ChatGPT

Methods for Comparing Image-Derived Matrices for Defect Detection

Introduction: When analyzing fiber-optic material images for defects (scratches, digs, blobs, etc.), we often represent images as matrices (with pixel coordinates and BGR intensity values). Comparing two or more such matrices can reveal small local pixel-level differences as well as large-scale pattern variations. Below is an exhaustive list of classical (non-machine-learning) techniques – spanning mathematical, statistical, and information-theoretic methods – to compare image matrices. These methods can handle matrices of different sizes (with similar spatial/intensity trends) by appropriate normalization or alignment. We include relevant formulas, computational strategies (amenable to Python/numpy), and examples of each method.

1. Mathematical & Numerical Comparison Techniques

1.1 Norm-Based Pixel Difference Metrics

These methods compute direct differences between corresponding pixel intensities, treating images as matrices and measuring norms of the difference matrix:

- **Mean Squared Error (MSE):** Calculates the average of squared intensity differences. For two images (matrices) I_1 and I_2 of size $m\times n$:

$$\mathrm{MSE} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\big(I_1(i,j) - I_2(i,j)\big)^2 \quad 【27†L1 - L4】 .$$

  Strengths: Simple to compute (one-pass difference); highlights large pixel deviations. **Weaknesses:** Treats all differences equally (sensitive to overall intensity shifts, not perceptual structure [2]). Minor misalignments or lighting changes can inflate MSE. **Python:** Use NumPy arrays: e.g. `np.mean((A.astype(float)-B.astype(float))**2)`. **Computational Cost:** $O(mn)$ (linear in number of pixels).
  Example: In fiber images, a scratch yields high local MSE in scratch pixels; overall MSE quantifies total deviation.

- **Mean Absolute Error (MAE) / Sum of Absolute Differences (SAD):** Uses absolute pixel differences instead of squared.

$$\mathrm{MAE} = \frac{1}{mn}\sum_{i,j}|I_1(i,j) - I_2(i,j)|,$$

  or SAD = $\sum_{i,j}|I_1 - I_2|$.
  Strengths: Less sensitive to outliers than MSE (no squaring), yielding a more robust measure when a

few pixels differ greatly. **Weaknesses:** Not differentiable at 0 (if using in calculus), and still ignores spatial context. **Python:** `np.mean(np.abs(A-B))`. **Cost:** $O(mn)$.
Usage: Highlights average pixel deviation; useful if we expect isolated defects (scratches) and want a general difference magnitude.

- **Max or L^∞ Norm Difference:** The maximum absolute difference: $\max_{i,j}|I_1(i,j)-I_2(i,j)|$. **Use:** To detect the single largest deviation (e.g. a particularly bright defect pixel). **Strengths:** Very sensitive to outlier differences (e.g. a deep scratch pixel). **Weaknesses:** Ignores overall pattern; only one value. **Python:** `np.max(np.abs(A-B))`. **Cost:** $O(mn)$.
Interpretation: If this value exceeds a threshold, a significant defect might be present at some pixel.

- **Peak Signal-to-Noise Ratio (PSNR):** A derivative of MSE used in imaging, defined in decibels. For maximum pixel value `MAX` (e.g. 255 for 8-bit images):

$$\mathrm{PSNR} = 10 \log_{10} \frac{\mathrm{MAX}^2}{\mathrm{MSE}} \quad 【29†L87-L90】.$$

Higher PSNR indicates closer images. **Strengths:** Widely used for image quality; compresses MSE into a log scale (dB) emphasizing perceptual differences. **Weaknesses:** Like MSE, insensitive to structural differences (two different kinds of distortion can yield same MSE [4]). **Python:** Compute MSE then `10*np.log10((MAX**2)/MSE)`. **Cost:** Same as MSE.
Example: A PSNR below a threshold (relative to a reference) could signal unacceptable defects in fiber image transmission.

- **Image Difference Map:** Instead of a single metric, one can compute the **difference image** $D(i,j) = I_1(i,j) - I_2(i,j)$. Analyzing D can localize defects. For example, one might threshold $|D|$ to find pixels where the difference exceeds noise levels, or compute statistics on D. **Strengths:** Preserves spatial info – small local differences appear as bright spots in D. **Weaknesses:** Requires further processing to summarize differences. **Python:** `D = A.astype(int) - B.astype(int)`. **Use:** Visualizing D or computing its entropy/variance (see Sections 3) helps identify and quantify anomalies.

- **Ratio Image (Pixel-Wise Ratio or Log Difference):** For illumination changes, comparing the ratio $R(i,j) = \frac{I_1(i,j)+\epsilon}{I_2(i,j)+\epsilon}$ (or log difference $\log I_1 - \log I_2$) can be more robust to global brightness shifts. **Strengths:** Emphasizes relative changes (a scratch might produce a large ratio locally even if absolute difference is moderate). **Weaknesses:** Undefined for zero intensities (hence ϵ small constant); sensitive to noise when intensities are low. **Python:** `ratio = (A+1)/(B+1)` (assuming intensity scaled 0-255). **Use:** A scratch (darker line) on an otherwise uniform fiber face would yield a low ratio in that region, even if overall brightness differs between images.

