Simulation Report on CLAHE and GAP Pixel Detection in Polymeric Images

# Abstract

This report presents a simulation and analysis pipeline for detecting GAP pixels in polymeric images. The pipeline employs CLAHE enhancement and pixel-wise evaluation based on grayscale values and neighborhood criteria. Results, including CSV pixel data and GAP-highlighted images, are generated and summarized herein.

# Introduction

Image analysis is crucial in materials science and engineering, especially for studying polymeric structures. Accurate enhancement and characterization of features within such images facilitate further scientific understanding. This project focuses on the automated detection of 'GAP' pixels—pixels of interest defined by specific grayscale thresholds and neighborhood criteria—in images prefixed with 'Poly\_'. The goal is to enhance these images, identify GAP pixels, and present the results in both tabular and visual formats.

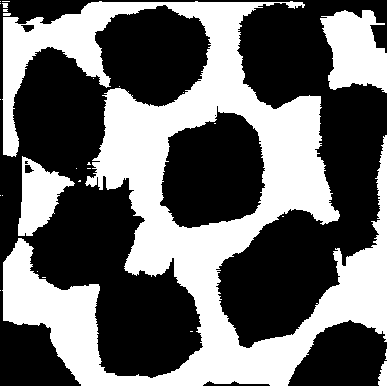
# Methods

The analysis pipeline consists of several stages:  
1. \*\*Image Selection & Enhancement:\*\* All images with filenames beginning with 'Poly\_' and in PNG or JPG format were selected from the specified input directory. Each image was processed using CLAHE (Contrast Limited Adaptive Histogram Equalization) with clipLimit=3 and tileGridSize=(10, 10) to improve local contrast and feature visibility.  
2. \*\*Grayscale Conversion & GAP Detection:\*\* The enhanced images were converted to grayscale. For each pixel, a 'GAP' flag was assigned. A pixel is considered a GAP pixel if its grayscale value lies between 1 and 150 (inclusive), and in at least one of the four main directions (up, down, left, right), there exist 25 contiguous neighboring pixels meeting the same grayscale condition.  
3. \*\*Data Output:\*\* For each image, two outputs were produced:  
 - A CSV file listing every pixel's coordinates, grayscale value, and GAP flag.  
 - A new binary PNG image where GAP pixels are shown in black (RGB 0,0,0) and all others in white (RGB 255,255,255).  
Automation and processing were implemented in Python, leveraging OpenCV for image enhancement, Pillow for grayscale conversion, and standard Python libraries for data management.

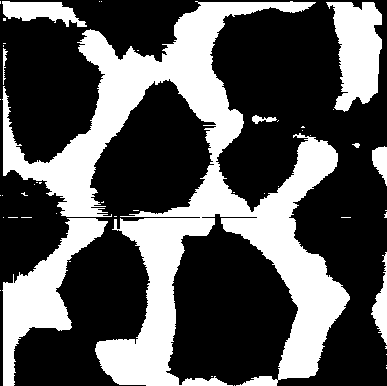
# Results

The pipeline processed all eligible images, generating detailed CSV tables and visual binary maps of GAP pixel locations. Below are the GAP-highlighted images for each processed input, demonstrating the effectiveness of the enhancement and detection algorithms.

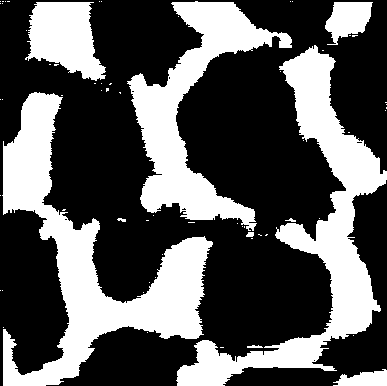
Poly\_01\_gap.png



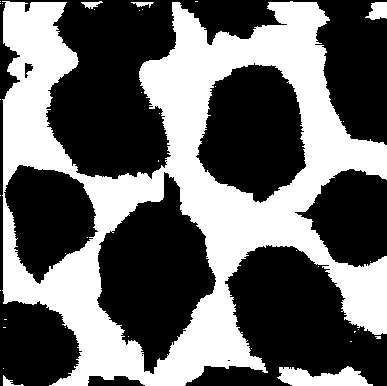
Poly\_02\_gap.png



Poly\_03\_gap.png



Poly\_04\_gap.png



Poly\_05\_gap.png

