INDEX

1. Problem #1:Optical Character Recognition 5
   1.1 Decision tree for reference data 5
      1.1.1 Motivation 5
      1.1.2 Approach and Implementation 5
      1.1.3 Results 14
      1.1.4 Discussion 15
   1.2 OCR Testing : Simple Cases 16
      1.2.1 Motivation 16
      1.2.2 Approach and Implementation 16
      1.2.3 Results 19
      1.2.4 Discussion 21
   1.3 OCR Testing : Advanced Cases 22
      1.3.1 Motivation 22
      1.3.2 Approach and Implementation 22
      1.3.3 Results 26
      1.3.4 Discussion 28
2. Problem #2: Contour Modeling 29
   2.1 Motivation 29
   2.2 Approach and Implementation 29
      2.2.1 Snakes: Active contour models 29
      2.2.2 Active Contours without edges 31
   2.3 Results 33
   2.4 Discussion 46
3. Problem #3: Salient Point Descriptors and Image Matching 47
   3.1 Extraction and Description of Salient Points 47
      3.1.1 Motivation 47
      3.1.2 Approach and Implementation 47
         3.1.2.1 Scale Invariant Feature Transform (SIFT) 47
         3.1.2.2 Speeded Up Robust Feature (SURF) 52
      3.1.3 Results 55
      3.1.4 Discussion 58
   3.2 Image Matching 59
      3.2.1 Motivation 59
      3.2.2 Approach and Implementation 59
      3.2.3 Results 61
      3.2.4 Discussion 63
   3.3 Bag Of Words 64
      3.3.1 Motivation 64
      3.3.2 Approach and Implementation 64
      3.3.3 Results 68
      3.3.4 Discussion 69
4. References 70

LIST OF FIGURES

1.1	Training image	5
1.2	ROI around every character	6
1.3	Euler number =-1	7
1.4	Euler number = 0	7
1.5	Euler number = 1	7
1.6	Number of Vertical Intersections = 1	8
1.7	Number of Vertical Intersections = 2	8
1.8	Number of Vertical Intersections = 3	8
1.9	Number of Horizontal Intersections = 1	8
1.10	Number of Horizontal Intersections = 2	8
1.11	Number of Horizontal Intersections = 4	8
1.12	Object pixels in first 10 rows > 50%	9
1.13	Object pixels in first 10 rows < 50%	9
1.14	Object pixels in last 10 rows > 75%	10
1.15	Object pixels in last 10 rows < 75%	10
1.16	Object pixels in first 10 columns > 60%	10
1.17	Object pixels in first 10 columns < 60%	10
1.18	Decision Tree of OCR	13
1.19	(a) Test image 1	16
1.19	(b) Test image 2	16
1.20	(a) Pre-processed test image 1	17
1.20	(b) Pre-processed test image 2	17
1.21	(a) Segmented Test image 1	18
1.21	(b) Segmented Test image 2	18
1.22	(a) Test night Image after histogram equalization	23
1.22	(b) Test shade Image after histogram equalization	23
1.23	(a) Test night thresholded image	24
1.23	(b) Test shade thresholded image	24
1.24	(a) Test night segmented image	25
1.24	(b) Test shade segmented image	25
2.1	(a) Initial contour points away from the object boundary	33
2.1	(b) Snake contour for the incorrect initialization	33
2.1	(c) Initial contour points near the object boundary	34
2.1	(d) Snake contour for the incorrect initialization	34
2.2	Snake contour with alpha = 0	35
2.3	Contour with beta = 0	36
2.4	Contour with gamma = 4	36

2.5 Contour with kappa = 0 37
2.6 Contour with W-line = 0 37
2.7 Contour with W-edge = 0 38
2.8 Contour with iterations = 10 38
2.9 Extraction of the spine contour 40
2.10 Extraction of the coronary outline 40
2.11 Separation of white and red blood cells 41
2.12 Isolation of the tumor in brain MRI 41
2.13(a)Level set contour of spine 42
2.13(b)Final level set function of spine after 400 iterations 42
2.14(a)Level set contour of coronary 43
2.14(b)Final level set function of coronary after 600 iterations 43
2.15(a)Level set contour of blood cells 44
2.15(b)Final level set function of blood cells after 600 iterations 44
2.16(a)Level set contour of brain 45
2.16(b)Final level set function of brain after 400 iterations 45
3.1 Comparison of pixels to its 26 neighbors 48
3.2 SIFT descriptor computation 51
3.3 SURF descriptor 54
3.4 SIFT features for the Bus image 55
3.5 SIFT features for the sedan image 55
3.6 Bus image with SIFT features and 15 descriptors 56
3.7 Sedan image with SIFT features and 10 descriptors 56
3.8 SURF features for the bus image 57
3.9 SURF features for sedan image 57
3.10 Image matching between school bus 1 and school bus 2 61
3.11 Image matching between school bus 1 and Bus 61
3.12 Image matching between school bus 1 and sedan 62
3.13 Concept of representing an image as bag of words 64
3.14 Codeword of an image as a histogram 66
3.15(a)Histogram of school bus 1 68
3.15(b)Histogram of sedan 68
3.15(c)Histogram of Bus 69
3.15(d)Histogram of school bus 2 69

LIST OF TABLES

Table 1.1 OCR for training image 14
Table 1.2 OCR for test image 1 19
Table 1.3 OCR for test image 2 20
Table 1.4 OCR for advanced test image 1 26
Table 1.5 OCR for advanced test image 2 27

Amruta Kulkarni,
USC ID: 6914970827
Email: arkulkar@usc.edu
Date: 29 Nov 2015

EE 569 HOMEWORK #4

---

1. Problem #1: Optical Character Recognition

1.1 Decision Tree for Reference Data

1.1.1 Motivation:

Optical character recognition is defined as the conversion of images of handwritten or printed text into machine-encoded text. It is mainly used for processes such as text-to-speech conversion and text mining. OCR has various applications in the field of computer vision and pattern recognition. An optical character recognition system can be trained by using a set of text symbols that we want recognize. In this problem, we will train our OCR system by using 18 different characters that we later want to identify in the test images.

1.1.2 Approach and Implementation

- To start with, we have the following image that contains 18 characters that include 10 numbers and 8 alphabets.

Figure 1.1 Training image

- This is an RGB image that we will first convert into a black and white image.
- By using thresholding, this image can be converted into a binary image.
- Now, we want to segment this image into 18 segments where each segment contains one character.
- Obtaining the Region Of Interest (ROI) around each character can do this.
- We can obtain the ROI around each character by determining X-min, X-max, Y-min and Y-max co-ordinates of each of the 18 ROIs.
- To do this, we can perform the 2-step image labeling method and give different labels to every character. Then, we can easily find the ROI which is nothing but a rectangular bounding box around every character as shown in the figure below:

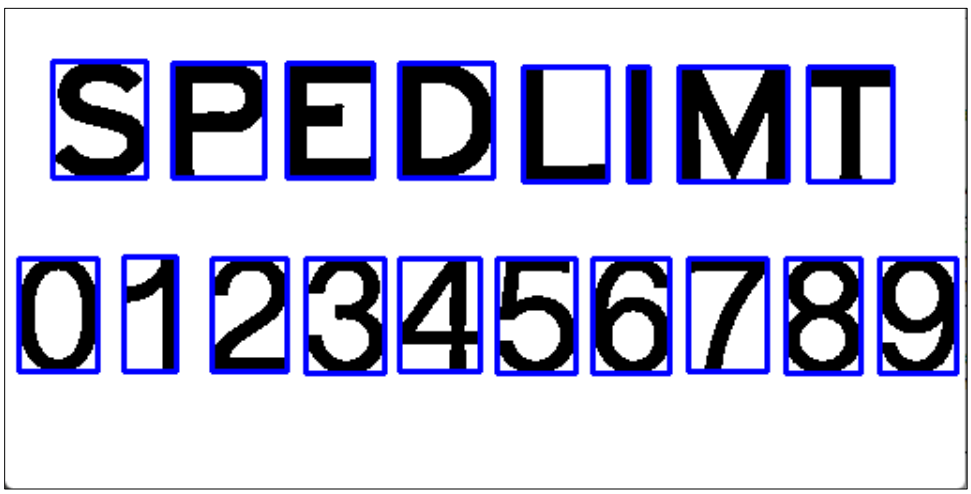

Figure 1.2 ROI around every character

- Now, for every binarized segment, we need to extract different features from every segment and then generate a decision tree classifier using these extracted features
- In this problem, we have extracted the following features:

1) <u>Euler number</u>

- Euler number is basically the difference of the total number of connected objects and the total number of holes in that object.
- Euler number for every character can be calculated with the help of bitquads.
- The formula for Euler number calculation is as follows:

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

- The parameters n{Q1}, n{Q3} and n{Qd} are calculated with the help of the following bit quad patterns:

Q_0

0	0
0	0

Q_1

1	0
0	0

0	1
0	0

0	0
0	1

0	0
1	0

Q_2

1	1
0	0

0	1
0	1

0	0
1	1

1	0
1	0

Q_3

1	1
0	1

0	1
1	1

1	0
1	1

1	1
1	0

Q_4

1	1
1	1

Q_D

1	0
0	1

0	1
1	0

- Thus, we need to traverse the image and find the total number of matches for all the above patterns.
- Using these parameters, we can get the euler number.
- For this problem we got 3 values of euler numbers i.e. -1,0 and 1.
- The examples of characters in the training image with these euler numbers are as shown below:

Figure 1.3 Euler number = -1

Figure 1.4 Euler number = 0

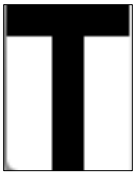

Figure 1.5 Euler number = 1

2) <u>Number of intersections of a vertical line from the center</u>

- If we draw a vertical line passing from the center of the rectangular bounding box (i.e. the ROI), then this line will cut every character in different number of intersections.
- For the characters in our training image, we got 3 values for the number of intersections i.e. 1,2 and 3.
- For example, we have the following three characters with 3 different number of intersections of the vertical line:

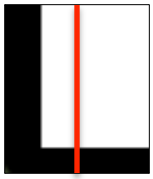

Figure 1.6 No. of vertical intersections = 1

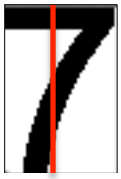

Figure 1.7 No. of vertical intersections = 2

Figure 1.8 No. of vertical intersections = 3

3) <u>Number of intersections of a horizontal line from the center</u>

- Similar to the above idea, if we draw a horizontal line passing through the center of the rectangular bounding box, then we will obtain different number of intersections of this line with the different characters.
- In our training image, we obtained 3 different intersection values for this horizontal line and they are 1,2 and 4.
- The following examples show the characters that have different number of horizontal intersections:

Figure 1.9 No. of horizontal intersections = 1

Figure 1.10 No. of horizontal intersections = 2

Figure 1.11 No. of horizontal intersections = 4

4) <u>Vertical symmetry</u>

- To calculate the symmetry percentage of the characters of the training image, we can first divide each ROI into two halves vertically.

- Then we can simply traverse the first half from bottom to top and at the same time the second half from top to bottom.
- Thus, for every object element we can compare if they are the same in both the parts.
- If they are, then they are the mirror reflections and each other and this will contribute towards symmetry of that object.
- The percent symmetry for all the 13 characters has been calculated.
- For characters like '0', '3' and 'I' the symmetry is much more than that for characters like '1' and '2'.
- This is a helpful feature to distinguish between two characters.

5) <u>Percentage of object pixels in the first 10 rows of the ROI</u>

- Let us look at the following two characters:

Figure 1.12 object pixels in first 10 rows > 50%

Figure 1.13 object pixels in first 10 rows < 50%

- If we count the total number of object pixels that are present in the first 10 rows of the ROI and then divide this number by 10 times the width of that ROI then we will get the percentage of object pixels in the first 10 rows of that ROI.
- In these two examples, the character 'D' has more number of object pixels in the first 10 rows of its ROI than the character '4' has in its ROI.
- To get the percentage and to get normalized values, so that we can compare these percentage values of different characters, we can divide this object pixel number by the total pixels that are present in the first 10 rows.
- Thus, 'D' has more than 50 % object pixels and lesser background pixels whereas '4' has less than 50% of object pixels and more background pixels in the first 10 rows of their respective ROI.

6) <u>Percentage of object pixels in the last 10 rows of ROI</u>

- Consider the following two characters:

Figure 1.14 object pixels in last 10 rows > 75%

Figure 1.15 object pixels in last 10 rows < 75%

- By using the similar idea discussed above, we can calculate the percentage of object pixels present in the last 10 rows of the ROI of every character by counting the number of object pixels and then dividing it by the total number of pixels in the last 10 rows of the ROI.
- Thus, for '2', the percent object pixels are greater than 75% and for 'S' they are less than 75%
- This feature is mainly useful in distinguishing between '2' and 'S' and also '1' and 'L'.

7) <u>Percentage of object pixels in the first 10 columns of the ROI</u>

- Consider the following two characters:

Figure 1.16 object pixels in first 10 columns > 60%

Figure 1.17 object pixels in first 10 columns < 60%

- Now, instead of considering the first 10 rows, we can also consider the first 10 columns.
- Using the same method discussed above, we can obtain the percentage of object pixels in the first 10 columns of the ROI.
- For the character '6', this percentage will be greater than that for the character '9'.
- This feature is mainly used to distinguish between '6' and '9'.

8) <u>Number of intersections of a horizontal line at the bottom</u>

- Sometimes, the letter 'M' may get misclassified because its classification mainly relies on the number of intersections of the horizontal line from the center.
- But, in some cases for e.g. after dilation or closing operation, the shape of this character may change and it might just result in 3 intersections instead of 4.
- Thus, to avoid this misclassification, we will use this additional feature just to avoid the misclassification of the character 'M'.

- Thus, by using the above techniques, we can extract 8 features from every character of the training image.
- Now, these extracted features can be used to construct a decision tree that will act as our classifier for OCR.
- To construct a decision tree, we can consider different features at different nodes and also use different thresholds at each node.
- The decision tree that has been constructed by using these 8 features can successfully classify the 18 characters of the training image.
- The first layer of the decision tree checks the Euler number of the characters.
- As we saw, this number is -1, 0 or 1. Depending upon its value, this node will have 3 branches and then each character will get labeled according to the branch (or parent node) it belongs to.
- Next, we compare the number of vertical intersections and then again the nodes grow into more number of branches.
- In the third level, we have compared the number of horizontal intersections and then split the characters into different branches based on this feature.
- Similarly, at every level, we will consider the feature, at every node, that can separate the characters in the best possible way.
- Thus, in short, we compare different features on every level of the tree and the give a temporary label to the characters at these intermediate levels.
- After we reach the final leaf nodes, we can assign the final label to the character.
- This idea has been depicted in the decision tree shown in figure 1.18.
- This decision tree has been used to recognize the characters in the training image.
- The results show the set of features for all these 18 characters, their true labels and their computed labels using this decision tree.
- In the decision tree, the red circles are the true labels whereas the black underlined letters outside the pink boxes, that have the unclassified characters, are the temporary labels.
- The green boxes are the features whereas the blue boxes are the threshold values of those features.
- The final nodes are highlighted in florescent green color and they contain the final recognized characters.

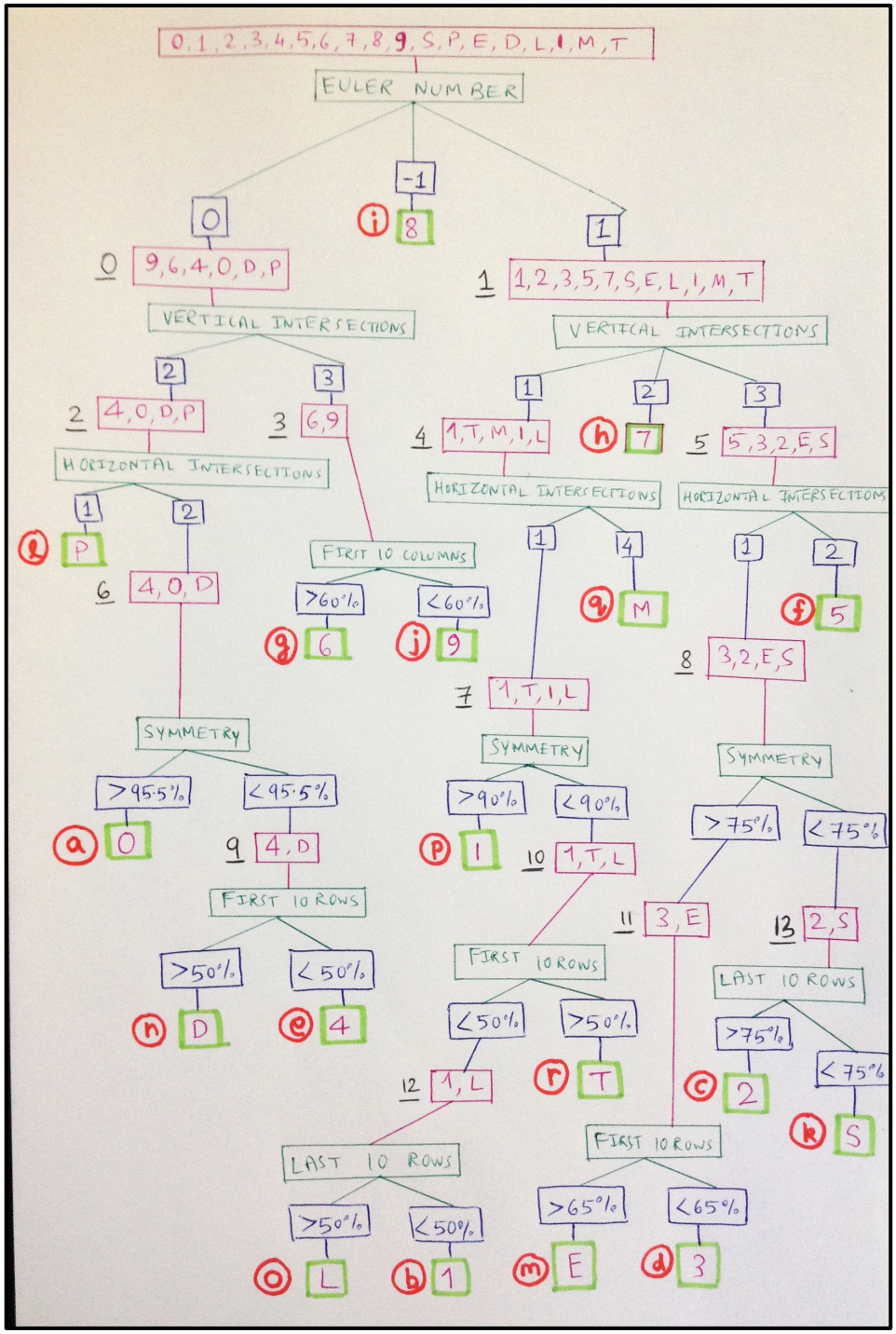

Figure 1.18 Decision tree of OCR

1.1.3 Results

Character	True label	Assigned label	Euler No.	Total vertical intersections	Total horizontal intersections	Symmetry	% object pixels in first 10 rows	% object pixels in last 10 rows	% object pixels in first 10 columns	Total intersections of bottom line
0	a	a	0	2	2	95.88%	55.34%	53.48%	65.24%	1
1	b	b	1	1	1	75.86%	34.48%	27.58%	14.67%	1
2	c	c	1	3	1	58%	61.46%	85.12%	49.83%	1
3	d	d	1	3	1	78.99%	56.51%	58.37%	34.35%	1
4	e	e	0	2	2	71.90%	28.40%	20.45%	25.24%	1
5	f	f	1	3	2	54.76%	67.44%	58.83%	52.09%	1
6	g	g	0	3	2	72.58%	56.90%	56.90%	64.35%	1
7	h	h	1	2	1	53.92%	84.65%	22.09%	14.09%	1
8	i	i	-1	3	1	76.78%	60%	60%	67.25%	1
9	j	j	0	3	2	73.50%	59.28%	55.47%	57.41%	1
S	k	k	1	3	1	67.87%	60.58%	69.80%	33.33%	1
P	l	l	0	2	1	62.90%	87.4%	24%	100%	1
E	m	m	1	3	1	85.44%	97.87%	100%	100%	1
D	n	n	0	2	2	95.31%	72.15%	71.56%	100%	1
L	o	o	1	1	1	75.45%	23.91%	100%	100%	1
I	p	p	1	1	1	100%	100%	100%	100%	1
M	q	q	1	1	4	68.67%	47.28%	49.49%	99.83%	3
T	r	r	1	1	1	73.77%	100%	26.08%	19.35%	1

