Hengyue Liu
4107-2966-75
hengyuel@usc.edu
EE 569 Homework #4
November 25, 2015

Problem 1: Optical Character Recognition (OCR)

a) Decision Tree for Reference Data

I Abstract and Motivation

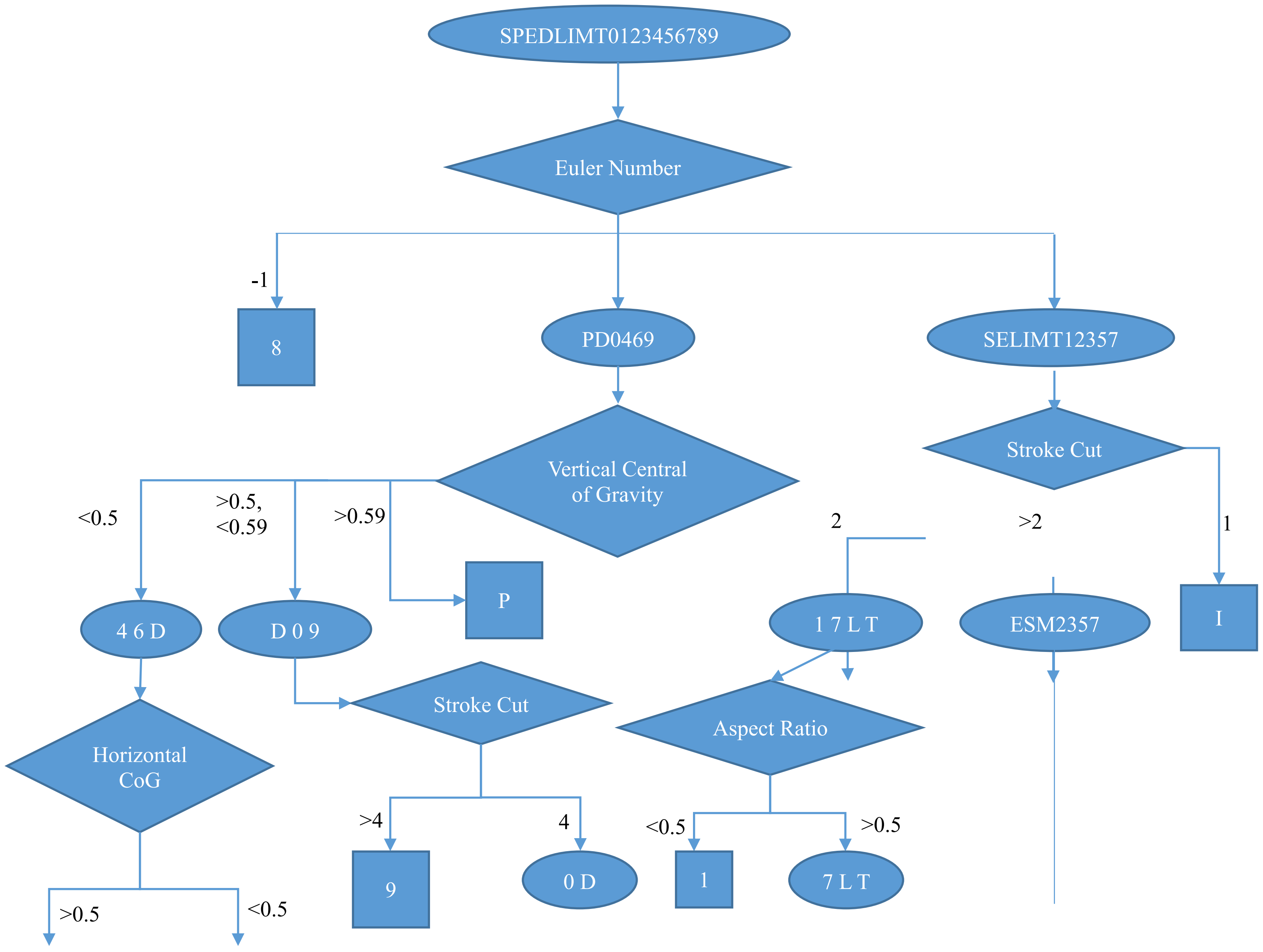
Optical Character Recognition is a useful application of pattern recognition and computer vision. It is a realization of machine "reading" images. OCR systems can be strong and robust dealing with different fonts, size, dynamic range and distortion. However, a very simple and limited implementation is developed in this section. Here, an OCR system is designed for recognizing numbers and alphabets from given speed signs. Some pre-processing steps are necessary for segmentation, like changing color to gray-scale, brightness enhancement, histogram equalization and image binarization. To recognize the numeric and alphabetic symbols, images need to be segmented well and each symbol is contained in a particular Region of Interest (ROI). In this implementation, connected component labeling method (discussed in discussion for homework 3) is used for object segmentation. Next, for each segment, compute the geometric features and build a decision tree to distinguish different characters. Some features are discriminant and robust, while some are unstable and useless for recognition. So, choosing the proper features is important in the whole system. Another important factor is the organization of the decision tree. Different construction order gives different efficiency and complexity of the tree. A well-organized decision tree can be built by maximizing decision entropy with several trials. However, efficiency and complexity are not significant for this small OCR system, thus the decision tree is built manually. In next part, a detailed procedure is introduced.

II Approach and Procedures

1. Read the training raw image, and change the color space to gray-scale.
2. Since the train image is ideal, the gray-scale image can be binarized directly. Alternatively, some pre-processing steps can also be applied but with little influence on it.

3. Conducting the connected component labeling to the binarized image. It gives 18 ideal segments in total, each standing for a particular character.
4. For each segmented character, compute the geometric features. Selected features are: Euler Number; Stroke Cut; Normalized Area; Normalized Perimeter; 1st order Central Spatial Moment; Circularity; Aspect Ratio; Symmetry.
5. Build decision tree based on the features calculated in last step. A prototype decision tree is built in this step. With several trials, the decision thresholds will be refined later.

The decision tree is designed as:



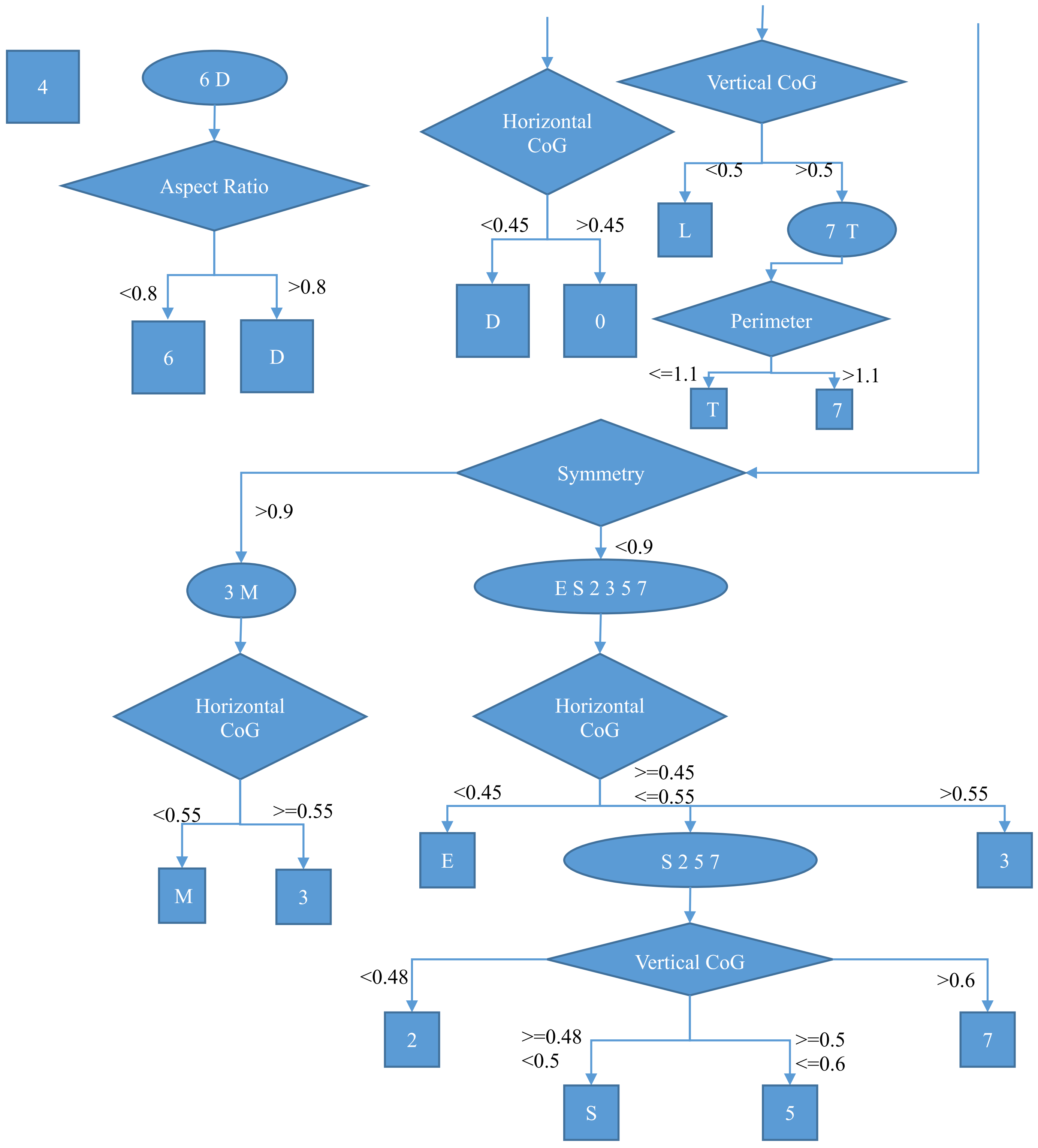

4
6 D
Aspect Ratio
<0.8
>0.8
6
D
Horizontal CoG
<0.45
>0.45
D
0
Vertical CoG
<0.5
>0.5
L
7 T
Perimeter
<=1.1
>1.1
T
7
Symmetry
>0.9
<0.9
3 M
E S 2 3 5 7
Horizontal CoG
<0.55
>=0.55
M
3
Horizontal CoG
<0.45
>=0.45
<=0.55
>0.55
E
S 2 5 7
3
Vertical CoG
<0.48
>=0.48
<0.5
>=0.5
<=0.6
>0.6
2
S
5
7

Figure 1 Decision tree

III Experimental Results

Shown below.

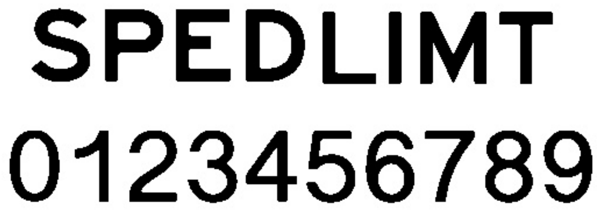


Figure 2 Binary train image after pre-processing

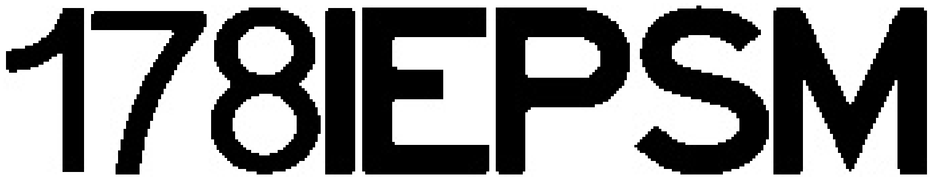
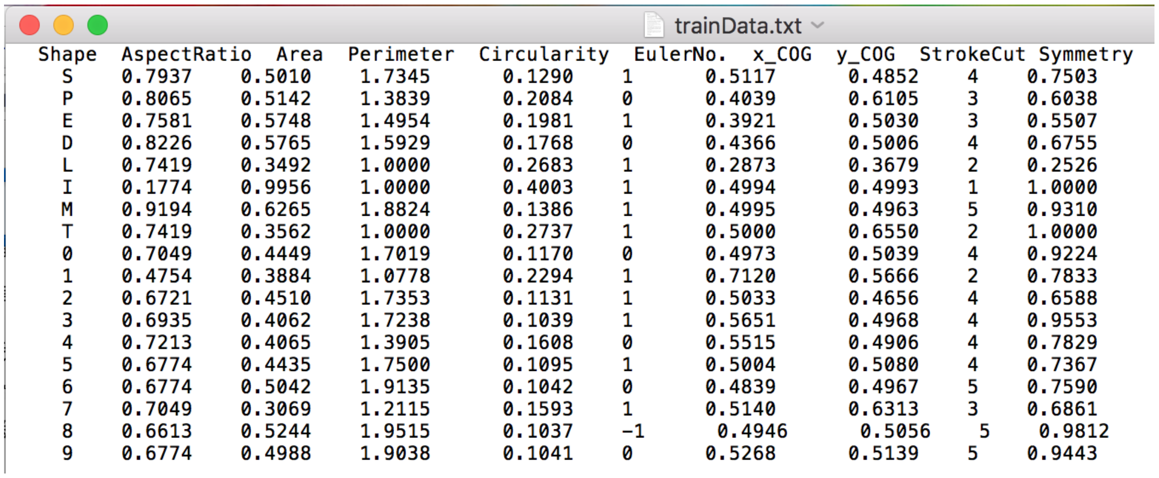


Figure 3 some segments of train image



Shape	AspectRatio	Area	Perimeter	Circularity	EulerNo.	x_COG	y_COG	StrokeCut	Symmetry
S	0.7937	0.5010	1.7345	0.1290	1	0.5117	0.4852	4	0.7503
P	0.8065	0.5142	1.3839	0.2084	0	0.4039	0.6105	3	0.6038
E	0.7581	0.5748	1.4954	0.1981	1	0.3921	0.5030	3	0.5507
D	0.8226	0.5765	1.5929	0.1768	0	0.4366	0.5006	4	0.6755
L	0.7419	0.3492	1.0000	0.2683	1	0.2873	0.3679	2	0.2526
I	0.1774	0.9956	1.0000	0.4003	1	0.4994	0.4993	1	1.0000
M	0.9194	0.6265	1.8824	0.1386	1	0.4995	0.4963	5	0.9310
T	0.7419	0.3562	1.0000	0.2737	1	0.5000	0.6550	2	1.0000
0	0.7049	0.4449	1.7019	0.1170	0	0.4973	0.5039	4	0.9224
1	0.4754	0.3884	1.0778	0.2294	1	0.7120	0.5666	2	0.7833
2	0.6721	0.4510	1.7353	0.1131	1	0.5033	0.4656	4	0.6588
3	0.6935	0.4062	1.7238	0.1039	1	0.5651	0.4968	4	0.9553
4	0.7213	0.4065	1.3905	0.1608	0	0.5515	0.4906	4	0.7829
5	0.6774	0.4435	1.7500	0.1095	1	0.5004	0.5080	4	0.7367
6	0.6774	0.5042	1.9135	0.1042	0	0.4839	0.4967	5	0.7590
7	0.7049	0.3069	1.2115	0.1593	1	0.5140	0.6313	3	0.6861
8	0.6613	0.5244	1.9515	0.1037	-1	0.4946	0.5056	5	0.9812
9	0.6774	0.4988	1.9038	0.1041	0	0.5268	0.5139	5	0.9443

Figure 4 geometric features of 'Training'

IV Discussion

Before talking about features, the segmentation method should be discussed. Connected component labeling is applied in this implementation since the method is easy to understand with good segmentation results. For ideal training image, the results of segments are actual characters. However, the segmentation results of testing images will contain irrelevant segments, which will be discussed later. With extracted characters, the features can be computed.

