electronics

MDPI

Article

OpenWeedGUI: An Open-Source Graphical Tool for Weed Imaging and YOLO-Based Weed Detection

Jiajun Xu, Yuzhen Lu * and Boyang Deng

Department of Biosystems & Agricultural Engineering, Michigan State University, East Lansing, MI 48824, USA; xujiaju1@msu.edu (J.X.)
* Correspondence: luyuzhen@msu.edu

Abstract: Weed management impacts crop yield and quality. Machine vision technology is crucial to the realization of site-specific precision weeding for sustainable crop production. Progress has been made in developing computer vision algorithms, machine learning models, and datasets for weed recognition, but there has been a lack of open-source, publicly available software tools that link imaging hardware and offline trained models for system prototyping and evaluation, hindering community-wise development efforts. Graphical user interfaces (GUIs) are among such tools that can integrate hardware, data, and models to accelerate the deployment and adoption of machine vision-based weeding technology. This study introduces a novel GUI called OpenWeedGUI, designed for the ease of acquiring images and deploying YOLO (You Only Look Once) models for real-time weed detection, bridging the gap between machine vision and artificial intelligence (AI) technologies and users. The GUI was created in the framework of PyQt with the aid of open-source libraries for image collection, transformation, weed detection, and visualization. It consists of various functional modules for flexible user controls and a live display window for visualizing weed imagery and detection. Notably, it supports the deployment of a large suite of 31 different YOLO weed detection models, providing flexibility in model selection. Extensive indoor and field tests demonstrated the competencies of the developed software program. The OpenWeedGUI is expected to be a useful tool for promoting community efforts to advance precision weeding technology.

Keywords: machine vision; Qt; user interface; weed detection; precision agriculture

Citation: Xu, J.; Lu, Y.; Deng, B. OpenWeedGUI: An Open-Source Graphical Tool for Weed Imaging and YOLO-Based Weed Detection. *Electronics* **2024**, *13*, 1699. https://doi.org/10.3390/electronics13091699

Academic Editor: Luca Mesin

Received: 18 March 2024
Revised: 24 April 2024
Accepted: 24 April 2024
Published: 27 April 2024

1. Introduction

Weeds are a constant threat to crop production. Improper management of weeds can potentially lead to reductions of more than 50% in crop yield [1,2]; if weeds are left uncontrolled, the yield loss can escalate to 90% [3]. In the United States, weed control costs more than insect and disease control combined, with an average annual cost of $33 billion [4]. Traditional approaches to weed control include, among others, mechanical cultivation, hand weeding, and herbicide application [5]. While these methods have proven effective to varying degrees, they have many drawbacks. Mechanical weeding entails high costs and or low efficacy, and hand weeding is not economically feasible because of labor shortages and rising labor costs. Herbicides are the most widely used for weed management, but currently, intensive, repeated applications of herbicides have resulted in the evolution of herbicide-resistant weeds, substantially reducing control efficacy and increasing management costs. Herbicide application also poses risks to the environment and human health [6,7]. There is a critical need to develop effective and sustainable weed control methods.

Machine vision technology plays a crucial role in the realization of future site-specific weed management and sustainable cropping systems [8]. Integrating imaging sensors, actuators, and computer algorithms, machine vision has enabled the automation of a wide range of tasks in agriculture, which otherwise would be laborious and costly. In weed

management, it provides the ability to distinguish between weeds and crops and identify weed species, especially with the aid of high-performance deep learning methods [9]. Once weeds are detected and localized, appropriate controls can be activated and applied site-specifically to remove weeds, such as targeted herbicide spot spraying [10], the activation of mechanical cultivators or fingers [11], or the implementation of thermal methods [12]. These machine vision-based weeding approaches offer promising solutions for combating today's herbicide resistance in weed populations.

However, reliable discrimination between crops and weeds and among weed species remains a substantial challenge for current state-of-the-art machine vision technology. To address this challenge, efforts are called for to develop large-scale publicly accessible image datasets and benchmarking models. In recent years, a number of labeled weed datasets, alongside AI models trained on the datasets for performance benchmarking, have been created and released [13], such as DeepWeeds [14], Weed25 [15], CottonWeedID15 [16], and CottonWeedDet12 [17], just to name a few. The availability of these datasets inspires research into deep learning-based weed recognition [18]. Among various deep learning architectures, You Only Look Once (YOLO) detectors have drawn prominent attention for real-time object detection, achieving good tradeoffs between accuracy and speed [19,20]. As a one-stage detector, YOLOs operate by dividing an image into a grid of cells, each responsible for predicting bounding boxes and class probabilities for objects within that cell. A single convolutional neural network processes the entire image and generates predictions for all cells simultaneously, making the detectors exceptionally efficient. This advantage is particularly attractive for weed control by mobile systems for which timely responses are essential for efficiently targeting localized weeds. Wang, et al. [21] enhanced YOLOv5 with an attention mechanism, achieving a precision of 94.65% and recall of 90.17% in real-time detection of invasive *Solanum rostratum* Dunal seedlings. Dang, et al. [13] evaluated a large suite of 25 YOLO architectural variants for weed detection and reported detection accuracy ranging from 88% to 95% on the CottonWeedDet12 dataset [17]. Despite recent developments, significant efforts are still needed to develop large-scale weed datasets and robust machine vision-based weeding systems for practical applications.

Graphical user interfaces (GUIs) are software tools or applications that enable users to interact with hardware, data, and models, which are important for prototyping and the evaluation of machine vision systems in precision agriculture [22]. GUIs can provide a user-friendly interface that allows individuals, such as researchers, developers, and end-users, to easily access and utilize machine vision tools [23]. Particularly, in weed recognition, GUIs bridge the gaps between curated image datasets and machine learning models and offer the ease of deploying and evaluating models (trained offline) for real-time application. Making GUIs publicly accessible, in addition to datasets and models, would be significantly beneficial for stimulating collective and community efforts for repaid development iterations and progress, allowing developers to focus on solving other challenging tasks [24]. Although dedicated efforts have been made as described above to develop dedicated datasets and models to power machine vision systems for weed management, there have been comparatively scarce efforts directed at the development of open-source GUI tools for weed imaging and recognition. The lack of such tools hurts further progress in advancing the development of machine vision-based and AI-powered weeding technology.

