

DESIGNING OF OBJECTS USING SMOOTH CUBIC SPLINES

BY

AIMAN RASHEED

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

INFORMATION & COMPUTER SCIENCE

DECEMBER 2005

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **Aiman Rasheed** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**

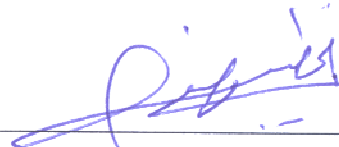
Thesis Committee



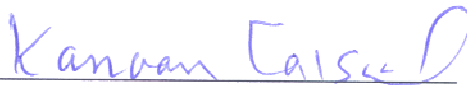
Dr. Muhammad Sarfraz (Advisor)



Dr. Muhammed Al-Mulhem (Member)



Dr. Wasfi G. Al-Khatib (Member)



Dr. Kanaan Faisal
(Department Chairman)



Dr. Muhammad Abdulaziz Al-Ohali
(Dean of Graduate Studies)

٢٥٧٧/٦/٢٩

Date 25-7-2006



Dedicated to

My beloved
family

ACKNOWLEDGMENTS

In the name of Allah (Sub 'hanahu-wa-Ta'ala), the Most Gracious, the Most Merciful

Alhumdulillah, All praise be to Allah (SWT), the self-sufficient Master, the Lord of the Worlds, Master of the Day of Judgment. The Creator of All and to Whom we all shall return. And there is none co-equal or comparable unto Him. Thee do we worship and Thee aid we seek.

Peace and blessings of Allah (SWT) be upon the last Prophet Muhammad (Sallallaahu'alaihi wa Sallam), his family and companions.

Acknowledgement is due to King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, for providing support and facilities for this research work.

With deep sense of gratitude, my appreciation goes to the thesis advisor, Dr. Muhammad Sarfraz, for his indomitable support and advice. Without his sheer guidance and suggestions, this work would have become a daunting task. His extensive knowledge and experience made it possible to shape the thesis. My unrestrained recognition also goes to the committee members, Dr. Muhammed Al-Mulhem and Dr. Wasfi G.Al-Khatib for their interest, invaluable cooperation and support.

I would like to render my profound acknowledgements to Dr. M. Balah for his countless hours that he spent on discussing and solving problems that I faced. It was his keen interest in this area that made this unsurmountable task viable.

Acknowledgements are due to my friends and colleagues who made my stay at the university a cherished and an unforgettable era. In particular I would like to express my gratefulness to Syed Zeeshan Muzaffar for always showing interest in the field of Computer Graphics. Saad A⁺, for his guidance in software for symbolic evaluation of mathematical formulations. Syed Akhtar Ghazi, Syed Adnan Shahab, Mudassir Masood, Imran Naseem, Imran Azam, Khawar Khan, Sarfraz Abbasi, Kashif Khan, Saad Azhar and Moin BHAI for their ever helpful natures. Finally, and most importantly I would like to thank Syed Adnan Yusuf who was always by my side, encouraging me for this achievement. His brainstorming habit made it possible for me to furnish new ideologies.

Finally, I am most profoundly thankful to my family members for being the source of inspiration and motivation. They have always helped me spur my spirits and encourage me throughout my life with their inspiring and hortatory words. Their celebrations of my minutest successes elate me to the pinnacle of confidence. Their prayers, patience, guidance and support lead to the successful accomplishment of this thesis.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	ii
LIST OF FIGURES	vii
LIST OF TABLES	xx
THESIS ABSTRACT	xxiv
CHAPTER 1 INTRODUCTION	1
1.1. Motivation.....	6
1.2. Methodology	7
1.2.1. Contour Extraction.....	7
1.2.2. Corner Detection.....	7
1.2.3. Spline Modeling.....	8
1.2.4. Curve Fitting Technique	11
1.3. Objectives and Contribution	11
1.4. Organization of Chapters	11
CHAPTER 2 CORNER DETECTION.....	13
2.1. Proposed Algorithm.....	15
2.1.1. First Phase.....	16
2.1.2. Second Phase	18
2.1.3. Third Phase	18
2.1.4. Tuning Parameters	20
2.2. Results and Analysis.....	20

2.3. Conclusion	50
CHAPTER 3 GENERIC CUBIC SPLINE MODELING	52
3.1. Interpolant Form	55
3.2. Local Support Basis Form	63
3.2.1. Curve Design	72
3.2.2. Curve Representation.....	73
3.3. Extension to Special Class: Timmer Parametric Cubic Spline.....	75
3.3.1. Introduction to Timmer Parametric Cubic.....	75
3.3.2. Properties of Timmer Parametric Cubic	78
3.3.3. Designing GC^2 Continuous Piecewise Timmer Curve using Iterative Scheme.....	83
3.3.4. Shape Control for Timmer Curves.....	88
3.4. Results and Analysis	92
3.4.1 Interpolant Form	94
3.4.2 Local Support Basis Form	98
3.4.3 Timmer Parametric Cubic.....	112
3.5. Conclusion	116
CHAPTER 4 CURVE DESIGN	117
4.1 Preprocessing	123
4.1.1 Finding Boundary of Planar Object.....	123
4.1.2 Corner Detection.....	125
4.2 Curve Fitting with Cubic Spline Model.....	127

4.2.1 Cubic Spline Interpolant Form.....	128
4.3 Knot Insertion	131
4.3.1. Randomized Knot Insertion	132
4.3.2. Fuzzy Random Knot Insertion	142
4.4 Results and Discussion	146
4.5 Conclusion	464
CHAPTER 5 CONCLUSION AND FUTURE WORK.....	465
5.1. Conclusion	465
5.1.1. Corner Detection.....	466
5.1.2. Spline Modeling.....	466
5.1.3. Introduction of Smoothness in Curve Design.....	467
5.1.4. Introduction of Shape Parameters.....	468
5.1.5. Development of Interpolant Form of Spline Model	468
5.1.6. Development of Local Support Basis Form.....	469
5.1.7. Planar Object Approximation and Designing	469
5.2. Future Work.....	470
5.2.1. Designing of tuning parameter independent corner detector	470
5.2.2. Enhancement of curve fitting technique in terms of time and space complexity.....	471
5.2.3. Extension of concepts to 3D geometry	471
References:.....	472

LIST OF FIGURES

Figure 2.1 Possible slope transitions in digital curves.....	17
Figure 2.2 Close coordinates.....	18
Figure 2.3 Close coordinates removed after phase 2	18
Figure 2.4 Unwanted clustered points	19
Figure 2.5 Points sequence for angle calculation.....	19
Figure 2.6 Removal of clustered points after phase 3.....	19
Figure 2.7 Detected corner points for im1 as per parameters given in Table 2.1	23
Figure 2.8 Detected corner points for im2 as per parameters given in Table 2.1	25
Figure 2.9 Detected corner points for im3 as per parameters given in Table 2.1	27
Figure 2.10 Detected corner points for im4 as per parameters given in Table 2.1	29
Figure 2.11 Detected corner points for im5 as per parameters given in Table 2.1.....	31
Figure 2.12 Detected corner points for im6 as per parameters given in Table 2.1	33
Figure 2.13 Detected corner points for im7 as per parameters given in Table 2.1	35
Figure 2.14 (a) Corner points detection using (a) Marji, results without collinear points suppression [22] (b) Marji, results with collinear points suppression [22].....	37
Figure 2.15 (b) Corner points detection using	38
Figure 2.16 Corner points detection using (a) Guru [14] (b) Chang [77] (c) Tsai [61] and (d) SRM05	42
Figure 2.17 Testing SRM05 with different tuning parameters	46
Figure 2.18 Results of SRM05 for functions using default tuning parameter values. Default are $ZCT = 7$, $DT = 5$ and $TA = 152$	47

Figure 2.19 Rotation testing of SRM05 using default tuning parameter values Default are ZCT = 7, DT = 5 and TA = 152.....	49
Figure 3.1 Blending function of Timmer curve	76
Figure 3.2 Timmer curve disobeying convex hull property	79
Figure 3.3 Timmer parametric cubic disobey VDP	80
Figure 3.4 Default values of shape parameters. $\omega_i = 1$ and $\nu_i = 0, \forall i$	94
Figure 3.5 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = 10, \forall i$	95
Figure 3.6 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = 50, \forall i$	95
Figure 3.7 Interval values of shape parameter. $\omega_i = 100$ and $\nu_i = 0, \forall i$	96
Figure 3.8 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = -3, \forall i$	96
Figure 3.9 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = -15, \forall i$	97
Figure 3.10 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = -25, \forall i$	97
Figure 3.11 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = -50, \forall i$	98
Figure 3.12 Default values of parameters $\omega_i = 1$ and $\nu_i = 0$ for shape Pot.....	98
Figure 3.13 Default values of parameters $\omega_i = 1$ and $\nu_i = 0$ for shape Square.....	99
Figure 3.14 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = 10$	99
Figure 3.15 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = 100$	100
Figure 3.16 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = 10$	100
Figure 3.17 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = 100$	101
Figure 3.18 Interval tension values of parameters for bottom segment $\omega = 10$ and $\nu = 0$	101

Figure 3.19 Interval tension values of parameters for bottom segment $\omega = 100$ and $\nu = 0$	102
Figure 3.20 Interval tension values of parameters for bottom segment $\omega = 10$ and $\nu = 0$	102
Figure 3.21 Interval tension values of parameters for bottom segment $\omega = 100$ and $\nu = 0$	103
Figure 3.22 Point tension values at specified points $\omega = 1$ and $\nu = 10$	103
Figure 3.23 Point tension values at specified points $\omega = 1$ and $\nu = 100$	104
Figure 3.24 Point tension values at specified points $\omega = 1$ and $\nu = 10$	104
Figure 3.25 Point tension values at specified points $\omega = 1$ and $\nu = 100$	105
Figure 3.26 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -3$	105
Figure 3.27 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -5$	106
Figure 3.28 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -8$	106
Figure 3.29 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -10$	107
Figure 3.30 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -20$	107
Figure 3.31 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -25$	108
Figure 3.32 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -50$	108
Figure 3.33 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -3$	109
Figure 3.34 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -5$	109
Figure 3.35 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -8$	110
Figure 3.36 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -10$	110
Figure 3.37 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -20$	111

Figure 3.38 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -50$	111
Figure 3.39 Global values of shape parameters $\alpha_i = 1$ and $\beta_i = 1$	112
Figure 3.40 Global values of shape parameters $\alpha_i = 2$ and $\beta_i = 2$	112
Figure 3.41 Global values of shape parameters $\alpha_i = 3$ and $\beta_i = 3$	113
Figure 3.42 Global values of shape parameters $\alpha_i = 4$ and $\beta_i = 4$	113
Figure 3.43 Global values of shape parameters $\alpha_i = 20$ and $\beta_i = 20$	114
Figure 3.44 Interval tension at base $\alpha = 15$ and $\beta = 15$, for <i>default</i> = 1	114
Figure 3.45 Global values of shape parameters $\alpha_i = -10$ and $\beta_i = -10$	115
Figure 3.46 Interval tension at base $\alpha = -5$ and $\beta = -5$, for <i>default</i> = 3	115
Figure 4.1 Outline capturing of the digital images	122
Figure 4.2 Bitmap image.....	124
Figure 4.3 Contour of the bitmap image.....	125
Figure 4.4 Corner Points of the Image.....	127
Figure 4.5 Contour Division into Pieces.....	128
Figure 4.6 Spline fit over Object Contour	131
Figure 4.7 Calculation of random knot in Algorithm 4.1	134
Figure 4.8 Calculation of random knot in Algorithm 4.2 and 4.3	137
Figure 4.9 Calculation of random knot in Algorithm 4.4	140
Figure 4.10 Calculation of random knot in Algorithm 4.5	141
Figure 4.11 Fuzzy membership criteria	143
Figure 4.12 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =1	149

Figure 4.13 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =2	152
Figure 4.14 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =3	155
Figure 4.15 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =1	159
Figure 4.16 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =2	162
Figure 4.17 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =3	165
Figure 4.18 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =1	169
Figure 4.19 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =2	172
Figure 4.20 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =3	175
Figure 4.21 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =1	179
Figure 4.22 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =2	182
Figure 4.23 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =3	185

Figure 4.24 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =1	189
Figure 4.25 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =2	192
Figure 4.26 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =3	195
Figure 4.27 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =1	199
Figure 4.28 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =2	202
Figure 4.29 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =3	205
Figure 4.30 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =1	208
Figure 4.31 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =2	211
Figure 4.32 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =3	214
Figure 4.33 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =1	217
Figure 4.34 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =2	220

Figure 4.35 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =3	223
Figure 4.36 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =1	226
Figure 4.37 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =2	229
Figure 4.38 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =3	232
Figure 4.39 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =1	235
Figure 4.40 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =2	238
Figure 4.41 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =3	241
Figure 4.42 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =1	244
Figure 4.43 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =2	247
Figure 4.44 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =3	250
Figure 4.45 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =1	253

Figure 4.46 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =2	256
Figure 4.47 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =3	259
Figure 4.48 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =1	262
Figure 4.49 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =2	265
Figure 4.50 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =3	268
Figure 4.51 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =1	271
Figure 4.52 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =2	274
Figure 4.53 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =3	277
Figure 4.54 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =1	280
Figure 4.55 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =2	283
Figure 4.56 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =3	286

Figure 4.57 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =1	289
Figure 4.58 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =2	292
Figure 4.59 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =3	295
Figure 4.60 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =1	298
Figure 4.61 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =2	301
Figure 4.62 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =3	304
Figure 4.63 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =1	307
Figure 4.64 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =2	310
Figure 4.65 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =3	313
Figure 4.66 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =1	317
Figure 4.67 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =2	320

Figure 4.68 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =3	323
Figure 4.69 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =1	327
Figure 4.70 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =2	330
Figure 4.71 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =3	334
Figure 4.72 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =1	338
Figure 4.73 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =2	342
Figure 4.74 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =3	346
Figure 4.75 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =1	350
Figure 4.76 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =2	353
Figure 4.77 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =3	356
Figure 4.78 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =1	360

Figure 4.79 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =2	364
Figure 4.80 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =3	368
Figure 4.81 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =1	371
Figure 4.82 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =2	374
Figure 4.83 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =3	377
Figure 4.84 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =1	381
Figure 4.85 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =2	384
Figure 4.86 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =3	388
Figure 4.87 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =1	392
Figure 4.88 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =2	396
Figure 4.89 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =3	400

Figure 4.90 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =1	404
Figure 4.91 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =2	408
Figure 4.92 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =3	412
Figure 4.93 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =1	416
Figure 4.94 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =2	419
Figure 4.95 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =3	422
Figure 4.96 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =1	426
Figure 4.97 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =2	429
Figure 4.98 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =3	433
Figure 4.99 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =1	436
Figure 4.100 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =2	439

Figure 4.101 Algorithm 4.6: Demonstration of spline fitting at each iteration using

threshold =3 442

LIST OF TABLES

Table 2.1 Parameter values for 8 tested shapes	22
Table 2.2 Comparison of algorithm evaluation for im1	24
Table 2.3 Comparison of algorithm evaluation for im2	26
Table 2.4 Comparison of algorithm evaluation for im3	28
Table 2.5 Comparison of algorithm evaluation for im4	30
Table 2.6 Comparison of algorithm evaluation for im5	32
Table 2.7 Comparison of algorithm evaluation for im6	34
Table 2.8 Comparison of algorithm evaluation for im7	36
Table 2.9 Comparison of algorithm evaluation	39
Table 2.10 Comparison of algorithm evaluation for figure 2.15.1	42
Table 2.11 Comparison of algorithm evaluation for figure 2.15.2	43
Table 2.12 Comparison of algorithm evaluation for figure 2.15.3	43
Table 2.13 Comparison of algorithm evaluation for figure 2.15.4	44
Table 2.14 Comparison of algorithm evaluation for figure 2.15.5	44
Table 2.15 Comparison of algorithm evaluation for figure 2.15.6	45
Table 4.1 Evaluation of algorithm 4.1 for Arabic word “Ali”	155
Table 4.2 Evaluation of algorithm 4.2 for Arabic word “Ali”	165
Table 4.3 Evaluation of algorithm 4.3 for Arabic word “Ali”	175
Table 4.4 Evaluation of algorithm 4.4 for Arabic word “Ali”	185
Table 4.5 Evaluation of algorithm 4.5 for Arabic word “Ali”	195
Table 4.6 Evaluation of algorithm 4.6 for Arabic word “Ali”	205

Table 4.7 Evaluation of algorithm 4.1 for object “Apple”	214
Table 4.8 Evaluation of algorithm 4.2 for object “Apple”	223
Table 4.9 Evaluation of algorithm 4.3 for object “Apple”	232
Table 4.10 Evaluation of algorithm 4.4 for object “Apple”	241
Table 4.11 Evaluation of algorithm 4.5 for object “Apple”	250
Table 4.12 Evaluation of algorithm 4.6 for object “Apple”	259
Table 4.13 Evaluation of algorithm 4.1 for object “Plane”	268
Table 4.14 Evaluation of algorithm 4.2 for object “Plane”	277
Table 4.15 Evaluation of algorithm 4.3 for object “Plane”	286
Table 4.16 Evaluation of algorithm 4.4 for object “Plane”	295
Table 4.17 Evaluation of algorithm 4.5 for object “Plane”	304
Table 4.18 Evaluation of algorithm 4.6 for object “Plane”	313
Table 4.19 Evaluation of algorithm 4.1 for English alphabet “D”	323
Table 4.20 Evaluation of algorithm 4.2 for English alphabet “D”	334
Table 4.21 Evaluation of algorithm 4.3 for English alphabet “D”	346
Table 4.22 Evaluation of algorithm 4.4 for English alphabet “D”	356
Table 4.23 Evaluation of algorithm 4.5 for English alphabet “D”	368
Table 4.24 Evaluation of algorithm 4.6 for English alphabet “D”	377
Table 4.25 Evaluation of algorithm 4.1 for Object “Mult_Seg_Plane”	388
Table 4.26 Evaluation of algorithm 4.2 for Object “Mult_Seg_Plane”	400
Table 4.27 Evaluation of algorithm 4.3 for Object “Mult_Seg_Plane”	412
Table 4.28 Evaluation of algorithm 4.4 for Object “Mult_Seg_Plane”	422
Table 4.29 Evaluation of algorithm 4.5 for Object “Mult_Seg_Plane”	433

Table 4.30 Evaluation of algorithm 4.6 for Object “Mult_Seg_Plane”	442
Table 4.31 Evaluation of algorithm 4.1 for object ‘Ali’	443
Table 4.32 Evaluation of algorithm 4.2 for object ‘Ali’	443
Table 4.33 Evaluation of algorithm 4.3 for object ‘Ali’	444
Table 4.34 Evaluation of algorithm 4.4 for object ‘Ali’	444
Table 4.35 Evaluation of algorithm 4.5 for object ‘Ali’	445
Table 4.36 Evaluation of algorithm 4.6 for object ‘Ali’	445
Table 4.37 Evaluation of algorithm 4.1 for object ‘Apple’	446
Table 4.38 Evaluation of algorithm 4.2 for object ‘Apple’	446
Table 4.39 Evaluation of algorithm 4.3 for object ‘Apple’	447
Table 4.40 Evaluation of algorithm 4.4 for object ‘Apple’	447
Table 4.41 Evaluation of algorithm 4.5 for object ‘Apple’	448
Table 4.42 Evaluation of algorithm 4.6 for object ‘Apple’	448
Table 4.43 Evaluation of algorithm 4.1 for object ‘Plane’	449
Table 4.44 Evaluation of algorithm 4.2 for object ‘Plane’	449
Table 4.45 Evaluation of algorithm 4.3 for object ‘Plane’	450
Table 4.46 Evaluation of algorithm 4.4 for object ‘Plane’	450
Table 4.47 Evaluation of algorithm 4.5 for object ‘Plane’	451
Table 4.48 Evaluation of algorithm 4.6 for object ‘Plane’	451
Table 4.49 Evaluation of algorithm 4.1 for English character ‘D’	452
Table 4.50 Evaluation of algorithm 4.2 for English character ‘D’	452
Table 4.51 Evaluation of algorithm 4.3 for English character ‘D’	453
Table 4.52 Evaluation of algorithm 4.4 for English character ‘D’	453

Table 4.53 Evaluation of algorithm 4.5 for English character ‘D’	454
Table 4.54 Evaluation of algorithm 4.6 for English character ‘D’	454
Table 4.55 Evaluation of algorithm 4.1 for object Mult_Seg_Plane	455
Table 4.56 Evaluation of algorithm 4.2 for object Mult_Seg_Plane	455
Table 4.57 Evaluation of algorithm 4.3 for object Mult_Seg_Plane	456
Table 4.58 Evaluation of algorithm 4.4 for object Mult_Seg_Plane	456
Table 4.59 Evaluation of algorithm 4.5 for object Mult_Seg_Plane	457
Table 4.60 Evaluation of algorithm 4.6 for object Mult_Seg_Plane	457
Table 4.61 Evaluation of space efficiency in terms of reduction of dataset	458

THESIS ABSTRACT

NAME: AIMAN RASHEED

TITLE: DESINGING OF OBJECTS USING SMOOTH CUBIC SPLINES

MAJOR FIELD: COMPUTER SCIENCE

DATE OF DEGREE: DECEMBER 2005.

With the growing trend of computer graphics application in various fields such as, Automobile industry, Aeronautical engineering, Ships designing, Mechanical instrumentation, Animation, Fonts designing etc, it is desirable to have mathematical representation of physical objects in curve and/or surface formulation.

A lot of research is done in order to acquire an optimized curve designing scheme. The curves which are robust and easy to control and compute are of more interest. Of all the possibilities in curve designing, the cubic functions are the most powerful as they can be used to define space curves and curves with inflections. More over these functions are easy and efficient in computation. The ideas such as end point interpolation, detection of characteristic points, least square approximation, recursive subdivision and parameterization can be used for curve fitting.

MASTER OF SCIENCE DEGREE

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS, DHAHRAN.

DECEMBER 2005

ملخص الرسالة

الاسم: ايمن رشيد

عنوان الرسالة: تصميم الأشياء باستخدام الشرائح الملساء من الدرجة الثالثة.

التخصص: علوم الحاسب الآلي والمعلومات

تاريخ التخرج: ذو الحجة 1426

مع تزايد تطبيقات الرسم بالحاسوب في تخصصات مختلفة، منها صناعة السيارات، هندسة الطيران، تصميم السفن،
التجهيزات الميكانيكية، تحريك الأجسام بالحاسب، تصميم خطوط الكتابة إلى غير ذلك. كل هذا يتطلب إيجاد تمثيل
رياضي لشتى المجسمات عن طريق منحنيات أو أسطح.

لقد نفذت أبحاث كثيرة للحصول على طريقة مثلى لتصميم المنحنيات... تشكل المنحنيات القوية والتي يسهل التحكم فيها
الاهتمام الأكبر من قبل الباحثين. من ضمن الحالات المتوفرة لتصميم المنحنيات تعتبر الدوال من الدرجة الثالثة
الأكثر قوة لأنها الدوال الأقل درجة التي يمكن استخدامها لتعريف منحنيات ذات ثلاث أبعاد ومنحنيات تتوفر على
نقاط انثناء. زيادة على ذلك، تعتبر هذه الدوال سهلة وفعالة فيما يخص حسابها. تعتبر الأفكار التي تخص إقحام النقاط
النهائية، اكتشاف النقاط المميزة، التقريب بطريقة المربعات الصغيرة، التقسيم المتكرر والتمثيل بالعامل طرق تمكن
استعمالها في ملائمة المنحنيات

CHAPTER 1

INTRODUCTION

The term spline goes back to the long flexible strips of metal used by draftsmen to lay out the surfaces of ships, cars, aircrafts etc. The weights were attached to those strings in order to give smooth shapes. The Splines are actually piecewise polynomial parametric curves, generated by varying a parameter over a specified range. Spline curve fitting techniques can be found in [72,73,76]. Among all other splines, B-Splines, Hermite and Beziér are the basis for research work in this field.