There are 7 geometric features used in this decision tree: Aspect Ratio; Normalized Perimeter; Euler Number; Horizontal Central of Gravity; Vertical Central of Gravity; Stroke Cut Number; Symmetry. The reason why selecting these features is that they are relatively stable than other features like Area and Circularity. For example, the area of a shape can be largely influence by the thickness of the shape, which may be improved by applying thinning first then calculate the area (computationally exhausting). With several comparisons between testing images and training image, the normalized Area and Circularity showed low liability and discriminability, so discard these two features although being computed.

Euler Number is the most stable feature compared to others. So, take Euler Number as the first decision node. There are 3 different results for the training image: -1, 0 and 1. If the Euler Number is -1, then there is an end node with the character recognized as “8”. Meanwhile, other characters will be divided into 2 groups.

Stroke Cut is another great feature for recognition. The stroke cut is computed like this: first do the thinning for the image; draw a horizontal line and a vertical line on the thinned image, and count the number of intersections; move the 2 cutting lines and scan the whole thinned image again, until all possible combinations of cutting lines are visited; find the majority of the count of stroke cut and take it as the final stroke cut number. In this way, the stroke cut becomes more stable even if the shape is slightly different for the same character.

Horizontal and vertical Central of Gravity(CoG) are also important features. The centroid is computed using spatial central moment of Area, and the position of centroid is used for recognizing different characters. Take the example of the character “E”, the horizontal CoG must be less than 0.5 (actually less than 0.45) since more pixels are on the left side of the character. Hence, these two features are significant in the decision tree.

Aspect Ratio is a simple but efficient geometric feature. For example, in the decision tree, “1”, “7”, “L”, “T” have the same stroke cut 2. “1” can be separated out since its aspect ratio is smaller 0.5 for sure since it is thinner than the other three. In some decision nodes, it can recognize end leaf node quickly.

Symmetry (right-left symmetry) is similar to aspect ratio in some sense, but it is not robust. Because in many situations, the symmetry axis does not always lie on the center line of the image. So, in the implementation, multiple choices of symmetry axis are tried, then find the largest value of symmetry. For example, “I” and “T” have symmetry value equals to 1.0, while “L” has small symmetry value.

With these selected features, one can build a primitive decision tree. The tree must be modified later since the model should satisfy at least all other test images. This primitive tree should output the right recognitions with ‘Training.raw’ as test image.

b) OCR Testing for both simple and advanced cases

I Abstract and Motivation

The most important part for testing cases is proper pre-processing for non-ideal images, which involved several extra steps different from dealing with the ideal training image. Pre-processing steps are necessary for segmentation, like changing color to gray-scale, brightness enhancement, histogram equalization and image binarization. From the description of the assignment, there are two kinds of situation: simple and advanced cases. However, in my implementation, all images are treated the same, even for training image. So, part b) & c) are combined via the same procedures, and the program will decide which particular pre-processing step is needed. For example, intensity adjustment will be applied if the image is too bright or dark, or the histogram is not smooth enough, transfer-function-based histogram equalization will be applied.

II Approach and Procedures

1. Read arbitrary test image, and do the pre-processing step by step:
   a. Changing color space to gray-scale.
   b. Apply Gaussian filter. Mean filter reduce the influence of small parts, which will generate redundant and useless small segments in object segmentation step.
   c. Compute the average intensity (brightness) of the test image, do the brightness adjustment: if the image is too dark, increase the intensity values; if the image is too bright, decrease the intensity values. (naïve manipulation of each pixel value)
   d. Histogram equalization via transfer function method. Histogram equalization gives smooth transition of image color, which is important for binarizing image correctly. If the input image's histogram has too many sharp transitions, then do the equalization.
   e. Binarization. 2 methods are implemented, one is dual maximum method, the other is naïve binary thresholding. However, given the simple situation, setting threshold equals to 128 is easy and enough for achieving good results.
   f. Noise removal. There may be some isolate dots or holes in the binary image, and removing the noise gives more accurate and robust geometric features.
2. Similar procedure with step 3&4 in part a), do the segmentation and features computation for test image.
3. Based on the features of each segment of test image, make prediction from the decision tree built in part a).
4. If applicable, refine the decision tree by changing the thresholds or organization.

III Experimental Results

Shown below.

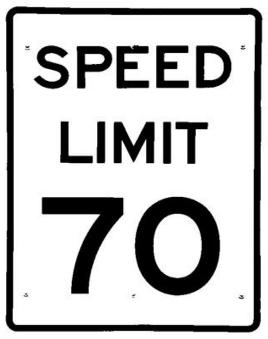


Figure 5 Binary 'Test_ideal1' after pre-processing

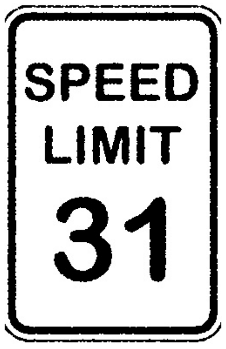


Figure 6 Binary 'Test_ideal2' after pre-processing

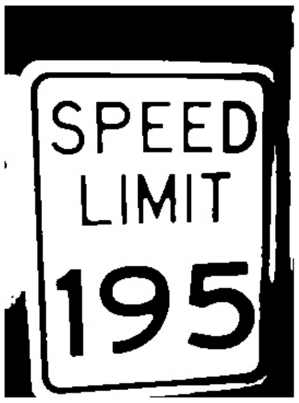


Figure 7 Binary 'Test_night' after pre-processing

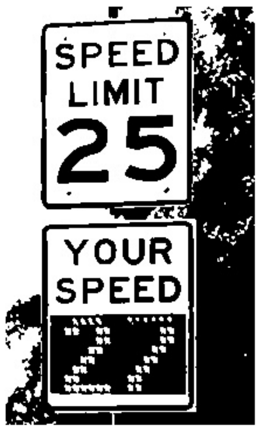


Figure 8 Binary 'Test_shade' after pre-processing

Following are some examples of connected component segmentation.

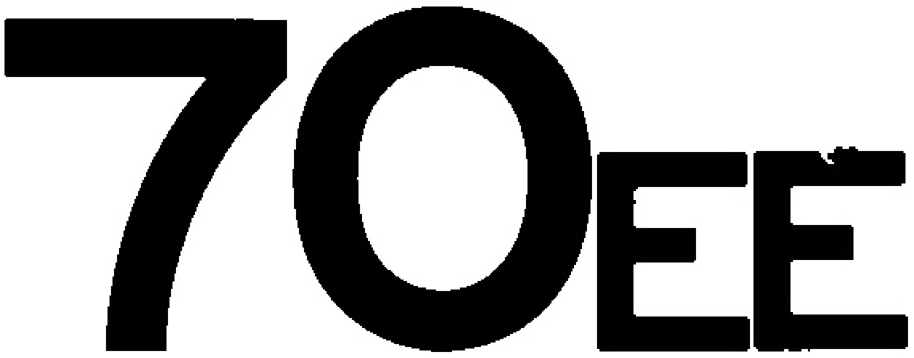


Figure 9 some segments of 'Test_ideal1' (scaled)

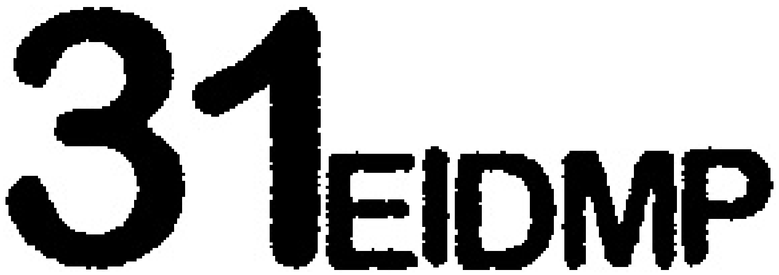


Figure 10 some segments of 'Test_ideal2'

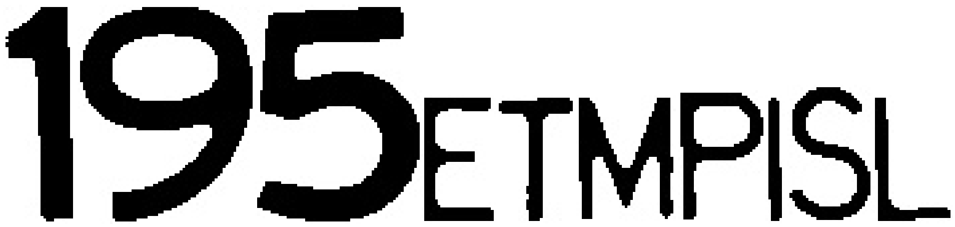


Figure 11 some segments of 'Test_night'

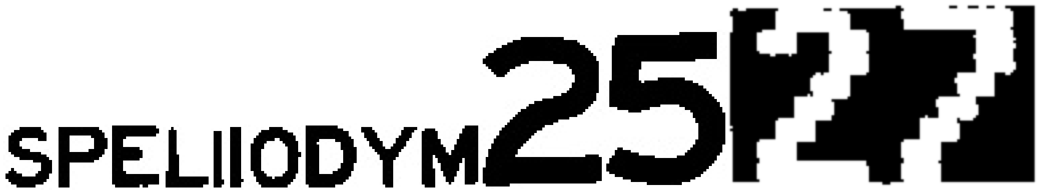
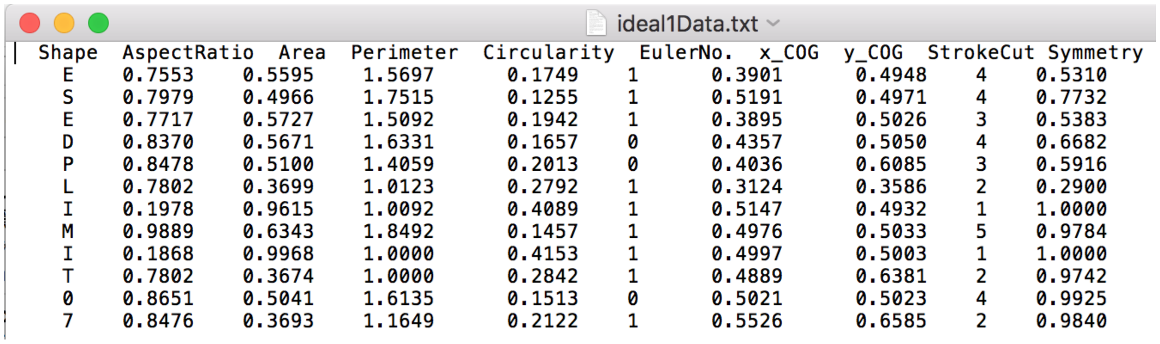


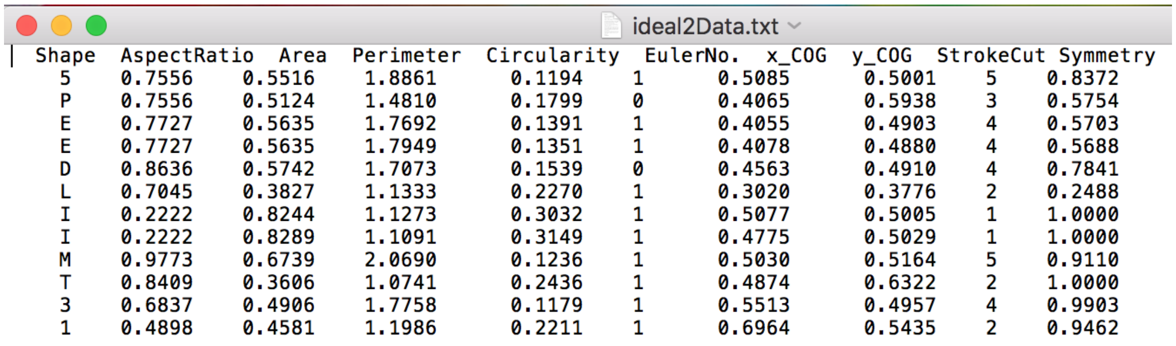
Figure 12 some segments of 'Test_shade'