There are different software frameworks available for developing GUIs for agricultural applications. The GUIDE and App Designer tools in Matlab (Mathworks, Natick, MA, USA) are commonly used for GUI design for image data analytics. Mobaraki, et al. [25] developed a Matlab-based GUI for the analysis of hyperspectral images. Khandarkar, et al. [26] performed an image-processing technique to detect artificially ripened bananas using a Matlab-based GUI. Furthermore, Lu, et al. [27] presented a Matlab-based GUI for fruit defect detection using structured illumination imaging. As a graphical programming language, LabVIEW is well suited for GUI design for instrumentation and measurement systems. Deshmukh, et al. [28] designed LabVIEW GUI to classify the ripening stages

of the various fruits. However, both Matlab and LabVIEW, which require expensive licenses, are not preferred for developing open-source, free software. Instead, packages in Python such as Tkinter, PySide, and PyQt offer alternatives for creating such GUIs. Bauer, et al. [29] developed a Tkinter-based GUI for automated phenotypic analysis of lettuce using aerial images. ElManawy et al. (2022) designed a Tkinter-based software platform to process hyperspectral images for high-throughput plant phenotyping. Among various GUI designing tools, Qt (https://www.qt.io/, assessed on 23 April 2024) is well known as a free, open-source toolkit for developing user-friendly, scalable, cross-platform applications. In addition to supporting GUI development in C++, Qt has pythonic wrappers, such as PyQt and PySide for Python developers, which can be readily integrated with mainstream computer vision and deep learning libraries to develop machine vision and data analytics systems. Lu, et al. [30] used Qt in C++ with OpenCV for the system operation and online computer vision tasks of an apple harvest-assistive and in-field sorting machine. Gao, et al. [31] developed a software platform with PyQt5 for the processing and modeling of photosynthetic data in protected agriculture production. However, currently, there has been a dearth of research efforts on developing GUIs dedicated to weed imaging and the deployment of deep learning models for weed detection towards machine vision-based robotic weeding.

With the goal of advancing machine vision technology for weed recognition and control, the overall objective of this research was therefore to develop a user-friendly, open-source GUI, called OpenWeedGUI, for weed imagery collection and deployment and the evaluation of deep learning models for weed detection. Specifical objectives were to (1) design a Qt-based GUI that interfaces with a camera for weed imaging and a suite of off-line trained YOLO models for real-time weed detection, and (2) develop and field test a mobile machine vision platform integrated with the GUI for weed imaging and detection. Source codes of the OpenWeedGUI software program have been made publicly available (https://github.com/XU-JIA-JUN/OpenWeed-GUI-PyQt5-YOLO, assessed on 23 April 2024).

2. Materials and Methods

2.1. GUI Design Features

Figure 1 illustrates the design pipeline of OpenWeedGUI. Bridging the gap between users and imaging hardware and computer vision algorithms, the software has two primary functionalities, i.e., image collection and weed detection, which are realized through camera interfacing and the deployment of deep learning models, respectively. For camera interfacing, users are allowed to control camera parameter settings, which affect image acquisition, display, and weed detection. Model deployment provides the flexibility of choosing from a range of pre-trained YOLO models to detect weeds in image streams. Moreover, the GUI displays weed detection statistics and saves raw and detected images in real-time, facilitating the subsequent offline assessment of weed detection performance. Figure 2 shows the main window of OpenWeedGUI. The interface features a menu bar on the top comprising dropdown options supporting different functionalities as described below. Multiple functional modules are laid out vertically on the left of the interface to enable various user settings and results logging, and their specific functionalities are described below. A display window is placed on the right for live visualization of image streams and weed detection results, and below the window is a playback control module to visualize the weed detection of videos uploaded from external sources.

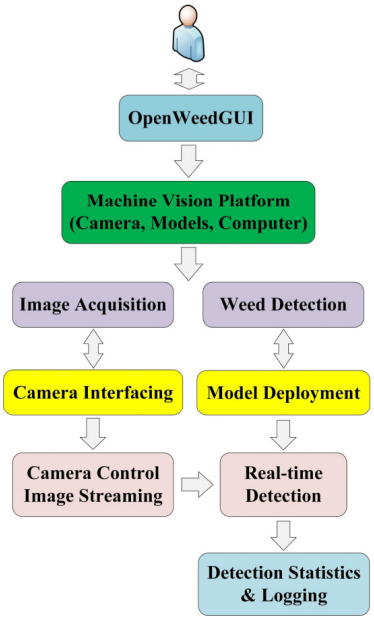

Figure 1. The design pipeline of OpenWeedGUI.

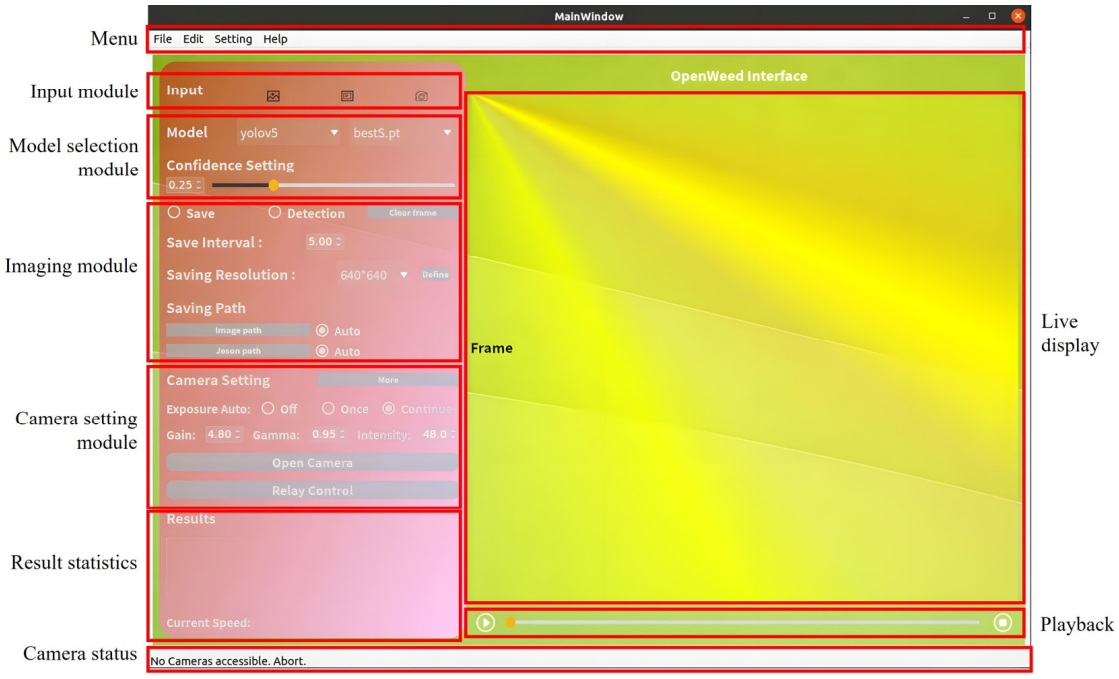

Figure 2. Graphical user interface layout.

2.1.1. GUI Framework

Figure 3 shows the framework of OpenWeedGUI, delineating the methodical interrelation between different functional modules. There are three distinct working threads, each of which is responsible for one of three input sources: images, videos, and camera livestreams. When an input option is checked (Figure 2), the corresponding thread is activated, enabling various downstream operational adjustments. For instance, with the camera option selected, upon a change in camera parameters made by the user, the GUI, driven by the active thread, instantaneously adjusts the imaging module's operations, ensuring that the detection output immediately reflects the parameter modifications. The input module interfaces with the model selection module and allows the user to choose one among a variety of YOLO models, which will be subsequently fed into the imaging

module, coordinating pivotal operations such as image acquisition and weed detection. The imaging operations are responsive to camera parameters, and the feedback from these processes subsequently prompts adjustments to the camera settings. The results statistics module prints a summary of weed detection results in real-time. The display module provides a live display of different types of image data input, which are internally linked to the playback module in case of playing an external video. The status bar conveys the real-time camera interfacing status.