1.2 Derivative and Transform-Based Metrics

These methods leverage derivatives or transforms to emphasize structural differences (edges, textures, or frequency content), capturing small local anomalies and large-scale pattern shifts:

- **Gradient and Edge Map Differences:** Compute image gradients (e.g. via Sobel filters) or edges (via Canny detector) for each image, then compare these derivative representations. For example, let ∇I_1 and ∇I_2 be gradient magnitude images; compare them using MSE or correlation. Alternatively, compare binary edge maps E_1, E_2.
  Strengths: Defects like scratches produce strong edges – comparing gradients or edges isolates such structural differences that raw intensity MSE might dilute. **Weaknesses:** Susceptible to noise; requires tuning (e.g. edge detection thresholds). **Python:** Use OpenCV/`cv2.Sobel` for gradients or `cv2.Canny` for edges, then e.g. `np.mean((grad1 - grad2)**2)`. **Example:** A tiny crack on a fiber endface might be barely visible in intensity but yields a distinct edge; edge-map subtraction would highlight it.

- **Structural Similarity Index (SSIM):** A perceptual metric that compares local patterns of luminance, contrast, and structure [5]. SSIM is computed on local windows (e.g. 8×8) across the image and combines three terms: luminance l, contrast c, and structure s. For two windows x and y:

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad 【6†L181 – L189】,$$

  where μ = mean, σ^2 = variance, σ_{xy} = covariance, and C_1,C_2 are small constants for stability. SSIM ranges from –1 to 1 (1 means identical images).
  Strengths: Accounts for human-visual-relevant differences – structural distortions are penalized more than simple noise [4]. Good for large-scale pattern comparison and fine detail (via the local window SSIM map). **Weaknesses:** More complex than MSE; slightly slower [7]. Assumes images are aligned and same size. **Python:** Use `skimage.metrics.structural_similarity`. **Use:** SSIM is high if fiber texture is similar, but drops in regions where a scratch disrupts local structure. (One can examine the SSIM map to locate the defect.)

- **Fourier Transform and Frequency Comparison:** Compute the Discrete Fourier Transform (DFT) of each image (e.g. via 2D FFT). Compare **power spectra** to detect large-scale pattern changes or periodic differences. For example, the **Power Spectral Density (PSD)** $P_1(f)$ vs $P_2(f)$ can be compared with an L^2 norm or correlation. A significant difference in low-frequency components indicates a large-scale intensity trend difference, whereas high-frequency discrepancies indicate differences in fine details or noise.
  Strengths: Frequency domain analysis can reveal subtle pattern changes (e.g. periodic fiber bundle arrangements) not obvious in spatial domain. It's invariant to spatial shifts for magnitude comparisons. **Weaknesses:** Loses spatial localization (you know something differs at a certain frequency, but not where in image). Comparing phase requires alignment (see *Phase Correlation* below). **Python:** Use `np.fft.fft2` on each image. Compare `np.abs(F1)**2` and `np.abs(F2)**2` arrays (e.g. via MSE or by looking at specific frequency bands).
  Example: A circular blob defect might introduce extra low-frequency content (broad intensity change) and high-frequency content (sharp edges); by examining the FFT magnitude difference, one can quantify these contributions.

- **Phase Correlation (Fourier Shift Theorem):** A method for measuring translation or shift between images using the Fourier phase [8] [9]. If $F_1(\xi,\eta)$ and $F_2(\xi,\eta)$ are the Fourier transforms, **phase correlation** uses the normalized cross-power spectrum:

$$C(\xi,\eta) = \frac{F_1(\xi,\eta)\,F_2^*(\xi,\eta)}{|F_1(\xi,\eta)F_2(\xi,\eta)|}\,,$$

then finds the peak of the inverse DFT of C, which corresponds to the translation (t_x,t_y) between I_1 and I_2 [10] [11].
Strengths: Robust way to detect large-scale shifts or misalignments (the peak gives the offset to align images). Useful if images differ in position/size. **Weaknesses:** Assumes differences are mostly translational; other differences (defects) appear as noise in correlation. **Python:** Use `numpy.fft` or OpenCV's `phaseCorrelate` to get shift. **Use:** Before comparing intensities, one can align images via phase correlation. In defect detection, once aligned, any remaining differences are likely actual anomalies rather than mere misregistration.

- **Wavelet or Multi-Scale Analysis:** Decompose images using wavelet transforms or Gaussian pyramids. Compare corresponding scale components. **Strengths:** Captures differences at multiple resolutions – small scratches might appear in high-frequency sub-bands, whereas large blobs show up in low-frequency bands. **Weaknesses:** More complex to implement; need to aggregate differences across scales. **Use:** Compute e.g. a Daubechies wavelet transform for each image and compare coefficient matrices (MSE or entropy per band). This can isolate anomalies by size: fine defects vs broad illumination changes.