Following are the computed geometric features.


ideal1Data.txt

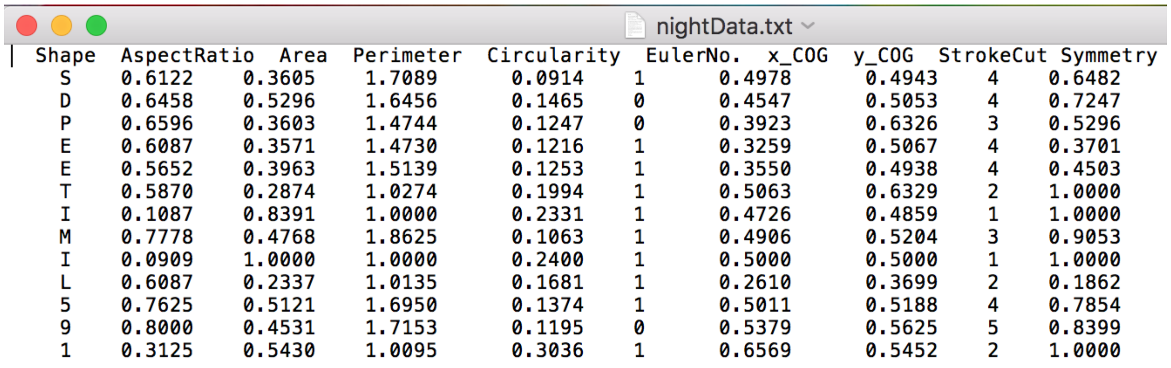
```
 Shape  AspectRatio  Area   Perimeter   Circularity  EulerNo.  x_COG    y_COG  StrokeCut Symmetry
   E    0.7553    0.5595    1.5697     0.1749     1       0.3901     0.4948    4     0.5310
   S    0.7979    0.4966    1.7515     0.1255     1       0.5191     0.4971    4     0.7732
   E    0.7717    0.5727    1.5092     0.1942     1       0.3895     0.5026    3     0.5383
   D    0.8370    0.5671    1.6331     0.1657     0       0.4357     0.5050    4     0.6682
   P    0.8478    0.5100    1.4059     0.2013     0       0.4036     0.6085    3     0.5916
   L    0.7802    0.3699    1.0123     0.2792     1       0.3124     0.3586    2     0.2900
   I    0.1978    0.9615    1.0092     0.4089     1       0.5147     0.4932    1     1.0000
   M    0.9889    0.6343    1.8492     0.1457     1       0.4976     0.5033    5     0.9784
   I    0.1868    0.9968    1.0000     0.4153     1       0.4997     0.5003    1     1.0000
   T    0.7802    0.3674    1.0000     0.2842     1       0.4889     0.6381    2     0.9742
   0    0.8651    0.5041    1.6135     0.1513     0       0.5021     0.5023    4     0.9925
   7    0.8476    0.3693    1.1649     0.2122     1       0.5526     0.6585    2     0.9840
```

Figure 13 geometric features of 'Test_ideal1'


ideal2Data.txt

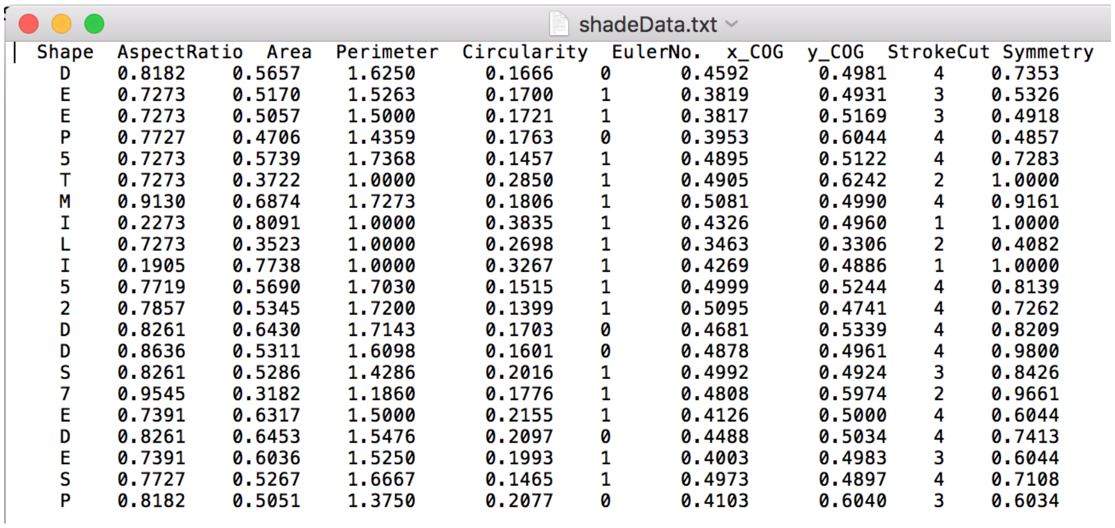
```
 Shape  AspectRatio  Area   Perimeter   Circularity  EulerNo.  x_COG    y_COG  StrokeCut Symmetry
   5    0.7556    0.5516    1.8861     0.1194     1       0.5085     0.5001    5     0.8372
   P    0.7556    0.5124    1.4810     0.1799     0       0.4065     0.5938    3     0.5754
   E    0.7727    0.5635    1.7692     0.1391     1       0.4055     0.4903    4     0.5703
   E    0.7727    0.5635    1.7949     0.1351     1       0.4078     0.4880    4     0.5688
   D    0.8636    0.5742    1.7073     0.1539     0       0.4563     0.4910    4     0.7841
   L    0.7045    0.3827    1.1333     0.2270     1       0.3020     0.3776    2     0.2488
   I    0.2222    0.8244    1.1273     0.3032     1       0.5077     0.5005    1     1.0000
   I    0.2222    0.8289    1.1091     0.3149     1       0.4775     0.5029    1     1.0000
   M    0.9773    0.6739    2.0690     0.1236     1       0.5030     0.5164    5     0.9110
   T    0.8409    0.3606    1.0741     0.2436     1       0.4874     0.6322    2     1.0000
   3    0.6837    0.4906    1.7758     0.1179     1       0.5513     0.4957    4     0.9903
   1    0.4898    0.4581    1.1986     0.2211     1       0.6964     0.5435    2     0.9462
```

Figure 14 geometric features of 'Test_ideal2'



```
| Shape  AspectRatio  Area   Perimeter   Circularity  EulerNo.  x_COG   y_COG  StrokeCut Symmetry
   S    0.6122     0.3605    1.7089     0.0914    1        0.4978     0.4943    4     0.6482
   D    0.6458     0.5296    1.6456     0.1465    0        0.4547     0.5053    4     0.7247
   P    0.6596     0.3603    1.4744     0.1247    0        0.3923     0.6326    3     0.5296
   E    0.6087     0.3571    1.4730     0.1216    1        0.3259     0.5067    4     0.3701
   E    0.5652     0.3963    1.5139     0.1253    1        0.3550     0.4938    4     0.4503
   T    0.5870     0.2874    1.0274     0.1994    1        0.5063     0.6329    2     1.0000
   I    0.1087     0.8391    1.0000     0.2331    1        0.4726     0.4859    1     1.0000
   M    0.7778     0.4768    1.8625     0.1063    1        0.4906     0.5204    3     0.9053
   I    0.0909     1.0000    1.0000     0.2400    1        0.5000     0.5000    1     1.0000
   L    0.6087     0.2337    1.0135     0.1681    1        0.2610     0.3699    2     0.1862
   5    0.7625     0.5121    1.6950     0.1374    1        0.5011     0.5188    4     0.7854
   9    0.8000     0.4531    1.7153     0.1195    0        0.5379     0.5625    5     0.8399
   1    0.3125     0.5430    1.0095     0.3036    1        0.6569     0.5452    2     1.0000
```

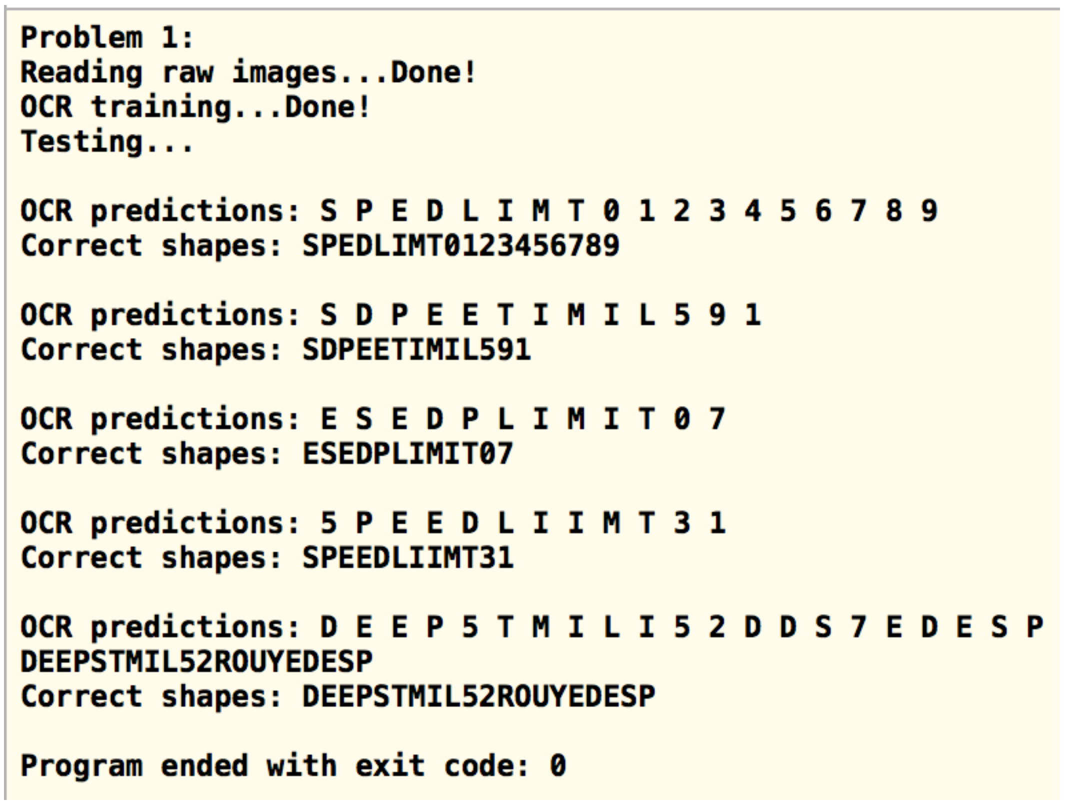
Figure 15 geometric features of ‘Test_night’



```
| Shape  AspectRatio  Area  Perimeter  Circularity  EulerNo.  x_COG   y_COG  StrokeCut Symmetry
   D    0.8182    0.5657   1.6250     0.1666    0       0.4592    0.4981    4     0.7353
   E    0.7273    0.5170   1.5263     0.1700    1       0.3819    0.4931    3     0.5326
   E    0.7273    0.5057   1.5000     0.1721    1       0.3817    0.5169    3     0.4918
   P    0.7727    0.4706   1.4359     0.1763    0       0.3953    0.6044    4     0.4857
   5    0.7273    0.5739   1.7368     0.1457    1       0.4895    0.5122    4     0.7283
   T    0.7273    0.3722   1.0000     0.2850    1       0.4905    0.6242    2     1.0000
   M    0.9130    0.6874   1.7273     0.1806    1       0.5081    0.4990    4     0.9161
   I    0.2273    0.8091   1.0000     0.3835    1       0.4326    0.4960    1     1.0000
   L    0.7273    0.3523   1.0000     0.2698    1       0.3463    0.3306    2     0.4082
   I    0.1905    0.7738   1.0000     0.3267    1       0.4269    0.4886    1     1.0000
   5    0.7719    0.5690   1.7030     0.1515    1       0.4999    0.5244    4     0.8139
   2    0.7857    0.5345   1.7200     0.1399    1       0.5095    0.4741    4     0.7262
   D    0.8261    0.6430   1.7143     0.1703    0       0.4681    0.5339    4     0.8209
   D    0.8636    0.5311   1.6098     0.1601    0       0.4878    0.4961    4     0.9800
   S    0.8261    0.5286   1.4286     0.2016    1       0.4992    0.4924    3     0.8426
   7    0.9545    0.3182   1.1860     0.1776    1       0.4808    0.5974    2     0.9661
   E    0.7391    0.6317   1.5000     0.2155    1       0.4126    0.5000    4     0.6044
   D    0.8261    0.6453   1.5476     0.2097    0       0.4488    0.5034    4     0.7413
   E    0.7391    0.6036   1.5250     0.1993    1       0.4003    0.4983    3     0.6044
   S    0.7727    0.5267   1.6667     0.1465    1       0.4973    0.4897    4     0.7108
   P    0.8182    0.5051   1.3750     0.2077    0       0.4103    0.6040    3     0.6034
```

Figure 16 geometric features of 'Test_shade'. Note: the first column is the prediction of the shapes of each segment. For this image, there are many wrong prediction because the OCR system can not identify ‘Y’, ‘O’, ‘U’ and ‘R’.

Following is the OCR result using the decision tree discussed before.



```
Problem 1:
Reading raw images...Done!
OCR training...Done!
Testing...

OCR predictions: S P E D L I M T 0 1 2 3 4 5 6 7 8 9
Correct shapes: SPEDLIMT0123456789

OCR predictions: S D P E E T I M I L 5 9 1
Correct shapes: SDPEETIMIL591

OCR predictions: E S E D P L I M I T 0 7
Correct shapes: ESEDPLIMIT07

OCR predictions: 5 P E E D L I I M T 3 1
Correct shapes: SPEEDLIIMT31

OCR predictions: D E E P 5 T M I L I 5 2 D D S 7 E D E S P
DEEPSTMIL52ROUYEDESP
Correct shapes: DEEPSTMIL52ROUYEDESP

Program ended with exit code: 0
```

Figure 17 OCR results for 5 test images (including 'Training')

IV Discussion

The assumptions for this OCR system are that all characters are clearly complete, regardless of fonts and distortion or skew angle. This means the system does not account for most real-world situations, which also be the limitation of the system. However, the chosen geometric features reveal some ground truths.