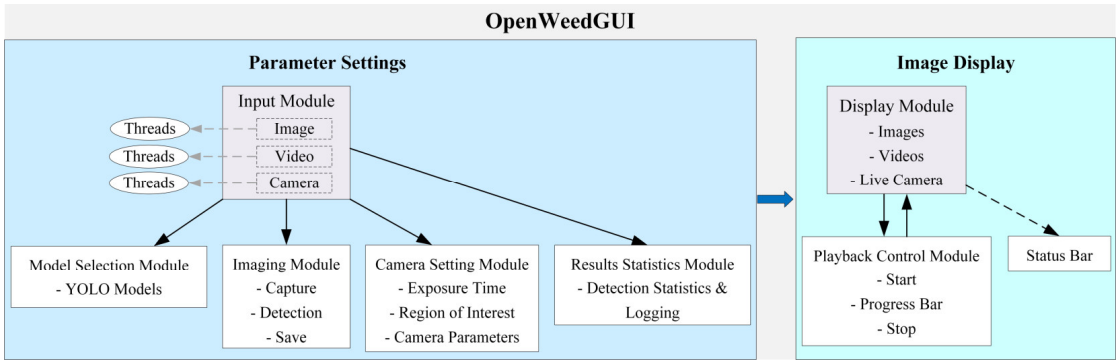

Figure 3. The OpenWeedGUI framework.

2.1.2. GUI Module Description

Presented below is a description of individual functional modules of OpenWeedGUI designed to deliver an efficient and friendly user experience.

- Menu: It offers streamlined navigation of the interface functionalities including system settings, help documentation, module shortcuts, and advanced camera control features, such as setting the region of interest (ROI). The ROI setting allows one to work with a specific rectangular area of input images or live image streams for improved weed detection. With the menu bar, users can effortlessly switch among different operation tasks, enhancing the overall user experience.
- Input Module: This module allows users to select their preferred type of input data, i.e., image, video, and real-time camera image streams (primary target). This flexibility makes the GUI well-suited for real-time, online, and offline weed detection tasks.
- Model Selection Module: As an important technical feature in the GUI, this module enables users to choose from and deploy a collection of pre-trained YOLO models (Section 2.3) for weed detection. Additionally, users can adjust the confidence threshold for enhanced weed detection. This module offers ease of model switching during the weed detection process, enabling users to decide on a model that best suits detection needs.
- Imaging Module: This module offers options for capturing and saving detected images. It has two checkboxes for saving and detection functions, respectively, with the flexibility to enable or disable image saving during the detection process. Users can specify a saving interval (in seconds) that determines how frequently images (in .jpg format), alongside associated detection files (in .json format), are saved locally. Images can be saved at either the original or customized resolution, and upon clicking the "Save" checkbox, three separate folders will be created, each timestamped, for saving raw images, the corresponding images overlaid with detection bounding boxes, and JSON files, respectively. The "detection" will perform weed detection by a specified model for camera video streams in real-time.
- Camera Setting Module: This module allows users to interact with imaging hardware and adjust camera parameters, including exposure times, gain, gamma, and intensity. A handy "More" button is added, upon clicking, to enable additional color space settings including Saturation, Hue, and Value. The permissible adjustment

ranges (specific to the imaging hardware described in Section 2.2.1) for these camera parameters are given in Table 1.

- Display Module: This module displays the source image data, with weed detection bounding boxes overlaid if the detection option is checked. It supports the presentation of static images, external videos, and real-time camera video frames. The display dimensions are configured to a square shape of 640 × 640 pixels, dictated by the input size of YOLO models. However, the size of the display module can be readily modified to cater to specific input requirements.
- Playback Control Module: This module incorporates three essential components including a start button, a progress bar, and a stop button, for weed detection of external videos. With a video loaded, users can initiate playback by selecting the start button. Throughout the playback, the checkbox option for image detection and saving remains accessible, and the progress bar allows users to define the specific range for weed detection by adjusting its position.
- Status Output Module: This module communicates real-time operational updates, whether it is tracking the processing status, alerting the user about potential issues, or updating on successful tasks. For instance, it displays connection statuses, such as verifying if the camera is successfully connected or indicating any disconnections. It also specifies the source of the current video stream, whether it is from a local file or a real-time camera. This design keeps the users informed every step of the way.

Table 1. Description of camera parameter settings.

Camera Parameter	Description	Range
Gain	Control the sensitivity of the camera's sensor to light	[0, 24]
Gamma	Perform gamma correction	[0, 2.4]
Intensity	Set a specific target intensity level for the camera to maintain.	[0, 100]
Saturation	Adjust the intensity of colors, making them more or vivid	[0, 2]
Hue	Modify the overall tint and tone of the image	[0, 300]
Value	Adjust the brightness of the image	[0, 1]

2.2. Hardware and Software

2.2.1. Hardware

A lightweight mobile machine vision platform was designed and prototyped for the implementation and testing of OpenWeedGUI. As schematically shown in Figure 4, the platform mainly consists of a color camera (Allied Vision, Stadtroda, Germany) with a resolution of 2465 × 2056 pixels, attached with an 8.5-mm focus lens, and a Jetson AGX Orin developer kit (NVIDIA, Santa Clara, CA, USA) connected with supporting peripherals (monitor and keyboard), which are mounted with a wheeled, motorized frame. The platform was powered by a 12 V battery, which was connected to an inverter to power the computer and monitor. The Jetson kit, with 2048-core Ampere architecture and 64 Tensor Cores, is an advanced, energy-efficient system-on-a-chip edge computer designed for AI applications. The mobile platform was deployed for weed imagery in field conditions and the evaluation of onboard weed detection with the aid of OpenWeedGUI.

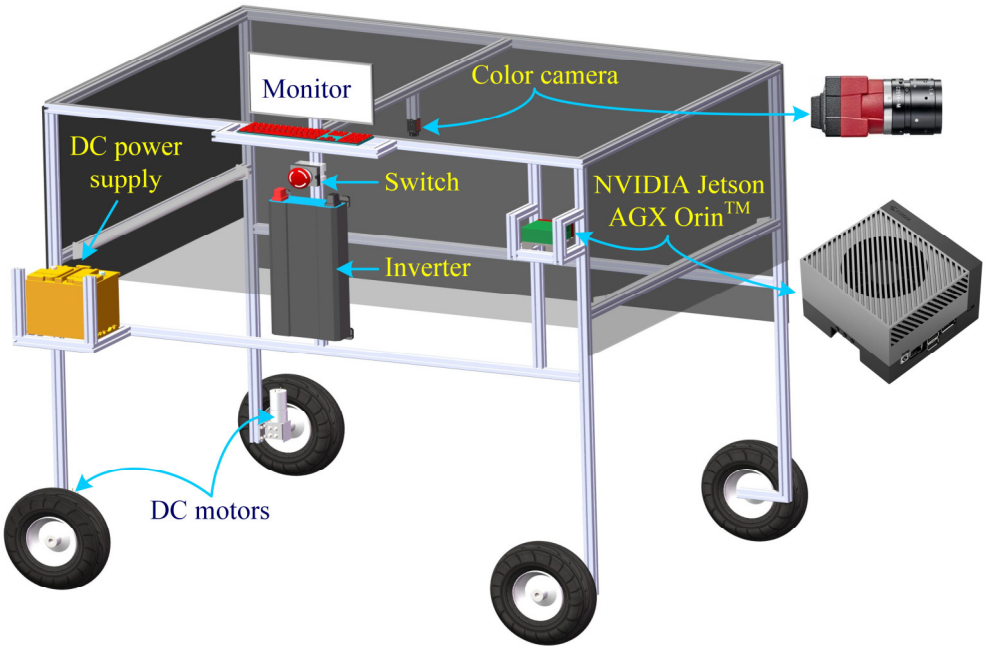

Figure 4. Conceptual illustration of a mobile machine vision system for in-field weed imaging.

