

Article

Analysis of Automated Digital Multimeter Readings Using LabVIEW OCR and YOLOv8s-Based Object Detection

Anna Szlachta ^{1,*}, Jakub Wnęk ^{1,*}, Jakub Drzał ¹, Piotr Kubiszyn ¹ and Tetiana Bubela ²

¹ Department of Metrology and Diagnostic Systems, Rzeszów University of Technology, Wincentego Pola 2, 35-021 Rzeszów, Poland; j.drzal@prz.edu.pl (J.D.); p.kubiszyn@prz.edu.pl (P.K.)

² Department of Measuring Information Technologies, Lviv Polytechnic National University, Stepana Bandery St, 12, 79013 Lviv, Ukraine; tetiana.z.bubela@lpnu.ua

* Correspondence: annasz@prz.edu.pl (A.S.); j.wnek@prz.edu.pl (J.W.)

Abstract

Automation of measurement data acquisition is particularly important when digital instruments do not provide a direct communication interface or when long-term measurements require high repeatability and reduced operator involvement. This paper presents a comparative study of two image-based optical reading methods to acquire indications from a digital multimeter. The reference signal was generated using a Fluke calibrator, while the multimeter display was recorded with an industrial camera. The acquired images were processed independently using two different recognition strategies. The first approach was implemented in the Python environment using a You Only Look Once version 8 small (YOLOv8s) deep learning object detection model trained to classify individual display characters. The second approach was implemented in the Laboratory Virtual Instrument Engineering Workbench (LabVIEW) using a template-based optical character recognition (OCR) method prepared in NI Vision Assistant. Acquisition series were performed for different voltage ranges, with 100 samples acquired for each pipeline and measurement case. All image-acquisition experiments were conducted under controlled laboratory conditions using a fixed camera position, a constant region of interest (ROI), and stable illumination. The detected or recognised characters were reconstructed as numerical values and compared with the reference settings of the calibrator on statistical parameters that describe the dispersion within the series and the deviation of the readings. The study evaluates the practical applicability of both approaches and demonstrates the potential of non-invasive optical reading methods for automated acquisition of indications from measurement instruments.

Keywords: laboratory automation; optical character recognition; YOLO; Python; LabVIEW; industrial camera; image processing; automated measurement system



Academic Editor: Kefeng Ji

Received: 23 June 2026

Revised: 24 July 2026

Accepted: 25 July 2026

Published: 28 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Camera-based measurement systems have been widely described in the literature and are used in many research and industrial areas. The dynamic development of artificial intelligence methods now enables not only image analysis, but also the integration and processing of information obtained from various data sources. Among the most commonly used solutions for object detection are algorithms from the You Only Look Once (YOLO) family, including the latest version, YOLOv26, which is used in object detection, segmentation, and image analysis tasks [1,2].

In the context of metrology, it is of particular importance to ensure consistency of measurement and the reliability of the results obtained using AI-assisted vision systems. Ensuring the quality of the measurement results requires proper calibration of the camera and optical system, determination of the relevant metrological characteristics, and identification of all significant sources of error that affect the measurement result. The final uncertainty of the measurement is influenced, among other factors, by the resolution of the image sensor, optical distortions of the lens, the stability of the lighting parameters, sensor noise, the accuracy of the calibration procedure, the properties of the image processing algorithms used, and the type of neural network model employed.

The evaluation of the quality of measurement results obtained using cameras should be carried out according to the principles set forth in the international guide [3], which specifies methods for determining and presenting measurement uncertainty. In the case of the validation of measurement methods and the evaluation of the accuracy and precision of results, the requirements of ISO 5725, regarding the trueness and precision of measurement methods and results, also apply [4].

In vision systems designed for geometric measurements, the requirements of the VDI/VDE 2634 standards, which describe methods for evaluating the accuracy of optical measuring systems that use imaging techniques, are also of significant importance [5]. For coordinate measurement systems, the reference is the requirements of the ISO 10360 series of standards, which specify procedures for verifying the accuracy of coordinate measurement machines [6]. Although these standards primarily apply to coordinate measuring systems, their requirements for accuracy verification, calibration, and error assessment are also widely used in the design and validation of industrial vision systems, which are used successfully in the automatic measurement of geometric dimensions.

In [7], an integrated system for the automatic real-time detection of metal edge features based on the YOLOv5 machine vision algorithm was presented. This solution was developed to automate the metal grinding and chamfering process. In turn, ref. [8] describes an industrial vision system intended for the automatic dimensioning of plastic components and the inspection of displays used in the automotive industry.

Compliance with the requirements of these standards and guidelines ensures measurement consistency, comparability of results, and traceability to the units of the International System of Units (SI). This is particularly important in industrial and laboratory applications, where results obtained using image recognition systems form the basis for product quality assessment, process diagnostics, and decision making related to quality control and equipment operational safety.

Automatic identification and reading of measurements from instruments play an increasingly important role in modern industrial systems, autonomous vehicles, and medical applications. Digital devices typically display multi-digit values, which makes it difficult to recognise them correctly using classical image processing methods, such as thresholding or pattern matching. The development of deep learning-based methods has enabled a significant improvement in the effectiveness of automatic meter reading; however, their application requires the preparation of large, properly labelled training datasets.

The article [9] presents an automatic calibration system designed for measuring instruments without a digital interface. The solution uses image analysis algorithms to automatically read measurements from both analogue and digital displays. The research conducted showed that the accuracy of determining the position of the pointer in analogue instruments is characterised by an uncertainty smaller than the resolving power of the human eye.

In [10], a method for the automatic reading of analogue meter readings was presented, based on a combination of the Faster Region-based Convolutional Neural Network (Faster

R-CNN) object detection algorithm with classical computer vision methods. The results of the study showed that the proposed method effectively performs the automatic reading of analogue meters in power substation environments and provides an effective solution for the detection and the interpretation of analogue meter readings.

Analogue instruments are used in systems that monitor the condition of power equipment and other utilities (including water and gas). In such monitoring systems, the automatic reading of analogue meter readings remains a significant challenge. To address this problem, numerous methods based on computer vision techniques have been developed. However, most of them are highly sensitive to changes in lighting conditions and require the camera to be positioned perpendicular to the meter dial, which limits their reliability and practical applicability.

In [11], a system was proposed for the automatic recognition of analogue readings, enabling accurate readings under various lighting conditions and for images captured at different angles. The results of the experiments confirm the high resistance to changes in lighting conditions and camera angle, which demonstrates its suitability for real-world industrial applications. This paper utilises the Hough transform, which is one of the fundamental methods used for detecting lines in digital images. It is based on the transformation of points from image space to parameter space and is used to identify measurement results in vision systems based on analogue instruments [11,12].

Automatic recognition of digital instrument readings plays a key role in the automation of measurement processes and in the implementation of intelligent monitoring and control systems. However, in real-world industrial conditions, recognition performance is significantly limited by adverse imaging conditions. A particular challenge is the simultaneous presence of dim, uneven lighting and reflections on instrument panel surfaces, which lead to partial obscuration or distortion of the displayed information. Traditional image processing methods, designed to improve visibility under low-light conditions, can indeed enhance image legibility, however, they often result in overexposure of bright areas and unwanted amplification of background noise, which consequently reduces the effectiveness of the subsequent recognition process.

In addition to standard measuring instruments for a broad range of electrical and non-electrical quantities, most modern home medical devices display measurement results on digital screens that lack the ability to automatically record data. This limits the ability to remotely monitor a patient's condition, integrate with telemedicine systems, and automatically maintain electronic medical records. In [13], the authors demonstrated the use of images from a smartphone camera and artificial intelligence methods to automatically extract and interpret data from various medical devices used by patients at home (blood pressure, glucose levels or oxygen saturation). The study compared the effectiveness of the YOLO and single-shot multibox detector (SSD) models in detecting and recognising information displayed on the screens of medical device screens. Additionally, a hybrid approach was analysed, combining the YOLOv8l model for object detection with a convolutional neural network (CNN) responsible for classifying the detected data.

The digital display on a blood pressure monitor allows measurement results to be presented numerically, however, manually reading and recording these values can lead to user errors and incorrect documentation. To address this issue, a method has been proposed to automatically read the display using the YOLOv8 model. This solution enables the recognition of systolic and diastolic blood pressure values, as well as heart rate, based on an image of the display. Automating the reading process reduces the risk of user errors and streamlines the monitoring and archiving of measurement results [14].

The studies cited above demonstrate that illumination conditions, reflections, and viewing geometry are relevant challenges in automatic instrument reading.

2. Motivation

Many measuring instruments used in metrological and industrial practice do not provide communication interfaces for direct data transfer and therefore still require manual reading of displayed values [15]. This limits the automation of measurement procedures, particularly during long-term measurements, and justifies the implementation of automated acquisition solutions [16]. A practical alternative is the use of non-invasive computer vision methods, in which a camera serves as a virtual communication interface and image-recognition algorithms automatically extract numerical indications from LCD or LED displays [17]. In the proposed system, the multimeter display is captured by a camera connected to a PC and processed automatically.

A key aspect in the design of this type of automated system is the selection of an appropriate programming environment for image acquisition and analysis. Two fundamentally different approaches were compared. The first approach was based on LabVIEW, a graphical programming environment dedicated to measurement and automation applications. A significant advantage of this environment is the possibility of rapid prototyping of measurement applications without the need to write extensive source code. LabVIEW enables the creation of a user interface in the form of a front panel and the development of programme logic by connecting functional blocks in a graphical block diagram. This approach facilitates the integration of measurement hardware and supports the development of virtual instruments used for data acquisition, processing, and visualisation [18]. The second approach was based on Python, a general-purpose programming language which, thanks to its extensive ecosystem of libraries, has also gained wide application in the field of computer vision and image processing. Based on a review of the literature, the YOLOv8 model was selected, as it is characterised by its established use in applied computer-vision studies, the availability of a mature implementation, and its moderate computational complexity.

