# Task 5

## b) Thinning and thickening

To perform morphological operations *thinning* and *thickening* on an image, one must make sure that the image is binary. An RGB image must be converted to greyscale, and a greyscale image can be directly converted to binary, *B* . A threshold value, *T*, has to be specified in the formula for converting a greyscale image to binary, shown in Eq. 5.1.

Eq. 5.1

After converting the input image to binary, the morphological operations can be applied. If n = Inf, thinning eliminates pixels in objects without holes, and objects with holes shrink halfway between each hole and the outer boundary to a connected ring. Thickening is the process, much like dilation or closing, to grow selected pixels in the foreground of binary images. Thickening objects will result in eight-connected objects by adding pixels to the exterior of unconnected objects until connection, also with If n = Inf. Thinning and thickening preserve the *Euler number*.

The following commands can be used in MATLAB to perform thinning and thickening:

L\_thin = bwmorph(L\_binarized, ‘thin’);

L\_thick = bwmorph(L\_binarized, ‘thicken’);

The function bwmorph uses the thinning algorithm as described by Lam, Lee and Suen (1992) and shown in the conditions in Eq. 5.2, Eq. 5.3 and Eq. 5.4.

1. In the first part of the iteration, delete pixel *p* if conditions/equations 5.2, 5.3 and 5.4 are satisfied.
2. In the second part of the iteration, delete pixel *p* if conditions 5.2, 5.3 and 5.5 are satisfied.

Eq. 5.2

where,

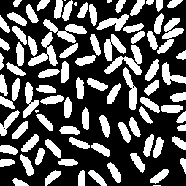
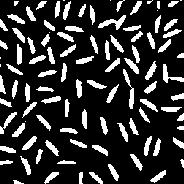
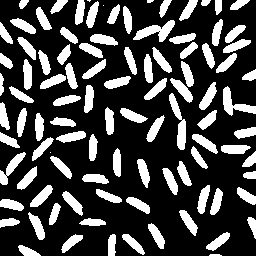
Eq. 5.3

where,

Eq. 5.4

Eq. 5.4

Fig. 5.1 shows the effect of thinning and thickening, where n is not specified so the algorithms do not continue iterating.



original

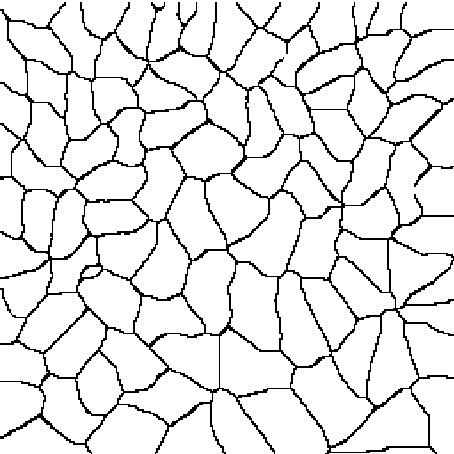
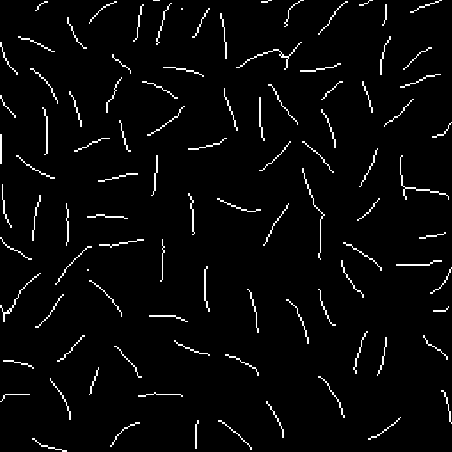
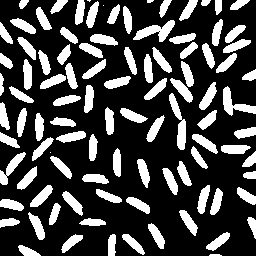
binary after thinning, T = 100

binary after thickening, T = 100

**Fig. 5.1**: Effect of thinning and thickening on binary version (T = 100) of rice image.

Fig. 5.2 shows the same binary image again after the same morphological operations, but now with a specified number of iterations, n = 5.

The next figure, Fig. 5.3, shows the same image again, but now infinitive iterations of the algorithm until the morphological operation is not possible. Note, how thinning was already at its maximum with n = 5 (Fig. 5.2), and that thickening continues until impossible (Fig. 5.3).

