POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Machine Vision for Symmetry Verification in Cardboard Manufacturing Lines

Laurea Magistrale in Automation and Control Engineering - Ingegneria Dell'Automazione

Author: Massimiliano Migliorini, Cristiano Sartini, Pietro Ughini

Advisor: Prof. Marco Tarabini

Co-advisor: Luca Pini

Academic year: 2024-2025

Contents

1. Introduction

1.1. Objective and Evaluation Criteria

The goal of our system is to assess the geometric symmetry of closed-boundary shapes on the surface of a product. This objective is designed to be scalable across different product types, enabled by a high-level configuration interface that allows easy adaptation by the target company. A key principle in our approach is the use of a **majority-vote criterion** among extracted features. For instance, if four symmetric features are identified and three are in mutual agreement while one differs significantly, the system assumes the majority to be correct. This avoids overreacting to isolated outliers and avoids relying on absolute geometric thresholds, which would reduce the generality and scalability of the algorithm.
Feature comparison is thus performed through **relative matching** within clusters of similarity, rather than against fixed reference values. This approach can be applied to various products and requires little manual adjustment.
Beyond scalability, our design also prioritizes **robustness** to environmental variability in production lines. Both the hardware and the software components are engineered to tolerate fluctuations in lighting, positioning, and surface conditions, as detailed later in the report.

Working Assumptions

To ensure reliable inspection in a production context, we base our system design on the following assumptions:

- The main conveyor belt moves at approximately **1 m/s**. We inspect **one piece out of ten** for the following reasons:
  1. Due to lighting and contrast requirements, we cannot afford a low exposure time. Hence, image acquisition must occur with the item **at rest**. We assume that approximately **10 seconds** are required to divert a piece onto a secondary belt, decelerate it, perform the vision-based inference, and either discard it or return it to the main line.
  2. The production process is standardized and consistent, so a **sampling-based quality control** is sufficient.
  3. At the end of the production cycle, the process becomes **discrete** rather than continuous, which simplifies the tracking of inspected items, whether accepted or rejected.
- The **machine vision system** is implemented on a **secondary conveyor belt**. Stopping the main conveyor would reduce throughput and place unnecessary demands on the actuators. The use of a secondary belt allows controlled deceleration and accurate positioning without compromising the efficiency of the main line.

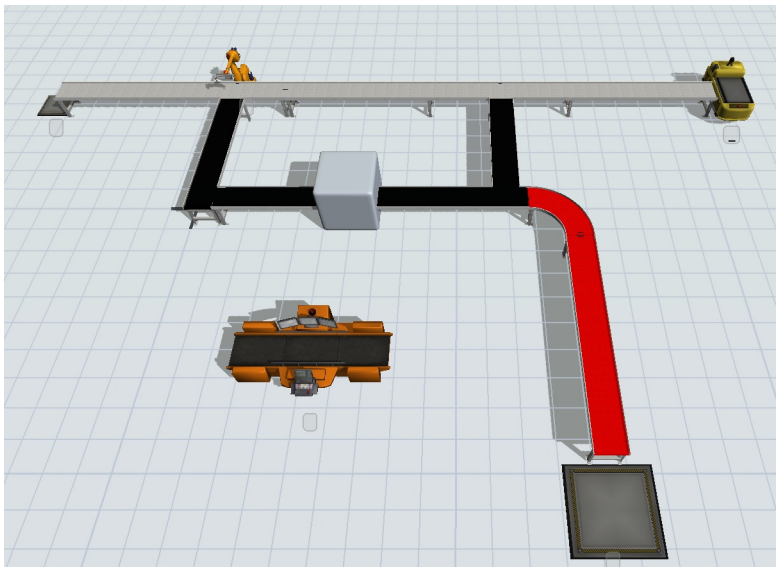


Figure 1: Simulation of the machine vision system setup

2. Image Acquisition

The machine vision setup has to provide a very high quality image, creating the highest level of **contrast** between features of interest and the surrounding background; with the greatest possible **resolution**, accordingly to the target resolution provided as input by client side. We aim to follow these two objectives by correctly defining these three integrated parts of the machine vision setup:

- **Sensor**
- **Lens**
- **Lighting System**

2.1 Setup

First, we select the camera position relative to the cardboard position.

- **Working Distance:** We chose the distance between the cardboard and the lens by balancing industrial requirements and laboratory constraints:
  - The mounting column has a maximum height of 104 cm, which sets the upper limit on WD.
  - A minimum safe distance is always required to avoid damaging the machine vision system.
  - Short WDs force the use of short focal length lenses (e.g. 6-8 mm), which are more expensive and prone to barrel distortion. Correcting for this in preprocessing would increase the calibration complexity and may not fully compensate.
  - Therefore, we set the working distance to the maximum of the column: $WD = 104\,\text{cm}$.

- **Camera Orientation:** We aligned the optical axis perpendicular to the cardboard surface to eliminate perspective distortion and simplify calibration.

- **Background:** A dark blue paper sheet was placed behind the cardboard to simulate the conveyor belt as background, avoiding pure black to better reflect real world lighting.

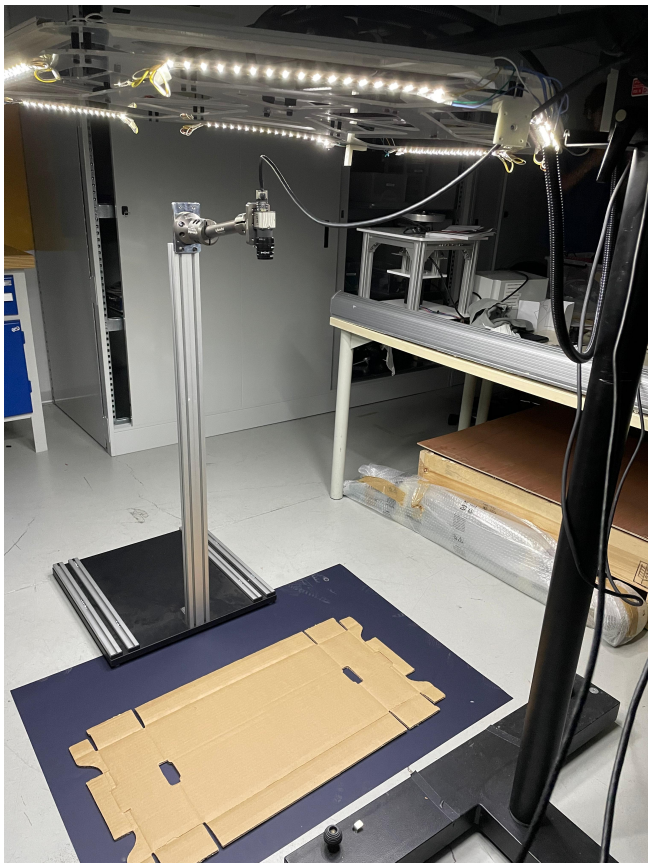


Figure 2: Laboratory image acquisition system.

2.0.1 Sensor

We have selected the *IDS UI-3360CP-M-GL* monochrome **CMOS** sensor with a global shutter and a **resolution** of **2048 × 1048 pixels**.

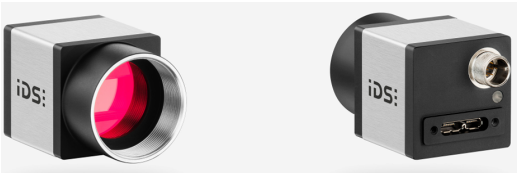


Figure 3: Sensor.

First of all, this choice supports our vision inspection task on several levels:

- **Feature contrast:** we verify symmetry of slots, holes, and corner geometries on a uniform, monochromatic surface, where contrast arises from intensity gradients rather than color differences.
- **Algorithm compatibility:** Capturing directly in grayscale aligns better with our image processing pipeline — axes of symmetry search, edge based segmentation, contour detection — avoiding the three RGB channels conversion, and thus reducing the computational effort of all the algorithms involved .
- **Effective resolution:** At the same sensor size, a monochrome CMOS sensor has an higher effective pixel density, with respect a Bayer-filtered RGB version.
- **Motion robustness:** The global shutter, unlike the rolling shutter, improves image fidelity under dynamic conditions, such as vibrations of the quality control station.

Sensor type	CMOS Mono
Shutter	Global Shutter
Sensor characteristic	Linear
Readout mode	Progressive scan
Pixel Class	2 MP
Resolution	2.23 Mpix
Resolution (h x v)	2048 x 1088 Pixel
Aspect ratio	17:9
ADC	12 bit
Color depth (camera)	12 bit
Optical sensor class	2/3""
Optical Size	11.264 mm x 5.984 mm
Optical sensor diagonal	12.75 mm (1/1.25")
Pixel size	5.5 µm
Manufacturer	ams/CMOSIS
Sensor Model	CMV2000-3E5M
Gain (master/RGB)	4x/-

Figure 4: Sensor datasheet provided by the manufacturer.

Furthermore, to meet our machine vision system's **resolution** requirements, we must perform an in-depth technical evaluation of the sensor parameters.