B-Spline consists of curve segments whose polynomial co-efficient depend on just a few control points. This behavior is called as local control. Thus moving a control point affects only a small part of the curve. This local behavior is due to the fact that each vertex is associated with a unique basis function. The B-Spline basis allows the order of basis function and hence the degree of the resulting curves to be changed without changing the number of defining polygon vertices. Some evolutionary methods, using B-splines, can be seen in [32,42-45]. These methods are based upon knot selection. Hermit

curves on the other hand are defined by two end points P_1 and P_4 , and their respective tangent vectors R_1 and R_4 . Bézier curves are developed by Pierre Bézier for use in designing automobiles at Renault. The Bézier form of cubic polynomial curve segment has four control points P_0 , P_1 , P_2 , and P_3 . Two intermediate points P_1 and P_2 are not on the curve. The Bézier curve interpolates the two end control points P_0 and P_3 and approximates the two intermediate points P_1 and P_2 .

The two basic ways of manipulating curve design and shape using control polygon are through developing B-Spline like basis functions and control points interpolation. A combined technique is shown in [70]. Another useful class of spline curves is known as Rational Cubic Spline [78]. Rational curves can define space curves and curves with inflections. A technique to fit a curve to planar digital data is presented in [33]. For large data point set, the characteristic points are identified in a recursive manner by enhancing Davis algorithm. The rational cubic is converted to rational hermit cubic form to manipulate the shape. These characteristic points are searched based upon high curvature point. For achieving a better fit, sub-division is done on the basis of a threshold value. Conic representation of curves and surfaces is presented in [69]. In this case the rational cubic is broken at its mid point to form two conic sections. The advantage of this scheme is in terms of computational cost. Smoothness to some extent is also achieved. The price is paid when shape control is considered. The rational cubic splines with linear denominator sometimes do not work as they are not bounded in the specified regions [56]. The introduction of weights in rational cubic spline solves this problem. Local

support basis form is formulated for C^2 rational cubic curve and the effect of weight is analyzed in [55]. The spline curves and surfaces can also be modeled using trigonometric functions [21]. Trigonometric blending functions are used in the construction of curvature continuous curves. Since such parameterization is disadvantageous because of slow computation of trigonometric values and instability in the neighborhood of 0 degrees, therefore it is usually converted into polynomial or rational form.

The splines can generally be represented in implicit, explicit or parametric form.

Implicit Form

In this case we can express y as explicit function of x (e.g. $y = f(x)$). The difficulties with this approach are;

- It is impossible to get multiple values of y for a single value of x , so curves such as circle and ellipse must be represented by multiple curve segments.
- Such curves are not rotationally invariant and may require breaking a curve segment into many segments.
- Describing curves with vertical tangents is difficult, because a slope of infinity is difficult to represent.

Explicit Form

We can choose to model curves as solutions to implicit equations of the form $y = f(x)$. The difficulties with this approach are,

- The given equation may have more solutions than required, for example in modeling a circle, we might want to use $x^2 + y^2 = 1$, which is fine. But how do we model one-half of a circle? We must add constraints such as $x \geq 0$ which cannot be contained within the implicit equation.
- If two implicit defined curve segments are joined together, it may be difficult to determine whether their tangent directions agree at joining point.

Parametric Form

The parametric form overcomes the problems caused by functional and implicit forms. The points on a curve are represented as ordered set of values, $p_i = [x_i, y_i]$. There are corresponding parametric functions that may be used to represent arbitrary curves; these are of the form $Q(t) = [x(t), y(t)]$. The parameter t takes the values from a specified range; conventionally from 0 to 1. a curve represented in this way can be thought of as the projection of the curve in 3-D, t as the third dimension, perpendicular to x and y plane. A unit circle can be represented in parametric form $Q(t) = [\cos(t), \sin(t)]$. Parametric form of curve allows multiple values of y for single or more values of x . Parametric curves replace the use of geometric slopes (which may be infinite) with parametric tangent vectors (which are never infinite). A parametric curve is approximated by piecewise polynomial curves. Cubic polynomials are most often used because lower degree polynomials give little flexibility in controlling the shape of the curve, and higher degree polynomials require more computation and can introduce unwanted wiggles.

Before going into the peculiarities of this research work, it is necessary to encompass a very important issue related to smooth curves, known as Continuity.

To describe the shape of a free form curve, it is recommended to use several curve segments which are joined together using the constraints known as the degree of continuity. It is one of the active research areas of Computer Aided Geometric Design CAGD. The designing of continuous curves and surfaces for CAD software development has been a tough problem to deal with for a long time. Early efforts can be seen in [30,71]. There are two types of continuities; parametric continuity and geometric continuity.

Parametric continuity is denoted as C^i , which means that two adjacent curve pieces have i^{th} degree parametric continuity and all lower derivatives. C^0 means that the two pieces are joined through a common point. C^1 means that the two curves not only share the same endpoint but also their tangent vector is the same, the direction and the magnitude. C^2 means that two curves are C^1 and also their second order parametric derivate matches at the common end point.

Geometric continuity is denoted as G^i . Unlike parametric continuity it is less stringent in the way that here only the direction of tangent vector should match. The magnitude of the tangent vector could be different. G^0 is similar to C^0 . G^1 amounts to have a common

tangent vector direction. G^2 means that the adjacent segments have common tangent vector and same curvature.

It is to be noted that the line segments joining the two curve segments must be collinear. Further more if a curve is said to be geometric continuous then it means that it is also parametric continuous but this is not possible the other way round.

1.1. Motivation

A common practice of formulating an object in the physical world into digital form is by first converting the object into gray level image by scanning it. Contours or boundary points are then obtained from this gray level image. These contours lead to the corner or significant point extraction. Finally splines are used to approximate or interpolate the significant points.

Traditional approaches of object digitization [79] have the draw back of greater Human Computer Interface (HCI). Users are supposed to specify significant points by some interactive means, for example, mouse pointer or pen pointer etc. These significant points produce erroneous impact on the shape of the curve. Further more, a user is needed to keep on specifying the significant points until he gets a desired curve fit. This kind of system is not only tedious to interact with but also very inefficient and especially for complex objects it becomes very inconvenient. Usually accuracy up to desired tolerance

limit is difficult to achieve. Moreover a user has to be familiar with the underlying mathematical model of the system in order to use it properly.

A great deal of research has been done to minimize the HCI factor and automate the whole process to greater extent. This has lead to digitization of objects in much efficient and accurate way. Unlike traditional approaches, researches have proposed some very good automated corner detection algorithms. Optimal curve fitting is done by segmenting the contour outline at the corner points. The curve fitting techniques used are usually based on Bézier Cubic function where as many researchers have used other spline models as well [6,41,46,60].

1.2. Methodology

1.2.1. Contour Extraction

The first and foremost step for this research work is to find the boundary of planar object. This can be done using chain code. Chain codes give the list of edge points and their directions along the boundary. The selection of these boundary points are based on their corner strength and contour fluctuations.

1.2.2. Corner Detection

Corners in digital images give important clues for shape representation and analysis. Generally objects information can be represented in terms of its corners, thus corner

points play a very vital role in object recognition, shape representation and image interpretation.

1.2.3. Spline Modeling

An efficient way of representing 2D planar objects is by using splines, which are piece wise polynomial curves. Generally splines are used in the form of parametric equations. Suppose that $x(t)$ and $y(t)$ can supply points $(x(t), y(t))$ along the curve as t is varied then we can write the parametric form as under,

$$\begin{aligned} x(t) &= a_0 + a_1t + a_2t^2 + \dots + a_nt^n \\ y(t) &= b_0 + b_1t + b_2t^2 + \dots + b_nt^n \end{aligned} \tag{1.1}$$

The value of n describes the degree of the spline in the above equation. For $n=2$, it is known as conic. For $n=3$, it is known as cubic and so on and so forth. The approximation is done piecewise by breaking the planar object into segments. The joining points of the segments are made continuous by careful selection of polynomial coefficients. For a polynomial of degree n , smoothness or continuity of degree $n-1$ can be achieved. This approach is advantageous in a manner that it allows to derive multiple shapes from a single stored object. Further more translations are applicable here.

It is desired to come up with a spline model which has properties like smoothness, local control, and point tension. Along with these properties it is also desirable to have the representation of spline model in interpolant form and local support basis form.

- **Introduction of Smoothness in curve designing**

To attain a better shape it is required that the spline should permit the mixing of sharp and smooth sections within the same description. Continuity condition provides the solution for this requirement. To achieve shapes with cusps, a zero order continuity condition can be used. For smooth shapes higher order continuity conditions are satisfied.

- **Introduction of Local Control in curve handling**

Each control point, if moved, should only exert influence on the shape of the spline in a neighborhood rather than producing impact on the whole curve. A given spline fitting method may offer varying degrees of local control depending on the influence of any given set of control points.

- **Introduction of Point and Interval Tension in curve rendering and their effect on object shapes**

To achieve point and interval tension we need to introduce the shape parameters associated with each point and interval. The change of shape parameter such that it affects only on the neighborhood of a specific point is known as point tension. The change of the shape parameters such that it affects the curve in the specified interval is known as interval tension. The increase in shape parameter of two consecutive

points tightens the curve towards a line segment joint by those control points, thus producing the same effect as that of interval tension.

- **Development of Interpolant Form of the Spline Model**

In this approach, the parametric values of the spline are made to pass through all the given set of data points. An interpolating function is devised to find those parametric values which do not match with the given set of data points. This technique is suitable in cases when the data points describing the contour of the object are sufficiently smooth and accurate with no sharp edges.

- **Development of Local Support Basis Form**

This approach is not as much restrictive as capture by Interpolation. In this case it is sufficient that the spline is made to pass close to the given set of data points. The proximity criterion between parametric point and given point is usually taken as distance along a coordinate or along a normal to the captured curve. A tolerance limit is defined for this distance, which could be an approximation based on the average. This approach is useful when the object to be approximated is not smooth. We can introduce the local support in the spline by transforming it from the control points to piece wise Bernstein-Bézier representation.

1.2.4. Curve Fitting Technique

Different techniques, like recursive algebraic fitting, piecewise polynomial fitting etc, have been employed for curve fitting. It is desired to formulate an efficient technique in such a way that there is no tradeoff for quality.

1.3. Objectives and Contribution

The research work is aimed to propose an efficient strategy for object designing using smooth cubic splines. To achieve this objective, several areas needed to be analyzed and studied in depth. The objectives are as under;

- Finding set of corner points from the object contour in such a way that they describe the shape of the object and as well as they are not in greater number.
- Designing of interpolant form as well as local support basis form of smooth parametric cubic spline model.
- Introduction of point and interval tension in the formulation of spline model.
- Designing of objects using spline model.
- Approximating the object contour using the spline model.

1.4. Organization of Chapters

The organization of this thesis is as follows. Chapter 1 is introduction to the objective and motivation. Chapter 2 covers Corner Detection. We have presented our proposed strategy. We cover the development of Interpolant form and Free form of cubic generic

spline model in Chapter 3. This chapter is also the backbone of our thesis contributions.

In Chapter 4 we discuss the proposed methodologies of object design and approximation.

Finally we conclude and present future possibilities in Chapter 5.

CHAPTER 2

CORNER DETECTION

Corners in digital images give important clues for shape representation and analysis. Generally objects information can be represented in terms of its corners, thus corner points play a very vital role in object recognition, shape representation and image interpretation [8,9,11,12,22,60,64]. Corners are robust features in the sense that they provide important information regarding objects under translation, rotation and scales change. A shape can be presented compactly, efficiently and accurately if corners are detected aptly. Since the mathematical notion of a corner is that of a high curvature point in planer curves [15,22-24,36,60], therefore most of the corner detection algorithms are based on curvature evaluation or calculation of opening angle at each contour point . This approach is effective for smoother shapes. In case of noisy shapes it detects false corners. Like other approaches, we are also considering the corner point as any point with a change in slope with respect to previous state of slope. The enhancement is that we have developed a scheme to reject false corners. Moreover our algorithm can also detect non-sharp corners and thus we have lower rate of false rejections. This algorithm also preserves the shape of the object.

The algorithm proposed by [23] relies on calculating the shape curvature function [57] using an adaptive filtering to remove as much noise as possible without losing corners. The authors have defined corners as the peaks of the function. The approach is stable against noise, scale and rotation affects.

An improved chain-code methodology is adapted to get a better characterization of contour [36]. This process helps in calculating the curvature at each point in an adaptive manner and thus works efficiently even if working with poor signal to noise ratio.

The curvature estimation technique [24] is approximation to curvature analysis since incase of digital curve, there does not exists any fixed definition of curvature. In this scheme the points with local maximum curvature are considered as dominant points.

A boundary based corner detection method is proposed by [14,61,77]. The geometrical centroid of the symmetrical boundary segment within the neighborhood of the point, on the boundary, under consideration is used. The distance between the centroid and the point under consideration defines the evidence of a point to be a corner point. This scheme is transformation invariant. The problem with this approach is that curves with smoother corners or edges will result in the increasing rate of false rejections.

The algorithm proposed by [15] is using both the information of local extrema as of curvature and modulus of transform through a specially designed wavelet transform function to detect corners and arcs effectively.

This chapter is organized as follows. In Section 2.1 the new corner detection algorithm is explained. The corner detection results of algorithms (including ours) are demonstrated and compared in section 2.2. Finally section 2.3 concludes this chapter.

2.1. Proposed Algorithm

The algorithm is composed of three phases. First phase is the basis of the whole algorithm where candidate points are detected using slope analysis. Points detected in the first phase are passed to the second phase for refinement, which is done by removing adjacent points. This adjacency is based upon a certain threshold. The third phase takes the refined points and removes the cluster of points lying on the jaggedly planner shape to give the final set of corner points. The cluster of points is removed using angular measurements. The tangents at each point for slope analysis are calculated as in (Equations 2.1-2.3) [69];

$$T_0 = 2(P_1 - P_0) - \frac{(P_2 - P_0)}{2} \quad (2.1)$$

$$T_n = 2(P_n - P_{n-1}) - \frac{(P_n - P_{n-2})}{2} \quad (2.2)$$

$$T_i = a_i(P_i - P_{i-1}) + (1 - a_i)(P_{i+1} - P_i) \quad (2.3)$$

where,

$$a_i = \frac{\|P_{i+1} - P_i\|}{\|P_{i+1} - P_i\| + \|P_i - P_{i-1}\|} \quad (2.4)$$

The tangent calculation is taken as a preprocessing step.

2.1.1. First Phase

The slope analysis is done on the basis of transitions. For example, there are only three possible states of slope.

- Increasing, i.e. +ve.
- Decreasing, i.e. -ve. and
- Constant, i.e. moving along any of the axis in 2D-plane.

Depending upon slope states, we have only four transitions,

- +ve \rightarrow 0
- 0 \rightarrow -ve
- -ve \rightarrow 0
- 0 \rightarrow +ve

Now any point P_i is taken as a candidate point if it comes after any one of the transitions, which means that there is a change in slope. This is shown in (Figure 2.1). Notice that if we encounter transitions 1 and 3 then we also need to take care of a small jaggy. To cater

such problem we have defined a threshold, which is describing if such transition is a proper one or just a small jaggy. Incase where it is taken as a small jaggy, we ignore it and keep record of our last transition state.

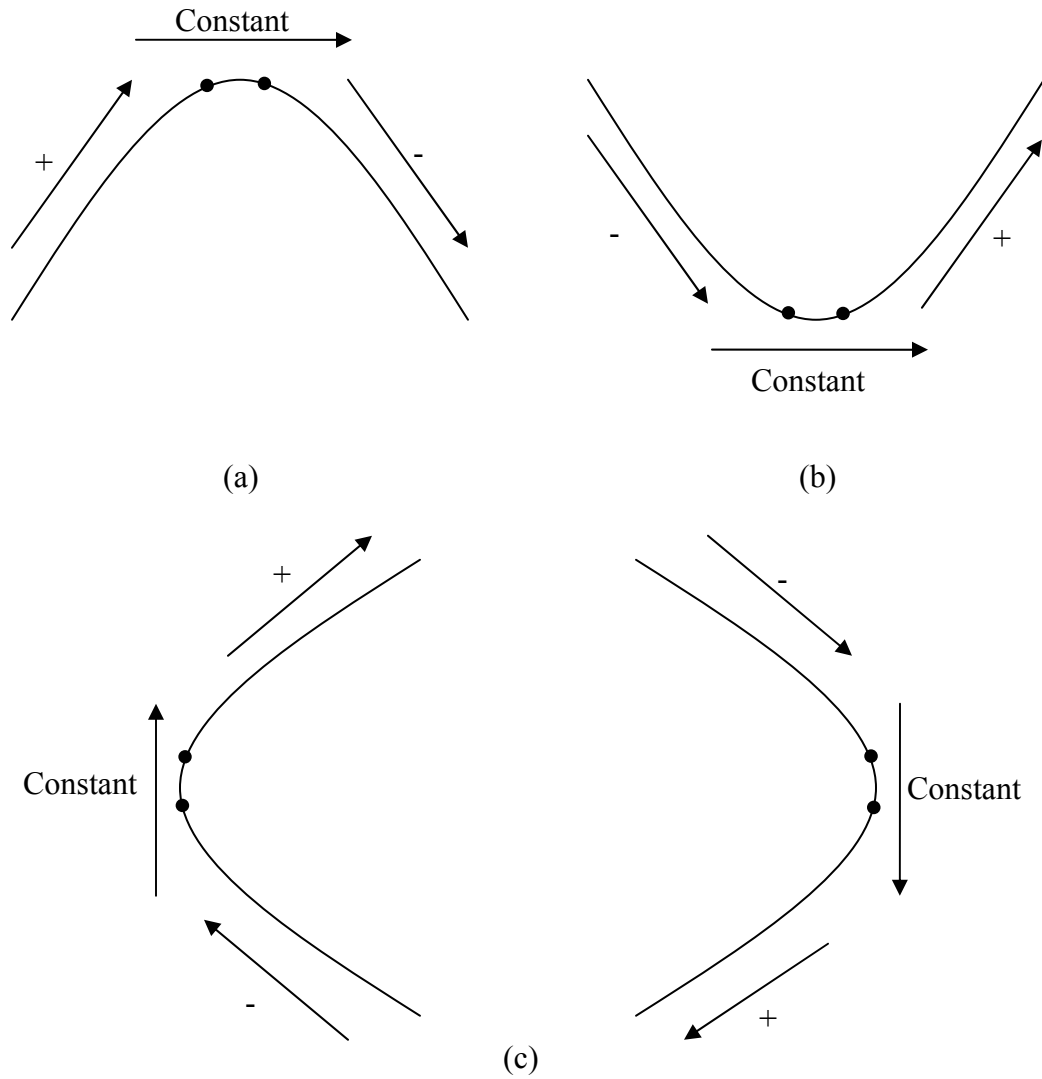


Figure 2.1 Possible slope transitions in digital curves

The advantage of this technique is that, we can apply this algorithm for smooth functions as well as irregular objects with jaggies.

2.1.2. Second Phase

Sometimes the corners to be detected are not the sharp angle points or they are the result of sharp jaggies and we may detect superfluous candidate corner points in first phase. These superfluous points are discarded in this phase. The superfluous points in this case are the ones with very close co-ordinate positions as shown in (Figure 2.2). So, in order to get refined points we remove such points. This phase actually acts as a preprocessing step for the next phase.



Figure 2.2 Close coordinates

(Figure 2.3) shows the Figure after second phase. The close coordinate points are removed.



Figure 2.3 Close coordinates removed after phase 2

2.1.3. Third Phase

The basic requirement of this phase is to remove the cluster of unwanted points as shown in (Figure 2.4). The points shown in big spots would have been enough rather than having so many intermediate points. We solved this problem by taking the angle between them. We considered $\hat{a} = P_i P_{i+1}$ as a vector and $\vec{b} = P_{i+1} P_{i+2}$ as another vector. Points are

shown in (Figure 2.5). The angle is calculated using dot product between them as in (Equation 2.5).

$$\vec{a} \cdot \vec{b} = |\hat{a}| |\hat{b}| \cos \theta \quad (2.5)$$

If this angle is greater than a certain threshold, the points are then taken as collinear and therefore the middle point is no more a corner point. This is demonstrated in (Figure 2.6).

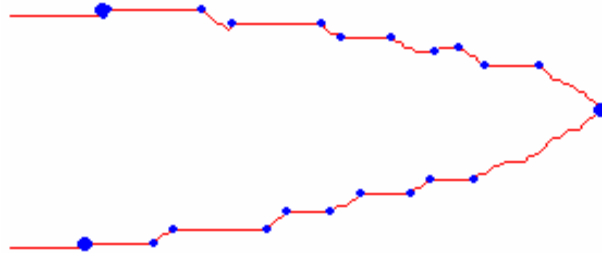


Figure 2.4 Unwanted clustered points

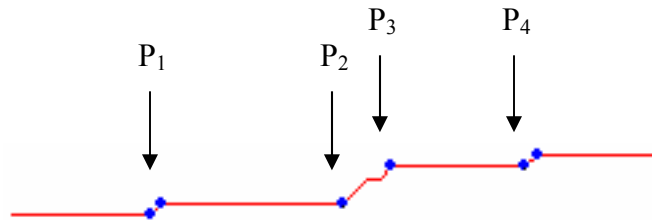


Figure 2.5 Points sequence for angle calculation

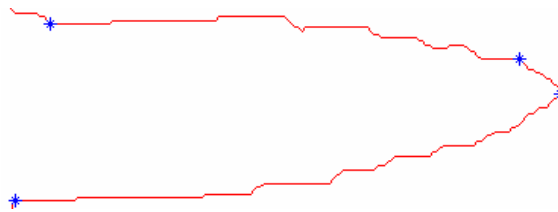


Figure 2.6 Removal of clustered points after phase 3

2.1.4. Tuning Parameters

The algorithm needs three different tuning parameters at different phases. In the first phase *Zero Count Threshold* (ZCT) is needed to differentiate a jaggy from a line. In case of a line the end points of it as corner points. The jaggies are ignored. *Distance Threshold* (DT) is used in second phase in order to remove the points nearer by this parameter. The last parameter is *Tolerance Angle* (TA) which is used to remove the clustered points. If points are clustered so that the intermediate points are not needed then such superfluous are removed from the list of final corner points. The hypothesis is that if three consecutive points are making an angle greater than TA then such points are taken as collinear points and due to which middle point is neglected. The default value of ZCT is 7, DT is 5 and TA is 152° .

A distinct property of our algorithm is that the default values of tuning parameters work equally well for almost all the shapes, either it be a smooth curve or irregular object boundary.

2.2. Results and Analysis

A number of frequently cited corner detectors were discussed in the survey by [60,85]. They selected four algorithms among them and compared the results with their corner detector. Along with these algorithms we are comparing our algorithm with [13]. The algorithms are referred as SAM04, IPAN99, BT87, FD77, RW75 and RJ73 respectively

in this paper. The default values of each algorithm is shown in (Table 2.1). The comparisons are represented in (Figures 2.7 – 2.13). SRM05 is our proposed algorithm. Further more we have also compared our algorithm with [22,62,80-84,86]; shown in (Figure 2.14). Also we tested the objects present in [14,61,77] against our approach as depicted in (Figure 2.15). After this we have tested our algorithm for different tuning parameters, shown in (Figure 2.16). Also we have demonstrated in (Figure 2.17) that our algorithm works perfect in case of smooth functions. Finally we have tested our approach for rotation affects and results are shown in (Figure 2.18).

Criteria for performance evaluation of corner detectors were given by Chetverikov and Szabo [12], which are as follow;

- *Selectivity*: It is the most important factor for any corner detector. The rate of correct detections should be high and the wrong ones should be low.
- *Single response*: Each corner should be detected only once.
- *Precision*: The positions of detected corners should be precise.
- *Robustness to noise*: The algorithm should perform well for noisy shapes as well.
- *Easy setting of parameters*: Parameters should be logical and easy to tune for variety of shapes.
- *Robustness to parameters*: Minor changes in parameter should not cause drastic changes in performance.
- *Speed*

Table 2.1 Parameter values for 8 tested shapes

Fig	Algorithm	Im1	Im2	Im3	Im4	Im5	Im6	Im7	Im8
A	SAM04	D	D	D	D	D	D	D	D
B	IPAN99	D	D	D	D	D	D	D	D
C	BT87	D	D	D	D	D	D	D	D
D	FD77	D	7,2500	D	500	1000	1300	D	1000
E	RW75	D	.15	5,2500	5,500	D	7,1000	D	D
F	RJ73	D	.15	D	D	D	D	D	D
G	SRM05	D	D	D	D	D	D	D	D

D stands for Default values for tuning parameters.

