ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 14, Issue 4, pp: (26-46), Month: October – December 2026, Available at: www.researchpublish.com

AI CNN Verification of Printed Brand Logos for Cross-Platform Deployment

Ray Wai Man Kong[1]

[1]Department of Systems Engineering, City University of Hong Kong, Hong Kong, China

DOI: https://doi.org/10.5281/zenodo.24531269

Published Date: 25-November-2026

Abstract: **Finished garments have two important printed marks — the brand logo and the wash label — which determine if they can move on from the assembly cell. Traditionally, these marks are checked by eye, a process that is tiring, inconsistent, and difficult to audit. Sometimes, even when a presence check shows 97% similarity and a color correlation of 0.99, a label can still fail due to ink issues. Standard inspection systems struggle here: fabric-defect detectors look at the entire surface instead of just the mark, learned detectors require lots of annotated data and a GPU, and one-class anomaly models need to be retrained each time a label changes. Feature-similarity programs can only confirm if a mark is present, not whether it's good quality.**

The main innovation is a CNN verification including image recognition from a template image, requiring no dataset, model file, or data scientist. During our experimental test, on-device training takes 2.7–3.0 seconds; each region takes 2.05 milliseconds to process. Across 70 test patches, scores from 0 to 1 separated good prints (minimum 0.766) from the worst bad print (maximum 0.616), with no false accepts or rejects at thresholds of 0.65–0.75. Compared with international standards, it offers greater accuracy, speed, and efficiency through one program, five bilingual Android-tablet commands, and one-tap product changes.

Keywords: **logo verification, wash label inspection, ORB, RANSAC, CNN verification, metric learning, on-device training, cross-platform deployment.**

I. INTRODUCTION

A finished garment has a few printed or woven marks that determine if it can be shipped: the brand logo, usually found on the chest or sleeve, and one or more wash and composition labels inside the neck or side seam. These marks are small and sit on flexible material, making them low in contrast compared to the surrounding fabric. In manual production lines, people check these marks by eye at the end of sewing—a process that's tiring, varies between operators, and is hard to audit afterwards.

Automating this check isn't the same as finding fabric defects. Fabric inspection looks for things like holes, slabs, oil marks, or broken yarns across the whole surface, and this field is well-studied: Hanbay et al. [1] reviewed it, Kumar [2] covers the computer vision methods, and Ngan et al. [3] discussed automated detection. In contrast, a mark-verification station faces a more focused—and sometimes tougher—challenge: it has to confirm that a specific, known graphic is present, complete, correctly positioned, and printed in the right colour, even when the garment might be rotated, shifted, or partly covered. Plus, it has to do all this within the line's cycle time.

Learned detectors—like those described by He et al. [4] for residual networks, LeCun et al. [5] for deep learning, and Roth et al. [6] for anomaly detection—can be very accurate, but only when you have thousands of labeled examples. That's rarely the case on a real production line. New logos come out with each season, license plates change, and wash labels get updated, meaning the system needs to be re-taught quickly by a line engineer, not retrained overnight by a data scientist. That's why one-class inspection methods—like those by Bergmann et al. [7], Defard et al. [8], and Roth et al.—are appealing: they learn what's "normal" from just a few good samples. The catch is that a print can look normal but still be wrong, and the model won't spot it if it's only seen good prints before.

Local features are at the other end of the spectrum. SIFT (scale invariant feature transform) [9] and SURF (speeded up robust features) [10] are both very good at matching but the computational cost is such that they're simply not feasible on a low power panel computer. ORB (Oriented FAST and Rotated BRIEF) [11] was introduced by Rublee et al. It's the corner detector paired with the binary descriptor and runs perfectly within a frame. A RANSAC homography (Fischler & Bolles [12] for the consensus search, Hartley & Zisserman [13] for the geometry) fitted to those correspondences gives you not just a match but also the pose of the mark (scale, in-plane rotation and translation) which is precisely what a station needs in order to place an inspection rectangle and check orientation.

Geometry cannot supply appearance. The field readings of the station described here show the limitation concretely: on a correctly printed label the ratio of inliers to putative matches is routinely 99 per cent and the colour correlation 1.00, and in a recorded frame of the deployed tablet a mark that was nevertheless rejected reported 99 per cent similarity and 95 per cent colour correlation. Those are ratios and correlations over a mask, not print-quality measurements. A worn edge removes a few keypoints from a population of hundreds, and a missing block of a graphic still leaves the surrounding structure intact; both leave the statistics high. The gate opens, the bad garment passes, and the error is discovered downstream — if at all.

This paper addresses that gap by introducing a smart, affordable second opinion that runs right on the same device. Here's what's new: (i) a hybrid pipeline where ORB and RANSAC locate and fix the position of the mark, and then a compact convolutional network checks the quality of that patch; (ii) a two-part score that mixes an overall similarity with a cell-by-cell comparison, putting extra weight on the parts that look worst, so local problems don't get hidden; (iii) a setup process that lets the network learn from a single captured template, taking about three seconds—all on the device; (iv) a way to turn the network's verdict into an easy-to-understand score showing how much of the mark would have to be damaged for it to fail; and (v) a single program that runs as-is on Windows, Android, and Orange Pi, along with fixes that made sure the Android version works reliably. The paper is just as careful about describing where the method's limits are as it is about what it can do.

Here's how the rest of the paper is laid out: Section II looks at related work, Section III explains the station setup and software, Section IV goes through the method, Section V covers how it's implemented and works across different platforms, Section VI shares results, how damage is measured, and what the system offers in practice, and Section VII wraps up with the conclusion.

II. RELATED WORK

A. Local Features and Binary Descriptors

SIFT, created by Lowe, finds important points in an image and describes them with a detailed 128-value histogram. It's still the standard that others are measured against, as Mikolajczyk and Schmid [14] confirm in their studies. SURF, from Bay et al., speeds things up by using box filters instead of more complex math. ORB, introduced by Rublee et al., takes things further: it combines FAST (for corner detection) with a simple way to measure orientation and uses a binary code that's much quicker to match than floating-point alternatives. For a station inspecting small marks at a set distance, losing some scale flexibility isn't a big deal—the speed boost is what really matters.

Machine-learned detectors and matchers have now surpassed the older, hand-crafted features in tough situations. SuperPoint, from DeTone et al. [15], learns to spot and describe key points together, while SuperGlue, by Sarlin et al. [16], uses a graph neural network instead of the usual nearest-neighbour rule. Radenović et al. [17] tune their network so it can handle big changes in viewpoint. But all these methods depend on large, pretrained models, which doesn't fit the quick-enrollment approach used here. They're great when you have a fixed set of marks for years—but not when your marks change every season.

B. Geometric Verification

When you try to match points between images, there are always some outliers, and standard least-squares methods can't handle them well. That's where RANSAC comes in—introduced by Fischler and Bolles, it repeatedly tries small random sets, fits a model to each, and picks the one most points agree with. For images, it only needs four matches at a time to figure out the transformation. This gives you a 3 × 3 matrix (a homography) that tells you how to rotate and scale the image patch, as Hartley and Zisserman explain. A more traditional approach is template matching (see Brunelli [18]), which is still common in label stations, but it can't handle rotation or perspective changes and quickly falls apart if the print varies much.

C. Learned Verification and One-Class Inspection

Industrial anomaly detection has pretty much settled on the one-class approach. MVTec AD (Bergmann et al.) was the benchmark for this; PaDiM (Defard et al.) is a patch distribution model which uses a Gaussian estimate of the patch distribution; PatchCore (Roth et al.) maintains a memory bank of nominal patches and scores a test patch by its distance to that bank. MVTec AD is the public benchmark for this class of methods, providing defect free training images and defective test images of industrial objects and textures so that methods can be fairly compared without recourse to a private dataset. PaDiM is patch distribution modelling in that it uses a multivariate Gaussian to model the feature vector at each patch position, estimated from the good samples, and flags a test patch whose vector is unlikely under that Gaussian. These methods do a good job at localising defects, but they are based on the assumption that the nominal class is homogeneous and that a defect is a deviation from that. A perfectly printed mark which is just wrong is not a deviation of that kind; it's a different, equally nominal object. That's why a verification formulation is actually the more appropriate one in this case rather than an anomaly formulation.

Deep convolutional networks (LeCun et al. on the general architecture, Simonyan and Zisserman [19] on depth and the residual variants in He et al.) form the basis of such methods, with the lightweight designs of MobileNetV3 (Howard et al. [20]) and MobileNetV2 (Sandler et al. [21]) being the ones that make them viable for embedded hardware. MobileNetV2 replaces a dense convolution with a depthwise separable one and stacks inverted residual blocks with linear bottlenecks, reducing the arithmetic cost of a layer by an order of magnitude at a very slight cost in accuracy. MobileNetV3 takes the same approach and improves on it with squeeze-and-excitation blocks and a hardware-aware architecture search so that a practical level of accuracy can be achieved on a tablet or phone processor. Both are the de facto standard for the embedded case and both still require pretrained weights and an offline training pipeline. The accuracy of these models is not in doubt. The issue is their cost, their memory footprint and their need for pretrained weights. A station that has to be taught by an operator, offline, on a device without a GPU simply can't rely on them.

D. Metric Learning for One-Shot Verification

The alternative is to learn a metric rather than a classifier. The Siamese network approach of Koch et al. [22] and the triplet objective of Schroff et al. [23] both map samples into an embedding space where distance is the measure of identity, so a single enrolment sample defines a class. The contrastive loss of Hadsell et al. [24] is the same in principle but operates on pairs with a margin. The network described in Section IV-D is from this family with two practical differences: it is trained at run time rather than offline and is small enough to be evaluated in NumPy on a CPU.

