University of Louisville

ThinkIR: The University of Louisville's Institutional Repository

Electronic Theses and Dissertations

8-2011

Coronal loop detection from solar images and extraction of salient contour groups from cluttered images.

Nurcan Durak
University of Louisville

Follow this and additional works at: <https://ir.library.louisville.edu/etd>

Recommended Citation

Durak, Nurcan, "Coronal loop detection from solar images and extraction of salient contour groups from cluttered images." (2011). *Electronic Theses and Dissertations*. Paper 386.
<https://doi.org/10.18297/etd/386>

This Doctoral Dissertation is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.

CORONAL LOOP DETECTION from SOLAR IMAGES and
EXTRACTION of SALIENT CONTOUR GROUPS from
CLUTTERED IMAGES

By
NURCAN DURAK
B.S., Baskent University, 2001
M.S., Middle East Technical University, 2004

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

August 2011

Copyright 2011 by Nurcan Durak

All rights reserved

CORONAL LOOP DETECTION FROM SOLAR IMAGES and EXTRACTION of SALIENT CONTOUR GROUPS from CLUTTERED IMAGES

By
NURCAN DURAK
B.S., Baskent University, 2001
M.S., Middle East Technical University, 2004

A Dissertation Approved on
August 8, 2011

by the following Dissertation Committee:

Olfa Nasraoui

Adel Elmaghraby

Ahmed Desoky

Ming Ouyang

Ayman El-baz

Dedicated to the memory of My Father, Selahattin Durak,

who passed away on 08/08/2010.

His stories are an inspiration to my journey.

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my advisor, Dr. Olfa Nasraoui, for her positive guidance, suggestions, corrections, support, and belief in me. I feel very lucky to have worked with her, to be a student of hers, and to simply have known her. She gave me opportunities to have experiences at every level of academia from mentoring junior students in research to grant writing. She made so many things possible for me. To her, I owe a lot.

I would like to thank all the committee members for accepting to serve on my committee and for their critique and suggestions on my dissertation. I would like to thank Dr. Joan Schmelz for her Astrophysics related feedback on my journal papers. I would like to thank Dr. Li Feng for the data and knowledge that she shared during my studies. Special thanks to the Solar Physics community at the Solar Image Processing workshops for their valuable feedback on my research. I am also thankful for my lab mates and undergraduate students who got involved in the Solar Loop Mining project.

I am grateful to NASA and to NSF for their financial support during my doctoral studies including the financing of my travels to attend workshops and conferences. I would like to thank the Computing Research Association (CRA) for supporting me in order to attend the CRA workshops starting from my first year. I became more motivated with inspiration from the many successful women researchers at their meetings. I would like to express my appreciation to the Association of Computing Machinery (ACM) for

their student research competitions, their travel support, and their student volunteering opportunities as well.

I am greatly indebted to all of my teachers during my education from primary school to graduate school. Each of them contributed much to me, affecting me greatly and helping me along my way. I am especially indebted to my Masters thesis advisor, Dr. Adnan Yazici, who gave me insight into doing research.

I would like to thank all of my family members: my mom, my dad, my brothers, my sisters, my nieces, and my nephews. They have always supported me during my education. Even though I am thousands of miles away from them, their care, prayers, and endless love always reached me when I was in need. I am as proud of them as they are proud of me.

My sincere thanks go to my dear friends in Turkey who encouraged me to pursue my PhD abroad and ease my homesickness with their friendship that no distance can diminish. I am grateful for the wonderful people that I have met in Louisville; their friendship enriched my life and left me with unforgettable memories. I am thankful to all the people who have touched my life in a way.

Last but not least, I would like to express my thanks to Thomas, through good times and bad, his patience, positive attitude, and comforting words were always with me.

ABSTRACT

CORONAL LOOP DETECTION from SOLAR IMAGES and EXTRACTION of SALIENT CONTOUR GROUPS from CLUTTERED IMAGES

Nurcan Durak
August 8, 2011

This dissertation addresses two different problems: 1) coronal loop detection from solar images: and 2) salient contour group extraction from cluttered images.

In the first part, we propose two different solutions to the coronal loop detection problem. The first solution is a block-based coronal loop mining method that detects coronal loops from solar images by dividing the solar image into fixed sized blocks, labeling the blocks as “*Loop*” or “*Non-Loop*”, extracting features from the labeled blocks, and finally training classifiers to generate learning models that can classify new image blocks. The block-based approach achieves 64% accuracy in 10-fold cross validation experiments. To improve the accuracy and scalability, we propose a contour-based coronal loop detection method that extracts contours from cluttered regions, then labels the contours as “*Loop*” and “*Non-Loop*”, and extracts geometric features from the labeled contours. The contour-based approach achieves 85% accuracy in 10-fold cross validation experiments, which is a 20% increase compared to the block-based approach.

In the second part, we propose a method to extract semi-elliptical open curves from cluttered regions. Our method consists of the following steps: obtaining individual smooth contours along with their saliency measures; then starting from the most salient contour, searching for possible grouping options for each contour; and continuing the grouping until an optimum solution is reached. Our work involved the design and development of a complete system for coronal loop mining in solar images, which required the formulation of new Gestalt perceptual rules and a systematic methodology to select and combine them in a fully automated judicious manner using machine learning techniques that eliminate the need to manually set various weight and threshold values to define an effective cost function. After finding salient contour groups, we close the gaps within the contours in each group and perform B-spline fitting to obtain smooth curves. Our methods were successfully applied on cluttered solar images from TRACE and STEREO/SECCHI to discern coronal loops. Aerial road images were also used to demonstrate the applicability of our grouping techniques to other contour-types in other real applications.

Keywords: coronal loops, solar images, coronal loop detection, feature extraction, pattern recognition, classification, curve tracing, contour extraction, contour grouping, perceptual rules

TABLE OF CONTENTS

1	INTRODUCTION	1
1.1	Problem Description and Objectives	3
1.2	Challenges	10
1.2.1	Challenges of Coronal Loop Detection	11
1.2.2	Challenges of salient contour group extraction from clutter	13
1.3	Contributions	16
1.3.1	Coronal Loop Detection	16
1.3.2	Salient Contour Grouping	20
1.3.3	Image Specifications	22
1.4	Organization of this Dissertation	23
2	BACKGROUND and RELATED STUDIES	24
2.1	Automatic Detection of Solar Events	25
2.2	Curve Processing	31
2.2.1	Curve Tracing.....	31
2.2.2	Curve Extraction	35
2.2.3	Curve Segmentation	37
2.3	Feature Extraction.....	42
2.3.1	Statistical Features.....	43
2.3.2	Edge Histogram Descriptors	44
2.3.3	Histogram of Oriented Gradients	45
2.3.4	Curvature Features	46
2.4	Classification	50
2.5	Ellipse Detection.....	56
2.5.1	Ellipse Fitting	57
2.5.2	The Hough Transform.....	60
2.6	Contour Grouping.....	63
2.6.1	Perceptual Organization	63
2.6.2	Salient Curve Detection	65
2.6.3	Grouping Methods.....	69
2.6.4	Grouping Measures	73
2.7	Discussion of the Limitations of Related Work.....	78
3	CORONAL LOOP DETECTION FROM THE SOHO/EIT IMAGE COLLECTION.....	80
3.1	Block-based Solar Loop Mining Approach on Raw Images	81
3.1.1	Image Preparation	82

3.1.2	Block Extraction and Labeling.....	85
3.1.3	Feature Extraction	89
3.1.4	Training Classifiers	105
3.1.5	Feature Evaluation.....	108
3.1.6	Solar Cycle-based Experimental Results.....	109
3.1.7	Testing Phase	111
3.2	Block-based Approach on IDL-cleaned Images	116
3.2.1	Image Preparation	117
3.2.2	Feature Extraction	122
3.2.3	Classification Experimental Results.....	127
3.3	Contour-based model on IDL-cleaned images.....	129
3.3.1	Strip Extraction	131
3.3.2	Principal Contour Extraction.....	133
3.3.3	Geometric Feature Extraction	140
3.3.4	Classification and Experimental Results	148
3.3.5	Experimental Results on Outside Blocks	149
3.3.6	Testing Inside Disk Blocks.....	151
3.3.7	Image Retrieval Tool.....	153
4	EXTRACTION of SALIENT CONTOUR GROUPS from CLUTTER.....	157
4.1	Discrete Contour Extraction	158
4.1.1	Curve Segmentation	159
4.1.2	Saliency Computation	164
4.1.3	Neighbor Association.....	165
4.2	Pairing Contours	167
4.2.1	Angular dissimilarity.....	168
4.2.2	Ellipticity.....	169
4.2.3	Eccentricity	176
4.2.4	Measures based on Point-to-Chord distance	176
4.2.5	Proximity.....	182
4.2.6	Weighing the measures	183
4.3	Contour Grouping.....	190
4.4	Experimental Results	199
4.5	Summary.....	206
5	CONCLUSION	208
5.1	Discussion.....	209
5.2	Further Directions.....	216
	REFERENCES.....	218
	CURRICULUM VITAE	228

LIST OF TABLES

Table 1-1 Data specifications used in the dissertation.....	24
Table 2-1 Confusion matrix for a two-class problem.....	57
Table 3-1 Statistical features	91
Table 3-2 Hough Transform based features	93
Table 3-3 Hough Transform based EHD features	93
Table 3-4 Edge based spatial features	94
Table 3-5 Curvature features	106
Table 3-6 The investigated classifiers	107
Table 3-7 Block based precision and recall values of various classifiers.....	108
Table 3-8 Cycle based testing results for image retrieval.....	113
Table 3-9 Confusion matrix for image based testing results	116
Table 3-10 Histogram of Eigenvector Orientations.....	127
Table 3-11 Second Order Derivatives Statistics	128
Table 3-12 Eigen-based Features	129
Table 3-13 Curvature based Features	130
Table 3-14 Block based precision and recall values of various classifiers.....	131
Table 3-15 Accuracy of the Principal Contour Extraction from cluttered regions	138
Table 3-16 Elliptical features from the contours	148
Table 3-17 Point-to-Chord distance features	150
Table 3-18 Classifier Results of Contour based approach.....	151
Table 3-19 Sample regions along with the extracted contours and their predicted contour label.	153
Table 3-20 Inside blocks and their decisions.....	155
Table 3-21 Confusion matrix for image based testing results	160

Table 4-1 Pair-wise versus group-wise measures (Red = Positive, Blue = Negative).....	190
--	-----

LIST OF FIGURES

<i>Figure 1-1</i> The interaction between the Sun and Earth along with the designated satellites	1
<i>Figure 1-2</i> SOHO/EIT image on the left versus TRACE image on the right. SOHO/EIT captures the Sun globally while TRACE provides fine detail with high resolution images.....	2
<i>Figure 1-3</i> Coronal loop regions (a) A coronal loop region outside of the Sun taken by SOHO/EIT under 171 Ångstroms (b) A coronal loop region taken by TRACE	4
<i>Figure 1-4</i> (a) Original image taken by STEREO/SECCHI, (b) Corresponding image after applying the ridge detection method in (Inhester, et al., 2007), (c) Ideal coronal loop segments are extracted manually from the clutter.....	7
<i>Figure 1-5</i> (a) TRACE image taken on May 19, 1998, at a wavelength of 173 Ångstroms, (b) Steger curve tracing algorithm is applied on the image (Aschwanden, et al., 2007). Note that top parts of the loops disappear.	8
<i>Figure 1-6</i> The result of road extraction on aerial image in an agricultural area in IRS data. Note that there are gaps between related road segments (Bacher, 2004).....	9
<i>Figure 1-7</i> Urban regions may generate clutter in Steger’s road detection algorithm (Steger, et al., 1999). (a) original image, (b) Extracted roads which contain heavy clutter in the urban region.....	10
<i>Figure 1-8</i> Expert marked loops. (a) A big loop (b) A vague loop (c) A noisy loop (d) A small loop (e) A loop interfering with another solar event (prominence)	12
<i>Figure 1-9</i> Regions that have no loops, but contain other activities that hard to distinguish from loops	13
<i>Figure 1-10</i> Existence of squiggles, intersecting curves, and wishbones make individual contour extraction challenging	14
<i>Figure 1-11</i> The global salient group consists of {red, orange, yellow} contours. The candidate list for the red segment is {green, blue, pink, orange}. The selection of the contour to be grouped is critical.	15

<i>Figure 1-12</i> General structure of the block-based coronal loop mining approach.....	17
<i>Figure 1-13</i> General structure of the contour-based approach.....	18
<i>Figure 1-14</i> Architecture of the extraction of of salient contour groups.....	20
<i>Figure 2-1</i> Full disk H α images (a) Prominences are on the solar limb, (b) Filaments are on the solar disk	28
<i>Figure 2-2</i> Sunspot evolution over three days. The shape and the size of sunspots change in time. Images from TRACE: http://trace.lmsal.com/	29
<i>Figure 2-3</i> Steger's method (Steger, 1998) applied on a brain image. The curve region in the red circle is missed.	34
<i>Figure 2-4</i> The red point is the last point of the traced curve. The segment starting with yellow dot is supposed to be selected. Since the curve structure starting with the blue point is close to the last point and in the range of the search space, the wrong curve will be traced by Raghupathy's algorithm (Raghupathy, et al., 2004).	35
<i>Figure 2-5</i> (a) Curvature definition with K -cosine, (b) Corner detection based on the K -cosine curvature plot (Sun, et al., 2007).....	38
<i>Figure 2-6</i> Segmenting a curve at the corners, then finding linear and elliptical curves according to Paramanand's method (Paramanand, 2006).....	39
<i>Figure 2-7</i> The curve is segmented at the transitions. A transition likelihood graph is shown on the left. (Fischler, et al., 1994)	41
<i>Figure 2-8</i> Spatial filter masks: vertical, horizontal, 45-degree, 135-degree, non-directional.....	44
<i>Figure 2-9</i> Different values of the shape index correspond to different shapes.....	48
<i>Figure 2-10</i> According to 1-NN, the label of X becomes "rectangle". According to 5-NN, the label of X becomes "circle"	52
<i>Figure 2-11</i> Eccentricity values for different ellipses. It gets close to 0 when the ellipse becomes more circular and close to 1 when the ellipse becomes more elongated.	60
<i>Figure 2-12</i> The input and output spaces of Hough Transformation	61
<i>Figure 2-13</i> Determining the center of an ellipse using three random points. The ellipse center is located at the intersection of the bisections of three tangent lines to the ellipse.	63
<i>Figure 2-14</i> Gestalt laws of perceptual organization	64

<i>Figure 2-15</i> Two segments and their parameters for modeling the proximity, good continuation, and similarity (Goldberg, et al., 2002)	65
<i>Figure 2-16</i> A circle in a cluttered background on the left and its saliency map on the right (Ullman, et al., 1988)	66
<i>Figure 2-17</i> Endpoints are the vertices, the real segments are solid lines, and the virtual segments are dashed lines. (Wang, et al., 2005)	68
<i>Figure 2-18</i> Cartesian and polar representations (Stojmenovic, et al., 2007)	75
<i>Figure 3-1</i> Image preprocessing techniques: (a) Original SOHO/EIT image in 171 Ångstroms. (b) Image after speck removal. (c) Image after background subtraction. (d) Image after binarization and global thresholding.....	83
<i>Figure 3-2</i> Background subtraction. (a) Raw solar image region with a coronal loop. (b) Background image created using the rolling ball algorithm (Ball size = 5). (c) Subtracted image (Background image is subtracted from the original image)	84
<i>Figure 3-3</i> Extracting out-of-disk blocks.....	85
<i>Figure 3-4</i> One loop region spreads out over consecutive blocks	87
<i>Figure 3-5</i> A snapshot of the training label correction tool which is developed in JAVA. The red underlines indicate the “Loop” blocks while the gray underlines indicate “Non-Loop” blocks. (a) gray-level blocks after background extraction, (b) binary blocks after binarization	88
<i>Figure 3-6</i> Loop blocks: (a) and (c) Gray level blocks, (b) and (d) Binary blocks	89
<i>Figure 3-7</i> (a) The length of the longest line (b) The number of edge pixels	90
<i>Figure 3-8</i> Four bands in (a) a Non-Loop block (b) a Loop block	92
<i>Figure 3-9</i> (a) A loop block. (b) A near-positive non-loop block (c) A small sized imperfect loop (d) A medium sized half loop	93
<i>Figure 3-10</i> Curves shown in red in (b) and (d) embedded in the noisy regions of (a) and (c).	94
<i>Figure 3-11</i> Black points are the traced curve so far. <i>E</i> is the current point. The gray points with question marks are candidate points. The problem in curve tracing is to determine which point should be selected to obtain the correct curve?	95
<i>Figure 3-12</i> Directions relative to the center point	95

<i>Figure 3-13</i> Search tree for n candidate points	98
<i>Figure 3-14</i> Success of the further tracing approach	99
<i>Figure 3-15</i> Tracing the curves from the cluttered regions: (a, c, e, g) are the cluttered images, (b, d, f, h) are the automatically traced curves from these cluttered regions, respectively.....	100
<i>Figure 3-16</i> Example of an extracted curve	101
<i>Figure 3-17</i> Curvature strengths for some extracted curves	102
<i>Figure 3-18</i> Curvature strength feature for <i>Loop</i> blocks versus <i>Non-Loop</i> blocks	103
<i>Figure 3-19</i> Peak angle and curve distance measures for a curve.....	104
<i>Figure 3-20</i> Peak angles for different cases: (a) $\alpha = 180^\circ$, (b) $\alpha = 135^\circ$, (c) $\alpha = -15^\circ$	104
<i>Figure 3-21</i> ROC curves of all classifiers.....	107
<i>Figure 3-22</i> Misclassified blocks (a) False negatives, (b) False positives	108
<i>Figure 3-23</i> Information gain values for the investigated features	109
<i>Figure 3-24</i> Images from different cycles: (a) a minimum cycle image, (b) a maximum cycle image	110
<i>Figure 3-25</i> General structure of the block-based testing phase	112
<i>Figure 3-26</i> A snapshot of the developed loop mining tool interface with the red rectangle indicating a predicted loop region. Note that non-loop regions on the image are also eliminated correctly by the system.	113
<i>Figure 3-27</i> Test results on unseen data: (a, b) Images with loops. In (a), we can see one false positive region in the top-right red box. (c, d) Images without loops, correctly classified	114
<i>Figure 3-28</i> Sample snapshots of the online image retrieval tool: (a, b) images with loops, (c) image without any loops	116
<i>Figure 3-29</i> Out of disk loop region (a) Raw image, (b) After cleaning with IDL	116
<i>Figure 3-30</i> Image denoised with the Wavelet transform, note that the speckles are still present (in the circles).....	118
<i>Figure 3-31</i> Noise removal with the Wavelet transform: (a) Original image, (b) Denoised image with the Wavelet transform, note that the top part of the loop is lost	119
<i>Figure 3-32</i> (a) An image segment obtained after <i>eit_prep</i> with circled significant specks, (b) image after removing specks.....	119

<i>Figure 3-33</i> A zoomed loop segment (a) after removing outliers (b) after smoothing with Wavelet Transform.....	120
<i>Figure 3-34</i> Background subtraction algorithm: (a) original image, (b) background image, (c) background image in (b) subtracted from original image in (a).....	121
<i>Figure 3-35</i> Comparison of the binarization method used in this study to other standard methods. (a) original block, (b) after a standard morphological thinning operator, (c) after the Canny edge detector, (d) after the binarization method used in this study.....	122
<i>Figure 3-36</i> Second order directional derivatives of a loop block. Different derivatives highlight different oriented loop points.....	124
<i>Figure 3-37</i> Information gain of the explored features in this section	129
<i>Figure 3-38</i> Some false positives (non-loop blocks are classified as loop)	130
<i>Figure 3-39</i> (a) Original Image with to be extracted strip around the solar disk, (b) Extracted strip of $H = 110$. (c) Strip after background extraction. (d) Strip after binarization	132
<i>Figure 3-40</i> (a) a magnified portion from the strip generated outside the solar disk, (b) the same strip after binarization	133
<i>Figure 3-41</i> (a) Input region, (b) desired salient contour (c) possible undesired contour	134
<i>Figure 3-42</i> The search space is inside the green triangle for the red point.....	134
<i>Figure 3-43</i> Principal contours in (b, c, d) extracted from the region in (a), (b) Non-Loop Contour, (c, d) Loop Contours.....	137
<i>Figure 3-44</i> Comparison of our results to Steger's results (a) original image, (b) curve points according to Steger's algorithm, (c) extracted curves by Steger's algorithm, (d) edge points by our diagonal gradients based method, (e) extracted curves by our Principal Contour Extraction algorithm.....	138
<i>Figure 3-45</i> Starting points for different oriented curve structures. (a) left-bottom point, (b) top-right point, (c) top-left point, (d) bottom-left point.....	139
<i>Figure 3-46</i> On the left: Original image with differently oriented curves intersecting each other. On the right: Automatically extracted curves. Intersecting individual curves are shown with different colors.....	139
<i>Figure 3-47</i> The distance (h) between the curve point (P_i) and chord (L)	140

<i>Figure 3-48</i> The contour extraction algorithm is applied on a non-loop region. The linearity of the extracted contours is high. <i>Linearity</i> = 0.95 for the left contour in (b) and it is equal to 0.92 for the right contour...	142
<i>Figure 3-49</i> Dividing the curve into windows	143
<i>Figure 3-50</i> (a) Non-loop contour, (b) loop contour which is asymmetric and half complete.....	144
<i>Figure 3-51</i> Computing the bell shape existence of a loop region: (a) traced curve, (b) chord line and traced contour, (c) plot of the distance values between the traced contour and the chord, indicating a bell shape.	146
<i>Figure 3-52</i> (a) and (c) Non-Loop contours, (b) and (d) their distance plots respectively, showing the lack of the existence of a bell shape.....	147
<i>Figure 3-53</i> ROC curve for the Adaboost.....	149
<i>Figure 3-54</i> Testing architecture for contour-based approach	153
<i>Figure 3-55</i> (a) Strip with extracted principal contours, (b) Contours classified as “Loop”, (c) Mapping the Loop contours to the original image	155
<i>Figure 3-56</i> (a) Image Retrieval Tool that uses the contour-based model, (b) Automatically detected loops both inside and outside the solar disk.....	155
<i>Figure 4-1</i> Vectors between consecutive windows along the contour	159
<i>Figure 4-2</i> Segmenting a curve at a corner point with two different window sizes. Critical points are shown on the image with a red dot.	161
<i>Figure 4-3</i> Sample results of curve segmentation. Red dots show the location of the critical points that are detected.	163
<i>Figure 4-4</i> Sample contours ordered by their saliency measures. Saliency decreases from left to right in each column (a) contours with an arc shape, (b) linear contours	165
<i>Figure 4-5</i> The search space for each end is shown as a gray region	166
<i>Figure 4-6</i> We combine the contours from their closest endpoints. Dashed lines represent the four distances (d_1, d_2, d_3, d_4) among the end points of the two contours C_1 and C_2	167
<i>Figure 4-7</i> Arrows indicate the vector directions at the connection part of two contours	169
<i>Figure 4-8</i> Two different contour combinations and angle α between two vectors v_1 and v_2	169

Figure 4-9 (a) Two contours, (b) Fitted ellipse on the two contours, (c) Residual space of the error of fit. Notice that there is a spike at the connection point shown in a red circle.	170
Figure 4-10 (a) Two contours, (b) Fitted ellipse on the contours, (c) Residual space of the error of fit. Notice that the $T_{overall}$ statistics is not reflecting the difference on both sides of the connection point, which is shown in a red circle.	171
Figure 4-11 (a) Two contours, (b) Fitted ellipse for both contours, (c) Residual space of the error of fit. Notice that the $T_{avg-comp}$ and $T_{avg-overall}$ statistics are very low, since we compare only K values on the left side and the right side.	172
Figure 4-12 Test statistics and eccentricity value are given along with the contours, fitted ellipses, and the residual space of the fits.	176
Figure 4-13 Computed measures for this pair: $concavity=0.75$, $peakValue = 0.5$, $Bell-Existence = 0.72$, $pseudo-curvature = 0.15$ (a) Two contours and the chord between their end points. Note that points are on both sides of the chord, (b) Point-to-chord distance plot, $\mathbf{D}$, (c) absolute value of point-to-distance plot, $ \mathbf{D} $	178
Figure 4-14 Different combinations of contours along with the point-to-chord distance measures, (a, c, d) show both the distance and smoothed absolute distance plots, (b) shows only distance plot, (e) shows only the absolute distance plot, (f) shows both the absolute distance plot and gradient distance plot.	181
Figure 4-15 The role of proximity. (a) For C_1 , both C_2 and C_3 are good candidates. We favor the close contour C_2 over the farther contour C_3 . (b) If C_1 and C_2 form a group, then should we include C_3 into the group? C_3 is far away and its length is shorter than the minimum distance between C_3 and the group.	182
Figure 4-16 A synthetic image used for pair and group sample generation.	183
Figure 4-17 Generated contour pairs, (a) positive pairs, (b) negative pairs.	184
Figure 4-18 Generated contour groups, (a) positive groups, (b) negative groups.	185
Figure 4-19 A decision tree built for pair-wise instances.	188
Figure 4-20 A decision tree built for group-wise instances.	188
Figure 4-21 Information gain values for group-wise and pair-wise instances. Note that only the ellipticity value is positive among the pair-wise instances.	189

<i>Figure 4-22</i> Initial seeds for the contour grouping algorithm. Each group is represented by a different color	193
<i>Figure 4-23</i> Sample contour groups along their labels	196
<i>Figure 4-24</i> (a) A sample of a synthetic cluttered image, (b) Salient contour groups obtained with our technique	200
<i>Figure 4-25</i> (a, c, e) Cluttered solar image regions after Ridgelet transform (Inhester, et al., 2007) on STEREO/SECCHI images, (b, d, f) Sample outputs.....	201
<i>Figure 4-26</i> Contour grouping results on a TRACE image, (a) original image, (b) cleaned image, (c) salient contour groups before B-spline fitting, (d) salient contour groups after B-spline fitting	202
<i>Figure 4-27</i> Road detection in aerial images. The results of (Bacher, 2004) (a) original IRS image, (b) detected roads (Bacher, 2004).....	203
<i>Figure 4-28</i> Road detection results (a) the input of our algorithm, (b) the output of the algorithm.....	205

1 INTRODUCTION

“All men have stars, but they are not the same things for different people. For some, who are travelers, the stars are guides. For others they are no more than little lights in the sky. For others, who are scholars, they are problems...”

~ The Little Prince Antoine de Saint-Exupéry

The Sun, the source of our life, is a highly energetic star where several gigantic energy revealing events occur. Some events such as coronal mass ejections or the solar wind affect the Earth and might cause damage to grids or satellites. Several satellites have been deployed to closely monitor the solar events, to understand their dynamics, and to take precautions from possible damage on Earth and to orbiting satellites in space. *Figure 1-1* illustrates several satellites monitoring the Sun and how a coronal mass ejection affects the Earth.

Figure 1-1 The interaction between the Sun and Earth along with the designated satellites¹

¹ The Sun and Earth Connection, <http://edmall.gsfc.nasa.gov/99invest.Site/SUN-EARTH1/Sun1.html>

These satellites, which include the Solar Dynamic Observatory (SDO), Solar and Heliospheric Observatory (SOHO), Transition Region and Coronal Explorer (TRACE), and YOHKOH, have been taking pictures of the Sun regularly and storing the images in public databases². Among those, SOHO³, the oldest satellite, was launched in 1996. The instrument of Extreme ultraviolet Imaging Telescope (EIT) on SOHO has been taking images of the solar corona in the ultraviolet range. EIT is using four different wavelengths: 171, 195, 284, and 304 Ångstroms. Different solar events are more visible in different wavelengths. SOHO has collected more than 500.000 snapshots of the Sun over the years, which are stored in the SOHO online database². Thanks to the images of SOHO/EIT, several unknown facts about the solar corona were revealed and some misconceptions about it were cleared out. However, the resolution of SOHO/EIT was not sufficient to observe the fine details of solar events.

Figure 1-2 SOHO/EIT image on the left versus TRACE image on the right. SOHO/EIT captures the Sun globally while TRACE provides fine detail with high resolution images

² EIT: Extreme ultraviolet Imaging Telescope on board Solar and Heliospheric Observatory (SOHO): <http://umbra.nascom.nasa.gov/eit/>

³ Solar Heliospheric Observatory: <http://sohowww.nascom.nasa.gov/>

In 1998, the TRACE satellite was launched to capture high spatial (1 arc second) and temporal resolutions (1-5 seconds) of the upper solar atmosphere⁴. TRACE images allowed researchers to study the relations between magnetic fields and plasma structures. SOHO and TRACE are used complementarily by researchers. SOHO provides a global picture of the Sun to roughly monitor solar events in the low resolution image. In case of existence of interesting events, TRACE closes up into their regions and collects fine details about these events with high spatial and temporal resolution images. *Figure 1-2* demonstrates a picture of the Sun taken by SOHO/EIT on the left and a sub region shown in the box. That sub region was captured with TRACE and shown on the right side of Figure 1-2. Note that the magnetic fields are much more visible in the TRACE image.

With the increasing number of solar images taken by several satellites, solar image databases^{2, 5} have grown over the years and manual search for solar events has become impossible. This has motivated the need for automated detection of solar events from image databases. In this dissertation, we propose a methodology and several novel algorithms for the automated detection of coronal loops from solar databases.

1.1 Problem Description and Objectives

Coronal loops are immense arches of plasma that are confined by the magnetic field, anchored in the solar photosphere, and stretch up for tens or hundreds of thousands of kilometers into the atmosphere. They can reach temperatures of several million K and are visible at X-ray and EUV wavelengths. The plasma contained in these loops can be quiescent, flowing, or exploding. Coronal loops are the basic building blocks of the solar

⁴ Transition Region and Coronal Explorer (TRACE) : <http://trace.lmsal.com/>

⁵ TRACE Data Center : http://trace.lmsal.com/trace_cat.html

corona and have been linked to basic unanswered questions such as the flare trigger and the coronal heating problem. The population of coronal loops can be directly linked to the solar cycle.

Loops are ideal structures to observe to understand the transfer of energy from the solar body into the corona. *Figure 1-3* shows coronal loop regions on an image taken by SOHO/EIT and TRACE. Footpoints, which are visible in *Figure 1-3* (b), the two ends of a coronal loop and lie in regions of the photosphere, where sunspots are located.

Figure 1-3 Coronal loop regions (a) A coronal loop region outside of the Sun taken by SOHO/EIT² under 171 Ångstroms (b) A coronal loop region taken by TRACE⁴

Coronal loops have attracted considerable attention from scientists, studying various subjects including the Coronal Heating Problem (Schmelz, et al., 2003; Schmelz, et al., 2007) which is one of the longest standing unsolved mysteries in astrophysics. The Coronal Heating Problem is essentially concerned with understanding and modeling the exact properties of temperature distribution along coronal loops. In order to make progress, scientific analysis requires data observed by instruments such as SOHO/EIT and TRACE. The biggest obstacle to completing studies of the Coronal Heating Problem

has been putting the relevant data set together. Currently physicists are looking at each image in the database separately to decide whether an image contains desired coronal loops or not. This process is very time consuming, tedious, and open to human errors.

Problem1: Coronal Loop Detection

As image databases got larger, the manual search for coronal loops became more challenging. For example, team members of the TRACE instrument⁴ looked at every image at the beginning of the mission to find interesting regions and events (Handy, 1999). The coronal loops analyzed by Lenz *et al.* were found manually as well (Lenz, 1999). For the work described in (Schmelz, et al., 2007), a team of undergraduate students search for loop candidates manually in the TRACE database⁵. The difficulty with these manual searches has sparked interest in automated or semi-automated methods for the extraction of coronal loops. Various algorithms have been developed to trace curvilinear features in solar images (Aschwanden, 2005; Lee, et al., 2006; Biskri, et al., 2010; Inhester, et al., 2007). These automated methods apply some kind of objective criterion optimization for the detection of loops and the measurement of loop properties. Most of these algorithms, however, were tested and compared on images that were *already known* to contain loops (*e.g.*, (Aschwanden, et al., 2007)).

Objective 1: Automated Retrieval of Coronal Loops

Our *objective* is to take this analysis one important step further, by building an image retrieval system that can detect coronal loops automatically from large image data sets *without knowing about the loop presence*, even when most of the images do not contain any coronal loops. Our work paves the way toward automated solar feature detection on new missions like the Solar Dynamics Observatory (SDO), where the

Atmospheric Imaging Assembly (AIA) takes approximately 4800 images per hour (compared with about 100 images per hour for TRACE). Even though the number of images is going to increase with SDO, the number of coronal loops will remain the same. Therefore searching for coronal loops manually from the huge SDO image database is not going to be possible. Thus, a robust automated loop detection system is needed.

Problem 2: Extracting Salient Contour Groups from Clutter

Another challenging problem in coronal loop studies is highlighting or bringing out coronal loops from cluttered solar regions. Once high resolution solar images became available, extracting individual loop segments from images automatically became even more difficult. The reason was that more magnetic fields in different temperatures were captured in a single image. As a consequence, countless magnetic fields intersect each other, which makes curve tracing more challenging. Another difficulty arises when image cleaning techniques are applied on these high resolution images. Since the background is very busy, image cleaning techniques tend to retain more undesired patterns in the images.

Ideally, researchers want to trace loops from one foot point to the other to analyze their characteristics (Aschwanden, 2005; Schmelz, et al., 2007). However, coronal loops are surrounded by other solar events or intersect with other loops. Also, the intensity levels near the footpoints are strong, while the top parts of loops tend to be faint. Therefore, image cleaning techniques may erase the faint parts of the loops, which may cause gaps among loop segments. To highlight the loops, curve tracing methods can be applied on solar images (Lee, et al., 2006; Raghupathy, et al., 2004; Steger, 1998). However, curve tracing methods are not only easily affected by the presence of noise

around the loop points, but also cannot handle the gaps within the loop segments, and can easily follow wrong paths at junctions or wishbones. In one approach, Inhester *et al.* detect ridge points in solar images and then link ridge points based on their closeness and edge orientation (Inhester, et al., 2007). However, they highlight not only coronal loop structures but also other curvilinear structures in the images as shown in *Figure 1-4* (b).

Figure 1-4 (a) Original image taken by STEREO/SECCHI⁶, (b) Corresponding image after applying the ridge detection method in (Inhester, et al., 2007), (c) Ideal coronal loop segments are extracted manually from the clutter.

After obtaining the curvilinear structures via the Ridgelet transform, they select loop segments and eliminate non-loop segments manually. Then they group the related loop segments once again manually to obtain the ideal results shown in *Figure 1-4* (c). Manual loop extraction is not only time consuming and tedious but also open to human errors due to clutter.

⁶ STEREO: http://stereo-ssc.nascom.nasa.gov/beacon/beacon_secchi.shtml

Figure 1-5 (a) TRACE image taken on May 19, 1998, at a wavelength of 173 Ångstroms, (b) Steger curve tracing algorithm is applied on the image (Aschwanden, et al., 2007). Note that top parts of the loops disappear.

Yet another example of fragmented segments of loops could be observed on the results of curve tracing based methods (Steger, 1998; Lee, et al., 2006). Steger's method is applied on solar images to highlight the curvilinear points and then trace them (Aschwanden, et al., 2007). However, Steger's method is sensitive to the threshold value used in curvilinear point selection. High thresholds remove the clutter points but also remove the top faint parts of the loops and cause fragmented loops as shown in *Figure 1-5 (b)*, whereas low thresholds keep more curvilinear points, which increases clutter in the results.

In addition to solar images, different applications may suffer from fragmented segments in clutter. In real life images, a single curve in an image could be broken into pieces due to many reasons such as poor image capture, image cleaning, subtle transition between foreground and background regions, etc. The human eye has the ability to perceive smooth curves from cluttered regions and complete the gaps within coherent segments easily whereas automatic techniques cannot achieve the same results as quickly and as accurately. The human visual system groups elements into meaningful or coherent

clusters using perceptual rules which are known as *Gestalt laws of perceptual organization* (Koffka, 1935). The most commonly used rules are proximity, similarity, continuity, and closure. In addition to these, co-linearity and co-curvilinearity are also used in contour grouping (Zhu, 1999).

Separating salient contours from clutter and grouping the related contours is needed in many real life applications such as object boundary detection in natural scenes (Felzenszwalb, et al., 2006; Wang, et al., 2005; Ullman, et al., 1988), road and mountain crest detection in satellite images (Alquier, et al., 1996; Wang, 2007; Bacher, 2004; Steger, et al., 1999), and blood vessel extraction from medical images. *Figure 1-6* illustrates a result of road extraction in an agricultural area (Bacher, 2004). Due to the similarity between the features of agricultural areas and the road, the roads are not extracted correctly and several gaps among roads occur. Road extraction studies (Bacher, 2004; Steger, et al., 1999) also face the problem of clutter in their results. In particular, urban regions may generate more cluttered regions, which make road extraction challenging as demonstrated in *Figure 1-7 (b)*.

Figure 1-6 The result of road extraction on aerial image in an agricultural area in IRS data⁷. Note that there are gaps between related road segments (Bacher, 2004)

⁷ IRS satellite: <http://www.nrsc.gov.in/satellites.html>

Figure 1-7 Urban regions may generate clutter in Steger's road detection algorithm (Steger, et al., 1999). (a) original image, (b) Extracted roads which contain heavy clutter in the urban region

Objective 2: Salient Contour Group Extraction from Cluttered Images

In this dissertation, another *objective* is to extract salient contour groups from cluttered images accurately and quickly. To reach this objective, we propose a contour grouping method using perceptual Gestalt rules and Markov Random Fields. The automated salient contour group extraction method alleviates the manual process of clutter elimination and speeds the process toward ideal results. Our method does not target only coronal loop extraction, but can be used for other applications in need of salient contour group extraction from clutter.

1.2 Challenges

We analyze the challenges that we have faced during our studies into two sections: challenges of coronal loop detection and challenges of salient contour group extraction.