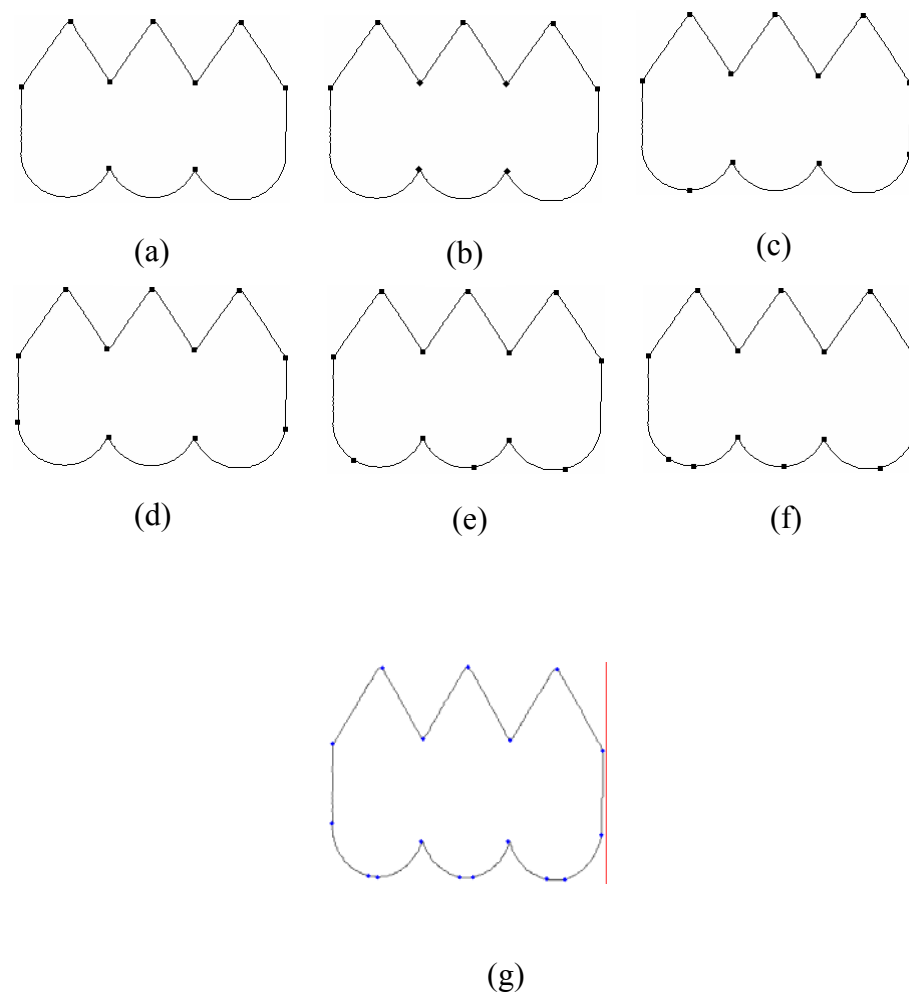


Figure 2.7 Detected corner points for im1 as per parameters given in Table 2.1

(a) SAM04. (b) IPAN99. (c) BT87. (d) FD77. (e) RW75. (f) RJ73 (g) SRM05

Table 2.2 Comparison of algorithm evaluation for im1

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
SAM04	9	9	8	0
IPAN99	9	9	8	0
BT87	11	11	6	0
FD77	11	11	6	0
RW75	12	12	5	0
RJ73	13	13	4	0
SRM05	17	17	0	0

A close inspection of results from (Figure 2.7) and (Table 2.2) shows that no false corners are detected. In fact the detected corners are representing the shape more explicitly. Unlike all other algorithms which are missing some very important corners, there is no corner missed in our approach as well. Moreover no corners are repeated.

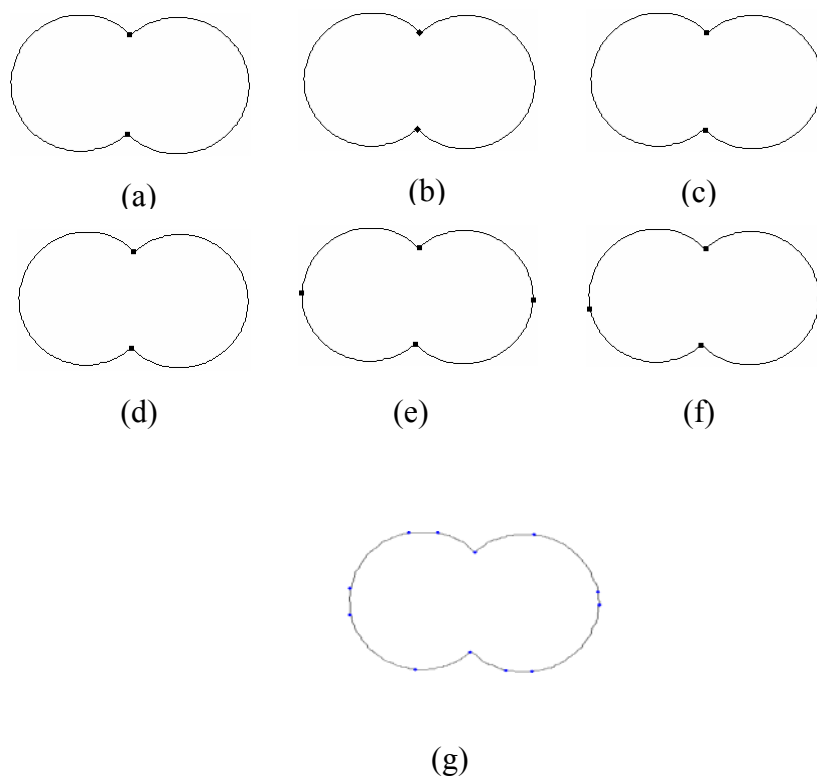


Figure 2.8 Detected corner points for im2 as per parameters given in Table 2.1

(a) SAM04. (b) IPAN99. (c) BT87. (d) FD77. (e) RW75. (f) RJ73 (g) SRM05

Table 2.3 Comparison of algorithm evaluation for im2

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
SAM04	2	2	12	0
IPAN99	2	2	12	0
BT87	2	2	12	0
FD77	2	2	12	0
RW75	4	4	10	0
RJ73	4	4	10	0
SRM05	12	12	2	0

We can examine from (Figure 2.8) and (Table 2.3) that a couple of corners are missed using our algorithm but still as compared to other algorithms the number of missed corners is very low. Shape of the object is captured in a much better way than any other algorithm. Other than our approach only (Figure 2.8e and Figure 2.8f) are preserving the shape to some extent.

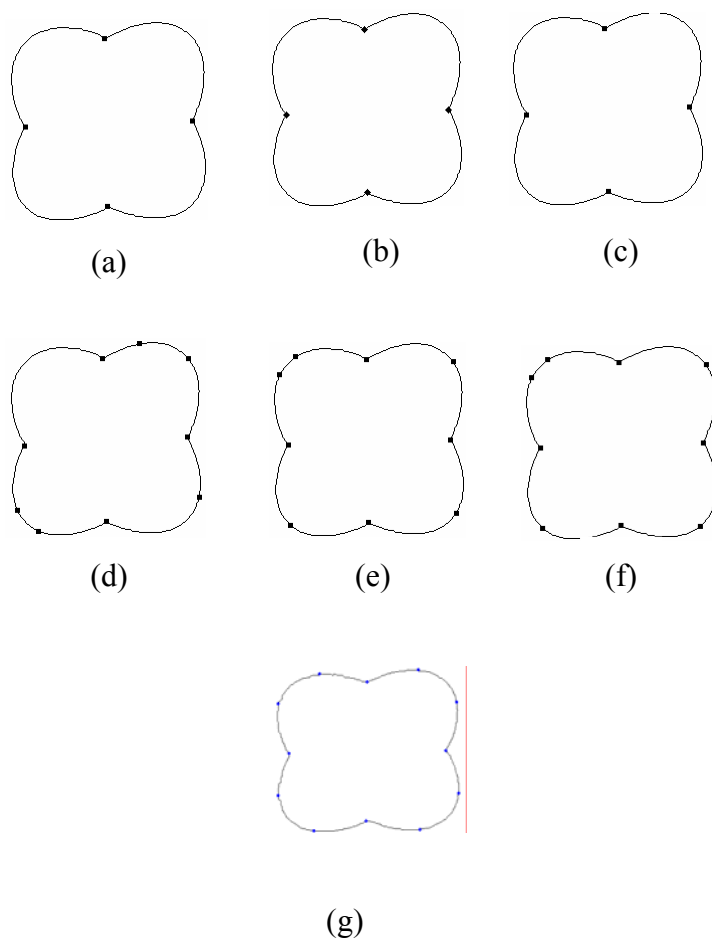


Figure 2.9 Detected corner points for im3 as per parameters given in Table 2.1

(a) SAM04. (b) IPAN99. (c) BT87. (d) FD77. (e) RW75. (f) RJ73 (g) SRM05

Table 2.4 Comparison of algorithm evaluation for im3

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
SAM04	4	4	8	0
IPAN99	4	4	8	0
BT87	4	4	8	0
FD77	9	9	3	0
RW75	9	8	3	1
RJ73	9	8	3	1
SRM05	12	12	0	0

Analysis of (Figure 2.9) and (Table 2.4) shows that our algorithm has chosen the most optimal point set as corners. Only (Figure 2.9e) and (Figure 2.9f) are good enough to preserve the shape of object except our approach. No corner is missed using this approach and also no extra corner is detected.

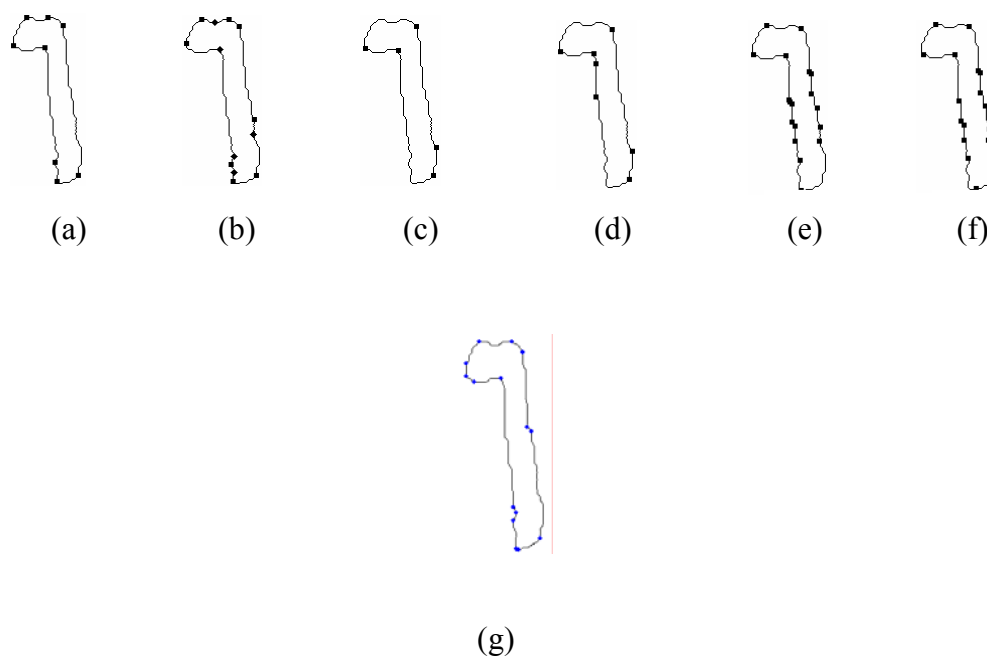


Figure 2.10 Detected corner points for im4 as per parameters given in Table 2.1

(a) SAM04. (b) IPAN99. (c) BT87. (d) FD77. (e) RW75. (f) RJ73 (g) SRM05

Table 2.5 Comparison of algorithm evaluation for im4

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
SAM04	8	7	8	1
IPAN99	13	8	8	5
BT87	5	5	6	0
FD77	7	4	6	3
RW75	17	5	5	12
RJ73	16	4	4	12
SRM05	14	9	0	5

Analysis of (Figure 2.10) and (Table 2.5) depicts that our approach works fine since it is not missing any important corner point therefore preserving the object's shape. Some of the points are still superfluous, but the number is still lesser and in the acceptable range unlike (Figure 2.10e) and (Figure 2.10f). In case of (Figure 2.10a), (Figure 2.10c) and (Figure 2.10e) the false detection percentage is low but they have missed a great number of vital corners. The analysis shows that percentage of corners missed should be less than the percentage of false detected corners.

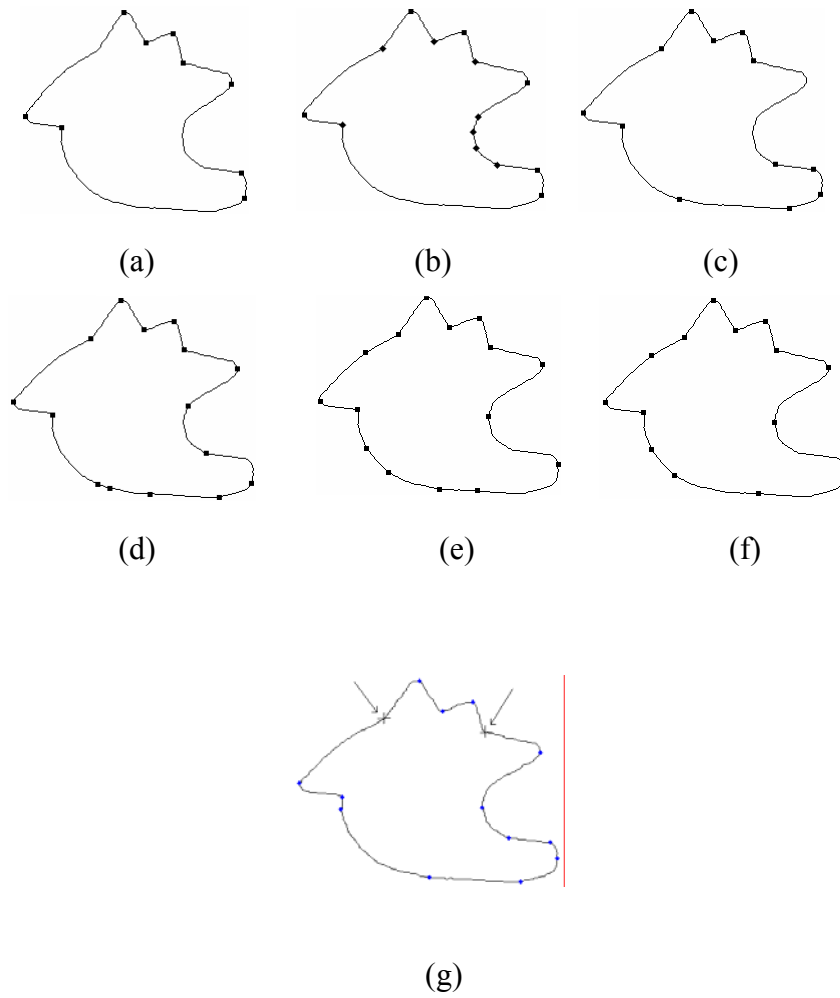


Figure 2.11 Detected corner points for im5 as per parameters given in Table 2.1.

(a) SAM04. (b) IPAN99. (c) BT87. (d) FD77. (e) RW75. (f) RJ73 (g) SRM05

Table 2.6 Comparison of algorithm evaluation for im5

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
SAM04	8	8	6	0
IPAN99	14	12	3	2
BT87	12	12	6	0
FD77	15	12	6	3
RW75	15	11	5	4
RJ73	14	11	2	3
SRM05	17	16	2	1

We can see that our algorithm is missing two vital corner points as shown in (Figure 2.11g). This is because of the fact that our phase three is removing any point lying on almost a straight line. Due to this the shape is not properly preserved. We can also observe that (Figure 2.11e) is performing best for this shape. Moreover, if we compare our algorithm against (Figure 2.11a) and (Figure 2.11c) which have not detected any false corners, we can see that those algorithms have missed more corners than ours, as depicted in (Table 2.6), and thus overall our algorithm performs better.

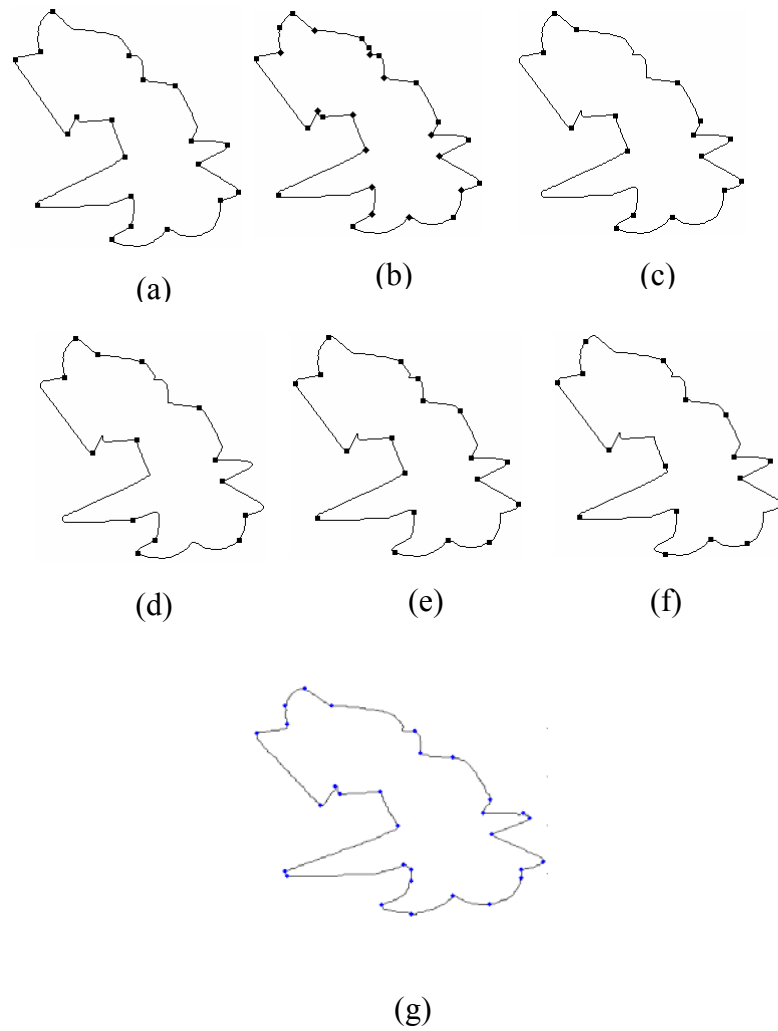


Figure 2.12 Detected corner points for im6 as per parameters given in Table 2.1

(a) SAM04. (b) IPAN99. (c) BT87. (d) FD77. (e) RW75. (f) RJ73 (g) SRM05

Table 2.7 Comparison of algorithm evaluation for im6

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
SAM04	20	20	9	0
IPAN99	28	26	4	2
BT87	16	16	13	0
FD77	14	12	17	2
RW75	19	18	9	1
RJ73	17	12	15	5
SRM05	30	27	2	3

The analysis of (Figure 2.12) shows that in case of highly irregular images, our algorithm has outperformed all other approaches. Our approach is preserving the shape most accurately since the least number of corners are missed, as we can see in (Table 2.7). Unlike other algorithms, our algorithm is detecting the most optimal set of corner points. Further more the percentage of false detected corners is very small and insignificant.

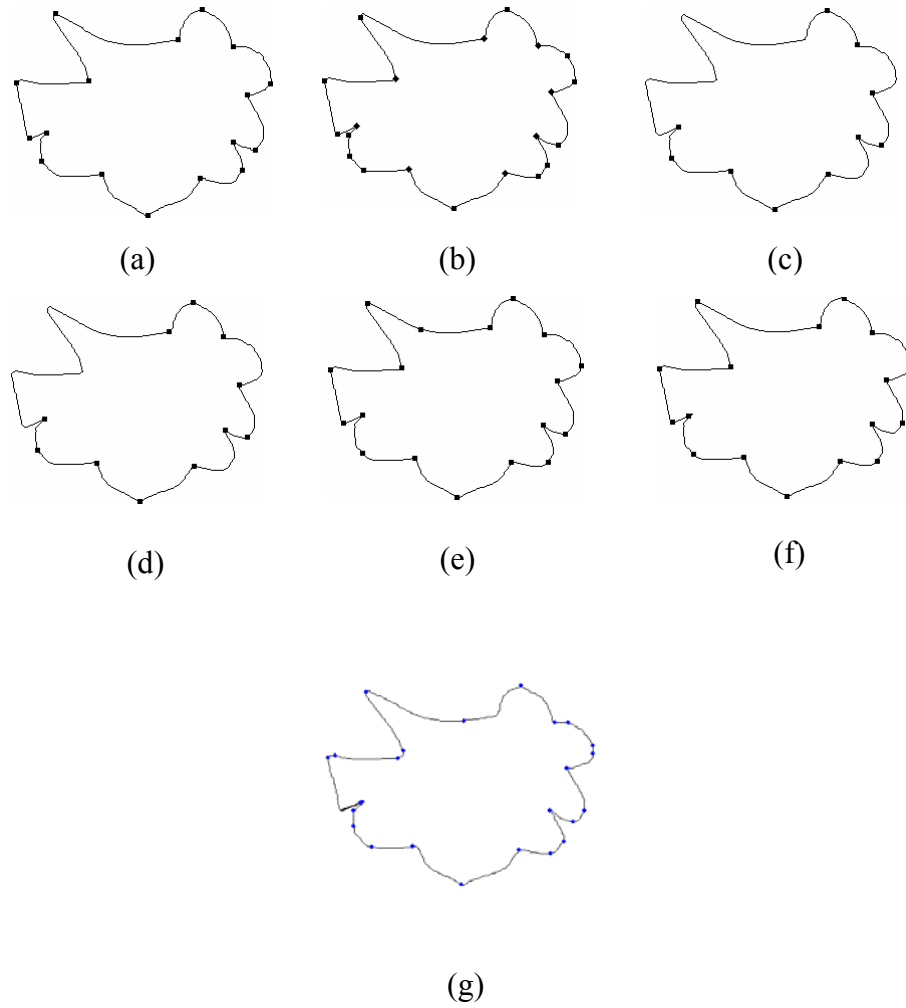


Figure 2.13 Detected corner points for im7 as per parameters given in Table 2.1

(a) SAM04. (b) IPAN99. (c) BT87. (d) FD77. (e) RW75. (f) RJ73 (g) SRM05

Table 2.8 Comparison of algorithm evaluation for im7

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
SAM04	17	17	5	1
IPAN99	21	21	2	1
BT87	10	10	14	1
FD77	11	11	11	1
RW75	18	18	4	0
RJ73	17	17	6	1
SRM05	24	20	2	4

In (Figure 2.13) we analyze that except for (Figure 2.13c) and (Figure 2.13d), all the approaches work almost the same. Most of them are properly preserving the shape of object. Even though a couple of corner points are missed but still no significant corner point is missed. Furthermore extra detected corner points are in the range of acceptable point set. In this specific case only (Figure 2.13b) has better results in terms of lesser false detections and overall shape preservation as we can observe in (Table 2.8).

