

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN
SCIENZE E TECNOLOGIE AGRARIE, AMBIENTALI E ALIMENTARI

Ciclo 38

Settore Concorsuale: 07/C1 - Ingegneria Agraria, Forestale e dei Biosistemi
Settore Scientifico Disciplinare: AGRI/09 - Meccanica Agraria

OPTICAL SENSORS APPLIED ON VINEYARD MONITORING

Presentata da: Marco Mingrone

Coordinatore Dottorato
Diana Di Gioia

Supervisore
Chiara Cevoli

Co-supervisori
Marco Seracini
Lorenzo Medici

Esame finale anno 2026

Finanziato dall'Unione europea- Next Generation EU, Missione 4, Componente 2, Investimento 3.3 (D.M. 117/2023) CUP J33C22001560009.



Contents

1 Introduction **6**

1.1 Precision viticulture: objectives and most frequent studies 6

1.1.1 Physical basis and validity of NIR spectroscopic analysis. . . . 6

1.1.2 Frequent study themes in PV 7

1.1.3 From sensing to decision. 8

1.1.4 Outlook. 8

1.2 Sensors used: camera types and monitoring approaches 8

1.2.1 Spectral imaging systems and camera architectures 9

1.2.2 Monitoring configurations: remote and proximal sensing . . . 9

1.3 Common challenges in spectral imaging 11

1.3.1 Illumination variability and environmental instability 11

1.3.2 Surface anisotropy, scattering, and specular reflection 11

1.3.3 High dimensionality and data redundancy 12

1.3.4 Spectral and spatial resolution trade-offs 12

1.3.5 Calibration transfer and model generalization 12

1.3.6 Operational constraints 13

1.4 Image calibration 13

1.4.1 Radiometric calibration and reflectance computation 14

1.4.2 Vignetting, sensor non-uniformity, and illumination correction 14

1.4.3 Spectral and temporal normalization 14

1.4.4 Integration of calibration into acquisition workflow 15

1.4.5 Physical meaning and validation 15

1.5 Image segmentation: techniques and algorithms 15

1.5.1 Classical and rule-based segmentation methods 16

1.5.2 Clustering and unsupervised segmentation 16

1.5.3 Machine learning-based segmentation 16

1.5.4 Deep learning and semantic segmentation 17

1.5.5 Spectral spatial segmentation in hyperspectral imagery 17

1.5.6 Evaluation metrics and validation 18

1.5.7 Challenges and perspectives 18

1.6 Predictive models: classification and regression approaches 18

1.6.1 Regression models for quantitative prediction 19

1.6.2 Classification models for categorical analysis 19

1.6.3 Feature extraction, dimensionality reduction, and variable selection 20

1.6.4 Model calibration, validation, and transferability 20

1.6.5 Explainability and integration with agronomic knowledge . . . 21
1.6.6 Perspectives and future directions . . . 21

2 Radiometric Calibration: white reference over distance 25
2.1 Introduction . . . 25
2.1.1 Empirical in-field methods . . . 25
2.1.2 Absolute calibration . . . 26
2.1.3 Direct Reflectance Transformation and model-based methods . 26
2.1.4 Spectral refinement and illumination calibration . . . 26
2.2 Dark-Bright Target Method: distance effect . . . 27
2.3 Pixel camera model . . . 29
2.4 Materials and Methods . . . 31
2.4.1 Multispectral imaging system . . . 31
2.4.2 Radiometric reference and illumination setup . . . 31
2.5 Data treatment . . . 32
2.6 Data Analysis . . . 34
2.7 Results and Discussion . . . 34
2.8 Conclusion . . . 37

3 Field-Based Evaluation of Vineyard Berry Quality Using Multispectral Sensing 40
3.1 Introduction to field multispectral acquisitions for grape quality assessment . . . 40
3.2 Materials and Methods . . . 42
3.2.1 Vineyard . . . 42
3.2.2 Multispectral set up and spectral acquisition . . . 43
3.2.3 Reference chemical parameters . . . 45
3.2.4 Hypercube extraction . . . 46
3.2.5 ROI selection . . . 48
3.2.6 Datastore . . . 49
3.2.7 Model development . . . 49
3.3 Results . . . 50
3.3.1 Chemometric models . . . 51
3.4 Conclusions . . . 58

4 Spectral Reconstruction Applied in Precision Agriculture: On-Field Solutions 60
4.1 Introduction . . . 61
4.2 Literature Search and Study Selection Methodology . . . 63
4.2.1 Databases, Publisher Platforms, and Coverage . . . 63
4.2.2 Search Strategy and Keywords . . . 63
4.2.3 Eligibility Criteria . . . 63
4.2.4 Screening Workflow . . . 63
4.3 Traditional HSI On-Field . . . 65
4.3.1 Limited Acquisition Speed . . . 74
4.3.2 Low Robustness in Outdoor Conditions . . . 75
4.3.3 Low Spatial Resolution . . . 75

4.3.4 HSI Data Management and Complexity . . . 76
4.3.5 Lighting Control . . . 76
4.3.6 Radiometric Image Calibration . . . 76
4.4 Spectral Super-Resolution: Methods, Challenges, and Perspectives . . . 79
4.5 Regression Models from RGB via Spectral Reconstruction . . . 87
4.5.1 Tomato Quality Assessment . . . 88
4.5.2 Sweet Potato Quality Evaluation . . . 89
4.5.3 Glutamate Estimation in Beef . . . 91
4.5.4 Early Detection of Maize Diseases . . . 91
4.5.5 UAV-Based In-Field Phenotyping . . . 92
4.6 Discussion . . . 92
4.7 Conclusions . . . 93

5 Spectral Reconstruction of Hyperspectral Data Acquired in a Vineyard 103
5.1 Introduction . . . 103
5.2 Materials and Methods . . . 104
5.2.1 Field Hyperspectral Image Acquisition . . . 104
5.2.2 Data Pre-processing . . . 104
5.2.3 Dataset Preparation . . . 106
5.2.4 Network Architecture and Training Parameters . . . 107
5.3 Results . . . 108
5.3.1 Training Configuration and Performance . . . 108
5.3.2 Test Performance . . . 109
5.4 Conclusion . . . 110

Appendices 112

A h.estrazione_ipercubo.m 112

B h.processing 114

C camera.app 117

D Class Hypercube 126

E HSCRNN-R 144

F MRAE Loss Class Function 149

Abstract

This doctoral thesis investigates the use of optical imaging systems (RGB, multispectral, and hyperspectral) for non-destructive vineyard monitoring, with particular emphasis on practical and cost-effective field applicability. While most studies in the literature rely on laboratory-based hyperspectral imaging (HSI) systems that are expensive and difficult to deploy under real outdoor conditions, this work focuses on the development and validation of more accessible multispectral solutions capable of operating directly on agricultural machines in the field.

A custom-built multispectral imaging system, mounted on a tractor, was designed and integrated with a dedicated MATLAB-based acquisition software. Laboratory experiments established a robust radiometric calibration procedure based on a fixed white reference, ensuring signal reliability under variable illumination. Field trials conducted in the Cadriano vineyard demonstrated that multispectral data, when properly calibrated and processed, can predict key grape quality indicators, such as soluble solids content, pH, and titratable acidity, with accuracy comparable to hyperspectral systems, yet at a fraction of their cost and complexity.

The final part of the research extended this concept toward the next frontier: spectral super-resolution, or the reconstruction of hyperspectral information from RGB images. A deep residual neural network (HSCNN-R) was trained on vineyard datasets to explore how common RGB cameras could emulate the spectral richness of hyperspectral imaging. This approach points toward a new generation of lightweight, low-cost, and easily deployable sensing systems for precision viticulture.

Overall, the thesis advances both the methodological understanding and the technological feasibility of optical sensing in the field, demonstrating that practical, economically sustainable systems can achieve performance levels once attainable only with laboratory-grade hyperspectral instruments.

Chapter 1

Introduction

1.1 Precision viticulture: objectives and most frequent studies

Precision viticulture adapts the principles of precision agriculture to the specific biophysical, technological, and managerial constraints of vineyards. Its overarching goal is to observe and exploit spatial and temporal variability in order to optimize grape yield and composition while reducing inputs (water, fertilizers, crop protection products) and environmental impact. Practically, it hinges on reliable, repeatable, and timely measurements of key plant and fruit state variables (canopy architecture, water status, nutrient status, and berry quality) followed by data-driven decisions such as variable-rate irrigation, targeted canopy management, and selective harvest [1, 2]. Early adoption of PV was driven by vigor zoning using vegetation indices (e.g., NDVI) from remote platforms, as well as by proximal measurements capturing fine-scale variability that remote sensing could not resolve at the time. Over the last decade, however, the research emphasis has progressively shifted from vigor proxies to *quality* and *physiology*-oriented traits (soluble solids, phenolic composition, acidity, stomatal regulation, and stem water potential), leveraging advances in optical sensing, mobile platforms, and machine learning [3, 4]. In this evolution, multi- and hyperspectral imaging have played a central role by relating reflectance signatures to chemical and structural attributes of leaves and berries through chemometrics and, more recently, deep learning [5, 6, 7].

1.1.1 Physical basis and validity of NIR spectroscopic analysis.

The rationale for correlating near-infrared (NIR) spectral data with physical and biochemical properties lies in the fundamental interaction between electromagnetic radiation and molecular bonds. When incident photons interact with matter, their energy can excite electronic or vibrational transitions in molecules depending on wavelength. In the visible region, photons primarily excite electronic transitions, which are related to pigments such as chlorophylls and anthocyanins. In the infrared region—particularly the NIR between 700 and 2500 nm—the photon energy is lower

and instead induces vibrational overtones and combination bands of fundamental molecular vibrations [5, 6].

These vibrations are associated mainly with X–H bonds (O–H, C–H, and N–H), which are abundant in organic molecules such as water, carbohydrates, proteins, and lipids. Consequently, the reflected or transmitted NIR spectrum encodes information about the chemical composition and physical structure of plant tissues. In grapevine leaves and berries, specific absorption features correspond to the presence and concentration of water, sugars, phenolic compounds, and pigments—attributes that directly reflect the physiological and ripening status of the plant [4, 8]. The resulting spectral signatures can therefore be empirically and mechanistically correlated with measurable traits such as leaf water potential, soluble solids content, and anthocyanin concentration. In this sense, the predictive validity of NIR spectroscopy derives from well-established molecular physics: each pixel in a spectral image captures a reflectance pattern that is a reproducible function of the tissue's molecular composition and its optical path. When calibrated with reference analytical data through chemometric or machine-learning models, these patterns provide robust quantitative estimations of plant physiological parameters [7, 9].

1.1.2 Frequent study themes in PV

Research topics recurrently addressed in PV can be grouped into four categories:

(i) Yield components and early yield prediction. A substantial body of work targets automated quantification of flowers and berries as predictors of yield. Classical computer vision pipelines and, more recently, deep learning have been used to detect and count flowers and berries under field conditions, including on-the-go acquisitions [10, 11, 12, 13]. These studies typically combine robust segmentation (morphological or CNN-based) with statistical models that relate visible organs to total counts (including occluded ones) and to final yield, acknowledging occlusion and illumination variability as primary sources of uncertainty.

(ii) Canopy structure and exposure. The spatial arrangement of gaps, leaves, and fruit exposure within the fruiting zone is a key determinant of microclimate and, ultimately, composition. On-the-go RGB imaging and supervised classification enable the derivation of canopy porosity and the quantification of exposed leaves and clusters, showing good agreement with reference point quadrat analysis and supporting variable-rate canopy operations [14]. These approaches scale to mapping and can inform leaf removal strategies aimed at balancing sunlight exposure and disease risk.

(iii) Water status assessment and irrigation scheduling. Water status is one of the most influential drivers of vegetative growth, yield, and secondary metabolism. Two major optical routes have been investigated. Thermal imaging quantifies canopy temperature to derive stress indices (e.g., CWSI, I_g) that correlate with stem water potential; recent ground-mobile thermal platforms have demonstrated vineyard-scale mapping and external validation across sites [15, 16]. Multispectral and NIR spectroscopy provide a complementary route by exploiting water-sensitive and structure-related bands; proximal on-the-go NIR and combined multispectral–thermal acquisitions have achieved predictive performance suitable for supporting irrigation

decisions under operational conditions [17, 18, 19].

(iv) Berry quality and ripening dynamics. Quantifying soluble solids content (SSC), acidity, and pigments non-destructively is a long-standing objective in PV. Field and near-field hyperspectral imaging, paired with careful calibration and preprocessing (e.g., Savitzky–Golay, SNV/MSC), have enabled robust SSC and pigment prediction and classification of ripeness classes; both push-broom and snapshot modalities have been used in motion, with predictive R^2 values that meet decision-support needs [4, 8, 20]. Beyond composition, proximal multispectral imaging and indices sensitive to chlorophyll or water content have also been explored to track nutrient-related variability at canopy scale and across phenological stages [21].

1.1.3 From sensing to decision.

Across these themes, the common methodological pipeline is: (1) standardized acquisition and calibration; (2) segmentation of regions of interest; (3) feature extraction and/or dimensionality reduction; and (4) predictive modeling (classification or regression) tuned to the viticultural decision at stake. Traditional chemometrics such as Partial Least Squares Regression (PLSR) and Support Vector Regression (SVR) remain the workhorses for quantitative prediction, whereas CNN-based models have gained traction for detection/segmentation and end-to-end inference, provided that sufficient annotated data and robust validation schemes are in place [7, 13]. Increasingly, PV studies emphasize multi-season, multi-site validation and the fusion of proximal with remote measurements to improve generalization.

1.1.4 Outlook.

The field is moving toward integrative PV frameworks that combine multi-scale sensing (UAV–proximal–IoT), modern analytics, and automated platforms to close the loop between measurement and actuation. Within this context, spectral imaging—when paired with rigorous calibration and modeling—serves as a quantitative, scalable, and non-destructive cornerstone for monitoring the key physiological and compositional traits that matter most to vineyard management and wine style [1, 22].

1.2 Sensors used: camera types and monitoring approaches

Optical sensors are the core technological enablers of precision viticulture, allowing the non-destructive measurement of canopy and fruit characteristics through the interaction between light and plant tissues. The evolution from RGB to multispectral and hyperspectral imaging has progressively increased both the spectral richness and the quantitative reliability of vineyard monitoring systems. Depending on their spectral range, acquisition mechanism, and deployment configuration, imaging systems can be classified according to (i) the type of camera used and (ii) the monitoring approach adopted (remote or proximal sensing).

1.2.1 Spectral imaging systems and camera architectures

Multispectral and hyperspectral cameras differ primarily in the number and continuity of the spectral bands they record. Multispectral imaging acquires reflectance information in a discrete number of spectral bands (typically 4–10), often pre-selected to match vegetation indices or specific absorption features. This configuration offers a trade-off between acquisition speed, cost, and spectral sensitivity. Hyperspectral imaging (HSI), by contrast, measures hundreds of contiguous and narrow spectral bands, capturing the continuous reflectance spectrum of each pixel. Such dense sampling enables the detection of subtle spectral variations linked to molecular composition and structural differences in vine organs [5, 6].

Two main imaging architectures are commonly adopted: *line-scan* (push-broom) and *snapshot* systems. In a line-scan camera, a single spatial line is captured at a time, and the third dimension of the hyperspectral cube is reconstructed as the platform moves. These sensors typically employ a dispersive element such as a diffraction grating or a prism to separate wavelengths, providing high spectral fidelity and radiometric accuracy. Push-broom cameras, however, require constant motion and stable illumination, which limit their usability in outdoor or mobile settings [3, 15].

In contrast, snapshot cameras acquire all spectral bands simultaneously for each pixel through the integration of micro-interferometric filters (e.g., Fabry–Pérot) directly on the sensor surface. This design eliminates motion artefacts, drastically increases acquisition speed, and allows the implementation of *on-the-go* systems for proximal monitoring of canopy and berry characteristics [20]. The spectral range of most snapshot sensors lies between 400 and 1000 nm (VNIR), covering the visible and near-infrared regions, where the main absorptions of chlorophyll, carotenoids, and water occur. Although snapshot systems usually provide lower spectral resolution than line-scan instruments, their temporal resolution and robustness to motion make them well suited for real-field and robotic applications [23, 24].

1.2.2 Monitoring configurations: remote and proximal sensing

Spectral monitoring in viticulture operates within two complementary frameworks: remote sensing and proximal sensing.

Remote sensing. Remote sensing includes satellite, aircraft, and unmanned aerial vehicle (UAV) platforms that acquire reflectance data over large vineyard areas. The main advantages of this configuration are the capacity to perform spatially comprehensive mapping and to detect large-scale variability in vigor, water status, or nutrient content [25, 26]. UAV-based hyperspectral sensors, operating in the VNIR or SWIR regions, enable higher spatial resolution and flexible acquisition timing compared to satellite systems, supporting fine-scale zoning of canopy health and disease symptoms. However, atmospheric disturbances, illumination variability, and geometric distortions can limit radiometric accuracy, requiring careful pre-processing and calibration protocols [22].

Proximal sensing. Proximal sensing refers to sensors mounted on ground platforms—such as tractors, robots, or handheld devices—placed at short distances (0.3–2.0 m) from the target. These systems offer high spatial and spectral resolution, enabling the detection of physiological and compositional traits at the organ level. The proximity reduces atmospheric interference, facilitates radiometric standardization, and allows for multi-modal integration (e.g., thermal, multispectral, and environmental data). Ground-based hyperspectral imaging has been applied to estimate canopy porosity, berry ripeness, and water status with precision comparable to laboratory measurements [4, 14, 18]. Furthermore, tractor-mounted platforms and ground robots can execute repeated scans throughout the growing season, capturing temporal dynamics that are unattainable through UAVs [27, 23].

Hybrid and IoT-based systems. Recent developments have moved toward hybrid configurations that merge the advantages of both scales. For example, aerial multispectral data provide spatial context for vineyard zoning, while proximal sensors deliver detailed ground truth for model calibration. IoT-based spectral devices have further extended the concept of continuous monitoring, integrating miniaturized sensors with wireless communication and cloud analytics. These autonomous systems are capable of tracking grape ripening and microclimate conditions in real time, bridging temporal gaps between UAV surveys and offering new possibilities for autonomous decision-making in vineyard management [24].

Spectral range and vegetation indices. The selection of the spectral range is dictated by the physiological trait of interest. The visible range (400–700 nm) is sensitive to pigments such as chlorophyll and anthocyanins, providing indicators of photosynthetic activity and ripening. The near-infrared region (700–2500 nm) responds primarily to water and structural compounds, thus informing about leaf turgor and internal tissue composition [6, 9]. Many vegetation indices derived from these regions, such as NDVI, PRI, and red-edge indices, have been adapted to proximal sensors, linking spectral reflectance ratios to physiological functions like chlorophyll fluorescence, photochemical efficiency, and canopy water status [21, 1].

Integration and data fusion. The convergence of spectral imaging with other sensing modalities—thermal cameras, LiDAR, and environmental probes—has become an emerging research frontier. Multi-sensor data fusion can enhance the interpretability and robustness of spectral measurements by combining structural and radiometric information. For instance, integrating hyperspectral imaging with canopy temperature and microclimatic data enables the simultaneous estimation of vine water potential and stress indicators with improved accuracy [18, 19]. Similarly, fusing spectral data with 3D canopy models from LiDAR or stereo vision facilitates the correction of shading and occlusion effects, enhancing the physical validity of reflectance-based indices. Overall, the diversity of optical sensors now available for vineyard monitoring provides a flexible technological ecosystem that can be adapted to the spatial, temporal, and physiological scales of interest. The ongoing miniaturization of components, improvements in sensor sensitivity, and the rise of edge

computing are expected to make high-resolution spectral monitoring increasingly accessible and autonomous within precision viticulture frameworks.

1.3 Common challenges in spectral imaging

Despite the rapid progress in optical technologies and data analytics, several challenges still constrain the operational deployment of spectral imaging in precision viticulture. These challenges arise from the intrinsic complexity of outdoor acquisitions, the physical properties of plant tissues, and the high dimensionality of hyperspectral data. Addressing them is crucial to ensure data reproducibility, model transferability, and the physical interpretability of derived indices.

1.3.1 Illumination variability and environmental instability

Field acquisitions are unavoidably affected by variations in natural illumination. Changes in solar angle, cloud cover, and canopy self-shading cause fluctuations in the irradiance received by the sensor, leading to inconsistencies in the recorded reflectance. In proximal sensing, illumination instability can generate artificial differences among frames, especially when data are collected under mixed sunlight and shadow conditions. These variations propagate into derived vegetation indices and prediction models, reducing their robustness and comparability across dates and sites [9, 20].

Different approaches have been proposed to mitigate illumination effects. Empirical line calibration, which uses reference panels of known reflectance within the scene, allows for the normalization of measured radiance values. Alternatively, relative calibration methods exploit invariant features, such as soil reflectance or non-varying canopy zones, to correct for illumination drift. Some recent systems integrate artificial light sources (halogen or LED arrays) into proximal platforms to provide stable and uniform illumination, thus decoupling the acquisition from solar conditions [23]. However, such active illumination solutions increase energy demand and require careful synchronization with the imaging system.

1.3.2 Surface anisotropy, scattering, and specular reflection

Plant tissues are optically heterogeneous and anisotropic. Leaf orientation, surface roughness, and the waxy cuticle of berries all contribute to complex light–matter interactions, producing both diffuse and specular reflection components. These phenomena cause local variations in apparent brightness that are not directly related to chemical composition. For example, the glaucous layer on grape berries (pruina) scatters light strongly, while leaf microstructure and water content modulate absorption in the near-infrared region [28, 16].

Scattering and specular reflection distort the linear relationship between reflectance and analyte concentration assumed by most chemometric models. Several pre-processing and correction techniques have therefore been developed, including multiplicative scatter correction (MSC), standard normal variate (SNV), and detrending transformations [8]. These methods adjust for baseline shifts and multiplicative

effects, isolating the chemical signal from geometric and optical artefacts. Nonetheless, their efficiency depends on proper calibration and noise management at the acquisition stage.

1.3.3 High dimensionality and data redundancy

Hyperspectral imaging typically produces hundreds of spectral bands per pixel, resulting in massive and highly collinear datasets. The spectral signatures of plant tissues often exhibit broad, overlapping absorption features due to the presence of numerous X–H bonds (O–H, C–H, N–H). This multicollinearity complicates model development and increases the risk of overfitting, especially when the number of spectral variables greatly exceeds the number of calibration samples [29].

Dimensionality reduction techniques are therefore essential to extract the most informative features. Classical approaches such as Principal Component Analysis (PCA) project the original data into lower-dimensional subspaces that maximize variance or covariance with the target variable. More recent methods, including wavelet transforms, genetic algorithms, and deep feature extraction using CNNs, have been applied to select relevant wavelengths while preserving spatial coherence [7, 13]. Efficient feature selection not only reduces computational demand but also improves the physical interpretability of predictive models by highlighting specific absorption regions associated with biochemical compounds.

1.3.4 Spectral and spatial resolution trade-offs

The design of spectral imaging systems inherently involves a compromise between spectral, spatial, and temporal resolution. High spectral resolution (narrow bandwidths) allows for the discrimination of fine absorption features but often requires longer integration times and produces lower signal-to-noise ratios (SNR). Conversely, high spatial resolution reduces the photon flux per pixel and may increase noise sensitivity. In field applications, this trade-off is compounded by platform motion, vibration, and ambient lighting variations, which can blur or distort hyperspectral cubes [4, 3]. Snapshot sensors partially mitigate these limitations by acquiring all bands simultaneously, but they typically offer fewer spectral channels and lower radiometric depth.

Instrument manufacturers and researchers are thus optimizing optical designs and acquisition strategies to balance these trade-offs. Examples include adaptive exposure control, sensor binning to increase SNR, and hybrid configurations combining hyperspectral cameras with RGB or multispectral sensors to enhance both spatial and spectral information content [20]. The combination of different modalities through data fusion has also emerged as a strategy to overcome individual sensor limitations, improving overall scene characterization.

1.3.5 Calibration transfer and model generalization

A persistent challenge in PV is the *transferability* of models developed under specific environmental and instrumental conditions. Spectral response depends on illumination geometry, sensor optics, and acquisition configuration; consequently, a model

trained on one platform or season may not generalize well to others. Calibration transfer techniques, such as piecewise direct standardization (PDS) or orthogonal signal correction (OSC), have been introduced to align spectral spaces between instruments [5, 9]. However, robust cross-season and cross-platform generalization still requires extensive datasets that capture variability in environmental conditions and plant phenology.

Recent research emphasizes multi-year data collection and the use of transfer learning to improve model adaptability. By fine-tuning neural networks on small, domain-specific datasets, models can leverage previously learned spectral–chemical relationships while accommodating sensor-specific variations [7]. Such approaches, coupled with standardized calibration procedures and open spectral libraries, represent a promising path toward operational and interoperable spectral monitoring in viticulture.

1.3.6 Operational constraints

Finally, operational factors such as acquisition speed, data storage, and processing complexity remain non-trivial barriers. Hyperspectral datasets can exceed several gigabytes per hectare, requiring high-throughput storage and computational resources for real-time analysis. On-the-go imaging further demands robust synchronization between motion sensors, GPS, and cameras to ensure georeferenced accuracy. Moreover, environmental factors—wind-induced canopy motion, dust, or changing humidity—can introduce additional artefacts that degrade data quality [23]. Continuous efforts are therefore directed toward simplifying system setup, automating calibration, and integrating edge-computing solutions that allow preliminary data processing directly on-board the acquisition platform.

In summary, spectral imaging in viticulture faces a complex combination of physical, environmental, and analytical challenges. While methodological advances in calibration, pre-processing, and data modeling continue to mitigate many of these limitations, achieving consistent, reproducible, and transferable results across vineyards and seasons remains an active research frontier.

1.4 Image calibration

Accurate image calibration is an essential prerequisite for any quantitative use of spectral data. The digital numbers (DN) recorded by imaging sensors correspond to radiance values that depend on both the intrinsic properties of the target and on external factors, such as illumination, sensor response, optical geometry, and atmospheric or environmental conditions. To compare acquisitions made at different times, locations, or by different instruments, these DN values must be converted into standardized reflectance values. Reflectance calibration aims to isolate the spectral response of the object itself, enabling physically meaningful interpretation and reliable correlations with biochemical or structural traits [18, 20].

1.4.1 Radiometric calibration and reflectance computation

Radiometric calibration transforms raw pixel intensities into physical units of reflectance or radiance. The standard method relies on the acquisition of two reference images: a white reference (highly reflective Lambertian surface) and a dark reference (sensor offset with the lens closed or a black target). The normalized reflectance $R_i(x, y)$ for each spectral band i and pixel (x, y) is computed as:

$$R_i(x, y) = \frac{DN_i(x, y) - DN_{dark}(x, y)}{DN_{white}(x, y) - DN_{dark}(x, y)} \tag{1.1}$$

This normalization compensates for variations in illumination intensity, sensor gain, and offset, producing values between 0 (no reflection) and 1 (full reflection). The accuracy of this conversion depends on the stability and homogeneity of the white reference and on the constancy of environmental conditions between reference and target acquisitions. In proximal sensing, where illumination can change rapidly due to clouds or canopy shadowing, dynamic calibration using real-time white reference panels mounted near the field of view has proven effective [20, 23].

1.4.2 Vignetting, sensor non-uniformity, and illumination correction

Vignetting, or the gradual decrease in brightness toward the image periphery, is a common artifact caused by lens geometry and filter transmission characteristics. Its correction is critical for maintaining radiometric consistency, particularly in snapshot sensors with wide-angle lenses or multiple micro-filters. A practical approach is to divide each acquired frame by a reference image obtained from a uniformly illuminated surface, creating a pixel-wise correction map that normalizes brightness across the field of view.

Illumination non-uniformity, especially under natural sunlight, can also be mitigated using polynomial surface fitting to approximate the spatial illumination gradient and remove it from the data cube. When imaging vine canopies, where the geometry of leaves and clusters introduces self-shading, multi-view or bidirectional acquisitions can reduce angular dependence and improve reflectance stability [3, 14].

To correct these effects, several spectral pre-processing techniques are applied prior to modeling. The standard normal variate (SNV) transformation removes multiplicative effects by centering and scaling each spectrum, while multiplicative scatter correction (MSC) aligns each spectrum to a reference mean spectrum through a linear regression approach. Derivative filtering, particularly the first or second derivative computed via the Savitzky–Golay algorithm, enhances weak absorption features and corrects baseline shifts due to surface irregularities [7, 8]. In some cases, a combination of these methods is used sequentially to optimize the balance between noise reduction and spectral fidelity.

1.4.3 Spectral and temporal normalization

In time-series or multi-platform analyses, it is necessary to ensure the comparability of reflectance spectra acquired under different conditions. Spectral normalization

can be achieved through linear scaling of intensities based on invariant spectral regions (e.g., near 550 nm for vegetation). Temporal normalization compensates for day-to-day variations in solar irradiance by adjusting measurements relative to a radiometric baseline or by incorporating solar radiation sensors co-located with the imaging device. Such corrections are particularly important when correlating reflectance with physiological indicators such as leaf water potential or berry sugar content, which require sub-percent precision in spectral response to capture subtle changes [18, 19].

1.4.4 Integration of calibration into acquisition workflow

A practical challenge in operational spectral imaging is integrating calibration procedures into high-throughput acquisition workflows. In on-the-go vineyard applications, where imaging speed often exceeds 5 km/h, calibration cannot interrupt the acquisition process. Automated exposure control, real-time white reference acquisition, and embedded processing units capable of performing radiometric normalization in situ are therefore becoming standard. Some robotic platforms now integrate both calibration and pre-processing pipelines directly on board, reducing the need for post-acquisition corrections and enabling near real-time reflectance mapping in the field [24, 27].

1.4.5 Physical meaning and validation

Ultimately, calibration ensures that the recorded signal corresponds to a physically meaningful property of the target: the fraction of incident radiation reflected as a function of wavelength. When accurately performed, reflectance calibration enables the empirical correlation of spectral features with measurable physical and biochemical attributes, uch as pigment concentration, water status, or surface morphology—grounded in the principles of vibrational spectroscopy and radiative transfer. Validated calibration protocols are thus the foundation upon which reliable spectral, biophysical models are built, allowing reflectance images to serve as quantitative proxies of plant condition in precision viticulture [4, 9].

1.5 Image segmentation: techniques and algorithms

Image segmentation is a critical stage in the spectral imaging workflow, as it defines the regions of interest (ROIs) from which spectral or spatial features are extracted. In viticulture, segmentation aims to isolate leaves, clusters, and canopy gaps from the background and other non-relevant objects such as sky, soil, or trellis components. The accuracy of segmentation directly determines the quality of the derived indices, chemometric models, and classification results. However, vineyard scenes pose substantial challenges due to natural illumination variability, occlusions, and heterogeneous textures.

1.5.1 Classical and rule-based segmentation methods

Traditional segmentation methods rely on color information and simple image processing rules to separate object classes. Early applications used thresholding on color spaces such as RGB, HSV, or CIELAB, exploiting the contrast between vegetation and background [10]. In these approaches, specific color channels are selected based on their discriminatory power, and global or adaptive thresholds are applied to create binary masks. The method is computationally efficient but sensitive to illumination and background variability.

Color-based segmentation has been refined through the use of mathematical morphology. Morphological operators such as erosion, dilation, opening, and closing remove small artefacts and connect adjacent structures, producing more homogeneous masks. In the study of grapevine inflorescences, Aquino et al. [10] combined color filtering with morphological reconstruction to detect individual flowers without requiring artificial backgrounds, achieving high accuracy even under uncontrolled lighting. Similarly, Diago et al. [14] used morphological operations to enhance the segmentation of canopy elements, gaps, leaves, and bunches, captured by RGB cameras during nighttime acquisitions with artificial illumination.

Other rule-based techniques include region growing, edge detection (Canny, Sobel), and watershed segmentation. The watershed algorithm, in particular, has proven effective for separating overlapping berries or flowers, using gradient information to delineate boundaries between contiguous objects. These methods are robust in controlled conditions but may fail in highly variable vineyard environments where color and texture overlap significantly among classes.

1.5.2 Clustering and unsupervised segmentation

When explicit thresholds or rules are inadequate, unsupervised clustering methods can group pixels with similar spectral or color properties without prior labels. Common algorithms include k-means, fuzzy c-means (FCM), and self-organizing maps (SOMs). These techniques have been used to partition hyperspectral or multispectral data into homogeneous regions, often as a pre-processing step before classification or regression.

Unsupervised segmentation is particularly valuable for hyperspectral imagery, where hundreds of bands create a high-dimensional feature space. Dimensionality reduction using PCA or Minimum Noise Fraction (MNF) transformation is often performed prior to clustering to reduce noise and highlight dominant spectral variance. Despite their simplicity, clustering methods are limited by their sensitivity to initialization and by the assumption that clusters are compact and separable in the feature space. Nonetheless, they provide a foundation for automated segmentation when ground-truth data are limited [5, 6].

1.5.3 Machine learning-based segmentation

Supervised learning approaches overcome some of the limitations of rule-based and unsupervised methods by leveraging labeled training data. In this paradigm, spectral and spatial features are extracted from each pixel and used to train a classifier that

assigns class labels. Common algorithms include Support Vector Machines (SVM), Random Forests (RF), and k-Nearest Neighbors (kNN). These methods have been successfully applied to segment canopy elements, discriminate clusters from leaves, and identify stressed vegetation [12, 14].

For example, SVM classifiers trained on combined RGB and HSV color features achieved recall values above 85% in identifying bunches and leaves in field images, outperforming single color-space approaches [14]. Random Forests and ensemble classifiers further improve robustness to illumination and noise by integrating multiple decision boundaries. However, these algorithms require representative training datasets that capture the natural variability of vineyard conditions to ensure reliable generalization.

1.5.4 Deep learning and semantic segmentation

Recent advances in computer vision have introduced deep learning architectures that perform end-to-end segmentation through CNNs. These models learn hierarchical features directly from raw image data, eliminating the need for manual feature extraction. Fully Convolutional Networks (FCNs), U-Net, and SegNet architectures are among the most commonly used for agricultural applications [12, 30].

In viticulture, Palacios et al. [12] implemented a SegNet-based approach for automatic flower detection and counting using on-the-go RGB imaging under night-time illumination. The model achieved precision and recall above 90% and 85%, respectively, demonstrating the ability of CNNs to cope with complex backgrounds and occlusions. Deep learning has also been applied to berry detection, canopy segmentation, and disease mapping, providing accurate spatial delineation even in heterogeneous field conditions.

Despite their superior performance, deep models require large, well-annotated datasets and considerable computational resources. Data augmentation (e.g., rotation, scaling, illumination variation) is often used to increase the diversity of training samples. Transfer learning from pre-trained networks can further reduce the need for large agricultural datasets, enabling the reuse of generic visual features learned from natural images. Hybrid architectures that combine convolutional layers with spectral attention mechanisms have recently shown promise for hyperspectral segmentation, capturing both spatial and spectral dependencies.

1.5.5 Spectral spatial segmentation in hyperspectral imagery

Unlike RGB or multispectral data, hyperspectral imagery contains hundreds of narrow spectral bands, allowing segmentation to exploit not only color and texture but also spectral shape information. Spectral spatial segmentation integrates both domains: spectral similarity identifies materials with comparable composition, while spatial continuity ensures coherence across neighboring pixels. Techniques such as three-dimensional (3D) convolutional networks, superpixel segmentation, and Markov random fields (MRFs) are widely used to model these relationships [7, 9].

Superpixel-based methods divide the image into small homogeneous patches based on spectral distance metrics (e.g., Euclidean or Mahalanobis distance) and spatial

proximity. These patches serve as processing units for subsequent classification or regression, reducing noise sensitivity and computational cost. Deep 3D CNNs, in contrast, process both spectral and spatial dimensions simultaneously, learning complex correlations that traditional 2D models cannot capture.

1.5.6 Evaluation metrics and validation

The accuracy of segmentation is typically assessed using metrics derived from confusion matrices, such as precision, recall, F1-score, and overall accuracy. Intersection over Union (IoU) or the Dice coefficient are also employed in pixel-wise classification to quantify the overlap between predicted and reference masks. In vineyard applications, segmentation models are often validated against manual annotations or physical measurements (e.g., actual flower or berry counts, canopy porosity from point quadrat analysis). Proper validation is essential to ensure that segmentation outputs are not only visually plausible but also quantitatively meaningful for subsequent agronomic modeling [10, 12].

1.5.7 Challenges and perspectives

Despite significant progress, several open challenges remain in vineyard image segmentation. High variability in lighting, occlusions from overlapping leaves and clusters, and the presence of complex backgrounds continue to hinder robust performance. Moreover, models trained in one vineyard or season often fail to generalize to others due to differences in cultivar morphology and environmental conditions. Current research focuses on domain adaptation, active learning, and semi-supervised methods to reduce dependence on large annotated datasets.

In parallel, the fusion of multispectral or hyperspectral data with depth information from LiDAR or stereo imaging is emerging as a strategy to improve segmentation robustness by adding structural context. Such multimodal approaches enable a more complete description of the canopy and its spatial organization, facilitating accurate feature extraction for downstream modeling. As imaging sensors and computational models co-evolve, segmentation in precision viticulture is progressively shifting from purely image-based classification to physically grounded, multi-sensor scene understanding.

1.6 Predictive models: classification and regression approaches

Once images have been properly calibrated, segmented, and pre-processed, the next step in the spectral imaging workflow is to extract quantitative or categorical information about plant physiology, fruit composition, and canopy condition. Predictive modeling establishes a mathematical relationship between the acquired spectral data (independent variables) and the target traits (dependent variables), enabling the estimation or classification of vineyard parameters under field conditions. Depending on the objective, models can be broadly categorized into *regression*—to predict

continuous variables—and *classification*—to assign discrete categories or classes. Both approaches rely on statistical, machine learning, or deep learning frameworks that differ in interpretability, computational demand, and data requirements.

1.6.1 Regression models for quantitative prediction

Regression techniques are the backbone of quantitative spectroscopy. In viticulture, they are employed to predict continuous variables such as soluble solids content (SSC), acidity, anthocyanins, or water potential from spectral reflectance or absorbance data. The most widely used method is PLSR, which projects high-dimensional spectral data into a lower-dimensional latent space maximizing covariance with the target variable [31]. PLSR is robust to multicollinearity and provides interpretable loading vectors that can be associated with specific wavelength regions, facilitating the physiological interpretation of spectral features.

Alternative linear approaches include Principal Component Regression (PCR) and Multiple Linear Regression (MLR), which are effective for low-dimensional multispectral data. However, when nonlinear relationships prevail, as in the case of pigment concentration or water content—nonlinear regression methods such as Support Vector Regression (SVR), Gaussian Process Regression (GPR), and Random Forest Regression have been successfully adopted [17, 29]. These methods model complex spectral, biochemical relationships by mapping input data into high-dimensional feature spaces or by aggregating ensemble predictors.

In recent years, ANNs and deep learning architectures have demonstrated their ability to learn intricate, non-linear spectral-chemical mappings without explicit feature extraction [7]. Feed-forward multilayer perceptrons (MLPs) and CNNs have been trained to predict SSC, anthocyanin concentration, and pH with determination coefficients (R^2) above 0.9 under field conditions. The main advantage of neural networks lies in their flexibility and automatic feature learning; however, they require large, representative datasets to prevent overfitting and ensure generalization across vineyards and seasons.

1.6.2 Classification models for categorical analysis

Classification models are designed to assign observations to predefined classes, such as ripeness stages, stress levels, or disease categories. In precision viticulture, classification tasks include the discrimination of canopy elements (leaf, berry, gap), the identification of ripeness categories, and the detection of water stress or nutrient deficiencies [12, 14].

Support Vector Machines (SVMs) are among the most common algorithms, using kernel functions to create non-linear decision boundaries that maximize class separation in the feature space. They have shown strong generalization performance even with limited training data. Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are simpler statistical classifiers that assume Gaussian class distributions and are effective for low-dimensional, normally distributed features. Ensemble methods such as Random Forests (RF) and Gradient Boosting Machines (GBM) combine multiple weak learners to enhance robustness against noise and

overfitting, and have been used to classify vine vigor zones, canopy porosity levels, and water status categories [16, 19].

Deep learning has further extended the capabilities of classification models through convolutional and recurrent neural networks. CNNs can jointly analyze spatial and spectral patterns, while hybrid 1D-2D CNN architectures exploit both the spectral signature of each pixel and the texture of its local neighborhood. Recurrent networks such as Long Short-Term Memory (LSTM) models have also been applied to time-series spectral data for monitoring phenological development. These architectures provide high accuracy but require substantial computational resources and annotated data, emphasizing the need for well-curated spectral libraries.

1.6.3 Feature extraction, dimensionality reduction, and variable selection

Regardless of the modeling paradigm, the success of predictive analysis depends strongly on the quality of input features. Hyperspectral data often contain hundreds of redundant or noisy variables; therefore, dimensionality reduction and wavelength selection are indispensable steps [29]. Traditional feature extraction methods such as PCA, PLS, or Independent Component Analysis (ICA) compress the data into orthogonal or statistically independent components. Alternatively, heuristic or optimization-based selection methods, such as genetic algorithms, successive projections algorithm (SPA), or variable importance in projection (VIP), identify the most informative wavelengths associated with target properties. These approaches not only improve computational efficiency but also enhance model interpretability by highlighting specific spectral regions linked to chemical constituents like water, sugars, or pigments.

In deep learning frameworks, feature extraction is performed automatically by convolutional filters or attention mechanisms that adaptively emphasize informative spectral bands. For instance, attention-based CNNs can dynamically weight spectral channels according to their contribution to the prediction task, mimicking variable selection in classical chemometrics but within a data-driven context.

1.6.4 Model calibration, validation, and transferability

Model calibration requires a representative and well-distributed dataset covering the expected range of variability in both spectral and reference data. The performance of regression and classification models is evaluated using statistical metrics such as the coefficient of determination (R^2), root mean square error (RMSE), residual predictive deviation (RPD), accuracy, precision, and recall. Cross-validation, bootstrapping, or independent test sets are standard practices to assess model generalization. However, achieving reproducibility across platforms and vintages remains a challenge due to variations in illumination, sensor characteristics, and environmental conditions [9, 5].

To address the issue of calibration transfer, techniques such as piecewise direct standardization (PDS), orthogonal signal correction (OSC), and external parameter orthogonalization (EPO) are used to align spectral spaces from different instruments or acquisition setups. Recently, transfer learning has emerged as a powerful strategy

for deep models, enabling them to adapt pre-trained networks to new datasets with limited fine-tuning [7, 13]. Domain adaptation methods further adjust model parameters to account for shifts in illumination spectra or sensor response, improving inter-season and inter-vineyard robustness.

1.6.5 Explainability and integration with agronomic knowledge

An emerging focus in predictive modeling is the interpretability of results. While deep learning and complex ensemble models achieve high predictive performance, their internal mechanisms are often opaque. Explainable artificial intelligence (XAI) techniques, such as saliency maps, layer-wise relevance propagation, and Shapley additive explanations (SHAP), are being applied to spectral models to identify which wavelengths or regions contribute most to predictions. This interpretability bridges the gap between data-driven predictions and physiological understanding, ensuring that models remain grounded in viticultural knowledge [6, 5].

Integrating predictive models with mechanistic and empirical agronomic knowledge allows the translation of spectral information into actionable vineyard management decisions. For example, thresholds in predicted water potential can inform irrigation scheduling, while maps of predicted SSC or anthocyanin concentration can guide selective harvesting. The coupling of spectral models with geographic information systems (GIS) and decision-support tools closes the loop between sensing, modeling, and management, making spectral imaging a central component of next-generation precision viticulture [1, 2].

1.6.6 Perspectives and future directions

The convergence of spectral imaging, machine learning, and automation is redefining vineyard monitoring from a reactive to a predictive discipline. The development of hybrid models that combine physical radiative transfer principles with data-driven learning promises to enhance model robustness and interpretability. Moreover, the increasing availability of open datasets and computational tools will accelerate reproducibility and benchmarking in the field.

Ultimately, predictive models in precision viticulture are moving toward integrative frameworks capable of fusing data across spatial scales (leaf, canopy, plot) and modalities (spectral, thermal, structural). As hardware miniaturization, edge computing, and real-time analytics continue to advance, the operational deployment of spectral predictive models will enable continuous, autonomous monitoring of vine physiology and fruit quality, supporting sustainable and data-driven vineyard management.

Bibliography

[1] Javier Tardaguila et al. “Smart Applications and Digital Technologies in Viticulture: A Review”. In: *Smart Agricultural Technology* 1 (2021), p. 100005. DOI: 10.1016/j.atech.2021.100005.

[2] Massimo Vincenzo Ferro and Pietro Catania. "Technologies and Innovative Methods for Precision Viticulture: A Comprehensive Review". In: *Horticulturae* 9.3 (2023), p. 399. DOI: 10.3390/horticulturae9030399.

[3] Alessandro Matese et al. "Assessing Grapevine Biophysical Parameters From Unmanned Aerial Vehicles Hyperspectral Imagery". In: *Frontiers in Plant Science* 13 (2022), p. 898722. DOI: 10.3389/fpls.2022.898722.

