Homework 5

1. A description of what your idea:

The first experiment that I attempted was applying dilation to the image in order to make the lines thicker, since I believed this would help with the recognition rate (due to the fact that many of the characters were disjoint causing the low size threshold to pick up more connected components than characters in the image). Also I had to make sure that I had an accurate list of true labels, since I had gotten that wrong in the previous assignment. I sorted the Features matrix by maxr to make sure the accurate list of true labels was correct. In order to sort the Features matrix by maxr I created another boundingBox called boundingBoxR, which tagged maxr onto the end of the Features matrix temporarily until Features is sorted. Then, after every image I planned on eroding the image and then dilating the image in order to remove any blur on the outside of the characters.

1. The code you wrote to implement it

function RunRecog(filename)

%A training

im = imread('a.bmp');

se = strel([111;111;111]);

im = imerode(im, se);

im = imadjust(im);

figure

imagesc(im)

im1 = reshape(im,prod(size(im)),1);

th = 220;

im2 = im;

im2(im>=th) = 0;

im2(im<th) = 1;

L = bwlabel(im2);

boundingBox

AFeatures = Features;

idA = ones(size(AFeatures(:,1)));

%D training

im = imread('d.bmp');

se = strel([111;111;111]);

im = imerode(im, se);

im = imadjust(im);

im1 = reshape(im,prod(size(im)),1);

th = 220;

im2 = im;

im2(im>=th) = 0;

im2(im<th) = 1;

L = bwlabel(im2);

boundingBox

DFeatures = Features;

idD = ones(size(DFeatures(:,1)))\*2;

%M training

im = imread('m.bmp');

se = strel([111;111;111]);

im = imerode(im, se);

im = imadjust(im);

im1 = reshape(im,prod(size(im)),1);

th = 220;

im2 = im;

im2(im>=th) = 0;

im2(im<th) = 1;

L = bwlabel(im2);

boundingBox

MFeatures = Features;

idM = ones(size(MFeatures(:,1)))\*3;

%N training

im = imread('n.bmp');

se = strel([111;111;111]);

im = imerode(im, se);

im = imadjust(im);

im1 = reshape(im,prod(size(im)),1);

th = 220;

im2 = im;

im2(im>=th) = 0;

im2(im<th) = 1;

L = bwlabel(im2);

boundingBox

NFeatures = Features;

idN = ones(size(NFeatures(:,1)))\*4;

%O training

im = imread('o.bmp');

se = strel([111;111;111]);

im = imerode(im, se);

im = imadjust(im);

im1 = reshape(im,prod(size(im)),1);

th = 220;

im2 = im;

im2(im>=th) = 0;

im2(im<th) = 1;

L = bwlabel(im2);

boundingBox

OFeatures = Features;

idO = ones(size(OFeatures(:,1)))\*5;

%P training

im = imread('p.bmp');

se = strel([111;111;111]);

im = imerode(im, se);

im = imadjust(im);

im1 = reshape(im,prod(size(im)),1);

th = 220;

im2 = im;

im2(im>=th) = 0;

im2(im<th) = 1;

L = bwlabel(im2);

boundingBox

PFeatures = Features;

idP = ones(size(PFeatures(:,1)))\*6;

%Q training

im = imread('q.bmp');

se = strel([111;111;111]);

im = imerode(im, se);

im = imadjust(im);

im1 = reshape(im,prod(size(im)),1);

th = 220;

im2 = im;

im2(im>=th) = 0;

im2(im<th) = 1;

L = bwlabel(im2);

boundingBox

QFeatures = Features;

idQ = ones(size(QFeatures(:,1)))\*7;

%R training

im = imread('r.bmp');

se = strel([111;111;111]);

im = imerode(im, se);

im = imadjust(im);

im1 = reshape(im,prod(size(im)),1);

th = 220;

im2 = im;

im2(im>=th) = 0;

im2(im<th) = 1;

L = bwlabel(im2);

boundingBox

RFeatures = Features;

idR = ones(size(RFeatures(:,1)))\*8;

%U training

im = imread('u.bmp');

se = strel([111;111;111]);

im = imerode(im, se);

im = imadjust(im);

im1 = reshape(im,prod(size(im)),1);

th = 220;

im2 = im;

im2(im>=th) = 0;

im2(im<th) = 1;

L = bwlabel(im2);

boundingBox

UFeatures = Features;

idU = ones(size(UFeatures(:,1)))\*9;

%W training

im = imread('w.bmp');

se = strel([111;111;111]);

im = imerode(im, se);

im = imadjust(im);

im1 = reshape(im,prod(size(im)),1);

th = 220;

im2 = im;

im2(im>=th) = 0;

im2(im<th) = 1;

L = bwlabel(im2);

boundingBox

WFeatures = Features;

idW = ones(size(WFeatures(:,1)))\*10;

Features = [AFeatures; DFeatures; MFeatures; NFeatures; OFeatures; PFeatures; QFeatures; RFeatures; UFeatures; WFeatures];

id = [idA; idD; idM; idN; idO; idP; idQ; idR; idU; idW];

means = mean(Features);

stddev = std(Features);

for i=1:length(means)%size(means)

Features(:,i) = Features(:,i) - means(:,i);

Features(:,i) = Features(:,i) / stddev(:,i);

end

D = dist2(Features,Features);

figure

imagesc(D);

sortMatrix = D;

Prediction = [];

for i=1:length(D)

sortMatrix(i,:) = sort(sortMatrix(i,:));

p = sortMatrix(i,2);

idIndex = findIndex(D(i,:),p);

predictionID = id(idIndex,1);

Prediction = [Prediction; predictionID];

end

id = id';

predictionID = predictionID';