The main contribution of this work is a measurement-oriented end-to-end evaluation approach to compare two heterogeneous optical reading implementations. Rather than treating the internal metrics of template-based OCR and object detection as directly equivalent, both pipelines reconstruct the recognised character sequences into numerical multimeter indications, which are then evaluated using the same statistical measures of within-series dispersion.

3. Selection of the Object Detection Methods

The Python-based implementation required a model for character-level object detection. The use of deep learning-based methods in automatic meter reading and instrument monitoring is also supported by the literature. In such systems, the image processing pipeline often includes display- or text-region detection followed by character recognition [19]. YOLO-based approaches have also been used to detect and capture indications from digital instruments, while few-shot learning and data augmentation methods can reduce the amount of manually annotated training data required for reliable instrument monitoring [20]. At the stage of model selection, YOLOv8 was also widely represented in the available scientific literature concerning automatic reading, object detection, and applied vision systems. This was one of the practical reasons for selecting this model family for the developed Python workflow.

The decision workflow shown in Figure 1 was used as an auxiliary guide during the selection of the object detection model. The figure was prepared by the authors based on the model-selection scheme presented by Ultralytics [21]. The workflow indicates that model selection should take into account the deployment conditions, the required speed–accuracy trade-off, and support for task-specific capabilities such as segmentation, pose estimation,

tracking, and oriented bounding-box detection, as well as architectural features such as end-to-end inference without non-maximum suppression (NMS). In the proposed application, these additional functionalities were not required. The task was limited to detecting a few character classes in a controlled laboratory setting. Therefore, the selection was narrowed to standard YOLO detection models intended for a balanced detection task.

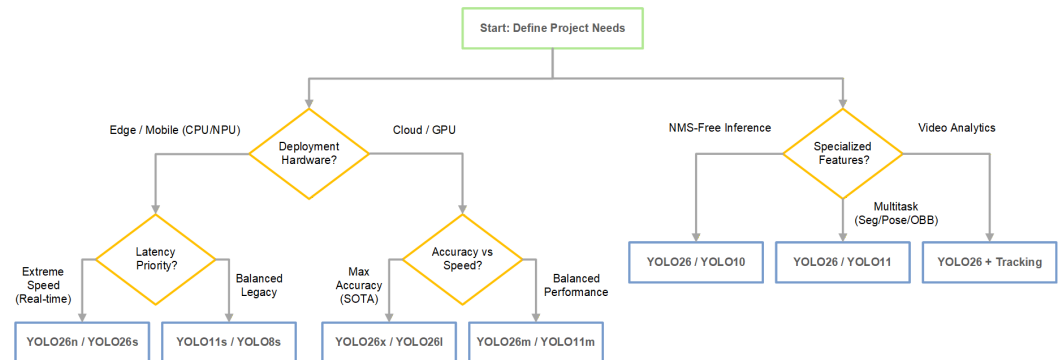


Figure 1. Decision workflow used during the selection of the Python-based object detection model.

The workflow shown in Figure 1 also indicates that both YOLOv8s and YOLO11s can be considered for balanced object detection tasks. In this study, YOLOv8s was selected because it was already well established in the literature and the Ultralytics ecosystem at the time of implementation. Moreover, the differences between successive YOLO generations were not the central research problem of this paper. The investigated task consisted of detecting a limited number of simple, repeatable, and visually regular character classes on a digital multimeter display. Therefore, architectural differences between YOLOv8s and newer models such as YOLO11s were not considered critical for the purpose of this study. The aim was not to determine the best YOLO generation, but to compare a Python-based YOLOv8s character-detection workflow with a LabVIEW template-based OCR workflow under the same laboratory acquisition conditions [22].

YOLOv8s was selected prior to the main experimental comparison as the detector implemented in the Python-based reading pipeline. The selection was guided by the established use of YOLOv8 in the available literature, the maturity of the Ultralytics implementation, its moderate computational requirements, and its compatibility with the available computing hardware [23].

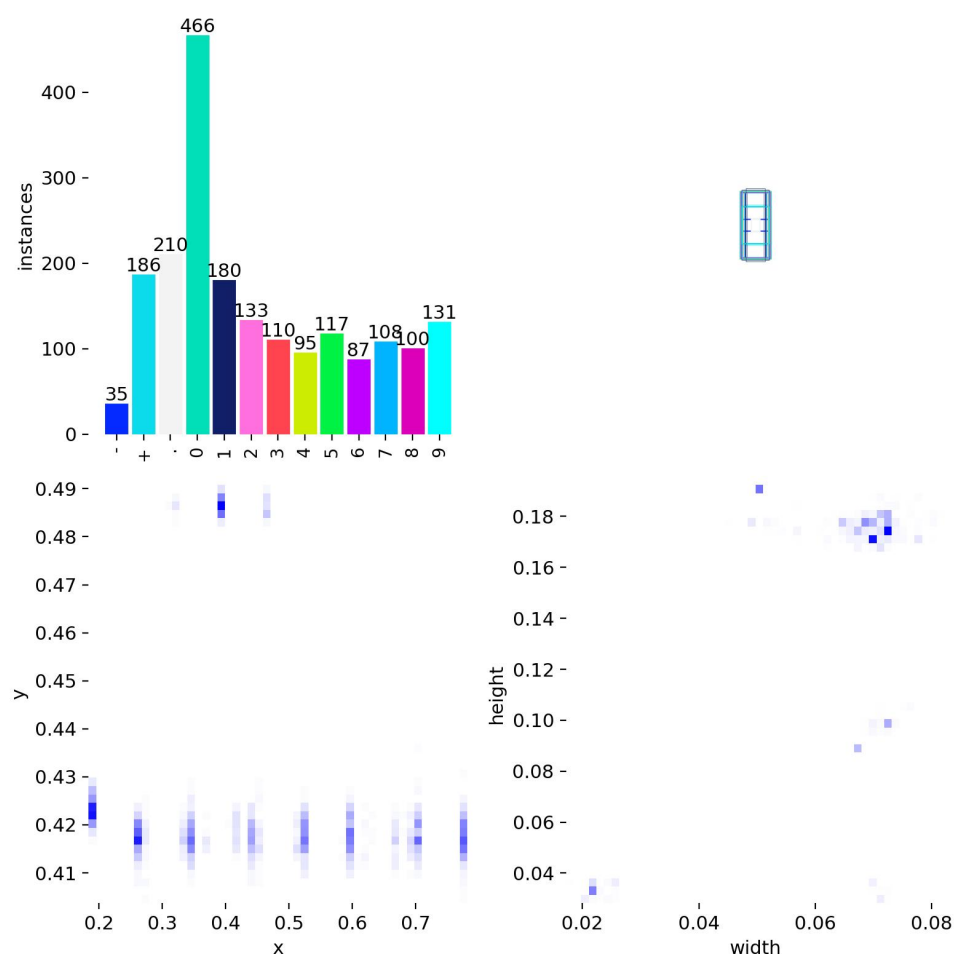
Table 1 summarises the benchmark characteristics of the YOLOv8 variants. The results are based on the official Ultralytics evaluation of the YOLOv8 detection models on the COCO dataset using an input resolution of 640 pixels. YOLOv8s provides a clear increase in detection performance compared to YOLOv8n, while its computational cost remains moderate. The increases from YOLOv8s to YOLOv8m, YOLOv8l, and YOLOv8x result in higher benchmark mAP₅₀₋₉₅ values, but also in a substantial increase in the number of parameters and FLOPs. In the considered application, this additional capacity of the model was not necessary because the model was trained to detect only a limited set of display characters under controlled acquisition conditions [24]. YOLOv8s was adopted as the implementation variant for the present comparison. The benchmark data are not treated as evidence that it is the optimal variant for the developed dataset.

Table 1. Published reference characteristics of the variants of the YOLOv8 detection model evaluated on the COCO dataset.

Model	mAP ₅₀₋₉₅	CPU ONNX, ms	A100 TensorRT, ms	Parameters, M	FLOPs, B
YOLOv8n	37.3	80.4	0.99	3.2	8.7
YOLOv8s	44.9	128.4	1.20	11.2	28.6
YOLOv8m	50.2	234.7	1.83	25.9	78.9
YOLOv8l	52.9	375.2	2.39	43.7	165.2
YOLOv8x	53.9	479.1	3.53	68.2	257.8

4. Training of a YOLOv8s Model in the Python Environment

The training dataset was prepared from images of the multimeter display acquired under laboratory conditions. The complete dataset consisted of 300 images and 2447 annotated character instances. Each image was manually annotated by marking the visible characters on the display. Before training, an automatic verification of data completeness was performed by checking the presence of corresponding image–label pairs. The dataset was then randomly divided into training and validation subsets in an 80:20 ratio. As shown in Figure 2, the training subset contained 1958 character instances distributed among 13 classes: digits from 0 to 9, decimal point, plus sign, and minus sign.

**Figure 2.** Distribution of annotated character instances and boundary box parameters in the YOLOv8 dataset.

The class distribution was not fully uniform, reflecting the actual occurrence of characters in the acquired measurement series. The digit 0 was the most frequent class, while the minus sign occurred the least often because most of the analysed measurements were performed for positive voltage values. The unequal class frequencies reflected the actual distribution of characters in the acquired multimeter-display images. The class distribution was therefore retained as a characteristic of the experimental dataset rather than artificially balanced before training.

The spatial distribution of bounding boxes showed that most characters were located in a narrow horizontal region corresponding to the multimeter display area. This confirmed that the dataset was geometrically consistent and that the detected characters appeared in repeatable positions in consecutive frames.

The YOLOv8s model was trained in the Python environment using the Ultralytics framework. The main training parameters are summarised in Table 2. Training was performed using pretrained weights, an input image size of 640 pixels $\times$ 640 pixels, a batch size of 4, and the AdamW optimiser. The model was trained for 200 epochs with early stopping patience set at 30 epochs.

During training, online data augmentation was applied using HSV-based image transformations, small rotations, translations, scaling, shear, and perspective transformations, as well as mosaic and mix-up. These operations were part of the training configuration and their individual contribution to the final reading repeatability was not evaluated separately. Horizontal and vertical flipping was disabled because digits and symbols on the multimeter display have a fixed orientation.