Segmentation and extract character with bounding box is one key step. Connected component labeling is useful for segment symbols out. However, the results contain every segment including irrelevant background or small group of dots. Thus, in the implementation, some conditions are added to avoid useless segments. For example, the character size should be not too small or large, the aspect ratio of a character should be less or equal to 1. With these modifications, characters are extracted out within bounding box.

However, these conditions are hard criteria that may not be working for a few special images, but it will work for most situations.

The feature computation is the same with part a). The decision tree is modified several times based on the testing results so that it will work for most images. Finally, Figure 17 shows the testing results of this OCR system. The results are acceptable since there are just several wrong recognitions. The results of “Test_ideal1” and “Test_ideal2” are perfect, and one error occurs in “Test_night”; one error occurs in “Test_shade”, and 4 defects since the OCR system is trained without the characters “Y”, “O”, “U” and “R”. In conclusion, the performance is reasonable.

The main problem of the system is the uncertainty of recognizing “5” from “S”. These two characters are very similar. Taking all 4 testing images into account, with the current features used, there is no decision branch that can separate “5” and “S” clearly. In other words, if one decision tree settings work for “Test_ideal1” in separating “5” from “S”, then it may not work for “Test_shade”. Hence, one possible solution is adding other useful features dealing with “5” and “S”, like gradient (since “5” has sharp corner not shown in “S”), or central symmetry.

If without histogram equalization, the results can be better for those testing images to be honest. However, for the robustness of the system, this step is necessary. If some characters’ intensities are very close to background, the characters may not be recognized. With equalization, the difference between the character and background can be magnified so that the character can be extracted successfully.

Another defect of the system is that it cannot identify “white” characters. So, “27” in the “Test_shade” is not defined as characters in the program. However, it can be segmented out successfully shown in Figure 12 (without histogram equalization). If applying other criteria, these two numbers should be recognized easily.

If one wants to enhance the robustness of the system, the decision tree should be built automatically by learning. Another way is utilizing probability decision tree, or going through multiple decision trees and take the most frequent recognition results (similar to random forest). Anyway, there are numbers of places need to improve for this system working in real-world situations.

Problem 2: Contour Modeling

(a), (b), (c), (d) combined

I Abstract and Motivation

In this part, contour modeling is used for image segmentation to medical images. There are two state-of-the-art algorithms, the snake algorithm and level-set algorithm. There are 4 gray-level medical images of different kinds, the main tasks is to extract the specific objects from the images based on contours. These two algorithms are very useful in medical image analysis since the images are gray-level with clear separation between foreground and background. The two algorithms are discussed later.

The snake model, also called as active contour model, is an outline finding mechanism via spline deformation based on energy minimizing. The spline is influenced by external force of constraint and image energy that pull it to the image features like edge and intensity, and the internal force resist deformation. The model involves user interaction that users select initial interest points to restrict the range of ROI. With users take control of some parameters, the object can be segmented more precisely. This model requires the knowledge of approximate boundary shape, so it is not fully automatic but effective. The pros and cons of snake algorithm will be discussed later.

Level-set algorithm is also a contour evolution model. However, the snake one is parametric model while the level-set one is a geometric model which can represent contours more precisely. The concept is easy to understand: the boundary curve is represented as the function of the intersection of the level-set function surface with zero plane. So the level-set function (LSF) is defined as $z = \varphi(x,y,t)$, and the contour then is the intersected curves $\{(x,y) \,|\varphi(x,y,t) = 0\}$. The active contour can be seen as a function of time then $C(s,t)$, and the derivative of t can be decomposed to the speed function controlling the motion of the contour F, and the inward normal vector N of the contour C. The level-set function z takes negative values inside the zero level contour and positive values outside so that N can be expressed as $-\nabla z/|\nabla z|$ where ∇ is the gradient operator. Then, the LSF can be computed by solving partial differential equation (PDE). In the paper[1] of Li, Chunming, et al., numbers of mathematic inductions and calculations are discussed. However, it will not be discussed in this report. The main purpose is to get familiar with the idea and master the program. There are several parameters to initialize the level-set algorithm as well. Besides, the

[1] Li, C., Xu, C., Gui, C., & Fox, M. D. (2010). Distance regularized level set evolution and its application to image segmentation. Image Processing, IEEE Transactions on, 19(12), 3243-3254.

ROI should be set at first. The initial position of ROI can be easily calculated by some pre-processing procedures. In this program, one can choose 2 rectangles as ROIs, then perform the algorithm.

II Approach and Procedures

Snake algorithm

1. Open the GUI of snk.fig with Matlab, and run.
2. Choose the image and Gaussian filter parameter sigma. The larger the sigma, the image is smoothed or blurred more.
3. Choose the initial interest points (will be interpolated as initial contour), and change the parameters:

   α (alpha): Specifies the elasticity of the snake. This controls the tension in the contour by combining with the first derivative term.

   β (beta): Specifies the rigidity in the contour by combining with the second derivative term.

   γ (gamma): Specifies the step size

   κ (kappa): Acts as the scaling factor for the energy term.

   W (Eline): Weighing factor for intensity based potential term.

   W (Eedge): Weighing factor for edge based potential term.

   W (Eterm): Weighing factor for termination potential term.

   Iteration times can also be modified, for comparison set all to 200 times.

4. Run the filter and show the results. Based on the results, change the parameters to find the ideal results.

Level-set algorithm

1. Run the DRLSE demos.
2. Modify demos, change the input images and assign related ROIs.
3. Change parameters:

   Time step Δt: related to the distance regulation term in implementation.

   λ: weighted length term.

   α: weighted area term. This parameter needs to be tuned multiple times.

   ε: width of the dirac-delta function.

   σ: Gaussian kernel standard deviation.

   Iteration times: inner and outer times, the total times is the multiplicity of this two terms.

4. Show the segmentation results and the level-set function.

5. Change the parameters to find ideal results.

III Experimental Results

1. Snake algorithm.

First, to find out each parameter's influence on the results, change the parameter one by one to get familiar with how it works.

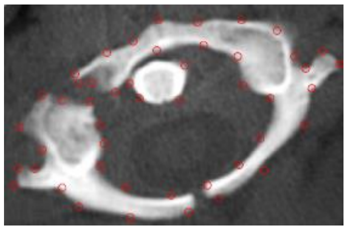
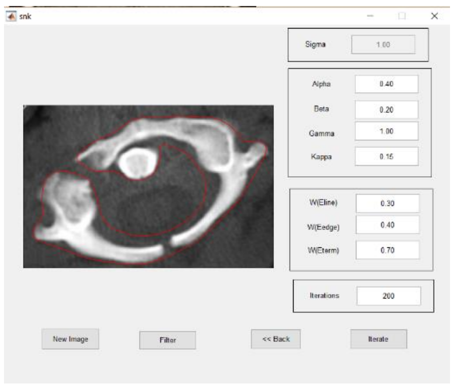
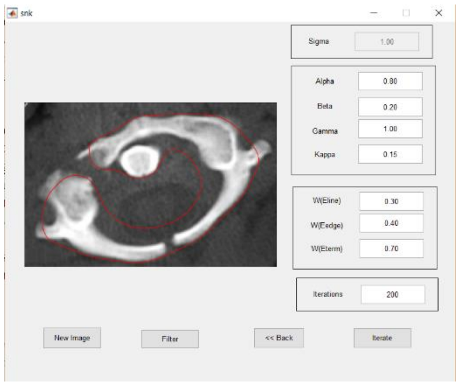


Figure 18 Initial interest points selected



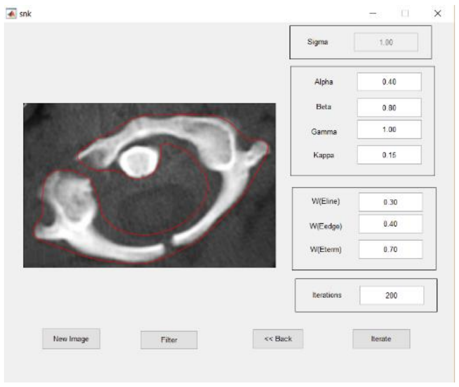
Sigma 1.00
Alpha 0.40
Beta 0.20
Gamma 1.00
Kappa 0.15
W(Eline) 0.30
W(Eedge) 0.40
W(Eterm) 0.70
Iterations 200

Figure 24 default parameter settings result



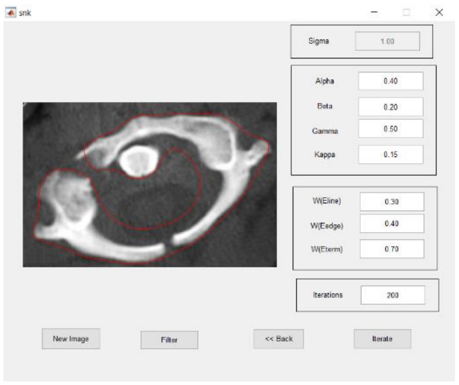
Sigma 1.00
Alpha 0.80
Beta 0.20
Gamma 1.00
Kappa 0.15
W(Eline) 0.30
W(Eedge) 0.40
W(Eterm) 0.70
Iterations 200

Figure 23 alpha=0.80 result



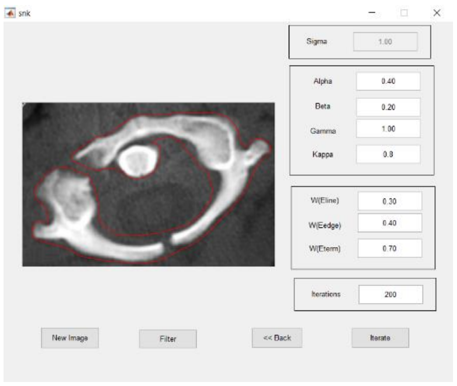
Sigma 1.00
Alpha 0.40
Beta 0.80
Gamma 1.00
Kappa 0.15
W(Eline) 0.30
W(Eedge) 0.40
W(Eterm) 0.70
Iterations 200

Figure 22 beta=0.80 result



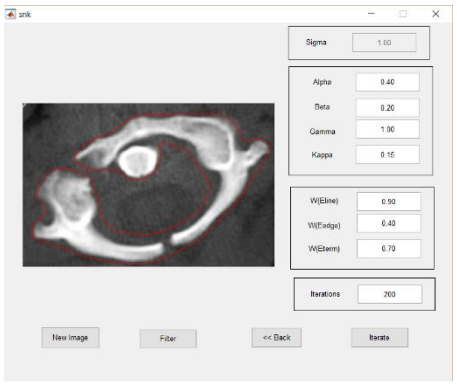
Sigma 1.00
Alpha 0.40
Beta 0.20
Gamma 0.50
Kappa 0.15
W(Eline) 0.30
W(Eedge) 0.40
W(Eterm) 0.70
Iterations 200

Figure 21 gamma=0.50 result



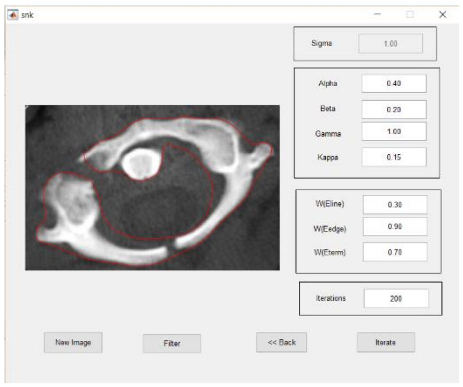
Sigma 1.00
Alpha 0.40
Beta 0.20
Gamma 1.00
Kappa 0.8
W(Eline) 0.30
W(Eedge) 0.40
W(Eterm) 0.70
Iterations 200

Figure 20 kappa=0.80 result



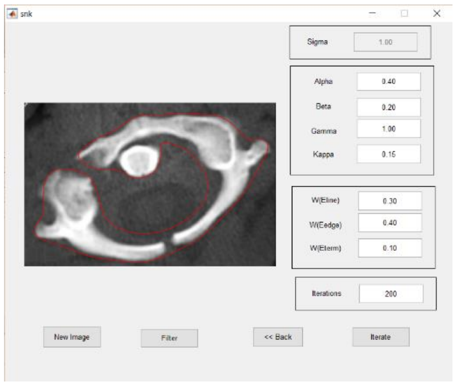
Sigma 1.00
Alpha 0.40
Beta 0.20
Gamma 1.00
Kappa 0.15
W(Eline) 0.90
W(Eedge) 0.40
W(Eterm) 0.70
Iterations 200

Figure 19 W (Eline) =0.90 result



Sigma 1.00
Alpha 0.40
Beta 0.20
Gamma 1.00
Kappa 0.15
W(Eline) 0.30
W(Eedge) 0.90
W(Eterm) 0.70
Iterations 200

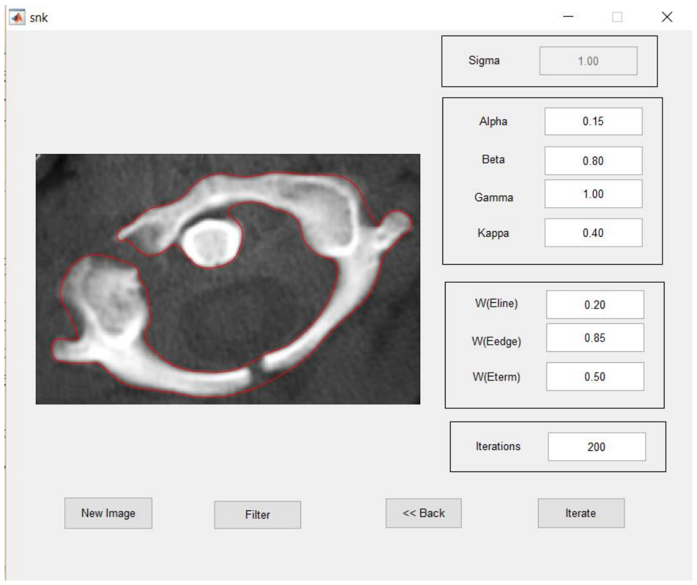
Figure 26 W (Eedge) =0.90 result



Sigma 1.00
Alpha 0.40
Beta 0.20
Gamma 1.00
Kappa 0.15
W(Eline) 0.30
W(Eedge) 0.40
W(Eterm) 0.10
Iterations 200

Figure 25 W (Eterm) =0.10 result

Segmentation result of "spine.raw".



