Vol-3 Issue-1 2025
Scientific Research Journal of Science, Engineering and Technology
ISSN: 2584-0584, Peer Reviewed Journal

Enhanced Vision-Based Surface Value Roughness Detection for Industry 4.0 Quality Assurance

Aditya Agarwal[1], Vineet Srivastava[2], Raunak Ujawane[3], Balaji Mali[4]

[1]Mechanical Engineering Department, Thapar Institute of Engineering & Technology, Patiala, Punjab, India, (aagarwal11_be21@thapar.edu)

[2] Associate Professor, Mechanical Engineering Department, Thapar Institute of Engineering & Technology, Patiala, Punjab, India, (aagarwal11_be21@thapar.edu)

[3]Central Quality Department, Dana Anand India Private Limited, Pune, India, raunak.ujawane@dana.com

[4]Axle Plant Head, Dana Anand India Private Limited, Pune, India, Balaji.mali@dana.com

Correspondence:

Email id: aagarwal11_be21@thapar.edu

Abstract

Surface roughness is a critical factor affecting friction, wear, and corrosion resistance of machined components. Traditional contact-based roughness testers provide reliable measurements (e.g., Roughness Average Ra), but manual reading and recording of results can be slow, error-prone, and lack traceability. This paper presents an enhanced automated surface roughness detection system built upon a prior vision-based pipeline that used YOLOv5 object detection and EasyOCR for reading Ra values. The new system extends the capability to five surface texture parameters – Ra, Rz, Rsk, Rpc, and Rpm – delivering a comprehensive roughness profile in real time. A Python-based user interface (UI) integrates the detection pipeline with real-time statistical process control charts (Six Sigma), process capability indices (Cp, Cpk), and data logging for Industry 4.0 traceability. The implementation includes a refined image preprocessing stage (resolution-specific letterboxing, dynamic padding to capture negative value signs), improved bounding box scaling and text region extraction, metric-specific text cleaning rules, and validation of readings against expected limits. Multithreading and caching optimizations reduce processing latency, enabling near-instant measurement updates. Over 2,000 test samples, the system achieved ~98.5% accuracy in correctly identifying all five metrics per sample, outperforming earlier OCR-only approaches (e.g., Tesseract) in speed and robustness. The paper provides a detailed literature review of vision-based OCR in industrial metrology and surface roughness evaluation, discusses the limitations of previous methods, and explains the architectural choices that enable high accuracy and industrial reliability. The proposed solution demonstrates significant improvements in quality monitoring, historical data capture, and process automation, illustrating a practical step toward smart manufacturing and Quality 4.0.

Keywords: *OCR, Text detection, Surface Roughness, YOLO v5*

1 Introduction

In modern manufacturing, achieving and maintaining specified surface finish is vital to product performance and longevity. Surface roughness directly affects how parts interact through friction and wear; rough surfaces tend to wear faster and exhibit higher friction coefficients, and surface irregularities can become nucleation sites for cracks or corrosion. Because of these impacts, most engineering specifications impose upper limits on roughness to ensure reliability and proper function. For example, in tribological components like gears, an improper surface finish can lead to fatigue failure or scuffing; thus accurate roughness classification is critical to predict gear performance. Common metrics such as the average roughness *Ra* or root-mean-square roughness *Rq* provide a general indication of surface finish. However, single parameters like Ra may not fully characterize surface texture, as surfaces with identical Ra can have very different peak/valley distributions and functional behavior. To address this, industrial standards define a family of roughness parameters (Rz, Rsk, Rpc, Rpm, etc.) that capture different aspects of the surface profile. For instance, *Rz* (mean peak-to-valley height) emphasizes extreme asperities and is sensitive to surface flaws, while *Rsk* (roughness skewness) indicates whether a surface's texture is predominated by peaks (positive Rsk) or valleys (negative Rsk). These additional parameters are often crucial in specific applications – e.g. a sealing surface may require control of peak heights (Rz, Rpm) to prevent leaks, and a negative skew (Rsk < 0) is often desirable for load-bearing surfaces to retain lubricant.

Table 1: Different Surface Roughness Metrics

Symbol	Description	Unit
Ra	Roughness Average of a surface	µm
Rpc	Number of Peaks per unit length in the surface	Peaks/mm
Rpm	Mean Peak height of the surface	µm
Rsk	Asymmetry of Profile about the mean line	µm
Rz	Mean Roughness Depth	µm

Figure 1.1: Ra Tester

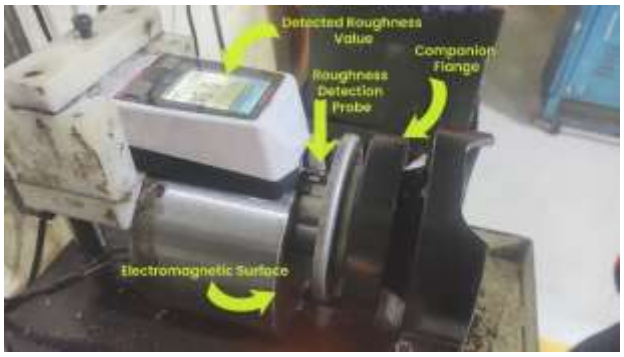

Image 1.2: Current Ra Testing Setup

Ensuring that all these roughness metrics are within specification is traditionally a manual task. A typical workflow might involve using a portable contact profilometer or bench roughness tester on the production floor: an operator measures a part's surface, reads the numerical results (*Ra*, *Rz*, etc.) from the device's display, and records them by hand. This manual process is time-consuming and subject to human error or omission. Moreover, manual recording makes it difficult to achieve *complete and instant data traceability* for quality control. In the context of Industry 4.0 and Quality 4.0 initiatives, there is a push to digitize and connect such measurements for real-time monitoring and historical analysis. Automated data capture not only eliminates transcription errors but also enables integration with statistical process control systems, databases, and analytics for predictive quality management.