2.2.2. Software

The OpenWeedGUI was developed using Python 3.8 in the Ubuntu 22.04.2 operating system, employing the PyQt5 toolkit for its user interface design. As a pythonic wrapper of Qt (C++), PyQt5 takes advantage of the former for GUI design while enabling extensive Python libraries for computer vision and deep learning tasks. In the framework of Qt, camera interfacing was achieved by the Vimba Python API (version 1.2.1) (https://github.com/alliedvision/VimbaPython, assessed on 20 June 2023) (Allied Version, Stadtroda, Germany) and downstream image and video processing tasks were undertaken by OpenCV (version 4.7.0). In conjunction with OpenCV, PyTorch (version 1.14) was used for deploying already trained YOLO object detection models for weed detection. Particularly for YOLOv5 models, TensorRT (version 7.x) (https://github.com/wang-xinyu/tensorrtx, assessed on 5 September 2023) and PyCUDA (https://github.com/inducer/pycuda, assessed on 5 September 2023) were used for accelerating real-time inference of YOLOv5 models, which is desirable for weeding systems with resource-constrained computing devices. The software programs for OpenWeedGUI will be made publicly available.

2.3. YOLO Models

OpenWeedGUI supports the deployment of a variety of YOLO object detectors as summarized in Table 2. These models were trained on the CottonWeedDet12 dataset [17] through transfer learning in PyTorch. The dataset consists of 5648 color images collected in diverse cotton field conditions, and a total of 9370 bounding box annotations for 12 weed classes, which is publicly accessible in the Zenodo repository (https://zenodo.org/records/7535814, assessed on 15 January 2023). Details about the training and benchmarking performance metrics of YOLO models, as well as the statistics of CottonWeedDet12, are given elsewhere [13]. It is noted that YOLOX and YOLOv8 models were not trained for weed detection in [13], but were trained in a separate study [32] and included in the OpenWeedGUI for completeness. The functionality of supporting a large suite of YOLO models makes the OpenWeedGUI a versatile tool. It can be used in offline scenarios for facilitating comparative analysis and performance evaluation of weed detection models, assisting in model selection. Users also have the convenience of deploying models that best align with their specific needs regarding accuracy and efficiency for real-time onboard weed detection.

Table 2. YOLO models supplied by OpenWeedGUI for weed detection.

YOLO Variant		Model in GUI	Input Size	Number of Parameters (Million)	URL
YOLOv3	YOLOv3-tiny	best-tiny.pt	640 × 640	8.6	https://github.com/ultralytics/YOLOv3, assessed on 9 July 2023
	YOLOv3	best3.pt	640 × 640	61.5	
	YOLOv3-SPP	bestSPP.pt	640 × 640	62.6	
YOLOv4	YOLOv4pacsp-s	bestv4_pacsp-s.pt	640 × 640	8.1	https://github.com/WongKinYiu/PyTorch_YOLOv4, assessed on 9 July 2023
	YOLOv4pacsp	bestv4_pacsp.pt	640 × 640	45.4	
	YOLOv4	bestv4.pt	640 × 640	63.2	
	YOLOv4pacsp-x	bestv4_pacsp-x.pt	640 × 640	97.8	
Scaled-YOLOv4	YOLOv4-P5	bestsv4-p5.pt	640 × 640	72.6	https://github.com/WongKinYiu/ScaledYOLOv4, assessed on 9 July 2023
	YOLOv4-P6	bestsv4-p6.pt	640 × 640	128.6	
	YOLOv4-P7	bestsv4-p7.pt	640 × 640	287.9	
YOLOR	YOLOR-P6	bestR-P6.pt	640 × 640	37.5	https://github.com/WongKinYiu/yolor, assessed on 9 July 2023
	YOLOR-CSP	bestR-CSP.pt	640 × 640	53.3	
	YOLOR-CSP-X	bestR-CSP-X.pt	640 × 640	102.4	
YOLOv5	YOLOv5n	bestN.pt	640 × 640	1.9	https://github.com/ultralytics/YOLOv5, assessed on 9 July 2023
	YOLOv5s	bestS.pt	640 × 640	7.2	
	YOLOv5m	bestM.pt	640 × 640	21.2	
	YOLOv5l	bestL.pt	640 × 640	46.5	
	YOLOv5x	bestX.pt	640 × 640	86.7	
YOLOv5 + TensorRT	YOLOv5n	bestN.engine	640 × 640	1.9	https://github.com/wang-xinyu/tensorrtx/tree/master/yolov5, assessed on 9 July 2023
	YOLOv5s	bestS.engine	640 × 640	7.2	
	YOLOv5m	bestM.engine	640 × 640	21.2	
	YOLOv5l	bestL.engine	640 × 640	46.5	
	YOLOv5x	bestX.engine	640 × 640	86.7	
YOLOX	YOLOX-s	bests_ckpt.pth	640 × 640	9.0	https://github.com/Megvii-BaseDetection/YOLOX, assessed on 9 July 2023
	YOLOX-m	bestm_ckpt.pth	640 × 640	25.3	
	YOLOX-l	best_ckpt.pth	800 × 800	54.2	
	YOLOX-x	bestx_ckpt.pth	800 × 800	99.1	
YOLOv8	YOLOv8s	yolov8bestS.pt	640 × 1200	28.6	https://github.com/ultralytics/ultralytics, assessed on 9 July 2023
	YOLOv8m	yolov8bestM.pt	640 × 640	78.9	
	YOLOv8l	yolov8bestL.pt	800 × 800	165.2	
	YOLOv8x	yolov8bestX.pt	640 × 640	257.8	

3. Experimentation and Discussion

3.1. Indoor Experimentation

Initial indoor experiments were conducted to assess the operational validity of functional modules of OpenWeedGUI deployed on the mobile platform (Figure 4). These examinations included camera activation, image setting adjustment, the responsiveness of model switching, image and video detection, the efficiency of real-time weed detection, and the reliability of results saving, under indoor lighting conditions. For weed detection, representative weed images were printed on paper sheets in color and placed on the ground beneath the camera. Additionally, an external video, captured for weed plants in natural field conditions in August of 2022, was imported to the GUI to test video-based weed detection.