[4] Alessandro Benelli et al. "Precision Viticulture: Automatic Selection of the Regions of Interest from Moving Wagon Hyperspectral Images of Grapes for Efficient SSC Prediction". In: *Smart Agricultural Technology* 7 (2024), p. 100434. DOI: 10.1016/j.atech.2024.100434.

[5] Gamal M. ElMasry, Da-Wen Sun, and Paul Allen. "Principles and Applications of Hyperspectral Imaging in Agro-Food Product Quality Evaluation: A Review". In: *Critical Reviews in Food Science and Nutrition* 52.11 (2012), pp. 999–1023. DOI: 10.1080/10408398.2010.543495.

[6] Renfu Lu and Bosoon Park. "Introduction". In: *Hyperspectral Imaging Technology in Food and Agriculture*. Ed. by Bosoon Park and Renfu Lu. New York, NY: Springer, 2015, pp. 3–7. ISBN: 978-1-4939-2836-1. DOI: 10.1007/978-1-4939-2836-1_1.

[7] Véronique Gomes, Ana Mendes-Ferreira, and Pedro Melo-Pinto. "Application of Hyperspectral Imaging and Deep Learning for Robust Prediction of Sugar and pH Levels in Wine Grape Berries". In: *Sensors* 21.10 (2021), p. 3459. DOI: 10.3390/s21103459.

[8] Armando M. Fernandes et al. "Brix, pH and anthocyanin content determination in whole Port wine grape berries by hyperspectral imaging and neural networks". In: *Computers and Electronics in Agriculture* 115 (2015), pp. 88–96. DOI: 10.1016/j.compag.2015.05.013.

[9] Puneet Mishra et al. "Close-range hyperspectral imaging of plants: A review". In: *Biosystems Engineering* 164 (2017), pp. 49–67. DOI: 10.1016/j.biosystemseng.2017.09.009.

[10] Arturo Aquino et al. "Grapevine flower estimation by applying artificial vision techniques on images with uncontrolled scene and multi-model analysis". In: *Computers and Electronics in Agriculture* 119 (2015), pp. 92–104. DOI: 10.1016/j.compag.2015.10.001.

[11] Borja Millán et al. "Image analysis-based modelling for flower number estimation in grapevine". In: *Journal of the Science of Food and Agriculture* 97.3 (2017), pp. 784–792. DOI: 10.1002/jsfa.7793.

[12] Fernando Palacios et al. "Automated grapevine flower detection and quantification method based on computer vision and deep learning from on-the-go imaging using a mobile sensing platform under field conditions". In: *Computers and Electronics in Agriculture* 178 (2020), p. 105788. DOI: 10.1016/j.compag.2020.105788.

[13] Alexander G. Olenskyj et al. “End-to-End Deep Learning for Directly Estimating Grape Yield from Ground-Based Imagery”. In: *Preprint, arXiv* (2022). eprint: 2208.02394. URL: https://arxiv.org/abs/2208.02394.

[14] María P. Diago et al. “On-the-go assessment of vineyard canopy porosity, bunch and leaf exposure by image analysis”. In: *Australian Journal of Grape and Wine Research* 25.3 (2019), pp. 363–374. DOI: 10.1111/ajgw.12404.

[15] Salvador Gutiérrez et al. “Vineyard water status assessment using on-the-go thermal imaging and machine learning”. In: *PLOS ONE* 13.2 (2018), e0192037. DOI: 10.1371/journal.pone.0192037.

[16] Salvador Gutiérrez et al. “Assessing and mapping vineyard water status using a ground mobile thermal imaging platform”. In: *Irrigation Science* 39 (2021), pp. 457–468. DOI: 10.1007/s00271-021-00735-1.

[17] María P. Diago et al. “Development and Validation of a New Methodology to Assess the Vineyard Water Status by On-the-Go Near Infrared Spectroscopy”. In: *Frontiers in Plant Science* 9 (2018), p. 59. DOI: 10.3389/fpls.2018.00059.

[18] María Paz Diago et al. “Combination of multispectral imagery, environmental data and thermography for on-the-go monitoring of the grapevine water status in commercial vineyards”. In: *European Journal of Agronomy* 140 (2022), p. 126586. DOI: 10.1016/j.eja.2022.126586.

[19] Andrea Zorer et al. “Modelling Water Stress in a Shiraz Vineyard Using Proximal and Remote Sensing Data”. In: *Agricultural Water Management* 285 (2023), p. 108312. DOI: 10.1016/j.agwat.2023.108312.

[20] Riccardo Bertoglio et al. “On-the-go Table Grape Ripeness Estimation via Proximal Snapshot Hyperspectral Imaging”. In: *Computers and Electronics in Agriculture* 226 (2024), p. 109354.

[21] Xuelian Peng et al. “Prediction of the Nitrogen, Phosphorus and Potassium Contents in Grape Leaves at Different Growth Stages Based on UAV Multispectral Remote Sensing”. In: *Remote Sensing* 14.11 (2022), p. 2659. DOI: 10.3390/rs14112659.

[22] Angelos Alexopoulos et al. “Complementary Use of Ground-Based Proximal Sensing and Airborne/Spaceborne Remote Sensing Techniques in Precision Agriculture: A Systematic Review”. In: *Agronomy* 13.7 (2023), p. 1942. DOI: 10.3390/agronomy13071942.

[23] Florian Rançon et al. “Designing a Proximal Sensing Camera Acquisition System for Vineyard Applications: Results and Feedback on 8 Years of Experiments”. In: *Sensors* 23.2 (2023), p. 847. DOI: 10.3390/s23020847.

[24] Hugo M. Oliveira et al. “An Autonomous Internet of Things Spectral Sensing System for In-Situ Optical Monitoring of Grape Ripening: Design, Characterization, and Operation”. In: *Computers and Electronics in Agriculture* 217 (2024), p. 108599. DOI: 10.1016/j.compag.2023.108599.

[25] Alfonso López et al. “Classification of Grapevine Varieties Using UAV Hyperspectral Imaging”. In: *Remote Sensing* 16.12 (2024), p. 2103. DOI: 10.3390/rs16122103.

[26] Yeniu Mickey Wang, Bertram Ostendorf, and Vinay Pagay. “Evaluating the Potential of High-Resolution Hyperspectral UAV Imagery for Grapevine Viral Disease Detection in Australian Vineyards”. In: *International Journal of Applied Earth Observation and Geoinformation* 130 (2024), p. 103876. DOI: 10.1016/j.jag.2024.103876.

[27] Juan Fernández-Novales et al. “Monitoring and Mapping Vineyard Water Status Using Non-Invasive Technologies by a Ground Robot”. In: *Remote Sensing* 13.14 (2021), p. 2830. DOI: 10.3390/rs13142830.

[28] Juan Fernández-Novales et al. “In-field quantification and discrimination of different vineyard water regimes by on-the-go NIR spectroscopy”. In: *Biosystems Engineering* 165 (2018), pp. 47–58. DOI: 10.1016/j.biosystemseng.2017.08.018.

[29] Rui Silva and Pedro Melo-Pinto. “A review of different dimensionality reduction methods for the prediction of sugar content from hyperspectral images of wine grape berries”. In: *Applied Soft Computing* 113 (2021), p. 107889. DOI: 10.1016/j.asoc.2021.107889.

[30] Zifei Luo et al. “Semantic Segmentation of Agricultural Images: A Survey”. In: *Information Processing in Agriculture* 11 (2024), pp. 172–186. DOI: 10.1016/j.inpa.2023.02.001.

[31] Alessandro Biacncolillo and Federico Marini. “Machine Learning and Pattern Recognition Methods in Chemistry from Multivariate and Data Driven Modeling”. In: *Elsevier* (2023), pp. 27–46. DOI: 10.1016/B978-0-323-90408-7.00001-0.

Chapter 2

Radiometric Calibration: white reference over distance

2.1 Introduction

Radiometric calibration establishes the mathematical relationship between the sensor's digital number (DN) values, at-sensor radiance, and surface reflectance. The general transformation, connecting the DN with the radiance $L(\lambda)$ reacing the pixel, follows the equation:

$$L(\lambda) = G(\lambda) \cdot (DN - DN_{dark}) \tag{2.1}$$

where $G(\lambda)$ is the sensor gain, and DN_{dark} represents the dark current. Further corrections such as vignetting, flat-field, and spectral response normalization are typically applied [1, 2].
The reflectance $\rho(\lambda)$ is then computed from the radiance by accounting for the downwelling irradiance $E(\lambda)$, coming from the sun, as:

$$\rho(\lambda) = \frac{\pi L(\lambda)}{E(\lambda)} \tag{2.2}$$

This framework assumes that both the radiance and irradiance are correctly calibrated to absolute physical units. However, achieving this condition in outdoor environments requires compensating for sensor non-linearities, environmental illumination variability, and atmospheric effects [3].

Several strategies have been proposed for converting raw DN to reflectance, which can be broadly divided into four major categories: (i) *empirical in-field methods* (ELM and SELM), (ii) *absolute laboratory calibration*, and (iii) *direct or model-based approaches*.

2.1.1 Empirical in-field methods

The most widely used approach for drone-based multispectral data is the *Empirical Line Method (ELM)* [4]. The ELM establishes a linear relationship between image DN values and known reflectances of calibration targets on the ground, typically using two to four panels of different reflectivities. Although operationally simple, ELM

assumes uniform illumination across the scene and linear sensor response, conditions that are often violated under variable cloud cover or non-Lambertian surfaces [5, 6]. Simplified or multi-target ELM variants have been developed to improve robustness and reduce the need for multiple field references [7, 8].

A simplified variant of the ELM is the so-called Two-Point Normalization, also referred to as Simple Reflectance Calibration or the Dark-Bright Target Method, and sometimes cited as the Simplified Empirical Line Method (SELM) or the ratio method. This method is one of the calibration techniques commonly adopted, especially in field applications. It computes the surface reflectance as

$$\rho = \frac{DN - DN_{\text{dark}}}{DN_{\text{white}} - DN_{\text{dark}}}$$

where DN_{dark} and DN_{white} represent the digital numbers of the dark and bright reference targets, respectively.

2.1.2 Absolute calibration

Absolute calibration methods are typically performed in laboratory conditions using traceable light sources, integrating spheres, and calibrated detectors [1]. These approaches ensure a physically consistent relationship between the radiance and DN values, with the advantage of long-term stability and traceability to standards such as NIST or GLAMR. However, their practical implementation in field conditions is limited due to instrument portability, environmental sensitivity, and differences between laboratory and operational environments.

2.1.3 Direct Reflectance Transformation and model-based methods

Recent developments have introduced direct reflectance transformation (DRT) workflows that use irradiance sensors co-registered with the imaging device [9]. These methods convert DN values directly to reflectance factors in near real time, without requiring ground reference panels, by continuously measuring downwelling irradiance. While these systems enhance operational efficiency and are robust under dynamic illumination, they demand precise calibration of both the irradiance and imaging sensors, and careful correction for sensor tilt and cosine response [10].

2.1.4 Spectral refinement and illumination calibration

In addition to radiometric scaling, wavelength calibration and illumination estimation represent key challenges. Spectral misregistration and stray-light can introduce non-negligible radiometric errors. Spectrum-matching algorithms [3] and data-driven illumination estimation methods [11, 12] have been developed to refine wavelength alignment and compensate illumination inconsistencies. These corrections are increasingly combined with physics-based or deep-learning models to further enhance reflectance retrieval accuracy.

Method	Main Advantages	Limitations
Absolute calibration (sphere-based)	Physically traceable; long-term stable; high accuracy	Requires laboratory setup; not feasible for field calibration; limited adaptability to varying conditions
Empirical Line Method (ELM)	Simple implementation; no specialized hardware	Sensitive to illumination variability; assumes linearity; depends on calibration panels
Simplified ELM	Reduced setup cost; adaptable to small fields	Lower accuracy; still dependent on scene uniformity
Direct Reflectance Transformation (DRT)	Real-time operation; no need for ground references	Requires irradiance sensor; sensitive to tilt and cosine errors; complex calibration
Spectral refinement	Corrects wavelength shifts and spectral distortions	Needs high SNR data; computationally intensive

Table 2.1: Summary of radiometric calibration methods, their main advantages and limitations.

Despite the progress in calibration strategies, several challenges remain. Variable illumination conditions, sensor temperature drifts, and stray-light contamination can significantly affect reflectance accuracy [10]. Field-based systems often face non-uniform illumination and bidirectional reflectance effects due to sun-target-sensor geometry [9]. Emerging studies propose hybrid approaches combining irradiance sensors, real-time atmospheric models, and data-driven optimization. New trends also include cross-calibration workflows using both empirical and physical models, and integration of deep-learning illumination estimators for autonomous reflectance correction.

Based on this literature, it is evident that achieving a robust radiometric calibration under variable outdoor conditions remains an open problem. Traditional ELM approaches are limited by illumination variability, whereas direct irradiance-based methods require careful hardware alignment and frequent recalibration. The approach explored in this thesis aims to address these issues by introducing a fixed in-frame reference panel, enabling continuous reflectance normalization within each image acquisition. This strategy aims to improve calibration repeatability and reduce the dependence on external reference measurements, contributing to more reliable proximal multispectral imaging in vineyard environments.

2.2 Dark-Bright Target Method: distance effect

In the present study, the Dark-Bright Target Method was employed, with the white reference positioned at a different distance from the target undergoing normalization.

In theory, the reflectance of a Lambertian surface (white in this case) should be invariant with respect to the camera-target distance, as the reduction in incident irradiance with distance ($1/r^2$ law) is compensated by a proportional increase in the number of scene points projected onto each pixel. In practice, however, this balance is often disrupted by non-ideal optical behavior, such as vignetting, lens aberrations, and non-Lambertian reflectance of natural surfaces. Experiments conducted with multispectral snapshot cameras under varying target distances have shown measurable intensity attenuation (up to 15%) across spectral bands due to optical transmission losses and atmospheric scattering [13]. When an image of a white reference is acquired with a multispectral camera, the value reported by the sensor ideally should not change based on the target camera distance. This is because there are two opposing effects that should compensate for each other on the pixel:

- decrease in Brightness: the brightness of each real point decreases with the square of the distance, it reflects less onto the lens and, consequently, onto the pixel.
- increase in Real Points per Pixel, the number of real points projected onto the same pixel increases as the distance increases.

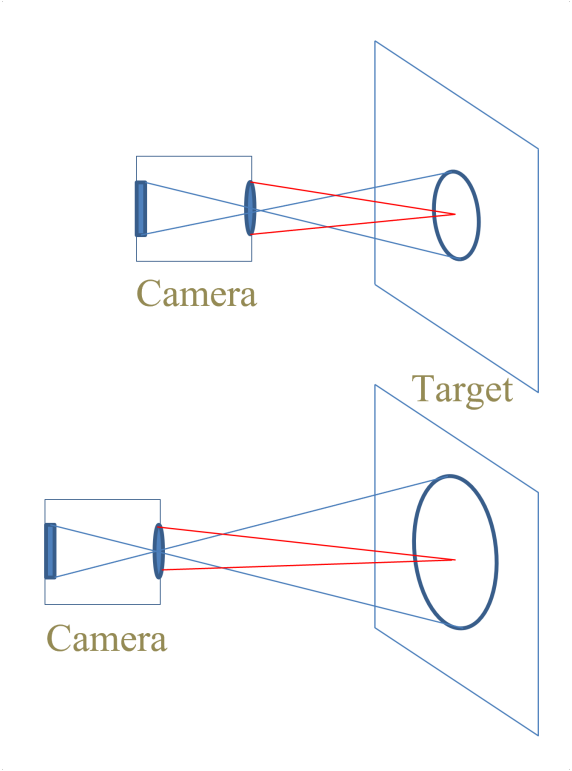


Figure 2.1: Diagram on the constancy of the detected intensity as the camera-target distance varies.

These effects should hold for a homogeneous material with constant reflectance across all points. In reality, this does not happen because of disturbances occurring on the lens and band-pass filters.

2.3 Pixel camera model

Before discussing the experimental section of the work, it is important to clarify what the DN extracted from each pixel represents. Consider one of the 25 types of pixels present on the sensor used in this work (Pothonfocus MV1-D2048x1088-HS02-96-G2) with the Quantum Efficiency (QE) at each band reported in Figure 2.2. The DN, which varies according to the pixel's position on the sensor, is influenced by several factors: the illumination source, the reflective properties of the material, the optical system (lens and filters), and the specific spectral filter associated with that pixel. In addition, the recorded value depends linearly on the electronic gain and the exposure time. The following section introduces all these variables in detail.

Following Figure 2.3, the solar illumination spectrum incident on the white reference

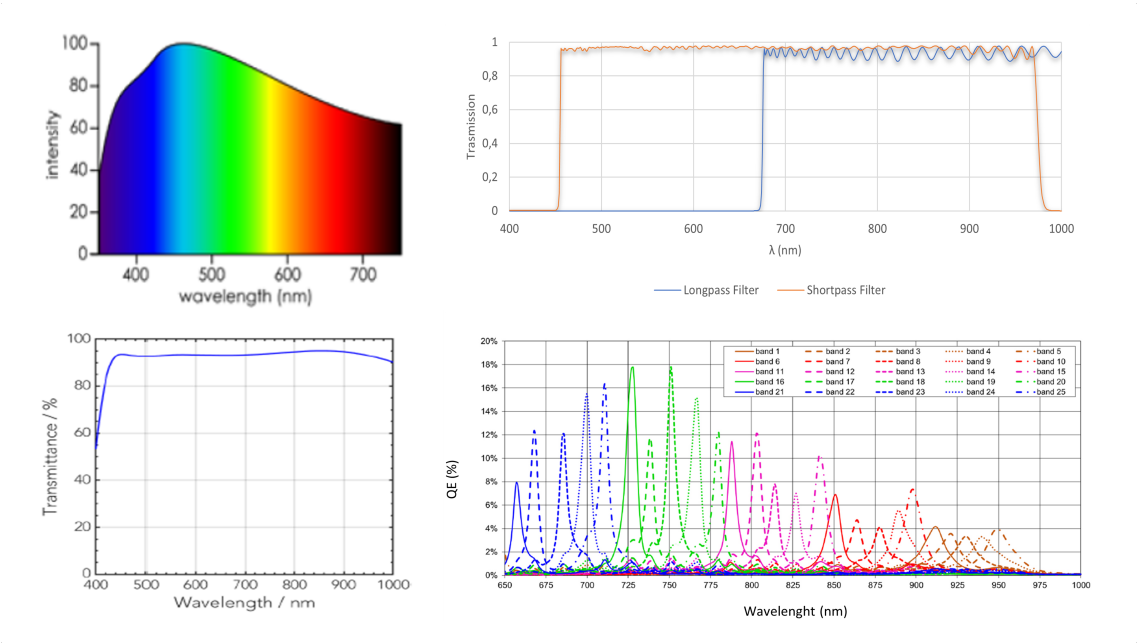


Figure 2.2: Spectral signature of sun, trasmittance of optical filter and the lens and quantum efficiency of Fabry Perot filter mounted on the CMOS sensor.

panel at position (X', Y')

$$S(x', y', \lambda, T)$$

is reflected toward the camera according to the local reflectance factor of the panel at that point.

$$r(x', y', \lambda)$$

The reflected radiant flux is then modified by the optical system, which redirects it toward the image sensor at pixel coordinates (X, Y), depending on the lens transmission efficiency

$$O(x, y, \lambda)$$

Before reaching the CMOS pixel, the signal is spectrally modulated by the response of the Fabry-Pérot interferometric filter mounted above it (x', y'), for example with central wavelength "i"

$$b_i(x, y, \lambda)$$

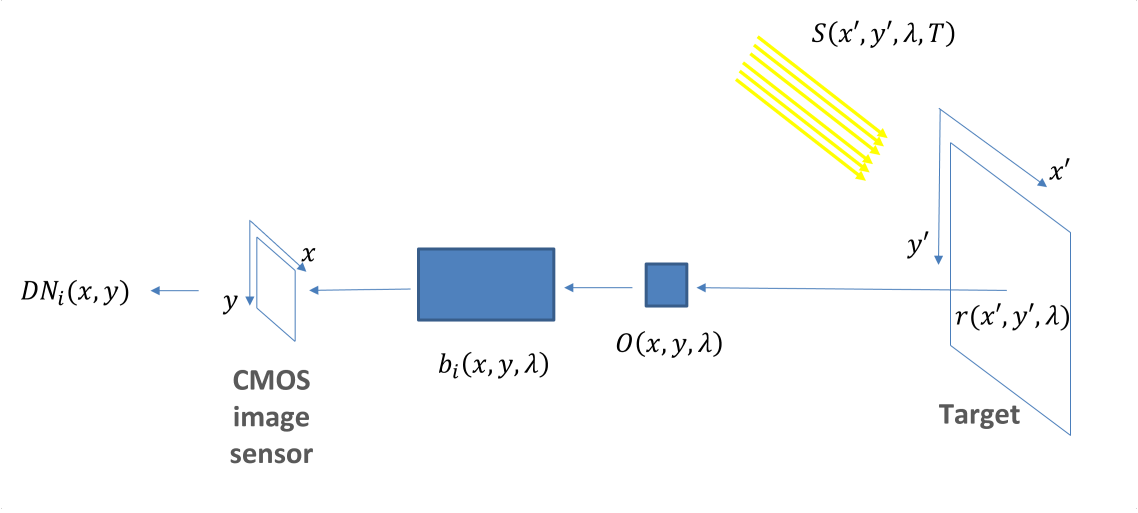


Figure 2.3: Digital Number: from sun to sensor.

Together with exposure time (t) and conversion Gain (k) the digital number we get from a pixel (x,y), at i-th band is:

$$DN_i(x,y) = kt \int_{\lambda_{\min}}^{\lambda_{\max}} b_i(x,y,\lambda)\, O(x,y,\lambda)\, S(x,y,\lambda,T)\, r(x,y,\lambda)\, d\lambda + e(x,y,\lambda)$$

The following assumptions were considered: all filters of the same type consistently behave within the sensor, and the incident light spectrum on the target (white reference) is assumed to be spatially uniform across the object. The contribution from the optical system and filters can be decomposed into spectral and spatial components. This assumption implies:
1) Band uniformity, meaning that the spectral characteristics are identical for all pixels equipped with the same color filter.

$$b_i(x,y,\lambda) = b_i(\lambda)$$

2)The source spectrum is assumed to vary only as a function of time (T), remaining spatially uniform over the target surface.

$$S(x', y', \lambda, T) = S(\lambda, T)$$

3)Optical consistency assumes that the optical system and filters are identical for all pixels, differing only by a multiplicative scalar factor.

$$O(x,y,\lambda) = \hat{O}(x,y)\, \tilde{O}(\lambda)$$

4)Reference Homogeneity with white standard reference homogeneous

$$r(x,y,T) = r(\lambda)$$

Under these assumptions, the digital number recorded by each pixel can be expressed as follows:

$$DN_i(x,y) = kt\, \hat{O}(x,y) \int_{\lambda_{\min}}^{\lambda_{\max}} b_i(\lambda)\, \tilde{O}(\lambda)\, S(\lambda,T)\, r(\lambda)\, d\lambda + e(x,y,\lambda)$$

2.4 Materials and Methods

2.4.1 Multispectral imaging system

The experiments were carried out using a custom-built multispectral imaging system designed for proximal sensing applications in vineyard environments. The imaging unit consists of a compact camera (Pothonfocus MV1-D2048x1088-HS02-96-G2) equipped with a 25-band interference filter covering the near-infrared (NIR) spectral range (approximately 650-950 nm). Each spectral band has an average full width

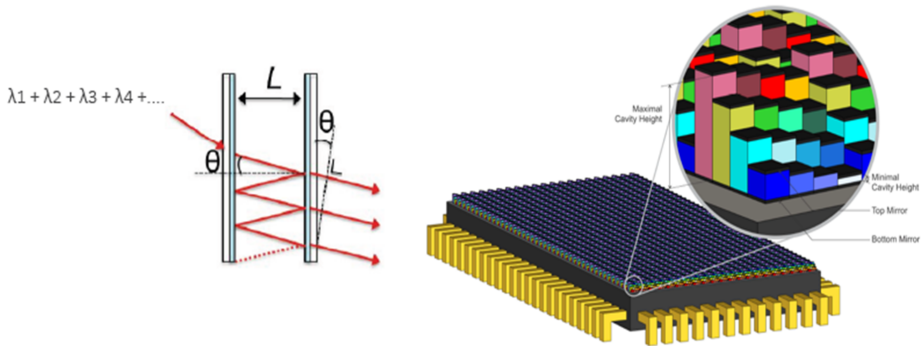


Figure 2.4: Fabry Perrot filter scheme on CMOS sensor.

at half maximum (FWHM) of 10 nm. The sensor operates with a 8-bit dynamic range and a nominal radiometric linearity across all bands. Image acquisition is controlled via custom Matlab application, to ensure flexibility and consistent exposure across acquisitions. The optical system was mounted on a stable tripod at a fixed height above the target surface, ensuring a nadir acquisition geometry. The field of view (FOV) was adjusted to fully include the calibration reference area and the region of interest, typically a homogeneous vineyard canopy or uniform surface. All acquisitions were conducted under solar illumination, in sunny day conditions to minimize short-term radiometric fluctuations.

2.4.2 Radiometric reference and illumination setup

A white reference panel (poster board, assumed to be homogeneous and opaque) was placed within the camera field of view (FOV) at a fixed distance of 10 cm during all image acquisitions. This panel served as a constant radiometric reference, allowing continuous normalization of reflectance values without the need for repeated reference acquisitions.

Dark reference measurements were also performed prior to each acquisition session by covering the camera lens with an opaque cap, in order to account for the electronic dark current offset.

To characterize the radiometric attenuation as a function of the camera-target distance, a second white reference panel of the same material was positioned at increasing distances from the camera (20, 30, 50, 70, and 90 cm), as shown in Figure



Figure 2.5: Test support for evaluating intensity attenuation over distance.

2.5. Acquisitions were repeated under identical optical conditions (focal length = 17 mm; aperture = f/2.8) while varying exposure parameters according to the experimental plan described below.

Experimental plan:

- Distance variation: 10, 20, 30, 50, 70, and 90 cm between camera and target.
- Exposure time variation: from 299 µs to 3600 µs, in 200 µs increments.
- Time of day: 9:00, 12:00, 17:00, and 18:00, to assess the effect of solar inclination.

Images were acquired using a custom MATLAB application (camera.app), which automatically adjusted the exposure time while maintaining a constant camera-target distance. This setup enabled the acquisition of 104 images across all combinations of distance and exposure settings within approximately five minutes.
All experiments were conducted under clear-sky conditions, ensuring negligible variation in both the incident solar spectrum and solar inclination during each acquisition sequence. This configuration allowed for a robust evaluation of radiometric attenuation as a function of sensor-target geometry and irradiance distribution.

2.5 Data treatment

An image acquired by the 25-band snapshot multispectral camera is a two-dimensional frame composed of a mosaic of macropixels as shown in Figure 2.6. Each macropixel consists of 25 individual pixels, each equipped with a Fabry-Pérot interferometric

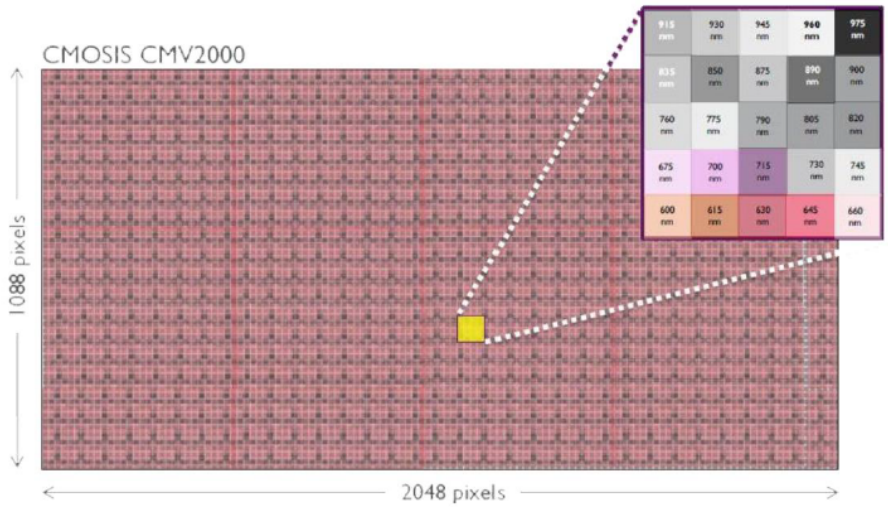


Figure 2.6: Snapshot mosaic Photonfocus.

filter that selects one of the twenty-five spectral bands within the approximate wavelength range of 650 to 950 nm.
Each multispectral image was pre-processed through a standardized pipeline consisting of:

1. Dark current subtraction using pre-recorded dark frames.
2. Ipercube extrapolation from 2D snapshot image via custom Matlab function called estrazione ipercubo.m[1].
3. Flat-field correction to compensate for pixel-to-pixel sensitivity variations.
4. Reference ROI extraction and computation of Median value (vector of 25 components).
5. Target ROI extraction and computation of Median value (vector of 25 components).

Starting from the experimental data the correction factor $k(d)$ for each spectral band was calculated.

Using the linear camera model and the previously stated assumptions, and by normalizing the data of each spectral band relative to the value measured at 10 cm, we isolate the optical contribution by removing the effects of source intensity, camera gain, and exposure time, which do not vary with distance. This correction factor describes the optical attenuation as a function of distance.

$$\frac{DN_i^{\mathrm{d\ cm}}}{DN_i^{10\ \mathrm{cm}}} = \frac{kt\hat{O}(x,y)^{\mathrm{d\ cm}}}{kt\hat{O}(x,y)^{10\ \mathrm{cm}}}$$

[1]we apply bilinear interpolation on the RAW image to maintain the original image size, forming a data cube, or hypercube, composed of 25 photos.

2.6 Data Analysis

To quantify the effect of distance on the recorded signal, the concept of attenuation percentage was introduced. The attenuation factor expresses the reduction in DN as a function of the distance (d) between the target and the reference panel, and is defined as:

$$\text{Attenuation}\%(d) = \frac{DN_{10} - DN_d}{DN_{10}} \cdot 100$$

This formulation allows the attenuation to be expressed as a percentage relative to the reference condition. Based on the experimental data, including acquisition time (and thus solar angle), wavelength, exposure time, and DN values measured on the white reference (10 cm) (mean, median, and standard deviation), several predictive models were tested for estimating the DN of the target. The analysis began with a simple linear regression model, and was subsequently extended to a Multilayer Perceptron Artificial Neural Network (MLP-ANN) trained using the Levenberg-Marquardt backpropagation algorithm. Due to the observed collinearity among the statistical descriptors of DN (mean, median, and standard deviation), linear models such as multiple regression were developed using only the median values as input features. The evaluation and comparison of the models were primarily based on the Root Mean Square Error (RMSE) calculated on the test set. All models were trained using the same dataset and validated on an identical test set, consisting of 25% of the data randomly selected from the entire sample.

2.7 Results and Discussion

As shown in the Figure 2.7, attenuation increases with distance for all the spectral bands acquired by the camera. This confirms that the phenomenon is related to the optical setup rather than to the presence of the Fabry-Perot filters (Figure 2.8). In all the plots, zero-attenuation values can be observed at various distances; these depend on the exposure time. Since the exposure times were automatically adjusted, some acquisitions are saturated regardless of the camera-target distance. To remove this contribution, we therefore focus on the maximum attenuation values recorded at a single wavelength, across the different distances, grouped by hour.

Figure 2.9 shows the trend of the maximum attenuation values for a fixed wavelength. These results highlight the dependence of attenuation on the time of day: as the angle between the light source and the normal to the target increases, the attenuation also increases.
The predictive performance of the different regression models developed for target DN estimation was evaluated using the coefficient of determination (R^2) and the Root Mean Square Error (RMSE), both for the calibration and test sets. Tabel 2.2 summarizes the results obtained for the simple linear regression, multiple linear regression (MLR), partial least squares (PLS) regression, and multilayer perceptron artificial neural network (MLP-ANN) models. All models exhibited high accuracy, with R^2 values exceeding 0.99, indicating a strong correlation between predicted and measured values. The simple linear regression already provided a good fit,

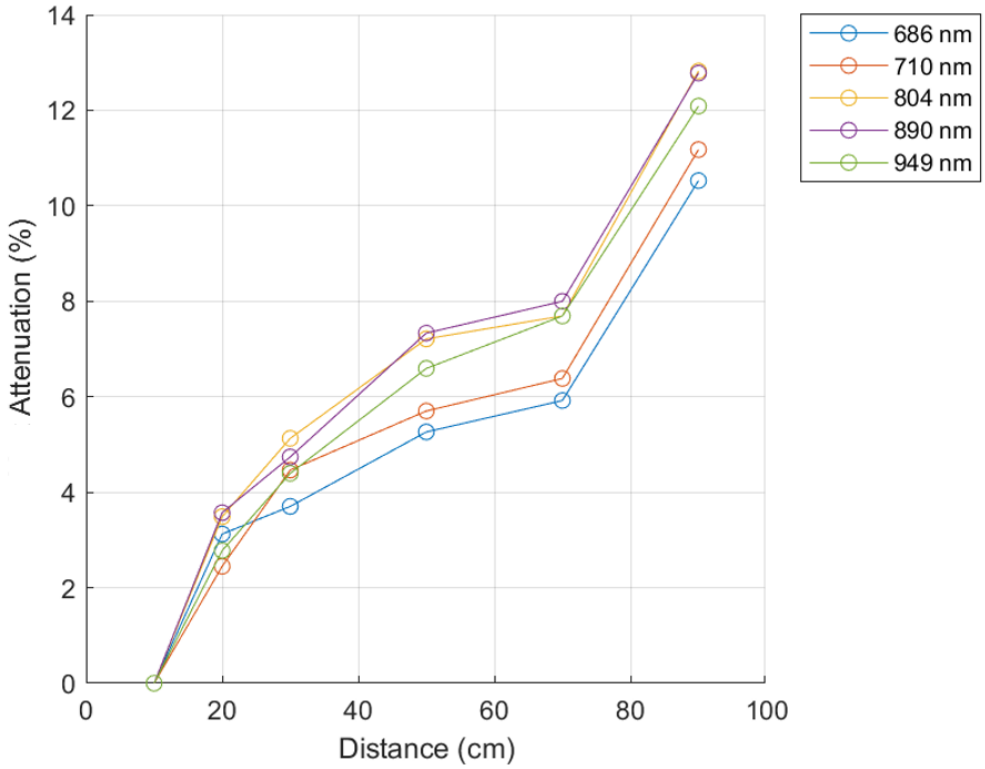


Figure 2.7: Attenuation at 9:00, fixed exposure time, for different wavelenghts.

although with a higher RMSE (≈ 12). Both MLR and PLS regression significantly reduced the prediction error (RMSE $\approx 3-4$), showing similar performance in both calibration and validation. The MLP-ANN achieved the best overall performance, with the highest $R^2(0.999)$ and the lowest RMSE (≈ 2.6) in both calibration and test phases. The network architecture consisted of one hidden layer with three neurons, using a sigmoid activation function. The output layer employed a linear activation to ensure direct mapping to DN values. Training was carried out using the Levenberg-Marquardt backpropagation algorithm. To prevent potential overfitting, the training process was monitored through an early stopping technique, in which the learning was automatically halted when the validation error failed to decrease for a predefined number of iterations. This approach ensured that the model maintained good generalization capability across the test set. To further assess the predictive

	Calibration		Test set	
	R^2	RMSE	R^2	RMSE
Simple linear regression	0.993	12.28	0.994	11.37
MLR	0.998	3.17	0.998	3.25
PLS regression	0.996	3.80	0.996	3.81
MLP-ANN	0.999	2.64	0.999	2.65

Table 2.2: Regression models results.

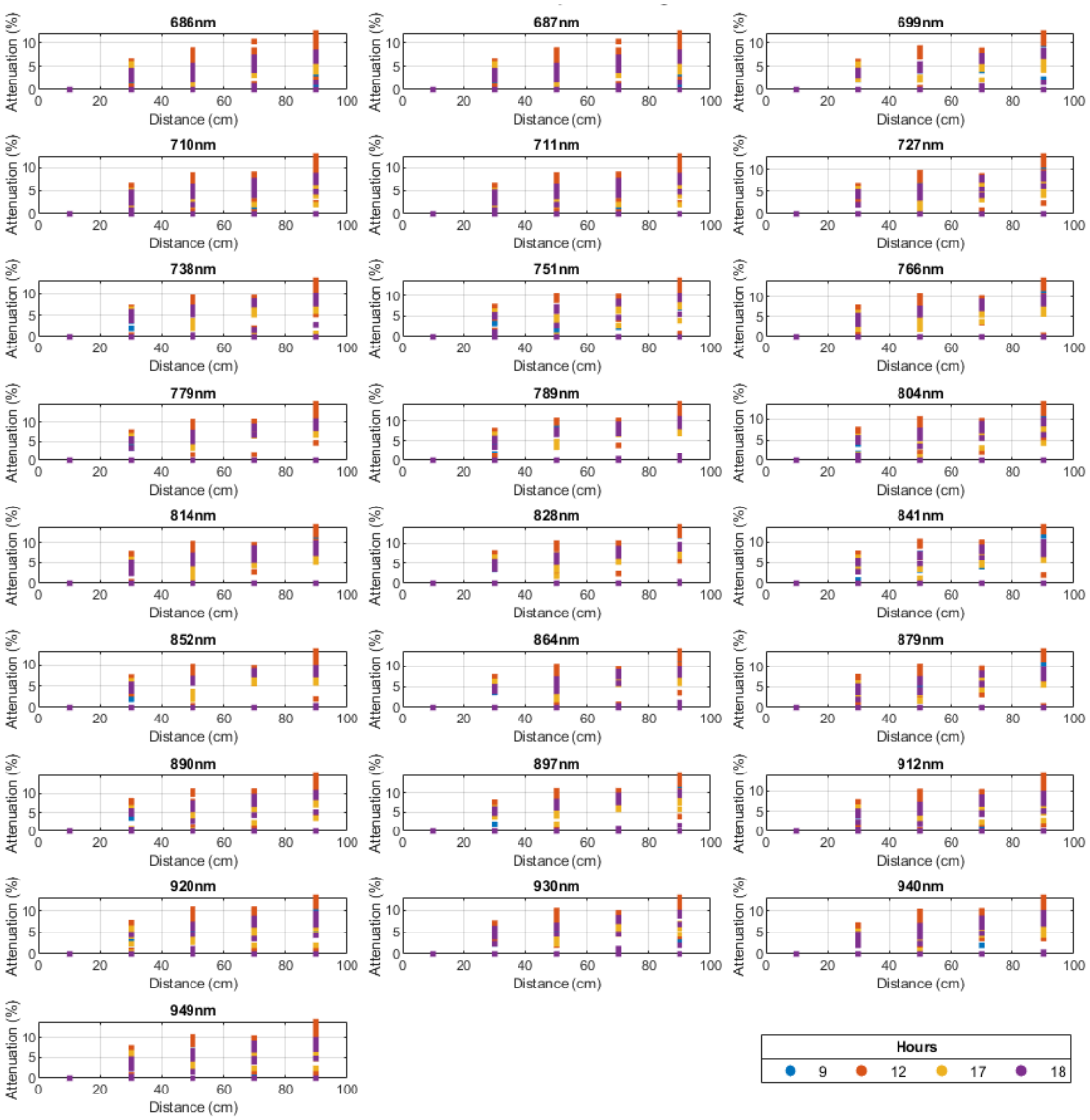


Figure 2.8: Test results: attenuation over distance for all bands. Acquisitions were performed at different times of the day and with various exposure times.

performance and robustness of the tested models, the relationship between the observed and predicted reflectance values was analysed. Figures 2.9 show the scatter plots of measured versus predicted values for the Multiple Linear Regression (MLR), Partial Least Squares (PLS), and MLP-ANN models, respectively. Each plot includes the 1:1 reference line, representing the ideal prediction, and the fitted regression line. All three models exhibit a strong linear correlation between predicted and measured data, confirming the reliability of the calibration process. The MLR and PLS models show a slight dispersion around the 1:1 line, particularly for lower reflectance values, where prediction uncertainty tends to increase due to lower signal intensity. The MLP-ANN, by contrast, demonstrates a nearly perfect alignment along the identity line, with minimal residuals across the full DN range, confirming its superior predictive capability and stability under varying conditions.

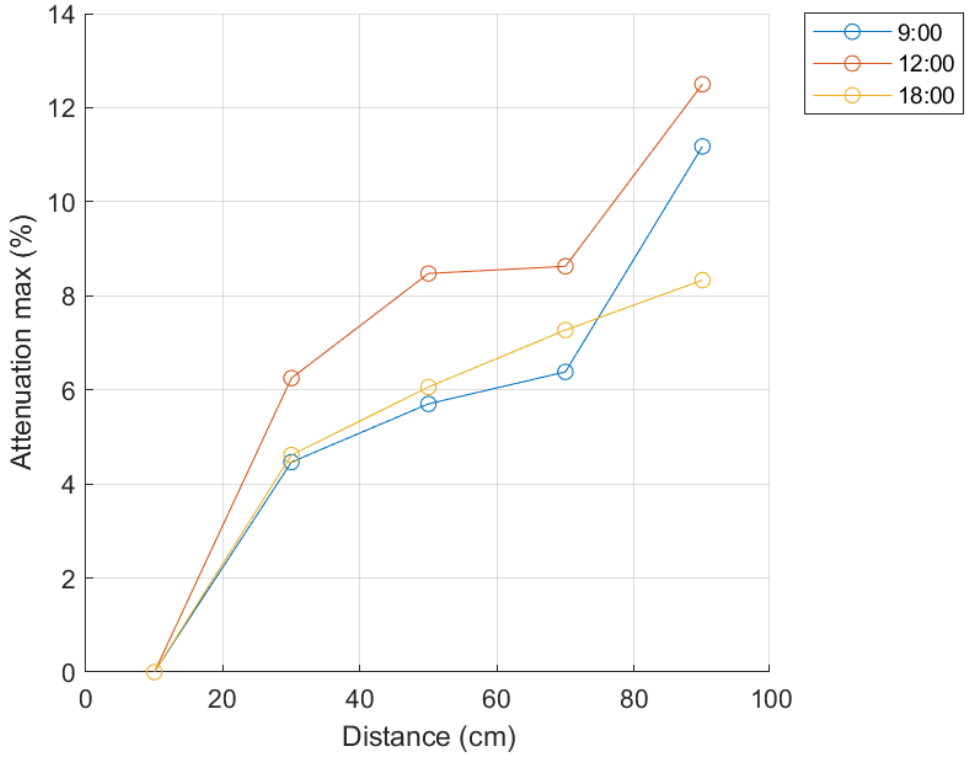


Figure 2.9: Attenuation max at 711 nm. Solar elevation angles at 09:00, 12:00, and 18:00 are approximately 46°, 71°, and 17°, respectively (source: NOAA Solar Position Calculator).

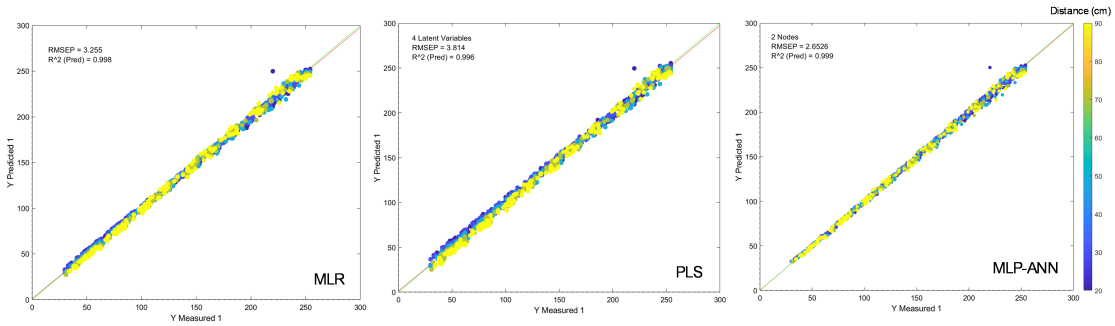


Figure 2.10: Observed vs. predicted DN values for the three tested models (MLR, PLS, and MLP-ANN).

2.8 Conclusion

This preliminary study on signal attenuation as a function of distance provides a foundation for the forthcoming acquisitions in the vineyard. The results obtained confirm that the camera-canopy distance selected for field measurements (approximately 80 cm) falls within the range of distances analyzed in these tests, ensuring the applicability of the derived attenuation model (Figure 2.11). In the subsequent field setup, a permanently mounted white reference strip will be integrated near the

camera to provide continuous reflectance calibration. This reference will enable the estimation of attenuation and normalization of the images at different wavelengths, while also assisting in the dynamic adjustment of exposure time to maintain the camera within its linear response range. Although the presence of the reference strip slightly reduces the field of view, this limitation is considered acceptable in view of the expected improvement in radiometric accuracy and data consistency.