Sigma 1.00
Alpha 0.15
Beta 0.80
Gamma 1.00
Kappa 0.40
W(Eline) 0.20
W(Eedge) 0.85
W(Eterm) 0.50
Iterations 200

Figure 27 A good example result with settings: alpha=0.15; beta=0.80; gamma=1.00; kappa=0.40; W (Eline) =0.20; W (Eedge) =0.85; W (Eterm) =0.50; iterations=200. Some explanations will be in the discussion part.

Segmentation result of “coronary.raw”.

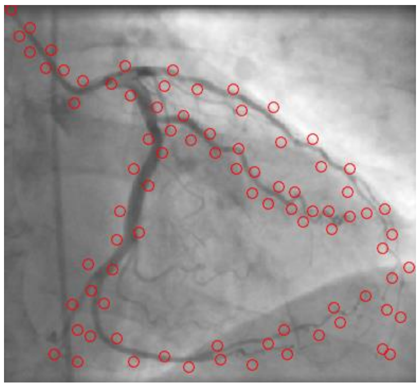
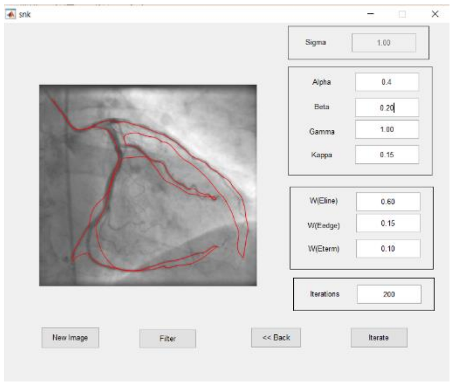
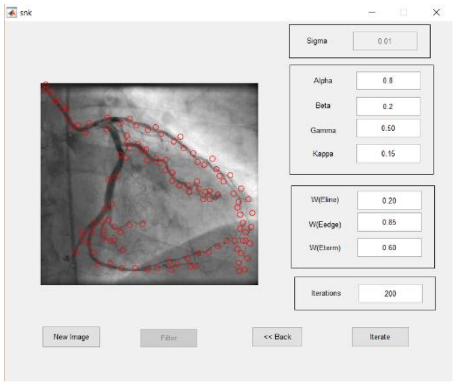


Figure 31 Initial points selected for "coronary"



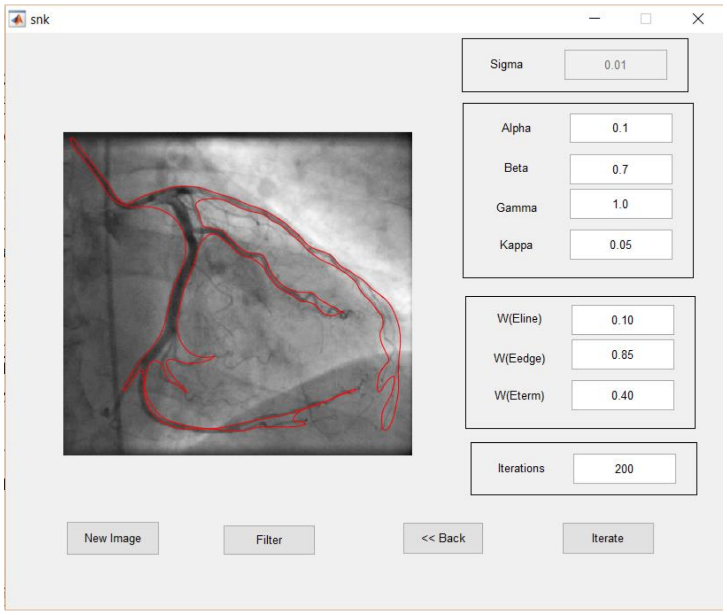
Sigma	1.00
Alpha	0.4
Beta	0.20
Gamma	1.00
Kappa	0.15
W(Eline)	0.60
W(Eedge)	0.15
W(Eterm)	0.10
Iterations	200

Figure 30 segmentation result (not good)



Sigma	0.01
Alpha	0.8
Beta	0.2
Gamma	0.50
Kappa	0.15
W(Eline)	0.20
W(Eedge)	0.85
W(Eterm)	0.60
Iterations	200

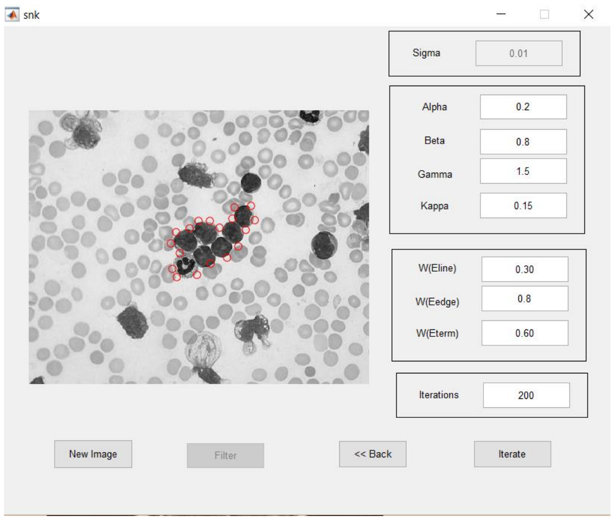
Figure 29 Re-select interest points



Sigma	0.01
Alpha	0.1
Beta	0.7
Gamma	1.0
Kappa	0.05
W(Eline)	0.10
W(Eedge)	0.85
W(Eterm)	0.40
Iterations	200

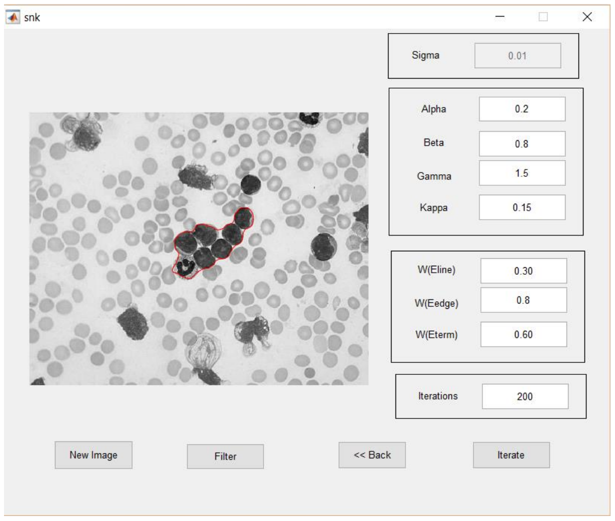
Figure 28 one acceptable result (still not good enough). Alpha=0.10; beta=0.70; gamma=1.00; kappa=0.05; W (Eline) =0.10; W (Eedge) =0.85; W (Eterm) =0.40; iterations=200.

Segmentation result of “blood cells.raw”.



Sigma 0.01
Alpha 0.2
Beta 0.8
Gamma 1.5
Kappa 0.15
W(Eline) 0.30
W(Eedge) 0.8
W(Eterm) 0.60
Iterations 200

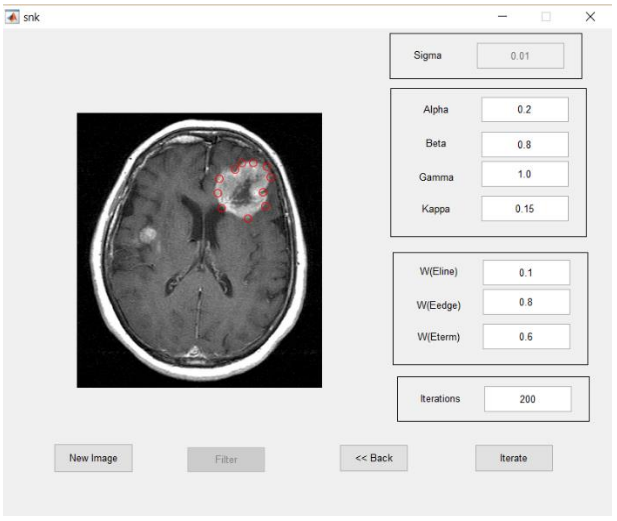
Figure 33 initial selected points for “blood cells”



Sigma 0.01
Alpha 0.2
Beta 0.8
Gamma 1.5
Kappa 0.15
W(Eline) 0.30
W(Eedge) 0.8
W(Eterm) 0.60
Iterations 200

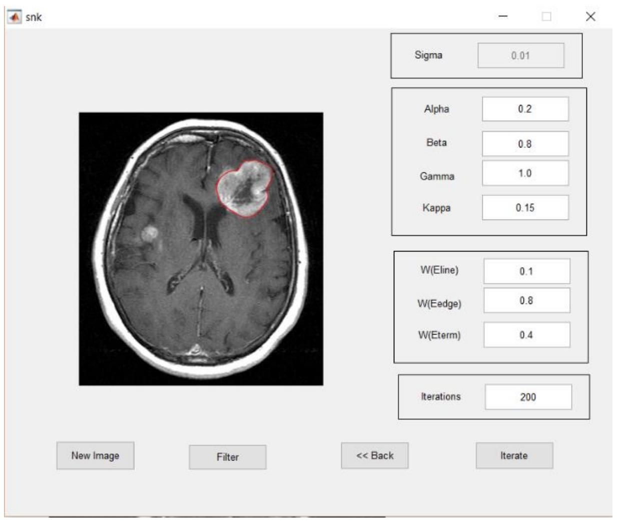
Figure 32 Segmentation result (this implementation can only segment one object out at a time, if using too large ROI, the result is not practical). Alpha=0.20; beta=0.80; gamma=1.50; kappa=0.15; W (Eline) =0.30; W (Eedge) =0.80; W (Eterm) =0.60; iterations=200.

Segmentation result of “brain.raw”



Sigma 0.01
Alpha 0.2
Beta 0.8
Gamma 1.0
Kappa 0.15
W(Eline) 0.1
W(Eedge) 0.8
W(Eterm) 0.6
Iterations 200

Figure 35 Initial interest points selected for "brain.raw"



Sigma 0.01
Alpha 0.2
Beta 0.8
Gamma 1.0
Kappa 0.15
W(Eline) 0.1
W(Eedge) 0.8
W(Eterm) 0.4
Iterations 200

Figure 34 Segmentation result. Note: again, one object at a time. A small tumor is on the left side which can be extracted out too. Alpha=0.20; beta=0.80; gamma=1.00; kappa=0.15; W (Eline) =0.10; W (Eedge) =0.80; W (Eterm) =0.40; iterations=200.

2. Level-set algorithm

First, find the influences of each parameter.

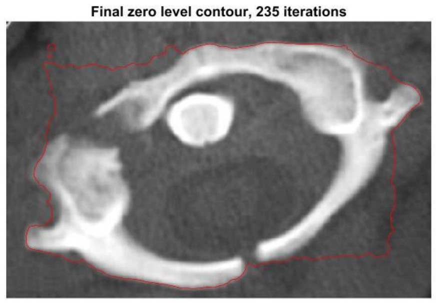


Figure40 timestep=2;iter_inner=15;iter_outer=15; lambda=5;alfa=2.5;epsilon=1.5;sigma=.1;

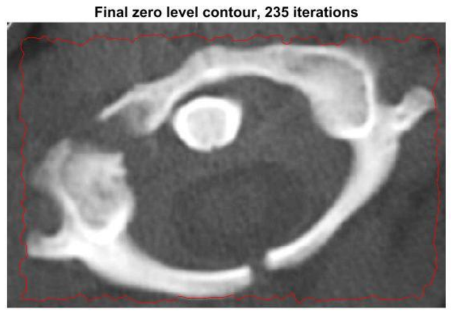


Figure39 timestep=2;iter_inner=15; timestep=2;iter_inner=15;iter_outer=15 ;lambda=5;alfa=0.5;epsilon=1.5;sigma=.1;

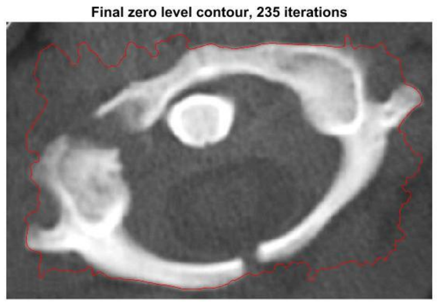


Figure38 timestep=2;iter_inner=15;iter_outer=15; lambda=10;alfa=2.5;epsilon=1.5;sigma =.1;

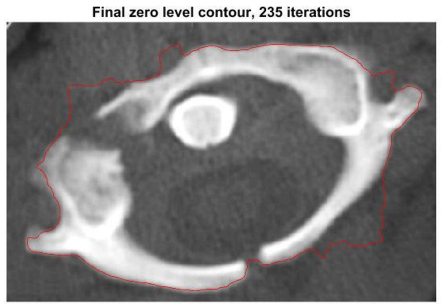


Figure37 timestep=5;iter_inner=15;iter_outer=15; lambda=5;alfa=2.5;epsilon=1.5;sigma=.1;

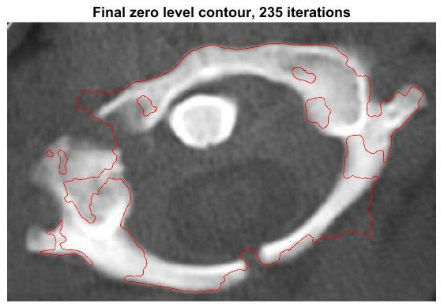


Figure36 timestep=2;iter_inner=15;iter_outer=15; lambda=5;alfa=2.5;epsilon=2.5;sigma=.1;

Segmentation results.