Table 2. Main parameters used during the YOLOv8s model training.

Parameter	Value
Model variant	YOLOv8s
Framework	Ultralytics YOLOv8, version 8.4.45
Training environment	Python 3.14.2
CPU	AMD Ryzen 7 4800H
RAM	16 GB RAM DDR4
GPU	NVIDIA GeForce GTX 1650, 4 GB VRAM
Input image size	640 px $\times$ 640 px
Maximum number of epochs	200
Completed epochs	94
Batch size	4
Optimizer	AdamW
Initial learning rate	0.001
Early stopping patience	30 epochs
Data augmentation	HSV transformations; rotation ($\pm 3^\circ$); translation (0.1); scaling (0.3); shear ($\pm 2^\circ$); perspective transformation (0.0003); mosaic (1.0); and mix-up (0.1)
Flipping	Disabled for both horizontal and vertical directions

The training curves presented in Figure 3 show a systematic decrease in training losses and convergence of the main validation metrics.

A slight increase in validation classification loss was observed during the final stage of training, while precision and recall remained at approximately 0.95 and 0.98–0.99, respectively. This behaviour indicates that further improvement in validation classification became limited and may suggest the beginning of mild overfitting.

The training curves are treated as diagnostics for the YOLOv8s model. Its practical behaviour in the developed reading pipeline is evaluated using the reconstructed numerical indications presented in Section 8.

The precision, recall, and confusion matrix reported in this section are supplementary diagnostics of the YOLOv8s training and validation process. They are not used as direct comparative metrics against the LabVIEW OCR implementation because the two methods are based on fundamentally different recognition mechanisms and do not provide equivalent internal output.

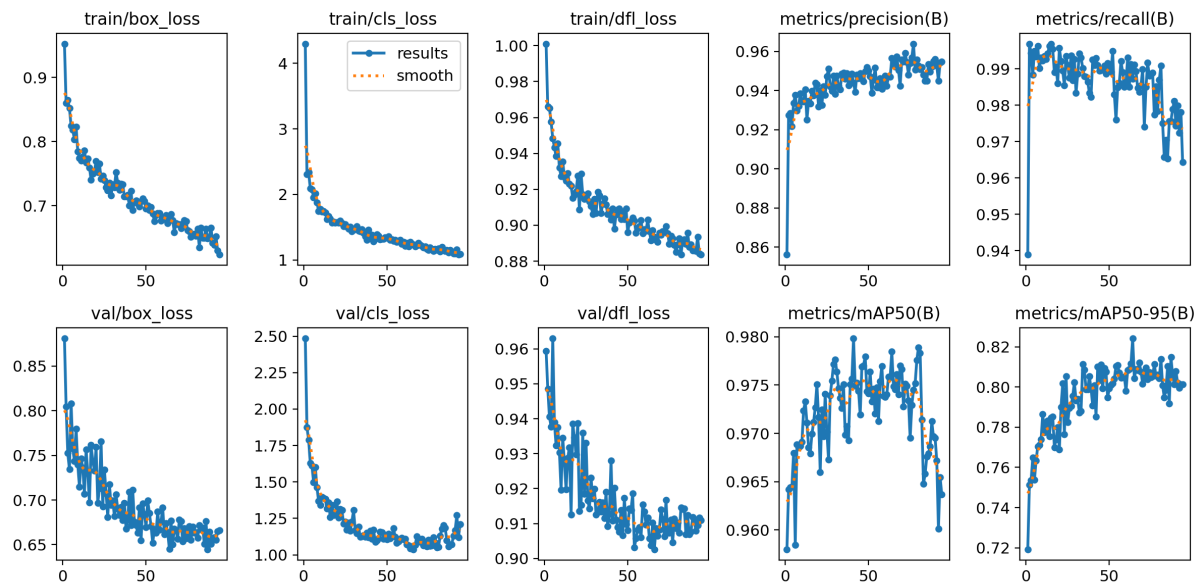


Figure 3. Training process of the YOLOv8s model.

Figure 4 shows example character detections obtained on multimeter-display images, including predicted classes, bounding boxes, and confidence scores.



Figure 4. Example results of character detection on the multimeter display obtained using the trained YOLOv8s model.

The class-wise performance of the trained YOLOv8s model was evaluated using a normalised confusion matrix. This matrix allows the identification of correctly recognised character classes as well as possible misclassifications between digits, signs, and the background class. The confusion matrix is presented in Figure 5.

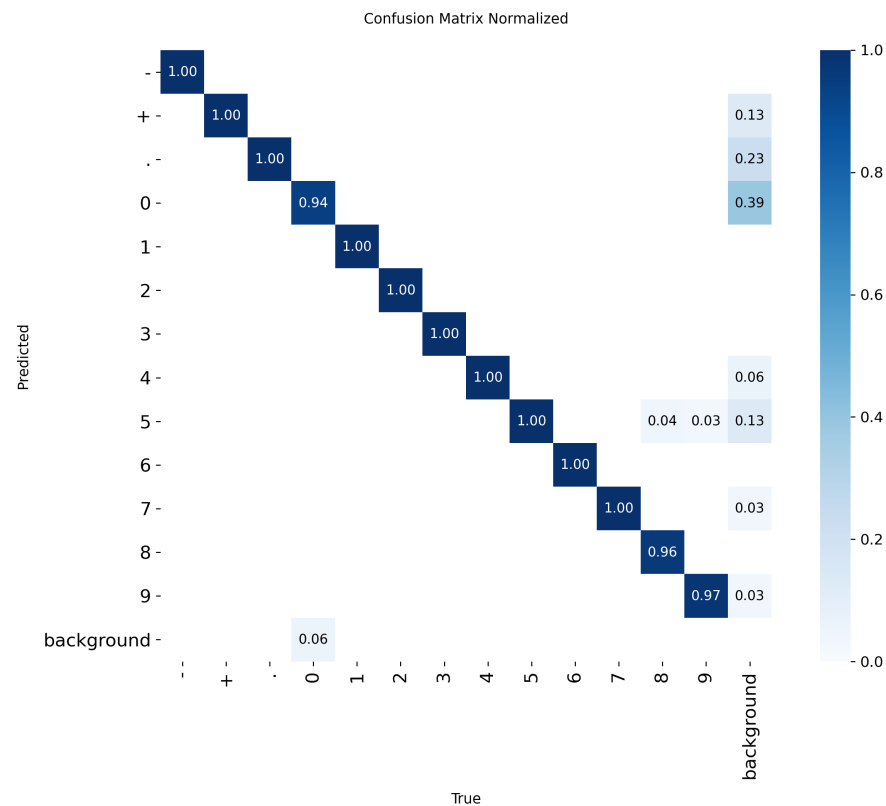


Figure 5. The confusion matrix of the model.

The matrix shows that most recognition results are concentrated on the main diagonal, which indicates correct classification of the majority of detected character classes. The minus sign, plus sign, decimal separator, and most digit classes achieved normalised diagonal values equal to 1.00, confirming that these classes were correctly separated by the trained model. Minor errors were observed only for selected digits, mainly 0, 8, and 9, and were related either to missed detections or to occasional confusion with visually similar character shapes. Some false detections were also associated with the background class, indicating that selected display artefacts could be interpreted as valid characters.

The confusion matrix was used to examine the class-wise validation behaviour of the YOLOv8s detector. However, it does not directly determine the numerical consequence of a classification or detection error because this consequence depends on the type and position of the affected character.

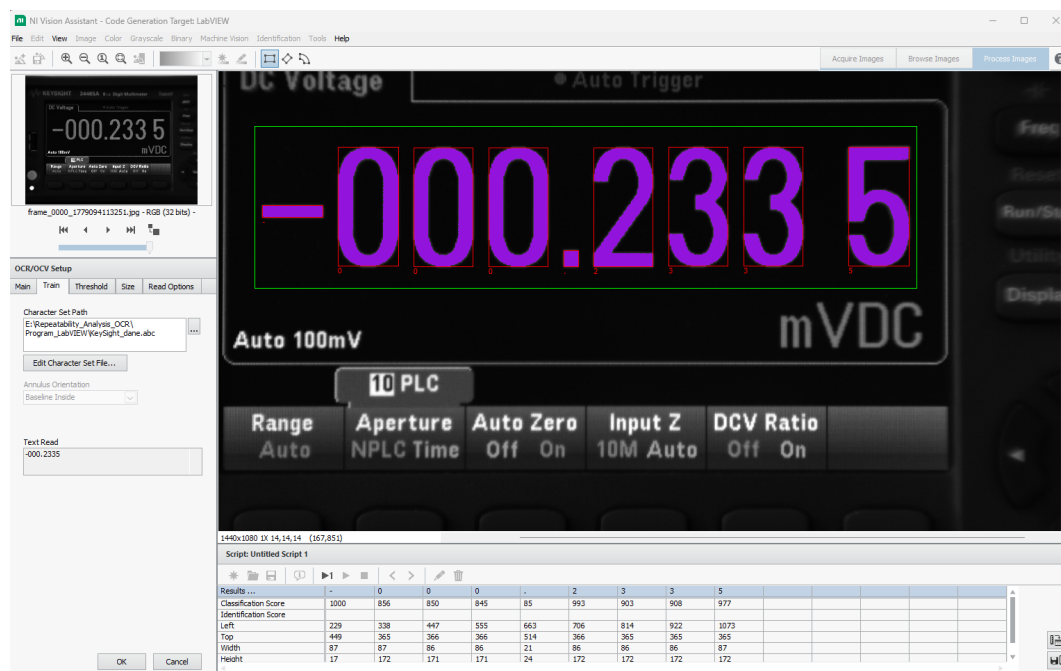
5. Configuration of the OCR in LabVIEW Vision Assistant

The OCR character set was prepared in NI Vision Assistant using multimeter-display images acquired during preliminary measurements. The OCR parameters applied in the final implementation are listed in Table 3.

Table 3. Main parameters used in the LabVIEW OCR-based reading method.