Table 1.1 OCR for training image

1.1.4 Discussion

The decision tree and the table of feature values along with true and OCR labels have been successfully generated for the training image. For every character, 8 features are extracted and depending upon its features value at every node, it flows down the branch of the decision tree and ends up at the leaf node. The main features that have been extracted are the Euler number, number of intersections for a vertical line from the center, number of horizontal intersections for a horizontal line from the center, symmetry, Percentage of object pixels in the first 10 rows, Percentage of object pixels in the last 10 rows and Percentage of object pixels in the first 10 columns. The 8th feature i.e. the number of intersections for a horizontal line at the bottom is mainly extracted to avoid the misclassification of the character 'M'. It is calculated after the temporary label 7 has been assigned. In the training image, M does not get misclassified and hence this branch is not included in the decision tree shown above, but this feature is calculated just to be on safer side.

1.2 OCR Testing : Simple Cases

1.2.1 Motivation

In the above section, we have trained our classifier to label the characters by using the decision tree on the extracted features of different characters. In this problem, we will consider two different images that have a subset of characters from the training image. Thus, we now want to see how well our decision tree classifier works on the testing data. To test our Optical Character Recognition system designed above, we will again have to use image segmentation and feature extractions techniques.

1.2.2 Approach and Implementation

- The test images are RGB images and we first need to convert them to gray-scale images. Then we need to perform thresholding to get binary images. The two test images that are obtained after the above steps are as shown below:

Figure 1.19(a) Test image 1

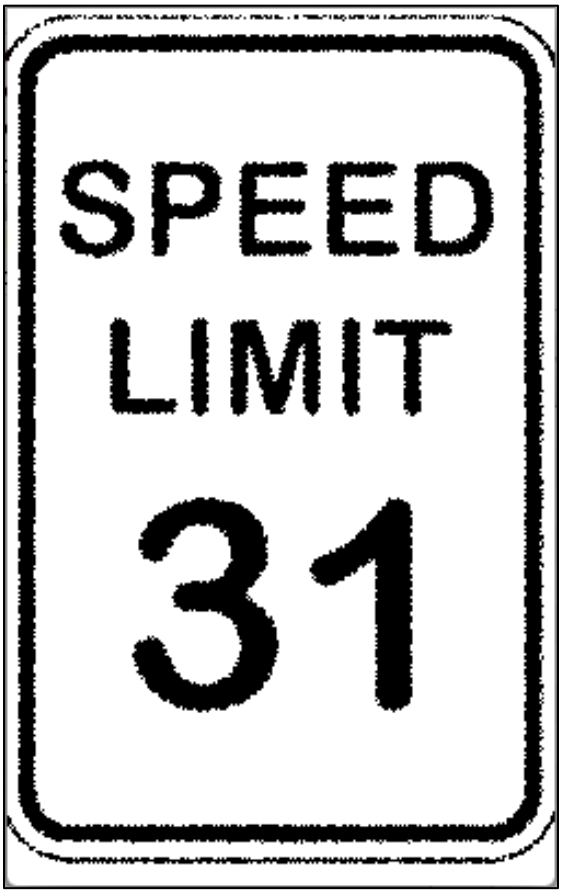

Figure 1.19(b) Test image 2

- From the above images, we can see that the speed boards have boundaries and also some other non-uniformity, like holes, which we want to get rid of.
- Thus, we need to remove the boundaries and also perform some morphological operations to smoothen the image and perform hole-filling.
- Now, the largest ROI will be that of the boundary in both the test images.
- Thus, after getting the x-min, x-max, y-min and y-max of all the ROIs, we can just consider ROI of the largest area.
- Now, we can simply discard those boundary pixels by iteratively setting the boundary pixels to the background pixels around the boundary ROI's co-ordinates.
- After we get rid of the boundaries, we need to perform the closing operation that is nothing but dilation followed by erosion.
- After these pre-processing steps, we get smooth images that contain only the characters that we want to recognize.
- The pre-processed images are as shown below:

Figure 1.20 (a) Pre-processed test image 1

Figure 1.20 (b) Pre-processed test image 2

- Now, we need to perform image segmentation to get the ROI i.e. the rectangular bounding box around every character in both the images, just like we did for the training image.
- After this we get the following two images:

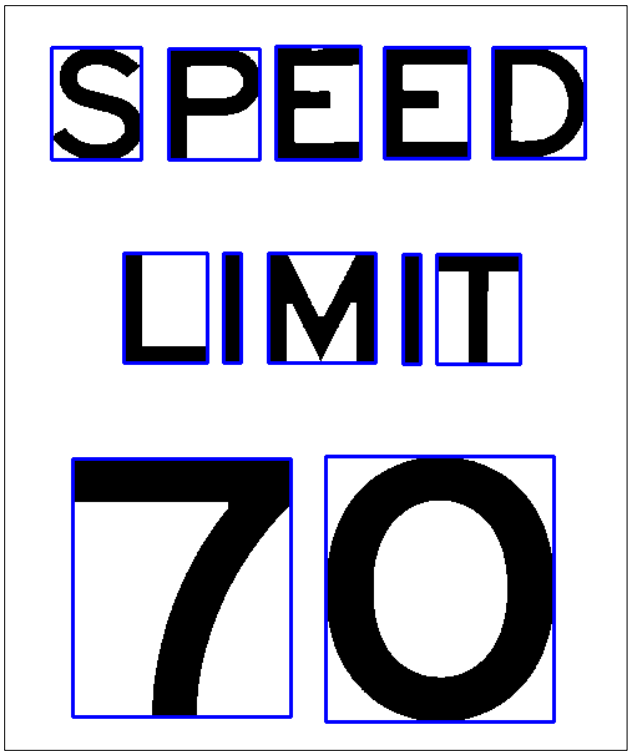

Figure 1.21(a) Segmented Test image 1

Figure 1.21(b) Segmented Test image 2

- Now, we need to consider every ROI of both the test images separately and extract the 8 features from each of the characters.
- The feature extraction part has already been explained in the previous section.
- After the features are extracted from all the 12 characters from both the respective images, we can use the decision tree classifier that was designed during the training phase of OCR. This decision tree is shown in figure 1.21.
- Thus, we simply need to pass in the extracted features of the characters to check if OCR is correctly able to detect these characters.
- The labels that get assigned by the OCR system as well as the true labels, along with the extracted features are shown in the results.

1.2.3 Results

The OCR labels, true labels and the extracted features for the test image 1 are shown in the following table:

Character	True label	Assigned label	Euler No.	Total vertical intersections	Total horizontal intersections	Symmetry	% object pixels in first 10 rows	% object pixels in last 10 rows	% object pixels in first 10 columns	Total intersections of bottom line
S	k	k	1	3	1	69.76%	49.04%	57.26%	15.38%	1
P	l	l	0	2	1	63.63%	82.70%	19.18%	99.11%	1
E	m	m	1	3	1	91.14%	77.68%	90%	96.63%	1
E	m	m	1	3	1	91.33%	90.28%	99.85%	98.66%	1
D	n	n	0	2	2	93.71%	64.4%	60.26%	97.66%	1
L	o	o	1	1	1	76.10%	22.05%	95.29%	98.87%	1
I	p	p	1	1	1	94.15%	98.66%	93.33%	99.77%	1
M	q	q	1	1	4	64.40%	36.16%	38.16%	98.08%	3
I	p	p	1	1	1	98.65%	100%	96%	99.77%	1
T	r	r	1	1	1	75.57%	92.79%	22.05%	15.73%	1
7	h	h	1	2	1	52.40%	98.92%	20.45%	16.82%	1
0	a	a	0	2	2	97.98%	28.20%	28.64%	31.63%	1

Table 1.2 OCR for test image 1

The OCR labels, true labels and the extracted features for the test image 2 are shown in the following table:

Character	True label	Assigned label	Euler No.	Total vertical intersections	Total horizontal intersections	Symmetry	% object pixels in first 10 rows	% object pixels in last 10 rows	% object pixels in first 10 columns	Total intersections of bottom line
S	k	k	1	3	1	72.15%	56.47%	58.23%	53.33%	1
P	l	l	0	2	1	60.17%	73.43%	23.12%	83.25%	1
E	m	m	1	3	1	88.85%	67.57%	73.63%	85.58%	1
E	m	m	1	3	1	90.26%	70.90%	72.72%	87.44%	1
D	n	a	1	3	1	95.60%	68.33%	70%	84.18%	1
L	o	o	1	1	1	76.09%	24.83%	74.19%	84.54%	1
I	p	p	1	1	1	92.77%	82.5%	87.5%	92.88%	1
M	q	q	1	1	1	58.53%	65.36%	51.70%	88.63%	3
I	p	p	1	1	1	91.11%	88.75%	82.5%	90%	1
T	r	r	1	1	1	78.28%	73.61%	19.44%	14.77%	1
3	d	d	1	3	1	78.52%	49.40%	48.80%	27.44%	1
1	b	b	1	1	1	79.24%	26.80%	27.87%	11.63%	1

Table 1.3 OCR for test image 2

1.2.4 Discussion

Features have been extracted from all the characters in both the images. As we can see from the results, all the characters were accurately determined for the first test image. For the second test image, all the characters except one were recognized correctly. As we can see from table 1.3, the character 'D' has been misclassified and the incorrect label has been shown in red. It got classified as '0', labeled as 'a'. If we look at the decision tree, 'D' and '0' are separated by using the symmetry feature. The threshold is a hard threshold of 95.5 % and if the symmetry value is greater than this threshold, then it is classified as '0'. In this case, 'D' crosses this threshold value as its symmetry value is 95.60% and thus it gets misclassified as '0'. The rest of the characters have been correctly classified.

1.3 OCR Testing : Advanced Cases

1.3.1 Motivation

We have already trained the OCR system using the 18 characters from the training image. Moreover, we have also tested if the decision tree classifier is able to classify 2 simple test images. In this problem, we will check if it works on 2 images that have not been captured under ideal conditions. Thus, we first need to perform pre-processing on these two images and then segment them to get characters. The remaining steps of feature extraction and classification stay the same as before.

1.3.2 Approach and Implementation

- The 2 test images need a lot of pre-processing so as to get proper character segments from them.
- First, we have performed transfer function based histogram equalization on both the images to improve their contrast.
- To do this, we first need to separate Red, green and blue channels of the image and then get the histogram of each of the channels separately.
- Now, we obtain the probability of each of the intensity values in the image by dividing the frequency of that value by the total number of pixels.
- After getting the probability, we need to calculate the cumulative distribution function by adding up the probabilities in a cumulative manner.
- Thus, we get the cumulative distribution function of the intensity values of the three channels.
- Now, we can just replace those corresponding intensity values in the original image by the multiplication of its cdf value with the maximum intensity that we want i.e. 255
- In this manner, we have now achieved an intensity range of 0 to 255 and thus increased the image contrast.
- The images after histogram equalization are as shown below:

Figure 1.22(a) Test night Image after histogram equalization

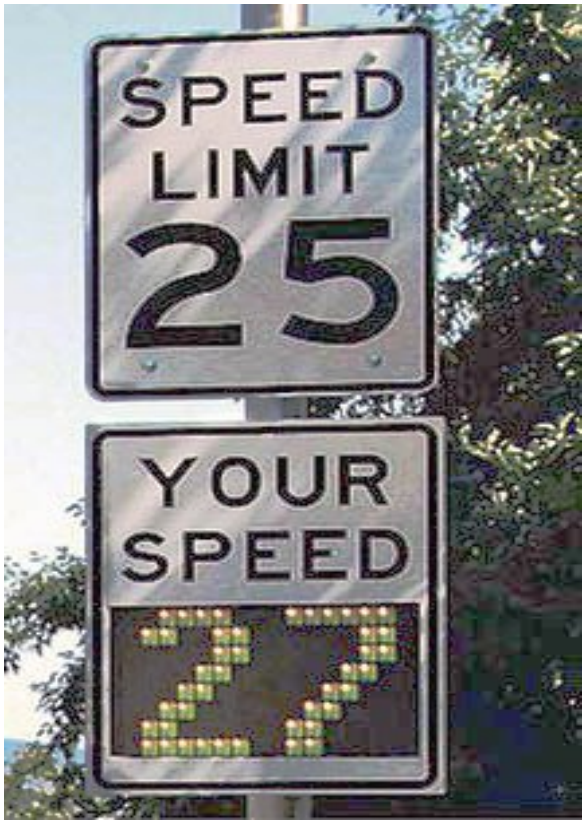

Figure 1.22(b) Test shade Image after histogram equalization

- After we have improved the contrast of the image, we now need to convert both the color RGB images to gray-scale images.
- Using the formula 0.21R+0.72G+0.07B does the conversion.
- Now, we need to perform thresholding on both the images.
- For the first image, thresholded has been performed on the entire image together.
- For the second image, using their x-min, x-max, y-min and y-max co-ordinates has cropped out the two plates.
- The second plate is further cropped into two more regions, one containing the alphabets and the other containing the numbers.
- The thresholding and morphological operations have then been performed on the two cropped regions separately.
- The images obtained after thresholding are as shown below:

Figure 1.23(a) Test night thresholded image

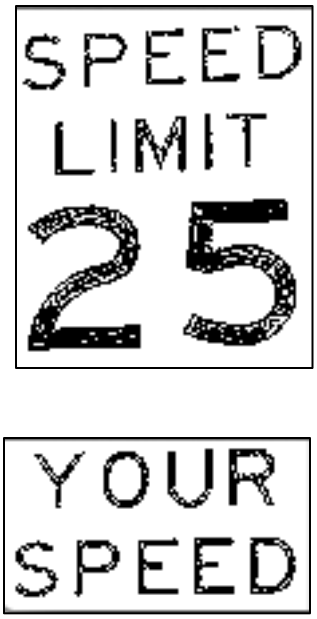

Figure 1.23(b) Test shade thresholded image

- We can see that the first image has borders and some non- uniformity in the background whereas the second image contains holes in between the characters.
- Now, we need to do boundary removal for the first image by using the largest area contour technique that has been discussed in section 1.2.2.
- Then we need to perform closing on the first image to get proper characters without any breaks or holes.
- Similarly, on the second image, we can perform image closing on both the plates.
- After this, we get images containing just the characters.
- By using image segmentation, we can again construct a rectangular bounding that specifies the ROI around every character like we have done for the training and the test images.
- The segmented images are shown below:

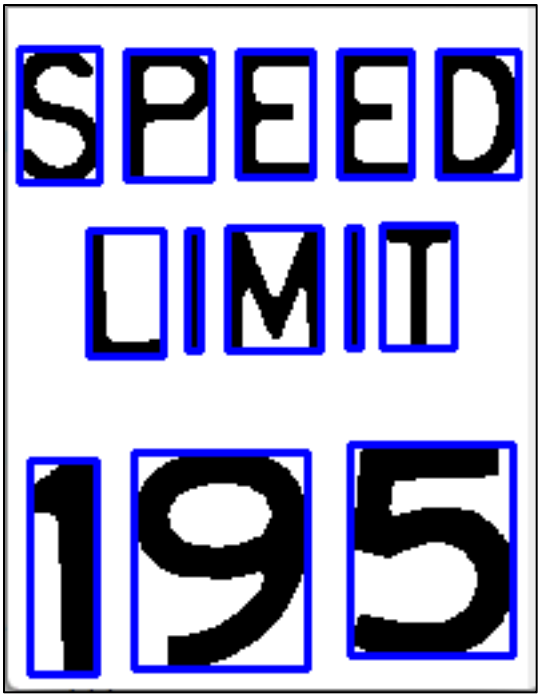

Figure 1.24(a) Test night segmented image

Figure 1.24(b) Test shade segmented image

- Thus, we now have the character segments from both the images.
- We can now extract features from each of these ROI independently.
- The feature extraction procedure has been explained in section 1.1.2.
- Thus, we now have 8 features of every character.
- Classification by using the decision tree that was generated by using the training image will now be performed.
- The process remains the same as it was for the previous test images.
- Thus, we now have the assigned labels by our decision tree classifier and the features of all the characters.

