24-678: Computer Vision for Engineers

Carnegie Mellon University

PS5

Due: **10/25/2024 (Fri) 5 PM @ Gradescope**

Issued: **10/2/2024 (Wed)**

Weight: **5% of total grade**

Note:

PS5-1 Binary image processing – detecting blobs, contours, and central axes

Unlike the Pantheon in Rome, which has lasted thousands of years, the lifespan of modern concrete structures is roughly 50-100 years. The regular and thorough inspection and monitoring of concrete structures are essential to avoid catastrophic failures such as the collapse of a residential tower in Surfside, Florida (Figure 1 left) and the Morandi Bridge in Genoa, Italy (Figure 1 right).

Figure 1: Concrete structure failures

In the next several decades, the US needs to fix existing infrastructure, nearly all of which is concrete. According to the American Society of Civil Engineers, the amount of money needed is estimated to be roughly $6 trillion.

To prioritize and plan for the repair operations of concrete infrastructure, we need to inspect and monitor concrete structures on a regular basis. Since a manual inspection is time-consuming, labor-intensive, and dangerous for workers, the mobile-robotics and machine-learning research communities have been making significant efforts to automate the inspection process.

Autonomous drones have been deployed to take images of concrete walls (https://www.youtube.com/embed/N-Xhbgwa4hI, https://www.youtube.com/embed/f6rbAVIwvnk), and various machine-learning algorithms have been developed to detect cracks automatically.

In this problem set, you will write a program that takes as input images of concrete walls with cracks and then finds the blob contours and crack central axes. <u>You may use any OpenCV built-in functions.</u>

Your program should take as input a binary image, “wall1.png,” shown in Figure 2 (b). This is a binary image generated by a machine-learning method, showing pixels that may be part of a crack as black pixels. Your program should:

(1) Apply dilation and erosion operations to the input image to turn black regions into clean blobs. Name the new image, “wall1-blobs.png,” and save the file,

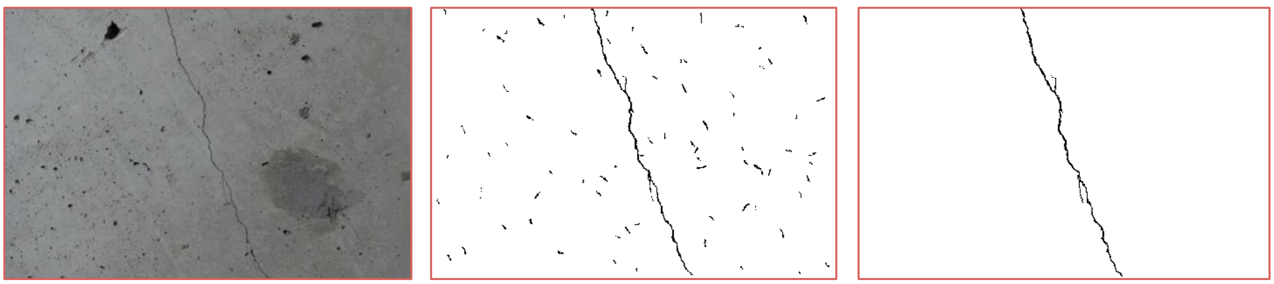
(2) Detect blobs and their contours; create an image that shows blob contours with randomly assigned colors, name the image,”wall1-contours.png,” and save the file,

(3) Define your own thresholds for detecting cracks, and apply the thresholds to the blobs to keep only the blobs that are likely to be actual cracks,

(4) For each blob that is likely to be an actual crack, find its central axis by thinning operations. Create an image that shows the central axis, name the image, “wall1-cracks,png,” and save the file,

Also, apply your code to the other image, “wall2.png,” shown in Figure 2 (e).

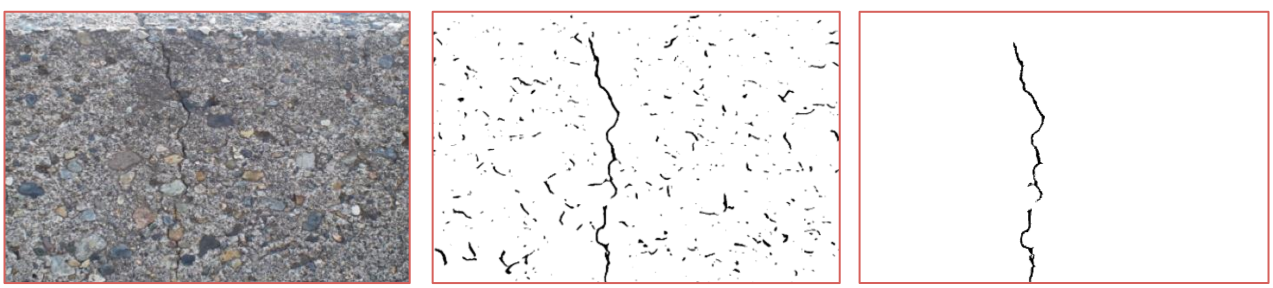
Note that the sample crack images shown in Figures 2 (c) and 2 (f) are prior to thinning operations.