Parameter	Value
Software environment	LabVIEW with NI Vision Assistant
Image processing module	OCR Setup
Input image format	RGB image, 32-bit
Image resolution	1440 px × 1080 px
Region of interest	Multimeter display area containing the numerical indication
Thresholding mode	Auto: Linear
Character type	Light characters on dark background
Number of threshold blocks	4
Read strategy	Conservative
Read resolution	High
Acceptance level	700
Multiline and rotation detection	Auto-detect
Separate character algorithm	None
Minimum character size	20 px
Minimum character spacing	1 px
Horizontal element spacing	1 px
Vertical element spacing	0 px

Figure 6 presents an example character set prepared for the recognition of digital multimeter indications. The figure shows samples of individual characters, assigned to their corresponding classes.

**Figure 6.** View of the NI Vision Assistant OCR character set editor.

Comparison of Recognition Assumptions for the Analysed Methods

To clarify the differences between the LabVIEW-based OCR method and the Python-based YOLOv8s method, their recognition principles, training requirements, handling of input variability, dependence on image processing parameters, and practical implementation limitations were compared. The comparison is presented in Table 4.

Table 4. Comparison of recognition assumptions for the LabVIEW OCR and Python YOLOv8s methods.

Aspect	LabVIEW OCR	Python YOLOv8s
Recognition principle	Template matching based on manually prepared OCR character samples.	Object detection model trained to detect individual characters on the multimeter display.
Recognised classes	Digits 0–9, decimal separator, plus and minus signs.	Digits 0–9, decimal separator, plus and minus signs.
Training data	Representative character templates prepared manually in the NI Vision Assistant OCR.	Annotated images of the multimeter display containing labelled character bounding boxes.
Templates or annotations per class	At least one representative template was prepared for each character class; additional templates were added only when needed.	Multiple annotated instances were used for each class, depending on their occurrence in the image set.
Handling of variability	Limited mainly by fixed camera position, constant ROI, and stable illumination conditions.	Handled by the training dataset and data augmentation, including brightness changes, small rotations, translations, and scaling.
ROI dependence	Incorrect ROI placement could remove part of the indication or include unnecessary display elements.	The model detected characters within the analysed image region, although proper display framing was still required.
Thresholding dependence	The retention parameters directly affected character segmentation and recognition quality.	No explicit thresholding was required during recognition.
Reading reconstruction	The recognised characters were ordered according to their position and combined into a complete numerical indication.	The detected characters were ordered according to their bounding-box position and combined into a complete numerical indication.
Main limitation	Requires stable acquisition conditions, fixed ROI placement, and properly adjusted OCR parameters.	Requires an annotated dataset and model training prior to use.

6. Experimental Measurement Setup

The experimental measurement setup presented in Figure 7 was prepared to compare the optical reading methods obtained using the two independent image processing approaches. The reference values were generated by a Fluke 5522A calibrator (Fluke Corporation, Everett, WA, USA). and applied to a Keysight 34465A digital multimeter (Keysight Technologies, Inc., Santa Rosa, CA, USA). The indication displayed by the multimeter was recorded with a Basler acA1440-220um monochrome industrial camera (Basler AG, Ahrensburg, Germany). The camera was connected to a personal computer (PC) via a universal serial bus (USB) interface, and the acquired images were processed independently in Python and LabVIEW environments.

During the measurements, the camera observed only the selected ROI containing the numerical indication of the multimeter. Limiting the analysed area to the display reduced the influence of irrelevant parts of the image and simplified further processing. Both methods were evaluated in separate acquisition series using the same camera, region of interest, measurement settings, and laboratory arrangement. The comparison therefore con-

cerns the repeatability of two complete optical reading pipelines operated under matched experimental conditions.

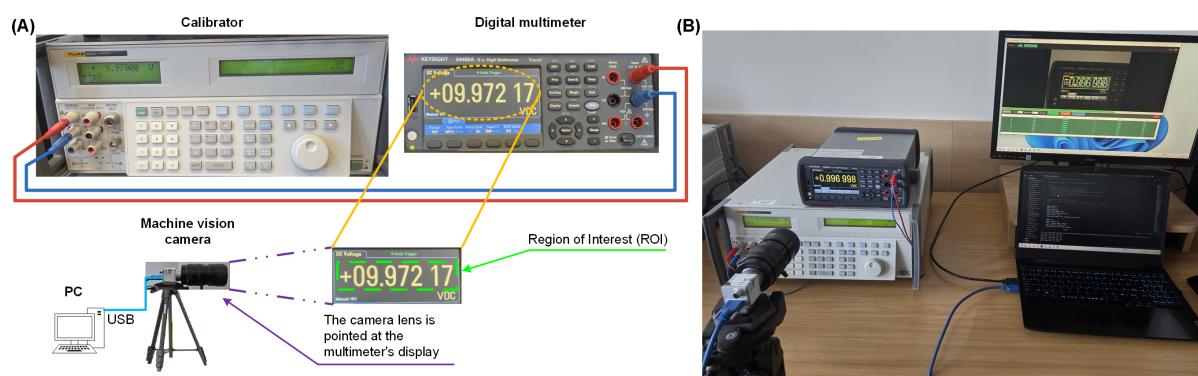


Figure 7. Experimental measurement stand used for optical reading of multimeter indications: (A) schematic representation of the system; (B) physical laboratory setup.

The basic camera parameters relevant to the image acquisition process are summarized in Table 5. The technical data presented in the table were prepared based on the manufacturer's documentation [25].

Table 5. Technical specifications of the Basler acA1440-220um camera.

Parameter	Value
Camera model	Basler acA1440-220um
Sensor	Sony IMX273 *
Shutter type	Global Shutter
Sensor format	1/2.9"
Sensor type	CMOS
Sensor size	5 mm × 3.7 mm
Resolution	1440 px × 1080 px
Resolution	1.6 MP
Pixel size	3.45 μm × 3.45 μm
Frame rate	227 fps
Camera type	Monochrome

* Sony Semiconductor Solutions Corporation, Atsugi, Kanagawa, Japan.

The camera was equipped with a KOWA LMZ68M zoom lens (Kowa Company, Ltd., Nagoya, Aichi, Japan), which allows adjustment of the focal length and manual focusing. The basic technical parameters of the lens are summarised in Table 6. The lens specifications were obtained from the referenced product documentation [26].

Table 6. Technical specifications of the KOWA LMZ68M lens.

Parameter	Value
Lens model	KOWA LMZ68M
Format	1/2"
Focal length	8–48 mm
Aperture	F1.0
Iris type	Manual iris
Mount	C-mount

7. Implementation of Image Acquisition and Reading Algorithms

After defining the common measurement setup, complete software procedures were developed for image acquisition, character recognition, result reconstruction, and data storage. Although both solutions operated on images obtained from the same camera, their internal processing workflows were implemented separately in LabVIEW 2021 SP1 and Python 3.13.13. The following subsections describe the structure of each algorithm and the way in which the recognised display indications were recorded for further analysis.

7.1. LabVIEW-Based Reading Algorithm

In this approach, image acquisition and character recognition were implemented using National Instruments Vision Development Modules. The algorithm included camera selection, continuous image acquisition, retrieval of frames from the image buffer, and processing of the selected region of interest containing the multimeter display. Within this region, the OCR session was applied to recognise the characters displayed repeatedly during the measurement loop. After the required number of samples had been acquired, the session was closed and the recognised readings were saved to a file for further analysis. Figure 8 illustrates the sequential structure of the LabVIEW implementation, from camera initialisation to automatic recording of the recognised multimeter readings.

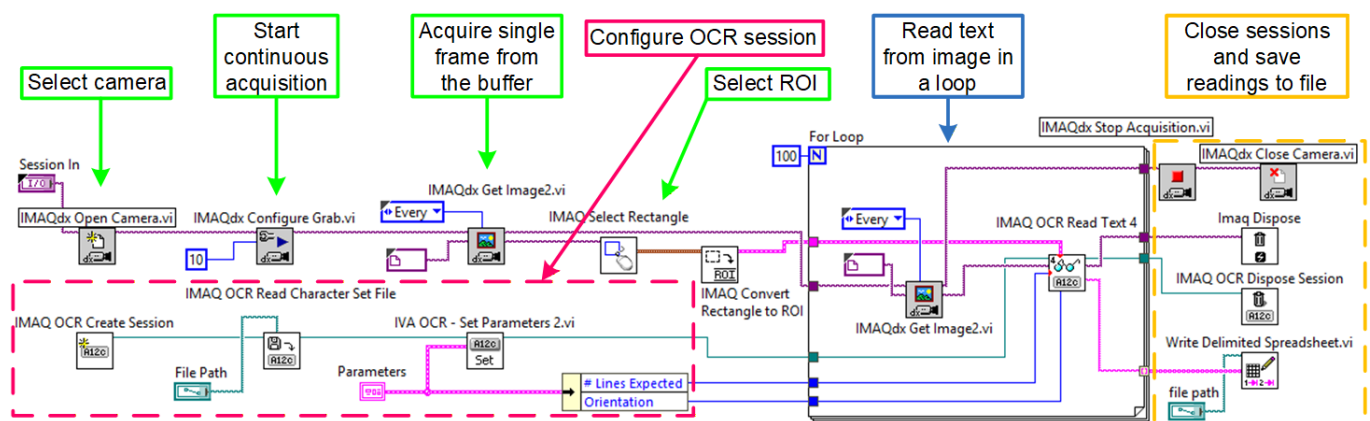


Figure 8. Block diagram of the image acquisition and OCR-based multimeter-reading algorithm implemented in LabVIEW.

The block diagram presents the general structure of the LabVIEW implementation, and indicates the main stages of the OCR-based reading procedure. However, to clarify the exact order of operations performed during a single measurement sequence, the workflow was additionally described in the form of pseudocode. Algorithm 1 summarises the initialization, acquisition, OCR recognition, data storage, and resource release steps used in the LabVIEW-based method.