Figure 5 showcases the real-time weed detection, model switching, and image acquisition outcomes of OpenWeedGUI. Upon clicking the "Open Camera" button, the button label is updated to "Close Camera", signaling that the camera has become active. Concurrently, the GUI starts to detect weeds in the camera image stream when the "Detect" checkbox is enabled as shown in Figure 5a. The process involves deploying a specific YOLO model (YOLOv5s by default) for weed detection, which immediately engages the

display module to present detection results enclosed by bounding boxes. Instantly, the result statistics module presents a summary of the quantities of detected weed instances within the current frame. The GUI supports switching among different models, as shown in Figure 5b where users can select one among YOLO variants available in a dropdown list. Upon selection, the interface intuitively updates a second list of architecture options specific to the chosen YOLO variant. This allows users to efficiently navigate through and experiment with different types of YOLO models.

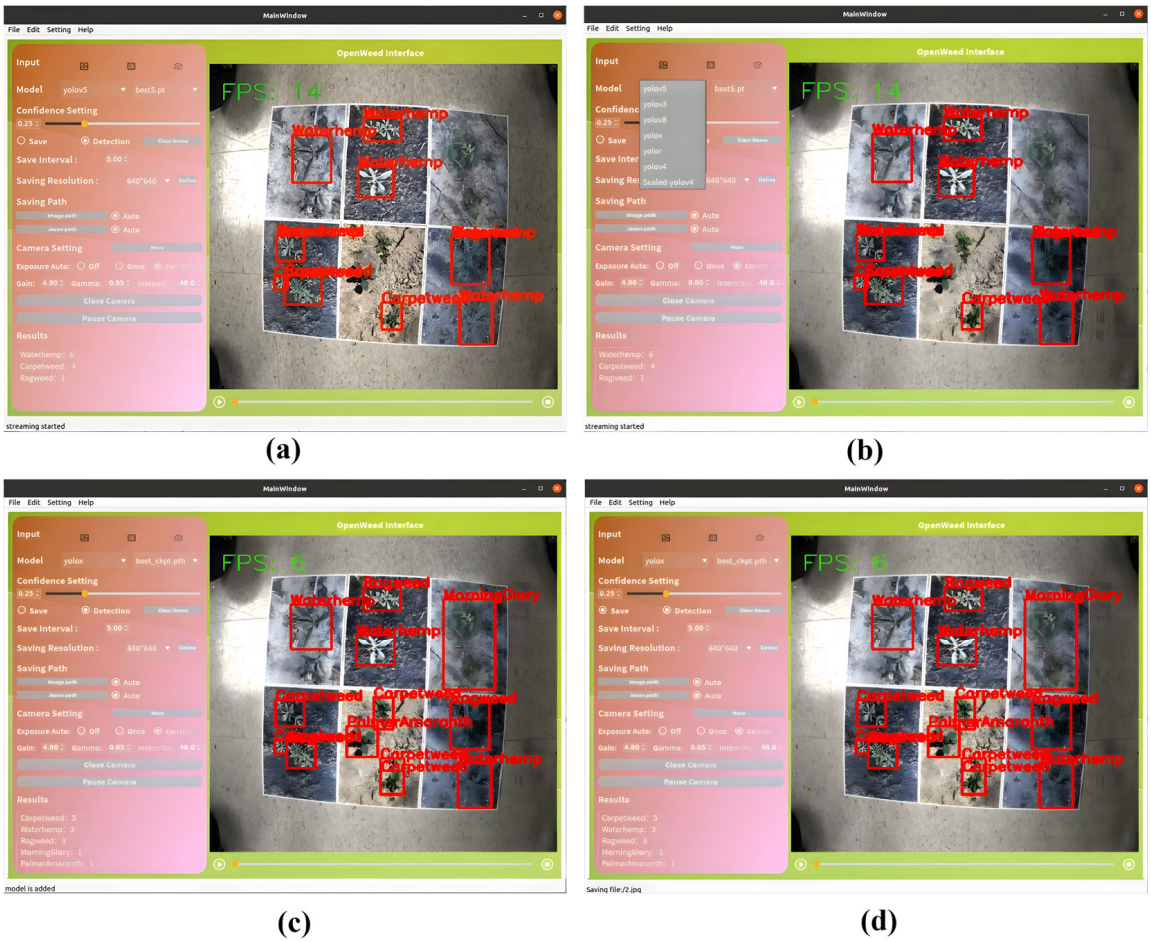

Figure 5. Illustration of OpenWeedGUI for weed detection: (**a**) run detection; (**b**) model selection; (**c**) change model; (**d**) save results.

The GUI allows for changing the detection model dynamically at run time, as shown in Figure 5c where YOLOv5s is switched to YOLOX, accompanied by updates in detection results. At the same time, the status bar at the bottom of the interface responds to this adjustment, indicating that a new model has been deployed. The "Save" checkbox of the GUI allows users to save images in their original and detected forms to a user-defined directory. As shown in Figure 5d, the status of each saving event is promptly communicated in the status bar. In addition to saving images (raw and detected with bounding boxes), the GUI also saves the detection results in .json files at a fixed user-defined time interval, as shown in Figure 6.

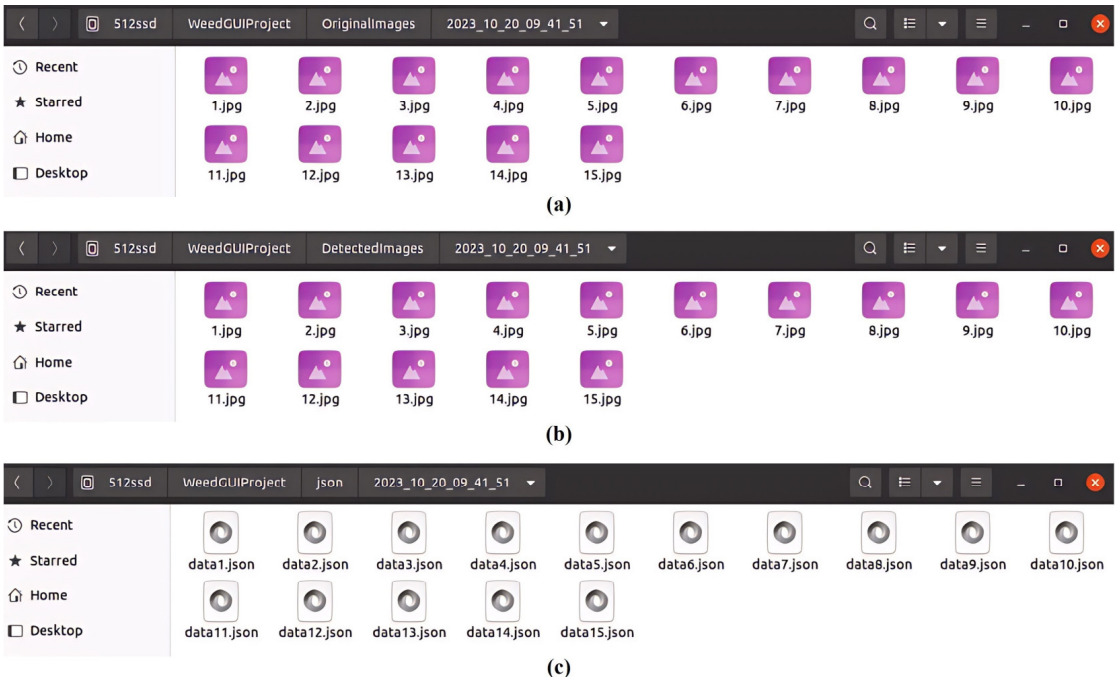

Figure 6. Illustration of saving operations by OpenWeedGUI in three separate folders. (**a**) Original images; (**b**) detected images with bounding boxes; (**c**) corresponding detection files (in .json format).