To cover all cardboard variants up to the largest sample plus some tolerance margin, we set our **field of view** to:

$$\mathrm{FOV}_x = 976\,\mathrm{mm} \quad \mathrm{FOV}_y = 519\,\mathrm{mm}.$$

This choice aligns with our scalability metric, enabling inspections across every cardboard size up to the maximum sample, and the extra FOV margin accommodates random rotations on the conveyor belt, which we have tried to simulate during the image dataset creation in the laboratory. The margin can't be too large, since it would deteriorate the cardboard imaging accuracy

The FOV specification and the **imaging accuracy** required by the company are therefore the key factors in verifying our sensor's **spatial resolution**. We expect the client to define the imaging accuracy as the size of the smallest symmetric feature to be detected. Once this minimum feature size $f_{\min}$ is established, accordingly to the table in *figure 4*, we require at least 5 pixels to cover this minimum size, in order to distinguish this feature from its symmetric counterparts.

Inspection	Pixels required (usually)
Defect detection	Min. 2x2
Feature location	Min. 3x3
Feature differentiation	Min. 5x5
Gauging	Sub-pixel resolution must be 1/10th the desired tolerance

Figure 5: Pixels required depends on the application

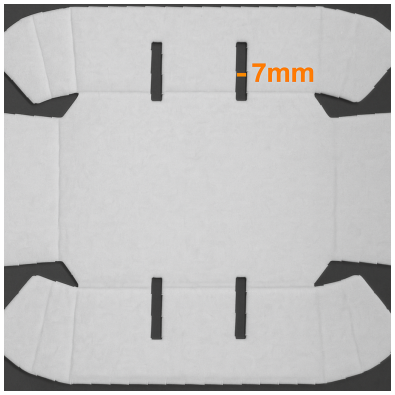


Figure 6: Smallest measured feature

Our measurements show a minimum feature width of 7 mm, yielding a maximum allowable pixel pitch of:

$$\mathrm{SR}_{\max} = \frac{7\,\mathrm{mm}}{5\,\mathrm{px}} = 1.4\,\mathrm{mm/px}.$$

Given our selected FOV and sensor resolution (2048×1088 px), the actual spatial resolution along each axis is:

$$\mathrm{SR}_x = \frac{\mathrm{FOV}_x}{2048} = \frac{976\,\mathrm{mm}}{2048\,\mathrm{px}} \approx 0.477\,\mathrm{mm/px}, \quad \mathrm{SR}_y = \frac{\mathrm{FOV}_y}{1088} = \frac{519\,\mathrm{mm}}{1088\,\mathrm{px}} \approx 0.477\,\mathrm{mm/px}.$$

$$\mathrm{SR}_x,\ \mathrm{SR}_y \ \leq\ \mathrm{SR}_{max} \quad \checkmark$$

2.0.2 Lens and Camera Parameters Tuning

From the available focal lengths (8 mm, 12 mm, 16 mm), we have chosen a 12 mm lens as the best compromise:

- **8 mm** is too extreme wide-angle, introducing excessive barrel distortion that cannot be fully corrected in calibration.
- **16 mm**, at our working distance (WD $\approx$ 1 m) and desired FOV, it could miss full coverage of the cardboard. Increasing WD to compensate would make the mount more prone to vibration.

Given our working parameters, we can verify their compatibility with $\boldsymbol{f = 12\,\mathrm{mm}}$, by using:

- **Thin-Lens (Pinhole) model:** a first rough approximation

$$\frac{\mathrm{FOV}_x}{\mathrm{WD}} = \frac{S_x}{f} \quad \Longrightarrow \quad f_x = \frac{\mathrm{WD}\,S_x}{\mathrm{FOV}_x} = \frac{1040\,\mathrm{mm} \times 11.264\,\mathrm{mm}}{976\,\mathrm{mm}} \approx 12.0\,\mathrm{mm}, \quad \checkmark$$

$$\frac{\mathrm{FOV}_y}{\mathrm{WD}} = \frac{S_y}{f} \quad \Longrightarrow \quad f_y = \frac{1040\,\mathrm{mm} \times 5.984\,\mathrm{mm}}{519\,\mathrm{mm}} \approx 12.0\,\mathrm{mm}. \quad \checkmark$$

- **Optics calculator:** *pointsinfocus.com*

Depth of Field (DoF): by fixing the focus plane to the conveyor-belt surface, the resulting depth of field encompasses any stacked cardboard. This ensures sharp images at inference time without requiring autofocus.
Parameter Tuning Pipeline: the following pipeline has inspired our tuning of the remaining camera parameters under these assumptions:

- Images are captured with the cardboard at rest, located on the quality control station.

- All of our algorithms rely on intensity gradients at the slot edges against the dark background, so their effectiveness depends on the sharpness of that contrast. Consequently, we adopted a processing pipeline that maximizes the achievable brightness contrast between the features and the background, while respecting the system's depth-of-field constraints.

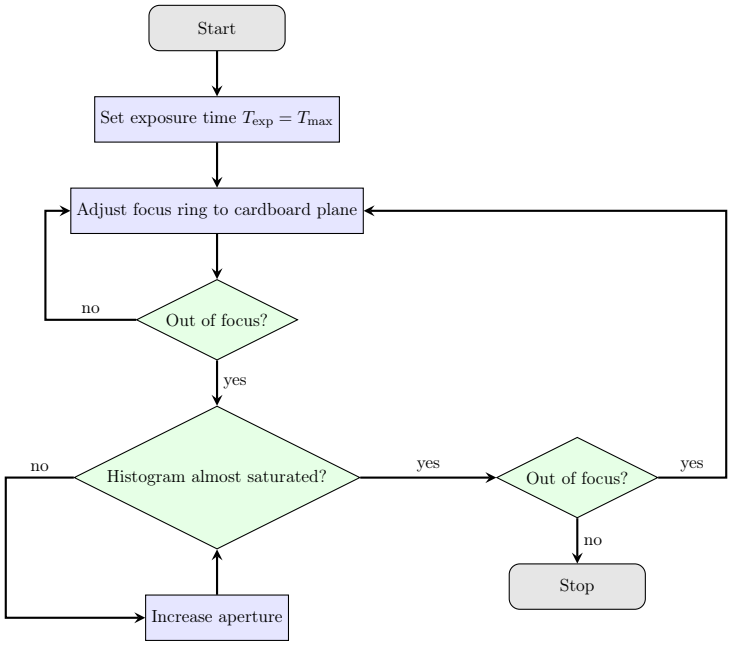


Figure 7: Flowchart for exposure, focus and aperture tuning

The pipeline implied by the flow chart follows these steps:

1. **Set exposure time:** fix the exposure to the sensor's maximum allowed value $T_{exp} = 500\,\text{ms}$, in order to maximize image brightness.

2. **Set focus plane:** manually focus the lens on the cardboard plane, achieving a maximum usable focal distance equals to the WD.

3. **Adjust aperture:** gradually increase the diaphragm opening until at least one of the following conditions are met:

   1) The grayscale histogram is nearly saturated at the white end.
   2) Further aperture increase would reduce depth of field below acceptable limits.

   - If condition (1) is reached but (2) is not, stop increasing the aperture.
   - If condition (2) is reached, readjust the focus plane once more, then stop.

Filter

The sensor's infrared sensitivity tail —visible in the quantum-efficiency curve of the datasheet—does not provide useful signal for distinguishing the cardboard from the background. Infrared reflectivity depends on the material composition, and since our cardboard and background are made of very similar materials, capturing IR wavelengths would only degrade the sensor's signal-to-noise ratio.

Therefore, we installed an IR-block filter (700 nm low-pass) between the sensor and the lens.

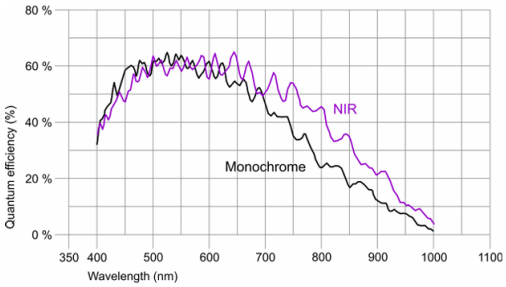


Figure 8: Quantum Efficiency

Figure 9: IR blocking filter

2.0.3 Illumination Technique

The objective of illumination is to maximize the brightness contrast between the cardboard and the background, to achieve sharp slot–edge detection, while ensuring uniform illumination across the entire cardboard surface.

Back-lighting analysis

We first evaluated a back-lighting setup, which produces strong silhouettes against a bright background. However, in our case contrast is already sufficient due to the difference in reflectivity between the beige cardboard and the opaque conveyor belt. Moreover, back-lighting would complicate installation on the production line, as it requires a transparent illuminated conveyor.

Partial bright-field analysis

Next, we considered partial bright-field or directional lighting using a linear LED array mounted above the cardboard. In the laboratory, the available LED strip was too short to illuminate the full field of view uniformly.

Figure 10: Ring Light

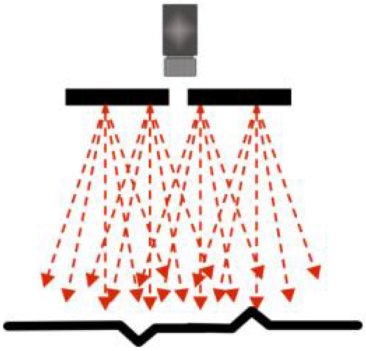


Figure 11: Flat diffuse implementation

Diffuse flat lighting solution

Finally, we adopted the laboratory's LED ring light (Figure 9), not for the purpose of dark-field illumination, but to create a diffuse flat-lighting effect (Figure 10); hence by positioning it above the camera. Care must be taken to mount the ring light at a sufficient distance from the cardboard; if placed too close, low-angle reflections accentuate surface defects (bubbles, scratches) in a dark-field manner, making it harder for our algorithms to distinguish true slot edges from unwanted artifacts.

