Assignment A8: Image Segmentation

CS 4640
Fall 2019

Assigned: October 2019

Due: 5 December 2019

Note: DO NOT USE SCRIPTS. No function should write to the interpreter, draw, read or write files, etc., unless explicitly required in assignment. Headers must be indented correctly.

1. Develop an algorithm which uses kmeans to segment the characters in an image in the following way:

- extract each connected component
- regularize the connected component into a 25x15 window
- create a set of points (nx25*15) for the n components where each row is the linearized 25x15 window of the normalized character
- use kmeans to find a reasonable number of segments

Report the results of this method in terms of percentage characters correctly recognized (use the im45 data: chars45.mat, truth.mat).

```
function chars = CS4640_kmeans_seg(im)
% CS4640_kmeans_seg - use kmeans to segment characters
% On input:
%     im (MxN array): binary image
```

```
% On output:
%     chars (MxN array): characters are classed according to their
%     index in
%        the templates data; if not a recognized character, then
%        assigned 0
% Call:
%     ccc = CS4640_kmeans_seg(mask);
% Author:
%     <Your name>
%     UU
%     Fall 2019
%
```

2. The problem here is to use the watershed method with some other analysis to produce the best possible set of character segmentations for a binary image. This requires some assumptions:

- The characters are part of the foreground.
- There will be enough words along horizontal lines to allow the determination of the most common character height and width.
- Characters may be disconnected or eroded in the image.

Develop your own algorithm to estimate the height and width of the characters. At a minimum, find the most common size; if possible find others. In addition, find the supporting lines (geometric) for the lowest parts of any lines (textual) of text. Provide the function *CS4640_char_lines* described below to extrsct these properties.

```
function [dims,rows] = CS4640_char_lines(im)
% CS4640_char_lines - find heights, widths and baseline for characters
% On input:
%     im (MxN array): binary image
% On output:
%     dims (nx2 array): for n connected components:
%          dims(cc,:) = [height,width]
%     rows (kx4 array): for k baselines:
%          rows(r,:) = [r1,c1,r2,c2] -- two points on line
```

```
% Call:
%     [dims,rows] = CS4640_char_lines(mask);
% Author:
%     T. Henderson
%     UU
%     Fall 2019
%
```

Develop your own watershed function based on the algorithm given in class. Use it to segment the image. Develop the function *C4640_watershed* as described below.

```
function L = CS4640_watershed(im)
% CS4640_watershed - compute watershed (basins and dams) in image
% On input:
%     im (MxN array): binary image
% On output:
%     L (MxN array): basins and dams
%       L(r,c) in [1,n] is a basin
%       L(r,c) = 0 is background
%       L(r,c) < 0 is a dam pixel
% Call:
%     L = CS4640_watershed(mask);
% Author:
%     T. Henderson
%     UU
%     Fall 2019
%
```

Finally, develop a high-level function that uses *CS4640_char_lines* and *CS4640_watershed* to produce a labeled output image of the characters (labeled 1-50) of an input image.

```
function imc = CS4640_segw(im)
% CS4640_segw - segment and label characters in image
% On input:
%     im (MxN array): binary image
% On output:
%     imc (MxN array): character segmentation
%       L(r,c) in [1,50]: character label (from templates)
```

```
%       L(r,c) = 0 is non-character
% Call:
%     imc = CS4640_segw(im);
% Author:
%     T. Henderson
%     UU
%     Fall 2019
%
```

Discuss performance on im45 in terms of percentage correct labels.