1.2.1 Challenges of Coronal Loop Detection

To recognize the coronal loop regions automatically, first we download images from the SOHO/EIT database to use them in the training phase. Experts then label the coronal loop regions on the downloaded images by marking the location within a bounding rectangle. From the labeled images, we build training models to learn the characteristics of coronal loops and distinguish them from the rest of the events. The most challenging aspects of coronal loop detection are listed below with an explanation.

Finding the most appropriate image cleaning sequence: The image preparation phase is very critical to achieve a high accuracy from classifiers. If we bring out the coronal loops from their surroundings clearly and suppress the other solar events as much as possible, we can improve the classification results. However, this aim is hard to achieve due to the nature of solar images and coronal loops. Most of the time, the intensity level along the entire loop varies significantly. The loops might be embedded into bright regions. Moreover, coronal loops and other solar events may coincide in the same region, as shown in *Figure 1-8 (e)*. Some of the loop shapes are so vague that after applying cleaning techniques, low intensity valued portions may disappear or there may be nothing left from the loop shape, if the loop is vague as in *Figure 1-8 (b)*. Also, the sequence of the performed techniques plays an important role in the results. The wrong order might lead to undesired results. Because of these problems, applying appropriate image cleaning techniques on the images is very critical. Some cleaning techniques may cause data loss from the coronal loop parts, whereas other techniques may retain or enhance the undesired solar events. Our wish is to keep as many of the coronal loop points as possible while getting rid of other forms from the images.

Figure 1-8 Expert marked loops. (a) A big loop (b) A vague loop (c) A noisy loop (d) A small loop (e) A loop interfering with another solar event (prominence)

Finding the most appropriate feature set: Selecting the right features to represent the patterns to be learned is at the core of automatic detection systems. Considering the nature of the patterns and the scenes, the most matching features should be investigated to achieve high accuracy from the classifiers. For our case, finding common features to represent all kinds of coronal loops was another challenge. When we analyzed the marked coronal loops in the training set, we observed that each coronal loop has unique characteristics, and thus finding common features for all of them is difficult. Their sizes and orientations vary from one loop to another, as shown in Figure 1-8 (a), (d). Even though their shape generally resembles an arch, we see different variations of arches in each loop, for example, they might be asymmetric semi-elliptic shapes. Therefore, performing well known ellipse detection methods (McLaughlin, 1998; Tsuji, et al., 1978; Duda, et al., 1972; Donoho, et al., 2001) is not a solution.

Distinguishing coronal loops from other solar events: Coronal loops are not the only events occurring on the solar corona, there are other kinds of activities or events, such as solar flares, prominences, or filaments that are hard to distinguish from coronal loops sometimes even for the human eye. Examples of image regions without any coronal loops but containing other solar events, are shown in Figure 1-9. These solar events

might show similar characteristics to coronal loops and cause a decrease in the accuracy of classifiers. They might cause high false alarms. We wish to reduce false alarms as much as possible and obtain coronal loops with high recall.

Figure 1-9 Regions that have no loops, but contain other activities that hard to distinguish from loops

1.2.2 Challenges of salient contour group extraction from clutter

To extract salient contour groups, first we divide the image into a set of smooth discrete contours and sort the contours according to their saliency measure. After that, we group the related discrete contours to obtain salient contour groups and eliminate background contours. The most challenging aspects of salient contour grouping are listed below.

Extracting individual contours: The first step of our approach is representing the image with a set of contours. Since the images are cluttered, we have to be very careful during individual contour extraction. The success of the results depends on the clarity of discrete contours. Final contours should be free of corners, jaggedness, and squiggle. In the existence of junctions, wishbones or intersecting curves as shown in *Figure 1-10*, it is hard to decide from where to cut the curves into pieces.

Figure 1-10 Existence of squiggles, intersecting curves, and wishbones make individual contour extraction challenging

Defining a saliency measure for discrete contours: The saliency measure of a contour represents a measure of how much a contour pops-out from the background and captures attention in the scene. Saliency depends on several factors, including smoothness, co-linearity, proximity, closure, and curvature consistency (Ullman, et al., 1988; Wang, et al., 2005). Different applications may need different definitions for saliency measures. For instance, object boundary detection favors closure and smoothness (Felzenszwalb, et al., 2006; Ullman, et al., 1988; Wang, et al., 2005). Defining saliency for individual contours might require prior information about the application. To obtain the optimal results from contour grouping accurately and quickly, the saliency measure should be defined carefully.

Choice of perceptual rules: During grouping, one or more of the perceptual rules might be at work in determining the perceived group. If there is more than one perceptual rule in the image scene, then those rules might be cooperating or competing. Choosing the appropriate rules and adjusting their weights is also critical in salient contour grouping. *Figure 1-11* illustrates an example of a challenging condition. Suppose that we are looking for grouping options for the red segment. The global optimum salient contour group consists of the red, orange and yellow contours. The candidate list of the red segment consists of the blue, pink, orange, and yellow segments. Favoring different

criteria (co-linearity, good continuation, proximity, co-circularity, co-elliptic, length) might yield different results. For example, the good continuation criterion could be indecisive between the orange and green contours. For the proximity criterion, the close choices would be between blue and green contours. On the other hand, for the elliptical or circular criteria, the decision would be between the blue and pink contours. In the cluttered region, the length of grouped contours could be misleading, too. Therefore, the weights of different criteria should be adjusted carefully to reach the desired optimal solution.

Figure 1-11 The global salient group consists of {red, orange, yellow} contours. The candidate list for the red segment is {green, blue, pink, orange}. The selection of the contour to be grouped is critical.

Finding the global optimum: The global optimum varies depending on the application. In our case, the longest and smoothest semi-elliptical curve groups are the optimal solution. There might be more than one contour group in an image. We are supposed to find all the groups. The biggest challenge of salient contour grouping in cluttered regions is getting stuck in local optima easily and thus missing the global optimum. When there is more than one good candidate in the search space for a contour, the algorithm might select the candidate giving lower cost but that might cause eliminating the right candidate yielding the global optimum. For example, using a greedy

search in the candidate selection process tends to cause erroneous results (Felzenszwalb, et al., 2006). Hence, the optimization part of the grouping algorithm should handle local minima problems.

Size of the search space: Varying gaps between contours make it difficult to choose the size of the search space during the candidate selection stage. When the search space gets bigger, the time complexity of the algorithm increases particularly with the existence of severe clutter in the image. A small search space might not group two related contours when they are far away from each other. The size of the search space should allow the related distant segments to be grouped and avoid unrelated candidates in order to maintain a low complexity.

1.3 Contributions

The contributions of this research can be divided into two parts: 1) a coronal loop detection system, 2) salient contour group extraction from clutter.

1.3.1 Coronal Loop Detection

1. *Building and validating a solar loop detection system:* In order to retrieve images with coronal loops from large image data sets, we have developed an image retrieval system. To the best of our knowledge, there is no automated retrieval system for solar images containing coronal loops from online solar image data sets. In (Lee, et al., 2006); (Aschwanden, et al., 2007); (Inhester, et al., 2007), coronal loops were traced in *predetermined* regions that were *known* to contain coronal loops, rather than detected automatically without knowing their presence as in our case. Their aim was to develop curve tracing algorithms to highlight the loop structures in given sub-regions of solar

images. In our problem which is different, we do not even know whether an image has any loops. In fact, this knowledge is exactly our goal, since we desire to retrieve only images having coronal loops. There have been several efforts to recognize other kinds of solar events, such as sun spots, filaments, plages, coronal mass ejections, and solar flare analysis (e.g. (Zharkova, et al., 2005); (Hill, et al., 2001), (Colak, et al., 2010) (Colak, et al., 2010), (Colak, et al., 2005)), all of which share no structural characteristics with coronal loops.

We developed two approaches to solve the coronal loop detection: a block-based approach, (Durak, et al., 2007; Durak, et al., 2008; Durak, et al., 2009) and a contour-based approach (Durak, et al., 2010; Durak, et al., 2010). In the block-based approach (as shown in *Figure 1-12*), we divide the solar images into fixed size blocks and label the blocks as “*Loop*” and “*Non-Loop*” according to the existence of a loop in a block. Then we extract block-based features and then train classifier models with the extracted block-based features. We achieved 65% precision and 67% recall from the best feature set.

Figure 1-12 General structure of the block-based coronal loop mining approach

One drawback of the block-based approach is that the “*Non-Loop*” blocks with solar activities were causing high false alarms and decreasing the accuracy. To overcome

this problem, we decided to work with individual contours rather than blocks in the contour-based approach (Durak, et al., 2010). In this approach, we first extract a strip around the solar disk and extract contours directly from this strip. As a result of eliminating the block extraction step, our image retrieval system was sped up significantly. Then the experts label the extracted contours. We investigated several shape features for the labeled contours and train the classifiers. An Adaboost classifier based on C4.5 decision tree was able to achieve 85% precision and 83% recall with the contour-based approach. The general architecture of the contour-based approach is shown in Figure 1-13.

Figure 1-13 General structure of the contour-based approach

2. Image Cleaning Sequence: Considering the sensitivity of coronal loops, we investigated several techniques to bring out coronal loops while suppressing other solar events (two conflicting goals). First, we perform an image cleaning technique with the IDL solar software (ssw) (Handy, 1998) to clean images from instrument defects. Later we apply speck removal and smoothing techniques. After that, we perform background extraction using the Wavelet transform. We also propose a binarization scheme to reduce all of the flux tubes into one-pixel width lines without changing the essential structure of

the flux tubes. The resulting binarization method brings out the general structure of the forms without causing any change in the original shape.

3. *Principal Contour Extraction:* We desire to accurately extract each coronal loop as an entire contour. To overcome the aforementioned challenges, we introduced a principal contour extraction method that extracts the desired principal contours in cluttered regions (Durak, et al., 2010). Our algorithm deals with the discontinuity problem in noisy environments. Previous curve tracing algorithms (Raghupathy, et al., 2004; Lee, et al., 2006; Steger, 1998; Sargin, et al., 2007; Cheng, et al., 2004) did not address the difficulties of curve tracing in noisy regions in the presence of discontinuity problems. We tested our algorithm on coronal loops embedded in cluttered regions and succeeded to extract coronal loop contours with 85% accuracy.

4. *Designing special features:* We investigated standard features such as statistical features (Gonzalez, 2007), histogram of gradients (Dalal, et al., 2005), and edge histograms (Won, et al., 2002). However, they could not yield a satisfactory classifier model accuracy. Therefore, we designed specific features to characterize the coronal loops better. We enhance the angle range in Edge Histogram Descriptors (EHD) by using the Hough transform. According to the spatial distribution of loops in the blocks, we proposed spatial edge histograms. Since coronal loops are curvilinear, we also propose curvature based features from curve structures obtained from a specialized curve tracing algorithm that we developed for this study. In the block-based approach, we investigated the histogram of second order derivatives, curvature histograms, Hough-based features, and eigenvalue histograms of the Hessian matrix. In the contour-based approach, we investigated linearity; elliptical features such as eccentricity, minor axis over major axis

ratio, etc.; chord-to-point distance features such as pseudo-curvature, bell-existence, arch height; perceptual features such as smoothness, proximity, and corner points.

1.3.2 Salient Contour Grouping

The general architecture of the extraction of salient contour groups is shown in *Figure 1-14*. We have two main sections in this part: contour extraction and contour grouping.

Figure 1-14 Architecture of the extraction of of salient contour groups

Our contributions are as follows:

1. **Dividing the curves into contours:** In this stage, we first perform the curve tracing method that we proposed in (Durak, et al., 2010). With this method, we obtain curves in length. These curves might contain subtle corner points or squiggles. We divide the curves into sub-segments using the curvature distribution along the curve that helps to obtain corner points and subtle transitions.

2. Defining a saliency measure: Considering the requirements of the coronal loop highlighting problem, we define a saliency measure based on linearity and length. We assign high saliency values to long semi-elliptical arcs, while low values to short straight lines. Previous saliency measures are based on curvature consistency (Murphy, et al., 2003) or smoothness (Ullman, et al., 1988; Wang, 2007).

3. Defining a measure for the goodness of ellipse fitting: In our study, we want to know whether combined contours lie on the same ellipse. Previous studies (Rosin, 1996) calculate the error of fit for ellipses but they do not provide a measure for the goodness of ellipse fitting. After direct least squares fitting of an ellipse (Fitzgibbon, et al., 1999), we calculate a gradient weighted algebraic distance for each point in the contour group and generate the residual space. We measure the goodness of fit by analyzing the statistical features of this residual space.

4. Defining measures based on point-to-chord distance: We investigate the shape of the signed point-to-chord distance plot to check whether the combined contours form an arc or irregular forms. Using point-to-chord distance plot is more appropriate for open curves. Most studies (Roussillon, et al., 2010; Nguyen, et al., 2010) propose arc measures for closed curves, whereas we define the arc existence measure for open curves.

5. Weight estimation of perceptual rules: Most algorithms (Felzenszwalb, et al., 2006) only consider smoothness as a measure when they extract salient curves. However, this is not sufficient in the existence of heavy clutter. We use smoothness, ellipticity, proximity, concavity, and circularity. To overcome the challenge of combining perceptual rules, we train a multiperceptron classifier with positive and negative contour

combinations. We use the estimated weights by Multiperceptron in the cost of function of the optimization phase of contour grouping.

6. *Salient contour group extraction from cluttered region*: Previous studies related to salient contour grouping (Felzenszwalb, et al., 2006; Wang, et al., 2005; Ullman, et al., 1988) have concentrated on the object detection problem from image scenes. There is no salient contour grouping study specializing in open curves in cluttered regions. In our study, we propose a method for extracting salient open curves from cluttered regions. To obtain salient contour groups, we propose the contour grouping technique based on Markov Random Fields (MRF) and perceptual rules (Durak, et al., 2011). One other difference from previous studies (Murino, et al., 1996; Schlüter, 1997; Felzenszwalb, et al., 2006) is that we group a contour with at most one other contour in each end. This constraint helps us avoid wishbone structures and obtain smooth semi-elliptical curves.

We tested our method on synthetic data which has heavy clutter, as well as for coronal loop highlighting from real solar images, and road detection in aerial images (to illustrate applicability to other applications) and successfully acquired salient contour groups.

1.3.3 Image Specifications

In this dissertation, we have used four different kinds of data. We list the details of the data in **Table 1-1**.

Table 1-1 Data specifications used in the dissertation

Data Name	Data Source	Image Specifications
SOHO/EIT	http://umbra.nascom.nasa.gov/eit/	FITS images 171 Angstroms Size = 1024 x 1024 Resolution: 8 bits per pixel Gray-level
TRACE	http://trace.lmsal.com/trace_cat.html	FITS images 171 Ångstroms Size = 1024x1024 Resolution: 24 bits per pixel Gray-level
STEREO/SECCHI	http://stereo-ssc.nascom.nasa.gov/beacon/beacon_secchi.shtml	171 Ångstroms Size = 1024x1024 Gray-level
IRS	http://www.nrsc.gov.in/	5.8 meter spatial resolution 256 gray-levels

1.4 Organization of this Dissertation

The rest of this dissertation is organized as follows. The background and related work about coronal loop detection and salient contour group extraction methods are presented in Chapter 2. Coronal Loop Detection from the SOHO/EIT image collection is described in Chapter 3, where we describe both a block based approach and a contour based approach. The salient contour group extraction method is presented in Chapter 4. Finally, our conclusion and future directions are given in Chapter 5.

2 BACKGROUND and RELATED STUDIES

“I do not know what I may appear to the world, but to myself I seem to have been only a boy playing on the sea-shore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me.”

~ Isaac Newton

This dissertation addresses two different problems: coronal loop detection from solar images and salient contour grouping from cluttered images. Within the scope of the first problem, we examine the previous coronal loop detection studies along with other solar event detection systems in Section 2.1. To extract coronal loops, we analyze the features of the curves. In Section 2.2, we review curve tracing algorithms, curve extraction, and curve segmentation methods. An essential component in any pattern recognition problem is feature extraction. Thus, we examine the features related to coronal loop detection in Section 2.3. To train classifier models, we have investigated several classifier techniques, which are described in Section 2.4. Since coronal loops could be semi-elliptic, we review ellipse detection methods under ellipse fitting and the Hough transform in Section 2.5. Within the scope of the second problem, we examine contour grouping studies along with perceptual organization, saliency detection, grouping methods and grouping measures in Section 2.6.

2.1 Automatic Detection of Solar Events

Coronal loop detection has been studied from different aspects (Lee, et al., 2006; Aschwanden, et al., 2007; Biskri, et al., 2010; Inhester, et al., 2007). The aim of these studies has been to highlight loop structures from the given solar image regions. One drawback of these studies is that they test their algorithms on only one image which is a TRACE image from May 1998 and not on different or recent images to challenge the difficulties of broad-spectrum loop detection problem. Thus, they do not validate their algorithms on sets of images.

Lee *et al.* segment coronal loops from solar images by estimating the magnetic fields using the local orientations (Lee, et al., 2006). They first preprocess the image to remove non-loop pixels. For preprocessing, they perform median filtering, unsharp masking, global thresholding (eliminating the pixel under the median value of the intensity image), and local thresholding (dividing the image into 31x31 regions and eliminating the points with intensity below the median value of the region). To label the pixels into loop and non-loop, they apply Strous's loop labeling algorithm. This algorithm compares the intensity level of a point (x, y) with its four directional neighbors which are horizontal neighbors, vertical neighbors, and diagonal neighbors. They examine whether the intensity of a point is higher than its neighbor pixels in each direction. If the intensity level of the point is higher than at least two different directions, then they label this point as a loop point. After labeling the pixels, they join disconnected loop pixels to form complete loops. They start from any loop pixel to form a coronal loop and add one pixel at a time to the current loop structure. To find the best continuation point, they look for the points within the search region around a given point. The search

region has a fan shape bounded by minimum and maximum angular directions. The best pixel in the region is the one that best preserves the loop continuity in the position and tangent direction. After linking the loop points, they apply post-processing to reduce the jaggedness and connect disconnected loop segments. They perform B-Spline fitting to connect the disconnected segments smoothly. For the B-Spline fitting, they provide a number of control points according to the length of the coronal loops. B-spline bends the curves from the control points. Each control point is associated to a basis function (Cham, et al., 1999).

Instead of Strous's loop detection method, Inhester *et al.* detect ridge points along with their orientation using Taylor coefficients (Inhester, et al., 2007). Then they connect ridge points and smooth connected points via polynomial fits. Finally, they connect related loop segments and eliminate non-loop structures with the help of a semi-automated procedure. Another method by Biskri and Inhester uses a 2D Morlet continuous Wavelet transform to detect loop points instead of the Ridgelet transform (Biskri, et al., 2010). Next, the image is segmented to produce thinner loop traces, followed by thresholding to eliminate falsely labeled loop points and thus to obtain clear loops.

Ashwanden *et al.* compare existing algorithms developed for tracing curvilinear features in solar images in terms of detected length of the loop and the completeness of the loop (Ashwanden, et al., 2007). They first detect the coronal loops in an image manually for ground truth. After that they apply Lee's method (Lee, et al., 2006), Inhester's method (Inhester, et al., 2007), and Steger's methods (Steger, 1998) on the original image separately. The results confirm the following limitations of automatic loop

highlighting method: (i) the top parts of coronal loops are untraceable since emission measure drops below the noise threshold, (ii) the footpoints may not be visible due to temperature drops towards the transition regions, and (iii) the complexity of the background disrupt loop tracing.

In addition to coronal loop detection, other solar features (prominences, filaments, sunspots, and active regions) have been detected automatically. Prominences are cool and dense gas on the solar atmosphere. Prominences are observed above the solar limb as in *Figure 2-1 (a)*, while the same physical structures observed on the solar disk are named filaments as shown in *Figure 2-1(b)*. Fu *et al.* develop a method to detect prominences on the solar limb from consecutive image frames (Fu, 2007). They learn the characteristics of the prominences from the training data. They first apply polar transformation on the surrounding region of the solar disk. Then they apply a linear diffusion filter on the angular image to obtain the contrast image which brings out the prominences out of bright regions. Later they perform thresholding on the contrast image to remove noisy points from the image. They extract features and measure following properties of prominences: time span, position angle, angular width, radial height and brightness. With the extracted features, they trained a Support Vector Machine (SVM) classifier with the labeled limb objects and achieved 93% accuracy with the leave-one-out validation strategy.

Figure 2-1 Full disk H α images (a) Prominences are on the solar limb, (b) Filaments are on the solar disk

Filaments are similar to prominence structures (dense and cool plasma) except that they appear on the solar disk. Filaments are in the shape of a twisted flux magnetic rope. They look darker than their surrounding and have elongated fibril shapes. Bernasconi *et al.* first delete sunspots from the images and then perform thresholding on images to keep the pixels with the same level of average intensity level (Bernasconi, 2005). After that, they calculate the following properties of the remaining clusters: position, length, area, average tilt of axis with respect to the Sun's equator, and chirality of the magnetic flux rope. This system does not offer any learning method. Colak *et al.* also propose automatic filament detection based on thresholding and segmentation (Colak, et al., 2005). First they threshold the image to find the filament candidates, and then they perform a region growing algorithm to detect solar events.

Figure 2-2 Sunspot evolution over three days. The shape and the size of sunspots change in time. Images from TRACE: <http://trace.lmsal.com/>

Another interesting solar event is the sunspot which is a region on the Sun's surface that is marked by a lower temperature than its surroundings and has intense magnetic activity. Because of their lower temperature, sunspots look like dark compact features on the quiet Sun background and are visible from Earth without the aid of a telescope. Sunspots change in size and shape, and usually last about 30 days, but some can last much longer or shorter than the others. Their shape and size evolve during their life span. *Figure 2-2* shows a sunspot evolution over three days. Notice that the size, position, and shape of the sunspot changes over time.

Zharkova *et al.* extract and index spots from image sequences (Zharkova, et al., 2005). The spatio-temporal behavior of each object is captured by their intensity and size in a time series. Each time series captures the entire life cycle of a sunspot, throughout its evolution. They first segment each image into spots and then track these spots over the sequence of images. To detect sunspots from an image, they segment the image into regions using a region growing method. Region growing starts from one or more pixels, then incorporates neighboring pixels into regions according to certain homogeneity criteria, and terminates when a specific termination criterion is met. In (Zharkova, et al., 2005), the similarity between intensity levels is used as a homogeneity criterion and if it

is higher than a specified low threshold then growing is terminated. After segmenting the image into small regions, regions having similar brightness value are merged. A two-level segmentation is used to decrease over-segmentation. After segmentation, regions are labeled as 'dark' or 'bright' spots based on their brightness value. If the brightness value of the region is over an empirical threshold, then they label this region as 'dark'. Once they extract dark spots from the images, they assemble dark objects into a time-series object to follow changes in the object's positions. If there is overlap between two dark objects' areas in the current image and the consecutive image, then the two objects are assumed to be the same object and the time and position information are kept into a time series object.

Turmon *et al.* also consider the temporal characteristics of sunspots in a three phase: identification, tracking, and trajectory analysis (Turmon, et al., 2002). In the identification phase, they detect objects in consecutive images using classification techniques. They train the system using a combination of expert-provided labels and unlabeled data for classifying image regions as sunspots. In the tracking phase, they associate current objects to past objects to optimize the total overlap. They compute overlap in the object's position between previous and current images and associate the current objects with the past objects. In the trajectory analysis phase, they aim at learning objects by modeling their trajectories through a Hidden Markov Model (HMM).

Another kind of solar events are "active regions" which are regions with a strong magnetic field. Sunspots generally form within active regions that may last for several weeks or even several months. Active regions have been identified automatically using thresholding and region growing algorithm in several studies (Zharkova, et al., 2005).

These studies perform thresholding to separate background from foreground and then perform a region growing algorithm to obtain bright regions. They compare the detected regions to manually generated synoptic maps to validate their results.

2.2 Curve Processing

Different stages of our methodology have required different curve processing techniques. To separate coronal loops from the noisy background, we resort to curve tracing methods. For extracting salient contours, we strive to represent the image with a set of smooth curves. Therefore, we divide the long curves into atomic contours through curve segmentation. This subsection reviews existing curve tracing, curve extraction, and curve segmentation techniques.

2.2.1 Curve Tracing

Curve tracing aims at obtaining each individual curve from an image (Raghupathy, et al., 2004; Lee, et al., 2006; Steger, 1998; Sargin, et al., 2007; Cheng, et al., 2004). It generally starts from a given starting point and follows a curve even if it crosses other curves. Curve tracing is needed in many image applications such as in medical images, aerial images, and so on. In aerial images, curve tracing can be used to detect roads, rivers, and railroads. In medical imaging, curve tracing can be used to detect blood vessels.

Most curve tracing methods (Steger, 1998), (Raghupathy, et al., 2004) consist of two phases: first highlighting the curve points, and then linking them. The success of curve tracing depends on both steps. If the first step causes data loss at the curve edges,

then the curves cannot be extracted correctly. If the curve points are extracted correctly, then they should be connected appropriately to extract the desired curves.

One of the breakthroughs in curve tracing was Steger's algorithm that consists of two stages: classifying curve points and linking the curve points (Steger, 1998). First a Gaussian kernel is convolved with the image to decrease the amount of noise in the image. Then curve points are classified by calculating first and second derivatives of the image. At a curve point, the first directional derivative should vanish and the second derivative should be large in absolute value. A pixel (x, y) has a boundary defined by the unit square $[x-1/2, x+1/2] \times [y-1/2, y+1/2]$. Let the direction perpendicular to the curve be $n(t)$ where t is given in Eq. (2-1). A pixel in the image is classified as a curve point if the first derivative of the intensity level along $n(t)$ vanishes within a square centered around the pixel. Calculating the direction of a point is solved with respect to the Hessian matrix.

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix}$$

The partial derivatives I_{xx} , I_{xy} , I_{yx} , and I_{yy} are computed using partial differences after convolving the image with a Gaussian smoothing kernel. The direction perpendicular to the curve $n(t)$ can be computed by finding the eigenvector corresponding to the maximum absolute eigenvalue of the Hessian matrix. Let this eigenvector be (n_x, n_y) . A quadratic polynomial is then used to determine whether the first directional derivative along the curve vanishes at the current pixel. Let (p_x, p_y) be the quadratic polynomial of eigenvector at point (x, y) .

$$(p_x, p_y) = (t \cdot n_x, t \cdot n_y) \quad 2-1$$

$$\text{where, } t = -\frac{n_x l_x + n_y l_y}{n_x^2 l_{xx} + 2n_x n_y l_{yy} + n_y^2 l_{yy}}$$

The point (x, y) is classified as a curve point if $(p_x, p_y) \in \left[-\frac{1}{2}, \frac{1}{2}\right] \times \left[-\frac{1}{2}, \frac{1}{2}\right]$ where (p_x, p_y) is defined in Eq. (2-1). The maximum eigenvalue is used as a measure of strength of the curve point. After classifying curve points, points are linked starting from the pixel with the maximum strength. Curves are constructed by adding the appropriate neighbor points to the current curve point. For this purpose, three neighboring pixels in the direction of the point are examined. Considering the angle difference and the distance values, the next curve point is added to the curve structure. The linking terminates when there are no more curve points in the neighborhood of the current pixel.

Steger's curve tracing algorithm has a few major disadvantages. First, the linking procedure searches only three points in the neighborhood of the last point added to the curve, and does not consider points which are part of the curve but are not in the immediate search space. Another drawback is that the algorithm is very sensitive to the Gaussian blurring parameter, σ , which easily causes data loss at curve points. The algorithm also needs a global threshold value to eliminate pixels with low intensity values and high thresholds cause additional data loss. Thus, method considers the local gradient values but misses the global picture. *Figure 2-3* illustrates a sample output of Steger algorithm on a brain image and the portion missed by the algorithm.

Figure 2-3 Steger's method (Steger, 1998) applied on a brain image. The curve region in the red circle is missed.

Raghupathy *et al.* propose some amendments to the linking procedure of Steger's method (Raghupathy, et al., 2004). Their curve point extraction method is exactly the same as Steger's method. Their aim is reducing the mistakenly traced curves at the junctions, which is another drawback of Steger's algorithm. As a remedy to this problem, Raghupathy *et al.* search for more appropriate points in the orientation of the last portion of the traced curve. They avoid linking of two points, if there is a big change in the angle from one point to another point. Hence, they follow the right path at the junctions. They also desire to solve the problem of the gaps among the related curve segments. When there is no point in the immediate neighborhood of the last added point of the traced curve, they search for the points having the same orientation as the final traced part within a further distance.

Raghupathy's method also has some problems. Steger starts to trace curves from the point with maximum strength (Steger, 1998), while Raghupathy picks the starting point manually for each curve (Raghupathy, et al., 2004). Manual starting point selection is not feasible. Raghupathy's approach for tracing the correct curve at the junctions does

not guarantee the correct curves, since they check the possibility of a strong angle change difference at the pixel level. With this micro level checking, it is hard to catch the changes at the macro level. For example in *Figure 2-4*, suppose that Raghupathy's curve tracing algorithm is at the red point and looking for the best point to continue. The correct point is the yellow one at the macro-level. Raghupathy's algorithm will however select the blue point at the blue circle and trace the wrong curve. Even though there is a big change of orientation between the longer portions of the connection part. Thus, the algorithm will not pick the correct route which is further away. To overcome such wrong selections at the micro-level, we thus concentrate on macro-level curve grouping.

Figure 2-4 The red point is the last point of the traced curve. The segment starting with yellow dot is supposed to be selected. Since the curve structure starting with the blue point is close to the last point and in the range of the search space, the wrong curve will be traced by Raghupathy's algorithm (Raghupathy, et al., 2004).

Another curve tracing algorithm considers alternative ways to trace curves (Sargin, et al., 2007). They extract possible traces in k candidate directions satisfying a given threshold in the second derivative along the local direction of a given pixel and select the best trace with the least distortion (Sargin, et al., 2007).

2.2.2 Curve Extraction

Curve extraction and curve tracing terms can be confusing. In contrast to curve tracing, curve extraction methods highlight the possible curve points from the image

without distinguishing between the individual curves. Hence, the linkage part is not addressed in curve extraction methods. This point differentiates curve extraction methods from curve tracing methods. The wavelet, ridgelet, curvelet, or beamlet transform can be used for curve extraction (Inhester, et al., 2007; Biskri, et al., 2010). These transformations are used to extract directional details from the image. The Wavelet transform breaks the signal into scaled windows and represents each signal in terms of wavelet signals. Wavelet analysis uses long time intervals where more precise low-frequency information is needed, and shorter regions where high-frequency information is needed (Mallat, 1989). The wavelet transform extracts directional details and captures horizontal, vertical and diagonal activity. These three directions might not be enough in noisy images.

An extension of the Wavelet Transform is the Ridgelet transform which provides multi-resolution texture information (Semler, 2006). It is effective in detecting linear radial directions in the frequency domain. The Ridgelet transform is optimal to find lines in the image. To detect line segments, the image is decomposed into blocks, and the Ridgelet transform is applied on each block. The Ridgelet transform is used in the curve point detection phase in (Inhester, et al., 2007), which was the first step of their coronal loop highlighting method.

Another improvement of the Wavelet transform is the Curvelet transform which detects image details along curves instead of radial directions. Curvelets decompose the image into a set of wavelet bands and apply the Ridgelet Transform on each band.

Wavelets, Ridgelets, and Curvelets are used for noise removal or contrast enhancement (Zhang, et al., 2008; Starck, 1999). To remove noise, the signal to noise

ratio (SNR) of the signal is used. For higher SNR values, the transformation stops and noise is not carried to the frequency space. Wavelets remove noise and retain the data in the horizontal or vertical directions, whereas Curvelets retain the data on curves. Curvelet based noise removal is applied on astronomical images in (Starck, 1999).

2.2.3 Curve Segmentation

Curve segmentation or dividing the curve into sub-segments plays an important role in the contour grouping part of our study. For the success of contour grouping, smooth, squiggle and jaggedness-free contours are required. The tangent and curvature are necessary for locating corners or breakpoints. The locations of abrupt changes in orientation or in curvature are the candidate locations where a curve can be segmented into sub-segments.

Paramanand *et al.* divides the curve into smaller segments at its corners which are detected with the help of curvature value (Paramanand, 2006). Their method calculates the curvature of the points using the K -cosine measure. Let P be the point set of a curve. K -cosine, $c_i(K)$ in Eq. (2-2), calculates the angle between the vectors from P_i to P_{i-k} and from P_i to P_{i+k} . P_{i+k} and P_{i-k} are the pixel values of K pixels further and K pixels behind the current pixel, respectively (Sun, et al., 2007). *Figure 2-5 (a)* shows the vectors of P_i for the K -cosine curvature calculation.

$$c_i(K) = \frac{\vec{a}_i(K) \cdot \vec{b}_i(K)}{|\vec{a}_i(K)| |\vec{b}_i(K)|} \quad 2-2$$

For each point along the P curve, the K -cosine value is calculated for each point and the curvature plot of the curve is obtained as shown in *Figure 2-5 (b)*. Corner points

are detected on the curvature plot via thresholding. If the curvature value is above a certain threshold, then that point is determined as a corner (Paramanand, 2006; Sun, et al., 2007). One of the drawbacks of the K -cosine is the selection procedure of the K value. If K is too small, the curvature values will be steady on the plot and will not create corner points. If K is too big, then determining the exact place of the corner point will be a problem. Another issue is the necessity of using different thresholds for different K values.

Figure 2-5 (a) Curvature definition with K -cosine, (b) Corner detection based on the K -cosine curvature plot (Sun, et al., 2007)

After detecting corner points, Paramanand *et al.* segment the curves at the corner points and obtain sub-segments. Later they investigate the shape of the sub-segments and determine whether a sub-segment corresponds to a line segment, an elliptical arc, or a curve with a smooth joint. They perform direct Least Squares fitting to the points of a sub-segment, and check the error of linear fitting to determine whether the segment is a straight line segment. If the sub-segment cannot fit a line segment, then ellipse fitting is applied. They use the average of the ellipse fit errors given by Eq. (2-3).

$$EOF_1 = \frac{\sum_{i=1}^n Ax_i^2 + Bx_iy_i + Cy_i^2 + Dx_i + F}{n} \quad 2-3$$

If the elliptical error of fit is small, then the segment is classified as an elliptical arc. If the error is too high for a good ellipse fit, then the presence of smooth joints are checked. To detect smooth joints, they divide the curve into windows, then take the average of each window to find a representative (x', y') , and calculate the tangent vector and normal vector. If the angle between tangent vectors is equal to the angular change between normal vectors, then there is no change along the curve. Otherwise, the shape of the curve is considered to be changing at the location where the angle change occurs, and there is a smooth joint at that location. At the location where they determined a smooth joint, they divide the curve into sub-segments and apply line fitting or ellipse fitting to the sub-segments again. *Figure 2-6* illustrates an original image, detected corners, and elliptical arcs.

Figure 2-6 Segmenting a curve at the corners, then finding linear and elliptical curves according to Paramanand's method (Paramanand, 2006)

Another line segmentation method using ellipse information can be found in (Kawaguchi, et al., 1998). Kawaguchi *et al.* first extract line-support regions from the original image, then select candidates for elliptical arcs from those line groups. They compute the eccentricity of the line-support region, and if the eccentricity is greater than a certain threshold, they keep the line group. Then, they partition the line segment into

three segments with equal lengths, such that the gradient orientations of the consecutive parts are in a monotonic ascending or descending order. They perform a genetic algorithm on the line segments to find the optimal ellipse fits, then they calculate the fitness values for all the generated ellipses, and select the ellipse with the highest fitness value.

In another curve segmentation method, Ichoku *et al.* propose to first fit a line to an entire curve based on end-to-end straight line fitting (Ichoku, et al., 1996). Then deviations from the straight line are determined, with the error criterion possibly being the maximal deviation, mean deviation, mean square deviation, or normalized maximal deviation. If the fit is bad, the algorithm fits a circular arc to the entire badly fitted curve. If the circular fit is also bad, the entire curve is shortened from one end, then the process of fitting a line and then fitting a circle is repeated for the bad fits. This process continues until either a line or a circular curve fits the progressively shortened curve. The procedure continues until the input curve is completely segmented. This process tends to cause data loss since important parts of the curves can be deleted easily.

Fischler *et al.* propose a method that partitions a curve at discontinuity points (Fischler, et al., 1986). The algorithm labels each point as a point in a smooth interval, a critical point, or a point in a noisy interval based on analyzing deviations from a chord or a line that joins the two endpoints of the curve. If the curve is close to the chord, then it is considered as a curve point. If the curve makes a single excursion which is an abrupt change, then the point farthest away from the chord is considered a critical point. If the curve makes two or more excursions, then the points in the interval are labeled as noise

points. The algorithm finally divides the curve structure at the critical points to accomplish curve partitioning.

Fischler *et al.* extend their study and segment the lines at transitions which they call the Saliency Selection System (SSS) (Fischler, et al., 1994). In this study, lines are segmented based on a transition likelihood histogram where the peaks of the histogram are determined as transitions. Each point along the line is given a histogram value for the likelihood that it is a transition point, based on the severity of direction and curvature change around it. This study measures the severity of transition by iteratively sliding a fixed-length “stick” or chord along the segment. Since the endpoints of a chord must touch the segment, the center of the stick usually should not be far from the segment’s points either. Whenever a point deviates significantly at the various stick positions, at least beyond a predefined noise threshold, a transition exists at that peak value. *Figure 2-7* demonstrates the detected transitions along a curve with a transition likelihood graph.

Figure 2-7 The curve is segmented at the transitions. A transition likelihood graph is shown on the left. (Fischler, et al., 1994)

One application area of line segmentation can be found in handwriting analysis (Zhang, et al., 2006). They break down segments into arcs and lines to isolate individual pencil strokes during handwriting analysis. The typical definition of a transition used in

this study is an anomalous point of the first or second order where either direction, curvature, or both, undergo a sudden change.

2.3 Feature Extraction

Features are used for describing the characteristics of the patterns to be learned (Duda, et al., 2001). Good features discriminate the desired patterns from the irrelevant patterns and from the background. Different features are useful for different tasks. For example, color information can be useful to distinguish between the Sun and the sky in an image, whereas texture information would be more appropriate to distinguish between grass and tree. To select the appropriate features, we have to examine the attributes of the desired patterns and other patterns wisely.