1.3 Geometric & Intensity Normalization

When matrices differ in size or orientation, a preprocessing step is needed before direct comparison:

- **Rescaling and Interpolation:** If one image is a scaled version of another, resample one to the same resolution as the other (using bilinear or bicubic interpolation). **Caution:** Resampling may introduce slight blurring; ensure to account for that in comparisons (e.g. use a slight tolerance for differences).
- **Cropping or Padding:** If one image is a subregion of another, align coordinates (possibly via feature matching or known fiducials) and then crop the larger image to the same region.
- **Intensity Normalization:** Adjust global intensity scale/bias (e.g. histogram equalization or simple linear contrast adjustment) so that overall brightness/contrast differences are removed. *For example, if one image is uniformly brighter, subtracting a constant or matching histograms before comparison will make defect comparison more sensitive.*

After such normalization, apply the comparison metrics described above. For instance, one might first use phase correlation or a few **keypoint matches** (from classical feature detectors like SIFT/ORB, which are not learning-based [12] [13]) to register images, then compute pixel-wise differences or SSIM on the aligned images. This ensures differences are due to defects, not misalignment.

2. Statistical Comparison Methods

2.1 Correlation and Covariance Measures

These treat pixel intensities as random variables and measure the statistical relationship between images:

- **Pearson Correlation Coefficient (PCC):** Treat the two images as vectors X and Y of length $N=mn$ (flattened pixels). PCC is the normalized covariance [14] [15]:

$$r = \frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_i (X_i - \bar{X})^2 \sum_i (Y_i - \bar{Y})^2}},$$

  which ranges from –1 to 1. $r=1$ implies one image is a perfect linear function of the other (identical up to scale/offset), and $r=0$ means no linear relationship [16] [17].
  Strengths: Captures overall **linear similarity** – insensitive to uniform brightness shifts or scale (because of mean-centering and normalization). Useful for large-scale pattern similarity.
  Weaknesses: If images have non-linear intensity relationship (e.g. one has a contrast enhanced version of the other) or localized differences, Pearson r may still be high. Also assumes images are aligned pixel-by-pixel. **Python:** `np.corrcoef(A.flatten(), B.flatten())[0,1]`. **Example:** Two fiber images with identical pattern but one slightly brighter could still have $r \approx 1$ (since correlation ignores mean differences), whereas if a big scratch alters many pixels, r will decrease. A significantly lower correlation than expected might indicate the presence of anomalies.

- **Normalized Cross-Correlation (NCC) for Templates:** This is similar to Pearson r, but computed *locally* or at offsets. In template matching, a smaller image (template) slides over a larger image; NCC at a position measures local similarity [18] [19]. The NCC formula for a template T and image window I of the same size is:

$$\mathrm{NCC}(T, I) = \frac{\sum (T - \bar{T})(I - \bar{I})}{\sqrt{\sum (T - \bar{T})^2 \sum (I - \bar{I})^2}},$$

  which is essentially Pearson correlation on that window [20]. A **peak NCC = 1** indicates a perfect match of structure (up to brightness/contrast).
  Use for Different Sizes: If I_1 and I_2 differ in size or slight translation, one can use NCC (or convolution) to find where I_2 best aligns in I_1. The highest NCC value over all shifts gives the best match [21]. **Computational Strategy:** Use FFT-based convolution to accelerate sliding correlation (Python: `cv2.matchTemplate` uses NCC if specified). **Strengths:** Effective for detecting a small sub-image or pattern inside a larger image, even with slight lighting differences (due to normalization). **Weaknesses:** Computationally heavy for large images (though FFT can reduce complexity to $O(N\log N)$). If rotation/scale differ, NCC drops (need feature-based methods in that case).
  Defect application: You could take a defect-free reference patch and slide it over a new image to locate where correlation drops (indicating a defect). Alternatively, if images are supposed to be identical, NCC = 1 everywhere when aligned; any region with low NCC could signal a localized anomaly.

- **Covariance Matrix and Hotelling's T^2 (for multiple images):** If comparing more than two images (say a set of reference images vs a test image), one can consider the pixel intensities across images as a multivariate sample. For a given pixel location, you have a vector of intensities from each reference image; you could compute the mean vector and covariance. Hotelling's T^2 statistic (a multivariate generalization of a t-test) could test if the test image's pixel fits the reference distribution. **Use:** Identify pixels that significantly deviate from the normal variation across multiple reference images (e.g., a scratch pixel intensity is outside the multivariate confidence ellipsoid). **Weaknesses:** Requires enough reference samples; complex to implement for large image (one T^2 per pixel). This is a rigorous statistical approach to flag abnormal pixels.