E. Embedded and Cross-Platform Deployment

OpenCV (as described by Bradski [25]) and NumPy (as described by Harris et al. [26]) provide the numerical underpinning on all three target platforms. OpenCV, the Open Source Computer Vision library, is used in this instance as a Python library (the program imports it as the cv2 module and calls its functions for image capture and decoding, colour conversion, warping and feature detection). NumPy provides the array arithmetic on which the verifier itself is written. Both are free Python libraries which run unmodified on Windows, Android and the Orange Pi and are available to a Python program without the need for a compiler, which is why a single source produces all three builds. The literature on deployment is considerably thinner than the literature on accuracy: a system that works perfectly on a desktop may well fail on a tablet due to the camera texture being mirrored, the touch coordinates being in window space rather than screen space, or the OS terminating the process when it runs out of memory. Section V details the type of defects of this nature that were discovered and rectified, as they are the ones that determine whether a method even gets to the line.

F. Position of the Present Work

The station in question is right at the low end of the trade-off. It has a rigid flat mark, a fixed camera and one enrolment sample per mark. In return it gives you a decision in a few milliseconds on hardware that costs less than the camera. The convolutional stage is not in any way a substitute for the geometric stage. It's a second opinion which is only invoked after the geometric stage has already given its verdict. That's what makes it both small and fast.

The novelty claimed is not in any new architecture but in the application of the architecture. The families of systems that are efficient in other contexts are all inefficient for this application: whole surface defect detectors answer a different question; learned detectors and one class anomaly detectors (Bergmann et al., Defard et al. and Roth et al.) require annotated data, a GPU and a retrain every time the label changes; and feature similarity/template based systems just report the presence of a defect. The verification stage which is supervised by the template image alone removes all three of those dependencies. The comparison in this paper is therefore with published international practice rather than a single in-house system. The outcome is therefore transferable in terms of accuracy, milliseconds and one tap product.

III. SYSTEM DESCRIPTION

A. Physical Configuration

The station consists of a Universal Serial Bus (USB) camera on a flexible arm, a diffuse light source, a fixed working distance and a matte white plinth on which the garment or the label is placed (Fig. 1). The camera is mounted normal to the plinth at a distance that makes the mark occupy between one fifth and one half of the frame width. Ambient light is reduced, and the camera's automatic white balance is locked before the template is captured, because the colour statistics of Section IV-C are only meaningful under a stable illuminant.

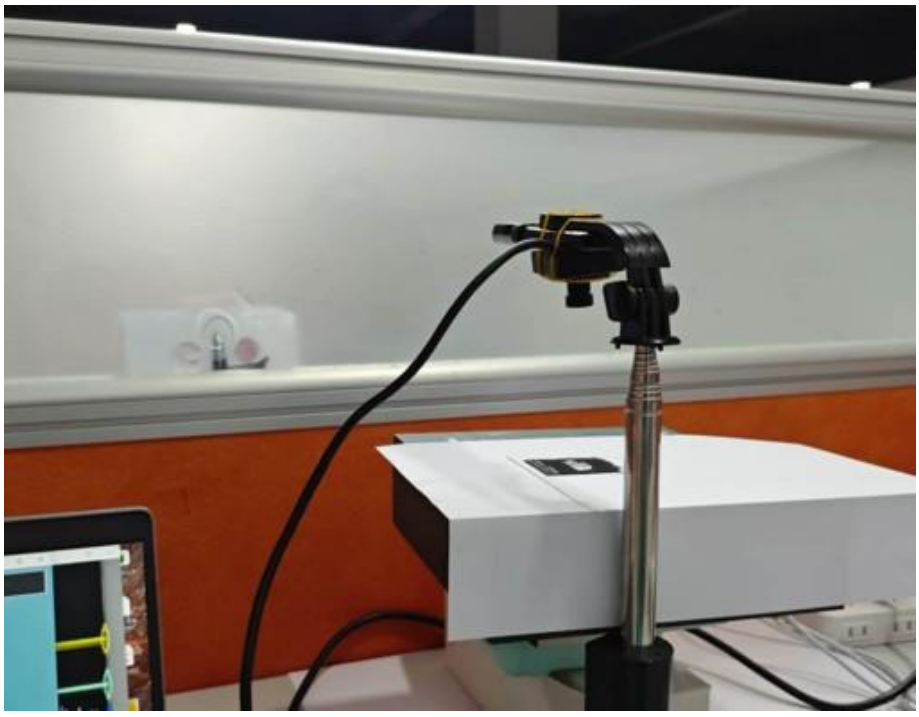

Fig. 1. The inspection station. A USB camera on a flexible arm is looking down on a matte plinth; the display on the left is running the same program as the desktop build.

The computing unit is either a small desktop running Windows, a tablet running Android 7 or later, or a single-board computer running Linux. The same program runs on all three; differences are confined to the camera backend, the white-balance path, and how results leave the process.

B. Software Architecture

It's a single process with one acquisition loop. Each loop iteration reads a frame, does the software white balance (as there's no hardware control available on the platform) and passes the frame to a worker thread. The worker then runs the chain of Fig 2 for each enabled region, creates an annotated version of the frame and publishes a result record which the interface thread reads at its own rate. The separation of acquisition from processing means that the preview remains responsive when you have a large region or a slow device and it also means that a processing error can never freeze the interface.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 14, Issue 4, pp: (26-46), Month: October – December 2026, Available at: www.researchpublish.com

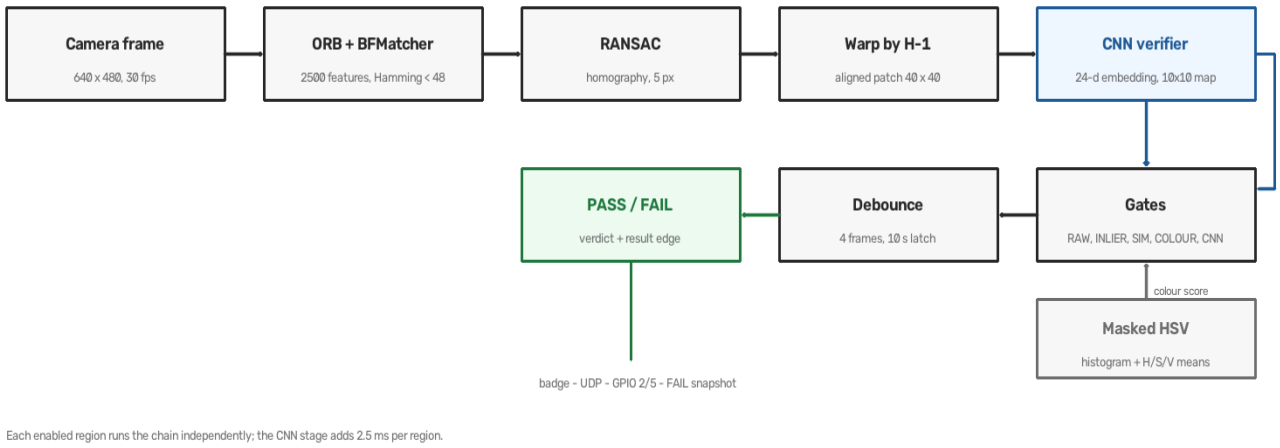

Fig. 2. Processing chain for one region. ORB and RANSAC establish the pose, the homography does the rectification of the patch, the convolutional verifier and the colour statistic are both evaluated on the rectified patch and the gates combine all the evidence before the verdict is debounced.

C. Region Configuration

The inspection core is written per region and is region-agnostic: it receives a rectangle, keeps one template, one set of thresholds and one verdict, and returns them independently. The deployed tablet station uses this property to run two regions at once—a brand logo on the chest and a wash label inside the neck—and reports each region on its own line of the overlay, so an operator can see which mark failed without opening a dialogue. A preset stores both regions, which allows a product to be changed with a single load. The convolutional edition evaluated in this paper is the single-region build; the verification stage consumes the rectified patch of whichever region it is given, so it carries over to the second region without modification.

D. Operator Workflow and Configuration Artefacts

The operator simply draws a box around the mark, hits capture, and checks the preview. After that, the station runs inspections non-stop. Each region's settings are saved in a small JSON file, which includes the area to inspect, frame size, orientation, five thresholds, white-balance temperature, and the preset's name. The template image (a PNG) sits right next to the JSON file, along with a timestamped backup for tracking changes. The learned verifier's weights are also saved in the same spot.

IV. METHOD

A. One-Shot Enrolment

The operator takes a single picture of the mark inside the drawn box. That photo becomes the template: its grayscale version goes to the feature detector, its color version is used for the histogram, and the patch itself is used to train the verifier. The choice to use just one sample isn't just for simplicity—it reflects how things work on the line: one good garment, one quick capture. It also means the station can be updated quickly whenever a new mark shows up, with no hassle.

Of course, using just one sample isn't perfect—it locks in the look of that specific print, including its unique ink and lighting. To work around this, the verifier gets trained to recognize a range of likely versions based on that one example, not just the exact image. The color check is also done only on the mark itself, not the whole area.