3. Preprocessing

After selecting the most suitable hardware for our application and working conditions (a form of hardware preprocessing aimed at minimizing the need for software corrections), we focused on software preprocessing. The goal was, first, to make our images robust to variations from ideal conditions, and second, to make them as suitable as possible for our main algorithm.

To ensure metrological accuracy in image-based measurements, the camera was calibrated using a checkerboard with known dimensions (15 mm square sides) (*CameraCalibration.py*). Using the OpenCV library, the internal corners of the pattern were automatically detected in multiple images acquired from different angles. This process yielded the intrinsic parameters of the camera (`mtx`) and the optical distortion coefficients (`dist`), which were then saved in a `calib.npz` file for later import and use. These parameters were subsequently used to remove geometric distortion from new images (`_undistort_image`), improving the precision of the measurements.

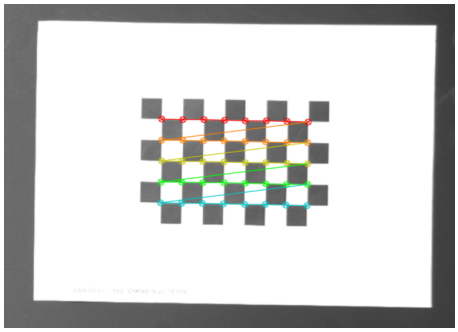


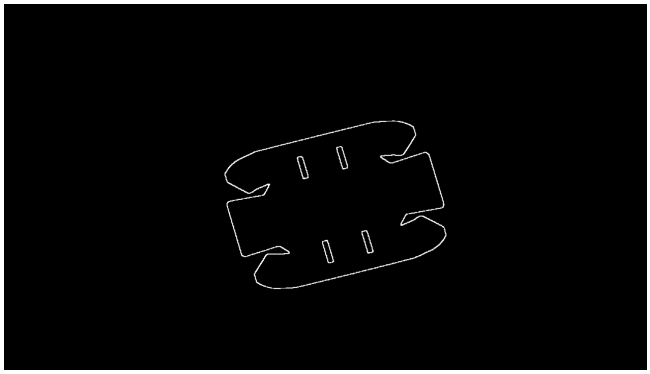
Figure 12: Chessboard detected

3.1. Effective preprocessing steps

Following the camera calibration, it was necessary to split the preprocessing into two components, as our algorithm is divided into two main procedures: one for processing internal features using rectangles, and another for also handling external features through the moments-based technique (which will be introduced later in the report).

3.1.1 Preprocessing for symmetry_via_rectangles

1. **Preprocessing: Gaussian blur, Canny, and morphology**



A Gaussian blur is applied to reduce noise, followed by edge detection using the Canny operator (*pre_processing*). Threshold and kernel parameters are loaded from a *config.yaml* file to facilitate adaptation to different images. After Canny, a dilation and a morphological closing are performed to consolidate the edges and close any discontinuities.

Figure 13: Original image binarized with Canny

2. **Alignment and cropping** The binary image is used to estimate the orientation of the main object through the minimum bounding rectangle of the external contour (*_external_contour*). The image is then rotated to align the object horizontally (*alignment_cropping*), and a crop centered on the object is performed. The result is an aligned binary image cropped on the cardboard, and a grayscale image, ready for further analysis.

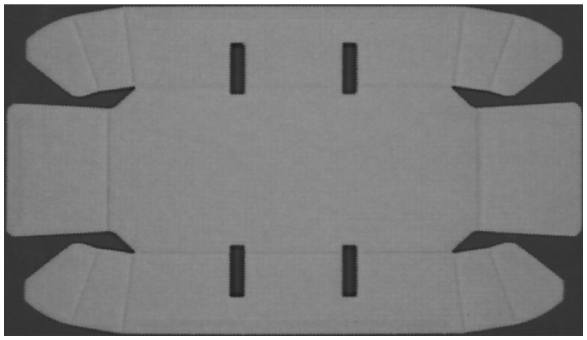


Figure 14: gray image aligned and cropped

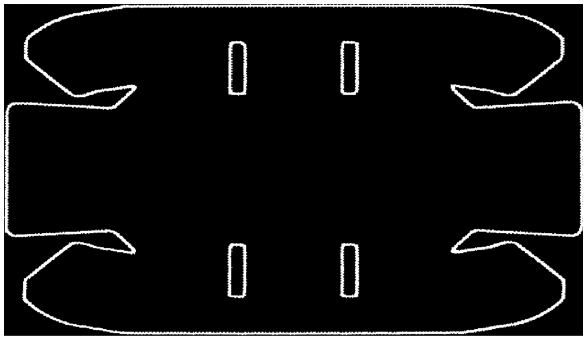


Figure 15: binary image aligned and cropped

3. **Symmetry axis detection**

Starting from the aligned binary image, two symmetry axes are computed: one along the main orientation of the object and one perpendicular to it (*axes_of_symmetry*). The coordinates of the center are saved in a CSV file for further analysis within the algorithm.

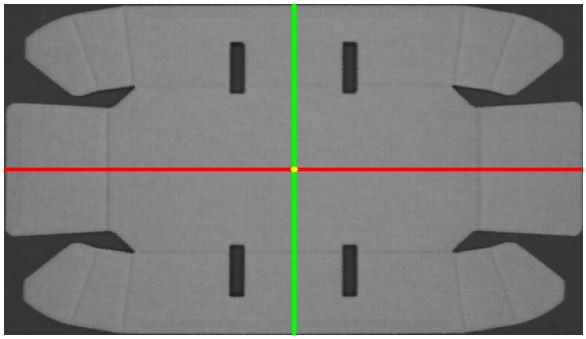


Figure 16: Gray image with axis of symmetry

3.1.2 Preprocessing for symmetry_via_Moments

In the case where external features of the cardboard are also to be processed using the moments and centroid-based technique, it was necessary to replicate the preprocessing steps previously introduced (already explained above and therefore not repeated here), with the addition of one specific phase: the *convex_hull*.

1. **Preprocessing: Gaussian blur, Canny, and morphology** As previously introduced, this step is used for contour extraction.
2. **convex hull**

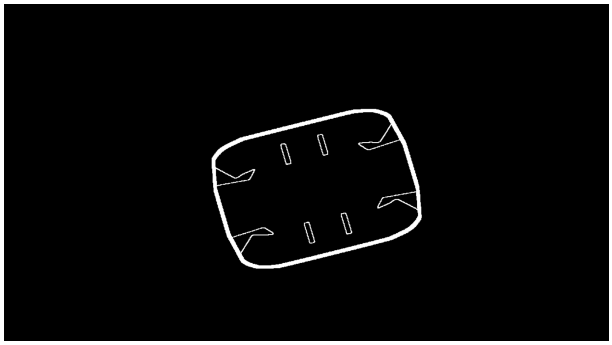


Figure 17: Convex hull

In the variant using moment-based analysis, a convex hull step was added to the preprocessing to handle external features. These features, located along the cardboard edge, appear as open shapes and cannot be detected as valid contours. The convex hull ensures that the entire external region is uniformly enclosed, allowing these shapes to be closed and processed like internal slots. Importantly, since the hull wraps the object symmetrically on all sides, it avoids introducing artificial asymmetries when none are actually present.

   To address this issue, the convex hull was applied to the main contour (*Convex_Hull*), which is the convex envelope that completely encloses the largest detected shape. Specifically:
   - The convex hull of the largest contour in the image is computed.
   - This contour is drawn (*cv2.drawContours*) with a certain thickness to enforce a geometric closure of the external region.
   - A mask of the convex region is created, and a morphological opening is applied only along a border band, obtained as the difference between the original mask and an internal erosion of it (*cv2.erode*).

   The morphological opening improves the uniformity of the borders and reduces false positives caused by noise and small details along the perimeter.
   This procedure ensures a well-defined and closed external region, enabling the detection of open features along the cardboard contour and their comparison.
3. **Alignment and cropping** As before, this step is used to process the rotated and cropped image.
4. **Symmetry axis detection** As before, this step extracts the symmetry axes and the center, which will be used in the core of the algorithm.

3.2. choice of Canny parameters

We begin by motivating the choice of using the Canny operator for contour extraction. To identify the features, we decided to rely on edge detection (a logic that will be explained later), and for this purpose, we selected the Canny algorithm. Canny was ideal for our application because it leverages the image gradient and offers high flexibility through four tunable parameters—blur kernel size, Gaussian sigma, threshold1, and threshold2—making it well-suited both for accurate edge detection and for our goal of scalability.
As previously mentioned, the effectiveness of Canny in our pipeline relies on careful tuning of these four key parameters:

- **Blur kernel**:

This parameter defines the kernel size used in the initial Gaussian blurring step (convolution with a gaussian filter), which helps in reducing noise before computing the gradient. The purpose of this blur is to prevent the algorithm from detecting false edges caused by image noise. Typical values are 3×3, 5×5, or 7×7.

Larger kernels like 7×7 or above are usually used for ultra-high-resolution images (e.g., 4K or 8K), as smaller kernels in such cases may retain too many minor edges or noise. On the other hand, a 3×3 kernel is suitable for small or very sharp images but may lead to excessive noise sensitivity, detecting edges where there are none.

For these reasons, we chose a 5×5 blur kernel as a balanced solution for our image resolution. It avoids over-detecting fine details (and potential noise), while still capturing the essential contours required by our algorithm—even under lighting disturbances or small imperfections in the hole-cutting process, as further discussed in the *Flexibility* paragraph.