Algorithm 1 LabVIEW template-based OCR reading workflow

Input: Number of readings N ; camera identifier C ; camera acquisition parameters P_{cam} ; OCR character-set file K ; OCR session parameters P_{OCR} ; output file path f_{out}

Output: Result table $T = \{(i, s_i)\}_{i=1}^N$, where s_i is the OCR text obtained from the i th acquired frame

- 1: Open camera C .
- 2: Configure continuous image acquisition using P_{cam} .
- 3: Create OCR session O .
- 4: Load the OCR character-set file K into session O .
- 5: Apply the OCR session parameters P_{OCR} .
- 6: Acquire a reference image frame I_0 from the camera buffer.
- 7: Display I_0 and allow the user to select a rectangular display area Q .
- 8: Convert the selected rectangle Q into ROI descriptor R .
- 9: Initialise an empty result table T .
- 10: **for** $i = 1$ to N **do**
- 11: Acquire the latest image frame I_i from the camera buffer.
- 12: Read OCR text s_i from frame I_i within ROI R using session O .
- 13: Append (i, s_i) to result table T .
- 14: **end for**
- 15: Stop image acquisition.
- 16: Close camera C .
- 17: Dispose of the image resources.
- 18: Dispose of OCR session O .
- 19: Export result table T to the delimited spreadsheet file f_{out} .

7.2. Python-Based Reading Algorithm

The algorithm was integrated with a graphical user interface (GUI), which allowed the user to configure the measurement parameters, load the detection model, initialise the camera, and select the region of interest corresponding to the display area of the multimeter (Figure 9).

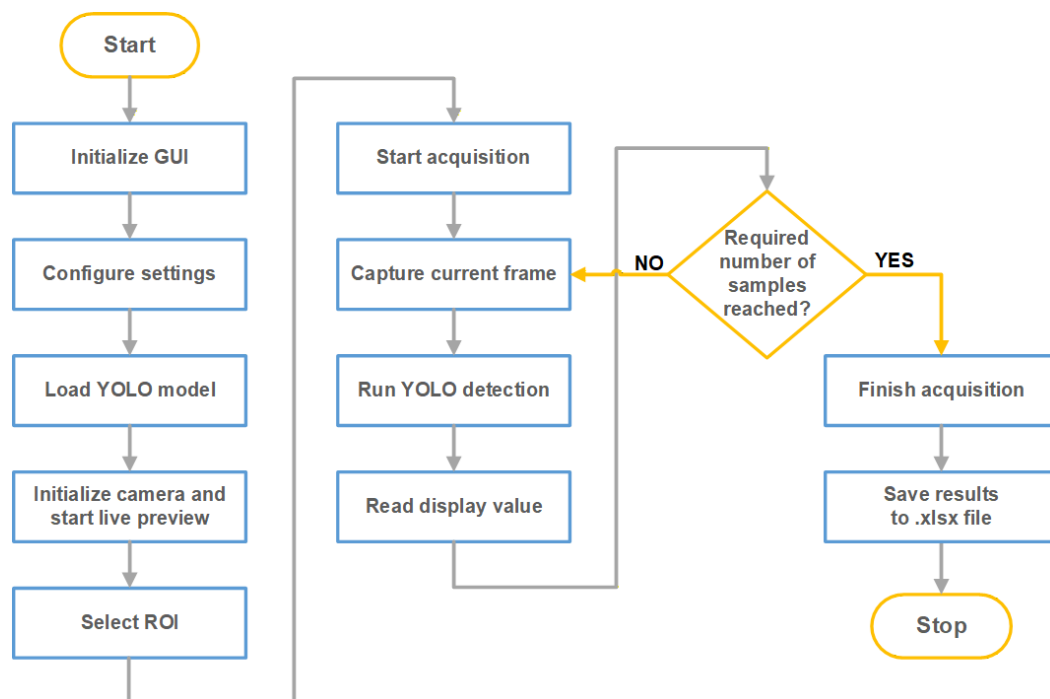


Figure 9. Flowchart of the Python-based multimeter-reading algorithm using the YOLOv8s model.

After the configuration stage, image acquisition was started and individual frames were captured from the camera. Each acquired frame was processed by the model to

detect individual display characters. The detected characters were then sorted according to their position on the horizontal axis and combined into a complete numerical value. This procedure was repeated in a loop until the required number of samples was reached. After completion of the acquisition results, the obtained data was saved to an .xlsx file for further statistical analysis and comparison with the results obtained in LabVIEW.

The block diagram presents the general structure of the Python-based implementation, including GUI configuration, camera initialisation, ROI selection, character detection, and results export. To provide a more detailed description of the processing sequence, the complete workflow was also expressed in the form of pseudocode. Algorithm 2 summarises the main steps of the YOLOv8s-based reading method, including image acquisition, optional ROI extraction, character detection, confidence-based filtering, left-to-right ordering of detected symbols, numerical reconstruction, and export of the results to an .xlsx file.

Algorithm 2 Python-based YOLOv8s multimeter-reading workflow

Input: Number of analysed images N ; camera identifier C ; trained YOLOv8s model file p_M ; confidence thresholds Θ ; IoU threshold τ_{IoU} ; multi-scale input sizes $S = \{640, 320\}$; output file path f_{out}

Output: Sequence of reconstructed indications $T = \{(i, r_i)\}_{i=1}^N$ exported to an XLSX file

- 1: Initialise the GUI.
 - 2: Configure N , camera C , confidence thresholds Θ , the IoU threshold τ_{IoU} , and the output file path f_{out} .
 - 3: Load the trained YOLOv8s model M from p_M .
 - 4: Open camera C and start the live preview.
 - 5: Allow the user to select the display region of interest R .
 - 6: Initialise an empty result table T .
 - 7: Start the measurement acquisition.
 - 8: **for** $i = 1$ to N **do**
 - 9: Acquire the current image frame I_i .
 - 10: Apply CLAHE to I_i and extract ROI R , obtaining J_i .
 - 11: Run multi-scale YOLOv8s inference on J_i using $S = \{640, 320\}$.
 - 12: Filter the detections using Θ and merge the results using non-maximum suppression with threshold τ_{IoU} , obtaining D_i .
 - 13: Sort the detections in D_i from left to right.
 - 14: Reconstruct the multimeter indication r_i from the ordered character labels.
 - 15: Append (i, r_i) to T .
 - 16: **end for**
 - 17: Finish the measurement acquisition.
 - 18: Export T to the XLSX file f_{out} .
 - 19: Stop the live preview and close camera C .
 - 20: Close the GUI.
-

8. Results

The results were analysed for selected direct current (DC) and alternating current (AC) voltage ranges. In the case of DC measurements, two power line cycle (PLC) settings were additionally considered, evaluating the influence of integration time on reading stability.

The comparison was performed at the common output level of both complete optical reading pipelines, namely, the reconstructed numerical indications obtained from the multimeter display. Both implementations were evaluated using the same measurement configurations and descriptive statistical parameters to compare the within-series dispersion of the readings.

The limits of the tolerance band correspond to the maximum permissible error (MPE) of the digital multimeter, denoted in this work as Δ_{mpe} . According to the manufacturer's

datasheet, the accuracy specification is expressed as $\pm(a\% \text{ of reading} + b\% \text{ of range})$ [27]. Therefore, the corresponding absolute limit, expressed in volts, was calculated as:

$$\Delta_{\text{mpe}} = \pm \left(\frac{a}{100} V_{\text{set}} + \frac{b}{100} V_{\text{range}} \right), \quad (1)$$

where a is the percentage-of-reading coefficient, b is the percentage-of-range coefficient, V_{set} is the voltage applied by the calibrator, and V_{range} is the selected measurement range of the digital multimeter.

Table 7 summarises the calibrator and multimeter settings applied in each measurement case. These conditions were kept the same for the LabVIEW-based method and the Python-based solution, which enabled a consistent comparison of the obtained readings.

Table 7. Reference settings and multimeter accuracy limits used during tests.

Measurement Case	Calibrator Setting	Multimeter Range	PLC Setting	Accuracy ($p_{\text{rdg}} + p_{\text{rng}}$)	Δ_{mpe}
DC voltage	0.980 V	1 V	10 PLC	0.0105 + 0.0010%	± 0.000113 V
DC voltage	0.980 V	1 V	1 PLC	0.0105 + 0.0010%	± 0.000113 V
AC voltage	0.997 V	1 V	—	0.07 + 0.03%	± 0.000998 V

The calculated values of Δ_{mpe} were used to define the shaded tolerance bands shown in the corresponding measurement plots.

8.1. DC Voltage Measurement at 1 V Range with 10 PLC Setting

Figure 10 presents a comparison of the values acquired using the two analysed methods for the DC 1 V measurement with the 10 PLC setting, together with the shaded tolerance band representing the digital multimeter accuracy limit Δ_{mpe} .