Features can be calculated globally or locally: global features are extracted from the entire image, whereas local features are extracted from the local regions of an image. Histograms are commonly used for global feature extraction and are invariant to image translation or rotation. Also, after applying normalization on histograms, they become invariant to scale. Histograms are used for indexing and retrieval of images (Swain, et al., 1991).

To extract local features, an image is divided either into fixed sized blocks or interesting regions which are extracted using segmentation techniques. Sliding window approaches can raster an entire image to check the existence of an object at a certain location in the image. Local features could also be extracted from curve segments, edges, and contours (Schmid, et al., 2004). There is a wide variety of features for pattern

recognition from images. Below, we review the features that have investigated for coronal loop detection.

2.3.1 Statistical Features

Statistical features are calculated using the intensity values of an image (Gonzalez, 2007). The intensity histogram is expected to be an approximation of the intensity probability density, $p(\cdot)$. Statistical features are computed based on the *central moments* of the histogram defined in Eq. (2-4). *Central moments* are the moments of the distribution around its mean μ .

$$\mu_n = \sum_{i=0}^{L-1} (z_i - m)^n p(z_i) \quad 2-4$$

The mean, m in Eq. (2-5), measures the average intensity of the given image. L is the number of intensity values and it is 256 for gray scale images. $p(z_i)$ is the estimate of the probability of value z_i occurring in the image. The standard deviation as given in Eq. (2-6) measures the average contrast.

$m = \sum_{i=0}^{L-1} z_i p(z_i)$	2-5
$\sigma = \sqrt{\mu_2}$	2-6
$R = 1 - 1(1 + \sigma^2)$	2-7
$\mu_3 = \sum_{i=0}^{L-1} (z_i - m)^3 p(z_i)$	2-8
$U = \sum_{i=0}^{L-1} p^2(z_i)$	2-9
$e = -\sum_{i=0}^{L-1} p(z_i) \log_2 p(z_i)$	2-10

Smoothness, R which is given in Eq. (2-7), measures the smoothness of the intensity in a region. R is 0 if a region has constant intensity and gets close to 1 if intensity levels fluctuate in a region. Skewness, μ_3 given in Eq. (2-8), measures the symmetry of distribution. Skewness is 0 for symmetric histograms, positive for right skewed histograms, and negative for left skewed histograms. Uniformity, U given in Eq. (2-9), measures the uniformity of intensities in the histogram. If all gray values are equal in the image, then this measure takes the maximum value. Entropy, e given in Eq. (2-10), measures the randomness in the histogram.

2.3.2 Edge Histogram Descriptors

Edge histogram descriptors (EHD) represent the local edge distribution of an image with a histogram (Won, et al., 2002). The edge histogram represents the frequency of five directions in the image, which are vertical, horizontal, 45-degree, 135-degree, and non-directional. The edge information is extracted from each sub-image through spatial filters.

$$\begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, \begin{bmatrix} \sqrt{2} & 0 \\ 0 & -\sqrt{2} \end{bmatrix}, \begin{bmatrix} 0 & \sqrt{2} \\ -\sqrt{2} & 0 \end{bmatrix}, \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$$

Figure 2-8 Spatial filter masks: vertical, horizontal, 45-degree, 135-degree, non-directional

The image is divided into 16 equal sized sub-blocks (Won, et al., 2002). For each sub-image, the edge histograms of five directions are computed. Each local histogram contains 5 bins. To represent an entire image, 80 bins are required. Each histogram bin

value is normalized by dividing the value by the total number of edges in the sub-image, so that the bin value becomes between 0 and 1. For monotone blocks which are absent of directions, edge histograms do not change much. Only blocks which have strong directions affect the edge histograms.

2.3.3 Histogram of Oriented Gradients

The histogram of oriented gradients (HOGs) counts the occurrences of gradient orientation in localized portions of an image (Dalal, et al., 2005). They divide the image into small spatial regions. For each region, they accumulate the edge orientations of the region. For each small sub region, they keep an orientation histogram. The main difference from edge histograms (Won, et al., 2002) is that this method is not restricted to five directions.

The gradient defines the tangent at that point, and its direction is the normal to the curve at that point. For a function $f(x, y)$, the gradient of f at coordinates (x, y) is defined as the two dimensional column vector given in Eq. (2-11).

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad 2-11$$

The gradient magnitude, $mag(\nabla f)$ given in Eq. (2-12), gives the steepness of direction at a point. It is the maximum rate of increase of $f(x,y)$ per unit distance in the direction of ∇f .

$$mag(\nabla f) = \sqrt{G_x^2 + G_y^2} \quad 2-12$$

The direction of the gradient vector at $\nabla f(x,y)$ is also important. The direction $\theta(x,y)$ is given in Eq. (2-13).

$$\theta(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \quad 2-13$$

After calculating the orientation of the gradient for each pixel, orientations in the sub-image are binned in the histogram. Here the angle range is mapped from $[-\pi, \pi]$ to $[-180^\circ, 180^\circ]$. Then this range is divided into n channels. According to Dalal, using unsigned gradient orientations in 9 histogram channels tends to perform best in image retrieval problems (Dalal, et al., 2005).

2.3.4 Curvature Features

Curvature is the rate of change in the edge direction. The edge direction changes rapidly at the corners, whereas it changes little at smooth junctions. To calculate curvature features, second-order differential geometry can be useful. Hessian matrix, H , is the equivalent of gradient for second-order geometry.

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix}$$

The direction of a point is calculated using the Hessian matrix. The partial derivatives, I_{xx} , I_{xy} , I_{yx} and I_{yy} , are computed using partial differentials after convolving the image with a Gaussian smoothing kernel that is essential to remove noise from the image. The eigenvalues and eigenvectors of the Hessian matrix have the following geometric meaning:

- The first eigenvector (the one whose corresponding eigenvalue has the largest absolute value) is the direction of greatest curvature (second derivative).
- The second eigenvector is the direction of least curvature.

- The corresponding eigenvalues, λ_1 and λ_2 , are the respective amounts of these curvatures.

The eigenvectors of H are called principal directions of pure curvature. The eigenvalues of H are called principal curvatures, and are denoted as, λ_1 and λ_2 . Based on principal curvatures, the following curvature features are derived in (Wang, et al., 2008): Gaussian curvature, mean curvature, curvedness, and shape index. Gaussian curvature is the product of two curvatures as given in Eq. (2-14) and is denoted as K . It is also called total curvature. Mean curvature is the average of two curvatures as given in Eq. (2-15) and is denoted as H . Curvedness measures the magnitude of curvature of a surface and the amount of deviation from flatness. It is the root square of the summation of squared curvatures as given in Eq. (2-16) and is denoted as C .

$K = \lambda_1 * \lambda_2$	2-14
$H = (\lambda_1 + \lambda_2)/2$	2-15
$C = \sqrt{(\lambda_1^2 + \lambda_2^2)/2}$	2-16
$S = -\frac{2}{\pi} \arctan \frac{\lambda_2 + \lambda_1}{\lambda_2 - \lambda_1} \quad \lambda_1 \geq \lambda_2$	2-17

The shape index, S , given in Eq. (2-17), is another measure using curvatures. Its value ranges from -1 to 1 and describes the morphology of the surface independently of scale. The surface corresponds to a bowl ($S = -1$), a valley ($S = -1/2$), a ridge ($S = +1/2$), a dome ($S = 1$) or a saddle ($S = 0$). *Figure 2-9* illustrates the different shapes corresponding to different value of the shape index.

Figure 2-9 Different values of the shape index correspond to different shapes

In addition to the shape index, the eigenvalues measure the convexity and concavity in the corresponding eigen directions. A ridge is a region where $\lambda_1 \approx 0$ and $\lambda_2 \ll 0$. Elliptic points occur where $\lambda_1 * \lambda_2 > 0$, and Hyperbolic points occur where $\lambda_1 * \lambda_2 < 0$.

The curvature feature is commonly used in biomedical imaging problems. Martinez-Perez uses second directional derivatives to extract blood vessels in retinal images (Martinez Perez, et al., 2001). They first convolve the image with second derivatives of the Gaussian function. Then they compute the eigenvalues, λ_1 and λ_2 , of the Hessian matrix. The eigenvalues measure the convexity and concavity in the corresponding eigen (principal) directions. They keep the maximum eigenvalue of the pixel and its magnitude as features which are used to classify pixels as a background or vessel (curve) point. After point classification, they perform region growing by starting from selected seeds.

In another biomedical imaging application, Wang uses the histogram of curvatures to match polyp candidates from different views (Wang, et al., 2008). From polyp candidates, they extract curvature related descriptors such as shape index, curvedness, Gaussian and mean curvatures. These descriptors are rotation, translation, and scale invariant. They extract about 1400 features from polyp candidate pairs. To reduce dimensions, they apply a diffusion map algorithm.

Aside from the curvature feature, the Curvature Scale Space (CSS) descriptor has been used frequently in image classification and image retrieval in the last decade. CSS is one of the MPEG-7 features which describe the shape of planar curves (Mokhtarian, et al., 1996). The curvature value $k(u, \sigma)$ given in Eq. (2-18) is calculated for each pixel for increasing σ values. For each value, the image is convolved with the Gaussian kernel $g(u, \sigma)$. $X_u(u, \sigma) = x(u) * g(u, \sigma)$ and $Y_u(u, \sigma) = y(u) * g(u, \sigma)$. $X_{uu}(u, \sigma)$ and $Y_{uu}(u, \sigma)$ are the second derivatives. The curvature values for each point at different scales are then accumulated. CSS is scale and rotation invariant, and the CSS graph can be used to easily detect the salient points of the image. Almeida uses the curvature scale space descriptor for shape based image retrieval. They use CSS as a shape descriptor and the Self Organizing Map (SOM) model as a CSS space organizer and summarizer (Almeida, et al., 2007).

$$k(u, \sigma) = \frac{X_u(u, \sigma)Y_{uu}(u, \sigma) - X_{uu}(u, \sigma)Y_u(u, \sigma)}{(X_u^2(u, \sigma) + Y_u^2(u, \sigma))^{\frac{3}{2}}} \quad 2-18$$

To calculate the curvature along the curves, angle changes among consecutive windows are commonly used. In addition to angle changes or second derivatives, curvature along the curves can be calculated by chord-to-point distance functions (Han, et al., 2001; Fu, et al., 1997). The chord is a straight line between the end points of a contour. Han calculates the discrete curvature by accumulating the distance from a point in the boundary to a specified chord with length L (Han, et al., 2001). They browse the chord from one endpoint to the other by sliding one point each time. For each chord movement, they calculate the distance of a point from the chord and accumulate the

distances to calculate the discrete curvature for each move. The corner points have high curvature in this method.

Fu extracts the main features of a contour based on a curve bend function which measures the bending degree at each point on the contour (Fu, et al., 1997). The curve bend function characterizes properties such as convexity or concavity of the curve segments. It is defined using the distance between the chord and the point.

2.4 Classification

Classification is the task of assigning objects to one of several predefined categories (Duda, et al., 2001). In the image domain, classification techniques are used to predict the labels of objects in the images, to filter the images with certain labels, to tag the segmented regions in the images, etc. In our problem domain, we want to determine the class label of the given solar image.

Suppose that we have K categories. Given a set of features, x , a classifier determines the class label, c^* , of the data instance. The probability of each class is $P(c_k | x)$. The class label of the data instance is assigned to the label giving the highest probability given by Eq. (2-19).

$$c^* = \underset{k}{\operatorname{argmax}} P(c_k | x), \quad k = 1, 2, \dots, K \quad 2-19$$

The task of a classifier is to estimate the probability $P(c_k | x)$, which requires learning a model from some training data. There are mainly two types of classifiers:

- *Generative classifiers* model the common attributes among the objects of the same category. Generative classifiers estimate the likelihood $P(x | c_k)$ and obtain $P(c_k | x)$ using Bayes's rule. Naïve Bayes is one of the generative classifiers (Duda, et al., 2001).

- *Discriminative classifiers* model the difference between categories. They find discriminant surfaces to separate categories. They directly calculate the $P(c_k | x)$. Nearest neighbor (KNN) classifiers and Support Vector Machines (SVM) are examples of discriminative classifiers.

To model coronal loops, we investigate several classifiers including both generative classifiers and discriminative classifiers.

Naïve Bayes is a classification technique based on Bayes theorem which calculates the conditional probability of an instance with several features under each class and then classifies new instances into the class with the largest posterior probability (Duda, et al., 2001). *Eq. (2-20)* formulates the classification score based on Bayes theorem.

$$P(c_k | x) = \frac{P(x|c_k)P(c_k)}{P(x)} \quad 2-20$$

K-Nearest Neighbors (*K-NN*) is a lazy classifier in which instances are represented as points in a feature space and the parameter K is the number of nearest neighbors (Duda, et al., 2001). A label is assigned to a new point based on the majority class of these K neighbors. Thus, in 1-NN, the class of the closest neighbor is assigned to a new data instance, while in higher K values, the class with majority votes is assigned to the data instance. *Figure 2-10* shows an example of K-NN classification with different K values.

Figure 2-10 According to 1-NN, the label of X becomes “rectangle”. According to 5-NN, the label of X becomes “circle”

C4.5 is an algorithm used to generate a decision tree from a set of training data using the concept of information entropy (Quinlan, 1993). Entropy of node t is measured using Eq. (2-21), where $p(i|t)$ denotes the fraction of instances belonging to class i at node t . When entropy gets close to 0, the node becomes more discriminative. A decision tree consists of two types of nodes: a leaf that indicates the class, and a decision node that contains a value of an attribute. Each attribute of the data can be used to make a decision that splits the data into smaller subsets. C4.5 examines the normalized information gain given in Eq. (2-22), which is the difference between entropies of the parent node and summation of children nodes. The attribute that yields the highest normalized information gain is the one used to make the decision. The algorithm then continues building the tree recursively on the smaller sub-sets. C4.5 uses a simple depth-first construction and needs the entire data to fit in memory, thus it is unsuitable for large datasets.

$$Entropy(t) = - \sum_{i=0}^{K-1} p(i|t) \log_2 p(i|t) \quad 2-21$$

$$Gain(t) = Entropy(t) - \sum_{v \in Values(t)} \frac{t_v}{t} Entropy(t_v) \quad 2-22$$

Repeated Incremental Pruning to Produce Error Reduction (RIPPER) starts by ordering the classes according to increasing class prevalence (fraction of instances that

belong to a particular class) (Cohen, 1995). It then learns a rule set for the smallest class first, while treating the rest as the negative class. Then it repeats the same procedure with the next smallest class as the positive class, and so on. RIPPER creates a rule set by starting with an empty rule set and adds rules one by one until there are no more positive examples left. In each iteration, the training set is split into a *grow set* and a *prune set*. The *grow set* contains two thirds of the positive examples and two thirds of the negative examples, and is used to construct the rules. A rule is a conjunction of conditions. Starting from an empty conjunction rule, conditions are gradually added. Rules are grown, greedily, adding conditions with the largest information gain in the grow set compared to the rule without that condition, making the rule more specialized. After the rule is grown, it is pruned (simplified) using the prune set, making the rule more general. As conditions are added, the rule becomes more and more specific, therefore covering fewer positive examples and fewer negative examples. This continues until the rule covers no negative examples from the *grow set*.

Boosting is a process in which a strong classifier, H , is created by combining M weak classifiers, h_m using Eq. (2-23) where h_m is a weak classifier and α_m is the weight of h_m (Shapire, et al., 1999).

$$H(x) = \sum_{m=1}^M \alpha_m h_m \quad 2-23$$

AdaBoost is an iterative procedure that learns several weak classifier models while adaptively modifying the distribution of the training data (Freund, et al., 1997). In each iteration, it focuses more on the previously misclassified instances, and then combines all the resulting models. Initially, all N records are initialized with equal

weights. Then they are reweighted according to the classifier output: the correctly classified instances receive lower weight, while the misclassified instances receive a higher weight. In the next iteration, the weak classifiers are built to deal with the reweighted instances, so that it focuses more on the instances that have higher weight. The changes in the instances' weights depend on the overall error of the current classifiers.

A classifier model can be evaluated by comparing the predicted class labels of several data samples to the actual class labels of these samples. **Table 2-1** illustrates the basic performance measures built over a 2x2 confusion matrix, where TP and TN denote the numbers of correctly classified positive and negative samples respectively, while FP and FN denote the numbers of misclassified positive and negative samples, respectively. Accuracy can be calculated by dividing the number of correctly predicted samples to all samples, $A = (TP+TN)/(TP+TN+FP+FN)$. Precision represents how many of the positive predicted samples are really positive samples, $P = TP/(TP+FP)$. Recall measures the proportion of positive predicted samples to all positive samples, $R = TP/(TP+FN)$. Neither precision nor recall is a good measure by itself. Both values should be high for a good classifier. The F-measure (F1-score) combines both precision and recall, $F = 2RP/(R+P)$. Another measure is the geometric mean, which is defined as $G_{mean} = \sqrt{TP_{rate}TN_{rate}}$ where $TP_{rate} = TP/(TP+FN)$ is true positive rate and $TN_{rate} = TN/(TN+FP)$ is true negative rate. G_{mean} maximizes the accuracy on each of the two classes in order to balance both classes at the same time (Matwin, et al., 1997).

Table 2-1 Confusion matrix for a two-class problem

	Positive Prediction	Negative Prediction
Positive Class	True Positive (TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

The Receiver Operating Characteristics (ROC) characterizes the relation between positive hits and false alarms. ROC curves plot the *TP* (on the y-axis) against the *FP* (on the x-axis). A good model should reach a high *TP* and low *FP* quickly. The area under the ROC curve is another measure of goodness of a classifier. The ideal area under the curve is 1 (Tan, et al., 2006).

The error of classifiers can be divided into two types: training errors and generalization errors. Training error is the number of misclassified records in the training data. Generalization error is the expected error on unseen data. Sometimes the model fits the training data too well but does not fit on unseen data (test data) well. This situation is called overfitting. A good model must have low training error and low generalization error.

When the number of features used in the learning algorithm increases, the learning algorithms do not learn very well most of the time (Tan, et al., 2006). The algorithms work better when the number of attributes is lower. Dimension reduction can eliminate irrelevant or redundant features, reduce noise and yield more understandable models. Redundant features duplicate the information in one or more other attributes. Irrelevant features do not contain any useful information for the learning method. Another benefit of dimensionality reduction is reducing the time and memory required by

the learning algorithm. The reduction of dimensionality can be done through feature subset selection or Principal Component Analysis (PCA).

Feature selection can be done during the learning procedure. In this case, it is called embedded feature selection (Jain, et al., 1997). Hence, a learning algorithm decides which features to use in the model. Features can also be selected before learning and the selected subset of features can be used in the learning model. These feature selection methods are called filtered approaches. Another way uses the classification techniques to decide the best features and called wrapper approaches (Kohavi, et al., 1997).

The best subset contains the smallest number of features which contribute most the accuracy. Forward selection and backward selection can be used to select the best subset. Forward selection starts with an empty set and includes the most effective feature which increases the accuracy. This process continues until the point where the accuracy does not change significantly. Backward selection starts with the entire set of features and eliminates the least effective one in each time until any further feature removal hurts the accuracy significantly. The impact of a feature can be measured through mutual information, information gain, and entropy measures. To select the best feature subset, optimization techniques including Genetic Algorithms, Greedy Search, Best First Search, and Exhaustive Search could be used (Koller, 1996; Jain, et al., 1997).

2.5 Ellipse Detection

Ellipse detection has various application areas such as obtaining ellipsoid objects from satellite images (Soh, et al., 2009) and shape retrieval (Wu, et al., 1993). Ellipse fitting can also be used in curve segmentation (Paramanand, 2006). One interesting

application of ellipse fitting can be found in Soh's study which performs ellipse fitting on aerial images to find ellipsoid objects (Soh, et al., 2009).

Since coronal loops are approximately semi-elliptical shapes, we review the ellipse detection methods. Ellipse fitting techniques (Bookstein, 1979; Fitzgibbon, et al., 1999; Sampson, 1982) or the Hough transform (McLaughlin, 1998; Duda, et al., 1972; Tsuji, et al., 1978) can be used to determine whether the given curve is ellipsoid.

2.5.1 Ellipse Fitting

In ellipse fitting methods, the given data points are fitted to a conic section given in Eq. (2-24). The objective of ellipse fitting methods is computing the function parameters, $P = [A \ B \ C \ D \ E \ F]^T$.

$$F(x,y) = Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0 \quad 2-24$$

The parameters of the conic could be used in determining the shape of the conic. Some rules about function shape are: If $B^2 - 4AC < 0$, it is an ellipse; if $B^2 - 4AC = 0$, it is a parabola; and if $B^2 - 4AC > 0$ it is a hyperbola.

To compute the unknown parameters, Bookstein minimizes the sum of squared algebraic distances (Bookstein, 1979), $G = \sum_{i=1}^n (P \cdot x_i)^2$ where P is conic section parameters and $x_i = [x_i^2 \ x_i y_i \ y_i^2 \ x_i \ y_i \ 1]$. According to Bookstein, the following constraint on the parameters needs to be placed: " $A^2 + B^2/2 + C^2 = 1$." He solves the eigenvalue of this equation to obtain the conic parameters. Bookstein uses the algebraic distance to compute the parameters (Bookstein, 1979). The algebraic distance

puts the data points into the generated function and takes the average of the error as given in Eq. (2-3).

Sampson presents an iterative improvement of Bookstein's method by replacing the algebraic distance with the gradient distance given in Eq. (2-25) (Sampson, 1982). The disadvantage of the algebraic distance is high curvature bias which causes less influence of the data located near the ends of the fitted curve (Rosin, 1996).

$$EOF_2 = \frac{F(x_i, y_i)}{|\nabla F(x_i, y_i)|} \quad 2-25$$

Fitzgibbon fits ellipses to scattered data with a direct least square method by imposing the equality constraint " $4AC - B^2 = 1$ " (Fitzgibbon, et al., 1999). First Fitzgibbon constructs a scattered matrix of the given points. Let D be the matrix consisting of the given points as rows. Let $D^T D$ be the scatter matrix S .

$$D = \begin{bmatrix} x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & x_2 y_2 & y_2^2 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 & x_n y_n & y_n^2 & x_n & y_n & 1 \end{bmatrix}$$

Fitzgibbon method solves the equation " $SP = \lambda GP$ " for P , where P is a vector of the conic section parameters, λ is a Lagrange multiplier, and G is a 6x6 constraint matrix and its elements are:

$$G = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

λ is a relative eigenvalue of S with respect to G . The eigenvector with minimum eigenvalue is the solution for P . The error function of Fitzgibbon is given by

$$EOF_3 = \sum_{i=1}^n \frac{Ax^2 + Bxy + Cy^2 + Dx + Ey + F}{4AC - B^2} \quad 2-26$$

Wu derives the ellipse parameters such as ellipse center (x_c, y_c) or axes lengths (K, L) with respect to conic parameters, P (Wu, et al., 1993). To transform a conic function into the standard ellipse equation: " $x^2/K^2 + y^2/L^2 = 1$ ", they have two steps: 1) translating origin from $(0, 0)$ to (x_c, y_c) to eliminate the coefficients D and E in the conic function parameters, 2) rotating (x_c, y_c) counterclockwise by an angle θ to the eliminate coefficient B in the conic function parameters. We substitute x with $(x' + x_c)$ and y with $(y' + y_c)$ to obtain $A(x')^2 + Bx'y' + C(y')^2 + Dx' + Ey' + F = 0$, and then reduce the $(Dx' + Ey' + F)$ part to f' . After the changes, we reduce the conic function to $A(x')^2 + Bx'y' + C(y')^2 + f' = 0$. We obtain the center coordinates of an ellipse, which are given by

$$x_c = \frac{-2CD + BE}{4AC - B^2}, y_c = \frac{-2AE + BD}{4AC - B^2}, f' = F + \frac{Dx_c + Ey_c}{2}$$

The angle between the major axis and a horizontal line is

$$\theta = \frac{1}{2} \tan\left(\frac{A-C}{B}\right)$$

If we rotate the coordinate system with an angle θ and take $x' = x''\cos\theta - y''\sin\theta$ and $y' = x''\sin\theta + y''\cos\theta$, then we remove the B coefficient and obtain the ellipse equation, $a'(x'')^2 + c'(y'')^2 + f' = 0$ where

$$a' = A \cos^2 \theta + B \cos \theta \sin \theta + C \sin^2 \theta$$

$$c' = A \sin^2 \theta - B \cos \theta \sin \theta + C \cos^2 \theta$$

The major axis K and minor axis L of an ellipse are calculated in terms of a' , c' , and f' .

$$K = \sqrt{\frac{-f'}{a'}}, \quad L = \sqrt{\frac{-f'}{c'}}$$

Eccentricity is the ratio of the distance between the two foci to the length of the major axis length, K , as given in *Eq. (2-27)*.

$$\epsilon = \frac{\sqrt{|K^2 - L^2|}}{K} \quad 2-27$$

Eccentricity is a measure of how much the conic section deviates from being circular. The eccentricity of a circle is zero, the eccentricity of an ellipse which is not a circle is greater than 0 but less than 1, the eccentricity of a straight elongated line is 1, and the eccentricity of a hyperbola is greater than 1. *Figure 2-11* demonstrates different ellipsoid shapes along with their eccentricity.

Figure 2-11 Eccentricity values for different ellipses. It gets close to 0 when the ellipse becomes more circular and close to 1 when the ellipse becomes more elongated.

2.5.2 The Hough Transform

Another method for ellipse detection is based on the Hough Transform. The basic idea of the Hough transformation is to find curves that can be parameterized like straight

lines, polynomials, or circles. In our study, we use the Hough Transform for both line detection and ellipse detection in solar image regions. The Hough Transform (Duda, et al., 1972) is a well-known method for line detection from images. The straight line $y = mx + b$ can be represented in the image space whereas this line can also be parameterized in the form: $\rho = x \cos \theta + y \sin \theta$, where ρ is the line distance from the origin, while θ is the angle of the vector from the origin.

Figure 2-12 The input and output spaces of Hough Transformation

When we take different points from a straight line and transform them into the Hough space (as in Figure 2-12), we will see that their sinusoidal curves will intersect in the same θ angle. The resulting peaks in the Hough space represent strong evidence that a straight line exists in the image. There are a number of methods which extract these peak regions, or *local maxima*, from the Hough space (Illingworth, et al., 1988; Svalbe, 1989). Quantization is applied to the Hough space because of space and time limitations and the noisy characteristics of lines (Duda, et al., 1972). The parameters of a line can be estimated more accurately using a finer quantization of the parameter space. For noise tolerance, however, a coarser quantization is better.

The length of a line can be predicted roughly from the number of sinusoidal curves crossing at the θ angle of the specified line. The global maxima in the Hough space represent the longest and strongest line combination of the image. We can predict the length of the longest line in the image from the number of points in the highest peak of the Hough Space.

The same procedure for line detection can be used to detect other shapes or curves. An ellipse can be described by its center coordinates (p, q) , semi major axis length (a) , semi minor axis length (b) , and orientation (θ) . Finding the ellipse that passes through the given points requires calculating these five parameters (McLaughlin, 1998; Duda, et al., 1972; Tsuji, et al., 1978). Instead of trying each point, McLaughlin accelerates the procedure of ellipse detection through random point selection (McLaughlin, 1998). McLaughlin first finds the center of the ellipse by picking three random points (x_1, x_2, x_3) and then calculates the axes lengths. To detect the center of the ellipse, this method determines the line equations for three points separately and finds the intersection of the lines passing through x_1 and x_2 and x_2 and x_3 . Then, the bisector lines of those intersection points are determined. The intersection of two bisectors is located at the center of the ellipse. *Figure 2-13* illustrates the points, the tangent lines, bisectors, and the ellipse center. After finding the center of the ellipse, the remaining ellipse parameters (a, b, θ) are calculated via selecting three random points to generate three linear equations. The solutions of these equations give the ellipse parameters.

Figure 2-13 Determining the center of an ellipse using three random points. The ellipse center is located at the intersection of the bisections of three tangent lines to the ellipse.

2.6 Contour Grouping

In this part, we review perceptual organization, saliency, salient contour detection, grouping measures, and several contour grouping methods.

2.6.1 Perceptual Organization

In perceptual organization (Koffka, 1935), Gestalt factors in the human visual perception are highly utilized as a basis for contour grouping. Gestalt factors include proximity, similarity, closure, continuation, symmetry, etc. The human visual system groups elements of a perceived scene into meaningful or coherent clusters and partitioning the curves is not a generic task that is independent of purpose. Relations among the curves such as symmetry, repeated structure, and parallel lines increase the perception of the curves. Also, noise definitions can be different depending on the application.

Perceptual grouping associates structurally related entities together by taking the human visual system as a cue (Lowe, 1985). In the image plane, blobs, edge segments, and geometrical features of the image regions can be grouped. Perceptual grouping studies (Zhu, 1999) commonly build their models based on Gestalt laws (Koffka, 1935).

Gestalt psychologists developed a set of principles to explain perceptual organization and how smaller entities are grouped to form larger ones. These principles are often referred to as the "laws of perceptual organization" which are illustrated in *Figure 2-14*.

Figure 2-14 Gestalt laws of perceptual organization

The most common Gestalt laws used in perceptual grouping are as follows:

- *Similarity*: Items which share visual characteristics such as shape, size, color, texture, or value will be seen as belonging together in the viewer's mind. Goldberg *et al.* propose brightness and contrast cues for similarity (Goldberg, et al., 2002).
- *Proximity*: Objects or shapes that are close to one another appear to form groups. Even if the shapes, sizes, and objects are radically different, they will appear as a group if they are close together. According to Goldberg, the probability of grouping of two segments decreases when the distance between them increases (Goldberg, et al., 2002).
- *Continuity*: Humans tend to follow the shapes beyond their ending points. Thus, the edge of one shape will seem to continue into the space and meet up with other shapes or the edge of the picture plane. Goldberg *et al.* propose two cues for good continuation between two segments: first is the parallelism cue which is $\overrightarrow{\theta_{ij}} + \overrightarrow{\theta_{jl}}$ and approaches zero when segments become parallel; the other is the co-circularity cue which is $\overrightarrow{\theta_{ij}} - \overrightarrow{\theta_{jl}}$ and approaches zero when segments become more circular (Goldberg,

et al., 2002). *Figure 2-15* shows the angles of two segments. For the collinear segments, both parallelism and co-circularity values are zero.

Figure 2-15 Two segments and their parameters for modeling the proximity, good continuation, and similarity (Goldberg, et al., 2002)

- *Closure*: Objects that are seen as a whole tend to be grouped together. Closure is the effect of suggesting a visual connection or continuity between sets of elements which do not actually touch each other in a composition. The principle of closure applies when we tend to see complete figures even when part of the information is missing. Continuity in the form of a line, an edge, or a direction from one form to another creates a fluid connection among compositional parts.
- *Curvature Consistency (Prägnanz)*: Humans tend to discern curves with a constant curvature. Some regularity and simplicity are easily interpreted by our sensory information.

2.6.2 Salient Curve Detection

Certain objects or contours pop out from a scene and attract more attention. This behavior is measured by saliency. The Gestalt psychologists identify several factors such as continuity, co-linearity, or closure to clarify why certain objects or contours from crowded scenes attract more attention than others.

Ullman *et al.* propose structural saliency based on the length of the curve, its continuity, and total curvature (Ullman, et al., 1988). The total curvature of the curve is the summed slopes in consecutive windows along the curve. Humans connect fragmented segments in such a way that the total curvature is minimized. Hence, salient curves should have low total curvature. They take into account the gaps among curves and call them *virtual elements*. The real curve segments are called *active elements*. The saliency measure of a curve is the weighted sum of local saliency measures of its active and virtual elements. For each pixel in the image, this method calculates a saliency measure considering the orientation relation between the pixel and its neighbors. It generates a saliency map which assigns higher intensity levels to interesting locations in the image. *Figure 2-16* illustrates a circle standing out in the clutter and the saliency map of the image has higher intensity levels for the circle.

Figure 2-16 A circle in a cluttered background on the left and its saliency map on the right (Ullman, et al., 1988)

Guy and Medioni extract salient (perceptual) contours from the images using co-curvilinearity and proximity measures (Guy, et al., 1996). They convolve the image with a special mask called an extension field. This mask encodes the likelihood and orientation of possible continuations. For each pixel, the extension field collects votes from the other segments in the image. The pixels with high votes and consistent orientation represent the salient points. This voting system is somehow similar to the Hough transform, yet they

are looking for certain parameters (smoothness, etc.) instead of exact shapes (line, circles, etc.).

Cheng *et al.* detect principal curves from original maps to detect roads from aerial images (Cheng, et al., 2004). Based on the smoothness of curves, the shortest path and directional deviations are calculated to find the principal curves in complicated curve networks. They build graphs of curve segments, such that each pixel in the skeleton image is converted to a graph node. A graph node has geometric and topological connections with the other nodes, and every node keeps the number neighboring nodes which indicates whether a node is an isolated node, end node, chain node, or junction node. They eliminate redundant nodes from the basic graph and group chain nodes to obtain super nodes which keep the angle list of the connected node group. Super nodes keep the length, straightness, and turning point list of the node group. They detect the principal curves from the super graph based on smoothness. They connect curve nodes to form smooth curve groups with the help of a depth first search algorithm. They start searching for principal curves from an unvisited end node and use depth first search and Dijkstra's algorithm to find the shortest path until reaching a certain length. They calculate the directional deviation of the path, and if it is small, then it is considered a good entrance. They then perform depth first search and Dijkstra's algorithm to find the entire path from the entrance until the end of the path, and finally keep the smoothest curves.

In another work, Gao *et al.* detect salient curves based on perceptual organization, which involves partitioning and grouping of curve segments using curve tracking methods (Gao, et al., 1993). In the partitioning phase, they introduce eight generic

segments which are defined by the tangent function. Four of them are concave or convex curve segments while the other four are differently oriented line segments. A different combination of curve segments forms a total of eight curve partition points. They define several curve tracking rules based on the monotonic characteristics of the tangent function.

Wang *et al.* extract perceptually salient closed boundaries in images via a ratio contour algorithm (Wang, et al., 2005). In this method, an object boundary is represented using real edge segments and the virtual segments which are the gaps between the real segments. They use the following prominence rules: real segments are more prominent than virtual segments; short virtual segments are more prominent than long virtual segments; and smooth segments are more prominent than unsmooth segments. Based on these rules, they define a boundary cost function which is the ratio of the sum of the total curvature and total gap length among segments to the total length of the boundary. The most salient boundary has the minimum boundary cost. To find the salient closed boundaries, they model the problem with an undirected graph in which each endpoint is a vertex and the edges connect the endpoints. Real segments are solid edges, while virtual segments are dashed as shown in *Figure 2-17*. In this graph, a close boundary is modeled as a cycle consisting of solid and dashed edges.

Figure 2-17 Endpoints are the vertices, the real segments are solid lines, and the virtual segments are dashed lines. (Wang, et al., 2005)

Felzenswalb *et al.* represent an image with a set of salient curves and propose a model which separates salient curves from background (Felzenswalb, et al., 2006). In this model, a curve is represented by a sequence of adjacent segments. Since the aim is to extract object boundaries, they use the probability of boundary (*PB*) function. They determine the probability of boundary for each pixel in the image. Each pixel can belong to either background or a salient curve. The total cost of the model is the summation of the curve costs and the background model cost. The optimal solution minimizes the total cost of the image. Their algorithm starts with one pixel and search for possible extensions repeatedly. When there is no extension decreasing the ratio of the cost to the length, the search mechanism for that curve terminates. They use a greedy search algorithm and the cost between two segments is the smoothness measure.

2.6.3 Grouping Methods

Contour grouping can be solved through probabilistic methods, such as Markov Random Fields or Conditional Random Fields. In the probabilistic approach, each contour has a probability value indicating its strength in grouping. The probability of grouping two contours can be calculated using Gestalt laws (e.g. their proximity, good continuation, similarity, and so on). A probabilistic grouping algorithm searches for optimal contour sequence c^* as given by Eq. (2-28), whose probability is the maximum given the cues, D , where c is an individual contour and G is a contour group.

$$c^* = \underset{c}{argmax} p(c \in G|D) \quad 2-28$$

The Markov model is commonly accepted in grouping methods to simplify the model and computational needs. According to this assumption, the grouping decision is

made in a local neighborhood. Markov Random Fields (MRF) models the context dependent entities through mutual influences among those entities (Li, 2009; Dubes, et al., 1989). In MRF, entities are labeled with one of the given labels based on neighborhood information. The labeling problem is specified in terms of a set of sites and a set of labels. Let S be a set of m sites: $S = \{1, 2, \dots, m\}$. Let L be a set of K labels: $L = \{1, \dots, K\}$. Labeling (f) is a mapping from S to L , $f: S \rightarrow L$. f_i assigns a unique label to site i . If we assume that label assignment is independent from neighborhood, then $P(f) = \prod_{i \in S} P(f_i)$. However, in the Markovian property, label assignment considers the local neighborhood. Let N denote neighborhood relations, then the probability of f_i is computed in its neighborhood, $P(f_i | f_{S-\{i\}}) = P(f_i | f_{N_i})$. $f_{N_i} = \{f_{i'} \in N_i\}$. A clique Q for (S, N) is defined as a subset of sites in S .

A set of random variables F is said to be a Gibbs random field on S with respect to N if and only if its configurations obey Gibbs distribution. Gibbs distribution takes the form $P(f) = \frac{1}{Z} e^{-\frac{1}{T} U(f)}$ where $Z = \sum_{f \in F} e^{-\frac{1}{T} U(f)}$ is a normalization constant, T is the temperature, and $U(f)$ is the energy function. The energy $U(f) = \sum_{q \in Q} V_q(f)$ is a sum of clique potentials $V_q(f)$ over all possible cliques Q . The value of $V_q(f)$ depends on the local configuration of the clique q . If we only consider the cliques size up to two, we can rewrite the energy as follows:

$$U(f) = \sum_{i \in S} V_1(f_i) + \sum_{i \in S} \sum_{j \in N_i} V_2(f_i, f_j) \quad 2-29$$

Based on this reduction, the joint probability can be written as

$$P(f) = \frac{1}{Z} \prod_{i \in S} r_i(f_i) \prod_{i \in S} \prod_{i' \in N_i} r_{i,i'}(f_i, f_{i'}) \quad 2-30$$

where, $r_i(f_i) = e^{-\frac{1}{T}V_1(f_i)}$ and $r_{i,j}(f_i, f_j) = e^{-\frac{1}{T}V_2(f_i, f_j)}$.

