Problem 1: Optical Character Recognition (OCR)

(a) Decision Tree For Reference Data

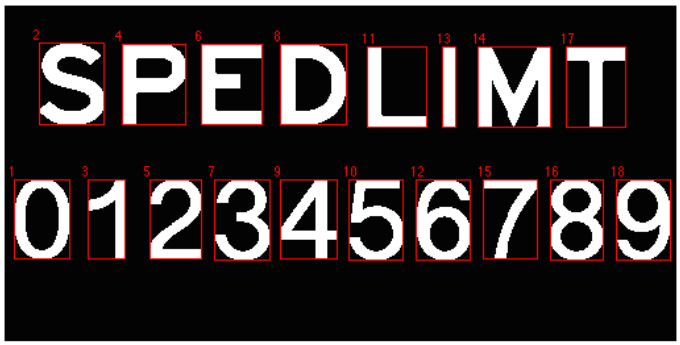
Introduction:

In this problem we are asked to segment each character from the training image and use it to train our system to implement an OCR system.

Methodology:

In the first step, the training image is binarized with a proper threshold. In the second step, we need to segment each character, for this purpose different characters are identified from each other by 8-connectivity.

After distinguishing every character, we can separate them from the whole image and analyze them separately, also for illustration we can draw bounding boxes around them to demonstrate segmentation of the binary image:



Bounding Boxes of the Characters

In the image above, the labels in the upper left corners of the characters are just indicating the object number, nothing to do with OCR.

In the next step, as we have each character image separately, we can do the feature extraction. The features used in my OCR system are:

- Normalized Area
- Normalized Perimeter
- Aspect Ratio
- Circularity
- Euler Number
- SymmetryX
- SymmetryY

Normalized Area: Normalized Area is calculated as,

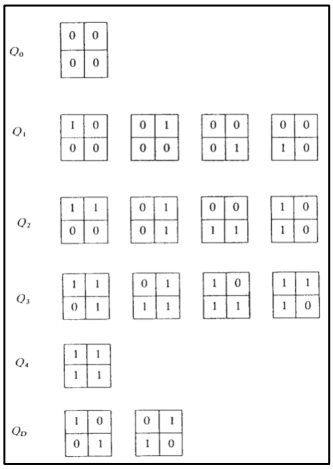
(Area of Object) / (Area of Bounding Box)

Area of the object is calculated as the total number of pixels that the object is occupying.

Normalized Perimeter: Normalized Perimeter is calculated as,

(Perimeter of Object) / (Perimeter of Bounding Box)

We get the Perimeter of the Object with the help of bit quads as shown below:



And the parameter of the object is calculated with the following formula:

$$P_O = n\{Q_2\} + \frac{1}{\sqrt{2}}[n\{Q_1\} + n\{Q_3\} + 2n\{Q_D\}]$$

Aspect Ratio: Aspect ratio is calculated as,

(Height of the Bounding Box) / (Width of the Bounding Box)

(In the lecture notes, this formula is defined as (Width / Height), but it does not make a difference while we are trying to distinguish different features with this value)

Circularity: Circularity is defined as,

Circularity = $(4 * \pi * A) / P^2$, where A=Area, P=Perimeter

Euler Number: Euler number is computed as,

$$E = \frac{1}{4}[n\{Q_1\} - n\{Q_3\} - 2n\{Q_D\}]$$

SymmetryX: This feature gives us a measure of the objects symmetry around the x-axis. If the object is highly symmetric around its X axis (X axis is the horizontal line passing through horizontally in the middle of the bounding box.), this value will be high. To compute this value, we flip the image around the X axis and take the difference, if the difference is large, then SymmetryX value will be low.

SymmetryY : This feature gives us a measure of the objects symmetry around the y-axis. If the object is highly symmetric around its Y axis (Y axis is the vertical line passing through vertically in the middle of the bounding box.), this value will be high. To compute this value, we flip the image around the Y axis and take the difference, if the difference is large, then SymmetryY value will be low.