Figure 7 shows the outcomes of video detection, wherein users can play the video to dynamically observe detection results. Furthermore, Figure 8 demonstrates weed detection given a single image input. Here, users can import an image and initiate the detection process by enabling the detection checkbox. The GUI accommodates real-time adjustments of the hue, saturation, and value of a color image, as shown in Figure 8b,c, where image manipulation in a color space yields different detection results. It is noted that the GUI supports enabling saving operations on demand during video and image detection processes, allowing for the flexibility to capture images as needed.

Figure 7. Illustration of weed detection of a video uploaded from an external source.

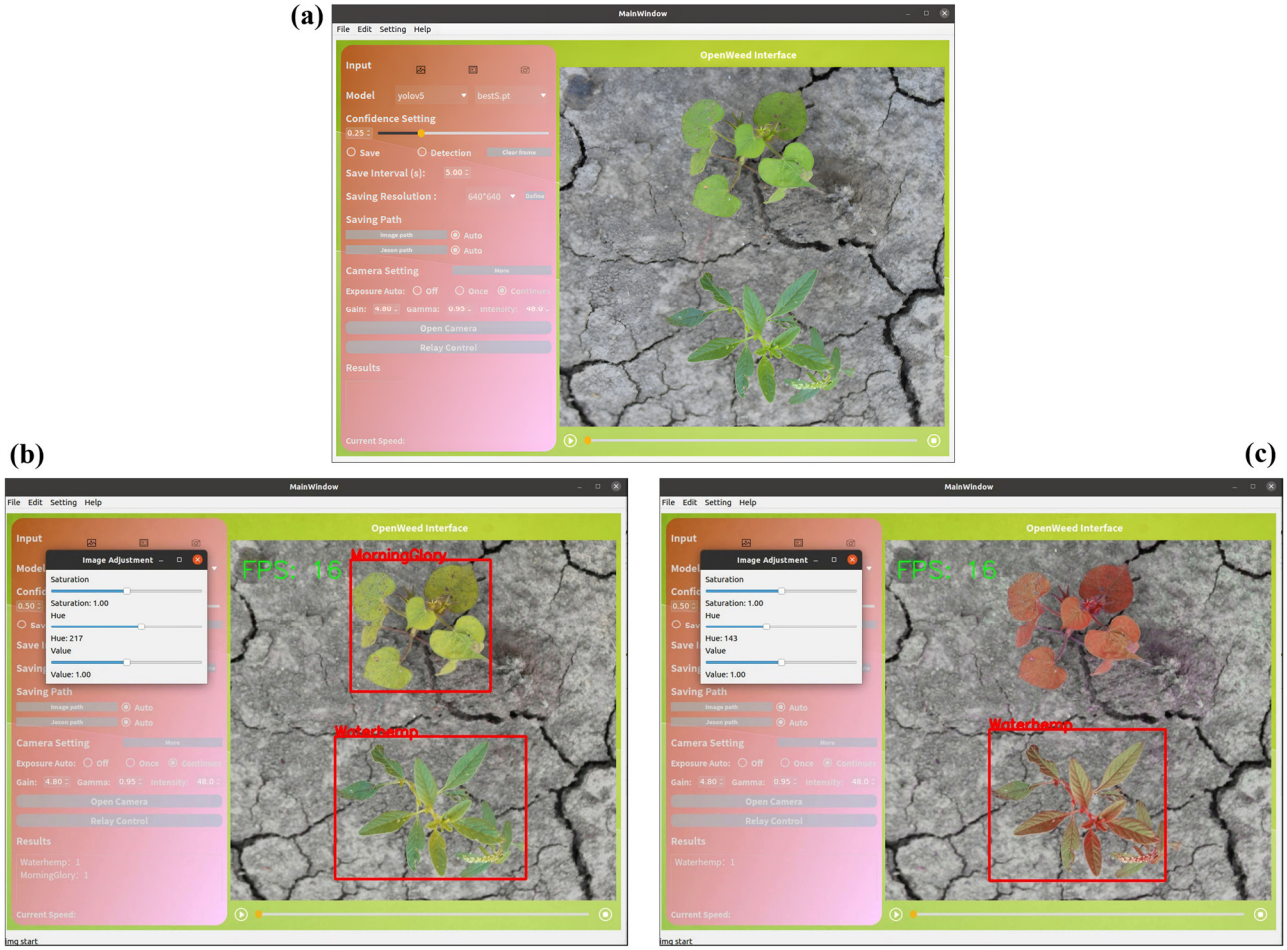

Figure 8. Illustration of weed detection for a single input image: (**a**) import an image, (**b**,**c**) detect weeds on the image modified differently in color space.

3.2. In-Field Testing

To further verify its operational functionalities, OpenWeedGUI was deployed on the mobile vision platform for in-field weed imaging and detection. Field tests were conducted throughout June and September of 2023 on the farmland of Michigan State University Horticulture Teaching and Research Center (Hort, MI). The field site was infested with naturally germinated weeds, among which Common Lambsquarters (*Chenopodium album* L.) was the dominant species. Since the weed was not included in the CottonWeedDet12 dataset for YOLO modeling [13], separate weed imagery was curated to update YOLO models to detect the weed, which was trained using default hyperparameter settings on a dataset consisting of 1439 images with 12,107 bounding box annotations for Common Lambsquarters. Before running the mobile platform in field test events, the camera parameters were adjusted to accommodate field light conditions and kept unchanged during the test. The mobile platform was operated at a traveling speed of approximately 0.35 m/s. Figure 9 (left) shows a typical field test scenario, and Figure 9 (right) shows weed detection with YOLOv8s enabled in OpenWeedGUI. Readers are referred to a video clip (https://youtu.be/KK36A4SWhH0, assessed on 15 December 2023) on the dynamic process of weed detection by OpenWeedGUI on a field test occasion. The field tests confirmed the capabilities of OpenWeedGUI for weed imaging and detection.