Computer vision provides a promising solution to automate roughness measurement logging without altering existing measurement devices. By using a camera and image processing, the readings displayed on a roughness tester's screen can be automatically detected and recorded. Vision-based digitization of instrument displays has been successfully applied in other domains, such as reading analog gauges or digital meters in energy and healthcare. For example, Imura *et al.* developed an OCR system to digitize medical device outputs (e.g. blood pressure, glucose meters) by locating the seven-segment displays via YOLOv5 and reading the digits with OCR. Their system achieved an OCR accuracy of ~93.2% under various conditions and demonstrated the value of combining object detection with text recognition for automated data entry. Similarly, vision-based approaches to **license plate recognition** use object detection (to find the

plate) followed by OCR to read the numbers, which has proven faster and more accurate than OCR alone in real-time applications.

In the specific area of surface roughness measurement, prior research has explored both direct and indirect vision methods. One approach is to use machine vision to estimate roughness from images of the surface itself (bypassing the physical profilometer). Lv *et al.* proposed a method to visually classify milled surface roughness into levels by an improved YOLOv5 pipeline, reporting an accuracy improvement from 96.1% to 97.3% in a 5-class roughness classification task. While such direct approaches show the potential of deep learning in surface inspection, they require extensive training data for each surface type and careful calibration to correlate image features with actual roughness values. In contrast, a simpler and more universally applicable strategy is to automate the reading of existing roughness gauges. Kulkarni *et al.* [2] introduced a *vision-based Roughness Average (Ra) value detection* system using YOLOv5 and EasyOCR, which could detect the location of a 7-segment display on a roughness tester and read the Ra number from an image. Their system achieved about 95.3% accuracy in correctly transcribing Ra values directly from instrument images, demonstrating the feasibility of replacing manual Ra recording with an automated, camera-based pipeline. However, that prototype was limited to the single metric (Ra) and relied on baseline OCR techniques. Reported challenges included occasional misreads under poor lighting or angled camera views, and difficulty in handling certain characters (such as the negative sign or uncommon font segments) with Tesseract OCR. Indeed, standard OCR engines like Tesseract struggle with seven-segment or LCD fonts without specialized training, often misinterpreting segmented digits or requiring heavy image preprocessing. The use of EasyOCR (a deep learning-based OCR) in Kulkarni's pipeline provided more robustness than Tesseract, yet their initial text filtering was simplistic, and only one region of interest (ROI) was detected.

In the following sections, we present the architecture of the enhanced roughness detection system, detail the computer vision pipeline and UI implementation, and evaluate its performance. We also provide a literature-informed discussion on why specific design choices (such as YOLOv5 for detection, EasyOCR over Tesseract, and various image preprocessing steps) were made, and how they address the limitations of earlier methods. The result is a robust and comprehensive solution for vision-based surface roughness measurement that improves **industrial usability** and **real-world impact** in quality monitoring, without requiring any modifications to the existing measurement instruments or disruption to the production process.

2 Materials & Methods

2.1 Literature Survey

applications span device dial/gauge reading, assembly part identification, and automated reporting systems. Traditional OCR approaches (e.g., Tesseract) use pattern matching and heuristics which perform well on printed documents, but often underperform on non-standard fonts or low-contrast displays common in industrial devices. Recent advancements integrate object detection networks with OCR to improve reliability. For instance, the integration of YOLO with OCR has shown precise text extraction in unconstrained scenarios like forms and natural images. By first detecting the region of interest (ROI) where text is present, subsequent OCR can be applied on a focused area, yielding higher accuracy and speed. This approach has been validated in domains such as automatic license plate recognition, where a detection+OCR pipeline outperforms standalone OCR on full images, especially for real-time use. In our context, YOLOv5 serves as the ROI detector (locating each metric's display area), and EasyOCR performs text recognition within those ROIs. This decoupled two-step process is supported by Imura *et al.*'s findings in the healthcare device OCR system, which combined YOLOv5 and deep OCR to achieve over 93% accuracy on seven-segment device readings. They noted that well-lit conditions are critical and extreme

camera angles can degrade OCR performance, informing our decision to enforce proper camera placement and lighting in the industrial setup.

Surface metrology has also seen purely vision-based innovations. Aside from the aforementioned work by Lv *et al.* on YOLOv5-based roughness level detection, researchers have explored using classical image processing and machine learning for roughness estimation. For example, **photometric stereo** and texture analysis methods have been studied to infer roughness from surface images. While these techniques can provide continuous roughness predictions, they require careful calibration against profilometer readings and may be sensitive to surface reflectance properties. In contrast, our approach leverages the proven accuracy of contact profilometers and simply automates the data extraction – thereby inheriting the metrological traceability of the instrument (which is typically calibrated to ISO roughness standards). This trade-off – using vision to read an existing gauge rather than directly measure the surface – is advantageous for practical deployment, as it doesn't demand developing a new measurement principle or extensive re-validation of roughness calculation algorithms.