- **Gaussian sigma**:
  In Canny, σ (the standard deviation of the Gaussian) controls the amount of blur applied during the initial smoothing step. A larger sigma results in a wider Gaussian curve, meaning intensity values are averaged over a broader neighborhood—this reduces noise more effectively but also causes loss of fine details.
  With a 5×5 kernel, a typical sigma value is around 1 when the image has moderate noise and a medium resolution, as in our case. This value preserves a sufficient level of detail for our algorithm while still reducing spurious edges. For these reasons, we selected $\sigma = 1$ which is also a commonly adopted default value in computer vision applications.
- **TH1 E TH2**:
  The two thresholds play a fundamental role in determining which edges to retain and which to discard. Threshold1 (low threshold) sets the lower bound: if the gradient magnitude is below this value, the pixel is discarded. If it falls between Threshold1 and Threshold2, the pixel is retained only if it is connected to a strong edge.
  Threshold2 (high threshold) sets the upper bound: if the gradient exceeds this value, the pixel is considered a strong edge with high confidence.
  Among the two, Threshold2 was the most critical parameter to tune. If set too high, important edges may be missed—for instance, the contours of larger cardboard regions or the edges of smaller slots, especially under non-ideal lighting conditions. Conversely, if set too low, there is a higher risk of detecting unwanted details, such as noise, cardboard cuts, or irrelevant thickness variations.
  To address this, we chose a value for Threshold2 that was low enough to ensure the detection of key edges and features essential to our project—even in suboptimal lighting—but not so low as to capture irrelevant noise or unwanted patterns. The final choice was Threshold2 = 118. Following the common thumb rule of setting Threshold1 between one-half and one-third of Threshold2, we selected Threshold1 = 50.

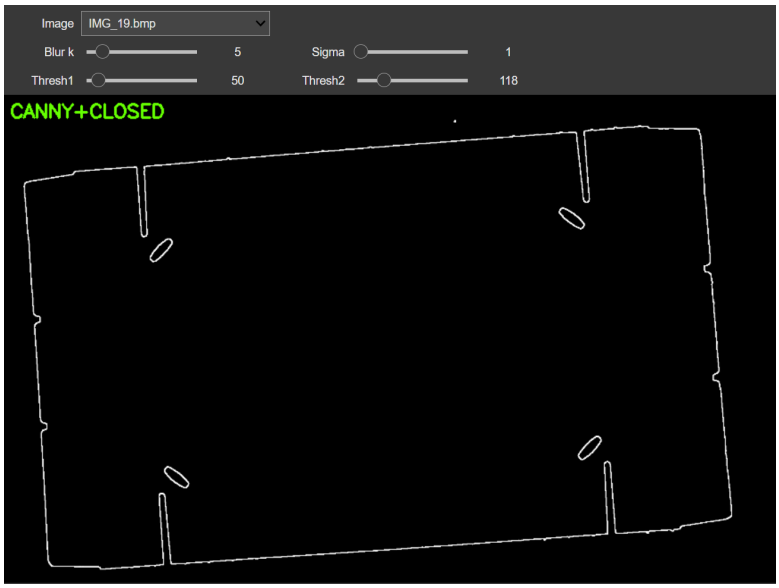


Figure 18: Temporary script done for canny parmeters tuning

3.3. flexibility and scalability

The choice of the parameters mentioned above is obviously crucial for our algorithm, which relies on feature detection through contour extraction. However, this choice may not generalize well when testing new images

under slightly different conditions, potentially making the algorithm less scalable and not robust to deviations from ideal scenarios.
To address these issues and increase the flexibility and robustness of our algorithm, we focused on two specific types of problems that we encountered during the development phase:

- Lighting disturbances with respect to the ideal setup We assume that the cardboard enters a closed imaging cabinet designed to block ambient light. When both curtains are lowered, optimal dark conditions are achieved, allowing the ring light to ensure high-contrast, uniform illumination. If one curtain is left open (e.g., partially closed entry), directional ambient light may disturb the scene. With both curtains open or missing, external light enters fully, simulating enclosure failure. These cases are used to evaluate system robustness under degraded lighting.
  To simulate and ensure robustness to such scenarios, we conducted several tests and acquisitions under different controlled lighting conditions in the lab, including:
  - complete darkness with both blinds closed (ideal setup with ring light only);
  - Full ambient light with both blinds open;
  - Grazing light from one side, with the nearest blind open and the far one closed;
  - Grazing light from the opposite side, with the nearest blind closed and the far one open.

Figure 19: acquisition in dark conditions

Figure 20: acquisition with both tents up

The selection of our Canny parameters was also guided by these tests, in order to ensure robustness under lighting conditions different from the optimal setup. We verified that, in each of the tested scenarios, the relevant edges were still correctly detected with our thresholds, allowing the algorithm to produce the desired output consistently.
This issue mainly resulted in defining an upper bound for Threshold 2, to avoid missing important contours under non-ideal lighting.

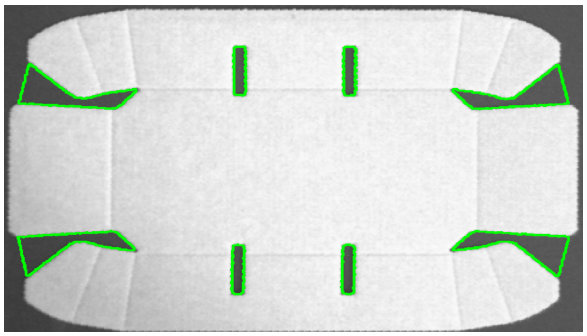


Figure 21: output with both tents up

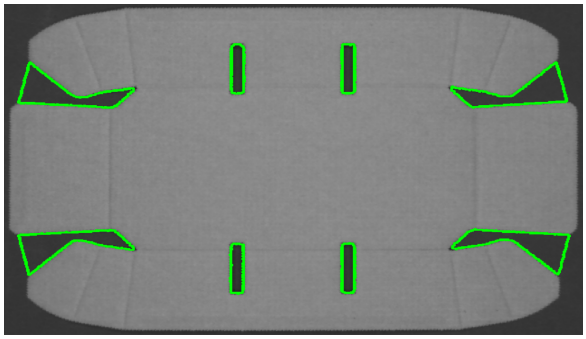


Figure 22: output with lights off

- To detect both external contours and less prominent edges, essential for feature detection in various lighting conditions and across different cardboard sizes, we were forced to choose a relatively low Threshold2 in the Canny algorithm. Lowering this threshold, however, introduced issues due to the detection of defective or uncut slots (which we simulated), which we actually intended to ignore within the logic of our algorithm.
  To address this problem, we implemented a filter that allows us to discard detected contours, which enclose defects (scratches or grooves) rather than holes. The filter works by computing the mean intensity of the pixels inside each contour and comparing it to a defined threshold ($INTENSITY\_HOLE\_MAX$). If the intensity exceeds this threshold, the feature is considered too bright to represent a real hole and is thus discarded.
  Thanks to this solution, we were able to further reduce Threshold 2 to detect as many relevant contours as possible, without being affected by noise or undesired cardboard imperfections, making our algorithm significantly more flexible in non-ideal conditions and more robust to manufacturing defects.
  In practice, the code applies the following logic to each candidate feature: a binary mask is created by filling the corresponding contour. The mean intensity inside this mask is then computed (on the grayscale image). If the result exceeds the maximum threshold ($INTENSITY\_HOLE\_MAX\_R$), the feature is discarded because it is too bright to be a true hole.

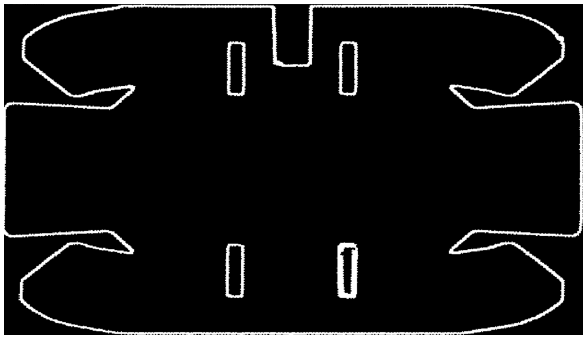


Figure 23: edge detection without defects filtering

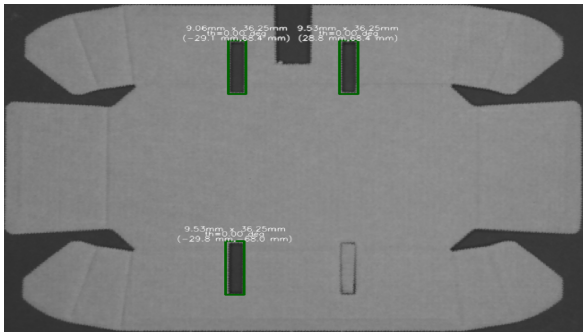


Figure 24: the defect is filtered, avoiding its detection as a slot

4. Feature detection algorithm

4.1. A Brief Overview of the symmetric Feature Detection Framework

Starting from the results obtained from the previous pre-processing step, we now introduce the "core component" of this project: the feature symmetry detection algorithm. Two separate algorithms for feature detection and symmetry analysis have been implemented. The first one is dedicated exclusively to measuring and classifying **internal features** ("slots") using **rectangular bounding boxes**, which, as we will see, enable a characterization that is both easily interpretable and fully compliant with company requirements. The second algorithm is instead designed to validate the conformity of all features on the cardboard surface, including **both internal and external ones** ("slots" and "cutouts"), by identifying their exact closed contours and subsequently classifying them based on the calculation of **moments and centroids**.
This dual comparison applied to internal features is far from redundant. On the contrary, it provides a valuable cross-validation of the results and allows a more conservative approach in detecting slots symmetry, which remains the central focus of this project. Moreover, it represents an effective strategy to combine interpretability (through "physical" thresholds applied to orientation and size of the rectangular bounding boxes) and precision (achieved by accurately tracing the contours).

