters). The applications of UAV are numerous and central to PA, and range from land mapping to high-resolution imaging. Readers can refer to [40–47] for a wide variety of in-depth surveys on the agricultural uses of UAVs. They are also heavily regulated by agencies such as the Federal Aviation Administration (FAA), and cannot be operated in certain airspaces [48].

UAVs can cover and inspect a large area very quickly, but are limited in range and flight time by their on-board fuel (e.g., batteries) [49–52]. The versatility of quadcopters in particular, combined with their cost-effective designs and commercial availability, make them ideal tools for fast and reliable data collection. However, they are limited by aerodynamics, and are therefore unsuitable for missions that require close-up imaging. Their inherent dynamic instability also require them to be actively controlled to maintain their position, further complicating the dynamics. While some quadcopter-based manipulators exist [53–55], running a manipulator off of the on-board power supply is a costly proposition from a fuel perspective, which makes controlling the quadcopter even more difficult, even for low DOF arms (see [56] for a survey of the state-of-the-art in aerial manipulation). As such, quadcopters are mostly used as top-down portable imaging platforms.

Multi-Robot Systems

Combining the complementary capabilities of UGVs and UAVs can provide many benefits to PA systems. For example, UAVs are uniquely equipped to survey large areas, whereas UGVs can perform tasks that require close contact or precision, thus making the combination of the two systems attractive for many PA tasks [57]. These types of integrated approaches for tasks such as data collection and processing are already common in many other application domains [58–67], but have not yet been implemented

fully in the agricultural domain. [68] provides a UAV-centric review of various UAV-UGV cooperation approaches in agriculture, which also includes an overview of the different platforms currently in use. [69] discusses adaptable solutions to UAV-UGV systems for PA, and [70] reviews current and future trends in cooperative UAV-UGV phenotyping. Challenges related to the use of multi-robot (and specifically, UGV-UAV) systems in agriculture are discussed in Section 1.2.3.

1.2.2 Computer Vision for Agricultural Applications

Computer vision (CV) is the use of optical signal and image processing techniques for the purposes of programmatically describing, understanding or otherwise interpreting an image [71]. Over the last decade or two, advances in computer vision have gone hand in hand with advances in artificial intelligence (AI). The proliferation machine learning (ML) methods such as deep neural networks (DNNs) and convolutional neural networks (CNNs) have paved the way for many modern PA techniques. The rapid increase in the computational capabilities of on-board and off-board computers have enabled the use of many ML methods developed in other application domains for plant inspections instead, and much of the research focus for PA has shifted to ML-related methods (see [72, 73] and [74, 75] for reviews of current ML trends in general and in agriculture, respectively). The result has been the automation of many agricultural tasks that would have been impossible a decade or two ago: Robots can now count [76–82] and pick [83–91] fruits, identify and destroy weeds [92–103], and perform high-throughput phenotyping [104–112], and much more, all of which require robust computer vision methods.

In many cases, successful application of CV techniques requires the fusion of different data modalities that correspond to different plant processes. For example, thermal (infrared) imaging is often used to detect plant water content and heat-related stresses [113, 114], whereas phenotyping often requires structural characterization, and therefore relies on three dimensional data captured using various ranging or imaging techniques instead [115]. The application of computer vision techniques to such multi-modal data therefore enables high-throughput analyses of many quantitative plant traits [116].

Modern plant phenotyping can therefore be understood as the use of non-invasive imaging technologies to capture essential plant information [117–119]. This type of plant data has three information facets: the spectral, the spatial, and the temporal. Since we consider temporality to be an inherent aspect of the data collection process, the discussion below focuses on the first two aspects. Comprehensive reviews of the applications of computer vision to agriculture in general can be found in [120, 121].

Spectral Data for CV-Based Phenotyping

Most spectral phenotyping tools are based on fundamental theories developed for visual near-infrared spectroscopy (VIS-NIRS), which utilize the fact that the chemical composition of a sample affects how it reflects incoming electro-magnetic waves. The chemical composition (or many properties related to the chemical composition) can therefore be determined by analyzing relative reflection intensities at different wavelengths [122]. The curve that is produced by repeating these intensity measurements for a range of wavelengths (i.e., over the visible and near-infrared spectrum) is called the reflectance of the sample, and is the primary variable of interest for many phenotyping applications. Reflectance curves can be used to determine water stresses [123], nitrogen deficiencies [124], senescence [125], photosynthetic capacity [126], and many other plant metrics [127–130] in a non-destructive manner. The estimation of these curves therefore forms a critical part of the PA inspection toolchain. For some applications (e.g., the calculation of vegetation indices [131]), knowing the reflectance values for certain critical wavelengths might be enough. In such cases, cameras with a small number of narrow-band channels are used, and the reflectances for the critical wavelengths are extracted from the multi-spectral images generated by these cameras. Applications that require finer spectral resolutions or continuous reflectance curves, on the other hand, use hyperspectral data. Hyperspectral signals are typically captured using cameras that have 100 or so bands spread across the VIS-NIR spectrum at a spectral resolution of around 1-2 nm. [132, 133] review the major differences between multi-spectral imaging and hyperspectral imaging, and [134–136] review recent developments in hyperspectral imaging.

Spectral data in and of itself is, however, not as useful as investigating the patterns that emerge across many samples. This is a challenging task that is a very good match for ML-based CV methods [137], which are often considered as much a part of pattern recognition [138] as CV. The central goal of these methods (and arguably, any ML methods) is to train a prediction or classification model using previous data, and use the trained model to analyze new samples [74]. In the context of PA, this often means collecting large datasets of plants with desirable metrics (e.g., healthy) and undesirable metrics (e.g., unhealthy) and designing an ML architecture that turns the dataset into a model. Many such architectures exist in literature, and taxonomies are discussed in [139, 140]. In [141], a stacked hourglass CNN is used to phenotype wheat. In [142], an anomaly detection network is presented based on long short-term memory (LSTM) nodes. In [143], a multi-class support vector machine (SVM) is used to detect diseases in potatoes. Some other works such as [144, 145] focus on designing architectures that are explainable (i.e., provide some physiologically meaningful insight).

Spectral data used in PA applications are also often collected using remote sensing via satellites [146,147] (see [148] for a review of sensor technologies deployed in modern satellites). Remote sensing is attractive to researchers and growers since satellites can survey large fields with ease compared to ground or aerial robots. The data generated by the satellites are geolocated and timestamped [149], and more importantly, the multispectral images generated by newer satellites can be used to extract vegetation indices for the entire field [150–154]. Alternatively, raw multispectral data can be used to classify each pixel and extract other useful metrics such as canopy cover or crop type [155–159]. In works such as [160], the temporal characteristics of the target plots are also considered. A review of remote sensing applications for precision agriculture is presented in [161]. It is worth noting that the methods used for satellite-based PA can also be applied to or used in conjunction with UAV-based PA [162, 163]. The benefits of using UAVs instead of satellites lie mainly in the convenience and availability of UAVs for on-demand and higher-resolution data collection, whereas the personnel and post-processing burden may be higher, since data must first be collected by a pilot or technician and then coregistered and stitched together [164]. Otherwise, the data

produced by a satellite and the data produced by a UAV can be treated similarly in PA pipelines. Furthermore, the sensors used for satellite- or UAV-based PA can be installed onto UGVs. Adding a multi-spectral camera onto the end effector of a mobile manipulator, for example, opens the possibility of collecting spectral data with spatial resolutions on the order of millimeters instead of meters, and is therefore an enabling technology for emerging applications such as early disease detection. These kinds of close-up crop inspections are sometimes called proximal sensing [165] and marks the transition between determining crop-level metrics (e.g., whether a field is affected by a disease) and leaf-level metrics (e.g., whether any leaves show disease symptoms). Several field robots designed for proximal sensing are reported in literature [166–168], although many similar robots also exist and use different terminology for the same processes (see Section 1.2.1 for details).

Spatial Data for CV-Based Phenotyping

In comparison to spectral data, spatial data captures the structure and shape of the plant and can be used to segment or classify parts [169], or to calculate parameters such as leaf area and angles, plant height, and stem width. The combination of spatial and spectral data also improves segmentation and classification results [170, 171]. Methods to capture spatial data (sometimes called reconstruction or digitizing) are varied. If sensing capabilities are limited by payload capacity, battery power or unit cost, spatial data can be extracted from 3D reconstructions generated through structure-from-motion [172], multi-view reconstruction [173], or similar methods based in processing 2D data. Alternatively, sensors that produce native 3D data can be used directly [174,175]. Robot-mounted cameras are typically practical and often a sufficient tool for phenotyping. For example, in [176], time-of-flight and digital single-lens reflex cameras are used to extract 3D reconstructions of canola branches and seedpods. If the plants are moved (e.g., in a greenhouse setting), then high-resolution methods such as x-ray computed tomography can also be used [177]. Several commercial solutions, such as

the Phenospex PlantEye[1], also exist. In-depth reviews of current methods to capture 3D plant data can be found in [178].

Processing spatial data remains a challenging task. ML methods exist to tackle the various challenges [179]. Some interesting results can be found in [179–187].

1.2.3 Challenges for Agricultural Multi-Robot Systems

The benefits of using multiple robots come at a cost. While increasing the number of robots in operation also improves overall throughput in theory, practical concerns such as routing (pathfinding) difficulties and data management also start to emerge. More data requires more processing and storage, and is not necessarily desirable from a crop management perspective if less data can achieve similar results. Furthermore, increasing the number of active robots increases the fiscal burden on the farmer, since each robot must be maintained and refueled. As such, it is desirable achieve more with fewer robots. One way to achieve this is to treat the inspection problem as a sampling problem where the objective is to select a smaller number of sampling tasks (i.e., individual plants to be scanned) that are representative of the field as a whole. The challenge is to then determine a scanning task for each robot in the multi-robot team such that a good estimate of the condition of the entire field is obtained as fast as possible.

Many methods exist to evaluate entire fields with a limited number of samples. When treated as an orienteering problem [188], where the goal is to visit an optimized subset of all targets, allocations are made using tour optimization on a graph induced by the field's topography. These methods optimize a variety of metrics such as spatial correlations [189] or cluster coverage [190]. The related problem of adaptive sampling [191–194] is instead concerned with the selection of optimal sampling locations. Adaptive sampling methods are also sometimes modeled as a partitioning problem where each robot selects a sampling location based on a robot-centered Voronoi region [195–197], a priority map [198], or polygon enclosures [199]. The allocation problem is approached

[1] `http://www.phenospex.com`

from a control systems perspective. Methods exist to design optimal trajectories between sampling points [200, 201], or to solve the equivalent multi-robot exploration [202–204] and multi-robot coverage problems [205–209].

Modeling the entire field from a limited number of samples requires an inference model. The majority of adaptive sampling literature uses Gaussian processes for their desirable properties [210], with modifications such as level set estimation [211] and sparse online Gaussian processes [212] to deal with the dimensionality limits of non-parametric inference. Since the spatial model for Gaussian processes depends on the selection of a kernel, the types of spatial correlations that can be captured by the model are similarly limited by the kernel. Most methods use either radial basis functions of anisotropic Gaussian kernels, and therefore fail to account for other types of spatial correlations (e.g., pathogen spread through water channels). Additionally, most adaptive sampling methods do not consider pathing constraints when selecting successive sampling locations, and therefore produce suboptimal allocations. Some methods [213,214] deal with this explicitly by looking at a row-based routing problem, but do not deal with the kernel issues otherwise.

Another important aspect of multi-robot coordination in agricultural systems is path planning, since the constrained workspaces and the robots' potential for causing damage to crops. A review of path planning approaches for agricultural ground robots can be found in [215]. The birds-eye view provided by UAVs can also be used to help plan collision-free paths. UAV-UGV coordination for efficient pathing is investigated in [216]. Cooperation between fuel-constrained UAVs and UGV-based refueling platforms must also be considered, especially for large-scale mapping or surveillance missions [217].

If the robots are performing inspection tasks, we must also plan a sequence of optimal viewing angles for 3D reconstruction. This is commonly known as the next-best-view (NBV) problem, which also serves as a cornerstone of both robotic exploration and 3D reconstruction [218–222]. Reconstruction methods work by maximizing an information gain metric across a set of possible upcoming views [223]. The information gain metrics used for these methods usually only consider spatial data, so view selections

do not necessarily produce high-quality spectral data (e.g., due to glares). Another issues is that state-of-the-art methods only optimize the immediate next-best-view and do not consider the combinatorial nature of selecting view sequences. Deep-learning based planners for NBV selection have also been investigated in [109].

1.2.4 Human-Robot Interaction for Crop Management

Human-robot interaction (HRI) is an active field of research that deals with the way humans interact with robots to accomplish shared tasks [224]. It is obvious that the use of autonomous robots alongside farmers has created the need to devise control and perception schemes that guarantee human safety while allowing robots to perform their tasks as expected (see [225] for a discussion of HRI safety). On the other hand, close interaction between farmers and robots allows robot-based inspections to be tailored to the farmer's needs, and can prove useful for a number of supervised learning tasks. Research into HRI in agriculture as a whole, however, has lagged behind other industries such as manufacturing, and the body of works dealing with agricultural HRI are limited. Reviews of current HRI trends in agriculture are presented in [226–229], while some similar reviews on concerns that are common across multiple application domains are presented in [230–233]. See [234–236] for some specific applications of HRI methods to agriculture.

1.3 Dissertation outline and contributions

The dissertation consists of seven chapters. Chapter 1 introduces the vision, planning and control problems relating to autonomous crop inspection platforms and reviews recent developments from literature. Chapter 2 describes the construction, technical specifications, and software architecture of the experimental inspection platform, including the system dynamics. Chapter 3 describes the inspection task itself and the lab setup used for experiments. It also details a methodology for generating 3D hyperspectral scans. Chapter 4 investigates the planning problem surrounding large-scale crop evaluations, and introduces a task allocation algorithms for multi-robot inspections.

Chapter 5 discusses the cooperative control of a UAV-UGV system during landing and takeoff. Chapter 6 details a human-robot interaction framework that uses gesture-based feedback from the farmer to improve the site-specificity of the system. Finally, Chapter 7 concludes the thesis with a summary of the main contributions and commentary on possible future work. The content of each chapter is described in detail below.

In Chapter 2, we first present the hardware setup for a cooperative UGV-UAV inspection platform. We present system dynamics for the various robots used to validate the developments in the subsequent chapters. We also outline common simplifying assumptions used in the control and planning approaches. We then provide details for the on-board sensors and describe common vision models for the cameras. Communication and storage models are briefly discussed in the context of robotic inspections. Finally, we provide specifications for an experimental prototype consisting of a single-track robot and a light-weight robotic manipulator and sensor system.

In Chapter 3, we present the central inspection task for robotic phenotyping. We introduce a formalization for generating and evaluating 3D hyperspectral scans based on a novel information gain metric, similar to those used in 3D reconstruction literature. Completion of the inspection task is then described in relation to this metric, and captures both the spatial and spectral dimensions of the task. We propose a novel method to determine spectrally-optimal next-best-views, which is used to efficiently construct 3D hyperspectral point clouds of the target crop. The hyperspectral reconstruction method thus leverages the differences in sensitivities (i.e., quantum efficiencies) for a heterogeneous set of on-board cameras to estimate surface reflectance curves, while the spatial reconstruction method uses a novel information gain metric to determine optimal view sequences for reconstruction. Point cloud results from a tabletop scanner are used to perform experimental validation.

In Chapter 4, the single-plant methodology is extended to an entire field of crops. We first define the multi-robot inspection problem associated with evaluating large numbers of crops in a field setting, and detail the challenges associated with timely inspections. We consider scanning tasks associated with each plant, and determine

optimal task selections and multi-robot allocations via kernel estimation and region-based task allocation. Kernel-based allocations reduce the number of scans required to inspect large fields, and therefore allow a smaller number of robots to be used for inspections. The associated path planning problem is also investigated. The planning methods are validated using multi-robot simulation studies based on real farm plots in New Jersey. We then consider cases where additional coarse aerial images are collected using UAVs. We modify the UGV-only planner to also incorporate UAV path and battery constraints. A solution method is then provided to maximize UAV coverage given these combined constraints. Simulation studies are performed to show that this new UGV-UAV planner reduces the time needed to scan an entire field when compared to the state-of-the-art, without increasing the overall travel distance for the robots.

In Chapter 5, we first describe a method to optimize landing trajectories for the UAVs based on on-board predictive controllers. We then extend this method to cooperative landing trajectories for heterogeneous UGV-UAV groups. Simulation studies are presented to verify the optimization method, and to further demonstrate the cooperative landing capabilities of the UGV-UAV system. We also introduce a novel learning-based estimation method for near-surface thrust and torque effects that dominate the aerodynamics of the UAV during landing. Experimental results and simulation studies are presented to verify the estimation method. An integration framework is finally outlined to combine the estimation and trajectory optimization methods.

In Chapter 6, we outline a human-robot interaction framework for human-guided inspections and demonstrate this framework using an experimental platform. We use the on-board vision sensors to determine gesture-based inspection suggestions and safe human following behavior for UGVs. The developments are then combined with results from earlier chapters for intuitive and responsive human-guided robotic crop inspections. We also propose a human-in-the-loop learning model to improve the convergence and specificity of the learning models. Some experiments are presented, but systematic implementation and evaluation are relegated to future work.

The main contributions of this dissertation are therefore the development of a platform and accompanying methods for autonomous UGV-UAV crop inspections. The

details of these contributions are detailed as follows.

1. The robotic crop inspection problem is formalized as a multi-robot next-best-view problem, and a joint spectral-spatial solution method is proposed. The formalization provides a novel information gain (IG) metric that is used to compare the quality of hyperspectral point clouds in relation to ground truth data. Unlike other IG metrics that only consider spatial information, the proposed IG metric includes spectral information and therefore allows the next-best-view method to determine view sequences to optimize the spectral quality of the reconstruction. Simulated results show that the proposed NBV method produces results within 1-3% of the theoretical maximum IG. An auto-calibration method is introduced for the on-board cameras and lights, which speeds up field phenotyping (to around 1% of the time it would take with a hyperspectral camera) and removes artifacts caused by changing lighting conditions. Reconstruction quality is further improved by introducing a specularity correction method which accounts for bright highlights caused by the cameras' viewing angles. A learning-based model is added to the method to counteract basis projection errors, which are a common problem for other spectral decomposition methods. This requires a large dataset for residual estimation, but can in turn reduce errors and allow data collection with a smaller number of cameras (e.g., 4 instead of 10).

2. New planning and task allocation methods for multi-robot inspection tasks are presented. A novel adaptive sampling method is presented for efficient data collection. The method uses similarity kernels learned by training a novel kernel network on historical data, and therefore captures a wider range of spatial correlations than the commonly used (anisotropic) Gaussian kernels. Since the learned kernel also considers temporal correlations, it can be used for early-season predictions as well as temporally-coherent metric estimates throughout the grow season. Simulated results show that the learned kernel provides accurate estimates up to 20 days earlier than the Gaussian kernel. The kernel is also used to select optimal sampling positions for a multi-robot team. The optimization is based on an

NP-hard allocation problem introduced alongside an algorithm to divide the field into geodesic Voronoi regions (i.e., regions that consider the movement of robots among row crops). This provides an efficient and collision-free task allocations for multi-robot crop inspection teams. The methods greatly reduce the number of scans needed to estimate the state of an entire field, and thus increase phenotyping throughput. The method is extended for UGV-UAV teams by considering energy-constrained planning and continuous image acquisitions, which allows the multi-robot team to cover the majority of the field 30% faster than other planners while expending similar energy and traveling similar distances.

3. The control structure for the multi-robot system is developed, and novel methods and algorithms are introduced for UGV-UAV cooperative landing. A method to determine non-blocking waiting poses for UGVs is developed to ensure UGVs do not block critical pathways for other robots or humans performing inspection tasks. The algorithm uses aerial data from the UAV to improve pose selections, and ensures aerodynamically favorable landing positions. Unlike other coordinated landing methods, the proposed approach lets the UGV relocate to avoid undesirable aerodynamics interactions that arise from the position of the landing pad (e.g., too close to a wall). On the UAV side, a predictive trajectory generation method is introduced to optimize landing trajectories that take unmodeled aerodynamic effects (e.g., the ground effect) into consideration. A novel neural network is proposed to learn the near-surface aerodynamic effects, and experiments are performed to show that the prediction errors for this network are within 4% on average for single-rotor effects. Simulation studies further demonstrate that the network predicts complex multirotor-surface interactions.

4. A new human-robot interactions framework is proposed for enhanced crop inspections. Human poses are estimated from camera data to extract pointing gestures, which are then used to send the UGV suggested inspection targets. Target tracking errors observed with an omni-directional robot are on the order of 10 cm for slow finger movements and 50 cm for fast finger movements. The pose estimates

are also used for safe and efficient human-following behavior, which allows the inspection robot to follow an expert for guided inspections. Simulation studies are performed to validate the approach, which show that the mean human-robot separation distance decreases by 3% when using the proposed human-following planner. The proposed framework replaces the screen or tablet interfaces that are traditionally used to control inspection robots and removes the need to learn new software, thus lowering one of the main adoption barriers for precision agriculture systems.

Chapter 2

Cooperative UGV-UAV Inspection Platform

2.1 Introduction

Crop inspections are an important part of managing large scale food production, as they ensure certain metrics are being met at the correct points within the growth season, and that the plants are free of stress and disease. Traditionally, this is done by the growers to inspect the crops manually at certain points in the growing season and making an assessment based on expert knowledge. Unfortunately, this practice also requires knowledge that can only be acquired over long periods of working with a certain crop, tool or site. Manpower requirements for manual inspections also create a severe and expensive throughput bottleneck, and therefore do not scale well for large-scale food production. The desirable solution to this problem is to use robotic platforms to perform the inspections, and to scale the production of these inspection platforms to match the throughput requirements. The trade-off for this switch is that robotic inspection systems are not yet turn-key solutions, and the technology requirements largely overlap with ongoing research, i.e., the technology readiness levels for some parts are severely low. The silver lining is that once a platform is deployed and operational, replicating the capabilities and knowledge of the platform is trivial in comparison to training an employee to become an expert. Roboticizing crop inspections is therefore a critical and natural direction for agricultural engineering research.

Methods to roboticize crop inspections are as varied as the crops themselves, but most methods rely on a set of vision and environmental sensors that are positioned by a robot to interact with nearby crops. Additionally, many robotic inspection methods require multiple robots in order to match the scale of agricultural production. In this

chapter, we propose an experimental UGV-UAV robotic platform for cooperative multi-robot crop inspections and describe the hardware and software components required for such operation. We also detail the system dynamics and on-board software and data architectures which enable large-scale inspections. Using these specifications, an experimental platform is designed as a single-track mobile manipulator and a quadcopter UAV. The discussions in later chapters therefore assume that the UGV-UAV systems deployed into the field match the capabilities of the platform described in this chapter.

The rest of the chapter is organized as follows. The proposed platform is presented in Section 2.2, and details are given for an experimental prototype in Section 2.3. The system dynamics for the experimental prototype are presented in Section 2.4. The software architecture including data and storage models are presented in Section 2.5. Finally, a concluding summary is presented in Section 2.6.

2.2 Proposed Platform

Robotic crop inspections require sensor systems to collect inspection data, manipulation or staging systems to situate and orient the sensor systems, and navigation and control systems to move to the inspection targets. In addition to these systems, on-board computers must provide GPU-accelerated processing for the combined navigation, planning and inspection system. In this section, we outline some of the hardware requirements for such a system, and provide details about the on-board sensors that should be mounted on this hardware. The remainder of the thesis assumes the UGV-UAV inspection platform satisfies these requirements, and an experimental prototype that was designed to these specifications is presented in Section 2.3.

2.2.1 Hardware requirements

The general hardware requirements for the proposed platform consist of a mobile manipulator and the peripheral inspection hardware. The mobile manipulator combines a wheeled ground robot that can position itself at arbitrary positions and orientations within the field, which acts as the base for the entire inspection system. A robot arm

can reach plants on either side of the base, which should include a gripper that acts as a mounting point for a suite of vision sensors. Additionally, for UAV/UGV operation, a dockable charging station should be mounted onto the mobile manipulator (e.g., above the back wheel or axle). The UAV should be chosen as a multirotor to enable hovering missions or single-plant inspections. The entire system should be integrated through industrial PCs and controller. This combined system will thus be capable of cooperative UGV/UAV crop inspection missions.

The hardware requirements for the proposed platform are outlined in Table 2.1

Table 2.1: Hardware requirements for proposed platform

Requirement	Description	Purpose
UGV base	A narrow, wheeled ground robot.	Allows the platform to navigate crop rows and operate on loose soil.
UAV	A multirotor UAV	Allows top-down views of the field. Can be used to inspect crops separately, or to provide the UGV with localization and scouting data.
36/48V battery	A large-capacity, high-voltage battery	Powers the control and vision systems, allows the UAV to recharge.
Manipulator	Robot arm with gripper	Allows the platform to position its vision and environmental sensors, and to manipulate crops/tools.
Navigation sensors	Lidar and base-mounted cameras	Allows the robot to simultaneously map the field and localize itself. Also provides rough phenotyping information.
Sensor suite	Thermal, RGB, and RGB-D cameras	Provides close-up phenotyping data for intensive inspections.
Ground probe	Environmental and soil sensors	Provides a single device to measure environmental and soil conditions. Designed to be planted by the manipulator.
On-board computers	An industrial PC and a GPU device	Allows on-board processing (e.g., planning, phenotyping, arm control). Provides an Ubuntu/ROS based software platform.

2.2.2 On-board sensors

The sensor requirements for the proposed platform should be crop and stress specific. However, in this section, we provide a general purpose sensor deployment that generates the types of data typical of robotic phenotyping. These sensors include imaging sensors such as RGB, RGB-D and thermal cameras, as well as non-imaging sensors such as temperature and humidity sensors. The combination of these sensors provide the platform with the necessary crop inspection data for stress detection, early disease detection, and more.

The sensor requirements for the proposed platform are outlined in Table 2.2.

Table 2.2: Sensor requirements for proposed platform

System	Sensor	Description
Navigation	Lidar	Front/back-mounted laser scanner that generates a point cloud.
	IMU	Inertial measurement unit that provides acceleration/orientation data.
	GPS	Positioning system that provides localization data.
	RGB-D	UGV-mounted color/depth sensors to detect humans, etc.
Vision	RGB	Color cameras used to extract hyperspectral data and to detect objects/stresses
	RGB-D	Arm-mounted color/depth sensor to capture geometries and RGB triplets
	Thermal	Thermal camera used to detect heat stress and related data
Ground probe	Thermocouple	Type-K thermocouple to measure soil temperatures
	UV/HDR	Light intensity sensors
	TPH	Combined environmental sensor for temperature, pressure and humidity
	Moisture	Capacitive soil moisture sensors
Battery	Charge sensor	Sensor to monitor battery charges

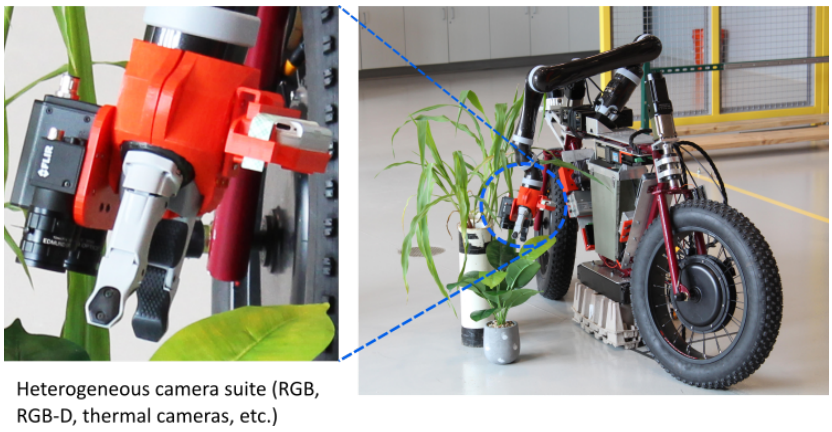


Figure 2.1: The experimental prototype designed by the Rutgers Robotics, Automation and Mechatronics Lab, consisting of a custom bikebot base, a Kinova arm with a KG-2 gripper, and a 3D-printed camera mount that houses an Intel RealSense depth camera, a Point Grey industrial GigE color camera, and a Flir A65 thermal camera.

2.3 Experimental Prototype

An experimental prototype was developed in the Rutgers Robotics, Automation and Mechatronics (RAM) Lab as part of a sponsored research project funded by Siemens Corporate Technology. The prototype's specifications were selected to satisfy the hardware and sensor requirements listed above. The sponsor was shown demonstrations of the manipulation and inspection subsystem at each project milestone.

A single-track bikebot base was chosen for the UGV base and designed and built by the RAM lab. A single-track robot was chosen because the target crops for the sponsored research project were row crops such as corn. A lightweight Jaco2 robot arm from Kinova was used to provide manipulation capabilities to the robot. A 3D-printed camera mount was designed to wrap around the KG-2 gripper that was attached to the Jaco2 arm. Fig. 2.1 shows the full mobile manipulator and camera suite.

A suite of sensors were attached to the camera mount to provide the mobile manipulator with phenotyping capabilities. Although the camera mount provides a brass-plated grid of mounting points, the prototype was designed to house two Point Grey industrial RGB cameras and a FLIR thermal camera on the top side, and an Intel Realsense RGB-D camera on the bottom side. The sides of the gripper were left free

to enable the fingers to open and close. The RGB-D camera was connected to the on-board computer via USB, whereas the remaining cameras were connected via ethernet using the on-board router and switch. Fig 2.2 shows a schematic of the camera mount.

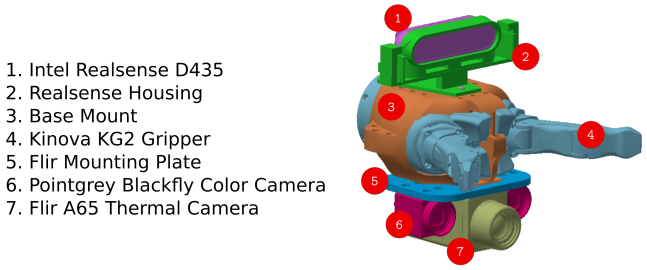


Figure 2.2: 3D-printed camera mount installed onto the KG-2 gripper. Several cameras can be mounted onto the gripper to provide phenotyping capabilities to the mobile manipulator.

A ground probe was also prototyped for use with the Kinova arm and KG-2 gripper. The ground probe housed several environmental sensors to capture temperature, humidity, barometric pressure and UV/HDR light conditions, as well as ground sensors to capture soil moisture and temperature. The probe was designed with an Arduino WiFi board to communicate with the on-board computer via wi-fi through the on-board router. Fig. 2.3 shows a schematic of the ground probe.

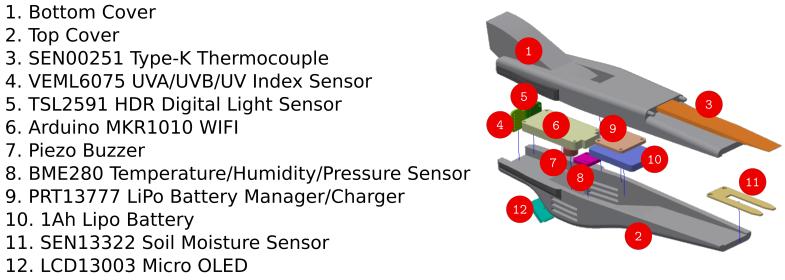


Figure 2.3: 3D-printed ground probe used to collect environmental and ground measurements.

2.4 System Dynamics

The system dynamics for the experimental prototype are useful for many planning and control applications, and form the basis for some of the predictive algorithms described in Chapters 4 and 5. In this section, we provide an overview of the dynamic models for the UGV and the UAV separately. Since the UGV dynamics are not a core part of this thesis, the derivations for any equations of motion are not provided here, and the readers are directed to the references provided within each subsection. However, the dynamics for the UAV are derived in full in this section, since they are a critical part of later chapters.

2.4.1 UGV dynamics

The dynamics of the bikebot UGV used in the experimental prototype are detailed in the work by Gong et al [237]. A quick summary of the results is provided below, but is not considered within the scope of this thesis in general otherwise. A simplified UGV model used for multi-robot simulations and the omni-directional UGV model used for in-lab experiments are provided below the bikebot model for reference and comparison.

Bikebot

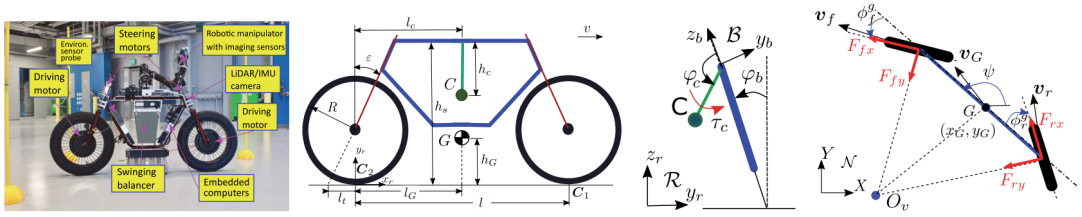


Figure 2.4: (a) Single-track two-wheel steering bikebot developed in the Rutgers RAM Lab. (b) Schematic for the bikebot. (c) Back view of the rolling motion. (d) Top view.

The bikebot is a two-wheel steering, single-track robot with identical rear and front wheels. The system kinematics and dynamics are described in [237] using front and rear contact points C_1 and C_2, respectively, as shown in Fig. 2.4. Three coordinate frames are used in the modeling process: a ground-fixed frame $\mathcal{N}(X, Y, Z)$, horizontal moving frame $\mathcal{R}(x_r, y_r, z_r)$ with origin C_2 and x-axis C_2C_1, and a body-fixed frame

$\mathcal{B}(x_b, y_b, z_b)$.

The rolling motion dynamics are then derived via the Lagrangian method as

$$\boldsymbol{M}_a(\boldsymbol{q})\ddot{\boldsymbol{q}}_a + \boldsymbol{C}_a(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{G}_a(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{B}_a(\boldsymbol{q})\boldsymbol{u}, \tag{2.1}$$

where matrices $\boldsymbol{M}_a$, $\boldsymbol{C}_a$, $\boldsymbol{G}_a$ and $\boldsymbol{B}_a$ are functions of the overall coordinates $\boldsymbol{q} = [\boldsymbol{q}_a^T, \boldsymbol{q}_p^T]^T$ and its derivatives, $\boldsymbol{q}_a$ is the generalized coordinates for rolling motion, $\boldsymbol{q}_p$ is the bikebot pose coordinate, $\boldsymbol{\tau}$ is the torque, and $\boldsymbol{u}$ is the control input. Planar motion and stationary dynamics are derived similarly. An external/internal convertible controller is then designed for tracking and stationary stabilization. For further details, readers are referred to [237].

Three-wheeled omni-directional UGV

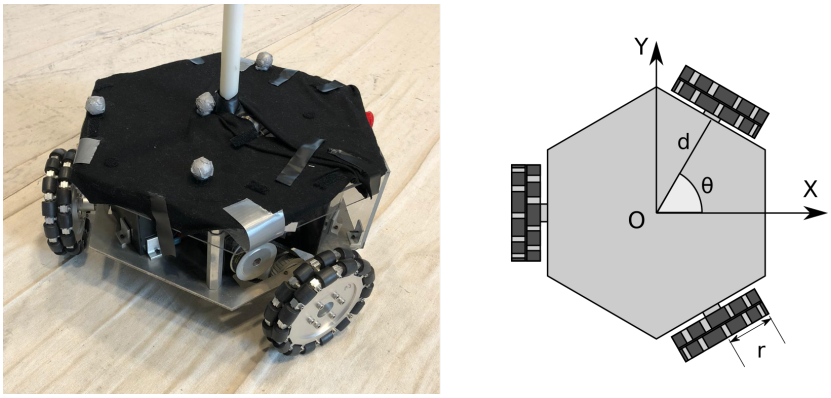


Figure 2.5: A three-wheel omni-directional robot used for in-lab experiments. The robot is fitted with color stereo and RGB-D cameras. Vicon motion tracking cameras are used to validate the experimental result.

Omni-directional UGVs allow greater flexibility in motion, but are unsuitable for field use in agricultural applications. The work in this thesis uses them in lieu of the bikebot for in-lab testing and validation for some of the vision and human-robot interaction algorithms. The robot models and low-level controllers for the omni-directional

UGVs used in this work are adapted from [238] with kinematic model

$$\dot{\boldsymbol{q}} = \bar{\boldsymbol{B}}(\theta)\boldsymbol{u},\ \bar{\boldsymbol{B}}(\theta) = \begin{bmatrix} \boldsymbol{R}_\theta & 0 \\ 0 & 1 \end{bmatrix},\ \boldsymbol{R}_\theta = \begin{bmatrix} c\theta & -s\theta \\ s\theta & c\theta \end{bmatrix}, \tag{2.2}$$

where $\boldsymbol{q} = [x, y, \theta]$ are the generalized robot coordinates, θ is the angle between the world-frame x-axis and the robot's leading wheel, $s\theta = \sin\theta$, $c\theta = \cos\theta$, and inputs $\boldsymbol{u} = [u_1, u_2, u_3]^T$ depend on wheel angular velocities ω_1, ω_2 and ω_3 as $u_1 = \frac{r\sqrt{3}}{3}(\omega_3 - \omega_2)$, $u_2 = -\frac{2r}{3}(\omega_1 - \omega_2 - \omega_3)$, $u_3 = \frac{r}{6r}(\omega_1 + \omega_2 + \omega_3)$. Additionally, $\bar{\boldsymbol{B}}^T(\theta) = \bar{\boldsymbol{B}}^{-1}(\theta)$.

Simulated UGV

The final robot model used in this thesis is the differential drive Clearpath Husky provided with ROS. The simulated robot approximately follows the model described by Lavalle [239],

$$\dot{x} = \frac{r}{2}(u_l + u_r)\cos\theta \tag{2.3}$$

$$\dot{y} = \frac{r}{2}(u_l + u_r)\sin\theta \tag{2.4}$$

$$\dot{\theta} = \frac{r}{L}(u_r - u_l), \tag{2.5}$$

where robot coordinates are given as $\boldsymbol{x} = [x, y, \theta]^T$, L is the wheelbase, u_l is the left wheel's angular velocity, u_r is the right wheel's angular velocity, and r is the radius of the wheel. Tracking controllers are provided by Clearpath as part of the ROS package and emulate the behavior of the physical robot.

2.4.2 UAV dynamics

In this section, we present a model for the multirotor UAV that comprises the second half of the proposed UGV-UAV inspection platform. The dynamics of a multirotor UAV are driven primarily by aerodynamic forces and interactions induced by the movement of each rotor. In this section, we outline a quaternion-based formulation for the rigid body dynamics and the associated aerodynamics of a generalized multirotor.

Rigid Body Dynamics

Let $\mathcal{O}$ be the inertial world frame with unit axes $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}\}$, and let $\mathcal{B}$ be the body frame with unit axes $\{\hat{\boldsymbol{b}}_1, \hat{\boldsymbol{b}}_2, \hat{\boldsymbol{b}}_3\}$. Denote the state of the multirotor in frame $\mathcal{O}$ as $\boldsymbol{x} = [\boldsymbol{\xi}^T, \boldsymbol{q}^T, \boldsymbol{v}^T]^T$, with position vector $\boldsymbol{\xi} \in \mathbb{R}^3$, velocity vector $\boldsymbol{v} \in \mathbb{R}^3$, and orientation quaternion $\boldsymbol{q} \in SU(2)$, where $SU(2)$ is the special unitary group of degree 2. The world frame state derivative is thus denoted as $\dot{\boldsymbol{x}} = [\dot{\boldsymbol{\xi}}^T, \dot{\boldsymbol{q}}^T, \dot{\boldsymbol{v}}^T]^T$, with linear velocity vector $\dot{\boldsymbol{\xi}}$, linear acceleration vector $\dot{\boldsymbol{v}}$, and quaternion derivative $\dot{\boldsymbol{q}}$ tied to body frame angular velocity vector $\boldsymbol{w}$ via rigid body dynamics

$$\dot{\boldsymbol{\xi}} = \boldsymbol{v}, \quad \dot{\boldsymbol{q}} = -\frac{1}{2}\begin{bmatrix} 0 \\ \boldsymbol{w} \end{bmatrix} \otimes \boldsymbol{q}, \quad \dot{\boldsymbol{v}} = \boldsymbol{q} \otimes \boldsymbol{F} \otimes \boldsymbol{q}^* - mg\hat{\boldsymbol{z}}, \tag{2.6}$$

where $\boldsymbol{F}$ is the body frame vector sum of all body and surface forces except gravity, $-g\hat{\boldsymbol{z}}$ is the gravity vector, $\otimes$ is quaternion-vector multiplication using skew-symmetric matrices (see [240]), and $\boldsymbol{q}^*$ is the conjugate of $\boldsymbol{q}$. The time derivative of $\boldsymbol{w}$ is defined via body frame torques $\boldsymbol{\tau} = [\tau_1, \tau_2, \tau_3]^T$ as

$$\dot{\boldsymbol{w}} = \boldsymbol{I}_o^{-1}\boldsymbol{\tau} - \boldsymbol{I}_o^{-1}[\boldsymbol{w} \times (\boldsymbol{I}_o\boldsymbol{w})], \tag{2.7}$$

where $\boldsymbol{I}_o$ is the 3×3 inertia matrix. More detailed derivations of the rigid body dynamics in (2.6)–(2.7) can be found in [241, 242]. [240] includes a review of quaternion math.

Aerodynamics

Total body-frame forces $\boldsymbol{F}$ and torques $\boldsymbol{\tau}$ are dominated by rotor-induced aerodynamic forces such as rotor thrusts and torques, as well as external aerodynamic forces such as winds and gusts. Accordingly, $\boldsymbol{F}$ and $\boldsymbol{\tau}$ can be further broken down as

$$\boldsymbol{F} = \boldsymbol{T}_{total} + \boldsymbol{F}_{ext} + \boldsymbol{\delta}_F, \tag{2.8}$$

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{total} + \boldsymbol{\tau}_{ext} + \boldsymbol{\delta}_\tau, \tag{2.9}$$

where $\boldsymbol{T}_{total}$ and $\boldsymbol{\tau}_{total}$ are the total rotor thrusts and torques, $\boldsymbol{F}_{ext}$ and $\boldsymbol{\tau}_{ext}$ are unmodeled or external forces and torques, and $\boldsymbol{\delta}_F \sim \mathcal{N}(0, \sigma_F^2 \boldsymbol{I})$ and $\boldsymbol{\delta}_\tau \sim \mathcal{N}(0, \sigma_\tau^2 \boldsymbol{I})$ are noise terms with variances σ_F^2 and σ_τ^2, respectively.

$\boldsymbol{T}_{total}$ and $\boldsymbol{\tau}_{total}$ for a hovering multirotor can be calculated based on the multirotor's configuration. In this thesis, we assume the rotors are evenly distributed about $\hat{\boldsymbol{b}}_3$, i.e., for a multirotor with n rotors, rotor i is placed at body-frame position

$$^{\mathcal{B}}\boldsymbol{r}_i = R[\cos\alpha_i, \sin\alpha_i, 0]^T,\ \alpha_i = \frac{2\pi}{n} - \frac{\pi}{8},\ i = 1, ..., n, \tag{2.10}$$

where R is the distance from the center of the body frame to the center of the rotor.

Per the blade element momentum (BEM) theory, the thrust and reaction torque produced by a free-standing rotor is given by the lumped-parameter models (see [241] for details), namely,

$$T_i = c_T \omega_i^2,\ \text{and}\ \tau_i = S_i c_\tau \omega_i^2,\ S_i = (-1)^i \tag{2.11}$$

respectively, where ω_i is the rotor angular velocity, c_T is thrust coefficient, c_τ is the reaction torque coefficient, and S_i denotes the direction of rotation for rotor i. The corresponding total body frame thrust is

$$\boldsymbol{T}_{total} = \sum_i T_i \hat{\boldsymbol{b}}_3 = c_T \sum_i \omega_i^2 \hat{\boldsymbol{b}}_3, \tag{2.12}$$

and the body frame torques are

$$\boldsymbol{\tau}_{total} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} \sum_i (\hat{\boldsymbol{b}}_2 \cdot {}^{\mathcal{B}}\boldsymbol{r}_i) c_T \omega_i^2 \\ \sum_i (-\hat{\boldsymbol{b}}_1 \cdot {}^{\mathcal{B}}\boldsymbol{r}_i) c_T \omega_i^2 \\ \sum_i S_i c_\tau \omega_i^2 \end{bmatrix}. \tag{2.13}$$

Non-hover aerodynamics are more complex and depend on the rigid body dynamics of the multirotor. In particular, near-surface aerodynamic effects depend on the pose and velocity of the multirotor, in addition to several other environmental parameters.

2.5 Software Architecture

The system presented in this thesis consists of many complex components that need to operate together autonomously and continuously. The software required to handle the communication between these components and the storage models needed to store the large amount of data produced by the components is similarly complex. In the remainder of this section, we provide an overview of the software architecture that enables autonomous robotic inspections. We further include the communication and storage models and component-level details for a proof-of-concept implementation of this architecture based on the Robotic Operating System (ROS) [243].

2.5.1 Overview of software architecture

First, we describe the overall architecture that would be necessary for a fully-integrated robotic inspection solution. The goal of such an architecture is to organize the dataflow between the respective system components and to create a streamlined management toolchain for the end users, which is both an information technologies challenge and an operational technologies challenge, in addition to the inherent robotics and controls challenges. Fig. 2.6 shows the system architecture for an industrial solution to these challenges. The overall architecture includes components to store and manage the large amounts of data generated by the robots, as well as components that manage robot planning and control, all of which must run in parallel and exchange data in real time.

In this thesis, we build a proof-of-concept implementation of the proposed architecture using open-source components provided by the ROS community. The individual components of the architecture are implemented as ROS nodes that communicate through a common (and distributed) network that uses ROS middleware to transport large amounts of data efficiently through a topic-based pub/sub model. Fig. 2.7 shows ROS architecture corresponding to the algorithms and methods described in the remainder of the thesis. We follow the ROS conventions of transporting data through topics, executing short-lived and on-demand calculations through services, and executing long-lived or time-intensive calculations through actions.