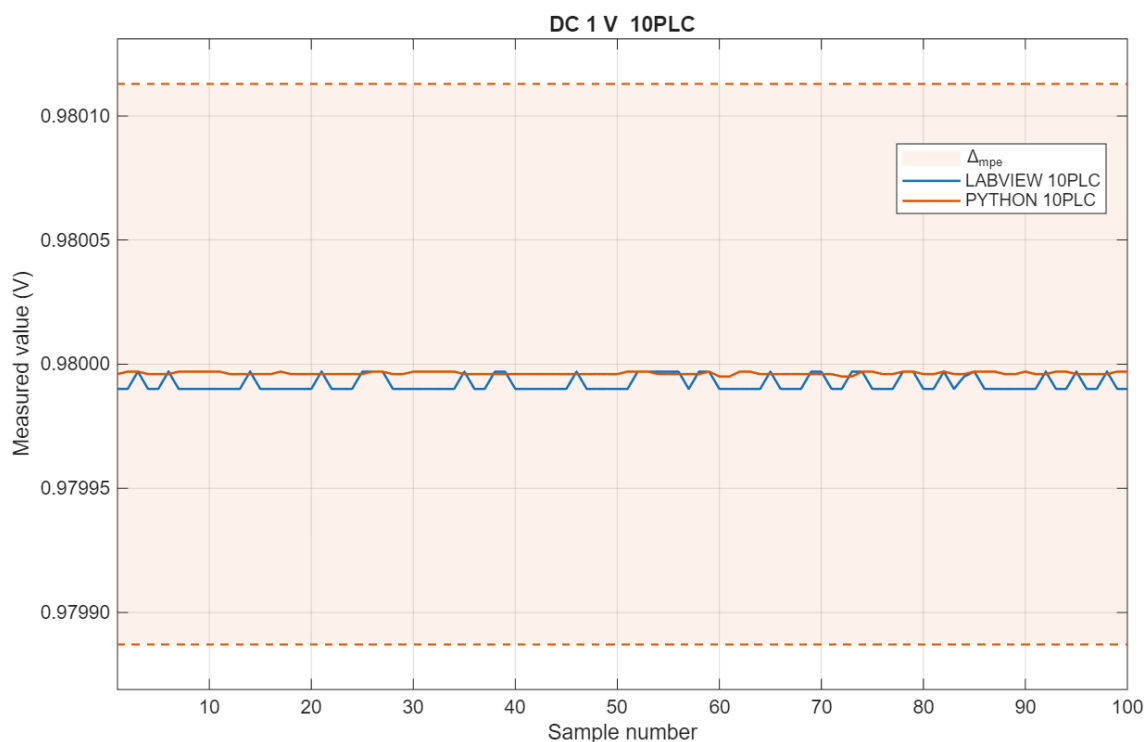


Figure 10. Comparison of results for the DC voltage measurement at 1 V with the 10 PLC integration setting.

Both algorithms provided stable readings, and the obtained values remained close to the reference indication throughout the entire measurement series. However, slight differences between the methods can be observed. The Python-based solution produced readings concentrated around one dominant value, whereas the LabVIEW-based method changed more frequently between two adjacent displayed values. This may indicate that, in this measurement case, the Python algorithm was less sensitive to small fluctuations or transitions visible on the multimeter display.

The statistical results presented in Table 8 confirm the visual observations from Figure 10. Both methods provided stable readings close to the reference value. However, the Python-based method achieved a lower standard deviation and a smaller range of values than the LabVIEW-based method. This indicates slightly higher repeatability for the Python readings for the DC 1 V measurement with the 10 PLC setting.

Table 8. Statistical summary of results for the DC 1 V measurement with the 10 PLC setting.

Parameter	LabVIEW 10 PLC	Python 10 PLC
Number of samples	100	100
Mean value, V	0.979992	0.979996
Median, V	0.979990	0.979996
Minimum, V	0.979990	0.979995
Maximum, V	0.979997	0.979997
Standard deviation, V	3.230×10^{-6}	5.412×10^{-7}
Variance, V ²	1.043×10^{-11}	2.929×10^{-13}
Type A uncertainty, V	3.230×10^{-7}	5.412×10^{-8}
Range, V	0.000007	0.000002

8.2. DC Voltage Measurement in the 1 V Range with 1 PLC Setting

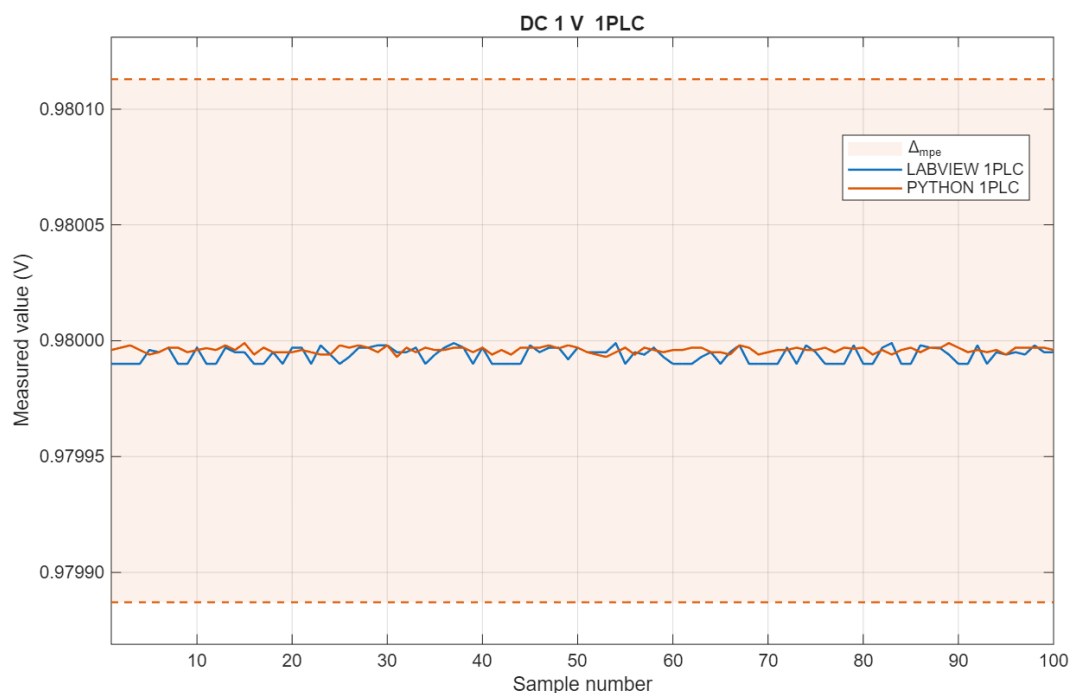
Compared with the 10 PLC configuration, noticeably larger fluctuations can be observed for the 1 PLC setting presented in Figure 11. This behaviour is consistent with expectations, since a shorter integration time provides less averaging of the measured signal and therefore results in lower reading stability. Nevertheless, for the DC 1 V measurement with the 1 PLC setting, both the LabVIEW and Python-based readings remained within the tolerance band defined by Δ_{mpe} .

However, the comparison of both methods shows that the Python-based solution produced smaller deviations than the LabVIEW-based method. The LabVIEW readings changed more frequently between adjacent displayed values, whereas the Python readings were more concentrated around the main indication level. This suggests that, for this measurement case, the Python-based algorithm was slightly less sensitive to short-term fluctuations visible on the multimeter display.

The statistical results presented in Table 9 confirm that the 1 PLC setting resulted in larger fluctuations than the 10 PLC configuration. However, the Python-based method achieved a lower standard deviation and a smaller range of values than the LabVIEW-based method. This indicates that, for the DC 1 V measurement with the 1 PLC setting, the Python solution provided slightly higher repeatability of the optical reading.

Table 9. Statistical summary of results for the DC 1 V measurement with the 1 PLC setting.

Parameter	LabVIEW 1 PLC	Python 1 PLC
Number of samples	100	100
Mean value, V	0.979994	0.979996
Median, V	0.979995	0.979996
Minimum, V	0.979990	0.979993
Maximum, V	0.979999	0.979999
Standard deviation, V	3.258×10^{-6}	1.317×10^{-6}
Variance, V ²	1.061×10^{-11}	1.734×10^{-12}
Type A uncertainty, V	3.258×10^{-7}	1.317×10^{-7}
Range, V	0.000009	0.000006

**Figure 11.** Comparison of results for the DC voltage measurement at 1 V with the 1 PLC integration setting.

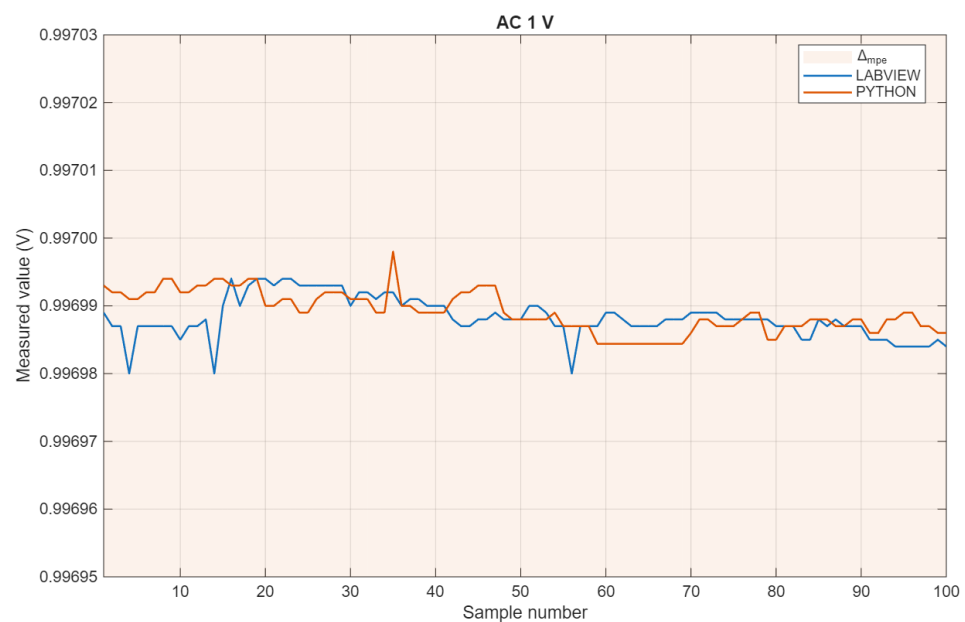
8.3. AC Voltage Measurement in the 1 V Range

The results for the 1 V AC voltage measurement are presented in Figure 12. In this case, the measured values also remained within a narrow range, although a slight variation between subsequent samples can be observed. Despite these small fluctuations, all measured values remained within the tolerance band defined by the digital multimeter accuracy limit Δ_{mpe} for the AC 1 V range. Both reading methods produced similar results, but local differences between the LabVIEW and Python readings were visible in some parts of the measurement series. These differences may result from frame acquisition, character recognition uncertainty, or small changes in the displayed indication during the measurement process.

The statistical results presented in Table 10 show that both the analysed methods achieved very similar repeatability for the AC 1 V measurement. The mean and median values obtained using the LabVIEW-based method and the Python-based method were almost identical. The standard deviation and variance were also at a comparable level, while the same range of measured values was observed for both approaches. This indicates that, for the analysed AC voltage case, both algorithms provided similar reading stability and comparable repeatability of the optical multimeter indication reading.