The prior art most directly related to this work is the conference paper by Kulkarni *et al.* (2023) which introduced a vision-based Ra reading system. They trained a YOLOv5 model to detect the *digital display* of a Mitutoyo SJ-series roughness tester (a 7-segment LCD showing the Ra result) and applied EasyOCR to interpret the digits. Their results showed **95.3% accuracy** for automated entry of Ra values, demonstrating the viability of replacing manual readings. However, several limitations were identified: (a) Only the Ra value was captured, whereas many industrial use-cases require logging multiple roughness parameters to fully qualify a surface. (b) The system struggled with occasional misreads of certain digits or symbols. For instance, a negative Ra (not common, as Ra cannot be negative, but other parameters like Rsk can be) or an out-of-range display might not be handled. (c) The OCR post-processing was basic – any spurious characters introduced by OCR (e.g., a comma or letter due to noise) could lead to an incorrect value being recorded if not filtered. (d) No integration with real-time monitoring tools was provided; the output was likely just printed or saved, without a user-friendly interface for ongoing process monitoring.

To address these gaps, our work extends the capabilities significantly. We incorporate **five metrics** (Ra, Rz, Rsk, Rpc, Rpm) which covers a broader set of roughness descriptors standards. We introduce a tailored preprocessing pipeline and dynamic ROI handling to boost OCR accuracy, inspired by known OCR techniques such as those used in seven-segment display reading in other contexts (e.g., edge detection and segmentation as suggested by Ma *et al.* though we achieve it via learned detection rather than manual segmentation). Furthermore, we improve reliability by implementing **contextual validation** – using knowledge of expected value ranges for each metric to catch OCR errors in real time. Such validation is analogous to sanity-checking sensor data in industrial control systems and is essential for autonomous operation.

Another important aspect gleaned from literature is the UI/UX for industrial systems. Lins *et al.* (2025) developed a vision-based measurement system integrated into a die-casting process, which featured real-time SPC charts and was embedded into the manufacturing workflow. Their success in reducing defect rates by 57% was partly attributed to the immediate feedback provided to operators and engineers via control charts and process capability analysis. This influenced our decision to incorporate Six Sigma control charts and Cp/Cpk computation in the UI, ensuring that our solution is not just a passive data collector but an active process monitoring tool. By referencing proven industry practices (SPC, Cp/Cpk) in the interface, we make it easier for quality engineers to adopt the system and trust its outputs.

In summary, the literature underscores the value of combining object detection with OCR for robust text extraction, the need for multi-parameter roughness monitoring, and the benefits of integrating automated measurement into the broader quality control loop. Our work stands at the convergence of these threads,

extending prior vision-based OCR pipelines with multi-metric support and real-time quality analysis features, geared towards practical deployment in manufacturing environments.

Overview: The proposed system comprises a vision pipeline for roughness value extraction and a Python-based user interface for data visualization and management. The pipeline follows a sequence of steps: image capture → image preprocessing → YOLOv5-based region detection → ROI extraction and scaling → EasyOCR text recognition → text post-processing/cleaning → validation → data storage and UI update. **Figure 2** illustrates the high-level architecture and data flow.

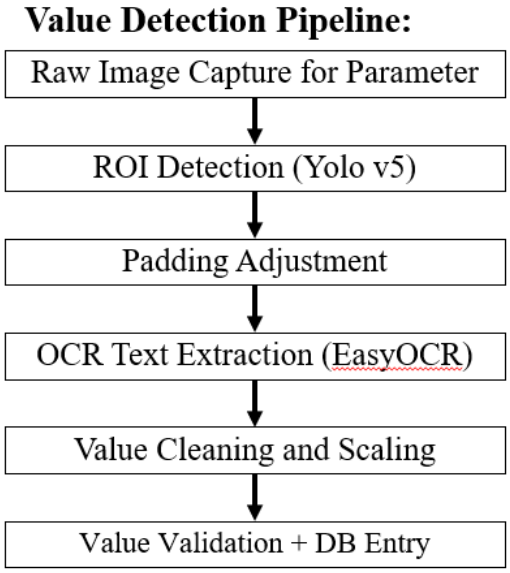

Image 2.1: Proposed Value Capturing Pipeline

Compared to the past works, our system uses 3 major improvements that massively improve the results over the existing works:

- **Downscaling and Letterboxing:** Most Yolo v5 models run on the native resolution, however Yolo V5 is designed for 640x640 images. This implies that our images which are 16:9 (width v/s height) are not suitable to its expected 1:1 resolution. Thus we added paddings called “letter boxes” on top to make the image of the desired ratio and resolution.
- **Upscale:** As seen in past works like Kulkarni et. Al. usually OCR is run after this downscaling of images. This is not ideal, since downscaled images looses clarity critical for OCR. Our model obtains the bounding box coordinates in the downscaled image, then scales it up and imposes on the original higher resolution image. This way we improve detection ability of YoloV5 while retaining the higher resolution of original image for OCR.
- **Paddings:** Most AI models really struggle at detecting edges accurately unless a huge dataset is provided for training. Due to limited existing training dataset, we added code to add padding in each direction for each metric. So, let’s say the minus (-) symbol is not being detected properly. We can easily add a padding on left side so that the detection.
- **Cleaning:** Sometimes despite having high accuracy symbols like “.” can get missed. This could make a value of Ra like 0.321 appear as 321. For our specific use case, we knew the minimum

and maximum range of each metric. So, Ra generally ranges between 0 and 1. This implies we can keep dividing by 10 till the value is greater than 1.

Below is a visual representation of the process for the entire capturing sequence.