4.2. Scalability, Flexibility, and Key Benefits

Scalability, flexibility, and ease of implementation are surely the main strengths of this algorithm. In fact, it operates without requiring any large dataset of images for feature detection (as would have been necessary in an AI-based approach with neural networks), and it works entirely in "run-time" conditions.
In measurement and feature identification systems, it is often necessary to provide the operator in charge of

supervision with a technical drawing that includes all the geometric specifications of the object, such as the shape and size of the holes, as well as their expected position on the cardboard. This reference can be useful for algorithms to compare the expected geometry with the one detected by the vision system.
However, the use of a so-called *"Golden Sample"* would have completely compromised the generality and scalability of our algorithm, making it necessary to integrate any changes (even minimal ones) with a new technical drawing to be provided and digitized.
To avoid this, we exploited a fundamental characteristic of the cardboard: its **symmetry** with respect to the axes. Leveraging this intuition, we were able, as will be detailed later in the two algorithms, to rely exclusively on **relative measurements**, thereby eliminating the need for any prior knowledge of the cardboard under analysis and maximizing the scalability of the approach.

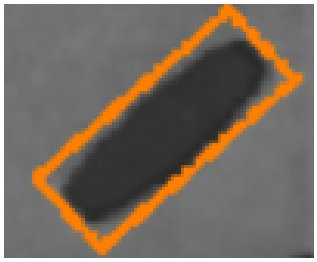
Note on the reliability of the measurements displayed on screen: Since this project is focused primarily on verifying the symmetry of the features and NOT on their precise measurement, the values corresponding to the positions of the centroids, dimensions, and orientations displayed on the screen should not be regarded as fully reliable. In addition to the inherent uncertainty of the image acquisition process itself, there is also uncertainty associated with contour tracking and the construction of the minimum-area rectangles, which is intrinsic in the definition of the algorithm.
It is important to emphasize that, within the framework of this project, performing a detailed estimation of measurement uncertainty (for example, calculating the sample mean and variance of the software measurements and comparing them with empirical results obtained using calipers) would have been inconsistent with our scalability objectives. Moreover, such uncertainty analysis would have offered limited practical benefit, because in symmetry verification we are essentially comparing relative differences between measurements extracted from the **same image**.
This means that any **systematic measurement error** introduced by the algorithm—such as a consistent overestimation or underestimation of dimensions—would very likely affect all measurements in the same way, resulting in errors that effectively cancel each other out when computing the deltas. In other words, even if the algorithm measures absolute sizes inaccurately, it remains capable of reliably identifying non-symmetric features because the same bias is applied uniformly across all elements.
Consequently, the absolute measurements obtained should be considered, also thanks to the camera calibration process, as "good approximations" of the real values. In conclusion, for the purposes of this project, we determined it was more valuable to focus on **flexibility** in terms of robustness to variations in the positioning of the cardboard on the conveyor belt and changes in ambient lighting conditions (as discussed in *Chapter about Flexibility*)

4.3. Internal Features Detection with Minimal-Area Rectangles

As previously anticipated, the purpose of this first algorithm is to identify, measure, and classify the internal features present on the cardboard, providing as output both a graphical visualization of the detection results and an acoustic signal to alert operators of any defective pieces on the line. A piece is considered defective whenever at least one of its features is classified as non-symmetric.
The inputs required by this algorithm are:

- **The binarized image**, containing all detected contours (as already specified in *Chapter about Preprocessing*
- **The spatial coordinates of the image symmetry axes**
- **The company-defined tolerances**, specifying the precision requirements to be applied

4.3.1 Minimal-Area Rectangle Extraction



The binarized image is first processed using the *cv2.findContours()* function, which extracts the boundaries of all closed contours as a vector of points.
Then, for each detected contour, the *cv2.minAreaRect()* function is applied. This function returns:

- **(cx, cy)**: the coordinates of the rectangle center
- **(w, h, theta)**: respectively, the width, height, and orientation (relative to the horizontal axis, positive in the clockwise direction)

The use of rectangles as bounding boxes has proven particularly effective for processing internal features across all the cardboard samples analyzed, due to their close resemblance to the typical shape of cardboard slots. Finally, by applying technical limits and thresholds *(see next paragraph)*, we were able to obtain a dictionary of internal features (specifically, the "slots") uniquely identified by the five parameters listed above, which will serve as the basis for their subsequent classification.

4.3.2 Technical Thresholds and Company Tolerances

To effectively filter the internal slots, it was necessary to define a set of technical thresholds capable of distinguishing them from external contours (*ASPECT_SLOT_MAX* and *AREA_SLOT_MAX_R*) as well as from nearly imperceptible holes or software artifacts (*ASPECT_SLOT_MIN* and *AREA_SLOT_MIN_R*).
Since the algorithm occasionally generated overlapping or duplicate features, an additional threshold on the distance between the centroids of already accepted slots (*TOL_RADIUS*) was introduced to discard such duplicates.
The last technical threshold, *INTENSITY_HOLE_MAX_R*, has already been extensively discussed in the *Chapter about Pre-processing.*

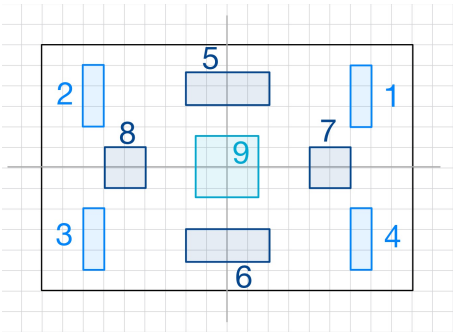
The main interface between the company's quality objectives and the vision software relies on just three parameters:

- ***position_mm:*** the maximum allowed difference in the centroid positions of two slots, used to limit misalignments (in [mm])
- ***length_mm:*** the maximum allowed difference in geometric dimensions (both length and height) between features with symmetric centroids, used to prevent production defects (in [mm])
- ***theta_degree:*** the maximum allowed difference in orientation between features with symmetric centroids, also used to prevent production defects (in [deg])

The use of these three indicators, each with a well-defined physical meaning, makes the algorithm easily understandable and actionable by the plant supervisor, in line with the scalability, interpretability, and flexibility objectives that this project aims to guarantee.

4.3.3 Sector Classification, Fuzzy Symmetric Matching, and Status Evaluation per Slot

Based on the information about the centroid positions and the dimensions of the rectangles (projected along the two symmetry axes of the image through the angle *theta*), the cardboard is divided into **9 regions of interest (or "sectors")**, as illustrated in the figure below.



This division is necessary to define the **expected number of symmetric copies**: if a slot lies on both symmetry axes, it will not have any symmetric counterparts; if it lies on only one symmetry axis, it is expected to have a single symmetric equivalent with respect to the other axis; if it is located off both axes, three symmetric counterparts are expected.

Once the 9 clusters containing the features (grouped by sector) have been created, they are iterated sequentially in increasing index order. For each sector, the algorithm applies the logic of **pop matching** (implemented in the *pop_match()* function): starting from the reference sector (based on the sector division described above), it searches for the expected features in the corresponding symmetric sectors by comparing the CENTERS POSITION. If a matching slot is found within the *position_mm* tolerance threshold, it is removed from its original sector list ("popped out") and added to a new cluster of **potentially symmetric features** (these clusters are based only on their centers position comparison). At the end of this process, the algorithm produces a series of new clusters, each containing a variable number of features with symmetric centers.
Next, it performs a comparison based on the values of the other characteristic parameters (width, height, and orientation), using a **tolerance-based comparison** similar to the centroid matching step. This process, referred to as **fuzzy matching**, relies on the *is_close_tuple* function, which compares the $\{w, h, theta\}$ values according to the company-defined tolerances discussed earlier. Each feature is then assigned one of the four possible statuses, as summarized in the table below. Special attention is dedicated to comparing the *theta* angle: before the comparison, its symmetric counterpart is first **mirrored** to derive the correct reference value.

For the cases involving features lying on the axes, the auxiliary *check_centers* function is used to verify their symmetry relative to the axis they are aligned with. This is done by measuring the distance between the axis and the feature's centroid and comparing it with the *position_mm* tolerance threshold.
Once a status has been assigned to each feature, the corresponding rectangular bounding box is drawn on the screen with the appropriate color. This makes the output immediately understandable to the supervisor, who can interpret the results at a glance using the provided reference table.

NOTE: in the table, the term *"Identical"* refers to the comparison that takes into account the reflection of the features and the tolerance criteria applied to them.

