SoC: Image Processing and Object Detection

Isha Yadav

June 2023

1 Introduction

This report presents the findings of a study focused on calculating the porosity of an image using image processing techniques. The objective of this analysis is to quantify the presence of void spaces or pores within the image, providing insights into the distribution and connectivity of these regions. The approach employed in this study utilizes the OpenCV library, a widely-used computer vision library, to perform image processing operations. The code employs a series of steps to process the image and estimate the porosity percentage.

2 Algorithm

- Start
- Accept the path of the image to be processed (imgPath)
- Define a function named porosityCal that takes imgPath as a parameter
- Inside porosityCal function:
  - Read the image using cv.imread() and store it in the img variable
  - Convert the image to grayscale using cv.cvtColor() and store it in the grayedImg variable
  - Apply Gaussian blur to the grayscale image using cv.GaussianBlur() with a kernel size of (7, 7) and store the result in the blurred variable
  - Apply a threshold to the blurred image using cv.threshold() to obtain a binary image, and store it in binary_image.
  - Create a 3x3 kernel using np.ones() and store it in the kernel variable
  - Perform morphological closing on the binary image using cv.morphologyEx() with cv.MORPH_CLOSE as the operation and kernel as the kernel, and store the result in the image variable
  - Find contours in the image using cv.findContours() and store the contours in the contours variable

  - Initialize a variable total_area as 0
  - Iterate over each contour in the contours list: Calculate the area of the contour using cv.contourArea() and add it to the total_area
  - Calculate the total area of the image by multiplying the height and width of the original image and store it in the image_area variable
  - Calculate the porosity percentage by dividing the total_area by image_area, multiplying by 100, and store it in the porosity variable
  - Return the porosity value.
- Stop

3 Code

```
import cv2 as cv
import numpy as np


def porosityCal(imgPath):
    img = cv.imread(imgPath)
    # cv.imshow('balls',img)

    grayedImg = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    #cv.imshow('Gray Image', grayedImg)

    blurred = cv.GaussianBlur(grayedImg, (7, 7), cv.BORDER_DEFAULT)
    #cv.imshow('Blur', blurred)

    _, binary_image = cv.threshold(blurred, 150, 255, cv.
                                   THRESH_BINARY)

    kernel = np.ones((3, 3), np.uint8)
    image = cv.morphologyEx(binary_image, cv.MORPH_CLOSE, kernel)

    contours, hierarchies = cv.findContours(image, 1, 2)  # cv.
                                   RETR_LIST, cv.
                                   CHAIN_APPROX_SIMPLE)

    total_area = 0
    for contour in contours:
        total_area += cv.contourArea(contour)

    # Calculate the porosity percentage
    image_area = img.shape[0] * img.shape[1]
    porosity = (total_area / image_area) * 100
    return porosity

imgPath = 'SampleImage.jpg'
porosity = porosityCal(imgPath)
print('Porosity: ',porosity)

cv.waitKey(0)
cv.destroyAllWindows()
```

3.1 Code Explanation

- Image Processing Steps:
  - The code starts by loading an image specified by the imgPath variable using cv.imread().
  - The loaded image is then converted to grayscale using cv.cvtColor() function.
  - Gaussian blur is applied to the grayscale image using cv.GaussianBlur() to reduce noise and smoothen the image.
  - A binary image is created by applying a threshold to the blurred image using cv.threshold(). Pixels with intensity values above the threshold are set to white (255), and pixels below the threshold are set to black (0).
  - Morphological closing operation is performed on the binary image using cv.morphologyEx() with a defined kernel size of (3, 3) to close small gaps in the image and enhance connectivity of regions.
- Contour Analysis and Porosity Calculation:
  - Contours are extracted from the processed image using cv.findContours().
  - The code iterates over the detected contours and calculates the area of each contour using cv.contourArea().
  - The total area of all contours is accumulated in the total_area variable.
  - The porosity percentage is then calculated by dividing the total area by the total area of the original image and multiplying by 100.
- Porosity Result:
  - The calculated porosity percentage is stored in the porosity variable and printed using print() function. In the provided code, the porosity value is printed as 'Porosity: {porosity_value}'.

4 Result

The porosity value of metal sample was calculated using image processing and object detection algorithm. The porosity of the given image is 33.26%.