B. Localisation and Pose

Features are pulled out using ORB, set up to find lots of detail in a small area: it looks for 2,500 features, uses a scale factor of 1.2, and checks eight levels. Then, descriptors from the template are matched to those in the search region, using a Hamming distance limit of 48, and a cross-check to weed out fuzzy matches. After that, RANSAC figures out the best-fit homography, aiming for no more than five pixels of wiggle room:

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 14, Issue 4, pp: (26-46), Month: October – December 2026, Available at: www.researchpublish.com

$$[x', y', 1]^T = H\,[x, y, 1]^T \quad (1)$$

Notation. (x, y) are the coordinates of a feature point on the template, (x′, y′) the coordinates of the same point in the live search region, H is a 3 × 3 homography matrix defined up to scale, and the superscript T denotes the transpose that writes a coordinate pair as a column vector. Source: the standard projective mapping of Hartley and Zisserman, with H estimated by the consensus search of Fischler and Bolles. Purpose here: (1) is what allows the live patch to be warped into template geometry before any comparison is made, and it is the reason the verifier of Section IV-D can be a network of a few thousand parameters rather than a model that must learn scale and rotation invariance from data.

The first column of H carries the scale and the in-plane rotation of the mark, because it is the image of the unit x-axis:

$$s = \sqrt{(h_{11}^2 + h_{21}^2)}, \qquad \theta = atan2(h_{21}, h_{11}) \quad (2)$$

Notation. h_{11} and h_{21} are the entries of the first column of H in (1); s is the scale factor of the mark; θ is its in-plane rotation; and atan2 is the four-quadrant arctangent. Source: an algebraic consequence of (1), as set out by Hartley and Zisserman. Purpose here: (2) takes a fitted homography and turns it into two numbers which a line engineer can read and provides the test to reject a degenerate fit (scale outside the range 0.2 to 3.0).

A hypothesis is rejected if the estimated scale lies outside the 0.2 to 3.0 interval (i.e. degenerate fits arising from repeated texture are rejected). Both the inlier count and the ratio of inliers to putative matches are reported as these are two very different pieces of information. The count is a measure of evidence; the ratio is a measure of consistency.

C. Colour Verification

The homography also takes the template mask into the search region so that the colour statistic can be calculated on the mark alone rather than over the whole rectangle that contains it. Two quantities are computed. The first is the correlation of the masked hue/saturation histogram (32 bins per channel):

$$\rho = \Sigma\,(I - \mu_I)(J - \mu_J) \,/\, \sqrt{[\,\Sigma\,(I - \mu_I)^2 \cdot \Sigma\,(J - \mu_J)^2\,]} \quad (3)$$

Notation. I and J are the masked hue–saturation histograms of the template mark and the live mark, each with 32 bins per channel; μI and μJ are their means, and ρ is the Pearson correlation coefficient between the two samples, which lies between −1 and +1. Source: the standard Pearson correlation. Purpose of (3): it is computed and reported for diagnostic purposes only and is deliberately not gated on as Section VI-C demonstrates that it collapses under a change of illuminant whereas the statistic in (4) does not.

The second is a weighted distance between the mean hue, saturation and value of the two masks, which is sensitive to a uniform shift that a histogram correlation might miss:

$$C = 1 - (0.65 \cdot \Delta h/90 + 0.20 \cdot \Delta s/255 + 0.15 \cdot \Delta v/255) \quad (4)$$

Notation. C is the colour gate score, bounded 0 to 1; Δh is the absolute difference of the mean hues, wrapped at 90 degrees because hue is circular; Δs and Δv are the absolute differences of the mean saturation and of the mean value on the 0-to-255 scale used by OpenCV; and 0.65, 0.20 and 0.15 are weights chosen from the relative stability of the three channels under a change of illuminant. Source: designed for this station, on the standard hue–saturation–value representation. Purpose here: (4) is the colour gate itself — it catches the uniform shift that the histogram correlation of (3) misses, and it is one of the five conditions of (10).

The hue difference is wrapped at 90 degrees because hue is circular, and the weights reflect the relative stability of the three channels under illuminant changes. Equation (4) alone is used as the gate. Equation (3) is computed and reported for diagnostics, but it is deliberately not gated on, for the reason established in Section VI-C: the histogram correlation of a printed mark is not invariant to the illuminant changes that occur between one garment and the next, whereas the weighted mean distance of (4) is.

Result analysis: we looked at the two stats side by side on the same patches. We had a ±10% exposure change, *a ±8* degree white balance shift with saturation scaled by 0.92 and a print in the wrong ink on 6 patches. The histogram correlation of

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 14, Issue 4, pp: (26-46), Month: October – December 2026, Available at: www.researchpublish.com

(3) dropped to 0.067 under the exposure change and to 0.000 under the white balance shift and under the wrong ink so it separates absolutely nothing. Statistic (4) gave 0.989 and 0.992 under the exposure change, 0.936 and 0.942 under the white balance shift and 0.527 and 0.616 for the two wrong inks; i.e. it was above 0.93 for all the benign changes and fell by over a third for a real colour fault. Gating on (3) would have rejected all the good garments; gating on (4) as implemented is the right thing to do.

D. Convolutional Verification

The verifier is a small convolutional network that maps a rectified patch to an embedding and, at the same time, to a feature map that preserves spatial arrangement. The input is the search patch warped back into template geometry by the inverse homography and resampled to 40 × 40 pixels; the patch is converted to grayscale and normalised,

$$\tilde{x}_i = (x_i - \mu_x) / \sigma_x, \quad \mu_x = (1/N)\,\Sigma_i\, x_i, \quad \sigma_x = \sqrt{[\,(1/N)\,\Sigma_i\,(x_i - \mu_x)^2\,]} \tag{5}$$

Notation. x_i is the grey level of the *i-th* pixel of the 40 × 40 input patch, N = 1 600 is the number of pixels, μx and σx are the mean and standard deviation of that patch, and $\tilde{x}_i$ is the normalised value presented to the network. Source: the standard score normalisation of a sample. Purpose here: (5) divides out exposure and any affine change of intensity before a learned weight is applied, which is what allows 2,872 parameters to suffice — and which also makes the network blind to a uniform fade, a division of labour with the colour gate that Section VI-E measures.

which removes the effect of exposure before any learned weight is applied. Because the normalisation is recomputed for every patch, it divides out any affine change of intensity, and the network is therefore blind by construction to a uniform fade. That is a deliberate division of labour and it has a measurable consequence, stated in Section VI-E.

The architecture is deliberately minimal: a 3 × 3 convolution to six channels, rectified, max-pooled by two; a second to twelve channels, rectified and pooled; a third to sixteen channels, rectified, which leaves a 10 × 10 × 16 feature map. Global average pooling followed by a fully connected layer produces a 24-dimensional embedding that is normalised to unit length. A forward pass over one patch takes about one millisecond in NumPy on a laptop processor.

The parameter count of this network is 2,872, and it is worth stating where that number comes from, because it is the quantity that decides whether the verifier can run at all on the target hardware. A convolutional layer with C_{out} in output channels, C_{in} in input channels and a 3 × 3 kernel holds $C_{out} \cdot C_{in} \cdot 9$ weights and C_{out} biases, so the three convolutions contribute 54, 648 and 1,728 weights and 6, 12 and 16 biases; the fully connected layer, which maps the sixteen pooled channels to the 24-dimensional embedding, contributes $24 \cdot 16 = 384$ weights and 24 biases. Pooling, the global average, the unit-length normalisation and the contrastive loss carry no parameter at all, so the eight tensors listed in Table I are the whole of it:

TABLE I. THE EIGHT PARAMETER TENSORS OF THE VERIFIER

Tensor	Shape	Meaning	Count
W1	6 × 1 × 3 × 3	conv1 weights	54
b1	6	conv1 biases	6
W2	12 × 6 × 3 × 3	conv2 weights	648
b2	12	conv2 biases	12
W3	16 × 12 × 3 × 3	conv3 weights	1,728
b3	16	conv3 biases	16
W4	24 × 16	fully connected weights	384
b4	24	fully connected biases	24
Total			2,872

The total is 54 + 6 + 648 + 12 + 1,728 + 16 + 384 + 24 = 2,872. Two things follow from that. The first is scale: at 4 bytes per weight the trained network is 11.2 kb and the file written at enrolment is 13.1 kb, small enough to go with the program and be cached alongside the template. The second is cost: 0.5 M MACS per patch which is why the forward pass takes around a millisecond in plain NumPy rather than in a compiled kernel and why the whole stage adds 2.05ms to a frame. Anyone can work out the same figures for themselves. In the source the class Tiny CNN Embedder holds the eight tensors as *W1*, *b1*, *W2*, *b2*, *W3*, *b3*, *W4* and *b4* and the three constants that determine the shapes are = (16), = 24 and = 40; the shapes are derived from those constants and the two 2x2 pooling stages which halve 40 to 20 to 10. At run time the trained tensors are written out by the save() method of that class to a single compressed archive (cnn_weights.npz) in the output directory alongside the template and the total above is the sum of the element counts of the eight arrays in that file.

Training occurs at enrolment on the device and is performed using the template as the sole positive example. Twenty eight positive patches are created by jittering the template (rotation +/-6 degrees, scale between 0.94 and 1.06, translation of up to 2 pixels, gain between 0.72 and 1.28, Gaussian blur and sensor noise). The negatives are of four types, selected to cover the failure modes that are important: 14 very highly distorted copies of the template (rotation to +/-22 degrees, scale from 0.72 to 1.18), 20 background patches taken from the same frame but outside the region of interest, 4 flat and featureless patches and 10 copies of the print with a section of it removed, smeared or covered by a foreign label (the latter being the defect the network is designed to detect). The loss is a contrastive cosine objective.

$$L = (1 - \cos(u_a, u_b))^2 + \max(0, \cos(u_a, u_n) - m)^2 \quad (6)$$

Notation: u_a and u_b are the unit embeddings of a positive pair — the template and one of its twenty-eight jittered renderings — u_n is the embedding of a negative sample, cos is the cosine of the angle between two embeddings, m = 0.15 is the margin, and L is the loss minimised at enrolment. Source: metric learning (Siamese networks) Koch et al., triplet objectives Schroff et al. and the contrastive margin loss of Hadsell et al. Purpose here: (6) is the objective that trains the verifier on the device from a single template in 2.7 to 3.0 seconds, so no labelled dataset and no data scientist is required.