Segmentation results of “spine”.

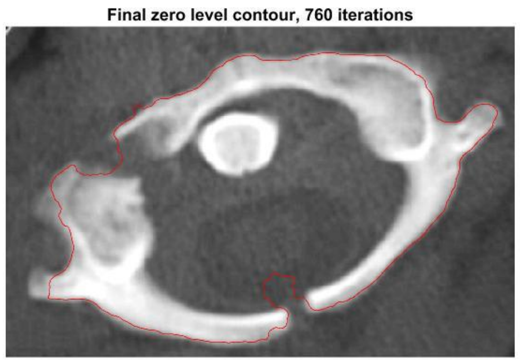


Figure 41 one example result using 1 large ROI

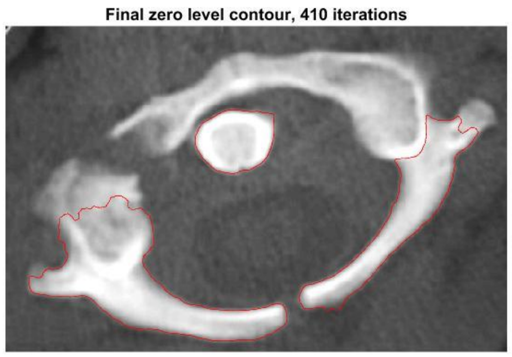


Figure 42 one example result using 3 small ROIs

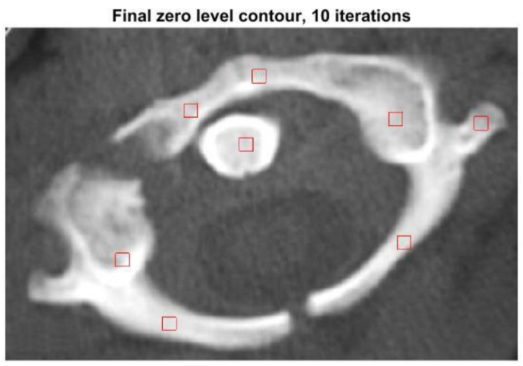


Figure 45 Initial 8 ROIs

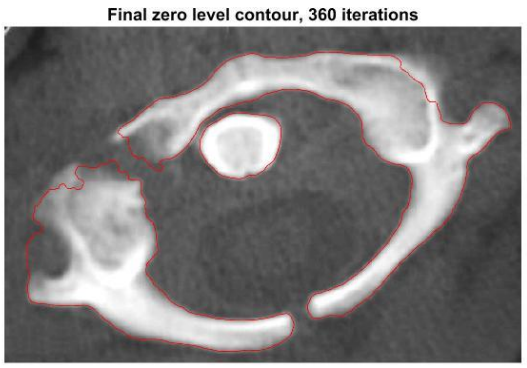


Figure 44 A acceptable result. timestep=75; alpha=-2.4; sigma=0.5

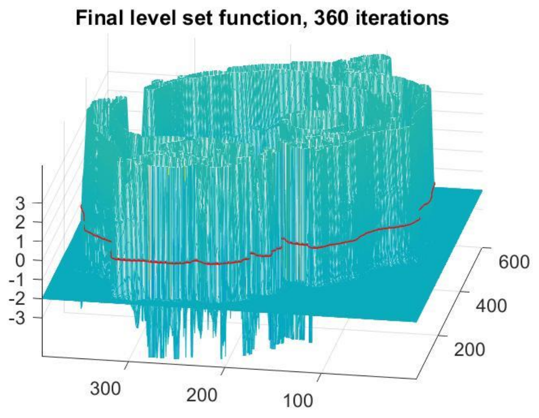


Figure 43 Final LSF

Segmentation results of “coronary”.

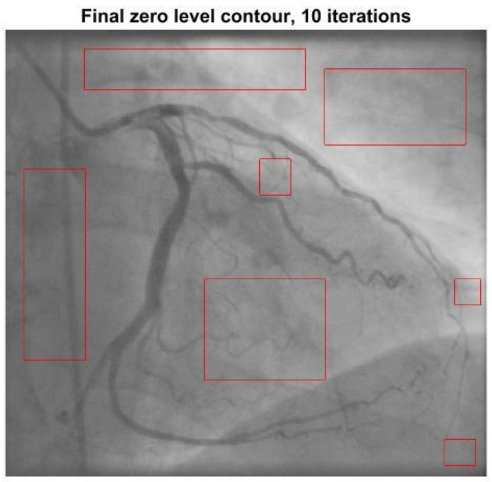


Figure 48 Initial 7 ROIs

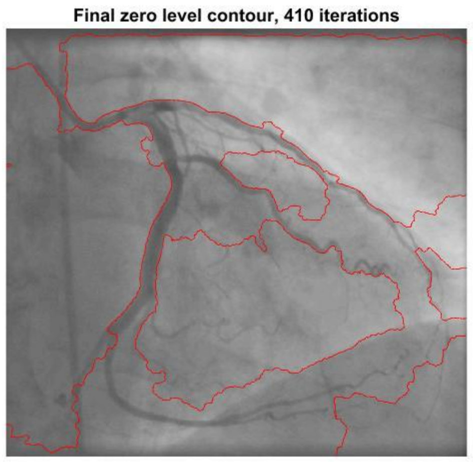


Figure 47 one acceptable result. timestep=65; alpha=-1.6; sigma=0.2. This can be tuned better (need times to try).

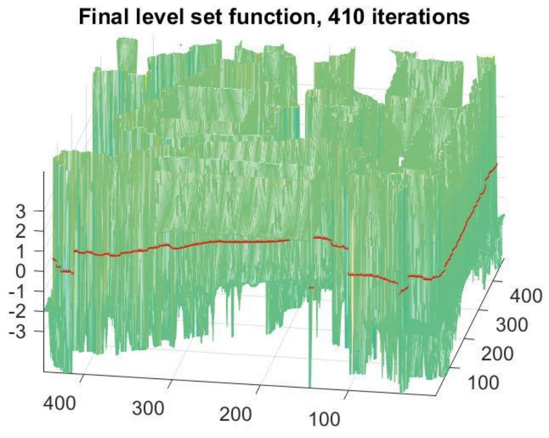


Figure 46 Final LSF

Segmentation results of “blood cells”.

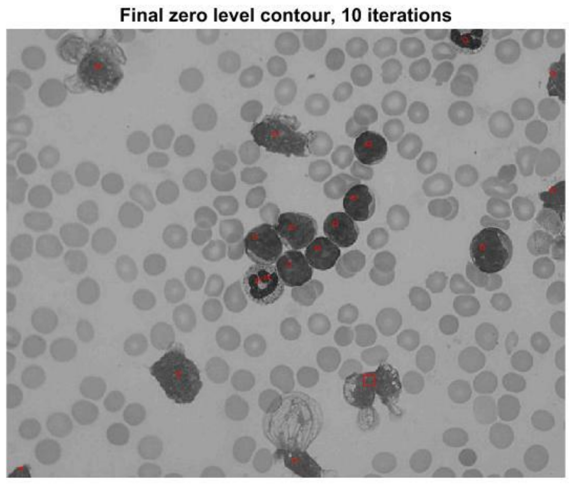


Figure 51 Initial ROIs

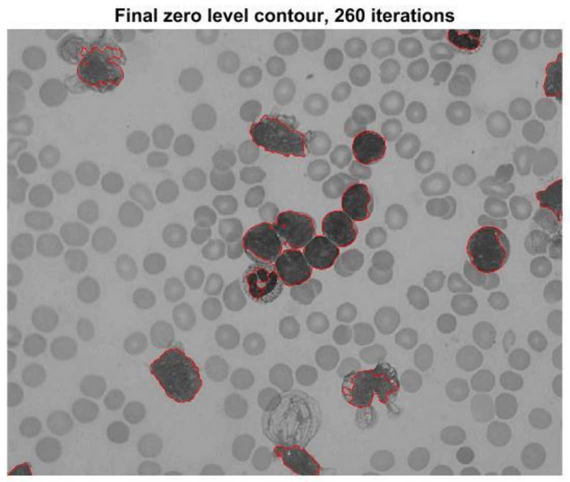


Figure 50 Final results. timestep=65; alpha=-3.8; sigma=0.2.

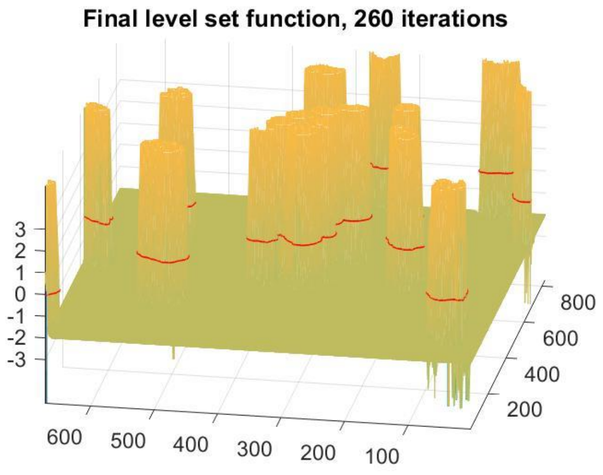


Figure 49 Final LSF

Segmentation results of “brain”.

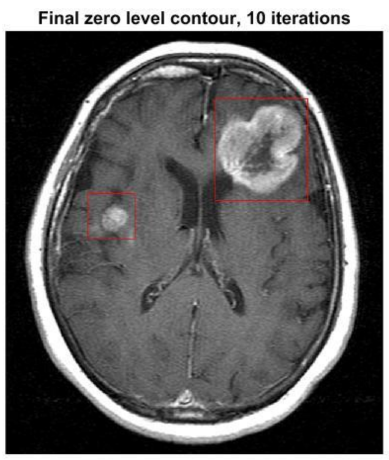


Figure 54 Initial ROIs

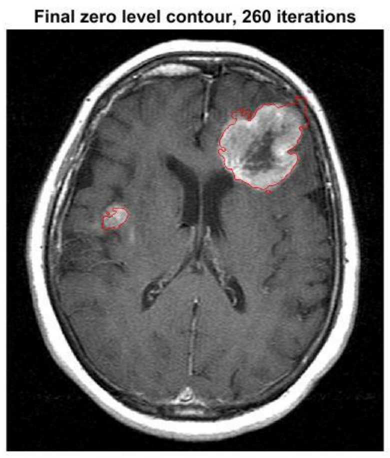


Figure 53 Final result. Timestep =50; alpha=3.5; sigma=0.6. The results can be refined by more iterations with less alpha.

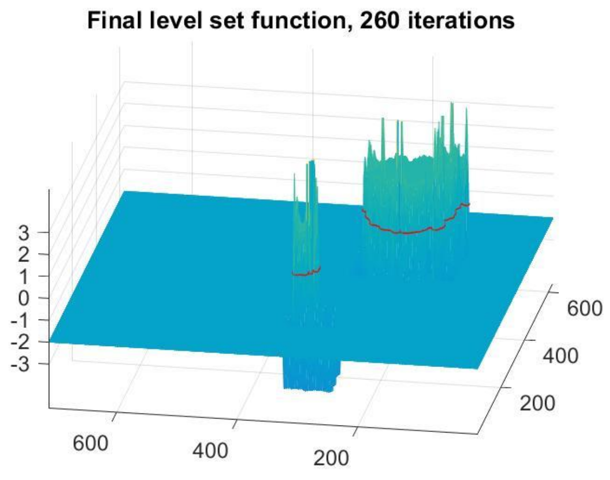


Figure 52 Final LSF

IV Discussion

1. Snake algorithm
   a. Parameter explanation

The performance of the snake algorithm depends on the user's settings. Comparing among Figure 19-26, which are the results by changing only one parameter each time from default settings. The influences of these parameters can be drawn from the observation.

Comparing Fig. 23 to Fig. 24, increasing the alpha gives smoother boundary with larger elasticity of the snake. As a result, the outside snake passes the true boundary while inside one moves away from the true boundary, giving the wrong segmentation.

Comparing Fig. 22 to Fig. 24, the beta controls the rigidity of contour; larger beta gives rougher contours resisting elasticity.

Comparing Fig. 21 to Fig. 24, larger step size gives larger movement of snake.

Comparing Fig.20 to Fig. 24, larger kappa makes the snake more attractive to the image features like edge.

Comparing Fig. 19 to Fig. 24, larger energy weight of line equation makes snake more attractive to intensity maximum or minimum around, so this parameter is highly dependent on the initial contour. If the initial interest points are too far from true boundary, the result will be erroneous.

Comparing Fig. 26 to Fig. 24, larger energy weight of edge equation makes snake more attractive to the gradient maximum around, so larger weight makes sure that the snake do not traverse the true boundary with similar intensity.

Comparing Fig. 25 to Fig. 24, if the termination weight is dominant than other two factors, then the snake will converge to corners first.

b. Results of the 4 testing images

The performances of the snake algorithm towards the four images are not equally good.

Fig. 27 shows a good result from multiple trials. However, the concave parts of the spine are not well converged. There is a balance between external force and internal force. In this situation, if still increasing the external energy weight, some parts of the snake will traverse the true boundary and loose the reality. So, this result is acceptable.