1.3.3 Results

The OCR labels, true labels and the extracted features for the test night image are shown in the following table:

Character	True label	Assigned label	Euler No.	Total vertical intersections	Total horizontal intersections	Symmetry	% object pixels in first 10 rows	% object pixels in last 10 rows	% object pixels in first 10 columns	Total intersections of bottom line
S	k	k	1	3	1	67.21%	59.35%	60.96%	49.80%	1
P	l	l	0	2	1	61.63%	69.69%	17.87%	70.81%	1
E	m	m	1	3	1	86.63%	65.17%	66.20%	73.95%	1
E	m	m	1	3	1	88.69%	65.35%	68.92%	77.29%	1
D	n	n	0	2	2	94.08%	70.96%	66.77%	72.70%	1
L	o	o	1	1	1	77.87%	20%	61.03%	61.45%	1
I	p	r	1	1	1	88.40%	98.33%	83.33%	92.60%	1
M	q	q	1	1	4	58.86%	45%	33.33%	71.91%	3
I	p	p	1	1	1	98.55%	98.33%	96.66%	98.91%	1
T	r	r	1	1	1	78.41%	72.85%	20.35%	12.55%	1
1	b	r	1	1	1	75.11%	63.07%	48.84%	16.29%	1
9	j	j	0	3	1	56.39%	62.57%	31.96%	30.97%	1
5	f	f	1	3	2	52.05%	77.58%	70.80%	48.25%	1

Table 1.4 OCR for advanced test image 1

The OCR labels, true labels and the extracted features for the test shade image are shown in the following table:

Character	True label	Assigned label	Euler No.	Total vertical intersections	Total horizontal intersections	Symmetry	% object pixels in first 10 rows	% object pixels in last 10 rows	% object pixels in first 10 columns	Total intersections of bottom line
S	k	c	1	3	1	73.2%	67.5%	78%	75.2%	1
P	l	l	0	2	1	55.83%	87%	35.5%	80.41%	1
E	m	m	1	3	1	88.18%	73.68%	75.26%	86.4%	1
E	m	m	1	3	1	90.66%	75%	78.33%	87.2%	1
D	n	n	0	2	2	88.09%	80%	75.71%	77.5%	1
L	o	o	1	1	1	68.85%	36.31%	74.21%	77.08%	1
I	p	p	1	1	1	100%	98.33%	100%	99.16%	1
M	q	q	1	1	1	83.27%	80.86%	89.13%	90%	3
I	p	r	1	1	1	89.655%	90%	98.57%	96%	1
T	r	r	1	1	1	69.29%	74.73%	31.57%	49.58%	1
2	c	c	1	3	1	64.84%	71.73%	92.60%	54.57%	1
5	f	f	1	3	2	55.63%	82.39%	76.30%	62.37%	1

Character	True label	Assigned label	Euler No.	Total vertical intersections	Total horizontal intersections	Symmetry	% object pixels in first 10 rows	% object pixels in last 10 rows	% object pixels in first 10 columns	Total intersections of bottom line
Y	x	x	1	1	1	43.56%	54.76%	20%	39.13%	1
O	a	n	1	2	2	92.20%	65.78%	64.21%	62.17%	1
U	x	x	1	1	2	61.84%	50%	63.68%	53.75%	1
R	x	x	0	2	1	66.05%	78.42%	61.05%	72.60%	2
S	k	K	1	3	1	65.21%	53.88%	62.22%	63.91%	1
P	l	l	1	2	1	56.88%	70%	27.36%	60.86%	1
E	m	d	1	3	1	88.71%	58.82%	58.82%	73.47%	1
E	m	m	1	3	1	92.30%	67.05%	64.70%	80.43%	1
D	n	n	0	2	2	92.66%	70.52%	70%	71.73%	1

Table 1.5 OCR for advanced test image 2

1.3.4 Discussion

The main pre-processing steps that have been performed are histogram equalization, morphological operations of dilation and closing and adaptive thresholding. For the first advanced testing image, two characters have been misclassified i.e. 'I' and '1'. They have been classified as 'T', labeled as 'r'. If we look at the decision tree, we can see that '1','T' and 'I' are separated using the feature of symmetry and then using the percentage of object pixels in the first 10 rows. For 'I', the symmetry value is 88.4% whereas the threshold is 90%. Thus it got misclassified as 'T'. For '1', the percentage of object pixels in the first 10 rows is 63.07% whereas the threshold is 50% and thus it got misclassified as 'T'. For the second test image, we got a total of 4 misclassifications out of 21 characters. The characters 'Y', 'U' and 'R' got labeled as 'X' because they were not recognized by the OCR system. This is because they were not present in the training image. The character 'S' got misclassified as '2', the character 'I' got misclassified as 'T', the character 'O' got misclassified as 'D' ('O' is same as '0' and hence it was recognized by OCR system) and 'E' got misclassified as '3'. The reason of misclassification is same as before that the threshold values at the respective nodes were not met for those particular features. In conclusion, we can say that this OCR system performs quite well on the advanced images.

2. Problem #2: Contour Modeling

2.1. Motivation

Contour modeling is mainly used for the detection of edges, lines and contours. It is a technique that is used to extract object outlines from images. Its main applications are in the field of medical image analysis, shape recognition, segmentation and object tracking. In this problem, we will consider two different techniques of contour modeling namely Snakes active contour modeling and Level-set active contours without edges.

2.2. Approach and Implementation

2.2.1 Snakes: Active contour models

- A snake is an energy minimizing spline.
- It is formed with the help of external constraint forces as well as the image forces.
- Snakes are active contour models that are pulled toward lines and edges and are then locked onto nearby edges.
- In this algorithm, we start with an initial contour near the desired object and then we keep on optimizing this contour till it fits the exact object boundary.
- The basic steps of this algorithm are:
  Step 1: Use the contour obtained a frame N as the initial contour for frame N+1.
  Step 2: Optimize the initial contour to fit the exact object boundaries for frame N+1.
  Step 3: Repeat the above 2 steps till we reach the total number of iterations N' i.e. till N' = N+1.
- The total energy of snake is a combination of three energies i.e. internal, image forces and external.
- The internal energy encourages smoothness as well as obtains prior knowledge about object boundary.
- The image forces push the snake toward image features like lines, edges and subjective contours.
- The external energy encourages the curve to fit onto image edges.

- The internal energy is given by:

$$E_{\text{int}} = (\alpha(s)|\mathbf{v}_s(s)|^2 + \beta(s)|\mathbf{v}_{ss}(s)|^2)/2$$

  Where the first term decides the elasticity and the second term decides the stiffness
- The image energy is given by:

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + W_{term}E_{term}$$

  Where E_{line} is the line functional that is the image intensity, E_{edge} is the edge functional and E_{term} is the termination functional.
- The external energy is given by:

$$E_{external}(\nu(s)) = -(\,|\,G_x(\nu(s))|^2 + |\,G_y(\nu(s))|^2\,)$$

  And this describes how well the curve matches the image data locally.
- Now, we want to find a contour that will minimize the combination of the above three energies.
- The iterative optimization of the contour, that minimizes the total energy, is done by using the gradient descent approach or by dynamic programming.

2.2.2 Active contours without edges

- The basic idea stays the same i.e. given an image; we want to evolve a curve to detect objects in that image and the curve must stop at the boundaries of the object.
- This algorithm assumes that the image is formed by two regions of piecewise constant intensities, say U_0^i and U_0^o i.e. inside the object and outside the object respectively.
- The fitting term is then given by:

$$F_1(C) + F_2(C) = \int_{inside(C)} |u_0(x, y) - c_1|^2 \, dx \, dy + \int_{outside(C)} |u_0(x, y) - c_2|^2 \, dx \, dy$$

  Where C is the variable curve and the constants c_1 and c_2 are derived from the averages of U_0^i and U_0^o.
- We can say that if the curve is inside the object or outside the object or both inside and outside the object, then the fitting energy will be grater than zero.
- Thus, in conclusion, the fitting energy is minimized only when the curve C is on the boundary of the object.
- This fitting term along with the regularizing terms of length of the curve and the area of the region inside the curve gives us the final energy term that we want to minimize in this algorithm.
- The energy functional is given by:

$$F(c_1, c_2, C) = \mu \cdot \text{Length}(C) + \nu \cdot \text{Area}(inside(C)) + \lambda_1 \int_{inside(C)} |u_0(x, y) - c_1|^2 \, dx \, dy + \lambda_2 \int_{outside(C)} |u_0(x, y) - c_2|^2 \, dx \, dy,$$

- Now, we just replace C by the level function Φ that defines the evolving curve.

- The final energy term that has to be minimized is defined as:

$$
\begin{aligned}
&F(c_1, c_2, \phi) \\
&\quad = \mu \int_{\Omega} \delta(\phi(x, y))|\nabla\phi(x, y)|\, dx\, dy \\
&\qquad + \nu \int_{\Omega} H(\phi(x, y))\, dx\, dy \\
&\qquad + \lambda_1 \int_{\Omega} |u_0(x, y) - c_1|^2 H(\phi(x, y))\, dx\, dy \\
&\qquad + \lambda_2 \int_{\Omega} |u_0(x, y) - c_2|^2 (1 - H(\phi(x, y)))\, dx\, dy.
\end{aligned}
$$

- The flow of the algorithm is as follows:
  Step 1: Initialize $\Phi^n$and set n to 0
  Step 2: Calculate $c_1(\Phi^n)$ and $c_2(\Phi^n)$
  Step 3: Deduce the Euler-Lagrange equation for Φ by using gradient descent and thus obtain Φ^{n+1}
  Step 4: Check if the solution has converged. If not, set n = n+1 and repeat the above steps till convergence is met.
- Generally, λ_1= λ_2=1 and ν = 0. The length parameter, μ, is kept small if we want to detect all the objects and large if we want to detect only the large objects.

2.3 Results

For implementing the snake active contour modeling, the source code given in [1] has been used. It has a GUI that lets the user to adjust parameter values of alpha, beta, gamma, kappa, w-line, w-edge, w-termination and iterations. Moreover, the user needs to specify the initial contours points around the object. Let us study the effects of each of these attributes.

a) Initial contour points

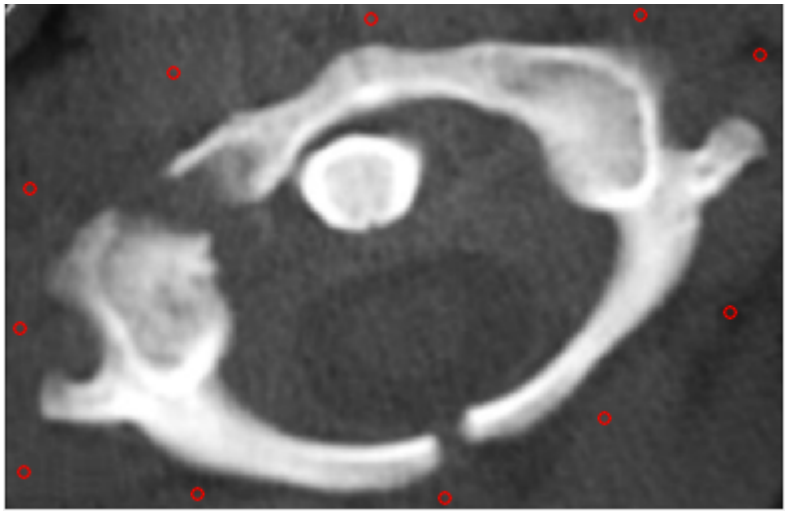

Figure 2.1 (a) Initial contour points away from the object boundary

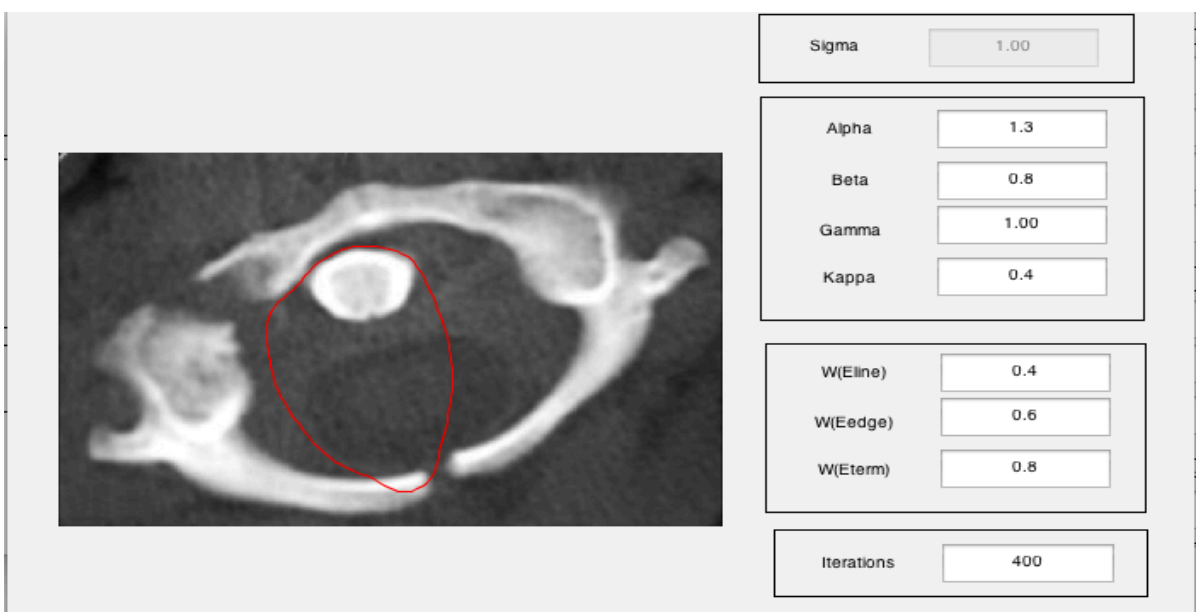

Figure 2.1(b) Snake contour for the incorrect initialization

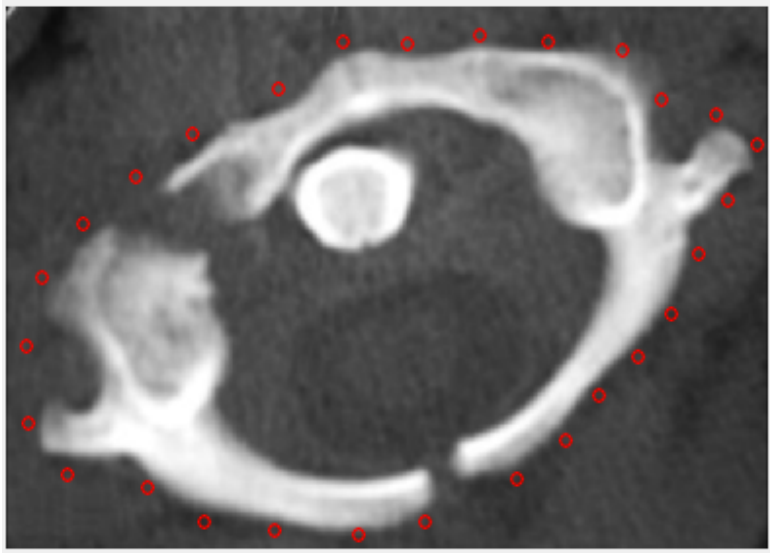

Figure 2.1(c) Initial contour points near the object boundary

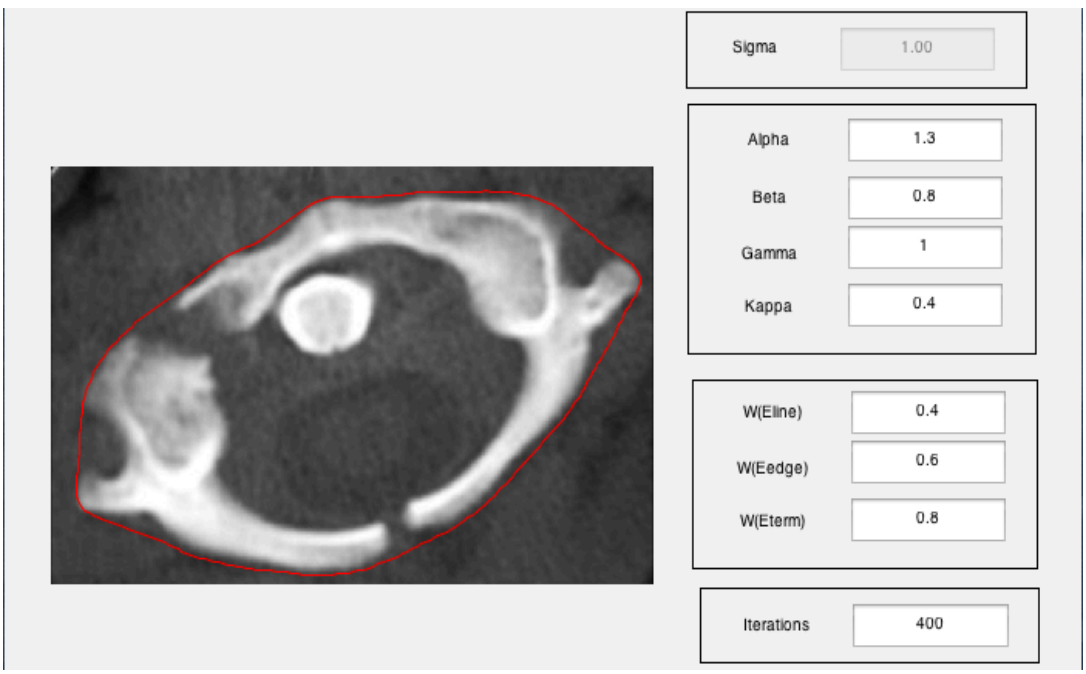

Figure 2.1(d) Snake contour for the correct initialization

From the above figures, we can see that when the initialization points (red circles) are away from the true object boundary, as shown in figure 2.1(a), we get a bad contour result (red boundary), as shown in figure 2.1(b). On the other hand, when the initialization points are close to the true object boundary, as shown in figure 2.1(c), we get the true contour of the object, as shown in figure 2.1(d).

b) *Alpha*

This term controls the tension of the snake. In other words, this term governs the elasticity of the contour. Let us see the effect by reducing it to zero.