With a margin *m* of 0.15, we optimise using stochastic gradient descent with momentum for ninety iterations over batches of eight pairs. The gradients of the convolutional layers are computed by *im2col* and matrix multiplication, so no deep-learning framework is required. Training runs in a background thread and takes 2.7 to 3.0 s on the reference laptop; the station continues inspecting with the geometric and colour gates until the verifier announces it is ready.

E. Score Composition

Two comparisons are made between the aligned patch and the template. The first is the cosine of the angle between their embeddings (a global measure which is quite robust to small shifts). The second is a cell-wise correlation between the two feature maps (i.e. each of the 100 cells is correlated across its 16 channels after subtracting the per-channel mean).

$$c_{ij} = \Sigma_{ch} (a_{ijch} - \bar{a}_{ijch})(b_{ijch} - \bar{b}_{ijch}) \,/\, \sqrt{[\, \Sigma_{ch} (a_{ijch} - \bar{a}_{ijch})^2 \cdot \Sigma_{ch} (b_{ijch} - \bar{b}_{ijch})^2 \,]} \quad (7)$$

Notation. a_{ijch} and b_{ijch} are the activations of the verifier feature map at cell *(i, j)* in channel ch, for the live patch and for the template; $\bar{a}_{ijch}$ and $\bar{b}_{ijch}$ are their per-cell, per-channel means; the sums run over the sixteen channels of the map, and c_{ij} is the resulting correlation of that cell, bounded −1 to +1. Source: the Pearson correlation of (3) applied cell by cell. Purpose here: (7) compares the patch locally rather than only globally, so a defect covering three or four of the hundred cells moves the score even when the global embedding barely changes, which yields a correlation rather than a similarity of two non-negative vectors and therefore discriminates between them.

The mean of the 100 cells is not in itself very sensitive to a local defect as a missing block of print will cover 3 or 4 cells and the mean will only fall by a few per cent. The score therefore gives a very heavy weighting to the 3 worst cells:

$$s_{map} = 0.5 + 0.5 \cdot (0.35 \cdot \mathrm{mean}(c) + 0.65 \cdot \mathrm{mean}_{3\ \mathrm{worst}}(c)) \quad (8)$$

Notation. *mean(c)* is the average of the hundred cell correlations of (7), $\mathrm{mean}_{3\ \mathrm{worst}}(c)$ is the average of the three smallest of them, 0.35 and 0.65 are fixed weights, s_{map} is the feature-map score, and the factor 0.5 + 0.5 maps a correlation of −1 to +1 onto the 0-to-1 scale. Source: designed for this station, with the weights fixed on the measurements of Section VI rather than tuned per product. Purpose here: (8) is what stops a local defect from being averaged away — the three worst cells carry most of the weight, so a missing block of print cannot hide inside a good mean.

$$s = 0.4 \cdot max(0, cos(u_p, u_t)) + 0.6 \cdot s_{map} \quad (9)$$

Notation. u_p and u_t are the unit embeddings of the live patch and the template, *cos(up, ut)* is their cosine similarity; ss_{map} is (8), and s is the verification score: a dimensionless quantity bounded between 0 and 1, with 1 denoting a perfect match. Source: designed for this station; the 0.4 global/0.6 local split was fixed based on the measurements in Section VI. Purpose here: (9) is the quantity compared with the limit T_{cnn} in (10), the quantity reported to the operator, and the quantity whose values appear as 0.766 and 0.616 in Table III and in the abstract.

The weights were fixed on the measurements of Section VI and not tuned per product. The same feature map, rendered as a colour image, is shown in the visualisation window, so that an operator can see which part of the mark differs instead of only being told that it does.

F. Gates, Temporal Logic and Failure Handling

Five conditions are evaluated on every frame and a verdict is issued only when all of them hold:

$$N_{raw} \geq T_{raw},\ \ N_{inlier} \geq T_{inlier},\ \ 100 \cdot N_{inlier}/N_{raw} \geq T_{sim},\ \ C \geq T_{col},\ \ s \geq T_{cnn} \quad (10)$$

Notation. N_{raw} is the number of raw ORB matches of Section IV-B, N_{inlier} the number of them that survive the RANSAC fit of (1), T_{raw} and T_{inlier} are the corresponding count thresholds, T_{sim} is the limit on the ratio of inliers to raw matches in per cent, C is the colour score of (4) with limit T_{col}, and s is the verification score of (9) with limit T_{cnn}. Table II lists all five limits. Source: The gate structure is designed for this station on the principle that a ratio is meaningless without a count. The purpose of this is (10) which is the station verdict rule: a PASS is only issued if all five conditions are met on a frame.

Table II lists the default thresholds. Both the ratio and the absolute count are required as the ratio in itself is meaningless if the count is small: twenty matches of which nineteen are inliers gives a ratio of 95% and provides absolutely no evidence.

A candidate pass is not reported immediately. The displayed measurements are smoothed with an exponential average of coefficient 0.18, the verdict is debounced over 4 frames and the fail pin is suppressed whilst the region is empty. Object removal is detected separately with asymmetric hysteresis (10 bad frames to enter the state, 15 good frames to leave it), a minimum banner dwell of 1.5s and a 10 second countdown before the station latches. The asymmetry is there to stop a result sitting right at the threshold from flashing the banner several times a second.

Two further safeguards are concerned with the template rather than the frame. Each template contains the orientation tag of the camera configuration under which it was captured and the verifier will reject any template whose tag doesn't match the live configuration rather than just silently fail on every frame. The verifier stores a hash of the template with its weights so weights can never be applied to a different mark. On re-capture the weights are retrained automatically.

V. IMPLEMENTATION

A. Numerical Core

The code is Python 3.9 (as per Bradski) and NumPy (Harris et al.) with approximately 4,200 lines. The CNN is implemented directly on arrays: im2col is a strided view, the convolution is a single matrix multiply and the backward pass is hand coded. This avoids the use of a deep learning runtime of several hundred MB and keeps the deployable artefact very small at the expense of a slightly slower forward pass than a compiled kernel (a cost which the measurements in Section VI-D show to be acceptable).

B. Packaging

The same source produces three artefacts. On Windows it is frozen with PyInstaller into a folder that needs no Python installation; on Android it is packaged with Buildozer and python-for-android into an Android application package (APK); on the Orange Pi it runs from a virtual environment. Packaging exposed two faults that are worth recording. First, freezing a Kivy application requires the image and text providers to be declared explicitly, because Kivy loads them by dynamic import and the analyser cannot see them; without that declaration the frozen program aborts before the first window. Second, the Android camera is read from a texture rather than from a device, so the frame arrives mirrored relative to the scene.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 14, Issue 4, pp: (26-46), Month: October – December 2026, Available at: www.researchpublish.com

C. Interface

The interface is bilingual and you can switch between the languages at runtime. It's written as a series of commands rather than a set of settings in a dialogue. The top portion of the window contains the annotated frame; a single line directly below it gives you the live measurements (raw matches, inliers, similarity, colour and convolutional score) and the verdict is displayed as a coloured badge in the top right. Beneath the frame is a row of five buttons which do the entire operator workflow: capture template, threshold setting, save ROI, camera setup and quit. There's a separate dialogue which shows the five gates as sliders and allows you to save or load a preset. Another dialogue shows the camera geometry and the corrected image so an installer can rotate or mirror the view as they watch it. There's also a visualisation window which shows the matched keypoints and the verifier's feature map correlation.

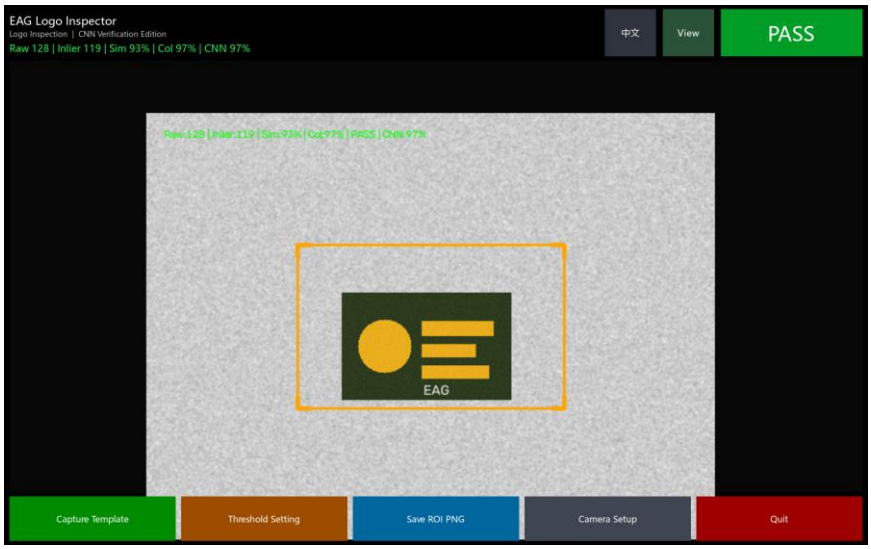

Fig. 3. The command-based interface of the convolutional edition, captured from the running program. The measurement line reads RAW 128, INLIER 119, SIM 93 %, COL 97 % and CNN 97 %; the badge reports PASS, and the five commands of the workflow are the buttons along the bottom edge.