ConfMat = ConfusionMatrix(id, Prediction, 10);

figure

imagesc(ConfMat);

recognitionRate = (trace(ConfMat))/(sum(sum(ConfMat)))

%PART 4

im = imread(filename);

se = strel([111;111;111]);

im = imerode(im, se);

im = imadjust(im);

im1 = reshape(im,prod(size(im)),1);

th = 220;

im2 = im;

im2(im>=th) = 0;

im2(im<th) = 1;

L = bwlabel(im2);

boundingBoxR

Features = sortrows(Features, 7);

Features(:, 7) = [];

FeaturesTest = Features;

idTestArray = [];

for i=1:10

for j=1:7

idTestArray = [idTestArray; i];

end

end

%NORMALIZATION

for i=1:length(means)%size(means)

FeaturesTest(:,i) = FeaturesTest(:,i) - means(:,i);

FeaturesTest(:,i) = FeaturesTest(:,i) / stddev(:,i);

end

testDistance = dist2(FeaturesTest, Features);

figure

imagesc(testDistance);

testsortMatrix = testDistance;

testPrediction = [];

for i=1:length(testDistance)

testsortMatrix(i,:) = sort(testsortMatrix(i,:));

p = testsortMatrix(i,2);

idIndex = findIndex(testDistance(i,:),p);

predictionID = idTestArray(idIndex, 1);

testPrediction = [testPrediction; predictionID];

end

idTestArray = reshape(idTestArray,70,1);

testPrediction = testPrediction';

ConfMatrix = ConfusionMatrix(idTestArray, testPrediction, 10);

figure

imagesc(ConfMatrix);

recognitionRate = (trace(ConfMatrix))/(sum(sum(ConfMatrix)))

boundingBoxR – Bounding Box function only used for test.jpg in order to output coordinates and bounding box image.

Nc=max(max(L));

f=figure;

imagesc(L);

hold on;

Features=[];

fileID = fopen('results.txt', 'w');

Results=[];

for i=1:Nc;

[r,c]=find(L==i);

maxr=max(r);

minr=min(r);

maxc=max(c);

minc=min(c);

rectangle('Position',[minc,minr,maxc-minc+1,maxr-minr+1],'EdgeColor','w');

height = maxr - minr;

width = maxc - minc;

threshold=7;

if( height >threshold && width > threshold)

cim=im2(minr-1:maxr+1,minc-1:maxc+1);

[centroid, theta, roundness, inmo] = moments(cim, 1);

Features=[Features; theta, roundness, inmo, maxr];

Results=[Results; minr, minc, height, width];

end

end

fileID = fopen('results.txt','w');

idTestArray = [];

for i=1:10

for j=1:7

idTestArray = [idTestArray; i];

end

end

Results = sortrows(Results, [1, 2]);

arrayResults = [Results idTestArray];

arrayResults = sortrows(arrayResults, [5, 2]);

for ii = 1:size(arrayResults,1)

fprintf(fileID,'%d, %d, %d, %d, %d',arrayResults(ii,:));

fprintf(fileID,'\n');

end

fclose(fileID);

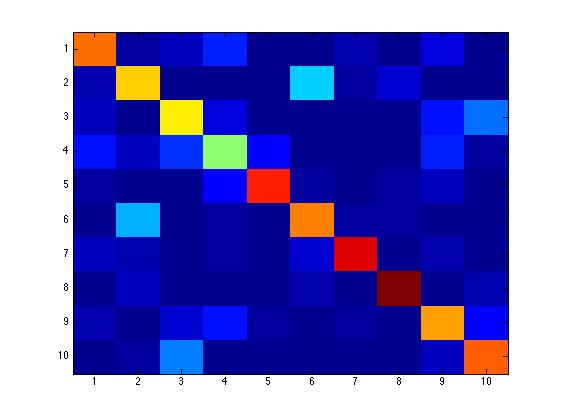
for i=1:size(arrayResults(:,1))

text(arrayResults(i,2)-10, arrayResults(i,1)-5, num2str(arrayResults(i,5)));

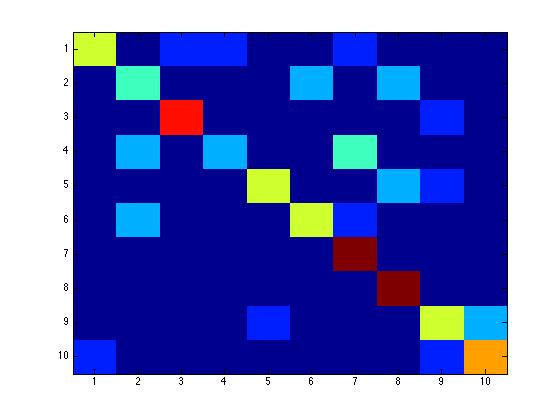
end

saveas(f,'results.jpg','jpg')

1. The results on the training data, as was done in part III of

assignment 4, in terms of a confusion matrix.

1. The results on “test.jpg” in terms of a confusion matrix.



1. Your analysis of the result: what did you learn from this experiment? If you get bad results, why is that? if you get better results, from where does the improvement come?

From this experiment, I learned that there are many ways of manipulating an image in order to speed up the recognition rate for optimal character recognition. I did obtain 65.7% recognition rate for the test.jpg, however, it was very difficult to get to that recognition rate. At first I had tried the erosion, and while by definition it is supposed to make the lines thinner it actually made it thicker surprisingly enough. Still a little confused about that, but also to boost the recognition rate some more I used the function imadjust, which increases the contrast. This helped the recognition rate a lot because the higher contrast helped fill the characters that were disjoint and possibly causing a problem with the threshold set at 7. This did cause a problem quite a bit so I did change the threshold to binarize the image to 220. This prevented the program from having more connected components than characters in the image. Another way that problem could have been solved was by increasing the threshold size which is set at 7.