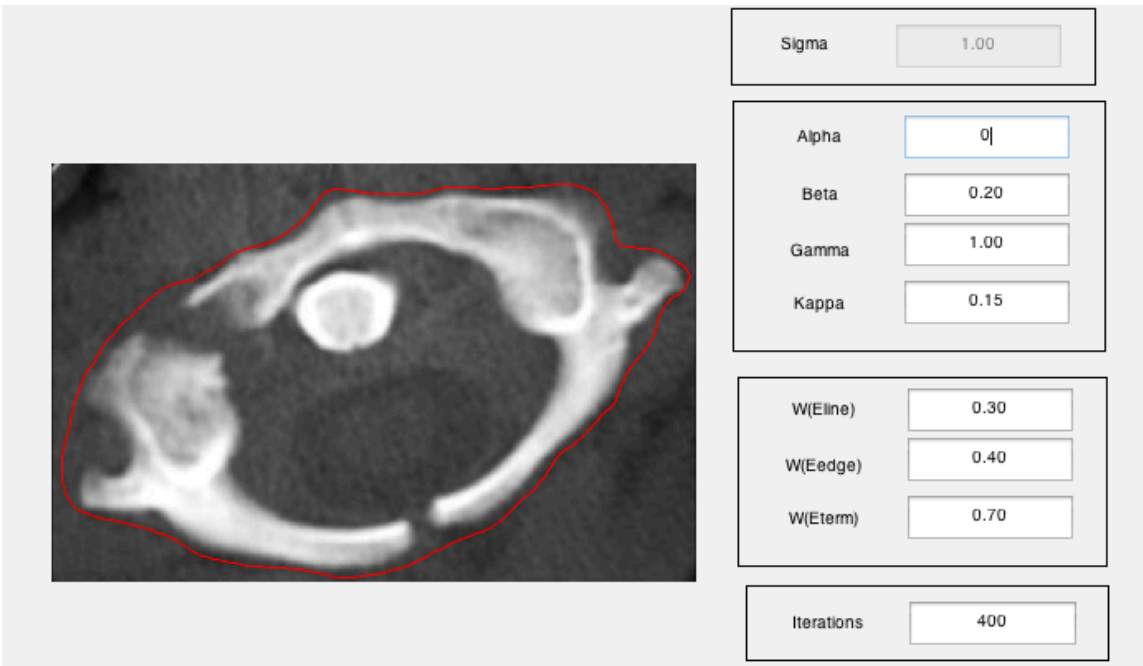

Figure 2.2 Snake contour with alpha = 0

Thus, from the above figure, we can see that tension is reduced due to which the contour is less elastic. The figures that follow will show the effect of non-zero alpha values.

c) *Beta*

This term controls the rigidity of the contour. If the value of beta is kept low, then the contour will be less stiff. Let us see the effect by reducing its value to 0.

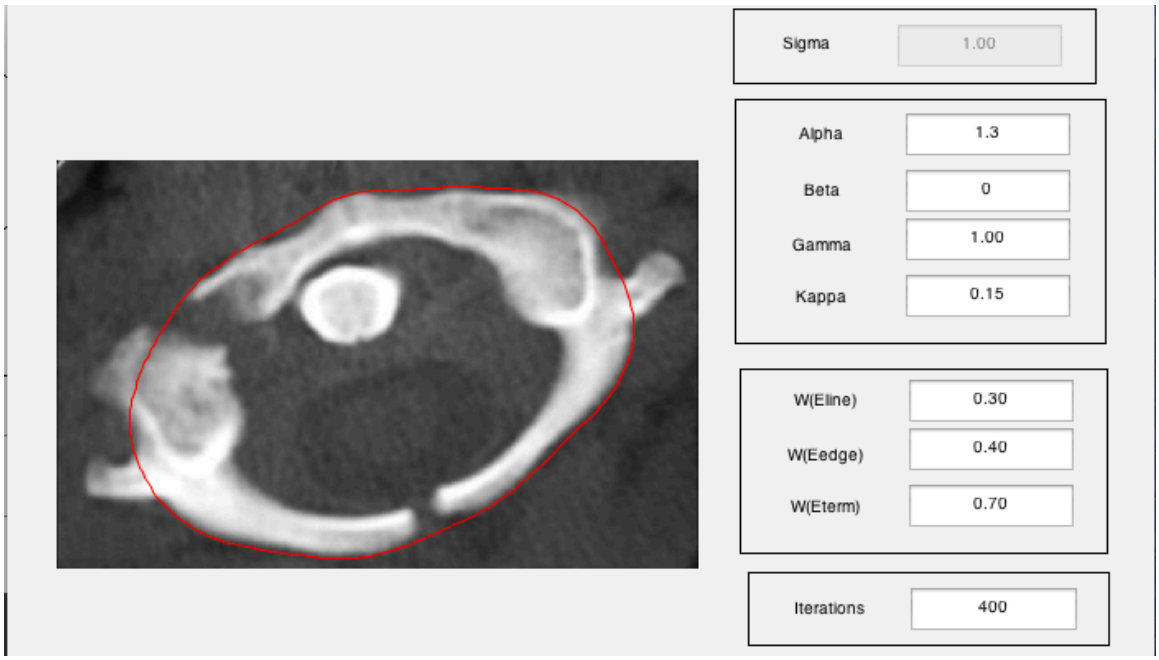

Figure 2.3 Contour with beta = 0

d) *Gamma*

This term decides the step size. It is related to the external energy. Let us see the effect of increasing it to 4.

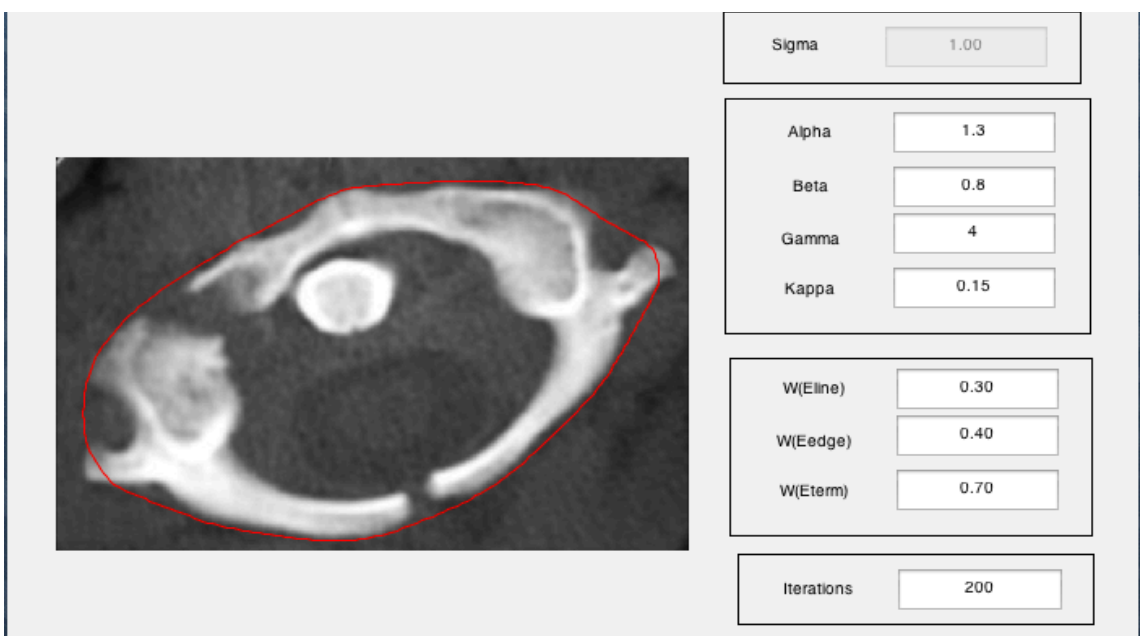

Figure 2.4 Contour with gamma = 4

e) *Kappa*

This parameter controls the energy term. Let us see the importance of this parameter by setting it to zero.

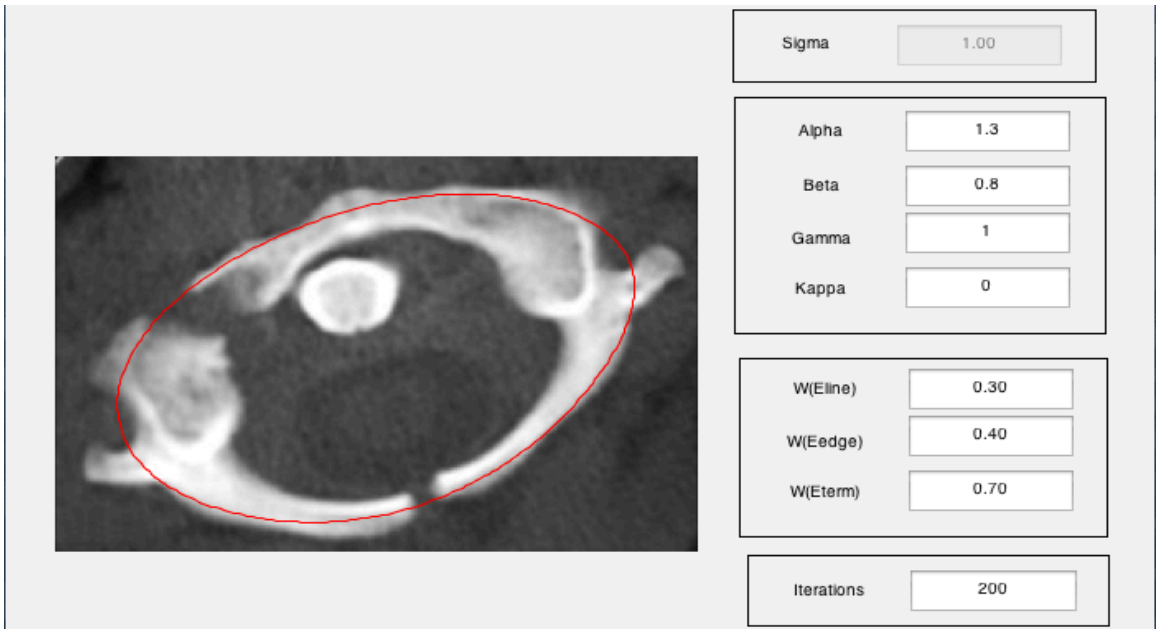

Figure 2.5 Contour with kappa = 0

We can observe that the contour has now entered the object region and thus resulted into an incorrect boundary.

f) *W-line and w-edge*

These three parameters control the weights for line, edge and terminal energy components. The effect of setting W-line to 0 is as follows:

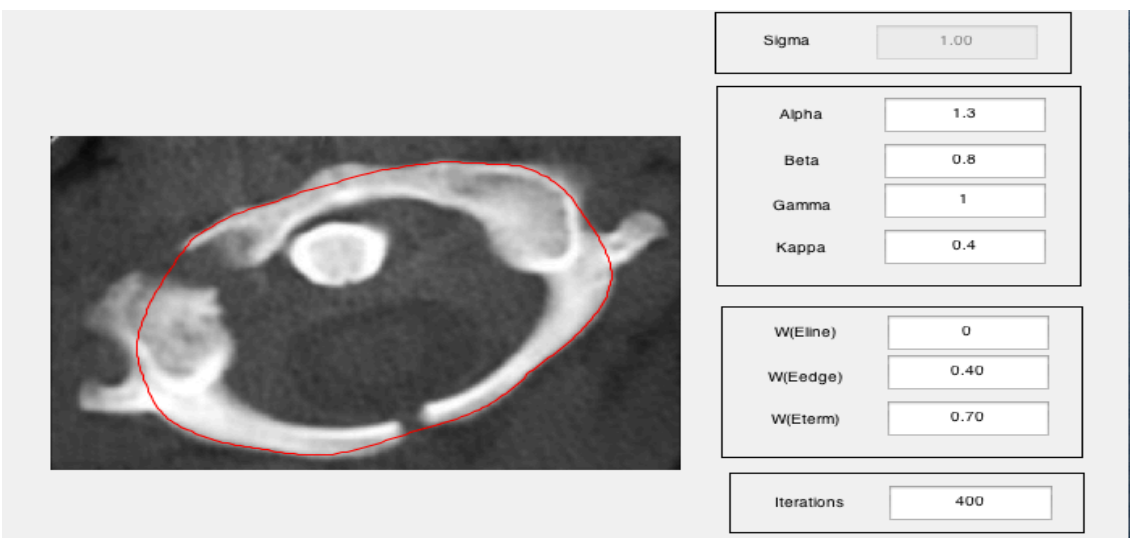

Figure 2.6 Contour with W-line = 0

The effect of setting W-edge to 0 is as follows:

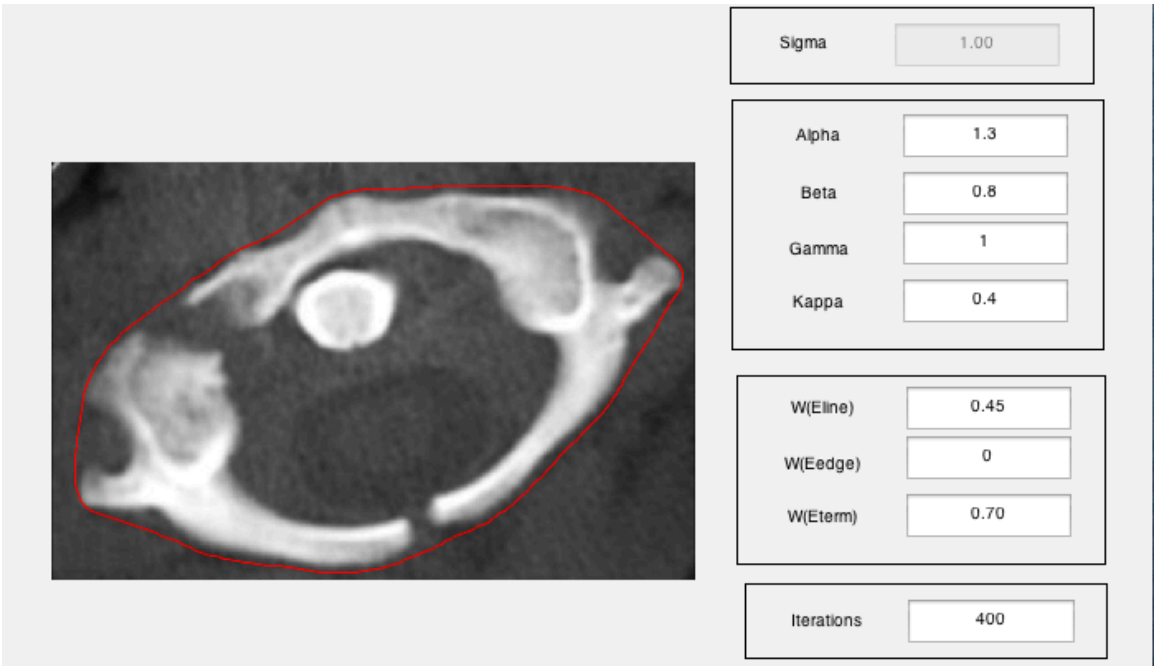

Figure 2.7 Contour with W-edge = 0

g) *Iterations*

If the number of iterations is smaller than required, then the contour will not converge and thus we will not get an accurate boundary. The following contour was obtained by setting the number of iterations to 10:

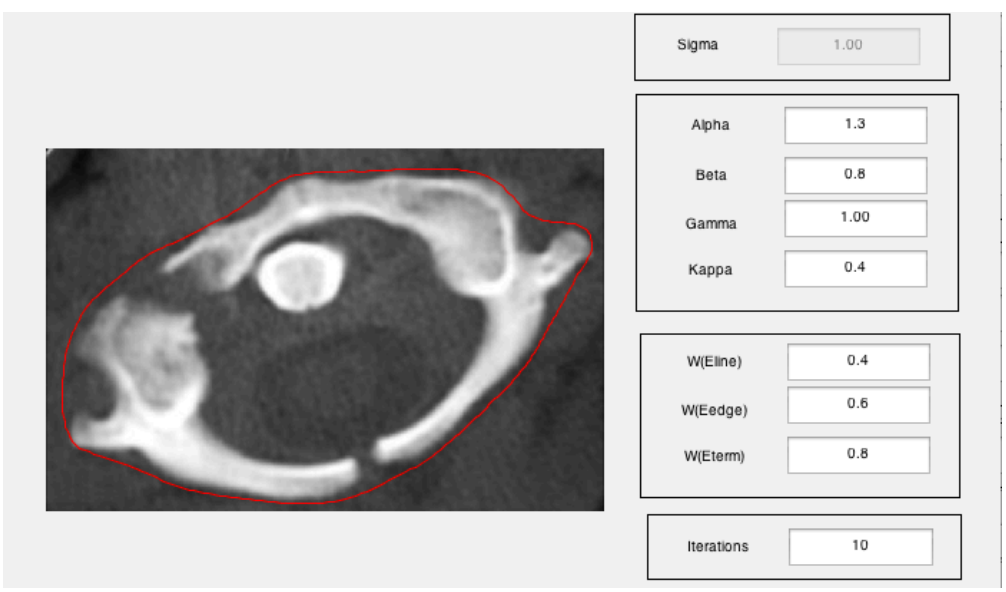

Figure 2.8 Contour with iterations = 10

The final contours obtained from extraction of the spine image, coronary outline, separation of red and white blood cells and isolating the brain tumor are as shown below:

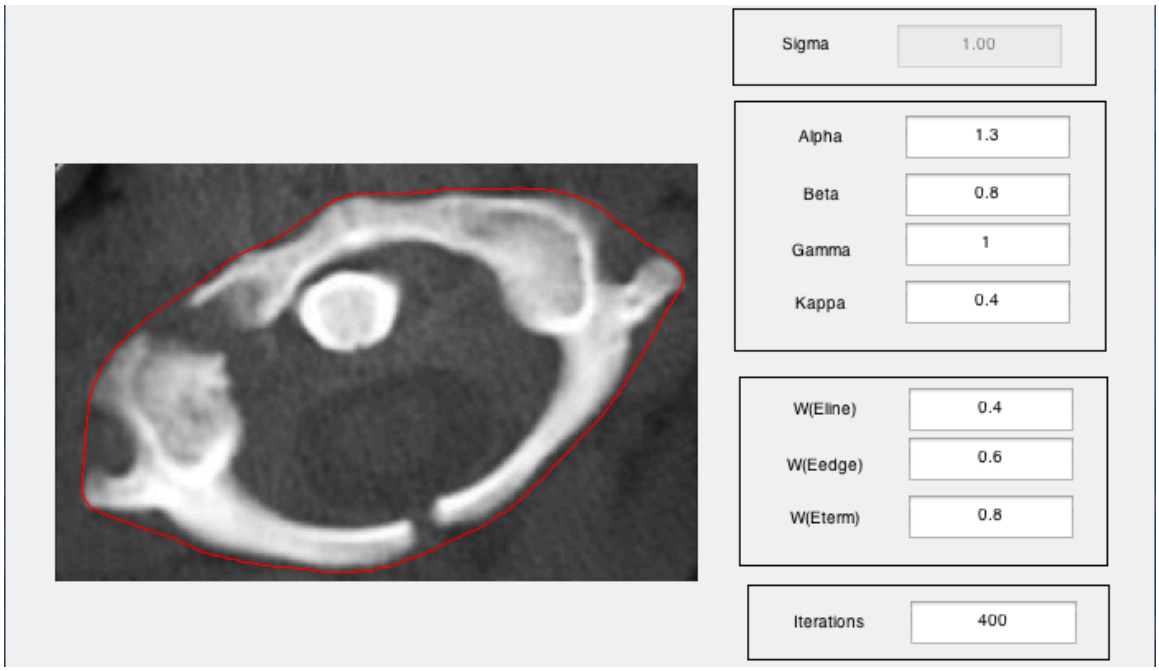

Figure 2.9 Extraction of the spine contour

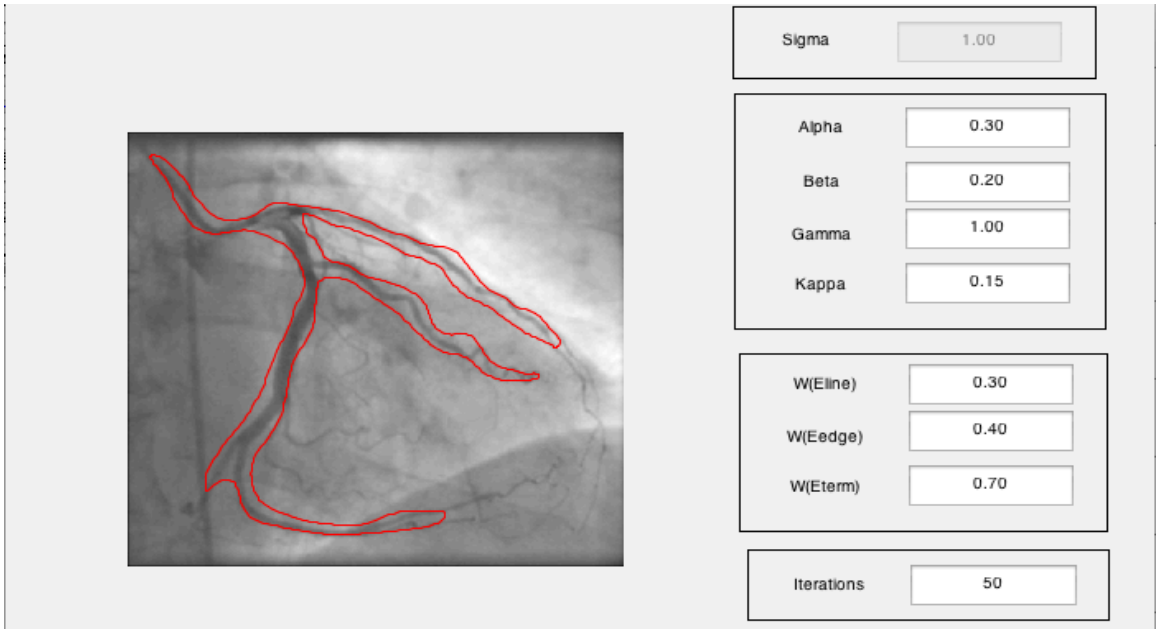

Figure 2.10 Extraction of the coronary outline

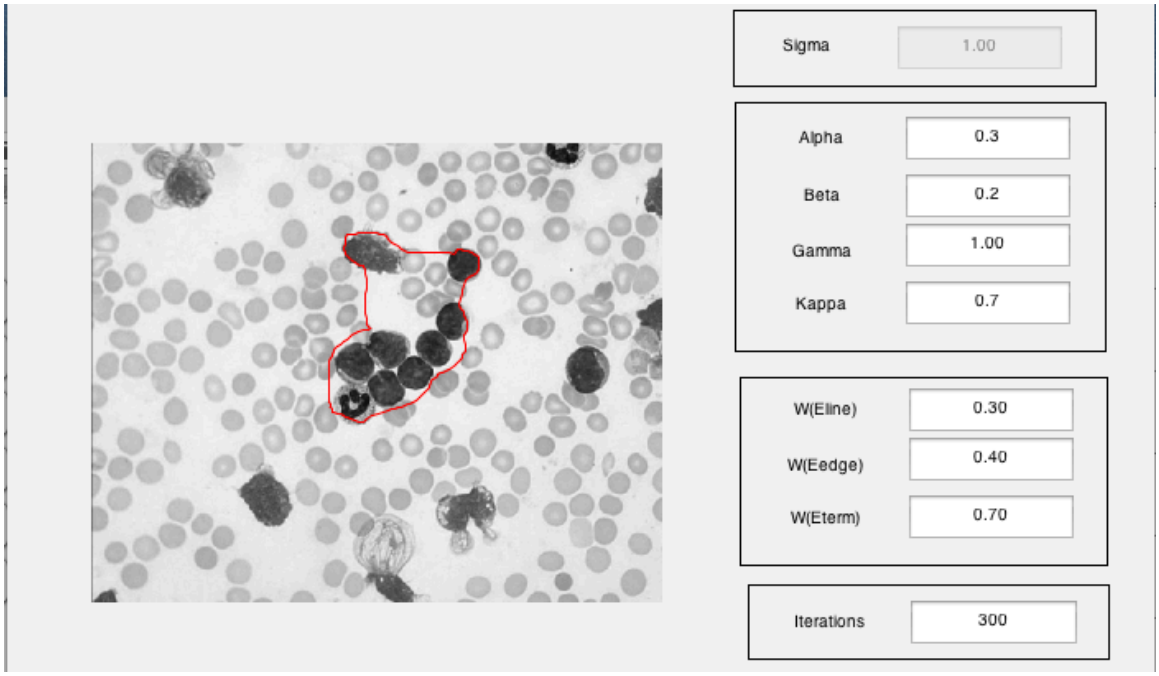

Figure 2.11 Separation of white and red blood cells

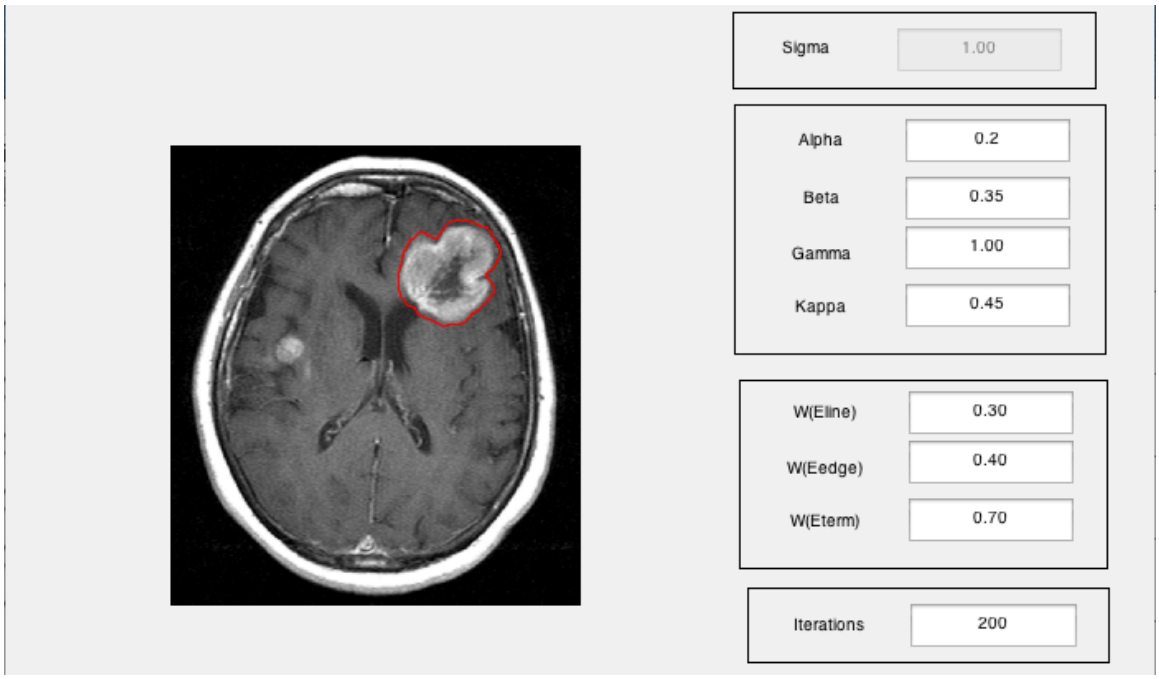

Figure 2.12 Isolation of the tumor in brain MRI

The contours obtained from the level set method of active contours without edges are as shown below:

