Assignment A6: Object Features

CS 6640
Fall 2020

Assigned: 27 October 2020

Due: 19 November 2020

For this problem, handin a report of your work, as well as the Matlab .m files for the functions described by the headers below. Instructions given in any A6_criteria document must be followed as patr of the assignment.

Some notes:

- Indent headers correctly (5 spaces indented lines)
- Do not exceed 72 characters per source line

None of the functions should write to the interpreter, draw, etc.

1. Develop a character classification method based on the shape parameters in Table 9.1 in the text (there are seven of those) and the Euler number; apply these to character classification in the test images *drawings_46* and *drawings_56*. This involves 2 Matlab functions which you must write, *CS6640_params_model* and *CS6640_params_classify*, which produces shape parameter models and use them to classify, respectively.

```
function P = CS6640_params_model(im)
% CS6640_params_model - build shape parameter model
% On input:
%     im (MxN array): binary image
% On output:
%     P (nx8 array): shape parameter values for n connected components
%       (k,1): form factor
```

```
%       (k,2): roundness
%       (k,3): aspect ratio
%       (k,4): solidity
%       (k,5): extent
%       (k,6): compactness
%       (k,7): convexity
%       (k,8): Euler number
% Call:
%     P = CS6640_params_model(im);
% Author:
%     <Your name>
%     UU
%     Fall 2020
%


function imo = CS6640_params_classify(im,P)
% CS6640_params_classify - classify with shape parameter models
% On input:
%     im (MxN array): binary image
%     P (nx8 array): shape models
%       (k,1): form factor
%       (k,2): roundness
%       (k,3): aspect ratio
%       (k,4): solidity
%       (k,5): extent
%       (k,6): compactness
%       (k,7): convexity
%       (k,8): Euler number
% on output:
%     imo (MxN array): classified objects (by model index)
% Call:
%     imo = CS6640_params_classify(im,P);
% Author:
%     <Your name>
%     UU
%     Fall 2020
%
```

2. Develop a character classification method based on the radial Fourier expansion, and ap-

ply these to character classification in the test images *drawings_46* and *drawings_56*. This involves 2 Matlab functions which you must write, *CS6640_RFE_model* and *CS6640_RFE_classify*, which produces shape parameter models and use them to classify, respectively.

```
function s = CS6640_RFE_model(im,n)
% CS6640_RFE_model - create Radial Fourier Expanson shape model
% On input:
%     im (MxN array): binary image
%     n (int): number of coefficients to keep
% On output:
%     s (nx1 vector): RFE shape model
% Call:
%     s = CS6640_RFE_model(classes==1,10);
% Author:
%     <Your name>
%     UU
%     Fall 2020
%
```

```
function imo = CS6640_RFE_classify(im,models)
% CS6640_RFE_match - classify with Radial Fourier Expanson shape
% models
% On input:
%     im (MxN array): binary image
%     models (pxq array): p RFE models with q coeffs each
% On output:
%     imo (MxN array): classified image (by model indexes)
% Call:
%     imo = CS6640_RFE_classify(letterst,models);
% Author:
%     <Your name>
%     UU
%     Fall 2020
%
```

3. Develop a character classification method based on the Hu moment invariants, and apply these to character classification in the test images *drawings_46* and *drawings_56*. This involves 2 Matlab functions which you must write, *CS6640_Hu_model* and *CS6640_Hu_classify*,

which produces shape parameter models and use them to classify, respectively. You also need to develop two helper functions, *CS6640_Hu_M* and *CS6640_Hu_eta*.

```
function m = CS6640_Hu_model(im)
% CS6640_Hu_model - build Hu shape models
% On input:
%     im (MxN array): binary image
% On output:
%     m (nx7 array): Hu moment values for n connected components
% Call:
%     m = CS6640_Hu_model(im);
% Author:
%     <Your name>
%     UU
%     Fall 2020
%
```

```
function imo = CS6640_Hu_classify(im,models)
% CS6640_Hu_classify - classify with Hu shape models
% On input:
%     im (MxN array): binary image
%     models (nx7 array): n Hu moment models
% On output:
%     imo (MxN array): classified connected components
% Call:
%     imo = CS6640_Hu_classify(im,m);
% Author:
%     <Your name>
%     UU
%     Fall 2020
%
```

```
function Mpq = CS6640_Hu_M(p,q,im)
% CS6640_Hu_M - compute p,q central moment
% On input:
%     p (int): power for x components
%     q (int): power for y components
%     im (MxN array): binary image
% On output:
```

```
%     Mpq (float): p-q th moment
% Call:
%     M11 = CS6640_Hu_M(1,1,im);
% Author:
%     <Your name>
%     UU
%     Fall 2020
%


function eta_pq = CS6640_Hu_eta(p,q,im)
% CS6640_Hu_eta - compute normalized p,q central moment
% On input:
%     p (int): power for x components
%     q (int): power for y components
%     im (MxN array): binary image
% On output:
%     eta_pq (float): normalized p-q th moment
% Call:
%     e11 = CS6640_Hu_eta(1,1,im);
% Author:
%     <Your name>
%     UU
%     Fall 2020
%
```

4. Develop a function, *CS6640_PCA_kmeans*, which reduces the dimensionality of a set of vectors using PCA methods, and then classifies them using kmeans.

```
function [cidx,ctrs] = CS6640_PCA_kmeans(T,p,k)
% CS6640_PCA_kmeans - reduce dimensionality and classify with kmeans
% On input:
%     T (nxd array): n d-vectors
%     p (float in [0,1]): percent of eignevectors to use
%     k (int): numer of classes for kmeans
% On output:
%     cidx (nx1 vector): class indexes for vectors
%     ctrs (kxq array): centers of q-vectors [q is floor(p*d)]
% Call:
```

```
%     [cidx,ctrs] = CS6640_PCA_kmeans(T,.2,6);
% Author:
%     <Your name>
%     UU
%     Fall 2020
%
```