2.2 Distribution Comparisons (Histograms & Statistical Tests)

Instead of one-to-one pixel correspondence, these methods compare the **intensity distributions** of images (useful for large-scale differences or when images aren't perfectly aligned):

- **Histogram Comparison:** Compute intensity histograms (or multi-channel histograms for BGR) for each image and use a similarity metric [22] [23] . Common metrics:
- *Histogram Correlation:* Essentially Pearson correlation between the frequency counts in each intensity bin [20] . Ranges from –1 to 1 (1 = histograms identical shape).
- *Chi-Square (χ^2) Distance:*

$$d_{\chi^2}(H_1, H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i)} \quad \text{【10†L42 – L49】},$$

  lower is better (0 if identical histograms). Emphasizes bins where one distribution has occurrences and the other does not. **Note:** Often histograms are normalized to form probability distributions before computing this.
- *Intersection:* Measures overlap: $I(H_1,H_2) = \sum_i \min(H_1(i), H_2(i))$ [25] . Higher intersection means more similar distributions (equal when one is subset of another).

- *Bhattacharyya Distance:*

$$d_B(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_1 \bar{H}_2 N^2}} \sum_i \sqrt{H_1(i)\, H_2(i)}} \quad \text{【10†L50 – L54】},$$

  which relates to the **Bhattacharyya coefficient** $\sum \sqrt{p_i q_i}$ measuring overlap. Smaller d_B indicates more similar distributions. This is useful for comparing color histograms, for example.
  Strengths: Histogram methods are robust to spatial changes – they compare overall intensity content. For example, if an image has a scratch (dark line), the overall histogram might show a slight increase in dark pixels and decrease in mid-tones. **Weaknesses:** They ignore spatial information – a scratch or a blob might not change the histogram much if small area, so histogram could miss localized defects. Also, different spatial patterns (one image has a dark spot vs another has a dark line) could still produce similar histograms. **Python:** Use OpenCV's `cv2.calcHist` and `cv2.compareHist` (which implements correlation, chi-square, Bhattacharyya, etc.) [27] [28] .
  Computational Cost: Computing a histogram is $O(mn)$; comparing is $O(bins)$.
  Example: If Image1 has a scratch (more dark pixels) and Image2 is pristine, histogram comparison

might yield a slightly lower correlation and a non-zero chi-square. It's more useful if differences are global (e.g. one image overall brighter or higher contrast than another).

- **Two-Sample Kolmogorov–Smirnov (K–S) Test:** A non-parametric test to decide if two samples come from the same distribution [29] [30]. It uses the maximum difference between the empirical cumulative distribution functions (ECDFs) of the two image intensity distributions. The K–S statistic $D = \max_x |F_1(x) - F_2(x)|$; a larger D (with small p-value) indicates the intensity distributions differ significantly [31].
  Use: Flatten each image into a list of intensities and apply the K–S test. If $p<0.05$, you conclude the images' intensities are not from the same distribution. **Strengths:** Makes no assumption about distribution shape; sensitive to any differences in distribution (not just mean or variance). **Weaknesses:** Like histograms, ignores spatial layout. Also, if image size differs, you may need to sample equal number of pixels or weight appropriately. **Python:** `scipy.stats.ks_2samp(intensities1, intensities2)`.
  Example: A fiber image with many small defects might have a broader intensity distribution (more dark and bright outliers) compared to a pristine image. K–S test could detect this difference in distribution with a high confidence.

- **Moments and Statistical Features:** One can compare summary statistics: mean intensity $\bar{I}$, standard deviation σ_I, skewness, kurtosis of each image. For instance, a scratch might reduce the mean brightness and increase the variance (due to dark pixels) in the defective image. A simple check could be: is $|\bar{I}_1 - \bar{I}_2|$ above a threshold? Or is the variance significantly different? These are simple but quick indicators. **Example:** In fiber images, a big blob defect might raise overall brightness variance.

- **Chi-Square Test on Contingency Table:** If one wanted to include spatial correspondence, one could form a contingency table of pixel intensities (e.g. quantize intensities and count how frequently intensity i in Image1 coincides with intensity j in Image2). Then use a chi-square test for independence to see if I_1 and I_2 intensities are independent (which would mean no meaningful correlation, implying differences). This approach is essentially related to mutual information (discussed next) and is more complex, so often not needed beyond MI.

2.3 Statistical Anomaly Detection (Local)

For defect localization, statistical techniques can be applied per pixel or region:

- **Z-score (Standardized Difference):** If we have an expected value and standard deviation for each pixel (say from a set of reference images or a known noise model), we can compute $Z(i,j) = \frac{I_{\text{test}}(i,j) - \mu_{\text{ref}}(i,j)}{\sigma_{\text{ref}}(i,j)}$. Pixels where $|Z|$ is large (e.g. >3) are statistically significant outliers, likely defects. **Strengths:** Accounts for natural variation (e.g. some pixels might always vary due to fiber texture, whereas a defect is an "unnatural" variation). **Weaknesses:** Requires a baseline mean and std image. If only one reference image is available, one might estimate noise globally. **Use:** Create a "defect mask" by thresholding $|Z|$; small scratches will show as clusters of high-Z pixels.