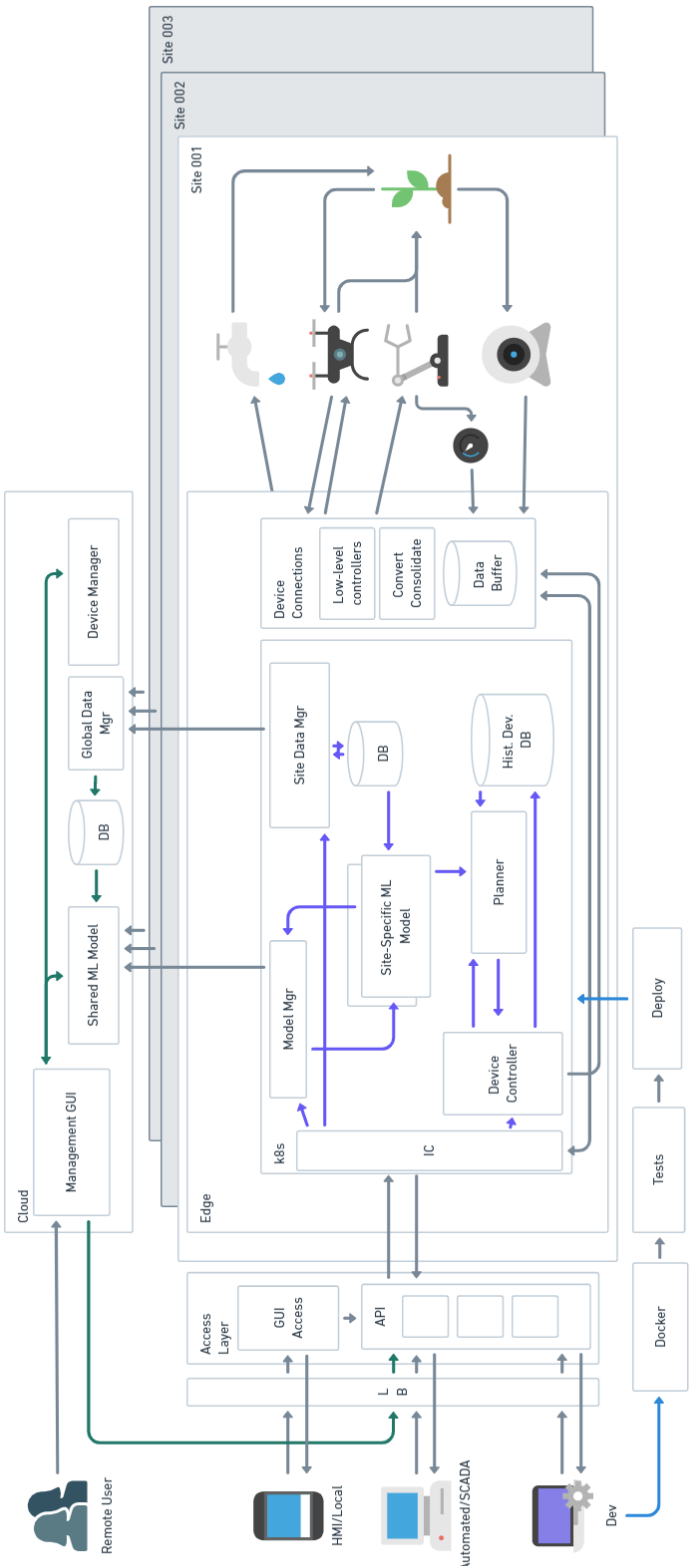


Figure 2.6: System architecture for a distributed multi-robot inspection system. The business logic for the system is handled by an on-premise industrial PC, while low-level controls and data collection is handled by on-board computers installed onto each robot.

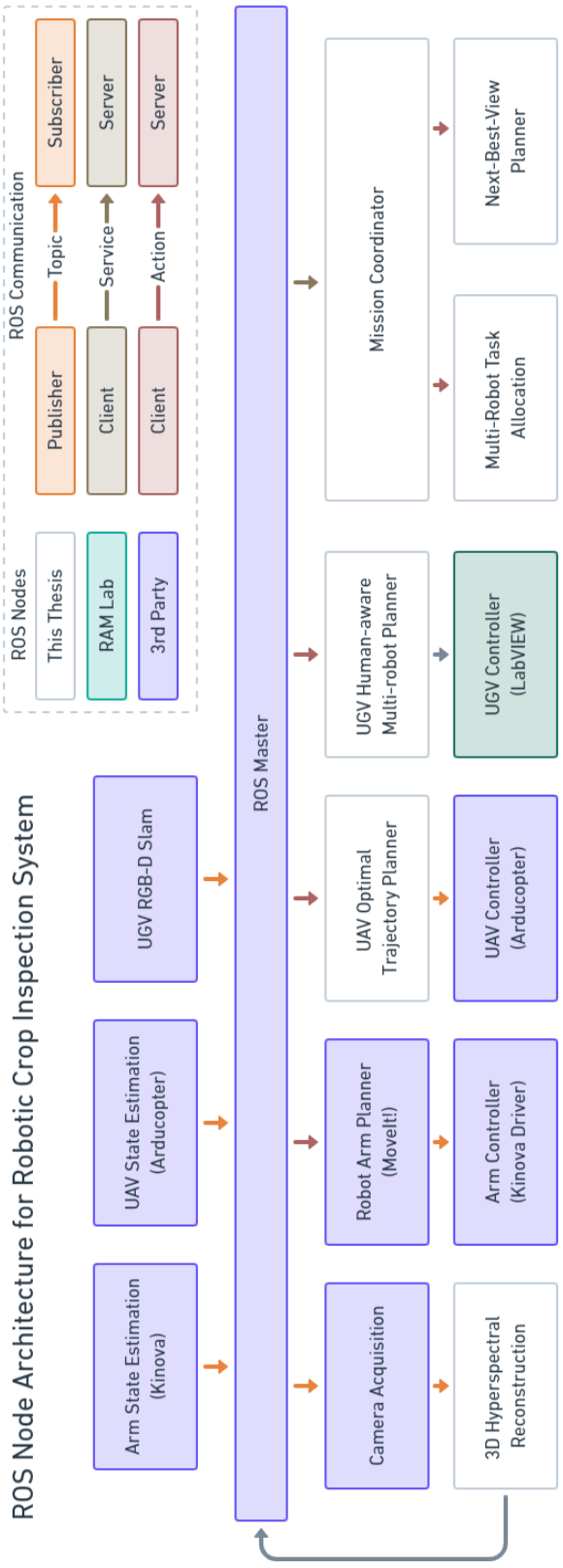


Figure 2.7: Node diagram for the ROS implementation of the proposed software architecture.

The components used in our proof-of-concept implementation are oftentimes developed by university researchers who do not plan on productizing the packages that make up these components. The implementation detailed in this section is therefore only meant to demonstrate the feasibility of the overall architecture. We note that an industrial implementation of this architecture would likely utilize different communication protocols and non-ROS components.

2.5.2 Communication and storage models

Inter-module communications are handled exclusively through ROS. This simplifies inter-module communications, as ROS includes a publisher-subscriber data transport model based on TCP/IP or UDP. ROS also provides straight-forward settings and tools for inter-device communications through its ROS-master architecture, where one device can act as the gateway between several devices on the same network. In this way, the ground probe can be connected directly with the GPU-accelerated on-board computer and the controllers. This also makes inter-robot communications simple, and provides an effective method of recording all data on a single ground control station.

2.5.3 Component-level details

Each component is designed as a ROS node with accompanying topics, services and internal data structures. The mapping and navigation modules can therefore operate effectively independently from the vision and reconstruction modules, since each node acts as a separate program with potentially multiple threads. Another benefit of this structure is that certain GPU and memory heavy modules can be put to sleep to conserve battery power or awoken to perform on-demand calculations. The entirety of this program flow can be controlled through ROS. A component-by-component desription for the nodes developed for the work presented in this thesis is provided below.

Mission Coordinator

The Mission Coordinator node keeps track of each robots mission, scan target and target pose. It creates mission plans by querying the various planner and allocation

nodes.

UAV Optimal Trajectory

The UAV optimal trajectory node uses a model predictive control based planner to generate desired UAV trajectories based on target poses provided by the Mission Coordinator node.

UGV Human-Aware Multi-Robot Planner

The UGV Human-Aware Multi-Robot Planner node uses a multi-robot planner to generate desired UGV trajectories based on target poses provided by the Mission Coordinator node.

Multi-Robot Task Allocation

The Multi-Robot Task Allocation node provides an allocation service that queries the current state of each robot and generates an optimal task allocation for the robot group as a whole.

3D Hyperspectral Reconstruction

The 3D Hyperspectral Reconstruction node stores and processes camera images and generates 3D hyperspectral point cloud reconstructions of the scene from these images.

Next-Best-View Planner

The Next-Best-View Planner node provides an optimal view selection service that queries the current state of a single robot, the state of its scan target, and the current reconstruction state for the target, and generates an optimal next-best-view for the robot.

2.6 Conclusion

This chapter presented a cooperative UGV-UAV platform for robotic crop inspection missions. The hardware requirements for such a platform were identified as an autonomous UGV that carries a large battery, an IPC, a light-weight robot arm, and a vision system mounted onto the end effector of the robot arm, and a UAV that carries downward-facing cameras for imaging, localization and coordination. An experimental platform matching these specifications was developed, and the system dynamics were derived for normal operation. The software architecture for a multi-robot, distributed inspection system was described both in terms of communication and storage models, and in terms of component-level details. A ROS-based framework was proposed to make subsystem integration easier.

The overall goal of this chapter was to develop a platform suitable for multi-robot crop inspections. The details presented in this chapter are therefore the basis for the remaining developments in this thesis, and each chapter deals with a different challenge related to the cooperative inspection platform. For example, the methods used to process the real-time data generated by the arm-mounted cameras are discussed further in Chapter 3. The proposed platform also combines the aerial capabilities of a multi-rotor UAV and the versatility and carrying capacity of a UGV. The associated cooperative planning and control problems are discussed in Chapters 4 and 5, respectively. The design of the UGV and the vision system also offer operational flexibility. We investigate this flexibility in Chapter 6 in the context of human-robot interactions and expert-guided inspections. All of these developments presuppose the UGV-UAV platform presented in this chapter.

Chapter 3

Robotic Crop Evaluations using 3D Hyperspectral Scans

3.1 Introduction

Robotic crop evaluations are central to PA methodology. Mobile robots equipped with vision sensors are used extensively for high-throughput, close-up crop inspections, and modern data management techniques have been developed to store and use the data generated by these robots for continuous and site-specific crop evaluations. In particular, modern phenotyping systems use hyperspectral imaging to detect plant stresses and diseases and 3D reconstruction methods to perform traditional phenotyping (e.g., determination of the plant's height). However, state-of-the-art robotic crop inspection methods treat hyperspectral imaging and 3D reconstruction as separate workflows when in fact they are two facets of the same workflow, i.e. the generation of a 3D model from spectral data captured by the on-board cameras. Furthermore, the morphology of a plant informs the spectral analysis and vice versa; it is impossible to separate the two facets when surface generation relies on how light is reflected off of the plant's surface and the determination of the reflective properties of the plant relies on knowledge of its surface geometry. This is a fundamental gap in the state-of-the-art for crop inspections, and limits how crop data is collected in today's systems. The next generation of robotic crop evaluation systems must therefore stand on two pillars: 1) an efficient scanning method to generate accurate phenotyping data, and 2) a robust evaluation method to complete the evaluation workflow. In this chapter, we introduce novel reconstruction methods and algorithms to perform 3D hyperspectral scans of single plants, and we further demonstrate the versatility of 3D hyperspectral reconstructions for downstream phenotyping tasks such as classification and metric calculations (e.g., canopy coverage). While these developments do not constitute a product-grade robotic crop

evaluation solution, they do form a cohesive and end-to-end crop evaulation pipeline and demonstrate the viability of the overall approach.

The main contributions of this chapter are a new field auto-calibration method using spectral decomposition, a novel multi-view hyperspectral reconstruction technique and information gain metric, a novel next-best-view selection algorithm, and a classification model based on latent space spectral representations. The contributions are presented in this order because each development informs subsequent developments, e.g., the reconstruction technique and information gain metric are used to develop the next-best-view algorithm, and so on. The field auto-calibration method addresses the need for an automated camera and light calibration method, e.g., due to changing outdoor lighting conditions, which is a shortcoming in most long-term field imaging methods. This shortcoming is typically addressed by adding strong artificial lights to the capture system or by controlling the scan environment through light boxes or shelters. In contrast, the calibration method developed here allows the robot to take images under arbitrary lighting conditions and removes the need for these additional steps. The 3D hyperspectral reconstruction method then uses the calibrated camera and light data to extract spatial and spectral information from multiple camera views. Unlike other multi-view reconstruction methods, the method presented in this chapter considers the effects of wavelength-based reflectances, and greatly reduces later computational burden by converting raw spectral curves to basis coefficients via spectral decomposition. Reconstructions created using this method also capture statistical information about the spectral reconstruction and are used to derive metrics such as the reconstruction confidence. Like all multi-view reconstruction methods, the reconstruction method presented in this chapter also requires the robot to take images from multiple views (i.e., end effector poses), so a next-best-view method is developed to determine these views. An approximate next-best-view sequence is found using a small subset of all possible views, and a method to incrementally improve the sequence through local refinement is proposed. The refined sequence is then used to reconstruct the plant. Finally, the reconstruction results are used to determine plant health using a deep neural network.

The rest of the chapter is organized as follows. The imaging model is detailed in Section 3.2. A novel spectral calibration method is described in Section 3.3. The inspection problem is then formalized in Section 3.4 using a probabilistic description of the surface shape and reflectance for the target crop. A 3D hyperspectral multiview reconstruction method is then proposed in Section 3.5, and a next-best-view method is proposed in Section 3.6 to improve the acquisition speed and quality for these reconstructions. A hyperspectral classification method is presented in Section 3.7. Experimental and simulation results are presented in Section 3.8. Finally, a concluding summary is presented in Section 3.9.

3.2 Multi-Camera Hyperspectral Imaging Model

Digital cameras capture images by converting incoming electromagnetic (EM) radiation into electrical signals. These signals are generated when photons hit the sensor elements, and the strength of the signal is dependent on both the frequency of the photon, which corresponds to a single wavelength in the EM spectrum, and the quantum efficiency of the sensor element, which measures the ratio of incident photons that are converted to photoelectrons for each wavelength. Since quantum efficiencies are wavelength-dependent, imaging sensors will produce different signals depending on both the photon composition of incoming light, and the types of sensor elements that were used to build the sensor. Sensors built with different sensor elements (e.g., elements sensitive to visible light versus elements sensitive to infrared light) will therefore generate images with different pixel intensities for the same scene. Arrays of sensor elements of different types are used to capture multiple regions (i.e., bands) of the electromagnetic spectrum, thus creating images with multiple channels (e.g., red, green and blue channels corresponding to the red, green and blue regions of the visible spectrum). The sensitivity curve of each channel is thus proportional to the quantum efficiencies for the corresponding sensor elements. The goal of hyperspectral imaging is to capture a separate image channel every 1–4 nm in the visible and near-infrared regions of the EM spectrum.

In this section, we describe the underlying model that enables hyperspectral imaging

using color cameras. First, a lighting model is outlined in which the photon distribution of the light is described as a wavelength-dependent curve. Second, a reflection model is introduced to capture how light reflects off of common plant surfaces such as leaves. Third, a camera model is selected to describe how the light coming from the plant is converted to an image. Lastly, the lighting, reflection and camera models are combined into a single multi-camera hyperspectral imaging model.

3.2.1 Lighting Model

Light is electromagnetic radiation with wavelengths in the visible part of the EM spectrum (VIS) corresponding to roughly 400–750 nm. Hyperspectral imaging additionally includes lower frequency near-infrared (NIR) radiation corresponding to roughly 750–1500 nm, or higher frequency ultraviolet (UV) radiation corresponding to roughly 200–400 nm. The vast majority of agricultural hyperspectral sensors capture data in the 400–1000 nm range (VNIR or VisNIR).

These wavelengths correspond to the energy of the photons emitted by the light source through the photon energy equation, $E = \hbar c \lambda^{-1}$, where $\hbar$ is Planck's constant, c is the speed of light in a vacuum, and λ is the wavelength of the photon. The energy ranges of the emitted photons are a direct result of the light source itself —light generated via stimulated emissions (i.e., lasers) are typically composed of photons that have a specific wavelength, whereas lights generated via complex chemical and physical processes (e.g., solar light) are typically composed of photons that correspond to a broader spectrum. Since pure reflections (i.e., elastic scattering without luminescence or iridescence) do not change the wavelength of reflected photons, the spectral signature of the light (i.e., distribution of emitted photons across all wavelengths) will have a direct effect on hyperspectral imaging results. That is, if the light source does not generate much light in the 400–500 nm range, it will be difficult to analyze the plant's reflective properties in that range since there are fewer photons being emitted with those wavelengths. It is therefore critical to determine the distribution of wavelengths for the light source.

In hyperspectral imaging, the concentration of photons emitted from the light source

(i.e., radiant flux) as a function of wavelength is called the spectral power distribution (SPD), which is defined as

$$s(\lambda) = \frac{\partial^2 \Psi}{\partial A_\Psi \partial \lambda}, \tag{3.1}$$

where Ψ is the radiant flux, A_Ψ is the area of integration, and λ is the wavelength, and is used to describe both the luminance (i.e., brightness) and chromacity (i.e., color) of the light source. Light that is incident on a plant's surface is similarly described using SPDs, and different light sources produce different hyperspectral images unless the images are corrected to remove the luminance and chromacity effects of the light source (e.g., via calibration). An ideal light source for hyperspectral imaging will therefore have a high-energy, broadband SPD, thus ensuring high signal to noise ratios for all wavelengths.

3.2.2 Reflectance Model

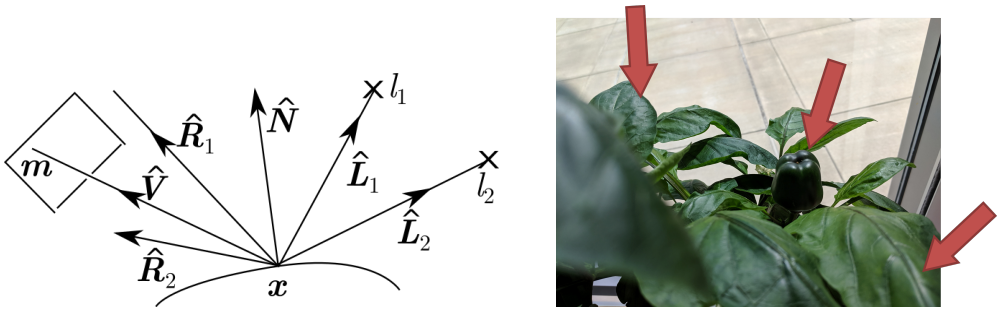
The reflectance of a surface describes the ratio of incident light that gets reflected off of the surface rather than getting absorbed or transmitted by the surface, i.e.,

$$R = \frac{\Phi_e^r}{\Phi_e^i}, \tag{3.2}$$

where R is the reflectance, Φ_e^i is the radiant flux that is incident on the surface, and Φ_e^r is the radiant flux that is reflected off of the surface. For opaque surfaces, it is observed that reflectances depend on the angles of the incoming and reflected rays and the SPD of the light source. The function that captures this relationship is called the bidirectional reflectance distribution function (BRDF), which replaces (3.2) as

$$R = f_r(\boldsymbol{\omega}_e^i, \boldsymbol{\omega}_e^r, \boldsymbol{x}, \lambda), \tag{3.3}$$

where $\boldsymbol{\omega}_e^i \in \mathbb{R}^2$ is the direction of incoming light, $\boldsymbol{\omega}_e^r \in \mathbb{R}^2$ is the direction of reflected light, $\boldsymbol{x} \in \mathbb{R}^2$ is the point of reflection on the plant's surface, and $\lambda \in \mathbb{R}$ is the wavelength of the incoming light. We assume that $\boldsymbol{x}$ sits on a two-dimensional manifold $\mathcal{S}$ defined



(a) (b)

Figure 3.1: (a) The Phong reflection model. Direction vectors $\hat{\boldsymbol{L}}$ and $\hat{\boldsymbol{V}}$ are dependent on the position of the light and the viewer, respectively, while $\hat{\boldsymbol{N}}$ depends on the point of reflection. (b) Example of specular reflections on a pepper plant, where the plant surface looks whiter due to direct reflections.

by the object's surface, i.e. $\boldsymbol{x} \in \mathcal{S} \subset \mathbb{R}^2$, and $\boldsymbol{\omega}_e^i$ and $\boldsymbol{\omega}_e^r$ are described in a coordinate system defined by the surface normal, tangent and cotangent vectors at $\boldsymbol{x}$, so R is described fully by seven variables (two azimuth angles, two zenith angles, two manifold coordinates, and a wavelength). In the case of plants with waxy leaves, the BRDF is phenomenologically described using a Phong reflection model [244], which prescribes three reflectance components to the total illumination: the ambient, which describes reflections due to light that is identically incident on all points along $\mathcal{S}$, the diffuse, which is reflected identically for all $\boldsymbol{\omega}_e^r$, and the specular, which captures angle-dependent highlights that are most intense when $\boldsymbol{\omega}_e^r$ is equal to the reflection of $\boldsymbol{\omega}_e^i$ about the surface normal at $\boldsymbol{x}$. Under the Phong reflection model, the illumination reflected from $\boldsymbol{x}$ toward a viewer (or sensor) at $\boldsymbol{v}_c$ is given as

$$I_p(\boldsymbol{x}, \boldsymbol{v}_c, \lambda) = k_a(\lambda) i_a(\lambda) + \sum_{j=1}^{N_l} \left[k_d(\lambda)(\hat{\boldsymbol{L}}_j \cdot \hat{\boldsymbol{N}}) i_{j,d}(\lambda) + k_s(\lambda)(\hat{\boldsymbol{R}}_j \cdot \hat{\boldsymbol{V}})^{\alpha} i_{j,s}(\lambda) \right], \quad (3.4)$$

where $k_a(\lambda)$ is the ambient reflection coefficient, $k_d(\lambda)$ is the diffuse reflection coefficient, $k_s(\lambda)$ is the specular reflection coefficient, $i_a(\lambda)$ is the incident ambient illumination, $i_{j,d}(\lambda)$ is the incident diffuse illumination from light j, $i_{j,s}(\lambda)$ is the incident specular illumination from light j, α is the specularity constant, $\hat{\boldsymbol{L}}_j$ is the direction vector from $\boldsymbol{x}$ to light j, $\hat{\boldsymbol{N}}$ is the surface normal at $\boldsymbol{x}$, $\hat{\boldsymbol{R}}_j$ is the reflection of $\hat{\boldsymbol{L}}$ about $\hat{\boldsymbol{N}}$, and $\hat{\boldsymbol{V}}$ is the direction vector from $\boldsymbol{x}$ to the viewer (or sensor). Fig. 3.1 shows a diagram of

the Phong illumination model and an example of specular highlights captured by the model.

If most of the illumination comes from a single light source and if specular reflectances are constant for all wavelengths, the BRDF defined in (3.3) is simplified using the Phong reflection model in (3.4) as

$$R(\boldsymbol{x}, \boldsymbol{v}_c, \lambda) = k_d(\lambda)(\hat{\boldsymbol{L}} \cdot \hat{\boldsymbol{N}}) + k_s(\hat{\boldsymbol{R}} \cdot \hat{\boldsymbol{V}})^{\alpha}, \tag{3.5}$$

such that $I_p(\lambda) = k_a i_a(\lambda) + R(\boldsymbol{x}, \boldsymbol{v}_c, \lambda)s(\lambda)$, where $s(\lambda)$ is the SPD.

3.2.3 Camera Model

Consider a system that contains a set of cameras $\boldsymbol{C} = \{C_1, C_2, \ldots, C_{N_c}\}$, where C_i, $i = 1, \ldots, N_c$, are single-channel 2D cameras. For cameras that capture more than one channel, such as RGB cameras which capture three separate bands in the red, green and blue parts of the visual spectrum, each channel can be represented by a co-located single-band virtual camera. We denote the known coordinate transformations between the base frame $\mathcal{B}$ and the sensor frames $\mathcal{C}_i$, $i = 1, \ldots, N_c$, that are attached to C_i at any time as ${}^{\mathcal{C}_i}\boldsymbol{T}_{\mathcal{B}}$. Virtual cameras that correspond to a multi-channel camera have identical coordinate transformations. Fig. 3.2 illustrates the configuration of a typical inspection scene.

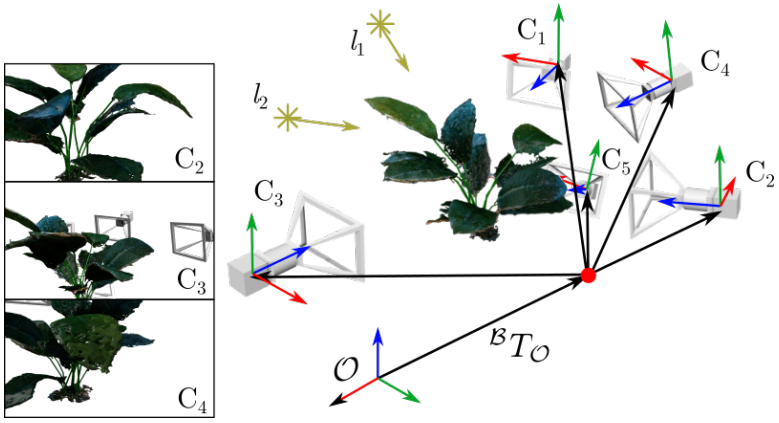


Figure 3.2: A typical scene for the NBV planning task. Cameras are fixed to a mounting system and pointed toward the object, capturing the images on the left. Illumination is provided by lights l_1 and l_2.

We use the pinhole camera model [245], where the projection of a 3D point in the scene onto the 2D image is obtained as $\tilde{\boldsymbol{m}} = \boldsymbol{P}\tilde{\boldsymbol{x}}$, with $\boldsymbol{P}$ calculated from the intrinsic and extrinsic camera matrices, and where $\tilde{\boldsymbol{m}} = [\boldsymbol{m}^T\, 1]^T = [u\, v\, 1]^T$ is the homogeneous image coordinates corresponding to the homogeneous scene coordinates $\tilde{\boldsymbol{x}} = [\boldsymbol{x}^T\, 1]^T = [x\, y\, z\, 1]^T$. The cameras are assumed to have zero axis skew and are calibrated so that their intrinsic and extrinsic matrices are known. Other models [246, 247] can also be used, as long as there exists a function $\boldsymbol{P}$ that is a bijective mapping.

We modify the model in [248] to describe the image captured by each camera. Particularly, we assume a Phong surface as shown in Fig. 3.1 to capture the sheen on plant leaves. The intensity captured by camera C_i at $\boldsymbol{m}$ is

$$p_i(\boldsymbol{m}) = \int_{\mathbb{L}_i} c_i(\lambda) \sum_j I_p(\boldsymbol{x}_j, \boldsymbol{m}, \lambda)\, \mathrm{d}\lambda, \tag{3.6}$$

where $\mathbb{L}_i$ is the support of C_i's spectral sensitivity curve $c_i(\lambda)$, and $I_p(\lambda, \boldsymbol{x}_j)$ is the Phong illumination from light source l_j reflected off the object at $\boldsymbol{x}_j$ towards $\boldsymbol{m}$. Illumination from self-emittance (i.e., thermal radiation) is approximately modeled using the diffuse term. Due to the aforementioned bijective mapping assumption, at most one non-trivial $(\boldsymbol{x}_j, C_i)$ pair exists for each $\boldsymbol{m}$ and therefore, $\boldsymbol{x}_j = \boldsymbol{x}$ for any j.

We define a view $\boldsymbol{v}$ for a group of heterogeneous sensors using the pose of the base frame $\mathcal{B}$ with respect to the world frame $\mathcal{O}$ as $\boldsymbol{v} = ({}^{\mathcal{B}}\boldsymbol{T}_{\mathcal{O}}, \boldsymbol{K}_2, \ldots, \boldsymbol{K}_{N_c})$, where $\boldsymbol{K}_i$ is the intrinsic matrix for camera C_i. We define a point $\boldsymbol{x}$ to be within a camera's field of view if its projection $\boldsymbol{m}$ onto the image plane falls within the bounds of the camera's sensor. For the pinhole camera model used here, this is equivalent to $\boldsymbol{x}$ being inside a viewing frustum defined by a pyramid that projects outward from the camera. We further define a point $\boldsymbol{x}$ on a surface to be occluded with respect to camera C_i if a ray cast from C_i towards $\boldsymbol{x}$ intersects any other surfaces before reaching $\boldsymbol{x}$, and say that $\boldsymbol{x}$ is un-occluded otherwise. For a point $\boldsymbol{x}$ to be visible to C_i, it must be both within C_i's field of view, and be un-occluded with respect to C_i. Note that we ignore illumination when defining visibility here.

Figure 3.3: (a) Field setup for a hyperspectral camera. (b) Color checker calibration tile used to determine camera sensitivity and light spectral power distribution curves.

3.3 Camera Spectral Calibration Method

Camera calibration is a critical part of the spectral reconstruction process. This section first presents a review of the typical calibration workflow for outdoor hyperspectral imaging using a spectroradiometer or hyperspectral camera as shown in Fig. 3.3a, in which calibrated white tiles are used to determine light SPDs before each series of images. The primary development of this section is a novel method to determine camera sensitivity curves and light SPDs from RGB images of a reference color chart as shown in Fig. 3.3b, which is a crucial development for fast, automated field calibration of arbitrary color cameras [249].

3.3.1 Ground Truth Curve Calculation

SPDs are typically measured in a controlled settings using spectroradiometers, which measure radiant flux as a function of wavelength λ. Spectroradiometers are calibrated devices that take the connected optics into account when calculating spectral curves. Given a calibrated light source (i.e., with a known SPD), camera sensitivities can be calculated by placing a monochromator between the calibrated light source and the target camera. The sensitivity curve is then determined by sweeping through the wavelength range of the monochromator and recording sensor responses (i.e., image intensities).

Given a calibrated hyperspectral camera (i.e., a camera with known $c(\lambda)$) that

produces irradiances $I_{HSI}(\lambda) = c(\lambda)s(\lambda)r(\lambda)$, reflectances are calculated as

$$r(\lambda) = \frac{r_0(\lambda)I_{HSI}(\lambda)}{I_0(\lambda)}, \tag{3.7}$$

where $I_0(\lambda)$ is the irradiance for a 95% reflectance calibration tile as recorded by the camera. Thus, the 95% reflectance calibration tile will yield $r(\lambda) = r_0(\lambda) = 0.95$ if scanned again after initial calibration, since $I(\lambda) = I_0(\lambda)$. The ground truth SPD $l'(\lambda)$ and camera sensitivity $c'(\lambda)$ are then calculated per wavelength using the initial calibration $(r_0(\lambda), I_0(\lambda))$ and new reflectance measurement $r'(\lambda)$. This process is the same for field calibration, but unlike laboratory conditions, light SPDs in the field are dependent on higly-variable solar fluxes and cloud covers throughout the day. As a result, the calibration procedure must be repeated at certain intervals to ensure that hyperspectral data collected throughout the day are comparable.

3.3.2 Field Auto-Calibration via Spectral Decomposition

Light SPDs and camera sensitivities can also be calibrated in the field using raw red, green, blue triplets captured by uncalibrated color cameras [249]. It is a well-known fact of spectral theory that any given function f in a Hilbert space $\mathcal{H}$ can be represented exactly as a projection onto an orthonormal basis $b_1, ..., b_{N_b}$ plus a residual R, i.e. $f = \boldsymbol{k}^T\boldsymbol{b}+R$, where $\boldsymbol{b} = [b_1, ..., b_{N_b}]^T$ is a column vector of basis functions and $\boldsymbol{k} = [k_1, .., k_{N_b}]$ is the basis coefficient vector. Since the span of $\boldsymbol{b}$ forms a subspace $\mathcal{G}$ of $\mathcal{H}$, the projection $\hat{f} = \boldsymbol{k}^T\boldsymbol{b}$ is unique and R lies in an orthogonal space $\mathcal{G}^{\perp}$. This concept extends to function spaces so that $f : \Lambda \to \mathbb{R}$ is represented exactly as $f(\lambda) = \boldsymbol{k}^T\boldsymbol{b}(\lambda) + R(\lambda)$, and can be used to describe SPDs, sensitivities and reflectances as basis projections.

We first assume that all variables are dependent on wavelength λ, and we define a tensor of reflectance coefficients $(\mathbf{k}_{\nu,q})(\lambda)$, where $q = a, d, s$ represents the ambient, diffuse and specular components, respectively. For each combination of lights Λ, a vector of spectral power distribution (SPD) functions $(\mathbf{s}_l)_\Lambda$ is constructed. The tensor

of intensities incident at the camera is therefore

$$(\mathbf{I}_{v,c,\nu})_\Lambda(\lambda) = (\mathbf{r}_{v,c,\nu,l})(\lambda)\bar{\times}_4(\mathbf{s}_l)_\Lambda(\lambda), \tag{3.8}$$

where $\bar{\times}_n$ is the mode-n tensor product with a vector. The reflectance tensor $(\mathbf{r}_{v,c,\nu,l})(\lambda)$ is defined element-wise as

$$(\mathbf{r})_{v,c,\nu,l}(\lambda) = (\upsilon_q)_{v,c,\nu,l} \cdot (\mathbf{k}_q)_\nu(\lambda), \tag{3.9}$$

where the view coefficients are $(\upsilon)_{v,c,\nu,l,q} = 1$ if $q = a$; $\hat{L}_{\nu,l} \cdot \hat{N}_\nu$ if $q = d$; or $(\hat{R}_{\nu,l} \cdot \hat{V}_{v,c,\nu})^\alpha$ if $q = s$, $\hat{L}_{\nu,l}$ is a unit vector from ν to l, $\hat{N}_\nu$ is the normal vector at ν, $\hat{R}_{\nu,l}$ is the reflection of $\hat{L}_{\nu,l}$ along $\hat{N}_\nu$, and $\hat{V}_{v,c,\nu}$ is a unit vector from ν to the center of c when viewed from v; see Fig. 3.1, and α is the material's shininess constant.

Reflectances, camera sensitivity curves, and light SPDs are linearized via orthogonal projection onto bases $\mathcal{B} = \{\mathbf{b}_{bk}\}_{bk=1}^{N_b}$, $\mathcal{G} = \{\mathbf{g}_{bc}\}_{bc=1}^{N_g}$, and $\mathcal{H} = \{\mathbf{h}_{bs}\}_{bs=1}^{N_h}$, where N_b, N_g and N_h are the basis cardinality, respectively.

$$(\mathbf{k})_{\nu,q}(\lambda) = (\boldsymbol{\rho}_{bk})_{\nu,q}(\mathbf{b}_{bk})(\lambda) + (\mathbf{R}^{(k)})_{\nu,q}(\lambda), \tag{3.10}$$

$$(\mathbf{c})_c(\lambda) = (\boldsymbol{\kappa}_{bc})_c(\mathbf{g}_{bc})(\lambda) + (\mathbf{R}^{(c)})_c(\lambda), \tag{3.11}$$

$$(\mathbf{s})_{l,\Lambda}(\lambda) = (\boldsymbol{\varsigma}_{bs})_l(\mathbf{h}_{bs})(\lambda) + (\mathbf{R}^{(s)})_l(\lambda),\ l \in \Lambda, \tag{3.12}$$

where $(\boldsymbol{\rho}_{bk,\nu,q})$, $(\boldsymbol{\kappa}_{bc,c})$, and $(\boldsymbol{\varsigma}_{bs,l})$ are projection coefficient tensors, and $(\mathbf{R}^{(k)}_{\nu,q})$, $(\mathbf{R}^{(c)}_{c})$, and $(\mathbf{R}^{(s)}_{l})$ are the projection residual tensors for $(\mathbf{k}_{\nu,q})$, $(\mathbf{c}_c)$, and $(\mathbf{s}_l)$, respectively.

The tensor of pixel intensities is then

$$(\mathbf{P})_{v,c,\nu,\Lambda} = \int_{-\infty}^{\infty} (\mathbf{c})_c(\lambda)\ (\mathbf{I})_{v,c,\nu,\Lambda}(\lambda)\,\mathrm{d}\lambda \tag{3.13}$$

and is estimated after calibration using (3.8) and (3.9) as

$$(\hat{\mathbf{P}})_{v,c,\nu,\Lambda} = \int_{-\infty}^{\infty} (\hat{\mathbf{c}})_c(\lambda)\big[\big((\hat{\upsilon}_{l,q})_{v,c,\nu}(\hat{\mathbf{k}}_q)_\nu(\lambda)\big)(\hat{\mathbf{s}}_l)_\Lambda(\lambda)\big]\,\mathrm{d}\lambda,$$

where a hat indicates an estimate or projection. We then find coefficients that minimize projection residuals, i.e.

$$(\boldsymbol{\rho})^*, (\boldsymbol{\kappa})^*, (\boldsymbol{\varsigma})^*, \alpha^* = \underset{(\boldsymbol{\rho}),(\boldsymbol{\kappa}),(\boldsymbol{\varsigma}),\alpha}{\operatorname{argmin}} \|(\mathbf{P}_{v,c,\nu}) - (\hat{\mathbf{P}}_{v,c,\nu})\|_F^2, \tag{3.14}$$

where $\|\cdot\|_F$ is the Frobenius norm.

Since $(\boldsymbol{P})_{v,c,\nu,\Lambda}$ has slice-wise form $c(\lambda)\boldsymbol{A}s(\lambda)$ in the (c,Λ) axes, where $\boldsymbol{A}$ is a matrix of known quantities calculated from the remaining axes of $(\boldsymbol{P})_{v,c,\nu,\Lambda}$, resolving the camera sensitivities under unknown lighting conditions requires solving a system of bilinear equations for each camera-light combination. In contrast to [248, 250], we assume tile reflectances are known, whereas SPD, sensitivity and object reflectances are all unknown. Lights are cycled on and off for each view to properly fill $(\mathbf{P}_{v,c,\nu,\Lambda})$. We use a Macbeth ColorChecker with known curves, r_{ref} [251], and $\mathcal{B}$, $\mathcal{G}$, and $\mathcal{H}$, are determined via principle component analysis (PCA) with N_b, N_g, and N_h components, respectively. Thus, the only unknowns that remain in (3.14) during the calibration step are the camera and light basis coefficients $(\boldsymbol{\kappa}_{bc,c})$ and $(\boldsymbol{\varsigma}_{bs,l})$.

Tikhonov regularization is used to penalize large derivatives and thus produce smoother solutions [252]. We fix $(\mathbf{k}_{q,\nu})$ for each patch as r_{ref} for $q = d$ and 0 for $q = a, s$, and capture one view. To resolve sensitivities, data from all light combinations must be considered, and to resolve SPDs, data from all cameras must be considered. We therefore obtain

$$(\hat{\boldsymbol{\kappa}})_c^*, (\hat{\boldsymbol{\varsigma}})_l^* = \underset{(\hat{\boldsymbol{\kappa}})_c,(\hat{\boldsymbol{\varsigma}})_l}{\operatorname{argmin}} \alpha_1\Psi_1 + \alpha_2\Psi_2 + \alpha_3\Psi_3 \tag{3.15a}$$

$$\text{s.t. } \Psi_1 = \sum_{c,\Lambda} \|(\mathbf{P})_{:,c,:,\Lambda} - (\hat{\mathbf{P}})_{:,c,:,\Lambda}\|_F^2 \tag{3.15b}$$

$$\Psi_2 = \sum_{c} \|\mathbf{W}(\tilde{\mathbf{g}}_{\lambda,bk})(\boldsymbol{\kappa}_{bk})_c\|^2 \tag{3.15c}$$

$$\Psi_3 = \sum_{l} \|\mathbf{W}(\tilde{\mathbf{h}}_{\lambda,bs})(\boldsymbol{\varsigma}_{bs})_l\|^2 \tag{3.15d}$$

$$0 \leq (\tilde{\mathbf{g}}_{\lambda,bk})(\boldsymbol{\kappa}_{bk})_c \leq 1 \tag{3.15e}$$

$$(\tilde{\mathbf{h}}_{\lambda,bs})(\boldsymbol{\varsigma}_{bs})_l \geq 0 \tag{3.15f}$$

where $\mathbf{W}$ is the difference matrix, and $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{h}}$ are matrices of λ sampled values of $\mathbf{g}$ and $\mathbf{h}$. (3.15a)-(3.15f) are solved via alternating least squares [248] with sub-optimizations

$$(\hat{\boldsymbol{\kappa}})_c^t = \underset{(\hat{\boldsymbol{\kappa}})_c}{\operatorname{argmin}} \; \alpha_1 \|(\mathbf{P})_{:,c,:,:} - (\tilde{\mathbf{P}})^{(c)}_{:,:,c,:,:} \bar{\times}_1 (\boldsymbol{\kappa}_{bk})_c\|_F^2 + \alpha_2 \|\mathbf{W}(\tilde{\mathbf{g}}_{\lambda,bk})(\boldsymbol{\kappa}_{bk})_c\|^2 \tag{3.16a}$$

$$\text{s.t.} \quad 0 \leq (\tilde{\mathbf{g}}_{\lambda,bk})(\boldsymbol{\kappa}_{bk})_c \leq 1, \tag{3.16b}$$

where $(\tilde{\mathbf{P}})^{(c)}_{:,:,c,:,:} = \int_{-\infty}^{\infty} (\mathbf{g}_{bk})(\lambda) \otimes (\mathbf{I}_{:,c,:,:}) \, \mathrm{d}\lambda$, and

$$(\hat{\boldsymbol{\varsigma}})_l^t = \underset{(\hat{\boldsymbol{\varsigma}})_l}{\operatorname{argmin}} \; \alpha_1 \sum_{l \in \Lambda_l \subset \Lambda} \|(\mathbf{P})_{:,:,:,\Lambda_l} - \sum_{l \in \Lambda_l} (\hat{\mathbf{P}})^{(l)}_{:,:,:,:,l} \bar{\times}_1 (\boldsymbol{\varsigma}_{bs})_l\|_F^2 + \alpha_3 \|\mathbf{W}(\tilde{\mathbf{h}}_{\lambda,bs})(\boldsymbol{\varsigma}_{bs})_l\|^2 \tag{3.17a}$$

$$\text{s.t.} \quad (\tilde{\mathbf{h}}_{\lambda,bs})(\boldsymbol{\varsigma}_{bs})_l \geq 0, \tag{3.17b}$$

with $(\tilde{\mathbf{P}})^{(l)}_{:,v,c,\nu,l} = \int_{-\infty}^{\infty} (\mathbf{h}_{bs})(\lambda) \otimes (\hat{\mathbf{c}})_c(\lambda)(\mathbf{r})_{v,c,\nu,l}(\lambda) \, \mathrm{d}\lambda$, and $\otimes$ as the tensor product.

The sub-optimizations are recast as constrained quadratic programs and solved using sequential least squares. Initial guesses are set to the closest known illuminant for l, and the camera averages for c. α is not a design variables here since diffuse reflections is assumed. We assume $\mathcal{G}$ and $\mathcal{H}$ sufficiently span the curve space and $(\mathbf{R}^{(s)}) \approx (\mathbf{R}^{(c)}) \approx 0$.

3.4 Formalization of the Inspection Task

3D hyperspectral scanning requires the determination of reflectance curves along an object's surface, and comprises two connected but parallel processes: the creation of a 3D surface from sensor data, and the population of this surface with spectral data. In [253], we reconstruct the surface as a hyperspectral point cloud, which is similar to RGB point clouds used in most robotic applications in that the *xyz*-point cloud is matched to reflectance curves instead of RGB triplets. In that way, the reconstructed surface is represented by a series of points with position, normal, and spectral components. We choose hyperspectral point clouds for this task, as they are a canonical respresentation of the object surface, i.e., we can convert the hyperspectral point cloud back into 2D images, 3D color meshes, or other representations as needed for downstream applications, while also preserving the integrity of the underlying spectral and spatial data

[253]. In this section, we formalize the inspection task by describing it in relation to a theoretical point cloud that perfectly captures the object's surface, which we refer to as the ground truth hyperspectral point cloud (GT HSPC).

Given an object for which we have high-quality 3D scans (or meshes, in the case of simulated objects), the GT HSPC is generated by sampling a dense *xyz*-point cloud from the 3D scan using Poisson disk sampling. Hyperspectral data from low-specularity regions from multiple captures are then transferred onto the *xyz*-point cloud to generate the GT HSPC. Any new hyperspectral 3D reconstructions are then compared to this GT HSPC to determine how much of the GT HSPC data is present in the reconstruction. A hyperspectral point cloud is said to be *complete* if it corresponds vertex-to-vertex to the GT HSPC, including both surface vertex coverage (how much of the surface has been reconstructed) and the quality of per-vertex spectral information (how well the reconstruction captures the reflectance properties of the surface). More formally, we compare a set of reconstructed point cloud vertices $\mathbb{V}$ to the set of ground truth vertices $\mathbb{V}_0$ from the GT HSPC by first determining a minimum weight matching M on a complete bipartite graph $\mathbb{G} = (\mathbb{V}, \mathbb{V}_0, \mathbb{E})$ where edge weights c_e for $e \in \mathbb{E}$ correspond to the Cartesian distances between the vertices connected by e. We then cull $\mathbb{E}$ to create a matching M' so that only edges with $c_e < c_{min}$ are selected, where c_{min} is a user-defined threshold. A subgraph $\mathbb{G}' = (\mathbb{V}, \mathbb{V}'_0, \mathbb{E}')$ is then defined such that $\mathbb{G}$ is perfectly matched by M'. The *completeness* of a reconstructed hyperspectral point cloud is then measured using this subgraph as

$$\Xi(\mathbb{V}_c) = \frac{1}{|\mathbb{V}_0|} \sum_{\nu \in \mathbb{V}} \left[\frac{1}{\int_{\mathbb{L}} \mathrm{d}\lambda} \int_{\mathbb{L}} \zeta(\lambda)\, \mathrm{d}\lambda \right] \tag{3.18}$$

The spectral gain (exploitation) terms in the completeness metric are assumed proportional to the spatial gain (exploration) terms. This implies that filling in unknown parts of the object for each camera is roughly equivalent to decreasing the overall spectral variance when the cameras are considered independently, that is, the summations in (3.18) are identically equal to 1 for all points and wavelengths. As such, we evaluate any candidate views by an approximate metric Ξ' and then reevaluate the top

performing candidates using the full completeness metric. We again parameterize Ξ' by a matching between the ground truth point cloud and a point cloud $\mathbb{V}_c$ as done for Ξ, and define it as

$$\Xi'\Big(\bigcup_i \mathbb{V}_{c_i}\Big) = \sum_{\mathbb{V}_{c_i}} \frac{w(C_i)}{(N_c-1)|\mathbb{V}_0|},\ i = 1, \ldots, N_c, \tag{3.19}$$

This definition of completeness works equally well for unknown objects for which we do not have high-quality 3D scans, as long as completeness is taken as a relative measurement and not an absolute one. This is an important distinction in theory, but a moot one in practice, since we often determine when to stop collecting new data for reconstructions by observing the change in the IG, i.e. we stop collecting data if the reconstruction quality hasn't changed much in the last couple of views. As such, we also propose the (practically) equivalent but less restrictive completeness metric

$$\Xi = \frac{1}{A_S}\int_{\mathcal{S}} \zeta_p \zeta_n \int_{\mathbb{L}} \zeta_\lambda \,\mathrm{d}\lambda \,\mathrm{d}\mathcal{S}, \tag{3.20}$$

where ζ_p and ζ_n capture the uncertainty of the position and normal of each point along $\mathcal{S}$ and are defined as

$$\zeta(\lambda) = \frac{1}{1+\exp(-b(\lambda)[\sigma_\nu^2(\lambda) - (\sigma_a^2(\lambda)+\sigma_p^2(\lambda))/2])}, \tag{3.21}$$

$$b(\lambda) = \frac{2}{\sigma_p^2(\lambda)-\sigma_a^2(\lambda)} \ln\left(\frac{1-\varepsilon_\Xi}{\varepsilon_\Xi}\right). \tag{3.22}$$

3.5 3D Hyperspectral Reconstructions

Reflectances, residuals and α are calculated by solving a three-step iterative optimization problem. At each step t, $(\boldsymbol{\rho}_{k,\nu,q})$ is estimated by fixing $(\mathbf{R}^{(k)})$ and α, and solving

$$\begin{aligned}(\hat{\boldsymbol{\rho}}_{k,q})_\nu^t = \operatorname{argmin}\ & \alpha_1 \|(\mathbf{P})_{:,:,\nu,:} - (\hat{\mathbf{P}})^{(k)}_{:,:,\nu,:}\|_F \\ & + \alpha_4 \|\mathbf{W}(\tilde{\mathbf{b}}_{\lambda,bk})(\boldsymbol{\rho}_{bk,q})_\nu\|_F^2 \end{aligned} \tag{3.23}$$

$$\text{s.t. } 0 \le (\tilde{\mathbf{b}}_{\lambda,k})(\boldsymbol{\rho}_k)_\nu \le 1. \tag{3.24}$$

where the predicted intensity tensor is $(\hat{\mathbf{P}})^{(k)}_{v,c,\nu,\Lambda} = (\hat{\mathbf{P}})_{v,c,\nu,\Lambda} + \mathbf{e}_R$, with residual term $\mathbf{e}_R = \int_{-\infty}^{\infty} (\hat{\mathbf{c}})_c(\lambda)\big[\big((\hat{v}_{l,q})_{v,c,\nu}(\mathbf{R}_q^{(k)})_\nu(\lambda)\big)(\hat{\mathbf{s}}_l)_\Lambda(\lambda)\big]\,\mathrm{d}\lambda$.