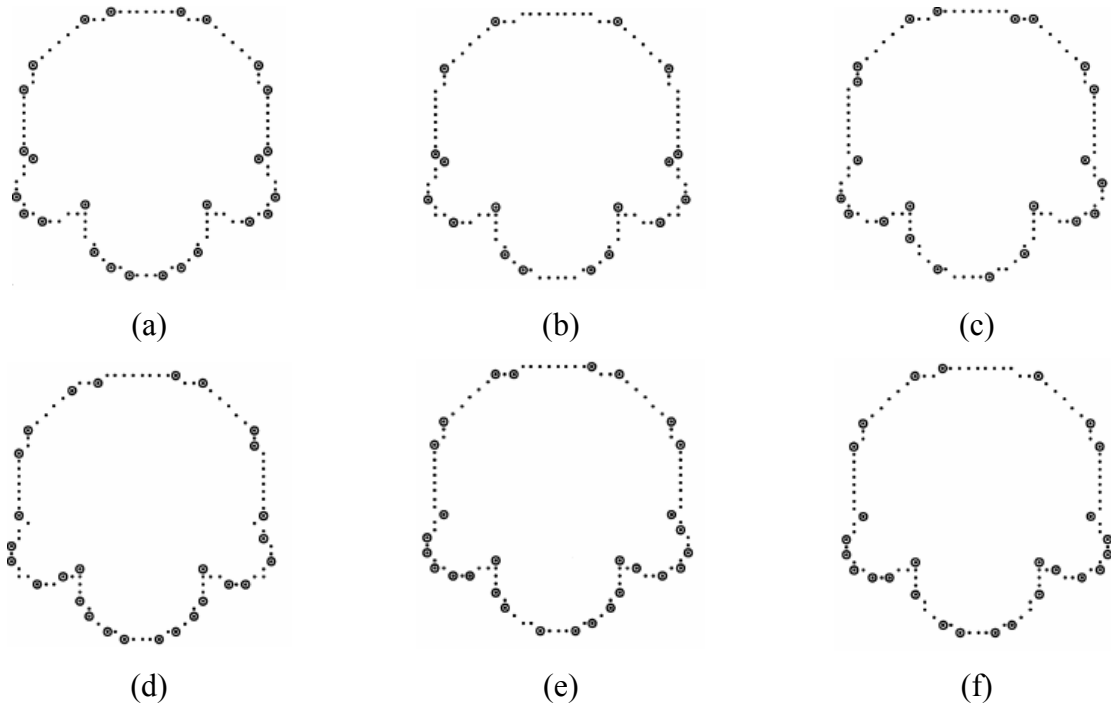


Figure 2.14 (a) Corner points detection using (a) Marji, results without collinear points suppression [22] (b) Marji, results with collinear points suppression [22] (c) The-Chin [86] (d) Ansari-Huang [84] (e) Ray-Ray [82] (f) Ray-Ray [83]

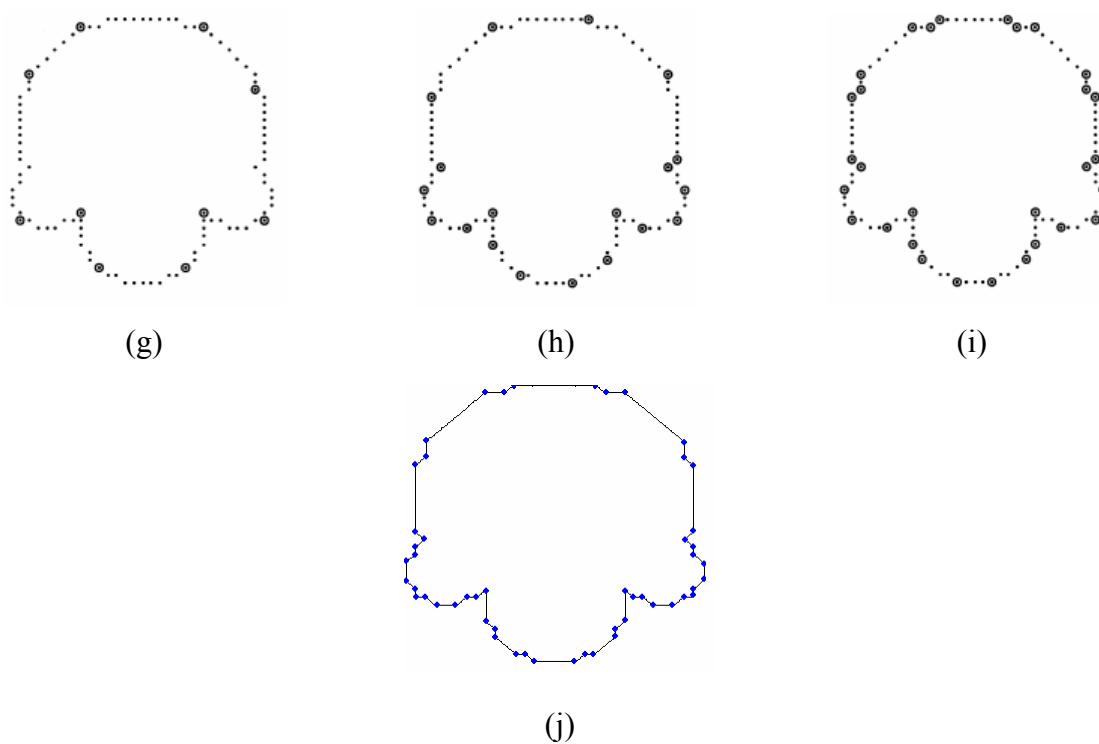


Figure 2.15 (b) Corner points detection using

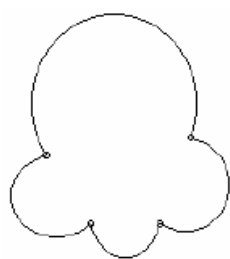
(g) Arcelli-Ramella [81] (h) Sarkar [80] (i) Cornin [62] and (j) SRM05

Table 2.9 Comparison of algorithm evaluation

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
Marji(a)	26	26	26	0
Marji(b)	18	18	34	0
The-Chin	22	22	30	0
Ansari- Huang	28	28	24	0
Ray-Ray(e)	29	29	23	0
Ray-Ray(f)	27	27	25	0
Arcelli- Ramella	10	10	42	0
Sarkar	19	19	33	0
Cornin	33	33	19	0
SRM05	52	52	0	0

We can clearly observe in (Figure 2.14) that our proposed algorithm is not missing any important corner point and each corner position is picked at a very precise position. Thus

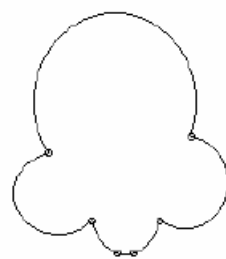
it is preserving the shape of the object. We can also see that there is no false rejection or false acceptance, as shown in (Table 2.9).



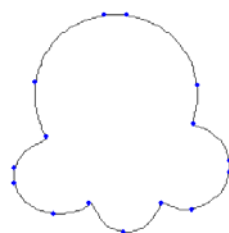
(a)



(b)

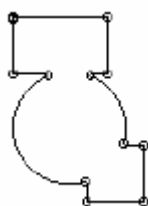


(c)



(d)

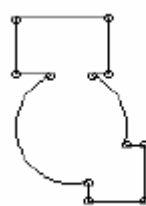
(15.1)



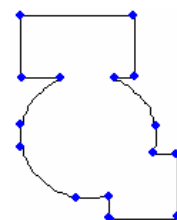
(a)



(b)

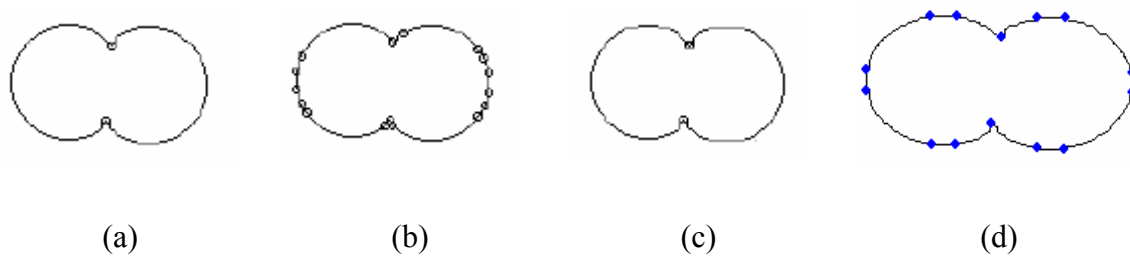


(c)

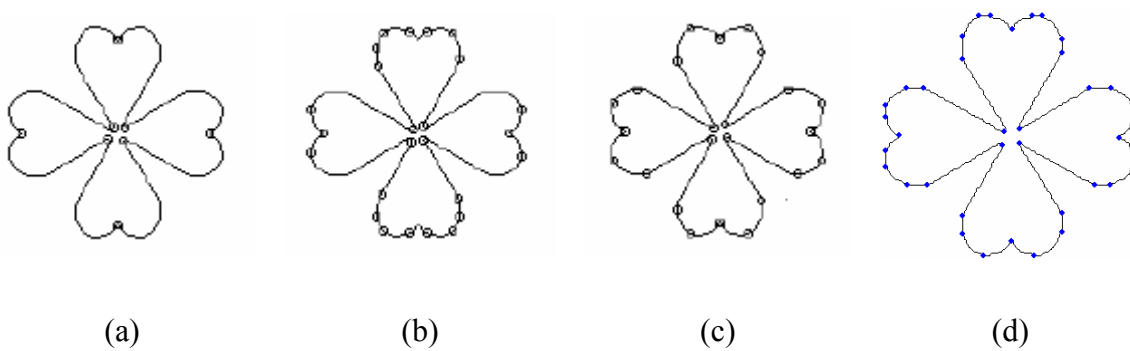


(d)

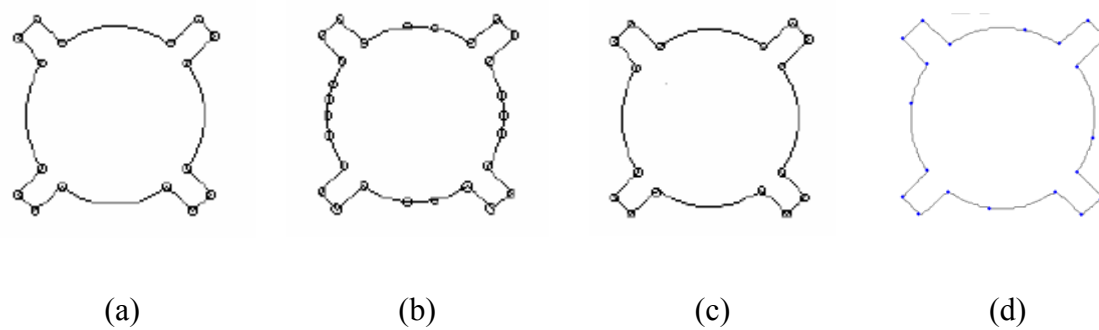
(15.2)



(15.3)



(15.4)



(15.5)

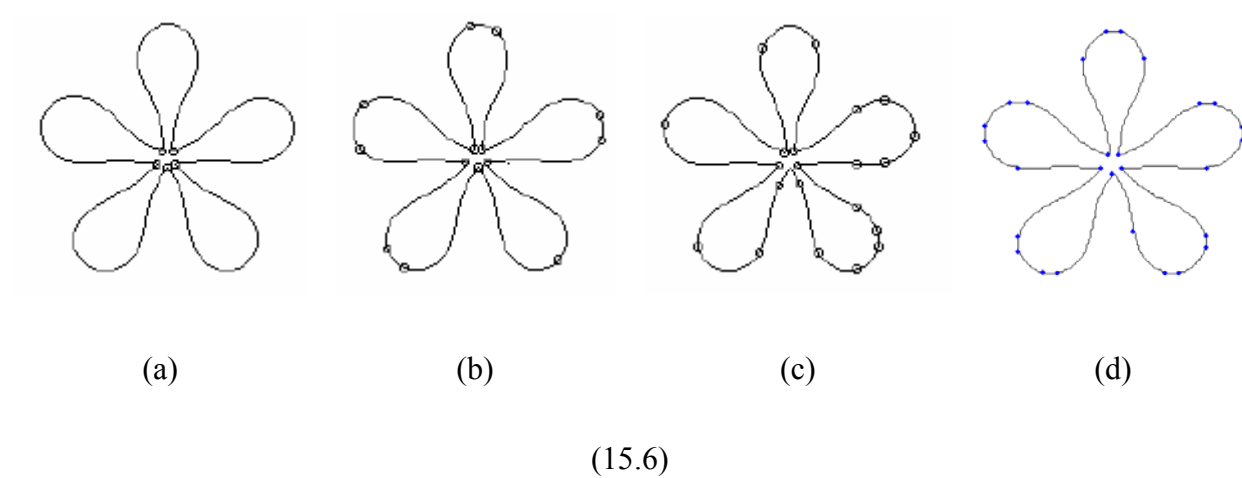


Figure 2.16 Corner points detection using (a) Guru [14] (b) Chang [77] (c) Tsai [61] and (d) SRM05

Table 2.10 Comparison of algorithm evaluation for figure 2.15.1

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
Guru	4	4	13	0
Chang	28	11	4	17
Tsai	6	6	9	0
SRM05	15	14	1	1

Table 2.11 Comparison of algorithm evaluation for figure 2.15.2

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
Guru	11	11	4	0
Chang	14	14	1	0
Tsai	11	11	4	0
SRM05	15	15	0	0

Table 2.12 Comparison of algorithm evaluation for figure 2.15.3

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
Guru	2	2	12	0
Chang	18	6	8	9
Tsai	2	2	12	0
SRM05	14	14	0	0

Table 2.13 Comparison of algorithm evaluation for figure 2.15.4

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
Guru	8	8	32	0
Chang	25	20	20	0
Tsai	24	24	16	0
SRM05	36	36	4	0

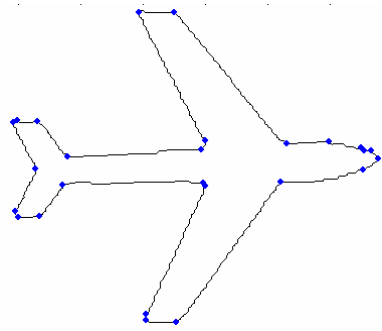
Table 2.14 Comparison of algorithm evaluation for figure 2.15.5

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
Guru	16	16	8	0
Chang	27	24	0	3
Tsai	16	16	8	0
SRM05	20	20	4	0

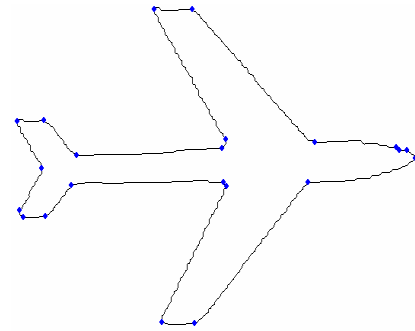
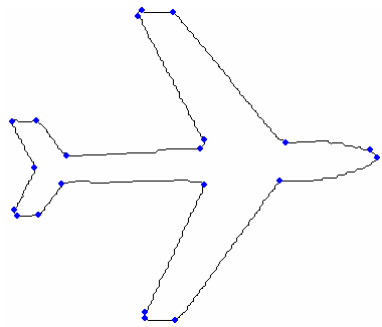
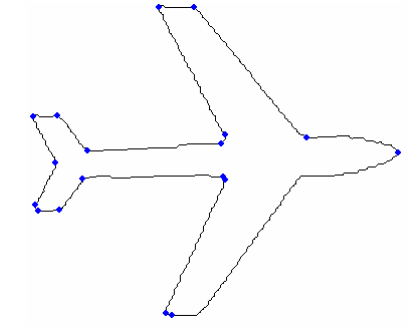
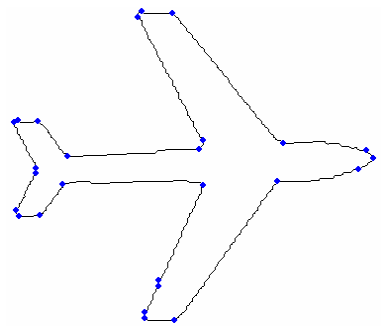
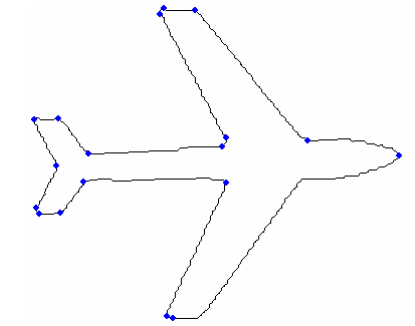
Table 2.15 Comparison of algorithm evaluation for figure 2.15.6

Algorithm	Total Detections	Correct Detections	Corners Missed	False Detections
Guru	5	5	20	0
Chang	14	9	16	5
Tsai	24	16	9	8
SRM05	28	22	3	6

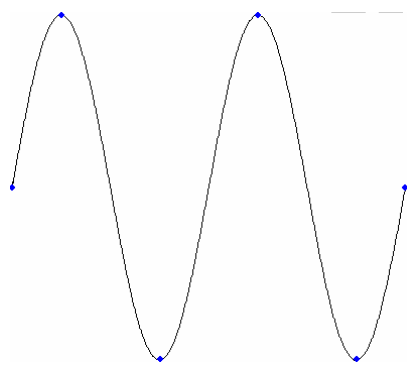
In (Figure 2.15) we examined that Chang is high on false acceptance and on the other hand Guru has high rate of false rejections. Incase of Tsai the rate of false acceptance and rejection is varying with the shape. Unlike other algorithms our approach is working in a similar fashion for all the objects with negligible false acceptance in a few cases as shown in (Tables 2.10 - 1.15). Further more since our approach is not missing any vital corner point therefore shape preservation is much better than other cases.



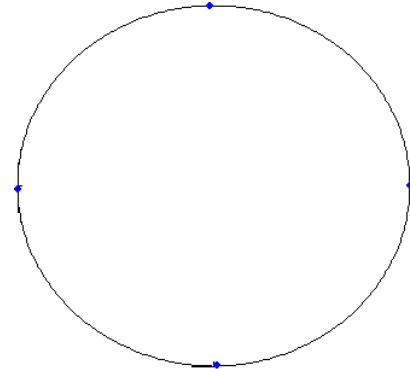
Default

 $ZCT = 7, DT = 5$ and $TA = 145^\circ$  $ZCT = 5, DT = 7$ and $TA = 145$  $ZCT = 7, DT = 5$ and $TA = 140$  $ZCT = 5, DT = 7$ and $TA = 152$  $ZCT = 5, DT = 7$ and $TA = 140$ **Figure 2.17 Testing SRM05 with different tuning parameters****Default are $ZCT = 7, DT = 5, TA = 152$.**

In (Figure 2.16) we have depicted that our approach is not much dependent upon the tuning parameters as is the case with other algorithms. Using our algorithm, one has to set these tuning parameters once in the beginning and then the set values work fine for most of the objects.



(a) $y = \sin(x)$

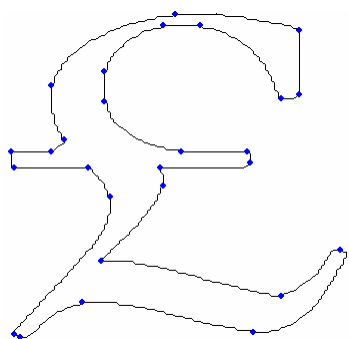


(b) $x^2 + y^2 = r^2$

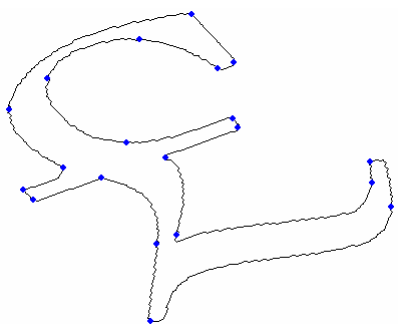
Figure 2.18 Results of SRM05 for functions using default tuning parameter values.

Default are ZCT = 7, DT = 5 and TA = 152.

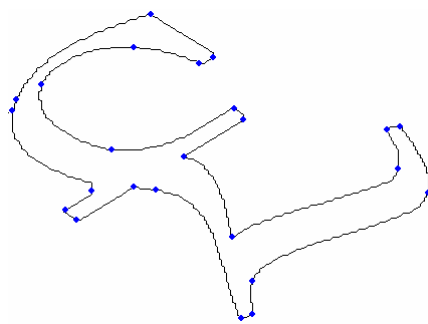
We have demonstrated in (Figure 2.17) that our approach works even better in case of smooth functions. However it is important to note that only 1st phase is enough for such mathematical functions.



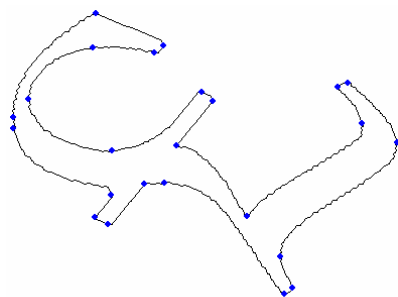
Rotation = 0



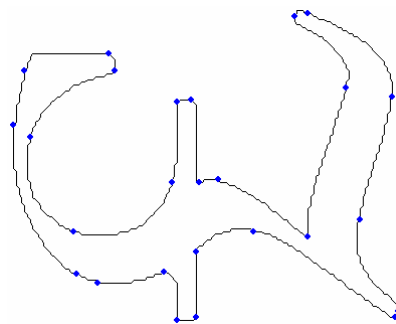
Rotation = 30



Rotation = 45



Rotation = 60



Rotation = 90

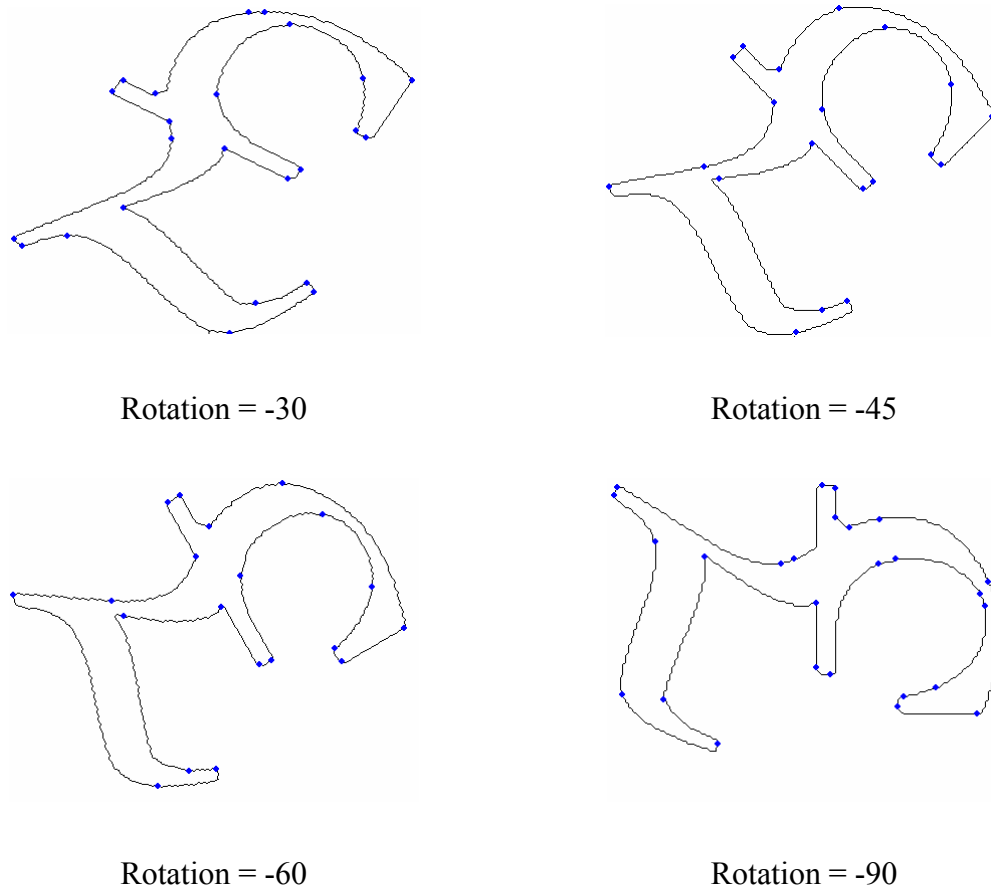


Figure 2.19 Rotation testing of SRM05 using default tuning parameter values

Default are ZCT = 7, DT = 5 and TA = 152.