In MRFs, the label of a node is assigned randomly from among the labels in the clique of the node. The aim is minimizing $U(f)$ with different combinations of labels for each node in each clique until $U(f)$ reaches to the optimal value. For local optimization problems, greedy search techniques could be used, while global optimization algorithms (e.g. simulated annealing) should be utilized to reach the global optimum.

Posch *et al.* perform contour-based grouping based on perceptual organization (Posch, et al., 2001). They apply MRF to model context dependencies and consistent interpretation of image data with groupings. Grouping of the contours is done by using co-linearities and curvilinearities. Two straight line segments form a collinearity if the line segments lie approximately on a straight line and if the gap between them is small compared to the length of the segments. Two elliptical arcs form a curvilinearity if two elliptical arcs lie on an ellipse and the gap between them is small compared to the length of the segments. Two linear groups form a proximity when the gap between them is relatively small. They concentrate on proximity, good continuation, symmetry, and closure relations. Good continuation is referred to as collinearity for line segments and curvilinearity for elliptical arcs and parallelism is used as a indication of symmetry.

Posch *et al.* first detect edges, and then group edge points into straight lines and elliptic arcs (Posch, et al., 2001). The resulting contour segments are grouped hierarchically, such that different levels of the hierarchy represent different groupings based on Gestalt principles. For example, the first level of the hierarchy deals with one dimensional phenomenon with three different groupings: collinearity, curvilineararity, or proximity, while the second level of grouping is based on principles of closedness.

Ren *et al.* introduce a framework for contour grouping using a conditional random field (CRF) (Ren, et al., 2005). The CRF is a little different from Markov Random Fields. In MRF, hidden labels based on joint distributions are sought, while in the CRF conditional distributions of labels are given in the observations. They compute the probability of being on a boundary for an edge using local continuity model. For the local model, they use curvilinear continuity of the two edges at both ends of an edge and they assume that these two edges are independent from each other. For the global model, they build a factor graph which is based on the Constrained Delaunay Triangulation graph modeling. In the factor graph, the detected edge segments in the image and the virtual segments among them are constructed. In order to capture longer contours, they perform conditional random fields on the factor graph.

Tu *et al.* parse images into regions, curves, and curve groups (Tu, et al., 2006). This study is interested in three types of curve structures: *free curves* which are independent and elongated structures; *parallel groups* which are curves that form a Markov chain structure along their normal directions; and *trees* which are structured as Markov tree structures.

Elder *et al.* search for the boundaries of lakes in satellite images through contour grouping (Elder, et al., 2003). They use prior models obtained from ground truth to calculate the likelihood ratio of binary cues such as proximity, good continuation, and similarity (Elder, et al., 2003). To calculate the probability of proximity cue, each gap distance value is assigned a contour probability and a random probability. The ratio of these values is used as the likelihood ratio for the proximity cue. The same approach is

repeated for other cues. They use lake boundaries extracted manually for the training images.

Ji *et al.* merge the adjacent arc segments belonging to the same ellipse (Ji, et al., 1999). Two arc segments should be close (tested by the proximity condition) to each other to be merged. The directions of arc segments are checked as well. The direction condition has two rules: the start of one arc segment and the end of the segment should have the same direction; the end of one segment and the end of the other should have opposite directions. Another condition they hold is elliptical goodness, which requires that two segments should have similar statistics in the residual space.

2.6.4 Grouping Measures

To group two segments, we have to check whether they satisfy certain conditions. Some basic conditions could be curvature consistency at the join, the angular similarity at the join, the distance between the segments. We could also check whether the segments are lying on circle, line, or ellipse.

2.6.4.1 Linearity

The basic idea of measuring linearity is fitting a line to the combined data and defining a measure based on the error of fit. Eccentricity is another way to measure the linearity. Eccentricity is 0 for the circle and 1 for the elongated line. Lowe introduces a significance measure which is the ratio of the maximal deviation of the curve to the length of the curve (Lowe, 1987). The significance measure can also be used to determine linearity. The deviation will be zero for a straight line. Another way is calculating ellipse

parameters and axes lengths and defining the linearity in terms of axes lengths as follows: $(1 - \text{minor-axis}/\text{major-axis})$. The slope changes between consecutive windows along the curve could also be used to compute the linearity. If the slope remains the same among the windows, the curve is linear.

2.6.4.2 *Circularity*

Circularity measures have been defined in many different ways, such as in terms of the distribution of the distance of contour points from a central point (Proffitt, 1982), or in terms of the tangent space (Nguyen, et al., 2010). A circularity measure for curves should be invariant to rotation, and scaling. When the curve becomes more circular, the circularity measure should increase. When the curve deviates from a circle, then this measure should decrease.

Proffitt measures the circularity based on the distances of the points from the center of gravity of the given points. Let the mean radius and standard deviation be μ_r , σ_r , respectively. The circularity measure is defined as: $\sqrt{1 - (\sigma_r/\mu_r)^2}$. Haralick's circularity measure is defined as follows: μ_r/σ_r (Haralick, 1974)

One simple approach to measure circularity is by fitting a circle to a curve with the least square norm and measuring the cost of the fit as a circularity measure (Roussillon, et al., 2010). They find the inner radius (r_1) and outer radius (r_2) of the given points and then defines the circularity measure as the ratio of the squared inner radius to the outer radius: $(r_1)^2/(r_2)^2$.

Lee proposes a circularity measure with respect to the area of the given shape S and the area of the fitted circle, C (Lee, et al., 1970). The ratio of the intersection of two areas to the union of those two areas gives the circularity measure: $(S \cap C) / (S \cup C)$.

Stojmenovic *et al.* define the circularity using the distances of the points from the center of the circle (Stojmenovic, et al., 2007). In this method, a set of points are transformed from Cartesian to polar coordinates as shown in *Figure 2-18*. Point (x, y) in the Cartesian space would be represented by $(\sqrt{x^2 + y^2}, \arctan(y/x))$ in the polar space. Circular points become linear in the new representation. They define the circularity in terms of linearity in the polar transform.

Figure 2-18 Cartesian and polar representations (Stojmenovic, et al., 2007)

Nguyen (Nguyen, et al., 2010) proposed a similar approach to Stojmenovic's method (Stojmenovic, et al., 2007). Nguyen's method transforms the points into tangent space which consists of the tangents between consecutive points on a polygonal curve. From a circular shape, they expect a straight line in the tangent space. Their measure is designed for closed curves.

2.6.4.3 Ellipticity

Stojmenovic *et al.* first find the focal points of the ellipse (Stojmenovic, et al., 2007). In an ellipse, the sum of the distances from a point to focal points is constant for

every point. Using this fact, they transform the original set of points to the polar representation. Let d_1 and d_2 be the distances from x to the focal points. Then the polar distance between x and the center will be the sum of the distances ($d_1 + d_2$). For a perfect ellipse, the points should be further from the center with the constant distance ($d_1 + d_2$). They calculate this distance for each point and plot the distances in Cartesian form. They expect a vertical line located at r from the plot for the perfect ellipse. They measure the linearity of the line and in the end they use this measure as an ellipticity of the shape. They also propose another ellipticity measure based on the ratio of the distance from a point to the curve to the distance from the point to the ellipse center.

Another ellipticity measure compares the area of the given region S to its ellipse fit R (Kopyrnicky, et al., 2004). They define the ellipticity measure in terms of set operations as follows: $(\text{area}(S \setminus R) + \text{area}(R \setminus S)) / (\text{area}(R \cup S))$. Note that this measure might generate values out of the [0-1] range.

Rosin defines the ellipticity using the elliptic variance which is calculated in terms of the center of gravity $u = [u_1 \ u_2]$ and the covariance C of the data points, $p_i = [x_i, y_i]$ (Rosin, 2003). The covariance is $C = \frac{1}{n} \sum_{i=1}^n (p_i - u)^2$. The mean radius of the shapes is given in Eq. (2-31) and the elliptic variance is given in Eq. (2-32). In the end, the ellipticity is calculated as follows: $PI = 1/(1+Evar)$. This measure suffers from high curvature bias and does not produce reliable measures.

$$v = \frac{1}{n} \sum_{i=1}^n \sqrt{(p_i - u)C^{-1}(p_i - u)^T} \quad 2-31$$

$$Evar = \frac{1}{nv^2} \sum_{i=1}^n (\sqrt{(p_i - u)C^{-1}(p_i - u)^T} - v)^2$$

Ji *et al.* merge the adjacent arc segments belonging to the same ellipse (Ji, et al., 1999). They calculate the elliptical goodness based on the residual space. They perform a least square ellipse fitting to the combined data of two segments. For a good fit, the residual errors should be distributed with mean 0 and variance σ^2 . They calculated the error of fit using the geometric distance which is the minimum distance between a point and the estimated ellipse. Let $\bar{e}$ be the sample mean, S^2 be the sample variance of the residual errors, and N be the number of point in the residual space. Then, a test statistics can be written as $T = \frac{N\bar{e}^2}{S^2}$. The distribution of T is used as a measure of the goodness of ellipse fitting. In their method, they first fit the ellipse on the combined data and then calculate residual errors for two segments separately and obtain two different T statistics, T_1 and T_2 . If two segments have different lengths, calculating statistics differently yields better results. For a good fit, the sample variance should be small. A large variance is an indication of bad fitting. T statistics cannot handle the variance very well. The ratio of variance values for each segment could be a good indication of a good fit as well. If the T statistics and the ratio of variances follow the same F distribution, then these two segments are merged.

To measure the goodness of fit, a Chi-Square test could be performed on the residual space. This test, which checks whether the distribution has zero mean, does not yield a reliable measure in the presence of high amounts of noise. Fitzgibbon *et al.* propose the run-distribution test to measure the goodness of fit (Fitzgibbon, et al., 1999). They first calculate the mean of the distribution. For the distributions with zero mean,

they build a histogram to keep the sign distribution of the residual space, then compare the distribution of histograms to the probability distribution function of the true run distributions. Their measure gives better results than the Chi-Square in the presence of high levels of noise. They also consider the sums of variances to segment the curves at the critical points. The sum of variances detects the abrupt changes in the residual space and therefore they segment the curves from the points generating a steep change.

2.7 Discussion of the Limitations of Related Work

Coronal loops detection studies ((Lee, et al., 2006), (Aschwanden, et al., 2007), (Inhester, et al., 2007), (Biskri, et al., 2010)) have concentrated on highlighting the loops on given images and validated their methods on only one image, which cannot confirm the reliability and generalization ability of these algorithms. So far, there have not been any automatic techniques to detect coronal loops. Other solar events exhibit different characteristics from coronal loops and were typically detected with the help of thresholding or region growing based segmentation techniques. After obtaining regions, researchers validate the events using the features of these regions. Lastly, not many studies have considered classification techniques to detect solar events.

Curve tracing can be stated as an optimization problem and in the existence of noise and gaps, reaching the optimal solution is much harder. Most tracing algorithms (Steger, 1998; Sargin, et al., 2007; Raghupathy, et al., 2004) offer local solutions and miss the global optima. Methods (Steger, 1998; Sargin, et al., 2007) based on local gradient information often fail at the junctions.

Since coronal loops are semi-elliptical, we examined ellipse fitting methods and ellipse detection based on the Hough Transform. Since our ellipses are not complete and

not perfect, we favor using ellipse fitting over Hough based ellipse detection. Some drawbacks of ellipse detection based on the Hough transform are: (i) the method fails in case of the existence of noise in the image, since even if there is no ellipse in the image, noise points could give rise to spurious ellipses as an output of the Hough transform, (ii) the method require perfect and closed ellipses, for semi-ellipses or imperfect ellipses, the method fails, (iii) in case of the existence of various sizes of ellipses in the image, finding one good threshold value to use in the Hough space is also difficult.

We presented several features including statistical, histogram of gradients, edge histograms, curvature scale space, and so on. We extract those features to solve the coronal loop detection problem. Since, the existing features were not sufficient for our problem, we designed new features. We presented how classifiers work and how they evaluate the results. We described all the classifier techniques that we have investigated for the solution of our problem.

We presented the contour grouping studies along with perceptual organization and grouping measures. Most of the studies (Ren, et al., 2005) target the object detection problem and propose algorithms for close curves, whereas in our study, we propose a system to detect open curves from clutter, a more challenging problem. Some of the current measures (e.g., proximity, smoothness) are still good measures for our problem. However, we need to define new measures to tackle the clutter and obtain semi-elliptical open curves. Almost all of the existing ellipticity or circularity measures were designed for close shapes, however in our contour grouping study, we deal with open curves, and we thus need to define ellipticity and circularity in a different way.

3 CORONAL LOOP DETECTION FROM THE SOHO/EIT IMAGE COLLECTION

“What we observe is not nature itself, but nature exposed to our method of questioning.”

~Heisenberg

In this chapter, we describe a procedure to automatically retrieve solar images with coronal loops from the SOHO/EIT image database. We developed two different approaches to solve the coronal loop detection problem: A block-based approach and a contour-based approach. In the early period of this dissertation (as described in Section 3.1), we had concentrated on identifying loop existence from fixed sized blocks, where we first divide the out of disk region of the Sun into fixed sized blocks and assign a label to each block as “*Loop*” or “*Non-Loop*”. Then we extract features from these labeled blocks and use them to train classifiers. When trained and tested on an independent set of raw EIT images, we achieved 65% precision and 67% recall from the best classifier result with the best feature subset.

Later, we investigated methods to clean the images to reduce noise and instrument defects by using the IDL ssw solar software (Handy, 1998). We also investigated different features (i.e., histogram of gradients, eigenvalue histograms, histogram of

second order derivatives, etc.) which are extracted from gray level blocks. With the cleaned images and new features, we achieve 63% precision and 79% recall. This approach is described in Section 3.2.

In the second phase, we propose a contour based approach which concentrates on individual contours rather than blocks. In this approach, we extract contours from an image strip around the Sun and label the contours as “*Loop*” or “*Non-Loop*”. Then we extract contour features and feed them to classifiers. Despite many challenges related to the coronal loop characteristics, we obtained promising results, namely 85% precision and 83% recall in loop retrieval. Compared to the block-based approach, the accuracy and performance of the contour-based approach are significantly better. We describe this approach in Section 3.3.

By using the best training model, we developed both an offline and a web-based coronal loop image retrieval tools that can separate images with loops from images without loops. These tools are presented in the end of each sub-section.

3.1 Block-based Solar Loop Mining Approach on Raw Images

The general structure of the block-based approach is demonstrated in *Figure 1-12*. We first download FITS images in the 171 Å wavelength from NASA’s EIT repository (SOHO) because this wavelength shows the coronal loops better than the other wavelengths (such as 191Å), due to their specific temperature range. These images are 1024x1024 in size and consist of gray level intensities. The training data set was initially prepared by astrophysics experts who marked each coronal loop in the downloaded solar

images by enclosing it within a minimum bounding rectangle whose coordinates are saved into the FITS header.

Image preprocessing techniques are then applied on the images to improve their quality, which will be described in Section 3.1.1. After image preprocessing, we divide the out-disk region into blocks and label them, as will be described in Section 3.1.2. The investigated features will be described in Section 3.1.3. We train several classifiers to learn the characteristics of coronal loops, as elaborated in Section 3.1.4 along with the results to validate our methods. We then evaluate the investigated features in Section 3.1.5 and analyze the effect of the different solar cycles on coronal loop mining in Section 3.1.6. Finally, we test the developed tool on unseen data in Section 3.1.7.

3.1.1 Image Preparation

The SOHO/EIT images can contain noise and artifacts resulting from instrumental defects, including an image wide grid artifact. To remove outliers, we first used the wavelet transform with the Daubechies second order wavelet family and soft thresholding. Even though this method was able to remove pixel level noise, it kept the bigger specks. At the same time, wavelet denoising caused data loss in the top part of loop structures. Therefore we resorted to an outlier removal technique which replaces a pixel with the median of its surrounding pixels if the pixel value deviates from the median by more than a certain threshold value. Since this method deals with only outliers, it does not cause data loss in other points and yields a higher resolution compared to standard denoising techniques. *Figure 3-1 (b)* shows the result of outlier removal on the image in *Figure 3-1 (a)*.

Figure 3-1 Image preprocessing techniques: (a) Original SOHO/EIT image in 171 Ångstroms. (b) Image after speck removal. (c) Image after background subtraction. (d) Image after binarization and global thresholding

Since loops tend to be embedded within bright regions, we need to separate loop structures from the background using background subtraction (Sternberg, 1983) which is a process of separating foreground objects from the background. This process is widely used to remove smooth continuous backgrounds from medical images or detect moving

objects from video scenes. The background image can be created using different methods such as the wavelet transform, curvelet transform, or “rolling ball” algorithm.

Figure 3-2 Background subtraction. (a) Raw solar image region with a coronal loop. (b) Background image created using the rolling ball algorithm (Ball size = 5). (c) Subtracted image (Background image is subtracted from the original image)

In the “rolling ball” algorithm (Sternberg, 1983), a local background value is calculated for every pixel by averaging over a very large ball around the pixel. This value is subtracted from the original pixel value to remove background intensities. Since background subtraction considers every single pixel’s intensity value, we can still preserve loop structures without data loss while we are getting rid of the background intensity. From this aspect, background subtraction is more suitable for our study compared to local or global thresholding which may easily cause irreplaceable data loss. *Figure 3-1* (c) shows the entire image after background subtraction. *Figure 3-2* shows the loop region, background image obtained by the rolling ball algorithm, and the subtracted image which is obtained by subtracting the background image from the original image.

We also perform binarization using the Sobel edge detector (Gonzalez, 2007). First, we convolve the image with the Sobel edge mask and then perform global thresholding which removes most of the undesired patterns. The global thresholding

value (τ) is selected using the magnitude of gradients. We calculate the magnitude of the gradient for each pixel and take the average of the magnitudes as given in Eq. (3-1). We replace the values smaller than τ with 0 and the values greater than τ with 1 on the convolved image. *Figure 3-1 (d)* illustrates the binary image.

$$\tau = \frac{1}{MN} \sum_{x \in M} \sum_{y \in N} ((\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2) \quad 3-1$$

3.1.2 Block Extraction and Labeling

We divide the solar image into several small blocks which will be used as the basic units of analysis. A block is defined by three values: width (W), height outside the solar disk (H), and height inside the solar disk (L). Although we have experimented with several schemes to determine the optimal width and height based on the size of the marked regions, we noticed that a fixed-sized block was actually preferable especially due to the diverse loop sizes in the data (i.e. there is really no single optimal value). We selected fixed block dimensions considering the average sizes of the loops in the given examples.

Figure 3-3 Extracting out-of-disk blocks

During block extraction, the solar image is repeatedly rotated by θ° at a time, and a block of size $W \times (H+L)$ is extracted from the upper middle of the image. Thus the arch of the loop was kept at the top in all blocks. As shown in *Figure 3-3*, blocks may overlap with each other to increase the coverage of an entire loop by a single block. The overlap ratio (ρ) is determined experimentally by considering the heap memory size available on the computer and the average loop coverage among the solar images.

If the overlap ratio (ρ) and width (W) are given, we can calculate the rotation angle θ according to *Eq. (3-2)* and the number of blocks to be extracted (N) according to *Eq. (3-3)*.

$$\theta = (1 - \rho)2 \sin\left(\frac{W}{2R}\right) \quad 3-2$$

$$N = \frac{2\pi}{\theta} \quad 3-3$$

We experimented with different block sizes and found that a size of 110×110 gives the best loop coverage. Therefore, we extracted blocks of this size. For the overlap ratio (ρ), we used 0.6. With these parameters, we extract between 53 and 56 blocks from each solar image. This number changes due to the changing value of the visible solar radius in the given solar images. This value is embedded in the metadata of the downloaded FITS images.

From each image, we extract two sets of blocks: gray-level blocks and binary blocks. We extract different sets of features from each type of blocks. We extract the gray-level blocks from the background subtracted images as shown in *Figure 3-1 (c)* and binary blocks from the binary images as shown in *Figure 3-1 (d)*.

To construct the training data set, the extracted blocks are labeled as either containing solar loops (i.e., *Loop* class), or not (i.e. *Non-Loop* class). We consider a block to be in the *Loop* class if its intersection with an expert-marked loop region is higher than a certain percentage (in our case 70%). Thus, if an image contains a loop, then that loop is generally spread out over 2 to 4 blocks that will be labeled as belonging to the “*Loop*” class. *Figure 3-4* shows an example of a loop spreading over four consecutive blocks. The remaining blocks (typically, approximately 50) from an image are labeled as part of the “*Non-Loop*” class. Therefore the number of “*Non-Loop*” blocks is very high compared to the number of “*Loop*” blocks.

Figure 3-4 One loop region spreads out over consecutive blocks

To show the blocks for an image, we have developed a tool that displays block regions along with their labels. If there are any mislabeled training blocks as a result of automatic labeling, we correct them using this tool. The extracted gray-level blocks and binary blocks are shown in *Figure 3-5*. This tool is also useful to understand the characteristics of blocks in different classes, and we have used it to identify misclassified blocks at the end of classification.

(a)

(b)

Figure 3-5 A snapshot of the training label correction tool which is developed in JAVA. The red underlines indicate the “Loop” blocks while the gray underlines indicate “Non-Loop” blocks. (a) gray-level blocks after background extraction, (b) binary blocks after binarization

3.1.3 Feature Extraction

In the intensity level blocks, in addition to loop structures, there are other kinds of grid artifacts and noise as illustrated in *Figure 3-6*. These shapes make our feature extraction more complicated. Because of their distinct characteristics, we extract different sets of features from the gray level and binary blocks.

Figure 3-6 Loop blocks: (a) and (c) Gray level blocks, (b) and (d) Binary blocks

3.1.3.1 Statistical Features from Gray Level Blocks

Table 3-1 lists the following *statistical features* (Gonzalez, 2007) which are extracted from the *intensity (gray)* level blocks: *Mean*, *Standard Deviation*, *Smoothness*, *Third moment*, *Uniformity*, and *Entropy*.

Table 3-1 Statistical features

Feature Name	Description
Mean	A measure of average intensity
Standard Deviation	A measure of average contrast
Smoothness	A measure of the relative smoothness of the intensity in a region
Third Moment	A measure of the skewness of a histogram
Uniformity	A measure of the uniformity of intensity in the histogram
Entropy	A measure of randomness

3.1.3.2 Specialized Edge Related Features from the Binary Blocks

The Hough Transform (Duda, et al., 1972) is a well-known method for line and curve detection from images. In our study, we assume that the number of lines is equal to the number of separate dense regions in the Hough Space as was done in (Illingworth, et al., 1988; Svalbe, 1989).

We perform quantization in the Hough space to save in memory and time requirements and to handle noise and imperfect lines in the blocks. As mentioned in (Illingworth, et al., 1988), the parameters of a line can be estimated more accurately using a finer quantization of the parameter space. However, a coarser quantization is better for noise tolerance. Since our case fits the second type, we apply a coarser quantization in the θ and ρ coordinates.

Figure 3-7 (a) The length of the longest line (b) The number of edge pixels

The global maximum in the Hough space represents the longest and strongest line in the image. Thus, we can estimate the length of the longest line in the image from the number of points in the highest peak of the Hough Space. Figure 3-7 (a) shows that despite the overlap, the lengths of the longest lines in the *Loop* blocks and *Non-Loop* blocks are distributed in different value ranges. Therefore we use the length of the longest line as well as the total number of lines in the blocks as features in classification. We also

use the number of total edge pixels in the image, which is typically higher in *Loop* blocks than total pixels in *Non-Loop* blocks as shown in *Figure 3-7 (b)*.

Table 3-2 Hough Transform based features

Feature Name	Description
Number of Lines	Number of dense regions in the Hough Space (higher than a threshold)
Length of Longest Line	Number of points in the global maximum of the Hough Space
Number of Edge Pixels	Total number of pixels on all kinds of edges

The *orientation* of the lines in the blocks seemed to provide another promising feature. For this purpose, the general Edge Histogram Definition (EHD) (Won, et al., 2002) can estimate the number of horizontal edges, vertical edges, 135° edges, 45° edges, and non-directional edges (i.e. none of the above). However, applying the standard method for calculating EHD descriptors did not give good results in our case because loop edges do not tend to exactly match the straight horizontal, straight vertical, straight 45°, and 135° lines. This caused an underestimation of the first four types of edges and overestimation of the non-directional edges. Thus, we resort to a more flexible interval-based angle mapping based on the line orientation (θ coordinate) estimated from the Hough Space.

Table 3-3 Hough Transform based EHD features

Feature Name	Description
Number of Vertical Edges	Number of lines with angle between 80° and 100°
Number of Horizontal Edges	Number of lines with angle either between 0°

	and 10° or between 170° and 180°
Number of 45° Edges	Number of lines with angle between 35° and 55°
Number of 135° Edges	Number of lines with angle between 125° and 145°
Number of Non-Directional Edges	Number of lines with angle which does not match any of the above criteria

3.1.3.3 Spatial Features

We observe that most edges in the “*Non-Loop*” blocks tend to be located in the bottom half of the block, whereas the edges are located in the top half of the block in the case of *Loop* blocks. Therefore we decided to consider the spatial edge distribution *within* the blocks to extract additional spatial features. For this purpose, a block is divided into four horizontal bands and the number of edge pixels is counted in each band as illustrated in Figure 3-8.

Figure 3-8 Four bands in (a) a *Non-Loop* block (b) a *Loop* block

Table 3-4 Edge based spatial features

Feature Name	Description
First Band Edges	Number of edge pixels in the first band
Second Band Edges	Number of edge pixels in the second band
Third Band Edges	Number of edge pixels in the third band
Fourth Band Edges	Number of edge pixels in the fourth band

3.1.3.4 Curvature Features from the Binary Images

Since coronal loops tend to have elliptical shapes, we have also attempted to apply Hough Transform based ellipse detection methods (McLaughlin, 1998; Duda, et al., 1972; Tsuji, et al., 1978). We implemented the random ellipse detection methodology (McLaughlin, 1998) to determine the ellipse parameters. However since most of our loops are not perfect ellipses, the random point selection often led to the incorrect center points. In particular, for near-positive *Non-Loop* blocks, as shown in *Figure 3-9* (b), we obtained a similar number of ellipses as in the *Loop* blocks shown in *Figure 3-9* (a). Noisy points also cause the overestimation of the dense regions which resulted in overestimating the number of ellipses in the blocks.

Figure 3-9 (a) A loop block. (b) A near-positive non-loop block (c) A small sized imperfect loop (d) A medium sized half loop

Some other problems faced during ellipse detection were that most loops are not perfect elliptical shapes, i.e. they tend to be asymmetric or incomplete as illustrated in *Figure 3-9* (c) and (d). Hence there is no optimal major axis length and minor axis length (a and b) range for the coronal loops. This makes it difficult to find an optimal threshold to decide whether a dense region in the Hough Space corresponds to a genuine loop.

Due to the non-promising Hough-based ellipse detection results, we resorted to curvature based features. However, the defective and noisy structures of the blocks does not allow the Curvature Scale Space descriptors (Mokhtarian, et al., 1996) or B-Spline

curve representation (Cham, et al., 1999) to discriminate between “Loop” blocks and “Non-Loop” blocks. This has motivated us to develop a new curve tracing algorithm and curvature strength features that specifically address the defective and noisy nature of loop shapes.

Figure 3-10 Curves shown in red in (b) and (d) embedded in the noisy regions of (a) and (c).

In most curve tracing algorithms, the direction of the previous points plays an important role in selection of the next point while tracing a curve (Raghupathy, et al., 2004; Lee, et al., 2006; Steger, 1998). Choosing the best next continuation point from noisy regions can pose a big challenge as can be illustrated by the curves embedded in the noisy regions of Figure 3-10. Previous curve tracing algorithms (Raghupathy, et al., 2004; Lee, et al., 2006; Steger, 1998; Sargin, et al., 2007; Cheng, et al., 2004) do not address the difficulties of curve tracing in noisy regions. Figure 3-11 shows various examples of challenging cases. Let us assume that the black pixels have been traced so far and that the algorithm is at pixel *E*. The gray pixels with question marks show the

candidate pixels for pixel E . Selecting different points among this candidate list will lead to different curves.

Figure 3-11 Black points are the traced curve so far. E is the current point. The gray points with question marks are candidate points. The problem in curve tracing is to determine which point should be selected to obtain the correct curve?

In our work, we calculate the relative direction of the next point based on the current point according to the chain code orientations shown in *Figure 3-12*, that we have adapted from Freeman's chain code (Freeman, 1961) in order to reduce the possibility of selecting an undesired point. In our algorithm (**Algorithm 3-1**), we save the direction of changes of the curve at each point in a direction vector, D . Thus, the average orientation of the curve in the last k points in D gives the path of the curve. All of the candidate points in the search space are stored in a vector C . **Algorithm 3-1** shows the steps of our curve tracing algorithm.

Figure 3-12 Directions relative to the center point

To find the major curves in a region, the selection of an appropriate starting point is also important. We start to trace a curve from the top-left edge pixel in the image, and

add other points to the curve structure based on the curve direction and distance between the last point on the curve and the next candidate point. This process is repeated until there are no more close points in the direction of the last portion of the curve structure. After adding all the points in one direction, we apply the same process in the opposite direction with the same starting point to include any points that were not traced before, but that belong to the same curve. At each step in the selection, the search space of the candidate points is not confined to the immediate 8-neighborhood of the current point, but also includes points in the $((2*gap+1)^2-1)$ -neighborhood. This increase in the size of the search space is an attempt to handle the broken nature of the loop structures. Different *gap* values were investigated, while trying to maximize the coverage of broken lines, while minimizing ventures inside noisy regions.

The criterion to select the best next point is estimated based on the Euclidean distance and the average direction change. In Eq. (3-4), we compute the weight of each candidate point C_i with respect to the last point on the curve, P_t . P is a list of traced points, and t is the number of points on the traced curve so far. In Eq. (3-5), the direction of candidate point C_i is compared to the average directions of the last k points of the curve structure.

$$w(C_i) = dis(C_i, P_t) + dis(C_i, P_t) * dirDif(C_i, D)^2 \quad 3-4$$

$$dirDif(C_i, D) = \left| dir(C_i) - \frac{1}{k} \sum_{j=t-k}^t dir(D_j) \right| \quad 3-5$$

Algorithm 3-1 Algorithm for Curve Tracing in Noisy Images

```
1.  $E[]$  = All white pixels
2. while no edge pixels remain in  $E$ 
  do
    2.1 Start forming a new curve structure  $P$  from the first element of  $E$ .
    2.2 Starting_Point =  $E[1]$ 
    2.3 Current_Point = Starting_Point
    2.4 while (1) //Trace curve from Starting_Point
      do
        2.4.1  $C[]$  = Find candidate points in the  $((2*gap+1)^2-1)$ -neighborhood of the
          Current_Point.
        2.4.2 Add Current_Point into curve structure  $P$ 
        2.4.3 Remove Current_Point from  $E$ .
        2.4.4 If no candidate points found then break the loop.
        2.4.5 For each candidate_point in  $C$ 
          do
            distance = Euclidian Distance between candidate_point and Current_Point
            direction = Find relative direction between candidate_point and Current_Point
            (see Figure 3-12)
            Calculate direction_difference using Eq.(3-5)
            Calculate weight of candidate point using Eq.(3-4)
          end for
        2.4.6 If minimum weighted candidate point < weight_threshold, then
          2.4.6.1 Current_Point = minimum weighted candidate point
          2.4.6.2 Add direction of Current_Point into direction array  $D$ 
        end if
      end while
    2.5 Trace curve from Starting_Point to detect the other half of the curve.
    2.6 Combine two halves
  end while
```

The complexity of the algorithm is proportional to the number of starting point and the size of the gap. Let M be the number of starting points and $CurveLen$ be the length of the traced curve. The algorithm complexity can be expressed as $O(M.CurveLen.gap^2)$.

One improvement to our curve tracing is considering the further points to make the selection. If there are more than one candidate points having the same weight value or if there are other candidate points having a very close weight value to the weight threshold, then we start a search for those points. Assume that we have n candidate points with similar weight values under the weight threshold. We trace all these candidate points for a further k points to see their orientation tendency. The search tree of n candidate points is shown in *Figure 3-13*. Selecting the k value is also application dependent. For our case, we selected a value of $k = 10$. After tracing k further points for n candidate points, we compute the orientation difference between the further traced segments and the last part of the previously traced curve structure by using *Eq. (3-6)*. For the i^{th} candidate point, we take average of the orientation change of the further traced k points. We subtract the average orientation of the last k points in the traced curve so far (*past-k average orientation*) from the *further-k average orientation*.

Figure 3-13 Search tree for n candidate points

The direction changes in the further curve segment are kept in the *FD* array. During selection, we also consider the length of the further traced curve segments. Longer further curve traces are more promising than the shorter ones. If the length of the curve segment

is too short (e.g., 2 or 3 points long), then we penalize this curve segment by using Eq. (3-7) where β is used for adjusting the weight of the orientation difference.

$$furtherDif(i) = \left| \frac{1}{k} \sum_{j=1}^k dir(FD_j) - \frac{1}{k} \sum_{j=N-k}^N dir(D_j) \right| \quad 3-6$$

$$furtherWeight(i) = \frac{\beta * furtherDif(i)}{length(CurveSegment(i))} \quad 3-7$$

We change the **Algorithm 3-1** and add the following condition after step 2.4.6:
“If there are more points less than the threshold, then trace candidate paths and pick the next point providing the longest curve and the smallest orientation difference Eq. (3-7), otherwise pick the next point with the smallest weight.”

Figure 3-14 Success of the further tracing approach

With this change, we improve the accuracy of the curve tracing results. The worst case of the algorithm complexity is $O(M.CurveLen.gap^2.n.k)$. But, further tracing occurs a few times along the curve. Therefore, the real algorithm complexity is $O(M.CurveLen.gap^2 + n.k)$. Figure 3-14 shows an incorrect curve tracing result with **Algorithm 3-1**. At the junctions, the algorithm picked the point with the smallest weight and followed an undesired path. After applying the further tracing search, we were able to obtain the desired path from the image. Figure 3-15 shows some curves that we traced using our final method.

Figure 3-15 Tracing the curves from the cluttered regions: (a, c, e, g) are the cluttered images, (b, d, f, h) are the automatically traced curves from these cluttered regions, respectively.

In our study, we apply the curve tracing algorithm on each block without knowing whether a coronal loop is present. Thus, the curve tracing algorithm will inevitably attempt to extract some curves from the “Non-Loop” blocks as well. Since we want to know which regions really contain a loop, we need to measure the curvature degree of each extracted curve structure. Curvature occurs when two line segments meet and form an angle in a digital arc sequence (Lee, et al., 1993). If a digital arc sequence is segmented into line segments based on its corner points, and the exterior angle between two consecutive line segments is found, then the curvature of all the line segments can be calculated by averaging the exterior angles along the curve (Freeman, et al., 1977; Pineda, et al., 1983; Haralick, et al., 1992). Corner points can be found using the Curvature Scale Space (CSS) technique which finds an optimum of T-corners or peak points in rounded corners (Mokhtarian, et al., 1996).

In our work, we divide the direction list D into segments based on the identification of sharp direction changes. Since D stores the direction changes along a curve, there is no need to calculate the exterior angle between the line segments of the arc. Also, there is no need to fill the gaps along the curve as in the CSS feature (Mokhtarian, et al., 1996). We start our line segmentation from sign change junctions and get positive and negative chunks. We then divide each chunk into different sub-segments whenever a new direction point is not close to the average of the directions in the segment so far. In some cases, a single point may be distorted, and the next point may maintain the continuity of the angle of the previous segment. To avoid segmenting a line at incorrect places, we compare the next point to the average directional change of the segment whenever a point is not close to the average of the segment. *Figure 3-16* shows a sample curve and its segments, with the segmentation $S = [(-2, -2, -1, -2), (0, -1, -1, -1, -1), (1, 1, 2, 1, 2)]$ according to the direction codes from *Figure 3-12*.

Figure 3-16 Example of an extracted curve

After segmenting D , we calculate the average angle of each segment and store these values in an angle list T , then sum the average angle of differences between all pairs of adjacent segments, and multiply it by 45° to map the chain code direction to an angle. Then we calculate the average angle change along the curve using *Eq. (3-8)*, where n is the number of segments in segment list, S . If there is a single segment in S , then δ is assigned the value 0° . To reduce the influence of shorter segments, we also take into

account the length of the segments as shown in Eq. (3-8). Shorter segments will have less effect on the average curvature value.

$$\delta = \frac{45}{n-1} \sum_{i=2}^n \frac{\min(|S(i)|, |S(i-1)|)}{|D|} |T(i) - T(i-1)| \quad 3-8$$

In the case of coronal loops, the optimal curve tends to be semi-elliptic, and the positive and negative angles tend to be distributed evenly (e.g. Figure 3-17 (a)). We name these arcs *two-sided* arcs. If all angles are of the same sign, then the arc is not semi-elliptically shaped (e.g. Figure 3-17 (b) - (d)). We call this kind of arc a *one-sided* arc. While *two-sided* arcs are considered more important than one-sided arcs, in one-sided arcs, the presence of rounded corners or strong angle differences along the curve still indicates some curvature strength. If there are no significant angle differences in *one-sided* arcs, then their curvature strength will be close to 0 (e.g. Figure 3-17 (d)). In the case of *two-sided* arcs, the arcs with a smaller radius should have less curvature strength compared to those having a large radius (e.g. Figure 3-17 (a) versus (c)).