We assume residuals are Gaussian processes, such that for any given residual, $(\mathbf{R}^{(k)})_{\nu,q}(\lambda) \sim \mathcal{GP}(\mu((\boldsymbol{\rho}_k)_{\nu,q}), k((\boldsymbol{\rho}_k)_{\nu,q}, (\boldsymbol{\rho}_k)'_{\nu,q})))$, with $k(\rho,\rho') = \sigma_R^2 \exp(-\frac{1}{2l^2}\|\rho - \rho'\|^2)$, where σ_R^2 and l are design parameters for signal variance and length scale. The reflectance coefficients and residuals from Section 3.3.2 are used as training points $(\boldsymbol{\rho}_T, \mathbf{R}_T)$ for a Gaussian process machine learning (GPML) model [254]. The residuals are then updated based on the new reflectance coefficient estimate $(\hat{\boldsymbol{\rho}}_{k,q})_\nu$ as

$$(\mathbf{R}^{(k)})^t_{\nu,q} = \mathbf{k}_*^T(\mathbf{K} + \sigma_R^2\mathbf{I})^{-1}\mathbf{R}_T, \tag{3.25}$$

where $\mathbf{k}_*$ is the vector of covariances between $(\hat{\boldsymbol{\rho}}_{bk,q})^t_\nu$ and $\boldsymbol{\rho}_T$, $\mathbf{K}$ is the training covariance matrix, and $\mathbf{I}$ is the identity matrix. Finally, α is updated to minimize errors due to specularity, by fixing $(\boldsymbol{\rho})$ and $(\mathbf{R}^{(k)})$ and minimizing

$$\begin{aligned} \alpha^t = \underset{\alpha}{\operatorname{argmin}} \sum_{v,c,\nu,\Lambda} \|(\mathbf{P}) - (\tilde{\mathbf{P}})^{(\alpha)}\|^2 \\ \text{s.t.} \quad \alpha > 0 \end{aligned} \tag{3.26}$$

where $(\tilde{\mathbf{P}})^{(\alpha)}$ is equivalent to $(\tilde{\mathbf{P}})$ with the specular term linearized using the first-order Taylor expansion $(\upsilon)_{v,c,\nu,l,s} \approx (a)^{\alpha^t} + \ln(a)a^{\alpha^t}(\alpha - \alpha^t)$, $a = (\hat{\boldsymbol{R}}_{\nu,l} \cdot \hat{\boldsymbol{V}}_{v,c,\nu})$. Optimization steps are then repeated until convergence, or maximum number of iterations. Note that (3.26) requires data from multiple views to solve for the reflectances of each point.

3.6 Next-Best-View Planning

Efficient data collection is enforced by selecting spectrally-optimal NBV sequences that maximize the approximate completeness Ξ' of the HPSC while minimizing total capture time T across all views

$$\mathcal{V}^* = \underset{\mathcal{V}}{\operatorname{argmax}}\ \Xi'(\mathcal{V}) - \alpha_T T(\mathcal{V}), \tag{3.27}$$

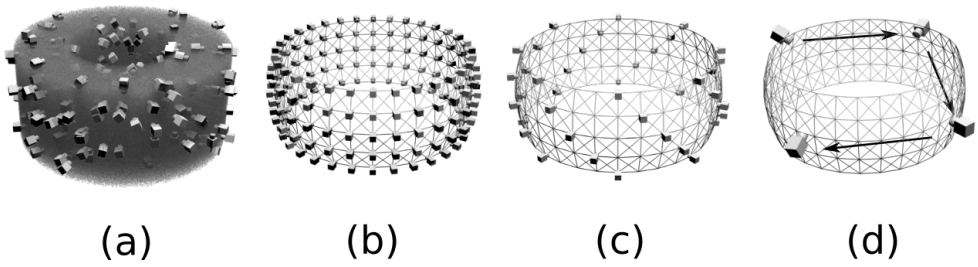


Figure 3.4: An illustration of the NBV procedure. (a) The space of all possible views is calculated as V. (b) A graph $\mathcal{G}$ of neighboring views is calculated by sampling V. (c) $\mathcal{G}$ is pruned to generate candidate views V_C. (d) An approximate NBV sequence is selected from V_C as $\mathcal{V}^*$.

where $\alpha_T \ll 1$ is a user-defined weighting factor.

We calculate spectrally-optimal next-best-views (SONBV) in a local, hierarchical fashion by first calculating a rough solution using a small, discrete subset of V, then iteratively following the local utility gradient around each view in $\mathcal{V}$. Fig. 3.4 shows a step-by-step illustration of the procedure.

A sequence $\mathcal{V}^*$ is optimal if it minimizes (3.27) with $T(\mathcal{V}) = \sum_{u=1}^{N_v-1}[\tau(v_u, v_{u+1})]$, where $\tau(v_u, v_{u+1})$ is the time to complete a trajectory from v_u to v_{u+1} and is measured and tabulated *a priori*. The selection of α_T presents a trade-off between completeness and total trajectory time.

Spectrally-optimal NBV selection is a combinatorial problem [253]. Due to memory and runtime limitations, we first discretize V as $\tilde{V}$ by uniformly sampling V, then store $\tau(v_u, v_{u+1})$ as a $|\tilde{V}| \times |\tilde{V}|$ matrix $\mathbf{T}$. We then generate a graph $\mathcal{G}$ with adjacency matrix $\mathbf{A}_\mathcal{G}$ as shown in Algorithm 1.

Algorithm 1 Adjacency Matrix Calculation

```
1: procedure AdjMat(T)
2:     A_G ← 1 − I, A'_G ← T
3:     while A_G ≠ A'_G do
4:         A'_G ← A_G
5:         for row in T do
6:             row← TrimFarNeighbors(row)
7:         A_G ← T ∨ T^T
```

A subgraph with $N_\mathcal{G}$ maximally-distant nodes is then calculated to sufficiently cover V, as shown in Algorithm 2. In practice, we assume the configuration space for the

cameras is a manifold, which allows us to use fast marching farthest point sampling with distances between $(v_i, v_{i+1}) \in V$ as $\mathbf{T}$ instead of geodesic distances along the manifold. The result is compiled into a list of candidate views.

Algorithm 2 Generation of Candidate Views

Require:
$\mathbf{T}$: Matrix of τs
$N_{\mathcal{G}}$: Desired graph size
procedure GenerateCandidateViews($\mathbf{T}$,$N_{\mathcal{G}}$)
  $V_C \leftarrow \emptyset$, $\mathbf{A}_{\mathcal{G}} \leftarrow$ AdjMat($\mathbf{T}$)
  $\mathbf{P} \leftarrow$ ShortestPaths($\mathbf{T}, \mathbf{A}_{\mathcal{G}}$)
  while $|V_C| < N_{\mathcal{G}}$ **do**
    $C \leftarrow$ FarthestNode($\mathbf{T}, \mathbf{P}$)
    $V_C \leftarrow V_C \cup \{C\}$

Algorithm 3 Approximate NBV Selection

Require:
V_C: List of candidate views
N_v: Sequence length
procedure ApproxSONBV(V_C,N_v)
  $\mathcal{V}^* \leftarrow \emptyset$, $J^* \leftarrow -\infty$, $\mathcal{V}^{N_v} \leftarrow$ perm(V_C, N_v)
  for all $\mathcal{V} \in \mathcal{V}^{N_v}$ **do**
    $\mathbb{V} \leftarrow$ ConstructCloud($\mathcal{V}$)
    $\tau_{tot} \leftarrow$ Sum($[\tau(\mathcal{V}[i], \mathcal{V}[i+1])$ for $1 \leq i < N_v]$)
    $J \leftarrow \Xi(\mathbb{V}) - \alpha_T(\tau_{tot})$
    if $J > J^*$ **then**
      $\mathcal{V}^* \leftarrow \mathcal{V}$, $J^* \leftarrow J$

An approximate NBV sequence with maximal J is chosen from permutations (or traveling salesman solutions for combinations) of length N_v of V_C by approximating completeness and total travel time for each permutation. The approximate NBV method is summarized in Algorithm 3.

Sequence refinement is performed as in Algorithm 4, by iteratively evaluating combinations of jumps from $u_v \in \mathcal{V}^*$ to $u'_v \in \tilde{V}$ for all $u_v, u'_v \in \tilde{V}$ such that $A_{\mathcal{G}}(u_v, u'_v) \neq 0$. This moves the optimal sequence along the (discrete) local cost gradient such that at each iteration i, it is guaranteed that $J(\mathcal{V}_i) \leq J(\mathcal{V}_{i+1})$. The search space with local refinement therefore includes any local maxima around $\mathcal{V}^*$, rather than being limited to the subset of V defined by $\tilde{V}$.

Algorithm 4 Local NBV Refinement

Require:
V_C: List of candidate views
$\mathcal{V}$: Approximate NBV sequence

procedure RefinedSONBV(V_C,$\mathcal{V}$)
  $\mathcal{V}^* \leftarrow \mathcal{V}$, $\mathcal{V}' \leftarrow \mathcal{V}$, $J^* \leftarrow J(\mathcal{V})$, $J' \leftarrow J^* + \varepsilon$, $u' \leftarrow \{\ \}$
  while $J' > J^*$ **and** $i{+}{+} < i_{max}$ **do**
    $J^* \leftarrow J'$, $\mathcal{V}^* \leftarrow \mathcal{V}'$
    for $u_v \in \mathcal{V}$ **do**
      for $u'_v \in \tilde{V}$ **do**
        if $A_{\mathcal{G}}(u_v, u'_v) \neq 0$ **then**
          $u'[u_v]$.append(u'_v)
    for $\{u'_v\} \in$ Comb(u') **do**
      if $J(\{u'_v\}) > J'$ **then**
        $\mathcal{V}' \leftarrow \{u'_v\}$, $J' \leftarrow J(\{u'_v\})$

3.7 Hyperspectral Classification

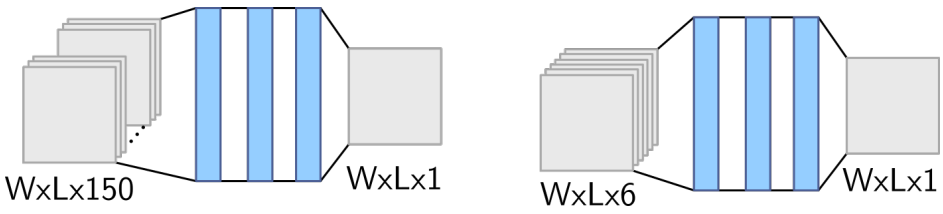


Figure 3.5: The proposed healthy/unhealthy hyperspectral classifiers based on a relatively shallow deep neural network. HSC-Full (left) takes high-resolution hyperspectral data as input, whereas HSC-Coeff takes reflectance coefficients as input.

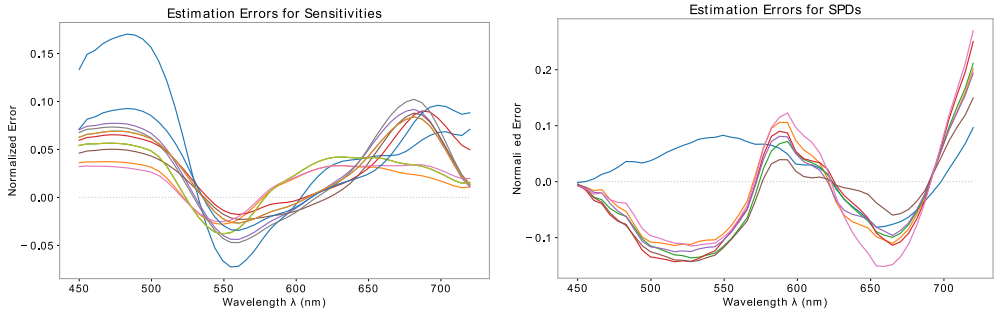
The hyperspectral data generated by the reconstruction algorithm is processed using a machine learning model to determine binary (healthy/unhealthy) classifications of the samples. We propose the two (relatively shallow) deep neural networks in Fig. 3.5 to learn the boundaries between spectral signatures that are healthy versus those that are unhealthy. The main difference between the two networks is that HSC-Full uses the raw hyperspectral data as input, whereas HSC-Coeff uses the reflectance coefficients as calculated in (3.25). HSC-Coeff is therefore shallower than HSC-Full. The hidden layers for each network are fully connected and narrow with a dropout of 0.8 to prevent over-fitting. The input is assumed to be regularized to $[0, 1]$ already. Furthermore, the absolute values for the reflectances are critical to accurate classifications, so the input for the two networks should not be regularized in general.

3.8 Results

In this section, we present experimental and simulation results to validate the algorithms presented above. Camera and light calibration results are presented first, since the calibration results inform the accuracy and precision of the remainder of the results. Reflectance estimation and spectrally-optimal NBV results are then presented separately for clarity, although it should be understood that they are both part of an iterative process and similarly inform one another.

3.8.1 Camera and Light Calibration

We calibrate camera and light data for 7 lights with peaks that span 450 nm–750 nm and 4 generic usb webcams with different Sony IMX color sensors (for a total of 12 channels) and compare our results to data from an imec Snapscan LS150+ camera. Figs. 3.6a and 3.6b show normalized camera and light calibration errors for each wavelength, respectively.



(a) (b)

Figure 3.6: (a) Estimation errors for each camera's sensitivity curve, normalized across all wavelengths for which data was available. The performance of the estimator decreases around the fringes for the selected basis, but is within acceptable limits. (b) Estimation errors for each light's spectral power distribution, normalized across all wavelengths for which data was available.

To ensure solution stability and convergence during the alternating least squares step, all sensitivities were optimized concurrently in the first step, followed by all SPDs in the next step, and so on until convergence. Fig. 3.7 shows a representative plot, where overall cost within each step is strictly decreasing. Weighting the Tikhonov term

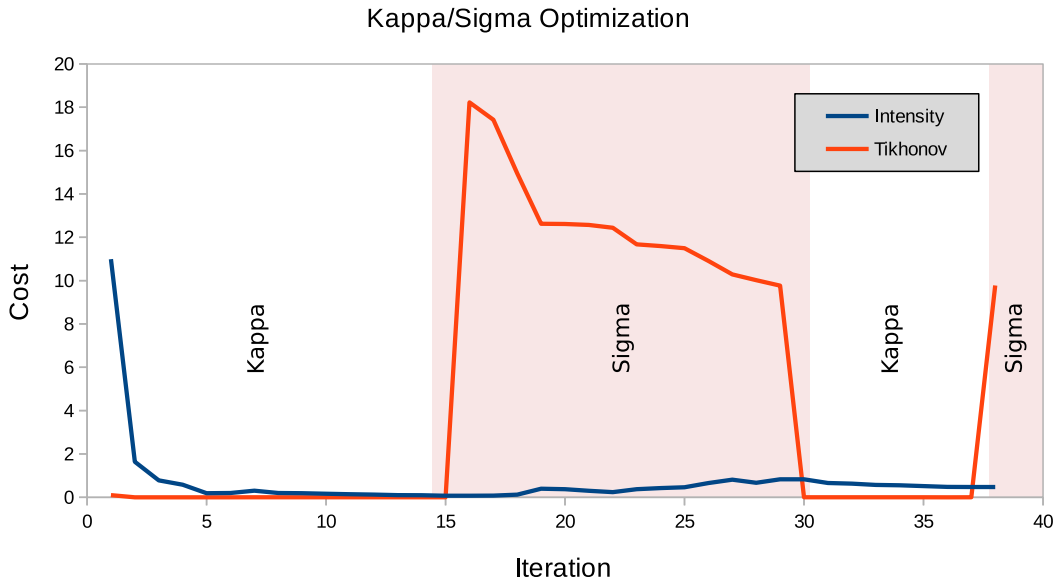


Figure 3.7: Camera and light coefficients converge after a few iterations of the alternating least squares estimator. Here, the optimizer alternates between minimizing costs for κ (white background) and for ς (light red background). The first cost terms (normed $\tilde{\mathbf{P}}$ errors) are plotted in blue, and the second cost terms (Tikhonov regularization cost) are plotted in orange. Cost coefficients are set to $\alpha_1 = \alpha_2 = \alpha_3 = 1$.

more than the intensity term can cause the estimation to fail, so α_2 should be selected so both terms are of comparable magnitude. Initial guesses are assumed to be close enough to the true solution to find the global optimum.

Solution smoothness can be increased at the expense of accuracy by changing α_2 and α_3, as shown in Fig. 3.8(a). Tikhonov regularization penalizes large coefficients for high-frequency basis functions, thus producing smoother solutions, as shown in Fig. 3.8(b), due to higher weightings for some basis functions, as shown in Fig. 3.8(c). This precludes the use of fluorescent bulbs, but works well with incandescents.

3.8.2 Reflectance Estimation

Reflectance curves were reconstructed for both known calibration tile reflectances and non-calibrated reflectances. Fig. 3.9a shows the estimates for the tile reflectances, while Fig. 3.9b shows mean errors for all tiles. Results are greatly affected by basis selectinos. Inaccurate SPD coefficients can cause the optimizer to produce consistently incorrect (and even physically-impossible negative) integrals for the $(\tilde{\mathbf{P}})^{(k)}$ terms. Since each step

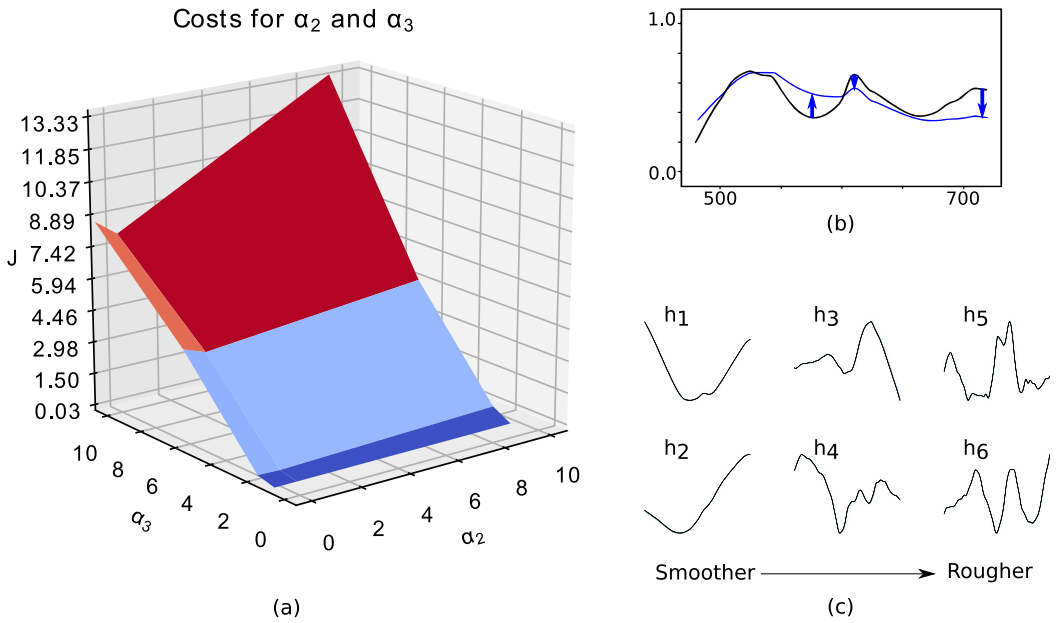


Figure 3.8: Effect of Tikhonov regularization on estimation results. (a) Estimation error cost generally decreases with increasing α_2 and α_3. (b) Increasing the Tikhonov cost weighting reduces sharp rises and falls in the final estimate. (c) Basis functions derived from principal components tend to have higher Tikhonov cost as the basis number increases.

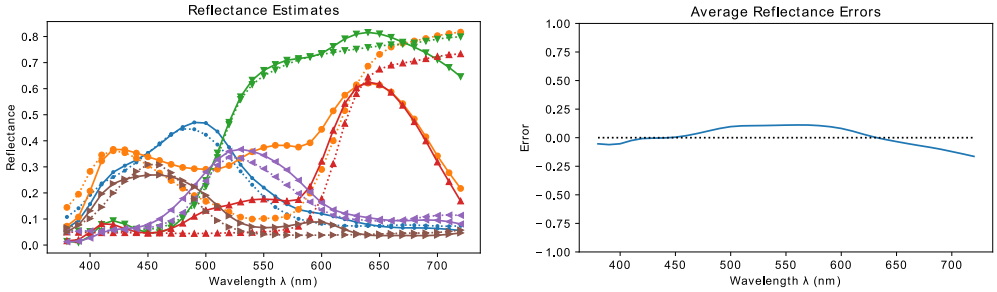


Figure 3.9: (a) Estimated (solid) and ground truth (dotted) reflectances for the primary color calibration tiles. (b) Average reflectance errors for all tiles. Reflectances are calculated by assuming diffuse-only reflections, and solving for ($\boldsymbol{\rho}$) like any other reflectance. Both κ and σ are fixed after camera and light calibration, so any errors in those coefficients carry over to the reflectance estimation, which can skew results.

in the three-step optimization behaves like a separate minimization, and the error terms are linearized around the previous step's optimal solution, the overall cost is minimized rapidly as shown in Fig. 3.10.

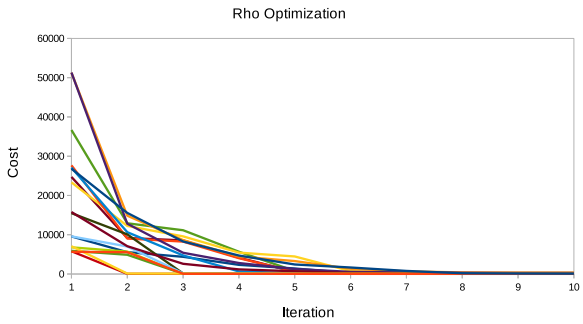


Figure 3.10: Reflectance coefficients ($\boldsymbol{\rho}$) are calculated by fixing all other variables, calculating the residual error term e_R and minimizing Eq. (3.23). Estimates for the coefficients converge rapidly (within a couple of optimization steps), but require several iterations of the three-step optimization process for quality estimates.

Table 3.1: Comparison of Spectrally-Optimal NBV Methods

	Exhaustive	Approx SONBV	Refined SONBV
Mean	0.9997	0.8714	0.9905
Max	0.9997	0.9923	0.9974
Min	0.9997	0.7612	0.9638
# evals	7880400	720	134727
# iters	1	1	3.3

3.8.3 Spectrally-Optimal NBV

We validated the spectrally-optimal NBV algorithm by comparing with an exhaustive search NBV algorithm. To ensure that the tests only compared NBV selections, a plant was scanned using the robot-mounted camera suite and the reconstruction shown in Fig. 3.11a was used to simulate both algorithms. The methods were compared by first sampling the space of all views to generate a discretized set $\tilde{V}$. Candidate views were then generated as shown in Algorithm 2. Mean, maximum and minimum NBV costs for an ε-tolerance of 0.3 were calculated for over 1000 trials of each method. Table 3.1 lists the results, and Fig. 3.12 shows the refinement steps for a random refined SONBV iteration chain. The refined SONBV method consistently yielded results comparable to an exhaustive search, even when the initial view sample was changed.

Data from combinatorial 3-sequences of 12 views for the first mesh were collected using the algorithm above, and each sequence was compared to the ground truth point cloud. Table 3.2 lists a summary of the results. Greedy single-view optimizations

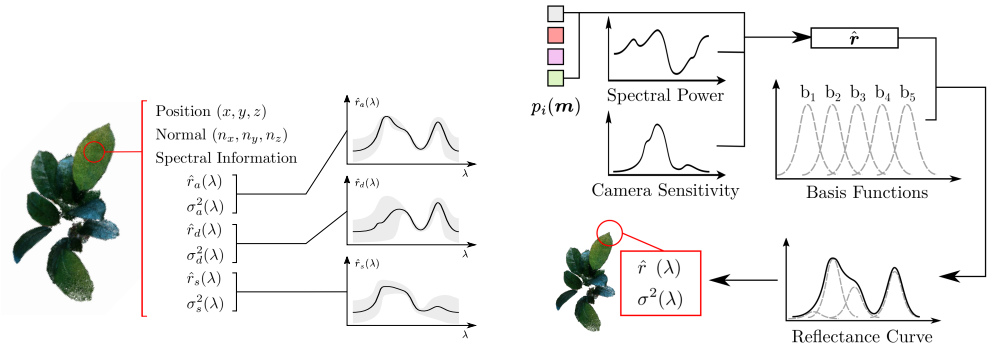


Figure 3.11: (a) A representative hyperspectral point cloud. Each point in the cloud contains both position and normal data as well as spectral data such as mean reflectances and their variances, etc. (b) Flowchart of hyperspectral point cloud generation from camera images. Raw sensor data $p_i(\boldsymbol{m})$ is used in conjunction with known spectral power distribution and camera sensitivity curves for the lights and the sensors, respectively, to generate an estimate of the reflectance curve. The reflectance curve is calculated and stored as a linear combination of basis functions, which allows for smaller storage and faster retrieval.

were often sub-optimal due to overlapping views or unexplored occlusions. This was mitigated by the combinatorial selection of sequences, i.e., NBVs must be selected as sequences rather than single views to ensure the resulting reconstruction is optimal.

To validate the simplifying assumptions used in defining the IG metrics, the approximate IG described in (3.19) was compared with the full IG in (3.18) for the same set of sequences. Table 3.3 shows the comparison results. We found that the approximate

Table 3.2: Completeness for sequences of one, two and three views.

View	1 View Max	2 View Max	3 View Max
$\theta = 30^\circ$ $\psi = 300^\circ$	**0.281932**	0.477162	0.616209
$\theta = 0$ $\psi = 120^\circ$	0.273646	**0.48363**	0.60084
$\theta = 0$ $\psi = 180^\circ$	0.262328	0.477162	0.616209
$\theta = 0$ $\psi = 300^\circ$	0.262126	**0.48363**	**0.621261**
$\theta = 0$ $\psi = 240^\circ$	0.255255	0.455133	0.598626
$\theta = 0$ $\psi = 0^\circ$	0.247373	0.464834	0.616209
$\theta = 0$ $\psi = 360^\circ$	0.247373	0.464834	0.616209
$\theta = 30^\circ$ $\psi = 360^\circ$	0.246968	0.464228	**0.621261**
$\theta = 30^\circ$ $\psi = 180^\circ$	0.246362	0.464834	**0.621261**
$\theta = 30^\circ$ $\psi = 240^\circ$	0.245554	0.458165	0.600849

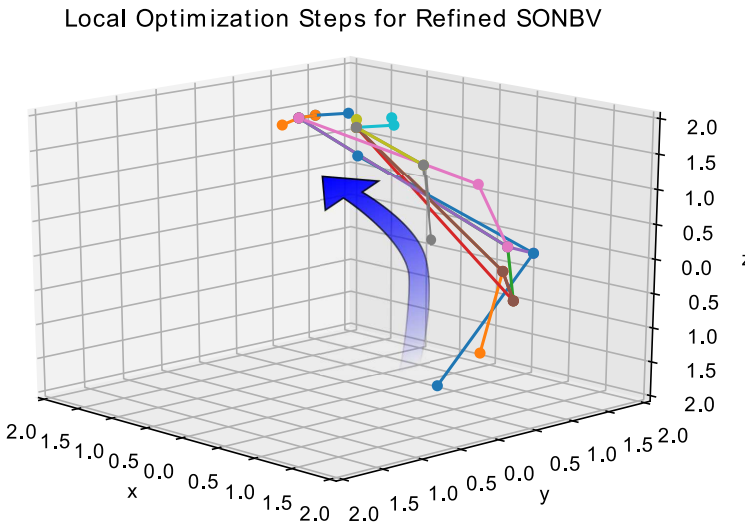


Figure 3.12: NBV refinement using Algorithm 4 for $N_v = 3$. Each dot represents the position of the central camera in the corresponding view candidate. The optimal view sequence moves along the utility gradient in the neighborhood of $\mathcal{V}$ at each step of the optimization. Cameras and the lines connecting them are the same color within a sequence.

IG consistently overestimated actual information gain by about 10%, but generally preserved the ordering across all sequences. The approximate IG does not consider data quality, and only considers visibility, whereas the IG also considers sensitivity curves and light angles, making it nearly intractable. By not performing the expensive calculations, the speed gains for the approximate IG are significant, by a factor of at least $O(npk\mathbb{L}/\,\mathrm{d}\lambda)$.

Fig. 3.13 shows resultant clouds for sequences of the same initial view. Fig. 3.14 illustrates a point cloud comparison which shows that the variance is smaller where the diffuse and specular terms can be eliminated, larger where the surface reflects the two light sources. Errors are concentrated along regions with similar normals. For points that receive indirect light, the system reduces to only the ambient terms. This is a fundamental limitation for real-world spectral data, as the diffuse and specular terms require incident light to be solvable. Conversely, points that receive direct light show aliasing between the ambient and diffuse terms, since they are both additive and constant at any viewing angle. Collecting data with both direct and indirect light helps

Table 3.3: Comparison of approximate and actual IG metrics.

Seq/Len	Full IG	Appx IG	%Diff	Abs Diff	# Full/# Appx
1/3	0.7934	0.8887	0.1072	0.0953	3/1
8/3	0.7929	0.8887	0.1078	0.0958	4/2
15/3	0.7919	0.8887	0.1089	0.0968	5/3
5/3	0.7916	0.8887	0.1092	0.0971	6/4
10/3	0.8127	0.8827	0.0793	0.0701	1/5
7/3	0.8116	0.8827	0.0805	0.0711	2/6
4/3	0.7767	0.8692	0.1063	0.0924	7/7
14/3	0.7763	0.8692	0.1068	0.0929	8/8
8/3	0.7756	0.8692	0.1076	0.0936	9/9
6/2	0.7229	0.8061	0.1032	0.0832	10/10

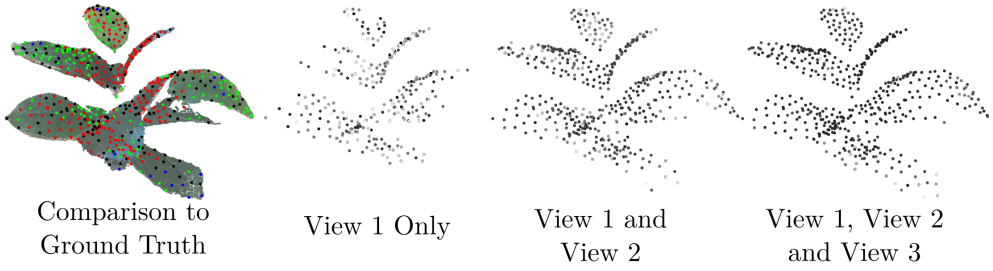


Figure 3.13: Comparison of multiple sequences containing the same initial view. (Left) Points captured by the sequence are shown on top of the full mesh. Points covered by the 1-sequence are colored red, points covered by the 2-sequence are colored green, points covered by the entire 3-sequence are colored blue, and points that were not covered by the sequence are colored black. (Center Left, Center Right, Right) Variances are shown in shades of gray. Lower variance data are darker.

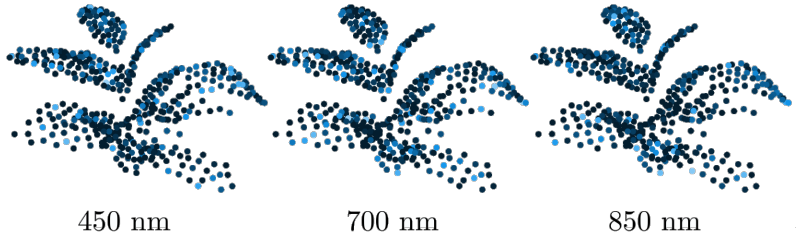


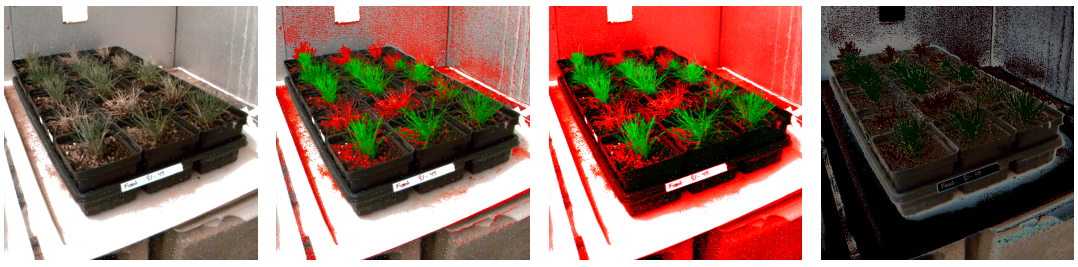
Figure 3.14: Comparison of a reconstructed experimental hyperspectral point cloud to the ground truth hyperspectral point cloud. Point intensity denotes relative reflectance error, darker is less error.

resolve this. We note that this is a known issue [255], and is generally mitigated by carefully controlling lighting conditions, which is not always possible.

A key limitation of the SONBV algorithm is that it requires one image per camera per light for each view. That is, the time it takes to capture a view is proportional to the number of lights, assuming cameras take images concurrently. The on-the-fly calculation of the spectrally-optimal neighboring view sequence also allows to calculate NBVs for large sets of views ($|\tilde{V}| > 10,000$), even when taking multiple shots of the same points, which would be infeasible using exhaustive search. Another limitation of the work lies in the use of high computational cost when calculating $\mathbf{T}$ and V_C. However both of these matrices can be calculated *a priori* and that would alleviate the computational impact in practice.

3.8.4 Hyperspectral Classification

The hyperspectral classification method was validated using hyperspectral cube data for fescue grass stress-tested in a growth chamber. Our method was compared against a commercial classification package sold by imec by first loading the cube into both classifiers, then segmenting the cube into healthy (green) and unhealthy (red) regions. Both classifiers were provided with limited sample data (1-2 examples of healthy and unhealthy plants) for training purposes, and both were asked to process the entire cube using the generated classifier models. The results are shown in Figs. 3.15 and 3.16.



(a) Original data (b) imec Classifier (c) HSC-full (d) Difference

Figure 3.15: Comparison of the imec Classifier and HSC-full. (a) Color image of original hyperspectral data, where pure white indicates spectral oversaturation, (b) Classification results obtained using the imec Classifier, (c) Classification results obtained using HSC-full, (d) Difference between the two classification results, where darker is better.

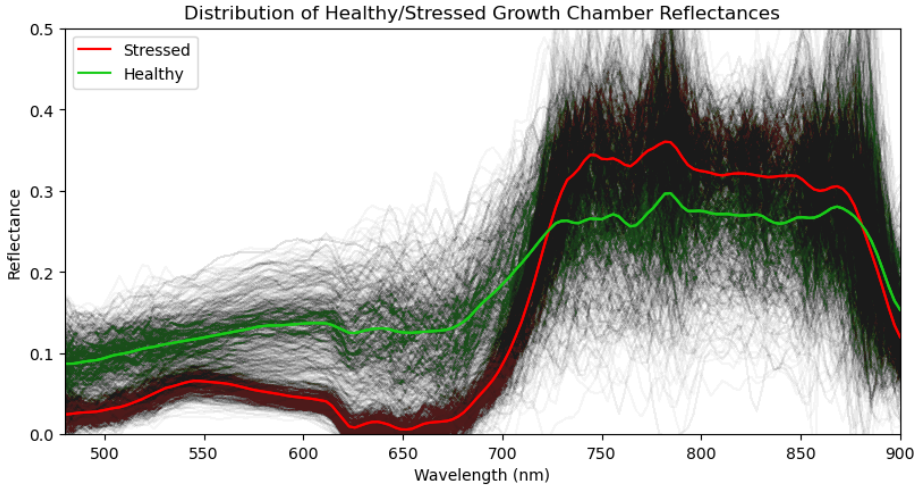


Figure 3.16: Classification results for stressed/healthy fescue grass samples raised in a growth chamber. Spectral curves are colored according to the predicted class, where green is healthy and red is stressed. Higher saturation indicates higher concentration of one class over the other.

The proposed classifier matched results generated by a commercial package for previously unseen hyperspectral cubes within 5 epochs of training, and achieved over 99% validation accuracy given a 80/20 split between training and validation data. The differences between the proposed method and the state-of-the-art classifier sold by imec in terms of inference time and accuracy for the 2048 x 1088 x 150 cube are minimal as shown in Fig. 3.15d, especially in critical inference regions corresponding to fescue samples. The key benefit of the proposed method is that it can operate on the basis coefficient vectors instead of the full spectrum, which greatly reduces the size of the overall classifier since the input size, as shown in Table 3.4, without incurring any penalties to the inference time. Additionally, the proposed model is developed in Tensorflow and is portable and lightweight. Integrating the proposed model into an agricultural automation workflow is therefore easier when compared to integrating proprietary classifiers that require specific software.

3.9 Conclusion

This chapter presented a comprehensive 3D hyperspectral reconstruction method, including camera spectral calibration and spectrally-optimal next best view planning. A

Table 3.4: Model and Inference Details for HSC

Classifier	Model Size (#params)	Single Inference (ms)		Batch Inference (ms)	
		CPU	GPU	CPU	GPU
HSC-Full	45602	33	33	51	48
HSC-Coeff	98	33	32	50	49

field calibration method was proposed for a set of RGB cameras mounted onto a robot arm, which allows the scanning system to self-calibrate at the start of each scan. A hyperspectral reflectance estimation method was developed based on spectral decomposition, where the reflectance curve is assumed to be a linear combination of a small set of basis functions plus a small residual term representing the difference between the ground truth reflectance curve and its projection onto the basis. It is shown that optimal basis selections greatly reduce the magnitude of the residual, however the residual must still be considered for RGB-only hyperspectral reconstructions since the span of the basis (and therefore accuracy of the predictions) is limited by the selection of cameras. An additional method is introduced to learn system-specific probabilistic residual functions based on the selected cameras and basis coefficient predictions. This residual method provides a probabilistic range for the reflectance estimates.

The 3D hyperspectral inspection task was formalized by considering the reflectance estimates for each point along the surface of the scan target. A novel information gain metric was proposed to compare two point clouds representing the same object. The metric was then used to compare experimental hyperspectral reconstructions to a ground truth hyperspectral point cloud. A matching between the points in these two point clouds was used to define a completeness metric for 3D hyperspectral reconstructions, and an approximate completeness metric was derived so hyperspectral point clouds could be evaluated without access to the ground truth point cloud. A 3D hyperspectral reconstruction method was then proposed to generate hyperspectral point clouds from RGB images collected from multiple cameras, and a next-best view algorithm was developed to maximize the completeness of a given reconstruction while also minimizing the number of images required to achieve a certain completeness. A

time-optimal extension to the next-best-view algorithm was proposed that incrementally refined a rough next-best-view solution obtained from a much smaller subset of possible views, and experiments were performed to show that this refined next-best-view method yielded results much closer to theoretically-optimal view sequences. Finally, a hyperspectral classification method was presented to segment the hyperspectral point cloud (or its 2D projections onto an image plane) according to a binary healthy-unhealthy classification. This classification method was then validated by comparing experimental results from a growth chamber to results obtained from a state-of-the-art hyperspectral classifier.

The goal of this chapter was to solve a fundamental issue with robotic crop inspections, i.e., how do we process the imaging data generated by our arm-mounted cameras to create a 3-dimensional description of the plant, and how can this description capture the spectral properties commonly used for phenotyping? The methods developed in this chapter allow the robot system described in 2 to plan successive views for its set of cameras and to then use the captured data to create canonical 3D hyperspectral reconstructions of the plant. Since the method described here is applicable to single plant scans, scanning large fields requires multiple such systems operating cooperatively. The planning and control challenges associated with the multi-robot system are investigated further in the following chapters.

Chapter 4

Multirobot Planning and Allocation for Inspection Tasks

4.1 Introduction

Inspecting crops at an industrial scale is a difficult task for a single robot but feasible for a group of robots operating in concert. However, collecting accurate and timely crop data for every single plant at such large scales is impractical, and autonomous crop inspection technologies must be used instead to achieve scalability [256]. Industry trends indicate growing demand for *ad hoc* inspection methods and continous monitoring using unmanned aerial and ground vehicles (UAVs and UGVs, respectively) [68], however these technologies are still limited in throughput when compared to satellite imaging. On the other hand, the types of data UGVs and UAVs can collect, and the level of detail that such data can achieve, make multi-robot systems a highly appealing option for crop inspection applications. The critical question that remains is whether the effectiveness of such systems can be supplemented on the software end of the equation, for example by predicting larger scale or regional trends from a limited set of data collected by a small group of robots. More importantly for the framework presented in this thesis, we want to utilize the hyperspectral scanning method presented in Chapter 3 to infer plot-wide metrics. In this chapter, we aim to answer this question by investigating planning and allocation methods directly connected to these metrics.

The main contributions of this work are threefold. First, we present a novel formulation for the multi-robot inspection problem for agricultural applications, including domain-specific constraints. This formulation reflects practical challenges associated with crop inspections, and in particular, with inspections performed on row crops by large (row-blocking) robots such as UGVs. Second, we present a learning-based method to turn the limited (i.e., single-plant or cluster) metric data collected by a multi-robot

group into plot-wide metric estimates using historical data and site-specific similarity kernels. This is an important development that enables larger scale metric analyses with throughput-limited data collection capabilities. Thirdly, we present a method to allocate these data collection tasks among the multi-robot group so that the plot-wide metric estimates are as accurate as possible. The partitioning and multi-robot task allocation methods and the metric estimation method are highly inter-dependent and affect the overall performance of the proposed multirobot planning and allocation framework.

The rest of the chapter is organized as follows. The multi-robot inspection problem for agricultural applications is presented in Section 4.2. A metric field estimation method is presented in Section 4.3 to utilize the inspection data collected by the multi-robot group. Two separate solution methods to the inspection problem are presented in Section 4.4 for discrete and continuous data collection, respectively, with corresponding partitioning and optimization methods. Results for the metric estimation and MRTA methods are presented in Section 4.5. Finally, a chapter conclusion is presented in Section 4.6.

4.2 Multi-Robot Inspection Problem

The multi-robot inspection problem (MRIP) is concerned with allocating a group of robots to collectively inspect a large group of targets. The problem is formalized in [257] for agricultural multirobot inspections as follows. Let $\boldsymbol{R} = \{r_1, ..., r_{N_R}\}$ be a group of robots, where N_R is the number of robots. The robots perform inspections on plants arranged along a plot topology $\mathcal{P}$, and each inspection performed at point $p \in \mathcal{P}$ at time $t \in \mathcal{T}$, where $\mathcal{T}$ is the growth cycle of the crop and t is relative to the start of this cycle, provides an observation o_k (e.g., a soil moisture reading). The set of all observations $\boldsymbol{O} = \{o_k | k = 1, ..., K\}$ are then used to calculate spatio-temporal metrics $\boldsymbol{m}(p,t) = [m_1(p,t), ..., m_n(p,t)]^T$, n is the number of metrics being investigated, which capture plant properties such as height, NDVI, and so on. For a large enough group of crops, these metrics can instead be represented as a time-variant mapping $f : \mathcal{P} \times \mathcal{T} \rightarrow \mathbb{R}^n$, which assigns each point in plot topology $\mathcal{P}$ a vector of metrics in $\mathbb{R}^n$ equivalent to $\boldsymbol{m}(p,t)$, denoted as $f(p,t)$ and estimated as $\hat{f}(p,t)$. The multi-robot

inspection problem is defined using these mappings as

$$\min J = \int_{\mathcal{T}'} \int_{p \in \mathcal{P}} \mathrm{E}[\|f(p,t) - \hat{f}(p,t)\|^2 | \boldsymbol{O}] \, \mathrm{d}p \, \mathrm{d}t, \tag{4.1}$$

where $\mathrm{E}[\,\cdot\,]$ is the expected value of the least-squares estimation error conditioned on the set of all observations collected by the group of robots. The minimization is performed over $\mathcal{T}' = [t_0, t_f]$, $t_0, t_f \in \mathcal{T}$, with $t_0 < t_f < t_K$ denoting inference, $t_K \leq t_0 < t_f$ denoting prediction, and $t_0 < t_K < t_f$ denoting a mix of inference and prediction.

Additional constraints are placed on (4.1) to ensure reliable results by requiring probably-approximately-correct (PAC) Bayesian bounds [258] on the estimate as

$$\Pr(\|f(p,t) - \hat{f}(p,t)\|^2 \geq \epsilon) \leq \delta, \ \forall (p,t) \in \mathcal{P} \times \mathcal{T}, \tag{4.2}$$

where Pr stands for probability. This constraint provides a theoretical upper bound on the estimation error given a set of observations, where ϵ is typically selected as a function of user-selected variable δ and the Kullback-Leibler divergence of f and $\hat{f}$. It also provides quantitative guidance on the number of samples required to achieve a certain confidence interval.

4.3 Estimation of the metric field

Optimal task selections and the allocation of these tasks to individual robots relies on the method used to estimate $\hat{f}$. We therefore first introduce a novel estimation method to determine $\hat{f}$ based on Gaussian process machine learning [210] before discussing task selection and allocation.

