AMERICAN UNIVERSITY OF ARMENIA
College of Science and Engineering
CS 260 / CS 371 Image Processing

Project 1 – Lines of Text in Stone Inscriptions

Preliminary Submission Deadline: Wednesday, **November 19 2025**, no later than 22:00
Final Submission Deadline: Monday, **November 24 2025**, no later than 22:00 **SHARP**
Textbook: W. Burger, M. J. Burge. "Digital Image Processing: An Algorithmic Introduction using Java", 2nd ed., 2016
Reading: Chapter 5, Chapter 6, Chapter 8, Chapter 9, Chapter 10, Chapter 18, Chapter 19.

The Project aims at detection / recognition of stone inscriptions. Due to weathering, vandalism, erosion, and the complexity of ancient scripts, many of these texts are hard to read. Study, apply and test Image Processing methods and algorithms to detect Armenian inscriptions.

The shapes of Armenian letters are based on vertical strokes. Based on this observation, implement and test a pipeline outlined below. Implement the steps as ImageJ plug-ins or menu commands, and save them using the ImageJ macro recorder. In addition to the recorded macro(s), submit the code of all implemented plug-ins. Other image processing and programming environments may be used only for testing purposes.

#	Description	Submission
0	Create a Google drive and share its link / send invitation to skhachat@aua.am and eduard_grigoryan@edu.aua.am. Make sure your name is explicitly reflected in the title. All project deliverables will be collected in its **\Lines** subfolder.	
1	Select several images of tombstones (tapanakars), crossstones (khachkars) or monuments with Armenian inscriptions. Make them grayscale and process as outlined below.	The image files and source links
2	Detect the vertical edges (both east and west edges) using Edge Detection Operators (chapter 6). Make the images of the detected edges binary. Combine the detected east and west edges in a binary image.	Kernels of the edge operators and the images of east, west and combined edges
3	Strengthen the detected vertical edges by applying 1 pixel-wide dilation in horizontal direction and 1 pixel-wide erosion in vertical direction (chapter 9).	Structuring elements and the images after dilation and erosion
4	Denoise the image of the strengthened vertical edges by applying linear and / or nonlinear filters of unit radius (chapter 5). Make sure the image stays binary after denoising.	The filter(s) and kernel(s), and the denoised image of vertical edges
5	Armenian text can be considered as a binary region that has a high-frequency structure. To detect such regions and blur, apply the Bandpass Filter. Try different values for large structures and small structures limits (for example, 40 and 30	The filtered image

	pixels respectively) to produce horizontally aligned regions that resemble words or entire text lines. (chapter 19).	
6	If necessary, use the filtered image from **step 5** as a binary mask for the denoised image of the vertical edges from **step 4** by applying AND operation	The masked image
7	If having implemented **step 6**, apply the same Bandpass Filter from **step 5** to the masked image.	The filtered masked image
8	Analyze the particles in the filtered image from **step 5** or **step 7** and show the fitting ellipses (chapter 10).	The image with fitting ellipses of the analyzed particles
9	Skeletonize the filtered image from **step 5** or **step 7** or the fitting ellipses from **step 8** (chapter 9).	The images after the skeletonization
10	Detect the horizontal lines by applying Hough Transform (chapter 7). Use Hough_**Transform.java PlugInFilter**. The width of the resulting image corresponds to the range of angles from 0 to π. The horizontal lines are identified by the angle $\pi/2$.	The result of the Hough Transform
11	Convert the image of the Hough Transform to grayscale format and apply a threshold to its region around angle $\pi/2$ to locate the horizontal lines.	Binary regions filtered by the threshold from the image of the Hough Transform.
12	Write a plugin that locates on the original image the horizontal lines detected by the Hough Transform. Check, if these lines represent the text lines.	The plugin and the image with the located text lines
13	Compute the horizontal projection of the binary image from **step 4** and locate the minimum points around the horizontal lines detected **in step 12**.	Image with the top and bottom boundaries of the detected text lines

Submission Conditions:

1. This is an individual assignment. Identical or similar submissions / files / results / reports / diagrams etc. will be disqualified – both the source(s) and receiver(s) will collect 0 point.
2. Group work will be accepted only if all group members are explicitly indicated in the submission. The individual contribution of each group member must also be explicitly stated, including all reasons of forming the group.
3. The submission deadline is rigidly strict. Submissions will be checked immediately after the stated deadlines. Submit even an unfinished work to get points and feedback. Late submissions will be disqualified and collect 0 point.
4. Not only precise solutions, but also free-format descriptions of ideas, difficulties, algorithms, simplifications, assumptions, etc. may be submitted.
5. You are welcome to use external sources, but all of them must be explicitly acknowledged and the links / references provided.