We also tested our approach against rotation affects, shown in (Figure 2.18). The close inspection shows that even though there is small affect of rotation but still shape preservation is valid. We can see that the rate of false rejection and false acceptance is not increasing.

The analysis of all experiments shows that our algorithm has less false rejection and less false acceptance rate as compared to any of the algorithms presented. Moreover our approach is giving the optimal corner point set for all shapes irrespective of its nature. Further more the corners are detected at the precise positions in all the cases. The important aspect of this approach is that it is robust to noise. Incase of smoother shapes, the algorithm works fine as shown in case of smooth functions. Therefore, we can say that it is kind of a generic solution for finding corners in digital objects. Another preeminence of this approach is that although we are using some of the tuning parameters for better output but changing them does not change the result by big margins. Also we do not need to change these tuning parameters with respect to different shapes.

2.3. Conclusion

In areas like pattern recognition, image matching, motion analysis, outline capturing, reconstruction of objects etc, the corners of an object play a very vital role as of features for shape representation and analysis. In this chapter we present a novel scheme for detecting corners of a planner object. The core of the algorithm is based upon slope analysis. The complexity of our algorithm is linear. It is very efficient and as well as accurate. This scheme works for both, smooth planner curves and irregular planner curves. Test results of this algorithm are compared with some commonly referred corner detectors. The experiments show that this method leads to good quality results and robustness to noise with low computational cost.

We have demonstrated a linear time corner detection algorithm which is simple to implement, efficient and robust to noise. It is non-recursive in nature, does not depend much on tuning parameters and it is not effected greatly by small changes in tuning parameters. It is very effective for shape preservation and representation. The problem with this algorithm is that if tuning is needed then it is done manually. This work can be extended for adaptive/dynamic tuning of parameters. Further a tuning parameter independent algorithm will be the best option.

CHAPTER 3

GENERIC CUBIC SPLINE MODELING

Computer Graphics and Geometric Modeling play a very vital role in modeling and simulation of real life objects, yet there are shapes that are difficult to represent. For example; modeling of hand drawn shapes is quite a cumbersome task. Also it is required to have memory efficient object representation system. Splines are the answer to these requirements. They are taken as considerably decent and accurate way of representing, designing and manipulating the hand drawn objects. Further they also provide memory efficient solution.

The simplest way of curve fitting and object designing is to apply linear interpolation/approximation for finding intermediate data values between pairs of data points. The problem is that such attempts are extremely unlikely to provide reliable results if the data being used is anything other than broadly linear. In an attempt to deal with inherent non-linearity, the next step usually involves some sort of polynomial interpolation/approximation. This generally leads to far more stable and robust solution, but is also potentially a difficult area as the end points, local convexity and continuity of

derivatives all make their influences felt in often-contradictory ways. One of the most popular ways of dealing with these issues is to use splines. In their most general form, splines can be considered as a mathematical model that associate a continuous representation of a curve or surface with a discrete set of points in a given space. Spline fitting is an extremely popular form of piecewise approximation using various forms of polynomials of degree n , or more general functions, on an interval in which they are fitted to the function at specified points, known as control points, nodes or knots. The polynomial used can change, but the derivatives of the polynomials are required to match up to degree $n-1$ at each side of the knots, or to meet related interpolatory conditions. Boundary conditions are also imposed on the end points of the intervals. The heart of spline construction revolves around how the selected control points are effectively blended together using the polynomial function of choice.

Given the various alternative forms of spline, the question of which type of spline is most applicable in any given situation naturally arises and is inevitably a difficult one to answer without clear criteria. Arguably the most important deciding question is whether the spline is required to approximate or interpolate the control points. In other words, does the user require the curve to pass through the control points with absolute precision, or is the overall shape of the curve more important?

An interpolating function is devised to find those intermediate values which do not match with the given set of data points. This technique is suitable in cases when the data points

describing the contour of the object are sufficiently smooth and accurate with no sharp edges. Approximation is not as much restrictive as capture by Interpolation. In this case it is sufficient that the spline is made to pass close to the given set of data points. This approach is useful when the object to be approximated is not smooth [59,75].

It is necessary to take the degree of the polynomial into account before going into the details. Cubic polynomials are most often used because as compared with other polynomials, they provide reasonable smoothness, economical computation and ideal storage facility. Lower degree polynomials give little flexibility in controlling the shape of the curve. Moreover they do not possess smoothness property. Whereas higher degree polynomials require more computation and can introduce unwanted wiggles. However cubic splines have limitations like lack of freedom in shape control and object design. Due to which they are not as useful for the designer as it is the requirement in present scenario [76].

In this chapter we will discuss the proposed formulation of interpolant form and local support basis form of a generic cubic spline model. Our proposed approach, in addition to enjoying the good features of cubic splines also possesses interesting shape design features. The methodology involves two families of shape design parameters. One of them is associated with intervals and the other is associated with points. These parameters give shape control properties like interval and point tension.

We have developed an interpolatory curve scheme which involves piecewise cubic spline in its description. It is desired to extend this idea to freeform curves, which can have all the properties similar to that of B-Spline. This will help preserve the geometric smoothness of the design curve while allowing the continuity conditions on the spline functions at the knots to be varied by certain parameters, thus giving greater flexibility. This gives the designer control over the curve shape in such a way that if shape parameters are changed in an interval then the shape is changed only in the neighborhood and it does not affect the over all shape of the curve.

B-Splines are amongst the most useful and powerful tool for Computer Graphics and Geometric Modeling. They form the basis for the splines of n^{th} degree having the continuity of class C^{n-1} . The properties of B-Spline include its non-negativity of n^{th} degree spline that is nonzero only on $n+1$ intervals. They form the partition of unity, that is, the basis functions sum up to one. The curves generated by the summation of the product of control points with the basis function have some very useful properties like local convex hull property and variation diminishing property.

3.1. Interpolant Form

In this section we will generalize the idea of curve design for any given amount of data. We will formulate the piecewise generation of curve by joining the segments together with GC^2 continuity constraints. The procedure for curve design is as follows;

For parametric interpolation, let $F_i = (x_i, y_i) \in R^m$, $i \in Z$, be the given data points at distinct knots $t_i \in R$. Also suppose that $v_i > 0$ for $i=1, \dots, n$ and $\omega_i > 0$ for $i=1, \dots, n$ be the respective point and interval weights for producing tension effect in a piece. If we let $X(t)$ be the spline interpolant to the data (t_i, x_i) and $Y(t)$ be the spline interpolant to the data (t_i, y_i) , then the parametric curve $P(t) = (X(t), Y(t))$, where $t_1 \leq t \leq t_n$, is the piecewise cubic generic spline model, given as in (Equation 3.5) with subject to one of the following end conditions:

- **Type 1:** First derivative end conditions,
- **Type 2:** Natural end conditions,
- **Type 3:** Periodic end conditions.

Necessary and sufficient condition for the function $P(t)$ to be the generic spline interpolant is that its derivatives M_i satisfy,

$$c_{k-1}M_{k-1} + \left(\frac{1}{2}v_k + 2c_{k-1} + 2c_k\right)M_k + c_kM_{k+1} = b_k(y_{k+1} - y_k) + b_{k-1}(y_k - y_{k-1}) \quad (3.1)$$

For $k=1, 2, \dots, n$, where $c_i = w_i/h_i$, $b_i = 3c_i/h_i$ and h_i is the interval spacing given by (Equation 3.7). The system of Equation given in (Equation 3.1) provides $(n-2)$ equations in n unknowns, M_1, \dots, M_n . The two unknown derivative values can be calculated using anyone of the end conditions. In this thesis we have used Type 1 first derivative end

conditions. Now we can transform the set of equations into tri-diagonal system of linear system in order to calculate the unknowns. Since we are using Type 1 end conditions, therefore we end up with diagonally dominant tri-diagonal system. Not only do they have unique solution, but also they can be efficiently solved. Once the unknown derivative values are calculated the piecewise parametric cubic spline interpolant form can easily be computed. The end condition equations are given as under;

The equations for Type 1 first derivative end conditions are represented in (Equation 3.2);

$$M_1 = P'(t_1) \text{ and } M_n = P'(t_n) \quad (3.2)$$

For Type 2 natural end conditions they are represented in (Equation 3.3);

$$\begin{aligned} \left(\frac{1}{2} \nu_1 + 2c_1 \right) M_1 + c_1 M_2 &= b_1 (y_2 - y_1) \text{ and} \\ c_{n-1} M_{n-1} + \left(\frac{1}{2} \nu_n + 2c_{n-1} \right) M_n &= b_{n-1} (y_n - y_{n-1}) \end{aligned} \quad (3.3)$$

For Type 3 periodic end conditions, the equations are represented in (Equation 3.4);

$$\begin{aligned} \left(\frac{1}{2} \nu_1 + \frac{1}{2} \nu_n + 2c_1 + 2c_{n-1} \right) M_1 + c_1 M_2 + c_{n-1} M_{n-1} &= b_1 (y_2 - y_1) + b_{n-1} (y_n - y_{n-1}) \text{ and} \\ M_1 &= M_n \end{aligned} \quad (3.4)$$

The generic spline model is given as in (Equation 3.5)

$$P_i(t) = (1-\theta)^2 \{1+\theta(2-\alpha_i)\} F_i + \alpha_i (1-\theta)^2 \theta V_i + \beta_i (1-\theta) \theta^2 W_i + \theta^2 \{1+(1-\theta)(2-\beta_i)\} F_{i+1} \quad (3.5)$$

Where,

$$\theta|_{[t_i, t_{i+1})}(t) = \frac{(t-t_i)}{h_i} \quad (3.6)$$

The interval spacing between the distinct knots is given by (Equations 3.7 - 3.9);

$$h_i = t_{i+1} - t_i > 0, \quad (3.7)$$

$$\Rightarrow t = t_i + \theta h_i, \quad 0 \leq \theta \leq 1, \quad (3.8)$$

Therefore for each interval the knots can be given as,

$$t_i \leq t \leq t_i + h_i \quad (3.9)$$

Also,

$$\begin{aligned} P'_i(t)|_{\theta=0} &= (-\alpha_i) \frac{F_i}{h_i} + \alpha_i \frac{V_i}{h_i} = M_i \\ \Rightarrow V_i &= F_i + \frac{M_i h_i}{\alpha_i} \end{aligned} \quad (3.10)$$

and

$$\begin{aligned}
P'_i(t)|_{\theta=1} &= -\beta_i \frac{W_i}{h_i} + \beta_i \frac{F_{i+1}}{h_i} = M_{i+1} \\
\Rightarrow W_i &= F_{i+1} - \frac{M_{i+1}h_i}{\beta_i}
\end{aligned} \tag{3.11}$$

From (Equations 3.10) and (Equation 3.11), we can analyze that the piecewise cubic spline model hold the following interpolatory properties;

$$\begin{aligned}
P(t_i) &= F_i, & P(t_{i+1}) &= F_{i+1} \\
P'(t_i) &= M_i, & P'(t_{i+1}) &= M_{i+1}
\end{aligned} \tag{3.12}$$

Where P' denotes the first derivatives with respect to t and M_i denotes derivative value computed at the knot t_i . This eventually leads the piecewise cubic to the Hermite interpolant.

Now, applying GC^2 constraint equation at the joining points of the segments or pieces in order to achieve second order geometric continuity and for the formulation of tri-diagonal system of linear equations;

The constraints are given as in (Equation 3.13);

$$\begin{bmatrix} P(t_i^+) \\ P'(t_i^+) \\ P''(t_i^+) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{V_i}{\omega_i} & \frac{\omega_{i-1}}{\omega_i} \end{bmatrix} \begin{bmatrix} P(t_i^-) \\ P'(t_i^-) \\ P''(t_i^-) \end{bmatrix} \quad (3.13)$$

This can also be written as,

$$P_i''(t_i) = \frac{V_i}{\omega_i} P_{i-1}'(t_i) + \frac{\omega_{i-1}}{\omega_i} P_{i-1}''(t_i) \quad (3.14)$$

To satisfy the constraint, calculate the unknowns written in (Equation 3.15).

$$P_i''(t_i), \quad P_{i-1}'(t_i) \quad \text{and} \quad P_{i-1}''(t_i) \quad (3.15)$$

Since these unknowns involve first and second derivative of the cubic spline model, therefore first and second derivatives of the generic cubic spline are calculated as under.

First derivative of the generic cubic spline is given in (Equation 3.16).

$$\begin{aligned} P_i'(t) = & (1-\theta)[(1-\theta)(2-\alpha_i) - 2\{1+\theta(2-\alpha_i)\}] \frac{F_i}{h_i} + \alpha_i(1-\theta)(1-3\theta) \frac{V_i}{h_i} + \beta_i\theta(2-3\theta) \frac{W_i}{h_i} \\ & + \theta[\theta(\beta_i-2) + 2\{1+(1-\theta)(2-\beta_i)\}] \frac{F_{i+1}}{h_i} \end{aligned} \quad (3.16)$$

The second derivative is given by the (Equation 3.17).

$$\begin{aligned}
P_i''(t) = & [2\{1 + \theta(2 - \alpha_i)\} - 4(1 - \theta)(2 - \alpha_i)] \frac{F_i}{h_i^2} - \alpha_i \{3(1 - \theta) + (1 - 3\theta)\} \frac{V_i}{h_i^2} + 2\beta_i(1 - 3\theta) \frac{W_i}{h_i^2} \\
& + [4\theta(\beta_i - 2) + 2\{1 + (1 - \theta)(2 - \beta_i)\}] \frac{F_{i+1}}{h_i^2}
\end{aligned} \tag{3.17}$$

To calculate the value of $P_i''(t_i)$ put $\theta = 0$ in (Equation 3.17), we get.

$$P_i''(t_i)|_{\theta=0} = 2(2\alpha_i - 3) \frac{F_i}{h_i^2} - 4\alpha_i \frac{V_i}{h_i^2} + 2\beta_i \frac{W_i}{h_i^2} + 2(3 - \beta_i) \frac{F_{i+1}}{h_i^2} \tag{3.18}$$

Putting the values of V_i and W_i in (Equation 3.18)

$$P_i''(t_i)|_{\theta=0} = \frac{2}{h_i} \{3\Delta_i - 2M_i - M_{i+1}\} \tag{3.19}$$

For $P'_{i-1}(t_i)$ we know that,

$$P'_{i-1}(t_i) = M_i \tag{3.20}$$

To calculate the value of $P''_{i-1}(t_i)$ put $\theta = 1$ in (Equation 3.17), we get.

$$P''_{i-1}(t_i)|_{\theta=1} = 2(3 - \alpha_{i-1}) \frac{F_{i-1}}{h_{i-1}^2} + 2\alpha_{i-1} \frac{V_{i-1}}{h_{i-1}^2} - 4\beta_{i-1} \frac{W_{i-1}}{h_{i-1}^2} + 2(2\beta_{i-1} - 3) \frac{F_i}{h_{i-1}^2} \tag{3.21}$$

Putting the values of V_{i-1} and W_{i-1} in (Equation 3.21), we get.

$$P''_{i-1}(t_i)|_{\theta=1} = \frac{2}{h_{i-1}} \{-3\Delta_{i-1} + M_{i-1} + 2M_i\} \quad (3.22)$$

Substituting (Equation 3.18), (Equation 3.20) and (Equation 3.22) in (Equation 3.14), we get the tri-diagonal system of linear equations.

$$\frac{2\omega_{i-1}M_{i-1}}{\omega_i h_{i-1}} + \left(\frac{4\omega_{i-1}}{\omega_i h_{i-1}} + \frac{\nu_i}{\omega_i} + \frac{4}{h_i}\right)M_i + \frac{2M_{i+1}}{h_i} = \frac{6\Delta_i}{h_i} + \frac{6\omega_{i-1}\Delta_{i-1}}{\omega_i h_{i-1}} \quad (3.23)$$

Where,

$$\Delta_i = (F_{i+1} - F_i)/h_i \quad (3.24)$$

Multiply (Equation 3.23) by $\omega_i/2$ and then put the $c_i = \omega_i/h_i$, we get.

$$c_{i-1}M_{i-1} + \left\{\frac{\nu_i}{2} + 2c_{i-1} + 2c_i\right\}M_i + 2c_iM_{i+1} = 3c_i\Delta_i + 3c_{i-1}\Delta_{i-1} \quad (3.25)$$

In order to study the behavior of the solution with respect to tension parameters, it is convenient to write the system depicted in (Equation 3.25) in unit diagonal form. Thus dividing (Equation 3.19) by the co-efficient of M_i will give us unit diagonal form,

$$a_i M_{i-1} + M_i + c_i M_{i+1} = b_i, \quad i = 1, \dots, n-1 \quad (3.26)$$

With $M^T = [M_1, \dots, M_n]$, this system has the matrix form,

$$(I + E)M = B \quad (3.27)$$

The terms involving M_0 and M_n have been transferred to the right hand side. E is the tri-diagonal matrix with zero diagonal.

3.2. Local Support Basis Form

In this section, we will construct the B-Spline like basis for the generic cubic spline curve with the same continuity constraints as those for interpolatory spline formulation. These are the local basis functions with local support and having the property like being positive everywhere. The curve designed using these local support basis functions possesses all idea geometric properties like partition of unity, convex hull and variation diminishing. The curve design not only provides interesting shape control properties like point and interval tension but also as special case it recovers classes of cubic curves like Bézier, Ball and Timmer splines.

To construct the local support basis form, we have adopted the methodology used in [7]. Here we will transform the curve scheme representation into piecewise defined Bézier form.

Let,

$$c_i = \omega_i / h_i, \quad b_i = 3c_i / h_i \quad \text{and} \quad a_i = 1/c_i$$

Substitute the values in (Equation 3.25). Multiplying both sides by ω_i and after that replace c_i and then multiply again by $a_{i-1}a_i$.

We will get,

$$a_i M_{i-1} + (2a_i + \frac{a_i a_{i-1} v_i}{2} + 2a_{i-1}) M_i + a_{i-1} M_{i+1} = 3a_i \Delta_{i-1} + 3a_{i-1} \Delta_i \quad (3.28)$$

Let additional knots be added outside the knot partition $t_1 < t_2 < \dots < t_n$ of the interval $[t_1, t_n]$, defined by,

$$t_{-2} < t_{-1} < t_0 < t_1 \quad \text{and} \quad t_n < t_{n+1} < t_{n+2} < t_{n+3} \quad (3.29)$$

Also defining the cubic spline $\varphi_i(t)$;

$$\varphi_i(t) = \begin{cases} 0 & t \leq t_{i-2} \\ 1 & t \geq t_{i+1} \end{cases} \quad (3.30)$$

After imposing the constraints defined in (Equation 3.13) on the cubic spline defined in (Equation 3.30) we get,

At $t = t_{i-2}$,

$$\varphi_i(t_{i-2}) = 0, \quad \varphi'_i(t_{i-2}) = 0, \quad \varphi''_i(t_{i-2}) = 0 \quad (3.31)$$

Now from (Equation 3.19)

$$\varphi_i''(t_{i-2}) = P_i''(t_{i-2}) = \frac{2}{h_{i-2}} \left[\frac{3}{h_{i-2}} \{\varphi_i(t_{i-1}) - \varphi_i(t_{i-2})\} - 2\varphi_i'(t_{i-2}) - \varphi_i'(t_{i-1}) \right] = 0 \quad (3.32)$$

Which is simplified to,

$$\varphi_i(t_{i-1}) = \frac{h_{i-2}}{3} \varphi_i'(t_{i-1}) \quad (3.33)$$

At $t = t_{i+1}$,

$$\varphi_i(t_{i+1}) = 1, \quad \varphi_i'(t_{i+1}) = 0, \quad \varphi_i''(t_{i+1}) = 0 \quad (3.34)$$

Now from (Equation 3.22),

$$\varphi_i''(t_{i+1}) = P_{i+1}''(t_{i+1}) = \frac{2}{h_i} \left[-\frac{3}{h_i} \{\varphi_i(t_{i+1}) - \varphi_i(t_i)\} + 2\varphi_i'(t_{i+1}) + \varphi_i'(t_i) \right] = 0 \quad (3.35)$$

Which is simplified to,

$$\varphi_i(t_i) = 1 - \frac{h_i}{3} \varphi_i'(t_i) \quad (3.36)$$

At $t = t_{i-1}$,

From (Equation 3.28)

$$\begin{aligned}
a_{i-1}\varphi'_i(t_{i-2}) + 2(a_{i-1} + a_{i-2})\varphi'_i(t_{i-1}) + a_{i-2}\varphi'_i(t_i) = \\
3\frac{a_{i-1}}{h_{i-2}}(\varphi_i(t_{i-1}) - \varphi_i(t_{i-2})) + 3\frac{a_{i-2}}{h_{i-1}}(\varphi_i(t_i) - \varphi_i(t_{i-1}))
\end{aligned} \tag{3.37}$$

Which is simplified to,

$$2(a_{i-1} + a_{i-2})\varphi'_i(t_{i-1}) + a_{i-2}\varphi'_i(t_i) = 3\frac{a_{i-2}}{h_{i-1}}\varphi_i(t_i) + \left(\frac{3a_{i-1}}{h_{i-2}} - \frac{3a_{i-2}}{h_{i-1}}\right)\varphi_i(t_{i-1}) \tag{3.38}$$

Substituting the values of $\varphi_i(t_{i-1})$ and $\varphi_i(t_i)$, we get

$$\left(a_{i-1} + 2a_{i-2} + \frac{a_{i-2}h_{i-2}}{h_{i-1}}\right)\varphi'_i(t_{i-1}) + \left(a_{i-2} + \frac{a_{i-2}h_i}{h_{i-1}}\right)\varphi'_i(t_i) = \frac{3a_{i-2}}{h_{i-1}} \tag{3.39}$$

At $t = t_i$,

From (Equation 3.28)

$$\begin{aligned}
a_i\varphi'_i(t_{i-1}) + 2(a_i + a_{i-1})\varphi'_i(t_i) + a_{i-1}\varphi'_i(t_{i+1}) = \\
3\frac{a_i}{h_{i-1}}(\varphi_i(t_i) - \varphi_i(t_{i-1})) + 3\frac{a_{i-1}}{h_i}(\varphi_i(t_{i+1}) - \varphi_i(t_i))
\end{aligned} \tag{3.40}$$

Which is simplified to,

$$2(a_i + a_{i-1})\varphi'_i(t_i) + a_i\varphi'_i(t_{i-1}) = 3\frac{a_i}{h_{i-1}}\varphi_i(t_i) - \frac{3a_i}{h_{i-1}}\varphi_i(t_{i-1}) + \frac{3a_{i-1}}{h_i} - \frac{3a_{i-1}}{h_i}\varphi_i(t_i) \tag{3.41}$$

Substituting the values of $\varphi_i(t_{i-1})$ and $\varphi_i(t_i)$, we get

$$\left(a_i + \frac{a_i h_{i-2}}{h_{i-1}}\right) \varphi'_i(t_{i-1}) + \left(2a_i + a_{i-1} + \frac{a_i h_i}{h_{i-1}}\right) \varphi'_i(t_i) = \frac{3a_i}{h_{i-1}} \quad (3.42)$$

(Equations 3.39) and (Equation 3.42) are in terms of two unknowns $\varphi'_i(t_{i-1})$ and $\varphi'_i(t_i)$, solving them simultaneously give us,

$$\varphi'_i(t_i) = \frac{B_3 A_1 - B_1 A_3}{B_2 A_1 - B_1 A_2} \quad (3.43)$$

$$\varphi'_i(t_{i-1}) = \frac{A_3}{A_1} - \frac{A_2}{A_1} \left(\frac{B_3 A_1 - B_1 A_3}{B_2 A_1 - B_1 A_2} \right) \quad (3.44)$$