Figure 3-17 Curvature strengths for some extracted curves

The *radius* can be computed from the direction change list D by looking at changes in the x -direction of the segments before and after the peak point. The amount of sign change can be estimated using Eq. (3-9), in which θ^+ is the number of positive

angles along the edge, θ^- is the number of negative angles, and l is the number of elements in D .

$$\beta = (l - |\theta^+ - \theta^-|) / l \quad 3-9$$

Finally, we calculate the “Curvature Strength” of D by adding the weighted radius, δ (average angle change), β (sign distribution), l (number of element in D) and n (number of segments in S) as shown in Eq. (3-10).

$$\text{Curvature-Strength}(D) = w_1 * \text{Radius} + w_2 * \delta + w_3 * \beta + w_4 * l + w_5 * n \quad 3-10$$

We set the weights to map strong curve shapes to the [50-100] range, weaker curve shapes to the [15-50] range, and other non-curved shapes to the [0-30] range. Figure 3-18 shows the curvature strength value distribution over 400 *Loop* blocks and 400 *Non-Loop* blocks. This plot shows that the curvature strength feature is promising for distinguishing the *Loop* blocks from the *Non-Loop* blocks, with most of *Non-Loop* blocks ranging in the [0-30] range, whereas *Loop* blocks range in the [15-100] range.

Figure 3-18 Curvature strength feature for *Loop* blocks versus *Non-Loop* blocks

In addition to the curvature strength feature, we have investigated computing the peak angle (α) in an alternative way by computing the angle between two segments when they intersect at the peak point. The simple formula for the peak angle is $\alpha = \theta_1 + \theta_2$ where θ_1 is the average angle of the segment to the left of the peak point, and θ_2 is the average angle of the segment to the right of the peak point. *Figure 3-18* shows the peak angle and curve distance on a curve.

Figure 3-19 Peak angle and curve distance measures for a curve

If there is only one segment such as the one shown in *Figure 3-20* (a), then α will be 180° . If the sides of the peak point have the same sign distribution as in *Figure 3-20* (b), then α will be an obtuse angle, otherwise α will be an acute angle. We also keep the following features from the traced curve: The Euclidean distance (d) between the two end-points of the traced curve, and the length (l) of the traced curve.

Figure 3-20 Peak angles for different cases: (a) $\alpha = 180^\circ$, (b) $\alpha = 135^\circ$, (c) $\alpha = -15^\circ$

Table 3-5 Curvature features

Feature Name	Expression
Curvature Strength	A measure between 0 to 100 of how curvy the traced curve is
Peak Angle	Angle between two segments when they intersect at a peak

	point
Curve Length	Total number of points on the traced curve
Curve Distance	Euclidean distance between the endpoints of the traced curve
Sign Distribution	Sign distribution along the curve

3.1.4 Training Classifiers

We present the results for a training data set consisting of 150 images that have been labeled by marking a minimum bounding rectangle around the loop shapes. The solar images were from the years 1996, 1997, 2000, 2001, 2004 and 2005. After block extraction and automatic labeling, we obtained 403 “*Loop*” blocks and 7950 “*Non-Loop*” blocks. Then we extracted features from both types of blocks as described in Section 3.1.3, and trained classifiers and evaluated them using 10-fold cross-validation. We resorted to a supervised learning strategy that uses labeled examples of blocks with and without loops to build a prediction model that can detect the occurrence of loops based on the extracted features. **Table 3-6** shows the classifier techniques that were investigated.

Table 3-6 The investigated classifiers

Classifier	Abbreviation	Brief description
Adaptive Boosting (Shapire, et al., 1999) (Using C4.5 base classifier)	Adaboost	Sequentially learns an ensemble of C4.5 base learners by focusing on examples that are hard to classify
Naïve Bayes (Duda, et al., 2001)	NB	Probabilistic (Bayesian) classifier
Multilayer Perceptron (Rumelhart, et al., 1986)	MLP	Neural Network Classifier trained using back propagation
C4.5 Decision trees (Quinlan, 1993)	C4.5	Learns a tree based classifier built with the most predictive attributes
Repeated Incremental Pruning to Produce Error Reduction (Cohen, 1995)	RIPPER	Learns an optimal set of rules that cover the training samples

<i>K</i> -nearest neighbor classifier (Duda, et al., 2001) (<i>K</i> = 5 determined based on trial and error)	K-NN	Lazy Instance based classifier
---	------	--------------------------------

In **Table 3-7**, we list the precision and recall values obtained from the different classifiers for each feature set. The precision and recall values that resulted in the best F1-Score (harmonic mean of precision and recall) are shown in bold in the table. By looking at these results, we observe that the statistical features give low recall, while the Hough-based features give better precision and recall than statistical features, spatial features and curvature features perform similar to Hough-based features. When we combine all features together, Adaboost, MLP, and RIPPER yield very similar results, with their F-score values almost the same, and with the best recall value around 69% and the best precision value around 62%. *Figure 3-21* shows the ROC curves of the classifiers, showing that the Adaboost classifier reached the best precision-recall pairs very quickly. Based on the ROC curve, we chose to use Adaboost for the next stage in our decision making, which is to retrieve the images containing coronal loops from an unlabeled collection.

Table 3-7 Block based precision and recall values of various classifiers

Classifier	Features in Table 1 (Statistical)			Features in Table 2 +Table 3 (Hough-based)			Features in Table 4 (Spatial)			Features in Table 5 (Curvature)			All Features		
	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre	Rec.	F1	Pre.	Rec.	F1
Adaboost	0.38	0.24	0.29	0.46	0.45	0.46	0.49	0.48	0.49	0.50	0.48	0.49	0.64	0.67	0.66
NB	0.18	0.08	0.11	0.52	0.60	0.56	0.52	0.56	0.55	0.37	0.87	0.52	0.36	0.77	0.49
MLP	0.54	0.22	0.31	0.5	0.58	0.54	0.51	0.42	0.46	0.57	0.56	0.57	0.62	0.7	0.66

C4.5	0.45	0.20	0.28	0.51	0.51	0.51	0.51	0.55	0.53	0.5	0.55	0.52	0.6	0.63	0.61
RIPPER	0.48	0.18	0.26	0.56	0.49	0.52	0.51	0.49	0.5	0.52	0.57	0.55	0.62	0.7	0.66
K-NN	0.42	0.2	0.27	0.49	0.44	0.47	0.54	0.44	0.49	0.56	0.48	0.52	0.64	0.61	0.63

Figure 3-21 ROC curves of all classifiers

The most important factor behind the low precision is the large number of near-negative “Loop” blocks and near-positive “Non-Loop” blocks. Figure 3-9 shows some samples of near-positive and near-negative blocks. Near-positive instances make up around 20% of all negative instances, while the near-negative instances constitute almost 40% of all positive instances. In addition to the data specific problems, another challenge to classification was the imbalanced distribution of the “Loop” versus “Non-Loop” instances, with the ratio of the positive (Loop) class (minority) to the negative (Non-Loop) class (majority) around 1 to 20. To summarize, the imbalanced class distribution and the high percentage of border-line instances make the classification task very difficult.

Figure 3-22 shows several misclassified blocks. In the false negative examples, we can observe that misclassified loops are generally defective (loops are discontinuous due to image cleaning or other reasons), small, or half of their complete length (the other half might be located another block). In the false positive examples, we can observe severe clutter which confuses edge histograms and Hough based features.

Figure 3-22 Misclassified blocks (a) False negatives, (b) False positives

3.1.5 Feature Evaluation

We evaluate the goodness of features by using the information gain measure given in Eq. (2-22). In our case, there are two classes: *Loop* and *Non-Loop*. We use the training data to calculate the information gain of each feature. The information gain confirms that the specialized loop features in Table 3-5 are more discriminative than the other extracted features shown in Figure 3-23.

Figure 3-23 Information gain values for the investigated features

We also use a greedy search algorithm (Vafaie, et al., 1994) to select an optimal subset of features for classification, then train classifiers on only the selected features which are: “Curvature Strength”, “Peak Angle”, “Curve Length”, “Number of Edge Pixels”, “Third Moment”, “First Band”, “Second Band”, “Third Band”, and “Fourth Band”. Using these features, we obtained 72% precision and 78% recall values from Adaboost. Even though these values are higher than the values obtained using all features, we notice an overfitting when we test the generated model on unseen testing data. Thus we ended up using the previous classifier model trained with all features because it was causing less overfitting.

3.1.6 Solar Cycle-based Experimental Results

Solar activities can be categorized into three cycles: the minimum cycle does not contain a lot of activity, and thus results in fewer loops on the corona; the maximum solar cycle contains a lot of activity including many loops, as well as other kinds of solar

events (e.g. solar flares and coronal mass ejections). Finally the medium cycle contains more activity than the minimum cycle and less activity than the maximum cycle. The years 1996 and 2005 were part of the minimum cycle period, while 2000 and 2001 fell in the maximum cycle period, and 1997 and 2005 fell in the medium cycle period. *Figure 3-24* shows an image from the minimum cycle in 1996 and another image from the maximum cycle in 2000.

Figure 3-24 Images from different cycles: (a) a minimum cycle image, (b) a maximum cycle image

Since it is trivial to automatically infer the solar cycle from the metadata contained in the header of each FITS image, we have attempted to train three different solar cycle- specific models, with each model trained using 60 images from the same cycle, and then tested each specialized model on a different test set containing 20 images (including 10 with loops and 10 without any loops) from the same solar cycle that was used for training. For comparison, a global model was also trained using all the images in all the cycles and tested with all the test images. **Table 3-8** shows the image-based precision and recall values of loop images and non-loop images for each cycle. For loop images, the lowest precision value among the three cycles occurs for the minimum cycle,

and the highest recall value occurs for the maximum solar cycle because there are more loop shapes in the training set of this solar cycle. As expected, we achieved a best trade-off between precision and recall when we used all the cycles to train one model.

Table 3-8 Cycle based testing results for image retrieval

Minimum Cycle		Medium Cycle		Maximum Cycle		All Cycles	
Pre.	Rec.	Pre.	Rec.	Pre.	Rec.	Pre.	Rec.
0.5	0.9	0.8	0.7	0.8	1	0.88	0.83

3.1.7 Testing Phase

To retrieve solar images containing loops from the EIT solar image repository (SOHO), a similar process to the training phase is applied on *unlabeled* test images. *Figure 3-25* shows the architecture of the testing phase. After image cleaning and feature extraction, we apply the top three classifier models (Adaboost, MLP, RIPPER) on the extracted features to generate the block labels in each image. Based on these block labels, a global decision is then made about whether the entire *image* contains a loop or not. The most accurate results were obtained from Adaboost which gave fewer false positives (non-loop regions predicted as loops) and higher true positives (loop regions correctly predicted as loops).

The final decision for an image is made based on the predicted labels of its blocks. If at least one block is predicted to be in the loop class, then the image is classified into the loop class, and we highlight all the predicted loop regions on that image with red rectangles. If several consecutive (neighboring) blocks are classified in the loop class, we

merge them into one large block and show their location using one big rectangle on the image.

Figure 3-25 General structure of the block-based testing phase

3.1.7.1 Image Retrieval Tool

We have developed an image retrieval tool in JAVA, where users can upload a set of solar images and the system separates the images containing loops from those without any loops. Users can then browse both categories of images and save the images containing loops in a directory. Figure 3-26 shows the user interface of the developed image retrieval tool.

To evaluate the final image retrieval system, we tested it on new unlabeled images from the same years as the training data. The testing set contained 100 images, half of which containing coronal loops. The final loop mining results are presented in **Table 3-9**, showing a precision of 78% and recall of 80% relative to the “Loop Image” class. Since separating *Non-Loop Images* from *Loop Images* accurately is as important as finding only loop images, we desire high precision and recall values in the *Non-Loop* class as well.

With high precision and recall values for both classes, we can conclude that our tool has succeeded for both types of images.

Figure 3-26 A snapshot of the developed loop mining tool interface with the red rectangle indicating a predicted loop region. Note that non-loop regions on the image are also eliminated correctly by the system.

Table 3-9 Confusion matrix for image based testing results

(precision = 78%, recall = 80%)

	Actual Loop Images	Actual Non-Loop Images	Total
Predicted Loop Images	40	10	50
Predicted Non-Loop Images	11	39	50
Total	51	49	100

Figure 3-27 illustrates some results obtained using our loop mining tool. If there is a loop in an image, the loop regions are located on the image. If there is no loop found in an image, then this image is included into the non-loop image list.

Figure 3-27 Test results on unseen data: (a, b) Images with loops. In (a), we can see one false positive region in the top-right red box. (c, d) Images without loops, correctly classified

3.1.7.2 Web Developments

We have developed a website that provides all the details about our project, along with a working prototype of the retrieval system, at the following URL:

http://webmining.spd.louisville.edu/Solar_Loop_Mining/Demo/interface.html

We prepared a web based image retrieval tool which aims at querying SOHO/EIT images according to their coronal loop existence on the outside of the solar disk. First we

downloaded images from the SOHO/EIT image collection and used our Loop Mining Tool with the previous years' data model. Since this is a small demo version, we provide results only for the following year and month intervals: July-1996, July-1997, July-1998, July-1999, July-2000, July-2001, July-2002, July-2003, July-2004, July-2005, July-2006, July-2007, July-2008. Thus, in this online user interface, the user can only browse results from these periods. By downloading our tool, users can try more images from periods that are different from the current collection. Some snapshots of the online system are shown in *Figure 3-28*.

(a)

(b)

(c)

Figure 3-28 Sample snapshots of the online image retrieval tool: (a, b) images with loops, (c) image without any loops

3.2 Block-based Approach on IDL-cleaned Images

Even though the block-based approach can separate images with loops from those without any loops with a certain level of reliability, the false alarm ratio is still hurting the reliability of the detection system. One of the biggest problems of the block-based approach on raw images is in handling the grid artifacts and other instrument related defects as shown in *Figure 3-29* (a). The grid pattern and noise make the feature extraction phase much harder and decrease the accuracy of the automated detection considerably.

(a)

(b)

Figure 3-29 Out of disk loop region (a) Raw image, (b) After cleaning with IDL

Following some discussions with members of the solar physics community at the Solar Image Processing Workshop 2008, we decided to apply the standard *eit_prep* procedure of the IDL solar software (ssw) library (Handy, 1998). The grid structures and other noise effects are reduced significantly after applying the *eit_prep* procedure as shown in *Figure 3-29* (b). After processing the images in this way, the extracted features behaved unexpectedly and our detection accuracy did not increase significantly as expected. In fact, our previous feature extraction and classification approach on the new properly cleaned images achieved a 56% F1-measure which is lower than obtained for the previous pre-processing approach. To improve the system, we designed new features using curvature histograms, eigenvalue statistics, and directional derivatives. With these newly proposed features, we increase the F1-measure to 70% as will be explained in the following subsections.

3.2.1 Image Preparation

After downloading FITS images from the EIT database, we clean them using the standard IDL *eit_prep* procedure to get rid of instrumental defects and grid artifacts. *Eit_prep* results in images without any grid artifacts, however specks and salt and pepper noise are still present in the images. The salt and pepper noise occurs due to a combination of Poisson photon noise, mostly Gaussian readout noise and noise coming from the flat-field and grid correction matrices. Note that since most of the EIT detector damage occurs at the limb, the noise tends to be highest in that region.

Figure 3-30 Image denoised with the Wavelet transform, note that the speckles are still present (in the circles).

To remove the specks which are noise structures that are bigger than 2×2 pixels, we first experimented with noise removal using the Wavelet transform with the Daubechies family, second order wavelet and soft thresholding. Even though wavelets were able to remove pixel level noise, they kept the bigger specks as shown in *Figure 3-30*. When we increased the threshold value in Wavelet denoising, we were able to get rid of bigger specks but we lost data from the top part of the loop structures as shown in *Figure 3-31*. Therefore, we resort to a median based outlier removal technique that replaces a pixel by the median of its neighboring pixels (within a radius of 2 pixels which creates a 5×5 window) if the pixel's intensity value deviates from the median by more than a certain threshold (in this study, the threshold is fixed at 50). Since this method only deals with big specks, it provides a higher resolution output compared to standard

denoising techniques such as median filtering. As shown in *Figure 3-32* (b), this technique succeeds in removing specks while retaining loop information.

Figure 3-31 Noise removal with the Wavelet transform: (a) Original image, (b) Denoised image with the Wavelet transform, note that the top part of the loop is lost

Figure 3-32 (a) An image segment obtained after *eit_prep* with circled significant specks, (b) image after removing specks

After removing significant specks, we apply Wavelet denoising to get rid of pixel-level noise. For wavelet denoising, we use the Symlet family of order 4, soft thresholding, and 2% coefficient retaining. *Figure 3-33* compares an image before and after smoothing a loop segment.

Figure 3-33 A zoomed loop segment (a) after removing outliers (b) after smoothing with Wavelet Transform

After smoothing, we desire to bring out coronal loops from the bright regions where they are embedded. Unlike the previous approach, we use the Wavelet transform to construct the background image to retain more loop points. We obtain the background image by performing the Wavelet transform using the Symlet family of order 4, with soft thresholding, 40% coefficient retaining. *Figure 3-34* shows the original image, background image, and the image obtained by subtracting the background image from the original image.

Figure 3-34 Background subtraction algorithm: (a) original image, (b) background image, (c) background image in (b) subtracted from original image in (a)

After background subtraction, we perform the block extraction described in Section 3.1.2. Similar to the previous approach, we binarize the image to extract different features. This time, we follow a different procedure to binarize the blocks. We perform a skeletonization which is to reduce all the forms in a block to lines without changing the essential structure of the forms. We first compute the mean value of a block and retain the points if the intensity level is greater than the mean value. We compare the intensity level of a point to its four cross-pair neighbors which are the horizontal pair, vertical pair, diagonal pair, and anti-diagonal pair. For a point (x, y) , the horizontal pair consists of the points at 0° or $(x+1, y)$ and 180° or $(x-1, y)$; vertical pair consists of the points at 90° or $(x, y-1)$ and 270° or $(x, y+1)$; the diagonal pair consists of the points at 45° or $(x+1, y-1)$ and 215° or $(x-1, y+1)$, and the anti-diagonal pair consists of the points at 135° or $(x-1, y-1)$ and 315° or $(x+1, y+1)$. If the intensity level of a point is equal to or greater than at least two of its two different cross-pairs, then we consider the point to be a skeleton point.

This method is slightly different from the classical edge detection such as the Canny, Sobel or Prewitt or skeleton extraction methods (Gonzalez, 2007). Checking whether the point's intensity is a maximum among its cross-pair neighbors allows us to

keep the most representative points in the central location of the forms in a block. With this simple method, we can discern loop structures and other forms much better. We also reduce the complexity of curve tracing by keeping the skeleton of the image instead of all pixels along with their intensity values. *Figure 3-35* compares the described method to a standard morphological thinning operator and the Canny edge detector (Gonzalez, 2007). The Canny edge detector brings out the boundaries of the forms as shown in *Figure 3-35* (c) while the morphological thinning method hurts the shape of the loop forms and connects close points as shown in *Figure 3-35* (b) . The binarization method brings out the general structure of the forms without causing any change in the original shape as shown in *Figure 3-35* (d).

Figure 3-35 Comparison of the binarization method used in this study to other standard methods. (a) original block, (b) after a standard morphological thinning operator, (c) after the Canny edge detector, (d) after the binarization method used in this study.

3.2.2 Feature Extraction

Similar to the previous approach, we extract different sets of features from gray-level blocks and binary blocks.

3.2.2.1 Statistical Features

As in the previous approach, we extract the statistical features (*Mean, Standard Deviation, Smoothness, Third moment, Uniformity, and Entropy*) from the gray-level blocks. These are the same features listed in **Table 3-1**.

3.2.2.2 Histogram of Oriented Gradients

The histogram of oriented gradients counts the occurrences of gradient orientation in localized portions of an image (Dalal, et al., 2005). The gradient defines the tangent at that point. The gradient direction is the normal to the level curve at that point, while the gradient magnitude measures the steepness of that ascent. In our problem, the gradient, gradient magnitude and gradient directions promise to be useful in loop characterization and detection. For each block, we accumulate the edge orientations in the region in an orientation histogram. The orientation of the gradient as follows: $\theta = \tan^{-1} \frac{G_y}{G_x}$. We first translate the orientation range from $[-\pi, \pi]$ to $[-180^\circ, 180^\circ]$. After that, we translate the range of the gradient from $[-180, 180]$ to $[0, 360]$ using,

$$\theta = \begin{cases} \theta, & \text{if } \theta \geq 0 \\ \theta + 360, & \text{if } \theta < 0 \end{cases}$$

After obtaining the orientation of gradients for each pixel, the orientations in the block are binned in the histogram. According to Dalal (Dalal, et al., 2005), using unsigned gradient orientations in nine histogram channels tends to perform best in image retrieval problems. We also kept 9 histogram bins in our problem as listed in **Table 3-10**.

Table 3-10 Histogram of Eigenvector Orientations

Feature Name	Description
Hogs-1	The number of points where $\theta \leq 40^\circ$
Hogs-2	The number of points where $40^\circ < \theta \leq 80^\circ$
.....	
Hogs-9	The number of points where $320^\circ < \theta \leq 360^\circ$

3.2.2.3 Directional Derivatives

A directional derivative in a single direction is interpreted as the rate of change in that direction. Second order directional derivatives are obtained by applying two first-order directional derivatives on an image. Second order derivatives highlight the loop points better than first order derivatives. Directional second order derivatives of a block are shown in *Figure 3-36*. Different second order derivatives highlight different directions in the image.

Figure 3-36 Second order directional derivatives of a loop block. Different derivatives highlight different oriented loop points.

Since different second order derivatives keep different loop points, we use the histograms of the derivatives as features. These are listed in Table 3-11.

Table 3-11 Second Order Derivatives Statistics

Feature Name	Description
Hist- I_{xx}	The number of points where $I_{xx} > \tau$
Hist - I_{xy}	The number of points where $I_{xy} > \tau$
Hist - I_{yy}	The number of points where $I_{yy} > \tau$

3.2.2.4 Eigenvalue Histograms

The calculation of the direction of a point is done using the Hessian matrix. The partial derivatives, I_{xx} , I_{xy} , and I_{yy} , are computed using partial differences after convolving the image with a Gaussian smoothing kernel. Gaussian smoothing is essential to remove noise from the image.

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix}$$

The eigenvalues and eigenvectors of the Hessian matrix have the following geometric meaning: the first eigenvector (the one whose corresponding eigenvalue has the largest absolute value) is the direction of greatest curvature (second derivative), the second eigenvector (which is orthogonal to the first) is the direction of the least curvature. The corresponding eigenvalues are the respective amounts of these curvatures. The eigenvectors of H are called principal directions.

The eigenvalues, λ_1 and λ_2 , measure the convexity and concavity in the corresponding eigen directions. A ridge is a region where $\lambda_1 \approx 0$ and $\lambda_2 \ll 0$. Elliptic points occur where $\lambda_1 * \lambda_2 > 0$. Hyperbolic points are the points where $\lambda_1 * \lambda_2 < 0$.

Considering the geometrical meanings of the eigenvalues of the Hessian matrix, we keep the eigen histograms listed in Table 3-12.

Table 3-12 Eigen-based Features

Feature Name	Description
Elliptic or Hyperbolic points	The number of points where $\lambda_1 * \lambda_2 > 0$ or $\lambda_1 * \lambda_2 < 0$
Eigen Distance	The number of points where $ \lambda_1 - \lambda_2 > \tau$
Eigen-Hist-Positive	Two bins for positive eigenvalues
Eigen-Hist-Negative	Two bins for negative eigenvalues

3.2.2.5 Curvature Histograms

The eigenvalues, λ_1 and λ_2 , are called principal curvatures and they are invariant under rotation (Wang, et al., 2008) and can be used to calculate the following metrics:

$$\text{Gaussian Curvature} \quad K = \lambda_1 \lambda_2$$

$$\text{Mean Curvature} \quad H = (\lambda_1 + \lambda_2)/2$$

$$\text{Curvedness} \quad C = \sqrt{(\lambda_1^2 + \lambda_2^2)/2}$$

For each block, we keep the maximum and minimum eigenvalues and calculate the Gaussian curvature, mean curvature and curvedness values (Wang, et al., 2008; Li, et al., 2004) based on the global maximum and minimum eigenvalues. In addition, we calculate the mean curvature, Gaussian curvature, and curvedness values for each point in the block and keep a histogram of these curvature values. Table 3-13 lists all the curvature related features. The threshold values (T_1 , T_2 , T_3 , T_4) for the histogram are found by examining the curvature distributions for both “*Loop*” blocks and “*Non-Loop*” blocks.

Table 3-13 Curvature based Features

Feature Name	Description
Gaussian Curvature (K)	$\lambda_1 \lambda_2$
Mean Curvature (H)	$(\lambda_1 + \lambda_2)/2$
Curvedness (C)	$\sqrt{(\lambda_1^2 + \lambda_2^2)}/2$
Mean Curvature Histograms	Two bins: One bin for the points where mean curvature where $H > T_2$ and one bin for the points where $T_1 < H < T_2$
Gaussian Curvature Histograms	Two bins: One bin for the points where $K < 0$ and one bin for the points where $K > 0$
Curvedness Histograms	Two bins: One bin for the points where $C > T_4$ and one bin for the points where $T_3 < C < T_4$

3.2.2.6 Hough-based Features

From the binary images, we extract Hough-based features (*number of lines, length of the longest line, number of edge pixels*) as was previously described in Section 3.1.3.2.

3.2.2.7 Spatial Features

From the binary images, we extract spatial features (*first band edges, second band edges, third band edges, fourth band edges*) as described in Section 3.1.3.3.

3.2.3 Classification Experimental Results

We present the results for a training data set consisting of 180 images which have been labeled by marking a minimum bounding rectangle around the loop shapes in solar images from the years 1996, 1997, 2000, 2001, 2004 and 2005. After block extraction and automatic labeling, we obtained 752 “*Loop*” blocks and 8,193 “*Non-Loop*” blocks.

Then we extracted features from both types of blocks, and trained the classifiers listed in **Table 3-6** and evaluated them using 10-fold cross-validation.

In **Table 3-14**, we compare the results of all classification methods on different feature groups. For each classification method, we show the Precision, Recall, and F1-Score measures. Statistical features give a maximum F1-Score of 59% using the RIPPER classification technique. The accuracy is the lowest compared to other feature sets. HOGs features achieve a 67% F1-Score, while the combination of Hough and spatial features gives a 69% F1-Score, and the curvature features result in a 70% F1-Score which is the best result among the different feature sets. Finally, combining all features achieves a 70% F1-Score. Thus, we can conclude that, using only the curvature features results in the same performance as using all features. We could also observe that almost every alternative group gives results in the [0.6 - 0.7] range. We cannot say that one feature group is extremely better than the others.

Table 3-14 Block based precision and recall values of various classifiers

	Features in Table 3-1 (Statistical)			Features in Table 3-10 Table 3-11 Table 3-12 (HOGs + Eigen- based + Second Order Derivatives)			Features in Table 3-2 Table 3-3 Table 3-4 (Hough +Spatial)			Features in Table 3-13 (Curvature)			All Features		
Classifier	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1	Pre	Rec.	F1	Pre.	Rec.	F1
Adaboost	0.57	0.53	0.55	0.63	0.62	0.62	0.62	0.58	0.60	0.64	0.61	0.62	0.66	0.66	0.66
NB	0.55	0.44	0.49	0.59	0.79	0.67	0.58	0.83	0.68	0.63	0.68	0.66	0.64	0.72	0.68
C4.5	0.56	0.59	0.57	0.67	0.65	0.66	0.63	0.76	0.69	0.64	0.75	0.70	0.64	0.63	0.64
RIPPER	0.59	0.60	0.59	0.63	0.72	0.67	0.61	0.74	0.67	0.63	0.75	0.69	0.63	0.79	0.7
K-NN	0.6	0.52	0.55	0.64	0.65	0.65	0.64	0.64	0.64	0.63	0.63	0.63	0.66	0.66	0.66

We evaluate the goodness of individual features by using their information gain measure in *Figure 3-37*. We can observe that eigen histograms, second order derivatives, and histogram of gradients have higher information gain values than statistical features and Hough-based features.

Figure 3-37 Information gain of the explored features in this section

3.3 Contour-based model on IDL-cleaned images

The results of the block-based approach on the IDL-cleaned images are certainly higher than the results of the block-based approach on raw images. However, it is still lower than our expectations. The main drawback of our previous study was that it was built on characteristics of regions. The non-loop regions containing other solar events (as shown in *Figure 3-38*) may have very similar characteristics to the loop regions, and hence they cause a decrease in the accuracy of the system and a high false alarm rate in the independent testing data.

Figure 3-38 Some false positives (non-loop blocks are classified as loop)

As a consequence, misclassifying those non-loop regions was inevitable. Therefore we modify our solution to the problem. Whereas in the block-based approach, we extracted features from the regions, in the new contour-based approach, we determine principal curves in each region and then calculate the geometric characteristics of the principal curves therein. Analyzing every single curve separately instead of a region as a whole gives more reliable classification results. We confirm the existence of a loop in a region based on the existence of a principal contour with loop characteristics.

In the new contour-based approach, instead of extracting blocks, we extract a strip around the solar disk, then binarize this strip and extract principal contours from it. Then we label the contours as either “*Loop*” or “*Non-Loop*” classes. We extract geometric features from the contours and then train the classifiers as usual. Compared to our previous system, this new method decreases the rate of misclassified regions and increases the efficiency of the loop detection system. In the current system, we achieve 85% precision and 83% recall on average in 10-fold cross-validation experiments. *Figure 1-13* illustrates the architecture of the contour-based approach. We describe the strip extraction, feature extraction, and classification results in the following sub-sections.

3.3.1 Strip Extraction

We prepare the image as described in Section 3.2.1. After cleaning the image, we extract a strip around the solar disk instead of dividing the image into blocks. Recall that one of the biggest time consuming parts was the block extraction in the previous approaches. Instead of rotating the image n times and cropping one block at a time, we decided to directly analyze the strip around the solar disk, thus significantly accelerating the overall solar loop mining procedure. Specifically, we eliminate the required time for block division, block labeling, and feature extraction from blocks. Moreover, we remove the possibility of loop blocks having partial loops due to block division.

We extract an image strip (see *Figure 3-39*) from outside the solar disk by using an angular transformation. Let R_0 be the radius of the Sun disk, x_c and y_c be the central coordinates of the solar disk, and H be the height of the strip. We create a strip of size $H \cdot (2\pi) \cdot R_0$ out of the original image. The algorithm for strip extraction is given in **Algorithm 3-2**.

Algorithm 3-2 Extracting a strip from outside the solar disk

```
Input: OriginalImage,  $H$ ,  $R_0$ ,  $x_c$ ,  $y_c$   
Output: Strip  
Set  $Circumference = 2\pi(R_0 + H)$   
for each  $i$  from 1 to  $Circumference$   
     $\theta = 2\pi i / Circumference$   
    for each  $j$  from 1 to  $H$   
         $x' = x_c + (R_0 + j) \cdot \cos(\theta)$   
         $y' = y_c + (R_0 + j) \cdot \sin(\theta)$   
        Strip( $i, j$ ) = OriginalImage( $x'$ ,  $y'$ )  
    end for  
end for
```

(a)

(b)

(c)

(d)

Figure 3-39 (a) Original Image with to be extracted strip around the solar disk, (b) Extracted strip of $H = 110$. (c) Strip after background extraction. (d) Strip after binarization

From the strip, we prefer keeping the central points of the forms instead of all gray values to reduce the system complexity and increase the loop detection speed. We obtain the central points of the fluxes by comparing the intensity value of a point to its four cross-pair neighbors. Here, we follow the same procedure described in Section 3.2.1. If the intensity level of a point is greater than or equal to that of at least its two different

cross-pairs, then we consider the point to be a central point, otherwise we eliminate this point. *Figure 3-40 (b)* illustrates the binary version of the image from the gray-level version in *Figure 3-40 (a)*.

Figure 3-40 (a) a magnified portion from the strip generated outside the solar disk, (b) the same strip after binarization

3.3.2 Principal Contour Extraction

Even though we obtain much cleaner images after the image preparation stages, we still need to extract salient contours separately and eliminate short independent segments in the strip. As mentioned before, loop segments can be fragmented due to several reasons. Human eyes can easily complete the gaps on the related loop segments whereas computers have a very hard time discerning salient contours and closing the gaps. In particular, if other forms intersect with the fragmented loop, then favoring the wrong line segment over the right one is highly possible. *Figure 3-41 (a)* shows a sample region obtained from the previous stages. *Figure 3-41 (b)* shows the desired loop contour to be extracted from the region, while *Figure 3-41 (c)* shows an undesired contour but

one that is likely to be extracted. The accuracy of coronal loop detection depends on extracting the salient contours accurately from the clutter.

Figure 3-41 (a) Input region, (b) desired salient contour (c) possible undesired contour

To overcome these problems, we propose a Principal Contour Extraction method that uses connected components as a hint for the existence of contours (Durak, et al., 2010). A connected component might consist of more than one contour and we wish to extract each individual contour separately. Therefore, we run our curve tracing method (**Algorithm 3-3**) which handles gaps and follows the correct path at the junctions. We start curve tracing from the top-left point of the longest component, and trace both sides of the starting point. We search a pie slice of radius R and with an area confined between $(Current_Angle - \alpha)$ and $(Current_Angle + \alpha)$. $Current_Angle$ is the orientation of the last traced K points, α is tolerance angle in the search space. Figure 3-42 shows the search space of the red point (which is $Current_Point$). For the immediate search space, we use a small R value such as 10, and we use $\pi/6$ for α .

Figure 3-42 The search space is inside the green triangle for the red point

For each candidate point in the search space, we calculate the angle change from the *Current_Angle* and the distance from the *Current_Point*. We select the closest point from the points with smaller angle change as the best continuation point. We delete the selected point from the original image. We continue tracing until there is no continuation point left in the search space. Then, we elongate and narrow down the search space to catch far away segments and escape from possible jumps to unexpected segments. If there is no suitable point neither in the immediate search space nor in the further search space, then we finalize the curve tracing for that contour. We continue extracting contours from the image, until no connected components longer than a certain length are left. **Algorithm 3-3** describes the principal contour extraction steps.

To test how well our contour extraction method catches the desired coronal loop contours in the cluttered regions (*Figure 3-41 (b)*), we tested our algorithm on 100 loop contours and 400 non-loop contours which are embedded in cluttered regions, and successfully extracted 88% loop contours and 90% non-loop contours as shown in **Table 3-15**.

Table 3-15 Accuracy of the Principal Contour Extraction from cluttered regions

	Desired	Undesired	Total
Loop Contours	88	12	100
Non-Loop Contours	362	38	400

Algorithm 3-3 Principal Contour Extraction

Input: Original Image, Minimum length (ML), Tolerance Angle (α)

Output: Curve list (P)

1. E : All white pixels
2. R : Radius of search space, α : search angle
3. K : Last K points of a traced curve
4. **While no edge pixels remain in E do**
 - 4.1 Find the connected components in E .
 - 4.2 If the longest component is shorter than ML , then break the loop.
 - 4.3 SP : Top-left point of the longest connected component in E
 - 4.4 *First Half*: The traced points traced from the left side of the SP ,
 - 4.4 *Second Half*: The traced points traced from the right side of the SP
 - 4.6 Trace First Half:
 $Current_Angle = \pi$, $Current_Point = SP$
 - 4.7 Calculate Initial Search Space from SP in direction of $Current_Angle$
 - 4.7. **While no points are left in the search space do**
 - 4.7.1 Find the candidate points in the search space
 - 4.7.2 For each candidate point, calculate angle change
 - 4.7.3 Pick the best candidate point by taking in consideration the Euclidian distance and angle change differences.
 - 4.7.4 **If** the difference between $Current_Angle$ and the angle change of the best candidate point is greater than $\pi/4$, **then** break from the loop, otherwise add this point to the Half; assign this point to $Current_Point$, and calculate $Current_Angle$ considering last K points; remove this point from E , calculate search space from $Current_Point$.
 - 4.7.5 If there is no point in the search space, compute search space using a $2*R$ radius but an $\alpha/4$ angle.
 - 4.8 Trace Second Half:
 $Current_Angle = 0$, $Current_Point = SP$
Repeat the same steps under 4.7
 - 4.8 Calculate smoothness of both halves; eliminate non-smooth half; if their junction is also smooth, then combine them.
 - 4.9 If the length of the final combination is greater than ML then add this combination into P , otherwise eliminate it.

The complexity of the algorithm is proportional to number of connected components in the image and the size of the search space. Let CC be the number of connected components, $CurveLen$ be the average contour lengths, and SS be the size of the search space. The average algorithm complexity is $O(CC.CurveLen.SS)$. The algorithm extracts each salient contour separately and then experts label them as “Loop” or “Non-Loop” for evaluation purposes. *Figure 3-43* shows the extracted contours from a region along with their labels.

Figure 3-43 Principal contours in (b, c, d) extracted from the region in (a), (b) Non-Loop Contour, (c, d) Loop Contours

We compare our principal contour extraction algorithm to Steger’s curve tracing algorithm (Steger, 1998) which is described in Section 2.2.1. When we apply their curve point classification method on *Figure 3-44* (a), we obtained the curve points in *Figure 3-44* (b). When we link the curve points in *Figure 3-44* (b) according to their algorithm, we obtained the final result shown in *Figure 3-44* (c). We also applied our curve point detection method on the same image and obtained the result shown in *Figure 3-44* (d). After applying our Principal Contour Extraction, we obtained the contours in *Figure 3-44* (e).

Figure 3-44 Comparison of our results to Steger's results (a) original image, (b) curve points according to Steger's algorithm, (c) extracted curves by Steger's algorithm, (d) edge points by our diagonal gradients based method, (e) extracted curves by our Principal Contour Extraction algorithm

To make the algorithm invariant to orientation, we can change the orientation θ in **Algorithm 3-3** with the orientation of the component. The orientation of the component is the angle between the x-axis and the major axis of the component and varies between -90° and 90° . Instead of selecting a fixed point for each component, selecting a different extreme point according to the component's orientation can increase the chance of obtaining the desired curves even if they are in different orientations. To make the selection process easier, we rely on only two rules. If θ is close to 90° or -90° , then the curve structure is more vertical. If θ is close to 0° , then the curve structure is more horizontal. Considering this fact, we decide to use the following rules to select the starting points:

- If θ is between 0° and 60° , then select the left-bottom point (Figure 3-45 (a)).
- If θ is between -60° and 0° , then we select the top-right point (Figure 3-45 (b)).
- If θ is greater than 60° , then select the top-left point (Figure 3-45 (c)).
- If θ is less than -60° , then select the bottom-right point (Figure 3-45 (d)).