Let $\hat{f}^{(t)}$ denote an estimator associated with metric field f at time t. The strong spatial correlations in f are used to construct $\hat{f}^{(t)}$ using GPML, which prescribes a conditional posterior distribution $\hat{f}^{(t)} | \boldsymbol{P}, \boldsymbol{M}, \boldsymbol{P}_* \sim \mathcal{N}(\mu_{f^{(t)}}, cov(\hat{f}^{(t)}))$ on the estimate

with mean $\mu_{\hat{f}^{(t)}}$ and covariance $cov(\hat{f}^{(t)})$ as

$$\mu_{\hat{f}^{(t)}} = \boldsymbol{\kappa}\boldsymbol{M},\ \boldsymbol{\kappa} = K(\boldsymbol{P}_*, \boldsymbol{P})[K(\boldsymbol{P}, \boldsymbol{P}) + \sigma_n^2 \boldsymbol{I}]^{-1} \tag{4.3}$$

$$cov(\hat{f}^{(t)}) = K(\boldsymbol{P}_*, \boldsymbol{P}_*) - \boldsymbol{\kappa} K(\boldsymbol{P}, \boldsymbol{P}_*), \tag{4.4}$$

where $K : \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{R}^n \times \mathbb{R}^n$ is the covariance function (kernel) and $\boldsymbol{\kappa}$ is a matrix constructed by evaluating K for $\boldsymbol{P}$ and $\boldsymbol{P}_*$, $\boldsymbol{M} \in \mathbb{R}^{N_s \times n}$ is the training metric matrix, where N_s is the number of training samples, $\boldsymbol{P} \in \mathbb{R}^{N_s \times dim(\mathcal{P})}$ is the training position matrix, $\boldsymbol{P}_* \in \mathbb{R}^{N_q \times dim(\mathcal{P})}$ is the query position matrix, where N_q is the number of query points, and σ_n is the variance of an independent, identically distributed noise variable $\varepsilon \sim \mathcal{N}(0, \sigma_n)$ such that $\boldsymbol{m}(\boldsymbol{p}) = f(\boldsymbol{p}) + \varepsilon$, $\boldsymbol{p} \in \mathcal{P}$, and $\boldsymbol{I}$ is the $N_s \times N_s$ identity matrix.

For many applications, K is selected as a Gaussian kernel such as a radial basis function (RBF), i.e., $K(\boldsymbol{P}, \boldsymbol{P}') = \exp[-\frac{\|\boldsymbol{P} - \boldsymbol{P}'\|^2}{2\sigma^2}]$, where σ is a (user-defined) free parameter, and the parameters of the kernel are selected to match K to experimental covariances. The inherent downside to this kernel selection is the loss of spatial flexibility: Gaussian kernels cannot capture complex correlations that are not radially distributed, and they smooth out any directional biases or other such correlations due to environmental conditions. Radial basis function kernels will therefore perform poorly for crop inspection applications, where pathogens can travel along irrigation channels or where there are strong correlations along (but not necessarily across) crop rows.

We propose to learn site-specific and time-dependent kernels to estimate $\hat{f}$, and by extension, assess the spatio-temporal evolution of $\boldsymbol{m}$ [257]. This is done by constructing a novel kernel estimation network $K_\theta^{(t)}(\boldsymbol{p}, \boldsymbol{p}')$ shown in Fig. 4.1. This network is used to learn pair-wise spatio-temporal correlations from historical data for the entire field, thereby capturing the kinds of correlations that Gaussian kernels cannot capture. The trade-off is that this method requires access to a large set of historical data specific to the plot or crop that is being inspected (since the model is highly site-specific).

The loss function used to train the kernel estimation network $K_\theta^{(t)}$ is

$$\mathcal{L} = \mathcal{L}_{MSE}(\theta^{(t)}, \boldsymbol{P}, t) + \gamma_L \mathcal{L}_{coh}(\theta^{(t)}, \theta^{(t')}), \tag{4.5}$$

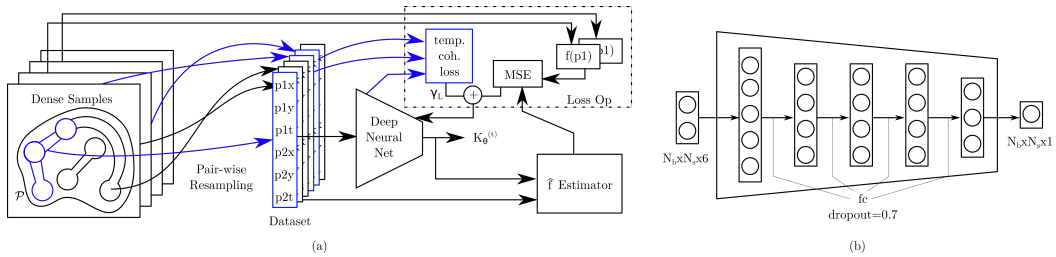


Figure 4.1: Schematic of kernel estimation network $K_\theta^{(t)}$. (a) Data flow diagram for the network. Time series data is resampled into pair-wise vectors and fed through the deep neural network (DNN). Resulting kernel estimates are used to construct custom mean squared error and temporal coherence losses as defined in (4.5). Training is performed using an Adam optimizer. (b) Overall DNN architecture, implemented in Tensorflow-Keras as a sequential network of dense layers (fc) with regularization and dropout.

where $\mathcal{L}_{MSE}(\theta^{(t)}, \boldsymbol{P}, t) = \|\boldsymbol{K}_\theta^{(t)}(\boldsymbol{p}, \boldsymbol{p'}) - \hat{\boldsymbol{K}}^{(t)}(\boldsymbol{p}, \boldsymbol{p'})\|_2^2$ is the mean squared error loss, and where $\boldsymbol{K}_\theta^{(t)}$ is the estimated kernel matrix, $\hat{\boldsymbol{K}}^{(t)}(\boldsymbol{P})$ is the sample covariance matrix calculated using pairwise Pearson correlation coefficients over the entirety of $\mathcal{T}$, $\mathcal{L}_{coh}(\theta^{(t)}, \theta^{(t')}) = \sum_{\boldsymbol{p}\in\boldsymbol{P}} \alpha_t \|\boldsymbol{K}_\theta^{(t)}(\boldsymbol{p}) - \boldsymbol{K}_\theta^{(t')}(\boldsymbol{p})\|_2^2$ is the temporal coherence loss with $\alpha_t = e^{-|t-t'|^2}$, and γ_L is a weighting hyper-parameter. This loss function provides a balance between estimation quality and temporal stability (coherence) that is tunable via γ_L during training. Decreasing γ_L therefore reduces the mean-squared error for the estimator but may cause jumps across t, whereas increasing γ_L would smooth out the kernel across t but may reduce estimation quality. This is an important balance, as $\mathcal{L}_{MSE}$ is a critical part of solving the MRIP defined in (4.1). Another option to keep γ_L low but reduce temporal jumps in $\hat{f}$ at the same time is to apply weight regularization to the dense layers of $K_\theta^{(t)}$.

The data used to train $K_\theta^{(t)}$ is extracted from dense multi-season crop data $\boldsymbol{P} = \{\boldsymbol{p}_i = (x_i, y_i, t_i) \in \mathcal{P} \times \mathcal{T} \mid i = 1, ..., N_s\}$ as follows. First, the $N_s \times 3$ data points are resampled into $N_s^2 \times 6$ pair-wise samples corresponding to the permutations of $\boldsymbol{P}$. We use all possible permutations to train the network, but it is possible to use only half of the permutations while enforcing covariance symmetry at a network architecture level. The training labels are then selected as the Pearson correlation coefficient,

$$\rho_{\boldsymbol{p},\boldsymbol{p'}} = \frac{var(\boldsymbol{p}, \boldsymbol{p'})}{(\sigma_{\boldsymbol{p}}\sigma_{\boldsymbol{p'}})}, \tag{4.6}$$

$\boldsymbol{p}$ and $\boldsymbol{p}'$ are the pairwise samples, which is guaranteed to return a value in $[-1, 1]$ due to the normalization by $\sigma_{\boldsymbol{p}}\sigma_{\boldsymbol{p}'}$. The temporal coherence is calculated over all samples rather than each pair, and requires a batch-wise (or larger) loss calculation as a result. When fully trained, the network returns the GPML covariance for any query $(\boldsymbol{p}, \boldsymbol{p}')$.

4.4 Optimal Multi-Robot Task Selection and Allocation

The selection of optimal per-robot tasks and the corresponding multi-robot task allocations (MRTA) depend on the modality of the observations that the robots are collecting. Inspection tasks that are heavily based on interacting with a single target for a longer period (e.g., 3D hyperspectral scanning) will lend themselves to capturing fewer targets that must be selected to maximize predictive impact across the entire field, whereas inspection tasks that are based on covering a larger area very quickly (e.g., 2D imaging) make it possible to capture more targets that are selected to maximize overall coverage instead. We therefore propose two alternative MRTA methods in this section to handle these two scenarios [257, 259]. Both methods are built on partitioning-based allocation strategies developed for multi-robot systems, with the main difference being the method by which potential inspection targets are prioritized.

4.4.1 Multi-Robot Task Allocation for Discrete Observations

Similar to other partition-based methods, [257] divides $\mathcal{P}$ into robot-centered regions before allocating tasks to each robot based on the regions they occupy. These regions are formed from $\mathcal{P}$ by first constructing an edge graph of $\mathcal{P}$ as $G = (V, E, \omega)$, where V is the set of nodes, E is the set of edges connecting the nodes, and ω are the edge weights, with a corresponding augmented geodesic Voronoi diagram $\text{Vor}_{G,K}$, where K is the set of nodes $v_i \in V$ that satisfy the minimal distance metric constraint $d_i(u, v_i) \leq d_j(u, v_j)$ for $j \neq i$. Here, the distance metric $d_i(u, v)$ is selected as the sum of the travel time over the shortest-path distance $d_{A^*}(u, v)$ for $r_i \in \boldsymbol{R}$ between nodes u and v and the

time-to-complete for r_i's current task, i.e.

$$d_i(u,v) = \frac{d_{A^*}(u,v)}{v_{max,i}} + \tau_{task,i}, \tag{4.7}$$

where d_{A^*} is the A* distance, $v_{max,i}$ is the maximum speed for r_i, and $\tau_{task,i}$ is the remaining time for r_i's task. For a set of robots with the same maximum speed and negligible task completion time, (4.7) can be approximated as $d_i \approx d_{A^*}$. Since each Voronoi region originates from a single robot's position, each robot $r_i \in \boldsymbol{R}$ sits at the center of a Voronoi region such that no other robot $r_j \in \boldsymbol{R}, j \neq i$ can reach a point in that region sooner than r_i. Since r_i is guaranteed to reach the points in its region before any other robot, any shortest path between two points contained within that region is collision-free.

Lemma 1 *Any shortest-path trajectory $\boldsymbol{x}_i(t)$ between two points $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$ that is fully contained within the interior of a region $K_i \subset \mathcal{P}$ of the augmented geodesic Voronoi diagram $\mathrm{Vor}_{G,K}$ is collision-free.*

Proof: Let $\boldsymbol{x}_c$ be the point of collision between r_i with shortest-path trajectory $\boldsymbol{x}_i(t)$ and r_j, $i \neq j$ with arbitrary trajectory $\boldsymbol{x}_j(t)$. At the time of collision, $\boldsymbol{x}_i(t) = \boldsymbol{x}_j(t) = \boldsymbol{x}_c$ and $d_i(x_0, x_f) = \int_{t_0}^{t} v_i(t)\,\mathrm{d}t$ and $d_j(x_0, x_f) = \int_{t_0}^{t} v_j(t)\,\mathrm{d}t$. If $\boldsymbol{x}_c \in K_i$, then $d_i = d_j$, and so $\boldsymbol{x}_c \in K_j$, and therefore it is also true that $\boldsymbol{x}_c$ must lie on the Voronoi edge between K_i and K_j. However, $\boldsymbol{x}_c \in \boldsymbol{x}_i(t)$ lies within the interior of K_i, and thus cannot lie on the Voronoi edge between K_i and K_j, so no point $\boldsymbol{x}_c$ exists such that $\boldsymbol{x}_c \in \boldsymbol{x}_i(t) \cap \boldsymbol{x}_j(t)$ and $\boldsymbol{x}_i(t)$ is collision-free. ■

Augmented geodesic Voronoi regions that satisfy (4.7) are calculated using Algorithm 5, and the Voronoi region for arbitrary points along edges in V can be queried using Algorithm 6. Note that Algorithm 5 first bisects the edge that r_i sits on to ensure that shortest distance paths do not cross the node occupied by r_i (i.e., robots block paths), and Algorithm 6 only calculates node-robot matchings after the graph retopography is complete. Each node in V is then assigned a robot based on (4.7), with distance tie-breakers resolving in favor of the first robot to perform the calculation.

Algorithm 5 Augmented Geodesic Voronoi Diagram

Require: G: Edge graph (V, E, ω)
$\boldsymbol{R}$: Set of robots

procedure AUGMENTEDGEODESICVORONOI
  $H \leftarrow G$, $\boldsymbol{R}_P = \{\}$
  for $r_i \in \boldsymbol{R}$ **do**
    $e \leftarrow$GETEDGE(r_i)
    $H \leftarrow$BISECTAT(H, e, r_i)
    for $r_j \in \boldsymbol{R} \backslash \boldsymbol{R}_P$ **do**
      if GETEDGE(r_j)==e **then**
        $e_c \leftarrow$GETEDGE$(r_i) \cap$GETEDGE(r_j)
        $r_j \leftarrow$REPLACEEDGENODE(r_j, e_c, r_i)
  return H
procedure BISECTAT(H, e, r)
  $H \leftarrow$DELETEEDGE(H, e)
  $H \leftarrow$ADDEDGE$(H, (e[0], r))$
  $H \leftarrow$ADDEDGE$(H, (r, e[1]))$
  return H

Algorithm 6 Voronoi Region Query

Require: G: Edge graph (V, E, ω)
H: Augmented geodesic Voronoi graph

procedure GETREGION(p)
  $e \leftarrow$GETEDGE(p)
  $r_{top}, d_{top} \leftarrow$CLOSESTROBOT(e[0])
  $r_{bottom}, d_{bottom} \leftarrow$CLOSESTROBOT(e[1])
  if $r_{top} = r_{bottom}$ **then**
    return r_{top}
  else
    $l_{edge} \leftarrow$EDGELENGTH(e)
    $l_{top}, l_{bottom} \leftarrow \delta l_{edge}, (1-\delta) l_{edge}$
    $d_{top}, d_{bottom} \leftarrow d_{top} + l_{top}, d_{bottom} + l_{bottom}$
    return r_{top} if $d_{top} < d_{bottom}$ else r_{bottom}
procedure CLOSESTROBOT(node)
  $d \leftarrow$ inf, $r_c \leftarrow \emptyset$
  for $r \in \boldsymbol{R}$ **do**
    $dist, path \leftarrow A^*(r, node)$
    $d, r_c \leftarrow d, r_c$ if $d \leq dist$ else $dist, r$
  return r_c, d

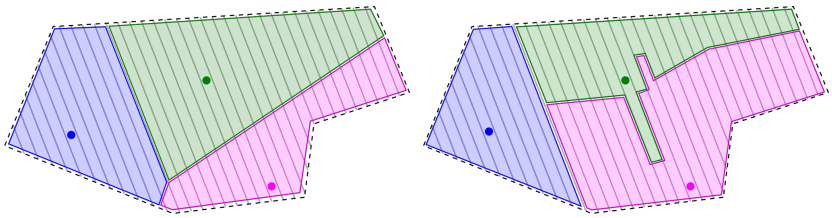


Figure 4.2: The difference in Voronoi partitions when the diagram is generated directly on a plot $\mathcal{P}$ (left) and when it is generated using G (right).

Fig. 4.2 shows the difference between Voronoi regions generated with L2 norms and augmented geodesic Voronoi regions generated using Algorithm 5. Lemma 1 guarantees that shortest-path trajectories for each region calculated using Algorithm 5 is collision-free, and thus traveling times and trajectories do not need to be optimized when selecting per-region inspection targets, as long as each robot selects a target within their Voronoi region at optimization time (i.e., the Voronoi region that is calculated when performing task allocation). The immediate implication for MRTA based on discrete observations is that task allocations can be calculated as robots complete their respective tasks, without having to consider possible trajectory overlaps or collisions, given a list of time-to-complete for each robot. Per-robot allocations are therefore made to minimize (4.1), which in turn minimize estimation errors across $\mathcal{P}$.

The optimal task selection problem is thus formed as the equivalent problem of maximizing the reduction in estimation error across $\mathcal{P}$ with each allocation. A more practical way of maximizing this quantity is to define a per-step cost functional

$$J_{step}(\boldsymbol{P}) = \int_{\boldsymbol{p}\in\mathcal{P}} cov(\hat{f}^{(t)}|\boldsymbol{P})\,\mathrm{d}\boldsymbol{p}, \tag{4.8}$$

where $cov(\cdot)$ is the covariance function, and $\boldsymbol{P}$ is the set of points to sample from. Since the temporal estimation and prediction errors must also be minimized, the total cost functional for an allocation is the cumulative posterior covariance over the entirety of $\mathcal{T}'$,

$$J_{task} = \sum_{t\in\mathcal{T}'} J_{step}(P^{(t)}), \tag{4.9}$$

where $P^{(t)}$ is the union of all previous (temporal) sample points and the sample points at step t. If $t_f > t_K$, then the problem must also account for future sample points, which creates a combinatorial problem. In fact, the decision problem associated with this optimization is NP-hard, as shown below.

Definition 1 *The optimal Voronoi task selection problem (VTS-OPT) consists of finding sets of metric sampling points* $P = \{\boldsymbol{p}_i : i = 1, ..., m, \boldsymbol{p}_i \in K_i\}$ *where a single sampling point* p_i *is selected from each region* K_i *such that* $J_{task}(P) < J_{task}(P')$ *for all* $P' \neq P$*, given a set of robots* $\boldsymbol{R}$ *and augmented geodesic Voronoi diagram* $\text{Vor}_{G,K}$*. Let* $H = (X, E)$ *be a weighted m-uniform m-partite hypergraph where* X *is the set of candidate sample points across* $\mathcal{P}$ *such that* m *sample points each are selected from* K_i*,* $i = 1, ..., m$*, and* E *is the set of non-empty subsets of* X *that form the hyperedges. Let* $J_{step}(e) : E \to \mathbb{R}$ *be a polynomial-time edge cost function such that* $J_{step}(e) \geq 0, \ \forall e \in E$*. The task selection problem above is equivalent to the decision problem (VTS-DEC) of determining whether* H *has a set of non-intersecting hyperedges* $\{e_i | i = 1, ..., t_f - t_K, e_i \in E\}$ *with summed cost less than* J_{task}*.*

Lemma 2 *VTS-DEC is NP-hard.*

Proof: It is straight-forward to see that VTS-DEC is in NP, as the edge cost of a solution can be calculated in polynomial time and the time complexity of the cost comparison is $O(1)$. We prove the problem is NP-hard by reducing the clustered coverage orienteering problem (CCOP) [190], which is NP-hard, to VTS-DEC, i.e., CCOP$\leq_P$VTS-DEC. Let $f_P(G'(V', E')) = H(X, E)$ be the reduction function, and let variables notated using primes follow their definitions in [190]. Construct $H(X, E)$ from nodes in $G'(V', E')$ so that cluster $V'_r \subset V'$ forms an independent set of X. Set $t_f = S_r^{min\ \prime} + t_K$ and $S'_r = C'_{max}$ and set $T'_{max} = \sum_{t>t_K} d^{(t)}_{max}$, where $d^{(t)}_{max}$ is the maximum edge distance of $Vor_{G,K}$ (bounded by $\mathcal{P}$). Define $\int cov(\hat{f}^{(t)}|P)$ as $Area'(y'_r)$. f_P is thus calculated in polynomial time, and the solution to $f_P(G'(V', E'))$ also solves the instance of CCOP. Therefore, VTS-DEC$\in$NP-hard. ■

As with other NP-hard problems, finding an globally-optimal solution to VTS-DEC (and its optimization equivalent, VTS-OPT) is a challenge that grows with the problem

scale. As such, [257] opts to find sub-optimal solutions via greedy allocation. With the greedy allocation method, (4.9) is replaced by a functional

$$J_{greedy} = \min \sum_{i \in \mathcal{I}} \left[\gamma_P d_i(x_i, p_i) + \sum_{j<i} K(p_i, p_j) \right], \tag{4.10}$$

where $\gamma_P d_i(x_i, p_i)$ is the weighted path cost for r_i and $\mathcal{I}$ is the set of indices for robots that have completed their assigned inspection tasks. Algorithm 7 illustrates the greedy allocation method that solves (4.10).

Algorithm 7 Region Based Task Allocation (RBTA)

Require: G: Edge graph (V, E, ω)
$Vor_{G,K}$: Augmented Voronoi diagram
$\boldsymbol{R}$: Set of robots

1: **procedure** RBTA
2: $P \leftarrow \{P^{(j)}\}$, $P^{(j)} \leftarrow \{\}, j = 1, ..., K$,
3: **for** $t = t_k, ..., t_f$ **do**
4: **for** $r_i \in \boldsymbol{R}$ **do**
5: $K_i \leftarrow$ VoronoiRegionAtTime(i, t)
6: $P' \leftarrow P^{(j)} \cup P^{(t)}$, $j = 1, ..., K$
7: $P^{(t)} \leftarrow P^{(t)} \cup \text{argmin}_{p \in K_i} J_{greedy}(p|P')$

Note that the greedy algorithm is concerned with selecting a single target for whichever robot is being currently assigned a task. The algorithm achieves this by selecting a target p_i within the robot's Voronoi region that is as different from all previous inspection targets across all robots (i.e., as uncorrelated to any previous inspection targets as possible). The user-defined weighting factor γ_P balances overlaps in K with the distance the robot has to travel to the next point. Therefore, large γ_P values would force the robot to move to closer points, whereas smaller γ_P values force the robot to move to points with the largest covariance difference. Observations from one $\boldsymbol{p}$ in a set of correlated positions P_{corr} increase the cost for the remaining member positions $\boldsymbol{p}_{corr} \in P_{corr}$ as

$$\Delta J_{greedy}(\boldsymbol{p}_{corr}|\boldsymbol{p}) = K(\boldsymbol{p}_{corr}, \boldsymbol{p}). \tag{4.11}$$

Note that collecting additional data around $\boldsymbol{p}$ reduces $cov(\hat{f}^{(t)}(\boldsymbol{p}))$ by (4.4), and

gives the condition to satisfy (4.12) via the quantile of the MSE distribution as

$$\Pr(MSE < \epsilon) = \text{erf}(\epsilon / cov(\hat{f}^{(t)})) > \delta, \tag{4.12}$$

where erf is the Gauss error function, which depends on the hyperparameters of the GPML model used to calculate $cov(\hat{f}^{(t)})$, and therefore corresponds to θ in (4.5). (4.11) can also be substituted into (4.4) to check whether the current sample set satisfies (4.12).

4.4.2 Multi-Robot Task Allocation for Continuous Observations

The solution method presented in the previous section is still valid and (sub-)optimal, but not necessarily efficient for task allocations with continuous observations which do not require extensive interactions with a single target, or for which a single observation captures the state of multiple targets. Such scenarios include large-scale 2D imaging inspections (e.g., via aerial drones or multirow-spanning UGVs) or collect-as-you-go inspections for air quality or atmospheric humidity. For these scenarios, simply passing through a region provides sufficient data. We propose a solution method for such inspections in [259], with specific focus on resolution-optimality for 2D imaging using a combined UGV-UAV team. The solution method described here follows a similar derivation, however it is noted that the method trivially generalizes to any other metric-optimal optimization as described in (4.1).

Like the previous section, the solution method for continuous observations first divides $\mathcal{P}$ into per-robot regions before allocating inspection tasks (i.e., routes along which the robot will collect data). However, since the method is now concerned with long-term data collection and continuous observations, additional constraints are added to the problem, and the metric-based optimization in (4.1) is recast as a maximization of a distance-based quality term $f(\boldsymbol{x}_i, \boldsymbol{p})$ that accumulates throughout each robot's trajectory $\boldsymbol{x}_i, i = 1, ..., N_R$, as

$$\max_{\boldsymbol{x}_i} J = \int_{\mathcal{T}'} \int_{\boldsymbol{p} \in \mathcal{P}} f(\boldsymbol{x}_i, \boldsymbol{p}) \, \mathrm{d}\boldsymbol{p} \, \mathrm{d}t, \tag{4.13}$$

where the integration is performed over the same domain as (4.1). Note that (4.13) is equivalent to (4.1) given a careful selection of f.

Since path selection now plays a critical role in maximizing J, we expand the definition of the robot set $\boldsymbol{R}$ to include state vectors $\boldsymbol{\xi}_i = [\boldsymbol{x}_i^T, \boldsymbol{q}_i^T, \dot{\boldsymbol{x}}_i^T, \dot{\boldsymbol{q}}_i^T]^T$ for each $r_i \in \boldsymbol{R}$, where $\boldsymbol{x}_i = [x_i, y_i, z_i]^T$ and $\boldsymbol{q}_i = [q_{1i}, q_{2i}, q_{3i}, q_{4i}]$ are the position and orientation components of the pose vector $\boldsymbol{p}_i = [\boldsymbol{x}_i^T, \boldsymbol{q}_i^T]^T \in \mathcal{P}_i$, respectively, with $\mathcal{P}_i$ as the set of feasible poses for r_i, and $\dot{\boldsymbol{x}}_i$ and $\dot{\boldsymbol{q}}_i$ are the linear and angular components of the velocity vector $\boldsymbol{v}_i \in \mathcal{V}_i$, respectively, with $\mathcal{V}_i$ as the set of feasible velocities for r_i. The set of feasible states is $\mathcal{X}_i \subseteq \mathcal{P}_i \times \mathcal{V}_i$. The dynamics equation for r_i with control input $\boldsymbol{u}_i(t) \in \mathcal{U}_i$, where $\mathcal{U}$ is the set of feasible inputs, is

$$\dot{\boldsymbol{\xi}}_i(t) = f_i(\boldsymbol{\xi}_i(t), \boldsymbol{u}_i(t)),\ \boldsymbol{\xi}_i(t) \in \mathcal{X}_i,\ \boldsymbol{u}_i(t) \in \mathcal{U}_i, \tag{4.14}$$

where $f_i(\cdot)$ is assumed deterministic and is selected as a single integrator for simplicity for the remainder of this section.

The energy capacity of r_i at time t is denoted as $B_i(t)$, and falls within a feasible range

$$\underline{B}_i \leq B_i(t) \leq \overline{B}_i, \tag{4.15}$$

where $\underline{B}_i$ and $\overline{B}_i$ are the minimum and maximum allowable energy capacities for r_i, respectively. The power consumption and recharge rates are denoted as $P_i(t) = P_{self,i} + P_{ext,i}$ and $C_i(t)$, respectively, where $P_{self,i}$ is the power consumption of r_i itself and $P_{ext,i}$ is the combined power consumption of any robots using r_i's batteries to recharge, such that

$$\dot{B}_i(t) = -P_i(t) + C_i(t). \tag{4.16}$$

The recharge rate for r_i is $C_i(t) = C_s$ when recharging at a charge station, where C_s is the charging rate of a station $s \in \mathcal{S}$, $\mathcal{S}$ is the set of charging stations, and $C_i(t) = 0$ otherwise. If r_i is an aerial robot, r_i can also land on a UGV to charge itself

from the UGV's batteries, at which point $C_i(t) = C_g$, where C_g is the charging rate of the UGV-mounted charger, and P_{ext} for the UGV now includes $C_i(t)$. We make a simplifying assumption that $P_{self,i} = \underline{P}_i$ when the robot is idle or hovering, where $\underline{P}_i$ is the average power drawn by the on-board computers when the robot is stationary, and $P_{self,i} = \overline{P}_i$ when the robot is moving, where $\overline{P}_i$ is the average power drawn by all subsystems at full speed, and $P_{self,i} = 0$ while charging.

Inspections are performed via on-board sensors with quality measure denoted as $f_{i,k}(t)$ for robot r_i at position $\boldsymbol{x}_i(t)$ and inspection target c_k that is at a discretized position $\boldsymbol{w}_k$ in $\mathcal{P}$, $k = 1, ..., K$, K is the dimensionality of the discretization. We assume that the quality term $f_{i,k}(t)$ for a target inside the range for the sensor attached to r_i scales inversely with the robot's distance to the target up to a maximum quality of $\overline{f}$, namely,

$$f_{i,k}(t) = \min(f_{0,i}/\|\boldsymbol{x}_i(t) - \boldsymbol{w}_k\|, \overline{f}), \tag{4.17}$$

where $f_{0,i}$ is the quality at ground level, which we assume is higher for UGV-mounted sensors compared to UAV-mounted sensors (due to the carry weight and battery capacity available to UGVs). Otherwise, the capture quality is $f_{i,k}(t) = 0$ if the target is outside the sensor range. A schematic overview of the quality calculations in (4.17) is illustrated in Fig. 4.3. To minimize quality reductions due to movement and to ensure similar performance at different heights, aerial speed is capped at

$$\|\boldsymbol{x}_i(t)\| \leq \frac{h_i(t)}{h_0} v_0, \tag{4.18}$$

where $h_i(t)$ is the height (distance above ground level) of the robot at t, h_0 is a reference height corresponding to $f_{0,i}$, and v_0 is the desired speed at h_0 as selected by the designer.

The inspection mission for planning horizon $t \in [0, T]$ is then formulated as the

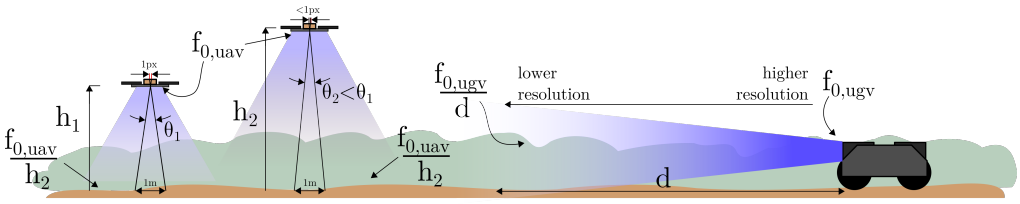


Figure 4.3: Unmanned aerial/ground crop inspection missions are often used to capture high-resolution crop imagery for phenotyping or yield estimation applications. The spatial (pixel) resolution of the plant image is inversely proportional to the distance.

constrained minimization,

$$\underset{i=1,..,N}{\boldsymbol{x}_i^*(t)} = \underset{\substack{\boldsymbol{x}_i(t)\\ i=1,...,N}}{\operatorname{argmax}} \int_0^T \left[\frac{1}{K} \sum_{k=1}^{K} \Big(\max_{\substack{\tau\in[0,t]\\ i=1,...,N}} f_{i,k}(\tau) \Big) \right] \mathrm{d}t \tag{4.19}$$

$$\text{s.t. } (4.14),\ (4.15),\ (4.16),\ (4.17),\ (4.18).$$

Note that the spatial discretization converts one of the integration terms in (4.13) to a sum over the discretized target list, so that the integration in (4.19) leads to a minimization with respect to the time it takes to achieve full coverage, whereas the summation leads to a maximization with respect to the quality of the coverage.

To determine optimal walks that solve (4.13), we first represent $\cup_i \mathcal{X}_i$, $i = 1, ..., N$, as an undirected graph $\mathcal{G} = (V, E)$ with vertices $v \in V$ and edges $e \in E$, such that each path $\boldsymbol{x}_i(t)$ for r_i can be discretized as a sequence of vertices and edges constituting a walk $w_i \in \mathcal{G}$. The optimization presented in (4.19) is therefore equivalent to finding walks $w_i \in \mathcal{G}$ that maximize the reward function

$$J(w_1, ..., w_N) = \sum_{k=1}^{K} \max_{w_i \in \mathcal{G}} \phi_k(\boldsymbol{\omega}_k, w_i), \tag{4.20}$$

where $\boldsymbol{\omega}_k$ is the discretization of $\boldsymbol{w}_k$, and $\phi_k(w_i)$ is the maximum resolution attained for plant k by r_i over w_i,

$$\phi_k(\boldsymbol{\omega}_k, w_i) = \max_{w \in w_i} \frac{f_{0,i}}{\|w - \boldsymbol{\omega}_k\|}. \tag{4.21}$$

We denote the reachable set for r_i at time t as $R_i(\boldsymbol{x}_i(t), \mathcal{U}_i)$, which is defined as the

subset of $\mathcal{X}_i$ for which the energy constraint in (4.16) is satisfied, i.e.,

$$B_i(t_0) + \int_{t_0}^{t_1} \dot{B}_i(\tau)\,\mathrm{d}\tau \geq \underline{B}_i,\ t_0, t_1 \in [0, T], \tag{4.22}$$

where t_0 and t_1 are any two time points along r_i's trajectory to and from any target pose $\boldsymbol{x}_t \in R_i(\boldsymbol{x}_i(t), \mathcal{U}_i)$. It is worth noting that (4.22) includes any number of recharge cycles, and $R(\boldsymbol{x}_i(t), \mathcal{U}_i)$ therefore represents the set of points that r_i can reach when cooperating with other robots in $\mathcal{R}$. Thus, for any UAV r_i and UGV r_j, $R_j(\boldsymbol{x}_j(t), \mathcal{U}_j) \subseteq R_i(\boldsymbol{x}_i(t), \mathcal{U}_i)$, as long as r_i can land on r_j. Since UGVs can only recharge at charging stations, the upper limit on $R_j(\boldsymbol{x}_j(t), \mathcal{U}_j)$ is the set of points that reduce B_i to $\underline{B}$, which are at most a path distance of

$$\overline{d}_j = \frac{\overline{B}_j}{\overline{P}_j} \frac{\|v\|_{0,j}}{2} \tag{4.23}$$

away from any charging station. Therefore, $R_j(\boldsymbol{x}_j(t), \mathcal{U}_i) \subseteq \cup_{s \in \mathcal{S}} \{\boldsymbol{x} :\ d(\boldsymbol{x}, \boldsymbol{x}_s) \leq \overline{d}_j\}$, where $d(\cdot, \cdot)$ is the geodesic (path) distance and $\boldsymbol{x}_s$ is the location of $s \in \mathcal{S}$. Per (4.22) and (4.23), any walk $w_j \in \mathcal{G}_j$ is fully contained within $R(\boldsymbol{x}_j(0), \mathcal{U}_j)$, so the search space for (4.19) is bounded by the reachable sets for each robot at $t = 0$.

Lemma 3 *Given a known topology for $\mathcal{G}$ and a fixed robot set $\mathcal{R}$, lower and upper bounds to $J(\boldsymbol{x}_1(t), ..., \boldsymbol{x}_N(t))$ can be found* a priori.

Sketch Proof of Lemma 3. Assume the trivial trajectories $X_0 = \{\boldsymbol{x}_i(t) : \boldsymbol{x}_i(t) = \boldsymbol{x}_i(0),\ i = 1, ..., N\}$ satisfy (4.14)–(4.18) and that the integral can be evaluated. Since the sum is monotonically increasing in t, any feasible set $X \neq X_0$ yields $J(X) \geq J(X_0)$ and $J(\mathcal{X}_0)$ constitutes a trivial global minimum and thus a lower bound on $J(\cdot)$. To show that an *a priori* upper-bound exists, assume that a walk w_i is constructed for each robot r_i. If r_i is a UGV, the longest walk will visit at least one charging station c_0 if there exists a charging station closer than $\overline{B}_i \|v\|_0 / \overline{P}_i$ away, in which case R_i is the union of the disjoint sets of all points starting at each c_0 that satisfy (4.23), and w_i visits all points in R_i. Walks for UAVs can be evaluated similarly by assuming that additional single-use charging stations exists for each point $\boldsymbol{x}_j$ inside the reachable set

for each UGV r_j, with a capacity of $B_j\overline{d}_j/2d(\boldsymbol{x}_j, c_0)$, where c_0 is the nearest charging station to $\boldsymbol{x}_j$, such that the UGV either transfers all remaining capacity to the UAV or carries the UAV to another point within R_j. The upper bound for $J(w_1, ..., w_N)$ is then found using (4.20) and (4.21). Since R_i is a superset of the set of unique points in w_i^*, $\phi_k(w_i)$ represents the theoretical maximum resolution that can be attained by r_i at the expense of r_j, $j \neq i$, and thus $J(w_1^*, ..., w_N^*) \leq J(w_1, ..., w_N) = \overline{J}$. ■

Since this problem is still NP-hard, and finding globally-optimal solutions is intractable for large K, we propose a greedy energy-constrained mission planner (ECMP) in [259] that generates sub-optimal trajectories with known optimality bounds via $\overline{J}$. Algorithm 8 outlines the steps required to plan optimal UAV/UGV crop inspection mission using ECMP, given *a priori* information about $\mathcal{G}$.

Algorithm 8 Energy-Constrained Mission Planner (ECMP)

Require:
$\mathcal{R}$: Set of robots
$\mathcal{G}$: Graph representation of plot

procedure ECMP
  GenReachable($\mathcal{R}, \mathcal{G}$) ▷ Algorithm 9, 10
  $\overline{R}_i \leftarrow \tilde{R}_i(\boldsymbol{x}_i(0), \mathcal{U}_i)$
  $\overline{J} \leftarrow J(\{w_i\}),\ w_i = \{(v_1, v_2) : v_1, v_2 \in \overline{R}_i\}$ ▷ (4.20)
  $\mathcal{T} \leftarrow$PartitionPlot($\tilde{R}_i, \emptyset$)
  $w_j \leftarrow$PlanWalks($\mathcal{T}$)
procedure PartitionPlot($\tilde{R}_i, \mathcal{T}_i$)
  for $\tilde{R}_i$, $i = 1, ..., N$, $v \in V$ **do**
    if $d(r_i, v) < d(r_j, v)$, $j \neq i$ **then**
      $\mathcal{T}_i = \mathcal{T}_i \cup v$
      $\mathcal{T}_j = \mathcal{T}_j \backslash v$, $j \neq i$
procedure PlanWalks($\mathcal{T}$)
  $w_i \leftarrow \emptyset$
  while any($v \in V \notin \cup w_i$) **do**
    for $r_i \in \mathcal{R}$ **do**
      $w_i \leftarrow w_i \cup$OptTarget

Since the calculation of the reachable set for the UAVs is a combinatorial problem and likely NP-hard itself, it is necessary to calculate $\overline{J}$ using an approximate reachable set $\tilde{R}_i$. When approximating $\tilde{R}_i$, we disregard the trivial case where the robot can complete an optimal path on a single charge, meaning r_i must recharge its batteries at least once to satisfy (4.15). Since UAVs can also use UGVs to recharge, the calculation

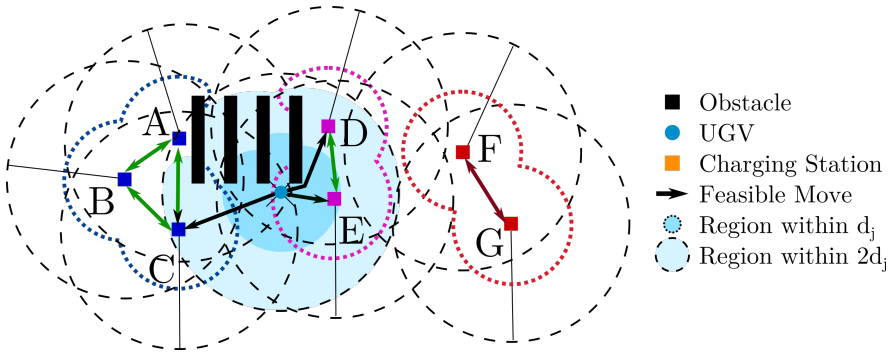


Figure 4.4: Energy constraints limit robots to subsets of their reachable set for long missions where charge depletion is undesirable. The reduced reachable set depends on the selection of a starting charging station, and is bounded by $\overline{d}_j$ afterwards. Here, the UGV (green circle) selects between the subset defined by stations A-B-C (dotted blue) or the subset defined by stations D-E (dotted pink). The subset defined by stations F-G (dotted red) is unreachable.

of $\tilde{R}_i$ for a UAV r_i depends on overlapping reachable sets for all UGVs, and thus we must first calculate $\tilde{R}_j$ for all UGVs before calculating $\tilde{R}_i$.

UGVs can only recharge at charging stations. The reachable set can therefore be calculated exactly, given the topology (including station locations) of the plot, and is equal to the union of all points within a distance of $\overline{d}_j$ of any charging station that is at most a distance of $2\overline{d}_j$ away from $\boldsymbol{x}_j(0)$. We disregard any points that fall outside this range, even if they are reachable in the strictest sense of the word (i.e. within a distance of $2\overline{d}_j$ of $\boldsymbol{x}_j(0)$), as they can only be visited once without violating the energy constraint, and end the UGV's mission (through energy depletion) when visited at any other point. Fig. 4.4 demonstrates the distinction between these two regions, and Algorithm 9 provides matching pseudo-code. Note that the choice of a first charging station creates a partition of the full reachable set for r_j and locks r_j into that partition.

Approximate reachable sets for UAVs are calculated similarly. First, we find reduced reachable sets that only visit charging stations. Then, we expand the reduced reachable sets using intersecting reachable sets for all UGVs. These two steps are repeated until convergence. The reachable set for a UAV r_i is approximated by first calculating the initial approximate reachable set $\tilde{R}_i^{(1)}$ using the method described above. An effective

Algorithm 9 UGV Reachable Set

Require:
c_i: An initial charging station
$\mathcal{G}$: Graph representation of plot

procedure GenReachableUGV($\mathcal{R}, \mathcal{G}$)
  $C \leftarrow \{c_i\}$, $C' \leftarrow \emptyset$
  while $C \neq C'$ **do**
    $C' \leftarrow C$
    $C \leftarrow C \cup \{c_1 : \exists c_2 \in C, d(c_1, c_2) \leq 2\overline{d}_j\}$
  $R_j \leftarrow \{v : d(c_j, v) < \overline{d}_j\}$, $c_j \in C$

range of $2\overline{d}_i$ is used instead of $\overline{d}_i$, and the set is expanded to include nearby UGVs as

$$\tilde{R}_i^{(2)} = \tilde{R}_i^{(1)} \cup \tilde{R}_\cap,\ \tilde{R}_\cap = \{\tilde{R}_j : \tilde{R}_i^{(1)} \cap \tilde{R}_j \neq \emptyset\}, \tag{4.24}$$

which is the union of $\tilde{R}_i^{(1)}$ with the reachable set for any UGV that r_i can reach by depleting its energy. The expanded approximate reachable set $\tilde{R}_i^{(2)}$ denotes the set of points reachable by the UAV if carried on a UGV. We therefore expand this region again to calculate the approximate reachable set,

$$\tilde{R}_i^{(3)} = \tilde{R}_i^{(2)} \cup \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}^{(2)}\| \leq \overline{d}_i,\ \boldsymbol{x}^{(2)} \in \tilde{R}_\cap\}, \tag{4.25}$$

which is the set of points further reachable by taking off from a UGV and landing back on it.

Note that UAVs can "hop" between the reachable sets for any UGV in range by landing on a UGV and riding it to the edge of the UGV's reachable set, then flying over to another UGV's reachable set so that it can land on the other UGV and be driven to a charging station. Fig. 4.5 demonstrates an example of UGV hopping. A similar approach also exists for bridging UAV reachable sets by landing on an intermediate UGV. Fig. 4.6 demonstrates a single iteration of this process, and Algorithm 10 presents pseudo-code.

The partitioning method divides the plot into regions that can and cannot be serviced by $\mathcal{R}$ long-term (i.e., as regions inside and outside $\cup\tilde{R}_i$, respectively). The partitioning algorithm can therefore be extended to determine the optimal placement of

Algorithm 10 Approximate Reachable Sets for UAVs

Require:
$\mathcal{R}$: Set of robots
$\mathcal{G}$: Graph representation of plot

1: **procedure** REGION(center, radius)
2: **return** $\{\boldsymbol{x} : d(\boldsymbol{x}, center) < radius\}$
3: **procedure** NEARESTSTATION($\boldsymbol{x}$, radius)
4: nearest$\leftarrow \emptyset$, $d_{min} \leftarrow$ inf
5: **for all** s$\in$REGION($\boldsymbol{x}$,radius) **do**
6: **if** d(nearest,s) $< d_{min}$ **then**
7: nearest$\leftarrow$s, $d_{min} \leftarrow d$(nearest,s)
8: **procedure** VIABLESTATIONS(region,visited)
9: vs$\leftarrow \emptyset$
10: **for all** station **within** $\overline{d}_i$ **of** region **do**
11: **if** station$\notin$visited **and** station **has** UGV **then**
12: vs$\leftarrow$vs $\cup$ station
13: **procedure** GENREACHABLEUAV($\mathcal{R}, \mathcal{G}$)
14: **for all** UGV $r_j \in \mathcal{R}$ **do**
15: $R_j \leftarrow$EXACTREACHABILITYSET(r_j)
16: **for all** UAV $r_i \in \mathcal{R}$ **do**
17: $s_0 \leftarrow$NEARESTSTATION($r_i, \overline{d}_i$)
18: APPROXREACHABLE($r_i, \mathcal{R}, \mathcal{G},$)
19: **procedure** APPROXREACHABLE($r, \mathcal{R}, \mathcal{G}, s_0$)
20: $\tilde{R}_i \leftarrow \emptyset,\ \tilde{R}_i^{(2)} \leftarrow \emptyset,\ \tilde{R}_i^{(3)} \leftarrow \emptyset$
21: $\tilde{R}_i^{(1)} \leftarrow$REGION($s_0$,$\overline{d}_i$)
22: visited$\leftarrow\{s_0\}$
23: **while** $\tilde{R}_i^{(1)} \neq \tilde{R}_i^{(3)}$ **do**
24: vs$\leftarrow$VIABLESTATIONS($\tilde{R}_i^{(1)}$)
25: **for all** s$\in$vs **do** ▷ (4.24)–(4.25)
26: $\tilde{R}_i^{(3)} \leftarrow \tilde{R}_i^{(1)} \cup$ REGION($s, \overline{d}_j + \overline{d}_i$)
27: visited$\leftarrow$visited $\cup$ s

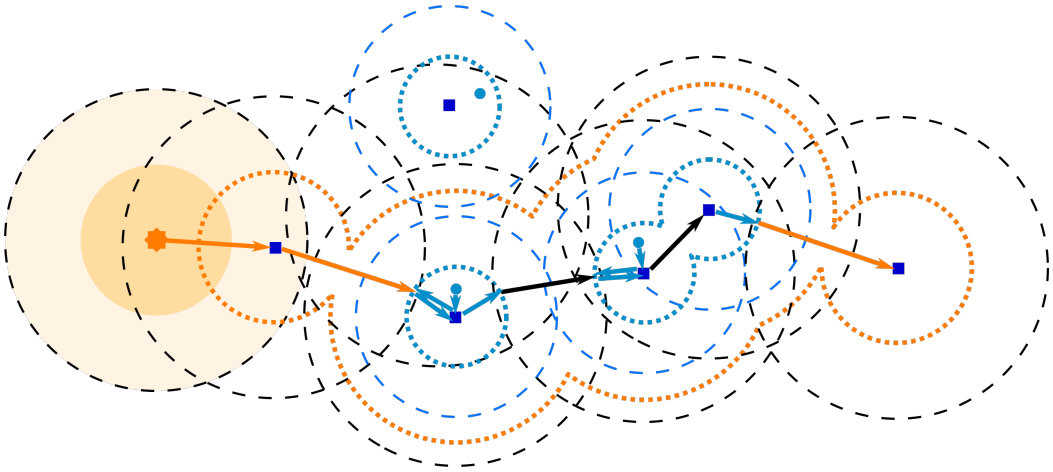


Figure 4.5: Unlike UGVs, UAVs can hop between regions outside their own range limit by landing on cooperative UGVs. Here, the UAV (orange star) can use the UGVs (blue circles) to hop between their reachable sets (dotted blue) by combining UAV-only moves (orange arrows), UGV only moves (blue arrows) and free moves (black arrows). The resultant reachable set for the UAV is shown as a dotted orange outline.

charging stations for continuous monitoring by ensuring the reachable sets fully cover the plot. Once the plot is partitioned, target selections are made by maximizing the cumulative resolution gain, estimated as follows: First, we convert the union of all previous paths into a binary map. Then, we subtract this binary map from a map created from the walk that is being tested. The estimate is then the multiplication of this binary map and the inverted maximum captured image resolution map. This process is then repeated in simulation to calculate the full walk for each robot, as shown in Algorithm 11.