(a) Wall 1 - original image

(b) Wall 1 – identified candidate crack pixels

(c) Wall 1 – detected cracks



(d) Wall 2 - original image

(e) Wall 2 – identified candidate crack pixels

(f) Wall 2 – detected cracks

Figure 2: Input images, candidate pixels, and detected cracks

Submission

To prepare for the submission of your work on Gradescope, create:

(1) a folder called “ps5-1,” that contains the following files:

- source code file(s)
- improved images created by your program:
    - wall1-blobs.png, wall2-blobs.png

    - wall1-contours.png, wall2-controus.png
    - wall1-cracks.png, wall2-cracks.png
- "readme.txt" file that includes:
    - Operating system
    - IDE you used to write and run your code
    - The number of hours you spent to finish this problem

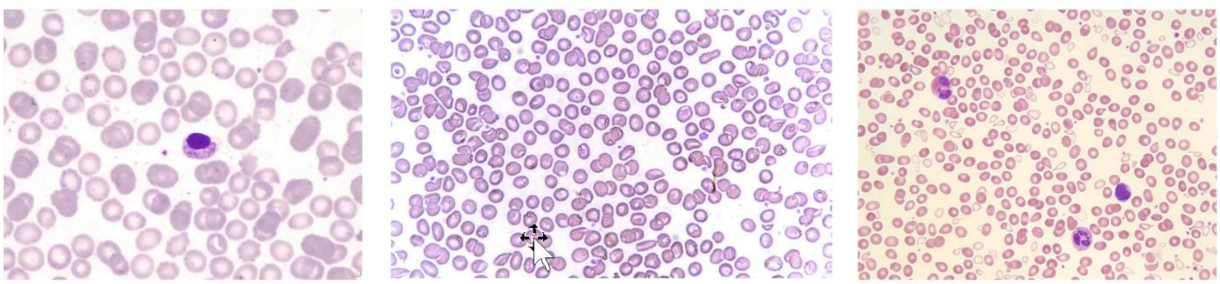
(2) a PDF file that contains the printouts and screenshots of all the files in the ps5-1 folder. Include an explanation of what thresholds you used to detect cracks. (Include, if any, the mathematical derivation and/or description of your method in the PDF file. Handwritten notes should be scanned and included in the PDF file.)

PS5-2 Blood-Cell Image Processing – RGB > Binary > Blobs > Blob Sizes and Principal Axes > Visualization

Analyzing blood cell images by image-processing techniques is crucial for several reasons:

- **Disease Detection**: It helps identify diseases like leukemia, malaria, and anemia by spotting abnormal cells early on.
- **Treatment Monitoring**: Doctors can track how well a treatment works by observing blood cell changes over time.
- **Research**: It provides valuable data for medical research, helping to understand diseases better and develop new treatments.
- **Precision**: Automated analysis reduces human error, ensuring more accurate and consistent results.
- **Efficiency**: It speeds up the diagnostic process, allowing for quicker decision-making in clinical settings.

Figure 3 below shows typical blood cell images: blood-cells1.jpg, blood-cells2.jpg, and blood-cells3.jpg.



blood-cells1.jpg blood-cells2.jpg blood-cells3.jpg

Figure 3: Blood cell images

Convert the three blood-cell images to binary black-and-white images, detect blobs, caliculate blob sizes and principal axes, and generate images called blood-cells1-catalog.jpg, blood-cells2-catalog.jpg, and blood-cells3-catalog.jpg that list all the detected blobs in order of size, as shown in Figure 4. In your catalog images, the major axis of each blob should be horizontal.

<u>You may use any OpenCV built-in functions including the ones listed on this webpage: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html.</u>

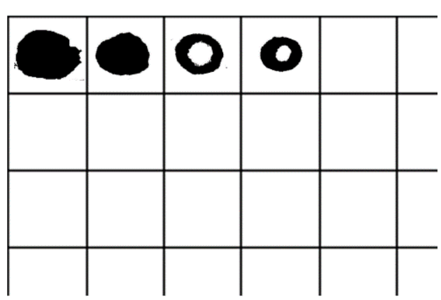


Figure 4: Blood cell images

Submission

To prepare for the submission of your work on Gradescope, create:

(1) a folder called "ps5-2," that contains the following files:
- source code file(s)
- binary images, blob images, and catalog images
    - blood-cells1-binary.jpg, blood-cells1-blobs.jpg, blood-cells1-catalog.jpg
    - blood-cells2-binary.jpg, blood-cells2-blobs.jpg, blood-cells2-catalog.jpg
    - blood-cells3-binary.jpg, blood-cells3-blobs.jpg, blood-cells3-catalog.jpg
- "readme.txt" file that includes:
    - Operating system
    - IDE you used to write and run your code
    - The number of hours you spent to finish this problem

(2) a PDF file that contains the printouts and screenshots of all the files in the ps5-2 folder. Include an explanation of what thresholds you used to detect cracks. (Include, if any, the mathematical derivation and/or description of your method in the PDF file. Handwritten notes should be scanned and included in the PDF file.)

Submit your work on Gradescope

Submit two files on Gradescope – replace "andrewid" with your own Andrew ID:

(1) **andrewid-ps5-files.zip** – this ZIP file should contain the ps5-1 folder and all the files requested.

(2) **andrewid-ps5-report.pdf** – this PDF file serves as the report of your work, and it should contain the printouts and screenshots of all the files in the "ps5-1" and "ps5-2" folders. (Include, if any, the mathematical derivation and/or description of your method in the PDF file. Handwritten notes should be scanned and included in the PDF file.)

Please organize pages with section titles and captions to make the report easy to read.