Status \ Sector	Quadrants (1–4)	Axes (5–6 or 7–8)	Center (9)
symmetric	Four identical and symmetric features	Two features that are identical, symmetric, and aligned on the axis	Single feature well-centered
missing	Three symmetric features found, with the fourth quadrant empty or containing a non-identical feature classified as an "outlier"	X	X
half	Two identical and symmetric features found among four (e.g., 2-2 split, 2-1-1 split, or two empty quadrants)	Two identical and symmetric features w.r.t. one axis, but with centroid misalignment beyond tolerance w.r.t. the other axis	X
outlier	Inconsistent feature w.r.t. the other ones in the cluster or a single feature without any symmetry matching	Two features dissimilar in size/angle (and possibly also misaligned w.r.t. their axis)	Single feature misaligned with the center

Subsequently, the *fuzzy_ groups* and *is_ close_ val* functions are used to add an additional layer of detail to the graphical output, explicitly highlighting the SOURCE OF THE ERROR. Once the characteristic parameters of each feature are displayed on the screen, they are colored in **red** according to the following criteria:

- The coordinates **(cx, cy)**: when the feature is classified as an "outlier" because it does not belong to any cluster (i.e., it did not find any correspondence based on centroid position matching).
- One or more parameters among **(w, h, theta)** if the comparison within the same symmetric cluster has failed specifically due to that value exceeding the threshold. By design choice, parameters that are present in the majority within the cluster are NOT marked in red (as they are assumed to be correct). In case of a tie, all the involved values are marked in red.

Note on tolerance within "fuzzy" clusters: The `is_close_val` and `is_close_tuple` functions organize values sequentially, starting from the sector with the lowest index: each new element is compared against the current representative within the tolerance range $\pm\theta_{th}$ (the same logic applies for w and h). This approach can lead to a total spread up to $2 \cdot \theta_{th}$.
Example: consider three features f_1, f_2, f_3 with absolute angular values of 42°, 39°, and 45°, respectively, and let $\theta_{th} = 3$:

$$f_1 = 42^\circ \xrightarrow{|39^\circ - 42^\circ| \leq 3^\circ} f_2 = 39^\circ \text{ added} \xrightarrow{|45^\circ - 42^\circ| \leq 3^\circ} f_3 = 45^\circ \text{ added}.$$

In this case, although $|45 - 39| = 6 > 3$, all three features end up in the same cluster because each one differs by only 3° from the initial representative f_1.
It is important to emphasize that this behavior does not result from any flaw in the algorithm itself but rather from the implicit choice of the threshold. To prevent features such as f_2 and f_3—which differ by 6°—from being clustered together while still meeting the 3° tolerance relative to f_1, it is necessary to reduce the initial threshold, for example by setting $\theta_{th} = 1.5^\circ$ so that the maximum spread between any pair of elements in the cluster is automatically limited to 3°.

4.3.4 Outputs of the Algorithm

At the end of the slot analysis, the code produces a dictionary named `slots_dict`, where we have two keys indicating:

- the aggregated *status* (`"OK"` if all features in the slot are `symmetric`, or `"KO"` otherwise),
- the list of potentially symmetric features, along with their characteristic parameters

A boolean flag called `defected` is also set to `true` as soon as at least one slot is classified as `"KO"`. This flag is used to immediately alert the operator with an acoustic signal indicating the presence of a potentially defective piece.
The dual-screen graphical output is shown in the *"Final Graphical Outputs and Results"* section below.

4.4. Symmetry Detection of External (and Internal) Features Using Moments and Centroids

Building upon the logic adopted in the algorithm described in the previous section, this approach aims to extend the symmetry analysis to external features, such as slots and cardboard flaps. To ensure consistency with our framework and to keep the algorithm as scalable as possible, we did not rely on predefined classifications. Instead, we chose to identify external features by extracting them from a convex hull and then using their contour (represented as a vector of spatial points) to compute their characteristic parameters, namely **moments** and **centroids**.
This framework also makes it possible to characterize internal features; however, as mentioned earlier, working in synergy with the previous algorithm remains essential to guarantee greater usability and interpretability of the results.
The inputs required by this algorithm are similar to those in the previous approach, with the difference that the binarized image now also includes the **convex hull**, as already specified in the *Chapter about Pre-processing*.

4.4.1 Characterization of Contours Using Moments and Centroids

Once the closed contours have been detected using the well-known *findContours()* function, a **binary mask** is applied to each identified feature: as illustrated in the figure on the side, the interior of each closed contour is filled with white pixels (maximum intensity), while the rest of the image is left black. This process results in a binarized image that isolates only the region corresponding to the feature, allowing for a more straightforward calculation of the CENTROIDS based on the average pixel intensity over the entire image.
Using the mask thus created, the *cv2.moments(mask)* function is applied to extract all the MOMENTS describing this region, starting from the zero-order moment (the simple area occupied by the contour) up to the third-order moment. This ensures a sufficiently accurate (though still approximate) characterization of the feature from this perspective. With the zero- and first-order moments, the centroid coordinates are computed as follows:

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

Once the centroid positions have been determined in the image reference system (cx, cy), and the positions of the feature's extreme pixels along the four main directions have been retrieved from the contour point vector, each feature can be located in one of the 9 sectors in the same way as described previously.

Theoretical Background on Moments: The *uniqueness theorem* states that the infinite sequence of moments uniquely determines a shape (up to measure-zero ambiguities), implying that any two-dimensional object can, in theory, be reconstructed exactly from all its moments.
Each binary feature is modeled as a discrete probability distribution $p(x, y)$ over foreground pixels, obtained by normalizing the pixel mask:

$$p(x, y) = \frac{f(x, y)}{\sum_{x,y} f(x, y)}$$

The **central moments**, which describe the shape independently of its position, are defined as:

$$\mu_{pq} = \sum_{x,y} (x - \bar{x})^p (y - \bar{y})^q \, p(x, y)$$

where $\bar{x}$ and $\bar{y}$ are the coordinates of the centroid. According to the uniqueness theorem, these moments capture all the information necessary to characterize the shape.

For comparing features and determining their symmetry status, the procedure follows exactly the same approach as described for the previous algorithm. Additionally, it is important to highlight the use of the auxiliary function *_anti_reflection*, which reflects a given contour according to its original sector. This function mirrors the contour across the appropriate symmetry axis so that all features—regardless of their initial quadrant (sectors 1–4)—are consistently realigned into Sector 1 (the same concept is applied to sectors 5–6 and 7–8). The transformation is performed relative to the symmetry center (cx_ax, cy_ax), which acts as the origin of the quadrant reference system, as illustrated in the images below:

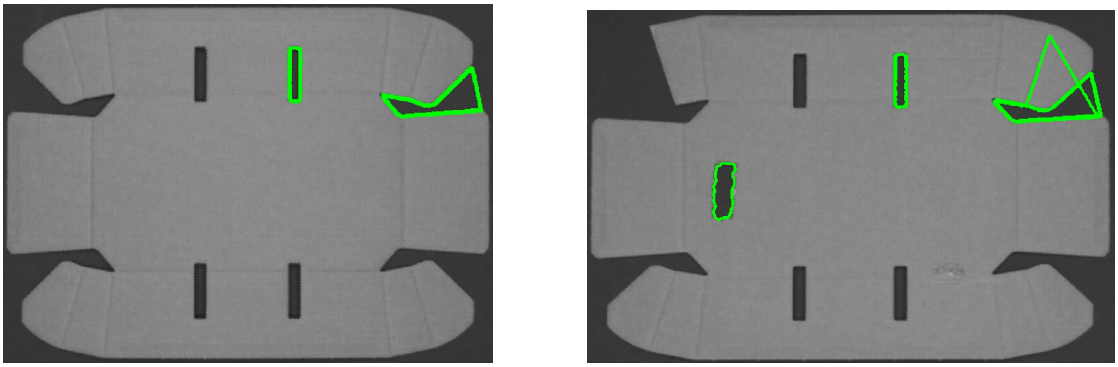


Figure 25: The left image shows the correct mirroring of the features into the first quadrant for subsequent overlap comparison. The right image shows an example of mirroring that would result in an error.

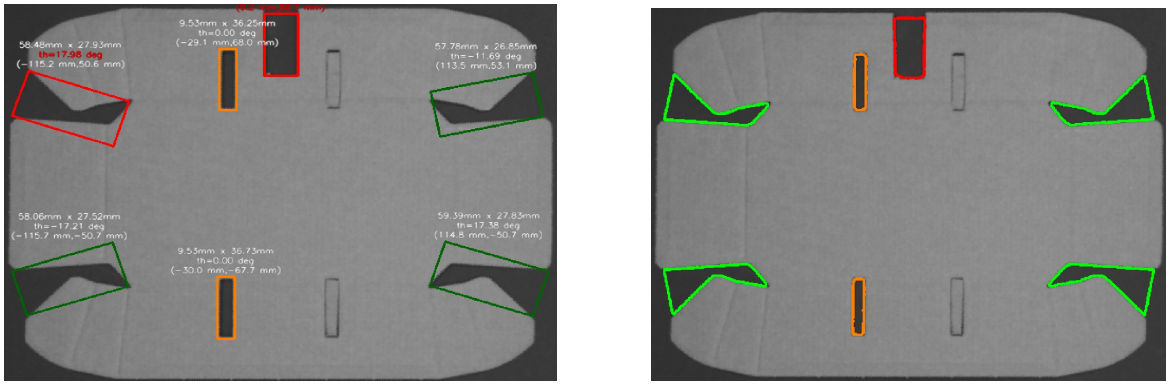
Finally, the symmetry status is assigned using methods analogous to those of the previous algorithm. However, instead of relying on the tolerance applied to dimensions and orientation, here a tolerance is applied to the relative deviation of each moment value. In particular, the *_is_close_tuple* function compares two tuples of raw spatial moments up to the third order to determine whether they are approximately equal. For each corresponding pair of moments, it computes the relative difference according to the formula:

$$\frac{|v_1 - v_2|}{\min(v_1, v_2)}$$

and verifies whether this value remains below a configured threshold ***m_th_rel***, which is a company-defined tolerance. The output of the algorithm can be seen in the next paragraph.