Figure 3-45 Starting points for different oriented curve structures. (a) left-bottom point, (b) top-right point, (c) top-left point, (d) bottom-left point

After the orientation and starting point changes, we tested our algorithm on synthetic images with differently oriented curves which are intersecting each other as shown in *Figure 3-46*. The starting points are automatically assigned according to the orientation of the connected components. Our current algorithm is also able to separate intersecting curves from each other. At the junction points, the algorithm follows the correct path all the time. The orientation invariant method is especially useful for the coronal loops inside the solar disk.

Figure 3-46 On the left: Original image with differently oriented curves intersecting each other. On the right: Automatically extracted curves. Intersecting individual curves are shown with different colors

3.3.3 Geometric Feature Extraction

To decide whether the given contour is a loop or not, we extract geometric features from the labeled contours, then we learn a classifier model. To calculate the arch height, curvature and linearity of the contour, we use the point- to-chord distance plot.

Figure 3-47 The distance (h) between the curve point (P_i) and chord (L)

Let n be the number of points in the contour, P be the point set of the contour, and L be the chord which is a line connecting two end points, $P_l = (x_l, y_l)$ and $P_n = (x_n, y_n)$. For each contour point, $P_i = (x_i, y_i)$, the distance between the point and the chord L is calculated using Eq. (3-11). Figure 3-47 illustrates a point on a curve and its distance to the chord, L .

$$d(P_i, L) = \frac{(y_1 - y_n)x_i + (x_1 - x_n)y_i + (x_1 y_n - x_n y_1)}{\sqrt{(x_n - x_1)^2 + (y_n - y_1)^2}} \quad 3-11$$

After calculating the distance for each point in P , we obtain a point-to-chord distance vector $\vec{D}$ of the contour, $\vec{D} = [d(P_l, L) \dots d(P_n, L)]$. We do not take the absolute value of the numerator in Eq. (3-11) and obtain a signed distance vector $\vec{D}$ with positive components d^+ and negative components d^- . Positive distance components are on one side of the chord while negative distance components are on the other side. The geometric features extracted from the contours are described in the following subsections.

3.3.3.1 Linearity of the contour

We run our principal contour extraction algorithm on every single block without any knowledge about the existence of loops in the blocks. Hence the contour extraction algorithm may end up tracing different shapes instead of only perfect loop shapes. The most common undesired traces tend to be linearly shaped or consisting of some close points forming small clusters. Based on this observation, we apply a line fitting algorithm which basically performs the first order polynomial fit on the traced curve. After that, we count how many points of the traced curve are located close to the fitted line. In Eq. (3-12), N is the number of points in the traced curve, P is the traced curve, and F is the fitted line.

$$Linearity = \frac{1}{N} \sum_{i=1}^N l_i \quad 3-12$$

$$\text{where, } l_i = \begin{cases} 1, & \text{if } d(P_i, F_i) < \tau \\ 0, & \text{otherwise} \end{cases},$$

where, $d(P_i, F_i)$ is the perpendicular distance from point P_i to the fitted line F_i

Figure 3-48 (b) shows the contour extraction results from a non-loop region in Figure 3-48 (a). In that region, there are two different tracing results which both have linear shape. With the *Linearity* feature, we can eliminate highly linear curves or the curves consisting of a small cluster and lacking an arch shape. In those cases, the extracted contours cannot be part of a loop structure.

Figure 3-48 The contour extraction algorithm is applied on a non-loop region. The linearity of the extracted contours is high. $Linearity = 0.95$ for the left contour in (b) and it is equal to 0.92 for the right contour

3.3.3.2 Pseudo-curvature

To calculate the pseudo-curvature of the contour, we employ the point-to-chord distance vector, $\vec{D}$. We calculate the curvature value as given in Eq. (3-13) by dividing the arch height of the contour (h) over the chord length ($\|L\|$) which is the Euclidean distance between the endpoints of the contour. Pseudo-curvature is close to 0 for straight-line, and the higher it gets, the more the contour deviates from a straight line.

$$Pseudo - curvature = \frac{h}{\|L\|} \quad 3-13$$

3.3.3.3 Smoothness

The automatically extracted contours may contain some jaggedness which is some rapid orientation changes along the curve. If the orientation change is severe at a point, then this might indicate the presence of a corner at that point. If the orientation change is not that severe, then that point deviates from the straight line a little bit, but is not a corner.

Loop contours are generally very smooth and do not contain many severe changes along the curve structure. Also, there may be several smooth junctions along the loop contour. Along non-loop contours however, the jaggedness ratio is higher and there are

more corner points. Considering these facts, we count the number of corner points along the curve. To detect corner points, we divide the curve into small windows and then determine representative pixels (red points in *Figure 3-49*) within each window as shown in *Figure 3-49*.

Figure 3-49 Dividing the curve into windows

We then calculate the angle change between two neighbor windows. Suppose that for the second window, the angle change of the window is calculated by subtracting the angle α_2 between the 2nd and 3rd windows from the angle α_1 between the 1st and 2nd windows. If the absolute value of the angle change $|\alpha_i - \alpha_{i-1}|$ is greater than τ , then that change is an indication of a corner point, as given in *Eq. (3-14)*. α_i is the tangent of the angle of the line connecting the two representative points of two consecutive windows.

$$\text{Corner Points} = \text{The number of point where } |\alpha_i - \alpha_{i-1}| > \tau \quad 3-14$$

In addition to the number of corner points, we calculate the smoothness of the curve, which might be called real curvature. If there are n windows along the curve, the smoothness is the average root square of angle changes among these windows, as given by *Eq. (3-15)*.

$$\text{Smoothness} = \frac{1}{n-1} \sqrt{(\sum_{i=2}^n (\alpha_i - \alpha_{i-1}))^2} \quad 3-15$$

3.3.3.4 Elliptic Features

Since coronal loops tend to be similar to a half-ellipse, we have also attempted to apply Hough Transform based ellipse detection methods which are described in Section 2.5.2 (Duda, et al., 1972; McLaughlin, 1998; Tsuji, et al., 1978). We implemented the random ellipse detection methodology (McLaughlin, 1998) to determine the parameters of the ellipses. However, since our loops are not perfect ellipses, (they are rather asymmetric or half ellipses as shown in *Figure 3-50 (b)*), the random point selection led to the incorrect center points. In particular, for near-positive *Non-Loop* contours such as the one in *Figure 3-50 (a)*, detecting center points or computing axis lengths is very challenging in Random Hough Transform based methods. In addition, loops come in different sizes and adjusting the size of the ellipse detector is another big problem for Hough based ellipse detection methods.

Figure 3-50 (a) Non-loop contour, (b) loop contour which is asymmetric and half complete

For all these reasons, instead of Hough based methods, we resorted to conic section fitting as described in Fitzgibbon (Fitzgibbon, et al., 1999) and apply direct least square fitting on the extracted contour to obtain the parameters of the conic section equation given in *Eq. (2-24)*. With the help of the computed parameters, we compute the major axis lengths, where the major axis is K and minor axis is L . We then calculate the

error of fit using the algebraic distance between the original values at the contour and the estimated conic section model. Finally, we define the ellipse related features listed in **Table 3-16**.

Table 3-16 Elliptical features from the contours

Feature Name	Description
Eccentricity	$\epsilon = \sqrt{K^2 - L^2}/K$
Minimum of Axis	The minor axis length (L)
Ratio of Axis	L/K
Function Shape	$B^2 - 4AC$
EOF-Ratio	$\frac{\text{\# of points having small error}}{\text{number of total points}}$

3.3.3.5 Point-to-Chord distance features

To distinguish loops from non-loop contours which have small linearity values but high curvature values, we check for the existence of a bell shape in the contour. To determine the bell existence, we plot the distance between each point on the contour C and its projection on the chord L versus the x-position on the chord (x is the distance between a projected contour point and the projection of the first contour point along the chord) as shown in *Figure 3-51* (c).

Figure 3-51 Computing the bell shape existence of a loop region: (a) traced curve, (b) chord line and traced contour, (c) plot of the distance values between the traced contour and the chord, indicating a bell shape.

To determine the presence of a bell shape, we first find the peak of the distance plot. Peak point is the maximum distance from the chord and peak location is the index of the peak point in distance vector. In an arc shape, we expect the peak point in the middle. In a bell shape, the distance values on both sides of the peak should decrease. Thus, we count the number of decreasing points on both sides of the peak point, and then compute the proportion of decreasing points on each side over the number of points in the corresponding side (these ratios are called *LeftRatio* and *RightRatio*). We also measure the skewness of the plot which takes values depending on the location of the peak-point. Finally, we take the minimum of *LeftRatio* and *RightRatio* and multiply it with the skewness to compute the *bell-Existence* feature in Eq. (3-16).

$$LeftRatio = \frac{\text{The number of decreasing points to left of peak}}{\text{Total points to left of peak}}$$

$$RightRatio = \frac{\text{The number of decreasing points to right of peak}}{\text{Total points to right of peak}}$$

$$skewness = \left| 0.5 - \frac{\min(\text{peakLocation}, \text{total points in the curve} - \text{peakLocation})}{\text{total points in the curve}} \right|$$

$$Bell-Existence = skewness + LeftRatio + RightRatio$$

3-16

Bell-Existence is a feature that complements the linearity and curvature features. It is particularly helpful in eliminating those non-loop curves that are non-linear as in illustrated in *Figure 3-52* (a) and (c).

Figure 3-52 (a) and (c) Non-Loop contours, (b) and (d) their distance plots respectively, showing the lack of the existence of a bell shape.

Table 3-17 Point-to-Chord distance features

Feature Name	Description
Bell-Existence	Measures how bell shaped a given contour is
ArchHeight	The maximum distance from the contour to the chord

3.3.3.6 Proximity

Since we allow gaps in the curve tracing phase, there might be some gaps between contour points. If the points are close to each other, then that contour is more

promising than a contour with several gaps. On the other hand, there is tendency for more gaps between the non-loop contour points. Therefore, we calculate the Euclidean distance between the consecutive points and take their average (as given in Eq. (3-17)) to obtain the proximity value feature.

$$proximity = \frac{1}{n-1} \sum_{i=2}^n EucDistance(P_i, P_{i-1}) \quad 3-17$$

3.3.4 Classification and Experimental Results

To form our training data set, we extracted principal contours from the image strips 600 images, then expert label them as “Loop” and “Non-Loop”. We gathered 150 “Loop” contours and 250 “Non-Loop” contours. After extracting all the features described in the previous sub-section, we trained the following classifiers: Adaboost based on C4.5, RIPPER, C4.5, Naïve Bayes and K-NN. **Table 3-18** shows the precision, recall, and F1-score obtained from the classifiers in 10-fold cross-validation experiments. Adaboost based C4.5 achieved 85% Precision and 83% Recall, an accuracy level is significantly higher than the accuracy level of the block-based approach on raw images (63% precision and 74% recall) or the block-based approach on the IDL solar software cleaned images (63% precision and 79% recall). Adaboost classifier reaches the best precision-recall pairs very quickly in ROC curve (*Figure 3-53*).

Table 3-18 Classifier Results of Contour based approach

Classifier	Pre.	Rec.	F1
Adaboost based C4.5	0.851	0.829	0.84
NB	0.726	0.803	0.763

C4.5	0.808	0.829	0.818
RIPPER	0.823	0.855	0.839
K-NN	0.791	0.697	0.741

Figure 3-53 ROC curve for the Adaboost

3.3.5 Experimental Results on Outside Blocks

To further evaluate our automated loop detection technique and show real examples of how it works in real life, we tested it on some challenging regions from a variety of EIT images that were not included in the training data. From each region, we extracted all contours that are longer than 15 pixels in length, and fed them as input to the Adaboost model. **Table 3-19** shows some challenging regions in column 1, as well as the extracted contours from these regions and their predicted labels in column 2 and 3. Our experimental results on these and other examples confirm that the extracted features are successful to reach correct decisions. The second and third columns in Table 3-19 show how our contour extraction algorithm generates correct contours from cluttered regions as well.

Table 3-19 Sample regions along with the extracted contours and their predicted contour label.

Binary region	Individual contours extracted from each region with the predicted label below each traced contour
<i>("Loop")</i>	<i>("Non-Loop")</i>
<i>("Loop")</i>	<i>("Non-Loop")</i>
<i>("Loop")</i>

(<i>“Non-Loop”</i>)	(<i>“Non-Loop”</i>)
	(<i>“Non-Loop”</i>)
(<i>“Non-Loop”</i>)	(<i>“Non-Loop”</i>)
	(<i>“Non-Loop”</i>)	(<i>“Non-Loop”</i>)

3.3.6 Testing Inside Disk Blocks

With the older block-based approach, we were not able to classify the regions inside the solar disk correctly. We want to extend the usage of the model on inside disk blocks. To detect the inside loops, we perform the same preprocessing techniques proposed for the outside loop detection phase, then divide the solar disk into fixed sized

blocks, and extract every contour that is longer than a certain length from each block. Then we use each contour's features as input to the Adaboost model to decide whether the extracted contour is a loop or not. If the decision is "*Loop*", then the block is labeled as "*Loop*", otherwise it is labeled as "*Non-Loop*." Table 3-20 lists the binary blocks along with the extracted contours and the label of the block. In this case, the orientation of the contours could point to different directions. Therefore, we use the orientation of the component as a hint to select the starting point in **Algorithm 3-3**.

Table 3-20 Inside blocks and their decisions

Binary Regions	Extracted Contours	Label of the Block
" <i>Loop</i> "
" <i>Loop</i> "
" <i>Non-Loop</i> "

3.3.7 Image Retrieval Tool

The steps that we follow in the contour-based image retrieval tool are different from the block-based image retrieval tool. *Figure 3-54* illustrates the architecture of the testing phase of the contour-based approach. Note that for best results, the input images should be cleaned using the IDL solar software (ssw) as explained in Section 3.2.1. After cleaning the data, we extract strips, then contours, then features and predict the label of the contours using the Adaboost classifier model. If any of the contours is labeled as "Loop", then that image is added to the list of images containing loops.

Figure 3-54 Testing architecture for contour-based approach

To summarize, for each image, we perform the following procedure:

- 1) Apply the preprocessing steps, which are despeckling, smoothing, and background subtraction.
- 2) Extract a strip from outside the solar disk and binarize the strip.
- 3) Extract principal contours from the strip.
- 4) Extract geometric features from the extracted contours.
- 5) If any contour is classified as “*Loop*”, then add this image into the image list with loops.
- 6) Map the location of the detected loops on the images by reversing the angular transformation used to extract to extract the strip.

For the image strip in *Figure 3-39*, we extracted the principal contours shown in *Figure 3-55 (a)*. The contours predicted as “*Loop*” are shown in *Figure 3-55 (b)*, and the mapped loop contour regions on the image are shown in *Figure 3-55 (c)*. These results show that our contour-based approach can automatically spot the exact location of the detected loop.

(a)

(b)

(c)

Figure 3-55 (a) Strip with extracted principal contours, (b) Contours classified as “Loop”, (c) Mapping the Loop contours to the original image

(a)

(b)

Figure 3-56 (a) Image Retrieval Tool that uses the contour-based model, (b) Automatically detected loops both inside and outside the solar disk

With a modest desktop computer (2GHz processor, 3 GB RAM), the completion of the above steps takes from 8 to 12 seconds per image. Compared to the block based

method, this is a big step forward in the performance of the system. For image retrieval, we use the same user interface that we have designed for image retrieval using the block-based method. The users upload a set of IDL ssw cleaned images and the system processes the images as described above and returns the images with loops in a separate list. *Figure 3-56 (a)* illustrates a result from the image retrieval tool. We also tested our method on loops located inside the solar disk. *Figure 3-56 (b)* shows the automatically detected regions both inside and outside the solar disk.

We tested our model on an unseen test image set that consists of 50 images with coronal loops and 50 images without loops. After performing the proposed image processing and contour extraction methods, then extracting geometric features of the detected contours, and we fed the features to an Adaboost trained model. If any contour in an image is classified as “*Loop*”, then we assume that the image contains a coronal loop. Out of 50 loop images, 45 images were classified as containing a loop correctly. Out of 50 non-loop images, 44 were classified correctly. Hence, we achieved 90% precision from the contour-based approach, as detailed in **Table 3-21**.

Table 3-21 Confusion matrix for image based testing results

(precision = 90%, recall = 88%)

	Actual	Loop Image	Non-Loop Image	Total
Predicted				
Loop Image		45	6	51
Non-Loop Image		5	44	49
Total		50	50	100

4 EXTRACTION of SALIENT CONTOUR GROUPS from CLUTTER

*“A bit beyond perception's reach
I sometimes believe I see
that Life is two locked boxes, each
containing the other's key.”*

~Piet Hein

Before obtaining salient contour groups, we need to obtain discrete contours from cluttered images by applying our curve tracing algorithm. To handle subtle corner points or transition points along the traced curves, we detect critical points and segment the curves at those points. Then, we associate each contour with its neighboring contours and compute the saliency measure of each contour. At the end of these steps, we obtain a set of smooth contours to be used in contour grouping. The details of the discrete contour extraction procedure are described in Section 4.1.

After obtaining individual contours, we group those that are related to form salient *contour-groups*. Contours should hold several criteria, (e.g., ellipticity, concavity, linearity, proximity, etc.) to be in the same group, that we combine in a cost optimization approach. We present the pairing measures along with the weight estimation of the measures in the cost function in Section 4.2.

In Section 4.3 we present the proposed algorithm to group related salient contours and to separate salient contour groups from the background clutter. In Section 4.4, we show the results of our approach on synthetic images, coronal loops on solar images, and road detection from aerial images.

4.1 Discrete Contour Extraction

Discrete contours should be smooth and the points of the contours should be adjacent to each other. We extract discrete contours with the help of the curve tracing algorithm that was presented in **Algorithm 3-3** (Durak, et al., 2010).

Before running the curve tracing algorithm, we preprocess the images according to the requirements of the application. If there are gaps among pixels, we first close the gaps among pixels and thin the image using morphological operators.

In the curve tracing algorithm, first connected components are detected, and then both sides of the top-left point of the longest component are traced. To add a new point to a traced curve, a pie slice with radius R and angle α is searched. New points are added to the traced curve until there is no point found in its immediate search space of the latest added point. The same procedure is repeated until no connected components are left in the image. Since severe angle changes are not allowed in this curve tracing algorithm, the final contours have less squiggles or jaggedness. As a result of each curve tracing, we obtain a discrete contour which consists of a set of points:

$$\mathcal{P} = \{ (x_1, y_1), \dots, (x_n, y_n) \} \quad 4-1$$

4.1.1 Curve Segmentation

Even though traced curves are free of severe angle changes at the pixel-level, there still could be corner points or concavity changes in the entire curve. The success of contour grouping relies on the smoothness of discrete contours. Therefore, we perform curve segmentation on the curves that we acquired from the curve tracing stage.

Curve segmentation can suffer from over-segmentation and under-segmentation issues. In the over-segmentation case, the algorithm divides the curves more than necessary, whereas in under-segmentation, the algorithm may miss subtle transitions and keeps some squiggles. In contour grouping, under-segmentation is more hurtful than over-segmentation. Even though, over-segmentation might increase the complexity of the contour grouping algorithm, we favor it over under-segmentation because in the over-segmenting cases, the contour grouping connects the segments if they are part of the same group. However, if there is a transition in a curve and we do not divide the curve at that point, then contour grouping will give a high cost for the possible grouping and will not combine the squiggly contour to other contours.

In order to detect critical points, we divide the contour into fixed size windows and compute the angle dissimilarity of the vectors from one window to the consecutive window as shown in *Figure 4-1*.

Figure 4-1 Vectors between consecutive windows along the contour

We compute the angular dissimilarity between consecutive vectors, v_1 and v_2 , using Eq. (4-2). This measure generates values between 0 and 1, in such a way that vectors in the similar directions have low dissimilarity value while vectors pointing in different directions will take higher values.

$$\delta = 0.5 - \cos(\alpha_i) / 2 \quad 4-2$$

$$\text{where, } \cos(\alpha_i) = \frac{v_i \cdot v_{i+1}}{\|v_i\| \|v_{i+1}\|}$$

After obtaining the angle dissimilarity plot, we detect the critical points using thresholding. If there is any change greater than a given threshold τ_l in the angle dissimilarity plot, than we take that point as a critical point. We should select the threshold value τ_l value carefully, because small thresholds cause over-segmentation while big thresholds miss subtle transition points.

Figure 4-2 Segmenting a curve at a corner point with two different window sizes. Critical points are shown on the image with a red dot.

Selecting the window size is also critical in curve segmentation. Small windows might generate several spikes and cause over-segmentation, while big windows might not locate the exact location of the corner points. Another problem is that we need to change the threshold value to locate the critical points for different window size. Figure 4-2 shows that the angle dissimilarity plots for two different window sizes are very different. Figure 4-2 (a) shows a bigger window size which misses the exact location of the corner point while Figure 4-2 (b) shows that the angle dissimilarity values for the smaller window size are close to each other in this case and thus selecting a wrong threshold easily cause over-segmentation.

Since we want to locate the exact locations of critical points on the curve, we favor small windows over bigger windows. However, we present two heuristics to alleviate the over-segmentation problem in small windows:

1) *Selecting the peak spike in a neighborhood:* When there is a spike in the plot, there might be other high values in its neighborhood. For example, in the case of a severe corner point along the curve, there could be several high dissimilarity values around the corner point. We thus divide the curve at the highest dissimilarity value and eliminate the rest of the candidate points to overcome over-segmentation.

2) *Having two thresholds:* We have two different thresholds, τ_1 and τ_2 , such that $\tau_1 \gg \tau_2$. We pick a high value for τ_1 to guarantee catching severe transitions, while we pick a smaller value for τ_2 to catch subtle transitions. To avoid an incorrect segmentation, we impose another condition for the dissimilarity values between τ_1 and τ_2 . By observing the characteristics of the angle dissimilarity plot, we were able to observe the following working condition. If there is a dissimilarity value (δ_1) between τ_1 and τ_2 , then there must be another point with similar dissimilarity value (δ_2) in its neighborhood. If there is only one isolated spike, it is probably noise, and we therefore ignore it. Mathematically speaking, these conditions can be formulated as: $\delta_1 > \delta_2$ and $\delta_2 > \tau_2$ and $|\delta_1 - \delta_2| < \epsilon$. We can observe this behavior in *Figure 4-2 (b)* and *Figure 4-3 (b)* in which the peak point is followed by another high dissimilarity value.

Figure 4-3 Sample results of curve segmentation. Red dots show the location of the critical points that are detected.

Figure 4-3 illustrates some outputs of our curve segmentation approach. In Figure 4-3 (a), a severe corner point is detected without a problem. In Figure 4-3 (b), even though the transition is very subtle, we catch the critical point. In Figure 4-3 (c), we catch the major transition with the τ_1 . We also detect another critical point due to a low τ_2 . We let contour grouping deal with these kinds of segmentations.

4.1.2 Saliency Computation

After curve segmentation, we obtain a set of discrete smooth contours, which are shaped like either arcs or lines. Since we are interested in *salient* contour groups, we assign saliency measures to each contour. The *saliency measure* ξ of a contour represents a measure of how much a contour pops-out from the background and captures attention in the scene. Different applications may need different definitions for saliency measures. For instance, object boundary detection studies favor closure and smoothness in saliency computation (Ullman, et al., 1988; Wang, 2007). Since coronal loops are semi-elliptical open curves, we define the saliency measure of a contour in Eq. (4-3) using the *contour length* which is the cardinality of $\mathcal{P}$ (given in Eq. (4-1)) and *linearity*, $\mathcal{L}$, which is given by Eq. (3-12). While short and linear contours are the least salient, long and circular arcs are the most salient in our system.

$$\xi = |\mathcal{P}| + |\mathcal{P}| * (1 - \mathcal{L}) \quad 4-3$$

We have also experimented with adding curvature consistency as another factor in the saliency measure. However, the curve segmentation component already returns smooth contours, making the curvature consistency pretty much the same for all contours. For this reason, curvature consistency was not helpful.

After calculating the saliency measures of all contours using Eq. (4-3), we sort the contours according to their saliency measure. *Figure 4-4* shows some contours ordered by their saliency measure.

Figure 4-4 Sample contours ordered by their saliency measures. Saliency decreases from left to right in each column (a) contours with an arc shape, (b) linear contours

4.1.3 Neighbor Association

For grouping purposes, we associate each contour to its neighboring contours. In our problem, each contour can be grouped with at most one other contour from each end. This constraint is set to prevent obtaining wishbone structures during grouping. In each end of contour point set $\mathcal{P}$, we search a region confined within an isosceles triangular region (as in shown in Figure 4-5) whose peak point is at the first quartile point $\mathcal{P}_{(n/4)}$ or last quartile point $\mathcal{P}_{(3n/4)}$ of the contour; side lengths are the half length of the contour ($R = |\mathcal{P}|/2$), and tolerance angle, α . We narrow the searchable regions in order to reduce the time complexity of the contour grouping. These parameters could be adjusted depending on the application. In order to give fewer grouping options to the short

contours while giving more options to the longer contours, the search space is adjusted proportional to the contour's length.

Figure 4-5 The search space for each end is shown as a gray region

For each contour, we search for the neighboring contours in each end and associate them to the contour. Let C_1 be the current contour and C_2 be a neighboring contour. In order to add C_2 into C_1 's neighbor list, one of C_2 's endpoints should reside in C_1 's search space. We keep the neighbors in each search space in different neighbor lists, called N_1 and N_2 .

At the end of neighbor association, contours are ready for the contour grouping stage. Each contour has a group label GL which represents the group to which the contour is associated. We assume that all contours belong to the background in the beginning. Therefore, they have an initial group label equal to zero. We give the definition of the contour structure in **Definition 4-1**.

Definition 4-1: *Contour* = $(\xi, \mathcal{L}, \mathcal{P}, N_1, N_2, GL)$ where ξ is the saliency measure, $\mathcal{L}$ is the linearity, $\mathcal{P}$ is the contour point list, N_1 is the neighbor list on one end, N_2 is the neighbor list on the other end, and GL is the group label of the contour.

4.2 Pairing Contours

When we pair two contours C_1 and C_2 , we combine their point lists cautiously to calculate pairing measures correctly. Combining points in a wrong order might cause wrong pairing measures and a wrong estimation of the cost value.

Before combining the points of two contours, we calculate four distances (d_1, d_2, d_3, d_4) between the end points of the two contours. In *Figure 4-6*, dashed lines represent the distances between the end points. We combine the contours from the end points having the minimum distance.

Figure 4-6 We combine the contours from their closest endpoints. Dashed lines represent the four distances (d_1, d_2, d_3, d_4) among the end points of the two contours C_1 and C_2 .

If the shortest distance between two contours is between one's end point ($\mathcal{P}_n$) and the other's starting point ($\mathcal{P}_1$), we can append the point list of the contour on the right side to the end of the point list of the contour on the left side. Notice that the curve tracing algorithm returns contours whose point list $\mathcal{P}$ is ordered from the left end point $\mathcal{P}_1$ to the right end point $\mathcal{P}_n$ as shown in *Figure 4-6* (b). We desire to keep the same order in the grouped contours.

If the shortest distance between two contours is either between one's end point ($\mathcal{P}_n$) and the other's end point ($\mathcal{P}_n$) or between one's starting point ($\mathcal{P}_1$) and the other's starting point ($\mathcal{P}_1$) (such as in *Figure 4-6* (a)), we have to flip one of the point lists to

obtain the correct combination. We follow the same rules when we add a contour into an existing group. In that case, the group acts like a contour and we only add contours from the end points of the group.

For each contour in the image, we pair the contour with each one of its neighboring contours in N_1 and N_2 separately and then calculate the pair-wise cost for each pair. In previous studies (Wang, et al., 2005; Felzenszwalb, et al., 2006), the cost function was defined using only the smoothness measure. However, in cluttered images, smoothness alone is not sufficient to extract semi-elliptical open curves. Thus, in order to discern coronal loops or other open curves, we define our cost function in terms of the following criteria: *angular dissimilarity, ellipticity, concavity, arch shape, eccentricity, proximity, and length*.

We pair the contours and compute the measures on the combined data points. We compute the pair-wise cost between neighboring contours and keep the pair-wise costs in a cost matrix, thus speeding up the optimization process.

4.2.1 Angular dissimilarity

If the vectors of the contours at the connection part follow the same direction, then it is possible that these two contours belong to the same contour group. To measure the similarity of vector directions, we calculate the angular dissimilarity, δ , between two vectors (v_1 and v_2) located at the connection part of two contours. *Figure 4-7* shows these vectors with red arrows at the connection part of the contours C_1 and C_2 .

Figure 4-7 Arrows indicate the vector directions at the connection part of two contours

A vector could be oriented either from $\mathcal{P}_l$ to $\mathcal{P}_{(n/4)}$ or from $\mathcal{P}_{(3n/4)}$ to $\mathcal{P}_{(n/4)}$ of a contour depending on which sides the contour will connect to the other contour. We calculate the angular dissimilarity, δ as given in Eq. (4-2), using the cosine of the angle α between the two vectors v_1 and v_2 . Figure 4-8 shows two different vector combinations and angle α between them. The δ measure takes values in the range $[0, 1]$ where low values are for similar angles and high values are for dissimilar angles.

Figure 4-8 Two different contour combinations and angle α between two vectors v_1 and v_2

4.2.2 Ellipticity

Since we are seeking semi-elliptical curves, we check whether the combined contours lie on the same ellipse. For obtaining their elliptical goodness, we fit an ellipse to the combined points of the contours using a direct least square ellipse fitting method (Fitzgibbon, et al., 1999) that calculates the optimal ellipse parameters. To calculate the error of fit for a point, we have used the gradient weighted algebraic distance given by Eq. (4-4).

$$\varepsilon_i = \frac{Q(x_i, y_i)}{|\nabla Q(x_i, y_i)|} \quad 4-4$$

$$\text{where, } Q(x_i, y_i) = Ax_i^2 + Bx_iy_i + Cy_i^2 + Dx_i + Ey_i + F$$

For a reasonably good fit, the mean of the residual errors ε should be close to zero and their variance should be low (Ji, et al., 1999). To normalize the residual errors, first we subtract the mean of the residual space (RS) from the entire residual points. Let $\bar{\varepsilon}$ be the sample mean and σ^2 be the sample variance of the residual space. We test whether the residual space has a normal distribution using Welch's T statistic, given in Eq. (4-5) which is expected to be low for contours lying on the same ellipse and high otherwise.

$$T_{overall} = \left| \sqrt{N} \frac{\bar{\varepsilon}}{\sigma} \right| \quad 4-5$$

We observe that at the joining part of two contours C_1 and C_2 , there is a spike in the residual space even when C_1 and C_2 lie on the same ellipse. Figure 4-9 shows the residual space for two contours on the same ellipse. These spikes at the connection points increase the $T_{overall}$ value and hurt the reliability of the test statistic. Hence, we first remove the spikes and then calculate the test statistic using Eq. (4-5).

Figure 4-9 (a) Two contours, (b) Fitted ellipse on the two contours, (c) Residual space of the error of fit. Notice that there is a spike at the connection point shown in a red circle.

In some contour combinations (as in *Figure 4-10*), the combined test statistic, $T_{overall}$, will have low value even the two contours are not on the same ellipse. Therefore, we compare the test statistics of the two parts separately to check if they are similar. Let n_1 and n_2 are the numbers of points in C_1 and C_2 . Let μ_1 and μ_2 be the mean values of the residual points of each part, and let σ_1^2 and σ_2^2 be the variance values of each part respectively. We compare these two test statistics using *Eq. (4-6)*.

$$T_{comp}(C_1, C_2) = \frac{|\mu_1 - \mu_2|}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \quad 4-6$$

Figure 4-10 (a) Two contours, (b) Fitted ellipse on the contours, (c) Residual space of the error of fit. Notice that the $T_{overall}$ statistics is not reflecting the difference on both sides of the connection point, which is shown in a red circle.

If there are more than two contours in the contour group, we take the average of the test comparisons between all consecutive contour pairs. Let N be the total number of contours in the group, we take the average test statistic using *Eq. (4-7)*.

$$T_{avg-comp} = \frac{1}{N-1} \sum_{i=1}^{N-1} T_{comp}(C_i, C_{i+1}) \quad 4-7$$

Another observation about the residual space is that even if two contours do not lie on the same ellipse, the comparison might still not reflect this fact. For example, in

Figure 4-11, two contours do not lie on the same ellipse. However, both sides of the connection point have almost the same error of fit mean and variance. Hence, both $T_{avg-comp}$ and $T_{avg-overall}$ are not solely enough to represent the truth. We observe that there is a jump between the left and right sides of the connection point and if we use all the points on each side, we apparently cannot capture this jump. Hence, instead of using all the points on each side, we propose to use only K points from the left side and right side of the connection point.

Figure 4-11 (a) Two contours, (b) Fitted ellipse for both contours, (c) Residual space of the error of fit. Notice that the $T_{avg-comp}$ and $T_{avg-overall}$ statistics are very low, since we compare only K values on the left side and the right side.

Let μ_3 and μ_4 be the mean values of K points of the left side and right side of the connection point respectively, and let σ_3^2 and σ_4^2 be the variance of K points from the left side and right side of the connection point, respectively. We compare the test statistics of these two fixed-size intervals using Eq. (4-8). For K , we experimented with different values and decided to use $K = 30$. Bigger windows might not catch a jump.

$$T_{jump}(C_1, C_2) = \frac{|\mu_3 - \mu_4|}{\sqrt{\frac{\sigma_3^2}{K} + \frac{\sigma_4^2}{K}}} \quad 4-8$$

If there are N contours in the contour group, we derive the T_{jump} statistics between consecutive contours and take the average of all of them as given in Eq. 4-9.

$$T_{avg-jump} = \frac{1}{N-1} \sum_{i=1}^{N-1} T_{jump}(C_i, C_{i+1}) \quad 4-9$$

Since, the three statistics $T_{overall}$, $T_{avg-comp}$, and $T_{avg-jump}$ reflect different behaviors of the residual space, instead of using only one test statistics, we average them to obtain a strong ellipticity value as given in Eq. (4-10). Different weights could be assigned to different statistics depending on the requirements of the application.

$$T_{final} = \frac{1}{3}T_{overall} + \frac{1}{3}T_{avg-comp} + \frac{1}{3}T_{avg-jump} \quad 4-10$$

The test statistics computed for different contour groups are illustrated in *Figure 4-12*. *Figure 4-12* (a, b) show examples where two contours are lying on the same ellipse. Note that T_{final} is low for these cases. *Figure 4-12* (c, d) show examples where two contours do not lie on the same ellipse. T_{final} are higher than one for both cases. We can also see how different statistics reach high values for negative cases. We also show one example in *Figure 4-12* (e) where the contours do not lie on the same ellipse but T_{final} is low. Notice that this group is very elongated (i.e., low eccentricity.) These cases show that we cannot rely on ellipticity alone for contour grouping. Different measures should cooperate to reach an optimal solution. In *Figure 4-12* (f, g), there are more than two contours. While *Figure 4-12* (f) is a positive case, *Figure 4-12* (g) is a negative case for our contour grouping. Even though T_{final} is higher in *Figure 4-12* (g) than in *Figure 4-12* (f), the difference between them is not significant and not very helpful in decision

making. One reason for the low difference is that three contours out of four lie on the same ellipse. Hence the effect of the fourth contour is not playing a big role. This supports the fact that we need more measures to group the contours correctly. For example, in the last case, the angle dissimilarity measure will be helpful.

(c)

$$T_{overall}=0.39 \quad T_{avg-comp} = 2.89 \quad T_{avg-jump} = 0.3837 \quad T_{final}=1.22, \text{ Eccentricity} = 0.3885$$

(d)

$$T_{overall}=0.2084 \quad T_{avg-comp} = 0.3262 \quad T_{avg-jump} = 0.5742 \quad T_{final}=0.3696, \text{ Eccentricity} = 0.06$$

(e)

$$T_{overall}=0.211 \quad T_{avg-comp} = 1.37 \quad T_{avg-jump} = 1.24 \quad T_{final}=1.0039, \text{ Eccentricity} = 0.75$$

(f)

Figure 4-12 Test statistics and eccentricity value are given along with the contours, fitted ellipses, and the residual space of the fits

4.2.3 Eccentricity

During ellipse detection, we compute the ellipticity of the entire shape using the minor axis length (L) and major axis length (K) using the eccentricity formula ($\epsilon = \sqrt{K^2 - L^2}/K$). The eccentricity values of the grouped contours were shown in Figure 4-12. We can see that elongated groups (as in Figure 4-12 (e)) have small eccentricity values, while circular groups (as in Figure 4-12 (c, f)) have higher eccentricity values.

4.2.4 Measures based on Point-to-Chord distance

We first combine the point lists of two contours and then compute different measures based on the point-to-chord distance for each point. Let n be the number of points on the contour and let (x_l, y_l) and (x_n, y_n) be the end points of the combined

contours $\mathcal{P}$. Chord L is a line that joins the two end points of the contour. For each point $\mathcal{P}_i = (x_i, y_i)$ on the combined contours, the distance between the point and the chord is calculated using Eq. (4-11).

$$d(\mathcal{P}_i, L) = \frac{(y_1 - y_n)x_i + (x_1 - x_n)y_i + (x_1y_n - x_ny_1)}{\sqrt{(x_n - x_1)^2 + (y_n - y_1)^2}} \quad 4-11$$

After calculating the distance for each point in $\mathcal{P}$, we obtain a point-to-chord distance vector $\vec{D}$ of the contour. We do not take the absolute value of the numerator in Eq. (4-11). Hence, we obtain a signed distance vector $\vec{D}$ with positive components d^+ and negative components d^- . Positive distance components are on one side of the chord while negative distance components are on the other side.