Algorithm 11 Calculate Full Cover

1: **procedure** FULLWALKS
2: **while** $\exists v \in V, v \notin \cup w_i$, $t \leftarrow t + \Delta t$ **do**
3: $\xi_i = \int_t^{t+\Delta t} \dot{\xi}_i(\tau)\, \mathrm{d}\tau$ ▷ (4.14)
4: **for** $r_i \in \mathcal{R}$ **do**
5: **if** $\|\boldsymbol{x}_i(t) - w_i[-1]\| < 0.01$ **then**
6: $w_i \leftarrow w_i \cup$ OPTTARGET

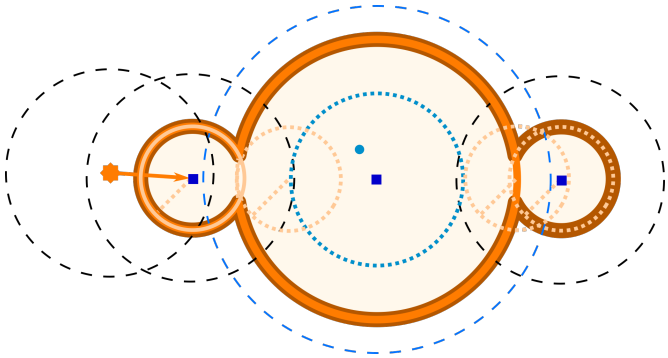


Figure 4.6: The approximate reachable set for a UAV r_i (orange star) is calculated by iteratively expanding an initial approximate reachable set $\tilde{R}_i^{(1)}$ (solid light orange) to first include a set of points $\tilde{R}_i^{(2)}$ (solid orange) reachable with the help of an adjacent UGV r_j with reachable set $\tilde{R}_j$ (dotted blue), then repeating the expansion to include a set of points $\tilde{R}_i^{(3)}$ (solid dark orange) reachable using adjacent charging stations.

4.5 Simulation results

4.5.1 Estimation of the metric field

The performance of the kernel method is evaluated by comparing $\hat{f}_K$ calculated via kernel estimation and $\hat{f}_G$ calculated via Gaussian RBFs to ground truth metrics f. Fig 4.7 shows the time evolution of $\hat{f}_K$ and $\hat{f}_G$ when identical samples are used. The RBF is designed such that the radius is the weighted norm $r = \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_W$, where W is a diagonal matrix of element weights, and is optimized for the training samples. The learned kernel is trained on a dataset collected by randomly inoculating a portion $I_0 = 0.01$ of the field on DAS0, where DASx is x days after sowing (DAS). Note that estimates with large prediction horizons (e.g., when t_2 =DAS0) fail for both models. Our method converges rapidly, with both 10-day and 1-day predictions closely matching the ground truth data on DAS29.

4.5.2 MRTA for discrete observations

We evaluate the performance of our graph-based pathfinding method used to generate Voronoi regions by comparing it to grid-based A* implemented by transforming a thresholded aerial image of the plot into the axis-aligned binary map. Finally, we generate a random group of robots and calculate path solution times and retopography

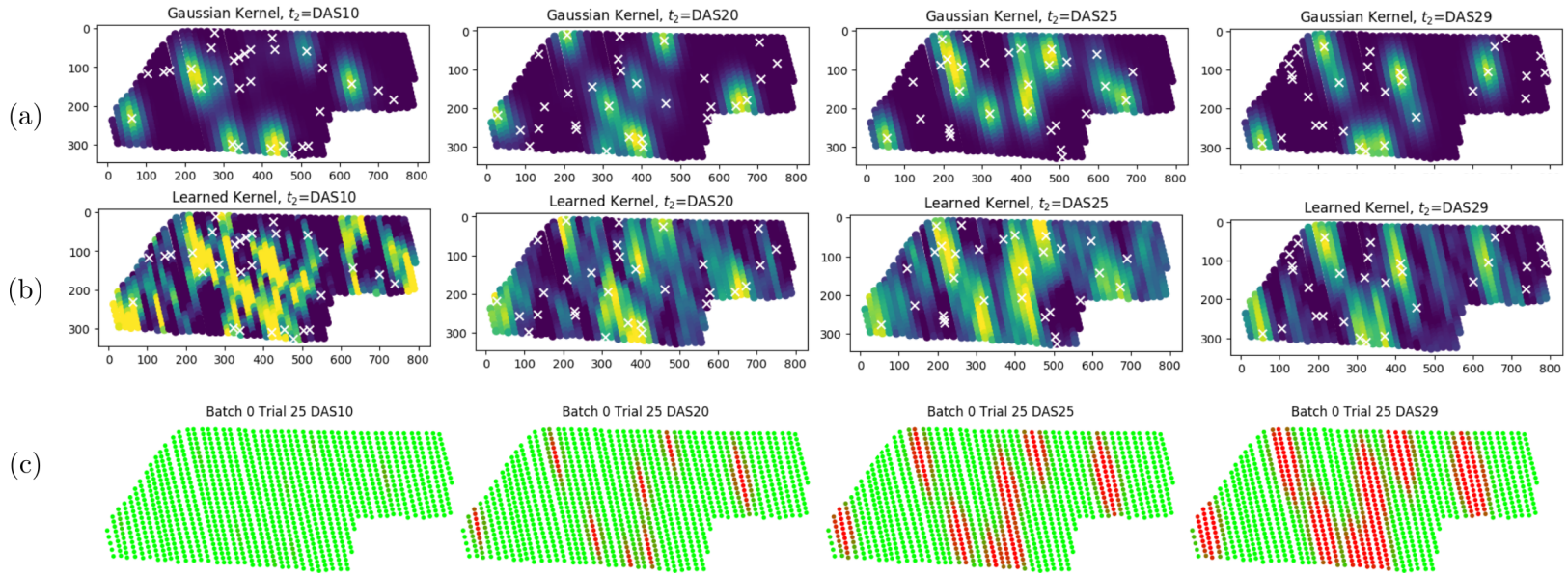


Figure 4.7: Comparison of metric fields for our simulated model (a) using anisotropic Gaussian RBF, (b) using kernel estimation, and (c) ground truth data. We sample only 30 randomly selected locations for each t. Our learned kernel produces better long-term predictions compared to the RBF-based kernel under identical sampling constraints.

(graph or grid generation) times as shown in Fig. 4.8. Our method maintains its performance for plots with curved rows, whereas the grid-based A* must convert these shapes to large grids due to the resolution needed to distinguish rows, thus degrading performance. We further evaluate our method by comparing graph-based and L2 Voronoi RBTA. Fig. 4.9 shows that while step-wise (i.e., one target per robot at each step) costs are similar, L2 Voronoi RBTA is slower when accounting for travel times, since the L2 cost function does not account for row constraints. Our method will therefore go through more data collection steps in the same time horizon, leading to lower costs.

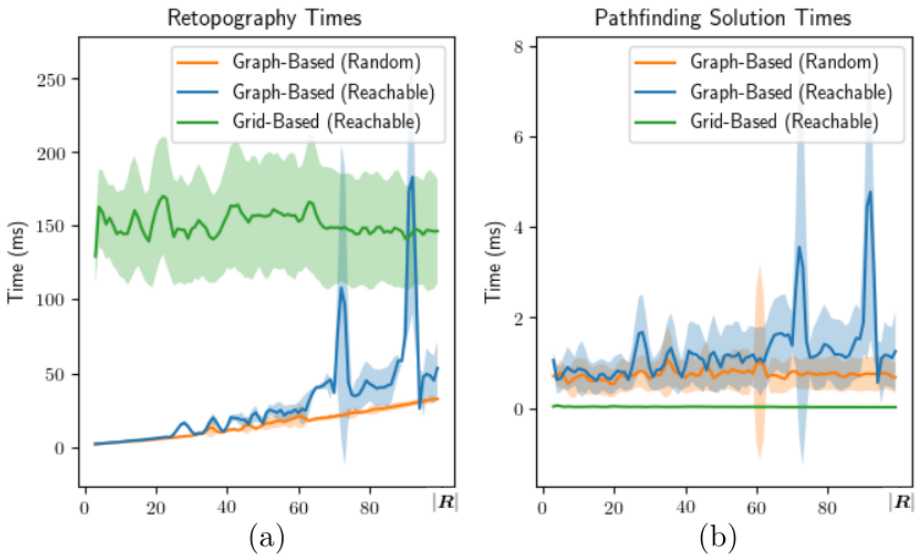
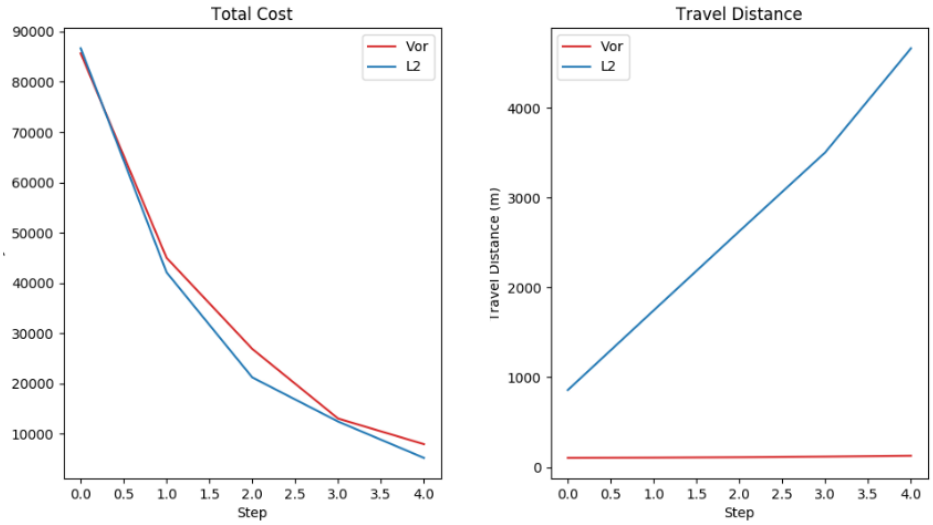


Figure 4.8: (a) Retopography times and (b) Pathfinding solution times for various $|\boldsymbol{R}|$ using Python's `networkx` and `pathfinding` libraries. Targets are selected either as random points in $\mathcal{P}$ that are not necessarily reachable from the robot's position, or as known reachable points. Retopography in this sense constitutes any restructuring of the underlying graph or map.

4.5.3 MRTA for continuous observations

We validate our proposed method by comparing our results to DARP [260], a boustrophedon cell decomposition-based multi-robot coverage planner that can be run with and without energy constraints. The methods are evaluated by first generating optimal paths using the same input set $(\mathcal{R}, \mathcal{G})$, and calculating $J(\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$ for each using the same simulation (i.e., paths are calculated *a priori* and executed with the same initial conditions). Table 4.1 lists the parameters used in the simulation. We select DARP as



(a) (b)

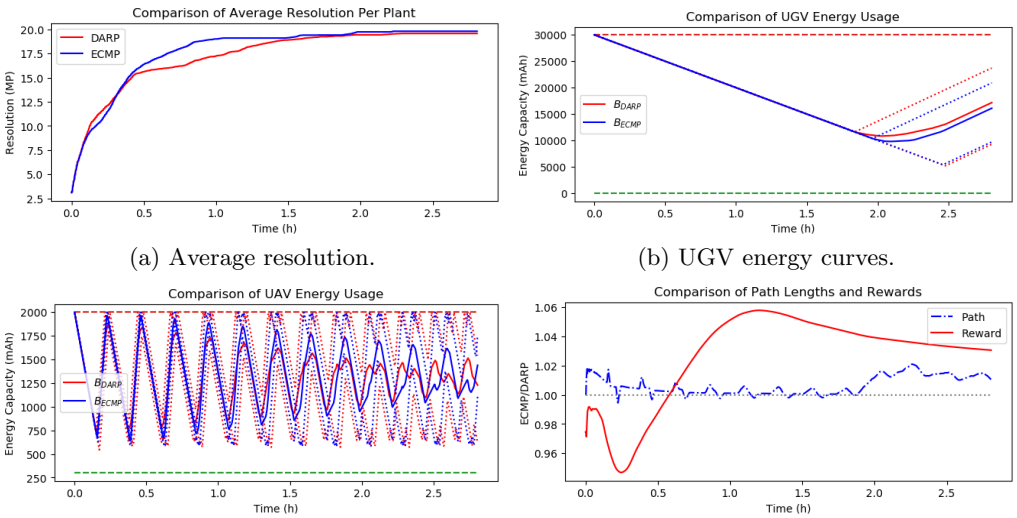
Figure 4.9: Comparison of (a) total cost and (b) total travel distance for the same plot using Voronoi diagrams that use graph distance (red) vs. L2 distance (blue). Both use our greedy allocation method, but with different d_i. Our method travels shorter distances for comparable cost.

Table 4.1: Simulation Parameters

Robot Type	$\underline{B}$ (mAh)	$\underline{P}$ (W)	$\overline{P}$ (W)	h_0 (m)	$\|v\|_0$ (m/s)
UGVs	30000	12	240	0.5	1
UAVs	3000	50	200	2	3

a baseline, and evaluate our method's performance by comparing the following metrics at each t to the baseline: total reward, average resolution per plant, energy usage, path length required to reach 95% maximum theoretical resolution, and overall trajectory differences.

Fig. 4.10 shows a side-by-side comparison of simulation results obtained using a DARP-BCD and the proposed ECMP. When compared to DARP-BCD, the ECMP method achieves similar average resolutions considerably faster with similar path length, since paths are selected for optimal resolution and energy usage, rather than for optimal coverage. Intuitively, the difference lies in how much each robot's sensor data overlaps with one another and their own previous trajectory. Algorithms that maximize coverage tend to waste fuel and time traversing locations that do not affect $\phi_k(w_i)$, and therefore produce sub-optimal trajectories. In contrast, our algorithm calculates minimally overlapping trajectories, such that robots avoid retracing their steps of spending too



(a) Average resolution.

(b) UGV energy curves.

(c) UAV energy curves.

(d) Relative path length and rewards.

Figure 4.10: Comparison of planning metrics for DARP without energy constraints, DARP with energy constraints, and ECMP.

much time on a region that has already been covered at greater resolutions by another robot. Our method also considers the combinatorial aspect of energy sharing between UGVs and UAVs, such that UGV and UAV movements are coordinated to increase the reachable set for each UAV.

Overall, our method achieves comparable resolutions by traversing fewer cells, since steps that do not change $\phi_k(w_i)$ by much are avoided. The optimization also trades short-term sub-optimality for long-term performance, as seen in Fig. 4.10a. Refueling constraints are enforced by the base simulation (independent of the planner) by diverting the robot to the nearest station when energy capacity gets too low.

4.6 Conclusion

This chapter presented a formulation for the multi-robot inspection problem (MRIP) with focus on estimating a metric field spanning the entire plot. An metric inference method was proposed that uses kernel estimations conditioned on multi-robot observations, calculated with a network trained on historical sample data and Gaussian process machine learning. Solution methods for multi-robot task allocation (MRTA) were divided into two as discrete and continuous, depending on how observations were collected and how the method of collection affected the metric calculations. For MRTA

with discrete observations, the kernel estimation method was used to determine optimal sampling points. Sampling tasks were then allocated to each robot by first partitioning the plot into Voronoi regions and then selecting the highest-reward task within each robot's Voronoi region. Finally, whole-field metrics were inferred from a limited number of samples across the entire field using Gaussian process machine learning and a learned kernel. Results were presented using large-scale simulation studies. For MRTA with continuous observations, a quality-optimal planner was described that solves an energy-constrained minimization recast from MRIP by first dividing the plot into energy- and topology-driven partitions, then selecting optimal walks within these partitions that maximize time-dependent capture quality. Simulation studies were performed to validate the approach, and an implementation of the algorithm was compared to similar algorithms found in literature.

The goal of this chapter was to utilize per-plant metrics collected as described in Chapter 3 by individual robots within a multi-robot group to then determine the metrics for an entire plot of crops, and to allocate the associated inspection tasks to this group of robots so that these plot-wide metrics can be collected as efficiently as possible. Some of the methods described here rely on the cooperative control of UGVs and UAVs within this multi-robot group, which is investigated in the next chapter. Further work to tailor the learning methods presented in this chapter are then presented in Chapter 6.

Chapter 5

Cooperative Landing and Takeoff for UAV-UGV Systems

5.1 Introduction

The cooperative inspection tasks presented in the previous chapter are predicated on reliable UGV-based landing and takeoff maneuvers. The UAVs must land on charging platforms installed onto the mobile ground robot, and for many agricultural applications, they must perform the landing maneuvers under large wind disturbances without harming nearby plants. This is a challenging control problem, since the aerodynamic forces involved in these maneuvers are a function of not just environmental conditions around the UAV and UGV, but also of the state and control inputs of the robots. In particular, landing and takeoff regimes are dominated by near-surface aerodynamic effects, which cause the UAV to experience varying lifts and torques based on the geometry of the landing pad and the state and input trajectories leading up to the maneuver. In other words, additional forces and torques depend on what object the UAV is landing on, and how it approaches that object. It is necessary from a safety perspective to develop a predictive and cooperative controller for UAV-UGV landing and takeoff maneuvers. This chapter therefore focuses on model identification for the aerodynamic forces and torques, and a predictive controller and planner for UAV landing maneuvers onto moving platforms.

The main contributions of this work are twofold. First, we present a dynamic model for UAV landing and takeoff regimes that includes the environmental conditions as an explicit state variable. This model includes terms that cannot be derived analytically, so we also propose a learning-based method to predict such terms from state, input, and environmental data. We design a novel network architecture that combines an auto-encoder based latent space representation method with a deep neural network

based thrust and torque prediction method, and we use this architecture to analyze near-surface effects for a multi-rotor UAV. Second, we present a predictive planning and control algorithm based on model predictive control. This algorithm utilizes the aforementioned prediction method to optimize receding horizon control trajectories, which allows the UAV to plan landing trajectories that compensate for predicted aerodynamic effects during landing. The optimization method uses a discretized prediction model to speed up calculations, and allows for dynamic replanning during landing.

The rest of the chapter is organized as follows. A near-surface dynamic model for multi-rotor UAVs is presented in Section 5.2. A learning-based model identification method that estimates and predicts the unmodeled terms from the previous section is proposed in Section 5.3. A trajectory optimization and predictive landing control method is proposed in Section 5.4. Results for these methods are presented in Section 5.5. Finally, a concluding summary is presented in Section 5.6.

5.2 Near-surface dynamic model for multi-rotor UAVs

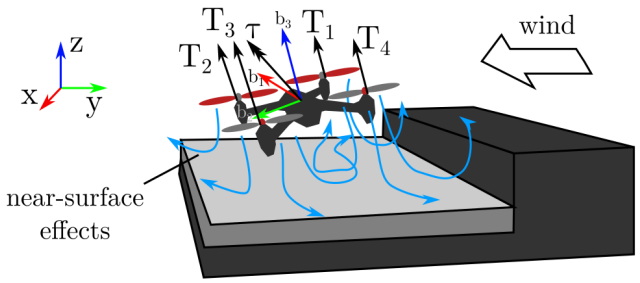


Figure 5.1: A multirotor landing on a mobile platform. Thrust and torques depend on the robot's proximity to any hard surfaces such as the ground.

The forces and torques induced by the rotors when the robot is near a surface such as a wall or floor cannot be fully captured by (2.12) and (2.13). These forces are functions of the velocity of the multirotor, the angular velocities of the rotors, the geometry of nearby surfaces, and other parameters. Considering the multirotor in isolation is thus insufficient to model near-surface thrusts and torques. For landing and takeoff maneuvers in particular, this means that the state $\boldsymbol{x}$ and the control input $\boldsymbol{u}$ alone are

often not enough to describe the dynamics of robots performing aggressive maneuvers in fluids (e.g., due to the Navier-Stokes convection and diffusion terms) [261]. Therefore, $\dot{\boldsymbol{x}}$ is often modeled to include disturbance terms, and in most cases, these disturbance terms are memoryless and only depend on current states and inputs. In [261], we propose the addition of a state-like disturbance term $\boldsymbol{\nu}(\cdot, \boldsymbol{\theta})$ into the dynamics equation $\dot{\boldsymbol{x}}$, and the addition of a corresponding parameter dynamics equation $\dot{\boldsymbol{\theta}}$:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\nu}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}), \tag{5.1a}$$

$$\dot{\boldsymbol{\theta}} = \boldsymbol{\zeta}(\boldsymbol{\theta}, \boldsymbol{u}) + \boldsymbol{\eta}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}), \tag{5.1b}$$

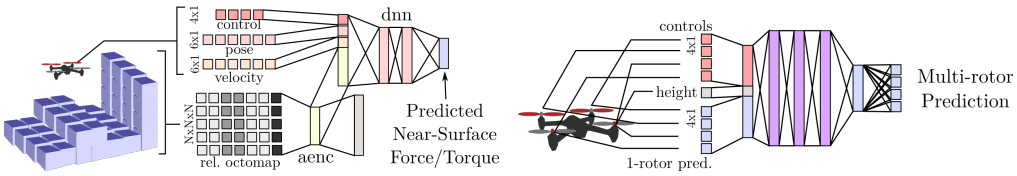
(5.1b) encapsulates any system-disturbance coupling (e.g., through actuation) as well as dynamics intrinsic to the environment (e.g. due to momentum conservation). In this new formulation, the disturbance behaves like a system state rather than an additive disturbance in the classical sense, and the disturbance is therefore termed as *state-like*. This has implications for any optimal landing and takeoff trajectories, as shown in Section 5.4.

For highly-transient state-like disturbances, $\boldsymbol{\nu}$ can be safely divided into a force term f_T and a torque term f_τ, with respective noise terms δ'_F and δ'_τ. For the remainder of the chapter, we only consider large scale aerodynamic effects and relegate high-frequency noise terms to existing disturbance rejection controllers. The unmodeled dynamics terms thus reduce to

$$T_i = f_T(\boldsymbol{x}, \omega_i, \mathcal{G}), \quad \tau_i = f_\tau(\boldsymbol{x}, \omega_i, \mathcal{G}), \tag{5.2}$$

where $\mathcal{G}$ is a 3D representation (e.g., point cloud or voxel occupancy grid) of the quadcopter's immediate environment (i.e., within a couple of rotor diameters). We assume that the hover-state ratios for c_T/c_τ carry over to these new models, such that

$$f_\tau(\boldsymbol{x}, \omega_i, \mathcal{G}) \approx \frac{S_i c_\tau}{c_T} f_T(\boldsymbol{x}, \omega_i, \mathcal{G}). \tag{5.3}$$



(a) Single-rotor network. (b) Multirotor network.

Figure 5.2: Data flow for (a) our near-surface estimation network and (b) our multirotor prediction network. Pose and velocity information is concatenated with latent-space representation of nearby surfaces and passed through a DNN-based prediction network. Single-rotor predictions are then used to predict corrected thrusts.

5.3 Model identification for maneuvers close to the ground

We propose a learning-based method to identify $f_T(\cdot)$ by first reducing $\mathcal{G}$ to a compact latent-space representation in $\mathbb{R}^{N_l}$, where N_l is the size of the latent-space vector, and learning a mapping $\tilde{f}_T : \mathbb{R}^3 \times SU(2) \times \mathbb{R} \times \mathbb{R}^{N_l} \to \mathbb{R}$ to approximate $f_T(\cdot)$ [262]. This mapping captures single-rotor thrusts with near-surface effects (as opposed to the free hover thrust T_0), and is determined using a deep neural network that is trained on experimental data as described in Section 5.5.1. The overall data flow for this method is presented in Fig. 5.2a.

Since rotor-rotor interactions can further modify the thrust and torque profiles of individual rotors, we also propose a method to estimate thrusts and torques for the full multirotor using a correction network. The data flow for the correction network is shown in Fig. 5.2b.

The near-surface model obtained by combining the single-rotor estimation network and the multirotor estimation network form the backbone of our proposed method. The models in (5.2) preserve the underlying dynamics of equation (2.11), meaning they can be used as drop-in replacements in any predictive controllers.

Single-rotor near-surface effects are learned using the neural network architecture shown in Fig. 5.2a, which combines the multirotor state and control inputs with a body-frame representation of the multirotor's immediate surroundings. Neural network based machine learning approaches are practical for this task, as (5.2) is equivalent to a regression problem with multimodal inputs, which are readily solved by neural networks at the cost of less reliable out-of-sample inferences. It is therefore important to collect

enough data for stable predictions before activating the network within any control loops.

To estimate single-rotor near-surface effects, the body-frame point cloud surrounding the robot is first cropped to a $10R \times 10R \times 10R$ axis-aligned bounding box and converted to a $N_o \times N_o \times N_o$ three-dimensional occupancy grid, where N_o is the dimension of one side of the grid. An N_l-dimensional latent space representation of the grid is then calculated using an auto-encoder, and is concatenated with the state and control input vectors. The concatenated vector is then used as in a deep force estimation network. We ensure physically-meaningful latent space representations by including the auto-encoder reconstruction loss as

$$\mathcal{L}_{NSEN} = \alpha_T \mathcal{L}_T + \alpha_R \mathcal{L}_R, \tag{5.4}$$

where $\mathcal{L}_{NSEN}$ is the total loss or the near-surface estimation network, $\mathcal{L}_T$ and $\mathcal{L}_R$ are thrust prediction and reconstruction losses, respectively, α_T and α_R are weighting variables. $\mathcal{L}_T$ is the batch mean-squared prediction error,

$$\mathcal{L}_T(\boldsymbol{T}_i, \boldsymbol{T}'_i) = \|\boldsymbol{T}_i - \boldsymbol{T}'_i\|_2^2, \tag{5.5}$$

where $\boldsymbol{T}_i$ and $\boldsymbol{T}'_i$ are the $N \times 1$ vectors for the ground truth and predicted thrust values for each batch, where N is the batch size, and $\|\cdot\|_2$ is the L^2-norm. The batch reconstruction loss is selected similarly as

$$\mathcal{L}_R = \sum_1^N \left[\frac{1}{N} \sum_{g \in \mathcal{G}} |occ(g) - occ'(g)|^2 \right], \tag{5.6}$$

where $occ(g)$ and $occ'(g)$ are the ground truth and predicted occupancy levels of a cell $g \in \mathcal{G}$, respectively. The selection of N_o and N_l constitute a trade-off between model accuracy, training time, and space complexity.

Single-rotor estimates are extended to the full multirotor system by observing hover-state relations between (ω_i, T_i) pairs. That is, we assume that T_i is affected not just by ω_i, but also ω_j, $j \neq i$, and that this effect is dependent on the motions of the multirotor.

We propose the learning-based method shown in Fig. 5.2b for this extension step.

The loss function for the multi-rotor estimation network is selected as the l^2-norm of the thrust prediction error,

$$\mathcal{L}_{MREN}(\boldsymbol{T}, \boldsymbol{T}') = \|\boldsymbol{T} - \boldsymbol{T}'\|_2, \tag{5.7}$$

where T and T' are the simulated and predicted thrusts, respectively.

We assume that the single-rotor-to-multirotor relation is learnable, and is similar to the empirical results reported by [263], and that the relation is based solely on the predicted single-rotor thrust, the configuration of the multirotor, and the states of the remaining rotors. As such, the multi-rotor network learns a function $f_{MR} : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ that behaves like a time-varying thrust coefficient.

5.4 Trajectory optimization based on predictive controller

The thrust and torque model described in the previous section also enables predictive controller and planner designs. In this section, we first introduce the trajectory optimization problem typical of multirotor UAVs landing on platforms, and we propose an iterative linear time-varying model predictive controller for systems with state-like disturbances such as those observed during landing and takeoff [261].

5.4.1 Trajectory optimization problem

The trajectory optimization described is formalized as the following minimization problem:

$$\min_{x,u} J(x,u) = \psi(\boldsymbol{x},\boldsymbol{u})\Big|_{t=t_f} + \int_{t_0}^{t_f} \phi(\boldsymbol{x},\boldsymbol{u}) \tag{5.8a}$$

$$s.t.\ \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) + \boldsymbol{\nu}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta}) \tag{5.8b}$$

$$\dot{\boldsymbol{\theta}} = \boldsymbol{\zeta}(\boldsymbol{\theta},\boldsymbol{u}) + \boldsymbol{\eta}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta}) \tag{5.8c}$$

$$\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta}) \leq 0 \tag{5.8d}$$

$$\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta}) = 0 \tag{5.8e}$$

$$\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n,\ \boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^m \tag{5.8f}$$

The admissable state and input spaces $\mathcal{X}$ and $\mathcal{U}$ are often described as convex polytopes, or intersections of a set of half-planes in $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively.

The Karush-Kuhn-Tucker (KKT) conditions for optimality can be derived for this modified problem using the principles of variational calculus, and show that the parameter trajectory is a part of the optimal solution along with the state and control trajectories, even though $J(\boldsymbol{x},\boldsymbol{u})$ doesn't explicitly depend on $\boldsymbol{\theta}$.

Lemma 1 *Optimal trajectories for landing maneuvers depend on the airflow around the multi-rotor, and the KKT conditions therefore include state-like disturbances.*

Proof of Lemma 1. First, find the augmented Hamiltonian and take the variation of the Lagrangian, using methodology from [264].

$$\mathcal{H}(\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{u},\boldsymbol{\theta},\dot{\boldsymbol{\theta}},t) = \phi(\boldsymbol{x},\boldsymbol{u},t) + \boldsymbol{\lambda}^{\boldsymbol{x}}(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},t) + \boldsymbol{\nu}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta},t) - \dot{\boldsymbol{x}}) +$$

$$\lambda^{\boldsymbol{\theta}}(\zeta(\boldsymbol{\theta},\boldsymbol{u},t) + \boldsymbol{\eta}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta},t) - \dot{\boldsymbol{\theta}}) + \boldsymbol{\lambda}(\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta},t)) + \boldsymbol{\mu}(\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\theta},t))),$$

$$\mathcal{L}(\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{u},\boldsymbol{\theta},\dot{\boldsymbol{\theta}},t) = \psi(\boldsymbol{x}(t_f),t_f) + \int_{t_0}^{t_f} \mathcal{H}\,\mathrm{d}t,$$

$$\delta\mathcal{L}(\boldsymbol{x},\dot{\boldsymbol{x}},\boldsymbol{u},\boldsymbol{\theta},\dot{\boldsymbol{\theta}},t) = \psi_{\boldsymbol{x}(t_f)}\delta\boldsymbol{x}(t_f) + \psi_{t_f}\delta t_f + \int_{t_0}^{t_f} [\mathcal{H}_x\delta\boldsymbol{x} +$$

$$\mathcal{H}_{\dot{x}}\delta\dot{\boldsymbol{x}} + \mathcal{H}_u\delta\boldsymbol{u} + \mathcal{H}_{\boldsymbol{\theta}}\delta\boldsymbol{\theta} + \mathcal{H}_{\dot{\boldsymbol{\theta}}}\delta\dot{\boldsymbol{\theta}} + \mathcal{H}_t\delta t]\,\mathrm{d}t,$$

$$\delta\mathcal{L}(\cdot) = [\psi_{x(t_f)} - \boldsymbol{\lambda}^{\boldsymbol{x}}]\delta\boldsymbol{x}(t_f) + [\boldsymbol{\lambda}^{\boldsymbol{x}}]\delta\boldsymbol{x}(t_0) + [\psi_{t_f} + \mathcal{H}(t_f)]\delta t_f$$

$$- [\mathcal{H}(t_0)]\delta t_0 + [-\boldsymbol{\lambda}^{\boldsymbol{\theta}}]\delta\boldsymbol{\theta}(t_f) + [\boldsymbol{\lambda}^{\boldsymbol{\theta}}]\delta\boldsymbol{\theta}(t_0) + \int_{t_0}^{t_f} \left[[\mathcal{H}_x + \dot{\boldsymbol{\lambda}}^{x}]\delta\boldsymbol{x} + [\mathcal{H}_{\boldsymbol{\theta}} +\right.$$

$$\left.\dot{\boldsymbol{\lambda}}^{\boldsymbol{\theta}}]\delta\boldsymbol{\theta} + [\mathcal{H}_u]\delta\boldsymbol{u} + [\mathcal{H}_{\lambda^x}]\delta\boldsymbol{\lambda}^{\boldsymbol{x}} + [\mathcal{H}_{\lambda^{\boldsymbol{\theta}}}]\delta\lambda^{\boldsymbol{\theta}} + [\mathcal{H}_\lambda]\delta\boldsymbol{\lambda} + [\mathcal{H}_\mu]\delta\boldsymbol{\mu}\right]\mathrm{d}t.$$

The variations are free, so to satisfy $\delta\mathcal{L} = 0$, we have

$$\boldsymbol{\lambda}^{\boldsymbol{x}}(t_f) = \psi_x(t_f),\ \boldsymbol{\lambda}^{\boldsymbol{\theta}}(t_f) = 0,\ \boldsymbol{\lambda}^{\boldsymbol{x}}(t_0) = 0,\ \boldsymbol{\lambda}^{\boldsymbol{\theta}}(t_0) = 0,\ \mathcal{H}(t_0) = 0,\ \psi_{t_f} = -\mathcal{H}(t_f)$$

$$\dot{\boldsymbol{\lambda}}^{x} = -\mathcal{H}_x,\ \dot{\boldsymbol{\lambda}}^{\boldsymbol{\theta}} = -\mathcal{H}_{\boldsymbol{\theta}},\ \mathcal{H}_{\lambda^x} = \mathcal{H}_{\lambda^{\boldsymbol{\theta}}} = \mathcal{H}_\lambda = \mathcal{H}_\mu = \mathcal{H}_u = 0.$$

The parameter trajectory appears in the following expansions of the above,

$$\dot{\boldsymbol{\lambda}}^{x} = -\left[\boldsymbol{\phi}_{\boldsymbol{x}} + \boldsymbol{\lambda}^{\boldsymbol{x}}(\boldsymbol{f}_{\boldsymbol{x}} + \boldsymbol{\nu}_{\boldsymbol{x}}) + \boldsymbol{\lambda}^{\boldsymbol{\theta}}(\boldsymbol{\eta}_{\boldsymbol{x}}) + \boldsymbol{\lambda}\boldsymbol{h}_{\boldsymbol{x}} + \boldsymbol{\mu}\boldsymbol{g}_{\boldsymbol{x}}\right], \tag{5.9}$$

$$\dot{\boldsymbol{\lambda}}^{\boldsymbol{\theta}} = -\left[\boldsymbol{\lambda}^{\boldsymbol{x}}(\boldsymbol{\nu}_{\boldsymbol{\theta}}) + \boldsymbol{\lambda}^{\boldsymbol{\theta}}(\boldsymbol{\zeta}_{\boldsymbol{\theta}} + \boldsymbol{\eta}_{\boldsymbol{\theta}}) + \boldsymbol{\lambda}\boldsymbol{h}_{\boldsymbol{\theta}} + \boldsymbol{\mu}\boldsymbol{g}_{\boldsymbol{\theta}}\right], \tag{5.10}$$

$$\psi_{t_f} = -\phi(t_f) - \boldsymbol{\psi}_{\boldsymbol{x}}(t_f)\left[\boldsymbol{f}(t_f) + \boldsymbol{\nu}(t_f) - \dot{\boldsymbol{x}}(t_f)\right], \tag{5.11}$$

and thus the state-like disturbance is a part of the optimal trajectory. ■

For a landing planner, the goal is to find an optimal trajectory from an initial state $\boldsymbol{x}_0$ to a final state $\boldsymbol{x}_f$, with $\boldsymbol{e} = \boldsymbol{x} - \boldsymbol{x}_f$, given

$$J(\boldsymbol{x},\boldsymbol{u}) = \|\boldsymbol{e}(t_f)\|_{\boldsymbol{P}} + \int_{t_0}^{t_f} [s + \|\boldsymbol{e}(t)\|_{\boldsymbol{Q}} + \|\boldsymbol{u}(t)\|_{\boldsymbol{R}}]\,\mathrm{d}t, \tag{5.12}$$

where $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are symmetric positive semi-definite weighting matrices for the

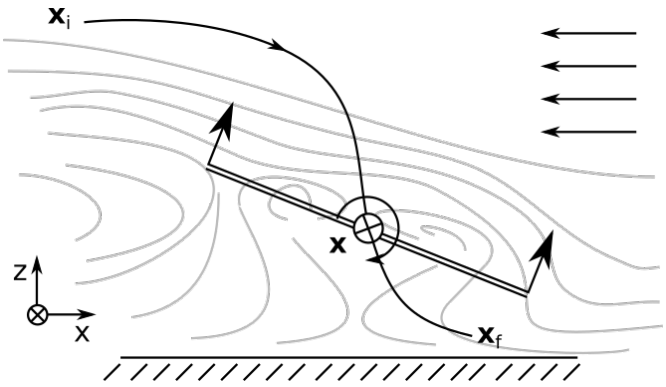


Figure 5.3: Planar quadcopter model with state-like disturbances.

final state, intermediate states, and the control effort, respectively, and s is a constant that defines the weighting of t_f (typically set to 1). We also constrain our states and inputs such that $z(1-\cos(\phi)) < r$ so the quadcopter never crosses the ground plane, and $0 \leq T_1 \leq T_{max}$ and $0 \leq T_2 \leq T_{max}$ so that input saturation is included in the optimization problem.

5.4.2 Trailing Window Disturbance Predictions

For $\boldsymbol{\nu} : \mathcal{X} \times \mathcal{U} \times \Theta \to \mathcal{V}$ that are not injective, there is no simple function $\boldsymbol{\nu}^{-1}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\nu})$ that provides an inverse mapping from $\mathcal{X} \times \mathcal{U} \times \mathcal{V}$ to Θ. However, learning-based methods and model identification for disturbances show that $\boldsymbol{\nu}$ itself is observable, and predictable. We therefore propose to consider and choose the time evolution of $\boldsymbol{\nu}$ for a trailing window $\boldsymbol{V} = \{\boldsymbol{\nu}(t) : t \in [t-T, t]\}$, where T is the duration of the window, such that there exists a function $\gamma : \mathcal{X} \times \mathcal{U} \times \mathrm{dom}V \to \mathcal{V}$ that makes $\boldsymbol{\nu}(\cdot, t+\Delta t) - [\boldsymbol{\nu}(\cdot, t) + \int_t^{t+\Delta t} \gamma(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{V}(\tau)\,\mathrm{d}\tau)]$ vanish for small Δt. We assume that $\boldsymbol{\nu}$ is continuous and observable (even if $\boldsymbol{\theta}$ is not, as discussed above).

We can then replace (5.1a) with a linearized system

$$\begin{bmatrix} \Delta\dot{\boldsymbol{x}} \\ \Delta\dot{\boldsymbol{\nu}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_{\boldsymbol{x}} & \boldsymbol{f}_{\boldsymbol{\nu}} \\ \boldsymbol{\gamma}_{\boldsymbol{x}} & \boldsymbol{\gamma}_{\boldsymbol{\nu}} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\nu} \end{bmatrix} + \begin{bmatrix} \boldsymbol{f}_{\boldsymbol{u}} \\ \boldsymbol{\gamma}_{\boldsymbol{u}} \end{bmatrix} \Delta\boldsymbol{u} + \begin{bmatrix} \boldsymbol{f}_0 \\ \boldsymbol{\gamma}_0 \end{bmatrix}, \tag{5.13}$$

where Δx, Δu and $\Delta\nu$ are perturbations to the nominal state, input and parameter trajectories, respectively, and subscripts denote partial derivatives such that $\boldsymbol{f}_x = \partial \boldsymbol{f}/\partial x$,

etc. Eqn (5.13) captures the state-like behavior of $\boldsymbol{\nu}(\boldsymbol{\theta})$ without having to deal with $\dot{\boldsymbol{\theta}}$. Note that $\boldsymbol{f_0}$ includes the affine gravity term.

The immediate challenge is that $\boldsymbol{\gamma}$ depends on previous states, inputs and parameters along the trajectory, meaning $\boldsymbol{\gamma_x}$, $\boldsymbol{\gamma_\nu}$ and $\boldsymbol{\gamma_u}$ are not constant along perturbations to $(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\theta}(t))$. This suggests the necessity for an iterative approach, which is discussed further in Section 5.4.4.

5.4.3 Linear Time-Varying Model Predictive Control

Linear time-varying model predictive control (LTV MPC) utilizes an updated system model at each prediction step, instead of using a constant model for the entirety of the path like its linear time-invariant counterpart, thus preserving both the computational benefits of linearizing the system and the time complexity of the model. For systems with state-like disturbances, LTV MPC also allows the optimizer to take the effects of the disturbance along the optimal trajectory into account. However, like other MPC methods, prediction errors depend on the model's accuracy, and the computational burden forces the horizon to be limited. If the cost function is quadratic, this burden can be greatly reduced by recasting the optimization problem in (5.8) to a quadratic program (QP). Note that

$$\|\boldsymbol{e}(t)\|_{\boldsymbol{Q}} = \boldsymbol{x_0}^T \boldsymbol{Q} \boldsymbol{x_0} + \Delta \boldsymbol{x}^T \boldsymbol{Q} \Delta \boldsymbol{x} + \boldsymbol{x_f}^T \boldsymbol{Q} \boldsymbol{x_f} - 2\boldsymbol{x_0}^T \boldsymbol{Q} \boldsymbol{x_f} - 2\Delta \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x_f} \tag{5.14}$$

and similarly for $\|\boldsymbol{e}(t_f)\|_{\boldsymbol{P}}$. Any nonlinear constraints can be linearized by assuming that around $(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\nu}_i)$ with $\boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\nu}_i) = \boldsymbol{h}_i$ and $\boldsymbol{g}(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{\nu}_i) = \boldsymbol{g}_i$,

$$\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\nu}) \approx \boldsymbol{h}_i + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \Delta \boldsymbol{x} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{u}} \Delta \boldsymbol{u} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\nu}} \Delta \boldsymbol{\nu},$$

and similarly for $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\nu})$. Now, we define a variable

$$\tilde{\boldsymbol{z}} = [\boldsymbol{x_z}, \Delta\boldsymbol{x_z}, \boldsymbol{x_f}, \boldsymbol{u_z}, \Delta\boldsymbol{u_z}, \boldsymbol{\nu_z}, \Delta\boldsymbol{\nu_z}]^T, \tag{5.15a}$$

$$\boldsymbol{x_z} = [\boldsymbol{x_0}, ..., \boldsymbol{x_{N_P}}]^T,\ \Delta\boldsymbol{x_z} = [\Delta\boldsymbol{x_1}, ..., \Delta\boldsymbol{x_{N_P}}]^T, \tag{5.15b}$$

$$\boldsymbol{u_z} = [\boldsymbol{u_0}, ..., \boldsymbol{u_{N_C\text{-}1}}]^T,\ \Delta\boldsymbol{u_z} = [\Delta\boldsymbol{u_0}, ..., \Delta\boldsymbol{u_{N_C\text{-}1}}]^T, \tag{5.15c}$$

$$\boldsymbol{\nu_z} = [\boldsymbol{\nu_0}, ..., \boldsymbol{\nu_{N_P}}]^T,\ \Delta\boldsymbol{\nu_z} = [\Delta\boldsymbol{\nu_1}, ..., \Delta\boldsymbol{\nu_{N_P}}]^T, \tag{5.15d}$$

where $\boldsymbol{x_i}$, $\boldsymbol{u_i}$, and $\boldsymbol{\nu_i}$ are the nominal states, inputs and disturbances, and N_P and N_C are the prediction and control horizons, respectively. The optimization problem in (5.8) thus becomes

$$\min\ \tilde{J} = \tilde{\boldsymbol{z}}^T\tilde{\boldsymbol{Q}}\tilde{\boldsymbol{z}} + \tilde{\boldsymbol{c}}\tilde{\boldsymbol{x}} \tag{5.16a}$$

$$s.t.\ \tilde{\boldsymbol{A}}\tilde{\boldsymbol{z}} \leq \tilde{\boldsymbol{b}}, \tag{5.16b}$$

where $\tilde{\boldsymbol{c}} = 0$, $\tilde{\boldsymbol{Q}}$ is constructed to satisfy $\tilde{J} \equiv J$, and $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{b}}$ are constructed to satisfy

$$(\boldsymbol{x} + \Delta\boldsymbol{x})_{i+1} = \boldsymbol{A}_{d,i}(\boldsymbol{x} + \Delta\boldsymbol{x})_i + \boldsymbol{B}_{d,i}(\boldsymbol{u} + \Delta\boldsymbol{u})_i + \boldsymbol{C}_{d,i}(\boldsymbol{\nu} + \Delta\boldsymbol{\nu})_i + f_i\Delta t \tag{5.17}$$

$$(\boldsymbol{\nu} + \Delta\boldsymbol{\nu})_{i+1} = \boldsymbol{D}_{d,i}(\boldsymbol{x} + \Delta\boldsymbol{x})_i + \boldsymbol{E}_{d,i}(\boldsymbol{u} + \Delta\boldsymbol{u})_i + \boldsymbol{F}_{d,i}(\boldsymbol{\nu} + \Delta\boldsymbol{\nu})_i + \boldsymbol{\gamma}_i\Delta t \tag{5.18}$$

$$\boldsymbol{u}_i + \Delta\boldsymbol{u}_i = \boldsymbol{u_{N_C\text{-}1}} + \Delta\boldsymbol{u_{N_C\text{-}1}},\ N_C \leq i < N_P \tag{5.19}$$

$$\underline{\boldsymbol{u_i}} \leq \boldsymbol{u_i} \leq \overline{\boldsymbol{u_i}},\ \underline{\boldsymbol{x_i}} \leq \boldsymbol{x_i} \leq \overline{\boldsymbol{x_i}}, \tag{5.20}$$

where $\boldsymbol{A}_d$, $\boldsymbol{B}_d$, $\boldsymbol{C}_d$, $\boldsymbol{D}_d$, $\boldsymbol{E}_d$, and $\boldsymbol{F}_d$ are the discretizations of $\boldsymbol{f_x}$, $\boldsymbol{f_u}$, $\boldsymbol{f_\nu}$, $\boldsymbol{\gamma_x}$, $\boldsymbol{\gamma_u}$ and $\boldsymbol{\gamma_\nu}$ for each timestep, respectively, and an overbar or underbar denote upper and lower limits, respectively. This QP can now be solved with off-the-shelf solvers [265].