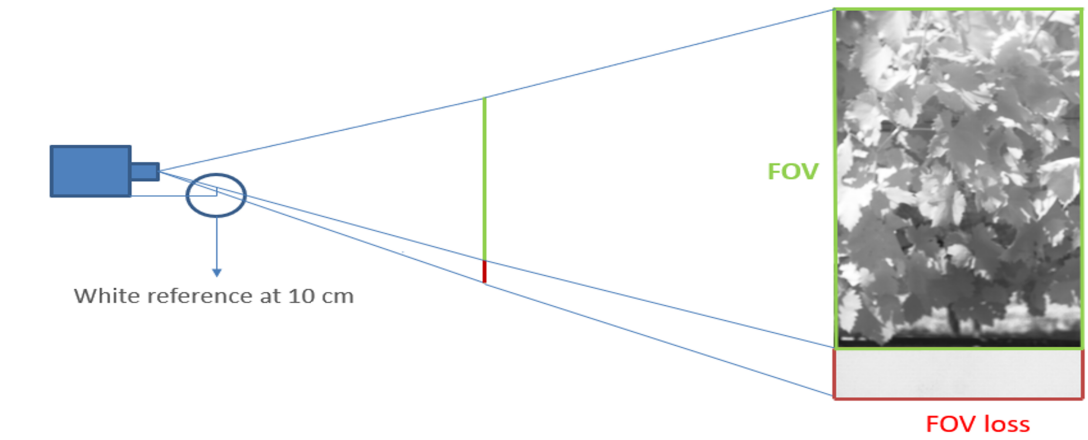


Figure 2.11: White reference scheme and resulting field of view.

Bibliography

[1] Zhipeng Wang et al. "Absolute Radiometric Calibration of an Imaging Spectroradiometer Using a Laboratory Detector-Based Approach". In: *Remote Sensing* 14.9 (2022), p. 2245. DOI: 10.3390/rs14092245.

[2] Vlado Kitanovski, Jean-Baptiste Thomas, and Jon Yngve Hardeberg. "Reflectance Estimation from Snapshot Multispectral Images Captured under Unknown Illumination". In: *Proceedings of the 29th Color and Imaging Conference (CIC29)*. Society for Imaging Science and Technology, 2021, pp. 264–269. DOI: 10.2352/issn.2169-2629.2021.29.264.

[3] Bo-Cai Gao, Marcos J. Montes, and Curtiss O. Davis. "Refinement of Wavelength Calibrations of Hyperspectral Imaging Data Using a Spectrum-Matching Technique". In: *Remote Sensing of Environment* 90 (2004), pp. 424–433. DOI: 10.1016/j.rse.2003.09.002.

[4] Geoffrey M. Smith and Edward J. Milton. "The Use of the Empirical Line Method to Calibrate Remotely Sensed Data to Reflectance". In: *International Journal of Remote Sensing* 20.13 (1999), pp. 2653–2662. DOI: 10.1080/014311699211994.

[5] W. M. Baugh and D. P. Groeneveld. "Empirical Proof of the Empirical Line". In: *International Journal of Remote Sensing* 29.3 (2008), pp. 665–672. DOI: 10.1080/01431160701352162.

[6] Faheem Iqbal, Arko Lucieer, and Karen Barry. "Simplified Radiometric Calibration for UAS-Mounted Multispectral Sensor". In: *European Journal of Remote Sensing* 51.1 (2018), pp. 301–313. DOI: 10.1080/22797254.2018.1432293.

[7] Chuyuan Wang et al. "A Simplified Empirical Line Calibration Method for sUAS-based Remote Sensing". In: *Proceedings of IEEE/ISPRS Workshop on Remote Sensing*. 2015. URL: https://www.researchgate.net/publication/280696498.

[8] J. Burdette Barker et al. "Calibration of a Common Shortwave Multispectral Camera System for Quantitative Agricultural Applications". In: *Precision Agriculture* (2019). DOI: 10.1007/s11119-019-09701-6.

[9] Juha Suomalainen et al. "Direct Reflectance Transformation Methodology for Drone-Based Hyperspectral Imaging". In: *Remote Sensing of Environment* 266 (2021), p. 112691. DOI: 10.1016/j.rse.2021.112691.

[10] Yuxiang Wang et al. "UAV Multispectral Remote Sensing for Agriculture: A Comparative Study of Radiometric Correction Methods under Varying Illumination Conditions". In: *Biosystems Engineering* 248 (2024), pp. 240–254. DOI: 10.1016/j.biosystemseng.2024.11.005.

[11] Haris Ahmad Khan et al. "Illuminant Estimation in Multispectral Imaging". In: *Journal of the Optical Society of America A* 34.7 (2017), pp. 1085–1098. DOI: 10.1364/JOSAA.34.001085.

[12] Yuqi Li, Qiang Fu, and Wolfgang Heidrich. "Multispectral Illumination Estimation Using Deep Unrolling Network". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Introduces a deep unrolling ADMM-based method for spectral illumination estimation. 2021, pp. 2672–2681.

[13] Riccardo Bertoglio et al. "On-the-go Table Grape Ripeness Estimation via Proximal Snapshot Hyperspectral Imaging". In: *Computers and Electronics in Agriculture* 226 (2024), p. 109354.

Chapter 3

Field-Based Evaluation of Vineyard Berry Quality Using Multispectral Sensing

3.1 Introduction to field multispectral acquisitions for grape quality assessment

The assessment of grape ripeness and quality is a crucial step in precision viticulture, directly influencing harvest decisions and the final characteristics of wine. Conventional methods for determining grape composition—such as destructive sampling and laboratory analysis of total soluble solids (TSS), titratable acidity (TA), pH, and anthocyanin concentration—are time-consuming and spatially limited. The need for rapid, objective, and non-destructive evaluation has therefore driven the development of optical sensing technologies, particularly those based on spectroscopy and imaging spectroscopy.

In the last two decades, near-infrared (NIR), hyperspectral (HSI), and multispectral imaging (MSI) have emerged as powerful tools for monitoring grape composition and ripening status. Early studies established the capability of NIR spectroscopy to predict TSS, pH, and anthocyanin content in grape homogenates and intact berries [1, 2]. These works demonstrated that multivariate modeling, particularly partial least squares regression (PLSR), could extract relevant chemical information from spectral data. Later, Cozzolino et al. [3] emphasized the methodological foundation of multivariate calibration—sample selection, spectral preprocessing, validation, and error propagation—which remains essential for developing robust models in spectroscopy-based quality assessment.

Recent research has extended these methods to proximal and in-field imaging. Benelli et al. [4] first demonstrated the feasibility of in-field, non-destructive hyperspectral imaging (400-1000 nm) for estimating grape ripeness through TSS prediction using PLSR models. Similarly, Gutiérrez et al. [5] employed an on-the-go hyperspectral system mounted on an all-terrain vehicle to map TSS and anthocyanins across a vineyard, achieving promising determination coefficients ($R^2 > 0.8$). Their subsequent work combined hyperspectral data with convolutional neural networks

(CNNs), improving robustness under variable field conditions [6].

Main Outcome	Sensor / Range (nm)	Target Variables	Model
Foundational NIR study on grapes and wine [1]	NIR spectrometer (400-2500)	TSS, pH, anthocyanins	PLSR, LOCAL
Framework for multivariate spectroscopy [3]	NIR (1100-2500)	Calibration review	PLSR, PCR, ANN
In-field, non-destructive maturity estimation [4]	HSI (400-1000)	TSS	PLSR
On-the-go mapping, $R^2 > 0.8$ [5]	HSI push-broom (400-1000)	TSS, anthocyanins	SVM, PLSR
First CNN applied to vineyard HSI [6]	HSI push-broom (400-1000)	TSS, pH, TA, anth.	CNN
Fast, on-the-go snapshot system [7]	Snapshot HSI (500-900)	TSS, anthocyanins	Mask R-CNN + PLSR
High accuracy ($R^2 > 0.9$) in-field estimation [8]	MSI (475-842)	TSS, pH, phenolics	ANN
Comparative modeling of linear/nonlinear methods [9]	HSI (400-1000)	TSS, pH, anthocyanins	Ridge, SVR, RF
Handheld device validation for table grapes [10]	Portable NIR (900-1700)	TSS, acidity, pH	PLSR, RF

Table 3.1: Summary of representative studies on proximal spectral and imaging techniques for grape quality assessment.

Multispectral systems, although limited to a smaller number of bands, have proven to be efficient for in-field applications. Pérez-Pérez et al. [8] used a proximal multispectral sensor (475-842 nm) coupled with artificial neural networks (ANNs) to estimate TSS, pH, and total phenolics, reporting R^2 values above 0.9. These results highlight the potential of multispectral cameras for rapid, cost-effective deployment in precision viticulture. Complementary work on portable and handheld NIR devices has further confirmed their accuracy in predicting ripeness and acidity of table grapes, underscoring the versatility of compact sensors for on-site monitoring [10].

A particularly relevant advancement is the integration of snapshot hyperspectral technologies, which acquire the entire datacube in a single exposure. Bertoglio et al. [7] applied a snapshot camera (Senop HSC-2, 500–900 nm) for on-the-go ripeness estimation in table grapes, combining Mask R-CNN segmentation and PLSR regression to predict TSS and anthocyanins. Compared to traditional push-broom systems, snapshot sensors overcome motion artefacts and enable fast acquisitions

suitable for robotic or handheld operation. These approaches represent the current frontier in proximal optical sensing for viticulture.

From a modeling perspective, linear methods such as PLSR remain the benchmark for spectral regression due to their interpretability and robustness. However, nonlinear algorithms—including support vector regression (SVR), random forest (RF), extreme gradient boosting (XGB), and neural networks—have demonstrated superior performance in complex or variable lighting conditions [9]. Deep learning architectures (CNNs, autoencoders) have also shown promise in handling high-dimensional data and in automating feature extraction [6]. The choice of algorithm depends on sensor characteristics, data dimensionality, and the target variable (TSS, anthocyanins, or acidity).

Overall, the reviewed studies converge on the importance of integrating advanced machine learning models with robust, field-ready imaging systems. The combination of proximal sensing, spectral modeling, and AI-driven analysis offers a feasible pathway toward continuous and non-destructive monitoring of grape quality. Table 3.1 summarizes the main characteristics of representative studies, highlighting the diversity of sensors, wavelengths, and modeling approaches applied to grape ripeness and compositional assessment.

In summary, the convergence of compact multispectral and hyperspectral technologies with advanced regression and deep learning models enables efficient, non-invasive assessment of grape maturity and compositional attributes directly in the vineyard. These developments represent a significant step toward data-driven precision viticulture, where high-throughput optical sensing can support decision-making on harvest timing, selective picking, and quality control throughout the production chain.

3.2 Materials and Methods

3.2.1 Vineyard

The field tests were conducted in a vineyard Sangiovese located at Cadriano, focusing on both sides of four rows. The distance between rows was approximately 2 m, while the spacing between individual vines within the same row was 1 m. This vineyard was originally used to test four different types of fertilization, each applied at two intensity levels, resulting in a total of eight distinct treatments. The four rows subjected to these fertilization types (rows 4, 6, 8, and 10) were further divided into plots of fifteen plants each, as shown in Figure 1.The vineyard is oriented from northwest to southeast, and because measurements were always carried out after 12 p.m., the west side of each row was directly exposed to the sun, while the east side remained in shadow. The same applies to the distribution of treatments across the vineyard rows. A similar situation occurs for the white reference target, which was affected by the shadow of the adjacent row at sunset. In this study, the effect of fertilization was not considered.

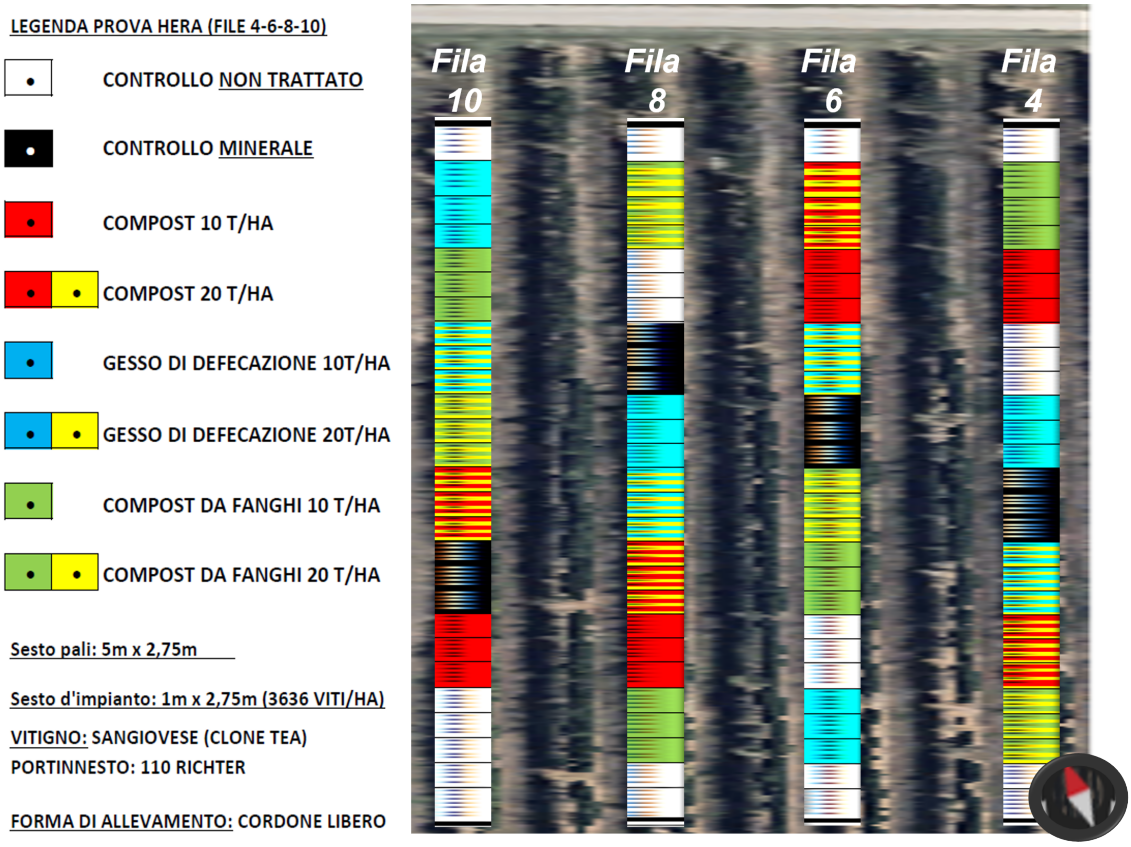


Figure 3.1: The map of the vineyard in Cadriano detailing the distribution of treatments across the vineyard rows.

3.2.2 Multispectral set up and spectral acquisition

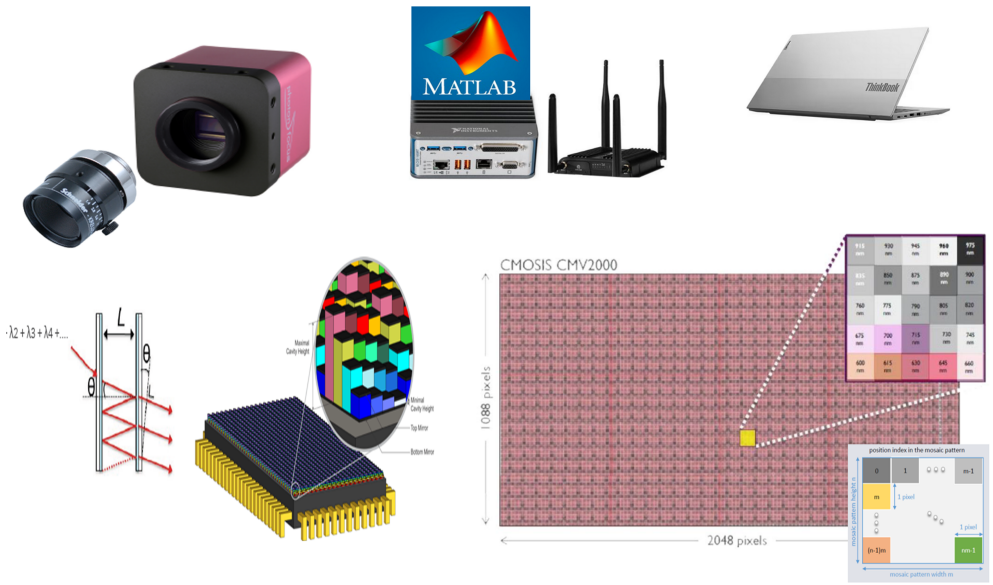


Figure 3.2: Multispectral Camera acquisition system, Fabry-Perot filter scheme and sensor grid.

The camera used was a Photonfocus multispectral model featuring 25 spectral bands ranging from 600 to 975 nm and a resolution of 2.2 megapixels. The multispectral camera, based on snapshot technology, employs a commercial CMOS sensor enhanced with pixel-by-pixel Fabry-Perot interferometers. This technology is particularly suitable for field applications due to its compact size, lightweight nature, and cost-effectiveness compared to hyperspectral cameras (Figure 2). Preliminary field tests carried out in July 2023 defined the optimal camera positioning relative to the tractor, which determined the ground clearance, target distance, field of view (FOV), and lens aperture:

- Focal length: 17 mm
- Target distance: 1 m ± 15 cm
- Lens aperture: f/2.8



Figure 3.3: Camera on tractors: image acquisition.

The camera mounted on the tractor was positioned at the fruiting zone height, approximately one meter from the canopy, at a fixed distance to maintain image sharpness with a constant aperture (Figure 3). Natural lighting conditions were used during the tests, with sunlight directly illuminating the row side to replicate real-world conditions and ensure accurate and relevant data collection for vineyard monitoring. Because lighting conditions were uncontrolled, a fixed white reference target was placed near the camera to enable image calibration and compensate for illumination variability (see Chapter 2). The tests were conducted on August 1 and 29, and September 4, 10, and 11, 2024. Photonfocus provided the eBUS Player software (Pleora Technologies) for camera configuration and image acquisition. For multispectral cube extraction and subsequent corrections, PfHyper was used. However,

both software tools lack automation for image acquisition and cube processing. To address this limitation, a specific MATLAB script named camera.app was developed to integrate and extend the existing functionalities, including real-time exposure time calculation based on a white reference within the field of view (Figure 4). While other parameters remained fixed, solar illumination, exposure time, and tractor speed had to be dynamically adjusted in real time to prevent overexposed or motion-blurred images. Solar illumination is uncontrollable, but the other two parameters were managed adaptively during acquisition. Lower solar radiation increased the optimal exposure time and consequently reduced the maximum allowable tractor speed. From the white reference attached to the camera, a 25×25 pixel area was extracted to compute statistical metrics on the raw image (without data cube extraction):

- standard deviation
- minimum and maximum intensity
- Sobel gradient analysis

The camera.app script is provided in Appendix A.



Figure 3.4: Control panel of the “camera.app”.

3.2.3 Reference chemical parameters

Grape berry ripening is commonly described in terms of sugar accumulation, measured as total soluble solids (TSS). However, other compositional variables are also considered to assess optimal harvest maturity, including berry acidity, often expressed as pH and titratable acidity (g/L). After image acquisition, TSS (expressed in °Brix) was measured on 20 randomly selected berries from each fertilized section using a portable digital refractometer (Maselli). Subsequently, the mean value for each section was calculated. pH and titratable acidity were measured using a Crison Compact titrator. 2.4 Data analysis

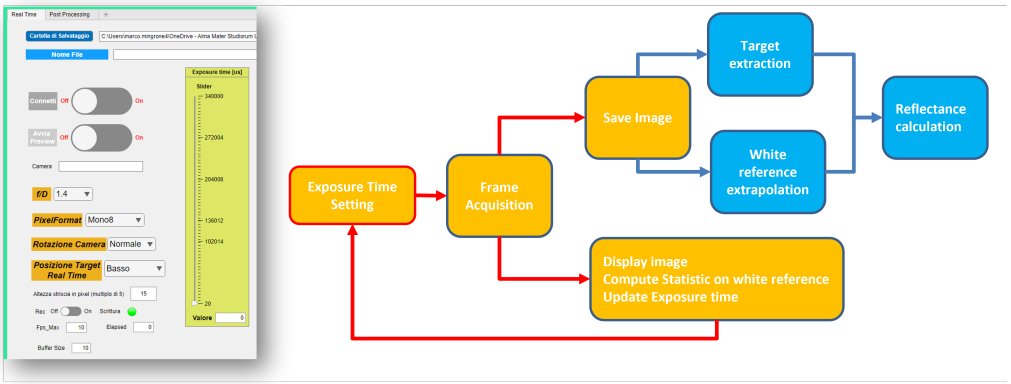


Figure 3.5: "camera.app" control panel and application workflow.

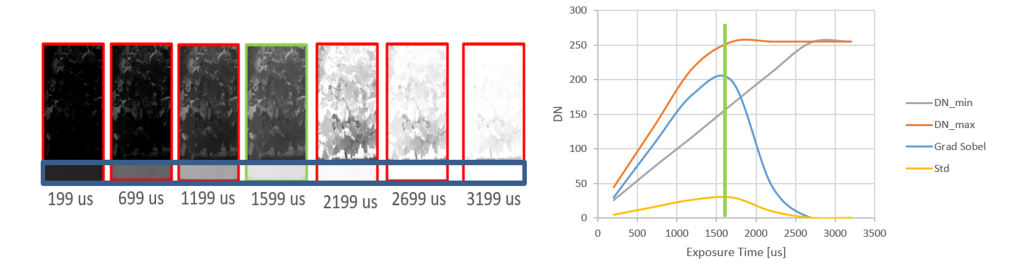


Figure 3.6: "camera.app" feedback for automatic exposure.

3.2.4 Hypercube extraction

A specific MATLAB script ("estrazione ipercubo.m") was developed to take an image or a selected region (box) as input and return the corresponding hypercube. The script applies bilinear interpolation to generate 25 single-band images at different wavelengths, all of the same size as the original RAW image. By default, the function extracts the hypercube of the entire input image, excluding a 5-pixel outer border, as these edge pixels cannot be correctly interpolated. Optional parameters can be specified to define an internal square region (defined by its top-left vertex and side length), allowing the function to return the hypercube corresponding to the selected area. The function outputs a structured variable called "ipercubo", containing the following fields:

- File name: name of the processed file
- Image: RAW image provided by the camera
- Wavelengths: array mapping each hypercube component to its corresponding wavelength
- Full cube: complete hypercube of the selected image
- White: hypercube of the white reference located at the bottom of the FOV
- White mean: mean spectrum of the white reference hypercube

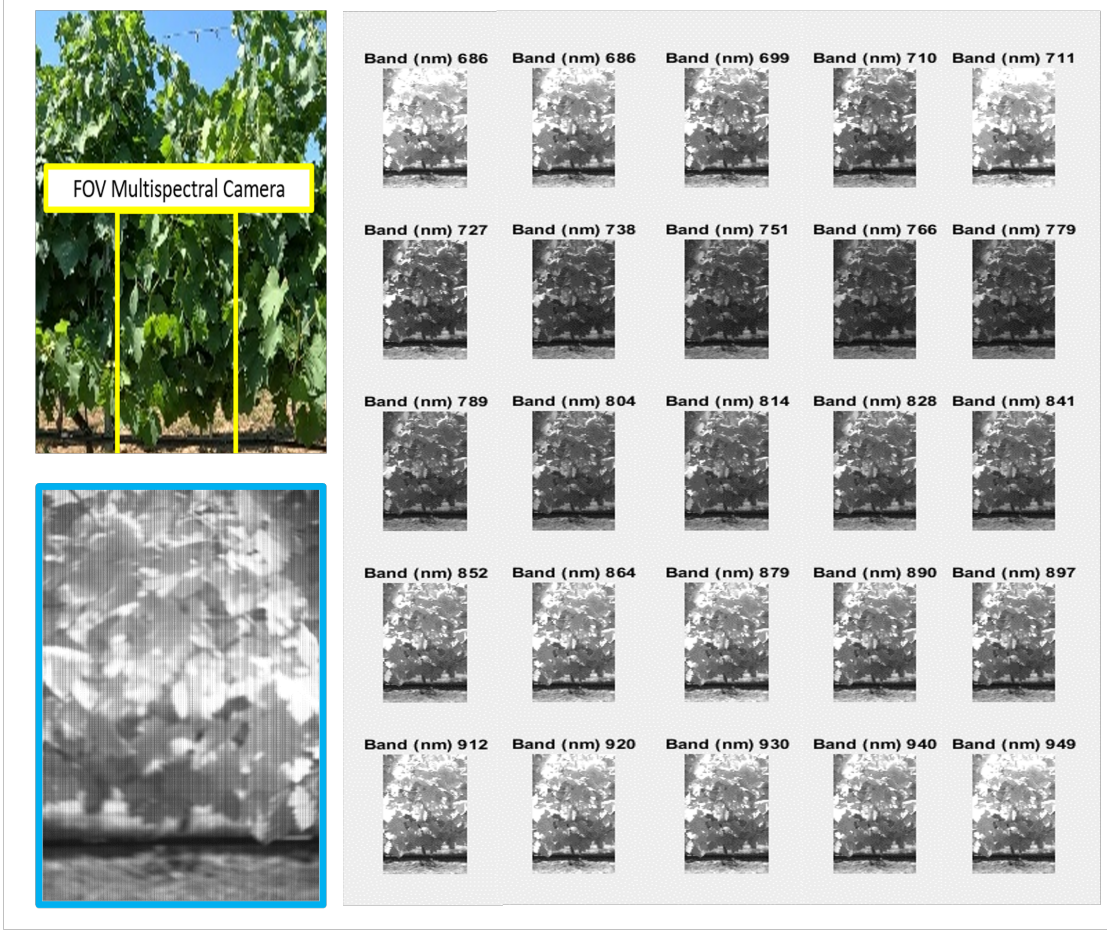


Figure 3.7: On the left, the FOV of the GoPro 11 and the Photonfocus Multispectral; on the right, the multispectral cube extracted with 25 bands at same size:1085x2045.

- White std . standard deviation of the spectrum
- Target cube - hypercube of the target image

Image orientation is crucial for correct hypercube extraction, as it affects the pixel-to-wavelength mapping (e.g., the top-left pixel no longer corresponds to 912 nm but to 949 nm when the superpixel is rotated). This rotation is required because the camera was positioned with a 90° left rotation to achieve a larger vertical than horizontal field of view. To ensure correct visualization, the image is rotated to the left, which does not introduce any artifacts. The cropping is performed using the following MATLAB operation: foto = foto(4:end, 1:end-3); This operation is necessary because the camera captures images of 1088 × 2048 pixels but disregards the last three columns and three rows. The useful portion is therefore 1085 × 2045 pixels (since the superpixel size is 5). If the image is rotated 90° to the left, the first three rows and the last three columns must be ignored. The function requires a variable named foto in the workspace. The complete code for estrazione ipercubo.m is provided in Appendix B.

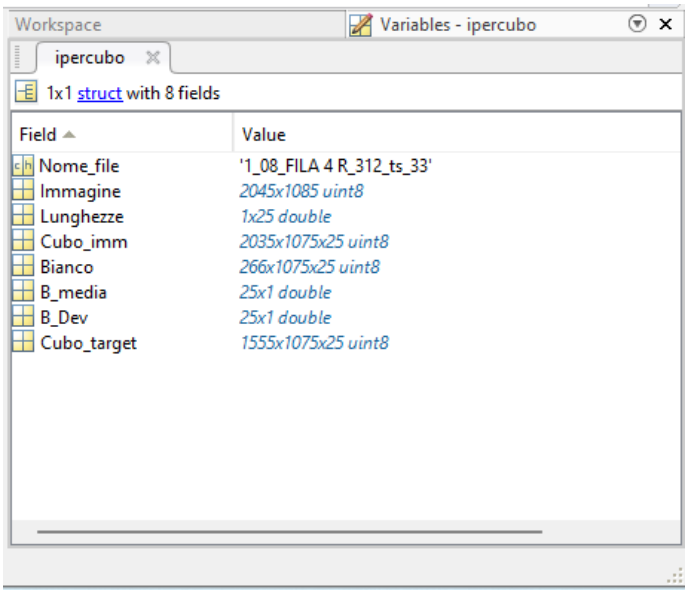


Figure 3.8: Output: "Ipercubo" data struscture.

3.2.5 ROI selection

For each extracted hypercube, the Regions of Interest (ROIs) were selected using the h.segmentation.m script. Specifically, a hybrid approach was employed to extract the ROIs corresponding to grape berries in the acquired images. This method combines spatial features related to reflectance with spectral characteristics. The steps involved can be divided into two categories: spatial and spectral. Spatial steps:

- computation of the 3D reflectance hypercube;
- application of an intensity threshold to a 2D image at 700 nm (a single slice of the 3D hypercube), highlighting only bright areas;
- application of the Circular Hough Transform to detect circular shapes within the 2D image.

Spectral steps:

- computation of the first-derivative (D1) hypercube, with the same dimensions as the 3D hypercube;
- application of an intensity threshold to the D1 image at 700 nm, highlighting only dark areas.

A mask corresponding to the identified circles is then applied, using the spectral derivative at 700 nm. As shown in Figure 9, this procedure highlights only the regions corresponding to pixels of grape berries directly exposed to sunlight. This result is obtained by applying thresholding techniques to both the original image and its spectral derivative at 700 nm.

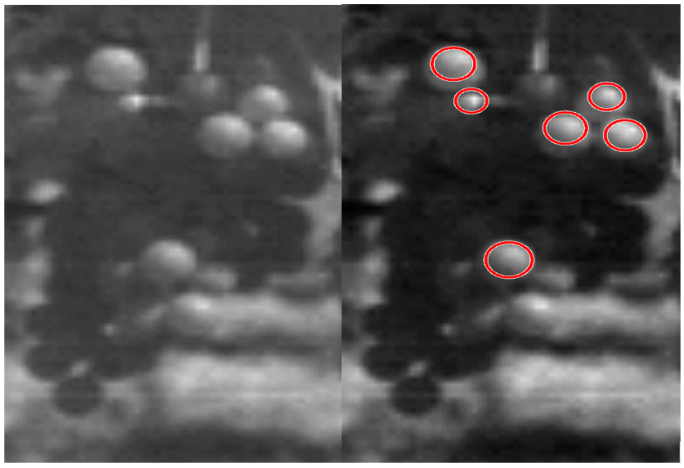


Figure 3.9: ROI, circles detected using Circular Hough Transform, first derivative D1 at 700 nm, circles corresponding to berries.

3.2.6 Datastore

For each acquired image, the average reflectance spectrum of the ROI was extracted and used to populate the data store. The corresponding reference parameters (°Brix, pH, and titratable acidity) were automatically associated with each spectral record. The resulting dataset consists of multispectral data comprising 25 spectral bands. This structured data store served as the foundation for subsequent modelling and analysis steps.

3.2.7 Model development

The spectral data (25 points) were first smoothed using the Savitzky–Golay method (polynomial order: 2; smoothing points: 3) and subsequently pre-treated by applying standard normal variate (SNV) correction, first and second derivatives (D1 and D2), and finally mean centering (MC). The SNV is among the most employed pre-processing techniques to correct spectral variations related to differences in optical path length and light scattering, while derivative transformations help remove both additive and multiplicative effects in the spectra (Rinnan et al., 2009). Principal Component Analysis (PCA) was employed as an exploratory tool to visualize data distribution in relation to °Brix and other quality parameters, as well as to identify potential outliers prior to model development. Subsequently, Partial Least Squares (PLS) regression models were developed to estimate the quality attributes (SSC, pH, and acidity). Model validation was performed using the Venetian blind cross-validation method (segments: 10). The robustness of the models and the absence of overfitting were further evaluated through a 50-permutation test. To prevent overfitting, the optimal number of latent variables (LVs) was selected by identifying the global minimum of the root mean square error in cross-validation (RMSECV). All the models were developed using PLS Toolbox for Matlab2023a. Initially, the models were developed considering all available spectra. However, since the reference measurements (°Brix, pH, and acidity) were obtained from samples collected by treatment per row, a second set of models was computed by averaging the spectral

data, specifically, by calculating the mean spectrum from the three spectral acquisitions per treatment (for each rows and for each treatment time), for a total of 52 samples. All the models were developed using PLS Toolbox for Matlab2023a.

3.3 Results

The reference analytical data used to calibrate the PLS models included soluble solids content (SSC, expressed as °Brix), titratable acidity (TA, g L^{-1}), and pH. The measurements covered a broad range of ripening stages, ensuring adequate variability for model development (Table 3.2). SSC ranged from 9.1 to 20.9 °Brix (mean = 16.9 ± 2.8), TA from 5.6 to 26.6 g L^{-1} (mean = 9.4 ± 5.3), and pH from 2.9 to 3.6 (mean = 3.3 ± 0.2). The relationships among these parameters reflected the typical evolution of grape composition during ripening (Figure 10). As expected, SSC showed a strong positive correlation with pH (R^2 = 0.81) and an inverse, nonlinear relationship with TA (R^2=0.87). These trends confirm that increasing sugar accumulation is associated with the progressive decrease in acidity and the corresponding rise in pH, consistent with the physiological processes underlying grape maturation.

Parameter	**°Brix**	**TA (g L^{-1})**	**pH**
Mean	16.9	9.4	3.3
SD	2.8	5.3	0.2
Min	9.1	5.6	2.9
Max	20.9	26.6	3.6

Table 3.2: Statistics of the reference parameters

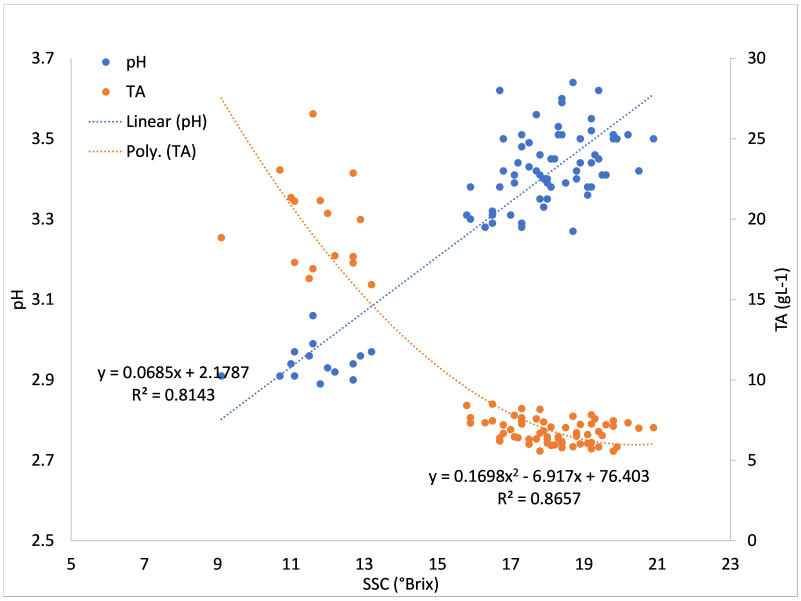


Figure 3.10: Relation between reference parameters.

3.3.1 Chemometric models

Figure 11 displays the raw, SNV-corrected, and first derivative (D1) spectra for all samples (25 wavelengths). The visual inspection of the overlaid spectra highlights that, in a multispectral configuration with a limited number of wavelength points, the application of the standard normal variate (SNV) correction substantially compresses the spectral variability among samples, leading to a marked reduction in the differences between curves. This occurs because, when only a few spectral points are available, the estimation of each spectrum's mean and standard deviation becomes less robust, and the normalization tends to flatten meaningful variations in signal intensity. SNV standardizes each spectrum by subtracting its mean and dividing by its standard deviation. With limited spectral resolution (few data points), the computed mean and standard deviation for each individual spectrum are unstable, small measurement noise or random fluctuations can have a disproportionately large effect. As a result, the normalization can artificially compress the dynamic range of the spectra and flatten real differences in spectral magnitude between samples. In contrast, with full hyperspectral data (hundreds of contiguous wavelengths), SNV is more reliable because the estimates of the mean and variance are statistically robust, and the method effectively corrects for scattering effects without erasing chemical information In contrast, the first derivative preserves and even enhances the spectral differences by emphasizing local gradients and inflection points rather than rescaling the overall signal. As a result, discriminative features related to chemical or physical differences among samples remain more clearly detectable. These observations indicate that, for multispectral datasets characterized by sparse wavelength sampling, derivative-based pre-treatments may be more effective in maintaining relevant spectral information than global normalization approaches such as SNV.

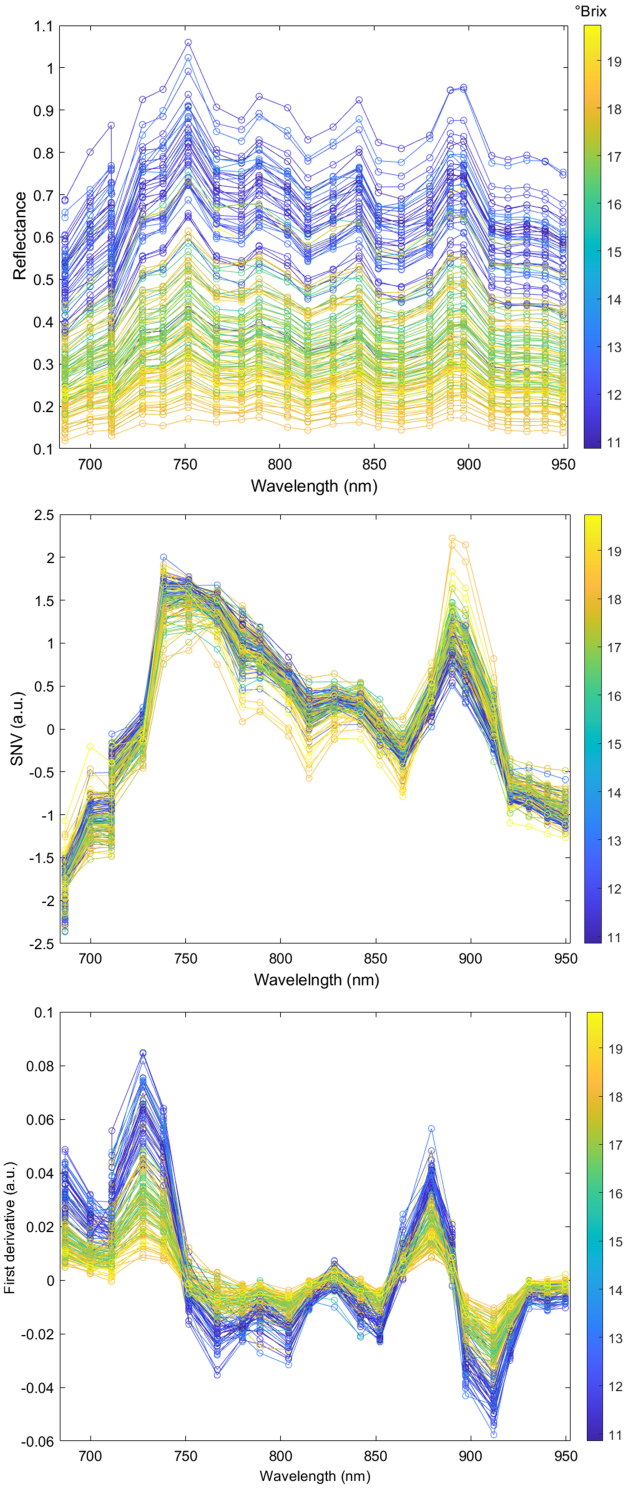


Figure 3.11: Raw and pre-processed spectra.

In the near-infrared region between 700 and 950 nm, the spectral features observed in grape samples are mainly associated with the absorption of water and the vibrational overtones of organic compounds such as sugars, acids, and phenolic constituents. Specifically, the region around 750–760 nm corresponds to weak overtones related to C–H stretching in organic molecules, while the broad absorption near

970 nm (often extending from 920 to 980 nm) is primarily attributed to the first overtone of the O–H stretching vibration of water molecules, which dominates the spectral profile due to the high moisture content of grape berries. In addition, subtle variations in reflectance across this range can be linked to differences in soluble solids content (°Brix) and acidity, since both parameters are correlated with changes in the concentration of sugars and organic acids that affect the overall absorption intensity of C–H and O–H overtone bands (Cozzolino et al. 2011; Cozzolino et al., 2006; Cayeta et al., 2010). The interpretation of the PCA results relies on the combined analysis of the score and loading plots, which together provide complementary information about the dataset structure. The score plot illustrates the distribution of the samples in the new coordinate system defined by the principal components, revealing trends, groupings, or gradients among samples (e.g., related to compositional parameters such as °Brix). In contrast, the loading plot describes how the original variables (in this case, the wavelengths) contribute to each principal component, indicating which spectral regions are most responsible for the observed separations or trends in the score space. By jointly examining the two plots, it is possible to identify the spectral features that drive the variation represented by each principal component: samples that move along a given direction in the score plot are influenced by variables showing high positive or negative loadings on the same component. Figure 12 show the score plot of the first two principal components (PC1: 86.81%; PC2: 6.62%) obtained from PCA performed on first-derivative spectra. Samples are distributed along PC1 according to °Brix values, with low °Brix samples on the right and high °Brix samples on the left, indicating that PC1 captures the spectral variability related to sugar accumulation. The corresponding loading plot (figure 13) revealed that the positive loadings on PC1 were mainly associated with the wavelength regions 686–751 nm and 851–879 nm. These spectral intervals are known to be influenced by overtones of C–H and O–H vibrations, which are sensitive to changes in sugar and water content in the grape berries. Therefore, the trend observed along PC1 can be interpreted as reflecting the progressive accumulation of soluble solids (°Brix) during ripening, as captured by variations in these specific spectral regions.

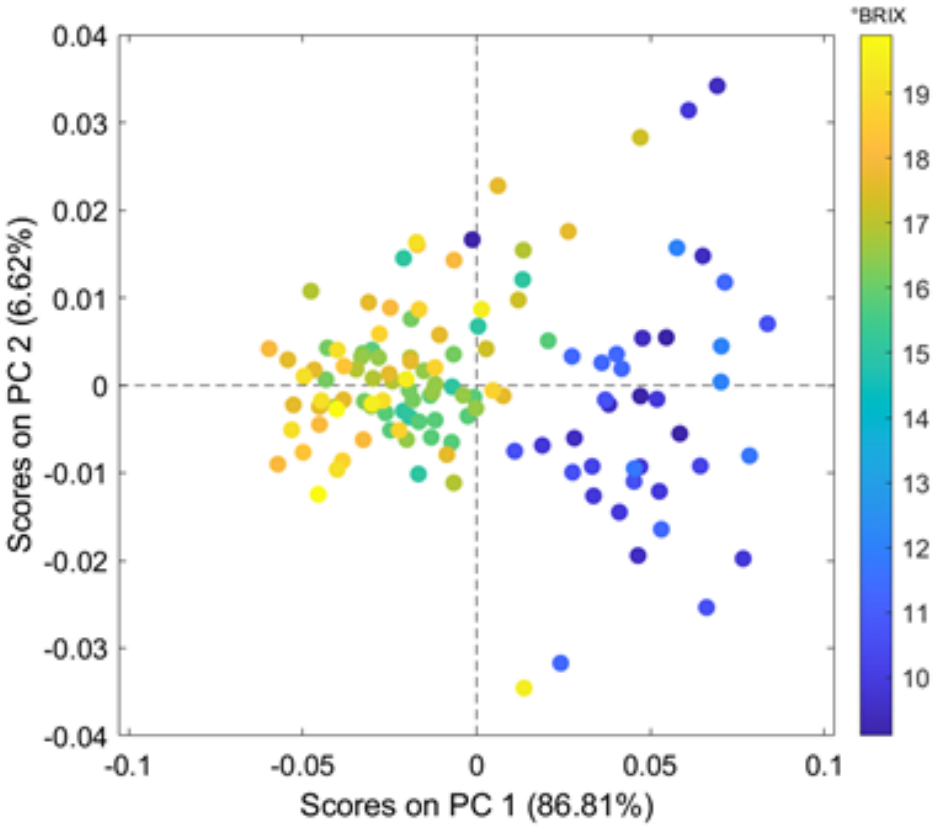


Figure 3.12: PCA score plot.

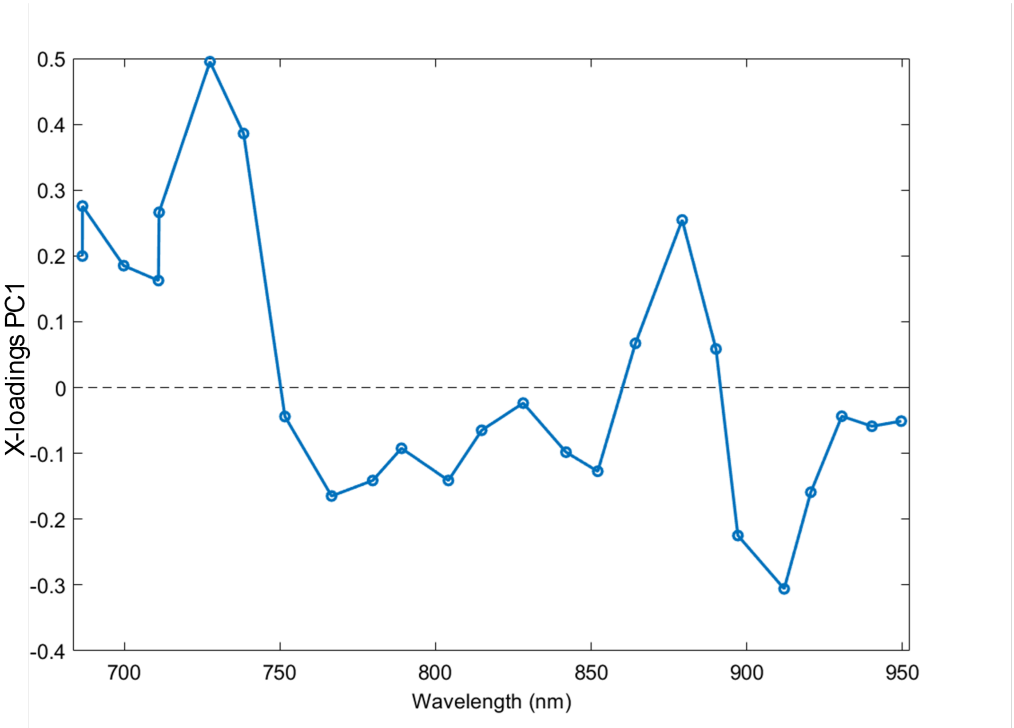


Figure 3.13: PCA X-loadings plot (PC1).