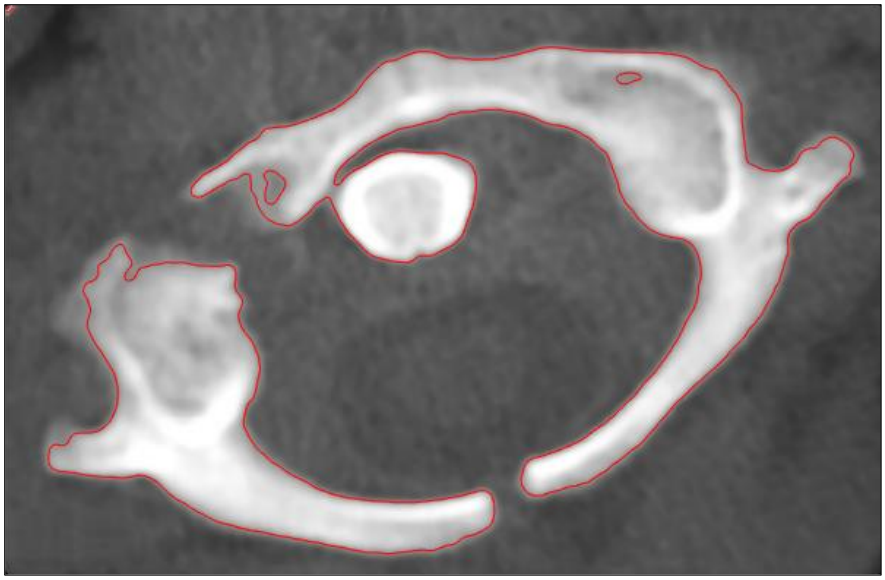

Figure 2.13(a) Level set contour of spine

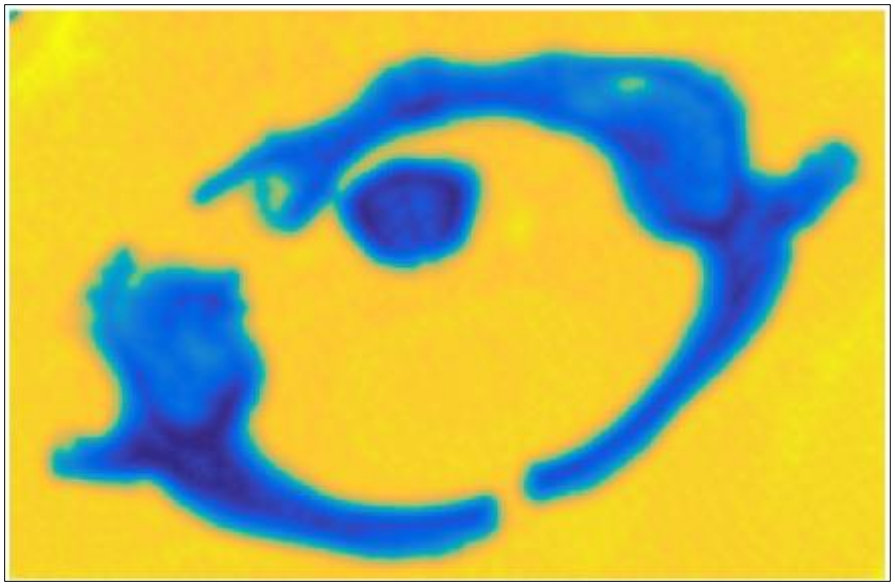

Figure 2.13(b) Final level set function of spine after 400 iterations

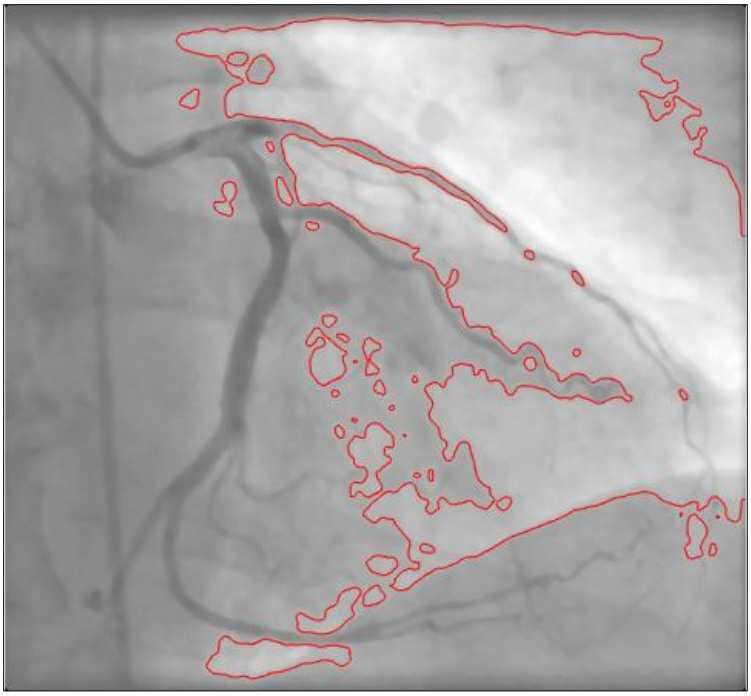

Figure 2.14(a) Level set contour of coronary

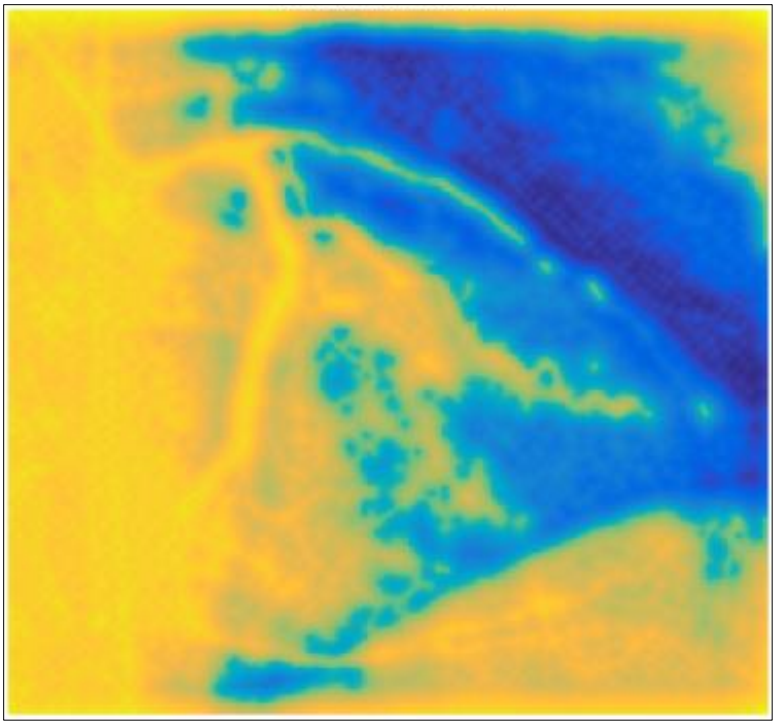

Figure 2.14(b) Final level set function of coronary after 600 iterations

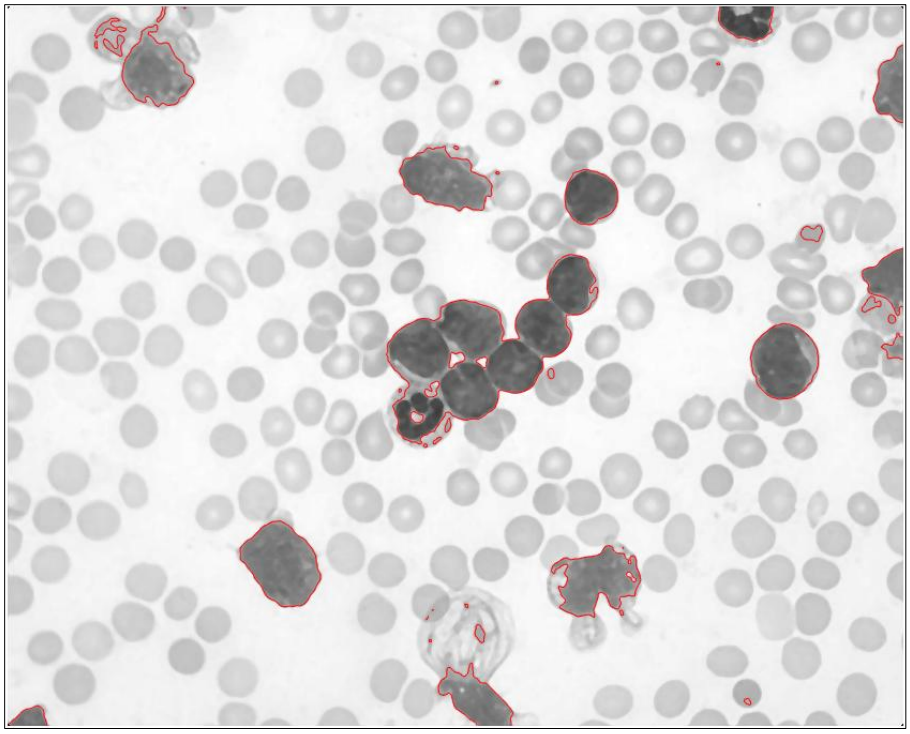

Figure 2.15(a) Level set contour of blood cells

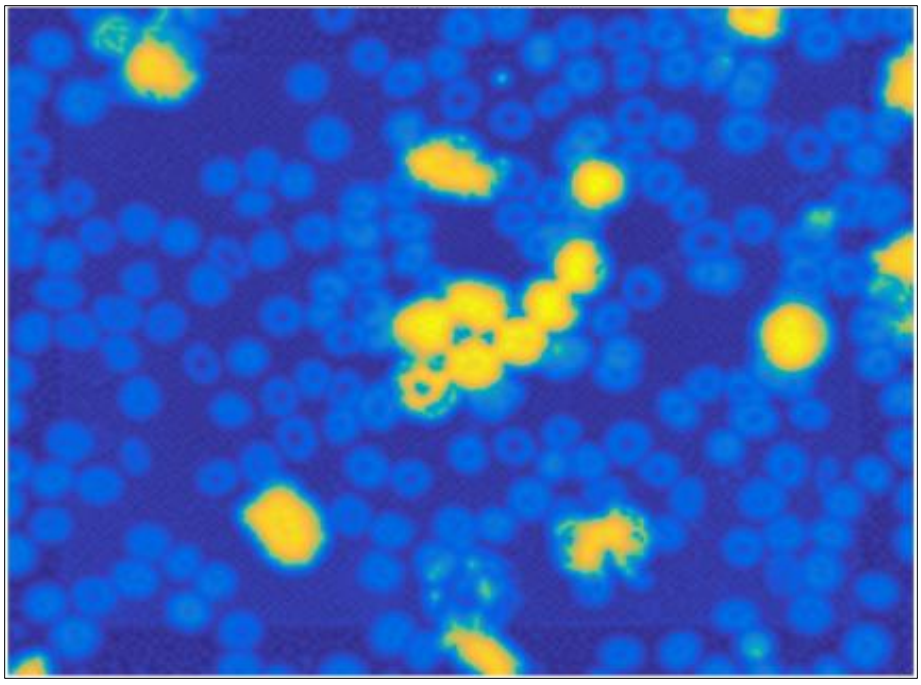

Figure 2.15(b) Final level set function of blood cells for 600 iterations

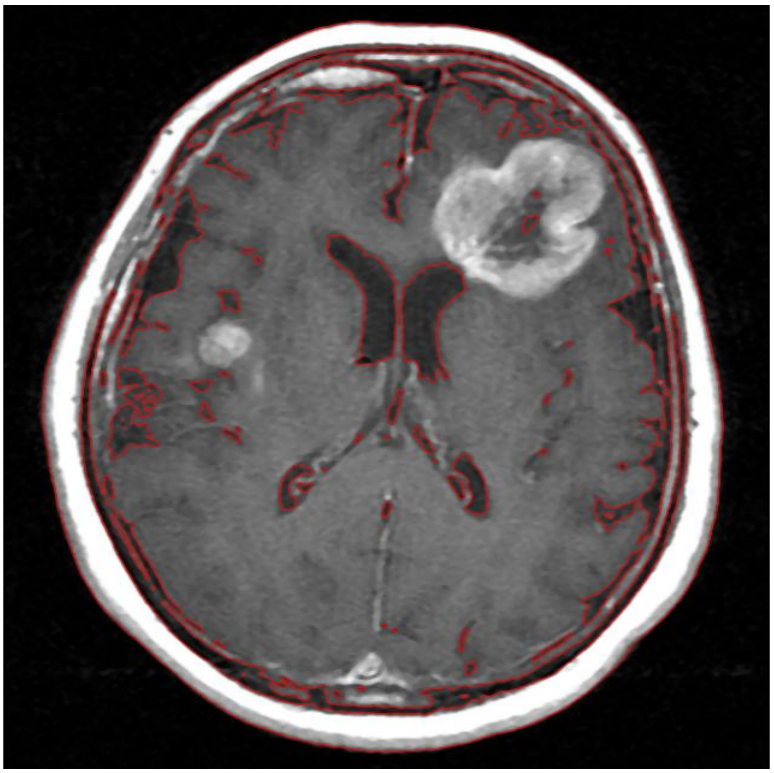

Figure 2.16(a) Level set contour of brain

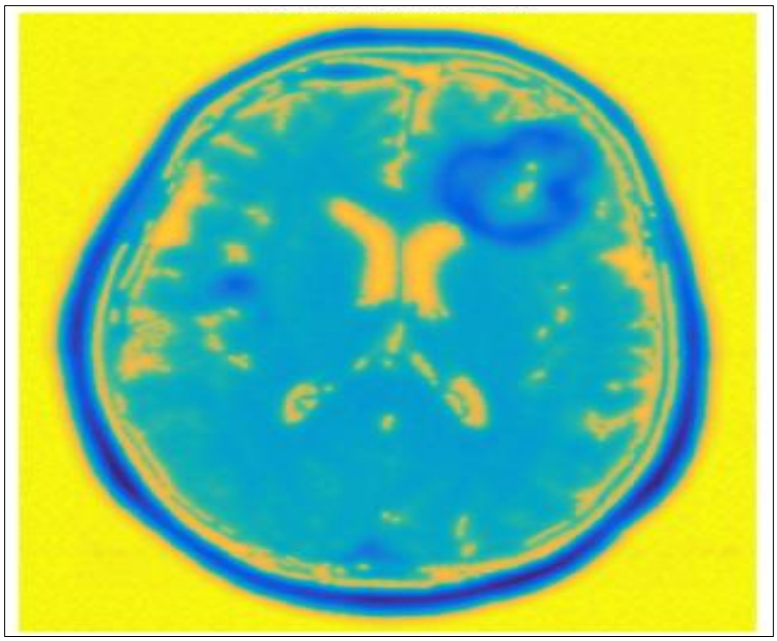

Figure 2.16(b) Final level set function of brain for 400 iterations

2.4 Discussion

Thus, a simple elastic snake is defined by a set of points, the internal elastic energy term and the external edge based energy term. We first need to initialize the contour in the vicinity of the object and then optimize the points to get the final contour that minimizes the total energy. The limitations of this algorithm are that the snake does not see the object boundaries unless it gets very close to it. Moreover, it may over smooth the boundary and it cannot follow topological changes of the object. The advantages of snake are that it is useful to track and fit non-rigid shapes, the contour remains connected, the user can suitably define energy function and weight with flexibility and that the snake contour fills in the subjective contours. The disadvantages of snake algorithm are that we must have a decent initialization near the true boundary and we need to have some prior information to set the function parameters. The snake algorithm performs well on isolating the brain tumor and detecting the spine. It gives bad results for separation of red and white blood cells whereas it gives moderate results for coronary outline detection. Level set algorithm is a more effective way to detect contours. The advantages of level set algorithm are that it automatically detects interior contours, it is efficient in case of concave objects and that noise does not affect the functioning of this algorithm. Moreover, unlike snake, the initial curve can be placed anywhere and it follows changes in the topologies of an object. Level set algorithm gives very good results for spine and separation of red and white blood cells. It gives moderate results for the brain and coronary contours. Thus, in conclusion we can say that level set algorithm needs fewer parameter adjustments than the snake algorithm and that it is and more efficient.

3. Problem #3: Salient Point Descriptors and Image Matching

3.1. Extraction and Description of Salient Points

3.1.1 Motivation

For applications like object recognition, gesture recognition and video tracking, we first need to detect and describe the local features in images. The points of interest of an object within an image can be extracted to get the feature description of that object. These features should be reliable enough so that we can detect objects even if the image is scaled or contains noise. This feature extraction and description can be done with the help of two algorithms namely Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF).

3.1.2 Approach and Implementation

3.1.2.1 Scale Invariant Feature Transform (SIFT)

- This technique is used to get the image features and descriptors for image matching and recognition
- SIFT transforms the image data into co-ordinates that are scale invariant.
- The main stages of these algorithm are as follows:
  1. Scale space extrema detection
  2. Keypoint Localization
  3. Orientation Assignment
  4. Keypoint Descriptor
- The following theory describes the details of these stages:

1. Scale space extrema detection

- A continuous function known as scale space is used to search the stable features within an image.
- These stable features allow the detection of locations that are invariant to the scale change of an image.

- Now, this scale space function is defined as follows:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Where *G(x,y,σ)* is a Gaussian function defined as

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

- Thus, the scale space function *L(x,y,σ)* is the convolution of the Gaussian function and the image.
- Now, as mentioned before, we need stable features to detect invariant locations. Thus, instead of using the scale space function, we use the scale space extrema function that convolves the image with the difference of Gaussian function (instead of Gaussian function as shown above).
- The scale space extrema function is defined as:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned}$$

  Where the image is convolved with the difference of Gaussian functions of two scales that are separated by a factor of k.
- Now, we want to detect the local maxima and minima of *D(x,y,σ).*
- To do that, each sample point is compared to its 26 neighbors that comprise of 8 neighbors in the current image and 9 neighbors each from the scale above the image and below the image respectively as shown below:

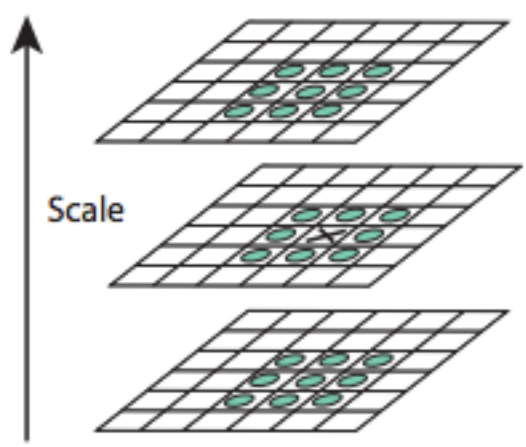

Figure 3.1 Comparison of pixels to its 26 neighbors [2]

- This sample point is selected only if it is either larger or smaller than all of these 26 neighbors.

- Now, to detect the extrema, we need to choose a proper frequency of sampling in the image domain and the scale domain.
- The scale space DoG function has a large number of extrema and we only want to detect the stable ones from them. To do this we use coarse sampling of scales in the scale domain.
- Extrema can be very close together and thus in image domain we can perform image doubling by using interpolation. This process increases the number of stable keypoints.

2. <u>Keypoint Localization</u>

- So till this point we have found potential keypoint candidates by comparing pixels to its 26 neighbors.
- After this, we need to obtain the location, scale and ratio of principal curvatures of these keypoints so that we can reject the redundant ones.
- The location of the extremum, $\hat{x}$, is obtained as follows:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}.$$

  where D(x) is the taylor expansion of the scale space function D(x,y,σ)
- The keypoints that are probable for rejection are those that have low contrast and poor localization along edges.
- The function value at the extremum D($\hat{x}$) is used to reject the unstable low contrast extrema.
- The principal curvature is obtained by using a Hessian matrix, H, which is computed at the location and scale of the keypoint.
- The ration of principal curvatures is nothing but the ratio of the square of trace of H divided by the determinant of H.
- By comparing this ratio with an appropriate threshold value, we can now reject the keypoints that have poor localization along edges.

3. Orientation assignment

- Now, once we have found the final stable keypoints, we can assign orientations to these keypoints by using the local image properties.
- First, the scale of the keypoint (that we obtained in the previous stage), is used to select the Gaussian smoothed image L.
- Now, for each image sample L(x,y), we compute the gradient magnitude, m(x,y), and orientation, θ(x,y) at that specific scale.
- This is done as follows:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

- Now, an orientation histogram is formed with the help of the sample point's gradient orientation.
- This, histogram has 36 bins for the 360 degree range of angles.
- The gradient magnitudes, as well as a Gaussian circular window, weight the image samples that were added to these histograms.
- The highest peaks in the histogram are now detected and thus the orientation is obtained.
- For local peaks that are within 80% of the highest peak are also used to create keypoints with that orientation.
- In short, we can have multiple keypoints at the same location and scale but they will have different orientations.

4. Keypoint Descriptor

- We have now obtained the image location, scale and orientation for every stable keypoint of the image.
- The next thing that has to be computed is the keypoint descriptor.
- To do this, a rectangular grid of 4 × 4 is laid on the keypoint in the image domain, such that the orientation of this grid is determined by the histogram peak.
- Now, for each point of the grid, a local histogram of gradient directions is calculated.
- These gradient directions are quantized into 8 discrete directions.

- A Gaussian window weights the entries in these histograms by assigning weights to the magnitude of each sample point.
- Thus, a local histogram that is calculated over 4 × 4 grid points and with 8 directions give rise to a descriptor with 4 × 4× 8 = 128 dimensions for every keypoint.
- This is known as the SIFT descriptor.
- The figure that depicts how the SIFT descriptor is computed from sampled values of the grid orientation and magnitude is shown below:

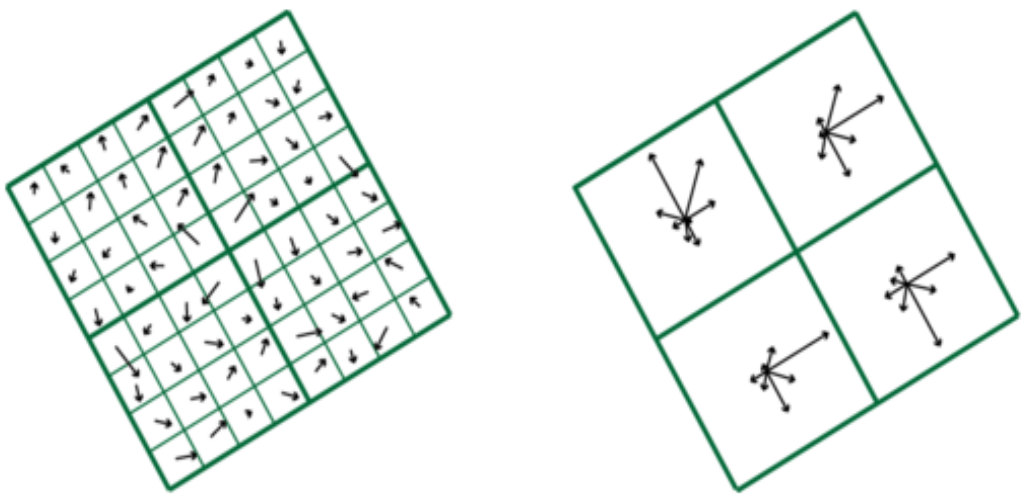

Figure 3.2 SIFT descriptor computation [3]

Thus, we have studied the SIFT technique that extracts features and descriptors from the image objects.

3.1.2.2 Speeded Up Robust Feature (SURF)

- This is another technique that extracts features and descriptors of the image objects.
- This algorithm has the following flow:
  Step 1: Find image interest points
  Step 2: Scale space representation of points of interest
  Step 3: Find feature direction
  Step 4: Compute feature vectors
- Let us study the details of the above steps.

<u>1. Find the image interest points</u>

- SURF uses a blob detector based on the Hessian matrix to find points of interest.
- This is done by using a Hessian matrix which contains the second order partial derivatives as follows:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}$$

  Where L_{xx}(x,σ) is the convolution of the Gaussian second order derivative and the image in point x (and similarly L_{xy}(x,σ) and L_{yy}(x,σ) are convolutions of Gaussian derivatives and the image)
- The box filter of size 9 × 9 is an approximation of a Gaussian with σ = 1.2 and it represents the highest spatial resolution.
- These approximations are denoted by D_{xx}, D_{xy} and D_{yy}.
- Now, blob responses at location x = (x,y,σ) are obtained as follows:

$$\det(\mathcal{H}_{\text{approx}}) = D_{xx}D_{yy} - (0.9D_{xy})^2$$

<u>2. Scale space representation of points of interest</u>

- Scale spaces are usually implemented as image pyramids like we saw in the SIFT algorithm.
- In SURF, as we have used the box filters and integral images, we apply box filters of different sizes instead of iteratively reducing the image size.

- The output of the 9 × 9 filter is considered as the initial scale layer to which we refer as scale s = 1.2.
- The following layers are obtained by filtering the image with bigger masks that take into account the discrete nature of the integral images as well as the filter structure.
- Thus, filters of size 9 × 9, 15 × 15, 21 × 21, 27 × 27 etc. are used.
- Now, to localize the interest points, non-maximum suppression in a 3×3×3 neighborhood is applied.
- The scale space interpolation is then carried out where the maxima of the hessian determinant are interpolated.

3. Find feature direction

- To achieve rotational invariance, the orientation of the points of interest must be found.
- For this purpose, the haar wavelet response is used.
- The responses in x and y directions within a circular neighborhood of radius 6s, where s is the scale of that point of interest, are calculated at that point of interest.
- These responses are then weighted by Gaussian function.
- Now, a 2D plot of the x response in the abscissa and the y response in the ordinate is generated.
- The sum of all the responses within a sliding window of size π/3 is calculated and is then used to estimate the dominant orientation.
- The sum of the horizontal and vertical responses within this window gives the local orientation vector.
- The longest vector defines the orientation of that point of interest.

4. Compute feature vectors

- First, a square window of size 20s is constructed around the point of interest along the orientation that was obtained in the previous step.
- This region is now split into smaller 4 × 4 sub regions and then for each sub region, features at 5 × 5 regularly sampled points are computed.
- The haar wavelet transforms in the horizontal and the vertical directions are d_x and d_y respectively.

- These responses are first weighted with a Gaussian and are then summed up over each sub region.
- Now, we also calculate the sum of the absolute values of these responses, $|d_x|$ and $|d_y|$.
- Thus, each sub region has a four dimensional descriptor vector as V = $(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$
- This results in a descriptor vector for all 4 × 4 sub regions of length 64.
- This idea is depicted in the following figure:

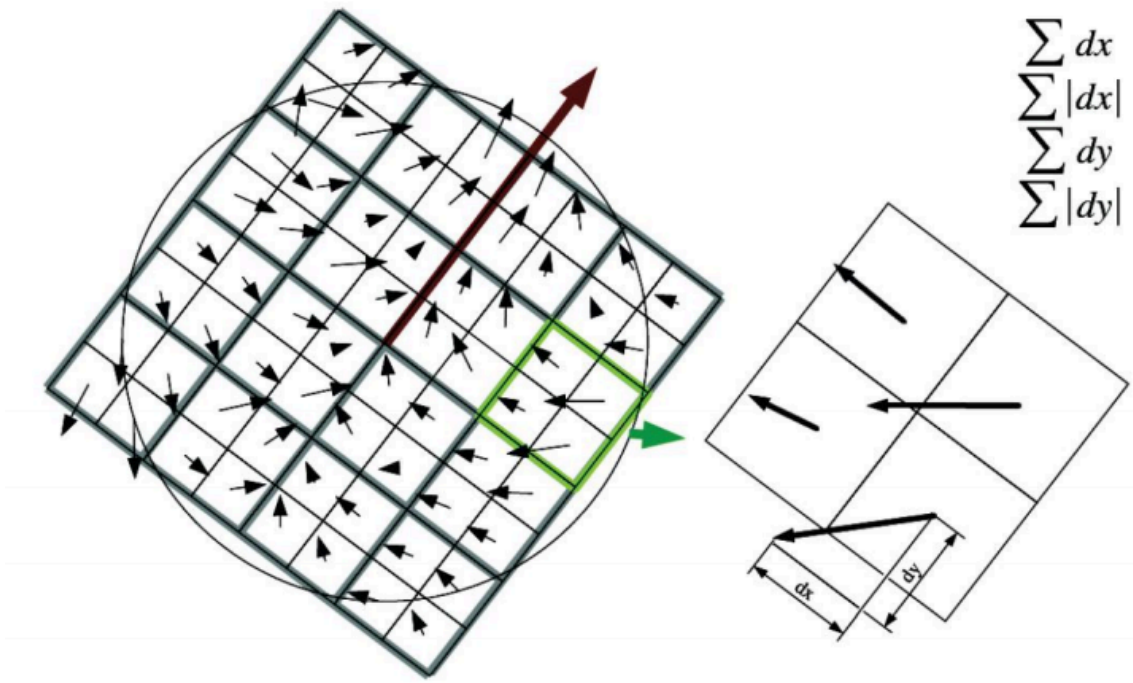

Figure 3.3 SURF descriptor [4]

Thus, we have studied the SURF technique that extracts features and descriptors from the image objects.