We expect that arc points will be on one side of L and distances will therefore have the same sign for arcs whereas S shaped curves will have points on both sides of L and distances will therefore have different signs. These S shapes occur when concave and convex contours are grouped. To prevent from forming these groups, we derive the *concavity* measure as given in Eq. (4-12). Concavity takes values in the [0-1] range in which arcs will have values close to zero whereas S shapes will have values close to 1.

$$\text{concavity} = \frac{\min(d^+, d^-)}{\max(d^+, d^-)} \quad 4-12$$

Figure 4-13 (a) demonstrates an example where a concave and a convex contour are paired. Figure 4-13 (b) shows the $\vec{D}$ plot which has negative distance values for the contour on the left hand side and positive distance values for the contour on the right hand side. For this pair, the *concavity* measure is 0.75.

Figure 4-13 Computed measures for this pair: *concavity*=0.75, *peakValue* = 0.5, *Bell-Existence* = 0.72, *pseudo-curvature* = 0.15 (a) Two contours and the chord between their end points. Note that points are on both sides of the chord, (b) Point-to-chord distance plot, $\vec{D}$, (c) absolute value of point-to-distance plot, $|\vec{D}|$.

Since we are favoring arc shapes, we expect a bell shape in the distance plot. Similar to Section 3.3.3.5, we use the *Bell-Existence* feature to observe this characteristic. Before calculating the *Bell-Existence* feature, we smooth the plot to remove any jaggedness. We also take the negative of the plot, if the maximum distance value from the chord is negative, (e.g. Figure 4-14 (a)). Then we calculate *LeftRatio*, *RightRatio*, and *skewness* on the smoothed distance plot, as presented in Section 3.3.3.5.

Another observation is that the absolute distance plot of arc shapes will have only one peak, whereas squiggly groups will have more than one peak, as illustrated in Figure 4-13 (c). To penalize this behavior, we derive a *peakValue* measure (given in Eq. (4-13)) using the number of peaks in the absolute distance plot. Let PS be a set of peaks in $|\vec{D}|$,

then the cardinality of PS will give the number of peaks. When obtaining the peaks, we looked for peaks 30 points apart from each other to avoid the wrong estimation of the number of peaks. The measure *peakValue* takes values in the [0-1] range. For a single peak, *peakValue* will be 0, while as the number of peaks increases, *peakValue* gets closer to 1.

$$peakValue = 1 - \frac{1}{|PS|} \quad 4-13$$

In different real life applications (e.g., in the road extraction problem), linear contour groups might be desired. To support them in our system, we keep the *pseudo-curvature* which measures how contour groups approach a circular arc and deviate from the straight lines. We consider the maximum value in the absolute distance vector as the chord height, $h = \max(|\vec{D}|)$, which tends to be small for lines and high for circular arcs. The ratio of $2h$ over the chord length $||L||$ should be close to 1 for a semi-circle. We use Eq. (3-13) to calculate the *pseudo-curvature*.

Another observation that we have made is that the gradient of the absolute distance plot has a high deviation from zero when a big discontinuation happens. This high deviation can be seen better in the gradient absolute distance plot as illustrated in Figure 4-14 (e). To examine this kind of deviation, we use the maximum absolute value in the gradient of absolute distance plot in a measure called *deviation*.

From the point-to-chord distance, we compute the *concavity*, *pseudo-curvature*, *peakValue*, *Bell-Existence*, and *deviation* metrics. Figure 4-14 demonstrates different contour groups, their distance plots, and the measures based on the distance plot. Figure 4-14 (a,b) show positive examples (i.e., these contours belong to the same group) of

contour grouping, and the *concavity* and *peakValue* measures are zero while the *Bell-Existence* measure is low for these positive examples. Figure 4-14 (c, d, e) show negative examples (*i.e.*, these contours do not belong to the same group) of contour grouping. Notice that the number of peaks is two in these examples, showing that *peakValue* is a good indicator for contour grouping. *Bell-Existence* values in these negative examples are also higher than the values in the earlier positive examples. *Concavity* eliminates cases such as the ones in Figure 4-14 (c, d). The *Deviation* value is higher in Figure 4-14 (d, e, f) than in the other cases.

= 1.06

(c)

Concavity = 0.5, Bell-Existence=0.81, peakValue=0.5, pseudoCurvature = 0.52, deviation = 6.46

(d)

Concavity = 0, Bell-Existence = 0.71, peakValue = 0.5, pseudoCurvature = 0.52, deviation = 4.07

(e)

Concavity = 0, Bell-Existence = 72, peakValue = 0, pseudoCurvature = 0.23, deviation = 14.3336

(f)

Figure 4-14 Different combinations of contours along with the point-to-chord distance measures, (a, c, d) show both the distance and smoothed absolute distance plots, (b) shows only distance plot, (e) shows only the absolute distance plot, (f) shows both the absolute distance plot and gradient distance plot

We have also experimented with fitting a polynomial to the distance plots; however, normalization of the error-of-fit values for different kinds and different lengths of contour groups was not convenient and the generated measures on the error-of-fit values were not reliable.

4.2.5 Proximity

Proximity, ρ , measures the closeness of two contours. As Gestalt declares, humans create association among close contours. We use the minimum distance among the end points of two contours as the proximity measure as shown in Eq. (4-14).

$$\rho = \min (d_1, d_2, d_3, d_4) \quad 4-14$$

If there are good candidates for contour grouping, we should favor the close ones over the farther ones. If the farther one is also part of the group, the grouping algorithm should include the farther one into the group later. *Figure 4-15* (a) illustrates a case where C_1 is looking for grouping options and both C_2 and C_3 are good candidates but favoring close contours yield more complete contour groups.

Figure 4-15 The role of proximity. (a) For C_1 , both C_2 and C_3 are good candidates. We favor the close contour C_2 over the farther contour C_3 . (b) If C_1 and C_2 form a group, then should we include C_3 into the group? C_3 is far away and its length is shorter than the minimum distance between C_3 and the group.

In real images, we observe that adding a very far and short contour to existing coherent long groups hurts the accuracy of the results. Hence, we derive a

distanceToLength (as given in Eq. (4-15)) measure which is the ratio of the minimum distance between two contours to the minimum cardinality of their point lists. In particular, including a short contour into a long contour group is risky. Therefore, we also take length ratio of the contours into account with the *lengthRatio* feature given by Eq. (4-16).

$$distanceToLength = \rho / (\min(|C_1.\mathcal{P}|, |C_2.\mathcal{P}|)) \quad 4-15$$

$$lengthRatio = \min(|C_1.\mathcal{P}|, |C_2.\mathcal{P}|) / \max(|C_1.\mathcal{P}|, |C_2.\mathcal{P}|) \quad 4-16$$

4.2.6 Weighing the measures

One of the biggest challenges of contour grouping problems is combining all of the parameters in a sound cost function. Previous studies do not make a difference between the pair-wise cost and group-wise cost. We observed that adding a contour into an existing group requires different measures compared to combining two contours. So, we propose two different cost functions one for pairing two groups and another for adding a contour into a group.

Figure 4-16 A synthetic image used for pair and group sample generation

To observe which factors play a role in the pair-wise cost and the group-wise cost, we created a synthetic image shown in *Figure 4-16* and generated contour pairs and contour groups from the image. We pair a contour with its neighbors on each end and assign a label for each contour pair. If they belong to the same group, then we assign a positive label, otherwise we assign a negative label. Some positive and negative contour pairs are shown in *Figure 4-17*. In all, we generated 18 positive and 76 negative contour pairs from the training image shown in *Figure 4-16*.

Figure 4-17 Generated contour pairs, (a) positive pairs, (b) negative pairs

We expand positive contour pairs and generate contour groups. Note that each contour group has more than two contours. We label each generated contour group as positive or negative and repeat expanding until there is no expansion options left for the groups. *Figure 4-18* illustrates some samples of positive and negative contour groups. Note that labeling can be different for different applications; here we considered the requirements of coronal loop characteristics. For example, in river detection from satellite

images, the right-most combination in the bottom of *Figure 4-18* (b) could be considered positive. From the synthetic image in *Figure 4-16*, we generated 50 positive and 424 negative contour groups.

Figure 4-18 Generated contour groups, (a) positive groups, (b) negative groups

In the end, we have two different training data sets, one for contour pairs and the other for contour groups. We present the distributions of the feature values of the training data in **Table 4-1**. Red points represent the positive instances while blue points represent

negative instances. We can observe that most of the measures are not very discriminative for the pair-wise case. However, the group-wise features behave differently from the pair-wise features. For example the necessity of certain measures such as peak-value, angle dissimilarity, and distance to length measures can be observed better in the group-wise measures.

Table 4-1 Pair-wise versus group-wise measures (Red = Positive, Blue = Negative)

Measure Name	Pair-wise	Group-wise
Angle Dissimilarity		
Ellipticity		
Eccentricity		
BellExistence

We trained a decision tree to see which features are selected by the decision tree (given in *Figure 4-19*), and the decision tree selected *ellipticity*, *concavity*, *bell existence*, and *deviation* for the pair-wise instances.

Figure 4-19 A decision tree built for pair-wise instances

We also trained a decision tree for the contour group instances as shown in *Figure 4-20*. We can see that different features are more discriminating in this case. Thus, using the same cost function for contour-pairs and contour-groups can be misleading for contour grouping.

Figure 4-20 A decision tree built for group-wise instances

We computed the information-gain measures for both types of instances as shown in *Figure 4-21*. For the pair-wise instances, only the ellipticity measure generates a non-zero value, while the rest of the measures take on zero values. For the group-wise instances however, almost all of the measures have an effect on the decision. The decision tree built for the group-wise instances given in *Figure 4-20* confirms information-gain values.

Figure 4-21 Information gain values for group-wise and pair-wise instances. Note that only the ellipticity value is positive among the pair-wise instances.

Based on the findings on the pair-wise and group-wise measures, we define two different costs. If we want to combine two contours, C_1 and C_2 , then we calculate the pair-wise cost, $\phi(C_1, C_2)$, given by *Eq. (4-17)*. We train a single layer neural network classifier without feature normalization to estimate the weights of the parameters and a threshold value for the class decision; *i.e.*, if the weighted summation of the feature values is under a certain threshold, then the instance is assigned to one class otherwise; it is assigned to another class. Let F be generated features (*i.e.*, *angleDissimilarity*, *distanceToLength*, *Concavity*, *Eccentricity*, *Ellipticity*, *pseudoCurvature*, *peakValue*,

BellExistence, Deviation, lengthRatio). Let WP be the weights and TP be a threshold, all generated by the Multiperceptron neural network for positive versus negative contour-pair discrimination. Then, the pair-wise cost $\varphi(C_1, C_2)$ of the contour-pairs is given by

$$\varphi(C_1, C_2) = \sum_{i=1}^{|F|} WP(i) \cdot F(i) \quad 4-17$$

Using the same procedure for the cost function for contour-groups, we train a single layer Multiperceptron without parameter normalization. Let WG be the weights generated for contour-groups and TG be the threshold generated by the Multiperceptron for the contour group. When we add a contour C_l to an existing group G_l , we compute the group-wise cost, $\psi(G_l, C_l)$, given in Eq. (4-18).

$$\psi(G_l, C_l) = \sum_{i=1}^{|F|} WG(i) \cdot F(i) \quad 4-18$$

For different applications, the weights of the parameters in the pair-wise and group-wise costs could be different. It is therefore good to generate several positive and negative instances and train a classifier with these instances before establishing the cost functions.

4.3 Contour Grouping

When contours are associated with each other, they form a contour group G which is defined by a group label GL , group energy GE , a set of contours $\mathcal{C}$, and $\mathcal{P}$ is the combined point set of all the contours in the group. GL is an element of the label set X . A contour group must have at least two contours. We define a contour group as follows:

Definition 4-2 *Group* = $(GL, GE, \mathcal{C}, \mathcal{P})$ where GL is the group label, GE is the group energy, $\mathcal{C}$ is the set of contours in the group, and $\mathcal{P}$ is the combined point set of all the contours.

The set of all contours in the image is denoted by $\mathfrak{S}$. Let n be the number of segments in the image, then $\mathfrak{S} = \{\mathfrak{S}_1, \dots, \mathfrak{S}_n\}$. In a cluttered image, contours could be either part of salient groups $\mathcal{G}$ or part of the background $\mathcal{B}$. The contours in $\mathfrak{S}$ are divided into two subsets such that $\mathfrak{S} = \mathcal{G} \cup \mathcal{B}$ and the intersection of two sets is empty $\mathcal{G} \cap \mathcal{B} = \emptyset$. Let k is the number of salient groups in an image, then $\mathcal{G}$ is the set of all contour groups in the image with $\mathcal{G} = \{G_1, \dots, G_k\}$. The background $\mathcal{B}$ is comprised of the contours which are not part of any salient group. If a contour $\mathfrak{S}_i$ is part of the background model, then its group label GL is 0, otherwise the label is the label of its group.

At the beginning, all contours belong to the background $\mathcal{B}$ and there are no contour groups in $\mathcal{G}$. The initial label set X has only one value, $X = \{0\}$. When contours form new groups, we expand X with new group labels. Grouping the contours mean changing their group labels when necessary and creating new groups G or updating the $\mathcal{G}$ list and the label set X .

The problem of extracting the salient contour groups from the cluttered background can be stated as finding an optimal configuration X^* which results in a minimum sum of costs for the salient contour groups $\mathcal{G}$ and the contours in the background model $\mathcal{B}$ as given by

$$X^* = \arg \min \{U(X) = \sum_{G \in \mathcal{G}} V_G(X_G) + \sum_{C \in \mathcal{B}} V_C(X_C)\} \quad 4-19$$

The first term in the configuration in Eq. (4-19) is the summation of contour group costs $V_G(X_G)$. The second term is the cost $V_C(X_C)$ summation of the contours X_C in the background $\mathcal{B}$. If a contour is part of the background $\mathcal{B}$, then it has a fixed cost A as shown in Eq. (4-20). The value of A is selected using the Multiperceptron generated threshold value T .

$$V_C(0) = A \quad 4-20$$

Before performing optimization, we initialize the contour labels. Every optimization problem requires a good initialization which accelerates reaching the global solution. For the contour grouping problem, if the contour pairs with small cost are grouped in the beginning, we can reach the solution much faster. To initialize the system, we build a cost matrix, $CM = [\infty]_{n \times n}$ where n is the number of discrete contours in the image. Then, we compute the pair-wise costs of each contour between its neighbors and update the cost matrix.

In the initialization phase, we generate only contour groups consisting of two contours. To find good seeds, we scan the cost matrix starting from the first row to the last row. Let $CM(i, j)$ be the minimum of the i^{th} row and let $CM(j, k)$ be the minimum of the j^{th} row. To consider the pair (C_i, C_j) as a seed, it should hold the following condition: $\{CM(i, j) < TP \text{ and } i=k\}$. If both i^{th} and j^{th} contours result in the minimum pair-wise cost among their combination and if the pair-wise cost is less than TP (i.e., the threshold value generated by the Multiperceptron neural network for contour-pairs), then we consider this pair as a seed pair and combine them and assign a new group label for them. When there is a new group, we just increment the number (GN) of existing groups and assign the new

number as a group label for the group. For the group energy, we calculate the new contour group's group-wise cost using *Eq. (4-18)*.

If a contour does not have any pair-wise cost under the threshold, its group label remains zero. We are very conservative in the initialization and only generate reliable seeds for the optimization problem. Since our contours are ordered by their saliency measure, it is likely that most salient contours will generate seeds in the initialization phase. *Figure 4-22* shows the initial seeds for the image used to generate the training data. For this image, the system generates seven contour-groups which are shown with different colors. It is possible to generate more groups with fewer conditions.

Figure 4-22 Initial seeds for the contour grouping algorithm. Each group is represented by a different color

After obtaining the initial seeds, we have a set of contours with group labels and the rest of the contours still belong to the background. With the acquired initial group labels, we update the other contours' group labels. In this phase, we examine each contour separately to see whether the contour is related to any other contours in its neighborhood N_1 , N_2 . Before describing the label updating, we want to remind the reader

that each contour could be related to at most one other contour on each end to form smooth curves and not wishbone structures.

A typical Markov Random Field changes the label of a site based on its neighbors' labels. We reverse the approach and tend to keep the label of the site but trying to change the labels of its neighbors. We start from the most salient contour and look for label updating options in the neighborhood N_l and N_2 . Let V_l be the label set of N_l for a contour $\mathfrak{S}_i$ and V_2 be the label set of N_2 and l be the group label of the contour $\mathfrak{S}_i$, such that $l > 0$.

We have following conditions for label updating:

- Condition 1: If $l \in V_1$ and $l \in V_2$, then this contour is connected to its neighbors as shown in *Figure 4-23 (a)*. However, this connection may be a local optimum and might need to be broken. Therefore, for each end, we eliminate the contour ($\mathfrak{S}_j$) which has the same label from the group, $G_l = \mathfrak{G}_l - \{\mathfrak{S}_j\}$ and add the other contours in that neighborhood into the shrunk group separately and compute their group-energy. If any new combination satisfies the following condition $\frac{GE(G_1)}{|\mathcal{R}(G_1)|} < \frac{GE(G_l)}{|\mathcal{R}(G_l)|}$, then we accept the change that requires setting the label of $\mathfrak{S}_j$ to zero and setting the contour ($\mathfrak{S}_m$) resulting in a lower cost/length ratio to l . If this new contour belongs to another contour group, we need to ensure that this change does decrease the total cost of the system and not increase it.

- Condition 2: If $l \in V_1$ xor $l \in V_2$, then this contour is connected to a contour either in N_l or N_2 (as shown in *Figure 4-23 (b)*). Let N be the neighborhood where there is no label l among the contour labels. Let N^+ be the neighborhood where there is a contour

label l . First, we search for any label change and group updating in N , then we look for label updating options in N^+ .

If the second condition holds, we check how any label change in the neighborhood affects the cost of the entire model. When a contour is associated to an existing contour group, we have to be sure that this expansion will not harm the consistency of the group and will work in favor of reaching the global solution.

Then, for each contour in the neighborhood ($\forall \mathfrak{J}_j \in N^-$), we compute the cost of including a contour ($\mathfrak{J}_j$) into the group of $\mathfrak{G}_l$. First we create an expanded group $G_l = \mathfrak{G}_l \cup \{ \mathfrak{J}_j \}$, then we compute the group-wise energy GE of the expanded group G_l using Eq. (4-18).

To change the label of a contour in the neighborhood, we check that the energy difference between before and after this change is negative. Since our goal is to obtain long and smooth semi-elliptical open curves, we check the ratio of the group energy to group length for both the newly generated group G_l and the old group $\mathfrak{G}_l$. If the cost/length ratio of G_l is smaller than that of $\mathfrak{G}_l$, then this change could be useful to reach the global optimum.

$\mathfrak{J}_j$ could belong to the background model or to another contour group. We handle these situations differently.

- If $\mathfrak{J}_j$ belongs to the background model (as shown in *Figure 4-23 (b)*), then the energy difference of the label change will be as follows:

$$(\Delta E = \frac{GE(G_1)}{|\mathcal{R}(G_1)|} - \frac{GE(\mathfrak{G}_l)}{|\mathcal{R}(\mathfrak{G}_l)|} - \frac{A}{|\mathcal{R}(\mathfrak{J}_j)|})$$

- If $\mathfrak{J}_j$ belongs to another contour group (say group k) as shown in *Figure 4-23* (c, d), we analyze possible changes with two different scenarios:

1. **Scenario 1:** (Possibility of merging two groups) First we merge two groups, $G_l = \mathfrak{G}_k \cup \mathfrak{G}_l$, then calculate the group energy of the merged group G_l . If the merged group results in a lower group energy than the sum of energies of the separate groups $\mathfrak{G}_k$ and $\mathfrak{G}_l$, then we accept merging these two groups, otherwise we reject the merging ($\Delta E = GE(G_l) - GE(\mathfrak{G}_l) - GE(\mathfrak{G}_k)$, $\Delta E < 0$). If we accept merging, then we change the group labels of the contours in $\mathfrak{G}_l$ with k and remove $\mathfrak{G}_l$ from the salient group list $\mathfrak{G}$. We also update the point list, contour list and group energy of $\mathfrak{G}_k$. *Figure 4-23* (c) illustrates sample contour groups of this scenario. Changing a label at a time would not converge in this case, therefore merging is the solution. Note that this time we do not divide the energies by the contour lengths, since a longer length of a merged contour can mislead the decision process.

Figure 4-23 Sample contour groups along their labels

2. **Scenario 2:** (Possibility of changing a group label of $\mathfrak{J}_j$) In some cases, a contour could be part of a wrong group (possibility a local optimum), and the system has to take that contour out of the wrong group and include it into the right

group. *Figure 4-23 (d)* illustrates two groups where a label change of $\mathfrak{J}_j$ resulted in lower global energy. In this situation, we have to exclude $\mathfrak{J}_j$ from $\mathfrak{G}_k$ in addition to expanding $\mathfrak{G}_l$ with $\mathfrak{J}_j$. We create a shrunk group, $G_2 = \mathfrak{G}_k - \{\mathfrak{J}_j\}$ and compute the group energy of G_2 . If only one contour is left in G_2 , then we assign the background cost A to the group energy of G_2 . The total energy change will be as follows:

$$\left(\Delta E = \frac{GE(G_1)}{|\mathcal{P}(G_1)|} - \frac{GE(\mathfrak{G}_l)}{|\mathcal{P}(\mathfrak{G}_l)|} + \frac{GE(G_2)}{|\mathcal{P}(G_2)|} - \frac{GE(\mathfrak{G}_k)}{|\mathcal{P}(\mathfrak{G}_k)|} \right)$$

At the end of label changing alternatives for the neighbors of contour $\mathfrak{J}_i$, we have a set of energy differences. If the minimum energy difference is negative (i.e., an update decreases the total energy of the system), then we accept the change, otherwise we keep the labels as they are.

We iterate the label updating for each contour in the system, until the system cost reaches a stable value. During optimization, we use simulated annealing to avoid local optima and first assign a high value to A which is the cost of belonging to the background. In the following iterations, we decrease A gradually to escape local minima. Our algorithm generally converges in two or three iterations. Algorithm 4.1 illustrates the steps of our approach. The input of the algorithm is a set of unlabeled contours and the output of the algorithm is a set of labeled contours and generated contour groups.

Algorithm 4-1 Contour Grouping

Input: $\mathfrak{J}$

Output: $X, \mathfrak{G}$

Variables:

CM : the cost matrix, $CM = [\infty]_{n \times n}$

X : the label set, $X = \{0\}$

GN : number of groups, $GN = 0$

TP : pairing threshold

TG : grouping threshold

Initialization:

for each contour $\mathfrak{S}_i$ in $\mathfrak{S}$

for each neighbor $\mathfrak{S}_j$ of $\mathfrak{S}_i$

$$CM(i,j) = CM(j,i) = \phi(\mathfrak{S}_i, \mathfrak{S}_j)$$

end for

end for

for each contour $\mathfrak{S}_i$ in $\mathfrak{S}$

$CM(i,j)$: minimum pair value of $\mathfrak{S}_i$

$CM(j,k)$: minimum pair value of $\mathfrak{S}_j$

if $CM(i, j) < TP$ and $i = k$ and $GL(\mathfrak{S}_i) = 0$, *then*

 Increase GN by 1

 Form a group structure, $\mathcal{G}_{GN} = \{GN, \psi(\mathfrak{S}_i, \mathfrak{S}_j), \{\mathfrak{S}_i, \mathfrak{S}_j\}, \mathcal{P}\}$

 Update X and $\mathcal{G}$

end if

end for

Label Updating:

$U_0(X)$: total energy of the system

$$A = (1 + \alpha_0) \cdot TG$$

repeat

$$A = (1 + \alpha_t) \cdot TG \quad // \text{Decrease } A \text{ by decreasing } \alpha \text{ value } \alpha_t < \alpha_0$$

for each contour $\mathfrak{S}_i$ in $\mathfrak{S}$

if $\mathfrak{S}_i (GL) > 0$

 Generate dummy groups by changing the labels of neighbor contours in N_1 and N_2

 Make necessary group energy calculations for each change

 Calculate the energy differences between the current energy and possible energy of change

<p style="text-align: right;">If any change results in lower cost/length ratio than the current group of $\mathfrak{F}_i$, then accept the label change, update $\mathcal{L}$ with the group changes, else reject the change and keep the labels.</p> <p style="text-align: right;">end if</p> <p style="text-align: right;">end for</p> <p>until $U_t(X) < U_{t-1}(X)$</p>

The complexity of the algorithm is proportional to the number of contours (n) and the number of neighbors of the contours. Let *AvgNeigh* be average number of neighbors for contours. The total complexity of the algorithm is the sum of initialization part and label updating part: $O(n) + O(n * AvgNeigh)$.

After obtaining the optimal labels of the contours, we have a set of contour groups. However, there might be gaps within the contours in these groups. Therefore, we close these gaps using straight lines, and then perform B-spline fitting to smooth the contour groups.

4.4 Experimental Results

We tested our system on three different image sets: synthetic images, coronal loops in solar images, and roads in aerial images. The first set is a collection of synthetic images that we created to represent possible challenges in real images. We tested our algorithm with several synthetic cluttered images that contain multiple intersecting contour groups in different orientations and sizes. Our technique successfully separated the background from the salient contours successfully as illustrated in *Figure 4-24*.

Figure 4-24 (a) A sample of a synthetic cluttered image, (b) Salient contour groups obtained with our technique

We then applied our method on cluttered solar images to automatically delineate coronal loops. The solar data consists of input images that were generated using the Ridgelet transform described in the (Inhester, et al., 2007) on solar images from STEREO/SECCHI in $\lambda=171^\circ$ wavelength. We were also given the desired ground truth results from the cluttered regions in each input image. We adjusted the weights in the pair-wise and group-wise costs based on the given ground truth. Our algorithm reaches the optimal solutions for the given test images with 90% accuracy. The average time taken to reach an optimal solution is between 30 and 50 seconds depending on the amount of clutter. Note that we implemented our algorithm in MATLAB and used a computer with 2.1 Ghz Dual core, 4GB RAM, and 64 bit operating system. Figure 4-25 (b, d) demonstrates sample outputs of our algorithm on solar images. In addition to elliptical curves, we kept long and smooth curves here as in the provided ground truth.

Figure 4-25 (a, c, e) Cluttered solar image regions after Ridgelet transform (Inhester, et al., 2007) on STEREO/SECCHI images, (b, d, f) Sample outputs

We also tested our technique on TRACE images such as the one shown in *Figure 4-26* (a). First we clean the image and obtain the curves as shown in *Figure 4-26* (b). Since, the results had gaps along the loops, we applied our technique to combine related loop segments and obtain the result in *Figure 4-26* (c). We performed the B-spline technique to smooth the detected contour groups but even though this yielded smoother results, it caused data loss in the ends of the contours as shown in *Figure 4-26* (d). We noticed the similar data loss due to B-Spline in *Figure 4-25* as well. The results after filling the gaps can be used if the users do not desire any data loss.

Figure 4-26 Contour grouping results on a TRACE image, (a) original image, (b) cleaned image, (c) salient contour groups before B-spline fitting, (d) salient contour groups after B-spline fitting

Another application area that requires contour grouping is road detection in aerial images. The algorithm proposed in (Bacher, 2004) detects the main roads but fails in the urban roads or the roads in agricultural regions as shown in *Figure 4-27* with the road network containing gaps in those regions. We improve the results of (Bacher, 2004) using our contour grouping as shown in *Figure 4-28* (b).

Figure 4-27 Road detection in aerial images. The results of (Bacher, 2004) (a) original IRS⁸ image, (b) detected roads (Bacher, 2004)

⁸ IRS data: www.nrsc.gov.in

The road detection problem is different from coronal loop detection, since roads are generally linear or *S* shaped. Therefore, most of the measures defined for the coronal loop detection problem are not needed in road detection, and the most important measures are linearity and proximity.

We also observed the required differences during curve segmentation and neighbor association. In this case, we narrowed down the search space by decreasing the tolerance angle α but increasing the side lengths of the triangle in which we are searching. As an input to algorithm, we used the result of (Bacher, 2004) shown in *Figure 4-27* and kept only the white pixels of the image. Then we ran our contour grouping method on the thinned image. The gaps among the roads are shown in *Figure 4-28 (a)* and the output of our algorithm is demonstrated in *Figure 4-28 (b)*. Our algorithm closed the gaps existing in *Figure 4-28 (a)*. If Steger's method (Steger, 1998) is used for road extraction, we could extract more roads and the results of contour grouping could reflect the exact road network.

(a)

(b)

Figure 4-28 Road detection results (a) the input of our algorithm, (b) the output of the algorithm.

4.5 Summary

There are two main components in the salient contour grouping approach: discrete contour extraction and contour grouping. For a given image from a certain domain, first we obtain the binary image by applying appropriate image preparation techniques, then we perform curve tracing to acquire curves from the image, and segment the curves at corner points or inflation points to attain smooth and squiggle-free contours. Considering the requirements of the application domain, we assign saliency measures to the individual contours, and associate each contour with its neighbors. At the end of the contour extraction phase, we have a set of contours to be used in contour grouping.

In the contour grouping phase, we first determine which perceptual rules or shape priors play a role in the application domain. To combine different perceptual rules in a cost function, we train a Multiperceptron classifier with positive and negative samples and obtain the weights of the perceptual rules that should be used in the application domain. In our contour grouping solution, contours can belong to either clutter or salient contour groups. If they are part of clutter, then a fixed cost is assigned to them. If they belong to a group, then the group cost is computed using the weights generated by the Multiperceptron. Then our system changes the labels of the contours and computes the total cost of the entire model. We continue the iterative changing of the labels of the contours and recompute the total cost of the entire model until the model reaches stability. The optimization phase gives us the labeled contours and the detected salient contour groups.

We tested our contour grouping approach on synthetic images, then tested it on coronal loop highlighting in cluttered images, and finally on road detection in aerial images. We achieved the optimal results in two iterations on average.

5 CONCLUSION

“I have not failed. I've just found 10,000 ways that won't work.”

~Thomas A. Edison

In this dissertation, we addressed two different problems: coronal loop detection from SOHO/EIT images and extracting salient contour groups from cluttered images. In the scope of the first problem, our contributions are towards curve tracing, feature extraction, feature selection, and developing an image retrieval tool for the coronal loop detection problem.

Our coronal loop detection system has evolved in time due to the challenges imposed by a new real life interdisciplinary problem. We first started with raw images and the block-based approach, and then switched to IDL ssw software-based cleaned images but still using the block-based approach. Lastly we resorted to a contour-based approach on the cleaned images. While in the early stages of this project, we were hardly able to exceed 40% accuracy, we have now reached 90% accuracy.

For the second problem, we proposed a contour grouping method based on Gestalt-inspired perceptual rules, Markov Random Fields, and novel features that are specialized to the contour extraction, grouping and classification problem. We have thus

defined a new ellipticity measure to merge different contours, and derived new measures from the signed point-to-chord distance plot. With the new shape features that we derived, we were able to successfully extract coronal loops from cluttered images. The amalgam of different perceptual rules was judiciously combined by training a Multiperceptron neural network and we observed via training a decision tree how different factors play a role in the combination of two contours as opposed to the combination of more than two contours. In addition to successful tests on mining coronal loops from solar images from two different instruments onboard two different satellites aimed toward the sun (TRACE and STEREO/SECCHI), we have also tested our method on road detection in aerial images showing the ability of our approach to close the gaps in road networks.

Our study is an interdisciplinary study between the fields of astrophysics and computer science. We have investigated several image cleaning techniques, features, classifiers, and several approaches to handle the imbalanced data problem, etc., many of which not performing to our expectations.

Below, we make conclusions from our research work in Section 5.1 and outline several future research directions in Section 5.2.

5.1 Discussion

Pattern recognition and machine learning on *real* data in new problems, is really challenging and is far from perfect scenarios of achieving almost magical results on *synthetic* images. It is also unlike applying pattern recognition and machine learning algorithms on 30 year old data sets (*e.g.*, segmenting the same tiger out of green grass,

detecting the same London busses, etc.), which are cited by hundreds of studies. In new real-life applications, every sample tells another story and may call for different off the beaten path approaches. In our case, loop shapes are very diverse in their shape, size, and direction, and in some cases, they are very hard to distinguish (even to the untrained human eye) from other solar phenomena that occur on the solar corona. Extracting common features for all positive samples and distinguishing positive samples from negative samples can be almost impossible. At times, 40% accuracy made us happy, and each 1% increase in accuracy seemed like a miracle. We have tried a large number of approaches over the years of this project, and in the end achieved 84% F1-score from cross-validation and 90% accuracy from the image based testing tool on unseen solar data, which is considered reliable given all the challenges of automatic detection of coronal loops.

Working with imperfect data: At the beginning of this project, we started working with noisy raw images. Because the instrument related grid artifacts in the raw images were really hurting the accuracy of the system, we devoted a lot of effort to clean the images and extract individual coronal loops from extremely noisy images with a variety of techniques, many of which have been described in this dissertation, and others have been omitted. Towards the end of this stage we have developed a system with 67% accuracy, which was neither wonderful nor desperate.

The IDL solar software (ssw) tools that have been developed by astrophysicists over the years for cleaning some instrument linked defects like grids were not mentioned in publications. But we were fortunate to discover them after our meeting and interactions

with solar physics experts during the Solar Image Processing workshop in 2008, a fruitful meeting for researchers from several disciplines.

Inconsistent markings: Another problem in our interdisciplinary project was the lack of consistency in the expert marked regions. When we were analyzing the loop markings, one structure was marked as a loop in one image but was not marked in another image. This problem, which was common in the training data, was making automated learning by a classifier impossible.

Data loss with cleaning: After discovering and using the IDL ssw solar software to clean solar data, we were faced with a new disappointment: This tool does not only delete the instrument caused grid artifacts, but it also deletes all the faint loops. Despite the deletion of faint loops from our training data, the markings that indicated their presence were still there. Thus “to clean or not to clean using IDL???” was another question to be answered. Since, this project’s goals were to support the data sifting needs of astrophysics experts; we decided to continue adopting the IDL based cleaning of images.

Extracting the right features: Extracting the right features is as important as working on the right images or having consistent training data. Analyzing the given positive and negative samples and investigating the right features according to the requirements of the problem can increase the accuracy and performance of the detection system significantly. During feature extraction, we should analyze how much that feature is affecting the result and how much time is required to extract that feature. If a feature does not affect the accuracy and its extraction takes forever, it is good to give up on that feature. For example, this was the case for the texture features that we investigated to

reduce the effect of grid artifacts and Hough-based features especially without quantization of the Hough space. In our case, the most reliable features are curvature and shape features.

Working on the right entity: In pattern recognition, the entity we are extracting features from is also important. First, we extracted features from fixed-sized blocks to obtain the exact position of the loops. Then, we changed the entity type and extracted features from the individual contours acquired from a strip around the Sun. This was a big step forward in our study. It not only resulted in a big increase in the accuracy but also a decrease in the false alarms we were getting using the block-based approach. After the contour-based approach, we carried the same learning model to the loops inside the solar disk and achieved very accurate results. With the block-based approach, we were supposed to train two different models for inside and outside the solar disk. Also the imbalanced data of the in-disk samples was a big problem in the block-based approach. We can say that it is good to try different entities in pattern recognition problems. Observing the pattern from different point of views might yield surprising results.

Solutions for Imbalanced Data: In the block-based approach, we were also suffering from an imbalanced data problem, since the ratio of loop blocks to non-loop blocks was about 1 to 20. We tried the SMOTE approach (Chawla, et al., 2002) that increases the minority class samples by generating fake samples based on the given minority class samples while under-sampling the majority class randomly to improve the classification accuracy. Although this method increased the accuracy in cross-validation (in fact increasing the precision from 60% to 97%) it decreased the accuracy on unseen

data, indicating the occurrence of overfitting. Balancing the classification accuracy on unseen data is one possible future direction.

Training samples should be diverse: We have investigated generating different data models for different solar cycles; however this has not helped the classification accuracy. The solar images from minimum cycles do not have enough positive samples to teach a classifier what a coronal loop is, while those images from the maximum cycle lack examples of quiet regions and thus do not teach the classifier what a non-loop region is very well. As a result, we achieved the highest precision and recall value when we combined the images from all the cycles together to train the model. Therefore, obtaining a generic model using diverse samples seemed more promising in our case and the comparison of generic training versus specialized training could be another future research direction.

Some classifiers perform better on unseen data: Another conclusion that we have made is that even though some classifiers (e.g., Naïve Bayes, RIPPER) seem to achieve equally good results on cross-validation results, they are not that successful on unseen data. In our case, Adaboost based on C4.5 was always yielding fewer false alarms and higher true positives on unseen data. This confirms the wisdom of never relying only on cross-validation results, when comparing different classifier models, and instead testing the models on unseen data. The comparison of the accuracy of several classifiers on unseen data versus in cross-validation is worth to analyze.

Feature selection might hurt: Even though feature selection seems to not hurt the results during cross-validation experiments, it might hurt the accuracy on unseen data,

confirming that one should always test the generated model from the selected features on unseen data before judging the reliability of feature selection.

Curve tracing is a challenging problem: In extremely noisy images, curve tracing becomes very challenging. If we pick the point giving the minimum cost value in a close neighborhood as in Algorithm 3-1, we might miss the trace of the correct curve. If we consider farther traces and pick the next point, then we increase the time complexity. Even though we increase the accuracy, it is still very challenging to extract coronal loops from clutter. It is better to clean the image as much as possible, and then run the curve tracing algorithm. Another strategy to avoid local search by looking for a continuation point in wider regions in noisy environments, turned out to increase the risk of ending up with wrong curves, and thus actually hurting global search. Thus instead of looking for farther points through curve tracing, it is better to obtain adjacent points and perform contour grouping later, to increase the chance of obtaining correct curves.

