GALAXY INTERNATIONAL INTERDISCIPLINARY RESEARCH JOURNAL (GIIRJ)
ISSN (E): 2347-6915
Vol. 13, Issue 4 April (2025)

AUTOMATED MORPHOLOGICAL ANALYSIS OF SEM IMAGES OF COPPER ALLOY MICROSTRUCTURES USING C++ AND OPENCV

Ilhomjon Rasulovich Samandarov
Ph.D. in Technical Sciences, Associate Professor, Department of Mathematics and Natural Sciences, Almalyk Branch of Tashkent State Technical University named after Islam Karimov, Republic of Uzbekistan, Almalyk
Email: samadarov.2003@gmail.com

ABSTRACT

This article presents a methodology for quantitatively assessing the degree of deformation in particle shapes observed in scanning electron microscope (SEM) images. The analysis utilizes the C++ programming language and the OpenCV library. The study includes preprocessing, segmentation of particles, and extraction of morphological parameters such as eccentricity, compactness, and shape factor. The obtained results enable comparative evaluation of particle deformation levels under different processing conditions for copper alloys.

Keywords: Deformation degree, image segmentation, morphological analysis, OpenCV, particle shape, SEM images, C++.

INTRODUCTION

The shape and size of microstructural particles observed in SEM images of metallic samples are key indicators reflecting the degree of plastic deformation of the material. Specifically, in the analysis of copper alloys, changes in grain shape can indicate processing conditions such as cold rolling, annealing, or severe plastic deformation.

Traditional methods for quantitative microstructure analysis are often time-consuming and require manual work. In contrast, automated image analysis using computer vision techniques significantly enhances the accuracy and reproducibility of results. The OpenCV library in C++ provides a comprehensive set of tools for image processing, contour extraction, and computation of morphological characteristics.

This study focuses on the development and implementation of an algorithm for the automatic analysis of SEM images to quantitatively assess particle deformation. Key metrics include shape parameters such as roundness, eccentricity, and compactness, which provide objective criteria for comparing microstructural states under various processing regimes.

The material analyzed consisted of SEM images of copper alloy microstructures subjected to different degrees of plastic deformation. All images had high resolution (at least 1024×1024 pixels) and were stored in PNG or JPEG formats.

The image processing was carried out using OpenCV version 4.x in the C++ programming environment, with code compilation and debugging performed in Visual Studio 2022. The analysis algorithm included the following stages:

1. Image Preprocessing

- Conversion to grayscale (cv::cvtColor)
- Noise filtering using a median filter (cv::medianBlur)
- Contrast enhancement via histogram equalization (cv::equalizeHist)

2. Binarization and Segmentation

- Application of Otsu's method (cv::threshold with THRESH_OTSU)
- Morphological operations (erosion and dilation) to improve particle separation (cv::morphologyEx)

3. Contour Extraction and Morphological Parameter Calculation

- Contour detection using cv::findContours
- Calculation of area, perimeter, and other particle-specific features
- Computation of shape coefficients, such as:
- Roundness coefficient: $C = \frac{4\pi \cdot \mathrm{Area}}{x^2 2}$
- Compactness: ratio of area to the minimum bounding rectangle
- Eccentricity: based on ellipse approximation via cv::fitEllipse

4. Classification and Visualization

- Particles with high elongation and low roundness were interpreted as more deformed
- Results were saved as annotated images and in tabular form with particle parameters

Figure 2 shows the original SEM image converted to grayscale, which is a necessary step prior to binarization and morphological analysis.
Figure 2. Image after grayscale conversion.
After filtering and thresholding using Otsu's method, binarized images were obtained (Figure 3) in which particles are successfully separated from the background. The application of morphological operations helped eliminate noise and merged elements, thereby improving the accuracy of subsequent contour extraction.
Figure 3. Binarized image using Otsu's method.
Figure 4. Image after morphological filtering.
As a result of the algorithm execution, particle contours were extracted and morphological parameters were calculated, including area, perimeter, circularity, and eccentricity. An example of contour and particle ID visualization is shown in Figure 5.
Figure 6. Final visualization with detected particles and assigned labels.