Feature Extraction:

By using these feature extraction functions, we extract features of each character in the training image, i.e. calculate every feature for each character. If we do this procedure we get the following feature table:

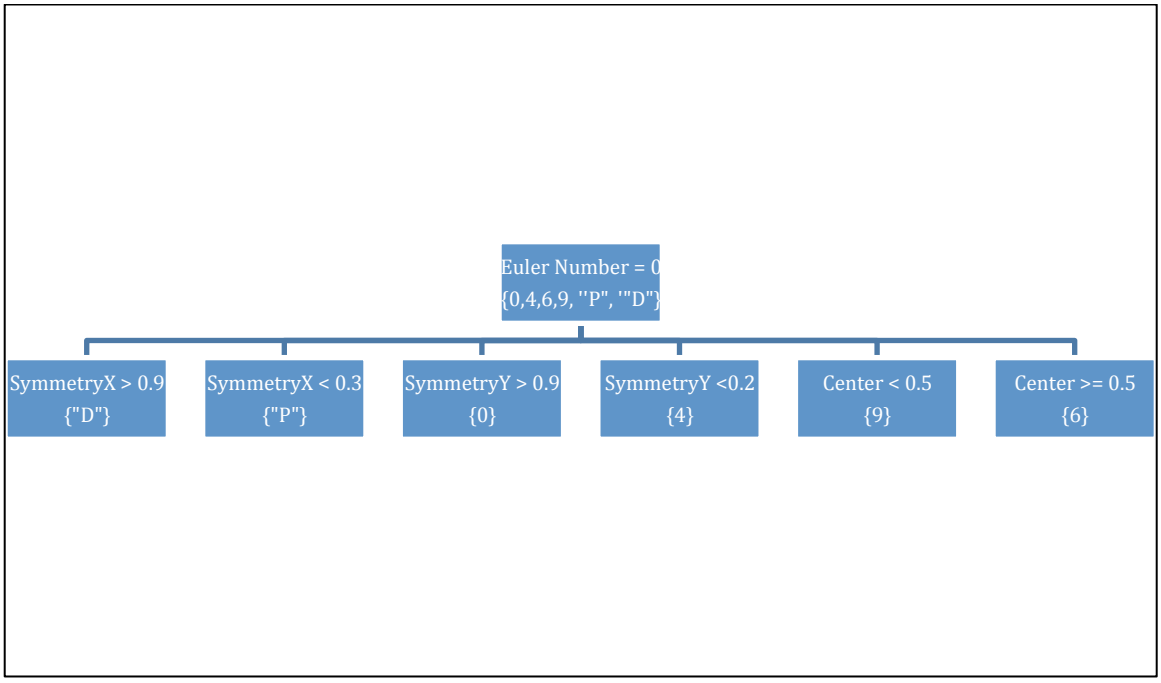
'*'	'NormalizedArea'	'NormalizedPerimeter'	'AspectRatio'	'Circularity'	'EulerNumber'	'SymmetryY'	'SymmetryX'
'0'	0.378723404	0.777343266	1.382978723	0.479536318	0	0.975799481	0.946413137
'S'	0.436981758	1.329777407	1.240740741	0.191846156	1	0.464895636	0.37254902
'1'	0.31981352	0.916829004	1.96969697	0.266959358	1	-0.402332362	0.422740525
'P'	0.446969697	0.862969577	1.222222222	0.466672475	0	0.451349655	0.260514752
'2'	0.384957265	1.359257181	1.444444444	0.158233932	1	0.373001776	0.072824156
'E'	0.497326203	1.361048211	1.294117647	0.207389414	1	0.379928315	0.870967742
'3'	0.346872985	1.328538864	1.404255319	0.149988075	1	0.466542751	0.576208178
'D'	0.50137741	0.832634193	1.2	0.563302964	0	0.421978022	0.98021978
'4'	0.35	0.831153188	1.354166667	0.388913908	0	0.106227106	0.353479853
'5'	0.37911726	1.422552695	1.434782609	0.142446711	1	0.525629887	-0.009556907
'L'	0.302424242	0.90874322	1.32	0.282151308	1	-0.170340681	0.366733467
'6'	0.428853755	1.114834247	1.434782609	0.262363979	0	0.539170507	0.511520737
'I'	0.687878788	0.872927244	4.4	0.427928144	1	0.997063142	0.997063142
'M'	0.541480378	1.594072615	1.081967213	0.167102413	1	0.966055046	0.458715596
'7'	0.261865794	0.979002074	1.382978723	0.209043463	1	-0.02	-0.4725
'8'	0.443434343	0.847596947	1.466666667	0.467423909	-1	0.908883827	0.630979499
'T'	0.308484848	0.903693337	1.32	0.291031174	1	1	0.218074656
'9'	0.423913043	1.115917465	1.434782609	0.258838113	0	0.613053613	0.529137529

Decision Tree:

We construct the decision tree starting with the Euler Number. The values Euler Number can take in our case are: -1, 0 ,1.