PLS regression models were developed to predict soluble solids content (SSC, °Brix), pH, and titratable acidity (TA) from the spectral data. Different spectral pre-treatments were applied prior to model calibration, including raw spectra, first derivative (1D), and second derivative (2D), in order to evaluate their influence on model performance. Model performance was assessed using the coefficient of determination (R^2) and the root mean square error (RMSE), both in calibration (C) and cross-validation (CV), along with the optimal number of latent variables (LV) determined by minimizing RMSECV (Table 2). The PLS models showed consistent and comparable performance across the three quality parameters. For SSC (°Brix), R^2 values ranged from 0.79 to 0.81 in calibration and 0.77–0.78 in cross-validation, with RMSECV values around 1.28–1.30 °Brix. The pH models exhibited slightly higher coefficients, with R^2 values between 0.85 and 0.86 in calibration and 0.77–0.78 in cross-validation, and RMSECV values close to 0.10. Similarly, titratable acidity (TA) models yielded R^2 values of 0.81–0.82 in calibration and 0.77 in cross-validation, with corresponding RMSECV values around 2.7 g L^{-1}. No noticeable effect of the spectral pre-treatment (raw, first, or second derivative) was observed for any of the three parameters, indicating that the limited number of spectral bands in the multispectral data was sufficient to capture the main variability associated with these quality attributes. The slight reduction in R^2 and increase in RMSE from calibration to cross-validation were consistent among parameters, confirming the overall stability and robustness of the developed model. In addition

Parameter	**Pre-processing**	**R^2 (C)**	**R^2 (CV)**	**RMSE (C)**	**RMSE (CV)**	**LV**
SSC (°Brix)	Raw	0.81	0.77	1.24	1.30	3
	1D	0.81	0.78	1.24	1.28	3
	2D	0.79	0.78	1.25	1.28	5
pH	Raw	0.85	0.78	0.08	0.10	8
	1D	0.86	0.78	0.08	0.10	7
	2D	0.85	0.77	0.09	0.11	8
TA (g L^{-1})	Raw	0.81	0.77	2.47	2.68	3
	1D	0.82	0.77	2.39	2.72	5
	2D	0.81	0.77	2.43	2.72	4

Table 3.3: PLS results obtained considering all the spectra.

to the models developed using all individual spectra, a second set of PLS models was built using the mean spectra calculated for each treatment (. This approach aimed to align the spectral dataset with the reference measurements (°Brix, pH, and acidity), which were collected as averaged values per treatment. The same pre-processing and validation procedures were applied as in the previous modelling stage (Table 3). The PLS models developed using mean spectra exhibited improved or comparable predictive performance relative to those based on all individual spectra, particularly for SSC (°Brix) and pH. For °Brix, R^2 values increased to 0.88–0.89 in calibration and 0.80–0.81 in cross-validation, while RMSECV decreased to about 1.2

°Brix, indicating that averaging spectra helped to reduce intra-treatment variability and slightly enhance prediction accuracy. The pH models showed a more evident improvement, with R^2 values up to 0.91 in calibration and 0.82 in cross-validation, and RMSECV values around 0.09, confirming that the mean spectra provided a more stable and representative input for modeling this parameter. Similarly, titratable acidity (TA) models exhibited high and consistent performance, with R^2 values up to 0.90 in calibration and 0.84 in cross-validation, and RMSECV values around 2.2 g L^{-1}.Regarding the effect of spectral pre-treatment, only minor differences were observed among the raw, first-, and second-derivative spectra. The application of the first derivative slightly improved model performance, particularly for pH, likely due to its ability to enhance subtle spectral features and reduce baseline offsets. However, the improvements were not substantial for SSC and TA, suggesting that the limited number of wavelengths in the multispectral data already provided a sufficiently clean signal, with minimal baseline and scattering effects to correct. This indicates that, in a multispectral configuration, the influence of pre-processing on model performance is limited and largely dependent on the specific parameter considered. The robustness of all PLS models was further verified by a permutation test coupled with statistical significance testing. The results confirmed that all models were statistically significant ($p < 0.05$), indicating that the relationships captured by the regression were not attributed to chance. The Variable Importance

Parameter	**Pre-processing**	**R^2 (C)**	**R^2 (CV)**	**RMSE (C)**	**RMSE (CV)**	**LV**
SSC (°Brix)	Raw	0.88	0.80	0.94	1.25	5
	1D	0.88	0.80	0.97	1.23	6
	2D	0.89	0.81	0.91	1.20	5
pH	Raw	0.84	0.78	0.08	0.10	3
	1D	0.91	0.82	0.06	0.09	5
	2D	0.88	0.78	0.07	0.10	4
TA (g L^{-1})	Raw	0.90	0.84	1.77	2.20	4
	1D	0.88	0.83	1.89	2.26	3
	2D	0.89	0.84	1.82	2.23	3

Table 3.4: PLS results obtained considering the mean spectra

in Projection (VIP) scores were calculated for all PLS models in order to evaluate the contribution of each spectral variable to the prediction of the different quality parameters. For clarity, only the VIP profile of the PLS model for SSC (°Brix) based on first-derivative spectra is presented and discussed here, as it is representative of the overall trends observed across models (Figure 14). Variables with VIP values greater than 1 are generally considered to have a relevant influence on the model and to contribute significantly to the prediction of the response variable. Notably, the wavelengths showing the highest VIP scores correspond to the spectral regions that exhibited the most influential loadings in the PC1 obtained during PCA, indicating a strong consistency between the exploratory (PCA) and predictive (PLS) analyses.

This correspondence suggests that the same spectral features responsible for sample distribution along the principal components are those most relevant for the prediction of °Brix in the PLS model.

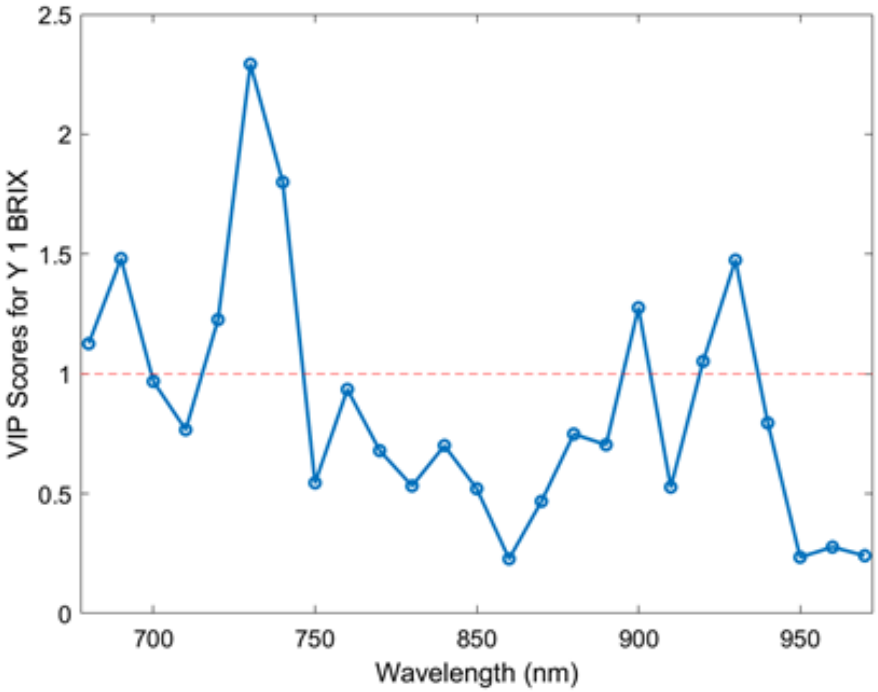


Figure 3.14: VIP scores of the PLS model developed to predict °Brix.

When comparing the predictive accuracy of the developed PLS models with similar in-field studies, the root mean square error (RMSE) provides a more consistent and interpretable measure of model performance than the coefficient of determination, which can vary with sample range and size. For SSC (°Brix), the mean-spectra models developed in this study achieved RMSECV values around 1.2 °Brix, in line with results obtained under comparable field conditions. Gutiérrez et al. (2019) reported RMSE values of 1.36 °Brix (cross-validation) and 1.27 °Brix (prediction) using on-the-go hyperspectral imaging for grape berries. Similarly, Benelli et al. (2021) achieved an RMSECV of 0.79 °Brix for SSC estimation using hyperspectral imaging in the field, while Fernandes-Novales et al. (2020) obtained RMSECV below 1.24 °Brix for TSS prediction through VIS–SW–NIR spectroscopy under vineyard conditions. These comparisons indicate that the performance of the mean-spectra PLS models developed in this work is fully consistent with those obtained from more complex hyperspectral systems, despite the use of a reduced number of multispectral bands. The achieved errors fall within or even below the typical range reported for in-field estimation of grape quality traits, confirming the suitability of the proposed approach for practical vineyard monitoring applications. The satisfactory predictive performance obtained for pH and titratable acidity (TA), despite the lack of specific absorption features in the NIR region, can be explained by indirect correlations with other physicochemical parameters, primarily soluble solids content (SSC). During grape ripening, the accumulation of sugars is typically accompanied by a decrease

in organic acids and a consequent increase in pH, resulting in strong physiological interdependence among these variables. Therefore, spectral variability primarily related to water and sugar absorptions can indirectly capture the ripening-related changes that also affect acidity and pH. Similar findings have been reported in previous studies, where NIR or hyperspectral models achieved reliable predictions for pH and TA even in the absence of direct spectral responses. Gutiérrez et al. (2019), Benelli et al. (2021), and Diago et al. (2021) highlighted that these parameters are often estimated through their covariation with SSC and other maturity indicators. Moreover, Gomes et al. (2021) and Ferrera et al., (2022) reported RMSE values of approximately 0.07-0.12 pH units and 2-4 g L^{-1} for TA under field or semi-controlled conditions, supporting the interpretation that such predictions arise from indirect correlations rather than direct NIR sensitivity to acids or hydrogen ions.

3.4 Conclusions

The results presented in this chapter demonstrate that multispectral imaging can provide reliable and consistent estimations of key grape quality attributes under real field conditions. Despite the reduced spectral resolution compared to hyperspectral systems, the combination of proper radiometric calibration, careful preprocessing, and chemometric modelling allowed the prediction of soluble solids content, pH, and titratable acidity with performance levels comparable to those obtained from more complex hyperspectral setups. These findings confirm that a well-designed multispectral system, when properly calibrated and operated, represents a viable and economically sustainable alternative for in-field sensing applications in precision viticulture.

Beyond the quantitative results, the study highlights the importance of integrating methodological robustness with practical deployability. The field trials validated not only the optical configuration and acquisition workflow but also the potential for dynamic and repeatable data collection directly from agricultural machinery. This transition from laboratory to operational environments marks a decisive step toward scalable and accessible monitoring systems for vineyard management.

Having established that multispectral sensing can achieve satisfactory predictive accuracy in real-world scenarios, the next logical step is to explore whether similar or even improved information can be retrieved from simpler RGB imagery through computational means. The following chapter therefore extends this investigation into the domain of *spectral reconstruction*, analysing how deep learning models can infer high-dimensional spectral data from conventional RGB inputs. This approach aims to bridge the gap between the cost-effectiveness of RGB systems and the analytical depth of hyperspectral imaging, paving the way toward a new generation of flexible, low-cost optical tools for precision agriculture.

Bibliography

[1] D. Cozzolino et al. “Analysis of grapes and wine by near infrared spectroscopy”. In: *Journal of Near Infrared Spectroscopy* 14.6 (2006), pp. 279–289. DOI: 10.1255/jnirs.642.

[2] J. A. Cayuela and C. Weiland. “Intact grape berry characterization by NIR reflectance spectroscopy”. In: *Postharvest Biology and Technology* 58.2 (2010), pp. 143–150. DOI: 10.1016/j.postharvbio.2010.06.005.

[3] D. Cozzolino et al. “Multivariate data analysis applied to spectroscopy: Potential application to juice and fruit quality assessment”. In: *Food Research International* 44.7 (2011), pp. 1888–1896. DOI: 10.1016/j.foodres.2011.01.047.

[4] A. Benelli et al. “In-field and non-destructive monitoring of grapes maturity by hyperspectral imaging”. In: *Biosystems Engineering* 207 (2021), pp. 59–67. DOI: 10.1016/j.biosystemseng.2021.05.004.

[5] S. Gutiérrez et al. “On-the-go hyperspectral imaging for the in-field estimation of grape berry soluble solids and anthocyanin concentration”. In: *Australian Journal of Grape and Wine Research* 25 (2019), pp. 127–133. DOI: 10.1111/ajgw.12377.

[6] S. Gutiérrez et al. “Hyperspectral imaging and CNN for on-the-go, non-destructive assessment of grape composition in the vineyard”. In: *Proceedings of the 21st GiESCO International Meeting*. Thessaloniki, Greece, 2019.

[7] F. Bertoglio, J. Tardaguila, and M. P. Diago. “On-the-go table grape ripeness estimation via proximal snapshot hyperspectral imaging”. In: *Computers and Electronics in Agriculture* 215 (2024), p. 108589. DOI: 10.1016/j.compag.2024.108589.

[8] J. Pérez-Pérez et al. “Vineyard proximal sensing using multispectral imaging to evaluate grape composition and quality parameters”. In: *Journal of Agricultural and Food Research* 14 (2025), p. 100895. DOI: 10.1016/j.jafr.2025.100895.

[9] G. A. Moreira et al. “Determination of sugar, pH, and anthocyanin contents in Port wine grape berries through hyperspectral imaging: An extensive comparison of linear and non-linear predictive methods”. In: *Applied Sciences* 11.21 (2021), p. 10319. DOI: 10.3390/app112110319.

[10] J. L. Aleixandre-Tudó, H. Nieuwoudt, and W. J. Du Toit. “Ripeness prediction in table grape cultivars by using a portable NIR device”. In: *Horticulturae* 8.7 (2022), p. 613. DOI: 10.3390/horticulturae8070613.

Chapter 4

Spectral Reconstruction Applied in Precision Agriculture: On-Field Solutions

Scientific publication

M.Mingrone[1,*] M. Seracini[2], C. Cevoli[1]

[1]Department of Agricultural and Food Sciences, Alma Mater Studiorum, University of Bologna, 40126 Bologna, Italy
[2]Department of Physics and Astronomy "Augusto Righi", Alma Mater Studiorum, University of Bologna, 40126 Bologna, Italy

*Author to whom correspondence should be addressed.

Appl. Sci. **2025**, 15(20), 10985; https://doi.org/10.3390/app152010985
Submission received: 25 August 2025 / Revised: 29 September 2025 / Accepted: 8 October 2025 / Published: 13 October 2025
(This article belongs to the *Special Issue: Signal and Image Processing: From Theory to Applications: 2nd Edition*)

Abstract

Over the past two decades, hyperspectral imaging (HSI) systems have shown significant potential in agriculture, from disease detection to the assessment of plant and fruit nutritional status. However, most applications remain confined to laboratory analyses under controlled conditions, with only a limited fraction implemented in field environments. In this scenario, spectral reconstruction techniques may serve as a bridge between the high accuracy of HSI and the challenges of on-field or even real-time applications. This review outlines the current state of the art of on-field HSI in the agrifood sector, highlighting existing limitations and potential

advantages. It then introduces the problem of spectral reconstruction and reviews current techniques used to address it. Laboratory and on-field studies will be taken into account. The final section offers our perspective on the limitations of HSI and the promising potential of spectral super-resolution to overcome current barriers and enable broader adoption of hyperspectral technology in precision agriculture.

Keywords: spectral super-resolution; multispecral imaging; hyperspectral imaging; deep learning; proximal sensing.

4.1 Introduction

Spectral reconstruction from RGB images - that is, generating multispectral or hyperspectral images from simple three-channel images (red, green, and blue) - is a topic that has received increasing attention in the last decade. Native hyperspectral images, acquired with hyperspectral cameras, provide richer information than traditional RGB images, capturing data across multiple narrow spectral bands, from tens to hundreds of bands. It allows for detecting chemical and physical properties in targets on the scene, related to the spectral signature, without any damage or invasive sampling. This capability enables advanced applications in various sectors, including remote sensing, agriculture, geology, medicine, environmental monitoring, and cultural heritage preservation [1]. Within environmental monitoring, air pollution is a topic where hyperspectral images are the best source of information. However, hyperspectral cameras are generally complex, expensive, and often impractical for large-scale applications or consumer contexts. To overcome these limitations, a line of research has developed aimed at reconstructing detailed spectral information from RGB images by leveraging advanced computational techniques and machine learning models [2]. Generally, the reconstruction of hyperspectral images from RGB images is a complex challenge due to information loss and limitations in data quality. First of all, there are problems of the non-uniqueness of the solution: reconstructing the spectrum from an RGB image is not unique, as different spectral distributions can correspond to the same RGB values. Given the sensor response, multiple input spectra can lead to identical integral values. Moreover, there is lack of an RGB camera frequency response. So, we do not know exactly how the camera works with the incoming spectrum. An important aspect of this problem is that many models require knowledge of the spectral response of the sensor (e.g. attention-block models) [1], which are not available for common RGB cameras. In addition, we must couple with data degradation. Raw image values contain unknown pixel responses directly related to the incoming electromagnetic waves, but JPEG compression, white balance, and gamma correction degrade the original spectral information. However, recent studies (from regression models to deep learning models) demonstrate promising results, including predictions of spectral information beyond the visible spectrum, such as in the NIR (near-infrared) region. The methods proposed in the literature can primarily be divided into two general categories: prior-based methods and data-driven methods. Prior-based methods exploit statistical information and

spatial or spectral characteristics, such as sparsity or spectral correlation [1]. On the other hand, data-driven methods mainly rely on deep neural networks, which have demonstrated a great ability to model complex relationships between RGB data and corresponding spectral information [3]. Recent research has significantly focused on the use of deep learning techniques to address the problem of spectral reconstruction. Approaches based on deep convolutional neural networks (CNNs), as well as more advanced architectures, such as Generative Adversarial Networks (GANs), have shown remarkable effectiveness, especially when trained on large datasets [1]. A critical issue in this field concerns the physical accuracy of reconstructed images. Many current methods, although achieving good spectral accuracy, may generate results that do not respect physical plausibility—meaning that reconstructed images do not precisely yield the original RGB values when reintegrated with the camera's spectral sensitivities. To address this challenge, recent studies have developed physically plausible reconstruction methods, introducing constraints based on the decomposition of spectral information into fundamental metameric components. This allows for a reconstruction that is physically coherent and robust to exposure variations [4]. A further advancement in research involves the introduction of category-specific prior information—additional details describing particular object types or surfaces present in the images—to further improve reconstruction quality [5]. Finally, an important driver for developing these approaches has been the creation of new datasets, such as ARAD-HS [6], which offers a wide variety of natural hyperspectral images. These datasets enable rigorous validation and performance comparisons among the various developed methods [7]. It is important to note that in the analyzed studies, RGB and hyperspectral images can be treated either as directly measured luminance data or as reflectance data, where the acquired signal is normalized against the light source and environmental conditions. However, it should be emphasized that many commonly cited datasets, such as ARAD-HS, CAVE, ICVL, and BGU-HS [1], do not provide detailed calibration procedures for precise conversion to reflectance, one of the most important steps in order to extract numerical information from the images related to the status of plant and fruits. The only dataset explicitly described with rigorous calibration via a standard white panel is the KAUST-HS dataset [1, 6]. Consequently, the use of reflectance data in studies based on these datasets should be approached cautiously, recognizing potential limitations in their accuracy and generalizability. Applying these models to images acquired under uncontrolled conditions remains an open challenge but represents a significant area of research for food-quality analysis and other natural material assessments. This study presents a detailed analysis of the topics previously introduced, beginning with the current state of the art in conventional hyperspectral imaging (HSI), with particular attention to methodologies, algorithms, and their limitations. On this basis, we then examine spectral reconstruction, highlighting the advantages it provides as well as the constraints that characterize it, with specific reference to the earliest applications reported in the agri-food domain.

4.2 Literature Search and Study Selection Methodology

4.2.1 Databases, Publisher Platforms, and Coverage

We conducted structured searches on two citation databases—Web of Science Core Collection and Scopus—and complemented them with Google Scholar to broaden recall, as reported in Table 4.1. Full texts were retrieved on publisher platforms when available, with a focus on ScienceDirect for Elsevier journals, and on MDPI, IEEE Xplore, SpringerLink, and Wiley Online Library for non-Elsevier titles. These choices align with the venues represented in our reference list (e.g., Elsevier journals such as *Information Fusion*, *Computers and Electronics in Agriculture*, and the *Microchemical Journal*).

4.2.2 Search Strategy and Keywords

Query strings combined controlled and free-text terms using Boolean operators and truncation. Core terms were derived from the review scope and keywords of the manuscript ("Spectral super-resolution", "Multispectral Imaging", "Hyperspectral Imaging", "Deep Learning", "Precision Agriculture", "Proximal sensing", "On-field") and were specialized per context (HSI general, on-field HSI, SSR, algorithms/models). Queries were iteratively refined to balance precision (algorithm/dataset-specific terms) and recall (broader agrifood terms).

4.2.3 Eligibility Criteria

Inclusion criteria: (i) peer-reviewed journal articles or full conference proceedings; (ii) relevance to hyperspectral imaging in agrifood or to spectral super-resolution (RGB-to-HSI/HSR) with agrifood applicability; (iii) English language; (iv) publication year 1998–2025. Exclusion criteria: non-peer-reviewed items; purely satellite-only remote sensing without proximal/on-field relevance; studies lacking spectral/agrifood focus; duplicates.

4.2.4 Screening Workflow

Titles/abstracts were screened to remove off-topic items. The remaining records underwent full-text assessment; reasons for exclusion (e.g., missing agrifood context, insufficient spectral content) were logged. Reference chaining (backward/forward) was used to capture seminal works (e.g., calibration, band selection, classic HSI datasets) and recent SSR applications.

Database	HSI	HSI On-Field/Proximal	Spectral Reconstruction/SSR	Algorithms/ML & CV	Filters
Web of Science	"hyperspectral imaging" AND (agriculture OR crop OR fruit OR food); "multispectral imaging" AND agrifood; "reflectance" AND (VIS OR NIR OR "red edge"); "band selection"	"on-field" OR field OR "proximal sensing" OR UAV OR "handheld" AND "hyperspectral"; "radiometric calibration" AND (panel OR white/-dark); "lighting control"	"spectral super-resolution" OR "spectral reconstruction" OR "RGB-to-hyperspectral" OR "RGB-to-HSI"; "camera spectral response" OR CSR; NTIRE AND ("spectral" OR HSR)	HSCNN OR HSCNN+; MST OR MST++; Transformer OR "spectral-wise transformer"; CNN OR GAN; PLSR; SVM; Random Forest; "band selection" AND (GA OR CARS)	Years: 1998–2025; Language: English; Doc types: Article, Review, Proceedings; WoS Categories: Agriculture; Imaging Sci. & Photonics; Food Sci. & Tech.; Comp. Sci. (AI/CV)
Scopus	Same as WoS plus: "vegetation indices" OR (NDVI OR NDRE OR GDVI); "SWIR"	Same as WoS plus: canopy OR "in situ"; platform: UAV OR trolley OR gantry	Same as WoS plus: "physically plausible" AND reconstruction; "exposure invariance"	Same as WoS plus: "domain adaptation"; "transfer learning"; "self-supervised"	Years: 1998–2025; English; Subject areas: AGRI/BIOL, ENGI, COMP; Doc types: Article/Conference/ Review
Google Scholar	As above, using quotes for phrases and minus terms to suppress noise (e.g., satellite when targeting proximal)	As above; query chaining with site: filters (e.g., site:unibo.it for theses if needed, later excluded)	As above; add dataset names (CAVE, Harvard, Chikusei, WHU-Hi) to find benchmarks	Model acronyms + task (e.g., MST "reconstrn")	Any time → custom range 1998–2025; Exclude patents/citations
ScienceDirect	"hyperspectral imaging" AND (agriculture OR postharvest OR phenotyping)	"field" OR "on-field" OR UAV within agrifood journals (e.g., Computers and Electronics in Agriculture, Agricultural Water Management)	"spectral super-resolution" OR "RGB-to-HSI" within Information Fusion, Results in Engineering, Smart Agricultural Technology	"deep learning" AND hyperspectral; "transformer"; "GAN" within target journals	Journals only; Subject Areas: Agricultural & Biological Sciences; Computer Science; 1998–2025

Table 4.1: Databases/platforms,context-specific keywords used in queries, and applied filters.

4.3 Traditional HSI On-Field

HSI is emerging as a key technology across a wide range of agri-food applications. The main advantage of this technology lies in its non-invasive approach, as it enables the analysis of objects within a scene by simultaneously combining imaging and spectroscopy. The result of this measurement is a three-dimensional data cube, or hypercube, consisting of two spatial dimensions and one spectral dimension. Each acquired pixel is characterized by a complete spectrum—reflected, transmitted, or absorbed by the sample—commonly referred to as its spectral signature. Beyond laboratory and industrial systems, hyperspectral imaging can also be implemented on different platforms, including ground-based setups, UAVs, airborne systems, and satellites, thus covering a wide range of spatial scales and application domains.

The development of regression and classification models based on hyperspectral imaging (HSI) in agriculture generally follows a consolidated methodological pipeline, which, despite its modularity, remains consistent across most studies as shown in Figure 4.1. The first critical decision concerns the acquisition system, where the choice of the hyperspectral camera and platform determines the spectral range, resolution, and signal-to-noise characteristics of the resulting datacube ($x \times y \times \lambda$). Indoor gantry systems and imaging boxes are frequently employed for controlled experiments and calibration datasets, while ground-based vehicles, such as tractors or unmanned ground vehicles (UGVs), allow for direct deployment in the field. Unmanned aerial vehicles (UAVs) and manned airborne platforms extend coverage to larger areas, with satellites providing long-term and wide-scale monitoring. Selecting the sensor technology (e.g., pushbroom, whiskbroom, snapshot, or spectral scanning cameras) is equally decisive, as each modality entails specific trade-offs in terms of spatial resolution, acquisition speed, and robustness to motion. Once data are acquired, extensive pre-processing is typically required to ensure reliability. This includes radiometric corrections, such as white/dark reference calibration and spectral normalization, geometric adjustments for distortion removal and georeferencing, and, in the case of airborne acquisitions, atmospheric corrections. Following pre-processing, a crucial step is segmentation and region-of-interest (ROI) extraction, where the objects of interest (leaf, fruit, canopy, or plot) are separated from background elements, such as soil, shadows, or crop residues. This is achieved through a variety of techniques, including morphological operations, thresholding, vegetation indices, spectral clustering approaches such as Principal Component Analysis (PCA), or more recently, deep neural segmentation methods. A critical aspect at this stage is the choice of the analytical strategy, which generally follows two main approaches [8]. The first relies on the ROI average spectrum, where the mean spectral signature of a region is used to represent the entire sample. This method is computationally efficient and suitable for relatively homogeneous materials or when only global classification is required, yet it inevitably overlooks local heterogeneity and may obscure subtle adulterations or localized defects. Conversely, the pixel-wise approach exploits the full spatial resolution by analyzing each pixel spectrum individually, thus enabling the visualization of chemical maps and the detection of spatially localized anomalies. While this strategy preserves the distinctive advantage of hyperspectral

imaging, it also introduces substantial computational demands, higher sensitivity to noise, and the need for advanced chemometric or deep learning models. Ultimately, the choice between the two reflects a trade-off between simplicity and representativeness, with recent studies often favoring the ROI approach for its practicality, even at the cost of underutilizing the spatial richness of hyperspectral data. The next phase concerns dimensionality reduction and feature extraction, necessary to condense the large spectral–spatial information into more manageable and discriminative representations. Beyond PCA, other methods, such as Minimum Noise Fraction (MNF), Independent Component Analysis (ICA), Partial Least Squares (PLS), or band selection strategies, are employed, while deep learning-based embeddings, such as autoencoders and Convolutional Neural Networks (CNNs), have gained increasing prominence. Building a robust dataset requires linking hyperspectral measurements to ground-truth labels obtained through expert annotation, field sampling and laboratory analysis—for instance, by measuring biochemical or physiological parameters (e.g., chlorophyll, nitrogen, soluble solids) through standard methods such as spectrophotometry, refractometry, or HPLC (High-Performance Liquid Chromatography). This is complemented by careful definition of training, validation, and test splits, with data augmentation techniques, such as geometric transformations (e.g., rotation, flipping, cropping), spectral perturbations (e.g., noise addition, spectral shifts), or synthetic sample generation via generative models, used to increase dataset variability and reduce overfitting in limited datasets. For modeling, regression tasks typically rely on algorithms such as Partial Least Squares Regression (PLSR), Random Forests (RF), or CNNs, targeting biochemical and physiological parameters. Classification problems—such as crop/weed discrimination or stress detection—are addressed with Support Vector Machines (SVMs), two- or three-dimensional CNNs, or hybrid models. Lightweight versions of these algorithms are increasingly adopted for real-time, on-board inference on GPUs (Graphics Processing Units) or FPGAs (Field-Programmable Gate Arrays). Model evaluation combines classical statistical measures with domain-specific metrics. Regression models are commonly assessed with the coefficient of determination (R^2), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), or the Ratio of Performance to Deviation (RPD). Classification models are evaluated through accuracy, precision, recall, F1-score, and confusion matrices. For spectral fidelity and image reconstruction, metrics such as the Spectral Angle Mapper (SAM), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM) are also employed. Finally, deployment translates model outputs into actionable tools. In agriculture, this includes the production of operational maps for nutrients or water stress, the integration of real-time predictions into precision spraying or micro-dosing systems, or the use of embedded pipelines for sorting and grading applications. Comparable workflows are reported in other domains—such as food quality and safety, medical imaging, and water or flood monitoring—where analogous sequences of acquisition, segmentation, feature extraction, and modeling are applied with domain-specific adaptations. For example, indoor calibration protocols are optimized for food safety, tissue- or biomarker-driven feature sets guide medical imaging, and water–soil–vegetation spectral separation is exploited in environmental monitoring [9, 10].

Table 4.2 compares the main platforms employed for hyperspectral data acquisi-

tion in precision agriculture and related domains. Ground-based systems (gantries, imaging boxes, UGVs) stand out for their high stability and controlled illumination, making them particularly suitable for calibration studies and reference dataset collection. UAVs offer a balance between flexibility and spatial detail, with established applications in vineyards, rice fields, and horticultural crops. Manned airborne systems provide regional coverage but are constrained by high costs and logistical complexity, whereas satellite missions deliver long-term time series on a global scale, albeit with a ground sampling distance (GSD) inadequate for micro-plots. Multi-UAV systems enable simultaneous coverage of multiple plots but require sophisticated coordination and data management strategies [10].

Table 4.3 highlights how the choice of acquisition modality directly affects data quality and suitability for specific scenarios. Pushbroom imagers remain the most widely adopted in agriculture due to their compact design and high signal-to-noise ratio (SNR), although they require careful motion correction. Whiskbroom scanners, despite their high spectral accuracy, are less suited to dynamic scenes or UAV deployment. Spectral scanning and snapshot systems are typically used in indoor or industrial sorting contexts, where rapid acquisition and reduced costs are prioritized.

Table 4.4 synthesizes the main families of algorithms applied to HSI analysis. Traditional machine learning approaches (PLSR, SVM, and RF) remain highly relevant for biochemical parameter estimation thanks to their efficiency with limited datasets and selected bands. Deep learning models (1D, 2D, 3D CNNs) leverage both spectral and spatial information, achieving superior accuracy in discrimination tasks such as weed detection or stress identification. Generative approaches (GANs, autoencoders) have emerged more recently to mitigate overfitting and generate synthetic samples. This convergence suggests that methodological innovations are increasingly transferable across application domains.

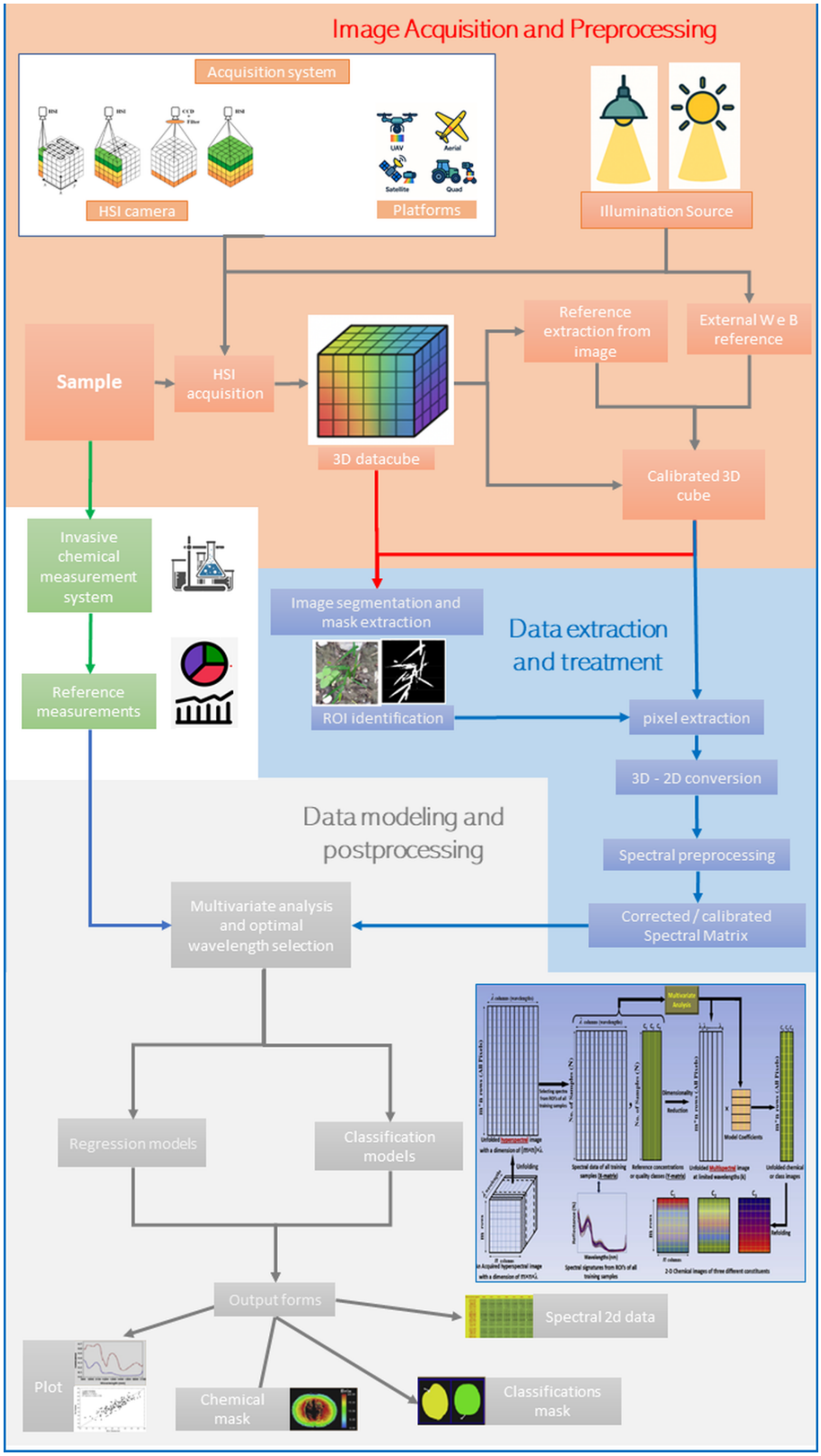


Figure 4.1: Traditional HSI workflow.

Platform	Typical Height	Coverage	Spatial Resolution	Advantages	Limitations	Products/Crops and Cited Studies
Ground-based fixed (gantry, imaging box, hypercarts)	1–5 m	m^2–tens of m^2	mm–cm	Illumination control; high stability; ideal for calibration/-training	Not scalable; indoor/-greenhouse	Maize/soy/wheat leaves; Hyper-cart/Imaging box [11, 12, 13]
Ground-based mobile (tractors, quad, UGV)	0.5–2 m	plots (ha)	cm–dm	In-field operability; integration with actuation (micro-spraying)	Wind/variable light; limited speed	Weed mapping, micro-dosing [14, 15, 16, 17, 18]
Helicopters	100 m–2 km	∼10 km^2	m–dm	Targeted coverage; hovering	Costs/logistics	Local/proximal monitoring [10]
UAV (multirotor/fixed-wing)	50–150 m	ha–km^2	cm–dm	Flexibility; high detail; rapid coverage	Autonomy; stability/-geometric registration	Vineyards, rice paddies, vegetables [19, 20, 21, 22]
Multi-UAV	<150 m	∼0.5 km^2	cm–dm (up to 0.01–0.5 m)	Acquisition parallelization	Overlap, communications	Multi-plot campaigns [10]
Aircraft (manned)	1–2 km	tens–hundreds km^2	m–dm	Regional coverage; area efficiency	Costs/logistics	Extensive crops (wheat/-maize) [23, 24]
Satellites (PRISMA, EnMAP, Hyperion)	400–700 km	global	10–30 m	Time series; wide coverage	GSD not suitable for micro-plots; not real-time	Vegetation/forests; Hyperion/CHRIS sensors [10]

Table 4.2: HSI acquisition platforms in precision agriculture and representative studies [9, 10].

Mode	Principle	Typical Applications	Pros	Cons and Studies
Pushbroom (line-scan)	Line acquisition (slit + dispersion)	UAV, vehicles, gantry	High SNR; contiguous bands	Sensitive to instability/motion; e.g.,: Specim FX10 on DJI M600 [20]
Whiskbroom (point-scan)	Point scanning with moving mirrors	Historical	Point spectral accuracy	Slow for dynamic scenes [9]
Staring imagers	Fixed 2D detector; stationary scene; spectral/optical scanning	Indoor, medical, lab	Simple optical scheme	Requires spectral modulation/filters, times > snapshot [10]
Spectral scanning (variable filters/spectrograph)	Variation of λ over time on mono sensor	Indoor, sorting, gantry	Lower cost; flexible	Long acquisition time; limited bands; Sony ICX414AL + LVF [14, 15]
Snapshot	Simultaneous spectral+spatial acquisition	Real-time, fast inspection	Fast; suitable for online	Lower resolution; storage; NIR snapshot for sorting [25, 10]

Table 4.3: HSI acquisition technologies/modes and use cases.

Category	Typical Algorithms	Operational Notes	Examples in Review
Traditional ML	PLSR, SVM, RF, DT, KPCA, SPCA	Feature engineering; efficient with few bands	Nutrients/stress; crop/weed discrimination [26, 27, 28]
Discriminant	PLS-DA, GAPLS	Classification/variable selection for food/medical	Cereals/salmon; breast cancer (FTIR+PCA) [10]
DL 1D (spectrum)	1D-CNN, MLP	Pixel-wise; fast; ignores spatial context	Leaf parameters [9]
DL 2D/3D (space+spectrum)	2D/3D-CNN, hybrids	Higher accuracy; computational cost	Weed/disease detection [29, 11]
Texture/Feature	GLCM, ANN	Spectral textures; food classification	Blueberries, salmon [10]
Generative/Augmentation	AE, GANs	Overfitting reduction; +synthetic samples	GAN for corn kernels [30]

Table 4.4: HSI analysis methods (ML/DL) with operational characteristics.

Table 4.5 illustrates the wide spectrum of applications reported in the literature, ranging from nutrient and water stress monitoring to crop/weed discrimination and pest detection. The availability of software tools (e.g., MATLAB, Python libraries, such as SPy and PlantCV, and embedded SDKs) is shown to play a decisive role, often requiring custom pipelines tailored to specific tasks. In sorting and quality control, embedded software is critical for enabling real-time implementation. The same software frameworks can be adapted across domains with different performance requirements—for instance, rapid decision-making in surgical guidance versus high-throughput processing in food sorting.

Task	**Spectral Range**	**Algorithms**	**Software Used**	**References**
Nutrients/Chlorophyll	400–1000 nm (VIS–NIR)	PLSR, RF, CNN	Proprietary SW (Specim/Headwall), MATLAB (Hyper-Tools)	[31, 22, 32]
Water stress	500–900 nm (RE, NIR)	VI, SVM, CNN	Python (SPy, PlantCV)	[33, 24]
Weed/crop discrimination	450–950 nm	2D/3D-CNN, SVM	Python (custom), UAV SDK	[14, 15, 16, 27]
Disease/pest detection	700–1700 nm	SVM, hybrid DL	Proprietary acquisition SW + MATLAB	[11, 29]
Sorting/product quality	660–980 nm (NIR snapshot)	Real-time CNN, embedded pipeline	Embedded SDK (Pleora eBUS, C/C++), Python	[25]
Food safety/quality (Heliyon)	500–2500 nm (NIR/SWIR)	PLS-DA, SVM, GLCM, ANN	MATLAB/Python; Proprietary SW	[10]
Medical imaging	VIS–NIR; FTIR (5882–6250 nm)	PCA, PLS-DA, CNN	Laboratory/hospital pipeline	[10]
Water/Flood monitoring	VIS–NIR (veg/-soil/water)	Indices, spectral signatures	GIS + HSI post-processing	[10]

Table 4.5: HSI applications in precision agriculture.

Table 4.6 emphasizes the trade-offs among computing platforms for managing hyperspectral datasets. GPUs remain the standard for training and inference of deep learning models, whereas FPGAs offer extremely low latency and high throughput in sorting and compression tasks, albeit at the expense of design complexity. Embedded systems (e.g., Jetson TX1/TX2/Xavier) are emerging solutions for UAVs and UGVs, enabling on-board processing but constrained by limited memory. Broader challenges are linked to costs and massive dataset management, where HPC clusters and dedicated storage infrastructures become necessary, particularly in cross-domain applications such as medical imaging or water quality monitoring.

HW	Applications	Advantages	Limitations	HW Examples/Performances
CPU	Offline analysis	Availability, simplicity	Slow on large hypercubes	–
GPU (desktop/ embedded)	DL training/inference; compression	+10–50× vs. CPU; DL & CUDA	Power/consumption	Tesla C1060: real-time unmixing [34]; Jetson TX1: efficient PCA (75 bands) [35]
FPGA	On-line sorting and compression	Very low latency; dedicated architectures	Complex development	SoC FPGA: 216× faster than 256-core GPU on 512 × 614 × 224 [36]; Artix-7: real-time MSVM [37]; VCA without pre-DR [38]
Embedded (Jetson TX2/Xavier)	In-field UAV/UGV	Portability; sensor integration	Limited memory	Jetson on DJI M600 + Specim FX10 for acquisition/recording [19, 20]

Table 4.6: Hardware and real-time implementation for HSI with examples/performances.

Beyond vineyard use, detailed in numerous recent studies [39, 40, 41, 42, 43, 44], ground-based HSI has been successfully applied to wheat, vegetables, stone fruit, citrus, and for quality assessment and defect detection in fresh produce, such as apples, potatoes, and tomatoes [45, 46]. For example, the review by Dale [45] and the overview provided by Benelli [46] report multiple sub-studies documenting the use of HSI for apple ripeness classification, tomato damage detection, and non-destructive evaluation of quality parameters in kiwis and bell peppers.

A wide range of predictive and classification approaches were employed in the studies reported in Table 4.7. Among regression techniques, Partial Least Squares Regression (PLSR) was by far the most commonly applied method, often combined with preprocessing or variable selection procedures. Other linear approaches such as multiple linear regression and optimized vegetation indices were also adopted, while non-linear methods, including convolutional neural networks (CNNs), support vector machines (SVMs), multilayer perceptrons (MLPs), and discriminant analysis, were implemented particularly for classification tasks, such as variety discrimination, disease detection, and weed recognition. With respect to predictive performance, most studies reported moderate to good accuracy, with determination coefficients (R^2) typically ranging from 0.70 to 0.88 for chlorophyll and nitrogen estimation, and classification accuracies above 90% in tasks such as fruit maturity assessment

and weed resistance detection. Even though environmental factors (e.g., illumination variability, shadows, canopy structure, or wind) often affected data quality, the majority of models achieved acceptable and reproducible results, demonstrating that ground-based hyperspectral imaging can reliably support both quantitative predictions and qualitative classifications in real field conditions.

In Table 4.7, some agricultural studies on-field using hyperspectral technology are listed. The only type of hardware-related information we aim to include, in order to provide a broader overview of these applications, is economic. The overall equipment required for hyperspectral image acquisition in the field comes at a significantly high cost. The aqcuisition system listed in Table 4.7 have an average cost of €20,000 and can reach up to €40,000 for the camera body alone, without lens/objective.

