CrysAI ^{TM} Evaluation Metrics for Crystal Shape & Size Descriptors

1 Size-Based Metrics

1.1 DL-IA Chord Length

The chord length is calculated as the horizontal distance between the leftmost and rightmost points of a contour intersected by a horizontal line. For each detected contour in the image, an axis-aligned bounding box is first obtained using the cv2.boundingRect function, defining the region where the contour lies. A mask is then created for the contour, and horizontal lines are iteratively drawn across the height of the bounding box. The intersection of each horizontal line with the contour is computed using the *cv2.bitwise_and* function from OpenCV library.

For each valid intersection (where at least two points exist), the leftmost and rightmost intersection points are identified, and the Euclidean distance between them is calculated as the chord length. This distance is given by the formula:

$$\mathrm{CL} = \frac{\sqrt{\left(x_{right} - x_{left}\right)^2 + \left(y_{right} - y_{left}\right)^2}}{pixelsPerMetric} \qquad (1)$$

where CL is the chord length, (x_{left}, y_{left}) and (x_{right}, y_{right}) are the coordinates of the leftmost and rightmost intersection points, respectively, and $pixelsPerMetric$ is the scaling factor that converts pixel distances to real-world units (e.g., µm). The calculated chord lengths from all valid intersections are stored and used for further analysis, such as determining chord length distributions, square-weighted mean chord length (SWMCL) used in the time-series plots etc.

1.2 DL-IA Particle Size/Length (Maximum Feret Diameter)

The Feret diameter, also known as the maximum calliper diameter, is used to quantify the size of crystal particles. It is defined as the greatest distance between any two points on the particle's contour. In the python script, the Feret diameter for a given crystal particle's contour is determined by iterating through all point pairs within the contour and computing their Euclidean distance.

Mathematically, the Maximum Feret diameter D_{feret} is calculated as:

$$\mathrm{D}_{feret} = \max\left\{dist(\mathrm{P}_i, \mathrm{P}_j) \mid \mathrm{P}_i, \mathrm{P}_j \in Contour\ Points\right\} \qquad (2)$$

Where, $dist(\mathrm{P}_i, \mathrm{P}_j)$ represents the Euclidean distance between two points $\mathrm{P}_i(x_i, y_i)$ and $\mathrm{P}_j(x_j, y_j)$ given by:

$$dist(\mathrm{P}_i, \mathrm{P}_j) = \frac{\sqrt{\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2}}{pixelsPerMetric} \qquad (3)$$

The maximum distance is then normalized by the scaling factor, $pixelsPerMetric$, which converts pixel distances to actual measurement unit (µm). The Feret diameter reflects the actual size of the particle, irrespective of the particle orientation or image resolution.

1.3 DL-IA Particle Width

The **particle width** represents the shortest dimension of a particle, measured perpendicular to its longest dimension (i.e. **Feret diameter**). This measurement complements the particle length to describe its overall shape. In the calculation, the **minimum bounding rectangle** around the particle contour is determined using the OpenCV function *cv2.minAreaRect()*. This function computes the smallest rectangle enclosing the contour, which may be rotated to align with the particle's major and minor axes. The rectangle provides two dimensions: width (W) and height (H), representing the lengths of its sides. The **particle width** is then defined as the smaller of these two dimensions. Mathematically, it can be expressed as:

$$\text{Particle width} = \frac{\min(W, H)}{pixelsPerMetric} \tag{4}$$

where: W and H are the dimensions of the bounding rectangle obtained from *cv2.minAreaRect*().

This approach ensures that the particle width is calculated in a manner that is robust to the orientation of the particle, making it particularly effective for irregularly shaped particles. By pairing the particle width with the particle length, additional shape descriptors i.e. the **aspect ratio** was derived from it, offering a comprehensive geometric characterization of the particle.

1.4 DL-IA Circular Equivalent Diameter (CircularED)

Circular Equivalent Diameter (CED) can be computed using two different approaches. The circular equivalent diameter based on projected area, **CED$_A$** (commonly used in many particle size analysis instruments), it calculates the **diameter of a circle with the same area** as the irregular crystal particle shape. The perimeter of the particle-based, **CED$_P$** is the **diameter of a circle with the same perimeter** as the irregular shape crystal particle. CED$_P$ is used to analyze edge roughness and shape irregularity.

Based on projected area

$$\text{Circular Equivalent Diameter } (\text{CED}_A) = \frac{\sqrt[2]{4A/\pi}}{pixelsPerMetric} \tag{5}$$

Based on perimeter

$$\text{Circular Equivalent Diameter } (\text{CED}_P) = \frac{\sqrt[2]{P/\pi}}{pixelsPerMetric} \tag{6}$$

1.5 DL-IA Spherical Equivalent Diameter (SphericalED)

The **Spherical Equivalent Diameter (SED)** of an irregularly shaped crystal particle, is estimated using the diameter of a sphere that has the same volume or surface area as the irregular shape. For an irregularly shaped crystal particle contours in 2D image, the SED can be using the area A:

$$\text{Spherical Equivalent Diameter (SED)} = \frac{2\sqrt[2]{A/\pi}}{pixelsPerMetric} \tag{7}$$

It estimates the diameter of a circle with the same area as the crystal particle’s 2D projection.