Table 10. Statistical summary of results for the AC 1 V measurement.

Parameter	LabVIEW	Python
Number of samples	100	100
Mean value, V	0.996988	0.996989
Median, V	0.996988	0.996989
Minimum, V	0.996980	0.996984
Maximum, V	0.996994	0.996998
Standard deviation, V	2.994×10^{-6}	2.978×10^{-6}
Variance, V ²	8.962×10^{-12}	8.867×10^{-12}
Type A uncertainty, V	2.994×10^{-7}	2.978×10^{-7}
Range, V	0.000014	0.000014

**Figure 12.** Comparison of results for the AC voltage measurement at 1 V.

8.4. Summary of Results Obtained for the Analysed Voltage Ranges

In addition to the detailed analysis of selected measurement cases, the statistical parameters obtained for all analysed voltage ranges were summarised in a compact form. The measurement cases are presented in columns, while the rows contain the main statistical parameters calculated separately for the LabVIEW-based and Python-based reading methods. This layout allows for a direct comparison of both approaches for different voltage ranges and PLC settings. The general statistical summary for all the measured cases is presented in Table 11. The table includes the number of samples, mean value, median, minimum, maximum, standard deviation, variance, and range calculated separately for the LabVIEW-based and Python-based reading methods.

The differences between the methods are visible mainly in the standard deviation and range of values, which describe the repeatability of the obtained readings. For several measurement cases, especially for the DC 1 V range with the 10 PLC setting, the Python-based method produced lower dispersion than the LabVIEW-based method.

Table 11. Overall statistical summary of results for the analysed voltage ranges.

Parameter	Method	AC 100 mV	AC 1 V	AC 10 V	DC 100 mV 1 PLC	DC 1 V 1 PLC	DC 10 V 1 PLC	DC 100 mV 10 PLC	DC 1 V 10 PLC	DC 10 V 10 PLC
Number of samples	LabVIEW Python	100 100	100 100	100 100	100 100	100 100	100 100	100 100	100 100	100 100
Mean	LabVIEW Python	99.671140 99.669569	0.996988 0.996989	9.997396 9.997429	97.997830 97.997927	0.979994 0.979996	9.998009 9.998050	97.997536 97.997738	0.979992 0.979996	9.998098 9.998095
Median	LabVIEW Python	99.671000 99.669000	0.996988 0.996989	9.997410 9.997420	97.998000 97.998000	0.979995 0.979996	9.998010 9.998020	97.997700 97.997700	0.979990 0.979996	9.998100 9.998100
Minimum	LabVIEW Python	99.670000 99.660000	0.996980 0.996984	9.997000 9.997370	97.997000 97.997000	0.979990 0.979993	9.997980 9.997980	97.997000 97.997000	0.979990 0.979995	9.998000 9.998000
Maximum	LabVIEW Python	99.673000 99.672000	0.996994 0.996998	9.997470 9.997744	97.998000 97.998700	0.979999 0.979999	9.998040 9.998400	97.998000 97.997900	0.979997 0.979997	9.998200 9.998100
Standard deviation	LabVIEW Python	7.788×10^{-4} 1.916×10^{-3}	2.994×10^{-6} 2.978×10^{-6}	5.238×10^{-5} 3.903×10^{-5}	3.775×10^{-4} 2.835×10^{-4}	3.258×10^{-6} 1.317×10^{-6}	9.587×10^{-6} 6.082×10^{-5}	3.274×10^{-4} 9.984×10^{-5}	3.230×10^{-6} 5.412×10^{-7}	2.454×10^{-5} 2.190×10^{-5}
Variance	LabVIEW Python	6.065×10^{-7} 3.671×10^{-6}	8.962×10^{-12} 8.867×10^{-12}	2.743×10^{-9} 1.524×10^{-9}	1.425×10^{-7} 8.037×10^{-8}	1.061×10^{-11} 1.734×10^{-12}	9.192×10^{-11} 3.699×10^{-9}	1.072×10^{-7} 9.967×10^{-9}	1.043×10^{-11} 2.929×10^{-13}	6.020×10^{-10} 4.798×10^{-10}
Type A uncertainty	LabVIEW Python	7.788×10^{-5} 1.916×10^{-4}	2.994×10^{-7} 2.978×10^{-7}	5.238×10^{-6} 3.903×10^{-6}	3.775×10^{-5} 2.835×10^{-5}	3.258×10^{-7} 1.317×10^{-7}	9.587×10^{-7} 6.082×10^{-6}	3.274×10^{-5} 9.984×10^{-6}	3.230×10^{-7} 5.412×10^{-8}	2.454×10^{-6} 2.190×10^{-6}
Range	LabVIEW Python	0.003000 0.012000	0.000014 0.000014	0.000470 0.000374	0.001000 0.001700	0.000009 0.000006	0.000060 0.000420	0.001000 0.000900	0.000007 0.000002	0.000200 0.000100

9. Discussion

The results obtained confirm that image-based reading of digital multimeter indications can support measurement automation, particularly in laboratories where instruments without direct communication interfaces are still used. This is relevant to LabbiKal, an accredited testing and calibration laboratory operating within the Department of Metrology and Diagnostic Systems at Rzeszów University of Technology, where handheld multimeters requiring manual readout are frequently submitted for calibration. The proposed approach could therefore reduce the need for manual transcription and support partial automation of the calibration procedure.

The proposed optical reading approach addresses this limitation by using a camera and image processing algorithms as a non-invasive interface between the measuring instrument and the computer. The instrument does not have to be modified electrically or mechanically, and its display can be read in a manner similar to visual observation by an operator. The comparison of the LabVIEW-based OCR method and the Python-based YOLOv8s method showed that both approaches were capable of automatically acquiring repeatable numerical indications from the multimeter display. This confirms the practical applicability of optical character recognition and object detection methods for automating data acquisition from instruments that were not originally designed for computer-based measurement systems.

For the analysed DC voltage measurements, the Python-based pipeline generally produced reconstructed indications that were less dispersed than the LabVIEW-based pipeline. A lower standard deviation was obtained in five of the six DC measurement cases, while a smaller range was observed in four of the cases. The clearest improvement occurred for the 1 V range, for which the Python-based implementation provided more repeatable optical readings at both PLC settings. Because the implementations were evaluated in separate acquisition series, these results characterise the behaviour of the complete acquisition and recognition pipelines and cannot be attributed exclusively to the internal recognition mechanism.

For the analysed AC voltage measurements, both implementations achieved very similar standard deviations and ranges. This indicates that the relative advantage of a particular optical reading pipeline depends on the measurement configuration and on the variability of the indications presented by the multimeter.

From a practical point of view, the developed systems may be useful for supporting calibration procedures, repeatability tests, and laboratory measurements carried out in environments such as LabbiKal. Optical reading can extend the possibilities of automating measurements performed with older or simpler instruments that do not provide digital communication interfaces. Such systems can reduce the need for manual transcription, support continuous data recording, and facilitate the preparation of measurement documentation.

In addition to the statistical characteristics of the reconstructed indications, the practical implementation aspects of both approaches should also be considered. Although both methods can be used for automatic reading of digital multimeter indications, they differ in terms of software accessibility, deployment, required environment, scalability, and support for code development.

The comparison presented in Table 12 supports the selection of an optical reading implementation according to the intended laboratory scenario. The LabVIEW OCR method is particularly suitable for laboratories that already use NI hardware, LabVIEW measurement applications, and NI Vision tools, especially when only a limited number of instruments with stable display geometry must be supported and rapid integration with an existing graphical application is required. The Python/YOLOv8s method is more suitable when an open and extensible environment is preferred, when further machine learning or image processing functions are planned, or when the system is expected to be adapted to additional instruments and display layouts. When both environments are available, the selection should be based on a preliminary end-to-end evaluation that considers the repeatability of reconstructed indications, integration with existing equipment, software availability, implementation effort, maintainability, and the effort required to prepare OCR templates or annotated training data.

However, some limitations should be considered. The quality of optical reading depends on camera positioning, lighting conditions, display contrast, reflections, and the correct selection of the region of interest.

In the LabVIEW implementation, recognition performance depends on the preparation of the OCR character set and the configuration of segmentation and thresholding parameters. In the Python implementation, performance depends on the composition and annotation quality of the training dataset and on the selected model configuration.

Future work should focus on evaluating both optical reading pipelines under systematically varied acquisition conditions. The planned experiments should include changes in illumination intensity, non-uniform lighting, and reflections from the display surface. To ensure repeatable and quantitatively controlled lighting conditions, a dedicated experimental setup should be developed using a lux meter and adjustable light sources.

Another important research direction is the evaluation of the influence of camera viewing angle and displacement on display localisation, ROI positioning, and final character-recognition performance. Such tests would make it possible to determine the sensitivity of both pipelines to deviations from the fixed camera geometry used in the present study.

Further development should also include an ablation study of the current YOLOv8s-based pipeline to assess the influence of selected training settings, data augmentation techniques, and image-preprocessing stages. In addition, the results should be compared with at least one detection or recognition model outside the YOLO family to evaluate whether the observed performance is specific to YOLOv8s or reflects a broader advantage of deep learning-based character recognition.

Table 12. Practical comparison of the Python-based and LabVIEW/NI Vision Assistant-based implementations.