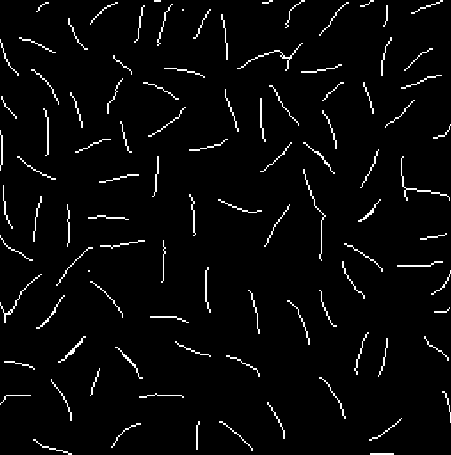
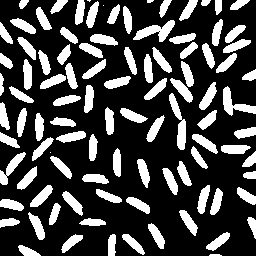


original

binary after thinning, T = 100, n = Inf

binary after thickening, T = 100, n = Inf

**Fig. 5.3**: Effect of thinning and thickening (infinite iterations of the operations, n = Inf) on binary (T = 100) rice image until operation is impossible.



original

binary after thinning, T = 100, n = 5

binary after thickening, T = 100, n = 5

**Fig. 5.2**: Effect of thinning and thickening (five iterations of the algorithm, n = 5) on binary (T = 100) rice image.

The last example is given with a binary image of the letter ‘J’ (j.png), threshold value *T = 100*, and the two morphological operations are conducted with n = Inf for thinning and n = 5 for thicken to visualize the changes best. The results are shown in Fig. 5.4.

original

binary after thinning, T = 100, n = Inf



binary after thinning, T = 100, n = 5

**Fig. 5.4**: Effect of thinning and thickening on the binary (T = 100) image of the letter ‘J’.

## Source Code

Source code of task5.m is added here. Visit MATLAB code for functions.

**task5.m**

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Title: Thinning and thickening

% Author: Samir Habibi

% Rev. Date: 24/11/2020

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear; % Delete all variables.

close all; % Close all windows.

clc; % Clear command window.

% Ask user for file by presenting options with menu() command.

fileChoice = menu('File', 'Letter J', 'Rice', 'Choose own');

% Use switch() to read file based on user's choice (fileChoice).

switch (fileChoice)

case 1

filename = ('j.png');

L = imread(filename);

case 2

filename = ('rice.jpg');

L = imread(filename);

case 3

filename = uigetfile('');

L = imread(filename);

end % End the switch-statement after obtaining image.

% Call function to inspect image for multiple colour bands, if so, convert

% to grey. The image L is used for the input argument.

L = inspectFileDimension(L);

% Call function to binarize an image before performing morphological

% operations, even if image is already binary because it would not change.

% The image L is used again for the input argument.

L\_binarized = binarizeImage(L);

% Use function bwmorph() to perform morphological operations (thinning and

% thicken) on binarized version of image L.

% 'n' is specified to 5 for thickening, and to Inf for thinning.

L\_thick = bwmorph(L\_binarized, 'thicken', 5);

L\_thin = bwmorph(L\_binarized, 'thin', Inf);

% Display three images in one figure: original/binary version,

% and version after thinning and thickening.

figure;

subplot(1, 3, 1);

imshow(L\_binarized);

title('Original image');

subplot(1, 3, 2);

imshow(L\_thin);

title('After thinning');

subplot(1, 3, 3);

imshow(L\_thick)

title('After thickening');

## References

Lam, L., Lee, S.W, and Suen, C. Y., 1992. Thinning Methodologies-A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [e-journal] 14(9), pp. 869-885. Available through: <https://pdfs.semanticscholar.org/0404/ba243ecbb8efc6bcb07a754b6f8770856131.pdf> [Accessed 23 November 2020].

## Bibliography

MathWorks, 2015. *bwmorph*. [online] Available at: <https://nl.mathworksom/help/images/ref/bwmorph.html > [Accessed 22 November 2020].

TheAILearner, 2020. *Thinning and Thickening*. [online] Available at: <https://theailearner.com/2019/07/31/thinning-and-thickening/> [Accessed 2 December 2020].