Over-segmentation versus under-segmentation: Some of the main critical decisions in curve segmentation are: (i) finding the right window size, (ii) finding the threshold values to deal with every kind of curve in the image, (iii) catching subtle transitions while avoiding over-segmentation at the same time, (iv) escaping from small jitters but still detecting all the corner points or inflation points. All these decisions seem to benefit by hurting other decisions, thus working in a trade-off relationship with each other. Thus generalizing curve segmentation for every kind of image is challenging and depending on the application, we should make the right decisions and obtain the curve segments maximizing the final goal.

Using different cost functions for contour-pairs and contour-groups:

Combining two contours is different from adding a contour into a consistent long contour group. Thus using the same cost function for contour-pairs and contour-groups could be misleading. Therefore, we use two different cost functions for contour grouping. We show that the necessity of two different cost functions through two different training sets. Also, the contour grouping model should be powerful enough to resist including clutter elements into the salient contour groups.

Contour grouping is context dependent: In different application domains, human beings can sense the context and perceive the entities using their previous knowledge. As humans, we know that roads are different from rivers in aerial images. Using our previous knowledge, it is easy to perceive them distinctly in aerial images. Therefore, using only proximity or smoothness for every kind of application or proposing a generic contour grouping model is not realistic. Shape priors could be very useful if we are searching for contour groups of a certain shape, and observing the nature of the domain is an obligation, to give more information to the contour grouping algorithm.

Good initialization is the key in optimization problems: When we generate good seeds to start with, we can reach optimal results very quickly and accurately. Good seeds also require some domain knowledge. Thus we need to define a saliency measure by considering the application requirements, and we should generate seed pairs considering the application as well. The system should also handle wrong seeds and converge to desired results. Our label changing approach updates the labels and gets rid of the wrong seeds.

5.2 Further Directions

With the launch of NASA's Solar Dynamics Observatory (SDO) in February 2010, the resolution of solar images has improved significantly. Hence, researchers should not use the exact same coronal loop detection algorithms designed for low resolution images. Also, the scalability and performance of older methods should be scrutinized when applied on images produced by the SDO instrument.

Even though SDO produces spectacular images and SOHO/EIT is not the main source of data now, the past 15 years of SOHO/EIT images still offer a lot of information to understand the dynamics of the Sun. With the techniques proposed in this dissertation, the entire SOHO/EIT data set could be analyzed and the detected coronal loops could be offered to researchers.

Since marking/labeling the images is time consuming, and sometimes inconsistent (due to multiple experts), semi-supervised classification techniques seem to be particularly promising to strengthen the modeling of the minority class (*Loop*) instances. It is also possible to add a feedback tool to refine the training data set with the help of feedback given to misclassified regions in the testing phase.

The effect of point-to-chord distance measures and ellipticity measures could be used in shape based image retrieval. In particular, if open curves are searched, these features could improve the results.

Different optimization techniques could be analyzed for contour grouping algorithm methods. The application areas of our contour grouping method could be

expanded to medical images (e.g., blood vessel detection in coronary images). Thus our curve tracing method could be applied in different domains as well.

REFERENCES

Almeida C.W.D [et al.] Image Retrieval using the curvature scale space (CSS) technique and the self organizing map (SOM) model under rotation [Conference]. : Digital Information Management, ICDIM, 2007.

Alquier L. and Montesinos. P. Perceptual grouping and active contour functions for the extraction of roads in satellite pictures [Conference]. : Image and Signal Processing for Remote Sensing III, 1996. - pp. 153-163.

Aschwanden M.J. . 2D Feature Recognition and 3D Reconstruction in Solar EUV Images [Journal]. : Solar Physics, 2005. - Vol. 228. - pp. 339–358.

Aschwanden M.J. and J.K. Lee, G.A., Allen, M., Smith, B. Inhester Comparison of Five Numerical Codes for Automated Tracing of Coronal Loops [Journal]. : Solar Physics, 2007. - 2 : Vol. 248. - pp. 359-377.

Bacher U. & Mayer, H Automatic road extraction from IRS satellite images in agricultural and desert areas. [Journal]. : In International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences, 2004. - B3 : Vol. 35. - pp. 1055-1060.

Bernasconi P.N., Rust, D.M., Hakim, D., Advanced Automated Solar Filament Detection and Characterization Code: Description, Performance, and Results [Journal]. : Solar Physics, 2005. - 99 : Vol. 228.

Bookstein F. Fitting conic sections to scattered data [Journal]. : Computer Vision Graphics and Image Processing, 1979. - Vol. 9. - pp. 56-71.

C.K. Lee R.M. Haralick, K. Deguchi Estimation of curvature from sampled noisy data [Conference]. : International Conference on Computer Vision and Pattern Recognition, 1993. - pp. 536-541.

Cheng M. and C.Cheng Y. Liu, A robust algorithm for image principal curve detection [Conference]. : ICPR, 2004.

Cohen W. W. Fast effective rule induction [Journal]. : ICML, 1995. - pp. 115-123.

Colak T. and Qahwaji R. Automatic Detection and Verification of Solar Features [Journal]. : International Journal of Imaging Systems Technology, 2005. - 4 : Vol. 15. - pp. 199-210.

Colak T. and R.Qahwaji AUTOMATED PREDICTION OF SOLAR FLARES: Integrating Image Processing and Machine Learning for the Creation of a Hybrid Computer Platform that Provides Real-Time Prediction of Solar Flares [Book]. : LAP LAMBERT ACADEMIC PUBLISHING, 2010.

D. Donoho X. Huo Applications of Beamlets to Detection and Extraction of Lines, Curves and Objects in Very Noisy Images [Conference]. - 2001 : In Proc. NSIP.

D. E. Rumelhart G. E. Hinton, R. J. Williams Learning internal representations by error propagation [Book Section] //, Parallel distributed processing: explorations in the microstructure of cognition. : MIT Press, 1986. - Vol. vol. 1.

Dalal N and Triggs B Histograms of Oriented Gradients for Human Detection [Conference]. : IEEE Computer Society Conference on Computer Vision and Pattern Recognitio, 2005. - pp. 886-893.

Domingos M. Pazzani On the optimality of the simple Bayesian classifier under zero-one loss. [Journal]. : Machine Learning, 1997. - Vol. 29. - pp. 103-137.

Dubes R.J. and A.K.Jain Random Field Models in Image Analysis [Journal]. : Journal of Applied Statistics, 1989. - 2 : Vol. 16.

Duda R. O. and and Hart P. E. Use of the Hough Transformation to Detect Lines and Curves in Pictures [Journal]. : Comm. ACM, 1972. - Vol. 15. - pp. 11-15.

Duda R. O., Hart P. E. and . and D. G. Stork. Pattern Classiffication [Book]. : John Wiley and Sons, 2001. - 2nd.

Durak N. and Nasraoui O. Extracting Salient Contours from Cluttered Solar Images [Conference]. : IEEE International Conference on Image Processing, 2011.

Durak N. and Nasraoui O. Feature Exploration for Mining Coronal Loops from Solar Images [Conference]. : IEEE International Conference on Tools and Artificial Intelligence, 2008.

Durak N. and Nasraoui O. Principal Contour Extraction and Contour Classification to Detect Coronal Loops from the Solar Images [Conference]. : ICPR, 2010.

Durak N. and O. Nasraoui J. Gomez, F. Gonzalez, H. Elgazzar, S. Sellah Mining Coronal Loops in Solar Images from the SOHO collection [Conference]. : NAFIPS, 2007. - pp. 514-519.

Durak N. and O. Nasraoui J. Schmelz Automated Coronal Loop Detection based on Contour Extraction and Contour Classification from the SOHO/EIT Images [Journal]. : Solar Physics, 2010. - pp. 383-402.

Durak N. and O. Nasraoui J. Schmelz Coronal Loop Detection from Solar Images [Journal]. : Pattern Recognition, 2009. - 11 : Vol. 42. - pp. 2481-2491.

EIT: Extreme ultraviolet Imaging Telescope on board Solar and Heliospheric Observatory (SOHO): [Online]. - <http://umbra.nascom.nasa.gov/eit/>.

Elder J. H., Krupnik A. and Johnston L. A. Contour grouping with prior models [Journal]. : IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003. - 6 : Vol. 25. - pp. 661 - 674.

Felzenszwalb P. and McAllester D. A Min-cover Approach for Finding Salient Curves [Conference]. : IEEE Workshop on POCV, 2006.

Fischler M.A. and Bolles R.C. Perceptual Organization and Curve Partitioning [Journal]. : IEEE PAMI, 1986. - 1 : Vol. 8.

Fischler M.A. and Wolf H.C. Locating Perceptually Salient Points on Planar Curves [Journal]. : IEEE Trans on PAMI, 1994. - 2 : Vol. 16. - pp. 113-129.

Fitzgibbon A., Pilu M. and Fisher and R.B. Direct Least Squares Fitting of Ellipses [Journal]. : IEEE Trans. of PAMI, 1999. - 5 : Vol. 21.

Freeman H. On the encoding of arbitrary geometric configurations [Journal]. : IEEE Trans. Electron. Comput., 1961. - Vol. 10. - pp. 260–268.

Freund Y. and Schapire R.E. A decision-theoretic generalization of on-line learning and an application to boosting [Journal]. : Journal of Computer and System Sciences, 1997. - 1 : Vol. 55. - pp. 119-139.

Fu A.M.N., Jan H. and Huang K. A Curve Bend Function Method to Characterize Contour Shapes [Journal]. : Pattern Recognition, 1997. - 10 : Vol. 30. - pp. 1661-1671.

Fu G. Shih, F. Y. Wang, H. Automatic Detection of Prominence Eruption Using Consecutive Solar Images [Journal]. : IEEE Transactions on Circuits and Systems for Video Technology, 2007. - 1 : Vol. 17. - pp. 79-85.

Gao Q. G. and Wong A.K.C Curve Detection Based on Perceptual Organization [Journal]. : Pattern Recognition, 1993. - 7 : Vol. 26. - pp. 1039-1046.

Goldberg J. H and M. Elder and R. Ecological statistics of Gestalt laws for the perceptual organization of contours [Journal]. : Journal of Vision, vol. 2, pp., 2002. - Vol. 2. - pp. 324-353.

Gonzalez R.C., Woods, R. Digital Image Processing [Book]. - New York City : Prentice-Hall, 2007.

Guy G. and Medioni G. Inferring global perceptual contours from local features [Journal]. : International Journal of Computer Vision, 1996. - Vol. 20. - pp. 113-133.

H. Freeman L. S. Davis A Corner-Finding Algorithm for Chain-Coded Curves [Journal]. : IEEE Trans. Computers, 1977. - 3 : Vol. 26. - pp. 297-303.

H. Vafaie and I.F. Imam Feature Selection Methods, Genetic Algorithms vs. Greedy-like Search [Conference]. : IFICS, 1994.

Han J.H. and Poston T. Chord-to-point distance accumulation and planar curvature: a new approach to discrete curvature [Journal]. : Pattern Recognition Letters, 2001. - Vol. 22. - pp. 1133-1144.

Handy B. N., L.W. Acton, C.C. Kankelborg et. al. The transition region and coronal explorer [Journal]. : Solar Physics, 1999. - Vol. 187. - p. 229.

Handy S. L. Freeland and B. N. Data analysis with the solarsoft system [Journal]. : Solar Physics, 1998. - 2 : Vol. 182. - pp. 497-500.

Haralick R. M. A measure for circularity of digital figures. [Journal]. : IEEE Transactions on Systems, Man and Cybernetics, 1974. - 4 : Vol. 4. - pp. 334-336.

Hill M. and V Castelli C.H. Li, Y.C. Chang, L. Bergman, J.R. Smith, B. Thompson Solarspire: querying temporal solar imagery by content [Conference]. : ICIP, 2001.

Ichoku C. and B. Deffontaines J. Chorowicz Segmentation of digital plane curves: A dynamic focusing approach [Journal]. : Pattern Recognition Letters, 1996. - 7 : Vol. 17. - pp. 741-750.

Illingworth J. and Kittler J. A survey of the Hough Transform [Journal]. : Computer Vision, Graphics, and Image Processing, 1988. - pp. 87-116.

Inhester B., Feng L. and Wiegelmann T. Segmentation of Loops from Coronal EUV Images [Journal]. : Solar Physics, 2007. - Vol. 248. - pp. 379–393.

J.C. Pineda P. Horaud An Improved Method for High-Curvature Detection with Applications to Automatic Inspection [Journal]. : SP, 1983. - Vol. 5. - pp. 117-124.

Jain A. and Zongker D. Feature selection: Evaluation,application, and small sample performance [Journal]. : IEEE Transactions on PAMI, 1997. - 2 : Vol. 19. - pp. 153-158.

Ji Q. and Haralick R.M. A Statistically Efficient Method for Ellipse Detection [Conference]. : ICIP, 1999.

Kawaguchi T. and Nagata R Ellipse Detection using grouping of edgels into Line Support Regions [Conference]. : ICIP, 1998. - pp. 70-74.

Koffka K. Principles of Gestalt Psychology [Book]. - New York : Harcourt Brace, 1935.

Kohavi R. and H. John G. Wrappers for feature subset selection [Journal]. : Artificial Intelligence, 1997. - 1-2 : Vol. 97. - pp. 273-324.

Koller D., and Sahami M. Toward optimal feature selection [Conference]. : ICML, 1996. - pp. 284-292.

Kopyrnicky M. and M. Ahmed M. Kamel, Contour description through set operations on dynamic reference shapes [Conference]. : ICIAR, 2004.

Lee D.R. and Sallee T. A method of measuring shape [Journal]. : Geographical Review, 1970. - 4 : Vol. 60. - pp. 555-563.

Lee J.K, Newman T.S. and Gary G.A. Oriented connectivity-based method for segmenting solar loops [Journal]. : Pattern Recognition, 2006. - Vol. 39.

Lenz D. D., DeLuca, E. E., Golub, L., Rosner, R., & Bookbinder, J. A. Temperature and Emission-Measure Profiles along Long-lived Solar Coronal Loops Observed with the Transition Region and Coronal Explorer [Journal]. : ApJ, 1999. - Vol. 517. - p. 155.

Li S. Markov Random Field Image Analysis [Book]. : Springer, 2009.

Lowe D.G. Three dimensional object recognition from single two dimensional images [Journal]. : Artificial Intelligence, 1987. - Vol. 31. - pp. 355-395.

Lowe David G. Perceptual Organization and Visual Recognition [Book]. : Kluwer Academic Publishers, 1985.

M. Turmon J. M. Pap, S. Mukhtar Statistical Pattern Recognition for Labeling Solar Active Regions: Application to SOHO/MDI Imagery [Journal]. : The Astrophysical Journal, 2002. - Vol. 568. - pp. 396-407.

Mallat S. G. A theory for multiresolution signal decomposition: the wavelet representation [Journal]. : IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989. - Vol. 11. - pp. 674- 693.

Martinez Perez M.E. [et al.] Segmentation of Retinal Blood Vessels Based on the Second Directional Derivative and Region Growing [Conference]. : ICIP, 2001.

Matwin S. and M. Kubat Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. [Conference] // ICML-97. - 1997. - pp. 179–186.

McLaughlin R.A. Randomized Hough Transform: Improved ellipse detection with comparison [Journal]. : Pattern Recognition Letters, 1998.

Mokhtarian F. and Abbasi S., and Kittler, J. Robust and Efficient shape indexing through curvature scale space [Conference]. : British machine learning conference, 1996. - pp. 53-62.

Murino V., Regazzoni C.S. and Foresti G.L. Grouping as a Searching Process for Minimum-Energy Configurations of Labelled Random Fields [Journal]. : Computer Vision and Image Understanding, 1996. - 1 : Vol. 64. - pp. 157-174.

Murphy T.M. and M. Matlin L. H. Frinkel Curvature Covariation as a Factor in Perceptual Saliency [Conference]. : IEEE conference on Neural Engineering, 2003. - pp. 16-20.

Nguyen T.P. and Debled-Rennesson I. Circularity measuring in linear time [Conference]. : ICPR, 2010.

Paramanand C. and Rajagopalan, A. N. An Efficient Representation of Digital Curves with Line Segments and Elliptical Arcs [Conference]. : Proceedings of the IET International Conference on Visual Information Engineering, 2006. - pp. 470-475.

Posch S. and Schluter D. Perceptual Grouping using Markov Random Fields and Cue Integration of Contour and Region Information [Journal]. - 2001.

Proffitt D. The measurement of circularity and ellipticity on a digital grid [Journal]. : Pattern Recognition, 1982. - 5 : Vol. 15. - pp. 383-387.

Q. Gang Gao A.K.C Wong Curve Detection Based on Perceptual Organization [Journal]. : Pattern Recognition, 1993. - 7 : Vol. 26. - pp. 1039-1046.

Quinlan J.R. C4.5: Programs for Machine Learning [Book]. : Morgan Kaufmann Publishers Inc., 1993.

R. M. Haralick L. G. Shapiro Computer and Robot Vision [Book]. - 1992. - Vol. Volume 1.

Raghupathy K. and Parks T. W. Improved Curve Tracing in Images [Conference]. : IEEE ICASSP, 2004. - pp. 581-584.

Ren X, Fowlkes C. C. and Malik and J. Scale-invariant contour completion using Conditional Random Fields [Journal]. : ICCV, 2005.

Rosin P.L. Analysing error of fit functions for ellipses [Journal]. : Pattern Recogn. Lett., 1996. - 14 : Vol. 17. - pp. 1461-1470.

Rosin P.L. Analysing error of fit functions for ellipses [Conference]. : BMVC, 1996.

Rosin P.L. Measuring shape: ellipticity, rectangularity, and triangularity [Journal]. : Machine Vision and Applications, 2003. - Vol. 14. - pp. 172-184.

Roussillon T. and I. Sivignon L. Tougne Measure of circularity for parts of digital boundaries and its fast computation [Journal]. : Pattern Recognition, 2010. - 1 : Vol. 43. - pp. 37-46.

Roussillon T. and I. Sivignon L. Tougne Measure of circularity for parts of digital boundaries and its fast computation [Journal]. : Pattern Recognition, 2010. - 1 : Vol. 43. - pp. 37-46.

S. Biskri J.-P. Antoine, B. Inhester and F. Mekideche Extraction of Solar Coronal Magnetic Loops with the Directional 2D Morlet Wavelet Transform [Journal]. : Solar Physics, 2010. - 2 : Vol. 262. - pp. 373-385.

Sampson P.D. Fitting conic sections to very scattered data: An iterative refinement of the Bookstein Algorithm. [Journal]. : CVGIP, 1982.

Sargin M.E. and A. Altinok K.Rose, B.S. Manjunath. Tracing Curvilinear Structures in Live Cell images. [Conference]. : ICIP, 2007.

Schlüter F. Ackermann and A. Maßmann and S. Posch and G. Sagerer and D. Perceptual Grouping of Contour Segments Using Markov Random Fields [Journal]. : IEEE Trans. PAMI, 1997. - pp. 11-17.

Schmelz J.T. and Cirtain J.W. et al. Coronal loops: Isothermal OR multithermal? [Journal]. : Advances in Space Research, 2003. - 6 : Vol. 32. - pp. 1109-1115.

Schmelz J.T. and Roames J.K. Nasraoui K. The coronal loop controversy: TRACE analysis [Journal]. : Advances in Space Research, 2007. - Vol. 39. - pp. 1497-1501.

Schmid C. and F. Jurie Scale-invariant shape features for recognition of object categories [Conference]. : In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2004. - pp. 90-96.

Semler L., & Dettori, L. A Comparison of Wavelet-Based and Ridgelet-Based Texture Classification of Tissues in Computed Tomography [Conference]. : Proceedings of International Conference on Computer Vision Theory and Applications, 2006.

Shapire R.E. and Singer Y. Improve boosting algorithms using confidence-rated predictions [Journal]. : Machine Learning, 1999. - 3 : Vol. 37. - pp. 297-336.

Soh Y. [et al.] A new Method for Ellipse Fitting in Satellite Images [Conference]. : ICICTA, 2009.

Solar Heliospheric Observatory [Online]. - <http://sohowww.nascom.nasa.gov/>.

Starck J.L., Donoho, D.L., & Candes, E.J. Astronomical Image Representation by the Curvelet Transform [Journal]. : Astronomy & Astrophysics, 1999. - Vol. 398. - pp. 785-800.

Steger C. An unbiased detector of curvilinear structures [Journal]. : IEEE Trans. On Pattern Analysis and Machine Intelligence, 1998. - pp. 113-125.

Steger C., Mayer H. and Radig B. The Role of Grouping for Road Extraction [Journal]. : Automatic Extraction of Man-Made Objects from Aerial and Space Images(II), 1999. - 256 : Vol. 245. - pp. 1931-1952.

Sternberg S. Biomedical Image Processing [Journal]. : IEEE Computer, 1983.

Stojmenovic M. and Nayak A. Direct Ellipse Fitting and Measuring Based on Shape Boundaries [Conference]. : Pacific-Rim Symposium on Image and Video Technology, 2007. - pp. 221-235.

Stojmenovic M. and Nayak A. Shape based circularity measures based of planar points sets [Conference]. : IEEE ICSSC, 2007.

Sun T.H. [et al.] Boundary-based corner detection using K-cosine [Conference]. : IEEE Systems, Man and Cybernetics, 2007. - pp. 1106 - 1111 .

Svalbe I. D. Natural Representations for Straight Lines and the Hough Transform on Discrete Arrays [Journal]. : IEEE Transactions on Pattern Analysis and Machine Intelligence, 1989. - 9 : Vol. 11. - pp. 941 – 950.

Swain M.J. and Ballard D.H. Color Indexing [Journal]. : International Journal of Computer Vision, 1991. - 1 : Vol. 7. - pp. 11-32.

T. J. Cham and R. Cipolla Automated B-spline curve representation incorporating MDL and error-minimizing control point insertion strategies [Journal]. : IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999. - 1 : Vol. 21. - pp. 49-53.

Tan P.N., Steinbach M. and Kumar V. Introduction to Data Mining [Book]. : Addison Wesley, 2006.

TRACE Data Center [Online]. - http://trace.lmsal.com/trace_cat.html.

Transition Region and Coronal Explorer [Online]. - <http://trace.lmsal.com/>.

Tsuji S. and Matsumoto F Detection of Ellipses by a Modified Hough Transformation [Journal]. : IEEE Transactions on Computers, 1978.

Tu Z. and Zhu S.C Parsing images into regions, curves, and curve groups [Journal]. : Int'l Journal of Computer Vision, 2006. - 2 : Vol. 69. - pp. 223-249.

Ullman S. and Sha'ashua A. Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network [Conference]. : ICCV, 1988.

Wang J. Yang and R. S. Classified road detection from satellite images based on perceptual organization. [Journal]. : Int. J. Remote Sens., 2007. - 20 : Vol. 28. - pp. 4653-4669.

Wang S. [et al.] Salient Closed Boundary Extraction with Ratio Contour [Journal]. : IEEE Trans on Pattern Analysis and Machine Intelligence, 2005. - 4 : Vol. 27. - pp. 546-61.

Wang S., Yao J. and Summers R.M Matching Colonic Polyps from Prone and Supine CT Colonography Scans based on Statistical Curvature Information [Conference]. : ICPR, 2008.

Won C. S., Park D. K. and Park S. J. Efficient Use of MPEG-7 Edge Histogram Descriptor [Journal]. : ETRI Journal, 2002. - 1 : Vol. 24. - pp. 23-30.

Wu W. and Wang M. Elliptical object detection by using its geometric properties [Journal]. : Pattern Recognition, 1993. - 10 : Vol. 28.

Y. Li E. Hancock Face Recognition using Shading-Based Curvature Attributes [Conference]. : ICPR, 2004. - pp. 538-541.

Zhang B., Fadili M.J. and and Starck J.-L. Wavelets, Ridgelets and Curvelets for Poisson Noise Removal [Journal]. : IEEE Transactions on Image Processing, 2008. - 7 : Vol. 17. - pp. 1093--1108.

Zhang X. [et al.] Extraction of Line Segments and Circular Arcs From Freehand Strokes Based on Segmental Homogeneity Features [Journal]. : IEEE Trans. On Systems, Man, and Cybernetics, 2006. - 2 : Vol. 36.

Zharkova V. and S. Ipson A. Benkhalil, and S. Zharkov Feature recognition in solar images [Journal]. : Artificial Intelligence Review, 2005. - 3 : Vol. 23. - pp. 359-377.

Zhu S. C. "Embedding Gestalt Laws in Markov Random Fields -- A theory for shape modeling and perceptual organization" [Journal]. : IEEE Trans. on PAMI, 1999. - 11 : Vol. 21. - pp. 1170-1187.

Zhu S. C. Embedding Gestalt Laws in Markov Random Fields -- A theory for shape modeling and perceptual organization [Journal]. : IEEE Trans. on PAMI, 1999. - 11 : Vol. 21. - pp. 1170-1187.

CURRICULUM VITAE

e-mail: nurcan.durak@gmail.com

webpage: <http://webmining.spd.louisville.edu/~ndurak>

RESEARCH INTERESTS

- Computer Vision, Pattern Recognition, Machine Learning, Feature Extraction, Text Analysis, Data Mining, Web Mining, Multimedia Retrieval, Image Retrieval, Video Retrieval

EDUCATION

- **Phd.** in Computer Engineering and Computer Science, University of Louisville, Louisville, KY, USA. (GPA 4.0, 2011)

Dissertation: Coronal Loop Mining from Solar Images and Extraction of Salient Contour Groups from Cluttered Images

- **M.Sc** in Computer Engineering, Department of Computer Engineering, METU, Turkey. 2004.

Thesis: Semantic Video Modeling and Retrieval with Visual, Auditory, Textual Sources.

- **B.Sc.** in Computer Engineering, Department of Computer Engineering, Baskent University, Turkey. 2001.

WORK EXPERIENCE

- August 2006 – Now: **Research Assistant** in Mining Coronal Loops in Solar Images from the SOHO collection, as part of a NASA supported Project at the Knowledge Discovery Lab in University of Louisville, Louisville, Kentucky.

Responsibilities:

- Designing and developing a Java based image retrieval tool that retrieves images with coronal loops from online solar image databases,
- Image processing to clean solar images using Matlab, Java and IDL,
- Feature extraction and selection to recognize coronal loops at MATLAB,
- Experimenting with various classification techniques on the problem using Java within the WEKA data mining package,
- Extracting salient contours from cluttered solar images using MATLAB,
- Mentoring an undergraduate student under NSF REU program, the student won an ACM CRA (Computing Research Association) nomination for

outstanding research and a best research poster at Speed School Undergraduate Research Competition in E-Expo with our study.

- Writing grant proposals, journal and conference publications.
 - Creating and maintaining the project web site, and maintaining lab webpage.
 - **Research Areas of the Project:** Image Processing, Machine Learning, Data Mining, Pattern Recognition, Feature Extraction
- November 2005- August 2006: **Teaching Assistant**, Department of Computer Engineering, METU, Turkey.

Assisted Courses: Compiler, Database Management System, and Software Engineering.

- September 2004 - November 2005: **Researcher** in Video Data Modeling and Querying on Surveillance Videos as part of a DPT Project in the Multimedia Lab in Department of Computer Engineering, METU, Turkey.

Responsibilities:

- Developing a Java based object and event retrieval tool from surveillance video archives,
 - Designing semantic video models for spatio-temporal objects and events,
 - Developing a query interface supporting spatio-temporal interactions of semantic entities,
 - Integrating system with automatically extracted moving objects and events which was developed at Visual C++ platform.
 - **Research Areas of the Project:** Video data modeling, spatio-temporal queries, pattern recognition, object extraction, event classification
- September 2001 - September 2004: **Teaching Assistant** in the Department of Computer Engineering, Baskent University, Ankara, Turkey.

Assisted Courses: Programming Language, Software Engineering, Computer Architecture, Object Oriented Programming, Operating System, Introduction to Programming with C.

-Mentoring two undergraduate students toward their graduation thesis. (Automatic Video Scene Detection for one student; Traveling Salesman problem for the other)

- 2000 Summer Internship: **Software Engineer** at Dolphinsoft GMBH, Bielefeld, Germany.
Worked on a project related to search engines evaluation using script languages and C++ programming.
- 1999 Summer Internship: **Software Engineer** at the Statistic Institute of Government, Ankara, Turkey.

Oracle database programming, developing software using Borland Delphi.

HONORS and AWARDS

- 2nd place winner at Grand Finals of all ACM Student Research Competitions, 2011 (Invited to ACM Award Banquet in San Jose with Dr. Olfa Nasraoui)
- Google Anita Borg Scholarship Finalist, 2011 (Invited to GooglePlex in Mountain View, CA)
- 3rd place at ACM Student Research Competition in Grace Hopper Conference 2010 sponsored by Microsoft Research.
- IEEE Outstanding Computer Engineering and Computer Science Student Award, University of Louisville, 2010
- Who's Who Among Students in American Universities and Colleges - 2010 Recipient
- NASA travel awards to attend Solar Imaging Workshops in 2008 and 2010.
- 2nd place winner in the photography category in the Art show, University of Louisville, 2010.
- IEEE Outstanding Computer Engineering and Computer Science Student Award, University of Louisville, 2009
- 3rd graduate class of 2001, Baskent University.
- Full scholarship for undergraduate studies at Baskent University, 1997 – 2001.

SUMMER SCHOOLS

- Machine Learning Summer School, Chicago, 2009.
- European Summer School in Information Retrieval, Dublin, Ireland, 2005.

TECHNICAL SKILLS

- Programming: JAVA, C, C++, Matlab, Phyton, Scheme, Haskell, lex-yacc, gcc, Prolog, Assembly Language, Delphi, Pascal
- Data Mining Tools: Weka, Cluto, Minitab, ImageJ
- Crawling and Indexing: Lucene, Nutch
- Object Oriented Design: UML, Microsoft Visio
- Solar Imaging Tools: IDL, SolarSoft
- Web programming: Perl, Php, Asp, CGI, HTML, MySQL, Photoshop, FrontPage

JOURNAL PUBLICATIONS

1. Nurcan Durak, Olfa Nasraoui: "Extracting Salient Contours from Cluttered Images with Perceptual Rules", (submitted to IEEE transactions on Image Processing)
2. Nurcan Durak, Olfa Nasraoui, Joan Schmelz: "Automated Coronal Loop Detection based on Contour Extraction and Contour Classification from the SOHO/EIT Images", Solar Physics, 2010, pp. 383-402.
3. Nurcan Durak, Olfa Nasraoui, Joan Schmelz: "Coronal Loop Detection from Solar Images", Pattern Recognition, 2009, Vol.42, No:11, (November 2009), pp.2481-2491.

CONFERENCE PROCEEDINGS (Peer Reviewed)

1. Nurcan Durak, Olfa Nasraoui: "Extracting Salient Contours from Cluttered Solar Images", *IEEE International Conference on Image Processing 2011 (ICIP 2011)*.
2. Nurcan Durak, Grace Hopper Conference, "Coronal Loop Detection from Solar Images", Atlanta, 2010. (Won the 3rd place in ACM- Student Research Competition sponsored by Microsoft Research, NSF scholarship)
3. Nurcan Durak, Olfa Nasraoui: "Principal Contour Extraction and Contour Classification to Detect Coronal Loops from the Solar Images", *IEEE International Conference on Pattern Recognition 2010. (ICPR 2010)*
4. Nurcan, Durak, Olfa Nasraoui, "Contour-based Approach to detect coronal loops from solar images", Conference on Intelligent Data Understanding (CIDU), NASA Ames Research Center, 2009
5. Nurcan Durak, Olfa Nasraoui: "Feature Exploration for Mining Coronal Loops from Solar Images". *IEEE International Conference on Tools and Artificial Intelligence 2008 (ICTAI 2008)*.
6. Nurcan Durak, Olfa Nasraoui: "Blog Profile Extraction using Text Categorization". *ACM Hypertext and Multimedia 2008*. (ACM – Student Research Competition, placed in the final five)
7. Carlos Rojas, Olfa Nasraoui, Nurcan Durak, Leyla Zhuhadar, Sofiane Sellah, Zhiyong Zhang, Basheer Hawwash: "Knowledge Discovery in Data with Selected Java Open Source Software", *CIML: Machine Learning Virtual Organizations, Computational Intelligence and Machine Learning*, October 24, 2008.
8. Nurcan Durak, Olfa Nasraoui, Jonatan Gomez, Fabio Gonzalez, Heba Elgazzar, Sofiane Sellah: "Mining Coronal Loops in Solar Images from the SOHO collection". *NAFIPS 2007: 514-519*. Published in *IEEE-Press*.
9. Nurcan Durak, Adnan Yazici, Roy George: "Online Surveillance Video Archive System". *13th International MultiMedia Modeling Conference*. Singapore, 2007. Published in *Lecture Notes in Computer Science*, Springer-Verlag.
10. Kemal Arda, Nurcan Durak, Nazan Ciledag, Nazife Baykal, "Assessing patients with swallowing disorder via tracking barium swallows", *National Radiology Conference*, Turkey, 2007
11. Nurcan Durak, Adnan Yazici: "Data Modeling and Querying of Surveillance Videos", *SIU 2006*, Published in *IEEE-Press (National Conference)*.
12. Nurcan Durak, Adnan Yazici: "Multimodal Video Database Modeling, Querying and Browsing". *ISCIS 2005: 802-812*, Published in *Lecture Notes in Computer Science*, Springer-Verlag.
13. Nurcan Durak, Adnan Yazici: "Video Database Modeling", *Informatics Encyclopedia*, 902-907, (Book chapter, in Turkish).

WORKSHOPS AND PRESENTATIONS

1. Solar Image Processing Workshop V, Switzerland, 2010. (Oral Presentation, NASA Travel Award)
2. Garrett Ridge, Nurcan Durak, Olfa Nasraoui: "Prominent Curve Detection from Cluttered Images", Posters at Capitol, Frankfurt, KY, 2010.

3. Garrett, Ridge, Nurcan Durak, Olfa Nasraoui, Automated Principal Curve Detection in Images of Solar Coronal Loops, Engineering-Expo 2009, 1st Place Winner: Undergraduate Research Competition at E-Expo 2009, University of Louisville.
4. Poster Presentation, Graduate Research Symposium 2009 at University of Louisville (Coronal Loop mining).
5. Poster Presentation, Engineering-Expo 2009 at University of Louisville (Blog text categorization).
6. Applied Imagery Pattern Recognition Workshop (AIPR), Washington DC, 2009 Oral Presentation).
7. Solar Image Processing Workshop IV, Baltimore, MD, 2008. (Oral Presentation, NASA Travel Award)
8. Poster Presentation, Engineering-Expo 2008 at University of Louisville (Coronal Loop mining).
9. NASA Applied Information Systems Research Program Investigator Workshop, (2008, 2009).
10. Women in Computing Research (CRA-W), Graduate Cohort Workshop, (2007, 2008, 2009),(Poster Presentation)
11. Richard Tapia Celebration of Diversity in Computing, Orlando, 2007,(Poster Presentation)
12. Demonstration and Oral Presentation of Object and Event Retrieval tool for Surveillance video archives at several DPT workshops around Turkey, 2004-2005.

GRANT WRITING EXPERIENCE

- Data Mining of Remote Sensing Imagery based on Non-Rigid Registration and Dynamic Clustering for Change Detection, NASA INSPIRE program, 2010
- Solar Event Detection and Retrieval System using Spatio-Temporal Dimensions, NASA AISPR program 2009 (got excellent reviews)

ORGANIZATIONAL SERVICE

- Senator, Student Government Association, 2009-2010
- Officer at Appropriation Committee, 2009-2010
- Information Chair and Executive Board member - Graduate Student Council of University of Louisville, 2008-2010
- Organizer and Speaker of a workshop entitled "Integration of International Students to US Education System", 2009

PROFESSIONAL SERVICE

- Computer Science and Engineering Judge at Regional Science Fair and Manual Dupont High School, Louisville, 2010
- Student Volunteer – ACM SIG Computer Human Interaction 2009
- Reviewer:
 - WI (Web Intelligence) 2008,
 - WSDM (Web Search and Data Mining) 2008,

- ICDM (International Conference on Data Mining) 2008,
- RecSys (Recommender Systems) 2008,2011
- SIGKDD (Knowledge Discovery and Data Mining) 2008,
- CEC (Congress on Evolutionary Computation) 2009,
- Statistical Analysis and Data Mining, Book Chapter, 2009.
- KDD (Knowledge Discovery and Data Mining) 2010, 2011

RELEVANT COURSES

- **PhD Coursework:** Web Mining, Human Computer Interaction, Multimedia and Hypertext Analysis, Adv. Tech. Internet Search, Content based Multimedia Retrieval, Data Mining, Advanced Database Applications, Experimental Design, Medical Imaging Techniques
- **Master Coursework:** Computational Linguistics, Pattern Recognition, Image Processing, Natural Language Processing, Distributed Computing Systems, Advanced Database, Computer Networks

Some Course Projects during Graduate Studies

- Medical Imaging Techniques: Tracking swallowing disorders from barium swallowing videos (It was presented at a medical conference)
- Hypertext and Multimedia: Designing a similarity based image retrieval tool using MPEG features.
- Web Mining: Classifying web pages into two groups: the ones with advertisements and the ones without advertisements.
- Advance Internet Searching Techniques: Crawling and indexing Images from Flickr using Nutch and Lucene and developing a content based image retrieval tool on Lucene
- Human Computer Interaction: Developing a zoomable browser for the categories of Yahoo!; Performed a User study on the developed zoomable tool; User study on the hidden functions of Internet Explorer.
- Experimental Design: Experimenting interactions among MPEG features in a similarity based image retrieval tool using multi-factorial design at Minitab.
- Computational Linguistics: NLP techniques for question answering; Analyzing Turkish language with respect to syntax and semantics using parser programming languages (ccg) and Prolog.

LANGUAGES

- Native Language: Turkish
- Fluent in English
- Beginner in German

AFFILIATIONS

- IEEE Student Member
- ACM Student Member
- Society of Women Engineers

HOBBIES

- Photography (Mostly abstract, viewing and capturing objects from different point of views).
- Blogging in Turkish (photography, movies, books, literature, traveling, music)
- Volunteering for Louisville Refugees (helping children in their homework, teaching English to adults)
- Camping, Hiking, Biking, Orienteering, Traveling, Swimming, Reading, Cinema, Playing flute.