Aspect	Python-Based YOLOv8s Implementation	LabVIEW/NI Vision Assistant OCR Implementation
Software accessibility	Python is an open-source programming language and can be used without purchasing a dedicated development environment.	LabVIEW is a commercial engineering environment and normally requires an appropriate NI software licence for development.
Execution environment	The application can be developed and executed using a standard Python interpreter and common code editors or IDEs.	The application is developed in the LabVIEW environment, while deployment requires the appropriate LabVIEW Runtime Engine.
Installation footprint	The basic Python environment is relatively lightweight, although additional libraries, especially deep learning frameworks, increase the total installation size.	The LabVIEW/NI Vision Assistant environment is typically larger because it includes integrated graphical programming, vision tools, and runtime components.
Code modification	Python uses text-based source code, which can be edited directly in a code editor and modified line by line.	LabVIEW uses graphical programming, in which the application is built by placing functional blocks and connecting them with wires.
Programming knowledge required	Python requires knowledge of text-based programming, but the development process can be supported by tools that assist with the creation, explanation, debugging, and modification of source code.	LabVIEW requires an understanding of graphical programming principles, including the arrangement of functional blocks and signal connections within the block diagram.
Hardware integration	Python generally requires the selection of appropriate libraries, communication protocols, and device drivers; therefore, hardware compatibility must be handled individually for each instrument or camera.	LabVIEW benefits from measurement-oriented tools and ready-to-use drivers provided by NI or instrument manufacturers, which can simplify integration with laboratory equipment.
Ease of deployment	Python applications can be deployed by installing the required interpreter, libraries, and model files, providing flexibility in the configuration of the target computer.	LabVIEW-based applications require appropriate NI runtime components and, depending on the implementation, additional Vision modules or licences.
Documentation and examples	Python benefits from extensive community documentation, tutorials, scientific publications, and open-source examples.	LabVIEW provides vendor documentation and examples, although detailed information concerning specific OCR workflows is more dependent on proprietary NI tools and documentation.
Main practical advantage	High flexibility, low entry cost, and straightforward integration with image processing and machine learning libraries.	Convenient graphical workflow and strong integration with measurement equipment and existing NI-based test systems.

10. Conclusions

The results demonstrate that neither optical reading pipeline was uniformly superior across all investigated voltage ranges. For the DC voltage measurements, the Python-based pipeline produced a lower standard deviation in five of the six analysed cases and a smaller range in four cases. The clearest difference was observed for the 1 V range, for which a lower dispersion was obtained at both PLC settings. In the AC voltage measurements, the relative performance depended on the selected range: the LabVIEW-based pipeline produced lower dispersion at 100 mV, both pipelines achieved nearly identical results at 1 V, and the Python-based pipeline produced lower dispersion at 10 V.

These results indicate that the selection of an optical reading method should be based on the intended measurement configuration rather than on the internal performance metrics

of the recognition mechanism alone. The common analysis of reconstructed numerical indications provides a practical basis for evaluating complete optical reading implementations and for selecting a solution that is compatible with the existing laboratory infrastructure and further development requirements.

The LabVIEW OCR implementation is particularly appropriate for laboratories that are already operating within the NI hardware and software ecosystem and require direct integration with existing graphical measurement applications. The Python/YOLOv8s implementation offers greater flexibility when an open development environment, further machine learning integration, or adaptation to additional instruments is required.

Because the two implementations were evaluated in separate live acquisition series, the observed differences represent the combined behaviour of image acquisition, recognition, and numerical reconstruction. They cannot therefore be attributed exclusively to the OCR or object detection algorithm. The conclusions concerning the neural-network-based approach apply specifically to the evaluated YOLOv8s implementation and to the controlled acquisition conditions used in this study.

Author Contributions: Methodology, A.S. and P.K.; software, J.W. and J.D.; validation, P.K. and T.B.; writing—original draft preparation, A.S., J.W. and J.D.; writing—review and editing, A.S.; visualization, J.D.; supervision, A.S. and T.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016*; IEEE: New York, NY, USA, 2016; pp. 779–788. [\[CrossRef\]](#)
2. Jocher, G.; Qiu, J.; Liu, M.; Lyu, S.; Akyon, F.C.; Kalfaoglu, M.E. Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models. *arXiv* **2026**, arXiv:2606.03748. [\[CrossRef\]](#)
3. JCGM 100:2008; Evaluation of Measurement Data—Guide to the Expression of Uncertainty in Measurement. Joint Committee for Guides in Metrology: Sèvres, France, 2008. [\[CrossRef\]](#)
4. ISO 5725-1:2023; Accuracy (Trueness and Precision) of Measurement Methods and Results—Part 1: General Principles and Definitions. International Organization for Standardization: Geneva, Switzerland, 2023.
5. VDI/VDE 2634; Part 2: Optical 3D Measuring Systems: Optical Systems Based on Area Scanning. Verein Deutscher Ingenieure and Verband der Elektrotechnik Elektronik Informationstechnik: Düsseldorf, Germany, 2002.
6. ISO 10360-13:2021; Geometrical Product Specifications (GPS)—Acceptance and Reverification Tests for Coordinate Measuring Systems (CMS)—Part 13: Optical 3D CMS. International Organization for Standardization: Geneva, Switzerland, 2021.
7. Kirda, A.W.; Majewski, P.; Bursy, G.; Bartoszek, M.; Yassin, H.; Królczyk, G.; Akbar, N.A.; Caesarendra, W. Integrating YOLOv5, Jetson Nano Microprocessor, and Mitsubishi Robot Manipulator for Real-Time Machine Vision Application in Manufacturing: A Lab Experimental Study. *Adv. Sci. Technol. Res. J.* **2025**, *19*, 248–270. [\[CrossRef\]](#) [\[PubMed\]](#)
8. Szlachta, A.; Odon, A.; Bednarek, P. Vision System as a Measurement Interface. *Przegląd Elektrotechniczny* **2015**, *91*, 292–295. (In Polish) [\[CrossRef\]](#)
9. Corrêa Alegria, F.; Cruz Serra, A. Automatic Calibration of Analog and Digital Measuring Instruments Using Computer Vision. *IEEE Trans. Instrum. Meas.* **2000**, *49*, 94–99. [\[CrossRef\]](#)
10. Wang, L.; Wang, P.; Wu, L.; Xu, L.; Huang, P.; Kang, Z. Computer Vision-Based Automatic Recognition of Pointer Instruments: Data Set Optimization and Reading. *Entropy* **2021**, *23*, 272. [\[CrossRef\]](#) [\[PubMed\]](#)

11. Zheng, C.; Wang, S.; Zhang, Y.; Zhang, P.; Zhao, Y. A Robust and Automatic Recognition System of Analog Instruments in Power Systems Using Computer Vision. *Measurement* **2016**, *92*, 413–420. [CrossRef]
12. Sidor, K.; Szlachta, A. The Impact of the Implementation of Edge Detection Methods on the Accuracy of Automatic Voltage Reading. *Meas. Sci. Rev.* **2017**, *17*, 93–99. [CrossRef]
13. Ferreira, P.; Lobo, P.; Reis, F.; Vilaça, J.L.; Morais, P. Digitization of Medical Device Displays Using Deep Learning Models: A Comparative Study. *Appl. Sci.* **2025**, *15*, 5436. [CrossRef]
14. Ahmadi, N.; Tanoto, H.V.; Munir, R. Automated Digit Recognition and Measurement-Type Classification from Blood Pressure Monitor Images Using Deep Learning. *Algorithms* **2025**, *18*, 377. [CrossRef]
15. Luśtyk, D. Recognition of Digital Display Indications of a Measuring Instrument Using a Hebbian Neural Network. *Sci. Pap. Fac. Electr. Control Eng. GdańSk Univ. Technol.* **2019**, *66*, 37–39. (In Polish) [CrossRef]
16. Grzeczka, G.; Klebba, M. Automated Calibration System for Digital Multimeters Not Equipped with a Communication Interface. *Sensors* **2020**, *20*, 3650. [CrossRef] [PubMed]
17. Zhou, W.; Peng, J.; Han, Y. Deep Learning-Based Intelligent Reading Recognition Method of the Digital Multimeter. In *Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 17–20 October 2021*; IEEE: New York, NY, USA, 2021; pp. 3272–3277. [CrossRef]
18. Olejnik, P.; Awrejcewicz, J. Intelligent Mechatronics in the Measurement, Identification, and Control of Water Level Systems: A Review and Experiment. *Machines* **2022**, *10*, 960. [CrossRef]
19. Carvalho, R.; Melo, J.; Graça, R.; Santos, G.; Vasconcelos, M.J.M. Deep Learning-Powered System for Real-Time Digital Meter Reading on Edge Devices. *Appl. Sci.* **2023**, *13*, 2315. [CrossRef]
20. Zhang, B.; Li, L.; Lyu, Y.; Chen, S.; Xu, L.; Chen, G. A New Instrument Monitoring Method Based on Few-Shot Learning. *Appl. Sci.* **2023**, *13*, 5185. [CrossRef]
21. Ultralytics. Model Comparisons: Quick Decision Guide. Available online: <https://docs.ultralytics.com/compare#quick-decision-guide> (accessed on 13 July 2026).
22. Wang, G.; Chen, Y.; An, P.; Hong, H.; Hu, J.; Huang, T. UAV-YOLOv8: A Small-Object-Detection Model Based on Improved YOLOv8 for UAV Aerial Photography Scenarios. *Sensors* **2023**, *23*, 7190. [CrossRef] [PubMed]
23. Liu, C.M.; Juang, J.C. Systematic Evaluation of YOLOv8 Variants for UAV-Based Object Detection. *Appl. Sci.* **2026**, *16*, 3559. [CrossRef]
24. Jocher, G.; Chaurasia, A.; Qiu, J. Ultralytics YOLOv8. 2023. Available online: <https://docs.ultralytics.com/compare/yolov8-vs-yolo11#performance-metrics> (accessed on 10 July 2026).
25. Basler AG. Basler Ace acA1440-220um USB 3.0 Camera. Available online: <https://www.baslerweb.com/en/shop/aca1440-220um/> (accessed on 6 July 2026).
26. RMA Electronics, Inc. Kowa LMZ68M 1/2-Inch 8–48 mm F1.0 Manual Zoom, Focus and Iris C-Mount Lens with Macro Mode. Available online: <https://www.rmaelectronics.com/kowa-lmz68m/> (accessed on 7 July 2026).
27. Keysight Technologies. Digital Multimeters 34460A, 34461A, 34465A (6½ Digit), and 34470A (7½ Digit): Truevolt DMMs for Your Next Generation of Insights; Data Sheet, Publication No. 5991-1983EN. Available online: <https://www.keysight.com/us/en/home.html> (accessed on 7 July 2026).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.