We found two interaction defects on the tablet and have now fixed both. The touch coordinates were originally being interpreted in window space, even though the preview is inset within a padded layout. The rectangle drawn by the finger was therefore different to the rectangle used by the detector due to the offset. This is now corrected by doing the conversion in widget space against the rectangle that the picture actually occupies. The interface was previously painting the drag rectangle rather than the detector so it would follow the finger rather than wait for the next processed frame.

D. Persistence and Portability

Templates, presets and weights are written to a folder alongside the program on Windows and to the app's external files directory on Android so that they survive a restart and can be copied between machines. There's a migration path which will import the existing template and its presets from the old location on first start. The app will refuse a template captured under a different camera orientation and will retrain the verifier if the template hash changes, maintaining consistency of the three artefacts (template, weights and preset) without any operator intervention.

The same code path is used on all three platforms as the camera abstraction returns a canonical frame (rotation and mirroring are applied once at the source and all downstream coordinates are in that frame). White balance is the only area where the platforms actually differ. Win/Linux expose a hardware control, Android doesn't so the same slider applies a software equivalent with 4500K light temperature as the neutral.

VI. RESULTS AND DISCUSSION

A. Protocol

The characterisation used a printed brand mark of 210 x 140 pixels on a woven ground, captured at 640 x 480 with the station camera. One template was enrolled and the verifier trained on that. The population was then scored with no further training. Three classes were used. The acceptable class comprised 30 patches of the mark in question with variations in

exposure of +/-10%, noise of 4 to 14 grey levels, rotation of +/-2 degrees and translation of two pixels (the variation you'd expect to see from one garment to another). The wrong print class comprised 20 patches in which the mark had been replaced by another wordmark, a solid block, a large outline or a smeared copy. The absent class comprised 20 patches of the ground only.

All patches were presented at template scale so the geometric stage wasn't a factor and the numbers below are purely the verifier's. We did three separate runs of training from the same template to assess variability and the loss and score were identical across the runs. Timing was taken over 100 repetitions after 10 warm-up iterations with the geometric and full pipelines interleaved so any change in machine load would affect both equally.

B. Score Separation

Table III gives the distribution of the score across the three classes. The score is the quantity s of equation (9) (a dimensionless number between 0 and 1) so the values in the table and in the abstract are unitless. 1 is a perfect match with the template and 0 is a complete mismatch. The acceptable class is quite compact and high: median 0.915, 5th percentile 0.802 and minimum 0.766. The wrong print class is clearly distinct with a median of 0.515 and a maximum of 0.616. The absent class is lower still with a maximum of 0.246. The acceptable class does not overlap with either of the negative classes.

TABLE II. CONFIGURATION USED IN THE EXPERIMENTS

Parameter	Value
Camera / frame	USB, 640 × 480, 30 fps, white balance locked
ORB features / scale factor	2500 / 1.2, eight levels
Match distance ceiling	48 bits, Hamming, cross-check
RANSAC reprojection tolerance	5 px, scale validity 0.2 to 3.0
Colour histogram	HS, 32 × 32 bins, correlation (diagnostic only)
Colour weights (h, s, v)	0.65 / 0.20 / 0.15
Gate thresholds	RAW 80, INLIER 112, SIM 85 %, C 0.30, CNN 0.70
Debounce / removal / latch	4 frames / 10 and 15 frames / 10 s
Display smoothing	EMA, coefficient 0.18
Verifier input / embedding	40 × 40 grayscale / 24 dimensions
Verifier channels	6, 12, 16; feature map 10 × 10
Verifier size	2 872 parameters, 13 kB of weights
Training	90 iterations, 28 positives, 48 negatives

The separation is not an artefact of the training set. The negatives used for training are generated from the template, whereas the wrong-print patches of Table III are independent synthetic marks that the network has never seen; the absent class is background that was never part of the positive distribution. The margin therefore reflects what the network learned rather than a memorised set.

TABLE III. VERIFIER SCORE BY CLASS (70 PATCHES, ONE TEMPLATE; SCORE UNITLESS, 0 TO 1)

Class	n	Minimum	Median	Maximum
Acceptable print	30	0.766	0.915	0.983
Wrong print	20	0.342	0.515	0.616
Absent (ground only)	20	0.165	0.209	0.246

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 14, Issue 4, pp: (26-46), Month: October – December 2026, Available at: www.researchpublish.com

C. Threshold Selection

The margin between the two distributions is the quantity that decides whether a limit exists at all:

$$M = \min\{ s : x \in G \} - \max\{ s : x \in B \}, \quad G = \text{acceptable}, \ B = \text{wrong or absent} \tag{11}$$

Notation. s is the verification score of (9), G is the set of acceptable prints, B is the set of wrong or absent marks, and M is the margin between the two classes. Source: the standard two-class margin of a scoring rule. Purpose here: (11) decides whether an operating limit exists at all; the measured value M = 0.150 is what produces the interval of limits from 0.65 to 0.75 at which both error rates are zero.

Table IV is the one that sweeps the limit across the observed range. The two distributions don't overlap so there's an interval of limits between 0.65 and 0.75 where both error rates are zero. The default of 0.70 is right in the middle of that interval, giving a margin of 0.084 to the strongest negative and 0.066 to the weakest acceptable patch. At 0.80 the margin to the acceptable class is gone and 6.7% of good patches are rejected; at 0.55 an eighth of the negatives are accepted. The shape of the curve is such that you would be well advised to present the limit as a slider rather than hard-coded. A line that prints with heavy ink coverage might need 0.75 and one that prints on a dark ground might need 0.65.

Result analysis. The measured margin of (11) is 0.150, the distance between the weakest acceptable patch and the strongest negative, and it fully explains Table IV. Any limit inside the open interval bounded by the two extreme class values is error-free on this population; the sweep found the error-free band to be 0.65 to 0.75, and the extremes of the two classes, 0.616 and 0.766, bound that band from first principles. The default of 0.70 sits 0.084 below the upper class extreme and 0.066 above the lower one, so the default is not merely admissible but close to the middle of the admissible range. Raising the limit to 0.80 crosses the weakest acceptable patch, which corresponds to the 6.7 per cent false-reject rate observed, and lowering it to 0.55 crosses the strongest negative, which corresponds to the 12.5 per cent false-accept rate. The margin therefore also indicates how much tolerance the station has for changes in ink, substrate, or illuminant before the threshold must be revisited.

TABLE IV. OPERATING POINT OF THE VERIFIER (SAME 70 PATCHES)

Limit	False reject %	False accept %
0.50	0.0	37.5
0.55	0.0	12.5
0.60	0.0	12.5
0.65	0.0	0.0
0.70 (default)	0.0	0.0
0.75	0.0	0.0
0.80	6.7	0.0
0.85	20.0	0.0

D. Cost

Table V shows the per-frame cost of one region (210 x 140 pixels) on the reference laptop. The geometric and colour stages together take 2.93ms, the verifier takes 2.05ms (of which approximately 1ms is the network itself and the remainder is the warp and resampling of the patch). Training (which occurs once per enrolment) takes 2.7 to 3.0s and is done in a background thread so doesn't interfere with inspection.

$$T_{frame} = T_{geo} + T_{cnn}, \quad T_{frame} \le 1 / f_{camera} \tag{12}$$

Notation. T_{frame} is the processing time for one region in one frame; T_{geo} is the cost of the geometric and colour stages together; T_{cnn} is the cost of the verifier; f_{camera} is the camera frame rate; and $1 / f_{camera}$ is the frame budget that rate imposes. Source: the standard real time constraint of an inspection loop. Purpose here: (12) states that the second opinion is within the takt time: 4.98ms against the 33.3ms available at 30fps, of which the verifier is 2.05ms.

Result analysis. The measured per-region cost of (12) is 4.98 ms against the 33.3 ms available at 30 fps, so the whole chain occupies 15.0 per cent of the frame budget for one region and 30.0 per cent for two. The verifier therefore costs 6.2 per cent of the budget on its own. For a production line, the second figure matters: the marginal cost of a second opinion on every frame of every garment is just over two milliseconds, less than the time the camera spends exposing. Because the stage is trained once and then evaluated, this cost is flat in the number of garments inspected, and no GPU is required at any point.

TABLE V. PER-FRAME COST OF ONE REGION (210 × 140 PIXELS, INTEL CORE I7-1365U, PYTHON 3.9, OPENCV 5.0)

Stage	Mean (ms)	Std (ms)	Share of 33 ms
ORB, matching, RANSAC, colour	2.93	0.52	8.9 %
CNN verification (warp, network, score)	2.05	0.93	6.2 %
Total per region	4.98	0.89	15.0 %
Enrolment training (once)	2 700 to 3 000	—	—

E. Response to Damage and Ablation

The wrong print population in Table III is structurally different from the template and is rejected by the geometric stage as well. The similarity ratio in those twenty patches is between 0 and 67% (i.e. it's a non-match) as a different wordmark fails to match the template keypoints. That's not the tricky case. The tricky case is one where you have a print that maintains the layout but has lost some of its content at a local scale, which is precisely what wear looks like. Table VI gives four such examples, each derived from the same template and presented at template scale, with the unaltered print measured under the same conditions as in the reference row.