Application	**Crop**	**Acquisition System**	**Spectral Range**
Phenotyping	Wheat, legumes [47]	Resonon Pika II	390.9–887.4 nm, 244 bands
Phenotyping	Cotton [48]	Middleton MRC-923-001	400–1000 nm, 236 bands
Phenotyping	Vine [41]	ATV, Resonon Pika L VNIR	400–1000 nm, 300 bands
Fruit ripening	Mango [49]	Resonon Pika II VNIR	400–900 nm, 240 bands
Chlorophyll	Rice [50]	Cubert S185 imaging spectrometer	450–950 nm, 4 nm
Chlorophyll, Nitrogen	Wheat [51]	Redlake MS4100 VNIR	400–1000 nm (VNIR)
Chlorophyll	Sugar beet [52]	HySpex VNIR 1600	400–1000 nm
Nitrogen	Rice [53]	Specim QE V10E	400–1000 nm, 5 nm
Nitrogen	Wheat [54]	HySpex VNIR 1600-160	400–1000 nm, 3.7 nm, 160 bands
Fungal diseases	Wheat, barley [55]	Gilden Photonics	400–1000 nm
Fungal diseases	Wheat (yellow rust) [56]	Specim ImSpector V10E	400–1000 nm, 2.8 nm
Water stress	Corn, barley [57]	Specim PS V10E	400–1000 nm, 2.8 nm
Weeds	Corn [17]	Specim V9	435–834 nm, 19 nm
Weeds	Wheat [27]	Specim V10E	400–1000 nm, 2.8 nm
Weed resistance	Palmer amaranth, etc. [58]	Resonon Pika II	400–900 nm, 240 bands
No-till/conserv. agric.	Corn (stubble) [59]	Headwall Micro-Hyperspec	347–952 nm, 329 bands

Table 4.7: On-field ground based applications of hyperspectral imaging in the agri-food sector.

Despite its great potential, the use of HSI on-field entails several practical and technical challenges. Below, we analyze the most important ones, with a greater focus on those less related to the specific hardware in use. Table 4.8 summarizes the various issues investigated. The first three are mainly tied to the type of technology currently available in research centers and on the market. It is not impossible that

prices and/or technical specifications will improve in the coming years to match current RGB cameras. The remaining issues concern data acquisition and processing methods for raw field data. In what follows, each issue is separately analyzed.

Issue	Main Cause	Practical Effect	Solutions/Methodologies and References
Acquisition speed	Push-broom, line-by-line scanning	Limited coverage, long times for large areas	Use of portable spectrometers or "on-the-go" systems [43, 44]
Outdoor robustness	Vibrations, motion, wind	Blurring, distortion, registration errors	Anti-vibration supports, stable platforms, software correction [46, 44]
Variable lighting	Sun, clouds, shadows	Spectral inconsistency, poor data comparability	Calibration targets, normalization algorithms [45, 46, 43]
Spatial resolution	Hardware limits (portable HSI)	Less detailed images, difficult small-structure analysis	High-resolution systems, integration with other technologies [45, 43]
Complex calibration	Field instability, lack of standards	Systematic errors and error propagation	Standardized procedures, frequent calibration [46, 43]
Data management	Voluminous multi-band data	Storage, processing time, analysis difficulty	Data cleaning, compression, feature selection [60, 45, 39]

Table 4.8: Main practical issues of on-field HSI in agriculture, related causes, effects, and references (tabella spostata).

4.3.1 Limited Acquisition Speed

Current HSI systems are often characterized by low acquisition speeds, especially when using push-broom or line-scan spectroradiometers with internal moving parts in the camera to generate spectral bands, which acquire one line at a time [45, 40, 44]. This strongly limits the extent of monitoring campaigns, particularly when large areas need to be covered or when working with moving platforms. For instance, studies reported by [46] highlight that scanning a complete tree canopy may take several minutes, making large-scale use or use in high temporal variability contexts difficult. Limited acquisition speed is not only a technological bottleneck but also a major barrier for operational scalability. In UAV-based applications, line-scanning systems require precise synchronization with platform velocity, which increases operational complexity and makes long campaigns prone to incomplete or misaligned coverage [9]. However, the use of portable NIR spectrometers allows for faster data collection,

making monitoring more efficient, as shown by [43]. On-the-go HSI systems also enable data acquisition on entire vineyard rows while in motion, further reducing the operational time [44]. Beyond agriculture, similar constraints are observed in food quality inspection and medical imaging, where rapid decision-making is critical, and pushbroom systems are often replaced by snapshot cameras, despite their lower spectral fidelity [10]. Another limiting factor for acquisition speed, in addition to the intrinsic behavior of the hyperspectral camera, is illumination. Under field conditions, reduced light levels—caused, for example, by cloud cover—often require longer exposure times to achieve an acceptable signal. Even snapshot cameras, which are designed for rapid cube reconstruction, suffer from low light sensitivity and reduced signal-to-noise ratios, thus constraining their effective speed in outdoor deployments. Furthermore, when the target is located several meters away from the sensor, as in many ground-based applications, increasing the exposure time inevitably leads to motion blur. This issue is commonly mitigated by reducing the platform's travel speed, which in turn further decreases the overall acquisition efficiency.

4.3.2 Low Robustness in Outdoor Conditions

This lack of robustness is consistently reported across the literature as one of the key obstacles for field deployment. According to Ram et al [9], the combination of platform vibrations and atmospheric interference severely impacts the reliability of outdoor acquisitions, particularly for UAVs. For ground-based application, vibrations due to ground unevenness can compromise acquisition quality, causing blurring, distortions, and misalignment between spectral bands [46, 40]. For example, in vineyard campaigns, it has been observed that even minor oscillations in the mounts lead to registration errors, which are difficult to correct in post-processing. Some authors suggest using anti-vibration supports or active stabilization systems [46].

4.3.3 Low Spatial Resolution

Despite the high spectral detail, many portable HSI systems offer lower spatial resolution compared to conventional cameras [45, 43].

Spatial resolution represents a critical parameter in hyperspectral imaging, as it directly affects the ability to capture meaningful spatial and spectral information. Four main challenges are typically encountered: (i) the trade-off between pixel size and optical quality, where small pixels may be limited by diffraction or optical aberrations, while large pixels reduce spatial detail; (ii) the balance between resolution and noise, since operating near the resolution limit often increases noise, requiring strategies such as spatial or spectral binning; (iii) the need to match the spatial resolution with the size of the target particles or contaminants, as insufficient resolution produces mixed pixels that dilute relevant signals; and (iv) the influence of surface irregularities and geometry, which introduce reflections, shadowing, and other artifacts that degrade effective resolution. When these limitations prevent reliable pixel-wise analysis, many studies have resorted to using the ROI average spectrum [61]. This approach reduces noise, simplifies data structures, and enhances computational feasibility, making it particularly attractive in applied and industrial contexts. Over the

last five years, this trade-off has resulted in more than 80% of published studies adopting the ROI-based strategy, as it compensates for spatial resolution constraints and improves the robustness of classification models [8]. Nevertheless, this choice inevitably limits the exploitation of hyperspectral imaging's unique capacity to map spatial heterogeneity, thereby reducing its potential for detecting localized defects, adulterants, or contaminants.

4.3.4 HSI Data Management and Complexity

Another critical issue relates to the size and complexity of HSI data. Each acquisition generates large volumes of multi-band data, often tens or hundreds of MB per image, posing significant challenges in terms of storage, transfer, and processing time [60, 45, 39]. Embedded platforms such as Jetson boards struggle to process full hypercubes in real time, necessitating dimensionality reduction or band selection prior to modeling, highlighting that data management is often a greater challenge than data acquisition itself [9]. Das et al. [60] emphasize that efficient data management is one of the key aspects for operational deployment of HSI, suggesting the adoption of automated pipelines for data cleaning and compression. In some cases, the cited literature suggests that the use of machine learning techniques for feature selection and dimensionality reduction is essential for handling large datasets [62]. Moreover advanced processing techniques like parallel and distributed architectures, and high-performance computing can help to reduce computational and storage cost [63].

4.3.5 Lighting Control

Natural light, variable in intensity and direction, is one of the main sources of error in outdoor HSI imaging [9, 45, 39, 43]. The presence of clouds, shadows, reflections, or rapid light variations can alter the sensor's spectral response, making it difficult to compare repeated acquisitions. Various methods (e.g., use of reference targets for reflectance calibration or normalization algorithms) have been proposed in the literature, but standardizing conditions remains problematic [45, 46].

4.3.6 Radiometric Image Calibration

Radiometric calibration is an essential phase in the acquisition and analysis of hyperspectral and multispectral images. This refers to the set of procedures that transform raw digital values (digital numbers, DN) produced by a sensor into absolute radiometric measurements, such as reflectance or scene radiance [64, 65, 66], considering both sensor response and lighting conditions. In the case of hyperspectral and multispectral images, calibration serves two purposes: ensuring data comparability across different acquisitions, sensors, and lighting conditions; and enabling reliable quantitative information extraction, essential for biophysical monitoring, species or material identification, diagnostics, and automatic classification.

It requires stable, controlled conditions, which are difficult to maintain in the field [45, 46]. Calibration errors can propagate throughout the data-processing chain, significantly affecting the accuracy of qualitative and quantitative estimates.

Benelli et al. [46] underline that the lack of standardization in calibration procedures among research groups still limits widespread outdoor HSI use. The evolution of methods, from classic reference-based strategies to computational and AI-based solutions, has expanded operational capabilities, allowing for use across a wide range of scenarios. However, the choice of calibration procedure must be carefully adapted to operating conditions, sensor type, and application objectives, always considering the challenges and limitations reported in the recent literature.

The most widespread method, using physical references (on-field and laboratory-based), involves reference panels with known reflectance (Spectralon or similar materials), acquired before, during, and after the acquisition session [64, 67, 68, 69, 66]. This strategy can be applied both in the lab and directly in the field and includes the following steps: acquisition of panel images under the same operating conditions as the scene; estimation of the sensor's radiometric response function for each band, typically through linear regression between panel values and their certified reflectances [64]; application of the calibration curve to all operational images to obtain radiometrically corrected data [64, 67, 69].

Panel choice (pure white, gray, black, multiple) and operational handling strongly affect calibration quality and final measurement accuracy [67, 69].

Calibration Based on Illumination Estimation and Compensation

Recently, techniques have spread that estimate the spectral distribution of the light source directly from images, then compensate its effects on the data [70, 71, 72, 73, 74]. These methods include using reference patches in the scene to extract the SPD (Spectral Power Distribution) of the light [70, 75]; applying color constancy algorithms adapted to multispectral data, such as gray world, max-RGB, shades of gray, or gray edge [72, 73]; and finally using statistical or machine learning (deep learning) methods for automatic, per-pixel estimation of the illuminant [74].

These approaches are particularly useful where physical references cannot be included, or in dynamic scenes and complex operational environments [70, 74, 71].

Calibration for Non-Conventional or Consumer Instruments

The spread of consumer digital cameras has stimulated the development of calibration procedures adapted to RGB sensors or non-scientific devices, especially under low-light conditions [66, 76]. Adopted methods include using reference panels, grayscale charts, and careful management of acquisition parameters (ISO, exposure time, dark correction) [66] and spectral calibration strategies using diffraction filters and simplified physical models [76]. The analyzed literature highlights several recurring issues in the radiometric calibration of multispectral/hyperspectral images: variations in natural light [68, 67, 75, 64]; errors due to panel misplacement, shadowing, surface contamination, or non-optimal angles [67, 64]; non-linear sensor response and temporal drift [66, 69, 77]; absence of physical references [72, 73, 71]; multiple or non-uniform illumination effects [74, 75, 78]; and application requirements and sector-specific constraints [78, 79, 77].

In Table 4.9, several of the radiometric calibration methods previously discussed are summarized. Particular attention is given to the last two columns, which concern

the practical aspects of their implementation. In many cases, the lack of knowledge of the sensor spectral response represents a major limitation, especially when non-scientific or consumer-grade cameras are employed. Conversely, methods requiring a reference target in the field may complicate certain in situ applications, where the deployment of calibration panels is not always feasible.

Calibration Type	**Advantages**	**Main Issues**	**Response Required?**	**Known Reference Required?**
On-field with reference panels [64, 67, 68, 69, 66]	High accuracy; compensates light variation; simple	Sensitive to placement errors, shadows, contamination, light variability	No	Yes
Laboratory-based [77, 69]	High repeatability, controlled conditions	Not representative of real world; needs updates	Yes	Yes
Illumination compensation [70, 72, 73, 74, 71, 75]	Applicable without references, flexible, automatable	More uncertain in heterogeneous scenes, noise-sensitive	Yes	No
Multi-reference (multiple panels) [67, 64, 66]	Better response function, reduces saturation	More targets, more complexity	No	Yes
Light source calibration [78, 79]	Essential for non-standard light sources	Extra measurements, may require advanced tools	Yes	No
Low-cost/consumer spectral calibration [66, 76]	Affordable, accessible	More noise-sensitive, drift, non-linearity	Yes	Yes

Table 4.9: Summary of radiometric calibration methods for hyperspectral and multispectral images.

4.4 Spectral Super-Resolution: Methods, Challenges, and Perspectives

Spectral Super-Resolution (SSR), that is, the reconstruction of hyperspectral images from RGB images, is a central topic in modern computer vision, with applications ranging from precision agriculture to medical diagnostics and cultural heritage preservation.

Nevertheless, it is important to emphasize that spectral reconstruction only replaces the traditional acquisition stage: once the hyperspectral cube is reconstructed, all subsequent steps—such as data preprocessing, dimensionality reduction, and the development of predictive models—remain essentially the same as in the traditional HSI pipeline, along with their inherent difficulties.

A phenomenon that has a direct implications on spectral reconstruction is metamerism. Metamerism arises from the fact that imaging devices reduce the continuous spectral information of light to a small number of channels (red, green, and blue). As a result, two different spectral reflectance functions may produce identical RGB values and thus appear as the same color, despite their underlying physical differences. This phenomenon highlights the inherent information loss in color imaging systems, where a high-dimensional spectral signal is projected onto a low-dimensional color space. Formally, the response R_c of a given channel $c \in \{R, G, B\}$ can be expressed as a weighted integral over the visible spectrum:

$$R_c = \int_\lambda S(\lambda)\, I(\lambda)\, Q_c(\lambda)\, d\lambda, \tag{4.1}$$

where $S(\lambda)$ denotes the surface spectral reflectance, $I(\lambda)$ the spectral power distribution of the illuminant, and $Q_c(\lambda)$ the spectral sensitivity function of the channel. Two distinct reflectance spectra, $S_1(\lambda)$ and $S_2(\lambda)$, are said to be metameric under a given illuminant if they produce identical responses in all channels:

$$\int_\lambda S_1(\lambda)\, I(\lambda)\, Q_c(\lambda)\, d\lambda \;=\; \int_\lambda S_2(\lambda)\, I(\lambda)\, Q_c(\lambda)\, d\lambda \quad \forall c \in \{R, G, B\}. \tag{4.2}$$

In this case, although $S_1(\lambda) \neq S_2(\lambda)$, the recorded color will be indistinguishable, which represents the essence of the metamerism problem. Recovering the full incident spectrum $S(\lambda)$ from only three RGB responses $\{R, G, B\}$ is intrinsically an *inverse ill-posed problem*. Since infinitely many different spectra can lead to the same set of RGB values, the mapping from RGB space back to the hyperspectral domain is not unique.

Despite this problem, the recent literature highlights the significant progress enabled by statistical models and, especially, deep learning approaches [1, 3].

Regression [80], one of the earliest techniques, has become popular because of its straightforward, fast, accurate, and closed-form solution. RGB and their spectral estimations are related in the most basic "linear regression" [80] by a single linear transformation matrix. Moreover, polynomial and root-polynomial regression [81, 82] expand the RGB into polynomial/root-polynomial terms, which are subsequently transferred to spectra using a linear transform, in order to add non-linearity. Regressions that minimize the mean squared error (MSE) in the

training set are sometimes referred to as "least-squares" regressions. Li et al. [83] proposed a locally linear embedded sparse coding approach for spectral reconstruction from RGB images, achieving competitive results on various benchmarks. Similarly, the A+ method based on Adjusted Anchored Neighborhood Regression, developed by Timofte et al. [84], is a reference in traditional super-resolution and is often used as a baseline in SSR applications.

Several advanced architectures have been proposed in recent years, such as HSCNN+ [85] reported in Figure 4.2, attention-based models, adversarial networks [86, 87], and techniques that exploit prior knowledge (e.g., the camera's spectral response) [2, 5]. In particular, Yan et al. [5] suggest to use informative priors on material categories, meta-learning, and dimensionality reduction techniques to improve generalization and model efficiency.

The New Trends in Image Restoration and Enhancement (NTIRE) challenges, held annually alongside the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), provide standardized benchmarks and competitive venues for low-level vision problems. They foster rapid progress by releasing carefully curated datasets and clear evaluation protocols, enabling fair, reproducible comparisons among state-of-the-art architectures in spectral reconstruction.

As shown in Table 4.10, most of the submitted solutions rely on deep learning architectures, predominantly convolutional and transformer-based models, whereas approaches grounded in sparse recovery are almost entirely absent from the recent NTIRE challenges. In the earliest editions (e.g., NTIRE 2018), convolutional neural networks rapidly supplanted traditional sparse coding or regression-based techniques, which had previously been considered effective for spectral recovery. From NTIRE 2020 onwards, virtually all competitive entries relied on increasingly deeper CNNs, often incorporating residual or dense connections to improve stability and representation capacity. More recent challenges (e.g., NTIRE 2022) show a further shift towards transformer-based designs, such as MST++ and hybrid CNN–Transformer frameworks, reflecting the field's convergence with broader trends in image restoration. Another noticeable trend in recent NTIRE challenges is the increasing integration of architectures that are sensitive not only to spectral dependencies but also to spatial features. Transformer-based models and hybrid CNN–attention frameworks explicitly leverage spatial context to improve reconstruction fidelity, marking a shift from purely spectral, pixel-wise mappings to more context-aware designs. Notably, dictionary learning and sparse recovery approaches are almost entirely absent from these recent competitions, confirming that they no longer represent competitive baselines when large-scale datasets and efficient GPU implementations enable the training of complex neural models. This progression underscores the decisive role of deep learning in setting the state of the art for spectral reconstruction.

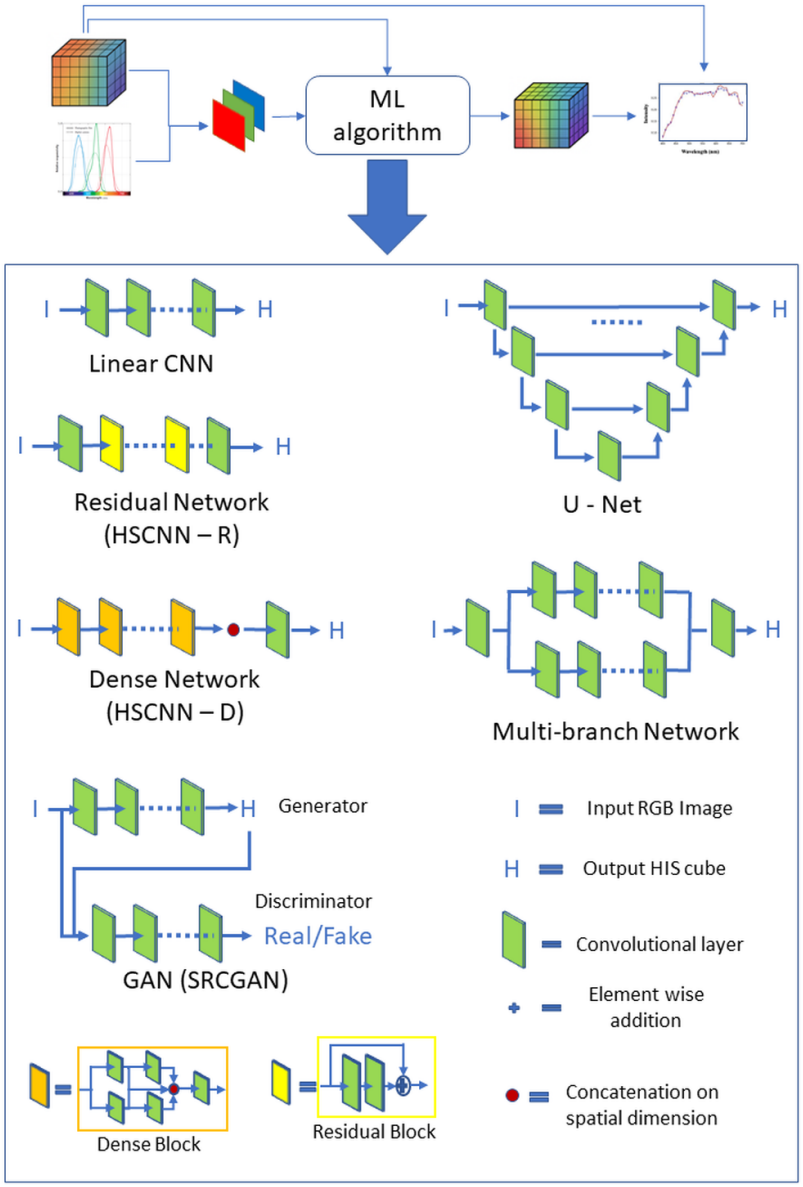


Figure 4.2: Spectral reconstruction workflow using deep learning.

Article	Representative Architecture Types/Examples
Arad et al., NTIRE 2018 [88]	CNN-based residual networks (e.g., HSCNN+); U-Net-like encoder–decoder models with skip connections; GAN-based solutions explored in some entries (e.g., IVRL Prime).
Arad et al., NTIRE 2020 [6]	Deep residual CNNs with multi-branch designs; ensemble models combining multiple residual backbones; architectures exploiting U-Net-like skip connections for multi-scale recovery.
Arad et al., NTIRE 2022 [7]	Transformer-based networks (MST++ with spectral-wise attention); residual learning frameworks (DRCR Net, RDAN); multi-branch and multi-feature fusion strategies; attention mechanisms (spectral, spatial, or dual).

Table 4.10: Architectures reported in the NTIRE challenges under consideration (representative/ backbone families used by participants).

The summary reported in Table 4.11 shows not only the technical diversity of the methods submitted to NTIRE competitions but also the gradual evolution of the benchmarking protocols. A distinctive element of these challenges is the division into tracks, each designed to emulate different acquisition conditions or levels of task difficulty. For instance, the NTIRE 2018 and 2020 spectral recovery competitions defined two parallel tracks: the Clean track, which considered RGB images generated from hyperspectral data through a known and noise-free camera response function, and the real-world track, which simulated the practical scenario of JPEG-compressed RGB images obtained with an unknown response function. This dual setup provided insight into both the theoretical upper bound of algorithmic performance and the more challenging case of uncontrolled acquisition. In the NTIRE 2022 spectral recovery challenge, the organizers opted for a single track with a more realistic camera simulation pipeline, which incorporated automatic exposure, sensor noise, basic in-camera processing, and compression. This choice reflects a shift in focus from idealized conditions to scenarios that better approximate real-world usage. Overall, the design of tracks across NTIRE competitions illustrates the organizers' intention to balance scientific rigor with practical relevance. By progressively moving from clean synthetic scenarios to complex, noise-affected, and compression-degraded data, the NTIRE challenges have established themselves as a benchmark series that both advances methodological innovation and pushes models closer to real-world applicability.

Study/Team	Technique	HW/SW	Ex. Time	Accuracy
VIDAR2	Dense CNN ensemble	GTX 1080Ti, PyTorch	0.96 s inference	MRAE ∼0.016 [88]
HypedPhoti	CNN (GAN variants)	GTX 1080Ti, Keras/Tensorflow	0.57 s inference	MRAE ∼0.017
IVRL Prime	Residual CNN + GAN	Titan X, Tensorflow	1.5 s (GPU)/2.5 m (CPU)	MRAE ∼0.015
IPIC SSR	CNN + ensemble	2× RTX 2080Ti, PyTorch	36 h training, 0.56 s inference	MRAE 0.030 (clean)/0.062 (real) [6]
MDISL-lab	CNN ensemble (10 models)	GTX 1080Ti, PyTorch	48 h training, 16 s inference	MRAE 0.031/0.062
OrangeCat	CNN ensemble (8 settings)	2× Titan Xp, PyTorch	7 d training 3.7 s inference	MRAE 0.032/0.062
MST++	Transformer (MST++)	RTX 3090, PyTorch	40 h train 0.10 s inference	MRAE 0.1131, RMSE 0.0231
MIALGO	Enhanced HAN	8× Tesla V100, PyTorch	5 d train 0.43 s inference	MRAE 0.1247, RMSE 0.0257
CVIA SSR	DRCR Net	RTX 3090, PyTorch	24 h train 0.15 s inference	MRAE 0.1766, RMSE 0.0322
IVL	Fast-n-Squeeze	GTX 1080, PyTorch/MATLAB	30 min train 0.01 s inference	MRAE 0.2915, RMSE 0.0541

Table 4.11: Summary of reported methods, hardware/software configurations, and performance in NTIRE challenges.

Table 4.12 summarizes both the official NTIRE datasets and the most widely used public hyperspectral benchmarks. A clear distinction can be observed between the NTIRE collections and traditional datasets. The NTIRE datasets (BGU HS, ARAD 1K, and LDV 2.0) were specifically curated to support competitive benchmarking, providing standardized train/validation/test splits and evaluation protocols. They are generally smaller in spatial resolution (e.g., ARAD 1K at 482×512 px) but offer carefully controlled acquisition pipelines, such as realistic camera simulations and JPEG compression, which are crucial for fair comparison across methods. In contrast, public hyperspectral datasets such as ICVL, CAVE, and Harvard remain indispensable for algorithm development due to their higher spatial resolution and scene diversity, though they lack standardized challenge settings. Remote sensing datasets (e.g., Chikusei, Houston, Pavia, Washington DC Mall, Botswana, and Cuprite) typically consist of large single-scene acquisitions with high spectral dimensionality, providing opportunities to test scalability but posing challenges for generalization. More recent UAV-based benchmarks, such as WHU-Hi, reflect a growing interest in domain-specific applications, particularly agriculture and crop monitoring. Overall, while NTIRE datasets drive methodological innovation through controlled and competitive benchmarking, the broader public datasets remain essential for testing robustness and adaptability of spectral reconstruction

methods across varied imaging conditions and application domains. Many public datasets used for training and testing models do not include standardized calibration procedures, further complicating the generalization of results [1, 4]. Chen et al. [89] highlight that deep networks achieve excellent results on benchmarks, but also the challenge of maintaining high performance on unseen data or under different acquisition conditions.

Dataset	Source	Size (W × H)	# Bands/Range	# Images	Notes
BGU HS (NTIRE 2018) [88]	Ben-Gurion Univ.	∼1392 × 1300	31 (400–700 nm)	256	Used in NTIRE 2018 (Clean/Real-World tracks).
BGU ext. (NTIRE 2020) [6]	Ben-Gurion Univ.	∼1392 × 1300	31 (400–700 nm)	450+	Camera simulation with JPEG compression and noise.
ARAD 1K (NTIRE 2022) [7]	Specim IQ camera	482 × 512	31 (400–700 nm)	1000	Large natural HSI dataset for spectral recovery.
KAUST-HS (NTIRE 2020) [74]	King Abdullah Univ. of Science and Technology	512 × 512 (typ.)	31 (400–700 nm)	∼20 scenes	Provides calibrated hyperspectral images with white reference panels, enabling direct reflectance extraction (unique among public datasets).
ICVL [90]	Ben-Gurion Univ.	1392 × 1300	31 (400–700 nm)	200	Indoor/outdoor natural scenes.
CAVE [91]	Columbia Univ.	512 × 512	31 (400–700 nm)	32	Studio scenes (toys, food, fabrics).
Harvard [92]	Harvard Vision Lab	1040 × 1392	31 (400–700 nm)	50	Natural scenes, indoor/outdoor.
Chikusei [93]	Univ. of Tokyo (UAV)	2517 × 2335	128 (363–1018 nm)	1 large image	Agricultural and urban area (UAV-borne).
Houston [94]	Univ. of Houston (GRSS)	349 × 1905	144 (380–1050 nm)	1	Airborne urban/-land cover.
Pavia Centre [95]	Univ. of Pavia	1096 × 715	102 (430–830 nm)	1	ROSIS sensor, urban area.
Pavia Univ. [95]	Univ. of Pavia	610 × 340	103 (430–830 nm)	1	ROSIS sensor, university campus.
Washington DC Mall [96]	HYDICE sensor	1208 × 307	191 (400–2400 nm)	1	Benchmark airborne scene.
Botswana [97]	NASA EO-1 Hyperion	1496 × 256	145 (400–2500 nm)	1	Okavango Delta, 30m resolution.
Cuprite [98]	NASA/JPL AVIRIS	512 × 614	224 (370–2480 nm)	1	Mineral mapping benchmark.
WHU-Hi (HanChuan) [99]	Wuhan Univ. (UAV)	1200 × 512 (typ.)	31 (VIS–NIR)	Multiple	UAV dataset for crop monitoring/classification.

Table 4.12: Public and NTIRE hyperspectral datasets commonly used in spectral recovery and super-resolution.

It is important to note that the NTIRE challenges are not specifically oriented toward agri-food applications. Rather, they are conceived as general-purpose benchmarks for low-level vision tasks, such as spectral recovery, image restoration, and super-resolution, using standardized datasets and evaluation protocols. While the methodological advances achieved in NTIRE competitions are of potential relevance for agri-food imaging—particularly in terms of spectral fidelity and reconstruction accuracy—their datasets are typically composed of generic natural scenes or video content. Consequently, transferring these approaches to agri-food requires additional validation under domain-specific conditions, including outdoor acquisition variability, crop heterogeneity, and application-driven performance metrics.

In this regard, it is also important to acknowledge recent contributions that go beyond the NTIRE framework and explicitly address some of the physical shortcomings of current spectral reconstruction methods. Lin and Finlayson [81] demonstrated that most deep learning architectures, while highly accurate on fixed benchmark datasets, are strongly dependent on exposure conditions. Their analysis revealed that leading CNN-based models fail to generalize when illumination or camera settings vary, a scenario that is unavoidable in field acquisitions. To mitigate this limitation, they revisited regression-based approaches and introduced root-polynomial regression, a model that is inherently exposure-invariant. Although simpler than deep neural networks, this method was shown to maintain stable performance across a range of exposure conditions, thus highlighting the importance of robustness over architectural complexity. Building on this line of work, Lin and Finlayson [4] proposed a physically plausible formulation of spectral reconstruction, directly addressing another fundamental limitation of existing methods. In conventional pipelines, reconstructed hyperspectral signals often fail to reproduce the RGB values that were originally captured, meaning that the mapping from RGB to spectra is not physically consistent with the image formation process. To resolve this, the authors decompose each spectrum into two components: a *fundamental metamer*, which is uniquely determined by the measured RGB values, and a *metameric black*, which represents the residual spectral degrees of freedom invisible to the camera. This guarantees that reconstructed spectra always integrate back to the exact RGB measurements, ensuring zero colorimetric error while still allowing for learning-based methods to predict the hidden spectral variation. Beyond improved spectral fidelity, this approach enhances robustness to changes in illumination and camera response, thereby aligning reconstruction with the physical constraints of image capture. Taken together, these studies suggest that for application domains, such as agri-food, where imaging is frequently conducted in uncontrolled outdoor conditions, advances in exposure invariance and physical plausibility may be as critical as improvements in benchmark accuracy. Consequently, progress in this field cannot rely solely on NTIRE-driven deep learning pipelines, but must also integrate physics-based constraints and robustness criteria to ensure reliable deployment in real-world agricultural monitoring and food quality assessment. A critical aspect concerns model portability: the output of RGB cameras heavily depends on the sensor's spectral response, calibration, white balance, and compression processes [3, 100, 4]. As noted by Koundinya et al. [100], models trained on data from a specific camera may degrade significantly when applied to data acquired with different devices. Normalization using a white reference and

the incorporation of physical constraints on the plausibility of the reconstructed response are promising strategies. Overall, spectral super-resolution remains an open challenge: excellent results have been achieved in controlled conditions and on standard datasets, but operational application requires robust pipelines for calibration, validation, and critical selection of data and models [1, 80]. All these factors make reconstruction highly dependent on the dataset, the camera, the resolution, the FOV, and the camera sensor, among others [1, 4, 81]. The validity of an SSR model on cameras/datasets different from those used in training is often limited. In particular, the SSR model depends on the spectral response function of the specific camera used, in addition to the nature of the scene and the acquisition methodology. In datasets acquired in reflectance (where the response is normalized to a white standard), this dependence is mitigated, but considering the RGB response of each system remains fundamental [4].

The main challenges of spectral super-resolution concern the need for large and heterogeneous datasets for effective training, the limited physical interpretability of deep learning models, and the difficulty of ensuring generalization across different cameras and datasets. In summary, spectral super-resolution represents a promising pathway to broaden spectral analysis applications in real-world scenarios; however, it remains an open scientific and technological problem with several hurdles still to be addressed. The following section examines recent applications of spectral reconstruction and super-resolution, both in controlled laboratory settings and under uncontrolled field conditions.

4.5 Regression Models from RGB via Spectral Reconstruction

HSI is a highly powerful tool for non-destructive characterization of food products and biological systems. However, its widespread application has been hindered by the high cost of equipment, the complexity of measurement procedures, and the low portability of traditional systems. In this context, spectral reconstruction (SR)—the ability to obtain hyperspectral data from simple RGB images captured with consumer cameras or smartphones—has recently emerged as a promising strategy to enable new forms of low-cost, highly accessible quantitative analysis. Through deep learning models trained to predict spectral information (or selected bands) from RGB photographs, it is theoretically possible to estimate chemical–physical and qualitative parameters directly on real products, both in the lab and potentially under field conditions.

Despite this theoretical potential, practical applications of spectral reconstruction for building quantitative regression models in the agri-food domain are still rare. Most studies focus on validating pipelines in laboratory settings as can be seen in Table 4.13, and only a few report real-world or in-field applications (Table 4.14). The main challenges include the standardization of RGB image acquisition and calibration procedures, model transferability across different matrices and lighting conditions, the management of instrumental and environmental variability, and the robustness of models when applied to real rather than simulated RGB images.

Application	Method	Range (nm)	Parameter	Performance	Reference
Tomato	HSCNN-R	400–1000	SSC	$R^2 = 0.51$	[101]
Tomato	HSCNN-R	400–1000	TTA	$R^2 = 0.61$	[101]
Tomato	HSCNN-R	400–1000	Lycopene	$R^2 = 0.92$	[101]
Rice	HR-ResNet	400–1000	Profilo spettrale	RMSE = 1.20, MAE = 0.61	[102]
Sweet potato	HSCNN-D	400–1000	SSC	$R^2 = 0.69$, RMSEP = 0.91	[103]
Sweet potato	HSCNN-D	400–1000	DMC	$R^2 = 0.84$, RMSEP = 2.74	[104]
Sweet potato	HRNet	400–1000	DMC	$R^2 = 0.88$, RMSEP = 2.35	[104]
Sweet potato	MST++	400–1000	DMC	$R^2 = 0.65$, RMSEP = 4.04	[104]
Egg	HRNet	400–1000	Embryo mortality classification	Acc. = 81.15%	[105]
Egg	Restormer	400–1000	Embryo mortality classification	Acc. = 80.64%	[105]
Beef	MST++	400–1000	Glutamic acid index	$R^2 = 0.84$, RMSEP = 0.17	[106]

Table 4.13: Applications of hyperspectral image reconstruction in laboratory settings.

Application	Method	Range (nm)	Parameter	Performance	Reference
Maize	HSCNN+	400–700	Pest disease	Precision = 94.27%	[107]
Rice	MST++ (pre-trained)	400–700	SPAD (chlorophyll)	$R^2 = 0.40$	[108]
Rice	MST++ (pre-trained)	400–700	Bending force	$R^2 = 0.37$	[108]
Rice	MST++ (pre-trained)	400–700	Plant height	$R^2 = 0.71$	[108]
Rice	MST++ (pre-trained)	400–700	DMC	$R^2 = 0.38$	[108]

Table 4.14: Applications of hyperspectral image reconstruction in field conditions.

This section presents a critical discussion of some of the few existing examples in the literature on the construction and validation of quantitative regression models from RGB images via spectral reconstruction, applied to fruits and food products.

4.5.1 Tomato Quality Assessment

One of the earliest examples of laboratory-based spectral reconstruction applied to agri-food products is reported by Zhao et al. [101]. The authors developed and tested a deep learning pipeline for reconstructing hyperspectral images from single RGB images of tomatoes, aiming to rapidly and non-destructively predict several key

quality parameters, including soluble solids content, titratable acidity, sugar-to-acid ratio, and anthocyanin/lycopene index. The study used tomato samples at different ripeness stages, which were analyzed using both hyperspectral imaging and RGB capture—partially simulated and partially real. RGB images were either simulated from hyperspectral reflectance data using standard illumination (CIE Standard illuminant D65) and camera response models (CIE 1931 Standard Observer with gamma correctio 1.4) or acquired directly by smartphone (Samsung Galaxy S9+). The reference parameters were measured through laboratory analyses. The reconstruction approach was based on a residual convolutional neural network (HSCNN-R), trained in a supervised manner with RGB inputs and hyperspectral outputs. Training was performed using only simulated RGB images. This ensured pixel-wise alignment between RGB and hyperspectral data. This network reconstructs a 31-channel spectra from 400 to 1000 nm with an RMSE of 6.22, MRAE of 0.05, and SAM of 1.68. The trained model was then validated both on simulated and real RGB images, in order to evaluate its robustness and transferability. The results showed that HSCNN-R was able to reconstruct hyperspectral pixel spectra with good accuracy—especially in the visible range—and enabled the estimation of quality parameters with R^2 values ranging from 0.5 to 0.9 (Table 4.13). The pipeline combined automatic fruit segmentation with regression on reconstructed spectra. However, one major methodological limitation was that training relied on perfectly standardized RGB simulations, making the model highly effective—but only under identical conditions to those used for training. When applied to real RGB images (captured via smartphone, with different lighting, sensor response, noise, and compression), the accuracy of both spectral reconstruction and quantitative predictions tended to decrease, although some robustness was still observed. Notably, the pipeline did not include explicit calibration of real RGB images using color charts or white references, limiting the model's transferability under variable conditions. The network basically links reflectance HSI to simulated and uncalibrated RGB images. The authors themselves highlighted that the model trained on tomatoes under lab conditions cannot be directly applied to other products, matrices, or lighting setups without retraining or including more diverse data in the training set. This study represents a milestone in validating spectral reconstruction for fresh product analysis in the lab. It demonstrates that it is possible to obtain reliable, non-destructive estimates of quality parameters from simple RGB images.

4.5.2 Sweet Potato Quality Evaluation

A relevant example of spectral reconstruction applied to quantitative regression in agri-food is the work by Ahmed et al. [103], which focused on sweet potatoes. This study showed how an advanced deep learning pipeline can reconstruct hyperspectral data from simulated RGB images, aiming to predict the soluble solids content (SSC), a key industrial and nutritional quality parameter for this crop. The authors used 141 sweet potatoes of three cultivars, acquired in a laboratory setting. Each sample was imaged using a Specim IQ hyperspectral camera, with two acquisitions at different angles under 750 W tungsten halogen lamps simmetrically positioned ensured even field-of-view illumination. RGB images were then simulated from hyperspectral reflectance data using a standard rendering process and later converted

to JPEG format for model input. The pipeline included careful segmentation to isolate the sweet potato from the background based on spectral differences. Only pixels from the region of interest (ROI) were used in both the HSI and RGB datasets. Spectra from the ROI were preprocessed using methods like Multiplicative Scatter Correction (MSC), Standard Normal Variate (SNV), and Savitzky–Golay smoothing. Informative bands were selected using a genetic algorithm to optimize SSC prediction. For training, each sample consisted of a pair: a simulated RGB image (not calibrated in absolute reflectance) and its corresponding HSI cube (reduced to selected bands and limited to the ROI). No real RGB images were used. The model employed was HSCNN-D, a dense hyperspectral convolutional neural network, with an RMSE of 0.05, MRAE of 0.86, and PSNR of 26 dB (Table 4.15). The model achieved more faithful spectral reconstructions than the previous study, particularly in the visible range, thanks to a deeper network (HSCNN-D versus HSCNN-R) at the cost of a higher computational load.

Metric	**Formula**	**Articles**
RMSE (Root Mean Squared Error)	$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$	(Zhao et al., 2020) [101]; (Ahmed et al., 2024) [103]; (Dong et al., 2025) [106]; (Zhao et al., 2022) [109] ; (Fu et al., 2022) [107]
MRAE (Mean Relative Absolute Error)	$MRAE = \frac{1}{n}\sum_{i=1}^{n}\frac{\lvert y_i - \hat{y}_i\rvert}{y_i}$	(Zhao et al., 2020) [101]; (Ahmed et al., 2024) [103]; (Dong et al., 2025) [106]; (Fu et al., 2022) [107]
SAM (Spectral Angle Mapper)	$SAM = \arccos\left(\frac{\mathbf{y}\cdot\hat{\mathbf{y}}}{\lVert\mathbf{y}\rVert\lVert\hat{\mathbf{y}}\rVert}\right)$	(Zhao et al., 2020) [101]
PSNR (Peak Signal-to-Noise Ratio)	$PSNR = 10\log_{10}\left(\frac{MAX^2}{MSE}\right)$	(Ahmed et al., 2024) [103]; (Dong et al., 2025) [106]
MAE (Mean Absolute Error)	$MAE = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$	(Zhao et al., 2022)[109]

Table 4.15: Evaluation metrics used in spectral reconstruction studies. In the formulas: i = generic pixel, y_i = ground truth value, $\hat{y}_i$ = predicted value, n = number of samples, $\mathbf{y}$ and $\hat{\mathbf{y}}$ = spectral vectors, MAX = maximum signal value.

As depicted in the study, traditional HSI steps follow the spectral reconstruction one in order to obtain a predictive model. Quantitative regression using Partial Least Square Regression (PLSR) on reconstructed data yielded a performance comparable to—and, in some cases, better than—that obtained using full-band real HSI, especially when only the selected informative bands were used. Feature selection helped reduce collinearity and noise compared to full-spectrum analysis.

This work extends spectral reconstruction to a complex matrix like sweet potato, showing that a segmented, feature-driven deep learning pipeline can deliver accurate regression models from simulated RGB inputs. However, the absence of real RGB data and field-calibrated imaging protocols remains a key challenge for practical deployment in production environments.

4.5.3 Glutamate Estimation in Beef

A recent study by Dong et al. [106] provides an advanced benchmark for spectral reconstruction applied to animal-based products. The authors explored the feasibility of reconstructing hyperspectral data from RGB images to predict glutamic acid content in beef, comparing the performance of eleven state-of-the-art deep learning models including Multi-stage Spectral-wise Transformer (MST) and MST++ [110], trained on simulated RGB-HSI pairs with separate subsets for training/testing and regression calibration/validation.. The dataset included 360 beef samples from three cuts and four regions in China. Hyperspectral images were acquired in the visible-NIR range using a 125-band system, and corresponding RGB images were captured using the same device. The pipeline involved white/dark correction for the HSI data and feature selection via competitive adaptive reweighted sampling (CARS), reducing the dataset to 31 informative bands. Regions of interest were segmented to isolate meat pixels, and multivariate analysis included correlation assessments and t-SNE visualizations. All deep learning models delivered strong results. MST++ and MST showed the highest consistency with real HSI data, with an RMSE of 0.02, MRAE of 0.01, and PSNR of 36.97 dB, with respect to the previous studies. For the regression task, the MST++ model performed nearly as well on the reconstructed HSI as on the real HSI, achieving a prediction R^2 of 0.842 vs. 0.853, respectively. In contrast, RGB images alone yielded a significantly lower accuracy, with an R^2 around 0.60. This study demonstrates the robustness of cutting-edge spectral reconstruction networks for quantitative prediction in meat products.

4.5.4 Early Detection of Maize Diseases

The study by Fu et al. [107] proposes a low-cost pipeline for early maize disease detection under field conditions, using hyperspectral data reconstructed from RGB images acquired in real operational settings. The dataset was collected at Jilin University (China) using the Xianyu 335 maize variety. Images were acquired with a Specim IQ camera (204 bands, 400–1000 nm, 512 × 512 pixels), from a distance of 1–1.5 meters. Each image contained both healthy and diseased areas, with pixel-wise annotations. While hyperspectral images were calibrated using a white reference panel (99% reflectance), RGB images—captured simultaneously and perfectly aligned—were not calibrated in reflectance. Image patches of 128 × 128 pixels were extracted and augmented through flipping and rotation. The HSI cubes were reduced to 31 bands spanning 400–700 nm. Spectral recovery was performed using an HSCNN+ architecture, a deep CNN with dense blocks, trained to reconstruct hyperspectral outputs from RGB inputs. The network was trained on 90% of the data and tested on the remaining 10%. A second CNN was used to classify disease status at the pixel level (healthy, infected, other), applied to both real and reconstructed HSI data. The results showed that HSCNN+ outperformed competing models like MST++, MIRNet, and HRNet, achieving an MRAE of 0.0713 and RMSE of 0.1204, with lowest errors in the central bands and higher errors at spectral extremes. In real field scenarios, disease detection based on reconstructed HSI outperformed pure RGB-based detection by 6–7 percentage points in overall accuracy, particularly in complex cases. The performance on the reconstructed HSI was very close to that on

the real HSI, and robustness was validated via a five-fold cross-validation.