4.4.2 Why the Minimal-Area Rectangles Algorithm Cannot Be Applied for External Features Classification?

To justify the introduction of a second method for the classification of external features, an attempt was made to use the minimal-area rectangles approach to classify them. As clearly shown in the image on the left, this classification proves inefficient because it excessively simplifies the geometry of the feature, which is instead accurately captured using the second algorithm (image on the right).



4.5. Final Graphical Outputs and Results

Unless otherwise specified, the threshold values used for the analysis are as follows:

- *position_mm:* 5.0 mm (centroid misalignment tolerance)
- *length_mm:* 3.0 mm (tolerance on width and height)
- *theta_degree:* 6° (tolerance on orientation)
- *m_th_rel:* 12% (relative difference between moments)

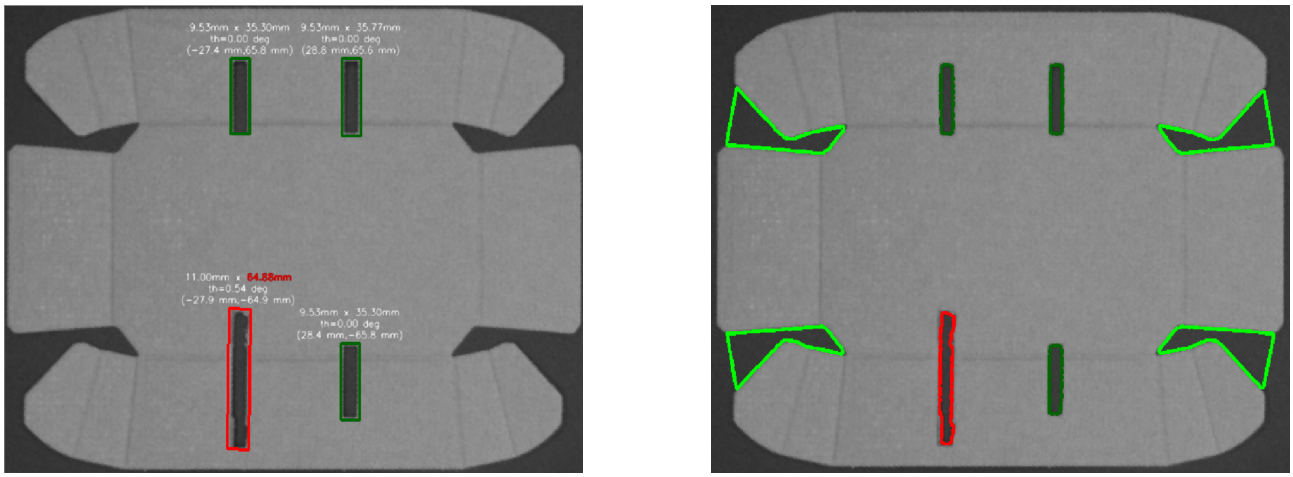


Figure 26: Defect on the height of the slot

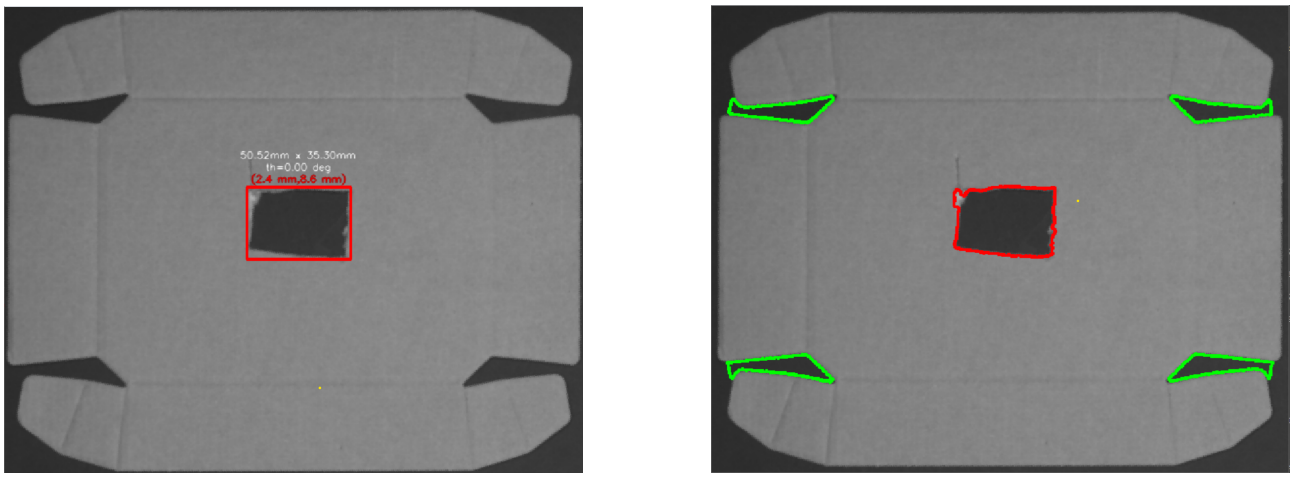


Figure 27: Misalignment of the center of a slot of both axes

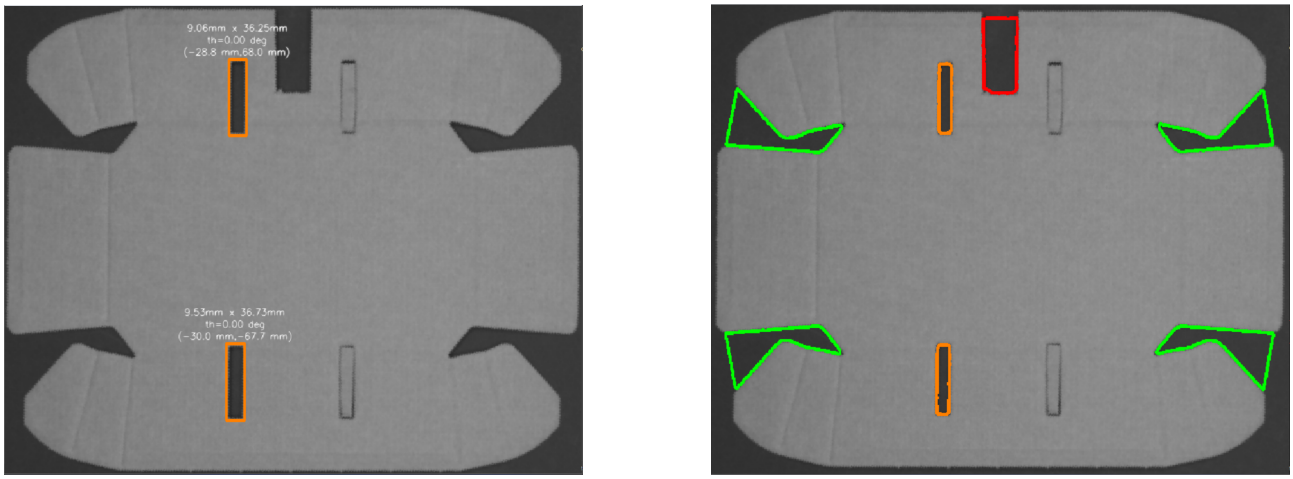


Figure 28: Features categorized as "half symmetric" due to manufacturing error and asymmetry on one external feature