Figure. 28 is the result of “coronary” image. Unfortunately, with several tests, the results of this image are not as expected as it is for “spine” image. This image has too many details, such as numbers of small veins and overlapping veins. This reveals the drawback of the snake that it cannot deal with contours overlapping. The algorithm is not able to separate two connected objects apart. To get best result for this image, the initial contour selected must be very close to the real boundaries for snake converging around small veins; then, some overlaps should be cut.

Figure. 32 is the result of “blood cells” image. Like discussed before, the snake can only segment one object at a time. This is a disadvantage of snake algorithm that it cannot deal with topology change. Hence, white cells in the middle of the image can be segmented out by ignoring other surrounding white cells.

Figure. 34 is the result of “brain” image. This result is acceptable and could be improved accordingly. The “tumor” is obvious in the image, in terms of intensity and edges. Thus, the “tumor” can be separated well.

Overall, there are some patterns found from running the program. Initial interest points should be close to true boundary, otherwise the snake may not converge. In other words, the results are highly dependent on the initial contour. Gaussian kernel sigma controls the range of snake deformation, so blurring edges can increase the chance of convergence.

By all means, from the facts above, the advantages of snake algorithm are: automatic and adaptive search of minimum energy; external energy is of user’s control; Gaussian filter is scale adaptive; dynamic objects are applicable. Of course, disadvantages are obvious: global minimum is not always achievable; details of the contour are ignored when deformation; it is sensitive to convergence policy.

2. Level-set algorithm
    a. Parameter explanation

There are 7 parameters to discuss: time step; iteration times; lambda; alpha; beta; epsilon; sigma.

Sigma is the Gaussian kernel, the same function with snake algorithm. Sigma should not be set too large, otherwise it can cause the zero level-set contour leakage. The level-set contour can be stuck at the position where gradient maximum is achieved but within the object. So, multiple ROIs are needed.

Comparing Fig. 39 with Fig. 40, the alpha is the area weight. Larger alpha gives larger energy of each iteration to pull the zero evolution contour to the real boundary. This term is of the most important since it change the evolution steps. If setting the alpha larger than 0 as Fig. 40, with one ROI assigned, the initial zero contour will shrink even separate itself apart at last; if setting smaller than 0 with 2 ROIs assigned, then the initial zero contour will dilate and even merge together.

Comparing Fig. 38 with Fig. 40, the lambda parameter does influence the result. However, as stated in the paper, lambda has small influences to the results. So, remain lambda unchanged and equals to 5.0.

Comparing Fig. 37 with Fig. 40, with the same number of iteration times, larger time step gives larger evolution contour movements between each iteration.

Comparing Fig. 36 with Fig. 40, epsilon is the width of the DiracDelta function. With larger epsilon, the evolution becomes much stronger. So, the parameter should be tuned carefully or just remain unchanged.

b. Results of the 4 testing images

Overall, the performance of level-set algorithm are much better than snake, especially in terms of topological features of several objects. One thing should be stated that each result can be modified even better by changing parameters more carefully, adding more ROIs and change the position of ROIs.

The ROIs can more than one to make sure the correct segmentation. Take “spine” image as example shown in Fig. 41 & Fig. 42, if one large ROI is selected to cove the whole spine and set alpha larger than zero, the segmentation can only get the outside contour of the object. Or if using 3 small ROIs inside the object and set alpha smaller than zero, then the result shows only parts of the object can be segmented out. Based on these trials, good results can be obtained. Still, the results can be tuned to get even better results.

The main advantage of the level-set approach is that it deals with the problem a higher dimension. So, with this concept, it can keep track of the contours even if it self-intersects. Meanwhile, it can solve topology changes like division and merge. These two items cannot be achieved by snake. Another advantage is that it does not require the initial contours to be very close to the true boundary, so it relaxes user effort.

The performance to “coronary” is much better than snake because level-set can deal with topology changes. The result is not ideal, but it can be tuned more detailed to get better result. Also, the performance of “blood cell” can segment multiple cells at a time, which is better than the results of snake.

On the other hand, the disadvantage becomes clear. To achieve very good results, the iteration time is much larger than snake’s. For snake, 200 iterations can get acceptable results. However, the iteration time for level-set should be over thousand. However, more iteration times make the final contour more accurate than snake’s. So, iteration time is a trade-off for better results.

One problem of level-set implementation is that, it requires the re-initialization to enhance the robustness. It deals with noise, but decreases the efficiency.

Problem 3: Salient Point Descriptors and Image Matching

a) Extraction and Description of Salient Points

I Abstract and Motivation

Scale Invariant Feature Transform (SIFT) is an approach for detecting and extracting local feature descriptors that are reasonably invariant to changes in illumination, image noise, rotation, scaling, and small changes in viewpoint.

Detection stages for SIFT features: - Scale-space extrema detection; - Keypoint localization; - Orientation assignment; - Generation of keypoint descriptors.

Interest points for SIFT features correspond to local extrema of difference-of-Gaussian filters at different scales. Given a Gaussian-blurred image.

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

where

$$G(x,y,\sigma) = 1/(2\pi\sigma^2)exp(-(x^2+y^2)/\sigma^2)$$

is a variable scale Gaussian, the result of convolving and image with a difference-of-Gaussian filter

$$G(x,y,k\sigma) - G(x,y,\sigma)$$

is given by

$$D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,\sigma)$$

Which is just the difference of the Gaussian-blurred images at scales σ and kσ.

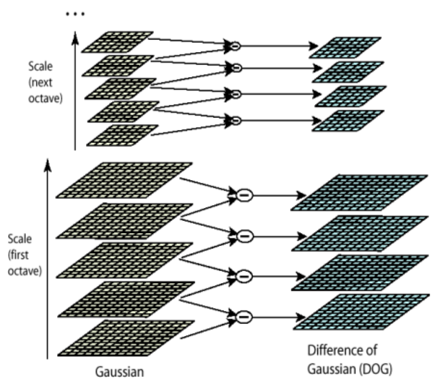


Figure 55 Diagram showing the blurred images at different scales, and the computation of the difference-of-Gaussian images (from Lowe, 2004)

The convolved images are grouped by octave (an octave corresponds to doubling the value of σ), and the value of k is selected so that we obtain a fixed number of blurred images per octave. This also ensures that we obtain the same number of difference-of-Gaussian images per octave.

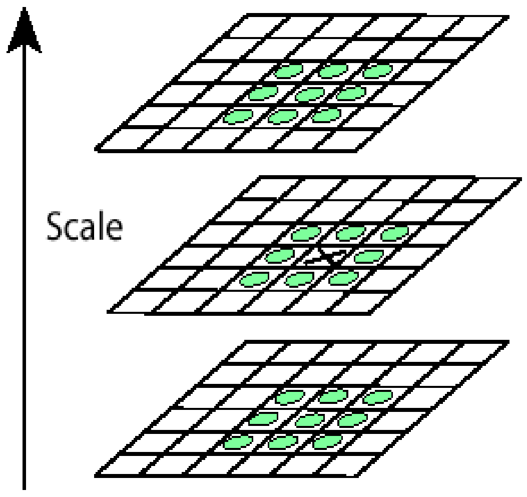


Figure 56 Local extrema detection, the pixel marked × is compared against its 26 neighbors in a 3 × 3 × 3 neighborhood that spans adjacent DoG images (from Lowe, 2004)

Interest points (called keypoints in the SIFT framework) are identified as local maxima orminima of the DoG images across scales. Each pixel in the DoG images is compared to its 8 neighbors at the same scale, plus the 9 corresponding neighbors at neighboring scales. If the pixel is a local maximum or minimum, it is selected as a candidate keypoint.

For each candidate keypoint: -Interpolation of nearby data is used to accurately determine its position. -Keypoints with low contrast are removed. -Responses along edges are eliminated. -The keypoint is assigned an orientation.

To determine the keypoint orientation, a gradient orientation histogram is computed in the neighborhood of the keypoint (using the Gaussian image at the closest scale to the keypoint's scale). The contribution of each neighboring pixel is weighted by the gradient magnitude and a Gaussian window with a σ that is 1.5 times the scale of the keypoint. All the properties of the keypoint are measured relative to the keypoint orientation, and this provides invariance to rotation.

Once a keypoint orientation has been selected, the feature descriptor is computed as a set of orientation histograms on 4 × 4 pixel neighborhoods. The orientation histograms are relative to the keypoint orientation, the orientation data comes from the Gaussian image closest in scale to the keypoint's scale. Histograms contain 8 bins each, and each descriptor contains an array of 4 histograms around the keypoint. This leads to a SIFT feature vector with 4 × 4 × 8 = 128 elements. This vector is normalized to enhance invariance to changes in illumination.

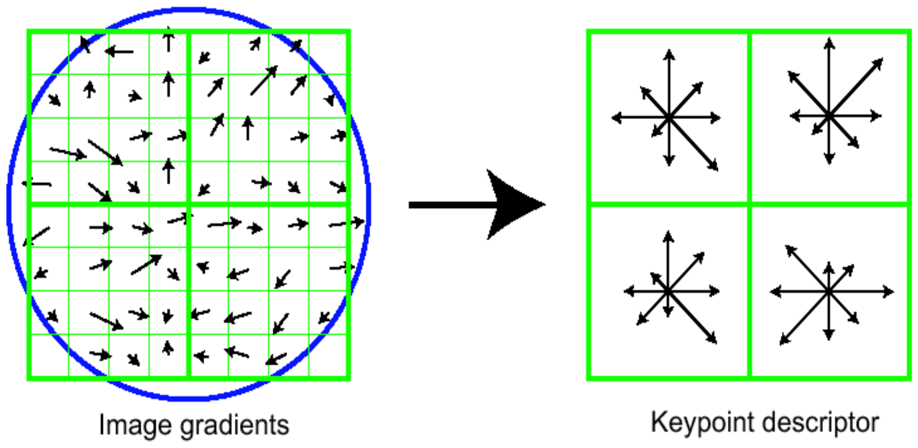


Figure 57 SIFT feature descriptor (from Lowe, 2004)

Speeded Up Robust Features (SURF) is similar to the SIFT, but with faster speed and more robust against image transformations. SURF uses the integration of image for speeding up filtering, then use a blob detector to find the interest points based on Hessian matrix. The maximum of the determinant of the matrix is used for measuring the change of the points around the point. So different from other methods, the scale space is the output of box filters. As for the descriptors, they are obtained by the response of interest points to the Haar Wavelet. And the matching part is similar to the SIFT.

II Approach and Procedures

1. Read testing raw image, and convert to matrix formant Mat().
2. Compute SIFT features or SURF features using built-in classes. Keypoints and corresponding descriptors are obtained.
3. Draw the keypoints on the original image, and show the image.

III Experimental Results

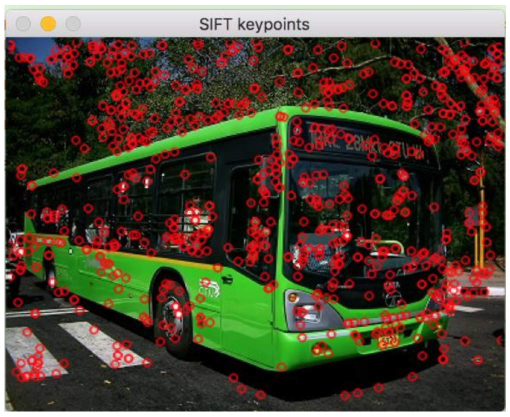


Figure 59 SIFT keypoints of “bus” image

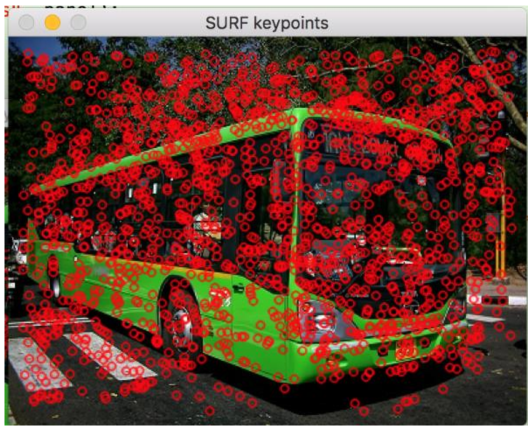


Figure 58 SURF keypoints of bus image

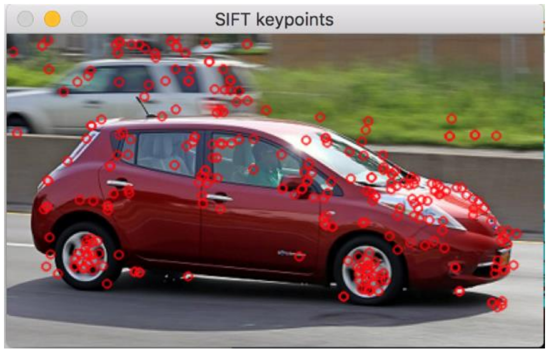


Figure 61 SIFT keypoints of “sedan” image

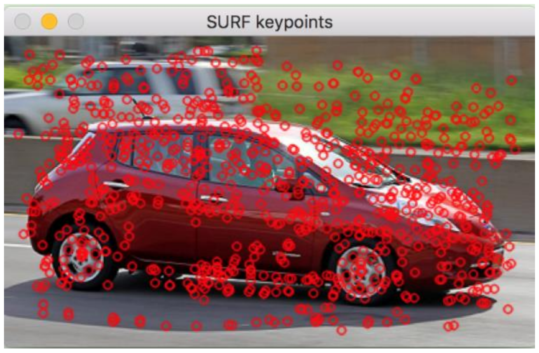
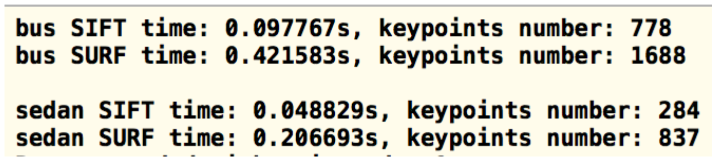


Figure 60 SURF keypoints of “sedan” image



```
bus SIFT time: 0.097767s, keypoints number: 778
bus SURF time: 0.421583s, keypoints number: 1688

sedan SIFT time: 0.048829s, keypoints number: 284
sedan SURF time: 0.206693s, keypoints number: 837
```

Figure 62 Runing time for 2 testing images with SIFT and SURF

IV Discussion

Comparing Fig.58 to Fig.59, Fig. 60 to Fig. 61, SURF key points are more than SIFT key points number. From Fig. 49, the running time of SIFT is less than SURF per point. However, the time is calculated by simple function of clock () which can be introduced error based on the computer environment.