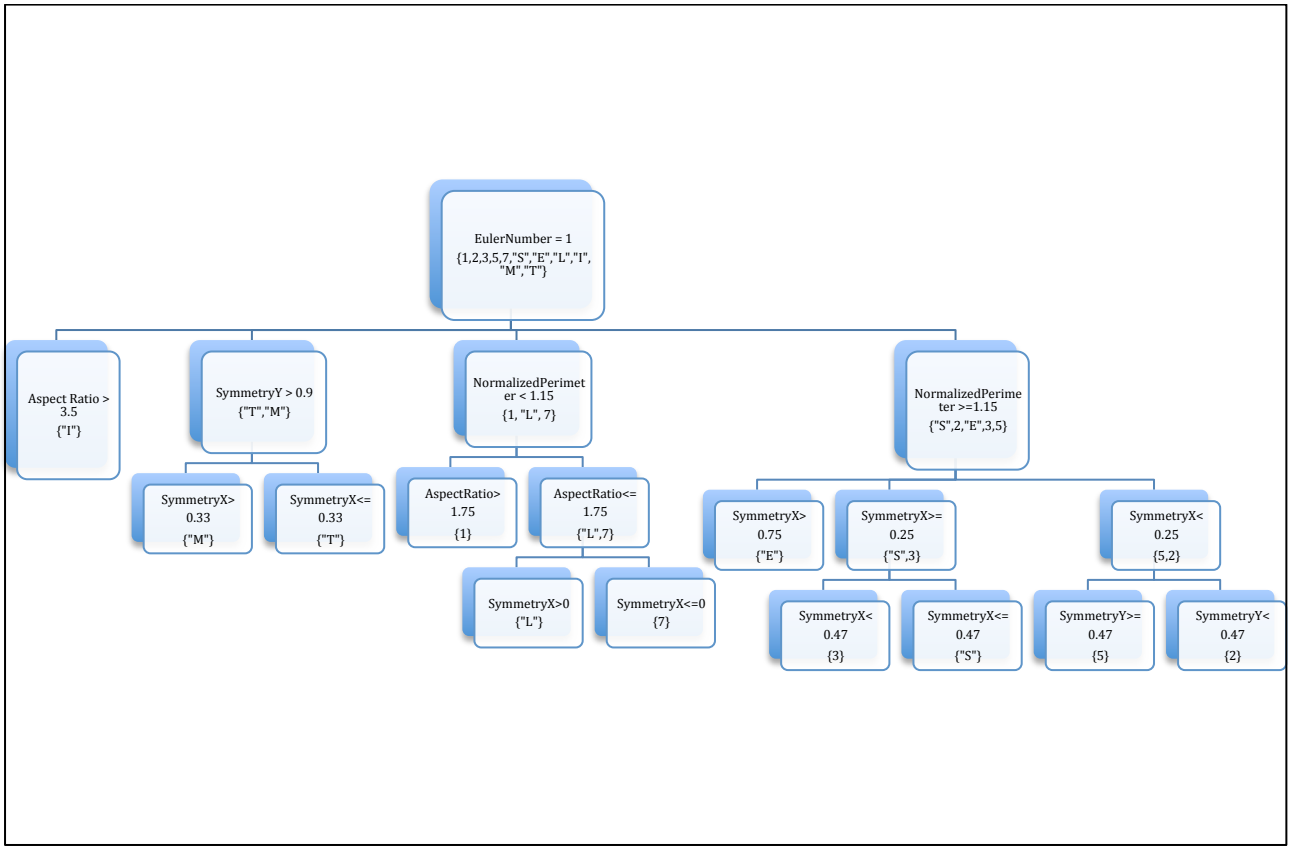
I will show the complete decision tree in three parts, starting with different Euler Numbers.