Where,

$$A_1 = a_{i-1} + 2a_{i-2} + \frac{a_{i-2} h_{i-2}}{h_{i-1}} \quad (3.45)$$

$$A_2 = a_{i-2} + \frac{a_{i-2}}{h_{i-1}} h_i \quad (3.46)$$

$$A_3 = \frac{3a_{i-2}}{h_{i-1}} \quad (3.47)$$

$$B_1 = a_i + \frac{a_i h_{i-2}}{h_{i-1}} \quad (3.48)$$

$$B_2 = 2a_i + a_{i-1} + \frac{a_i h_i}{h_{i-1}} \quad (3.49)$$

$$B_3 = \frac{3a_i}{h_{i-1}} \quad (3.50)$$

Substituting the values of $\varphi'_i(t_{i-1})$ and $\varphi'_i(t_i)$ in (Equations 3.33) and (Equation 3.36), we get

$$\varphi_i(t_{i-1}) = \frac{h_{i-2}}{3} \left\{ \frac{A_3}{A_1} - \frac{A_2}{A_1} \left(\frac{B_3 A_1 - B_1 A_3}{B_2 A_1 - B_1 A_2} \right) \right\} \quad (3.51)$$

$$\varphi_i(t_i) = 1 - \frac{h_i}{3} \left(\frac{B_3 A_1 - B_1 A_3}{B_2 A_1 - B_1 A_2} \right) \quad (3.52)$$

Substituting the values of variables $A_1 - A_3$ and $B_1 - B_3$, and simplifying we get,

$$\varphi_i(t_{i-1}) = \frac{a_{i-2}h_{i-2}d_i}{D_i} = \mu_{i-1} \quad (3.53)$$

$$1 - \varphi_i(t_i) = \frac{a_i h_i d_{i-1}}{D_i} = \lambda_i \quad (3.54)$$

$$\varphi'_i(t_{i-1}) = \frac{3a_{i-2}d_i}{D_i} = \hat{\mu}_{i-1} \quad (3.55)$$

$$\varphi'_i(t_i) = \frac{3a_i d_{i-1}}{D_i} = \hat{\lambda}_i \quad (3.56)$$

Similarly defining,

$$\varphi_{i+1}(t) = \begin{cases} 0 & t \leq t_{i-1} \\ 1 & t \geq t_{i+2} \end{cases} \quad (3.57)$$

After imposing the constraints defined in (Equation 3.13), we get,

$$\varphi_{i+1}(t_i) = \frac{a_{i-1}h_{i-1}d_{i+1}}{D_{i+1}} = \mu_i \quad (3.58)$$

$$1 - \varphi_{i+1}(t_{i+1}) = \frac{a_{i+1}h_{i+1}d_i}{D_{i+1}} = \lambda_{i+1} \quad (3.59)$$

$$\varphi'_{i+1}(t_i) = \frac{3a_{i+1}d_{i+1}}{D_{i+1}} = \hat{\mu}_i \quad (3.60)$$

$$\varphi'_{i+1}(t_{i+1}) = \frac{3a_{i+1}d_i}{D_{i+1}} = \hat{\lambda}_{i+1} \quad (3.61)$$

We can analyze a relation here,

$$\hat{\mu}_{i-1} = \frac{2}{h_{i-2}} \mu_{i-1} \quad (3.62)$$

$$\hat{\lambda}_i = \frac{3}{h_i} \lambda_i \quad (3.63)$$

Where,

$$d_i = \frac{1}{2} a_i a_{i-1} \nu_i + a_i + a_{i-1} \quad (3.64)$$

And

$$D_i = h_{i-1}d_{i-1}d_i + a_i(h_{i-1} + h_i)d_{i-1} + a_{i-2}(h_{i-2} + h_{i-1})d_i \quad (3.65)$$

Now for local support basis, define

$$B_i(t) = \varphi_i(t) - \varphi_{i+1}(t) \quad (3.66)$$

B_i has the local support (t_{i-2}, t_{i+2}) and an explicit representation of B_j on any interval (t_i, t_{i+1}) , $i = j-2, j-1, j, j+1$.

To calculate the local support basis formulation, we have,

$$\begin{aligned} B_j(t) = & (1-\theta)^2 \{1 + \theta(2 - \alpha_i)\} B_j(t_i) + \theta(1-\theta)^2 \{\alpha_i B_j(t_i) + B'_j(t_i)h_i\} + \\ & \theta^2(1-\theta) \{\beta_i B_j(t_{i+1}) - B'_j(t_{i+1})h_i\} + \theta^2 \{1 + (1-\theta)(2 - \beta_i)\} B_j(t_{i+1}) \end{aligned} \quad (3.67)$$

Where from (Equation 3.66)

$$B_j(t_i) = B'_j(t_i) = 0, \text{ for } i \neq j-1, j, j+1 \quad (3.68)$$

$$B_j(t_{j-1}) = \mu_{j-1}; \quad B'_j(t_{j-1}) = \hat{\mu}_{j-1} \quad (3.69)$$

$$B_j(t_j) = 1 - \lambda_i - \mu_i; \quad B'_j(t_j) = \hat{\lambda}_j - \hat{\mu}_j \quad (3.70)$$

$$B_j(t_{j+1}) = \lambda_{j+1}; \quad B'_j(t_{j+1}) = -\hat{\lambda}_{j+1} \quad (3.71)$$

3.2.1. Curve Design

Now that we have developed local support basis functions for freeform generic spline formulation, it is desired to devise a convenient methodology to compute the curve representation.

$$P(t) = \sum_{j=i-1}^{i+2} B_j(t) P_j, \quad t \in [t_i, t_{i+1}), \quad i = 0, \dots, n-1 \quad (3.72)$$

Using (Equation 3.67) with $j = i-1, i, i+1, i+2$

For $j = i-1$

$$B_{i-1}(t) = (1-\theta)^2 \{1 + \theta(2-\alpha_i)\} \lambda_i + \theta(1-\theta)^2 (\alpha_i \lambda_i - \hat{\lambda}_i h_i) \quad (3.73)$$

For $j = i$

$$B_i(t) = (1-\theta)^2 \{1 + \theta(2-\alpha_i)\} (1-\lambda_i - \mu_i) + \theta(1-\theta)^2 \{ \alpha_i (1-\lambda_i - \mu_i) + (\hat{\lambda}_i - \hat{\mu}_i) h_i + \theta^2 (1-\theta) (\beta_i \lambda_{i+1} + \hat{\lambda}_{i+1} h_i) + \theta^2 \{1 + (1-\theta)(2-\beta_i)\} \lambda_{i+1} \} \quad (3.74)$$

For $j = i + 1$

$$B_{i+1}(t) = (1-\theta)^2 \{1 + \theta(2-\alpha_i)\} \mu_i + \theta(1-\theta)^2 (\alpha_i \mu_i + \hat{\mu}_i h_i) + \theta^2 (1-\theta) \{\beta_i(1-\lambda_{i+1} - \mu_{i+1}) - (\hat{\lambda}_{i+1} - \hat{\mu}_{i+1}) h_i\} + \theta^2 \{1 + (1-\theta)(2-\beta_i)\} (1-\lambda_{i+1} - \mu_{i+1}) \quad (3.75)$$

For $j = i + 2$

$$B_{i+2}(t) = \theta^2 (1-\theta) (\beta_i \mu_{i+1} - \hat{\mu}_{i+1} h_i) + \theta^2 \{1 + (1-\theta)(2-\beta_i)\} \mu_{i+1} \quad (3.76)$$

To prove the partition of unity, add these four basis functions defined in (Equations 3.72-3.75), we get,

$$B_{i-1}(t) + B_i(t) + B_{i+1}(t) + B_{i+2}(t) = 1 \quad (3.77)$$

3.2.2. Curve Representation

By local support property,

$$P(t) = \sum_{j=i-1}^{i+2} B_j(t) P_j, \quad t \in [t_i, t_{i+1}), \quad i = 0, 1, \dots, n-1 \quad (3.78)$$

Where, $P_j \in R^N$, $j = 0, 1, \dots, n+1$ define the control points of the representation.

We can write (Equation 3.78) as under;

$$\Rightarrow P(t) = B_{i-1}(t)P_{i-1} + B_i(t)P_i + B_{i+1}(t)P_{i+1} + B_{i+2}(t)P_{i+2} \quad (3.79)$$

Substituting the values of basis functions, we get the transformation to Bézier form. This form is very convenient for computational purposes.

$$P_i(t) = (1-\theta)^2 \{1 + \theta(2 - \alpha_i)\} F_i + \alpha_i (1-\theta)^2 \theta V_i + \beta_i \theta^2 (1-\theta) W_i + \theta^2 \{1 + (1-\theta)(2 - \beta_i)\} F_{i+1} \quad (3.80)$$

where,

$$F_i = \lambda_i P_{i-1} + (1 - \lambda_i - \mu_i) P_i + \mu_i P_{i+1} \quad (3.81)$$

$$F_{i+1} = \lambda_{i+1} P_i + (1 - \lambda_{i+1} - \mu_{i+1}) P_{i+1} + \mu_{i+1} P_{i+2} \quad (3.82)$$

$$V_i = (\lambda_i - \frac{\hat{\lambda}_i}{\alpha_i} h_i) P_{i-1} + \{(1 - \lambda_i - \mu_i) + \frac{(\hat{\lambda}_i - \hat{\mu}_i)}{\alpha_i} h_i\} P_i + (\mu_i + \frac{\hat{\mu}_i}{\alpha_i} h_i) P_{i+1} \quad (3.83)$$

and

$$W_i = (\lambda_{i+1} + \frac{\hat{\lambda}_{i+1}}{\beta_i} h_i) P_i + \{(1 - \lambda_{i+1} - \mu_{i+1}) - \frac{(\hat{\lambda}_{i+1} - \hat{\mu}_{i+1})}{\beta_i} h_i\} P_{i+1} + (\mu_{i+1} - \frac{\hat{\mu}_{i+1}}{\beta_i} h_i) P_{i+2} \quad (3.84)$$

3.3. Extension to Special Class: Timmer Parametric Cubic Spline

Timmer Parametric cubic curve was proposed by Harry G Timmer of McDonnell Douglas [90]. This curve was modeled after the Bézier curve. The difference is that it follows the control polygon in more restrictive way. Timmer achieved this by forcing the Parametric cubic to pass through the two control points and also through the mid point of the line joining two intermediate points $P_{i,1}$ and $P_{i,2}$.

Even though this curve technique is a well accepted one in the field of computer graphics but the designers and practitioners did not opt for it. The rationale was its property of not satisfying the convex hull.

3.3.1. Introduction to Timmer Parametric Cubic

The blending functions of Timmer Parametric cubic are;

$$f_0(t) = (1 - 2t)(1 - t)^2 \quad (3.85)$$

$$f_1(t) = 4t(1 - t)^2 \quad (3.86)$$

$$f_2(t) = 4t^2(1 - t) \quad (3.87)$$

$$f_4(t) = (2t - 1)t^2$$

(3.88)

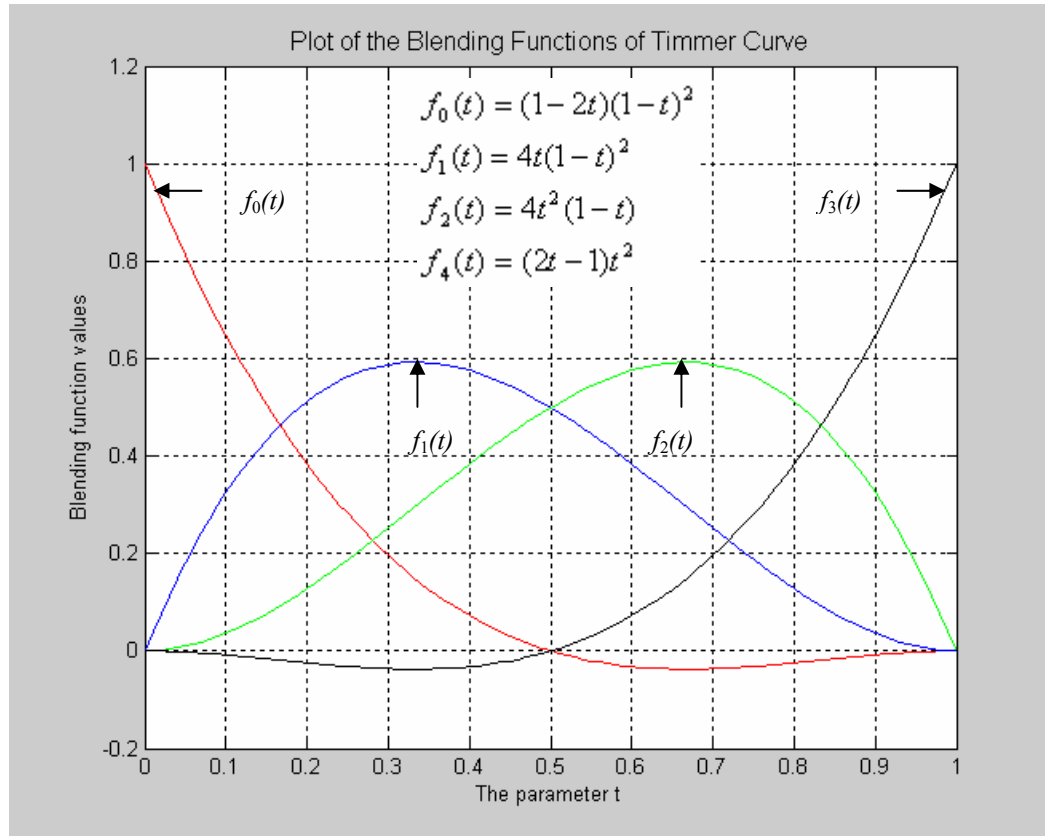


Figure 3.1 Blending function of Timmer curve

We can note that some part of f_0 and f_3 are negative, which shows that it does not follow the convex hull property.

So, the parametric form of Timmer curve is;

$$\begin{aligned}
H_i(t) &= (x(t), y(t)) \\
H_i(t) &= (1-2t)(1-t)^2 P_i + 4t(1-t)^2 P_{i,1} + 4t^2(1-t)P_{i,2} + (2t-1)t^2 P_{i+1}
\end{aligned} \tag{3.89}$$

Where as,

$$\begin{aligned}
P_{i,1}(x_{i,1}, y_{i,1}) &= P_i + \frac{1}{\alpha_i} T_i \\
&= (x_i + \frac{1}{\alpha_i} M_i, y_i + \frac{1}{\alpha_i} N_i)
\end{aligned} \tag{3.90}$$

$$\begin{aligned}
P_{i,2}(x_{i,2}, y_{i,2}) &= P_{i+1} + \frac{1}{\beta_i} T_{i+1} \\
&= (x_{i+1} - \frac{1}{\beta_i} M_{i+1}, y_{i+1} - \frac{1}{\beta_i} N_{i+1})
\end{aligned} \tag{3.91}$$

P_i and P_{i+1} are the two control points for i^{th} piece. $P_{i,1}$ and $P_{i,2}$ are two intermediate points that are calculated in order to render a piece. $T_i(M_i, N_i)$ and $T_{i+1}(M_{i+1}, N_{i+1})$ are unit tangent vectors at two control points respectively. α_i and β_i are real numbers, which are used as shape parameters.

Now, from (Equation 3.90) and (Equation 3.91), we can write (Equation 3.89) in its co-ordinate form. So, Timmer parametric cubic is represented as follows,

$$\begin{aligned}
H_i(t) = & (1-2t)(1-t)^2 P_i + 4t(1-t)^2 t \left(\frac{T_i}{\alpha_i} + P_i \right) + t^2 (2t-1) P_{i+1} \\
& + 4(1-t)t^2 \left(\frac{T_{i+1}}{\beta_i} + P_{i+1} \right)
\end{aligned} \tag{3.92}$$

3.3.2. Properties of Timmer Parametric Cubic

Following are the properties of Timmer curve.

- **Coordinate System Independent**

A coordinate system independent curve remains same even if the coordinates are changed. In order to follow this property the polynomial bases must identically sum to one;

$$\sum_{i=0}^3 f_i(t) \equiv 1 \tag{3.93}$$

This property can be proved by using (Equations 3.85) to (Equation 3.88).

- **Convex Hull Property**

The convex hull is a bounding polygon around all control points in such a way that the line joining any two of the control points remain inside the polygon. A curve is said to

fulfill this property if it is coordinate system independent and all the polynomial bases are non-negative. i.e.

$$\sum_{i=0}^3 f_i(t) \equiv 1, \text{ and } f_i(t) \geq 0, \quad 0 \leq t \leq 1 \quad (3.94)$$

Timmer parametric cubic does not follow this property as already shown in (Figure 3.1).

Also it can be seen from (Figure 3.2).

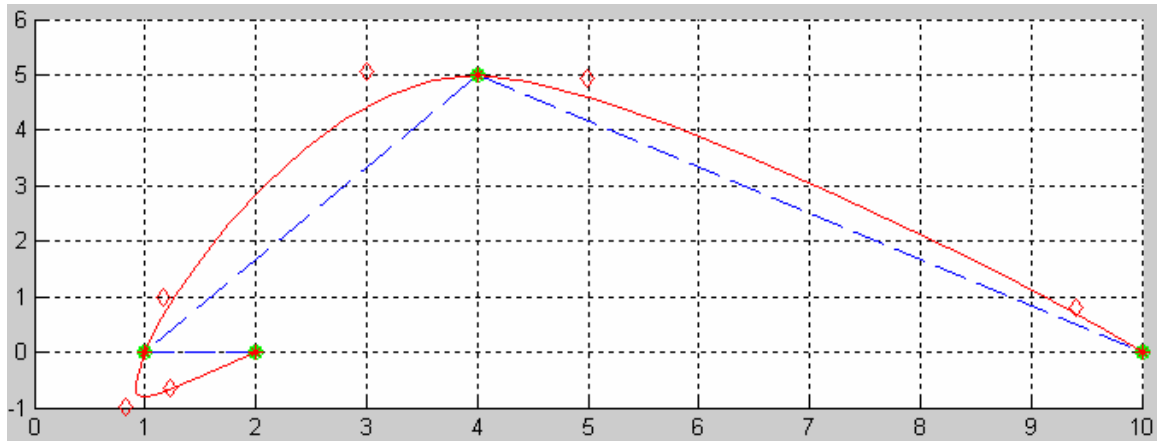


Figure 3.2 Timmer curve disobeying convex hull property

▪ Variation Diminishing Property

If a given straight line, lying in the same plane space as that of the curve, intersects the curve in ' c ' number of points and the control polygon in ' p ' number of points then,

$$c = p - 2j, \quad j \geq 0 \quad (3.95)$$

Considering (Figure 3.3), it is proved that Timmer parametric cubic does not obey this property as well. Here we can see that $c = 2$ and $p = 0$ and thus $j = -2$, which is negating the property.

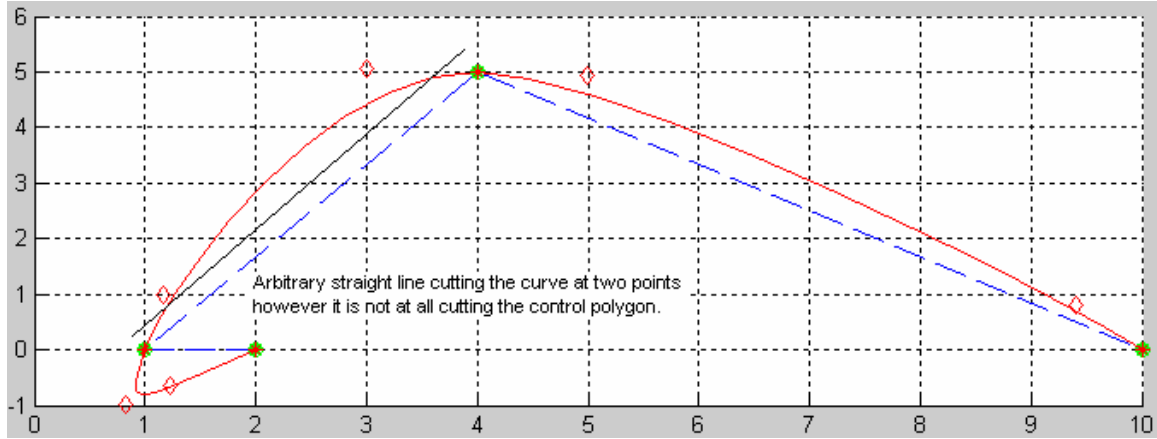


Figure 3.3 Timmer parametric cubic disobey VDP

▪ Symmetry

This property defines that the curve retains the shape even if the points are ordered in reverse order, thus

$$\sum_{i=0}^3 f_i(t)P_i \equiv \sum_{i=0}^3 f_i(1-t)P_{3-i} \quad (3.96)$$

Therefore Timmer follows this property.

▪ Invariant Form under Affine Transformation

The two ways for affine transformation are,

1. Transform sampled data points on the curve directly or,
2. Transform only the control points and use the transformed control points to generate new Timmer curve.

For an affine transformation to satisfy, we can write the form,

This can be proved for Timmer as follows,

$$M : X' = AX + b \quad (3.97)$$

$$M(H(t)) = \sum_{i=0}^3 f_i(t) M(P_i) \quad (3.98)$$

$$M(H(t)) = AH(t) + b \quad (3.99)$$

$$= \sum_{i=0}^3 f_i(t) AP_i + \sum_{i=0}^3 f_i(t) b \quad (3.100)$$

$$= \sum_{i=0}^3 f_i(t) (AP_i + b) \quad (3.101)$$

$$= \sum_{i=0}^3 f_i(t) M(P_i) \quad (3.102)$$

Thus, Timmer is affine transformation invariant.

▪ Endpoint Interpolation

As in the case of Bézier, Timmer also interpolates the end points of a piece. It meets the following conditions;

$$f_0(0) = 1, \quad f_i(0) = 0, \quad i = 1, 2, 3 \quad (3.103)$$

$$f_3(1) = 1, \quad f_i(1) = 0, \quad i = 1, 2, 3 \quad (3.104)$$

▪ Extra Interpolation

Timmer interpolates the point at $t = 0.5$, which is actually the mid point of the line segment joining $P_{i,1}$ and $P_{i,2}$.

$$f_1(0.5) = 0.5, \quad f_2(0.5) = 0.5, \quad f_i(0.5) = 0, \quad i = 1, 3 \quad (3.105)$$

3.3.3. Designing GC^2 Continuous Piecewise Timmer Curve using Iterative Scheme

This scheme was proposed by [6], in which, a GC^2 piecewise Timmer curve is obtained. Data set needed is a set of points in Cartesian coordinate where the points are in the form of $P_i(x_i, y_i)$ for $i = 0, 1, \dots, n$. When there are a small number of points given as interpolating points, one may easily form G^2 data for those points in order to render a Timmer Curve with G^2 continuity. However, when there are large numbers of points, then forming G^2 data for the given data points become troublesome. This scheme proposes a solution to this problem. In this case the curve passes through the points matching the unit tangent vectors and signed curvature at respective point.

The tangent vectors are calculated as follows [29],

$$T_0 = 2(P_1 - P_0) - \frac{(P_2 - P_0)}{2} \quad (3.106)$$

$$T_n = 2(P_n - P_{n-1}) - \frac{(P_n - P_{n-2})}{2} \quad (3.107)$$

$$T_i = a_i(P_i - P_{i-1}) + (1 - a_i)(P_{i+1} - P_i) \quad (3.108)$$

where,

$$a_i = \frac{\|P_{i+1} - P_i\|}{\|P_{i+1} - P_i\| + \|P_i - P_{i-1}\|} \quad (3.109)$$

Here (Equation 3.106) represents tangent vector at the first point, (Equation 3.107) represents the tangent vector at the last point and the rest of intermediate tangent vectors are calculated using (Equation 3.108).