SURF is faster than SIFT in terms of the algorithm of computing the DOG to approximate LOG in scale space with box filters. So, SURF is capable of computing more scales simultaneously. Normally, the SURF interest points are less than SIFT points. One possible explanation of more SURF points is that the images are too nature with too many textures and shapes.

The key point angle is calculated within a square of neighbors and find multiple angles in SIFT. As for SURF, the direction is the Haar wavelet response with largest norm within a sector.

The descriptors involve different numbers of neighbors. SIFT descriptors are calculated by the 128 dimension gradient histograms. SURF descriptors records the Haar wavelet response with 64 dimension.

On the other hand, SIFT is more robust in scale and rotation invariance. More computations make sure this robustness. However, SURF has better performances in resisting blurring and illuminance changing. This is mainly because the SURF descriptors are computed based on original image using its integral image, while SIFT ones are calculated based on the Gaussian pyramid.

b) Image Matching

I Abstract and Motivation

There are kinds of matching methods with different performances based on the images. In this part, SIFT features are used for image objects matching. Image matching is a hard problem since the results are highly influenced by view angle, partially observable objects, scale, rotation, illuminance and resolutions. So, SIFT can solve the problem of scale and rotation of the object, then match the key points pairs. However, there can be other solutions of better performances for these testing images.

II Approach and Procedures

1. To match "school_bus_1" to "school_bus_2", first extract SIFT features for both images.
2. Find match pairs by brute-force descriptor matcher.
3. Draw the matched pairs.

III Experimental Results

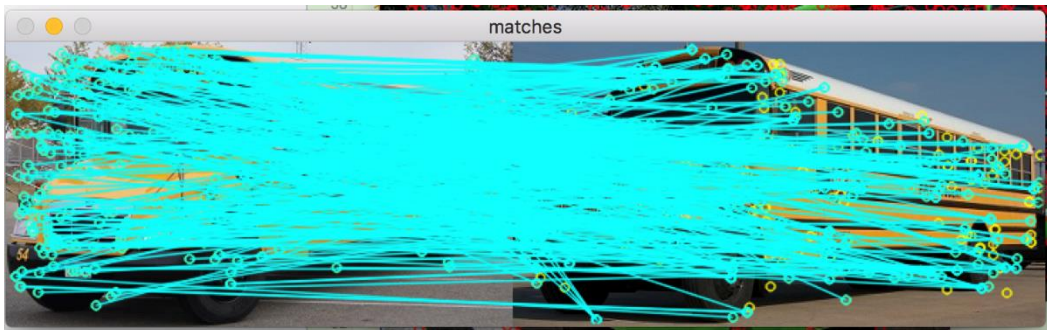


Figure 63 Image matching between "school_bus_1" and "school_bus_2"

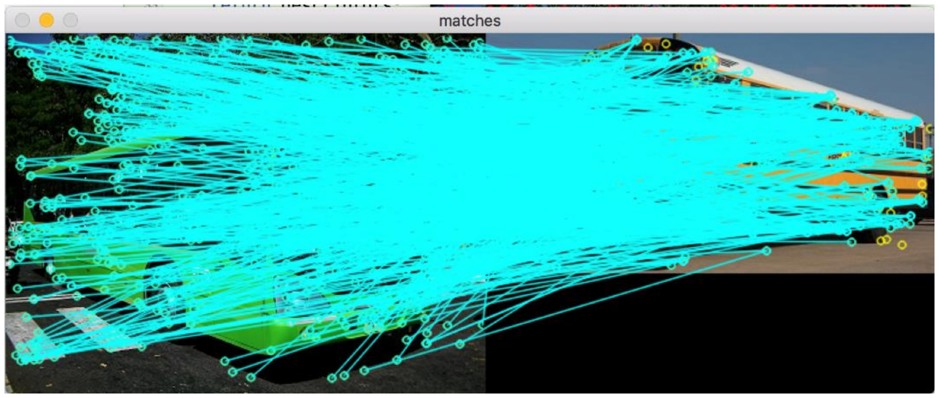


Figure 65 Image matching between "school_bus_1" and "bus"

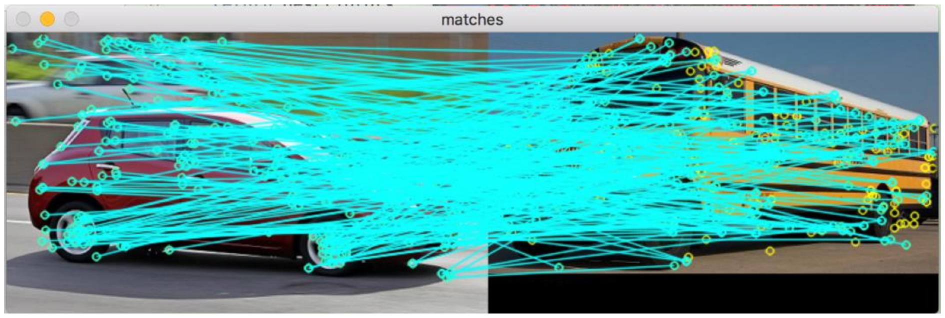


Figure 64 Image matching between "school_bus_1" and "sedan"

IV Discussion

Two factors to determine whether two images are matched. One is the parallelism of the lines drawn between matched pairs. The other the number of match pairs. The matcher will find matched key point from the second image to the key points in the first image. So, if the second image has more key points, then there will be a few unmatched key points in second image.

Fig. 63 has the best match, between two school buses. Most lines remain parallel even if there are some points remain unmatched. The view angle is different, so some lines of some parts of the school bus will intersect.

Fig. 64 and 65 do not have good matching results. Some obvious mismatch can be found. For example, in Fig. 52, the zebra crossing lines are matched to the body of the second school bus. In Fig.51, the white car in the background is matched. However, this is due to the corresponding key points are not found in the second image. More important, the lines in these two images are highly intersected losing much parallelism.

c) Bag of Words

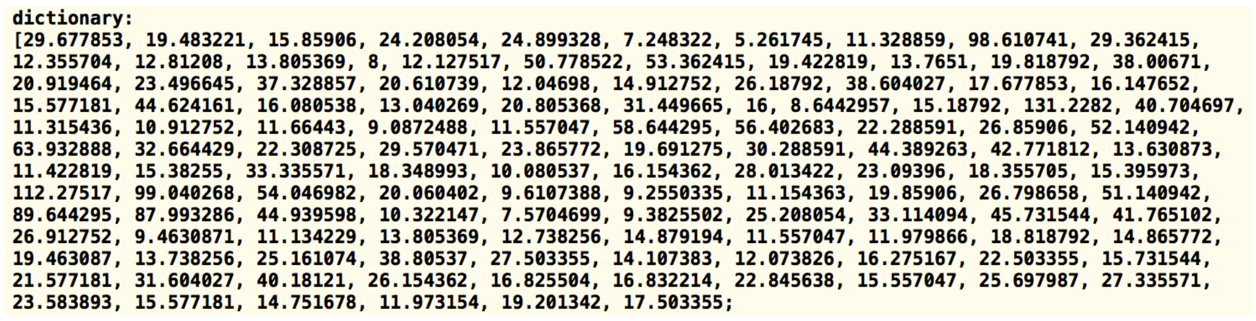
I Abstract and Motivation

The ideal of Bag of Words (BoW) is introduced from document classification. The BoW is a vector of occurrence counts of words in a document, which can be expressed as a histogram of each word. This can be applied to computer vision that small patches can be seen as words. The training of BoW is to cluster all training images into several clusters (words). With this training, a dictionary with cluster means can be generated. Then, the computed features of testing image can be compared to dictionary to get the histogram of BoW. Hence, one histogram stands for the feature conclusion of one image. Bag of Words is useful in image classification for not too large system, however the number of clusters should be defined by user.

II Approach and Procedures

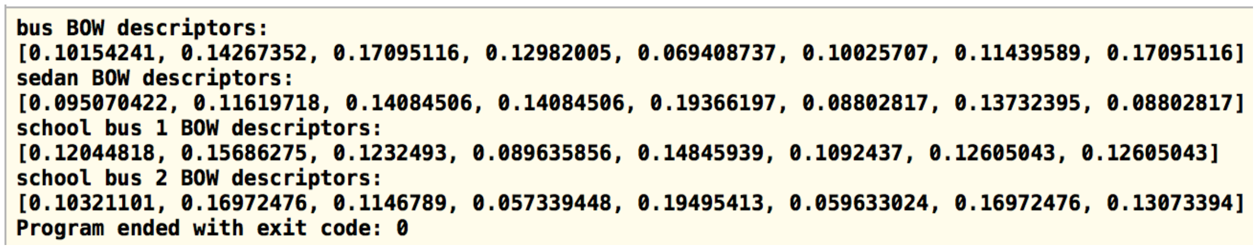
1. Read all five images, and compute SIFT features for all 5 images.
2. Combine the SIFT descriptors of "bus", "sedan", "school_bus_1" and "school_bus_2" as the training descriptors.
3. Apply k-means clustering to the training descriptors, k=8. Then, the dictionary is generated as an 8*128 matrix, with each row represents the mean descriptor of that cluster.
4. Test the descriptor of "school_bus_2" to match with the dictionary, and get the histogram of code words.

III Experimental Results



```
dictionary:
[29.677853, 19.483221, 15.85906, 24.208054, 24.899328, 7.248322, 5.261745, 11.328859, 98.610741, 29.362415,
12.355704, 12.81208, 13.805369, 8, 12.127517, 50.778522, 53.362415, 19.422819, 13.7651, 19.818792, 38.00671,
20.919464, 23.496645, 37.328857, 20.610739, 12.04698, 14.912752, 26.18792, 38.604027, 17.677853, 16.147652,
15.577181, 44.624161, 16.080538, 13.040269, 20.805368, 31.449665, 16, 8.6442957, 15.18792, 131.2282, 40.704697,
11.315436, 10.912752, 11.66443, 9.0872488, 11.557047, 58.644295, 56.402683, 22.288591, 26.85906, 52.140942,
63.932888, 32.664429, 22.308725, 29.570471, 23.865772, 19.691275, 30.288591, 44.389263, 42.771812, 13.630873,
11.422819, 15.38255, 33.335571, 18.348993, 10.080537, 16.154362, 28.013422, 23.09396, 18.355705, 15.395973,
112.27517, 99.040268, 54.046982, 20.060402, 9.6107388, 9.2550335, 11.154363, 19.85906, 26.798658, 51.140942,
89.644295, 87.993286, 44.939598, 10.322147, 7.5704699, 9.3825502, 25.208054, 33.114094, 45.731544, 41.765102,
26.912752, 9.4630871, 11.134229, 13.805369, 12.738256, 14.879194, 11.557047, 11.979866, 18.818792, 14.865772,
19.463087, 13.738256, 25.161074, 38.80537, 27.503355, 14.107383, 12.073826, 16.275167, 22.503355, 15.731544,
21.577181, 31.604027, 40.18121, 26.154362, 16.825504, 16.832214, 22.845638, 15.557047, 25.697987, 27.335571,
23.583893, 15.577181, 14.751678, 11.973154, 19.201342, 17.503355;
```

Figure 66 First row of dictionary



```
bus BOW descriptors:
[0.10154241, 0.14267352, 0.17095116, 0.12982005, 0.069408737, 0.10025707, 0.11439589, 0.17095116]
sedan BOW descriptors:
[0.095070422, 0.11619718, 0.14084506, 0.14084506, 0.19366197, 0.08802817, 0.13732395, 0.08802817]
school bus 1 BOW descriptors:
[0.12044818, 0.15686275, 0.1232493, 0.089635856, 0.14845939, 0.1092437, 0.12605043, 0.12605043]
school bus 2 BOW descriptors:
[0.10321101, 0.16972476, 0.1146789, 0.057339448, 0.19495413, 0.059633024, 0.16972476, 0.13073394]
Program ended with exit code: 0
```

Figure 67 Histogram of BOW of each image

IV Discussion

Fig. 53 is the first row of the dictionary. It is the mean vector of first cluster. The match is achieved by comparing each of descriptor of a testing image to each row of the dictionary; if the descriptor is closest to, say second cluster, then the descriptor will be clustered to second cluster, which means that the second bin will increase by 1 of the 8-bin histogram. With all descriptors matched, the histogram is normalized to 1.

Fig. 54 shows the histogram of each testing image. From observation, the histogram of "school_bus_1" and "school_bus_1" are similar and closed. The distance between the histograms demonstrate the similarity between images. Small distance means high similarity. For "bus" image, 2nd and 8th bins are of highest values; for "sedan", 5th bin has the highest value; for "school_bus_1" and "school_bus_2", 2nd and 5th bins has high values.

Actually, object classification can be achieved further then. For example, training images can be classified as 2 categories with label 1 and 2, 1 for "sedan" and 2 for "bus". Take one test image and compute its histogram, then by calculating the k-nearest neighbor distances, the major vote of the label is the prediction classification of the test image.