1.6 DL-IA Spherical Equivalent Perimeter (SphericalEP)

The **Spherical Equivalent Perimeter (SEP)** approximates the perimeter of a sphere’s 2D projection (i.e., the circumference of a circle that has the same projected area as the irregular shape). It can be used as a size-based metrics to compare irregular particle perimeters to an idealized sphere.

$$\text{Spherical Equivalent Perimeter } (SEP) = \frac{2\pi \sqrt[2]{A/\pi}}{pixelsPerMetric} \qquad (8)$$

2 Shape Descriptor-Based Metrics

2.1 Aspect Ratio

The aspect ratio describes the elongation of a particle by comparing its longest and shortest dimensions. It is computed as the ratio of the **major axis length** (longest dimension) to the **minor axis length** (shortest dimension). This measurement helps characterize whether a particle is **spherical, elongated, or needle-like**.

To determine the aspect ratio, the **minimum bounding rectangle** is fitted around the particle using OpenCV's *cv2.minAreaRect()* function. This function finds the smallest possible **rotated rectangle** enclosing the particle. The rectangle has two dimensions, W and H:

$$\text{AR } = \frac{\max{(W, H)}}{\min{(W, H)}} \qquad (9)$$

2.2 Circularity & Sphericity

Circularity quantifies how close a particle shape is to a perfect **circle** by comparing its 2D projected **area** and **perimeter** to a perfect circle. It is sensitive to **elongation and surface irregularities**. The python calculation is performed using OpenCV functions: *cv2.contourArea(contour)* computes the particle **area**, and *cv2.arcLength(contour, True)* to compute the particle **perimeter**.

Defined as:

$$\text{Circularity, C } = \frac{4\pi A}{P^2} \qquad (10)$$

where A is area and P is perimeter.

Sphericity is a **3D measure** that describes how closely the **overall shape of a particle** resembles a sphere. Defined as (for 3D object):

$$\text{Sphericity, } \varphi = \frac{A_s}{A_{sphere}} \qquad (11)$$

Or using volume and surface area:

$$\text{Sphericity}, \varphi = \frac{\pi^{1/3}(6V)^{2/3}}{A_s} \quad (12)$$

Where: A_s is the surface area of the particle. A_{sphere} is the Surface area of a sphere with the same volume. For **2D contour particle data,** a simple method is used to estimate the **3D sphericity** without assuming volume or thickness. Sphericity (φ) is approximated as the square root of circularity (C):

$$\text{Sphericity}, \varphi = \sqrt{\frac{4\pi A}{P^2}} \quad (13)$$

2.3 Compactness (or Roundness)

Compactness is a shape descriptor that quantifies how **efficiently a particle fills its bounding area** or **volume**. It helps differentiate dense, rounded particles from elongated or irregular ones. Roundness describes **how smooth and curved the edges of a particle are**, compared to a perfect circle. Unlike circularity, which is based on area and perimeter, **roundness** emphasizes **edge smoothness**. **Roundness** is approximated by comparing the **particle perimeter** to its **maximum Feret diameter**:

$$\text{Roundness} = \frac{4A_P}{\pi\, Feret_{max}^{2}} \quad (14)$$

$$\text{Compactness} = \sqrt{\frac{4A_P}{\pi\, Feret_{max}^{2}}} \quad (15)$$

2.4 Convexity (or solidity)

Convexity (or Solidity) measures the **compactness** of a particle by comparing its actual **area** to the area of its **convex hull**. It helps detect **particle porosity, fragmentation, or rough growth**. The **convex hull** of the particle is determined using *cv2.convexHull(contour)*, which computes the smallest convex shape that completely encloses the particle. Solidity is then computed as:

$$\text{Solidity or Convexity, S} = \frac{A}{A_{convex\ hull}} \quad (16)$$

2.3 Roughness factor (RF)

The **roughness factor** quantifies **edge irregularities** by comparing the **perimeter-based circular equivalent diameter (CED)** with the **area-based spherical equivalent diameter (SED)**. It is useful for **identifying crystal particle surface roughness and agglomeration effects**. Roughness factor is estimated as the ratio of CED_P and SED_A.

$$\text{Roughness Factor (RF)} = \frac{\text{CED}_P}{\text{SED}_A} \quad (17)$$

3 Data & Statistical Analysis-Based Metrics (To be added)

3.1 Mean, Standard Deviation, Mode

3.2 Histogram & Gaussian (Kernel Density Estimate) Distributions

3.3 Square and Volume Weighting

3.4 Percentiles, Surface & Vol. Mean Diameter (D10/D50/D90, D[4,3], D[3,2])

3.5 Moving Average and Gaussian-based Smoothing