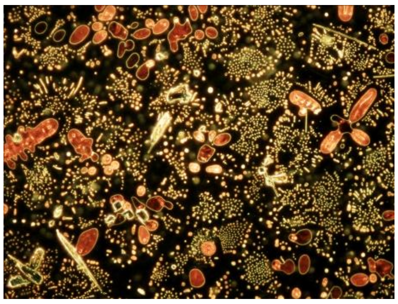
Figure 1. Original image

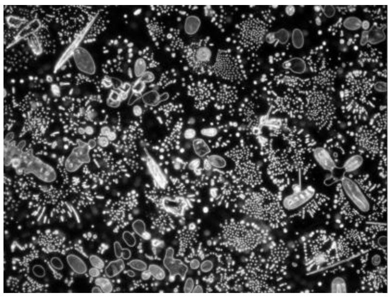
Figure 2. Original SEM image converted to grayscale

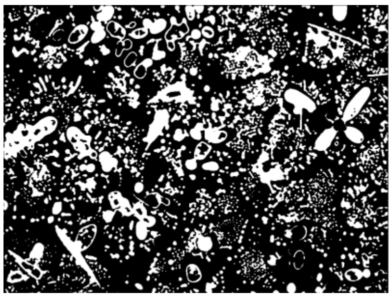
Figure 3. Binarized image using Otsu's method

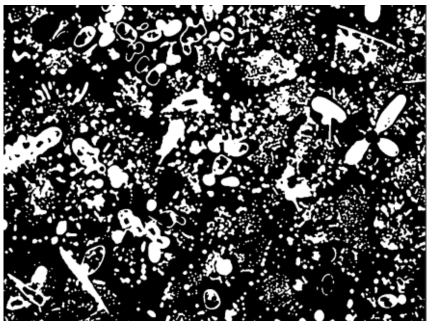
Figure 4. Image after morphological filtering

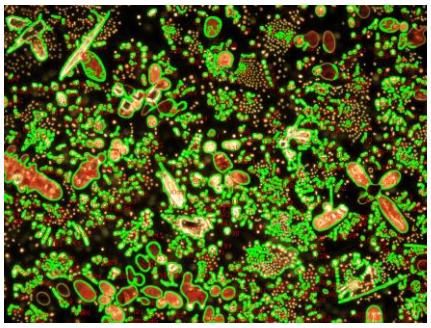
Figure 5. Example of contour and particle ID visualization

Table 1 presents sample quantitative characteristics obtained for several particles. In total, measurements were performed for 508 particles.

Table 1. Morphological characteristics of selected particles

ID	Area (px)	Perimeter (px)	Circularity	Eccentricity
16	63.50	35.90	0.62	0.83
21	108.50	43.56	0.72	0.31
30	83.00	38.63	0.70	0.84
36	109.50	44.38	0.70	0.86
42	179.50	64.53	0.54	0.56
43	310.50	86.33	0.52	0.73
47	83.00	47.11	0.47	0.84
51	85.00	68.43	0.23	0.92
57	56.50	34.38	0.60	0.95
60	53.00	42.28	0.37	0.98

The analysis revealed that particles with high eccentricity values (> 0.8) and low circularity (< 0.5) exhibit a high degree of elongation, indicating plastic deformation. In contrast, particles with circularity close to 1 have an almost spherical shape and were likely subjected to minimal deformation.

The developed algorithm, based on the OpenCV library and implemented in C++, allows for efficient analysis of SEM images of copper alloy microstructures. The resulting morphological parameters (eccentricity, circularity, and compactness) provide a quantitative evaluation of particle deformation.

This automated approach ensures the reproducibility of analysis and enables the comparison of microstructural changes resulting from different processing conditions, such as cold rolling or severe plastic deformation.

Software Implementation

The image analysis was implemented in the C++ programming language using the OpenCV library. The code includes stages for preprocessing, segmentation, contour extraction, and morphological parameter calculation.