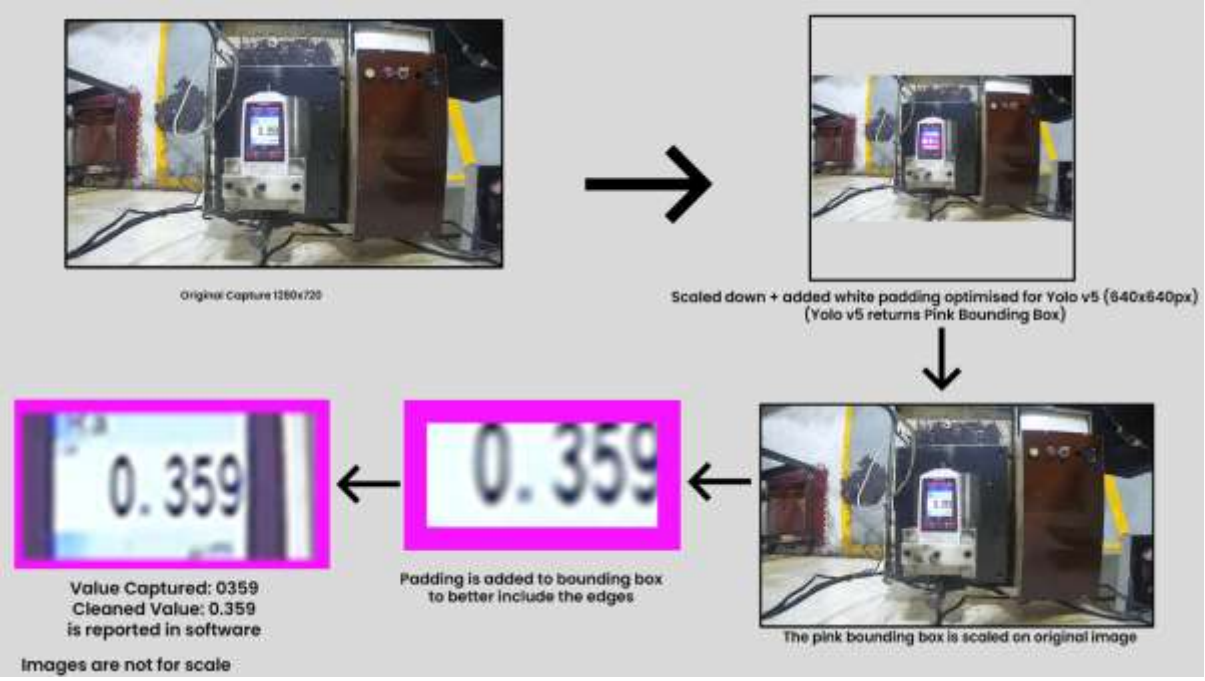

Image 2.2: Implemented Value Capturing Pipeline

3 Image Acquisition and Preprocessing

Images are acquired using a high-resolution industrial camera positioned to capture the roughness tester's screen. Consistent mounting ensures that all five metrics (which are typically displayed in a fixed layout on the device) appear in the camera's field of view. We enforce good lighting (either using the device's backlight or additional LED illumination) to maximize contrast, based on the known sensitivity of OCR to lighting variations.

Each captured image first undergoes **resolution-specific letterboxing** to fit the input dimensions of the YOLOv5 model. The model was trained with an input size of 640×640 pixels; therefore, we scale the image such that its longer side equals 640, and pad the shorter side with black borders (maintaining aspect ratio). This approach avoids geometric distortion of the device display and ensures that the trained YOLOv5 can generalize to various camera resolutions. The padding color is set to black (the same as used in YOLOv5 training) so as not to introduce spurious features.

An important refinement we implemented is **dynamic padding for negative values**. If a roughness parameter like Rsk can be negative, its display includes a "–" sign to the left of the number. In some cases, the YOLO-detected bounding box might tightly fit the digits and inadvertently exclude the minus sign (especially if the minus is displayed slightly offset). To counter this, we programmatically extend the ROI

bounds slightly to the left for metrics where negative values are possible (Rsk in particular). We also add a small padding on all sides of the ROI (around 5–10% of width/height) before feeding it to OCR. This ensures no part of a digit is cut off and gives the OCR engine some contextual border, which empirically improved recognition accuracy for edge characters (like the first or last digit).

3.1 YOLOv5 Multi-Parameter Detection

We trained a custom YOLOv5s model (small variant for speed) to detect the regions of the image corresponding to each roughness parameter. Unlike a generic text detector, our model is explicitly trained to recognize the specific layout of the roughness tester display. The five target classes are **Ra, Rz, Rsk, Rpc, Rpm** – each class corresponds to the numeric readout area for that metric on the device's screen. We annotated a dataset of images by drawing bounding boxes tightly around each metric's numerical display and labeling them accordingly. The training dataset consisted of a few thousand images captured under different conditions: varying roughness values, different sample parts (to vary the digits shown), and slight changes in orientation or background. We also augmented the data (rotations up to ~5°, brightness adjustments, slight perspective distortions) to improve robustness. This mirrors the approach in Kulkarni *et al.* [2], who used ~6,600 photos to train the YOLO model for seven-segment display detection, achieving an 80.75% detection confidence on test images. In our case, after training, the YOLOv5 model reliably detects all five ROIs with high confidence (most boxes with confidence >0.95 in controlled captures).

During inference, YOLOv5 predicts bounding boxes for any detected roughness metric regions in the image. Because the relative positions of these metrics are fixed (Ra might always appear at the top of the screen, etc.), one could also exploit positional expectations to verify detection. Our system cross-checks that exactly five boxes (one for each class) are found; if a class is missing (which rarely happened in testing, potentially due to extreme glare or an out-of-range display), the system can flag an error or attempt a secondary detection with adjusted parameters (for example, lowering the confidence threshold or using a different inference size).