5.4.4 Iterative LTV MPC with State-Like Disturbances

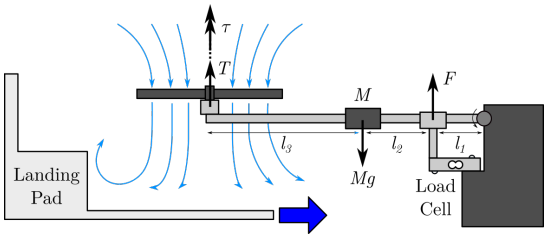
The optimization problem presented in Section 5.4.2 is solved iteratively using the LTV MPC with State-Like Disturbances (SLD) algortihm presented in Algorithm 12 below. Single-shot LTV MPCs are unsuitable in this case, as the SLD trajectory is affected by perturbations to the state trajectory.

Algorithm 12 LTV MPC with SLD

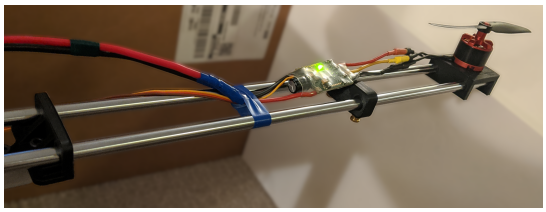
$\{x\}, \{u\} \leftarrow basectrl(A_0, B_0)$
$\text{traj}_{nom}[0] \leftarrow \text{traj}_0 \leftarrow (\{x\}, \{u\}, \{\nu_i, 0, \ldots, 0\})$
$i \leftarrow 0$
while $i = 0$ **or** $\|\text{traj}_i - \text{traj}_{nom}\| < \varepsilon$ **do**
    $x_0, u_0, \nu_0 \leftarrow \text{traj}_{nom}[-1]$
    for $k \leftarrow \{1 : N\}$ **do**
        $\{x_n\}, \{u_n\} \leftarrow \{x_0 \ldots x_{k-1}\}, \{u_0 \ldots u_{k-1}\}$
        $\{\nu_n\} \leftarrow \{\nu_0 \ldots \nu_{k-1}\}$
        $\{\nu_k\} \leftarrow \nu_{k-1} + \gamma(\{x_n\}, \{u_n\}, \{\nu_n\})\Delta t$
        $A_k \leftarrow A_q(\{x_n, x_k\}, \{u_n, u_k\}, \{\nu_n, \nu_k\})$
        $A_k \leftarrow B_q(\{x_n, x_k\}, \{u_n, u_k\}, \{\nu_n, \nu_k\})$
    $\text{traj}_i \leftarrow \text{traj}_{nom}$
    $i \leftarrow i + 1$
    $\text{u}_{nom} \leftarrow ltv\text{-}mpc(\{A_0 ... A_N\}, \{B_0 ... B_N\})$
    $\text{traj}_{nom} \leftarrow rollout(x_0, \{u_n om\}, \nu_0)$

The algorithm uses a base controller (e.g., a PID or LTI MPC) that ignores any disturbances to initialize a nominal trajectory traj_{nom}, and then iteratively improves this trajectory based on an LTV MPC which uses an updated model (5.13). The LTV MPC is designed to solve the QP presented in (5.16), but other solution methods should also work. If a feasible solution to the QP is found, the resultant input trajectory is then used to perform a zero-order hold open-loop rollout starting at the first timestep (x_0, u_0, ν_0). The rollout is then used as the nominal trajectory for the next iteration. If no feasible trajectory is found, the previous nominal trajectory is followed for one timestep, and a new nominal trajectory is generated from there. The time complexity of the algorithm depends on the application, but is multiplicative, as each refinement step requires the algorithm to solve a new optimization that requires N_C linearizations. The algorithm also preserves any stability guarantees of the underlying MPC step, so that the current formulation can be modified to incorporate robust methods.

To ensure successive iterations do not cause unbounded changes in the nominal trajectory and that the linearized system (and cost) is still valid after the update, the trajectories at each step are constrained to a neighborhood of the previous iterations via lower and upper bounds on state, input and disturbance deviations $\Delta\boldsymbol{x}$, $\Delta\boldsymbol{u}$ and $\Delta\boldsymbol{\nu}$. A sketch proof of the convergence and stability of this method is provided below.



(a) Schematic of proposed single-rotor experimental setup.



(b) Assembled single-rotor setup.

Figure 5.4: Experimental setup for thrust measurements. A 5kg load cell attached to a weighted cantilever arm is used to measure rotor thrust without impeding airflow. Polycarbonate sheets are positioned under the rotor to simulate a UGV-mounted landing pad and to change flow conditions.

Lemma 2 *Let C_i denote an MPC controller that can find a feasible (convergent) trajectory from $\boldsymbol{x_i}$ to $\boldsymbol{x_{i+1}}$ for system $(\boldsymbol{A_i}, \boldsymbol{B_i})$, where $\boldsymbol{x_{i+1}}$ is in the neighborhood $N(\boldsymbol{x_i})$. If there exists C_i for all $(\boldsymbol{x_i}, \boldsymbol{x_{i+1}}) \in \mathcal{X} \times \mathcal{X}$ such that $\boldsymbol{x_{i+1}} \in N(\boldsymbol{x_i})$, then a globally convergent iterative MPC controller can be found by using successive controllers $C_0, \ldots, C_N$ that take $\boldsymbol{x_0}$ to $\boldsymbol{x_f}$ if $\boldsymbol{x_{i+1}} \in R(\boldsymbol{x_i}, \mathcal{U}), i = 0 \ldots N\text{-}1$.*

5.5 Experimental results

In this section, we present experimental results for the learning-based disturbance model. We demonstrate the applicability of the predictive planner and controller to multirotor UAV landings through simulation studies.

5.5.1 Single-Rotor Data Collection Setup

While the single-rotor model should ideally be trained on untethered flight data near the landing pad, we propose the single-rotor setup as shown in Fig. 5.4 to speed up data collection for known landing pad geometries. Relative pose data is collected using retro-reflective markers and motion tracking cameras for both methods, but the bench setup allows for continuous operation in exchange for limited rotor mobility. The rotor is mounted onto a slender cantilever arm that minimally impedes airflow, and a load cell is used to record the apparent weight R of the arm assembly. The rotor thrust T

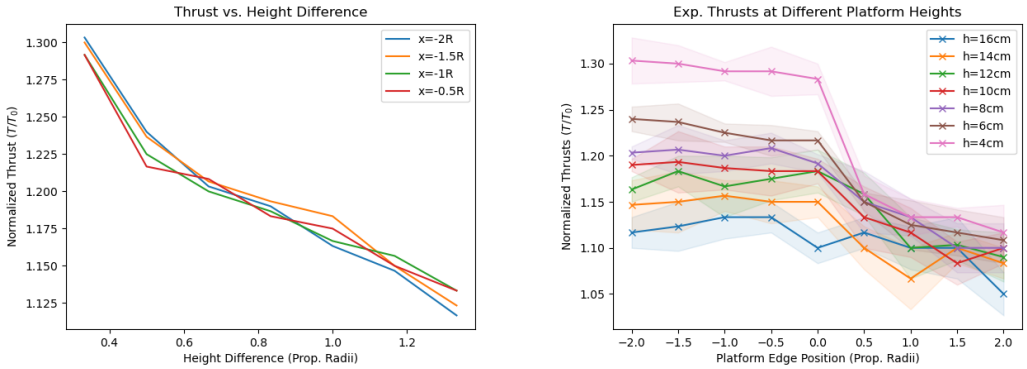
is then calculated as

$$T = \frac{Mg(l_1 + l_2) - Fl_1}{l_1 + l_2 + l_3}. \tag{5.21}$$

The arm is designed to minimize bending by selecting a material with a high Young's modulus (e.g., steel), and torsional stiffness is provided by doubling the rods laterally. Thrust data collected in this manner is used directly as training data, and the relative poses of the rotor and landing pad is used to construct sliding-window occupancy grids using a library such as Open3D [266].

Experimental data is collected by moving a box-like platform to a discrete set of configurations corresponding to a platform-rotor height difference of $h = 4, 6, ..., 16$ cm and a relative x-axis translation of $x = -2R, -1.5R, ..., 1.5R, 2R$, where $R = 12$ cm is the propeller diameter, i.e., $x = -2R$ corresponds to the configuration with the box fully covering the area under the rotor, while $x = 2R$ corresponds to the box leaving the area under the rotor empty. The rotor is then controlled using a microcontroller and electronic speed controller with a constant pulse width modulated signal of around 1300 μs, load cell voltages are converted to force measurements using an HX711 amplifier. The thrust data collected with this setup is shown in Fig. 5.5. Significant ground effects are observed for platform positions corresponding to $x < 0$, where both sides of the rotor are above the platform, with ground effects decreasing as the platform moves away to $x = 2R$. The increases in thrust at different height differences are consistent across $x = -2R, ..., -0.5R$.

Note that non-hover thrust predictions (e.g., during aggressive landing maneuvers) require large amounts of data corresponding to the non-hover states. Additionally, since the aerodynamics of a landing pad moving under a stationary rotor differ from a rotor moving over a stationary landing pad, non-hover data can only be collected by flying the multirotor over the landing pad. In these cases, models trained using static thrust tests can be used as a baseline for incremental training as new state-thrust data pairs are collected. This paper does not study these cases explicitly, but the same network architecture should work with arbitrary states and landing pad geometries, as long as



(a) Thrust vs height.

(b) Thrust vs platform position.

Figure 5.5: Experimental thrusts for different configurations.

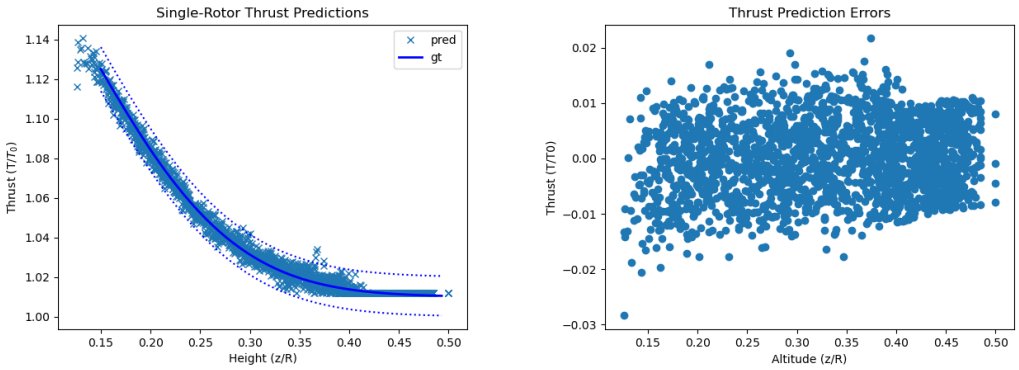
enough data can be collected.

5.5.2 Near-Surface Effect Prediction Results

We validate our proposed learning-based method by predicting single rotor thrusts across a dataset of 20000 samples that follow in-ground-effect thrust models reported by [267]. We split the data into training, validation and testing sets using a 80/10/10 ratio. The single-rotor and multi-rotor networks are both implemented in TensorFlow/Keras and are trained until the total losses described in (5.4) and (5.7) converge.

Thrust predictions for a single 10 cm rotor are shown in Fig. 5.6a. The prediction errors follow the ground truth thrust curve closely, and are generally within the variance of the training target dataset. Predictions are saturated at the lower end of the thrust range due to clipping observed in the voxel volume as the platform moves lower and eventually exits the bounding box. Prediction errors remain approximately the same in the saturated region, but the grid volume and N_o can be increased at the expense of training and prediction speeds to reduce clipping. Fig. 5.7 shows examples of input landing pad geometries alongside corresponding reconstructed occupancy volumes.

The single-rotor network and thrust models are also validated experimentally using the data described in Section 5.5.1. We train the single-rotor network on augmented thrust data corresponding to rotations of the voxel grid along the z axis. A heatmap of normalized errors corresponding to each experimental configuration is shown in Fig. 5.9.



(a) (b)

Figure 5.6: (a) Single-rotor thrust predictions. Predicted thrusts (light blue cross marks) follow the ground truth thrust distribution (mean at solid dark blue line), and (b) Single-rotor thrust errors corresponding to the per-sample thrusts shown in (a). Error distributions remain roughly constant across test heights.

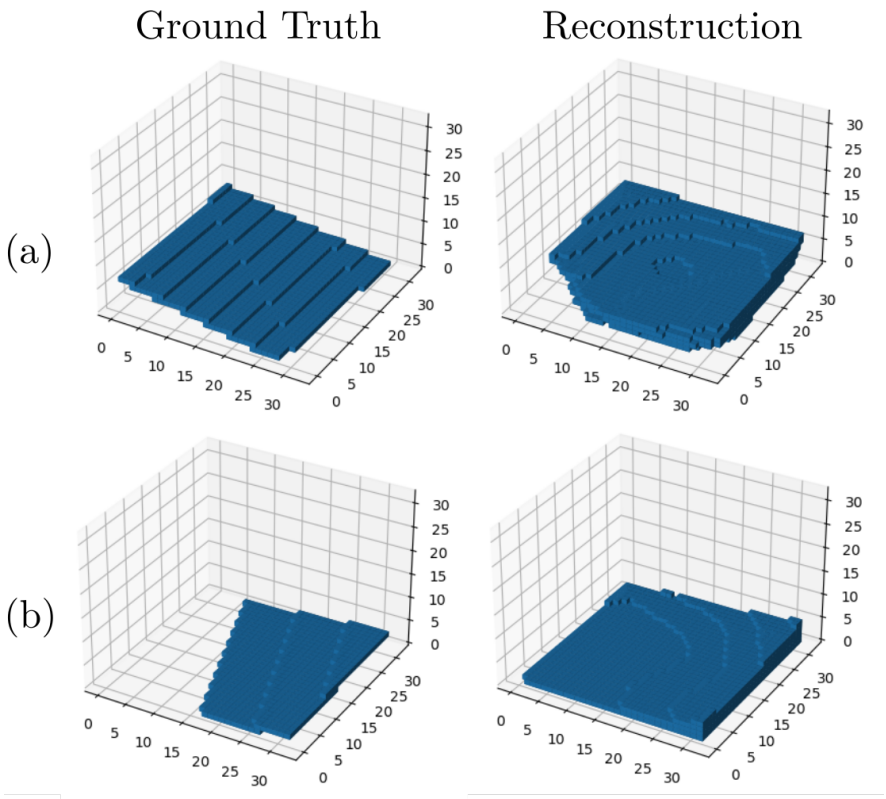


Figure 5.7: Ground truth (left) and reconstructed (right) landing pads. Note the direction and height differences between pads (a) and (b).

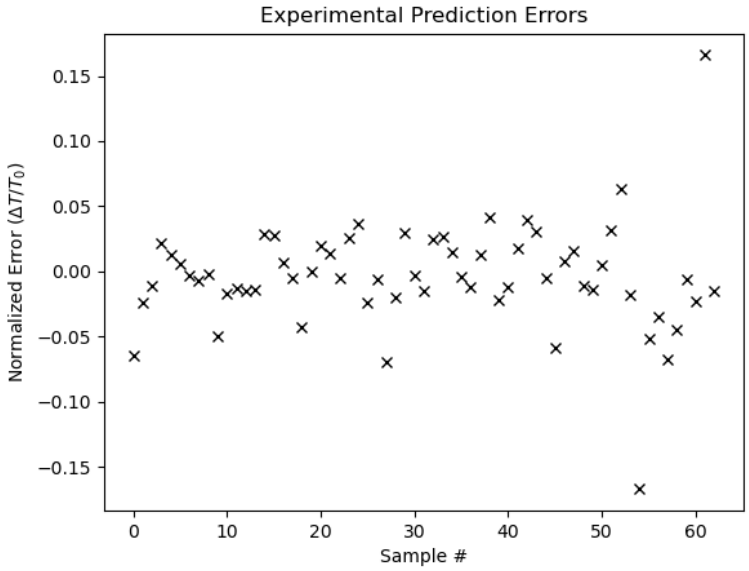


Figure 5.8: Normalized prediction errors for the trained single-rotor network.

The root mean squared errors for the single-rotor network are around 4% ($RMSE = 0.0408$).

The single-rotor network is also fast enough for real-time control applications. In our tests, we were able to get around 0.4 ms forward passes on a 559,769 parameter model corresponding to a $32 \times 32 \times 32$ grid on a laptop with an Intel i7-7700HQ processor and Nvidia GeForce GTX 960M graphics card with CUDA acceleration. The training and inference time complexity is investigated by comparing per epoch training times and average batch inference times. Fig. 5.10 shows that training time increases cubically with N_o and inference times are constant, while N_l does not affect training or inference times significantly.

Single-rotor thrust data is then corrected for multi-rotor validation using a model similar to the one in [263]. The network is trained using height, control input and uncorrected thrust data as inputs and corrected thrusts as outputs. Thrust predictions are shown in Fig. 5.11. Unlike the uncorrected thrusts, the corrected thrusts do not form a one-to-one function (i.e., multiple height values produce the same thrust). Consequently, the correction cannot be learned from rotor velocities and predicted single-rotor thrusts alone, and the network must include a height input.

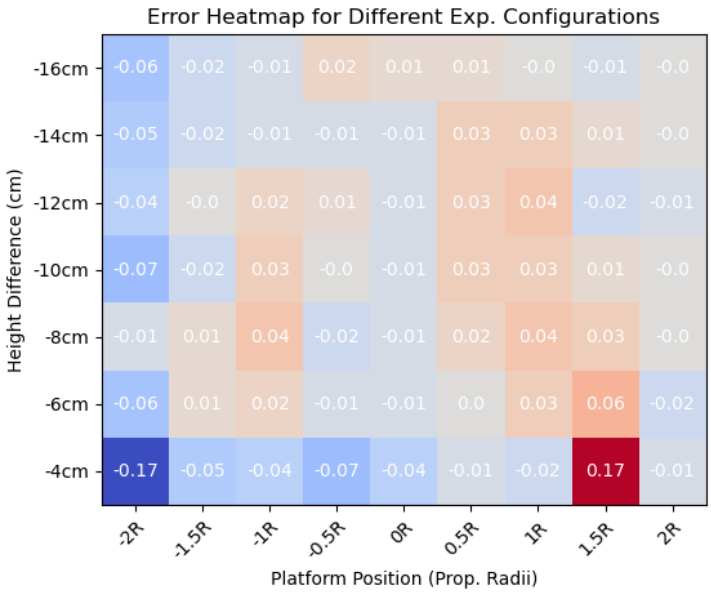


Figure 5.9: Heatmap of normalized experimental errors for various platform configurations.

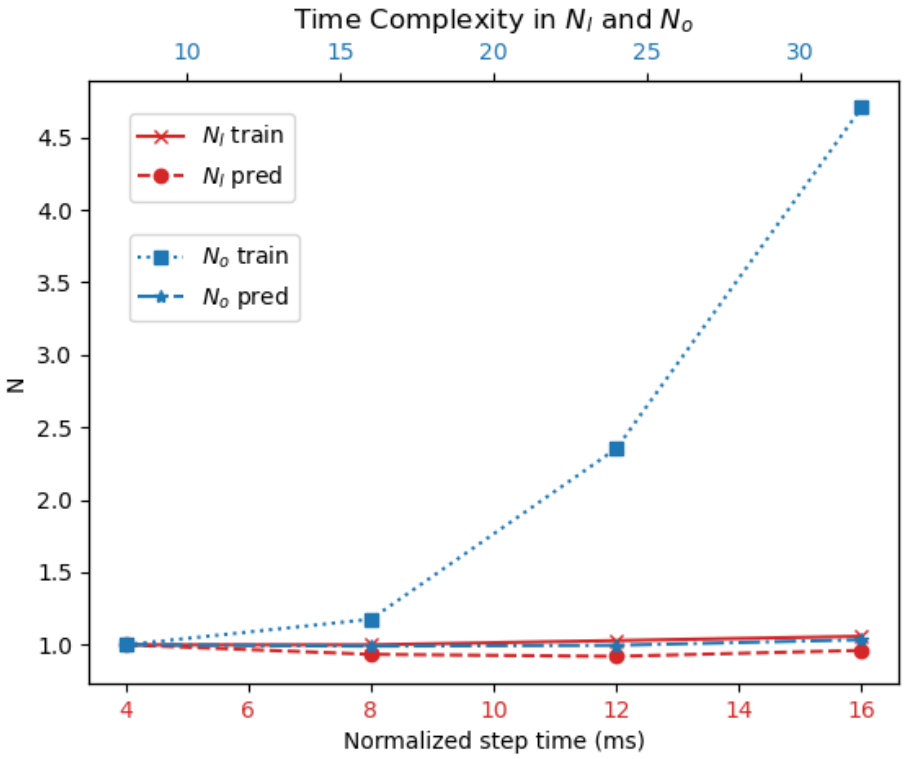
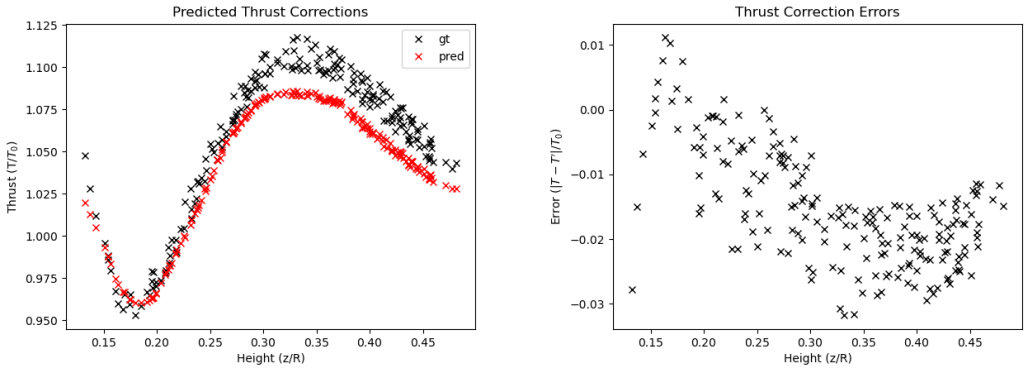


Figure 5.10: Time complexity for the prediction network with varying N_o.



(a) Thrust predictions.

(b) Prediction errors.

Figure 5.11: (a) Corrected thrust predictions for multi-rotor estimation network. (b) Prediction errors for the multirotor thrust network.

5.5.3 Trajectory Optimization Results

A simulation study was performed where the quadcopter was asked to land at the origin. Disturbances were set to zero at the starting point and allowed to evolve according to (5.13). To better assess performance, our MPC-based method was compared to a cascaded PID first tuned to minimize settling time, then tuned to minimize overshoot. The resultant trajectories are shown in Fig. 5.12. The time evolution of each trajectory is shown in Fig. 5.13. The trajectories were recorded starting from the same initial conditions, up to the point where the quadcopters touched the ground (or slightly went through it in the case of the PID tuned to minimize the settling time). Overall, the LTV MPC based approach outperformed the cascaded PIDs in both speed and convergence properties. This is likely due to the LTV MPC with SLD approach calculating optimal trajectories for not only the state and the input but the disturbance as well. This allows the optimizer to move in directions where a change in the disturbance decreases trajectory cost. Specifically, the Hessian includes derivatives along the disturbance trajectory, which implest that the optimizer considers the effects of $\boldsymbol{\nu}$ when calculating beneficial and detrimental solution directions. Assuming the model is accurate over the planning horizon, the constraints similarly incorporate $\boldsymbol{\nu}$, and naturally provide guarantees and bounds for the disturbances.

Iterative corrections reduce any compounding errors due to the linearization of the

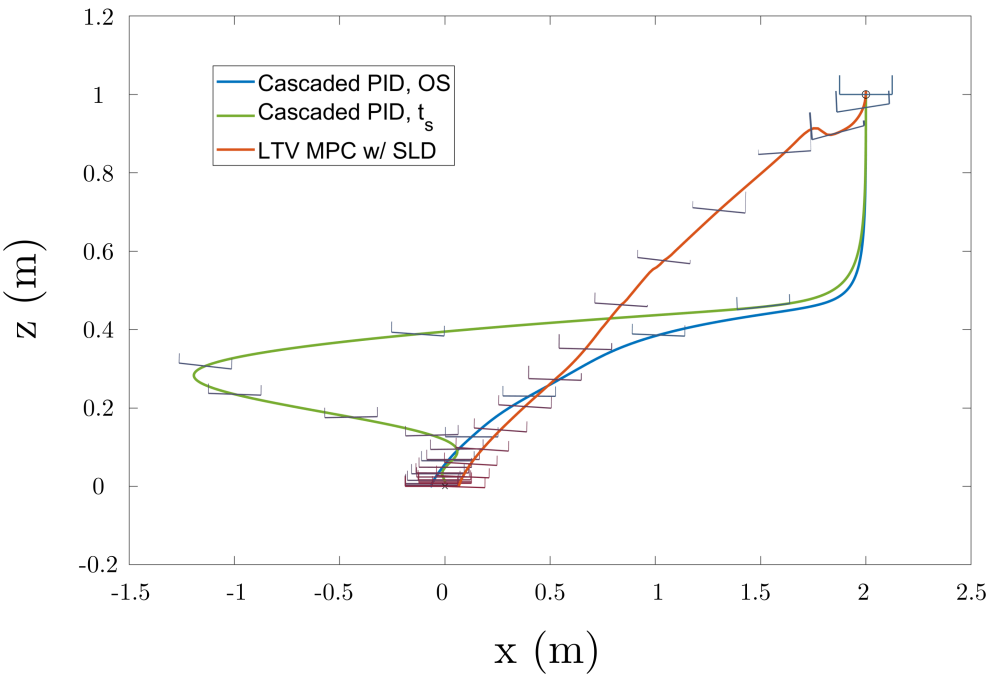


Figure 5.12: Comparison of the trajectories obtained by using a Cascaded PID$_{ts}$ tuned to minimize settling time (green), a Cascaded PID$_{os}$ tuned to minimize overshoot (blue), and our LTV MPC with SLD method (red). The quadcopter and the two inputs are plotted at 15 equitemporal points along each trajectory, with the trajectories encompassing roughly 20, 40 and 6 seconds, respectively. The length of the protruding segments correspond to the thrusts for each motor.

system around the nominal trajectory. The rollout similarly moves near-infeasible nominal trajectories back into the set of physically realizable trajectories. Fig. 5.14 shows an illustration of the process. At each step, the trajectory is updated based on the results of the previous optimization by performing a rollout. This rollout then determines the constraints and nominal trajectory for the next optimization.

Like most numerical optimization methods, the proposed method also requires some fine-tuning and meta-optimization to ensure good performance. Particularly, the cost weighting matrices must be selected carefully to ensure convergent and desired behavior. Some selections of $\boldsymbol{Q}$ and $\boldsymbol{R}$ can move the solution toward regions of instability, causing the QP method to fail.

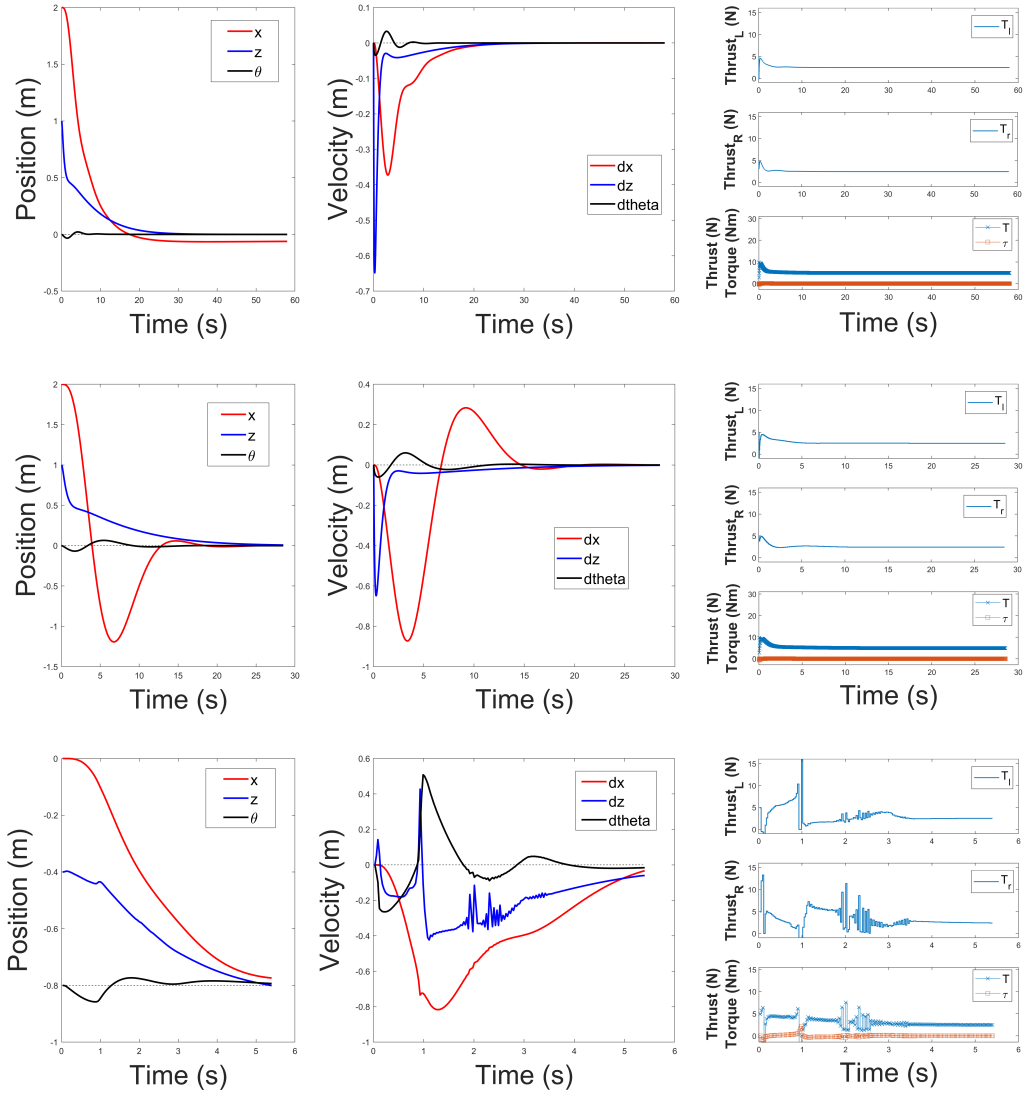


Figure 5.13: Comparison of state and input trajectories for PID_{os} (top), PID_{ts} (center), and LTV MPC w/ SLD (bottom). Note that the timescale for each set of plots is different.

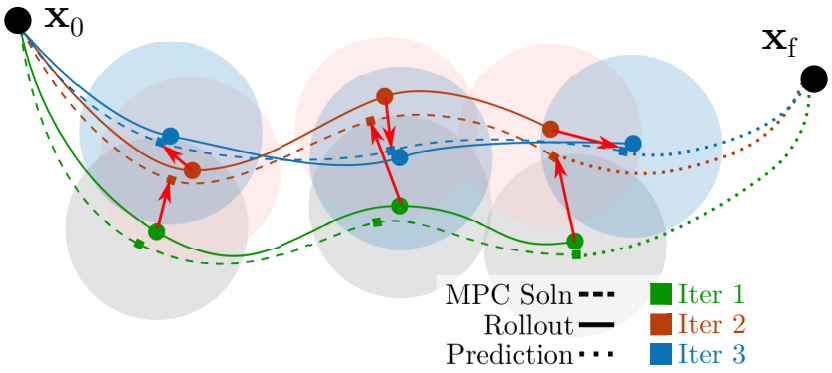


Figure 5.14: Iteration process for trajectory correction. At each step, an MPC solution is used to perform a rollout. The rollout is then used to constrain and initialize then next iteration. Corrections are made to not only the $\boldsymbol{x}$ trajectory shown here, but also to the $\boldsymbol{u}$ and $\boldsymbol{\nu}$ trajectories (not shown).

5.6 Conclusion

This chapter presented planning and modeling approaches for cooperative landing and takeoff manuevers for UAV-UGV groups. A near-surface dynamic model was proposed for multi-rotor UAVs landing on moving platforms. The unmodeled aerodynamics caused by the interaction of rotor-induced flows and landing pad geometries were captured by an additional state-like disturbance term. A model identification method was proposed to learn these disturbance terms from experimental data. The method combined several learning architectures into a novel deep neural network with three parts. The first part was an auto-encoder that converts robot-centered, body frame occupancy grid representations of the environment into a latent space representation. The second part was a single-rotor thrust and torque prediction network that concatenated this latent space representation with the state and control inputs of the multi-rotor. The third part was a multi-rotor correction network that compensated for vortex interactions between rotor-induced flows. The method was validated using simulated models and experiments on a bench-top static thrust tester. A trajectory optimization method based on model predictive control was also presented. The method utilized a predictive disturbance model to determine optimal landing trajectories and control inputs for multi-rotors landing on moving platforms. Simulation studies were used to compare the method to similar multi-rotor control methods.

The goal of this chapter was to solve some of the open questions related to UGV-based landings and takeoffs for agricultural multi-rotors. The methods presented in this chapter are a critical step toward ensuring the safe operation of UAVs in agricultural inspection missions. Specifically, the aerodynamic modeling and trajectory planning algorithms allow UAVs to land among densely-packed crop fields under challenging weather conditions. When combined with a disturbance rejection controller, these methods also pave the way toward UGV-UAV-human interactions for tailored crop inspections, which is investigated in the next chapter.

Chapter 6

Enhanced Inspections through Human-Robot Interaction

6.1 Introduction

Human-robot interactions create many opportunities for robotic crop inspections. On the user side, a well-designed human-robot interaction model can provide an intuitive interface between the complex robotic system and the grower, thus lowering one of the main barriers to robot adoption in agriculture. On the robot side, timely and contextual input from the grower can increase the efficacy of the learning models and clarify desired robot behaviors. The combination of the two sides improves both user experience and system performance. In this chapter, we explore the possibility of using the on-board sensors to enhance the inspection capabilities of the proposed platform through human-robot interactions. We focus on two aspects in particular: gesture-based inspection suggestions, and a human-in-the-loop learning model for more accurate inspections.

We first imagine a user guiding the platform via natural pointing gestures. This type of interaction is very intuitive for humans, but involves complex computer vision problems for robots. We utilize the on-board scanning sensors (e.g., RGB-D and lidar) to estimate the skeletal pose of the user and to localize them relative to the robot. We then determine whether the user is performing a pointing gesture, and if so, we project the pointing gesture onto a map of the environment to determine where they are pointing. This is then used as an inspection target, and the UGV plans a collision-free path to a nearby non-blocking pose that does not block the passage of other UGVs or humans. Since other UGVs and humans are also operating within the same environment, paths are planned cooperatively within the team of UGVs. Additional optimal poses and paths are calculated for human following and waiting behavior in a similar manner.

We then extend the learning-based planner presented in Section 4.4 to include these

gesture-based suggestions. The selection process for inspection targets is changed to prioritize targets that the user points to, and the learning model is updated to include a supervised labeling step if the robot is not able to identify any anomalies at that target. An updated software architecture is also presented for such a system. Steps that are necessary to fully integrate humans into the inspection loop are also discussed within the presented framework.

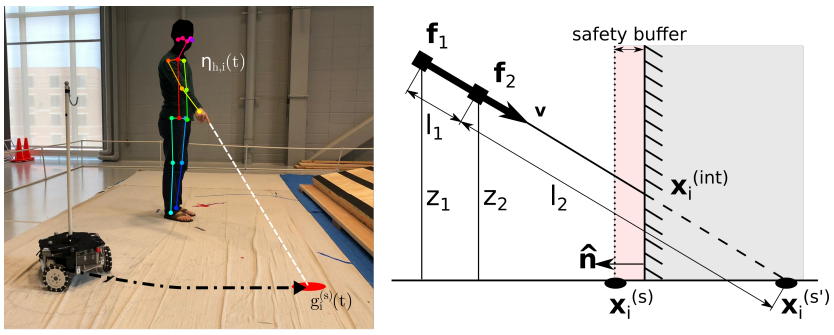
The rest of the chapter is organized as follows. A gesture-based suggestion method is presented in Section 6.2 for intuitive human-robot interactions during inspection missions. The human-aware path planning and navigation algorithms necessary for close human-UGV operations are described in Section 6.3. Experimental results are presented in Section 6.4. Finally, a chapter conclusion is presented in Section 6.5.

6.2 Gesture-Based Inspection Suggestions

The gesture-based inspection suggestion method in [268] is described as follows. Let $\mathbf{R}$ be the set of N cooperative human-following robots. The pose of robot $r_i \in \mathbf{R}$, $i = 1, ..., N$ at time t is $\boldsymbol{x}_i(t) = [x_i(t), y_i(t), \theta_i(t)]^T$, $\boldsymbol{x}_i(t) \in \mathcal{X}_i$, where $\mathcal{X}_i$ is the feasible pose set for r_i, with dynamics given as $[\dot{\boldsymbol{x}}_i^T(t), \ddot{\boldsymbol{x}}_i^T(t)]^T = \boldsymbol{f}(\boldsymbol{x}_i(t), \dot{\boldsymbol{x}}_i^T(t), \boldsymbol{u}_i(t))$, where $\boldsymbol{u}_i(t)$ are the control inputs for r_i at time t, and the pose goal for robot r_i is $\boldsymbol{x}_i^{(g)} = [x_i^{(g)}, y_i^{(g)}, \theta_i^{(g)}]^T$, $\boldsymbol{x}^{(g)} \in \mathcal{G}_i$, where $\mathcal{G}_i \subseteq \mathcal{X}_i$ is a polytope of goal states. We define $\boldsymbol{h}_i(t) = [\boldsymbol{\eta}_i(t)^T, \boldsymbol{\phi}_i(t)^T]^T$ as the concatenation of the pose vector $\boldsymbol{\eta}_i(t)$ [269] and the gait feature vector $\boldsymbol{\phi}_i(t)$ [270] for the human that r_i is targeting.

The user provides pose suggestions $\boldsymbol{x}_i^{(s)}$ through pointing gestures calculated form $\boldsymbol{h}_i(t)$. Rays are projected from the user's extended arm pose $\boldsymbol{\eta}_i(t)$ onto the environment as shown in Fig. 6.1 if the gait feature vector $\phi_i(t)$ matches a pointing gesture. The desired pose is selected as a feasible pose in $\mathcal{X}_i$ closest to the suggested pose, otherwise, the desired pose is selected as $\boldsymbol{x}_i(t)$ or the trailing pose described above, as set by the designer.

The intersection point $\boldsymbol{x}_i^{(int)}$ is follows the segment between the base $\boldsymbol{f}_1$ and tip $\boldsymbol{f}_2$



(a) (b)

Figure 6.1: (a) Point cloud data is used to estimate user pose and gait. (b) Human pose estimates determine robot pose suggestions by casting a ray from the extended arm onto the environment and selecting the closest feasible pose in $\mathcal{X}_i$. Optimal trajectories are calculated via (6.4) to a pose within a polytope $\mathcal{G}_i$ around the pose suggestion.

of the index finger, with direction vector $\boldsymbol{v} = \boldsymbol{f}_2 - \boldsymbol{f}_1$. The no-hit pose $\boldsymbol{x}_i^{(g')}$ is

$$\boldsymbol{x}_i^{(s')} = \boldsymbol{f}_2 + l_2\boldsymbol{v},\ l_2 = \frac{z_2}{z_1 - z_2}\|\boldsymbol{f}_1 - \boldsymbol{f}_2\|, \tag{6.1}$$

where $\|\cdot\|$ is the Euclidean norm, and the goal pose is selected with safety buffer δ_b as $\boldsymbol{x}_i^{(s)} = \boldsymbol{x}_i^{(s')}$ if $\boldsymbol{x}_i^{(int)} = \boldsymbol{x}_i^{(s')}$, and as $\boldsymbol{x}_i^{(s)} = [x_i^{(int)} + \delta_b n_x, y_i^{(int)} + \delta_b n_y, 0]^T$ otherwise, where $\hat{\boldsymbol{n}} = [n_x, n_y]^T$ is the map normal at $\boldsymbol{x}_i^{(int)}$.

6.3 Human-Aware Path Planning and Navigation

The optimal robot trajectories $\boldsymbol{x}_i(t)^*$ are solutions to a constrained multi-robot trajectory optimization problem,

$$\underset{i=1,\dots,N}{\boldsymbol{x}_i^*(t)} = \underset{\boldsymbol{x}_i(t)}{\operatorname{argmin}} \sum_{i=1}^{N}\left[\int_t^{t+T} \psi(\boldsymbol{x}_i(\tau), \boldsymbol{u}_i(\tau))\,\mathrm{d}\tau\right] \tag{6.2a}$$

$$\text{s.t. } \psi(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t)) = p_i\|\boldsymbol{x}_i(t) - \boldsymbol{x}_i^{(g)}\|, \tag{6.2b}$$

$$\dot{\boldsymbol{x}}_i(t) = \boldsymbol{f}(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t)), \tag{6.2c}$$

$$\boldsymbol{x}_i(t) \in \mathcal{X}_i,\ \boldsymbol{x}_i(t+T) \in \mathcal{G}_i,\ \boldsymbol{u}_i(t) \in \mathcal{U}_i, \tag{6.2d}$$

where T is free and p_i is the priority of the ith robot. The human-following robot problem with positioning suggestions is then defined using the optimization problem

as finding an optimal $\boldsymbol{x}_i(t)$ for each r_i to minimize (6.2a) for the entire robot group $\mathbf{R}$, given predictions for $\boldsymbol{h}_i(t)$. In particular, we wish to find optimal pose goals $\boldsymbol{x}_i^{(g)}$ to calculate $\boldsymbol{x}_i^*(t)$.

The combined planning and control problem is solved in two steps by first planning high-level optimal target positions and trajectories, and then tracking these trajectories using model predictive control (MPC) [268]. The target positions $\boldsymbol{x}_i^{(g)}$ are selected by optimizing desired poses $\boldsymbol{x}_i^{(d)}$ described in (6.3) using the optimization in (6.4), so that each robot either trails $\boldsymbol{h}_i(t)$ by a distance of δ_f, or moves to a waiting position that is within a distance of δ_w from a human-suggested pose, or approaches the human at a distance of δ_a so they can drop off any items.

The desired pose $\boldsymbol{x}_i^{(d)}$ is determined at each time step by a finite state machine (FSM) with *Following*, *Waiting* and *Approaching* states, which selects the corresponding robot pose as

$$\boldsymbol{x}_i^{(d)} = \begin{cases} \boldsymbol{x}_i^{(w)}, \text{ robot\_state=}\mathit{Waiting} \\ \boldsymbol{x}_i^{(f)}, \text{ robot\_state=}\mathit{Following} \\ \boldsymbol{x}_i^{(a)}, \text{ robot\_state=}\mathit{Approaching} \end{cases} . \tag{6.3}$$

State transitions between *Following (F)* and *Waiting (W)* are triggered at a robot-human distance of $R_{f\to w}$ for $F\to W$, and a distance of $R_{w\to f}$ for $W\to F$. Fig. 6.2 shows the transition conditions for each state, and Fig. 6.3 shows a schematic for the $F\to W$ transition. Note that the actual trajectory changes once the $F\to W$ threshold is crossed.

The following pose $\boldsymbol{x}_i^{(f)}$ is the pose trailing the user's trajectory by δ_f. For tractability, user trajectories are stored as a queue so the last pose is at least δ_f along the trajectory, and the penultimate pose is less than δ_f along the trajectory. Conversely, the approach pose $\boldsymbol{x}_i^{(a)}$ is the pose δ_a away from the user along a trajectory from the robot to the user.

At any given time, the robot solves a sub-optimization to determine the best goal

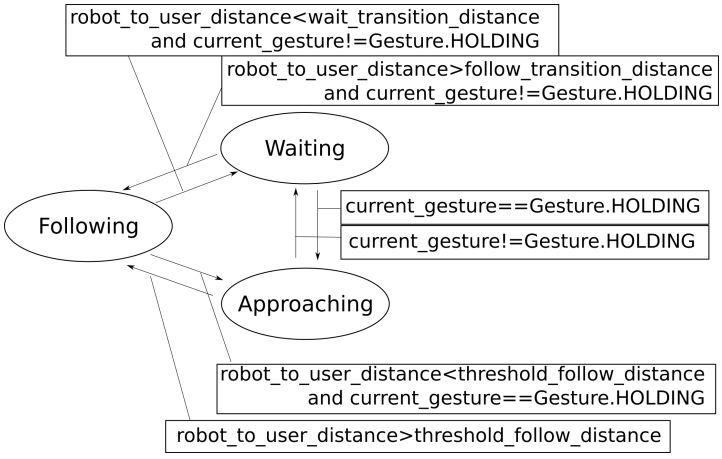


Figure 6.2: The robot's operational mode and its goal pose are determined by a state machine with three states: *Following (F)*, where the robot keeps within δ_f of the human, *Waiting (W)*, where the robot parks itself nearby, and *Approaching (A)*, where the robot approaches to within δ_a.

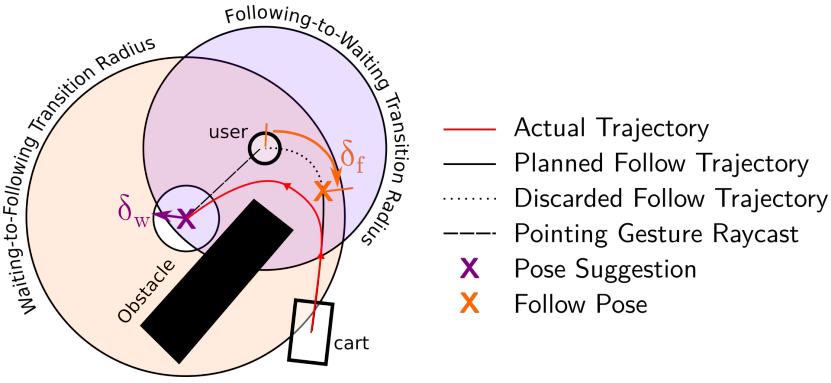


Figure 6.3: The robot transitions from its *Following* state to its *Waiting* state when the robot reaches the following-to-waiting transition radius $R_{f\to w}$, and resumes following by transitioning from its *Waiting* state to its *Following* state when the user reaches the waiting-to-following transition radius $R_{w\to f}$. Switching to the robot's *Approaching* state pre-empts this transition, and instead triggers an approaching-to-following or approaching-to-waiting transition.

pose $\boldsymbol{x}_i^{(g)}$,

$$\boldsymbol{x}_i^{(g)*} = \underset{\boldsymbol{x}_i^{(g)}}{\operatorname{argmin}} \sum_i \|\boldsymbol{e}_s\| + \alpha_2 f_c(\boldsymbol{x}_i^{(g)}) + \alpha_3 f_b(\boldsymbol{x}_i^{(g)}), \tag{6.4}$$

where $\boldsymbol{e}_s = \boldsymbol{x}_i^{(g)} - \boldsymbol{x}_i^{(d)}$ is the error between the goal pose $\boldsymbol{x}_i(t)$ and the desired pose $\boldsymbol{x}_i^{(d)}$, f_c is a pose cost that discourages waiting in poses that coincide with heavy foot traffic, and f_b is a blocking cost that discourages blocking the predicted trajectories of nearby humans. The relative strengths of these three cost components are determined by weighting variables α_2 and α_3.