4.5.5 UAV-Based In-Field Phenotyping

A recent study by Zhao et al. [109] presents one of the first applications of spectral reconstruction in open-field conditions for high-throughput phenotyping using UAVs. The goal was to reconstruct multispectral images from simple RGB images acquired by commercial drones to evaluate vegetation indices and crop parameters across large areas in a fast and cost-effective manner. The experiments were conducted on maize fields comprising 216 plots across three German sites, with image acquisitions on five dates during the growing season. UAVs equipped with both consumer-grade RGB cameras (DJI Phantom 4 Pro) and multispectral sensors (MicaSense RedEdge MX) were flown simultaneously. While the multispectral images were calibrated using reflectance panels, RGB images were not calibrated in absolute reflectance but were geometrically aligned with the reference images. The spectral reconstruction model was based on a modified DeepLabv3+ architecture, with RGB inputs of 256 $\times$ 256 pixels and outputs consisting of five multispectral bands (Blue, Green, Red, RedEdge, and NIR). The model was trained and validated using supervised learning with data augmentation and tested across sites using a cross-site validation strategy. The results were promising: RMSE values for band reconstruction ranged from 0.025 to 0.034, and pixel-level R^2 values were between 0.93 and 0.97. The NIR band proved the most difficult to reconstruct. Vegetation indices such as NDVI, GNDVI, and NDRE calculated from reconstructed data showed excellent agreement with reference data, with errors typically below 5%. This study confirms that UAV-RGB spectral reconstruction enables multispectral mapping and phenotyping in the field with an accuracy comparable to dedicated multispectral sensors. The approach offers an accessible solution for both research and precision agriculture, although future work must address absolute calibration and robustness under more extreme conditions.

4.6 Discussion

The first limitation observed in the analyzed studies concerns the very nature of the RGB inputs: rather than being acquired in parallel with hyperspectral data, RGB images are often mathematically derived from the hyperspectral cube. This raises questions about the realism of the experimental setup, as the neural networks are effectively trained on radiometrically uncalibrated inputs, without the effects of diverse camera spectral responses, and image compression and preprocessing. Moreover, the spectral reconstruction itself is typically performed on evenly spaced bands within the 400–1000 nm range, without prioritizing agronomically meaningful regions, such as the red edge. In this sense, many approaches appear to reproduce strategies developed for generic spectral reconstruction challenges (e.g., NTIRE) rather than being specifically tailored to agricultural applications. But not only the number and density of spectral bands seems to be a limitation. To date, no study has demonstrated the feasibility of reconstructing the full VIS–NIR interval (400–2500 nm), which would be of greater agronomic interest than the 400–1000 nm range.

Moreover, all of the analyzed studies rely exclusively on proprietary datasets collected for the specific experiment, without leveraging publicly available repositories that could provide broader coverage or crop-specific annotations. The pivotal role of datasets rely not only in terms of their size and diversity but also regarding the type of input data (e.g., reflectance vs. radiance, laboratory vs. in-field acquisitions) and the level of calibration provided. Most deep learning models for SSR rely heavily on large, well-annotated datasets. However, many commonly used datasets lack proper calibration, particularly with respect to reflectance standards. This introduces a substantial source of variability, limiting the transferability and generalizability of trained models. When SSR is applied to field conditions—where lighting is highly variable and the environment is complex—the choice of dataset becomes even more consequential. Models trained exclusively on laboratory-acquired, perfectly standardized data often fail to maintain performance when exposed to real-world variability, such as non-uniform illumination, different backgrounds, and the presence of occlusions or shadows. The issue of "domain shift" between training and deployment environments is particularly acute in agriculture, where seasonal changes, crop types, and imaging hardware may all differ from those represented in the training data.

4.7 Conclusions

In summary, field hyperspectral imaging remains constrained by high costs, complex instrumentation, and limited portability, which restrict its use largely to research contexts. Spectral super-resolution (SSR), leveraging low-cost RGB cameras, offers a more accessible alternative with potential to democratize spectral analytics in agriculture. While SSR methods usually deliver lower accuracy than direct HSI acquisition, they provide clear advantages in scalability, affordability, and the feasibility of near real-time applications. To move from laboratory prototypes to field-ready solutions, it is essential to establish standardized pipelines covering image acquisition, calibration, segmentation, and inference. Equally critical is the creation of large, open, and crop-specific datasets that capture environmental variability, along with robust calibration references to ensure transferability. Advances in adaptive and physically grounded SSR models will be key to overcoming domain shift across sites, devices, and crop types. Although SSR cannot yet replace hyperspectral imaging in all scenarios, it represents a pragmatic pathway toward the wider adoption of spectral tools in precision agriculture. Real progress will depend on coordinated efforts across academia, industry, and stakeholders to translate current advances into robust, user-friendly, and impactful solutions.

Bibliography

[1] J. Zhang et al. "A survey on computational spectral reconstruction methods from RGB to hyperspectral imaging". In: *Sci. Rep.* 12 (2022), p. 11905. DOI: 10.1038/s41598-022-16223-1. URL: https://doi.org/10.1038/s41598-022-16223-1.

[2] A. N. Fsian et al. “Spectral reconstruction from RGB imagery: A potential option for infinite spectral data?” In: *Sensors* 24 (2024), p. 3666. DOI: 10.3390/s24113666. URL: https://doi.org/10.3390/s24113666.

[3] J. He et al. “Spectral super-resolution meets deep learning: Achievements and challenges”. In: *Inf. Fusion* 97 (2023), p. 101812. DOI: 10.1016/j.inffus.2023.101812. URL: https://doi.org/10.1016/j.inffus.2023.101812.

[4] Y. T. Lin and G. D. Finlayson. “Physically Plausible Spectral Reconstruction”. In: *Sensors* 20 (2020), p. 6399. DOI: 10.3390/s20216399. URL: https://doi.org/10.3390/s20216399.

[5] L. Yan et al. “Reconstruction of hyperspectral data from RGB images with prior category information”. In: *IEEE Trans. Comput. Imaging* 6 (2020), pp. 1070–1081. DOI: 10.1109/TCI.2020.3000320. URL: https://doi.org/10.1109/TCI.2020.3000320.

[6] B. Arad et al. “NTIRE 2020 Challenge on Spectral Reconstruction from an RGB Image”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Seattle, WA, USA, 2020, pp. 882–883.

[7] B. Arad et al. “NTIRE 2022 Spectral Recovery Challenge and Data Set”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. New Orleans, LA, USA, 2022, pp. 863–864.

[8] M. Medina-García et al. “Strategies for analysing hyperspectral imaging data for food quality and safety issues—A critical review of the last 5 years”. In: *Microchem. J.* 214 (2025), p. 113994. DOI: 10.1016/j.microc.2025.113994. URL: https://doi.org/10.1016/j.microc.2025.113994.

[9] B. G. Ram et al. “A systematic review of hyperspectral imaging in precision agriculture: Analysis of its current state and future prospects”. In: *Comput. Electron. Agric.* 222 (2024), p. 109037. DOI: 10.1016/j.compag.2024.109037. URL: https://doi.org/10.1016/j.compag.2024.109037.

[10] A. Bhargava et al. “Hyperspectral imaging and its applications: A review”. In: *Heliyon* 10 (2024), e33208. DOI: 10.1016/j.heliyon.2024.e33208. URL: https://doi.org/10.1016/j.heliyon.2024.e33208.

[11] G. Polder et al. “Hyperspectral imaging for detection of plant diseases with deep learning”. In: *Plant Pathol.* 68 (2019), pp. 1017–1024. DOI: 10.1111/ppa.13011. URL: https://doi.org/10.1111/ppa.13011.

[12] E. van de Vijver, G. van der Heijden, and G. Polder. “A high-throughput hyperspectral phenotyping platform for plant traits.” In: *Plant Methods* 16 (2020), p. 123. DOI: 10.1186/s13007-020-00649-z. URL: https://doi.org/10.1186/s13007-020-00649-z.

[13] N. Virlet et al. “Field Scanalyzer: An automated robotic field phenotyping platform for detailed crop monitoring”. In: *Funct. Plant Biol.* 44 (2017), pp. 143–153. DOI: 10.1071/FP16163. URL: https://doi.org/10.1071/FP16163.

[14] D. Eddy, R. Gerhards, and W. Kühbauch. "Precision agriculture using hyperspectral imaging: Early results". In: *Precis. Agric.* 9 (2008), pp. 123–135. DOI: 10.1007/s11119-008-9056-2. URL: https://doi.org/10.1007/s11119-008-9056-2.

[15] J. Zhang, R. Pu, and J. Wang. "Hyperspectral band selection for crop disease detection". In: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. Munich, Germany, 2012, pp. 2456–2459. DOI: 10.1109/IGARSS.2012.6350764. URL: https://doi.org/10.1109/IGARSS.2012.6350764.

[16] J. Zhang, R. Pu, and J. Wang. "Detection of weeds in maize fields using hyperspectral imaging". In: *Comput. Electron. Agric.* 80 (2012), pp. 44–54. DOI: 10.1016/j.compag.2011.10.005. URL: https://doi.org/10.1016/j.compag.2011.10.005.

[17] X. E. Pantazi, D. Moshou, and C. Bravo. "Active learning system for weed species recognition based on hyperspectral sensing". In: *Biosyst. Eng.* 146 (2016), pp. 193–202. DOI: 10.1016/j.biosystemseng.2016.01.014. URL: https://doi.org/10.1016/j.biosystemseng.2016.01.014.

[18] M. Williams, S. Verhoeven, and J. Roelofs. "Hyperspectral weed detection using tractor-mounted imaging". In: *European Conference on Precision Agriculture (ECPA)*. Edinburgh, UK, 2017, pp. 121–128.

[19] P. Horstrand, R. Guerra, and J. Aranda. "Deep learning for weed detection in UAV-based hyperspectral imagery". In: *Remote Sens.* 11 (2019), p. 1559. DOI: 10.3390/rs11131559. URL: https://doi.org/10.3390/rs11131559.

[20] P. Horstrand, R. Guerra, and J. Aranda. "On-board hyperspectral data processing with UAVs for real-time applications". In: *ISPRS J. Photogramm. Remote Sens.* 158 (2019), pp. 36–48. DOI: 10.1016/j.isprsjprs.2019.09.005. URL: https://doi.org/10.1016/j.isprsjprs.2019.09.005.

[21] J. J. Sousa, L. Pádua, and T. Adão. "Hyperspectral UAV imaging for vineyard monitoring". In: *Remote Sens.* 14 (2022), p. 352. DOI: 10.3390/rs14020352. URL: https://doi.org/10.3390/rs14020352.

[22] X. Zhu, S. Liu, and X. Li. "Hyperspectral imaging for disease detection in crops: A case study in rice." In: *Plant Methods* 16 (2020), p. 54. DOI: 10.1186/s13007-020-00581-4. URL: https://doi.org/10.1186/s13007-020-00581-4.

[23] B. Lu et al. "Recent advances of hyperspectral imaging technology and applications in agriculture". In: *Remote Sens.* 12 (2020), p. 2659. DOI: 10.3390/rs12162659. URL: https://doi.org/10.3390/rs12162659.

[24] W. Maes and K. Steppe. "Perspectives for hyperspectral sensing in precision agriculture". In: *Trends Plant Sci.* 24 (2019), pp. 152–164. DOI: 10.1016/j.tplants.2018.11.001. URL: https://doi.org/10.1016/j.tplants.2018.11.001.

[25] J. Chen, W. Wang, and Z. Li. “Real-time fruit sorting using hyperspectral imaging and deep learning.” In: *Postharvest Biol. Technol.* 183 (2022), p. 111735. DOI: 10.1016/j.postharvbio.2021.111735. URL: https://doi.org/10.1016/j.postharvbio.2021.111735.

[26] D. Eddy and R. Gerhards. “Machine learning approaches for hyperspectral weed detection”. In: *Comput. Electron. Agric.* 105 (2014), pp. 173–182. DOI: 10.1016/j.compag.2014.04.003.

[27] I. Herrmann et al. “Ground-level hyperspectral imagery for detecting weeds in wheat fields”. In: *Precis. Agric.* 14 (2013), pp. 637–659. DOI: 10.1007/s11119-013-9321-x.

[28] Y. Zhao, L. Zhang, and J. Li. “Advances in hyperspectral band selection for crop monitoring”. In: *ISPRS J. Photogramm. Remote Sens.* 188 (2022), pp. 38–54. DOI: 10.1016/j.isprsjprs.2022.03.012.

[29] H. Qi, Y. Zhang, and H. Sun. “Deep learning for hyperspectral plant disease detection: A review.” In: *Front. Plant Sci.* 14 (2023), p. 1123456. DOI: 10.3389/fpls.2023.1123456.

[30] X. Zhang, Y. Chen, and Z. Wang. “Generative adversarial networks for hyperspectral data augmentation in crop classification”. In: *Remote Sens.* 14 (2022), p. 2894. DOI: 10.3390/rs14122894.

[31] H. Jiang, C. Yang, and Y. Li. “Estimating crop nitrogen content using UAV-based hyperspectral imagery.” In: *Precis. Agric.* 22 (2021), pp. 1432–1448. DOI: 10.1007/s11119-021-09777-9.

[32] A. Rubio, V. Rueda-Ayala, and R. Gerhards. “Hyperspectral and machine learning approaches for crop nutrient assessment”. In: *Comput. Electron. Agric.* 185 (2021), p. 106157. DOI: 10.1016/j.compag.2021.106157.

[33] M. Corti, D. Cavalli, and R. Colombo. “Assessment of crop water stress with UAV hyperspectral data”. In: *Agric. Water Manag.* 193 (2017), pp. 150–160. DOI: 10.1016/j.agwat.2017.08.012.

[34] S. Sanchez and A. Plaza. “GPU implementation of endmember extraction algorithms for hyperspectral imagery”. In: *IEEE Trans. Geosci. Remote Sens.* 49 (2011), pp. 4157–4171. DOI: 10.1109/TGRS.2011.2144602.

[35] M. Aragón, J. González, and A. Plaza. “Efficient hyperspectral PCA on embedded GPU platforms for precision agriculture”. In: *IEEE International Conference on Image Processing (ICIP)*. Taipei, Taiwan, 2019, pp. 2636–2640. DOI: 10.1109/ICIP.2019.8803007.

[36] J. M. P. Nascimento and J. M. Bioucas-Dias. “Real-time hyperspectral unmixing on FPGA: Design and performance evaluation.” In: *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* 13 (2020), pp. 5621–5633. DOI: 10.1109/JSTARS.2020.3023456.

[37] P. Gyaneshwar and R. R. Nidamanuri. “Real-time multispectral image classification using FPGA-based SVM.” In: *J. Real-Time Image Process.* 19 (2022), pp. 879–891. DOI: 10.1007/s11554-021-01079-0.

[38] D. Rosario, S. López, and A. Plaza. "Real-time FPGA implementation of the Vertex Component Analysis algorithm for hyperspectral unmixing". In: *SPIE 9244, Image and Signal Processing for Remote Sensing XX*. Amsterdam, The Netherlands, 2014. DOI: 10.1117/12.2066872.

[39] A. Benelli, C. Cevoli, and A. Fabbri. "In-field Vis/NIR hyperspectral imaging to measure soluble solids content of wine grape berries during ripening". In: *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. Trento, Italy, 2020, pp. 227–231. DOI: 10.1109/MetroAgriFor50201.2020.9277621.

[40] A. Benelli et al. "In-field and non-destructive monitoring of grapes maturity by hyperspectral imaging". In: *Biosyst. Eng.* 207 (2021), pp. 59–67. DOI: 10.1016/j.biosystemseng.2021.04.006.

[41] S. Gutiérrez et al. "On-The-Go hyperspectral imaging under field conditions and machine learning for the classification of grapevine varieties". In: *Front. Plant Sci.* 9 (2018), p. 1102. DOI: 10.3389/fpls.2018.01102.

[42] J. Fernández-Novales et al. "On The Go VIS SW NIR spectroscopy as a reliable monitoring tool for grape composition within the vineyard". In: *Molecules* 24 (2019), p. 2795. DOI: 10.3390/molecules24152795.

[43] G. Ferrara et al. "The prediction of ripening parameters in Primitivo wine grape cultivar using a portable NIR device". In: *J. Food Compos. Anal.* 114 (2022), p. 104836. DOI: 10.1016/j.jfca.2022.104836.

[44] S. Gutiérrez et al. "On the go hyperspectral imaging for the in field estimation of grape berry soluble solids and anthocyanin concentration". In: *Aust. J. Grape Wine Res.* 24 (2018), pp. 127–133. DOI: 10.1111/ajgw.12376.

[45] L. M. Dale et al. "Hyperspectral imaging applications in agriculture and agro-food product quality and safety control: A review". In: *Appl. Spectrosc. Rev.* 48 (2013), pp. 142–159. DOI: 10.1080/05704928.2012.705800.

[46] A. Benelli, C. Cevoli, and A. Fabbri. "In-field hyperspectral imaging: An overview on the ground-based applications in agriculture". In: *J. Agric. Eng.* 51 (2020), p. 1030. DOI: 10.4081/jae.2020.1030.

[47] J. Underwood et al. "Efficient in-field plant phenomics for row-crops with an autonomous ground vehicle". In: *J. Field Robot.* 34 (2017), pp. 1061–1083. DOI: 10.1002/rob.21728.

[48] Y. Jiang et al. "GPhenoVision: A ground mobile system with multi-modal imaging for field-based high throughput phenotyping of cotton". In: *Sci. Rep.* 8 (2018), p. 1213. DOI: 10.1038/s41598-018-19142-2.

[49] A. Wendel, J. Underwood, and K. Walsh. "Maturity estimation of mangoes using hyperspectral imaging from a ground based mobile platform". In: *Comput. Electron. Agric.* 155 (2018), pp. 298–313. DOI: 10.1016/j.compag.2018.10.021.

[50] B. Wang et al. “Retrieval of leaf chlorophyll content of paddy rice with extracted foliar hyperspectral imagery”. In: *2018 7th International Conference on Agro-Geoinformatics (Agro-Geoinformatics)*. Hangzhou, China, 2018, pp. 1–5.

[51] Q. Wu et al. “Field monitoring of wheat seedling stage with hyperspectral imaging”. In: *Int. J. Agric. Biol. Eng.* 9 (2016), pp. 143–148. DOI: 10.3965/j.ijabe.20160905.1707.

[52] S. Jay et al. “Estimating leaf chlorophyll content in sugar beet canopies using millimeter- to centimeter-scale reflectance imagery”. In: *Remote Sens. Environ.* 198 (2017), pp. 173–186. DOI: 10.1016/j.rse.2017.06.008.

[53] H. Onoyama et al. “Nitrogen prediction model of rice plant at panicle initiation stage using ground-based hyperspectral imaging: Growing degree-days integrated model”. In: *Precis. Agric.* 16 (2015), pp. 558–570. DOI: 10.1007/s11119-015-9394-9.

[54] N. Vigneau et al. “Potential of field hyperspectral imaging as a non destructive method to assess leaf nitrogen content in Wheat”. In: *Field Crop. Res.* 122 (2011), pp. 25–31. DOI: 10.1016/j.fcr.2011.02.003.

[55] R. L. Whetton, T. W. Waine, and A. M. Mouazen. “Hyperspectral measurements of yellow rust and fusarium head blight in cereal crops: Part 2: On-line field measurement”. In: *Biosyst. Eng.* 167 (2018), pp. 144–158. DOI: 10.1016/j.biosystemseng.2018.01.004.

[56] J. Zhao et al. “Vertical features of yellow rust infestation on winter wheat using hyperspectral imaging measurements”. In: *2016 5th International Conference on Agro-Geoinformatics*. Tianjin, China, 2016, pp. 1–4.

[57] C. Römer et al. “Early drought stress detection in cereals: Simplex volume maximisation for hyperspectral image analysis”. In: *Funct. Plant Biol.* 39 (2012), pp. 878–890. DOI: 10.1071/fp12060.

[58] K. N. Reddy et al. “Glyphosate-resistant and glyphosate-susceptible Palmer amaranth (Amaranthus palmeri S. Wats.): Hyperspectral reflectance properties of plants and potential for classification”. In: *Pest Manag. Sci.* 70 (2014), pp. 1910–1917. DOI: 10.1002/ps.3755.

[59] W. Chen et al. “Hyperspectral image segmentation for maize stubble in no-till field”. In: *2017 ASABE Annual International Meeting*. Spokane, WA, USA, 2017.

[60] M. Das, W. S. Yeo, and A. Saptoro. “A review of machine learning in hyperspectral imaging for food safety”. In: *Vib. Spectrosc.* 139 (2025), p. 103828. DOI: 10.1016/j.vibspec.2025.103828.

[61] I. Torres and J. M. Amigo. “An overview of regression methods in hyperspectral and multispectral imaging”. In: *Data Handl. Sci. Technol.* Vol. 32. 2019, pp. 205–230. DOI: 10.1016/B978-0-444-63977-6.00010-9.

[62] C. Zhang et al. “Application of near-infrared hyperspectral imaging for variety identification of coated maize kernels with deep learning”. In: *Infrared Phys. Technol.* 111 (2020), p. 103550. DOI: 10.1016/j.infrared.2020.103550.

[63] M. Z. Alom et al. “A state-of-the-art survey on deep learning theory and architectures”. In: *Electronics* 8 (2019), p. 292. DOI: 10.3390/electronics8030292.

[64] Raghav Khanna et al. “On Field Radiometric Calibration for Multispectral Cameras”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 6503–6510. DOI: 10.1109/ICRA.2017.7989710.

[65] Lei Deng et al. “A Subband Radiometric Calibration Method for UAV-Based Multispectral Remote Sensing”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2018). DOI: 10.1109/JSTARS.2018.2842466.

[66] A. Simoneau and M. Aubé. “Methods to Calibrate a Digital Colour Camera as a Multispectral Imaging Sensor in Low Light Conditions”. In: *Remote Sens.* 15 (2023), p. 3634. DOI: 10.3390/rs15143634.

[67] Aurelie M. Poncet et al. “Multispectral UAS Data Accuracy for Different Radiometric Calibration Methods”. In: *Remote Sensing* 11.1917 (2019). DOI: 10.3390/rs11161917.

[68] Sheng Wang et al. “Unmanned Aerial System multispectral mapping for low and variable solar irradiance conditions: Potential of tensor decomposition”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 155 (2019), pp. 58–71. DOI: 10.1016/j.isprsjprs.2019.06.017.

[69] J. B. Barker et al. “Calibration of a Common Shortwave Multispectral Camera System for Quantitative Agricultural Applications”. In: *Precis. Agric.* 21 (2020), pp. 922–935. DOI: 10.1007/s11119-019-09701-6.

[70] Z. Qin, X. Li, and Y. Gu. “An Illumination Estimation and Compensation Method for Radiometric Correction of Multispectral Images”. In: *IEEE Trans. Geosci. Remote Sens.* 60 (2022), p. 5545012. DOI: 10.1109/TGRS.2022.3225291.

[71] V. Kitanovski, J. B. Thomas, and J. Y. Hardeberg. “Reflectance Estimation from Snapshot Multispectral Images Captured under Unknown Illumination”. In: *29th Color and Imaging Conference Final Program and Proceedings*. Online, 2021, pp. 264–269. DOI: 10.2352/issn.2169-2629.2021.29.264.

[72] H. A. Khan et al. “Illuminant Estimation in Multispectral Imaging”. In: *J. Opt. Soc. Am. A* 34 (2017), pp. 1085–1099. DOI: 10.1364/JOSAA.34.001085.

[73] J. B. Thomas. “Illuminant Estimation from Uncalibrated Multispectral Images”. In: *2015 Colour and Visual Computing Symposium (CVCS)*. Gjovik, Norway, 2015, pp. 1–6. DOI: 10.1109/CVCS.2015.7274890.

[74] Yuqi Li, Qiang Fu, and Wolfgang Heidrich. “Multispectral Illumination Estimation Using Deep Unrolling Network”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Introduces a deep unrolling ADMM-based method for spectral illumination estimation. 2021, pp. 2672–2681.

[75] A. Amziane et al. “Frame-based Reflectance Estimation from Multispectral Images for Weed Identification in Varying Illumination Conditions”. In: *2020 IEEE International Conference on Image Processing (ICIP)*. Paris, France, 2020, pp. 1966–1970. DOI: 10.1109/ICIP40778.2020.9191200.

[76] S. Alvarez-Cortes, T. Kunkel, and B. Masia. “Practical Low-Cost Recovery of Spectral Power Distributions”. In: *Comput. Graph. Forum* 35 (2016), pp. 166–178. DOI: 10.1111/cgf.12717.

[77] M. Dinguirard and P. N. Slater. “Calibration of Space-Multispectral Imaging Sensors: A Review”. In: *Remote Sens. Environ.* 68 (1999), pp. 194–205. DOI: 10.1016/S0034-4257(98)00111-4.

[78] L. Ayala et al. “Light Source Calibration for Multispectral Imaging in Surgery”. In: *Int. J. Comput. Assist. Radiol. Surg.* 15 (2020), pp. 1117–1125. DOI: 10.1007/s11548-020-02195-y.

[79] I. Moya et al. “A new instrument for passive remote sensing Measurements of sunlight-induced chlorophyll fluorescence”. In: *Remote Sens. Environ.* 91 (2004), pp. 186–197. DOI: 10.1016/j.rse.2004.02.012.

[80] V. Heikkinen et al. “Evaluation and unification of some methods for estimating reflectance spectra from RGB images”. In: *J. Opt. Soc. Am. A* 25 (2008), pp. 2444–2458. DOI: 10.1364/JOSAA.25.002444.

[81] Y. T. Lin and G. D. Finlayson. “Exposure Invariance in Spectral Reconstruction from RGB Images”. In: *IS&T International Symposium on Electronic Imaging*. Burlingame, CA, USA, 2019, pp. 284–284. DOI: 10.2352/issn.2169-2629.2019.27.51.

[82] D. R. Connah and J. Y. Hardeberg. “Spectral recovery using polynomial models”. In: *Color Imaging X: Processing, Hardcopy, and Applications*. Vol. 5667. Bellingham, WA, USA, 2005, pp. 65–75. DOI: 10.1117/12.587995.

[83] Y. Li, C. Wang, and J. Zhao. “Locally Linear Embedded Sparse Coding for Spectral Reconstruction From RGB Images”. In: *IEEE Signal Process. Lett.* 25 (2018), pp. 363–367. DOI: 10.1109/LSP.2017.2776167.

[84] R. Timofte, V. De Smet, and L. Van Gool. “A+: Adjusted Anchored Neighborhood Regression for Fast Super-Resolution”. In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Vol. 9006. Singapore, 2015, pp. 111–126.

[85] Z. Shi et al. “HSCNN+: Advanced CNN-Based Hyperspectral Recovery from RGB Images”. In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Salt Lake City, UT, USA, 2018, pp. 11052–11060.

[86] P. Liu and H. Zhao. “Adversarial Networks for Scale Feature-Attention Spectral Image Reconstruction from a Single RGB”. In: *Sensors* 20 (2020), p. 2426. DOI: 10.3390/s20082426.

[87] A. Alvarez-Gila, J. van de Weijer, and E. Garrote. "Adversarial Networks for Spatial Context-Aware Spectral Image Reconstruction from RGB". In: *IEEE International Conference on Computer Vision Workshops (ICCVW)*. Venice, Italy, 2017, pp. 480–489. DOI: 10.1109/ICCVW.2017.64.

[88] B. Arad et al. "NTIRE 2018 Challenge on Spectral Reconstruction from RGB Images". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Salt Lake City, UT, USA, 2018.

[89] C. Chen et al. "A review of hyperspectral image super-resolution based on deep learning". In: *Remote Sens.* 15 (2023), p. 2853. DOI: 10.3390/rs15112853.

[90] A. Arad and O. Ben-Shahar. "Sparse Recovery of Hyperspectral Signal from Natural RGB Images". In: *Computer Vision—ECCV 2016*. Amsterdam, The Netherlands, 2016.

[91] F. Yasuma et al. "Generalized Assorted Pixel Camera: Postcapture Control of Resolution, Dynamic Range, and Spectrum". In: *IEEE Trans. Image Process.* 19 (2010), pp. 2241–2253. DOI: 10.1109/TIP.2010.2046811.

[92] S. Chakrabarti and T. Zickler. "Statistics of Real-World Hyperspectral Images". In: *CVPR 2011*. Colorado Springs, CO, USA, 2011, pp. 193–200.

[93] G. Shi et al. "Multi-manifold locality graph preserving analysis for hyperspectral image classification." In: *Neurocomputing* 388 (2020), pp. 45–59. DOI: 10.1016/j.neucom.2019.12.112.

[94] P. Ghamisi, J. A. Benediktsson, and S. Phinn. "Fusion of Hyperspectral and LiDAR Data in Classification of Urban Areas". In: *2014 IEEE Geoscience and Remote Sensing Symposium*. Quebec City, QC, Canada, 2014.

[95] P. Gamba et al. "Foreword to the 2008 GRSS Data Fusion Contest." In: *IEEE GRSL* 6 (2009), pp. 349–353.

[96] D. Landgrebe. "Hyperspectral Image Data Analysis". In: *IEEE Signal Process. Mag.* 19 (2002), pp. 17–28. DOI: 10.1109/79.974718.

[97] L. V. Katkovsky et al. "Fast Atmospheric Correction Method for Hyperspectral Data". In: *Remote Sensing* 10 (2018), p. 1698. DOI: 10.3390/rs10111698.

[98] R. O. Green et al. "Imaging Spectroscopy and the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)". In: *Remote Sens. Environ.* 65 (1998), pp. 227–248. DOI: 10.1016/S0034-4257(98)00064-9.

[99] Y. Zhong et al. "WHU-Hi: UAV-borne hyperspectral with high spatial resolution (H^2) benchmark datasets and classifier for precise crop identification based on deep convolutional neural network with CRF". In: *Remote Sens. Environ.* 250 (2020), p. 112012. DOI: 10.1016/j.rse.2020.112012.

[100] S. Koundinya et al. "2D-3D CNN based architectures for spectral reconstruction from RGB images". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Salt Lake City, UT, USA, 2020, pp. 957–958.

[101] J. Zhao et al. “Deep learning in hyperspectral image reconstruction from single RGB images—A case study on tomato quality parameters”. In: *Remote Sens.* 12 (2020), p. 3258. DOI: 10.3390/rs12193258.

[102] N. Lailyshofa and A. H. Saputro. “Hyperspectral Rice Grain Image Reconstruction Using HR-ResNet Algorithm to Construct Rice Spectral Reflectance Profile”. In: *2023 International Conference on Information Technology Research and Innovation (ICITRI)*. Jakarta, Indonesia, 2023, pp. 54–59. DOI: 10.1109/ICITRI59340.2023.10249254.

[103] M. T. Ahmed, O. Monjur, and M. Kamruzzaman. “Deep learning-based hyperspectral image reconstruction for quality assessment of agro-product”. In: *J. Food Eng.* 382 (2024), p. 112223. DOI: 10.1016/j.jfoodeng.2024.112223.

[104] M. Ahmed, A. Villordon, and M. Kamruzzaman. “Comparative analysis of Hyperspectral Image Reconstruction using Deep Learning for Agricultural and Biological Applications”. In: *Results Eng.* 23 (2024), p. 102623. DOI: 10.1016/j.rineng.2024.102623.

[105] M. Ahmed et al. “Hyperspectral image reconstruction for predicting chick embryo mortality towards advancing egg and hatchery industry”. In: *Smart Agric. Technol.* 9 (2024), p. 100533. DOI: 10.1016/j.atech.2024.100533.

[106] F. Dong et al. “Spectral reconstruction from RGB image to hyperspectral image: Take the detection of glutamic acid index in beef as an example”. In: *Food Chem.* 463 (2025), p. 141543. DOI: 10.1016/j.foodchem.2024.141543.

[107] J. Fu et al. “Maize disease detection based on spectral recovery from RGB images”. In: *Front. Plant Sci.* 13 (2022), p. 1056842. DOI: 10.3389/fpls.2022.1056842.

[108] W. Yang et al. “Investigating the impact of hyperspectral reconstruction techniques on the quantitative inversion of rice physiological parameters: A case study using the MST++ model”. In: *J. Integr. Agric.* 24 (2024), pp. 2540–2557. DOI: 10.1016/j.jia.2024.03.082.

[109] J. Zhao et al. “Deep-Learning-Based Multispectral Image Reconstruction from Single Natural Color RGB Image—Enhancing UAV-Based Phenotyping”. In: *Remote Sens.* 14 (2022), p. 1272. DOI: 10.3390/rs14051272.

[110] Y. Cai et al. “MST++: Multi-stage spectral-wise transformer for efficient spectral reconstruction”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. New Orleans, LA, USA, 2022, pp. 744–754. DOI: 10.1109/CVPRW56347.2022.00090.

Chapter 5

Spectral Reconstruction of Hyperspectral Data Acquired in a Vineyard

5.1 Introduction

The previous chapter presented a comprehensive overview of the emerging field of spectral reconstruction, outlining the fundamental principles, current methodologies, and research trends that aim to infer high-dimensional spectral information from standard RGB images. That discussion positioned spectral reconstruction as a promising alternative to traditional hyperspectral imaging, capable of drastically reducing hardware costs and acquisition complexity while maintaining a high level of analytical depth. Within this context, the adoption of deep learning models particularly convolutional and residual architectures has emerged as the most effective strategy for recovering the spectral richness lost during RGB acquisition.

Building upon that theoretical and methodological framework, this chapter focuses on the practical implementation and evaluation of a deep residual network (HSCNN-R) applied to hyperspectral data acquired in vineyard environments. The objective is to investigate whether such architectures, originally designed for general-purpose spectral super-resolution, can be effectively adapted to agricultural imaging, where the spectral response of plant tissues is influenced by complex factors such as illumination variability, geometry, and biochemical composition.

The following sections describe the dataset preparation, the network architecture and training strategy, and the quantitative assessment of reconstruction accuracy through standard spectral metrics. Particular attention is given to the analysis of reconstructed hyperspectral cubes in relation to the original reference data, in order to evaluate both numerical performance and spectral fidelity. By grounding the implementation in the agronomic context introduced in the previous chapters, this final part of the work aims to demonstrate the feasibility of data-driven spectral enhancement as a realistic step toward the operational deployment of low-cost, RGB-based imaging systems for precision viticulture.

5.2 Materials and Methods

5.2.1 Field Hyperspectral Image Acquisition

The experimental campaign was conducted on a commercial vineyard of *Vitis vinifera L.* cv. Sangiovese, located in the surroundings of Cesena, Italy[1]. The monitoring period covered 17 acquisition days between August 20th and October 4th, 2019. A single side of one vineyard row was selected and subdivided into 11 sections. For every acquisition date, one hyperspectral cube was collected from each of the 11 vineyard sections, yielding a total of 186 scans over the entire experimental period. Image acquisition was performed under clear sky conditions and direct sunlight, between 10:30 and 12:00 local time, to ensure uniform illumination. Before each acquisition dark and white references target were acquired to have a reference for the subsequent calibration procedure.

Hyperspectral image data were acquired using a push-broom camera (Nano-Hyperspec VNIR, Headwall Photonics, Inc., Fitchburg, MA, USA), operating in the visible and near-infrared (Vis/NIR) spectral range from 400 to 1000 nm. The sensor provided 272 contiguous spectral bands with a nominal spectral resolution of 2.2 nm and a spatial resolution of 640 pixels per line. As is often the case with spectral acquisition systems and other scientific imaging instruments, 12-bit HSI images are stored in a 16-bit format. This means that although the sensor's dynamic range corresponds to 12 bits (4096 discrete intensity levels), the data are saved in a 16-bit (uint16) container for compatibility and ease of processing. In this representation, only the lower (or sometimes higher) 12 bits are used, while the remaining bits remain unused. This practice preserves the full dynamic range of the spectral measurement while maintaining a standardized data format. The optical setup employed a 17 mm effective focal length lens, oriented perpendicularly to the scanned vineyard surface. The camera was mounted on a mobile support (a garden cart) at a height of 120 cm above the ground, positioned approximately 160 cm from the grapevine canopy. It was powered by a 12 V, 45 Ah automotive battery connected through a DC–AC inverter. Each hyperspectral scan was obtained by translating the camera along the scanning direction to reconstruct the two-dimensional scene from successive line acquisitions.

These hyperspectral data were later used as reference ground truth for the RGB-to-hyperspectral reconstruction model described in the following sections.

5.2.2 Data Pre-processing

The data pre-processing workflow included three main steps: RGB extraction, image segmentation, and radiometric calibration of hyperspectral data.

RGB extraction. RGB images were automatically generated by the acquisition software from the hyperspectral data cubes: Hyperspec III (Headwall Photonics

[1]The hyperspectral image acquisition campaign was conducted by collaborating researchers within the same project, and the complete dataset was previously used in a study focused on the qualitative evaluation of grape berry quality using full hyperspectral data [1].

Inc., Fitchburg, MA, USA). Specifically, the software selected the raw spectral bands centered at 472 nm, 538 nm, and 707 nm, and combined them through an internal rendering process to produce a standard RGB visualization (Figure 5.1).

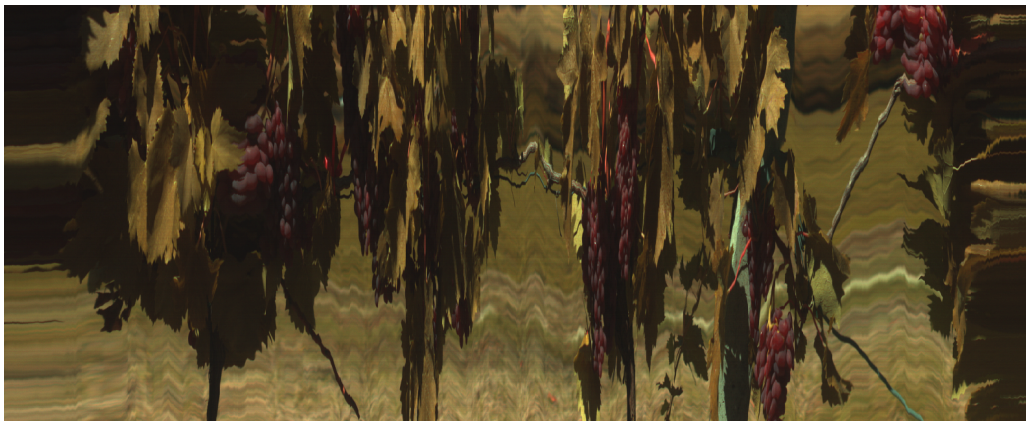


Figure 5.1: RGB image extracted by Hyperspec III from the corresponding HSI acquisition.

Although these images are directly derived from the hyperspectral acquisitions and therefore spatially and spectrally linked, the rendering pipeline includes undocumented operations intended for visualization purposes, such as color balance and tone mapping. This behavior mimics the typical processing applied in conventional RGB cameras, including white balance, gamma correction, and compression, and thus provides a realistic approximation of consumer-grade RGB imagery. It is important to note that the resulting RGB images are physically linked to the hyperspectral data, as they are derived directly from the same acquisition process. This behavior resembles the processing pipeline of conventional RGB cameras, where white balance, gamma correction, and compression are typically applied. Therefore, these rendered RGB images represent a realistic approximation of standard camera outputs, while preserving their spectral correspondence with the original hyperspectral scenes.

Segmentation. To isolate the grape berries from background elements such as leaves, branches, or sky, both the RGB and hyperspectral images were segmented using a spectral similarity approach based on the Spectral Angle Mapper (SAM), prior the reflectance computation. For each pixel, the spectral angle θ between its radiance spectrum $\mathbf{a}$ and the mean reference spectrum $\mathbf{b}$, computed as the average of several representative grape berry spectra, was calculated as:

$$\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \, \|\mathbf{b}\|} \tag{5.1}$$

where $\mathbf{a}$ and $\mathbf{b}$ are 272-dimensional vectors corresponding to the spectral signatures of the pixel and the reference, respectively. Pixels exhibiting a spectral angle less than or equal to 7° were classified as grape tissue, while all others were excluded from the mask. This threshold empirically ensured a good trade-off between sensitivity and specificity in isolating the grape berries, resulting in a clean segmentation of the fruit areas to be used in subsequent analysis.



Figure 5.2: RGB image segmented with angular vector method.

Radiometric calibration. Following segmentation, the hyperspectral images were radiometrically corrected to obtain reflectance values. The calibration was performed using reference white and dark images acquired under the same illumination conditions as the vineyard scans. The dark reference image accounted for sensor noise and dark current, while the white reference provided the baseline for reflectance normalization. The reflectance R for each pixel and wavelength was computed according to the standard formula:

$$R = \frac{I_{\text{raw}} - I_{\text{dark}}}{I_{\text{white}} - I_{\text{dark}}} \tag{5.2}$$

where I_{raw} represents the pixel intensity of the sample, I_{dark} the dark reference signal, and I_{white} the white reference signal. The resulting calibrated reflectance cubes were then used as ground truth data for the RGB-to-hyperspectral reconstruction model.

5.2.3 Dataset Preparation

The dataset used for the RGB-to-hyperspectral reconstruction task was derived from the calibrated hyperspectral images after segmentation. To reduce the influence of instrumental noise, the first and last spectral bands—corresponding to the extremes of the sensor's sensitivity range—were excluded from the analysis. Specifically, the first 14 bands and the last 3 bands were removed, as they exhibited high noise levels due to low sensor response and illumination instability.

After band exclusion, the remaining spectra were uniformly resampled to obtain a compact yet representative spectral profile. A total of 31 equally spaced bands were selected between 430 nm and 991 nm, covering the visible and near-infrared (Vis–NIR) region of interest. This spectral resampling provided a balanced compromise between data dimensionality and spectral information retention, ensuring compatibility with the network input while maintaining the essential spectral variability required for accurate reconstruction.

Each pixel belonging to the segmented grape regions was thus associated with a 31-dimensional reflectance spectrum, serving as the reference ground truth. The

corresponding RGB pixel values, spatially aligned with the hyperspectral data, were extracted from the rendered RGB images to form paired samples.

Finally, to prepare the data for training, the segmented pixels from each image were grouped into spatial patches (or "boxes") of 40×40 pixels. For each hyperspectral image, a set of paired RGB and hyperspectral boxes was generated, with tensor dimensions of $40 \times 40 \times 3$ for the RGB inputs and $40 \times 40 \times 31$ for the hyperspectral targets. Each box was assigned a unique filename following the convention `day_x_y_n_z`, where the variables indicate the acquisition day (x), the progressive image number within that day (y), the vineyard section (n), and the sequential index of the extracted patch (z). This systematic naming allowed each patch to be easily traced back to its corresponding original image and acquisition conditions.

The resulting dataset of paired RGB and hyperspectral patches was randomly divided into three independent subsets: training (70%), validation (15%), and test (15%). Importantly, this division was performed at the image level rather than the patch level. Consequently, all patches extracted from a given hyperspectral image (`day_x_y`) were assigned to the same subset, ensuring that no spatial or spectral information from a single acquisition could be shared across different data partitions. This strategy prevented data leakage and provided a more realistic evaluation of the model's generalization capability.

5.2.4 Network Architecture and Training Parameters

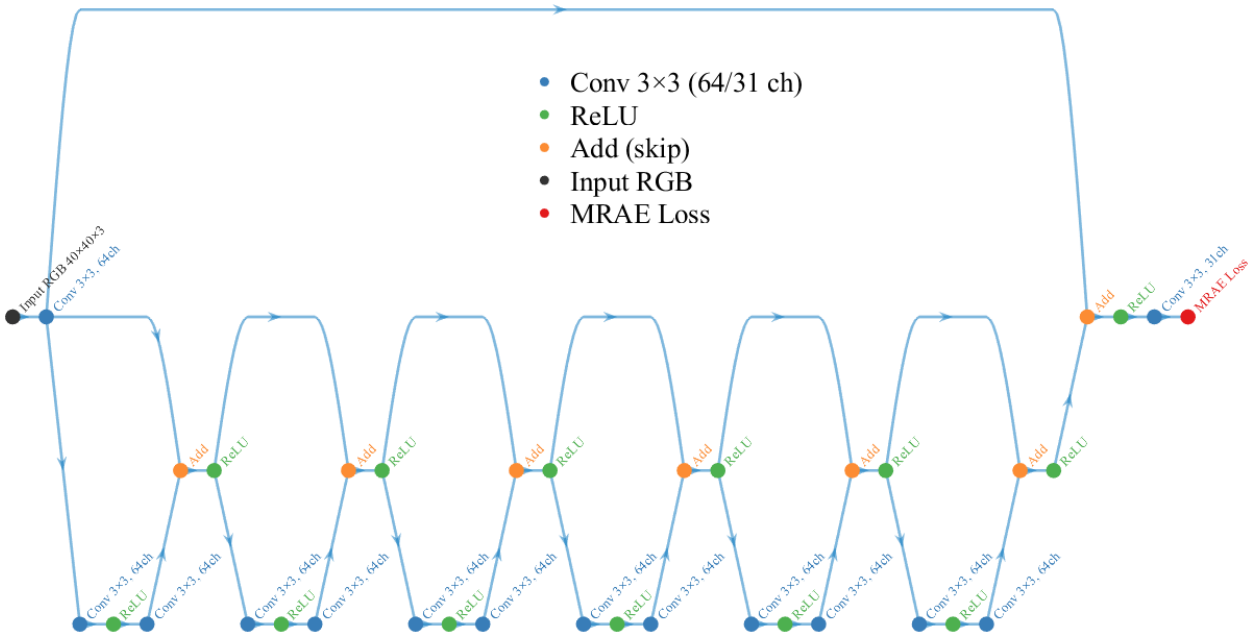


Figure 5.3: HSCNN-R.