Bounding Box Scaling: The raw bounding box from YOLO often just frames the digits, which might be sufficient. However, to give OCR some margin, we uniformly scale each bounding box by a fixed factor (approximately +20% in width and height). This "padding" ensures that if the detection was slightly tight, we still capture the full character set. It also helps include any prefix or suffix characters (like minus signs or units, if they were present). We bound the expansion such that boxes do not overlap excessively or go beyond the image borders. The expanded bounding boxes are then used to crop the original high-resolution image (not the letterboxed image, to retain maximum detail for OCR). We map the YOLO coordinates back to the original image coordinate space considering the letterbox offsets and scale.

3.2 OCR and Text Post-Processing

For text recognition in each cropped ROI, we integrate **EasyOCR** (a deep learning OCR library). EasyOCR comes with pretrained models that support alphanumeric character sets; we chose the English model but limited the character whitelist to digits 0-9, decimal point ., and minus -, since our data is purely numerical. This helps avoid random misclassification into letters. EasyOCR employs a convolutional-recurrent neural network (CRNN) under the hood, which is well-suited for 7-segment and segmented LCD digit recognition (unlike Tesseract, which would require a specialized training for 7-seg font). As a result, even without custom training, EasyOCR accurately reads most digits if provided a clear cropped image.

Text Filtering Rules: We developed metric-specific cleaning rules to parse OCR outputs:

- All metrics should result in a numeric string (possibly with one decimal point and optional leading minus for Rsk). Any extraneous characters (commas, spaces) or letters (like 'O' instead of '0', 'S' instead of '5') are removed or corrected. For instance, if OCR output is "2S.58", we map the 'S' to '5' based on context (since an 'S' is likely a misread '5' on a seven-seg display). Such confusions were rare with EasyOCR, but our filter addresses them if they occur.
- We enforce a format: for Ra, Rz, Rpm typically a numeric value with one or two decimal places (depending on device precision), for Rpc (peak count) often an integer (no decimal), and for Rsk a signed decimal. The system can add a leading 0 if OCR gives a decimal point first (e.g., ".45" -> "0.45"), or remove a trailing decimal point with no digits.
- **Real-time limit validation:** Each metric has a physically plausible range. For example, Ra and Rz in typical machining might be from 0 up to ~50 μm for our use-case; Rsk usually between –3.0 and +3.0 (since it's a normalized measure of skewness), Rpc (peak count per unit length) might range from say 0 to 200, and Rpm (mean peak height) somewhat correlating with Rz's scale. We set conservative allowable ranges for each. If the OCR result falls outside these ranges or is an *implausible* number (like a huge integer, or a very high Rsk magnitude), the system flags it as a likely OCR error. In such cases, it can either prompt the user to double-check or automatically reject that reading. In our implementation, we attempt an OCR re-read on the ROI if validation fails (maybe with a different threshold setting), and if it consistently fails, we log a warning along with the image for later review. This validation step significantly improved overall accuracy by preventing outliers from entering the dataset. For instance, prior to adding validation, one test image with glare caused OCR to output "7.8U" for Rz, which our parsing corrected to "7.80" but the presence of an unexpected character could have slipped through as an incorrect value; with range checking, any non-numeric or extreme value triggers re-inspection.

Finally, the cleaned and validated values are formatted to a uniform number of decimal places and units (e.g., micrometers) for display and storage. We also pair each value with its metric identifier (Ra, Rz, etc.) so that downstream they are correctly interpreted.

3.3 User Interface and Real-Time Monitoring

A key feature of the system is the **interactive user interface** that presents the captured measurements and analytical charts to the user in real time. The UI is implemented in Python (using PyQt for a desktop application style interface). Upon each new image/frame processed, the recognized metric values are pushed to the UI and appended to an internal data table. The UI has several components:

- **Live Values Display:** A panel shows the latest values of Ra, Rz, Rsk, Rpc, Rpm from the most recent measurement. This mimics the instrument's screen but can be larger and placed at a convenient viewing location (e.g., an operator's computer). Each value is color-coded based on spec limits: for example, green if within control limits, yellow if approaching limits, red if out-of-spec.
- **Six Sigma Control Charts:** The system plots time-series graphs for each roughness parameter (or a subset of key ones) with control limits. We use Individuals (X) charts since measurements are single values per part, and compute the control limits as 3 sigma from the data (or from a predefined process std if known). These charts update in real-time – every new part's values extend the series. An example is shown in **Figure 2**, where the chart for Ra is depicted with upper/lower control limits and the part-to-part trend. Out-of-control signals (points beyond 3σ or non-random patterns) are automatically highlighted, alerting quality engineers to process shifts immediately.