- **Moving Window Statistics:** Compute local mean/variance in each image (using a convolution or sliding window), then compare those statistics. For example, for each location, see if the mean

intensity in a local neighborhood differs significantly between images (could use a t-test). This can catch local brightness changes like blobs or scratches. **Cost:** Convolution-based methods can do this in $O(mn)$ with filtering.

In summary, statistical methods provide measures of **overall similarity** (correlation, histogram tests) and tools for **significance testing** of differences. They are especially useful when one needs to quantify whether differences are likely due to random chance or indicate real anomalies.

3. Information-Theoretic Comparison Techniques

Information theory measures treat images (or their features) as information sources, quantifying uncertainty and mutual information content. These methods are powerful for detecting both global and local differences, and they naturally handle cases where relationships are non-linear.

3.1 Entropy-Based Measures

Entropy measures the uncertainty or complexity in an image's intensity distribution. For an image I (consider its intensity as a random variable X with possible values x and probability mass function $p(x)$):

$$H(I) = H(X) = -\sum_{x} p(x)\,\log p(x) \quad 【31†L243 - L251】,$$

with log base 2 typically yielding units in bits. High entropy means pixel intensities are widely spread (image has lots of detail or noise); low entropy means a more uniform or simple image.

- **Shannon Entropy of Each Image:** By comparing $H(I_1)$ vs $H(I_2)$, one can see if one image has more randomness. **Use:** If a defect introduces randomness (e.g. scattering light causing pepper noise or many tiny features), the entropy might increase. Conversely, a large uniform blob (dark spot) might decrease entropy by adding a uniform region. **Strengths:** Single-number summary of complexity. **Weaknesses:** Cannot localize where differences are; different types of changes can offset (one part more random, another more uniform). **Python:** Use `np.histogram` to get probabilities then `-np.sum(p * np.log2(p))`.

- **Joint Entropy and Conditional Entropy:** Consider the joint distribution of pixel intensities of two images (X,Y) (for each corresponding pixel pair). The **joint entropy** $H(X,Y)$ measures the combined uncertainty. If images are identical, Y is a function of X, so $H(X,Y) = H(X) = H(Y)$. If completely independent, $H(X,Y) = H(X)+H(Y)$. One can also look at conditional entropy $H(Y|X)$ which is $H(X,Y) - H(X)$ – this measures how much uncertainty in Y remains given X. If images are identical, $H(Y|X)=0$ (no uncertainty in Y given X), whereas if they share little info, $H(Y|X) \approx H(Y)$. **Use:** High conditional entropy means knowing one image doesn't predict the other well – a sign of large differences or misalignment.

- **Entropy of Difference Image:** Another trick: compute $D = I_1 - I_2$ (or XOR for binary images) and measure $H(D)$. A perfectly identical pair yields a very low-entropy difference (mostly zeros). Defects show up as non-zero values in D, increasing its entropy (more uncertainty). **Strengths:** Directly captures how unpredictable the differences are. **Weaknesses:** If differences are systematic (e.g. one

image is uniformly brighter by a fixed offset), D would be a constant (low entropy) even though images differ – so this works best for localized or random differences.

3.2 Mutual Information (MI)

Mutual Information measures how much information one image contains about the other. It is a fundamental similarity measure used often in multi-modal image registration because it can capture non-linear dependencies. For images represented as random variables X and Y, MI is:

$$I(X;Y) = \sum_{x}\sum_{y} p_{XY}(x,y)\,\log\frac{p_{XY}(x,y)}{p_X(x)\,p_Y(y)} \quad 【18†L279-L287】,$$

where $p_{XY}(x,y)$ is the joint probability of intensity x in image1 and y in image2 (the joint histogram), and p_X, p_Y are the marginals. Intuitively, $I(X;Y)$ is high if knowing a pixel's intensity in I_1 reduces uncertainty of the corresponding pixel in I_2. It is **maximized when images are perfectly matched** (each intensity in one predicts the other), and minimized when they are independent [34] [35]. Notably, MI can detect a relationship even if it's not linear (unlike correlation) – for example, one image could be a contrast-stretched version of the other, which is a one-to-one mapping, and MI would be maximal whereas Pearson r might not be 1 due to non-linearity.

Strengths:
- Captures both linear and non-linear intensity relationships. Useful for comparing images from different modalities (not our case here, but highlights generality) [36].
- Very effective for **image alignment**: e.g., to find the correct registration, one can try various shifts/transformations and pick the one with highest MI [37] (since misaligned images appear more independent, lowering MI).
- For our case (same modality images), a high MI indicates the images share structure; a defect that breaks correspondence will lower MI. MI accounts for joint intensity distribution: e.g., if a scratch introduces intensity values in I_2 that never occur in I_1, the joint histogram spreads out and MI decreases.