The pose cost f_c for a pose $\boldsymbol{x}_i$ is determined by querying a heat map constructed from historical human trajectories using 2D kernel density estimation for query point $\boldsymbol{x}_i^{(g)}$ as

$$f_c(\boldsymbol{x}_i^{(g)}) = \frac{1}{|\mathcal{H}|} \sum_{\boldsymbol{h}(t)\in\mathcal{H}} \frac{1}{T_h} \int_{t_0}^{t_f} K_{\mathcal{H}}(\boldsymbol{x}_i^{(g)} - \boldsymbol{h}(t))\, \mathrm{d}t, \tag{6.5}$$

where $K_{\mathcal{H}}(\boldsymbol{x}) = \frac{1}{2\pi}|\mathbf{H}|^{-1/2} e^{-\frac{1}{2}\boldsymbol{x}^T\mathbf{H}^{-1}\boldsymbol{x}}$ is a user-defined bivariate anisotropic Gaussian kernel with covariance matrix $\mathbf{H}$, $\mathcal{H}$ is the set of all past human trajectories, and T_h is the total trajectory time for $\boldsymbol{h}(t) \in \mathcal{H}$.

Since human access to shelves should be prioritized at all times, the robot makes an effort to move out of the way to avoid blocking any humans when it is in a *Waiting* state. This is done by adding a blocking potential f_b to (6.4) as

$$f_b(\boldsymbol{x}_i^{(g)}) = \sum_{\boldsymbol{h}_j\in\mathcal{N}(\boldsymbol{x}_i(t))} \phi_\theta(\boldsymbol{x}_i^{(g)}, \boldsymbol{h}_j) K_b \tag{6.6}$$

that uses a radial basis function (RBF) with user-defined parameter γ_1, such that $K_b = e^{-\gamma_1\|\boldsymbol{x}_i^{(g)}(t) - \boldsymbol{h}_j(t)\|^2}$, to determine the influence of nearby humans in the neighborhood $\mathcal{N}(\boldsymbol{x}_i(t))$ of $\boldsymbol{r}_i$, which is modified by a weighting $\phi_\theta(\boldsymbol{x}_i^{(g)}, \boldsymbol{h}_j) = e^{-\gamma_2 t^2}$ with user-defined temporal attenuation parameter γ_2.

Once the goal pose is determined, the optimization problem in (6.2a)–(6.2d) can be solved in an iterative manner using a multi-robot receding-horizon planner described

as follows. Let $\boldsymbol{z}_i(t) = [\boldsymbol{x}_i^T(t), \dot{\boldsymbol{x}}_i^T(t)]^T$ be the state of the robot at time t, with dynamics $\dot{\boldsymbol{z}}_i(t) = \boldsymbol{f}(\boldsymbol{z}_i(t), \boldsymbol{u}(t))$, discretized around $\boldsymbol{z}_i(t_0)$ as $\boldsymbol{z}_i[k+1] = \boldsymbol{A}_d \boldsymbol{z}_i[k] + \boldsymbol{B}_d \boldsymbol{u}_i[k]$, with discretized matrices $\boldsymbol{A}_d = e^{AT}\boldsymbol{A}$, $\boldsymbol{B}_d = (e^{AT} - I)\boldsymbol{B}\boldsymbol{A}^{-1}$, and $\boldsymbol{A} = \frac{\partial \boldsymbol{f}(\cdot)}{\partial \boldsymbol{z}}|^{\boldsymbol{z}_i(t_0)}_{\boldsymbol{u}_i(t_0)}$, and $\boldsymbol{B} = \frac{\partial \boldsymbol{f}(\cdot)}{\partial \boldsymbol{u}_i(t)}|^{\boldsymbol{z}_i(t_0)}_{\boldsymbol{u}_i(t_0)}$, such that the dynamics are $\Delta\boldsymbol{z}_i[k+1] = \boldsymbol{A}_d \Delta\boldsymbol{z}_i[k] + \boldsymbol{B}_d \Delta\boldsymbol{u}_i[k]$ given $\Delta\boldsymbol{z}_i[k] = \boldsymbol{z}_i[k] - \boldsymbol{z}_i(t_0)$ and $\Delta\boldsymbol{u}_i[k] = \boldsymbol{u}_i[k] - \boldsymbol{u}_i(t_0)$.

An A* search is performed on a coarsely-discretized map at each time step to determine desired trajectory $\boldsymbol{x}_i^{(G)}(t)$, which provides a global trajectory plan. The local trajectory plan $\boldsymbol{x}_i^{(l)}(t)$ for optimization window $t_0 < t < t + T$ is calculated by first placing a moving boundary $\mathcal{W}$ with diagonal corners $[\boldsymbol{x}_i(t_0) - \boldsymbol{W}, \boldsymbol{x}_i(t_0) + \boldsymbol{W}]$, $\boldsymbol{W} = [w_x, w_y]^T$ around $\boldsymbol{x}_i(t_0),\ i = 1, ..., N$ such that $\|\boldsymbol{x}_i(t) - \boldsymbol{x}_i(t_0)\|_1 < \|\boldsymbol{W}\|_1,\ t_0 < t < t_0 + T$, and solving (6.2a)–(6.2d) for this window only, up to $\boldsymbol{x}_i^{(l)}(t_0 + T) = \boldsymbol{x}_i^{(d)}(t_0 + T)$ at $t = t_0 + T$, with tracking error $\boldsymbol{e}_i(t) = \boldsymbol{x}_i^{(l)}(t) - \boldsymbol{x}_i^{(G)}(t)$. We rewrite (6.2a)–(6.2d) as a near-equivalent constrained optimization problem

$$\underset{i=1,...,N}{\boldsymbol{x}_i^{(l)\,*}} = \underset{\substack{\boldsymbol{x}_i(t) \\ i=1,...,N \\ t_0<t<t_0+T}}{\operatorname{argmin}} \sum_{i=1}^{N} p_i \sum_{t=t_0}^{t_0+T} \|\boldsymbol{e}_i(t)\|_{\boldsymbol{Q}} \tag{6.7a}$$

$$\text{s.t. } \dot{\boldsymbol{z}}_i(t) = \boldsymbol{f}(\boldsymbol{z}_i(t), \boldsymbol{u}_i(t), \tag{6.7b}$$

$$\boldsymbol{z}_i^{(l)}(t_0) = \boldsymbol{z}_i(t_0), \tag{6.7c}$$

$$\boldsymbol{x}_i^{(l)}(t) \in \mathcal{W}, t_0 < t < t_0 + T, \tag{6.7d}$$

$$\|\boldsymbol{x}_i^{(l)}(t) - \boldsymbol{x}_j^{(l)}\| > R_r, j \neq i, \tag{6.7e}$$

$$\|\boldsymbol{x}_i^{(l)}(t) - \boldsymbol{h}_j^{(l)}\| > R_h, \tag{6.7f}$$

$$\boldsymbol{u}_i(t) \in \mathcal{U}_i, \tag{6.7g}$$

where $\|\cdot\|_{\boldsymbol{Q}}$ is the vector norm weighted by positive-definite matrix $\boldsymbol{Q}$, and (6.7e)–(6.7f) are collision constraints for robots and humans with distances R_r and R_h, respectively.(6.7) is then recast as a constrained quadratic program (QP) and solved with an iterative linear time-varying model predictive control solver [261], where all matrices are chosen so the QP satisfies the constraint equations in (6.7).

Fig. 6.4 shows the overall data flow for both systems. An algorithmic description of

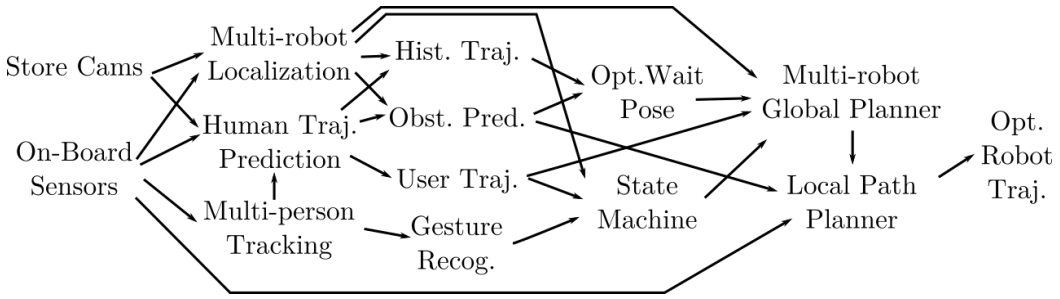


Figure 6.4: Data flows in a similar manner for both the simulation and the experimental platforms. Sensor data is used to perform skeleton tracking, trajectory prediction, localization and local obstacle mapping. Skeleton data is used to determine user gestures, which are then used to determine positioning suggestions. Gesture, trajectory and localization data is used to determine each robot's state as shown in Fig. 6.2. The goal pose is used to plan a global and local optimal trajectory.

the trajectory optimization method is given in Algorithm 13.

Algorithm 13 Multi-robot Receding-Horizon Planner

procedure GENERATE OPTIMAL TRAJECTORY
  $\boldsymbol{x}_i^{(d)} \leftarrow$ DESIREDPOSE($robot.State$) ▷ from (6.3)
  $\boldsymbol{x}_i^{(g)} \leftarrow$ OPTIMIZEPOSE($\boldsymbol{x}_i^{(d)}$) ▷ from (6.4)
  while $True$ **do**
    $\boldsymbol{x}_i^{(l)}(t) \leftarrow$ LOCALWINDOW($\boldsymbol{x}_i^{(G)}(t)$=$A^*(\boldsymbol{x}_i^{(g)})$)
    $\boldsymbol{x}_i^{(l)\,*}(t) \leftarrow$ QP($\boldsymbol{e}_i(t) = \boldsymbol{x}_i^{(l)} - \boldsymbol{x}_i^{(G)}$)
    $\boldsymbol{u}_i(t) \leftarrow$ LOWLEVELCTRL($\boldsymbol{x}_i^{(l)\,*}(t), \boldsymbol{u}_i^*(t)$)

6.4 Experimental Results

In this section, we present experimental results for the various subsystems and algorithms developed in this chapter. First, we give an overview of the experimental setup used to validate the methods. We then present our results for each subsystem in separate subsections for clarity.

6.4.1 Experimental Setup

In this section, we describe the setups for simulation studies performed using a differential drive Turtlebot3 platform, and the experiments conducted using a three-wheeled omni-directional robot. First, a multi-robot environment with rows is simulated in ROS/Gazebo using modified Turtlebot3 platforms. Fig. 6.5 shows one of the layouts

used to perform simulation experiments. All simulated robots are equipped with laser scanners, RGB cameras and Kinect-like depth sensors, and share floor plans and environmental scans.

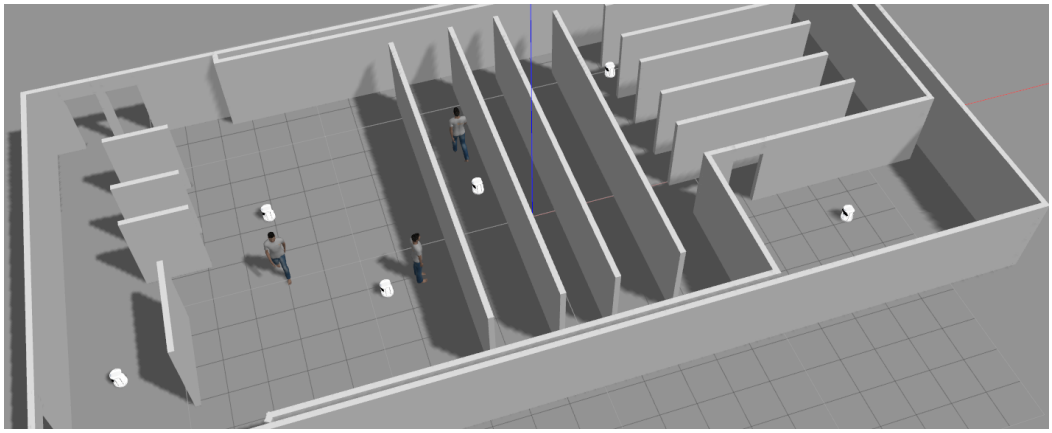
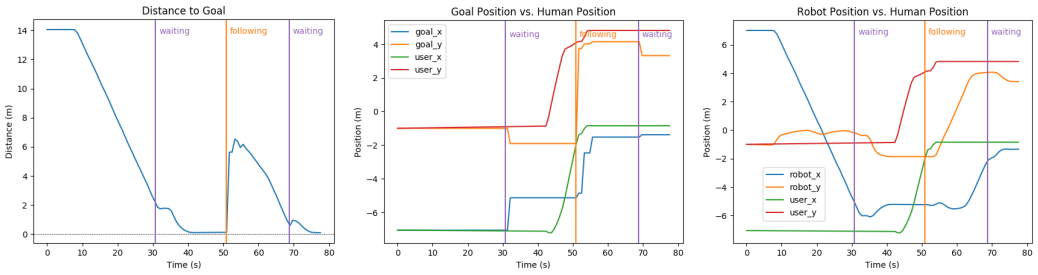


Figure 6.5: An environment is simulated in Gazebo, and includes up to 40 autonomous robots and simulated humans. Several humans can be actively controlled by the user to test pointing and walking behavior, and the rest are simulated using crowd dynamics.

Each robot's local pose estimated via adaptive Monte Carlo localization. Obstacle avoidance and goal selection is done locally (i.e., separately) in a moving window with on-board sensors and shared maps. We implement (6.6) as a costmap for a node that calculates (6.4). Similar to in-store conditions, the robots are given *a priori* information (floor plans and historical trajectories), but perform obstacle avoidance and human tracking online with on-board data. The simulations therefore test state and goal switching, obstacle avoidance and shelf blocking, and multi-robot planning. The experiments are performed with the three-wheel omni-directional robot described in Section 2.4.1, which is modified for human-following experiments as shown in Fig. 2.5.

6.4.2 State and Goal Switching

To test goal switching, the user is first asked to remain stationary while the robot navigates towards them. The user moves to the next aisle once the robot has reached a waiting position. The robot demonstrates state switching behavior by first following the user, then waiting at a position the user points to, then following them again once they move further than $R_{w \to f}$, and finally waiting at a second position determined by the



(a) (b) (c)

Figure 6.6: (a) Plot of the distance between the robot and its selected goal. (b) Comparison of the goal position and the human position. The goal pose initially aligns with the human's pose due to the absence of a trailing path, and is subsequently switched to an optimal waiting pose as the robot crosses $R_{f\to w}$. Following behavior resumes when the human starts moving away, with a goal pose that trails the human's trajectory by δ_f. The robot selects a waiting pose based on user suggestions once it reaches $R_{f\to w}$ again. (c) Comparison of robot and human positions as influenced by distance-dependent state switching.

user. Related metrics are shown in Fig. 6.6. Fig. 6.7 shows the simulation trajectories for a robot following a user through parallel aisles.

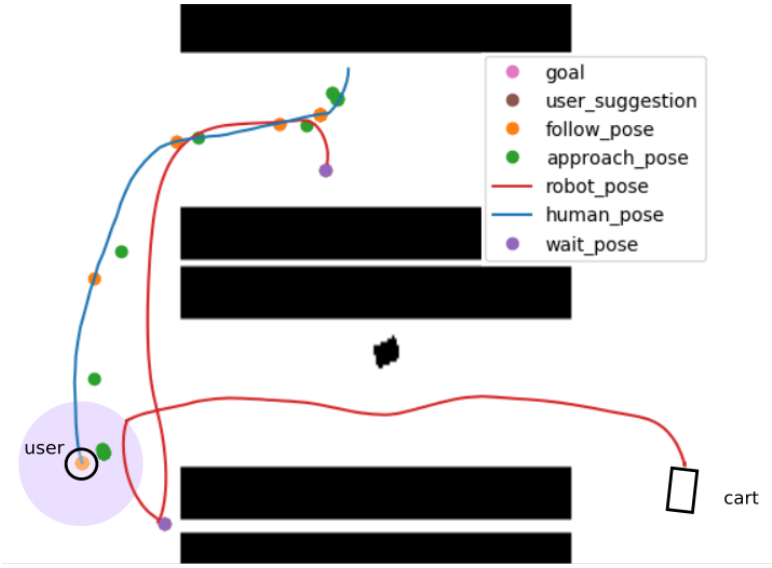


Figure 6.7: Results from a user following test, where the robot was asked to follow the user at a distance of $R_{f\to w} = 1$ m. The robot initially follows the user until it reaches $R_{f\to w}$, and then moves to the suggested waiting pose (purple). The robot resumes following once the user moves further than $R_{w\to f}$.

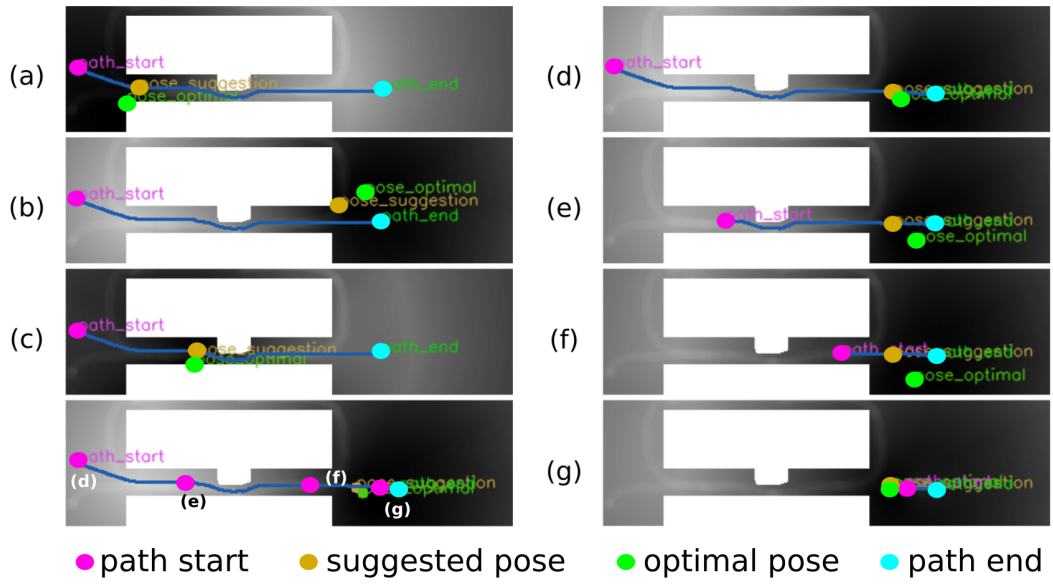


Figure 6.8: Optimal non-blocking poses selected for various scenarios. (a-c) User suggestion is varied along the projected path. (d-g) User approaches and passes robot.

6.4.3 Non-Blocking Pose Selection

Non-blocking pose selection allows the robot to make way for other robots and humans while still waiting close to its user. Non-blocking behavior relies on three costmap layers for L_2 distance cost, the pose cost based on historical robot and human trajectories, and the blocking cost based on the predicted human paths. The minimal cost across the combined costmap provides the optimal waiting position for the robot. The blocking cost converts predicted trajectories into a distance-based penalty that is discounted with time.

The pose cost is constant since the human heatmap is roughly constant over long-term observations, while the distance cost and the blocking cost are highly dependent on the suggested waiting position and the motions of nearby humans. We demonstrate this variability in Fig. 6.8 by varying the suggested waiting positions and the human trajectories.

The behaviors described in this section are tunable by the designer using α_2 and α_3 in (6.4). All figures shown here are calculated using $\alpha_2 = 0.7$ and $\alpha_3 = 0.4$ with a map resolution of 5 cm/pixel, but we note that different robot configurations or supermarket layouts require different αs.

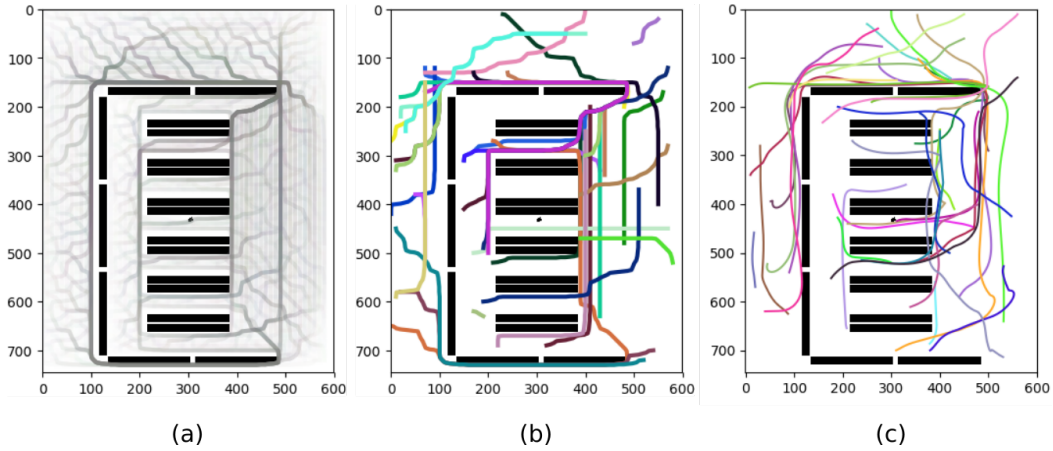


Figure 6.9: (a) Combined global trajectories for 50 people, using data from all trials. (b) Trajectories from a single trial (no robots) colored by person, showing high-activity areas and bottlenecks. (c) Trajectory data from a trial with robots using the multirobot planner. Map resolution is 5cm/px.

6.4.4 Multi-Robot Planning

We validate our approach, with simulations of 50 robots and 50 humans. Fig. 6.9 shows trajectories from 1000 trials with randomized start and goal poses.

We compared the performance of our planner against a scenario where no robots are present, a scenario where robots are present but exhibit pure pursuit behavior, and a scenario where robots plan individual trajectories. The results in Table 6.1 look at three metrics: mean robot-user separation, human wait time, and time to goal. The multirobot planner outperforms other methods when looking at wait time and time to goal metrics, whereas the individual planner has lower mean separation.

Table 6.1: Planner comparison for large-scale multirobot simulations

Metric	Mean Sep. Dist.	Wait Time	Time To Goal
No Robots (Base)	N/A	39.5 s	210.9 s
No Planner	2.41 m	117.3 s	235.0 s
Individual Planner	2.30 m	103.8 s	227.5 s
Multirobot Planner	2.34 m	100.2 s	225.9 s

6.4.5 Gesture-Based Suggestion Tracking

The gesture-based suggestion tracking algorithm was validated experimentally using the three-wheel omni-directional robot in Fig. 2.5. Pose suggestions were extracted

from human finger estimates using RealSense D435i RGB-D cameras, transformed into the world frame, and sent to the conebot at a rate of 20 Hz via wifi for online tracking. Pose suggestions were filtered with a Savitzky-Golay filter (1.5 s window, 3rd order) to reduce the effect of occasional jumps in finger tracking. Tracking errors are shown in Fig. 6.10 and were on average around 0.7 m across all trials and were calculated using the instantaneous pose suggestions. Practitioners are advised to filter finger estimates to reduce errors due to jitter.

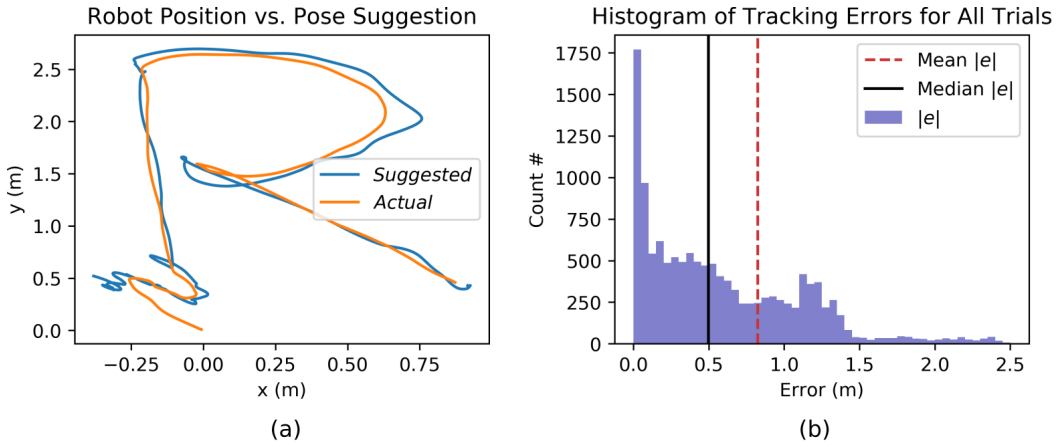


Figure 6.10: (a) Experimental gesture-based suggestion tracking results for a single trial. (b) Histogram of normed tracking errors across all trials, with mean error (dashed red) median error (solid black).

6.5 Conclusion

This chapter presented several developments relating to human-robot interactions within the agricultural domain. First, a human-robot interaction method based on pointing gestures was presented. This method utilized the on-board vision sensors (i.e., the RGB-D cameras, lidars, and 2D cameras) to detect the user and extract a skeletal pose using machine learning algorithms. The skeletal pose was then analyzed to determine whether the user was pointing at anything within the robot's vicinity, and if so, the pointing gesture was converted to an inspection target pose suggestion, corresponding to a location the user wanted the robot to inspect. The finger-tracking algorithm was validated experimentally using an omni-directional robot, and tracking results were compared to ground-truth suggestions collected using a motion capture system. Next,

a behavioral algorithm was introduced based on a three state FSM, which prescribed desired robot actions when following the user, moving in the inspect a target, or waiting for the user to complete manual inspections. Based on the FSM state, the robot then planned a human-aware, collaborative path to an optimal target robot pose such that any trajectories or target poses were non-blocking and collision-free. These behaviors were verified in simulation, and results were reported for state switching, pose selection, and multi-robot planning. Finally, the previous algorithms were situated within the larger robotic inspection framework. A human-in-the-loop learning method was proposed to utilize the gesture-based suggestions, and the learning methods proposed in previous chapters were extended to include the user's expert knowledge and suggestions. Preliminary results were demonstrated in simulation.

The goal of this chapter was to incorporate expert users into the autonomous robotic inspection framework. We demonstrated that the visual data necessary for this was already available to the robotic system through the selected on-board sensors, and that state-of-the-art machine learning methods could be used to extract human skeletal poses, thus providing a human-robot interaction method based on natural gestures (e.g., pointing). The planning algorithms presented in earlier chapters had to be extended to include human-awareness, which allowed for safe operation around the user, especially for human-in-the-loop training of the learning models. The framework extensions and algorithms presented in this chapter lay the groundwork for more effective, human-guided autonomous crop inspections.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this dissertation, we presented an autonomous system for crop inspections with specific focus on robotic phenotyping, planning and control. We first investigated the overall framework necessary to handle the autonomous collection of phenotyping data at large scales using a mobile manipulator based vision system. While there are many open questions associated with such a system, several technical challenges were identified as gaps in the state of the art and were chosen as the focus of this dissertation. Each of these challenges constituted one chapter of the thesis and investigated the development of a robotic phenotyping platform, a novel 3D hyperspectral scanning method, a multi-robot planning and allocation method, the cooperative control of UGV-UAV groups, and human-robot interactions. The cumulative work presented in these chapters therefore formed the basis for the proposed crop inspection system.

In Chapter 2, an architectural description of the framework and software was presented. An experimental platform was proposed for close-up inspections of row crops, and a simulation package was developed for this platform using the Robotic Operating System. The proposed platform is unique within the agricultural domain in that it consists of a single-track robot base with a light-weight 6 degree of freedom robotic arm with a custom camera mount and gripper end effector. A wireless ground probe was designed for use with the gripper, which allows the robot to take soil and atmospheric measurements.

In Chapter 3, a formal definition for the inspection task was presented, and a 3D hyperspectral reconstruction method and a next-best-view algorithm were introduced.

The reconstruction method was developed to utilize the fast capture times and accuracy of RGB-based multi-view photogrammetry. RGB images were collected at several viewing angles from several cameras, and the hyperspectral surface for the inspection target was generated using a learning-based reconstruction method. A novel information gain metric was derived for 3D hyperspectral reconstructions, which allowed the comparison of two hyperspectral point clouds to determine the change in reconstruction quality. This information gain metric was then used to determine optimal viewing angles for the cameras to speed up reconstruction and decrease capture times. The 3D hyperspectral reconstruction method was tested on simulated and real targets. A hyperspectral classification method was also introduced, and validated using growth chamber data for heat-stressed fescue grass.

In Chapter 4, the inspection problem was extended to multi-robot groups consisting of UAVs and UGVs, and the corresponding planning and task allocation problems were investigated. The multi-robot inspection problem was formulated as a least-squares optimization of the estimation error across the entire plot of crops, and an estimation method based on Gaussian process machine learning was proposed in which observations collected by the multi-robot group were used to construct a learning-based, site-specific and temporally-consistent similarity kernel. A task allocation method was proposed to leverage the learned kernel by first dividing the plot into robot-centered Voronoi regions and then assigning tasks to each robot within these regions. The planning problem was proven to be NP-hard, so an efficient, greedy allocation method was proposed to find sub-optimal solutions. An extension of this method was also proposed for multi-robot groups collecting continuous observations (i.e., capturing data as they move across the plot). The extended method also considered the energy limitations of the robots, and minimized the inference uncertainty for the entire plot while also ensuring the robots followed optimal recharge cycles. Simulation studies were performed for both allocation methods, and the estimation method was tested with simulated data on a real plot.

In Chapter 5, the cooperative control of UGVs and UAVs was investigated within the context of multi-robot inspection problem. The UGV-UAV landing problem was considered for outdoor environments. The complex aerodynamics resulting from the

interaction of the environment and the airflow generated by a multirotor were modeled using a learning-based method. A body-frame occupancy grid of the environment was converted to a latent-space representation and concatenated with the state of the robot, and sent to a deep neural network that predicted disturbance thrusts and torques. The learned aerodynamic model was compared to experimental data from a static thrust testbench. A disturbance-aware model predictive controller and planning algorithm based on the learned aerodynamic model was developed for multirotors landing on mobile platforms. The model was verified in simulation and compared to other common multirotor controllers.

Finally, in Chapter 6, human-robot interactions for agricultural inspection systems were investigated. A method to augment robotic inspections with an intuitive, gesture-based interaction method was proposed. The method utilized the on-board cameras to determine the pose and gait of experts working alongside the robot, allowing them to point at certain inspection targets that should be prioritized by the robot. The method was tested experimentally for a single robot, and multi-robot simulations were used to demonstrate how the method could be used at scale. Additional planning and path-finding methods were presented for UGV teams working alongside humans, with specific focus on safety and non-blocking behavior (e.g., keeping the aisles clear so other humans and robots can pass). A human-following algorithm was developed to enable active learning methods where the robot acts as an apprentice to an expert (similar to learning from demonstration).

7.2 Future work

This dissertation is not a comprehensive solution to all robotic crop inspection problems. While the chapters present critical questions and propose viable solutions in many key areas such as multi-robot planning and human-robot interactions, there are innumerable questions remaining within each chapter that would merit theses unto themselves. This section therefore aims to cover only the immediate next steps in achieving fully autonomous crop inspections, but it should be understood that there are many open questions within this domain that are equally as applicable as the ones presented here.

First and foremost, the 3D hyperspectral reconstruction method must be extended for real-time reconstructions, and tests must be performed on real crops in the field. This requires parallel work to be completed for the bikebot base, which is the topic of ongoing research. So far, the reconstruction method has only been tested in lab conditions with ground truth data available through either long-term studies or expensive sensors (such as the imec hyperspectral camera). The method should therefore also be tested with real-world dynamic lighting conditions (e.g., cloudy, rainy days). Scalability of the method can also be investigated for large-sample studies such as the ones performed for fescue grass by the Plant Biology Department on Rutgers Cook Campus. It would make sense to construct a permanent inspection station for such an application so as to streamline the collection of 3D hyperspectral scans for each sample. More data should be collected to investigate phenotyping applications. Algorithms to analyze the captured point clouds can be used to investigate the spatial aspects of plant stress. Target plants with more obvious physical responses to stress can be selected for initial testing (e.g., corn plants, which exhibit leaf curling, rolling or drooping under heat and drought stress).

The analysis performed on hyperspectral reconstructions should also be extended. The available data for healthy/stressed fescue grass can be analyzed further to determine how the classifier's sensitivity and specificity is affected by variations in individual plants. Larger studies can be performed on fescue samples to investigate the statistical significance of the indicators discovered by the deep neural network and to quantify or adjust classification results. The data can be further used to perform feature selection, in which a subset of the original spectra or coefficients are selected to greatly reduce the model size. If this subset can be reduced to a couple of wavelengths while still preserving classification accuracy, then the selected subset can be used to determine a plant-specific vegetation index that can then be calculated using data from cameras tuned to the selected wavelengths (e.g., via optical bandpass filters). This process might also give some insight into which biological mechanisms dominate certain stress responses based on which wavelengths are found to be more important. The study could also be extended to look at multi-class classification using the selected wavelengths.

The multi-robot method can also be tested in more realistic conditions, but such tests would require the construction of several more inspection platforms. A minimal experimental setup involving several UAVs and one more ground robot might be a feasible alternative in the short term. This would allow the testing of the partitioning and plot-wide metric estimation methods with real data over an entire summer season, and would help identify any issues arising from the transition from theory and simulation to real-world applications. Depending on the use case (e.g., turfgrass versus row crop), this might also lead to the development of a reusable model (e.g., to track and predict heat stress) for plant biology research. In any case, the data collection and model training steps must be streamlined for full autonomy.

Future research into UGV-UAV cooperation can go in several directions. Initially, research can focus on running experimental landing and take-off tests to gather a larger dataset to train the aerodynamic model. This will likely also highlight the shortcomings of the network architecture proposed in this work, as the testing and validation was limited to a couple of scenarios under lab conditions. It would be especially beneficial to combine the disturbance-aware planner with some more robust disturbance rejection method and perform the same experiments in windy conditions. There are also gaps in how the UAVs coordinate landing maneuvers with the UGVs, and how several UAVs might land on the same UGV at the same time. These are complex coordination problems that would require robust controllers for both the UGVs and the UAVs, possibly with both vision systems in the loop.

Finally, the human-robot interaction aspect can be extended by developing an active learning method that does not require experts to manually label data collected by the inspection platform. It would be relatively straight-forward to integrate voice recognition into the system to allow experts to give commands to the robot or to verbally label inspection targets they are pointing at. The software engineering challenges for such developments would overshadow any robotics research outcomes, but it would greatly help in lowering the technical barriers that prevent such systems from being used in the real world, and the research outcomes would be a natural extension of the work presented here for human-following robots.

References

[1] "Global Food Crisis (2008) | UNITED NATIONS ECONOMIC and SOCIAL COUNCIL," 2008.

[2] UN, "Food Production Must Double by 2050 to Meet Demand from World's Growing Population, Innovative Strategies Needed to Combat Hunger, Experts Tell Second Committee | Meetings Coverage and Press Releases," 2009.

[3] FAO, "The future of food and agriculture – Alternative pathways to 2050 | Global Perspectives Studies | Food and Agriculture Organization of the United Nations," 2018.

[4] H. Ritchie and M. Roser, "Crop Yields," *Our World in Data*, Oct. 2013.

[5] T. J. Lark, S. A. Spawn, M. Bougie, and H. K. Gibbs, "Cropland expansion in the United States produces marginal yields at high costs to wildlife," *Nature Communications*, vol. 11, no. 1, p. 4295, Sep. 2020.

[6] P. R. Ehrlich and J. Harte, "Opinion: To feed the world in 2050 will require a global revolution," *PNAS*, vol. 112, no. 48, pp. 14 743–14 744, Dec. 2015.

[7] S. S. Nadakuduti, C. R. Buell, D. F. Voytas, C. G. Starker, and D. S. Douches, "Genome Editing for Crop Improvement – Applications in Clonally Propagated Polyploids With a Focus on Potato (Solanum tuberosum L.)," *Front. Plant Sci.*, vol. 9, 2018.

[8] R. W. Bruce, C. M. Grainger, A. Ficht, M. Eskandari, and I. Rajcan, "Trends in Soybean Trait Improvement over Generations of Selective Breeding," *Crop Science*, vol. 59, no. 5, pp. 1870–1879, 2019.

[9] Z. Huang, C. Cheng, Z. Liu, B. Feng, Y. Hu, W. Luo, G. He, X. Yu, and W. Fu, "Highly efficient potassium fertilizer production by using a gemini surfactant," *Green Chem.*, vol. 21, no. 6, pp. 1406–1411, Mar. 2019.

[10] A. Chlingaryan, S. Sukkarieh, and B. Whelan, "Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review," *Computers and Electronics in Agriculture*, vol. 151, pp. 61–69, Aug. 2018.

[11] S. R. Debats, D. Luo, L. D. Estes, T. J. Fuchs, and K. K. Caylor, "A generalized computer vision approach to mapping crop fields in heterogeneous agricultural landscapes," *Remote Sensing of Environment*, vol. 179, pp. 210–221, Jun. 2016.

[12] N. Kim, K.-J. Ha, N.-W. Park, J. Cho, S. Hong, and Y.-W. Lee, "A Comparison Between Major Artificial Intelligence Models for Crop Yield Prediction: Case

Study of the Midwestern United States, 2006–2015," *ISPRS International Journal of Geo-Information*, vol. 8, no. 5, p. 240, May 2019.

[13] T. Duckett, S. Pearson, S. Blackmore, B. Grieve, W.-H. Chen, G. Cielniak, J. Cleaversmith, J. Dai, S. Davis, C. Fox, P. From, I. Georgilas, R. Gill, I. Gould, M. Hanheide, A. Hunter, F. Iida, L. Mihalyova, S. Nefti-Meziani, G. Neumann, P. Paoletti, T. Pridmore, D. Ross, M. Smith, M. Stoelen, M. Swainson, S. Wane, P. Wilson, I. Wright, and G.-Z. Yang, "Agricultural Robotics: The Future of Robotic Agriculture," *arXiv:1806.06762 [cs]*, Aug. 2018.

[14] R. Gebbers and V. I. Adamchuk, "Precision Agriculture and Food Security," *Science*, vol. 327, no. 5967, pp. 828–831, Feb. 2010.

[15] NIFA, "Adoption of Precision Agriculture | National Institute of Food and Agriculture," 2020.

[16] P. Gonzalez-de Santos, R. Fernández, D. Sepúlveda, E. Navas, L. Emmi, and M. Armada, "Field Robots for Intelligent Farms—Inhering Features from Industry," *Agronomy*, vol. 10, no. 11, p. 1638, Nov. 2020.

[17] S. Fountas, N. Mylonas, I. Malounas, E. Rodias, C. Hellmann Santos, and E. Pekkeriet, "Agricultural Robotics for Field Operations," *Sensors (Basel)*, vol. 20, no. 9, May 2020.

[18] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, p. 1729881419839596, Mar. 2019.

[19] K. Tai, A.-R. El-Sayed, M. Shahriari, M. Biglarbegian, and S. Mahmud, "State of the Art Robotic Grippers and Applications," *Robotics*, vol. 5, no. 2, p. 11, Jun. 2016.

[20] M. Sereinig, W. Werth, and L.-M. Faller, "A review of the challenges in mobile manipulation: systems design and RoboCup challenges," *Elektrotech. Inftech.*, vol. 137, no. 6, pp. 297–308, Oct. 2020.

[21] R. V. Bostelman, T. H. Hong, and J. A. Marvel, "Survey of Research for Performance Measurement of Mobile Manipulators," vol. 121, pp. 342–366, Jun. 2016.

[22] K. Zadarnowska and K. Tchoń, "A control theory framework for performance evaluation of mobile manipulators," *Robotica*, vol. 25, pp. 703–715, Nov. 2007.

[23] K. G. Fue, W. M. Porter, E. M. Barnes, and G. C. Rains, "An Extensive Review of Mobile Agricultural Robotics for Field Operations: Focus on Cotton Harvesting," *AgriEngineering*, vol. 2, no. 1, pp. 150–174, Mar. 2020.

[24] S. Yaghoubi, N. A. Akbarzadeh, S. S. Bazargani, S. S. Bazargani, M. Bamizan, and M. I. Asl, *Autonomous Robots for Agricultural Tasks and Farm Assignment and Future Trends in Agro Robots.*

[25] L. Hall, "How NASA and John Deere Helped Tractors Drive Themselves," Nov. 2016.

[26] S. Opiyo, J. Zhou, E. Mwangi, W. Kai, and I. Sunusi, "A Review on Teleoperation of Mobile Ground Robots: Architecture and Situation Awareness," *Int. J. Control Autom. Syst.*, Oct. 2020.

[27] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[28] H. Chao, Y. Gu, and M. Napolitano, "A Survey of Optical Flow Techniques for Robotics Navigation Applications," *J Intell Robot Syst*, vol. 73, no. 1, pp. 361–372, Jan. 2014.

[29] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual Navigation for Mobile Robots: A Survey," *J Intell Robot Syst*, vol. 53, no. 3, p. 263, May 2008.

[30] J. Crespo, J. C. Castillo, O. M. Mozos, and R. Barber, "Semantic Information for Robot Navigation: A Survey," *Applied Sciences*, vol. 10, no. 2, p. 497, Jan. 2020.

[31] Y. D. V. Yasuda, L. E. G. Martins, and F. A. M. Cappabianco, "Autonomous Visual Navigation for Mobile Robots: A Systematic Literature Review," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 13:1–13:34, Feb. 2020.

[32] F. Gul, W. Rahiman, and S. S. N. Alhady, "A comprehensive study for robot navigation techniques," *Cogent Engineering*, vol. 6, no. 1, p. 1632046, Jan. 2019.

[33] M. S. Güzel, "Autonomous Vehicle Navigation Using Vision and Mapless Strategies: A Survey," *Advances in Mechanical Engineering*, vol. 5, p. 234747, Jan. 2013.

[34] G. N. Desouza and A. C. Kak, "Vision for mobile robot navigation: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 237–267, Feb. 2002.

[35] S. Bonadies, A. Lefcourt, and S. Gadsden, "A survey of unmanned ground vehicles with applications to agricultural and environmental sensing," May 2016, p. 98660Q.

[36] S. Bonadies and S. A. Gadsden, "An overview of autonomous crop row navigation strategies for unmanned ground vehicles," *Engineering in Agriculture, Environment and Food*, vol. 12, no. 1, pp. 24–31, Jan. 2019.

[37] K. Zhang, Y. Yang, M. Fu, and M. Wang, "Traversability Assessment and Trajectory Planning of Unmanned Ground Vehicles with Suspension Systems on Rough Terrain," *Sensors (Basel)*, vol. 19, no. 20, Oct. 2019.

[38] O. Liu, S. Yuan, and Z. Li, "A Survey on Sensor Technologies for Unmanned Ground Vehicles," in *2020 3rd International Conference on Unmanned Systems (ICUS)*, Nov. 2020, pp. 638–645.

[39] M. Russo and M. Ceccarelli, "A Survey on Mechanical Solutions for Hybrid Mobile Robots," *Robotics*, vol. 9, no. 2, p. 32, Jun. 2020.

[40] S. Jordan, J. Moore, S. Hovet, J. Box, J. Perry, K. Kirsche, D. Lewis, and Z. T. H. Tse, "State-of-the-art technologies for UAV inspections," *Sonar Navigation IET Radar*, vol. 12, no. 2, pp. 151–164, 2018.

[41] F. Nex and F. Remondino, "UAV for 3D mapping applications: a review," *Appl Geomat*, vol. 6, no. 1, pp. 1–15, Mar. 2014.

[42] U. R. Mogili and B. B. V. L. Deepak, "Review on Application of Drone Systems in Precision Agriculture," *Procedia Computer Science*, vol. 133, pp. 502–509, Jan. 2018.

[43] J. Kim, S. Kim, C. Ju, and H. I. Son, "Unmanned Aerial Vehicles in Agriculture: A Review of Perspective of Platform, Control, and Applications," *IEEE Access*, vol. 7, pp. 105 100–105 115, 2019.

[44] D. C. Tsouros, S. Bibi, and P. G. Sarigiannidis, "A Review on UAV-Based Applications for Precision Agriculture," *Information*, vol. 10, no. 11, p. 349, Nov. 2019.

[45] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges," *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.

[46] H. S. Abdullahi, F. Mahieddine, and R. E. Sheriff, "Technology Impact on Agricultural Productivity: A Review of Precision Agriculture Using Unmanned Aerial Vehicles," in *Wireless and Satellite Systems*. Cham: Springer International Publishing, 2015, pp. 388–400.

[47] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, "A compilation of UAV applications for precision agriculture," *Computer Networks*, vol. 172, p. 107148, May 2020.

[48] C. Stöcker, R. Bennett, F. Nex, M. Gerke, and J. Zevenbergen, "Review of the Current State of UAV Regulations," *Remote Sensing*, vol. 9, no. 5, p. 459, May 2017.

[49] B. Galkin, J. Kibilda, and L. A. DaSilva, "UAVs as Mobile Infrastructure: Addressing Battery Lifetime," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 132–137, Jun. 2019.

[50] S. K. von Bueren, A. Burkart, A. Hueni, U. Rascher, M. P. Tuohy, and I. J. Yule, "Deploying four optical UAV-based sensors over grassland: challenges and limitations," *Biogeosciences*, vol. 12, no. 1, pp. 163–175, Jan. 2015.

[51] H. V. Abeywickrama, B. A. Jayawickrama, Y. He, and E. Dutkiewicz, "Comprehensive Energy Consumption Model for Unmanned Aerial Vehicles, Based on Empirical Studies of Battery Performance," *IEEE Access*, vol. 6, pp. 58 383–58 394, 2018.

[52] K. Fujii, K. Higuchi, and J. Rekimoto, "Endless Flyer: A Continuous Flying Drone with Automatic Battery Replacement," in *2013 IEEE 10th International*

Conference on Ubiquitous Intelligence and Computing and 2013 IEEE 10th International Conference on Autonomic and Trusted Computing, Dec. 2013, pp. 216–223.

[53] M. F. Ballesteros-Escamilla, D. Cruz-Ortiz, I. Chairez, and A. Luviano-Juárez, "Adaptive output control of a mobile manipulator hanging from a quadcopter unmanned vehicle," *ISA Transactions*, vol. 94, pp. 200–217, Nov. 2019.

[54] T. Ikeda, K. Ohara, A. Ichikawa, and T. Fukuda, "Pilot study on control of one DoF manipulator on quadcopter for hammering check," in *2015 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, Nov. 2015, pp. 1–2.

[55] J. Thomas, J. Polin, K. Sreenath, and V. Kumar, "Avian-Inspired Grasping for Quadrotor Micro UAVs." American Society of Mechanical Engineers Digital Collection, Feb. 2014.

[56] H. Bonyan Khamseh, F. Janabi-Sharifi, and A. Abdessameud, "Aerial manipulation—A literature survey," *Robotics and Autonomous Systems*, vol. 107, pp. 221–235, Sep. 2018.

[57] C. Potena, R. Khanna, J. Nieto, R. Siegwart, D. Nardi, and A. Pretto, "AgriColMap: Aerial-Ground Collaborative 3D Mapping for Precision Farming," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1085–1092, Apr. 2019.

[58] K. Asadi, A. Kalkunte Suresh, A. Ender, S. Gotad, S. Maniyar, S. Anand, M. Noghabaei, K. Han, E. Lobaton, and T. Wu, "An integrated UGV-UAV system for construction site data collection," *Automation in Construction*, vol. 112, p. 103068, Apr. 2020.