*Case: Euler Number = 0

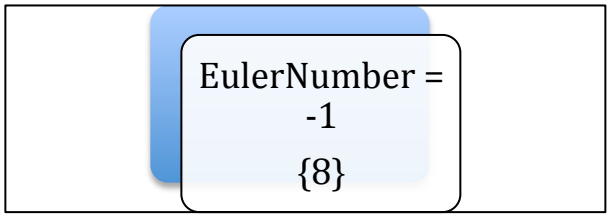


Here I explain the “Center” feature in this decision tree, It was hard to classify between the character “6” from “9”, to overcome this problem I filled inside the holes of “6” and “9” and computed the center of gravity for both. As a result, we are expecting the the Y component of the location of center of gravity of “9” to be higher than “6”. With this way, I distinguished “9” from “6”.

*Case: Euler Number = 1

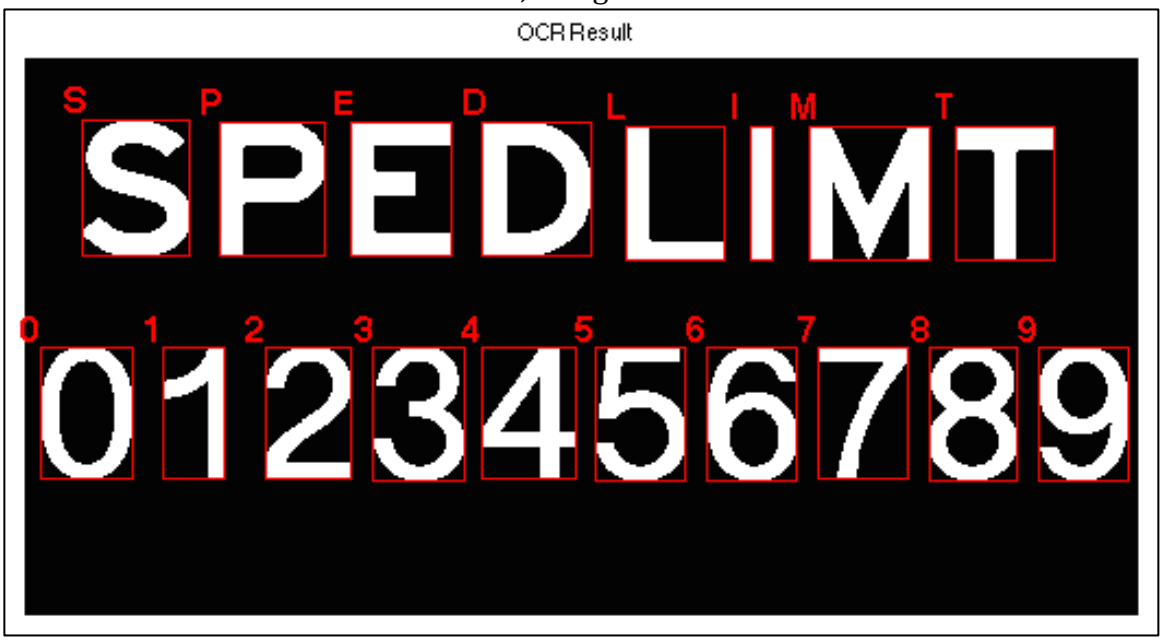


*Case: Euler Number = -1



Applying the Constructed Decision Tree on the training data:

If we apply the constructed decision tree to each of the character in the training image, we get this result. The labels on the upper-left corners are predicted character labels from the features of the character, using the decision tree.



As we can see, the decision tree gives 100% correct classification rate in the training data.

(b) OCR Testing: Simple Cases

Test_Ideal 1 Image:

Methodology:

Preprocessing:

After converting the RGB image to grey level. I am applying "Otsu" method to find a proper threshold to binarize the grey level image. Otsu threshold chooses the one to minimize the interclass variance of the black and white pixels.