Weaknesses:
- Computationally heavier: one must estimate the 2D joint histogram of intensities. If using 256 bins per image, that's 65,536 bins; computing and normalizing this histogram is $O(mn)$ for counting, and summing for MI also goes over all bins (though many might be zero). Efficient implementations use vectorized numpy or ITK libraries.
- MI is **not normalized** – its value depends on the entropy of images. For easier interpretation, sometimes **Normalized Mutual Information (NMI)** is used: $ \text{NMI} = \frac{H(X)+H(Y)}{H(X,Y)}$, or an alternative $ \frac{I(X;Y)}{\sqrt{H(X)H(Y)}}$. NMI ranges from 0 (no similarity) to 1 (perfect similarity) in some definitions, and can handle comparisons where images have different complexity.
- MI can be **fooled by interpolation artifacts** if images are not well-aligned (e.g., small spatial misalignments can still yield high MI if intensity distributions are broad).

Python Implementation: Compute a joint histogram (e.g. with `np.histogram2d` or `cv2.calcHist` 2D). Normalize to get p_{XY}, and also get marginals p_X, p_Y by summing over rows/columns. Then apply the formula: `MI = np.nansum(joint * np.log(joint/(pX[:,None]*pY[None,:])))` (taking care to avoid log(0)).

Example: Suppose image I_1 is defect-free and I_2 has a dark scratch. Most pixels have similar intensities in both, but pixels in the scratch are dark in I_2 while their counterparts in I_1 are not so dark. This widens the joint distribution (those pixel pairs contribute to less probable combinations), thus lowering MI. If the scratch covers significant area, the drop in MI will be noticeable. In contrast, if I_2 was just a globally contrast-adjusted version of I_1 (no defect), the joint histogram would be tight along a functional relationship, and MI would be high (even if Pearson correlation might drop if the relationship is nonlinear).

3.3 Relative Entropy (Kullback–Leibler Divergence) and Variants

Kullback–Leibler (KL) divergence measures how one probability distribution diverges from another [38]. It's defined for distributions P and Q as:

$$D_{KL}(P \parallel Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)} \quad 【22†L183-L190】 .$$

In our context, we can consider P as the intensity distribution of image I_1 and Q that of I_2. $D_{KL}(P\parallel Q)$ is the information loss when Q (image2's distribution) is used to model P (image1's) [38] [40]. If they are identical distributions, $D_{KL}=0$ [41]. Larger values mean the histograms differ significantly (each pixel intensity in one image is "surprising" when using the other's histogram as a model).

Strengths: More sensitive to distribution differences than simple histogram correlation. KL gives a directional measure: e.g., a small bright defect might change P slightly relative to Q (small KL one way) but if Q has a spike of values not in P, $D_{KL}(Q\parallel P)$ could be large. So it can highlight asymmetries in differences (useful if, say, one image adds a new intensity range). **Weaknesses:** Not symmetric ($D_{KL}(P||Q)\neq D_{KL}(Q||P)$ generally) [42], and it's undefined if $Q(x)=0$ for some x where $P(x)>0$ (in practice add a small epsilon to bins). Also, it ignores spatial info just like histogram comparison. **Normalized or Symmetric Versions:** *Jensen-Shannon Divergence (JSD)* is a symmetric, smoothed version of KL that is always finite and between 0 and 1 bit. JSD could be used similarly to measure similarity of distributions in a bounded way.

Use: Compute P and Q as normalized histograms of the two images. Calculate both $D_{KL}(P||Q)$ and $D_{KL}(Q||P)$ to see distribution shifts. For example, if I_2 (defective) has some new dark pixels not in I_1, then $D_{KL}(Q||P)$ will penalize that heavily (since Q has mass where P had none). This might be more sensitive to new anomalies than a symmetric measure. But if overall distributions are just shifted (one slightly brighter overall), KL will capture that as well.

3.4 Other Information-Theoretic Measures

- **Mutual Information Maps (Local MI):** One can compute MI in a sliding window over the image to get a *local mutual information map*. This could highlight regions where the correspondence between images is low (e.g. at a defect, the relationship between image1 and image2 intensities is weaker, yielding lower MI in that local window). This is computationally intense but conceptually useful for localization.

- **Variation of Information (VoI):** Defined as $H(X|Y)+H(Y|X)$, essentially the *information lost* and *gained* between images (it's related to $H(X)+H(Y) - 2I(X;Y)$). VoI behaves like a metric (0 if images identical, larger if they differ). It can be interpreted as the average "uncertainty" remaining when using one image to predict the other. In defect detection, a higher VoI means more unpredictability introduced by defects.
- **Information Gain of Defect vs No Defect:** If you have a probabilistic model of what defect-free images look like (distribution of pixel patterns), you can measure how much information (in bits) a given image deviates. This is more in the realm of statistical modeling, but essentially, something like a **Kolmogorov Complexity** or minimum description length could be invoked: a defective image might require a longer description (due to the irregularity of the defect) than a pristine one following a simple model. Though not easily computed, it's a theoretical angle on why entropy/MI changes signal defects (defects increase complexity).