[59] T. Miki, P. Khrapchenkov, and K. Hori, "UAV/UGV Autonomous Cooperation: UAV Assists UGV to Climb a Cliff by Attaching a Tether," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8041–8047, May 2019.

[60] L. Cantelli, M. Mangiameli, C. D. Melita, and G. Muscato, "UAV/UGV cooperation for surveying operations in humanitarian demining," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct. 2013, pp. 1–6.

[61] C. Phan and H. H. T. Liu, "A cooperative UAV/UGV platform for wildfire detection and fighting," in *2008 Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing*, Oct. 2008, pp. 494–498.

[62] M. Sinay, N. Agmon, O. Maksimov, G. Levy, M. Bitan, and S. Kraus, "UAV/UGV Search and Capture of Goal-Oriented Uncertain Targets," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 8505–8512.

[63] A. Lakas, B. Belkhouche, O. Benkraouda, A. Shuaib, and H. J. Alasmawi, "A Framework for a Cooperative UAV-UGV System for Path Discovery and Planning," in *2018 International Conference on Innovations in Information Technology (IIT)*, Nov. 2018, pp. 42–46.

[64] Q. Wang, H. Chen, L. Qiao, J. Tian, and Y. Su, "Path planning for UAV/UGV collaborative systems in intelligent manufacturing," *IET Intelligent Transport Systems*, vol. 14, no. 11, pp. 1475–1483, 2020.

[65] A. M. Khaleghi, D. Xu, A. Lobos, S. Minaeian, Y. Son, and J. Liu, "Agent-based hardware-in-the-loop simulation for UAV/UGV surveillance and crowd control system," in *2013 Winter Simulations Conference (WSC)*, Dec. 2013, pp. 1455–1466.

[66] M. Saska, T. Krajnik, and L. Pfeucil, "Cooperative uUAV-UGV autonomous indoor surveillance," in *International Multi-Conference on Systems, Signals Devices*, Mar. 2012, pp. 1–6.

[67] K. A. Ghamry, M. A. Kamel, and Y. Zhang, "Cooperative forest monitoring and fire detection using a team of UAVs-UGVs," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2016, pp. 1206–1211.

[68] Q. Vu, M. Raković, V. Delic, and A. Ronzhin, "Trends in Development of UAV-UGV Cooperation Approaches in Precision Agriculture," in *Interactive Collaborative Robotics*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 213–221.

[69] A. Pretto, S. Aravecchia, W. Burgard, N. Chebrolu, C. Dornhege, T. Falck, F. V. Fleckenstein, A. Fontenla, M. Imperoli, R. Khanna, F. Liebisch, P. Lottes, A. Milioto, D. Nardi, S. Nardi, J. Pfeifer, M. Popovic, C. Potena, C. Pradalier, E. Rothacker-Feder, I. Sa, A. Schaefer, R. Siegwart, C. Stachniss, A. Walter, W. Winterhalter, X. Wu, and J. Nieto, "Building an aerial–ground robotics system for precision farming: An adaptable solution," *IEEE Robotics Automation Magazine*, vol. 28, no. 3, pp. 29–49, 2021.

[70] J. A. Thomasson and X. Han, "The future of cooperative air and ground phenotyping," in *Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping V*, vol. 11414. International Society for Optics and Photonics, May 2020, p. 114140M.

[71] R. Szeliski, *Computer Vision: Algorithms and Applications*, ser. Texts in Computer Science. London: Springer-Verlag, 2011.

[72] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, Jul. 2015.

[73] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, "An Introductory Review of Deep Learning for Prediction Models With Big Data," *Front. Artif. Intell.*, vol. 3, 2020.

[74] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine Learning in Agriculture: A Review," *Sensors*, vol. 18, no. 8, p. 2674, Aug. 2018.

[75] A. K. Singh, B. Ganapathysubramanian, S. Sarkar, and A. Singh, "Deep Learning for Plant Stress Phenotyping: Trends and Future Perspectives," *Trends in Plant Science*, vol. 23, no. 10, pp. 883–898, Oct. 2018.

[76] M. Rahnemoonfar and C. Sheppard, "Deep Count: Fruit Counting Based on Deep Simulated Learning," *Sensors*, vol. 17, no. 4, p. 905, Apr. 2017.

[77] Y. Song, C. A. Glasbey, G. W. Horgan, G. Polder, J. A. Dieleman, and G. W. A. M. van der Heijden, "Automatic fruit recognition and counting from multiple images," *Biosystems Engineering*, vol. 118, pp. 203–215, Feb. 2014.

[78] X. Liu, S. W. Chen, S. Aditya, N. Sivakumar, S. Dcunha, C. Qu, C. J. Taylor, J. Das, and V. Kumar, "Robust Fruit Counting: Combining Deep Learning, Tracking, and Structure from Motion," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 1045–1052.

[79] S. W. Chen, S. S. Shivakumar, S. Dcunha, J. Das, E. Okon, C. Qu, C. J. Taylor, and V. Kumar, "Counting Apples and Oranges With Deep Learning: A Data-Driven Approach," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 781–788, Apr. 2017.

[80] X. Liu, S. W. Chen, C. Liu, S. S. Shivakumar, J. Das, C. J. Taylor, J. Underwood, and V. Kumar, "Monocular Camera Based Fruit Counting and Mapping With Semantic Data Association," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2296–2303, Jul. 2019.

[81] N. Häni, P. Roy, and V. Isler, "A comparative study of fruit detection and counting methods for yield mapping in apple orchards," *Journal of Field Robotics*, vol. 37, no. 2, pp. 263–282, 2020.

[82] W. S. Qureshi, A. Payne, K. B. Walsh, R. Linker, O. Cohen, and M. N. Dailey, "Machine vision for counting fruit on mango tree canopies," *Precision Agric*, vol. 18, no. 2, pp. 224–244, Apr. 2017.

[83] A. J. Scarfe, R. C. Flemmer, H. H. Bakker, and C. L. Flemmer, "Development of an autonomous kiwifruit picking robot," in *2009 4th International Conference on Autonomous Robots and Agents*, Feb. 2009, pp. 380–384.

[84] H. A. M. Williams, M. H. Jones, M. Nejati, M. J. Seabright, J. Bell, N. D. Penhall, J. J. Barnett, M. D. Duke, A. J. Scarfe, H. S. Ahn, J. Lim, and B. A. MacDonald, "Robotic kiwifruit harvesting using machine vision, convolutional neural networks, and robotic arms," *Biosystems Engineering*, vol. 181, pp. 140–156, May 2019.

[85] C. Schuetz, J. Baur, J. Pfaff, T. Buschmann, and H. Ulbrich, "Evaluation of a direct optimization method for trajectory planning of a 9-DOF redundant fruit-picking manipulator," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2660–2666.

[86] S. S. Mehta, W. MacKunis, and T. F. Burks, "Robust visual servo control in the presence of fruit motion for robotic citrus harvesting," *Computers and Electronics in Agriculture*, vol. 123, pp. 362–375, Apr. 2016.

[87] H. Yaguchi, K. Nagahama, T. Hasegawa, and M. Inaba, "Development of an autonomous tomato harvesting robot with rotational plucking gripper," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 652–657.

[88] L. Zhang, J. Jia, G. Gui, X. Hao, W. Gao, and M. Wang, "Deep Learning Based Improved Classification System for Designing Tomato Harvesting Robot," *IEEE Access*, vol. 6, pp. 67 940–67 950, 2018.

[89] Y. Zhao, L. Gong, Y. Huang, and C. Liu, "A review of key techniques of vision-based control for harvesting robot," *Computers and Electronics in Agriculture*, vol. 127, pp. 311–323, Sep. 2016.

[90] Z. De-An, L. Jidong, J. Wei, Z. Ying, and C. Yu, "Design and control of an apple harvesting robot," *Biosystems Engineering*, vol. 110, no. 2, pp. 112–122, Oct. 2011.

[91] S. Hayashi, K. Shigematsu, S. Yamamoto, K. Kobayashi, Y. Kohno, J. Kamata, and M. Kurita, "Evaluation of a strawberry-harvesting robot in a field test," *Biosystems Engineering*, vol. 105, no. 2, pp. 160–171, Feb. 2010.

[92] D. Andújar, J. Dorado, C. Fernández-Quintanilla, and A. Ribeiro, "An Approach to the Use of Depth Cameras for Weed Volume Estimation," *Sensors*, vol. 16, no. 7, p. 972, Jul. 2016.

[93] J. Li and L. Tang, "Crop recognition under weedy conditions based on 3D imaging for robotic weed control," *Journal of Field Robotics*, vol. 35, no. 4, pp. 596–611, 2018.

[94] O. Bawden, J. Kulk, R. Russell, C. McCool, A. English, F. Dayoub, C. Lehnert, and T. Perez, "Robot for weed species plant-specific management," *Journal of Field Robotics*, vol. 34, no. 6, pp. 1179–1199, 2017.

[95] M. Sujaritha, S. Annadurai, J. Satheeshkumar, S. Kowshik Sharan, and L. Mahesh, "Weed detecting robot in sugarcane fields using fuzzy real time classifier," *Computers and Electronics in Agriculture*, vol. 134, pp. 160–171, Mar. 2017.

[96] A. Heravi, D. Ahmad, I. A. Hameed, R. R. Shamshiri, S. K. Balasundram, and M. Yamin, "Development of a Field Robot Platform for Mechanical Weed Control in Greenhouse Cultivation of Cucumber," *Agricultural Robots - Fundamentals and Applications*, Nov. 2018.

[97] K. Buddha, H. J. Nelson, D. Zermas, and N. Papanikolopoulos, "Weed Detection and Classification in High Altitude Aerial Images for Robot-Based Precision Agriculture," in *2019 27th Mediterranean Conference on Control and Automation (MED)*, Jul. 2019, pp. 280–285.

[98] P. Lottes, R. Khanna, J. Pfeifer, R. Siegwart, and C. Stachniss, "UAV-based crop and weed classification for smart farming," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3024–3031.

[99] I. Sa, Z. Chen, M. Popović, R. Khanna, F. Liebisch, J. Nieto, and R. Siegwart, "weedNet: Dense Semantic Weed Classification Using Multispectral Images and MAV for Smart Farming," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 588–595, Jan. 2018.

[100] S. Haug, A. Michaels, P. Biber, and J. Ostermann, "Plant classification system for crop /weed discrimination without segmentation," in *IEEE Winter Conference on Applications of Computer Vision*, Mar. 2014, pp. 1142–1149.

[101] F. López-Granados, "Weed detection for site-specific weed management: mapping and real-time approaches," *Weed Research*, vol. 51, no. 1, pp. 1–11, 2011.

[102] A. Wang, W. Zhang, and X. Wei, "A review on weed detection using ground-based machine vision and image processing techniques," *Computers and Electronics in Agriculture*, vol. 158, pp. 226–240, Mar. 2019.

[103] C. Potena, D. Nardi, and A. Pretto, "Fast and Accurate Crop and Weed Identification with Summarized Train Sets for Precision Agriculture," in *Intelligent Autonomous Systems 14*, ser. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2017, pp. 105–121.

[104] T. Mueller-Sim, M. Jenkins, J. Abel, and G. Kantor, "The Robotanist: A ground-based agricultural robot for high-throughput crop phenotyping," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3634–3639.

[105] S. Jahnke, J. Roussel, T. Hombach, J. Kochs, A. Fischbach, G. Huber, and H. Scharr, "phenoSeeder - A Robot System for Automated Handling and Phenotyping of Individual Seeds," *Plant Physiology*, vol. 172, no. 3, pp. 1358–1370, Nov. 2016.

[106] J. Iqbal, R. Xu, S. Sun, and C. Li, "Simulation of an Autonomous Mobile Robot for LiDAR-Based In-Field Phenotyping and Navigation," *Robotics*, vol. 9, no. 2, p. 46, Jun. 2020.

[107] S. N. Young, E. Kayacan, and J. M. Peschel, "Design and field evaluation of a ground robot for high-throughput phenotyping of energy sorghum," *Precision Agric*, vol. 20, no. 4, pp. 697–722, Aug. 2019.

[108] N. Brichet, C. Fournier, O. Turc, O. Strauss, S. Artzet, C. Pradal, C. Welcker, F. Tardieu, and L. Cabrera-Bosquet, "A robot-assisted imaging pipeline for tracking the growths of maize ear and silks in a high-throughput phenotyping platform," *Plant Methods*, vol. 13, no. 1, p. 96, Nov. 2017.

[109] C. Wu, R. Zeng, J. Pan, C. C. L. Wang, and Y. Liu, "Plant Phenotyping by Deep-Learning-Based Planner for Multi-Robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3113–3120, Oct. 2019.

[110] A. Kicherer, K. Herzog, M. Pflanz, M. Wieland, P. Rüger, S. Kecke, H. Kuhlmann, and R. Töpfer, "An Automated Field Phenotyping Pipeline for Application in Grapevine Research," *Sensors*, vol. 15, no. 3, pp. 4823–4836, Mar. 2015.

[111] T. Gao, H. Emadi, H. Saha, J. Zhang, A. Lofquist, A. Singh, B. Ganapathysubramanian, S. Sarkar, A. K. Singh, and S. Bhattacharya, "A Novel Multirobot System for Plant Phenotyping," *Robotics*, vol. 7, no. 4, p. 61, Dec. 2018.

[112] A. Shafiekhani, S. Kadam, F. B. Fritschi, and G. N. DeSouza, "Vinobot and Vinoculer: Two Robotic Platforms for High-Throughput Field Phenotyping," *Sensors*, vol. 17, no. 1, p. 214, Jan. 2017.

[113] C. Still, R. Powell, D. Aubrecht, Y. Kim, B. Helliker, D. Roberts, A. D. Richardson, and M. Goulden, "Thermal imaging in plant and ecosystem ecology: applications and challenges," *Ecosphere*, vol. 10, no. 6, p. e02768, 2019.

[114] A. Prashar and H. G. Jones, "Infra-Red Thermography as a High-Throughput Tool for Field Phenotyping," *Agronomy*, vol. 4, no. 3, pp. 397–417, Sep. 2014.

[115] F. Y. Narvaez, G. Reina, M. Torres-Torriti, G. Kantor, and F. A. Cheein, "A Survey of Ranging and Imaging Techniques for Precision Agriculture Phenotyping," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2428–2439, Dec. 2017.

[116] L. Li, Q. Zhang, and D. Huang, "A Review of Imaging Techniques for Plant Phenotyping," *Sensors*, vol. 14, no. 11, pp. 20 078–20 111, Nov. 2014.

[117] A. Walter, F. Liebisch, and A. Hund, "Plant phenotyping: from bean weighing to image analysis," *Plant Methods*, vol. 11, no. 1, p. 14, Mar. 2015.

[118] C. Costa, U. Schurr, F. Loreto, P. Menesatti, and S. Carpentier, "Plant Phenotyping Research Trends, a Science Mapping Approach," *Front. Plant Sci.*, vol. 9, 2019.

[119] G. J. Rebetzke, J. Jimenez-Berni, R. A. Fischer, D. M. Deery, and D. J. Smith, "Review: High-throughput phenotyping to enhance the use of crop genetic resources," *Plant Science*, vol. 282, pp. 40–48, May 2019.

[120] J. F. S. Gomes and F. R. Leta, "Applications of computer vision techniques in the agriculture and food industry: a review," *Eur Food Res Technol*, vol. 235, no. 6, pp. 989–1000, Dec. 2012.

[121] D. I. Patrício and R. Rieder, "Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review," *Computers and Electronics in Agriculture*, vol. 153, pp. 69–81, Oct. 2018.

[122] F. García-Sánchez, L. Galvez-Sola, J. J. Martínez-Nicolás, R. Muelas-Domingo, and M. Nieves, "Using Near-Infrared Spectroscopy in Agricultural Systems," *Developments in Near-Infrared Spectroscopy*, Mar. 2017.

[123] N. Katsoulas, A. Elvanidi, K. P. Ferentinos, M. Kacira, T. Bartzanas, and C. Kittas, "Crop reflectance monitoring as a tool for water stress detection in greenhouses: A review," *Biosystems Engineering*, vol. 151, pp. 374–398, Nov. 2016.

[124] A. Elvanidi, N. Katsoulas, D. Augoustaki, I. Loulou, and C. Kittas, "Crop reflectance measurements for nitrogen deficiency detection in a soilless tomato crop," *Biosystems Engineering*, vol. 176, pp. 1–11, Dec. 2018.

[125] J. Anderegg, K. Yu, H. Aasen, A. Walter, F. Liebisch, and A. Hund, "Spectral Vegetation Indices to Track Senescence Dynamics in Diverse Wheat Germplasm," *Front. Plant Sci.*, vol. 10, 2020.

[126] D. Heckmann, U. Schlüter, and A. P. M. Weber, "Machine Learning Techniques for Predicting Crop Photosynthetic Capacity from Leaf Reflectance Spectra," *Molecular Plant*, vol. 10, no. 6, pp. 878–890, Jun. 2017.

[127] A. J. Foster, V. G. Kakani, and J. Mosali, "Estimation of bioenergy crop yield and N status by hyperspectral canopy reflectance and partial least square regression," *Precision Agric*, vol. 18, no. 2, pp. 192–209, Apr. 2017.

[128] J. Behmann, D. Bohnenkamp, and A.-K. Mahlein, "Image-based assessment of hyperspectral reflectance characteristics of plants in the field: Lessons Learned," in *Precision agriculture ?19*. Wageningen Academic Publishers, Jul. 2019, pp. 203–208.

[129] O. A. Montesinos-López, A. Montesinos-López, J. Crossa, G. de los Campos, G. Alvarado, M. Suchismita, J. Rutkoski, L. González-Pérez, and J. Burgueño, "Predicting grain yield using canopy hyperspectral reflectance in wheat breeding data," *Plant Methods*, vol. 13, no. 1, p. 4, Jan. 2017.

[130] T. S. Breure, A. E. Milne, R. Webster, S. M. Haefele, J. A. Hannam, S. Moreno-Rojas, and R. Corstanje, "Predicting the growth of lettuce from soil infrared reflectance spectra: the potential for crop management," *Precision Agric*, Aug. 2020.

[131] C. L. Wiegand, A. J. Richardson, D. E. Escobar, and A. H. Gerbermann, "Vegetation indices in crop assessments," *Remote Sensing of Environment*, vol. 35, no. 2, pp. 105–119, Feb. 1991.

[132] I. Mariotto, P. S. Thenkabail, A. Huete, E. T. Slonecker, and A. Platonov, "Hyperspectral versus multispectral crop-productivity modeling and type discrimination for the HyspIRI mission," *Remote Sensing of Environment*, vol. 139, p. 291305, 2013.

[133] E. Adam, O. Mutanga, and D. Rugege, "Multispectral and hyperspectral remote sensing for identification and mapping of wetland vegetation: a review," *Wetlands Ecol Manage*, vol. 18, no. 3, pp. 281–296, Jun. 2010.

[134] B. Lu, P. D. Dao, J. Liu, Y. He, and J. Shang, "Recent Advances of Hyperspectral Imaging Technology and Applications in Agriculture," *Remote Sensing*, vol. 12, no. 16, p. 2659, Jan. 2020.

[135] S. Pascucci, S. Pignatti, R. Casa, R. Darvishzadeh, and W. Huang, "Special Issue "Hyperspectral Remote Sensing of Agriculture and Vegetation"," *Remote Sensing*, vol. 12, no. 21, p. 3665, Jan. 2020.

[136] D. Caballero, R. Calvini, and J. M. Amigo, "Chapter 3.3 - Hyperspectral imaging in crop fields: precision agriculture," in *Data Handling in Science and Technology*, ser. Hyperspectral Imaging. Elsevier, Jan. 2020, vol. 32, pp. 453–473.

[137] A. Singh, S. Jones, B. Ganapathysubramanian, S. Sarkar, D. Mueller, K. Sandhu, and K. Nagasubramanian, "Challenges and Opportunities in Machine-Augmented Plant Stress Phenotyping," *Trends in Plant Science*, vol. 0, no. 0, Aug. 2020.

[138] C. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. New York: Springer-Verlag, 2006.

[139] J. R. Ubbens and I. Stavness, "Deep Plant Phenomics: A Deep Learning Platform for Complex Plant Phenotyping Tasks," *Front. Plant Sci.*, vol. 8, 2017.

[140] A. Singh, B. Ganapathysubramanian, A. K. Singh, and S. Sarkar, "Machine Learning for High-Throughput Stress Phenotyping in Plants," *Trends in Plant Science*, vol. 21, no. 2, pp. 110–124, Feb. 2016.

[141] M. P. Pound, J. A. Atkinson, D. M. Wells, T. P. Pridmore, and A. P. French, "Deep Learning for Multi-Task Plant Phenotyping," p. 9.

[142] A. Fuentes, S. Yoon, and D. S. Park, "Deep Learning-Based Phenotyping System With Glocal Description of Plant Anomalies and Symptoms," *Front. Plant Sci.*, vol. 10, 2019.

[143] M. Islam, Anh Dinh, K. Wahid, and P. Bhowmik, "Detection of potato diseases using image segmentation and multiclass support vector machine," in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, Apr. 2017, pp. 1–4.

[144] K. Nagasubramanian, S. Jones, A. K. Singh, S. Sarkar, A. Singh, and B. Ganapathysubramanian, "Plant disease identification using explainable 3D deep learning on hyperspectral images," *Plant Methods*, vol. 15, no. 1, p. 98, Aug. 2019.

[145] S. Ghosal, D. Blystone, A. K. Singh, B. Ganapathysubramanian, A. Singh, and S. Sarkar, "An explainable deep machine vision framework for plant stress phenotyping," *PNAS*, vol. 115, no. 18, pp. 4613–4618, May 2018.

[146] R. P. Sishodia, R. L. Ray, and S. K. Singh, "Applications of Remote Sensing in Precision Agriculture: A Review," *Remote Sensing*, vol. 12, no. 19, p. 3136, Jan. 2020.

[147] D. J. Mulla, "Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps," *Biosystems Engineering*, vol. 114, no. 4, pp. 358–371, Apr. 2013.

[148] L. Zhu, J. Suomalainen, J. Liu, J. Hyyppä, H. Kaartinen, and H. Haggren, "A Review: Remote Sensing Sensors," *Multi-purposeful Application of Geospatial Data*, Dec. 2017.

[149] X. X. Zhu, D. Tuia, L. Mou, G. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, Dec. 2017.

[150] A. Bannari, D. Morin, F. Bonn, and A. R. Huete, "A review of vegetation indices," *Remote Sensing Reviews*, vol. 13, no. 1-2, pp. 95–120, Aug. 1995.

[151] J. D. Tarpley, S. R. Schneider, and R. L. Money, "Global Vegetation Indices from the NOAA-7 Meteorological Satellite," *Journal of Applied Meteorology and Climatology*, vol. 23, no. 3, pp. 491–494, Mar. 1984.

[152] D. G. Hadjimitsis, G. Papadavid, A. Agapiou, K. Themistocleous, M. G. Hadjimitsis, A. Retalis, S. Michaelides, N. Chrysoulakis, L. Toulios, and C. R. I. Clayton, "Atmospheric correction for satellite remotely sensed data intended for agricultural applications: impact on vegetation indices," *Natural Hazards and Earth System Sciences*, vol. 10, no. 1, pp. 89–95, Jan. 2010.

[153] J. C. Price, "Calibration of satellite radiometers and the comparison of vegetation indices," *Remote Sensing of Environment*, vol. 21, no. 1, pp. 15–27, Feb. 1987.

[154] J. Xue and B. Su, "Significant Remote Sensing Vegetation Indices: A Review of Developments and Applications," May 2017.

[155] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy, "Neural Network Approaches Versus Statistical Methods In Classification Of Multisource Remote Sensing Data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 4, pp. 540–552, Jul. 1990.

[156] A. E. Maxwell, T. A. Warner, and F. Fang, "Implementation of machine-learning classification in remote sensing: an applied review," *International Journal of Remote Sensing*, vol. 39, no. 9, pp. 2784–2817, May 2018.

[157] G. Cheng, J. Han, and X. Lu, "Remote Sensing Image Scene Classification: Benchmark and State of the Art," *Proceedings of the IEEE*, vol. 105, no. 10, pp. 1865–1883, Oct. 2017.

[158] Y. Huang, Z.-x. Chen, T. Yu, X.-z. Huang, and X.-f. Gu, "Agricultural remote sensing big data: Management and applications," *Journal of Integrative Agriculture*, vol. 17, no. 9, pp. 1915–1931, Sep. 2018.

[159] E. Ndikumana, D. Ho Tong Minh, N. Baghdadi, D. Courault, and L. Hossard, "Deep Recurrent Neural Network for Agricultural Classification using multitemporal SAR Sentinel-1 for Camargue, France," *Remote Sensing*, vol. 10, no. 8, p. 1217, Aug. 2018.

[160] K. K. Gadiraju, B. Ramachandra, Z. Chen, and R. R. Vatsavai, "Multimodal Deep Learning Based Crop Classification Using Multispectral and Multitemporal Satellite Imagery," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 3234–3242.

[161] S. K. Seelan, S. Laguette, G. M. Casady, and G. A. Seielstad, "Remote sensing applications for precision agriculture: A learning community approach," *Remote Sensing of Environment*, vol. 88, no. 1, pp. 157–169, Nov. 2003.

[162] L. Deng, Z. Mao, X. Li, Z. Hu, F. Duan, and Y. Yan, "UAV-based multispectral remote sensing for precision agriculture: A comparison between different cameras," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 146, pp. 124–136, Dec. 2018.

[163] V. Lukas, J. Novák, L. Neudert, I. Svobodova, F. Rodriguez-Moreno, M. Edrees, and J. Kren, "The combination of uav survey and landsat imagery for monitoring of crop vigor in precision agriculture," in *ISPRS - International Archives of the*

Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XLI-B8. Copernicus GmbH, Jun. 2016, pp. 953–957.

[164] A. Mancini, E. Frontoni, and P. Zingaretti, "Satellite and UAV data for Precision Agriculture Applications," in *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2019, pp. 491–497.

[165] E.-C. Oerke, A.-K. Mahlein, and U. Steiner, "Proximal Sensing of Plant Diseases," in *Detection and Diagnostics of Plant Pathogens*, ser. Plant Pathology in the 21st Century. Dordrecht: Springer Netherlands, 2014, pp. 55–68.

[166] S. Cubero, E. Marco-Noales, N. Aleixos, S. Barbé, and J. Blasco, "RobHortic: A Field Robot to Detect Pests and Diseases in Horticultural Crops by Proximal Sensing," *Agriculture*, vol. 10, no. 7, p. 276, Jul. 2020.

[167] R. Vidoni, R. Gallo, G. Ristorto, G. Carabin, F. Mazzetto, L. Scalera, and A. Gasparetto, "ByeLab: An Agricultural Mobile Robot Prototype for Proximal Sensing and Precision Farming." American Society of Mechanical Engineers Digital Collection, Jan. 2018.

[168] B. Rey, N. Aleixos, S. Cubero, and J. Blasco, "Xf-Rovim. A Field Robot to Detect Olive Trees Infected by Xylella Fastidiosa Using Proximal Sensing," *Remote Sensing*, vol. 11, no. 3, p. 221, Jan. 2019.

[169] S. Paulus, J. Dupuis, A.-K. Mahlein, and H. Kuhlmann, "Surface feature based classification of plant organs from 3D laserscanned point clouds for plant phenotyping," *BMC Bioinformatics*, vol. 14, no. 1, p. 238, Jul. 2013.

[170] F. E. Oliva, O. S. Dalmau, and T. E. Alarcón, "A Supervised Segmentation Algorithm for Crop Classification Based on Histograms Using Satellite Images," in *Human-Inspired Computing and Its Applications*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 327–335.

[171] M. Zortea and E. R. Rodrigues, "Crop identification using superpixels and supervised classification of multispectral CBERS-4 wide-field imagery," in *Remote Sensing for Agriculture, Ecosystems, and Hydrology XXI*, vol. 11149. International Society for Optics and Photonics, Oct. 2019, p. 111491U.

[172] S. Jay, G. Rabatel, X. Hadoux, D. Moura, and N. Gorretta, "In-field crop row phenotyping from 3D modeling performed using Structure from Motion," *Computers and Electronics in Agriculture*, vol. 110, pp. 70–77, Jan. 2015.

[173] L. Lou, Y. Liu, M. Sheng, J. Han, and J. H. Doonan, "A Cost-Effective Automatic 3D Reconstruction Pipeline for Plants Using Multi-view Images," in *Advances in Autonomous Robotics Systems*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 221–230.

[174] X. Sun, X. Zhu, P. Wang, and H. Chen, "A Review of Robot Control with Visual Servoing," in *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, Jul. 2018, pp. 116–121.

[175] C. Xia, L. Wang, B.-K. Chung, and J.-M. Lee, "In Situ 3D Segmentation of Individual Plant Leaves Using a RGB-D Camera for Agricultural Automation," *Sensors*, vol. 15, no. 8, pp. 20 463–20 479, Aug. 2015.

[176] Thang Cao, K. Panjvani, Anh Dinh, K. Wahid, and P. Bhowmik, "An approach to detect branches and seedpods based on 3D image in low-cost plant phenotyping platform," in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, Apr. 2017, pp. 1–4.

[177] M. Li, M.-R. Shao, D. Zeng, T. Ju, E. A. Kellogg, and C. N. Topp, "Comprehensive 3D phenotyping reveals continuous morphological variation across genetically diverse sorghum inflorescences," *New Phytologist*, vol. 226, no. 6, pp. 1873–1885, 2020.

[178] S. Paulus, "Measuring crops in 3D: using geometry for plant phenotyping," *Plant Methods*, vol. 15, no. 1, p. 103, Sep. 2019.

[179] I. Ziamtsov and S. Navlakha, "Machine Learning Approaches to Improve Three Basic Plant Phenotyping Tasks Using Three-Dimensional Point Clouds," *Plant Physiology*, vol. 181, no. 4, pp. 1425–1440, Dec. 2019.

[180] I. Plyusnin, A. R. Evans, A. Karme, A. Gionis, and J. Jernvall, "Automated 3D Phenotype Analysis Using Data Mining," *PLOS ONE*, vol. 3, no. 3, p. e1742, Mar. 2008.

[181] Y. Jiang and C. Li, "Convolutional Neural Networks for Image-Based High-Throughput Plant Phenotyping: A Review," Apr. 2020.

[182] J. Ubbens, M. Cieslak, P. Prusinkiewicz, and I. Stavness, "The use of plant models in deep learning: an application to leaf counting in rosette plants," *Plant Methods*, vol. 14, no. 1, p. 6, Jan. 2018.

[183] J. Zhou, X. Fu, S. Zhou, J. Zhou, H. Ye, and H. T. Nguyen, "Automated segmentation of soybean plants from 3D point cloud using machine learning," *Computers and Electronics in Agriculture*, vol. 162, pp. 143–153, Jul. 2019.

[184] S. Jin, Y. Su, S. Gao, F. Wu, T. Hu, J. Liu, W. Li, D. Wang, S. Chen, Y. Jiang, S. Pang, and Q. Guo, "Deep Learning: Individual Maize Segmentation From Terrestrial Lidar Data Using Faster R-CNN and Regional Growth Algorithms," *Front. Plant Sci.*, vol. 9, 2018.

[185] M. Hobart, M. Pflanz, C. Weltzien, and M. Schirrmann, "Growth Height Determination of Tree Walls for Precise Monitoring in Apple Fruit Production Using UAV Photogrammetry," *Remote Sensing*, vol. 12, no. 10, p. 1656, Jan. 2020.

[186] S. Yang, L. Zheng, W. Gao, B. Wang, X. Hao, J. Mi, and M. Wang, "An Efficient Processing Approach for Colored Point Cloud-Based High-Throughput Seedling Phenotyping," *Remote Sensing*, vol. 12, no. 10, p. 1540, Jan. 2020.

[187] J.-X. Xu, J. Ma, Y.-N. Tang, W.-X. Wu, J.-H. Shao, W.-B. Wu, S.-Y. Wei, Y.-F. Liu, Y.-C. Wang, and H.-Q. Guo, "Estimation of Sugarcane Yield Using a Machine Learning Approach Based on UAV-LiDAR Data," *Remote Sensing*, vol. 12, no. 17, p. 2823, Jan. 2020.

[188] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering Problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research*, vol. 255, no. 2, pp. 315–332, Dec. 2016.

[189] J. Yu, M. Schwager, and D. Rus, "Correlated Orienteering Problem and its application to informative path planning for persistent monitoring tasks," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2014, pp. 342–349.

[190] W. Zhang, K. Wang, S. Wang, and G. Laporte, "Clustered coverage orienteering problem of unmanned surface vehicles for water sampling," *Naval Research Logistics (NRL)*, vol. 67, no. 5, pp. 353–367, 2020.

[191] J. Hwang, N. Bose, and S. Fan, "AUV Adaptive Sampling Methods: A Review," *Applied Sciences*, vol. 9, no. 15, p. 3145, Jan. 2019.

[192] Y. T. Tan, A. Kunapareddy, and M. Kobilarov, "Gaussian Process Adaptive Sampling Using the Cross-Entropy Method for Environmental Sensing and Monitoring," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 6220–6227.

[193] A. Renzaglia, C. Reymann, and S. Lacroix, "Monitoring the evolution of clouds with UAVs," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 278–283.

[194] T. Salam and M. A. Hsieh, "Adaptive Sampling and Reduced-Order Modeling of Dynamic Processes by Robot Teams," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 477–484, Apr. 2019.

[195] W. Luo and K. Sycara, "Adaptive Sampling and Online Learning in Multi-Robot Sensor Coverage with Mixture of Gaussian Processes," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 6359–6364.

[196] S. McCammon and G. A. Hollinger, "Topological Hotspot Identification for Informative Path Planning with a Marine Robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 4865–4872.

[197] S. Kemna, J. G. Rogers, C. Nieto-Granda, S. Young, and G. S. Sukhatme, "Multi-robot coordination through dynamic Voronoi partitioning for informative adaptive sampling in communication-constrained environments," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2124–2130.

[198] V. L. Somers and I. R. Manchester, "Priority Maps for Surveillance and Intervention of Wildfires and other Spreading Processes," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 739–745.

[199] D. Saldana, R. Assunção, and M. F. M. Campos, "A distributed multi-robot approach for the detection and tracking of multiple dynamic anomalies," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1262–1267.

[200] Q. Lu and Q.-L. Han, "Mobile Robot Networks for Environmental Monitoring: A Cooperative Receding Horizon Temporal Logic Control Approach," *IEEE Transactions on Cybernetics*, vol. 49, no. 2, pp. 698–711, Feb. 2019.

[201] M. Ghaffari Jadidi, J. Valls Miro, and G. Dissanayake, "Sampling-based incremental information gathering with applications to robotic exploration and environmental monitoring," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 658–685, May 2019.

[202] T. Miki, M. Popović, A. Gawel, G. Hitz, and R. Siegwart, "Multi-Agent Time-Based Decision-Making for the Search and Action Problem," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2365–2372.

[203] A. Viseras, T. Wiedemann, C. Manss, L. Magel, J. Mueller, D. Shutin, and L. Merino, "Decentralized multi-agent exploration with online-learning of Gaussian processes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 4222–4229.

[204] C. Baykal, G. Rosman, S. Claici, and D. Rus, "Persistent surveillance of events with unknown, time-varying statistics," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 2682–2689.

[205] A. Benevento, M. Santos, G. Notarstefano, K. Paynabar, M. Bloch, and M. Egerstedt, "Multi-Robot Coordination for Estimation and Coverage of Unknown Spatial Fields," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 7740–7746.

[206] W.-T. Li and Y.-C. Liu, "Dynamic coverage control for mobile robot network with limited and nonidentical sensory ranges," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 775–780.

[207] K. Yu, C. Guo, and J. Yi, "Complete and Near-Optimal Path Planning for Simultaneous Sensor-Based Inspection and Footprint Coverage in Robotic Crack Filling," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 8812–8818.

[208] J. M. Palacios-Gasos, Z. Talebpour, E. Montijano, C. Sagüés, and A. Martinoli, "Optimal path planning and coverage control for multi-robot persistent coverage in environments with obstacles," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1321–1327.

[209] A. Nguyen, D. Krupke, M. Burbage, S. Bhatnagar, S. P. Fekete, and A. T. Becker, "Using a UAV for Destructive Surveys of Mosquito Population," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 7812–7819.

[210] C. E. Rasmussen, "Gaussian Processes in Machine Learning," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 63–71.

[211] L. Bottarelli, M. Bicego, J. Blum, and A. Farinelli, "Orienteering-based informative path planning for environmental monitoring," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 46–58, Jan. 2019.

[212] K.-C. Ma, L. Liu, and G. S. Sukhatme, "Informative planning and online learning with sparse Gaussian processes," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4292–4298.

[213] T. C. Thayer, S. Vougioukas, K. Goldberg, and S. Carpin, "Multirobot Routing Algorithms for Robots Operating in Vineyards," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1184–1194, Jul. 2020.

[214] M. G. Plessen and A. Bemporad, "Shortest path computations under trajectory constraints for ground vehicles within agricultural fields," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2016, pp. 1733–1738.

[215] L. C. Santos, F. N. Santos, E. J. S. Pires, A. Valente, P. Costa, and S. Magalhães, "Path Planning for ground robots in agriculture: a short review," in *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Apr. 2020, pp. 61–66.

[216] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1498–1511, Dec. 2016.

[217] P. Maini and P. B. Sujit, "On cooperation between a fuel constrained UAV and a refueling UGV for large scale mapping applications," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2015, pp. 1370–1377.

[218] S. Dutta Roy, S. Chaudhury, and S. Banerjee, "Active recognition through next view planning: a survey," *Pattern Recognition*, vol. 37, no. 3, pp. 429–446, Mar. 2004.

[219] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1343–1377, Sep. 2011.

[220] M. Mendoza, J. I. Vasquez-Gomez, H. Taud, L. E. Sucar, and C. Reta, "Supervised Learning of the Next-Best-View for 3D Object Reconstruction," *Pattern Recognition Letters*, vol. 133, pp. 224–231, May 2020.

[221] C. Munkelt, M. Trummer, J. Denzler, and e. al, "Benchmarking 3D reconstruction from next best view planning," in *IAPR Conference on Machine Vision Applications, MVA 2007*, 2007, pp. 552–555.

[222] S. Arce, C. A. Vernon, J. Hammond, V. Newell, J. Janson, K. W. Franke, and J. D. Hedengren, "Automated 3D Reconstruction Using Optimized View-Planning Algorithms for Iterative Development of Structure-from-Motion Models," *Remote Sensing*, vol. 12, no. 13, p. 2169, Jan. 2020.

[223] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3D reconstruction," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3477–3484.

[224] M. A. Goodrich and A. C. Schultz, "Human–Robot Interaction: A Survey," *HCI*, vol. 1, no. 3, pp. 203–275, Jan. 2008.

[225] P. A. Lasota, T. Fong, and J. A. Shah, "A Survey of Methods for Safe Human-Robot Interaction," *FNT in Robotics*, vol. 5, no. 3, pp. 261–349, 2017.

[226] J. P. Vasconez, G. A. Kantor, and F. A. Auat Cheein, "Human–robot interaction in agriculture: A survey and current challenges," *Biosystems Engineering*, vol. 179, pp. 35–48, Mar. 2019.

[227] F. A. Cheein, D. Herrera, J. Gimenez, R. Carelli, M. Torres-Torriti, J. R. Rosell-Polo, A. Escolà, and J. Arnó, "Human-robot interaction in precision agriculture: Sharing the workspace with service units," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, Mar. 2015, pp. 289–295.

[228] P. Baxter, G. Cielniak, M. Hanheide, and P. From, "Safe Human-Robot Interaction in Agriculture," in *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '18. New York, NY, USA: Association for Computing Machinery, Mar. 2018, pp. 59–60.

[229] L. Benos, A. Bechar, and D. Bochtis, "Safety and ergonomics in human-robot interactive agricultural operations," *Biosystems Engineering*, vol. 200, pp. 55–72, Dec. 2020.

[230] T. B. Sheridan, "Human–Robot Interaction: Status and Challenges," *Hum Factors*, vol. 58, no. 4, pp. 525–532, Jun. 2016.

[231] K. A. Demir, G. Doven, and B. Sezen, "Industry 5.0 and Human-Robot Coworking," *Procedia Computer Science*, vol. 158, pp. 688–695, Jan. 2019.

[232] P. Tsarouchi, S. Makris, and G. Chryssolouris, "Human–robot interaction review and challenges on task planning and programming," *International Journal of Computer Integrated Manufacturing*, vol. 29, no. 8, pp. 916–931, Aug. 2016.

[233] C. Harper and G. Virk, "Towards the Development of International Safety Standards for Human Robot Interaction," *Int J of Soc Robotics*, vol. 2, no. 3, pp. 229–234, Sep. 2010.

[234] G. Adamides, C. Katsanos, Y. Parmet, G. Christou, M. Xenos, T. Hadzilacos, and Y. Edan, "HRI usability evaluation of interaction modes for a teleoperated agricultural robotic sprayer," *Applied Ergonomics*, vol. 62, pp. 237–246, Jul. 2017.

[235] Z. Huang, G. Miyauchi, A. S. Gomez, R. Bird, A. S. Kalsi, C. Jansen, Z. Liu, S. Parsons, and E. Sklar, "An Experiment on Human-Robot Interaction in a Simulated Agricultural Task," in *Towards Autonomous Robotic Systems*, ser. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 221–233.

[236] M. Kim, I. Koh, H. Jeon, J. Choi, B. C. Min, E. T. Matson, and J. Gallagher, "A HARMS-based heterogeneous human-robot team for gathering and collecting," *1*, vol. 2, no. 3, pp. 201–217, Sep. 2018.

[237] Y. Gong, K. Chen, J. Yi, and T. Liu, "Control of a two-wheel steering bikebot for agile maneuvers," in *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2019, pp. 984–989.

[238] K. Hunte and J. Yi, "Collaborative object manipulation through indirect control of a deformable sheet by a mobile robotic team," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 2019, pp. 1463–1468.

[239] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[240] J. C. K. Chou, "Quaternion kinematic and dynamic differential equations," *IEEE Trans. Robot. Automat.*, vol. 8, no. 1, pp. 53–64, Feb. 1992.

[241] R. Mahony *et al.*, "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, Sep. 2012.

[242] E. Fresk and G. Nikolakopoulos, "Full quaternion based attitude control for a quadrotor," in *2013 European Control Conference (ECC)*. Zurich: IEEE, Jul. 2013, pp. 3864–3869.

[243] M. Quigley *et al.*, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, vol. 3, 2009.

[244] B. T. Phong, "Illumination for computer generated pictures," *Comm. ACM*, vol. 18, no. 6, pp. 311–317, 1975.

[245] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, UK: Cambridge University Press, 2003.

[246] J. Courbon, Y. Mezouar, L. Eckt, and P. Martinet, "A generic fisheye camera model for robotic applications," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, San Diego, CA, 2007, pp. 1683–1688.

[247] D. Scaramuzza, "Omnidirectional camera," in *Computer Vision: A Reference Guide*, 2014, pp. 552–560.

[248] S. W. Oh, M. S. Brown, M. Pollefeys, and S. J. Kim, "Do it yourself hyperspectral imaging with everyday digital cameras," in *Proc. IEEE Conf. Comp. Vis. Pattern Recog.*, Las Vegas, NV, USA, 2016, pp. 2461–2469.

[249] M. Edmonds, T. Yigit, and J. Yi, "Auto-calibrated 3d hyperspectral scanning using a heterogeneous set of cameras and lights with spectrally-optimal next-best-view planning," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, 2020, pp. 863–868.

[250] D. K. Prasad, R. Nguyen, and M. S. Brown, "Quick Approximation of Camera's Spectral Response from Casual Lighting," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2013, pp. 844–851.

[251] "The ColorChecker Pages (Page 2 of 3)." [Online]. Available: http://www.babelcolor.com/colorchecker-2.htm

[252] M. M. Darrodi, G. Finlayson, T. Goodman, and M. Mackiewicz, "Reference data set for camera spectral sensitivity estimation," *J. Opt. Soc. Am. A*, vol. 32, no. 3, pp. 381–391, 2015.

[253] M. Edmonds, J. Yi, N. K. Singa, and L. M. Wang, "Generation of high-density hyperspectral point clouds of crops with robotic multi-camera planning," in *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, 2019, pp. 1475–1480.

[254] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Cambridge, Mass: MIT Press, 2006.

[255] J. Katrašnik, F. Pernuš, and B. Likar, "A method for characterizing illumination systems for hyperspectral imaging," *Opt. Express, OE*, vol. 21, no. 4, pp. 4841–4853, 2013.

[256] S. Wolfert, L. Ge, C. Verdouw, and M.-J. Bogaardt, "Big Data in Smart Farming – A review," *Agricultural Systems*, vol. 153, pp. 69–80, May 2017.

[257] M. Edmonds and J. Yi, "Efficient multi-robot inspection of row crops via kernel estimation and region-based task allocation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8919–8926.

[258] P. Germain, F. Bach, A. Lacoste, and S. Lacoste-Julien, "PAC-Bayesian Theory Meets Bayesian Inference," in *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016, pp. 1884–1892.

[259] M. Edmonds, T. Yigit, and J. Yi, "Resolution-optimal, energy-constrained mission planning for unmanned aerial/ground crop inspections," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, 2021, pp. 2235–2240.

[260] A. C. Kapoutsis *et al.*, "Darp: divide areas algorithm for optimal multi-robot coverage path planning," *J. Intell. Robot. Sys.*, vol. 86, no. 3-4, pp. 663–680, 2017.

[261] M. Edmonds and J. Yi, "A model predictive control based iterative trajectory optimization method for systems with state-like disturbances," in *2019 American Control Conference (ACC)*, 2019, pp. 1635–1640.

[262] ——, "Learning-based near-surface modeling for predictive multirotor landing control," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 711–716, 2021.

[263] S. A. Conyers *et al.*, "An Empirical Evaluation of Ground Effect for Small-Scale Rotorcraft," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2018, pp. 1244–1250.

[264] T. L. Friesz, *Dynamic Optimization and Differential Games*, ser. International Series in Operations Research & Management Science. Springer US, 2010.

[265] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: a parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, Dec. 2014.

[266] Q.-Y. Zhou *et al.*, "Open3D: A Modern Library for 3D Data Processing," *arXiv:1801.09847 [cs]*, Jan. 2018.

[267] S. Prothin *et al.*, "Aerodynamics of MAV rotors in ground and corner effect," *International Journal of Micro Air Vehicles*, vol. 11, p. 1756829319861596, Jan. 2019.

[268] M. Edmonds, T. Yigit, V. Hong, F. Sikandar, and J. Yi, "Optimal trajectories for autonomous human-following carts with gesture-based contactless positioning suggestions," in *2021 American Control Conference (ACC)*, 2021, pp. 3896–3901.

[269] Z. Cao *et al.*, "OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 43, no. 1, pp. 172–186, 2019.

[270] T. Yigit *et al.*, "Wearable IMU-Based Early Limb Lameness Detection for Horses Using Multi-Layer Classifiers," in *Proc. IEEE Conf. Automat. Sci. Eng.*, Hong Kong, 2020, pp. 956–961.