After Otsu thresholding we get the binary image below:

The binarized image contains some small pieces that should be removed for OCR. For this purpose, we perform binary image opening with a cross shaped structure element of size 5. After opening, we get the following result:

As we can see from the image above, we got rid of the small pieces by performing opening. However, we see some damage on top of the letter “E”, for this purpose we perform closing on the image with a square structure element of size 6.

Now, the preprocessing is complete, we apply our segmentation method (8-connectivity connected component labeling) to get each character and apply the decision tree rules to classify each character.

OCR Result:

The image below is the result of OCR, as we can see all characters are correctly classified.

Test_Ideal 2 Image:

Preprocessing:

After converting the RGB image to grey level, we need to perform noise removal. While doing the noise removal, we also need to preserve the edges for OCR, for this purpose, we perform adaptive noise removal.

Greyscale Image

Noise Removed

After cleaning the noise, we perform Otsu thresholding to binarize the image. After binarizing we realize that the locations around the characters are jagged, so we perform closing on the binary image with square structure element of size 4. After performing closing, the characters look more robust.

After Otsu Thresholding

After Performing Closing

We need to remove the outer frame before OCR, for this purpose we perform opening with a structural element with height 5 and width 1 (5,1). After that we perform closing with a square structural element of size 3.

Outer Frame Removed

After Performing Closing

OCR Result:

OCR Result

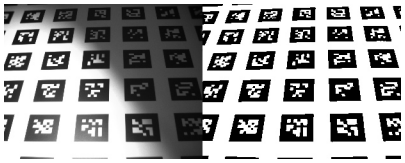
From the result above, we observe that, all characters are correctly classified except "S". In this example "S" is classified as 3. This might be due to the fact that they have the same euler number, similar aspect ratio and similar symmetryX (measure of how symmetrix around the X axis) value.

(c) OCR Testing: Advanced Cases

Test_Night Image

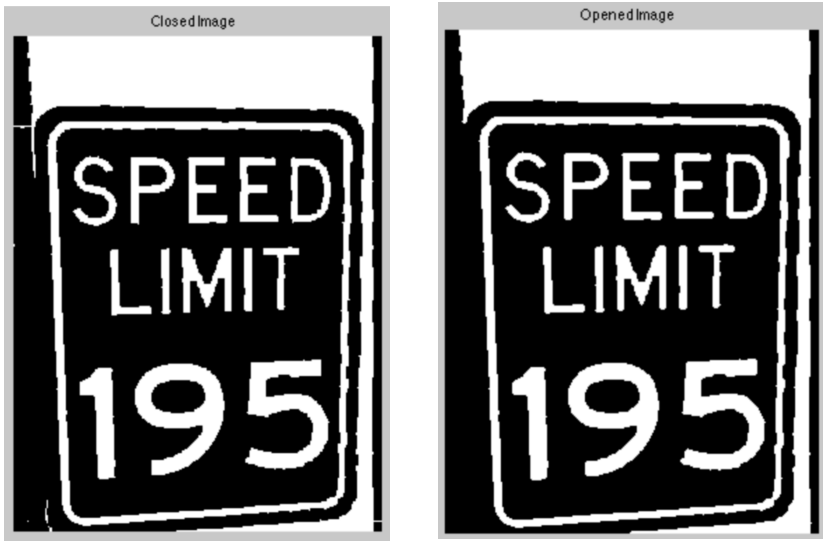
Preprocessing:

After converting the RGB image to grey level, we realize a shadow on the right half of the image, therefore global thresholding might cause problems. Therefore we need to apply local image thresholding, for this purpose I chose to apply, "Sauvola Local Image Thresholding", as we can see the example image below, it gives good results in case of a shadow.



Example of Sauvola Image thresholding

After thresholding the speed limit sign image, we perform closing and opening to make the characters more robust and readable for OCR:



OCR Result:

On the image above, we have the OCR result, as we can see some of the characters are incorrectly classified, some of them are correctly classified. The reason might be due to the fact that the image is taken with a camera angle, to overcome this problem, maybe from the orientation of the outer frame of the speed sign we can predict the camera viewing angle and then apply some computer vision techniques such as homography to adjust the viewing angle of the speed sign in a way that camera is looking directly to the sign with no angle.

Sources:

http://www.mathworks.com/matlabcentral/fileexchange/40266-sauvola-local-image-thresholding