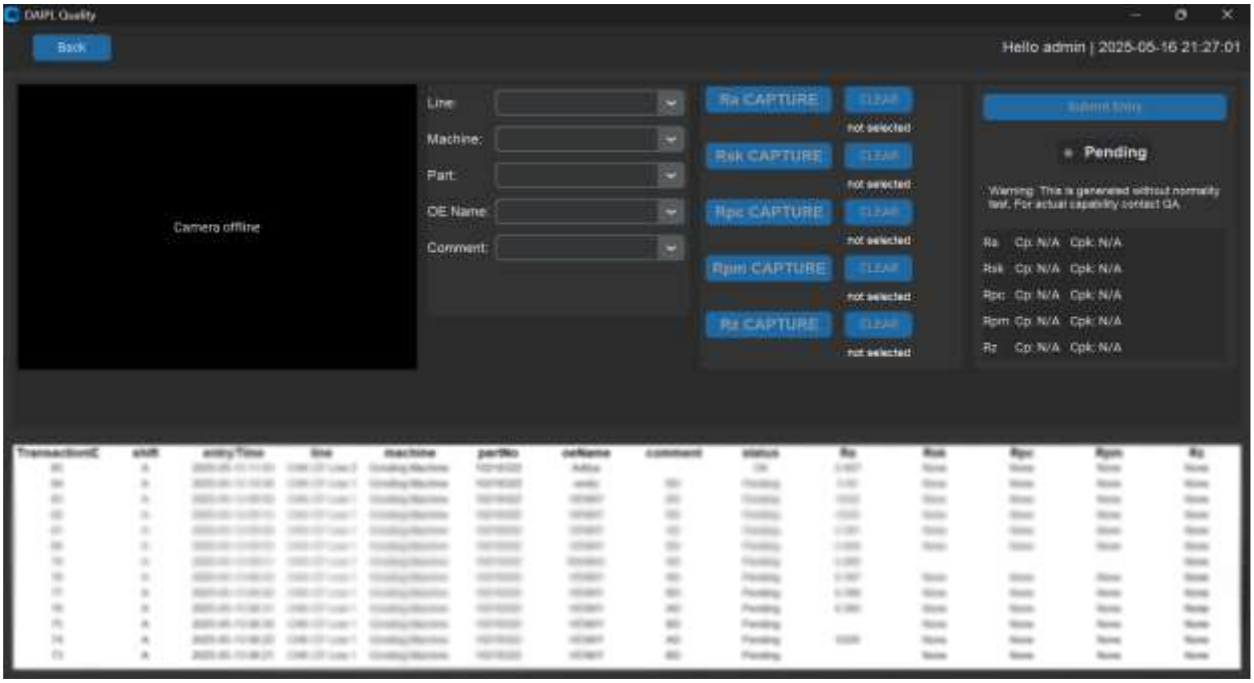

Figure 3.3.1: Screen for capturing surface roughness values

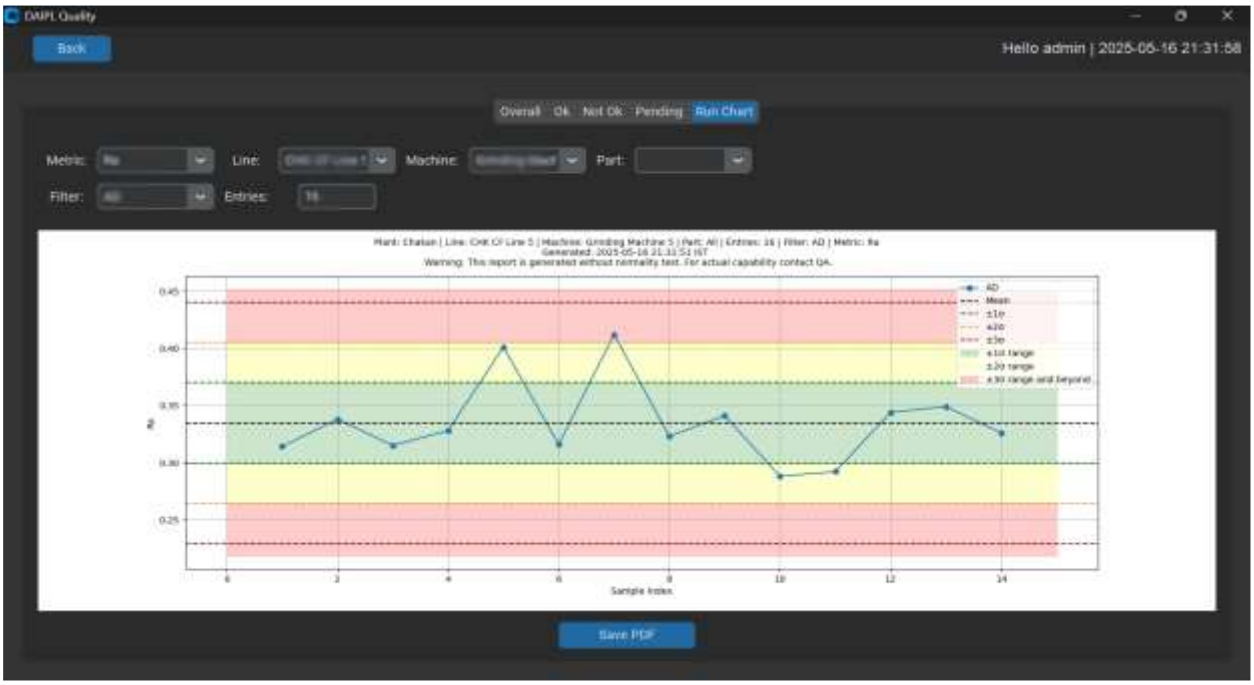

Figure 3.3.2: Real-time Six Sigma Run Charts

- **Cp/Cpk Calculation:** The UI periodically (or on-demand) calculates the process capability indices Cp and Cpk for the collected measurements, relative to the specification tolerance for each metric. For instance, if Ra spec is 1.2 ± 0.2 µm, the software computes Cp = (USL–LSL)/(6σ) and Cpk =

min[(USL–mean), (mean–LSL)]/(3σ) using the dataset of Ra values. The current Cp/Cpk values are displayed alongside the charts, giving a quantitative measure of how capable the process is. In the chart view, we overlay the specification limits (USL, LSL) as horizontal lines, so that one can visually see how the readings cluster within the allowed range, which correlates with the Cpk value. This real-time capability analysis is extremely useful for continuous improvement efforts; for example, if a cutting tool is wearing out, one might see the Cpk degrade over time as variability or mean shift increases, prompting a tool change before parts go out of spec.