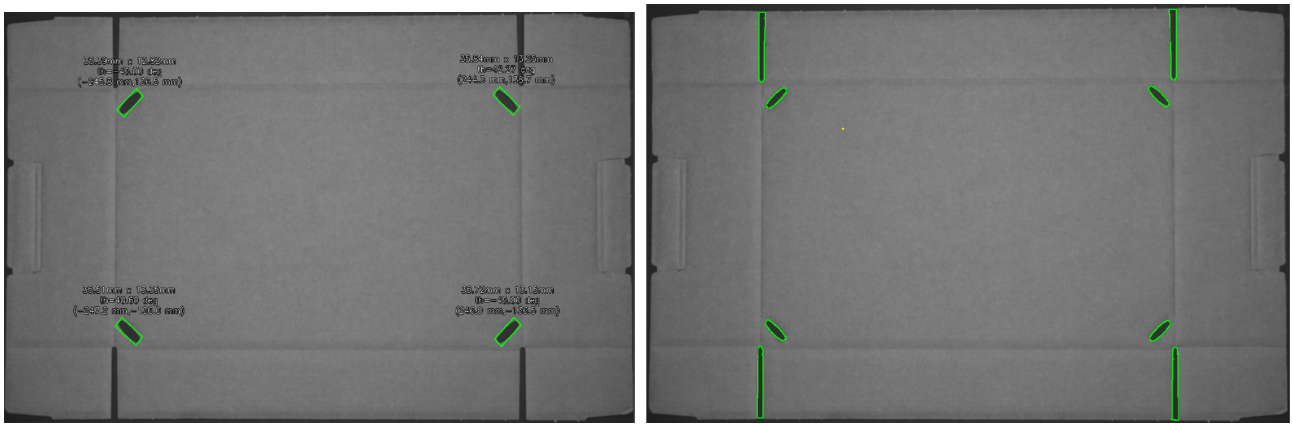


Figure 29: All external and internal features are symmetric

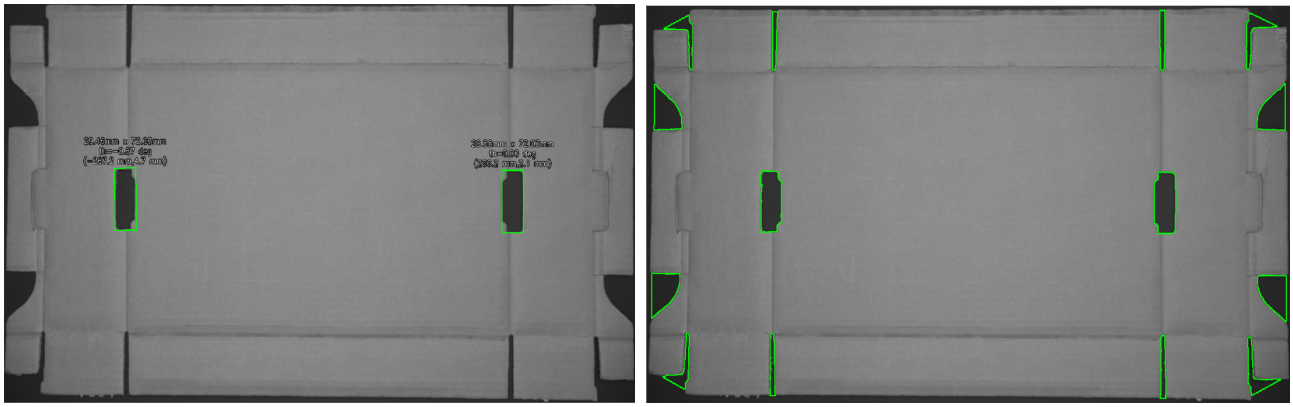


Figure 30: All external (a lot of them detected) and internal features are symmetric

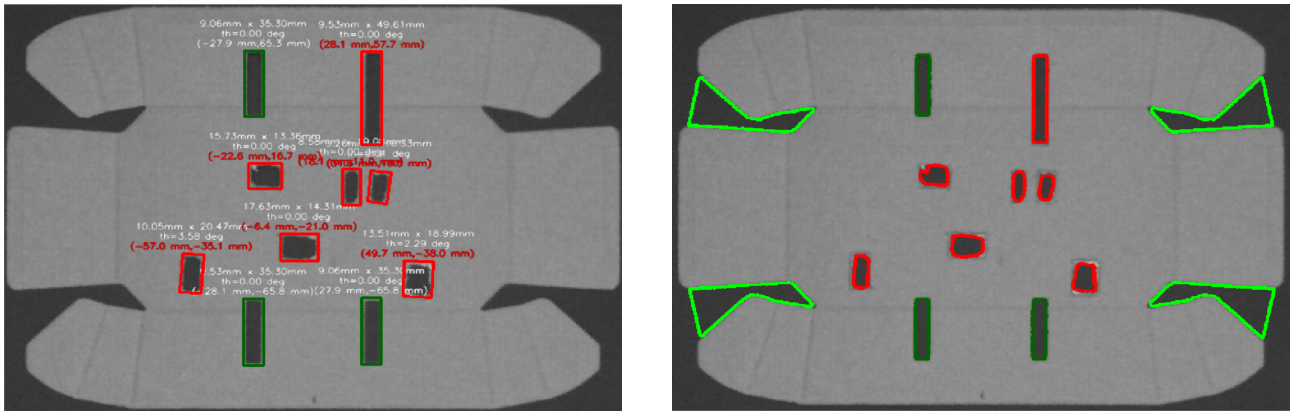


Figure 31: Several outliers in the internal part of the cardboard and a particular outlier in the upper-right part, due to the mismatch of the centers

5. Future Improvements

5.1. Software Developments

In the module *Simmetry_ Via_ Moments.py* of our codebase, we have just implemented the matching test between a template feature and a sampled one, by applying a simple thresholding on the relative difference between each moment pair of the moments vector $\mathbf{m}_T, \mathbf{m}_S \in \mathbb{R}^k$:

$$\forall i = 1, \ldots, k, \quad \frac{|m_{T,i} - m_{S,i}|}{\min(m_{T,i}, m_{S,i})} \leq \varepsilon$$

This method does not generalize well, because the relative threshold ε must typically be adjusted for each feature type based on the variance of its pixel-intensity distribution $\mathbf{I}(x, y)$. As a result, our application lacks scalability. To address this, we propose below a more robust solution that relies on a more sophisticated concept of distance in the moments space, between a template feature and its sample, giving also a statistical understanding of the results.

Statistical Hypothesis Testing with Central Moments

We now formulate symmetry verification as a statistical hypothesis test that compares the central moment vector $\hat{\mathbf{m}}_j$ of a candidate feature to the template descriptor $\mathbf{m}_T$. In this framework, **asymmetric features are interpreted as defective**, and therefore it is natural to define the null hypothesis as the absence of symmetry. This choice aligns with the goals of a quality control system, where the most critical error is to mistakenly accept a defective (i.e., asymmetric) object as symmetric (*Type I error*). On the other hand, rejecting a genuinely symmetric feature (*Type II error*) is considered less severe, as it corresponds to a conservative behavior of the system that errs on the side of caution by discarding more than strictly necessary.

Hypothesis:

$$\mathcal{H}_0: \quad \tilde{f}_j \not\sim T \quad \text{(feature is asymmetric)}$$
$$\mathcal{H}_1: \quad \tilde{f}_j \sim T \quad \text{(feature is symmetric)}$$

Each feature is modeled as a discrete probability distribution $p(x, y)$, defined over a finite support of n foreground pixels. For a single feature, the central moment vector $\hat{\mathbf{m}}_j \in \mathbb{R}^k$ is a deterministic function of the support points. However, due to imaging noise, variations in acquisition conditions, or slight deformations in shape, the observed moment vectors of geometrically equivalent features exhibit variability.
To statistically model this variability, we consider $\hat{\mathbf{m}}_j$ as a realization of a random vector. When moments are computed from a population of n pixels with empirical distribution $p(x, y)$, the resulting vector can be interpreted as a sample average over a bounded domain. Under this perspective, the variability across multiple realizations of similar features allows us to invoke the **multivariate Central Limit Theorem (CLT)**, which states that, provided finite second-order moments and sufficient sample size (sufficient number of pixel for each feature n), the sample moment vector converges in distribution to a multivariate normal:

$$\hat{\mathbf{m}}_j \sim \mathcal{N}(\mathbf{m}_j, \tfrac{1}{n}\Sigma),$$

Where $\mathbf{m}_j$ is the true moment vector of the underlying shape class, and Σ is the covariance matrix of the descriptors.

To test whether the observed feature $\tilde{f}_j$ is statistically consistent with the template T, we employ the **Hotelling's** T^2 statistic, which generalizes the univariate Student's t-test to the multivariate setting. Given that the descriptors are modeled as multivariate normal with unknown covariance matrix Σ, we estimate Σ from a training set of N known symmetric features:

$$\hat{\Sigma} = \frac{1}{N-1} \sum_{i=1}^{N} (\hat{\mathbf{m}}_i - \bar{\mathbf{m}})(\hat{\mathbf{m}}_i - \bar{\mathbf{m}})^\top, \quad \text{with} \quad \bar{\mathbf{m}} = \frac{1}{N} \sum_{i=1}^{N} \hat{\mathbf{m}}_i.$$

The Hotelling statistic comparing the candidate $\hat{\mathbf{m}}_j$ to the template mean $\mathbf{m}_T$ is:

$$T^2 = n\,(\hat{\mathbf{m}}_j - \mathbf{m}_T)^\top \hat{\Sigma}^{-1} (\hat{\mathbf{m}}_j - \mathbf{m}_T)$$

Under the alternative hypothesis $\mathcal{H}_0$ that the candidate feature is asymmetric instance of the template, the test statistic follows a chi-squared distribution:

$$T^2 \sim \chi^2_k.$$

For an asymmetric feature the statistic is stochastically larger, so small values of T^2 give evidence against the null.

Threshold Selection

Let α be the maximum probability of **false acceptance** (Type I error: accepting a defective part as good). We therefore pick the threshold in the lower tail of the reference distribution:

$$\tau \;=\; F^{-1}_{\chi^2_k}(\alpha).$$

Decision Rule

$$\boxed{T^2 \le \tau \implies \text{accept } \mathcal{H}_1 \text{ (symmetric)}} \qquad \boxed{T^2 > \tau \implies \text{retain } \mathcal{H}_0 \text{ (asymmetric)}}$$

This choice makes the test deliberately conservative: it only classifies a feature as symmetric when its moment vector falls within the lower α-quantile of the distribution learned from known good parts, thus limiting the risk of passing a defective (asymmetric) item.

5.2. Hardware Developments

To enhance robustness of our image analysis software, an improvement of the illumination technique can be performed, resulting in a better control of the product surface illumination, trying to reject all the sources of the factory illumination.

Given our CMOS sensor's quantum efficiency curve, which peaks in the blue and green bands and, and given the shifting of the beige color of the cardboard toward green - yellow wavelengths, it makes sense to align our illumination with those regions. To achieve the highest signal-to-noise ratio and suppress ambient light from overhead fluorescents lamps, we would ideally use a **narrow-band green LED** combined with a corresponding **band-pass filter**, plus a **notch filter** tuned to the strongest fluorescent emission lines (546 nm for green). Of course this environmental light blocking requires the knowledge of factory plant illumination.