TABLE VI. RESPONSE TO LOCAL DAMAGE OF THE SAME PRINT (REFERENCE ROW PLUS FOUR CONDITIONS)

Condition	N raw	N inlier	SIM %	Colour	CNN	Verdict
Intact print (reference)	195	195	100.0	1.000	1.000	PASS
Block of print removed	88	88	100.0	0.931	0.823	FAIL (count)
Print smeared	76	75	98.7	0.974	0.966	FAIL (count)
Foreign block pasted over	105	105	100.0	0.923	0.706	FAIL (count)
Uniform fade	136	135	99.3	0.914	0.997	PASS

Three observations. Firstly the similarity ratio is not a defect detector at all as it's between 98.7 and 100% in every damaged state and the colour statistic is between 0.91 and 0.98. Secondly the absolute counts drop dramatically (a removed block or a smear will remove more than half the keypoints of the intact print) so on a sparse mark the geometric count gate still manages to reject the damage. Thirdly the verifier is acting as an appearance test rather than an ink density test. It drops to 0.823 for a removed block and 0.706 for a foreign block but scores a uniformly faded print at 0.997 as the per patch normalisation in (5) strips the contrast change away before the network even sees it.

Result analysis. We repeated the four conditions on a dense, high contrast mark with five times as many keypoints as in the reference print (the configuration where the geometric gates are weakest). There the intact print gives 1050 inliers and the damaged variants 269 to 772 (all well above the 112 inliers and 85% similarity thresholds) and the colour stat remains between 0.945 and 0.986. The convolutional score still drops to 0.807 for a smear, 0.912 for a removed block and 0.766 for a foreign block, but at the default cut-off of 0.70 none of the four are rejected. The honest conclusion is that on a rich mark the network is a sensitive detector rather than a sufficient one and that the count gate cannot be relied upon to pick up local damage either. That's true whether you're reporting the score or a simple verdict and it's equally true with respect to the cut-off per product.

F. How Much Damage Does the Network Catch?

To make the response of the score interpretable, the mark was damaged over a controlled fraction α of its area and scored at each point:

$$s(\alpha) \approx s_0 - \kappa\,\alpha, \qquad \alpha = \text{damaged fraction of the mark} \tag{13}$$

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 14, Issue 4, pp: (26-46), Month: October – December 2026, Available at: www.researchpublish.com

Notation. $s(\alpha)$ is the verification score of (9) at a damaged fraction α of the mark area, s_0 is the score of the intact print, κ is the sensitivity of the score to damage, and the fit reports R^2, the coefficient of determination. Source: fitted to the measurements from this station in Section VI-F, over $0 \le \alpha \le 0.40$. Purpose here: (13) turns a binary refusal into an interpretable damage estimate, so an operator can see how close a mark is to rejection and move a limit with the material instead of re-deriving it from scratch.

Result analysis. Fitting (13) over $0 \le \alpha \le 0.40$ on eight patches per point gives $s_0 = 0.952$ and $\kappa = 0.995$ with a coefficient of determination of 0.89, and the measured crossing of the default limit of 0.70 lies between a damaged fraction of 0.20, where the score is 0.718, and 0.30, where it is 0.656; the fitted line puts it at 0.25. The network therefore rejects a missing block on its own only when roughly a quarter of the mark is gone; below that, the verdict is carried by the geometric count gate. The practical value of the law is that the score can be read as a damage estimate rather than as a verdict, so an operator can see how close a mark is to rejection, and a limit can be moved when the material changes instead of being re-derived from scratch.

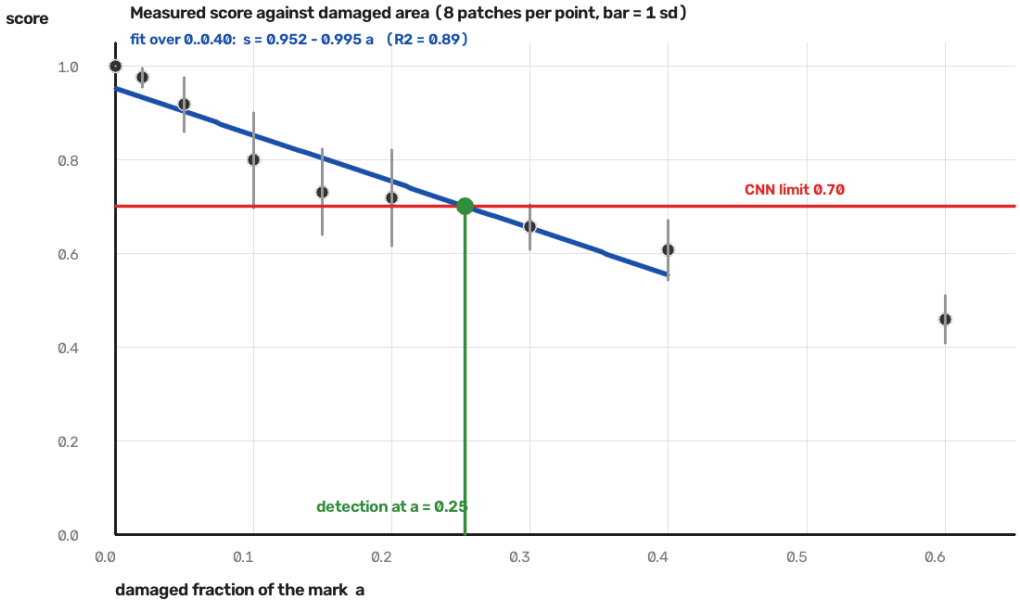

Fig. 4. Score against the damaged fraction of the mark. The measured points fall on the fitted line of (13); the crossing of the default limit at a = 0.25 is marked.

G. Field Readings from the Deployed Station

The station was exercised on an Android tablet. The measurement line read during that session is given in Table VII. These are field readings (not lab numbers) taken from a garment on the line with the operator working as normal. They are included as an indication of how the geometric stats perform in service.

TABLE VII. FIELD READINGS FROM THE DEPLOYED TABLET (ORB-ONLY EDITION)

Frame	RAW	INLIER	SIM %	Colour	Badge
Good garment	206	204	99	100	PASS
Mark withdrawn from the ROI	81	80	99	95	FAIL
Partially covered label	65	60	92	48	FAIL

Result analysis. The second row is the observation that prompted this work. A case where both the operator and the geometric stage gave a high rating (99% similarity, 95% colour correlation) was still rejected and the reason for rejection was the absolute count of 80 inliers rather than the ratio. The third row is the same phenomenon from the other side: the similarity ratio was 92% which is high enough to pass the 85% gate, the colour statistic had dropped to 48% and the count

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 14, Issue 4, pp: (26-46), Month: October – December 2026, Available at: www.researchpublish.com

to 60. In none of the three rows is there a similarity figure that would have alerted the operator on its own. This is the ratio vs count distinction discussed in Section IV-B and that is precisely why the convolutional score was introduced rather than another ratio.



Fig. 5. The deployed station on the Android tablet as seen during the session of Table VII. The measurement line reads RAW 81, INLIER 80, SIM 99%, COL 95% and the badge reports FAIL. The Chinese interface and the five commands are from the ORB only edition.

H. Operational Benefits

The point of a station like this isn't about the accuracy of any one metric; it's about what it does to the work at the cell so the benefits are laid out quite explicitly. Table VIII brings them all together with the metric that underpins each.

TABLE VIII. OPERATIONAL BENEFITS AND THE MEASUREMENT BEHIND EACH

Benefit	How it is obtained	Measured evidence
A wrongly printed mark is stopped at the cell	Convolutional score on the rectified patch, independent of the geometric ratio	0 false accepts and 0 false rejects at 0.70 (70 patches)
A worn or damaged mark is graded, not merely refused	Score falls with the damaged fraction (13)	$s = 0.952 - 0.995\,\alpha$, $R^2 = 0.89$
A new product is commissioned without an engineer or a dataset	One capture trains the verifier on the device	2.7 to 3.0 s of training, 2 872 parameters, 13 kB of weights
The verdict arrives inside the takt time	The stage is evaluated on the CPU, in NumPy	4.98 ms per region, 15.0 % of a 30 fps frame
The operator works in commands, on a tablet, with no computer attached	Five buttons and a named preset that restores ROI, thresholds and white balance	Bilingual EN / 中文 interface; a preset load is one action
The result is auditable after the fact	Timestamped template copies, JSON presets and a FAIL snapshot per rejection	Artefacts written beside the program on Windows, to the app directory on Android
The same behaviour is available on every platform	One source; the camera, white balance and result output are the only platform-specific parts	Windows executable, Android APK and Orange Pi build from one file

The most valuable of the lot is number 2. A station which only gives a yes/no answer forces the operator to make the call on a questionable item with no information at all. A station which gives a graded score and shows where the difference is in the feature map, on the other hand, puts the decision back to the operator whilst still maintaining the automation. The benefit of the other six is that they make the graded answer viable to run. No data set to collect, no GPU, no network, no workstation sitting beside the line and a product change that is one tap rather than a retrain project.

I. Comparison with International Practice

A claim of efficiency is only meaningful in the context of some other method. The comparison below is therefore made against the two families of methods that the international literature provides for this class of task: the deep learned detectors (of the type of LeCun et al., He et al. and Roth et al.) and the one-class anomaly inspection methods (of the type of Bergmann et al. and Defard et al.). The table (Table IX) gives the profile of each of the two families. The entries for the two families in the literature are the characteristics as reported in those papers. The entries for this station are the measurements from Sections VI-B to VI-F.