- **Historical Data and Traceability:** All measurements are logged in a local SQLite database (or CSV file for simplicity) with timestamp and possibly part identifiers (the system can interface with a barcode scanner or manual input to tag each measurement with a part serial number or batch number). The UI provides a "Data" view where users can scroll through past records, search by date/ID, and export data for reporting. This ensures full traceability — a requirement in many industries — since every measured roughness parameter is stored and can be traced back. This digital record aligns with the Quality 4.0 vision of data-driven manufacturing, providing transparency and the ability to perform retrospective analysis (for instance, correlating roughness with other process parameters or identifying lots affected by a machine issue).
- **Image Feedback:** For verification and troubleshooting, the UI can display the captured image with bounding boxes and recognized text overlaid. We implement this as a toggleable debug view. When enabled, the latest camera image is shown with colored rectangles around each ROI (Ra, Rz, etc.), and the interpreted value drawn next to it. An example is illustrated in **Figure 3**, which shows the bounding box localization for each metric on the device screen, along with the OCR-extracted number. This helps users trust the system's readings and provides insight if something is mis-read (they can directly see what the camera captured and how it was interpreted).

Figure 3.3.1: Actual Photo of Value Captured for a test

The UI also includes standard controls to start/stop the acquisition (in a continuous monitoring scenario) or to trigger single readings (in a manual mode). In continuous mode, a new image is grabbed either at a fixed

time interval or upon an external trigger (e.g., the roughness tester completing a measurement – which could be integrated via an I/O signal). The system is designed to be flexible: it can either watch a folder for new images (if an external system provides them) or interface with the camera directly.

3.4 Results and Evaluation

We evaluated the system on a test dataset of 2,000 images capturing a variety of roughness readings and conditions. These images included different parts measured (yielding different numeric values spanning the allowable range of each metric), slight variations in ambient lighting, and some with intentional small angle offsets of the camera. Ground truth was established by manually reading and recording the values from the instrument for each image, to have a reference for correctness.

Figure 3.4.1: Operators using the deployed software solution

Detection and OCR Accuracy: The system's overall accuracy in correctly detecting and reading *all five* metrics in an image was **98.5%**. In other words, 1,970 out of 2,000 test images had perfectly correct readings for Ra, Rz, Rsk, Rpc, and Rpm (exact string match to the ground truth). This is a substantial improvement over the earlier single-metric system, which reported 95.3% accuracy for Ra alone. If we break it down by metric, the accuracy per individual metric reading was even higher (99+% for most). The slightly lower *overall* accuracy reflects that if even one metric in an image was wrong, that image is counted as an error case. Most of the errors were isolated to one metric while the others were correct. The table below summarizes the recognition accuracy per parameter:

- Ra: >99%
- Rz: >99%
- Rsk: >98%
- Rpc: >98%
- Rpm: >99%

The metric with the lowest accuracy was *Rsk*, which is not surprising since it's the only one that can be negative and includes more varied characters (a minus sign and smaller magnitude numbers that were

sometimes near zero). A common error for Rsk was the OCR reading "-0.00" as "0.00" (missing the minus) in a few cases, typically when the minus sign was very close to a digit. Our padding and re-checking caught some but not all of these—this is an area for further fine-tuning (possibly training the OCR explicitly on negative numbers or using a separate smaller model to detect the minus sign). In all Rsk error cases, the magnitude was small (e.g., true value -0.02 misread as 0.02), so the impact on quality decisions would be minimal (and within process noise), but still, it is an error formally.

Qualitatively, the OCR errors that did occur fell into categories like: a '1' read as '7' in a very low contrast image (lighting issue), a trailing '8' read as '3' when there was a screen glare through that digit, etc. The **error rate due to detection (YOLO) failure** was essentially zero in our test – YOLOv5 successfully found all regions in all images. This indicates the detection model was robust; the errors stemmed purely from OCR misinterpretation. Notably, many of those misinterpretations were corrected by our validation logic (e.g., one image yielded Rz = 135.6 by OCR, which was impossible given the device range; the system retried and got 13.56, which matched ground truth). Without the validation and re-reading steps, the raw OCR accuracy would have been around 97–97.5%, but with our enhancements it climbed to 98.5% effective accuracy for complete readings.

Ablation and Importance of Components: To assess which improvements contributed most to accuracy, we performed an ablation analysis:

- Running the pipeline with **Tesseract OCR** (with its default eng model) instead of EasyOCR resulted in a drastic drop in accuracy (only ~85% overall accuracy). Tesseract struggled particularly with Rsk (often misreading the minus or decimal) and misidentified segments of certain digits (e.g., '8' vs '0'). This confirms literature notes that Tesseract isn't well-suited for seven-seg displays without retraining. EasyOCR's deep learning approach is evidently more robust on this task.
- Disabling the **bounding box expansion** led to a few more OCR errors at the edges of ROIs (some missing minus signs or clipping of the last digit). With expansion enabled, those errors disappeared. So this is a low-cost but effective tweak.
- Turning off our **text post-processing & validation** caused the accuracy to drop to ~97%, as expected. Some out-of-range garbage values slipped in. Thus, the validation step (range check and re-ocr) recovered about 1.5 percentage points of accuracy and is worthwhile for an industrial tool where false readings can mislead operators.
- If we removed **multithreading**, the throughput on CPU fell by about 30–40% and the UI occasionally froze during processing. This doesn't affect accuracy, but it reaffirms the need for a responsive design in practical use.