Computational Note: Information-theoretic measures often require discretizing intensities into bins (especially mutual info and KL, since you need probability distributions). For 8-bit images, one can use 256 bins directly; for higher bit-depth or color images (with 3 channels), the joint histogram grows large (e.g. 256^3 bins for 24-bit color, which is huge). In practice, one might convert to grayscale or reduce color space (e.g. compute joint histograms for each channel or use fewer bins per channel) for tractability. Python libraries like ITK or OpenCV (with `cv2.calcHist` in higher dimensions) can help. Ensure to have enough samples for joint histograms to avoid sparse data affecting MI/KL calculations.

4. Summary of Methods

Below is a summary table of the methods, their type, descriptions, strengths/weaknesses, and computational considerations:

Method	Type	Description	Strengths	Weaknesses	Computational Notes
Mean Squared Error (MSE)	Mathematical (Norm)	Average of squared pixel differences [1]. Lower MSE = more similar.	Simple; sensitive to large errors [2].	Not perceptual; equal weight to all pixels. Alignment required.	O(N) pixels. One pass diff. P arithmetic.
Mean Abs. Error (SAD/ MAE)	Mathematical (Norm)	Average of absolute pixel differences.	Simple; more robust to outliers than MSE.	Still doesn't consider structure; requires alignment.	O(N). Similar to MSE in cost
Peak SNR (PSNR)	Mathematical (Norm)	Log-scaled ratio of max signal to MSE [3] (in dB).	Common image quality metric; interpretable scale (dB).	Derived from MSE – inherits MSE's insensitivity to structure [4].	O(N) (via MSE). Used for ove assessment.

Method	Type	Description	Strengths	Weaknesses	Computational Notes
Structural Similarity (SSIM)	Mathematical (Perceptual)	Combines local luminance, contrast, structure comparisons [6]. Yields index in [–1,1].	Human-vision aligned; detects structural changes well [5].	Slightly slower; assumes local window analysis.	O(N) with window ops. Avai `skimage` [43].
Gradient/ Edge Difference	Mathematical (Derivative)	Compare image gradients or edge maps (e.g. Sobel, Canny) between images.	Emphasizes fine defects (scratches, edges).	Prone to noise; ignores flat regions differences.	O(N) to filter + compare. Us gradients.
Fourier/Freq. Comparison	Mathematical (Transform)	Compare DFT magnitudes (power spectra) of images. Large-scale differences show in low-freq; fine details in high-freq.	Captures pattern differences independent of shift (magnitudes).	Loses spatial info; phase differences not captured (unless using phase correlation).	O(N log N) for FFT. Use num Compare spectra via norms
Phase Correlation (FFT)	Mathematical (Alignment)	Uses Fourier phase to find translation between images [10]. Peak in cross-correlation indicates shift.	Robust alignment for translational offsets; can pre-align images.	Only handles shifts (not rotation/scale); defects act like noise.	O(N log N) with FFT. Implem or OpenCV.
Histogram Comparison	Statistical (Distribution)	Compares intensity histograms using metrics like correlation, Chi-square [24], intersection, Bhattacharyya [26].	Invariant to spatial rearrangements; handles global brightness changes (with correlation metric).	Insensitive to local defects (small area changes barely affect histogram); requires intensity normalization if large exposure differences.	O(N) to build histograms + compare. OpenCV `compar` available [28].

Method	Type	Description	Strengths	Weaknesses	Computational Notes
Correlation Coefficient	Statistical (Global)	Pearson r between images (linear intensity correlation) [14] . Ignores absolute scale/offset.	Simple global similarity; ignores uniform lighting differences.	Only linear relationships; a non-linear intensity mapping can lower r even if images match visually.	O(N). Can use `np.corrcoe`
Normalized Cross-Corr (NCC)	Statistical (Local)	Pearson correlation computed for a template/window sliding over an image [18] . Measures local similarity; peak indicates best align.	Finds pattern matches regardless of brightness/ contrast offsets [19] . Useful for sub-image location or local defect detection by correlation drop.	Expensive if done naively over all shifts; assumes small geometric distortion.	O(N * shift_area) or O(N log OpenCV `matchTemplate`
Kolmogorov–Smirnov Test	Statistical (Distribution)	Non-parametric test comparing two intensity distributions [29] . Yields statistic (max CDF difference) and p-value.	Detects any distribution difference (shift, spread, etc.) statistically.	No spatial info; needs many pixels for confidence.	O(N log N) to sort intensitie provides `ks_2samp` .
Z-score Anomaly Map	Statistical (Local)	Compute standardized difference per pixel: $ (I_{\text{test}} - \mu_{\text{ref}})/ \sigma_{\text{ref}} $. Large	Z	flags a defect at that pixel.	Accounts for expected nois each pixel; sensitive to abn