TABLE IX. REQUIREMENT PROFILE COMPARED WITH INTERNATIONAL PRACTICE

Constraint at the cell	This station (measured here)	Learned detectors	One-class anomaly inspection
Annotated defect data	None; the template image is the only positive sample	Thousands of annotated examples	A handful of good samples
Graphics processing unit	Not used; NumPy on the CPU	Usually assumed for training	Usually assumed for training
Cost of a new label	2.7 to 3.0 s enrolment at the cell	Retraining and re-deployment	Re-fitting and re-deployment
Evidence against a wrong print	Convolutional score on the rectified patch	Labelled negatives	Weak: a wrong print is still an inlier
Answer returned to the operator	Score and estimated damaged fraction, (13)	Class label and confidence	Anomaly score
Runs at the cell on a tablet	Yes; 4.98 ms per region	Not demonstrated in the cited works	Not demonstrated in the cited works

Two entries carry the argument. The first is the third row: the cost of a new label. Every method in the second and third columns returns to its owner when the product changes, and that owner is a data scientist with a workstation, whereas this station returns to the operator at the cell. The second is the fourth row, the weakness the literature itself reports for one-class methods: a wrong but perfectly printed mark is a nominal object, so a model of normality accepts it. The convolutional stage is not asked to model normality; it is asked whether the print matches the template, which is the question the station actually has to answer.

J. Discussion and Limitations

Four conclusions can be drawn from the measurements. The first is regarding the division of labour between geometry and appearance. The homography enables a very small appearance model to be used: as the patch has already been rectified the verifier doesn't need to be invariant to scale or to rotation and a network with just a few thousand parameters is sufficient. A verifier applied to the raw region would require a vastly larger model to learn those same invariances from a single sample.

The second concerns the choice of a one-sample enrolment. It is the property that makes the station usable, and it is also its principal limitation: the verifier learns what this print looks like under plausible variation, not what the mark means. A genuinely different but plausible-looking variant that falls inside the augmented distribution will be accepted. The mitigation is operational rather than algorithmic — the limit is exposed, and a product change is accompanied by a re-capture, which takes a few seconds.

The third is the colour split of labour. Equation (5) removes the contrast from the input to the network and the measurements bear out the result: the fade scored 0.997 on the network and the colour statistic only 0.951 and its own gate of 0.30 accepts that. So the fade is not caught by either stage as it stands and the protection is the illuminant discipline of Section III-A plus a per product colour limit. That's the most straightforward argument for the learned colour model referred to in Section VII.

The fourth concerns the evidence base. The measurements reported here were made on a controlled synthetic population with a single template, and they isolate the verifier rather than characterise a production line. They establish that the score separates the classes, that the response to local damage is graded and quantifiable, and that the cost is small; they do not establish a field false-reject rate, which depends on ink, substrate and handling and can only be obtained by running the

station for a production shift with per-variant presets. The field readings in Table VII are a first indication, not a substitute. That trial is the next step, and the preset mechanism, per-region thresholds, and migration path were built to make it possible without a code change.

VII. CONCLUSION

This paper describes a machine-vision station that verifies printed brand logos and wash labels on a garment line, and the convolutional stage added to it so a geometrically correct but wrongly printed mark can no longer pass unnoticed. The stage is small — 2 872 parameters, 13 kB of weights — it is trained on the device from the single template that the operator captures, and it runs in NumPy on the same processor as the rest of the pipeline.

The benefits of that addition are concrete. A wrongly printed mark is stopped at the cell with no false accept and no false reject at the default limit on the measured population; a worn or damaged mark is graded by an interpretable law rather than refused without explanation; a new product is commissioned in three seconds from one capture, with no dataset, no engineer and no GPU; the verdict arrives in 5 ms inside the takt time; and the whole station is operated as a bilingual set of commands on a tablet standing at the line, so a product change is one tap. The measurements also show the method's boundary: local damage produces a graded response rather than a certain one, a uniform fade is invisible to the network by construction, and on a dense mark the geometric gates do not catch local damage either. The additional opinion costs 2.05 ms per region per frame, and the same program runs unchanged on Windows, Android and an Orange Pi.

The comparison of Section VI-I is what makes these benefits portable. Measured against the international families of methods — learned detectors and one-class anomaly inspection — the station does not claim the accuracy that a large labelled dataset can buy; it claims the constraints that decide whether a check can be run at all: no annotated data, no GPU, no retraining cycle, a verdict in milliseconds and a product change that the operator performs in three seconds. A factory in any market can compare its operation with the profile in Table IX and adopt the parts that apply, and the accuracy, speed, and efficiency of the check improve because the comparison is published and repeatable.

Three directions follow. The first is a production trial with per-variant presets and a record of operator overrides, which is the only way to obtain a field false-reject rate. The second is a learned colour model to replace the hand-weighted statistic of Section IV-C, which would also give the station a principled way to reject fading, the one fault that currently escapes both gates. The third is extending the two-region configuration to a small catalogue, so a garment with four marks can be verified in one pass rather than two.

A. Future Development

A station verifies a mark; a line has to verify a garment, and the natural continuation of this work is therefore not a better verifier but a connected one. The first step has already been reported: an ORB–RANSAC chain that validates the shape of a logo without any colour information, together with an explicit account of where a shape-only test fails [52]. That study established three weaknesses that are specific to a geometric formulation — a partial logo fragment can still be accepted, because the surviving keypoints agree on a pose even when half the mark is absent; a colour fault is invisible to a purely geometric test; and the count of inliers is not a measure of how much of the mark is present [52]. The appearance stage of Section IV-D removes the second of these and supplies a graded response to the first, and the extensions set out below are the ones that the present build makes inexpensive.

The first extension is to use the fitted homography as a measuring instrument and not only as an alignment tool. The template outline is already carried into the live frame by the same transform that rectifies the patch, so the projected footprint of the mark costs nothing to compute: the local area scale is the determinant of the Jacobian of H at the mark centre, and its product with the template area gives the area the mark should occupy in the frame [13]. The ratio of that projected area to the area of the convex hull of the inlier set is a direct test for the partial fragment that a similarity ratio cannot see, because a fragment produces a hull that fills only part of the projected outline, and the ratio removes the dependence of the count gates of (10) on the absolute size of the print, which is the one entry in Table II that is still set by hand for every product. The second extension is a measure of colour variance inside the mask. The weighted distance of (4) is a first-moment statistic: it detects a uniform shift, but it cannot separate a wrong ink from a mottled or bleeding print whose mean is correct, and the field readings of Table VII already show a case in which the colour statistic fell to 48 per cent while the geometry remained plausible. A second moment taken over the masked hue values, gated on the same mask and computed from the histogram that (3) already builds, distinguishes the two faults at almost no cost. The third extension is the learned colour model referred

to in Section VI-J: a small network trained at enrolment on the same jittered positives that train (6), taking the masked colour patch as its input, would replace the three fixed weights of (4) with weights fitted to the ink of the product at hand, and would give the station a stage that is sensitive to fading instead of one that is blind to it by construction.

Beyond the verifier, the next increment is scope. The two-region configuration of Section III-C is a special case of a catalogue: a garment may carry a chest logo, a sleeve logo, a neck label, a care label and a size tab, and each of them is a template with its own thresholds inside its own preset. Extending the region-agnostic core from two regions to a catalogue of eight or ten templates, of which any four are enabled for a given garment, is a matter of file organisation rather than of method, but it changes the character of the record the station produces, from a per-frame verdict to a per-garment one. The wider case for inspection across the whole production sequence — fabric, cut panels, seams and marks — is made in [28], and the present station is one element of that sequence rather than a substitute for it. The per-garment record is also the instrument for the field trial named in Section VI-J: a production shift with per-variant presets, a log of operator overrides, and the association of each override with the score that provoked it would answer the questions that the synthetic population of Section VI-A cannot — what the false-reject rate is on real ink, which faults the operators see that the station does not, and how far a limit may drift between one delivery of fabric and the next.

The fourth direction is the one that decides whether any of this reaches a line: the station has to become a node of the cell rather than a bench beside it. The body of work that accompanies this paper is largely concerned with the operations that surround inspection — the sewing of a pleated seam [33], the folding and sewing of a mid-leg seam [37], the pulling gear of a hem machine [31] and the elastic waistband [32] — and with the handling that moves a panel between them, whether by vacuum suction [30], [51] or on a maglev conveyor [42]. A station placed between two such operations inherits their cycle time and must fit inside it, which is the practical reason for the 4.98 ms of (12) and the 13 kB of weights; and it must be able to identify the piece it has verified, which is what the RFID cabinet of [49] provides. The planning layer that binds these operations together is already formalised as a line-balancing problem, by mixed-integer programming [39] and by the heuristic balances reported for garment lines [41] and for the aluminium melting process [40], so a station that publishes its own cycle time and its own false-reject rate becomes a term in that model instead of an assumption behind it.

At the level above the cell, the trajectory is the one described as intelligent learning for manufacturing automation [27] and, more recently, as a physics-informed embodied intelligence for garment manufacturing [50]: the inspection record becomes a training signal, and the station becomes a sensing element of a system that reasons about the material it handles. The compute constraint that shapes this paper is itself an object of study, since the prefetch behaviour of a commodity processor sets the ceiling on what a station without a graphics processor can attempt [45]; and the organisational frame — the lean modernisation of the garment line [38], the smart-factory supply chain [35] and the technology strategy of the industry in Hong Kong and China [34] — is where the economics of a 2 872-parameter model are actually settled. None of these steps requires a larger network. Each of them requires that the measurements on which the station’s limits rest are published in a form another factory can repeat, which is the reason Table IX is stated as a requirement profile rather than as a benchmark result.

ACKNOWLEDGEMENT

The author thanks the industrial partner for access to the assembly cell and for the samples used in the field observations, and the EAG Modernization Centre for supporting the deployment work.