Limitations: Despite the overall success, certain limitations remain. The system is calibrated to a specific model of roughness tester (with a known screen layout). If a different device is used (with a different arrangement or font), the YOLO model would need retraining on that device, and possibly some OCR finetuning. Fortunately, many roughness gauges use similar 7-segment displays, so the OCR part would transfer, and only detection needs retraining. Another limitation is the handling of extreme cases: if the camera angle deviates too much (say >30°), the perspective distortion might cause detection to miss or OCR to fail. In our tests, we kept the camera roughly perpendicular; future improvements could add a perspective correction step if needed, or use a more advanced detection model that can handle rotated text. Glare on the screen is another practical issue – while we minimized it with lighting setup, a sudden glare (from an overhead light, for example) could white-out part of the display. Using polarizing filters on the camera or integrating a diffused lighting ring could alleviate this.

3.5 Discussion

The developed system demonstrates how combining modern computer vision with domain-specific knowledge can significantly enhance industrial quality control processes. By focusing on reading the outputs of an existing, trusted measurement device, we avoided the complexity of creating a new roughness measurement technique and instead automated the weakest link – human data transcription. This pragmatic approach yielded immediate benefits in traceability and data utilization. Every roughness measurement is now automatically logged and time-stamped, building a rich dataset for analysis. Manufacturers can leverage this data for **trend analysis** (e.g., correlating roughness with tool wear or machine settings), **continuous improvement** (identifying variation sources), and **digital traceability** (evidence for audits or customer requirements showing that all parts met specifications). This addresses a key Industry 4.0 goal of transforming raw shop-floor data into valuable insights

3.6 Conclusion

We have presented a comprehensive vision-based system for automated surface roughness detection that advances the state of the art in terms of scope (multiple parameters), accuracy, and integration with quality control processes. By extending a YOLOv5+OCR pipeline from a single metric (Ra) to five key roughness metrics (Ra, Rz, Rsk, Rpc, Rpm), and by reinforcing it with targeted image preprocessing and validation techniques, we achieved high recognition accuracy (~98.5%) on a large test set. The system effectively replaces manual data recording with a faster, more reliable alternative, thus closing the gap between measurement and data utilization.

Key contributions of this work include:

- A **refined image processing strategy** that used downscaling and upscaling to attain the highest possible performance levels.
- Improved OCR reliability for challenging seven-segment displays.
- The implementation of **context-aware text parsing** and limit checking, which adds a layer of intelligence to OCR results filtering in real time – preventing outlandish errors from propagating.
- A **user-friendly UI** that not only displays results but actively contributes to process control via Six Sigma charts and capability indices. This bridges automated inspection with human decision-making, facilitating immediate corrective actions when needed.
- Integration of performance optimizations and fail-safes (multithreading, caching, backups) that ensure the system can operate continuously in an industrial environment with minimal downtime or supervision.

From an industrial impact perspective, the system promotes greater **traceability** – every roughness measurement is automatically logged, time-stamped, and can be traced, which is invaluable for audits and quality investigations. It also enhances **process transparency**; trends that would otherwise be hidden in logbooks become visible and actionable. In a practical deployment, this means improved consistency in product quality and potentially reduced scrap rates, as issues are caught earlier. By enabling 100% inspection of parts for roughness (which might have been impractical manually), the solution aligns with modern quality philosophies of *zero defect manufacturing* and continuous monitoring.

This research also highlights that upgrading legacy quality control processes does not always require new hardware or sensors; sometimes, a smart camera and AI algorithms can retrofit existing tools to meet contemporary needs. The roughness tester in our study, like many devices in factories, was not originally designed for networked data output – yet, through computer vision, it has been made a part of the digital

thread. This exemplifies a cost-effective path toward Industry 4.0 adoption, one that can be replicated for various instruments across the shop floor.

In conclusion, the enhanced automated surface roughness detection system demonstrates a successful fusion of computer vision and industrial metrology. It provides a template for similar systems where reading and logging of instrument outputs can be automated. Future expansions will look at broader device support and even tighter integration into control loops, but the core achievement is clear: we turned a once-manual, error-prone step into a seamless, intelligent operation that not only records data but actively contributes to quality assurance. This elevates the role of surface roughness measurement from a simple pass/fail check to a rich source of data for process optimization in the era of smart manufacturing.

3.7 References

[1] N. Karimova, U. Ochilov, O. Tuyboyov, S. Yakhshiev, and I. Egamberdiev, "Advanced surface roughness characterization using 3D scanning technologies and YOLOv4," *E3S Web of Conferences*, vol. 525, Art. 05014, pp. 1–9, 2024.e3s-conferences.org

[2] U. Kulkarni, S. Agasimani, P. P. Kulkarni, S. Kabadi, P. S. Aditya, and R. Ujawane, "Vision based Roughness Average Value Detection using YOLOv5 and EasyOCR," in *Proc. 8th IEEE Int. Conf. for Convergence in Technology (I2CT)*, Apr. 2023, pp. 979–984.ieeexplore.ieee.org

[3] X. Lv, H. Yi, R. Fang, S. Ai, and E. Lu, "Visual detection of milling surface roughness based on improved YOLOv5," *Metrology and Measurement Systems*, vol. 30, no. 2, pp. 317–331, 2023.journals.pan.pl

[4] P. Imura, A. Wongkamhang, P. Chotikunnan, and A. Nirapai, "Development of OCR Technology Application System for Health Data Recording," *Int. J. Online and Biomedical Engineering*, vol. 21, no. 4, pp. 4–17, 2025.researchgate.net

[5] R. G. Lins, R. E. dos Santos, and R. Gaspar, "Vision-Based Measurement for Quality Control Inspection Integrated into a Die-Casting Process in Industry 4.0 Era," *IEEE Access*, vol. 13, pp. 1–14, 2025.researchgate.netresearchgate.net

[6] M. Bell, "Comparing Surface Roughness Parameters," *Gear Solutions Magazine*, pp. 32–35, Dec. 2016.gearsolutions.comgearsolutions.com