The program source code is provided below:

```
#include <opencv2/opencv.hpp>
#include <iostream>
```

```
#include <vector>
#include <cmath>
#include <iomanip>

using namespace cv;
using namespace std;

int main() {
   // 1. Load the image
   Mat image = imread("1.jpg");
   if (image.empty()) {
      cout << "Image not found!" << endl;
      return -1;
   }

   Mat gray, binary, morph;

   // 2. Convert to grayscale
   cvtColor(image, gray, COLOR_BGR2GRAY);
   imshow("1. Grayscale Image", gray);

   // 3. Remove noise using a median filter
   medianBlur(gray, gray, 5);

   // 4. Apply thresholding (Otsu's method)
   threshold(gray, binary, 0, 255, THRESH_BINARY + THRESH_OTSU);
   imshow("2. Binarized Image", binary);

   // 5. Morphological cleaning
   Mat kernel = getStructuringElement(MORPH_ELLIPSE, Size(3, 3));
   morphologyEx(binary, morph, MORPH_OPEN, kernel);
   imshow("3. Morphologically Cleaned", morph);

   // 6. Find contours
   vector<vector<Point>> contours;
   vector<Vec4i> hierarchy;
   findContours(morph, contours, hierarchy, RETR_EXTERNAL,
CHAIN_APPROX_SIMPLE);

   // 7. Clone original image for output display
   Mat output = image.clone();
```

```
    // 8. Structure to hold particle data
    struct Particle {
        int id;
        double area;
        double perimeter;
        double circularity;
        double eccentricity;
    };
    vector<Particle> particles;

    // 9. Analyze each contour
    for (size_t i = 0; i < contours.size(); ++i) {
        double area = contourArea(contours[i]);
        double perimeter = arcLength(contours[i], true);

        // Filter out small noise
        if (area < 50 || perimeter == 0) continue;

        double circularity = 4 * CV_PI * area / (perimeter * perimeter);

        // Calculate eccentricity if possible
        double eccentricity = 0.0;
        if (contours[i].size() >= 5) {
            RotatedRect ellipseFit = fitEllipse(contours[i]);
            double a = max(ellipseFit.size.width, ellipseFit.size.height) / 2.0;
            double b = min(ellipseFit.size.width, ellipseFit.size.height) / 2.0;
            eccentricity = sqrt(1 - (b * b) / (a * a));
        }

        particles.push_back({ static_cast<int>(i + 1), area, perimeter, circularity, eccentricity });

        // Draw contours and label them
        drawContours(output, contours, static_cast<int>(i), Scalar(0, 255, 0), 2);
        Moments mu = moments(contours[i]);
        int cx = static_cast<int>(mu.m10 / mu.m00);
        int cy = static_cast<int>(mu.m01 / mu.m00);
        putText(output, to_string(i + 1), Point(cx - 10, cy), FONT_HERSHEY_SIMPLEX, 0.5,
Scalar(0, 0, 255), 1);
    }

    // 10. Print summary table
    cout << left << setw(10) << "ID"
         << setw(15) << "Area"
        << setw(15) << "Perimeter"
        << setw(15) << "Circularity"
        << setw(15) << "Eccentricity" << endl;
    cout << string(70, '-') << endl;

    for (const auto& p : particles) {
        cout << left << setw(10) << p.id
            << setw(15) << fixed << setprecision(2) << p.area
            << setw(15) << p.perimeter
            << setw(15) << p.circularity
            << setw(15) << p.eccentricity << endl;
    }

    // 11. Show final annotated image
    imshow("4. Final Result", output);
    imwrite("output.jpg", output);

    cout << "\nAll images were displayed. 'output.jpg' has been saved." << endl;

    waitKey(0); // Keep windows open until a key is pressed
    return 0;
}
```

References

1. Gonzalez, R., Woods, R. Digital Image Processing. – Moscow: Tekhnosfera, 2005. – 1072 p.
2. Bradski, G., Kaehler, A. Learning OpenCV: Computer Vision with the OpenCV Library. – St. Petersburg: Piter, 2010. – 464 p.
3. Russ, J.C. The Image Processing Handbook. – CRC Press, 2016. – 1040 p.
4. Sedov, A.A., Morozov, P.N., Khromov, A.V. Automation of Microstructure Analysis of Metallic Materials Based on SEM Images // Metallurgy and Materials. – 2020. – No. 2. – P. 42–47.
5. Mishin, A.A., Solovyev, V.P. Morphological Analysis Methods for Structural Evaluation of Materials // MAI Proceedings, 2018. – No. 103.
6. Otsu, N. A Threshold Selection Method from Gray-Level Histograms // IEEE Transactions on Systems, Man, and Cybernetics, 1979. – Vol. 9(1). – P. 62–66.
7. Haralick, R.M., Shapiro, L.G. Computer and Robot Vision. Vol. 1. – Addison-Wesley, 1992.
8. Samandarov, I.R., Manshurov, Sh.T., Dushatov, N.T., Miratoyev, Z.M., Mustafin, R.R. Image Processing in C++ Using OpenCV Library // Universum: Technical Sciences. – 05.2023 – No. 5(110).
9. Starodubov, D., Sadykov, S., Samandarov, I., Dushatov, N., Miratoyev, Z. Method of Investigation of Stability and Informativeness of Basic and Derived Features in Microscopic and Defectoscopic Image Analysis of Cast Iron Microstructures // Universum: Technical Sciences. – 11.2024 – No. 11(128).