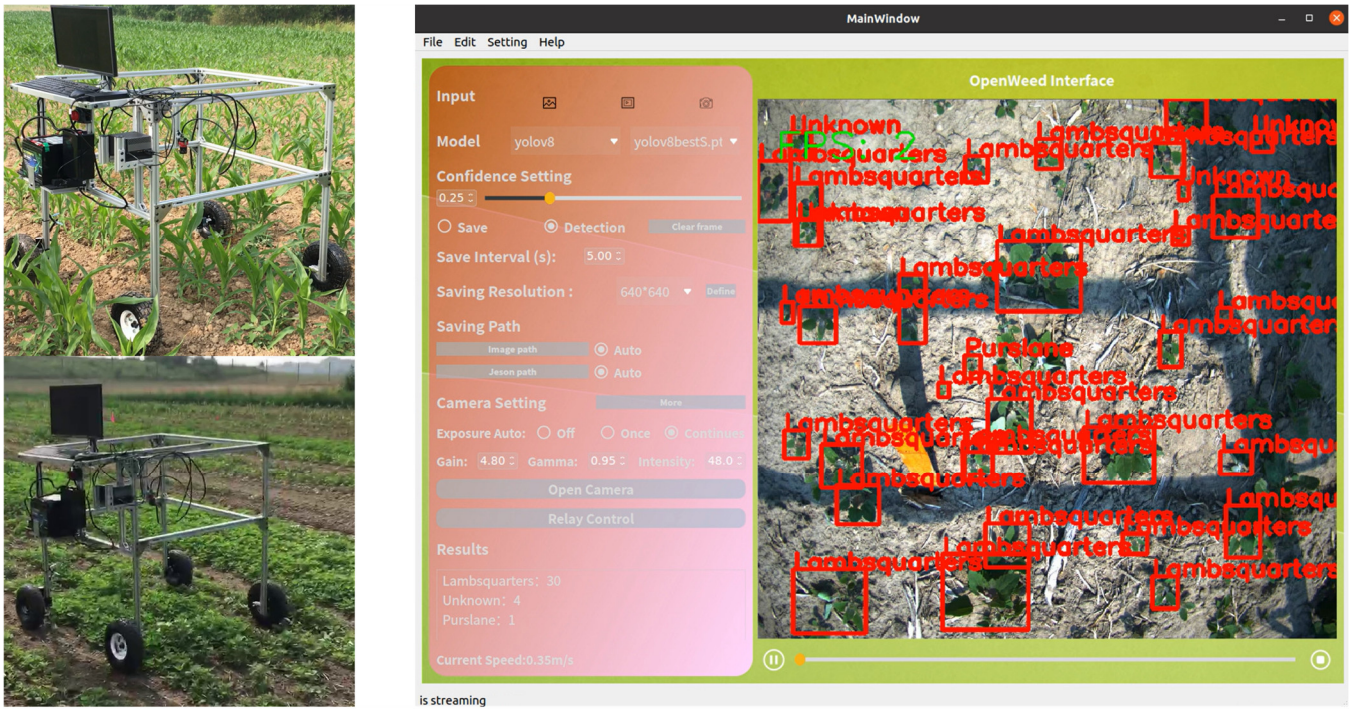

Figure 9. Photograph of field testing of a mobile weed imaging platform (**left**) and weed detection visualization of OpenWeedGUI (**right**).

3.3. Discussion

This study represents a novel attempt, in the spirit of developing and sharing open-source tools beneficial to the research community, to develop a GUI to facilitate greater efforts to advance machine vision-based weeding technology. The OpenWeedGUI introduced in this work is expected to be a useful tool for acquiring images and deploying YOLO models for weed detection. Both indoor and field tests validated the functionalities of various modules of the graphical tool, although these tests were not oriented toward assessing actual weed detection performance. Overall, the OpenWeedGUI met the design requirements in terms of achieving its two major functionalities (i.e., image acquisition and deployment of YOLO models for weed detection). However, it is important to point out some limitations and areas for further improvement.

The present OpenWeedGUI integrates pre-trained YOLO models into the software, which is however not memory efficient for rapid deployment. It would be more desirable to allow users to import external weed detection models separately from the interface, reducing the software file size and affording flexibility in model deployment. YOLOs are evolving rapidly. At the time of writing, YOLOv9 has been recently published [33], pushing the boundaries of real-time object detection. It is worthwhile to evaluate its performance in weed detection. While YOLOs are among the most successful real-time object detectors, there are other high-performant single-stage objectors, and very recently, transformer-based object detectors have also shown real-time efficiency while achieving high accuracy [34]. Supporting the deployment of all of these models for weed detection would conceivably enhance the utility of the OpenWeedGUI. This will demand dedicated efforts to benchmark these models for weed detection. It is noted that the OpenWeedGUI builds on an imaging hardware-specific package, i.e., Vimba API in Python, for camera communication, which restricts its applicability to compatible cameras. To support a wider range of camera options, it is necessary to overcome the dependence and employ more generally useful camera interfacing packages, which remain to be determined. Research is underway to make the OpenWeedGUI more user-friendly for practitioners with limited programming skills by providing complied, executable (.exe) versions across different operating systems (e.g., Windows, Linux, and Raspberry Pi) alongside improved technical features.

4. Conclusions

This study presents the OpenWeedGUI, an innovative, open-source graphical user interface designed for the ease of collecting images and deploying YOLO models for on-board weed detection. Created in the PyQt framework with the aid of various open-source enabling libraries, the OpenWeedGUI is featured with user-friendly functional modules for input selection, camera control, model selection, image saving, weed detection, and result visualization and logging. Indoor and field tests demonstrated the technical competencies and reliability of the graphical software. The open-source nature of the OpenWeedGUI makes it highly extensible and amenable to modifications for customized needs and will inspire community efforts to develop and share software products supporting precision weed control research. The OpenWeedGUI software is expected to be a useful tool for advancing the development and application of machine vision systems for weed detection and control. Improvements will be made to enhance its utility and user-friendliness in further revisions.

Author Contributions: J.X.—writing, original draft, software, formal analysis; Y.L.—writing, original draft, methodology, funding acquisition, data curation, conceptualization; B.D.—data acquisition, software debugging. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Discretionary Funding Initiative of Michigan State University.

Data Availability Statement: The datasets and software programs for the graphical user interface presented in this study are made publicly available. The links to the datasets and codes are provided within the article.