The RGB-to-hyperspectral reconstruction model was implemented using a residual convolutional neural network architecture based on the HSCNN-R framework

(Figure5.3). This architecture is specifically designed for hyperspectral image reconstruction tasks and leverages residual learning to facilitate the mapping between RGB inputs and high-dimensional spectral outputs. Residual connections enable the network to efficiently propagate both low-level and high-level features, mitigating vanishing gradient issues and improving convergence stability during training.

The network was implemented in MATLAB using the Deep Learning Toolbox. The input to the model consisted of RGB patches of size $40 \times 40 \times 3$, while the output corresponded to the reconstructed hyperspectral patches of size $40 \times 40 \times 31$. Prior to training, all RGB inputs were normalized by 255. The same scaling parameters were consistently applied during dataset loading for validation and testing, ensuring data consistency and preventing information leakage.

The loss function adopted for model optimization was the Mean Relative Absolute Error (MRAE), which is particularly suitable for hyperspectral reconstruction as it normalizes the absolute reconstruction error with respect to the true spectral values, thus reducing bias across different intensity levels.

The model was trained for a total of 1000 epochs with an initial learning rate of 1×10^{-3}, which was halved every 100 epochs to promote convergence. A mini-batch size of 64 samples was used throughout training. The optimization process was carried out using the Adam optimizer, configured with a gradient decay factor (β_1) of 0.9, a squared gradient decay factor (β_2) of 0.999, and $\varepsilon = 10^{-8}$.

All experiments were executed on MATLAB R2024b using GPU acceleration. The described configuration was found to provide a good trade-off between training stability and reconstruction accuracy, ensuring robust spectral recovery across the different test images.

5.3 Results

5.3.1 Training Configuration and Performance

The model was trained using a total of 2176 RGB–HSI paired patches from the training set, each of size 40×40 pixels. All experiments were conducted on a workstation equipped with an AMD Ryzen Threadripper PRO 5975WX CPU featuring 32 physical cores and 64 logical processors, with a base clock frequency of 3.6 GHz and 128 GB of system RAM.

GPU acceleration was provided by an NVIDIA Quadro T1000 graphics card with 8 GB of GDDR6 memory, a 128-bit memory interface, a bandwidth of up to 160 GB/s, and 896 CUDA cores operating at a clock frequency between 1395 and 1560 MHz.

Given the HSCNN-R architecture, the input and output dimensions ($40 \times 40 \times 3$ and $40 \times 40 \times 31$, respectively), and a mini-batch size of 64, the total memory requirement during training, computed as the sum of parameters, activations, and gradients, was approximately 1 GB. [2]

This value was estimated by analyzing the GPU utilization profile during training and confirmed to be consistent with theoretical expectations. The memory footprint

[2]The memory required during training depends on the number of network parameters, the size of the intermediate activations, and the gradients stored for backpropagation. The total memory

remained well within the available 8 GB of GPU memory, allowing the network to be trained without paging or memory overflow.

During training, the average GPU usage remained stable, with an estimated computational cost of about 1.4 GFLOPs per forward–backward pass. The total training time required to complete 400 epochs was approximately 1.25 hours. Figure 5.4 shows the evolution of the Mean Relative Absolute Error (MRAE) loss over the training epochs, indicating a stable convergence trend. The final training loss reached 0.2, while the validation loss stabilized around 3, confirming that the learning rate decay (reduced by a factor of 10 every 100 epochs) effectively supported convergence without overfitting.

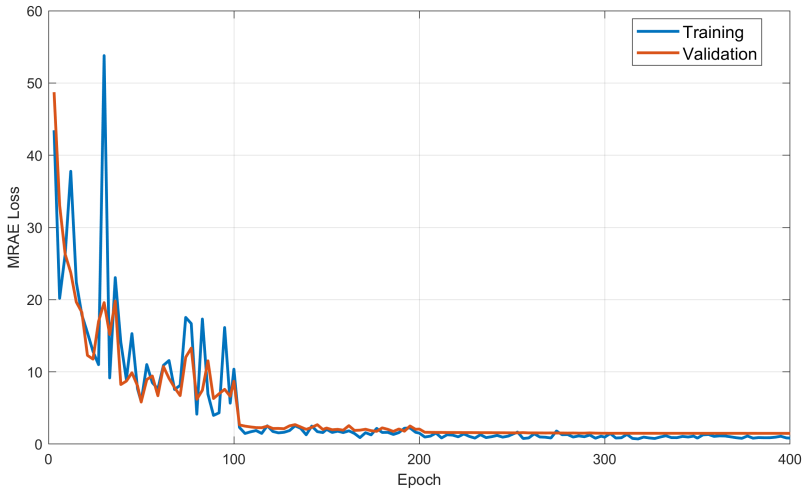


Figure 5.4: Training and validation MRAE loss evolution over 400 epochs.

5.3.2 Test Performance

To assess the quality of spectral reconstruction the Root Mean Square Error (RMSE) was employed. RMSE quantifies the root mean square difference between the recovered and reference spectra, reflecting the global reconstruction accuracy. Figure 5.5 illustrates the ground-truth and reconstructed spectral profiles for three samples characterized by distinct reference Brix values. As shown, the reconstructed spectra exhibit a high degree of consistency with the corresponding ground-truth curves. The spectral magnitudes and overall trends demonstrate strong agreement, particularly within the visible region, where an almost perfect overlap is observed. Minor discrepancies can be noticed in certain wavebands of the SW-NIR region for specific samples. The reported R^2 and RMSE values further confirm the strong correlation

load can be expressed as:

$$M_{\text{total}} = M_{\text{params}} + M_{\text{activations}} + M_{\text{gradients}} \tag{5.3}$$

where M_{params} is the memory occupied by trainable weights, $M_{\text{activations}}$ represents the temporary tensors generated during forward propagation, and $M_{\text{gradients}}$ denotes the memory needed to store gradient values for parameter updates. Among these terms, $M_{\text{activations}}$ is typically dominant and scales linearly with the batch size and the size of the input and output tensors.

and low reconstruction error, respectively, thus validating the reliability of the spectral recovery across samples with varying Brix levels.

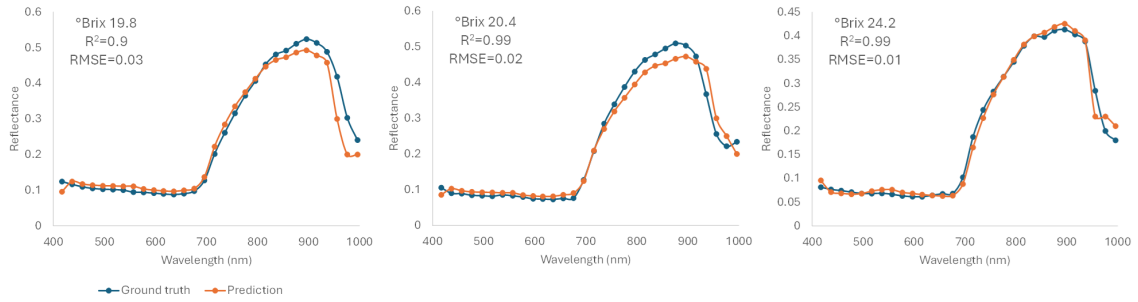


Figure 5.5: Example of ground truth (blue) and reconstructed (red) spectrum for a representative references Brix.

Figure 5.6 displays the complete set of reconstructed (Predicted) and ground-truth spectral reflectance curves across all samples. The reconstructed spectra generally follow the same trend and magnitude as the ground-truth data, demonstrating the model's ability to reproduce spectral behavior consistently across different samples. The reflectance values increase smoothly with wavelength, showing the characteristic transition around 700 nm, which is well preserved in the reconstructed spectra. Although minor discrepancies can be observed in certain regions of the SW-NIR range, the overall correspondence between predicted and reference spectra remains high. This close alignment suggests that the proposed reconstruction method captures both global and fine-grained spectral features effectively. The visual agreement observed across the entire dataset further supports the quantitative results obtained from the R^2 and RMSE metrics, confirming the robustness and generalization capability of the spectral reconstruction model.

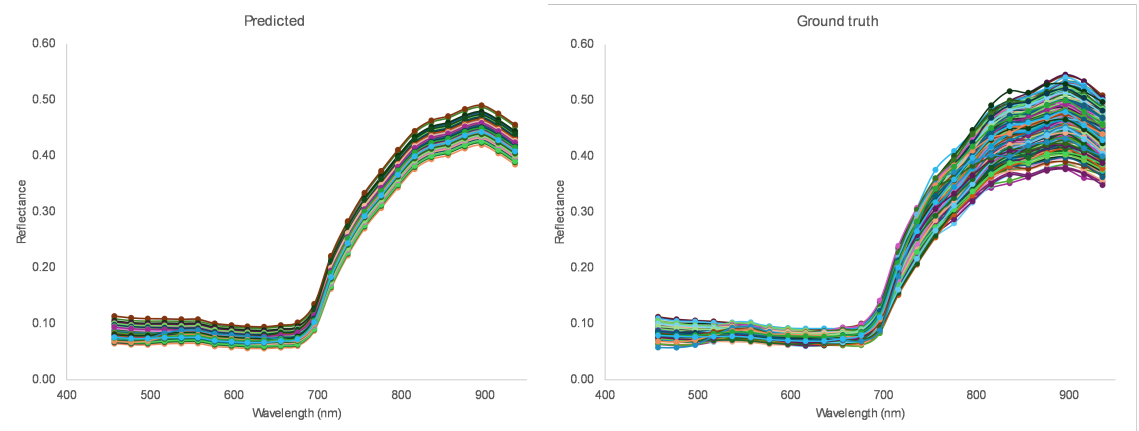


Figure 5.6: Ground and Predicted spectra for all Testset.

5.4 Conclusion

The results obtained in this study are promising and demonstrate the potential of the proposed spectral reconstruction approach as a solid starting point for further

investigation. Although the current model already achieves satisfactory performance, future improvements to the neural network architecture and training strategy are expected to enhance reconstruction accuracy even further. The reconstructed spectra will be subsequently employed to develop predictive models for estimating Brix values, enabling indirect quality assessment from reconstructed hyperspectral data. The performance of these models will be compared with previous studies that utilized full hyperspectral datasets, allowing for a comprehensive evaluation of the effectiveness and practical applicability of the proposed approach.

Bibliography

[1] A. Benelli et al. "In-field and non-destructive monitoring of grapes maturity by hyperspectral imaging". In: *Biosyst. Eng.* 207 (2021), pp. 59–67. DOI: 10.1016/j.biosystemseng.2021.04.006.

Appendix A

h.estrazione_ipercubo.m

```
1   function [ipercubo,lato] = estrazione\_ipercubo(foto,
        nome_file, x_start, y_start, lato)
2   foto = foto(4:end,1:end-3);
3   ipercubo = struct('Nome_file','', 'Immagine',[], 'Lunghezze
        ',[],'Cubo_imm',[],'Bianco',[], 'B_media',[],'B_Dev',
        [], 'Cubo_target',[]);
4   ipercubo.Immagine = foto;
5   ipercubo.Nome_file = nome_file;
6   if nargin > 2
7   if x_start < 5
8   x_start = x_start + 5;
9   end
10  if y_start < 5
11  y_start =  y_start +5;
12  end
13  colonna_start = floor(x_start/5)*5 +1;
14  riga_start = floor(y_start/5)*5 +1;
15  if lato < 5
16  lato = lato+5;
17  end
18  lato = floor(lato/5)*5;
19  lato = lato-1;
20  box = foto(riga_start-5:riga_start+lato+5, colonna_start-5:
        colonna_start+lato+5);
21  else
22  box = foto;
23  lato = 0;
24  end
25  Ind_Lung_Sinistra = [1,5,949; 2,10,897; 3,15,841; 4,20,779;
        5,25,710; 6,4,940; 7,9,890; 8,14,828; 9,19,766;
        10,24,699; 11,3,930; 12,8,879;
26  13,13,814; 14,18,751; 15,23,686; 16,2,920; 17,7,864;
        18,12,804; 19,17,738; 20,22,686; 21,1,912; 22,6,852;
        23,11,789; 24,16,727; 25,21,711;];
27  [rows,cols] = size(box);
28  num_band = 25;
29  l = 5;
30  for m=1:25
31  ipercubo.Lunghezze(m) = Ind_Lung_Sinistra(m,3);
32  end
```

```
        i_max = floor(rows/5)-1;
        i_max_min = i_max - 1;
        j_max = floor(cols/5)-1;
        j_max_min = j_max-1;
        band = 1;
        for r = 1:l
        for c = 1:l
        griglia = zeros(rows, cols);
        for i = 0:i\_max
        for j = 0:j\_max
        riga = (l*i)+r;
        colonna = (l*j)+c;
        griglia(riga,colonna) = 1;
        end
        end
        griglia = uint8(griglia);
        filtrata = box.*griglia;
        interpolata_righe = filtrata;
        for i = 0:i_max
        riga = r +(l*i);
        for j = 0:j_max_min
        colonna = c +(l*j);
        for k = 1:4
        interpolata_righe(riga,colonna+k)=(k/l)*filtrata(riga,
            colonna+l)+((l-k)/l)*filtrata(riga,colonna);
        end
        end
        end
        interpolata = interpolata_righe;
        for i=0:i_max_min
        riga = r +(l*i);
        for k=1:4
        for j=1:cols
        interpolata(riga+k,j)=(k/l)*interpolata_righe(riga+l,j) +((
            l-k)/l)*interpolata_righe(riga,j);
        end
        end
        end
        ipercubo.Cubo_imm{band} = interpolata(6:end-5,6:end-5);
        band = band+1;
        end
        end
        [LunghezzeOrdinato, indici] = sort(ipercubo.Lunghezze);
        ipercubo.Lunghezze = LunghezzeOrdinato;
        ipercubo.Cubo_imm = ipercubo.Cubo_imm(indici);
        ipercubo.Cubo_imm = cat(3, ipercubo.Cubo_imm{:});
        ipercubo.Bianco = ipercubo.Cubo_imm(1770:end,1:end,:);
        ipercubo.B_media = squeeze(mean(ipercubo.Bianco,[1, 2]));
        ipercubo.B_Dev = squeeze(std(double(ipercubo.Bianco),0,[1,
            2]));
        ipercubo.Cubo_target = ipercubo.Cubo_imm(1:1555,1:1075,:);
        end
```

Appendix B

h.processing

```
function ipercubo = process(ipercubo, mode, r)
if strcmp(r, 'ROI')
inputMatrix = double(ipercubo.ROI);
elseif strcmp(r, 'ROI_r')
inputMatrix = double(ipercubo.ROI_r);
inputMatrix = double(ipercubo.Cubo);
end
a = size(inputMatrix);
[~,c] = size(a);
if c == 3
[H, W, num_bands] = size(inputMatrix);
inputMatrix = reshape(inputMatrix,H*W,num_bands);
end
if strcmp(r, 'ROI')
ipercubo.SNV = applySNV(inputMatrix);
ipercubo.D = applyFirstDerivative(inputMatrix);
ipercubo.C = applyMeanCentering(inputMatrix);
ipercubo.SNV_D = applyFirstDerivative(ipercubo.SNV);
ipercubo.SNV_D_C = applyMeanCentering(ipercubo.SNV_D);
   ipercubo.SNV_D = reshape(ipercubo.SNV_D,H,W,num_bands);
ipercubo.SNV = reshape(ipercubo.SNV,H,W,num_bands);
ipercubo.D = reshape(ipercubo.D,H,W,num_bands);
ipercubo.C = reshape(ipercubo.C,H,W,num_bands);
ipercubo.SNV_D = reshape(ipercubo.SNV_D,H,W,num_bands);
ipercubo.SNV_D_C = reshape(ipercubo.SNV_D_C,H,W,num_bands);

elseif strcmp(r, 'ROI_r')
ipercubo.SNV_r = applySNV(inputMatrix);
ipercubo.D_r = applyFirstDerivative(inputMatrix);
ipercubo.C_r = applyMeanCentering(inputMatrix);
ipercubo.SNV_D_r = applyFirstDerivative(ipercubo.SNV_r);
ipercubo.SNV_D_C_r = applyMeanCentering(ipercubo.SNV_D_r);
ipercubo.SNV_r = reshape(ipercubo.SNV_r,H,W,num_bands);
ipercubo.D_r = reshape(ipercubo.D_r,H,W,num_bands);
ipercubo.C_r = reshape(ipercubo.C_r,H,W,num_bands);
ipercubo.SNV_D_r = reshape(ipercubo.SNV_D_r,H,W,num_bands);
ipercubo.SNV_D_C_r = reshape(ipercubo.SNV_D_C_r,H,W,
   num_bands);
elseif strcmp(r, 'ROI_filtrato')
ipercubo.SNV_filtrato = applySNV(inputMatrix);
```

```
39          ipercubo.D_filtrato= applyFirstDerivative(inputMatrix);
40          ipercubo.C_filtrato = applyMeanCentering(inputMatrix);
41          ipercubo.SNV_D_filtrato= applyFirstDerivative(ipercubo.
               SNV_filtrato);
42          ipercubo.SNV_D_C_filtrato = applyMeanCentering(ipercubo.
               SNV_D_filtrato);
43          ipercubo.Totale_filtrato = cat(2,inputMatrix,ipercubo.
               D_filtrato);
44          elseif strcmp(r, 'ROI_r_filtrato')
45          ipercubo.SNV_r_filtrato= applySNV(inputMatrix);
46          ipercubo.D_r_filtrato = applyFirstDerivative(inputMatrix);
47          ipercubo.C_r_filtrato = applyMeanCentering(inputMatrix);
48          ipercubo.SNV_D_r_filtrato = applyFirstDerivative(ipercubo.
               SNV_r_filtrato);
49          ipercubo.SNV_D_C_r_filtrato= applyMeanCentering(ipercubo.
               SNV_D_r_filtrato);
50          ipercubo.Totale_r_filtrato = cat(2,inputMatrix,ipercubo.
               D_r_filtrato);
51          elseif strcmp(r, 'Cubo')
52          ipercubo.SNV_Cubo = applySNV(inputMatrix);
53          ipercubo.D_Cubo = applyFirstDerivative(inputMatrix);
54          ipercubo.C_Cubo = applyMeanCentering(inputMatrix);
55          ipercubo.SNV_D_Cubo = applyFirstDerivative(ipercubo.
               SNV_Cubo);
56          ipercubo.SNV_D_C_Cubo = applyMeanCentering(ipercubo.
               SNV_D_Cubo);
57          ipercubo.Totale_Cubo = cat(2,inputMatrix,ipercubo.D_Cubo);
58          end
59          if strcmp(mode, 'uint8')
60          ipercubo.SNV_r= rescaleToUint8(SNV);
61          ipercubo.SNV_D= rescaleToUint8(SNV_D);
62          ipercubo.SNV_D_C= rescaleToUint8(SNV_D_C);
63          ipercubo.D_r= rescaleToUint8(D);
64          ipercubo.C= rescaleToUint8(C);
65          end
66
67          function snvMatrix = applySNV(matrix)
68          meanRow = mean(matrix, 2);
69          stdRow = std(matrix, 0, 2);
70          stdRow(stdRow == 0) = 1;
71          snvMatrix = (matrix - meanRow) ./ stdRow;
72          end
73
74          function derivativeMatrix = applyFirstDerivative(matrix)
75          derivativeMatrix = diff(matrix, 1, 2);
76          derivativeMatrix = [derivativeMatrix, zeros(size(matrix, 1)
               , 1)];
77          end
78
79          function centeredMatrix = applyMeanCentering(matrix)
80          meanriga = double(mean(matrix, 2));
81          centeredMatrix = matrix - meanriga;
82          end
83
84          function uint8Matrix = rescaleToUint8(matrix)
85          minVal = min(matrix(:));
```

```
86          maxVal = max(matrix(:));
87          scaledMatrix = (matrix- minVal) / (maxVal - minVal) * 255;
88          uint8Matrix = uint8(scaledMatrix);
89          end
```

Appendix C

camera.app

```
1         classdef camera < matlab.apps.AppBase
2
3         properties (Access = public)
4         Ind_Lung_Normale =[1,912; 2,920; 3,930; 4,940; 5,949;
              6,852; 7,864;8,879;9,890;10,897; 11,      789;12,
              804;13, 814;14, 828;15, 841;16, 727;17, 738;18, 751;
              19,766; 20,779; 21,711; 22,686; 23,687; 24,699; 25,710;]
5         Ind_Lung_Destra =[1,21,711; 2,16,727; 3,11,789; 4,6,852;
              5,1,912; 6,22,686; 7,17, 738; 8,12,804; 9,7,864;
              10,2,920; 11,23,686;  12,18,751; 13,13,814;
6         14,8,879; 15,23 686; 16,24,699; 17,19,766; 18,14,828;
              19,9,890; 20,4,940; 21,25,710; 22,20,779; 23,15,841;
              24,10,897; 25,5,949;]
7         Ind_Lung_Sinistra = [1,5,949; 2,10,897; 3,15,841; 4,20,779;
               5,25,710; 6,4,940; 7,9,890; 8,14,828; 9,19,766;
              10,24,699; 11,3,930; 12,8,879;
8         13,13,814; 14,18,751; 15,23,686; 16,2,920; 17,7,864;
              18,12,804; 19,17,738; 20,22,686; 21,1,912; 22,6,852;
              23,11,789; 24,16,727; 25,21,711;]
9         g %variabile globale per l'oggetto gigecam
10        immagine % variabile riempita durante la preview
11        durata_desiderata = zeros(10000,1)
12        tempoIterazione = zeros(10000,1) % array che mi contiene il
              tempo delle varie iterazioni ogni volta che faccio il
              salvataggio
13        tempo_pre_attesa = zeros(10000,1)
14        tempo_controlli = zeros(10000,1)
15        tempo_cattura = zeros(10000,1)
16        tempo_svuotamento = zeros(10000,1)
17        tempo_calcolo_fps = zeros(10000,1)
18        optionTable %tabella sheet nel popup
19        stopwhile
20        stop_acq
21
22        end
23
24        methods (Access = public)
25
26        function salvataggio(app,limite)
27
```

```
28          contaciclo = 0;
29          saveFolderPath = app.PERCORSOEditField.Value;
30          Iniziale_nome = char(app.NomeFile.Value);
31          bufferCounter = 0;
32          bufferSize = app.BufferSize.Value; % % Numero di immagini
               da accumulare prima di scriverle su disco Modifica
               secondo le tue esigenze
33          imageBuffer = cell(2,bufferSize); % Inizializza il buffer
34          tempo_iniziale = tic;
35          app.Scrittura.Color = 'red';
36          while app.ElapsedTime.Value < limite
37          try
38          esposizione = floor(app.g.ExposureTime);
39          catch
40          app.Messaggioerrori.Value = 'App.g.ExposureTime non trovato
               . Non trovo nessun oggetto gigecam';
41          end
42          inizio_iterazione = tic;
43          contaciclo = contaciclo + 1;
44          try
45          app.durata_desiderata(contaciclo) = 0.000001*app.g.
               ExposureTime*app.Sicurezzaduratataciclo.Value; % fisso la
               durata del ciclo al doppio dell'esposizione, ora so che
               acquisice frame diversi.
46          catch
47          app.Messaggioerrori.Value = 'App.g.ExposureTime non trovato
               . Non trovo nessun oggetto gigecam';
48          end
49          if strcmp(app.stop_acq, 'On')
50          app.stop_acq = 'Off';
51          app.LampStop.Color = 'white';
52          break
53          end
54          app.tempo_controlli(contaciclo) = toc(inizio_iterazione);
55          bufferCounter = bufferCounter + 1; % Incrementa il
               contatore del buffer
56          tempo_stamp = toc(tempo_iniziale);
57          imageBuffer{1,bufferCounter} = round(tempo_stamp,2);
58          try
59          imageBuffer{2,bufferCounter} = snapshot(app.g);
60          catch
61          if strcmp(app.stop_acq,'Off')
62          app.Messaggioerrori.Value = 'Non riesco ad acquisire frame
               e salvarli. Non trovo nessun oggetto gigecam, se non hai
               stoppato allora cade la linea.';
63          app.ListaCamere.Value = 'Nessuna Camera disponibile';
64          end
65          app.stop_acq = 'Off';
66          app.LampStop.Color = 'white';
67          disp('dentro catch del salvataggio. ');
68          break;
69          end
70          app.tempo_cattura(contaciclo) = toc(inizio_iterazione)-app.
               tempo_controlli(contaciclo);
71          if bufferCounter > bufferSize-1   % con il buffer pieno,
               scrivi le immagini su disco
```

```
72          inizio_svuotamento = tic;
73          if strcmp(app.TipoSalvataggio.Value, 'Continuo')
74          for i = 1:bufferSize
75          filename = fullfile(saveFolderPath, [Iniziale_nome '_'
                num2str(contaciclo - bufferSize + i) '_ts_' num2str(
                imageBuffer{1,i}) '_ET_' num2str(esposizione) '.bmp']);
76          imwrite(imageBuffer{2,i}, filename, 'bmp');
77          end
78          else
79          for i = 1:bufferSize
80          filename = fullfile(saveFolderPath, [Iniziale_nome '_'
                num2str(esposizione) '.bmp']);
81          imwrite(imageBuffer{2,i}, filename, 'bmp');
82          end
83          end
84          app.tempo_svuotamento(contaciclo) = toc(inizio_svuotamento)
                ;
85          deltat= zeros(1,bufferSize-1);
86          for i=1:bufferSize-1
87          deltat(i) = 1/(imageBuffer{1,i+1}-imageBuffer{1,i});
88          end
89          app.Framepersecond.Value = mean(deltat);
90          bufferCounter = 0;
91          app.tempo_calcolo_fps(contaciclo)= toc(inizio_svuotamento)-
                app.tempo_svuotamento(contaciclo);
92          end
93          app.tempo_pre_attesa(contaciclo) = toc(inizio_iterazione);
94          durata desiderata
95          tempo_di_attesa = app.durata_desiderata(contaciclo) - app.
                tempo_pre_attesa(contaciclo);
96          if tempo_di_attesa > 0
97          pause(tempo_di_attesa);
98          end
99          app.ElapsedTime.Value = toc(tempo_iniziale);
100         app.tempoIterazione(contaciclo) = toc(inizio_iterazione);
101         end
102         tic();
103         app.ElapsedTime.Value = 0;
104         app.Scrittura.Color = 'white';
105         T = table(app.tempo_controlli,app.tempo_cattura, app.
                tempo_svuotamento,app.tempo_calcolo_fps, app.
                tempo_pre_attesa,...
106         app.tempoIterazione,app.durata_desiderata, 'VariableNames',
                {'Controlli␣Preliminari␣e␣Ts','Riempimento␣Buffer', ...
107                 'Svuotamento␣Buffer','Calcolo␣FPS','Pre␣attesa','
                        Iterazione','durata␣desiderata'});
108         nome_tabella = fullfile(saveFolderPath, ['log_file_'
                Iniziale_nome '.xlsx']);
109         writetable(T, nome_tabella);
110         app.tempo_controlli = zeros(10000,1);
111         app.tempo_cattura = zeros(10000,1);
112         app.tempo_svuotamento = zeros(10000,1);
113         app.tempo_calcolo_fps = zeros(10000,1);
114         app.tempo_pre_attesa = zeros(10000,1);
115         app.tempoIterazione = zeros(10000,1);
116         end
```

```
117
118         function aggiungiriga(~,tabella)
119         currentdata = tabella.Data;
120         newRow = [0, 0];
121         newdata = [currentdata; newRow];
122         tabella.Data = newdata;
123         end
124
125         function aggiornaslider(app,value)
126         if value <= 349000/2
127         app.Slider.MajorTicks = [value/2 value 2*value];
128         app.Slider.MajorTickLabels = app.Slider.MajorTicks + "";
129         app.Slider.Limits = [value/2 2*value];
130         else
131         app.Slider.MajorTicks = [(2*value)-349000 value 349000];
132         app.Slider.MajorTickLabels = app.Slider.MajorTicks + "";
133         app.Slider.Limits = [(2*value)-349000 349000];
134         end
135         end
136
137         function statistiche(app)
138         app.TabellaStatistiche.Data = [];
139         tbox = app.TabellaBox.Data;
140         colonne le x
141         y = tbox(:,2);
142         x = tbox(:,1);
143         l = app.LatoBox.Value;
144         foto = app.immagine;
145         for j=1:size(tbox,1)
146         riga = y(j);
147         colonna = x(j);
148         box = foto(riga:riga+l, colonna:colonna+l);
149         grad_box = imgradient(box, 'sobel');
150         media_grad = mean(grad_box(:));
151         massimo= max(box(:));
152         minimo = min(box(:));
153         nuovaRigaDiDati = [j, floor(massimo),floor(minimo),floor(
               media_grad)];
154         currentdata = app.TabellaStatistiche.Data;
155         app.TabellaStatistiche.Data = [currentdata; nuovaRigaDiDati
               ];
156         end
157         end
158
159         function reset(app)
160         disp('stop␣premuto␣salvataggio␣stoppato');
161         app.ConnectSwitch.Value = 'Off';
162         connetti(app);
163         pause(1);
164         app.ConnectSwitch.Value = 'On';
165         connetti(app);
166         pause(2);
167         app.PreviewSwitch.Value = 'On';
168
169         end
170
```

```
171         function connetti(app)
172         app.TabellaET.Data = [500,1; 1000,1; 1500,1; 2000,1;
                2500,1; 3000,1; 3500,1; 4000,1; 4500,1];
173         app.TabellaBox.Data = [150,50; 150,540; 150, 1040];
174         app.LatoBox.Value = 30;
175         disp('chiamata␣alla␣funzione␣connetti');
176         if strcmp(app.ConnectSwitch.Value, 'On')
177         try
178         app.g = gigecam;
179         app.CameraConnessa.Value = app.g.DeviceModelName;
180         formato = app.PixelFormat.Value;
181         app.g.PixelFormat =  formato;
182         app.g.Timeout = 10; %era 5000 (millisecondi) bisogna capire
                il valore che non mi da errore (sotto i 1000 mi dava
                errore)
183         app.fDDropDown.Value = app.fDDropDown.Items{1}; %questo ci
                serve per riportare l'apertura nel foglio di salvataggio
184         app.Messaggioerrori.Value = '';
185         disp('connetti␣su␣on');
186         catch
187         app.Messaggioerrori.Value = 'Non␣vedo␣nessuna␣camera';
188         app.ConnectSwitch.Value = 'Off';
189         end
190         else
191         delete(app.g);
192         clear app.g;
193         app.CameraConnessa.Value = '';
194         app.PreviewSwitch.Value = 'Off';
195         disp('connetti␣off␣e␣preview␣off');
196         end
197         end
198
199         end
200
201         methods (Access = private)
202
203         function PixelFormatValueChanged(app, event)
204         value = app.PixelFormat.Value;
205         app.g.PixelFormat = value;
206         end
207
208         % Value changed function: ConnectSwitch
209         function Connetti(app, event)
210         connetti(app);
211         end
212
213         % Value changed function: PreviewSwitch
214         function AvviaPreview(app, event)
215         initial_value = app.PreviewSwitch.Value;
216         if strcmp(initial_value, 'On')
217         try
218         app.g.ExposureTime =10000;
219         app.Messaggioerrori.Value = '';
220         catch
221         app.Messaggioerrori.Value = 'Non␣riesco␣a␣settare␣ET.␣
                Riprova␣a␣lanciare␣la␣preview.';
```

```
222         end
223         inizial_value = app.g.ExposureTime;
224         app.valore_ET.Value = inizial_value;
225         aggiornaslider(app,inizial_value);
226         app.Slider.Value = inizial_value;
227         % imposta la profondity di bit nella visualizzazione per
                plottare correttamente l'immagine
228         if strcmp(app.PixelFormat.Value, 'Mono8')
229         range = [0 255];
230         elseif strcmp(app.PixelFormat.Value, 'Mono10')
231         range = [0 1023];
232         else
233         range = [0 4095];
234         end
235         % prendo i valori della tabella dei box e li plotto sulla
                preview
236         tbox = app.TabellaBox.Data;
237         lato= app.LatoBox.Value;
238         nBox = size(tbox,1);
239         rectangles = zeros(nBox,4);
240         for i=1:nBox
241         rectangles(i,1)= tbox(i,1);
242         rectangles(i,2)= tbox(i,2);
243         rectangles(i,3)=lato;
244         rectangles(i,4)=lato;
245         end
246         disp('fuori␣il␣while␣preview');
247         while true
248         value = app.PreviewSwitch.Value;
249         disp('dentro␣il␣while␣preview');
250         if strcmp(value, 'Off')
251         break;  % Esci dal ciclo se  valore "Off"
252         end
253         try
254         % disp('preview pre');
255         app.immagine = snapshot(app.g);
256         % disp('preview post');
257         catch
258         app.Messaggioerrori.Value = 'Non␣riesco␣ad␣acquisire␣frame.
                ␣Chiudo␣la␣preview.';
259         app.ListaCamere.Value = 'Nessuna␣Camera␣disponibile';
260         app.PreviewSwitch.Value = 'Off';
261         disp('preview␣dentro␣catch');
262         break;
263         end
264         % Inserisci i rettangoli nell'immagine
265         if strcmp(app.RotazioneCamera.Value,'0') %posizione normale
266         live = insertShape(app.immagine, 'Rectangle', rectangles, '
                Color', 'red', 'LineWidth', 3);
267         elseif strcmp(app.RotazioneCamera.Value,'1') %ruotata a
                destra
268         live = insertShape(app.immagine, 'Rectangle', rectangles, '
                Color', 'red', 'LineWidth', 3);
269         live = imrotate(live,-90);
270         else  % Ruotata a sinistra
```

```
271         live = insertShape(app.immagine, 'Rectangle', rectangles, '
                Color', 'red', 'LineWidth', 3);
272         live = imrotate(live,90);
273         end
274         imshow(live, 'Parent', app.UIAxesPreview, 'DisplayRange',
                range);
275         drawnow;
276         end
277         end
278         end
279
280         % Value changed function: Slider
281         function SliderTempoEsposizione(app, event)
282         value = app.Slider.Value;
283         app.valore_ET.Value = value;
284         try
285         app.g.ExposureTime = app.valore_ET.Value;
286         aggiornaslider(app,value);
287         catch
288         app.Messaggioerrori.Value = 'Il tempo di esposizione non si
                cambia tramite slider. Riprova.';
289         end
290         statistiche(app);
291         end
292
293         % Value changed function: valore_ET
294         function ValoreET(app, event)
295         value = app.valore_ET.Value;
296         try
297         app.g.ExposureTime = app.valore_ET.Value;
298         statistiche(app);
299         catch
300         app.Messaggioerrori.Value = 'il tempo di esposizione non si
                cambia tramite valore. Riprova.';
301         end
302         try
303         app.Slider.Value = value;
304         catch
305         app.Slider.Limits = [0 349000];
306         app.Slider.Value = value;
307         end
308         aggiornaslider(app,value);
309         end
310
311         % Button pushed function: AddboxButton
312         function AddboxButtonPushed(app, event)
313         aggiungiriga(app,app.TabellaBox);
314         end
315
316         % Button pushed function: StatisticsonwhiterefButton
317         function StatisticsonwhiterefButtonPushed(app, event)
318         statistiche(app);
319         end
320
321         % Button pushed function: FolderButton
322         function CartelladiSalvataggio(app, event)
```

```
323         saveFolder = uigetdir();
324         app.PERCORSOEditField.Value = saveFolder;
325         if ~exist(app.PERCORSOEditField.Value, 'dir')
326         mkdir(app.PERCORSOEditField.Value);
327         end
328         end
329
330         % Button pushed function: StartButton
331         function StartButtonPushed(app, event)
332         salvataggio(app,3600);
333         end
334
335         % Button pushed function: StopButton
336         function StopButtonPushed(app, event)
337         app.stop_acq = 'On';
338         app.LampStop.Color = 'red';
339         reset(app);
340         end
341
342         % Value changed function: StartLookUpSwitch
343         function StartLookUpSwitchValueChanged(app, event)
344         % value = app.StartLookUpSwitch.Value;
345         while true
346         value = app.StartLookUpSwitch.Value;
347         % app.ScritturaLamp.Color = 'green';
348         if strcmp(value, 'Off')
349         break
350         end
351         % Prendo i dati dalla tabella che si vede a schermo
352         t = app.TabellaET.Data;
353         % Ciclo esterno su tutte le righe della tabella
354         for i = 1:size(t, 1)
355         % Prendo l'esposizione in microsecondi dalla tabella
356         exposureTime = t(i,1);
357         % Imposto sulla camera il valore di esposizione in
               microsecondi
358         try
359         app.g.ExposureTime = exposureTime;
360         catch
361         disp('Il tempo di esposizione non si cambia, aspetto 0.5 s
               e riprovo');
362         % metto in pausa per mezzo secondo
363         pause(0.5);
364         app.g.ExposureTime = exposureTime;
365         end
366         % Aggiorno il display in microsecondi
367         app.valore_ET.Value = app.g.ExposureTime;
368         % Durata ad una data esposizione (s)
369         durata_in_secondi = t(i, 2);
370         salvataggio(app,durata_in_secondi);
371         end
372         app.StartLookUpSwitch.Value = 'Off'; % finito il ciclo
               spengo il pulsante ed esco
373         disp('ciclo terminato.');
374         end
375         end
```

```
376
377         % Button pushed function: addRowButton
378         function AggiungiRigaLookup(app, event)
379         currentdata = app.TabellaET.Data;
380         newRow = [0, 0];
381         newdata = [currentdata; newRow];
382         app.TabellaET.Data = newdata;
383         end
384
385         % Button pushed function: ComponiLookupButton
386         function ComponiLookup(app, event)
387         inizio = app.FromEditField.Value;
388         fine = app.ToEditField.Value;
389         passo = app.StepEditField.Value;
390         numero_steps= floor((fine-inizio)/passo);
391         disp(numero_steps);
392         app.TabellaET.Data=[];
393         for i=0:numero_steps
394         newROW=[inizio+(i*passo),1];
395         currentdata = app.TabellaET.Data;
396         app.TabellaET.Data = [currentdata;newROW];
397         end
398         end
399
400         % Button pushed function: FindCameraButton
401         function FindCameraButtonPushed(app, event)
402         tabella_camera = gigecamlist;
403         try
404         nome_camera = tabella_camera{1,1};
405         nome = nome_camera{1,1};
406         app.ListaCamere.Value = nome;
407         catch
408         app.ListaCamere.Value = 'Nessuna␣camera␣disponibile';
409         end
410         end
411         end
412         end
```