The unit tangent vectors are used to fix the direction of travel of a curve. Therefore the unit tangent vectors at each interpolating points are formulated as follows,

$$T_i = (M_i, N_i) = \frac{D_i}{\|D_i\|}, \quad i = 1, 2, \dots, n-1 \quad (3.110)$$

The Timmer curve is defined as in (Equation 3.89), however the intermediate control points are calculated as follows;

$$P_{i,1}(x_{i,1}, y_{i,1}) = (x_i + \frac{M_i}{\alpha_i}, y_i + \frac{N_i}{\alpha_i}) \quad (3.111)$$

$$P_{i,2}(x_{i,2}, y_{i,2}) = (x_{i+1} - \frac{M_{i+1}}{\alpha_i}, y_{i+1} - \frac{N_{i+1}}{\alpha_i}) \quad (3.112)$$

Now, substituting (Equation 3.111) into (Equation 3.89). Each Timmer curve is dependent upon a single variable, α_i .

$$H_i(t) = \frac{P_i \alpha_i + t\{4(t-1)[T_i(t-1) + T_{i+1}t] + t\alpha_i(2t-3)(P_i - P_{i+1})\}}{\alpha_i} \quad (3.113)$$

The first and second derivatives of $H_i(t)$ are;

$$H'_i(t) = \frac{4[T_i - 2(2T_i + T_{i+1})t + 3(T_i + T_{i+1})t^2] + 6t(t-1)(P_i - P_{i+1})\alpha_i}{\alpha_i} \quad (3.114)$$

$$H''_i(t) = \frac{-8(2T_i + T_{i+1}) + 24(T_i + T_{i+1})t + 6\alpha_i(2t-1)(P_i - P_{i+1})}{\alpha_i} \quad (3.115)$$

The formula for signed curvature is defined as follows,

$$K_i(t) = \frac{Hx'(t)Hy''(t) - Hx''(t)Hy'(t)}{\left(\sqrt{\{Hx'(t)\}^2 + H\{y'(t)\}^2}\right)^3} \quad (3.116)$$

At $t=0$;

$$K_i(0) = \frac{\alpha_i}{8} \{4M_{i+1}N_i - 4M_iN_{i+1} + 3[N_i(x_i - x_{i+1}) + M_i(y_{i+1} - y_i)]\alpha_i\} \quad (3.117)$$

And at $t = 1$;

$$K_i(1) = \frac{\alpha_i}{8} \{N_{i+1}[3\alpha_i(x_{i+1} - x_i) - 4M_i] + M_{i+1}[4N_i + 3(y_i - y_{i+1})\alpha_i]\} \quad (3.118)$$

An initial positive value of α_0 is required for this scheme to work. This α_0 is required to calculate the value of $K_0(1)$, by using the following constraint;

$$K_i(0) = K_{i-1}(1), \quad i = 1, \dots, n \quad (3.119)$$

By utilizing the (Equation 3.117) and (Equation 3.118), an equation is achieved in quadratic form.

$$a\alpha_i^2 + b\alpha_i + c = 0 \quad (3.120)$$

where,

$$\begin{aligned} a &= \frac{3}{8} [N_i(x_i - x_{i+1}) + M_i(y_{i+1} - y_i)] \\ b &= \frac{1}{2} (M_{i+1}N_i - M_iN_{i+1}) \quad \text{and} \\ c &= -K_{i-1}(1) \end{aligned} \quad (3.121)$$

The general solution for α_i is stated as follows;

$$\alpha_i = \frac{2[(M_i N_{i+1} - M_{i+1} N_i)]}{3\{N_i(x_i - x_{i+1}) + M_i(y_{i+1} - y_i)\}} \pm \frac{\sqrt{(M_i N_{i+1} - M_{i+1} N_i)^2 - 6\{N_i(x_i - x_{i+1}) + M_i(y_{i+1} - y_i)\}K_{i-1}(1)}}{3\{N_i(x_i - x_{i+1}) + M_i(y_{i+1} - y_i)\}} \quad (3.122)$$

If there exists a real positive solution for α_i , then it is selected to render Timmer curve.

The general algorithm to generate a G^2 piecewise Timmer curve is shown in (Algorithm 3.1):

Algorithm 3.1

Step1: For $i = 0, to \quad n$ do

Step1.1: Define $P_i(x_i, y_i)$

Step1.2: Calculate $T_i(M_i, N_i)$ using (Equation 3.110)

Step2: Define α_0

Step2.1: Calculate $K_0(1)$ using (Equation 3.118)

Step2.2: Calculate $P_{0,1}$ and $P_{0,2}$ using (Equation 3.112)

Step2.3: Render the curve $H_0(t)$, with $0 \leq t \leq 1$ where $H_0(t)$ is given by (Equation 3.89)

Step3: For $i = 0, to \quad n-1$ do

Step3.1: $K_i(0) = K_{i-1}(1)$ as defined in (Equation 3.119)

Step3.2: Solve (Equation 3.122) to obtain α_i

Step3.3: Select α_i with real positive number

Step3.4: Calculate $P_{i,1}$ and $P_{i,2}$ using (Equation 3.112)

Step3.5: Render the curve $H_i(t)$, with $0 \leq t \leq 1$ where $H_i(t)$ is given by (Equation 3.89)

3.3.4. Shape Control for Timmer Curves

A very useful and fascinating feature of cubic curve is the introduction of local control of a shape. This helps a user to define a specific shape and then allows him to play with the shape of the curve without changing the data set of control points. Related research and mathematical formulation of local control can be found in [7,87,88]. We have also developed the shape control for parametric cubic piecewise GC^1 and GC^2 continuous Timmer curves.

▪ Piecewise Timmer Curve with Shape control and GC^1 Continuity

The data set for GC^1 continuity is, set of points and shape parameters α and β for each piece. In this kind of interpolation, the curve passes through each point matching the unit tangent vectors.

In order to render a GC^1 curve, we will use (Equation 3.92). The steps required to program this scheme are;

Algorithm 3.2

Step1: Define all data points P_i , $i = 1, to \ n$

Step1.1: Define all shape parameters for each piece. α_i and β_i , $i = 1, to \ n-1$.

Step1.2: Calculate Tangent vectors for first and last point, using (Equation 3.106) and (Equation 3.107)

Step2: Calculate all intermediate Tangent vectors using (Equation 3.108)

Step2.1: Calculate the unit components of each vector.

Step3: Render the curve piece by piece using (Equation 3.92)

▪ Piecewise Timmer Curve with Shape control and GC^2 Continuity

In this scheme we are given a set of data points and the shape parameters α and β . To develop GC^2 continuity, we not only force the curve to pass through the control points but also the second derivative matches at the points that are joining two different pieces.

Here the condition to be satisfied is;

$$H_{i-1}''(1) = H_i''(0) \quad (3.123)$$

Taking 1st and 2nd derivative of (Equation 3.92), we will get (Equation 3.124) and (Equation 3.125) respectively.

$$H'_i(t) = \frac{2[2\alpha_i T_{i+1}(2-3t)t + \beta_i(t-1)\{T_i(6t-2) + 3\alpha_i t(P_i - P_{i+1})\}]}{\alpha_i \beta_i} \quad (3.124)$$

$$H''_i(t) = \frac{2[4\alpha_i T_{i+1}(1-3t) + \beta_i\{4T_i(3t-2) + 3\alpha_i(2t-1)(P_i - P_{i+1})\}]}{\alpha_i \beta_i} \quad (3.125)$$

Constrain given by (Equation 3.123) will give us the tri-diagonal system of $n-2$ linear equations for n unknown Tangent vectors.

$$w_i T_{i-1} + x_i T_i + y_i T_{i+1} = z_i, \quad i = 2, \dots, n-1 \quad (3.126)$$

where,

$$w_i = 4\alpha_i \beta_i \beta_{i-1} \quad (3.127)$$

$$x_i = 8\alpha_{i-1} \beta_i (\alpha_i + \beta_{i-1}) \quad (3.128)$$

$$y_i = 4\alpha_i \alpha_{i-1} \beta_{i-1} \quad (3.129)$$

$$z_i = -3\alpha_i \beta_i \alpha_{i-1} \beta_{i-1} (P_{i-1} - P_{i+1}) \quad (3.130)$$

Now we need two more equations to solve for n unknown Tangent vectors. These two equations are derived using the Type 1 first derivative end conditions.

$$m_1 = f'(t_1) \text{ and } m_n = f'(t_n) \quad (3.131)$$

This manipulation will provide us with a system in following form,

$$AT = B \quad (3.132)$$

Where, A is the tri-diagonal matrix of co-efficient. T is the unknown matrix of Tangent vectors and B is the constants matrix. After applying the end conditions we can transfer the terms involving the end conditions m_1 and m_n to the right hand side. Finally we can find the Tangent vectors by,

$$T = A^{-1}B \quad (3.133)$$

Once we have got all the data points, their respective unit tangents and shape parameters, we can easily compute the Timmer curve in a piece by piece fashion using (Equation 3.92).

The steps required to program this scheme are;

Algorithm 3.3

Step1: Define all data points P_i , $i = 1, to \ n$

Step1: Define all shape parameters for each piece. α_i and β_i , $i = 1, to \ n-1$.

Step2: Construct the tri-diagonal system of linear Equations using (Equation 3.126)

Step2: Calculate the Tangent vectors using (Equation 3.133)

Step3: Render the curve piece by piece using (Equation 3.130)

3.4. Results and Analysis

The tension behavior, including interval tension, point tension and global tension of the generic cubic spline model is tested for the interpolant and local support basis form. We have tested the effects for the range of values of data set in \mathbb{R}^2 . The default values of shape parameters ν_i will be assumed as zero $\forall i$ and parameters ω_i as 1 $\forall i$. The default values of shape parameters give us C^2 continuous curve design. The ν_i are termed point tension factors because they tighten a parametric curve at the i th point. The ω_i are termed interval weights because they tighten the curve on the i th interval. Further more it is important to note that parameters α_i and β_i define the class of spline as special case. For example, $\alpha_i = \beta_i = 2 \ \forall i$, define cubic Ball curves, $\alpha_i = \beta_i = 3 \ \forall i$, define cubic Bézier curves and $\alpha_i = \beta_i = 4 \ \forall i$, define cubic Timmer curves. By default we have demonstrated our results for cubic Bézier curves. As special case we have demonstrated the results and shown the effect of shape parameters for cubic Timmer curves.

In (Figures 3.4-3.10), we have demonstrated the shape control for the interpolant form of generic cubic spline model. We can observe that, the interpolant form follows the control polygon in a very restrictive manner. We see that positive increase in global values of point and interval tension parameters do not produce any affect at all as shown in (Figure 3.5 and 3.6). Where as, the progressive change in negative global values of shape parameters produce considerable effect on the shape of the curve. It is observed from (Figures 3.7 – 3.10) that for lower values, the curve tends to bulge inside but as we further decrease these values, the curve tends towards the control polygon. The inside bulging is due to the violation of geometric properties.

The freeform curve design is much more effective and a user can play with the shape parameters in greater control as compared to the interpolant form. This is depicted in (Figures 3.11- 3.37). In (Figures 3.13- 3.16) we can observe the impact of progressive increase in global values of point tension parameters. As we keep on increasing the values, the curve tends to follow the control polygon. Moreover we also observe the effect of point tension at specified points in (Figures 3.21 – 3.24). We observe that the increment in point tension parameters leads to cusp, which is in fact the condition of C^0 or G^0 parametric continuity. Moreover it is also observed that the change in point tension parameters exert influence of some degree on the adjacent curve pieces as well. (Figures 3.17 – 3.20) show the influence of change in parameters for interval tension at the base. We have demonstrated that increase in values produces the effect of control polygon. Finally we show the effect of negative values for global values of point tension parameters in (Figures 3.21 – 3.37). We get very interesting shapes as we keep on

decreasing the values and the further decrement result in remapping of the curve to the control polygon as in case of global values of point tension parameters

In case of cubic Timmer parametric spline's interpolant form, we can notice that when interval tension is applied at a certain piece the intermediate points tend to overlap the control points and due to which the curve is stretched and consequently it is then appeared as straight line for that specific piece. Looking at (Equations 3.111 and 3.112), one can notice that as we increase the values of α_i and β_i the tangent vector approaches zero and therefore making the intermediate point equal to the control point. Since the effect of tangent is nullified that's why points are joined as straight line.

3.4.1 Interpolant Form

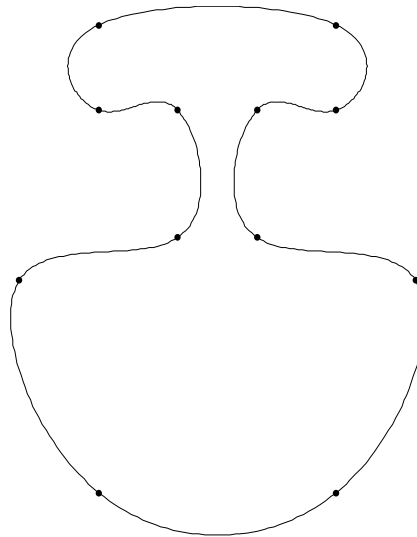


Figure 3.4 Default values of shape parameters. $\omega_i = 1$ and $v_i = 0, \forall i$

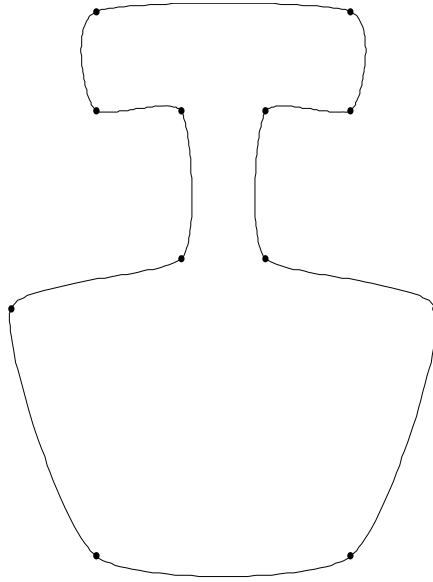


Figure 3.5 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = 10, \forall i$

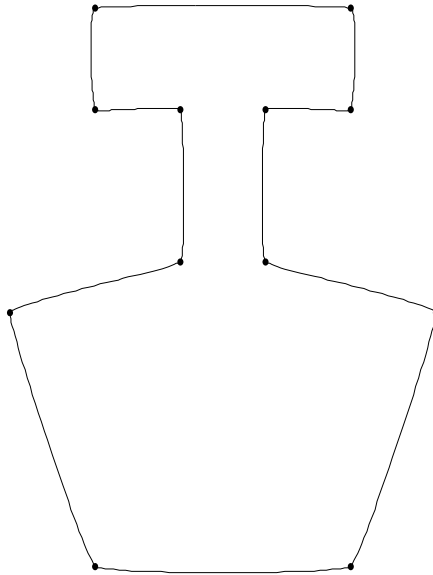


Figure 3.6 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = 50, \forall i$

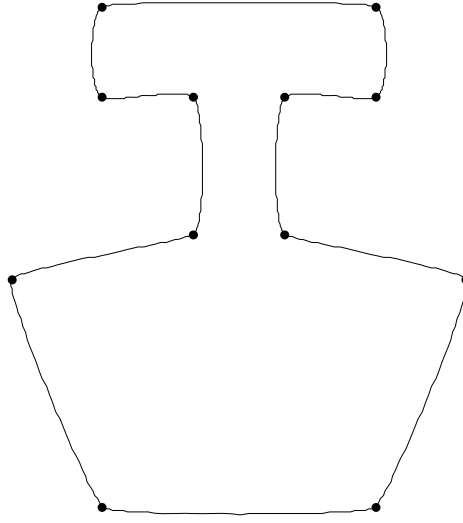


Figure 3.7 Interval values of shape parameter. $\omega_i = 100$ and $\nu_i = 0, \forall i$

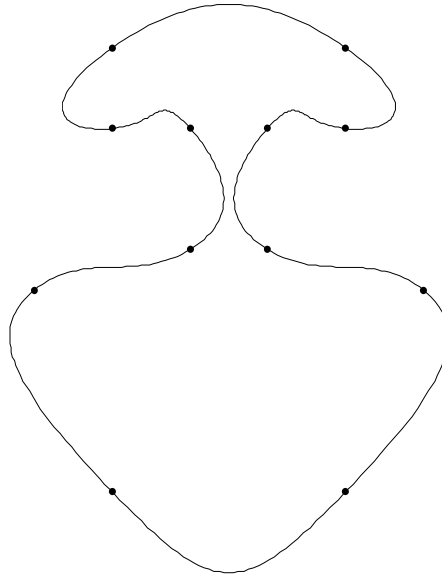


Figure 3.8 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = -3, \forall i$

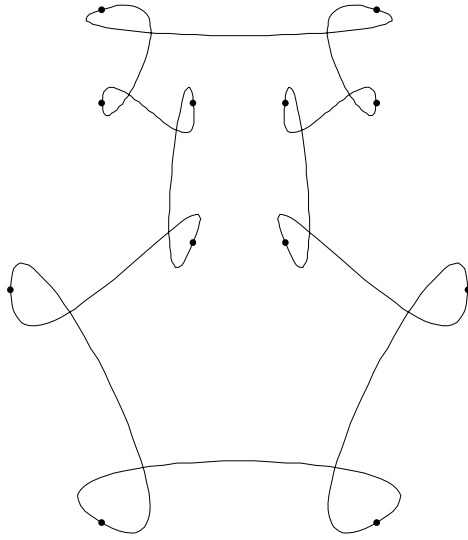


Figure 3.9 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = -15, \forall i$

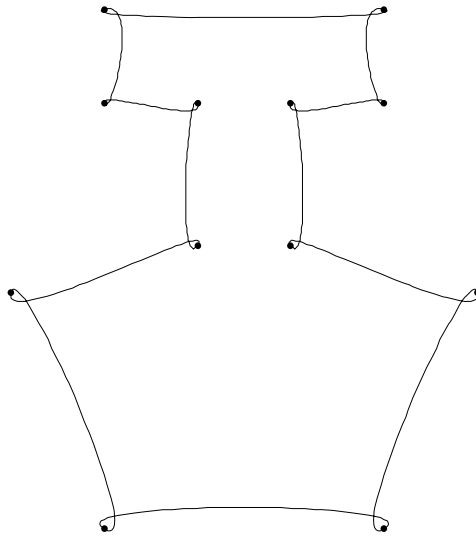


Figure 3.10 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = -25, \forall i$

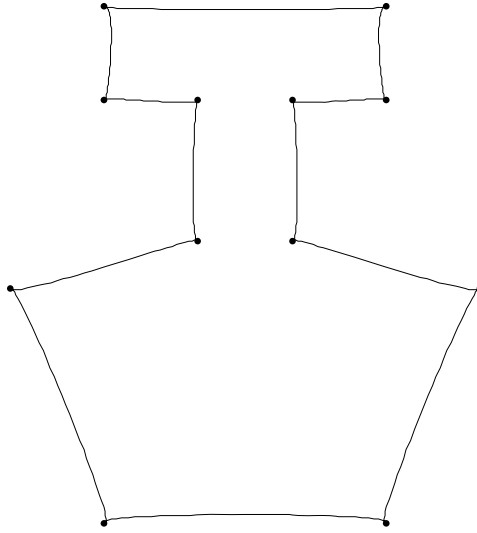


Figure 3.11 Global values of point tension shape parameter. $\omega_i = 1$ and $\nu_i = -50$, $\forall i$

3.4.2 Local Support Basis Form

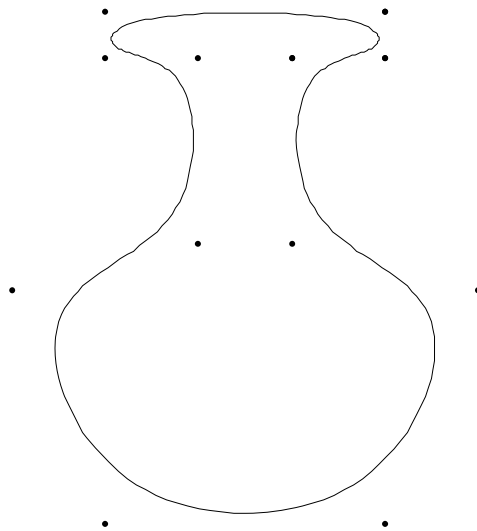


Figure 3.12 Default values of parameters $\omega_i = 1$ and $\nu_i = 0$ for shape Pot

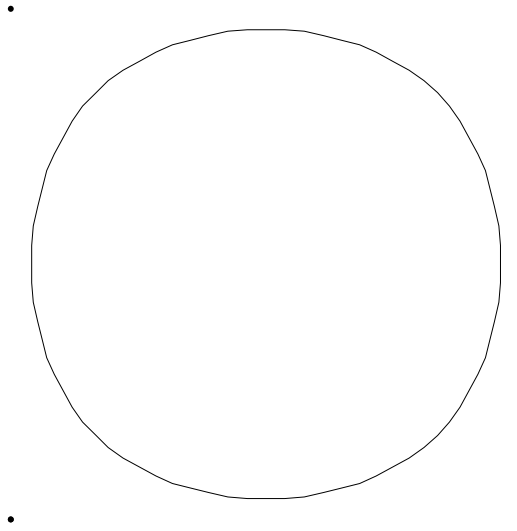


Figure 3.13 Default values of parameters $\omega_i = 1$ and $\nu_i = 0$ for shape Square