Acknowledgments: An earlier version of this work was presented at the 2023 SPIE Defense + Commercial Sensing conference (Orlando, FL, USA). The authors thank Daniel Brainard for providing field plots for conducting field tests of the developed graphical user interface.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. MacRae, A.; Webster, T.; Sosnoskie, L.; Culpepper, A.; Kichler, J. Cotton yield loss potential in response to length of Palmer amaranth (*Amaranthus palmeri*) interference. *J. Cotton. Sci.* **2013**, *17*, 227–232.
2. Morgan, G.D.; Baumann, P.A.; Chandler, J.M. Competitive impact of Palmer amaranth (*Amaranthus palmeri*) on cotton (*Gossypium hirsutum*) development and yield. *Weed Technol.* **2001**, *15*, 408–412. [CrossRef]
3. Manalil, S.; Coast, O.; Werth, J.; Chauhan, B.S. Weed management in cotton (*Gossypium hirsutum* L.) through weed-crop competition: A review. *Crop Prot.* **2017**, *95*, 53–59. [CrossRef]
4. Pimentel, D. Pesticides applied for the control of invasive species in the United States. In *Integrated Pest Management*; Elsevier: Amsterdam, The Netherlands, 2014; pp. 111–123.
5. Abbas, T.; Zahir, Z.A.; Naveed, M.; Kremer, R.J. Limitations of existing weed control practices necessitate development of alternative techniques based on biological approaches. *Adv. Agron.* **2018**, *147*, 239–280.
6. Del Prado-Lu, J.L. Insecticide residues in soil, water, and eggplant fruits and farmers' health effects due to exposure to pesticides. *Environ. Health Prev. Med.* **2015**, *20*, 53–62. [CrossRef] [PubMed]
7. Ecobichon, D.J. Pesticide use in developing countries. *Toxicology* **2001**, *160*, 27–33. [CrossRef] [PubMed]
8. Young, S.L. Beyond precision weed control: A model for true integration. *Weed Technol.* **2018**, *32*, 7–10. [CrossRef]
9. Hasan, A.M.; Sohel, F.; Diepeveen, D.; Laga, H.; Jones, M.G. A survey of deep learning techniques for weed detection from images. *Comput. Electron. Agric.* **2021**, *184*, 106067. [CrossRef]
10. Chostner, B. See & Spray: The next generation of weed control. *Resour. Mag.* **2017**, *24*, 4–5.
11. Kennedy, H.; Fennimore, S.A.; Slaughter, D.C.; Nguyen, T.T.; Vuong, V.L.; Raja, R.; Smith, R.F. Crop signal markers facilitate crop detection and weed removal from lettuce and tomato by an intelligent cultivator. *Weed Technol.* **2020**, *34*, 342–350. [CrossRef]
12. Bauer, M.V.; Marx, C.; Bauer, F.V.; Flury, D.M.; Ripken, T.; Streit, B. Thermal weed control technologies for conservation agriculture—A review. *Weed Res.* **2020**, *60*, 241–250. [CrossRef]
13. Dang, F.; Chen, D.; Lu, Y.; Li, Z. YOLOWeeds: A novel benchmark of YOLO object detectors for multi-class weed detection in cotton production systems. *Comput. Electron. Agric.* **2023**, *205*, 107655. [CrossRef]
14. Olsen, A.; Konovalov, D.A.; Philippa, B.; Ridd, P.; Wood, J.C.; Johns, J.; Banks, W.; Girgenti, B.; Kenny, O.; Whinney, J. DeepWeeds: A multiclass weed species image dataset for deep learning. *Sci. Rep.* **2019**, *9*, 2058. [CrossRef] [PubMed]

15. Wang, P.; Tang, Y.; Luo, F.; Wang, L.; Li, C.; Niu, Q.; Li, H. Weed25: A deep learning dataset for weed identification. *Front. Plant Sci.* **2022**, *13*, 1053329. [CrossRef] [PubMed]
16. Chen, D.; Qi, X.; Zheng, Y.; Lu, Y.; Li, Z. Deep Data Augmentation for Weed Recognition Enhancement: A Diffusion Probabilistic Model and Transfer Learning Based Approach. *arXiv* **2022**, arXiv:2210.09509.
17. Lu, Y. CottonWeedDet12: A 12-class weed dataset of cotton production systems for benchmarking AI models for weed detection. *Zenodo* **2023**. [CrossRef]
18. Coleman, G.R.; Bender, A.; Hu, K.; Sharpe, S.M.; Schumann, A.W.; Wang, Z.; Bagavathiannan, M.V.; Boyd, N.S.; Walsh, M.J. Weed detection to weed recognition: Reviewing 50 years of research to identify constraints and opportunities for large-scale cropping systems. *Weed Technol.* **2022**, *6*, 1–50. [CrossRef]
19. Diwan, T.; Anirudh, G.; Tembhurne, J.V. Object detection using YOLO: Challenges, architectural successors, datasets and applications. *Multimed. Tools Appl.* **2023**, *82*, 9243–9275. [CrossRef] [PubMed]
20. Terven, J.; Cordova-Esparza, D. A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. *arXiv* **2023**, arXiv:2304.00501. [CrossRef]
21. Wang, Q.; Cheng, M.; Huang, S.; Cai, Z.; Zhang, J.; Yuan, H. A deep learning approach incorporating YOLO v5 and attention mechanisms for field real-time detection of the invasive weed *Solanum rostratum* Dunal seedlings. *Comput. Electron. Agric.* **2022**, *199*, 107194. [CrossRef]
22. dos Santos, R.P.; Fachada, N.; Beko, M.; Leithardt, V.R. A rapid review on the use of free and open source technologies and software applied to precision agriculture practices. *J. Sens. Actuator Netw.* **2023**, *12*, 28. [CrossRef]
23. Mössinger, J.; Troost, C.; Berger, T. Bridging the gap between models and users: A lightweight mobile interface for optimized farming decisions in interactive modeling sessions. *Agric. Syst.* **2022**, *195*, 103315. [CrossRef]
24. Wisse, M.; Chiang, T.-C.; van der Hoorn, G. *D1. 15: Best Practices in Developing Open Platform for Agri-Food Robotics*; European Commission: Brussels, Belgium, 2020.
25. Mobaraki, N.; Amigo, J.M. HYPER-Tools. A graphical user-friendly interface for hyperspectral image analysis. *Chemom. Intell. Lab. Syst.* **2018**, *172*, 174–187. [CrossRef]
26. Khandarkar, S.; Wadhankar, V.; Dabhade, D. Detection and identification of artificially ripened fruits using MATLAB. *Int. J. Innov. Res. Comput. Commun. Eng.* **2018**, *6*, 9136–9140.
27. Lu, R.; Pothula, A.K.; Mizushima, A.; VanDyke, M.; Zhang, Z. System for Sorting Fruit. US Patent 9,919,345, 20 March 2018.
28. Deshmukh, L.; Kasbe, M.; Mujawar, T.; Mule, S.; Shaligram, A. A wireless electronic nose (WEN) for the detection and classification of fruits: A case study. In Proceedings of the 2016 International Symposium on Electronics and Smart Devices (ISESD), Bandung, Indonesia, 29–30 November 2016; pp. 174–178.
29. Bauer, A.; Bostrom, A.G.; Ball, J.; Applegate, C.; Cheng, T.; Laycock, S.; Rojas, S.M.; Kirwan, J.; Zhou, J. Combining computer vision and deep learning to enable ultra-scale aerial phenotyping and precision agriculture: A case study of lettuce production. *Hortic. Res.* **2019**, *6*, 70. [CrossRef] [PubMed]
30. Lu, Y.; Lu, R.; Zhang, Z. Development and preliminary evaluation of a new apple harvest assist and in-field sorting machine. *Appl. Eng. Agric.* **2022**, *38*, 23–35. [CrossRef]
31. Gao, J.; French, A.P.; Pound, M.P.; He, Y.; Pridmore, T.P.; Pieters, J.G. Deep convolutional neural networks for image-based Convolvulus sepium detection in sugar beet fields. *Plant Methods* **2020**, *16*, 1–12. [CrossRef] [PubMed]
32. Deng, B.; Lu, Y.; Xu, J. Weed database development: An updated survey of public weed datasets and cross-season weed detection adaptation. *Ecol. Inform.* **2024**, *81*, 102546. [CrossRef]
33. Wang, C.Y.; Yeh, I.H.; Liao, H.Y.M. YOLOv9: Learning what you want to learn using programmable gradient information. *arXiv* **2024**, arXiv:2402.13616.
34. Lv, W.; Xu, S.; Zhao, Y.; Wang, G.; Wei, J.; Cui, C.; Du, Y.; Dang, Q.; Liu, Y. DETRs beat YOLOs on real-time object detection. *arXiv* **2023**, arXiv:2304.08069.