3.1.3 Results

The SIFT [5] and SURF [6] algorithms were implemented to extract the features and descriptors from the Bus and Sedan image. The following images show the SIFT features of the bus image and the sedan image

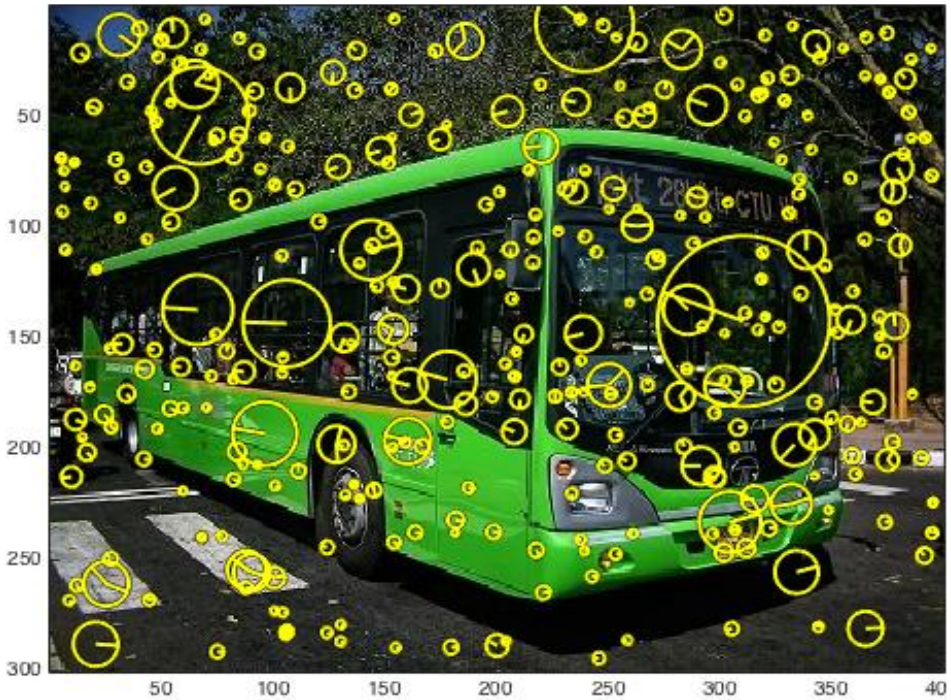

Figure 3.4 SIFT features for the Bus image

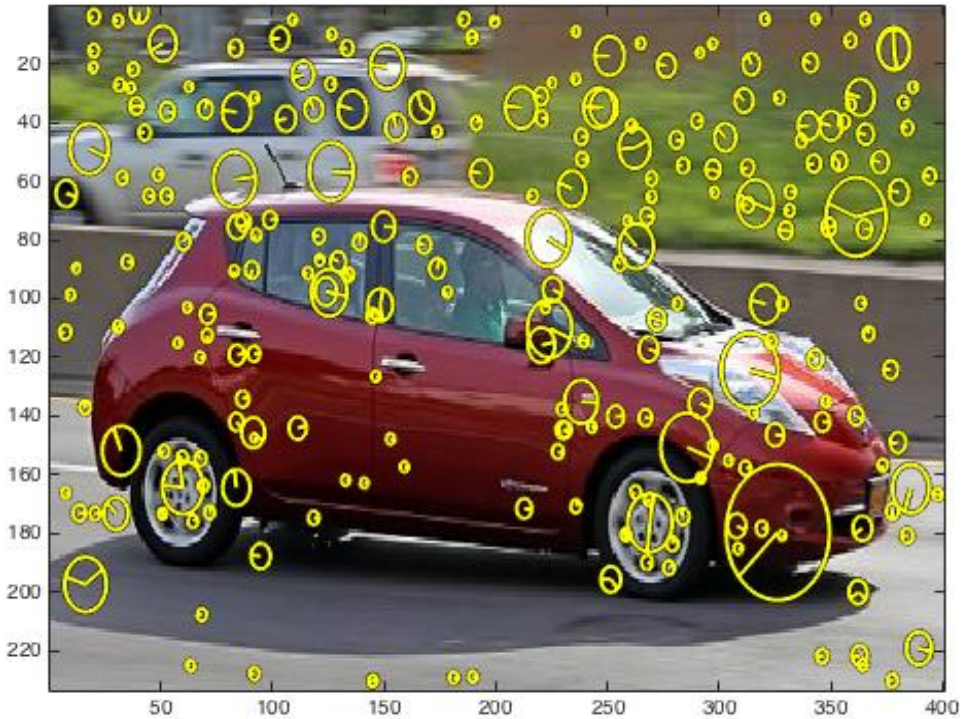

Figure 3.5 SIFT features for the sedan image

Each of the above features shown in yellow has its corresponding descriptor. The following images show 15 descriptors of bus and 10 descriptors of sedan. The descriptors will cover the image content if all of them are shown. Hence only a few descriptors of both the images have been displayed.

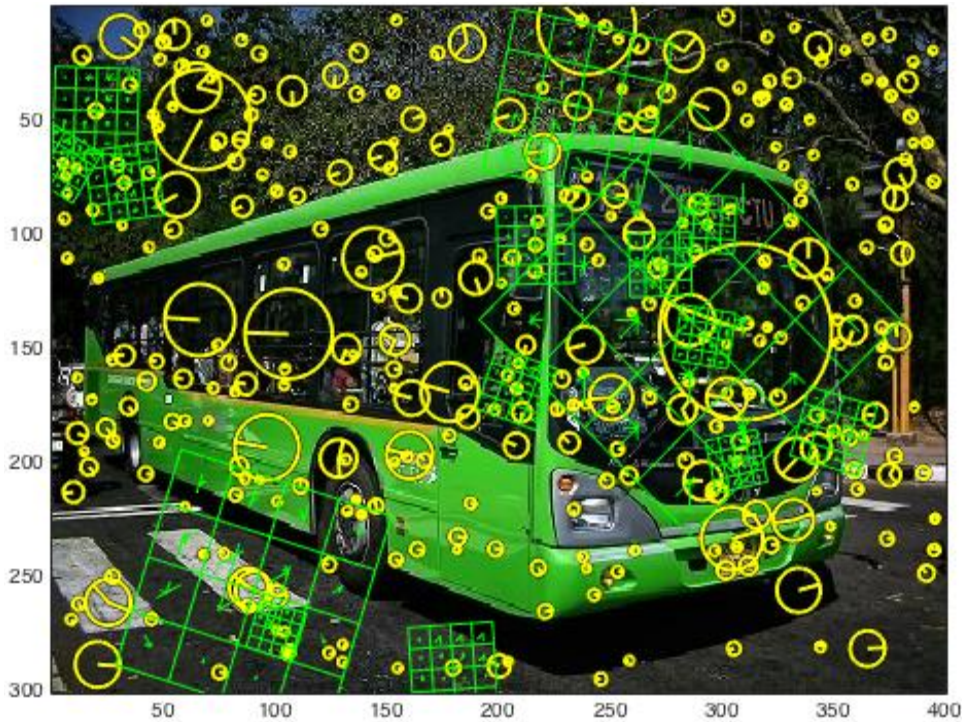

Figure 3.6 Bus image with SIFT features and 15 descriptors

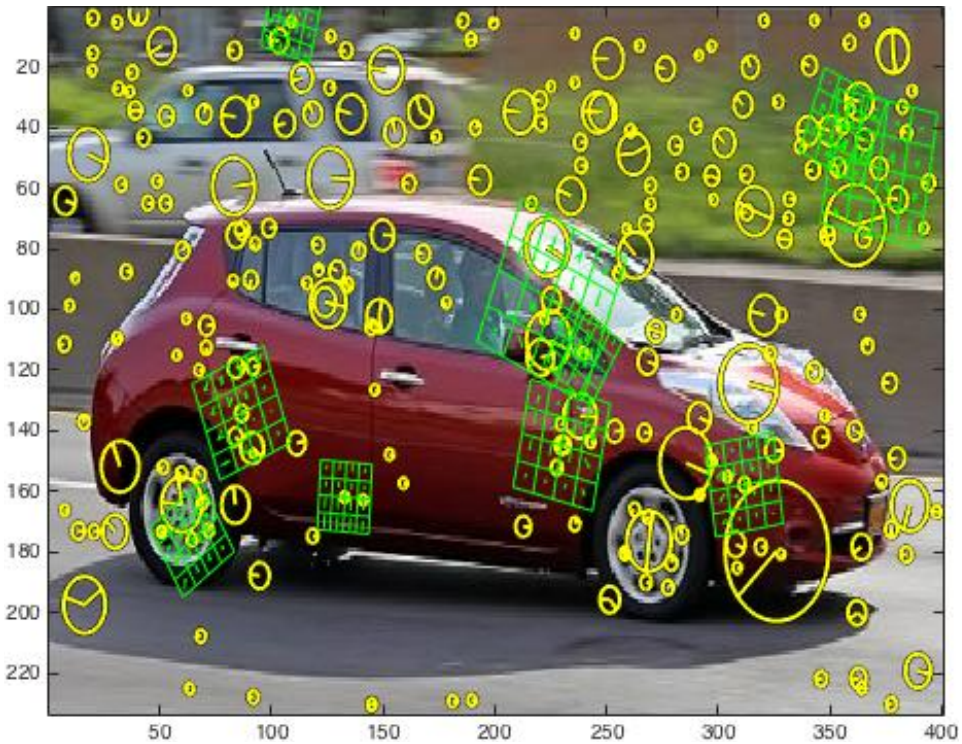

Figure 3.7 Sedan image with SIFT features and 10 descriptors

The SURF features for the bus and sedan image are as shown below:

Figure 3.8 SURF features for the bus image

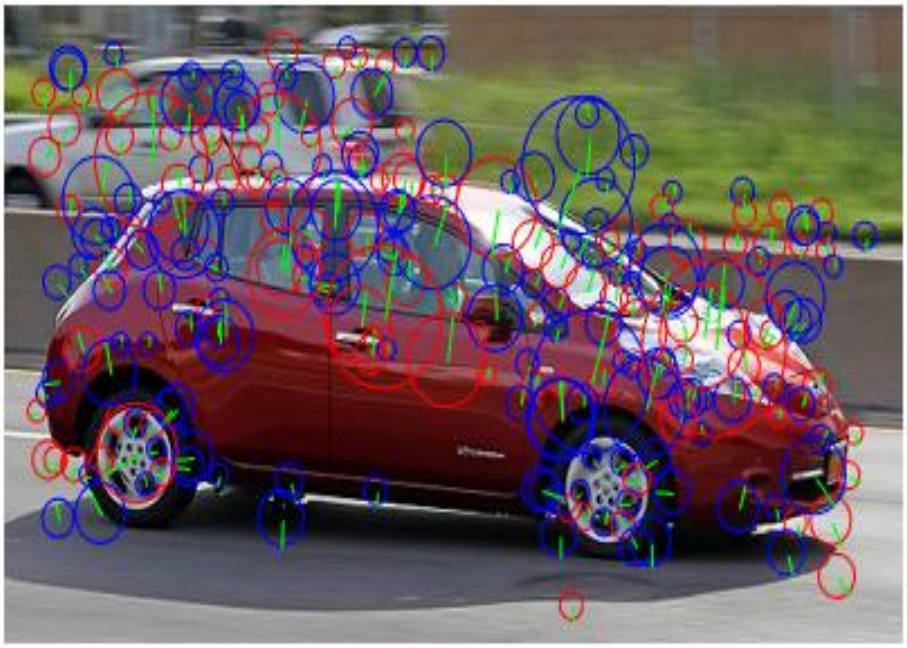

Figure 3.9 SURF features for the sedan image

3.1.4 Discussion

We can see the SIFT and SURF features for the bus and sedan images above. Both the algorithm use different techniques to extract useful and reliable image features. The differences in performance of SIFT and SURF are as follows:

1. SURF is 3 to 5 times faster than SIFT
2. SURF is more resilient to noise than SIFT
3. SURF is more easily adapted to parallel processing because each image is independently generated unlike SIFT.

SIFT uses difference of Gaussian over various scales of an image. SURF uses haar wavelet. In conclusion we can say that SIFT is the most accurate algorithm whereas SURF is the fastest algorithm for extracting features and descriptors. Both these algorithms are scale invariant.

SIFT uses difference of Gaussian and is the most accurate feature detector and descriptor

3.2. Image Matching

3.2.1 Motivation

The matching of images to calculate the measure of similarity between them is an important problem in various tasks. Object recognition, image registration and robot localization are some of these tasks. Using the SIFT algorithm, we can first extract the features of the two object images that we want to match. Now, by using different feature matching approaches, like the nearest neighbor procedure, we can compute the total number of matches between the two images. This problem performs image matching between various sets of 2 images.

3.2.2 Approach and Implementation

- To perform image matching, the SIFT features of both the images are determined independently.
- One image is considered as the model image (or training images for more than one image) whereas the other image is considered as the test image.
- To find corresponding features between the two images, different feature matching techniques can be used.
- In this problem, we will use the nearest neighbor technique.
- According to this approach, for each feature F_1^i in the model image, the corresponding feature F_2^j in the test image must be considered.
- This corresponding feature is the one with the smallest Euclidean distance from the feature F_1^i.
- This pair of the corresponding features (F_1^i, F_2^j) is then called a match.
- After getting all the SIFT features, many features from an image will not have any correct match probably because they arise from the background or they don't exist in the model/training database.
- Thus, we need to discard such features by using some thresholding technique.
- Next step is to determine if this match is positive or negative. For this purpose we need to use a threshold.
- So now, if the Euclidean distance between two features is below a certain threshold then the match is positive, otherwise it is negative.

- The threshold cannot be global because there are a lot of variations within the object from patch to patch.
- Thus, the threshold is defined as the ratio between the Euclidean distance to the nearest neighbor and the second nearest neighbor.
- A match is considered to be positive only if the distance to the nearest neighbor is 0.8 times larger than that from the second nearest one
- Using this image matching approach, we have calculated the image matching for 3 pairs of images in this problem.
- The first pair is the school bus 1 and school bus 2 pair
- The second and the thirst pair is school bus 1 with bus and school bus 1 with sedan respectively.
- This was implemented by using the VLFEAT source code [5]

3.2.3 Results

The object matching results for school bus 1 and school bus 2 are as shown below:

Figure 3.10 Image matching between school bus 1 and school bus 2

The object matching results for school bus 1 and bus are as shown below:

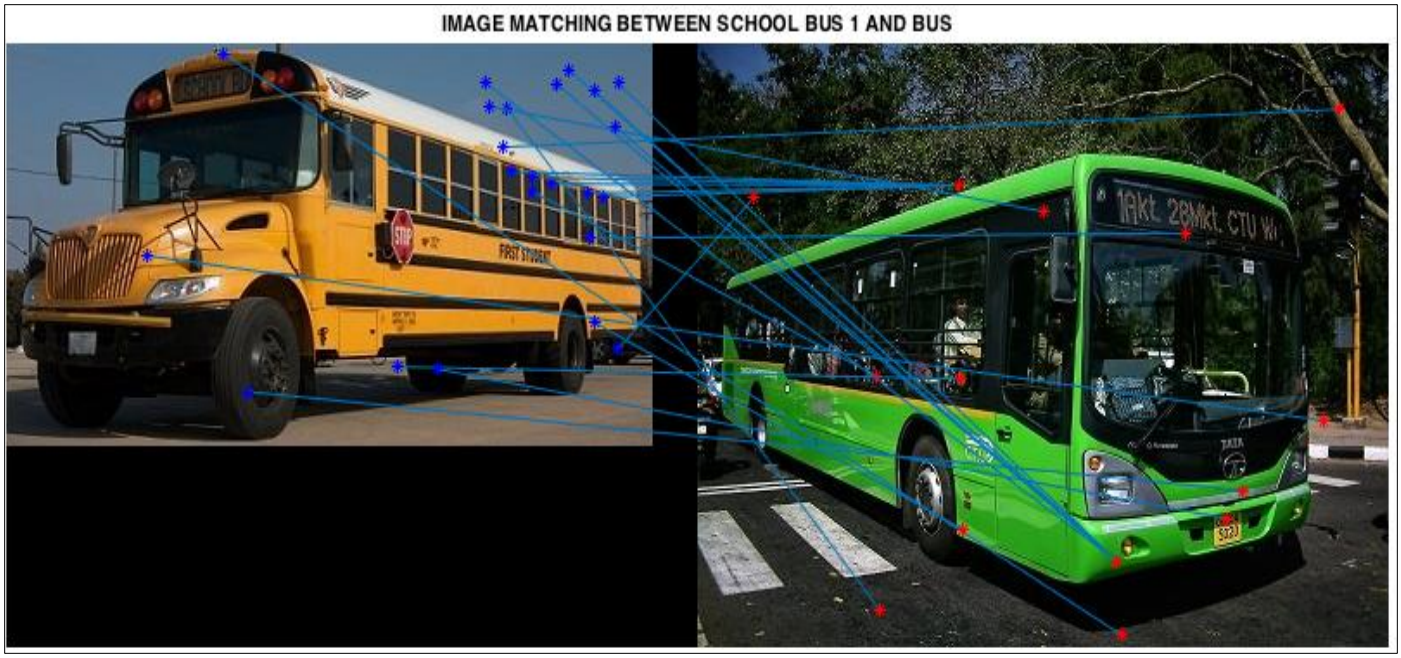

Figure 3.11 Image matching between school bus 1 and Bus

The object matching results for school bus 1 and sedan are as shown below:

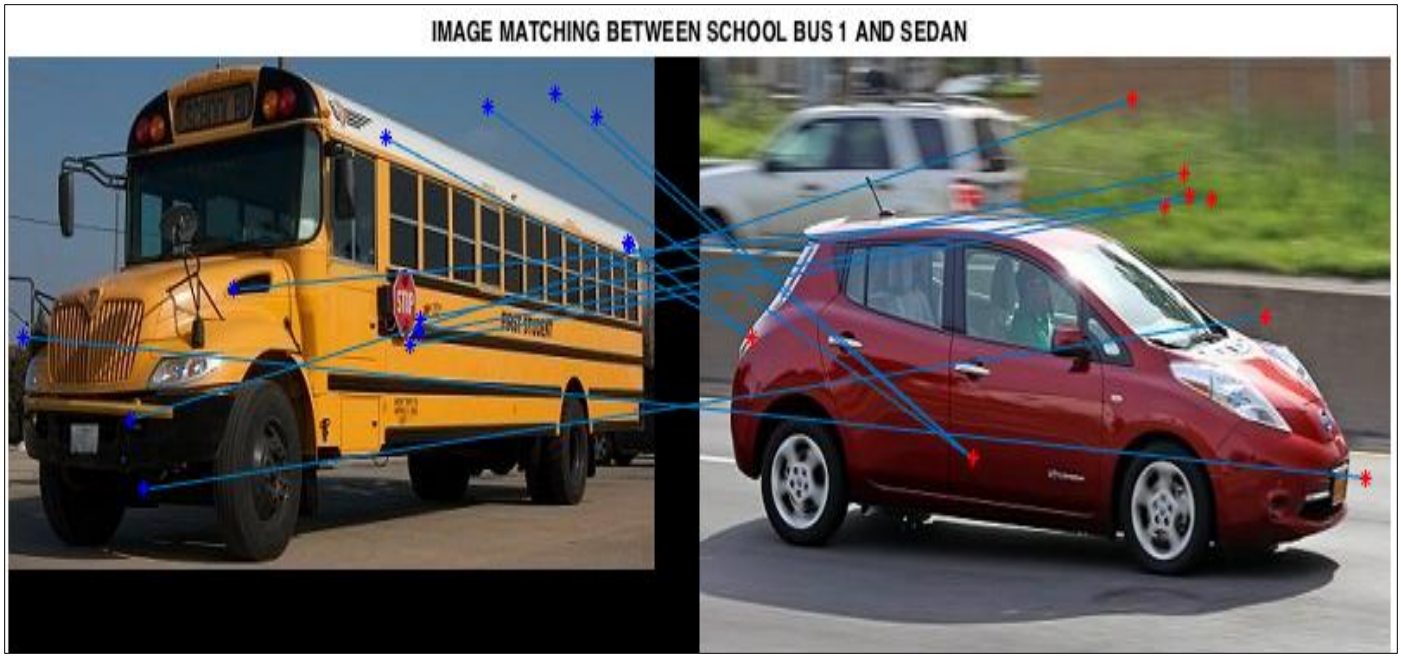

Figure 3.12 Image matching between school bus 1 and sedan

3.2.4 Discussion

The image-matching algorithm, when performed on the school bus 1 and school bus 2 images, gave 13 matches. The number of matches for school bus 1 and bus image are 23. The number of matches for school bus 1 and sedan are 13. Ideally, school bus 1 and school bus 2 should have given the maximum number of matches but it is not the case in this problem. The matching did not work well on the two images of the same object. We did not get a good number for matches between school bus 1 and school bus 2 because the features were not really correlated and they failed to produce positive matches that could overcome the threshold. On the other hand, we got good results for matching between school bus 1 and bus image because the features turned out to be quite similar. Similarly, we got a moderate umber of matches between school bus 1 and sedan image.

3.3. Bag Of Words

3.3.1 Motivation

The SIFT features extracted from various images can be used for object classification. After applying k-means clustering to all the SIFT features, that have been obtained from a set of training images, we can get k corresponding centroids for all these clusters. These centroids are used to get the codewords for the training as well as testing images. Thus, once we perform clustering on the training images, we can use the centroids for image classification. In this problem, we will explore the idea of bag of words for image classification.

3.3.2 Approach and Implementation

- In section 3.1, we saw how to get the SIFT features for object recognition and matching.
- We want to represent an object as a bag of words that are obtained from the object descriptors. This idea is shown below:

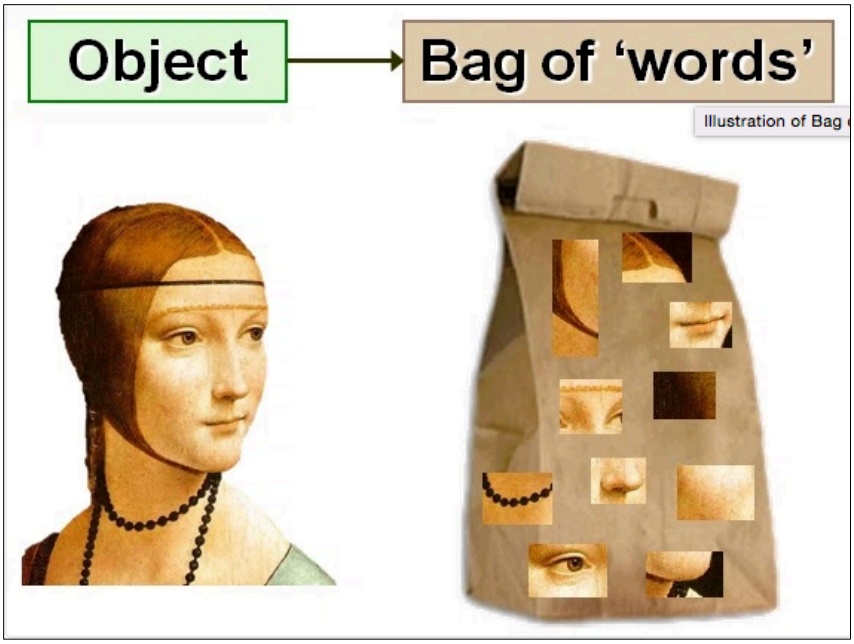

Figure 3.13 Concept of representing an image as bag of words

- This bag of words model can then be used for image classification.

- First, we need to construct a vocabulary of many visual words and then by using this vocabulary we need to generate histograms for different images from the training as well as the test set.
- So now, we take a set of training images and extract SIFT descriptors of these images by using the algorithm given in section 3.1
- Thus, we will get different number of 128 dimensional descriptors for every image.
- We need to concatenate all the image descriptors, obtained from the training image set, to create one large set of 128 dimensional descriptors.
- Now, we can perform k-means clustering on this set of descriptors.
- Clustering will result into k number of clusters of these descriptors.
- Every cluster will have its own centroid and thus we get a total of k centroids for k clusters.
- These centroids are nothing but cluster centers or cluster means.
- Centroids represent our dictionary's visual words and that is what we want to obtain in our first step.
- Once we have the centroids, we can now start generating histograms of different images of the training as well as test set.
- Now, suppose we want to get the histogram of an image A.
- We will first extract its SIFT features and descriptors.
- Now, we will assign each descriptor to one of the k clusters using the nearest neighbor algorithm.
- In nearest neighbor, each descriptor's distance from all the k centroids will be calculated and then that descriptor will be assigned to that particular cluster that gave the shortest distance value i.e. the cluster that is its nearest neighbor.
- In this manner, each descriptor of this image A will get assigned to one of the k clusters depending upon its distance from the k cluster centroids.
- Thus, we now have the cluster labels for each of these descriptors.
- We can generate a histogram of this image by counting the number of descriptors that were assigned to each of the k clusters.

- This idea is depicted below:

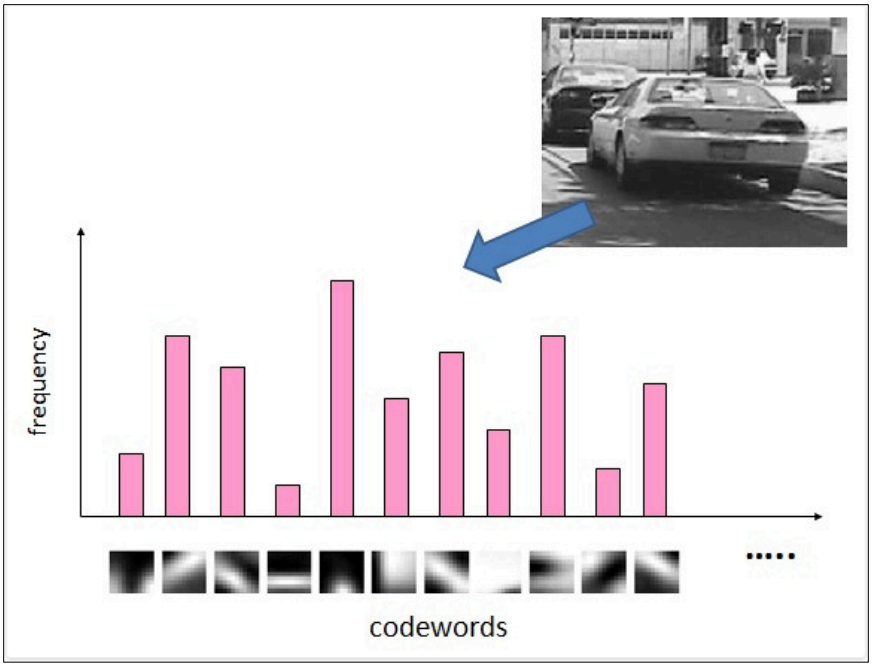

Figure 3.14 Codeword of an image as a histogram

- Thus, given a new image we can represent it by using the bag of words model.
- This approach has been used in this problem and the steps of implementation are given below:
- To start with, we have a training set of 3 images, namely school bus 1, bus and sedan.
- We first need to extract the SIFT features from each of these images.
- Now, we have different number of 128 dimensional descriptors for each of these images.
- We need to concatenate all these descriptors together and then perform k-means clustering on this entire set of 128 dimensional descriptors.
- In this problem, we have chosen k = 8. Thus after performing clustering, we have created 8 clusters of these 128 dimensional sets of descriptors.
- Each cluster has its centroid which can be thought of as the cluster mean.
- These centroids act like codewords and can be used to generate histograms of training as well as testing images.

- These centroids are nothing but our dictionary's visual words of our images.
- Now, after getting these centroids we want to generate histograms of 4 images that have 3 training set images and 1 test image that is the school bus 2 image.
- We already have the descriptors of the training images and we need to extract the descriptors of the test image.
- Once we get the independent sets of descriptors of all these images, we will consider one set at a time and then get the histogram for that corresponding image.
- Thus, every image descriptor will get assigned to one of the 8 clusters based upon the nearest neighbor output.
- Thus, for each image, its descriptors are labeled from 1 to 8 depending upon the cluster to which they are assigned.
- Now, we can count the number of descriptors for each cluster and then get the histogram of that image.
- This needs to be repeated for all the 4 images separately.
- This implementation has been done by using the VLFEAT source code [5].

3.3.3 Results

The codewords for school bus 1, sedan and bus images which is the training set are as follows:

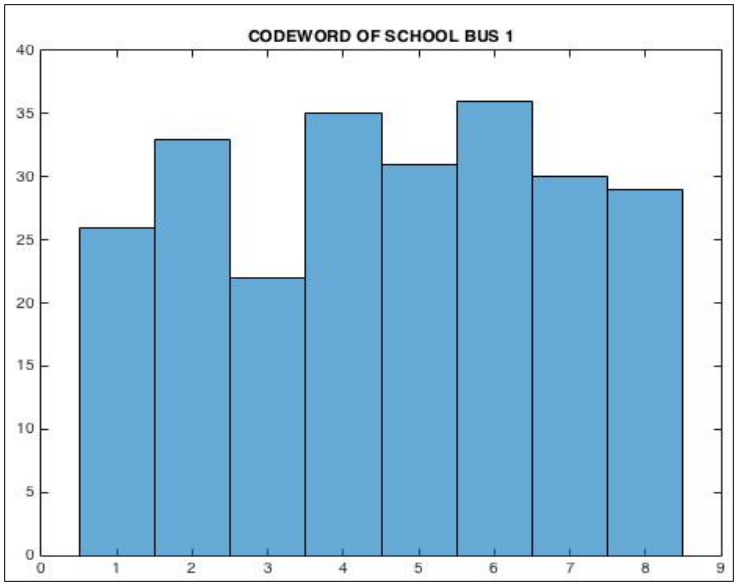

Figure 3.15 (a) Histogram of school bus 1

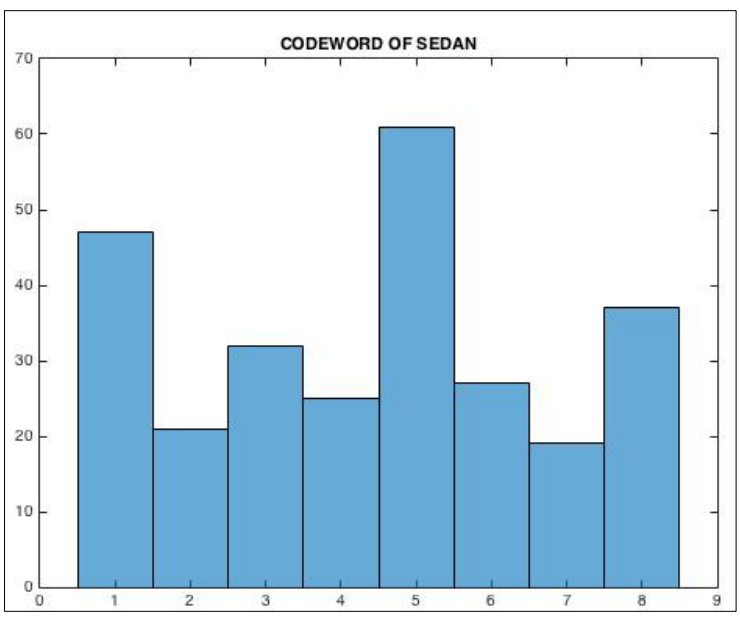

Figure 3.15(b) Histogram of Sedan

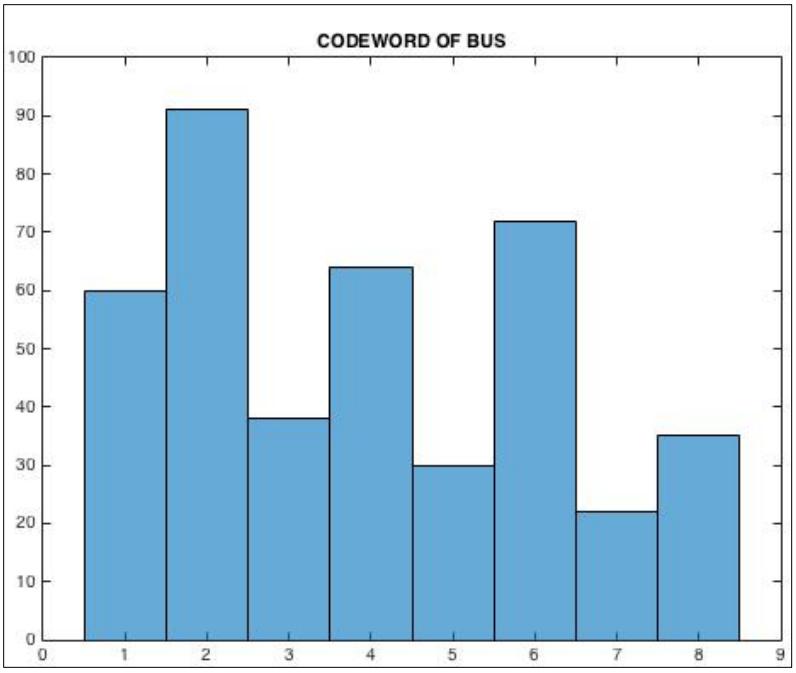

Figure 3.15(c) Histogram of Bus

The codeword for school bus 2 which is the test image is as follows:

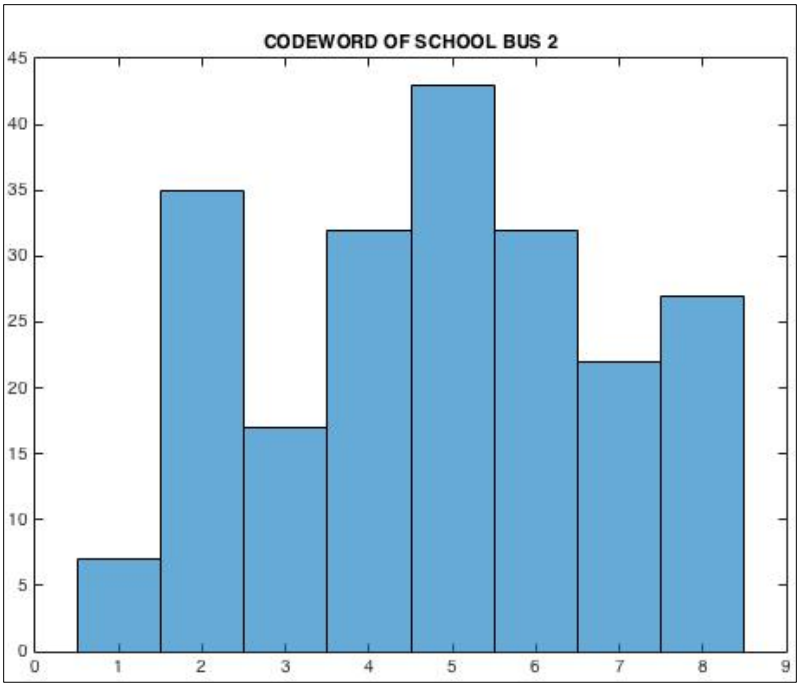

Figure 3.15(d) Histogram of school bus 2

3.3.4 Discussion

The codeword representation of an image in the form of histogram can be used for classification of images. First we have trained the classifier by using the school bus 1, bus and the sedan image. The testing of this bag of words model has to be performed by using the school bus 2 image. Now, after obtaining the codewords for all 4 images, as shown in the above results, we can compare the codeword of the school bus 2 image with other 3 codewords so as to classify this image. Maximum number of descriptors of school bus 2 has been assigned to clusters labeled from 4 to 8. Thus, the bars from 4 to 8 are quite high in a uniform fashion. For sedan and bus, the bars from 4 to 8 are not uniformly high as we have some low level bars in between. If we now look at the histogram of school bus 1, we can say that it also has maximum descriptors with labels from 4 to 8 and thus the bars are uniformly high in this same range. Thus, it shows a similarity with the histogram of school bus 2 image. Thus, in conclusion, the codeword of school bus 2 is most related to the codeword of school bus 1.

4. References

[1]http://www.mathworks.com/matlabcentral/fileexchange/28109-snakes--active-contour-models
[2] Distinctive Image Features from Scale Invariant Keypoints by David G. Lowe
[3]http://www.scholarpedia.org/article/Scale_Invariant_Feature_Transform
[4] http://www.sci.utah.edu/~fletcher/CS7960/slides/Scott.pdf
[5] http://www.vlfeat.org/overview/sift.html
[6] http://www.mathworks.com/matlabcentral/fileexchange/28300-opensurf--including-image-warp-