Method	Type	Description	Strengths	Weaknesses	Computational Notes
Mutual Information (MI)	Info-Theoretic	Measures shared information between images in bits [33]. Computed from joint intensity histogram. High MI = strong dependency/ similarity.	Captures non-linear intensity relationships; robust for multi-modal comparisons [37]. Effective for image alignment and detecting decorrelation due to defects.	Computationally heavy (joint histogram); value depends on image entropy (not normalized unless using NMI). Defects in small areas may have a subtle effect on total MI.	O(N) to gather joint hist + C calc (for 256×256 bins). Use Consider normalized MI for
Entropy (Shannon H)	Info-Theoretic	Measures uncertainty of an image's intensity distribution [32]. Higher entropy = more complex/ random image. Compare $H(I_1)$ vs $H(I_2)$, or compute joint entropy $H(I_1,I_2)$.	Simple scalar summary of image complexity; can indicate if a defect adds randomness or uniform regions.	Doesn't tell where changes occur; different defects can increase or decrease entropy.	O(N) to get histogram then sum.
KL Divergence	Info-Theoretic	Measures how one image's intensity distribution diverges from the other's [39]. $D_{KL}(P\parallel Q)=0$ if identical hists.	Sensitive to distribution differences (e.g. new intensity values introduced by defects). Directional measure can pinpoint which image has unexpected values.	Not symmetric; can blow up if one image has intensities absent in the other (mitigate by smoothing). Ignores spatial info.	O(bins) after obtaining distr with caution (smooth zeros

Method	Type	Description	Strengths	Weaknesses	Computational Notes
Earth Mover's Distance (EMD)	Statistical (Distribution)	Treats histograms as piles of "earth" and computes minimum cost to transform one into the other [44] [45] (Wasserstein metric). Essentially, a measure of how far intensity distributions shift.	Accounts for the magnitude of intensity shifts (not just differing counts). E.g., recognizes that a distribution shifted by +5 gray levels is a small change.	Computationally more expensive (requires solving a transportation problem); needs careful normalization if images differ in total pixels.	Can be O(bins^2) in worst c algorithms exist. Python: us `scipy.stats.wasserste` for 1D histograms.

Table: Summary of classical methods for comparing image-derived matrices. "Type" indicates the category (mathematical, statistical, or information-theoretic). Computational requirements assume images of N pixels.

Conclusion

By leveraging the techniques above, one can rigorously compare two or more image matrices to detect and analyze differences at multiple scales. A practical defect-detection workflow might combine several methods: for example, use phase correlation or feature matching to align images, then compute pixel-wise difference maps enhanced by gradient filtering, and finally quantify the significance of differences via statistical tests or mutual information drop. All the listed methods avoid machine learning and are implementable with Python's scientific libraries (NumPy, SciPy, OpenCV, scikit-image), making them suitable for integration into inspection pipelines. By examining both local pixel-level deviations and global pattern changes through these classical methods, one can reliably identify fiber optic material defects such as scratches, digs, and blobs and quantify their impact on the images.

---

[1] [6] [7] [12] [13] [22] [23] [27] [28] [43] Algorithms for Image Comparison - GeeksforGeeks
https://www.geeksforgeeks.org/computer-vision/algorithms-for-image-comparison/

[2] [4] [5] A Quick Overview of Methods to Measure the Similarity Between Images | by Data Monsters | Medium
https://medium.com/@datamonsters/a-quick-overview-of-methods-to-measure-the-similarity-between-images-f907166694ee

[3] How To Calculate PSNR
https://www.sciencing.com/how-to-calculate-psnr-12751768/

[8] [9] [10] [11] Phase correlation
https://sthoduka.github.io/imreg_fmt/docs/phase-correlation/

[14] [15] [16] [17] Pearson correlation coefficient - Wikipedia
https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

[18] [19] [21] Image Matching: A Comprehensive Overview of Conventional and Learning-Based Methods
https://www.mdpi.com/2673-8392/5/1/4

20 24 25 26 OpenCV: Histogram Comparison
https://docs.opencv.org/3.4/d8/dc8/tutorial_histogram_comparison.html

29 30 31 Kolmogorov–Smirnov test - Wikipedia
https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test

32 Entropy (information theory) - Wikipedia
https://en.wikipedia.org/wiki/Entropy_(information_theory)

33 34 35 Mutual information - Wikipedia
https://en.wikipedia.org/wiki/Mutual_information

36 37 Using mutual information and cross correlation as metrics for ...
https://www.researchgate.net/publication/
254440161_Using_mutual_information_and_cross_correlation_as_metrics_for_registration_of_images

38 39 40 41 42 Kullback–Leibler divergence - Wikipedia
https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence

44 45 Earth mover's distance - Wikipedia
https://en.wikipedia.org/wiki/Earth_mover%27s_distance