Appendix D

Class Hypercube

```
1
2         classdef HypeCube
3         methods(Static)
4         function crea_dataset(banda\_start, banda\_end, num_slices,
              modo, file_da_cercare)
5         mainFolder = uigetdir(pwd, 'Seleziona la cartella
             principale che contiene tutte le sottocartelle con le
             acquisizioni');
6         if mainFolder == 0
7         disp('Nessuna cartella selezionata, interruzione del
             programma.');
8         return;
9         end
10        folders = dir(mainFolder);
11        folders = folders([folders.isdir]);
12        folders = folders(~ismember({folders.name}, {'.', '..'}));
13        if nargin>3
14        if strcmp(modo,"controllo")
15        HypeCube.controllo_cartelle(mainFolder, folders,
             file_da_cercare);
16        return;
17        end
18        end
19        fprintf('Sottocartelle trovate:\n');
20        for i = 1:length(folders)
21        fprintf('%d - %s\n', i, folders(i).name);
22        end
23        selectedIdx = input('Inserisci i numeri delle sottocartelle
             da elaborare (es: [1 3 5] o [1:10]):');
24        indices = round(linspace(banda\_start, banda\_end,
             num_slices));
25        percorso\_cartella\_DS = uigetdir(pwd, 'Seleziona la
             cartella del dataset. deve contenere la parte input e la
             parte output');
26        for i = selectedIdx
27        nome\_sotto\_cartella = folders(i).name;
28        [newStruct, nome\_struttura] = HypeCube.
             importa_iperspettrali\_segmentate_normalizzate\_HSI(
             mainFolder,nome_sotto_cartella, indices);
```

```
29          HypeCube.saveImagesAndVariable(percorso_cartella_DS,
                nome_struttura, newStruct.RGB_blocchi,newStruct.
                Riflettanza\_blocchi);
30          clear newStruct;
31          disp(i);
32          end
33          end
34          function controllo_cartelle(mainFolder,folders)
35          selectedIdx =[];
36          ignoredIdx = [];
37          for i = 1:length(folders)
38          files = dir(fullfile(mainFolder, folders(i).name));
39                  nomiFile = {files.name};
40          if any(strcmp(nomiFile, file_da_cercare))
41          selectedIdx(1,end+1) = i;
42          else
43          ignoredIdx(end+1,1) = i;
44          end
45          end
46          disp(ignoredIdx);
47          disp(length(ignoredIdx));
48          disp(length(selectedIdx));
49          end
50          function saveImagesAndVariable(percorso_cartella_DS,
                nome_struttura, RGB_cellArray, Riflettanza_Array)
51          % Ottieni dimensioni del cell array
52          [numRows, numCols] = size(RGB_cellArray);
53          % Salva ogni immagine in formato PNG
54          for r = 1:numRows
55          for c = 1:numCols
56          indice = (r-1)*numCols + c;
57          indice = num2str(indice);
58          nome_RGB = [nome_struttura,'_n_',indice,'.png'];
59          filename_RGB = fullfile(percorso_cartella_DS,'Inputs_RGB',
                nome_RGB);
60          img = RGB_cellArray{r, c};
61          imwrite(img, filename_RGB);
62          % Salva la variabile in formato .mat
63          % nome_HSI = [nome_struttura,'_n_',indice];
64          nome_HSI_completo = [nome_struttura,'_n_',indice,'.mat'];
65          matPath = fullfile(percorso_cartella_DS,'Outputs_HSI',
                nome_HSI_completo);
66          Riflettanza = Riflettanza_Array{r,c};
67          save(matPath, 'Riflettanza');
68          end
69          end
70          end
71          %% Questa funzione entra nella cartella della singola
                acquisizione e genera le struttura newStruct che
                contiene Raw, White, Reflectance, RGB e
                Riflettanza_Blocchi e RGB_Blocchi
72          % mainfolder: il percorso che porta alla cartella
                principale che contiene tutte le sottocartelle con le
                varie acquisizioni
73          % nome_sotto_cartella: il nome (stringa) della cartella con
                 cui lavorare
```

```
74          function [newStruct,nome_struttura] = importa_iperspettrali
                (mainFolder,nome_sotto_cartella, indices)
75          settings = fullfile(mainFolder, nome_sotto_cartella,'
                settings.txt');
76          nome_struttura = HypeCube.legginome(settings);
77          try
78          Raw = hypercube(fullfile(mainFolder, nome_sotto_cartella,'
                raw_0'), fullfile(mainFolder, nome_sotto_cartella,'raw_0
                .hdr'));
79          catch
80          disp('Non␣riesco␣a␣trovare␣il␣RAW.')
81          return;
82          end
83          try
84          Dark = hypercube(fullfile(mainFolder, nome_sotto_cartella,'
                darkReference'), fullfile(mainFolder,
                nome_sotto_cartella,'darkReference.hdr'));
85          catch
86          disp('Non␣riesco␣a␣trovare␣il␣Dark.')
87          return;
88          end
89          try
90          White = hypercube(fullfile(mainFolder, nome_sotto_cartella,
                'whiteReference'), fullfile(mainFolder,
                nome_sotto_cartella,'whiteReference.hdr'));
91          catch
92          disp('Non␣riesco␣a␣trovare␣il␣White.')
93          return;
94          end
95
96          % converto gli ipercubi in double
97          Dark_d = double(Dark.DataCube);
98          Raw_d = double(Raw.DataCube);
99          White_d = double(White.DataCube);
100         Dark_d_sub = Dark_d(:, :, indices);
101         Raw_d_sub = Raw_d(:, :, indices);
102         White_d_sub = White_d(:, :, indices);
103         Riflettanza = (Raw_d_sub - Dark_d_sub) ./ (White_d_sub -
                Dark_d_sub);
104         Riflettanza(Riflettanza < 0) = 0;
105         newStruct = struct();
106         newStruct.Raw = Raw_d;
107         newStruct.White = White_d;
108         newStruct.Dark = Dark_d;
109         newStruct.Riflettanza = Riflettanza;
110         newStruct.Riflettanza\_blocchi = HypeCube.estrai\_blocchi(
                Riflettanza);
111         newStruct.RGB = imread(fullfile(mainFolder,
                nome_sotto_cartella,'image_0.png'));
112         [H,~,~]= size(newStruct.RGB);
113         if H==640
114         newStruct.RGB = rot90(newStruct.RGB, 1);
115         end
116         newStruct.RGB_blocchi = HypeCube.estrai_blocchi(newStruct.
                RGB);
117         end
```

```
118         function [newStruct,nome_struttura] =
                importa_iperspettrali_segmentate(mainFolder,
                nome_sotto_cartella, indices)
119         nome_struttura = nome_sotto_cartella;
120         a=load(fullfile(mainFolder,nome_sotto_cartella,'raw.mat'));
121         Dark_d = double(a.dark_I16);
122         Raw_d = double(a.raw_I16_chicchi);
123         White_d = double(a.white_I16);
124
125         % Estrazione delle fette
126         Dark_d_sub = Dark_d(:, :, indices);
127         Raw_d_sub = Raw_d(:, :, indices);
128         White_d_sub = White_d(:, :, indices);
129
130         % calcolo la riflettanza
131         Riflettanza = (Raw_d_sub - Dark_d_sub) ./ (White_d_sub -
                Dark_d_sub);
132         % elimino i valori negativi portandoli a 0
133         % (magari per determinate posizioni il non arriva nessun
                segnale
134         Riflettanza(Riflettanza < 0) = 0;
135         newStruct = struct();
136         newStruct.Raw = Raw_d;
137         newStruct.White = White_d;
138         newStruct.Dark = Dark_d;
139         newStruct.Riflettanza = Riflettanza;
140         newStruct.Riflettanza_blocchi = HypeCube.estrai_blocchi(
                Riflettanza);
141         newStruct.RGB = imread(fullfile(mainFolder,
                nome_sotto_cartella,'image_seg.png'));
142         newStruct.RGB_blocchi = HypeCube.estrai_blocchi(newStruct.
                RGB);
143         end
144         function [newStruct,nome_struttura] =
                importa_iperspettrali_segmentate_HSI(mainFolder,
                nome_sotto_cartella, indices)
145         nome_struttura = nome_sotto_cartella;
146         a=load(fullfile(mainFolder,nome_sotto_cartella,'raw.mat'));
147         Dark_d = double(a.dark_I16);
148         Raw_d = double(a.raw_I16_chicchi);
149         Dark_d_sub = Dark_d(:, :, indices);
150         Raw_d_sub = Raw_d(:, :, indices);
151         Intensita_normalizzata = (Raw_d_sub - Dark_d_sub) ./ 4095;
152         Intensita_normalizzata( Intensita_normalizzata < 0) = 0;
153         % Inizializza la struttura vuota
154         newStruct = struct();
155         % Riempio la struttura con i vari campi
156         newStruct.Raw = Raw_d;
157         newStruct.White = White_d;
158         newStruct.Dark = Dark_d;
159         newStruct.Intensita_normalizzata = Intensita_normalizzata;
160         newStruct.Intensita_normalizzata_blocchi = HypeCube.
                estrai_blocchi(Intensita_normalizzata);
161         newStruct.RGB = imread(fullfile(mainFolder,
                nome_sotto_cartella,'image_seg.png'));
```

```
162         newStruct.RGB_blocchi = HypeCube.estrai_blocchi(newStruct.
                RGB);
163         end
164         function nome = legginome(filename)
165         fid = fopen(filename, 'r');
166         if fid == -1
167         error('Impossibile␣aprire␣il␣file.');
168         end
169         % Leggi il file riga per riga
170         while ~feof(fid)
171         line = fgetl(fid); % Legge una riga dal file
172         % Controlla se la riga contiene "Prefix ="
173         if contains(line, 'Prefix␣=')
174         tokens = regexp(line, 'Prefix␣=␣(\S+)', 'tokens'); % \S+
                prende una parola senza spazi
175         if ~isempty(tokens)
176         nome = tokens{1}{1};
177         end
178         end
179         end
180         % Chiudi il file
181         fclose(fid);
182         end
183         % Questa funzione crea i blocchi 40x40 partendo dall'
                immagine in ingresso.
184         function blocks = estrai_blocchi(img)
185         % Definisci la dimensione dei blocchi
186         block_size = 40;
187         % Calcola il numero massimo di blocchi interi
188         righe = size(img,1);
189         colonne =size(img,2);
190         num_blocks_row = floor(righe / block_size);
191         num_blocks_col = floor(colonne / block_size);
192         blocks = cell(num_blocks_row,num_blocks_col);
193         for i = 1:num_blocks_row
194         for j = 1:num_blocks_col
195         row_start = (i-1) * block_size + 1;
196         row_end = i * block_size;
197         col_start = (j-1) * block_size + 1;
198         col_end = j * block_size;
199         blocks{i, j} = img(row_start:row_end, col_start:col_end, :)
                ;
200         end
201         end
202         end
203
204
205
206         'giorno_x_y'
207         function lista_generale = scegli_dati
208         % Scelta interattiva della cartella contenente le
                sottocartelle
209         cartella_iniziale = uigetdir([], 'Seleziona␣la␣cartella␣con
                ␣le␣sottocartelle␣dove␣stanno␣i␣dati');
210         % Lettura dei nomi delle sottocartelle
211         elenco = dir(cartella_iniziale);
```

```
212         elenco = elenco([elenco.isdir]); % Solo cartelle
213         elenco = elenco(~ismember({elenco.name}, {'.','..'}));
214         lista_generale = cell(length(elenco),1);
215         for i = 1:length(elenco)
216         nome = elenco(i).name;
217         parti = strsplit(nome, '_');
218         radice = strjoin(parti(1:3), '_');
219         lista_generale{i} = radice;
220         end
221         end
222         function [sotto_lista,lista_rimanente] = suddividi_dataset(
                lista_generale,percentuale)
223         if percentuale < 1
224         num_test = max(1, round(percentuale* length(lista_generale)
                ));
225         else
226         num_test = percentuale;
227         end
228         indici = randperm(length(lista_generale), num_test);
229         sotto_lista = lista_generale(indici);
230         sotto_lista = ordina(sotto_lista);
231         function array_sorted = ordina(array)
232         n = numel(array);
233         xy = zeros(n, 2);
234         for i = 1:n
235         str = array{i};
236         parts = split(str, '_');  % cell array del tipo {'giorno','
                x','y'}
237         x = str2double(parts{2});
238         y = str2double(parts{3});
239         xy(i, :) = [x, y];
240         end
241         [~, sortIdx] = sortrows(xy, [1 2]);
242         array_sorted = array(sortIdx);
243         end
244         lista_rimanente = HypeCube.rimanenze(lista_generale,
                sotto_lista);
245         end
246         % lista_rimanente = HypeCube.rimanenze(lista_generale,
                sotto_lista);
247         function lista_rimanente = rimanenze(lista_generale,
                sotto_lista)
248         % Mantiene l'ordine originale di lista_dataset
249         mask = ~ismember(lista_generale, sotto_lista);
250         lista_rimanente = lista_generale(mask);
251         end
252
253
254         %% Una volta generata la struttura lista andiamo a copiare
                le varie parti nelle cartelle che vogliamo
255
256         % Quindi a seconda della lista andiamo a copiare prima
                Inputs e poi Outputs:
257
258         % HypeCube.copia_file(lista_train); la uso puntando prima
                Inputs (delle cartelle Dataset e TrainSet)
```

```
% HypeCube.copia_file(lista_val);
% HypeCube.copia_file(lista_test);
function copia_file(sotto_lista)
% Scelta della seconda cartella contenente i file da copiare
cartella_dati = uigetdir([], 'Seleziona la cartella di partenza, contenente i dati da copiare');
% Scelta della cartella finale di destinazione
cartella_output = uigetdir([], 'Seleziona la cartella di arrivo, dove copiare i file');
% Lettura dei file nella cartella dati
files_dati = dir(fullfile(cartella_dati, '*'));
files_dati = files_dati(~[files_dati.isdir]); % Solo file
% Per ogni radice in sotto_lista, copiare i file corrispondenti
for i = 1:length(sotto_lista)
radice = sotto_lista{i};
pattern = [radice '_']; % Per cercare i file che iniziano con questa radice + underscore
for j = 1:length(files_dati)
nome_file = files_dati(j).name;
if startsWith(nome_file, pattern)
src = fullfile(cartella_dati, nome_file);
dest = fullfile(cartella_output, nome_file);
copyfile(src, dest);
end
end
end
end
function copia_cartelle(sotto_lista)
cartella\_dati = uigetdir();
cartella_output = uigetdir();
folders = dir(cartella\_dati);
folders = folders([folders.isdir]);
folders = folders(~ismember({folders.name}, {'.', '..'}));
for i = 1:length(sotto_lista)
radice = sotto_lista{i};
pattern = [radice '_'];
for j = 1:length(folders)
nome_file = folders(j).name;
if startsWith(nome_file, pattern)
src = fullfile(cartella_dati, nome_file);
dest = fullfile(cartella_output, nome_file);
copyfile(src, dest);
end
end
end
end
function [meanR , meanG, meanB, stdR, stdG, stdB] = standard_scaler()
imageDir = uigetdir(pwd, 'Seleziona la cartella principale');
imageFiles = dir(fullfile(imageDir, '*.png'));
sumR = 0; sumG = 0; sumB = 0;
sumR2 = 0; sumG2 = 0; sumB2 = 0;
numPixels = 0;
```

```
        for k = 1:length(imageFiles)
        imgPath = fullfile(imageDir, imageFiles(k).name);
        img = imread(imgPath);
        img = double(img);
        R = img(:,:,1);
        G = img(:,:,2);
        B = img(:,:,3);
        sumR = sumR + sum(R(:));
        sumG = sumG + sum(G(:));
        sumB = sumB + sum(B(:));

        sumR2 = sumR2 + sum(R(:).^2);
        sumG2 = sumG2 + sum(G(:).^2);
        sumB2 = sumB2 + sum(B(:).^2);
        numPixels = numPixels + numel(R);
        end
        meanR = sumR / numPixels;
        meanG = sumG / numPixels;
        meanB = sumB / numPixels;

        stdR = sqrt((sumR2 / numPixels) - meanR^2);
        stdG = sqrt((sumG2 / numPixels) - meanG^2);
        stdB = sqrt((sumB2 / numPixels) - meanB^2);
        end
        function [trainDS, valDS ] = carica_data(nome_dataset)
        [traininputFolder, ~, trainoutputFolder, ~] = HypeCube.
            ottieni_ipercorso('Seleziona la cartella del TrainSet (
            che contiene le sottocartelle Inputs_RGB e Outputs_HSI).
            ');
        [valinputFolder, ~, valoutputFolder, ~] = HypeCube.
            ottieni_ipercorso('Seleziona la cartella del ValidSet (
            che contiene le sottocartelle Inputs_RGB e Outputs_HSI).
            ');
        trainInputDS = imageDatastore(traininputFolder, '
            FileExtensions', {'.png', '.jpg', '.tif'}, "ReadFcn",  @
            (x) RGB_ReadFcn(x));
        valInputDS = imageDatastore(valinputFolder, 'FileExtensions
            ', {'.png', '.jpg', '.tif'}, "ReadFcn",  @(x)
            RGB_ReadFcn(x)); % sta funzione di lettura divide per
            255 i dati in ingresso --> scalati tra 0 e 1
        trainOutputDS = fileDatastore(trainoutputFolder, "ReadFcn",
             @(x) HS_ReadFcn(x), "IncludeSubfolders", false, "
            FileExtensions", ".mat");
        valOutputDS = fileDatastore(valoutputFolder, "ReadFcn", @(x
            ) HS_ReadFcn(x), "IncludeSubfolders", false, "
            FileExtensions", ".mat");
        trainDS = combine(trainInputDS, trainOutputDS);
        valDS = combine(valInputDS, valOutputDS);
        cartella_datastore = uigetdir( "current working directory"
            ,'Seleziona la cartella di salvataggio del dataset.');
        nome_file =[nome_dataset,'.mat'];
        file = fullfile(cartella_datastore,nome_file);
        file = string(file);
        file = file(1,1);
        save(file, "trainDS", "valDS");
```

```
347         disp("Dataset suddiviso correttamente in training e
                validation!");
348         end
349         function [inputFolder , inputFiles,  outputFolder,
                outputFiles] = ottieni_ipercorso(testo)
350         cartella_dataset = uigetdir( "current working directory" ,
                testo);
351         inputFolder = fullfile(cartella_dataset ,"Inputs_RGB");
                % Cartella delle immagini RGB
352         outputFolder = fullfile(cartella_dataset ,"Outputs_HSI");
                % Cartella delle immagini multispettrali
353         inputFiles = dir(fullfile(inputFolder, "*.png")); % sta
                funzione restituisce na struttura con all'interno i
                campi [name, folder,date,byte,isdir]
354         outputFiles = dir(fullfile(outputFolder, "*.mat"));
355         end
356         function curve_training = Estrazione_curve
357         Seleziona la cartella principale
358         mainFolder = uigetdir(pwd, 'Seleziona la cartella
                principale che contiene le varie strutture info');
359         if mainFolder == 0
360         disp('Nessuna cartella selezionata, interruzione del
                programma.');
361         return;
362         end
363         fileMat = dir(fullfile(mainFolder, '*.mat'));
364         [rig,~] = size(fileMat);
365         curve_training = struct();
366         curve_training.vettore_iterazioni =  zeros(rig,1);
367         curve_training.train_loss = zeros(rig,1);
368         curve_training.val_loss = zeros(rig,1);
369         for k = 1:length(fileMat)
370         nome_file = fullfile(mainFolder, fileMat(k).name);
371         dati = load(nome_file);
372         info = dati.info;
373         curve_training.vettore_lr(k) = info.BaseLearnRate;
374         curve_training.vettore_epoche(k) = info.Epoch;
375         curve_training.vettore_iterazioni(k) = info.Iteration;
376         curve_training.train_loss(k) = info.TrainingLoss;
377         curve_training.val_loss(k) = info.ValidationLoss;
378         end
379         [~, idx] = sort(curve_training.vettore_iterazioni);
380         curve_training.vettore_lr = curve_training.vettore_lr(idx);
381         curve_training.vettore_epoche =  curve_training.
                vettore_epoche(idx);
382         curve_training.vettore_iterazioni = curve_training.
                vettore_iterazioni(idx);
383         curve_training.train_loss = curve_training.train_loss(idx);
384         curve_training.val_loss = curve_training.val_loss(idx);
385         end
386         function plotta_curve(curve_training_normalizzata)
387         a = curve_training_normalizzata;
388         figure('Name','Loss e Validation');
389         plot(a.vettore_epoche, a.train_loss);
390         hold on
391         plot(a.vettore_epoche, a.val_loss);
```

```
392         legend('train loss','valodation loss');
393         figure('Name','Lr_ultima');plot(a.vettore_epoche, a.
                vettore_lr);
394         end
395         function plot_medie(newStruct)
396         iter = newStruct.vettore_iterazioni;
397         epoch = iter/377;
398         y1 = newStruct.train_loss;
399         y2 = newStruct.val_loss;
400         y3 = newStruct.vettore_lr;
401         if length(y2) == length(epoch)
402         x_val = linspace(iter(1), iter(end), length(epoch));  %
                Posizione delle epoche sulle iterazioni
403         else
404         x_val = iter;
405         end
406         figure('Name','Medie');
407         ax1 = axes('Position', [0.13 0.26 0.775 0.65]);
408         hold(ax1, 'on');
409         plot(ax1, iter, y3, '+', 'DisplayName', 'Validation Loss',
                'Color', 'k');
410         p1 = polyfit(iter, y1, 1);
411         trend_y1 = polyval(p1, iter);
412         plot(ax1, iter, trend_y1, '--k', 'LineWidth', 2, '
                DisplayName', 'Train Loss slope');
413         if length(y2) == length(x_val)
414         p2 = polyfit(x_val, y2, 1);
415         trend_y2 = polyval(p2, x_val);
416         plot(ax1, x_val, trend_y2, '-.r', 'LineWidth', 2, '
                DisplayName', 'Val Loss slope');
417         end
418         xlabel(ax1, 'Iterazione');
419         ylabel(ax1, 'Loss');
420         title(ax1, 'Loss function MRAE');
421         grid(ax1, 'on');
422         legend(ax1, 'show');
423         hold(ax1, 'off');
424                 ax2 = axes('Position', [0.13 0.13 0.775 0.01]);
425         set(ax2, 'XLim', [iter(1) iter(end)], 'YLim', [0 1], ...
426         'XTick', linspace(iter(1), iter(end), length(epoch)), ...
427         'XTickLabel', epoch, ...
428         'YTick', [], 'Box', 'off', 'Color', 'none');
429         xlabel(ax2, 'Epoca');
430
431         set(ax2, 'XAxisLocation', 'bottom');
432         set(ax2, 'Visible', 'on');
433         ax2.XColor = 'k'; % colore asse epoca
434
435         linkaxes([ax1 ax2], 'x');
436         end
437
438
439         function [newStruct, nome_originale] = modificastruttura
440         variabili = evalin('base', 'who');
441         fprintf('Variabli presenti nel workspace:\n');
442         for i = 1:length(variabili)
```

```
443         fprintf('%d - %s\n', i, variabili{i});
444         end
445         selectedIdx = input('Inserisci il numero della variabile da
                elaborare:');
446         nome_originale = variabili{selectedIdx};
447         newStruct = evalin('base',nome_originale); % a fare sta
                copia ci mette 0,02 s
448         evalin('base', ['clear ' nome_originale]);
449         end
450         function carica_struttura(banda\_start, banda\_end,
                num_slices)
451         %% Seleziona la cartella principale
452         mainFolder = uigetdir(pwd, 'Seleziona la cartella
                principale');
453         if mainFolder == 0
454         disp('Nessuna cartella selezionata, interruzione del
                programma.');
455         return;
456         end
457         folders = dir(mainFolder); % mi restituisce una struttura
                con diversi campi che sono degli array (quindi una
                struttura di array)
458         folders = folders([folders.isdir]); % Mantiene solo le
                cartelle
459         folders = folders(~ismember({folders.name}, {'.', '..'}));
                % Esclude '.' e '..'
460         fprintf('Sottocartelle trovate:\n');
461         for i = 1:length(folders)
462         fprintf('%d - %s\n', i, folders(i).name);
463         end
464         selectedIdx = input('Inserisci i numeri delle sottocartelle
                da elaborare (es: [1 3 5] o [1:10]): ');
465         indices = round(linspace(banda\_start, banda\_end,
                num_slices));
466         for i = selectedIdx
467         nome_sotto_cartella = folders(i).name;
468         [newStruct,nome_struttura] = HypeCube.importa_iperspettrali
                (mainFolder,nome_sotto_cartella, indices);
469         assignin('base', nome_struttura, newStruct);
470         end
471         end
472         function carica_struttura_da_nome(banda\_start, banda\_end,
                 num_slices,nome)
473         mainFolder = '1\_HYPE\_Chiara Dataset';
474         folders = dir(mainFolder);
475         folders = folders([folders.isdir]); % Mantiene solo le
                cartelle
476         folders = folders(~ismember({folders.name}, {'.', '..'}));
477         nomi = {folders.name};
478         indices = round(linspace(banda\_start, banda\_end, num\
                _slices));
479         nome\_sotto\_cartella = nomi(startsWith(nomi, nome));
480         [newStruct,nome\_struttura] = HypeCube.
                importa_iperspettrali(mainFolder,nome_sotto_cartella,
                indices);
481         assignin('base', nome_struttura, newStruct);
```

```
482         end
483         function coordinate = interactiveRectangle(immagine,nome\
                _file)
484         coordinate = struct( 'Id_imm', [], 'X', [], 'Y', []);
485         salvataggi = 1;
486         fig = figure;
487         imshow(immagine);
488         rect = rectangle('Position', [0, 0, 10, 10], 'EdgeColor', '
                r', 'LineWidth', 2);
489         uicontrol('Style', 'text', 'Position', [20, 20, 100, 20], '
                String', 'X:');
490         uicontrol('Style', 'text', 'Position', [200, 20, 100, 20],
                'String', 'Y:');
491         uicontrol('Style', 'text', 'Position', [380, 20, 100, 20],
                'String', 'Lato:');
492         uicontrol('Style', 'pushbutton', 'Position', [600, 20, 150,
                 20], 'String', 'Salva␣coordinate␣e␣rettangolo', ...
493         'Callback', @(~, ~) salvaCoordinate(rect));
494         set(fig, 'WindowButtonDownFcn', @(pos, ~)
                mouseClickCallback(rect));
495         xEdit = uicontrol('Style', 'edit', 'Position', [120, 20,
                50, 20], ...
496         'Callback', @(src, ~) updateRectangle(rect, src, 'x'));
497         yEdit = uicontrol('Style', 'edit', 'Position', [300, 20,
                50, 20], ...
498         'Callback', @(src, ~) updateRectangle(rect, src, 'y'));
499         latoEdit = uicontrol('Style', 'edit', 'Position', [480, 20,
                 50, 20], ...
500         'Callback', @(src, ~) updateRectangle(rect, src, 'lato'));
501         exitButton = uicontrol('Style', 'pushbutton', 'Position',
                [800, 20, 100, 20], 'String', 'Esci', ...
502         'Callback', @(~, ~) close(fig));
503         while ishandle(fig)
504         drawnow;
505         end
506         function mouseClickCallback(rect)
507         cp = get(gca, 'CurrentPoint');
508         xClick = cp(1, 1);
509         yClick = cp(1, 2);
510         xClick_corretto = floor(xClick / 5) * 5;
511         yClick_corretto = floor(yClick / 5) * 5;
512         pos = get(rect, 'Position');
513         pos(1) = xClick_corretto;
514         pos(2) = yClick_corretto;
515         set(rect, 'Position', pos);
516         set(xEdit, 'String', num2str(pos(1)));
517         set(yEdit, 'String', num2str(pos(2)));
518         end
519         function updateRectangle(rect, src, param)
520         pos = get(rect, 'Position');
521         val = str2double(get(src, 'String'));
522         if isnan(val) || val < 0
523         errordlg('Inserisci␣un␣valore␣valido.', 'Errore');
524         return;
525         end
526         switch param
```

```
527         case 'x'
528         pos(1) = val;
529         set(xEdit, 'String', num2str(val));
530         case 'y'
531         pos(2) = val;
532         set(yEdit, 'String', num2str(val));
533         case 'lato'
534         lato_corretto = floor(val / 5) * 5;
535         pos(3) = lato_corretto;
536         pos(4) = lato_corretto;
537         set(latoEdit, 'String', num2str(val));
538         end
539         set(rect, 'Position', pos);
540         end
541         function salvaCoordinate(rect)
542         pos = get(rect, 'Position');
543         coordinate.X = [coordinate.X; pos(1)];
544         coordinate.Y = [ coordinate.Y; pos(2)];
545         coordinate.Id_imm = [coordinate.Id_imm; nome_file];
546         line([pos(1) - 5, pos(1) + 5], [pos(2) - 5, pos(2) + 5], '
               Color', 'b', 'LineWidth', 1.5);
547         line([pos(1) - 5, pos(1) + 5], [pos(2) + 5, pos(2) - 5], '
               Color', 'b', 'LineWidth', 1.5);
548         text(pos(1), pos(2) + 10, num2str(salvataggi), 'Color', 'b'
               , 'FontSize', 10, 'FontWeight', 'bold');
549         colori (solo in corrispondenza delle x e del testo)
550         salvataggi = salvataggi + 1;
551         end
552
553         end
554         function n_blocco = coordinate_blocco(coordinate)
555         x = coordinate.X;
556         y = coordinate.Y;
557         [N,~] = size(x);
558         block_size = 40;
559         n_blocco = struct();
560         for i = 1:N
561         n_blocco.X(i) = ceil(x(i)/block_size);
562         n_blocco.Y(i) = ceil(y(i)/block_size);
563         end
564         end
565
566         function [blocco_RGB, blocco_IPER] =  estrai_blocco(
               struttura_foto, n_blocco)
567         [~,N] = size(n_blocco.X);
568         blocco_RGB = cell(N,1);
569         blocco_IPER = cell(N,1);
570         for i = 1:N
571         blocco_RGB{i} = struttura_foto.RGB_blocchi{n_blocco.Y(i),
               n_blocco.X(i)};
572         blocco_IPER{i}= struttura_foto.Riflettanza_blocchi{n_blocco
               .Y(i), n_blocco.X(i)};
573         end
574         end
575         function [imgsCell, matsCell, report] = carica_e_verifica()
```

```
imgDir = uigetdir(pwd,'seleziona la cartella con le immagini di test');
matDir = uigetdir(pwd,'seleziona la cartella i cubi di test');
pngFiles = dir(fullfile(imgDir, '*.png'));
N = numel(pngFiles);
imgsCell = cell(N, 2);

for i = 1:N
fpath = fullfile(pngFiles(i).folder, pngFiles(i).name);
img   = imread(fpath);
nomeBase = pngFiles(i).name;
nomeBase = nomeBase(1:end-4); % rimuove ".png"
imgsCell{i,1} = img;
imgsCell{i,2} = nomeBase;
end

%% --- MAT -> cell array {matrice, nome_base} ---
matFiles = dir(fullfile(matDir, '*.mat'));
M = numel(matFiles);
matsCell = cell(M, 2);

for i = 1:M
fpath = fullfile(matFiles(i).folder, matFiles(i).name);
S = load(fpath);              % carica struttura con le variabili del .mat
matrice = HypeCube.estrai_prima_matrice(S);
nomeBase = matFiles(i).name;
nomeBase = nomeBase(1:end-4); % rimuove ".mat"
matsCell{i,1} = matrice;
matsCell{i,2} = nomeBase;
end

if N > 0
[~, ordI] = sort(lower(string(imgsCell(:,2))));
imgsCell = imgsCell(ordI, :);
end
if M > 0
[~, ordM] = sort(lower(string(matsCell(:,2))));
matsCell = matsCell(ordM, :);
end
minLen = min(N, M);
uguali = false(minLen,1);
diffIdx = [];

for k = 1:minLen
uguali(k) = strcmp(imgsCell{k,2}, matsCell{k,2});
if ~uguali(k)
diffIdx(end+1) = k; %#ok<AGROW>
end
end

report = struct();
report.n_png  = N;
report.n_mat  = M;
report.tutti_corrispondono = (N == M) && all(uguali);
```

```
629         report.indici_non_corrispondenti = diffIdx;
630         report.extra_in_png = max(N - M, 0);
631         report.extra_in_mat = max(M - N, 0);
632 
633         if report.tutti_corrispondono
634         fprintf('OK: tutti i nomi corrispondono (%d elementi).\n',
               minLen);
635         else
636         fprintf('ATTENZIONE:\n');
637         if ~isempty(diffIdx)
638         fprintf('- %d righe non corrispondono (indici dopo l''
               ordinamento): %s\n', ...
639         numel(diffIdx), mat2str(diffIdx));
640         for d = diffIdx
641         fprintf('  riga %d -> PNG: "%s"  |  MAT: "%s"\n', d,
               imgsCell{d,2}, matsCell{d,2});
642         end
643         end
644         end
645         end
646         function A = estrai_prima_matrice(S)
647         campi = fieldnames(S);
648         if isempty(campi)
649         error('Il file .mat non contiene variabili.');
650         end
651         for i = 1:numel(campi)
652         val = S.(campi{i});
653         if (isnumeric(val) || islogical(val)) && ndims(val) >= 2
654         A = val;
655         return;
656         end
657         end
658         A = S.(campi{1});
659         end
660         function [prediction, net] = validation(riquadri\_RGB\
               _selezionati)
661         [file\_par\_in,percorso\_par\_in] = uigetfile('*.mat','
               Specifica i parametri con i quali fare il test, la
               prediction. File .mat');
662         Rerconstruction\Storico_Training\training 256 batch size
               learning rate 1e-4\Cartella Parametri\';
663 ........intero = fullfile(percorso\_par\_in,file\_par\_in);
664 ........params = load(intero);
665 ........net = params.net; % riacarico la rete (parametri e
    connessioni)
666 ........[n,~] = size(riquadri\_RGB\_selezionati);
667 ........prediction = cell(n,2);
668 ........for i=1:n
669 ........nuovo_riquadro = riquadri\_RGB\_selezionati{i,1};
670 ........nuovo_riquadro_scalato = HypeCube.RGB_ReadFcn_interna(
    nuovo_riquadro);
671 ........prediction{i,1} = predict(net, nuovo_riquadro_scalato);
672 ........prediction{i,2} = riquadri\_RGB\_selezionati{i,2};
673 ........end
674 ........end
675 ........function data = RGB_ReadFcn_interna(img)
```

```
␣␣␣␣␣␣␣␣img␣=␣double(img);
␣␣␣␣␣␣␣␣R␣=␣img(:,:,1);
␣␣␣␣␣␣␣␣G␣=␣img(:,:,2);
␣␣␣␣␣␣␣␣B␣=␣img(:,:,3);
␣␣␣␣␣␣␣␣HypeCube.standard_scaler
␣␣␣␣␣␣␣␣new_R␣=␣R/255;
␣␣␣␣␣␣␣␣new_G␣=␣G/255;
␣␣␣␣␣␣␣␣new_B␣=␣B/255;
␣␣␣␣␣␣␣␣data␣=␣cat(3,new_R,␣new_G␣,␣new_B);
␣␣␣␣␣␣␣␣end
␣␣␣␣␣␣␣␣function␣[m_media,m_mediana,m_std]␣=␣
    estrai_statistiche_predizione(A,x,y)
␣␣␣␣␣␣␣␣A(:,1)␣=␣cellfun(@double,A(:,1),'UniformOutput',false);
        pattern = sprintf('giorno_%d_%d',x,y);
        mask = cellfun(@(s) startsWith(s,pattern), A(:,2));
        C = A(mask,1);
        X4 = cat(4,C{:});
        Xperm=permute(X4,[3 1 2 4]);
        X = reshape(Xperm,31,[]).';
        radice = ['giorno_' num2str(x) '_' num2str(y) '_' ];
        m_media = mean(X,1,'omitnan');
        m_mediana = median(X,1,'omitnan');
        m_std = std(X,0,1,'omitnan');
        end
        function [Y, T , loss] = confronto_segmentati(groundT,
            prediction, indice)
        [Y,T] = HypeCube.scegli_pixel_segmentati(prediction{indice
            ,1}, groundT{indice,1},20,20);
        [loss, ~, ~, ~] = HypeCube.forwardLoss(Y,T);
        HypeCube.plot_spettri('Spettri_200', Y, T, loss);
        end
        function [Y,T]= scegli_pixel_segmentati(array_Y,array_T,
            riga_pixel,colonna_pixel)
        Y = squeeze(array_Y(riga_pixel,colonna_pixel,:));
        Y = cast(Y,'double');
        T = squeeze(array_T(riga_pixel,colonna_pixel,:));
        end
        function [Y, T, Y_norm, T_norm] = confronto(struttura,
            campo)
        [Y,T] = HypeCube.scegli_pixel(struttura.(campo), struttura.
            blocco_IPER,1,20,20);
        [loss, Y_norm, T_norm, loss_norm] = HypeCube.forwardLoss(Y,
            T);
        HypeCube.plot_spettri('Spettri', Y, T, loss);
        HypeCube.plot_spettri('Spettri␣norm',Y_norm, T_norm,
            loss_norm);
        end
        function [Y,T]= scegli_pixel(array_Y,array_T, Numero_punto,
             riga_pixel,colonna_pixel)
        Y = squeeze(array_Y{Numero_punto,1}(riga_pixel,
            colonna_pixel,:));
        Y = cast(Y,'double');
        T = squeeze(array_T{Numero_punto,1}(riga_pixel,
            colonna_pixel,:));
        end
```

```
720         function [loss, Y_norm, T_norm, loss_norm] = forwardLoss(Y,
                T)
721         epsilon = 1e-8;
722         if any(isinf(T(:)))
723         maxFiniteT = max(T(~isinf(T)));
724         T(isinf(T)) = maxFiniteT;
725         disp('Warning: Inf values in T replaced with max finite
                value');
726         end
727         T(isnan(T)) = 0.001;
728         Y(isnan(Y)) = 0;
729         loss_values = abs(Y - T)./(T + epsilon);
730         loss = mean(mean(loss_values, 1), 'all');
731         disp(['Loss attuale senza ReLu e normalizzazione: ',
                num2str(loss)]);
732         T_norm = (T - min(T(:))) / (max(T(:)) - min(T(:)) + epsilon
                );
733         Y_norm = (Y - min(Y(:))) / (max(Y(:)) - min(Y(:)) + epsilon
                );
734         loss_values_norm = abs(Y_norm - T_norm)./(T_norm + epsilon)
                ;
735         loss_norm = mean(mean(loss_values_norm, 1), 'all');
736         disp(['Loss attuale con normalizzazione: ', num2str(
                loss_norm)]);
737         if isnan(loss)  error('Loss  NaN! Controlla i dati di input
                e il modello.');
738         end
739         end
740         function plot_spettri(nom\e_plot, Y,T,loss)
741         [B,~] = size(Y);
742         figure('Name',nome_plot);
743         hold on;
744         plot(1:B, Y, '-o', 'DisplayName', 'predetto');
745         plot(1:B, T, '-s', 'DisplayName', 'GT', 'Color', 'r');
746         xlabel('Banda');  % Etichetta per l'asse x
747         ylabel('Value');  % Etichetta per l'asse y
748         str = ['loss = ' num2str(loss)];
749         text('Units','normalized','Position',[0.95 0.05], ...
750         'String', str, 'HorizontalAlignment','right')
751         title('Confronto tra spettro predetto e ground truth');
752         grid on;
753         legend('show'); % Mostra la legenda
754         hold off; % Rilascia il grafico per nuove figure
755         end
756         function Chicchi = test_puntuale(indice_nome)
757         suddivisione_dataset = load('suddivisione_dataset.mat');
758         lista_test = suddivisione_dataset.lista_test;
759         nome = lista\_test{indice\_nome};
760         HypeCube.carica\_struttura\_da\_nome(1,272,31,nome);
761         struttura\_caricata = evalin('base', nome);
762         Chicchi.coordinate = interactiveRectangle(struttura\
                _caricata.RGB,'nome\_ROIs');
763         Chicchi.n_blocco =  HypeCube.coordinate_blocco(Chicchi.
                coordinate);
764         [Chicchi.blocco\_RGB, Chicchi.blocco\_IPER] = HypeCube.
                estrai\_blocco(struttura\_caricata, Chicchi.n\_blocco);
```

```
765         Chicchi.prediction = HypeCube.validation(Chicchi.blocco_RGB
                );
766         [~, ~, Y\_norm, T\_norm] = HypeCube.confronto(Chicchi, '
                prediction');
767         HypeCube.plot\_spettri('Spettri norm',Y\_norm,T\_norm,22);
768         end
769         end
770         end
```

Appendix E

HSCRNN-R

```
[file\_data,percorso\_data] = uigetfile('*.mat','Specifica␣il␣datastore:','dataset.mat');
data = fullfile(percorso\_data,file\_data);
load(data);
percorso\_checkpoint = uigetdir('current␣working␣direcory','Specifica␣la␣cartella␣di␣salvataggio␣dei␣Parametri␣(DEVE␣ESSERE␣VUOTA).');
frequenza\_val = 50;
frequenza\_checkpoint = 68;
azione = questdlg('GPU␣o␣CPU?', 'Scegli␣HW␣training.', 'CPU', 'GPU', 'GPU');
if strcmp(azione, 'CPU')
options = trainingOptions("adam", ...
"CheckpointPath",percorso\_checkpoint,...
"CheckpointFrequency", frequenza\_checkpoint, ...
"CheckpointFrequencyUnit", "iteration",.....
"GradientDecayFactor",0.9,...
"SquaredGradientDecayFactor",0.999, ...
"Epsilon",  1e-8, ...
"InitialLearnRate", 1e-4, ...
"LearnRateSchedule","piecewise",...
"LearnRateDropFactor", 0.1, ...
"LearnRateDropPeriod", 100,...
"MiniBatchSize", 64, ...
"MaxEpochs", 1000, ...
"Shuffle", "every-epoch", ...
"ValidationData", valDS, ...
"ValidationFrequency", frequenza\_val, ...
"Plots", "training-progress", ...
"Verbose", false, ... % se mettere a display info tipo iterazione, epoca, tempo, training e validation loss
"VerboseFrequency", 10 ,...% ogni quante iteraizoni implementare il precedente output
"OutputFcn", @monitorLoss,...% Restituisce una struttura chiamata info: Epoch,Iteration,TimeElapsed,LearnRate,TrainingLoss,ValidationLoss,State
"OutputNetwork","best-validation-loss");
else
options = trainingOptions("adam", ...
"CheckpointPath", percorso\_checkpoint ,...
```

```
        "CheckpointFrequency", frequenza\_checkpoint, ...
        "CheckpointFrequencyUnit", "iteration",...
        "ExecutionEnvironment", "gpu",...
        "GradientDecayFactor",0.9,...
        "SquaredGradientDecayFactor",0.999, ...
        "Epsilon",  1e-8, ...
        "InitialLearnRate", 1e-3, ...
        "LearnRateSchedule","piecewise",...
        "LearnRateDropFactor", 0.1, ...
        "LearnRateDropPeriod", 100,...
        "MiniBatchSize", 64, ...
        "MaxEpochs", 1000, ...
        "Shuffle", "every-epoch", ...
        "ValidationData", valDS, ...
        "ValidationFrequency", frequenza_val, ...
        "Plots", "training-progress", ...
        "Verbose", false, ...
        "VerboseFrequency", 10,...
        "Verbose"
        "OutputFcn", @monitorLoss);
        end

        mode = questdlg('Con che parametri iniziamo il training?',
            'Scegli con quali parametri iniziare (o continuare) il
            training.', 'da checkpoint', 'da nuovo', 'da checkpoint'
            );
        if strcmp(mode,'da checkpoint')
        [file\_par\_in,percorso\_par\_in] = uigetfile('*.mat','
            Specifica i parametri iniziali con cui continuare il
            training. File .mat','RETE\_HSCNN\_R.mat');
        intero = fullfile(percorso\_par\_in,file\_par\_in);
        params = load(intero);
        net = params.net; % riacarico la rete (parametri e
            connessioni)
        net\_2 = layerGraph(net);
        net = trainNetwork(trainDS, net\_2, options);
        elseif strcmp(mode,'da nuovo')
        lgraph = layerGraph();
        tempLayers = [
        imageInputLayer([40 40 3],"Name","RGB\_input",'
            Normalization','none')
        convolution2dLayer([3 3],64,"Name","C3\_64\_1","Padding","
            same")];
        lgraph = addLayers(lgraph,tempLayers);
        tempLayers = [
        convolution2dLayer([3 3],64,"Name","C3\_64","Padding","same
            ")
        reluLayer("Name","relu")
        convolution2dLayer([3 3],64,"Name","C3\_64\_2","Padding","
            same")]
        lgraph = addLayers(lgraph,tempLayers);
        tempLayers = [
        additionLayer(2,"Name","addition")
        reluLayer("Name","relu\_1")];
        lgraph = addLayers(lgraph,tempLayers);
        tempLayers = [
```

```
        convolution2dLayer([3 3],64,"Name","C3\_64\_3","Padding","
           same")
        reluLayer("Name","relu\_2")
        convolution2dLayer([3 3],64,"Name","C3\_64\_4","Padding","
           same")];
        lgraph = addLayers(lgraph,tempLayers);
        tempLayers = [
        additionLayer(2,"Name","addition\_1")
        reluLayer("Name","relu\_1\_1")];
        lgraph = addLayers(lgraph,tempLayers);
        tempLayers = [
        convolution2dLayer([3 3],64,"Name","C3\_64\_5","Padding","
           same")
        reluLayer("Name","relu_3")
        convolution2dLayer([3 3],64,"Name","C3\_64\_6","Padding","
           same")];
        lgraph = addLayers(lgraph,tempLayers);
        tempLayers = [
        additionLayer(2,"Name","addition_2")
        reluLayer("Name","relu_1_2")];
        lgraph = addLayers(lgraph,tempLayers);

        tempLayers = [
        convolution2dLayer([3 3],64,"Name","C3\_64\_7","Padding","
           same")
        reluLayer("Name","relu_4")
        convolution2dLayer([3 3],64,"Name","C3\_64\_8","Padding","
           same")];
        lgraph = addLayers(lgraph,tempLayers);

        tempLayers = [
        additionLayer(2,"Name","addition_3")
        reluLayer("Name","relu_1_3")];
        lgraph = addLayers(lgraph,tempLayers);

        tempLayers = [
        convolution2dLayer([3 3],64,"Name","C3\_64\_9","Padding","
           same")
        reluLayer("Name","relu_5")
        convolution2dLayer([3 3],64,"Name","C3\_64\_10","Padding","
           same")];
        lgraph = addLayers(lgraph,tempLayers);

        tempLayers = [
        additionLayer(2,"Name","addition_5")
        reluLayer("Name","relu_1_4")];
        lgraph = addLayers(lgraph,tempLayers);

        tempLayers = [
        convolution2dLayer([3 3],64,"Name","C3\_64\_11","Padding","
           same")
        reluLayer("Name","relu_6")
        convolution2dLayer([3 3],64,"Name","C3\_64\_12","Padding","
           same")];
        lgraph = addLayers(lgraph,tempLayers);

```

```
125         tempLayers = [
126         additionLayer(2,"Name","addition_6")
127         reluLayer("Name","relu_1_5")];
128         lgraph = addLayers(lgraph,tempLayers);
129
130         tempLayers = [
131         additionLayer(2,"Name","addition_4")
132         reluLayer("Name","R finale")
133         convolution2dLayer([3 3],31,"Name","C3_da_64_a_31","Padding
                ","same")
134         mraeLossLayer("MRAE Loss")];
135         lgraph = addLayers(lgraph,tempLayers);
136         clear tempLayers;
137         lgraph = connectLayers(lgraph,"C3\_64\_1","C3_64");
138         lgraph = connectLayers(lgraph,"C3\_64\_1","addition/in2");
139         lgraph = connectLayers(lgraph,"C3\_64\_1","addition_4/in2")
                ;
140         lgraph = connectLayers(lgraph,"C3\_64\_2","addition/in1");
141         lgraph = connectLayers(lgraph,"relu_1","C3\_64\_3");
142         lgraph = connectLayers(lgraph,"relu_1","addition_1/in2");
143         lgraph = connectLayers(lgraph,"C3\_64\_4","addition_1/in1")
                ;
144         lgraph = connectLayers(lgraph,"relu_1_1","C3\_64\_5");
145         lgraph = connectLayers(lgraph,"relu_1_1","addition_2/in2");
146         lgraph = connectLayers(lgraph,"C3\_64\_6","addition_2/in1")
                ;
147         lgraph = connectLayers(lgraph,"relu_1_2","C3\_64\_7");
148         lgraph = connectLayers(lgraph,"relu_1_2","addition_3/in2");
149         lgraph = connectLayers(lgraph,"C3\_64\_8","addition_3/in1")
                ;
150         lgraph = connectLayers(lgraph,"relu_1_3","C3\_64\_9");
151         lgraph = connectLayers(lgraph,"relu_1_3","addition_5/in2");
152         lgraph = connectLayers(lgraph,"C3\_64\_10","addition_5/in1
                ");
153         lgraph = connectLayers(lgraph,"relu_1_4","C3\_64\_11");
154         lgraph = connectLayers(lgraph,"relu_1_4","addition_6/in2");
155         lgraph = connectLayers(lgraph,"C3\_64\_12","addition_6/in1
                ");
156         lgraph = connectLayers(lgraph,"relu_1_5","addition_4/in1");
157         plot(lgraph);
158         net = trainNetwork(trainDS, lgraph, options);
159         end
160         save('RETE_HSCNN_R_finale.mat','net');
161         Epoch,Iteration,TimeElapsed,LearnRate,TrainingLoss,
                ValidationLoss,State.
162         function stop = monitorLoss(info)
163         stop = false;
164         if isnan(info.TrainingLoss)
165         disp("Loss NaN! Interrompo il training.");
166         stop = true;
167         end
168         indice = num2str(info.Iteration);
169         if mod(info.Iteration, 100) == 0
170         info_name = indice + "_RETE_HSCNN_R_info_checkpoint.mat" ;
171         percorso_info = fullfile('Cartella␣Info', info_name);
172         save( percorso_info ,"info");
```

```
173         disp("Checkpoint salvato.");
174         disp(['Epoch: ', num2str(info.Iteration), ' - Loss: ',
                num2str(info.TrainingLoss), ' - Val Loss: ', num2str(info
                .ValidationLoss)]);
175         end
176         end
```

Appendix F

MRAE Loss Class Function

```
classdef mraeLossLayer < nnet.layer.RegressionLayer
methods
function layer = mraeLossLayer(name)
layer.Name = name;
layer.Description = "Mean Relative Absolute Error (MRAE)
   loss with NaN and Inf handling";
end
function loss = forwardLoss(~,Y, T)
epsilon = 1e-8;
if any(isinf(T(:)))
maxFiniteT = max(T(~isinf(T)));
T(isinf(T)) = maxFiniteT;
disp('Warning: Inf values in T replaced with max finite
   value');
end
T(isnan(T)) = 0.001;
Y(isnan(Y)) = 0.001;
T(T < 0) = 0.01;
loss_values = abs(Y - T)./(T + epsilon);
loss = mean(mean(loss_values, 1), 'all');
disp(['Loss attuale : ', num2str(loss)]);
if isnan(loss)
error('Loss NaN! Controlla i dati di input e il modello.');
end
```