REFERENCES

[1] Hanbay, K., Talu, M. F., & Özgüven, Ö. F. (2016). Fabric defect detection systems and methods — A systematic review. Optik, 127(24), 11960–11973. https://doi.org/10.1016/j.ijleo.2016.09.098

[2] Kumar, A. (2008). Computer-vision-based fabric defect detection: A survey. IEEE Transactions on Industrial Electronics, 55(1), 348–363. https://doi.org/10.1109/TIE.2007.896497

[3] Ngan, H. Y. T., Pang, G. K. H., & Yung, N. H. C. (2011). Automated fabric defect detection — A review. Image and Vision Computing, 29(7), 442–458. https://doi.org/10.1016/j.imavis.2011.02.003

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770–778). https://doi.org/10.1109/CVPR.2016.90

[5] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

[6] Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T., & Gehler, P. (2022). Towards total recall in industrial anomaly detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 14318–14328). https://doi.org/10.1109/CVPR52688.2022.01660

[7] Bergmann, P., Fauser, M., Sattlegger, D., & Steger, C. (2019). MVTec AD — A comprehensive real-world dataset for unsupervised anomaly detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 9592–9600). https://doi.org/10.1109/CVPR.2019.00395

[8] Defard, T., Setkov, A., Loesch, A., & Audigier, R. (2021). PaDiM: A patch distribution modeling framework for anomaly detection and localization. In Pattern Recognition. ICPR International Workshops and Challenges (pp. 475–489). Springer. https://doi.org/10.1007/978-3-030-68763-2_40

[9] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[10] Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). Computer Vision and Image Understanding, 110(3), 346–359. https://doi.org/10.1016/j.cviu.2007.09.014

[11] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2564–2571). https://doi.org/10.1109/ICCV.2011.6126544

[12] Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381–395. https://doi.org/10.1145/358669.358692

[13] Hartley, R., & Zisserman, A. (2004). Multiple view geometry in computer vision (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511811685

[14] Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(10), 1615–1630. https://doi.org/10.1109/TPAMI.2005.188

[15] DeTone, D., Malisiewicz, T., & Rabinovich, A. (2018). SuperPoint: Self-supervised interest point detection and description. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 224–236). https://doi.org/10.1109/CVPRW.2018.00060

[16] Sarlin, P.-E., DeTone, D., Malisiewicz, T., & Rabinovich, A. (2020). SuperGlue: Learning feature matching with graph neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4938–4947). https://doi.org/10.1109/CVPR42600.2020.00499

[17] Radenović, F., Tolias, G., & Chum, O. (2018). Fine-tuning CNN image retrieval with no human annotation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(7), 1655–1668. https://doi.org/10.1109/TPAMI.2018.2850773

[18] Brunelli, R. (2009). Template matching techniques in computer vision: Theory and practice. Wiley. https://doi.org/10.1002/9780470744055

[19] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations. https://arxiv.org/abs/1409.1556

[20] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q. V., & Adam, H. (2019). Searching for MobileNetV3. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1314–1324). https://doi.org/10.1109/ICCV.2019.00140

[21] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4510–4520). https://doi.org/10.1109/CVPR.2018.00474

[22] Koch, G., Zemel, R., & Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In Proceedings of the ICML Deep Learning Workshop. https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf

[23] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 815–823). https://doi.org/10.1109/CVPR.2015.7298682

[24] Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1735–1742). https://doi.org/10.1109/CVPR.2006.100

[25] Bradski, G. (2000). The OpenCV library. Dr. Dobb's Journal of Software Tools, 25(11), 120–125.

[26] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[27] Kong, R. W. M., Ning, D., & Kong, T. H. T. (2025). AI intelligent learning for manufacturing automation. International Journal of Mechanical and Industrial Technology, 13(1), 1–9. https://doi.org/10.5281/zenodo.15159741

[28] Kong, R. W. M., Ning, D., & Kong, T. H. T. (2026). AI visual inspection for garment production. International Journal of Computer Science and Information Technology Research, 14(3), 47–64. https://doi.org/10.5281/zenodo.21946142

[29] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2564–2571). https://doi.org/10.1109/ICCV.2011.6126544

[30] Kong, R. W. M., Liu, M., & Kong, T. H. T. (2024). Design and experimental study of vacuum suction grabbing technology to grasp fabric piece. Open Access Library Journal, 11, 1–17. https://doi.org/10.4236/oalib.1112292

[31] Kong, R. W. M., Kong, T. H. T., Yi, M., & Zhang, Z. (2024). Design a new pulling gear for the automated pant bottom hem sewing machine. International Research Journal of Modernization in Engineering Technology and Science, 6(11), 3067–3077. https://doi.org/10.56726/IRJMETS64156

[32] Kong, R. W. M. (2025). Innovative automated stretch elastic waistband sewing machine for garment manufacturing. International Research Journal of Modernization in Engineering Technology and Science, 7(3), 7347–7359. https://doi.org/10.56726/IRJMETS70275

[33] Kong, R. W. M. (2025). Automated seam folding and sewing machine on pleated pants for apparel manufacturing. International Journal of Mechanical and Industrial Technology, 13(1), 58–70. https://doi.org/10.5281/zenodo.16607787

[34] Kong, R. W. M. (2025). Innovative technology strategy in Hong Kong and China of the Asia industries. International Journal of Social Science and Humanities Research, 13(3), 103–116. https://doi.org/10.5281/zenodo.15911676

[35] Kong, R. W. M. (2025). Real world asset (RWA) supply chain for smart factory. International Journal of Management and Commerce Innovations, 13(1), 430–443. https://doi.org/10.5281/zenodo.16836199

[36] Kong, R. W. M. (2025). Woven air permeability textile fabric for garment automation. International Journal of Mechanical and Industrial Technology, 13(1), 31–45. https://doi.org/10.5281/zenodo.15806556

[37] Kong, R. W. M. (2026). Automated folding and sewing machine for the mid-leg seam pants. International Journal of Mechanical and Industrial Technology, 13(2), 67–78. https://doi.org/10.5281/zenodo.18169835

[38] Kong, R. W. M., Kong, T. H. T., & Huang, T. (2024). Lean methodology for garment modernization. International Journal of Engineering Development and Research, 12(4), 14–29. https://doi.org/10.1729/Journal.41971

[39] Kong, R. W. M., Ning, D., & Kong, T. H. T. (2025). Mixed-integer linear programming (MILP) for garment line balancing. International Journal of Scientific Research and Modern Technology, 4(2), 64–77. https://doi.org/10.5281/zenodo.14942910

[40] Kong, R. W. M., Ning, D., & Kong, T. H. T. (2025). Innovative line balancing for the aluminium melting process. International Journal of Mechanical and Industrial Technology, 12(2), 73–84. https://doi.org/10.5281/zenodo. 15050721

[41] Kong, R. W. M., Ning, D., & Kong, T. H. T. (2025). Line balancing in the modern garment industry. International Journal of Mechanical and Industrial Technology, 12(2), 60–72. https://doi.org/10.5281/zenodo.14800724

[42] Kong, R. W. M. (2025). AI magnetic levitation (Maglev) conveyor for automated assembly production. International Journal of Mechanical and Industrial Technology, 13(1), 19–30. https://doi.org/10.5281/zenodo.15599657

[43] Han, J., Cao, J., Wang, H., & Ji, X. (2022). A review of fabric defect detection methods based on computer vision. Journal of Liaoning Petrochemical University, 42(1), 78–85. https://doi.org/10.3969/j.issn.1004-0935.2022.01.013

[44] Mur-Artal, R., & Tardós, J. D. (2017). ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras. IEEE Transactions on Robotics, 33(5), 1255–1262. https://doi.org/10.1109/TRO.2017.2705103

[45] Ning, D., Kong, R. W. M., & Feng, S. P. (2026). A new DCache prefetch algorithm based on branch prediction information for AI industrial engineering. Academia Engineering, 3(2). https://doi.org/10.20935/AcadEng8332

[46] Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381–395. https://doi.org/10.1145/ 358669.358692

[47] Hartley, R., & Zisserman, A. (2004). Multiple view geometry in computer vision (2nd ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511811685

[48] Zhong, Q., Zeng, X., Groper, M., Tan, D. Q., Kong, R. W. M., & Liu, M. (2026). A data-driven dynamic modeling of fish-like robot enabled by artificial neural network. Ocean Engineering, 348, Article 124069. https://doi.org/10.1016/ j.oceaneng.2025.124069

[49] Kong, R. W. M. (2025). Automated RFID detection cabinet for garment manufacturing. International Research Journal of Modernization in Engineering Technology and Science, 7(9), 3402–3412. https://doi.org/10.56726/ IRJMETS83210

[50] Li, S., Dong, A., Kong, R. W. M., & Liu, C. (2026). A physics-informed embodied intelligence framework for smart garment manufacturing. Journal of Manufacturing Systems, 85, 307–317. https://doi.org/10.1016/j.jmsy.2026.01.015

[51] Kong, R. W. M., Ning, D., & Kong, T. H. T. (2025). Innovative vacuum suction-grabbing technology for garment automation. In K. M. Batoo (Ed.), Science and technology: Developments and applications (Vol. 6, pp. 148–170). BP International. https://doi.org/10.9734/bpi/stda/v6/4600

[52] Kong, R. W. M., Ning, D., & Kong, T. H. T. (2026). AI logo inspection system for inspecting the logo shape in quality control. International Research Journal of Modernization in Engineering Technology and Science, 8(9), 129–138. https://doi.org/10.56726/IRJMETS102581