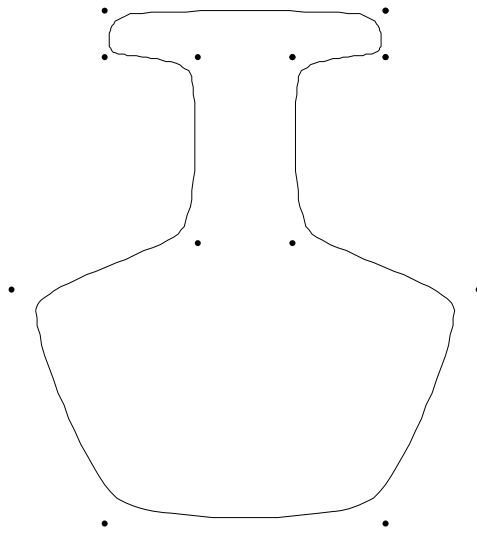


Figure 3.14 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = 10$

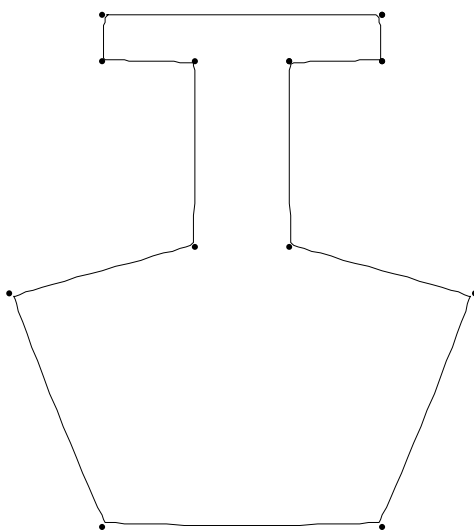


Figure 3.15 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = 100$

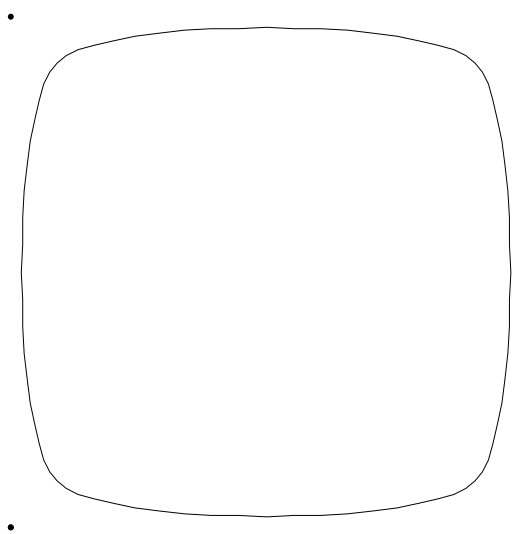


Figure 3.16 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = 10$

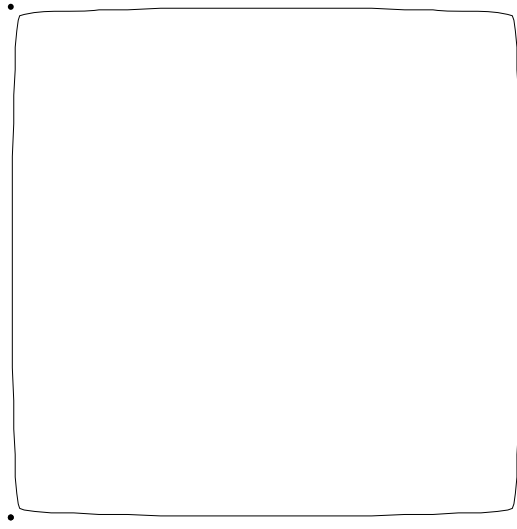


Figure 3.17 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = 100$

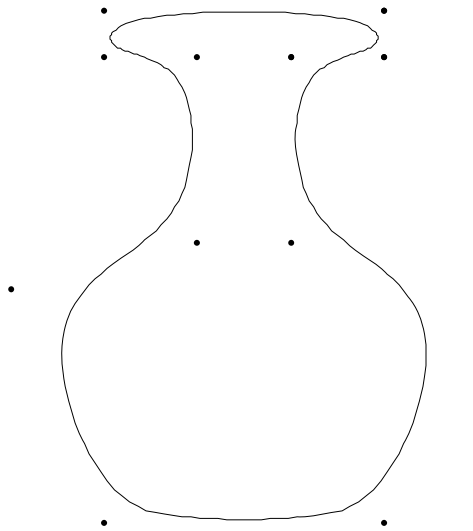


Figure 3.18 Interval tension values of parameters for bottom segment $\omega = 10$ and

$$\nu = 0$$

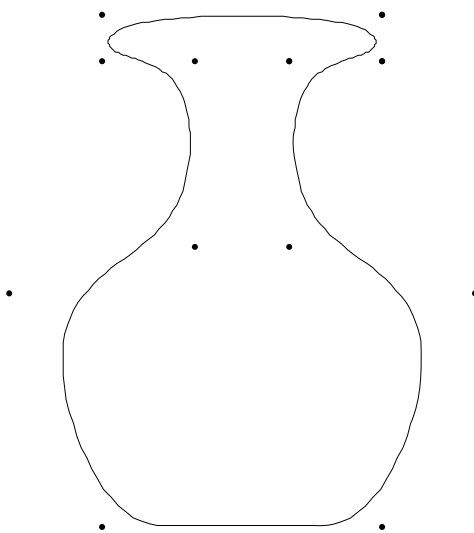


Figure 3.19 Interval tension values of parameters for bottom segment $\omega = 100$ and

$$\nu = 0$$

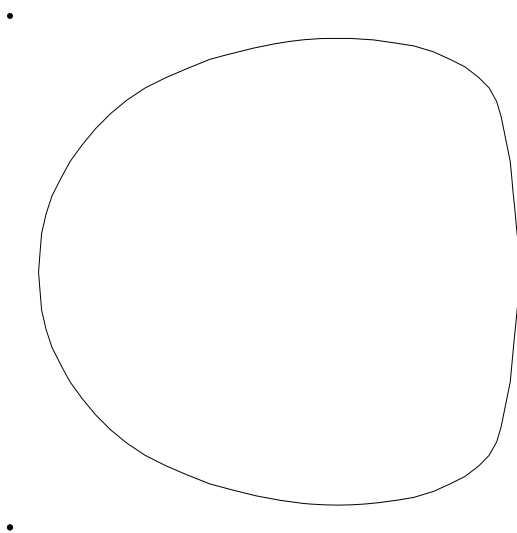


Figure 3.20 Interval tension values of parameters for bottom segment $\omega = 10$ and

$$\nu = 0$$

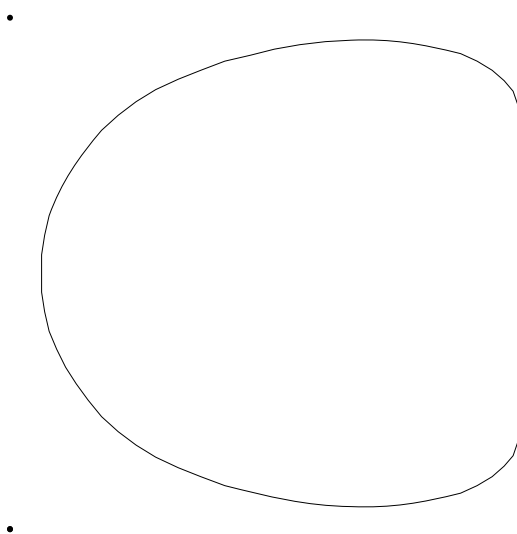


Figure 3.21 Interval tension values of parameters for bottom segment $\omega = 100$ and

$$\nu = 0$$

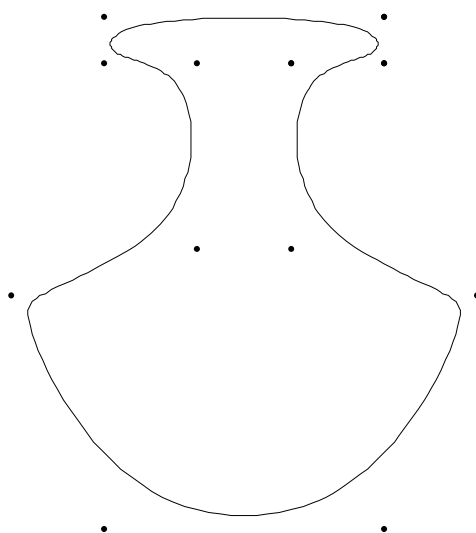


Figure 3.22 Point tension values at specified points $\omega = 1$ and $\nu = 10$

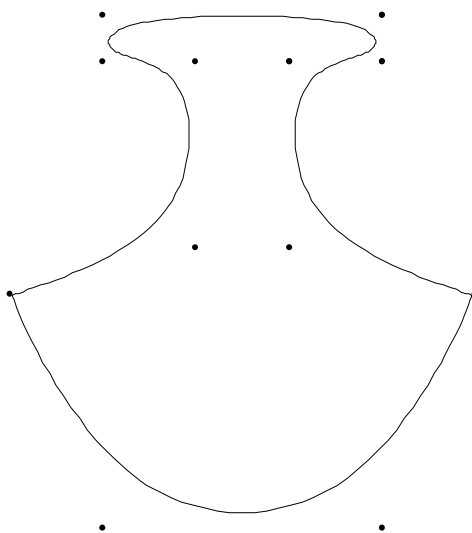


Figure 3.23 Point tension values at specified points $\omega = 1$ and $\nu = 100$

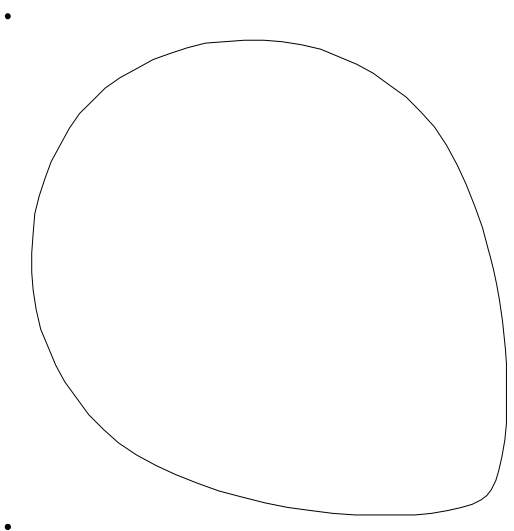


Figure 3.24 Point tension values at specified points $\omega = 1$ and $\nu = 10$

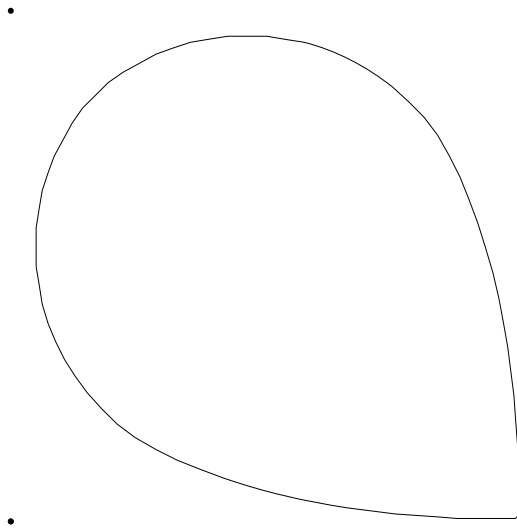


Figure 3.25 Point tension values at specified points $\omega = 1$ and $\nu = 100$

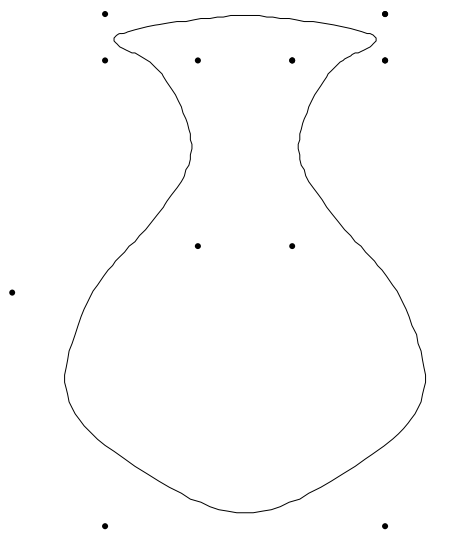


Figure 3.26 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -3$

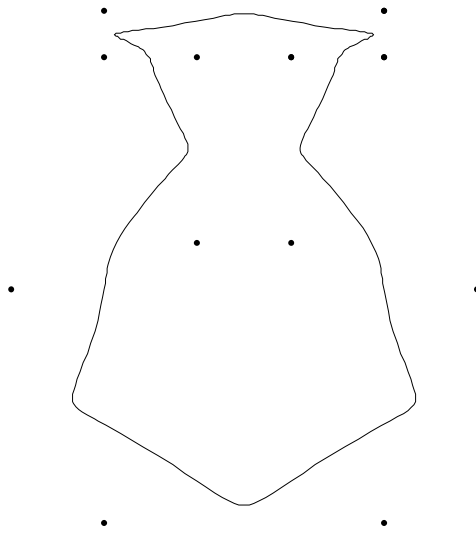


Figure 3.27 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -5$

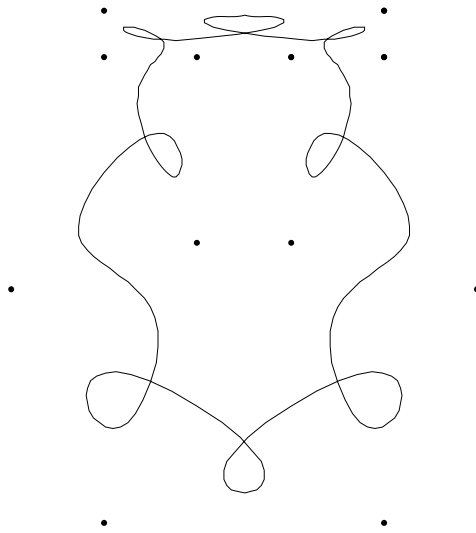


Figure 3.28 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -8$

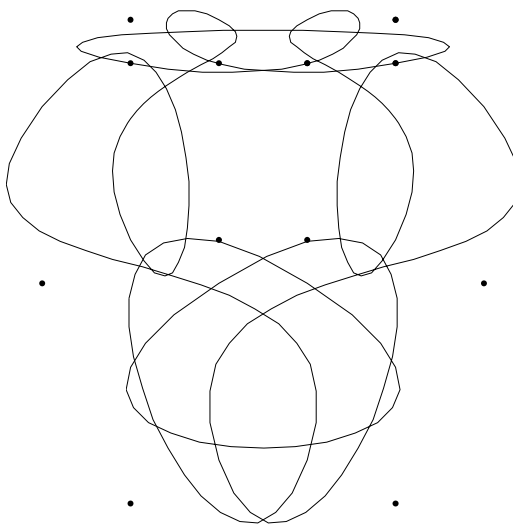


Figure 3.29 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -10$

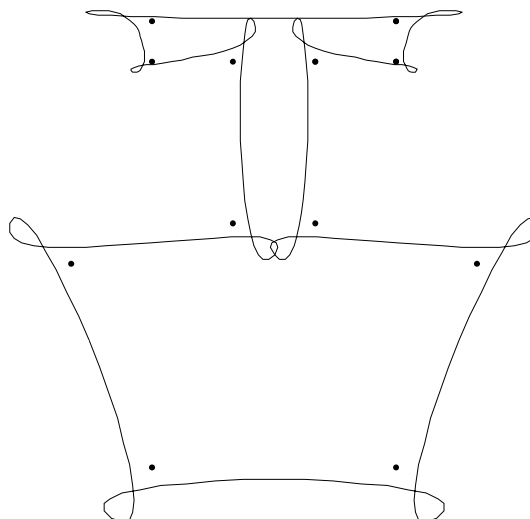


Figure 3.30 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -20$

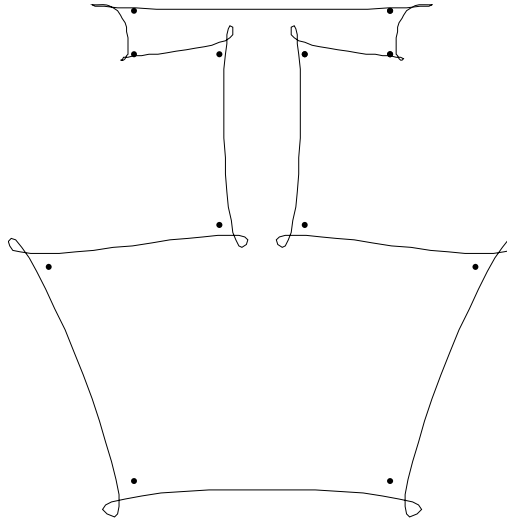


Figure 3.31 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -25$

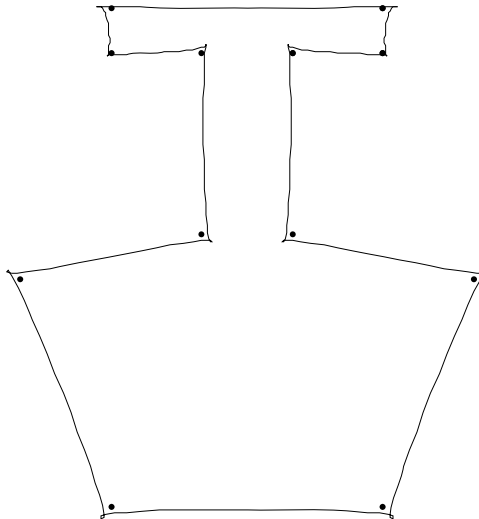


Figure 3.32 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -50$

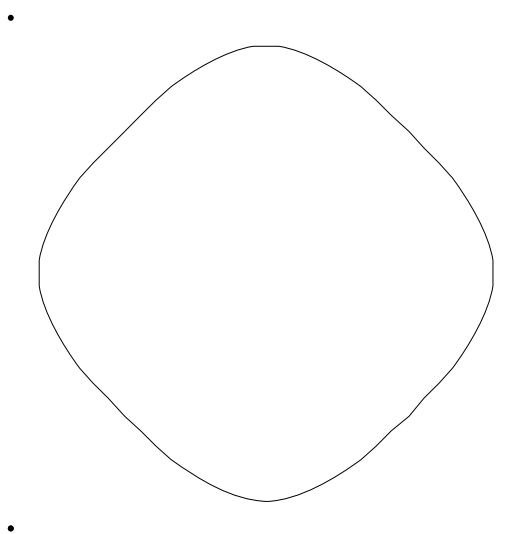


Figure 3.33 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -3$

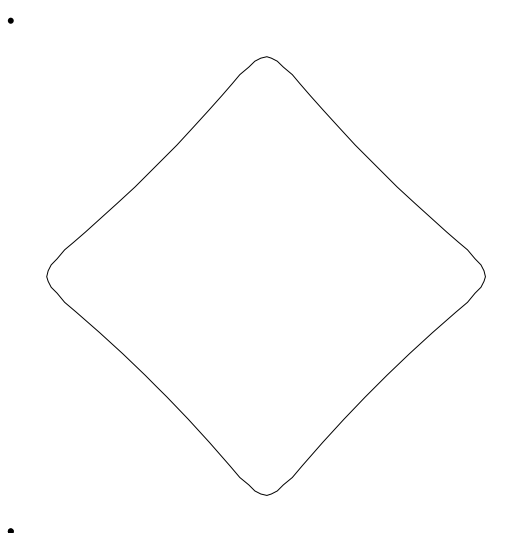


Figure 3.34 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -5$

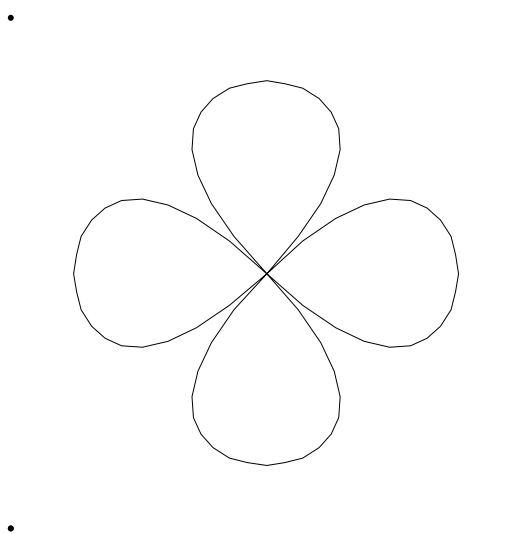


Figure 3.35 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -8$

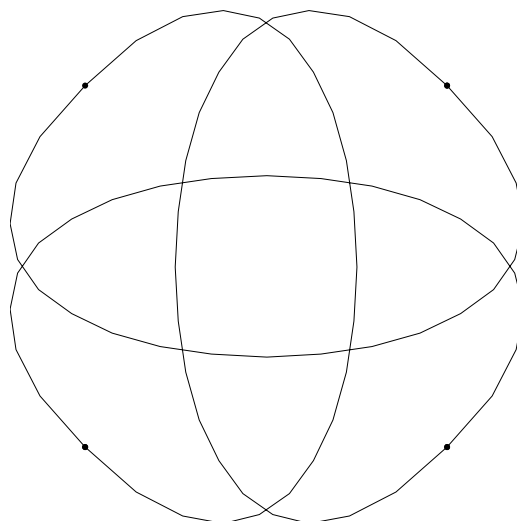


Figure 3.36 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -10$

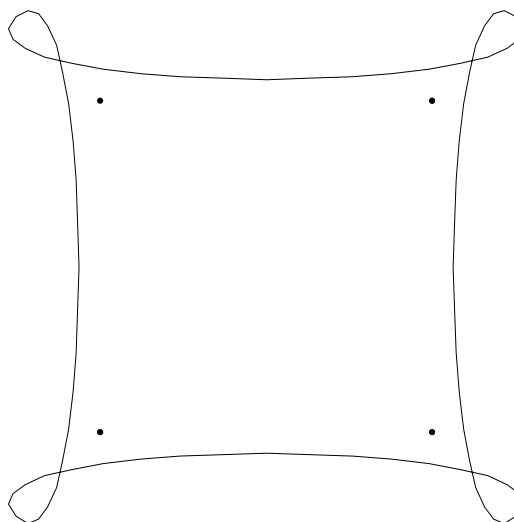


Figure 3.37 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -20$

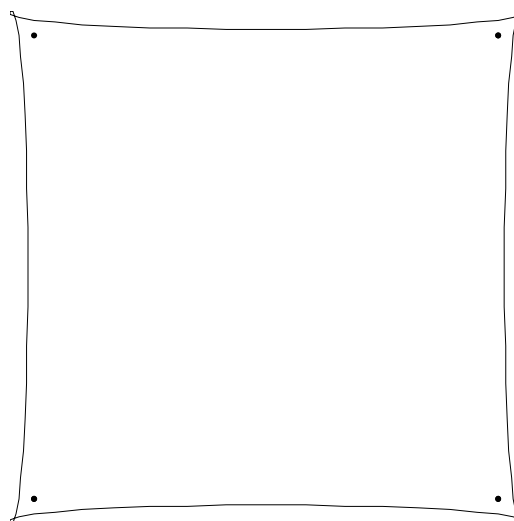


Figure 3.38 Global values of point tension parameters $\omega_i = 1$ and $\nu_i = -50$

3.4.3 Timmer Parametric Cubic

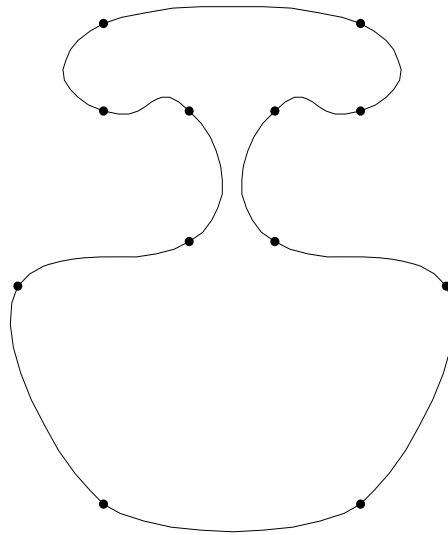


Figure 3.39 Global values of shape parameters $\alpha_i = 1$ and $\beta_i = 1$

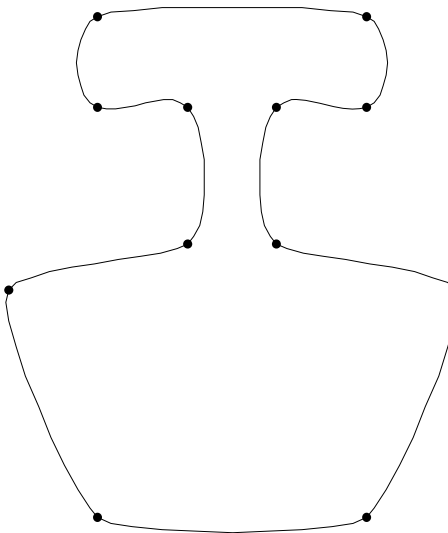


Figure 3.40 Global values of shape parameters $\alpha_i = 2$ and $\beta_i = 2$

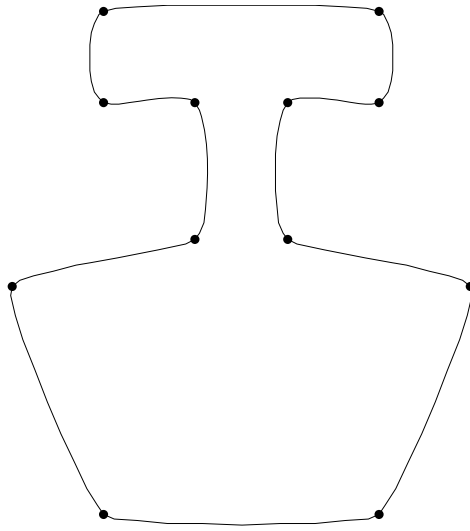


Figure 3.41 Global values of shape parameters $\alpha_i = 3$ and $\beta_i = 3$

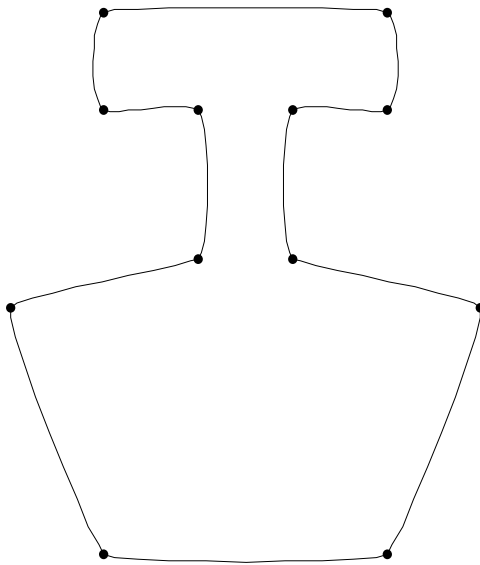


Figure 3.42 Global values of shape parameters $\alpha_i = 4$ and $\beta_i = 4$

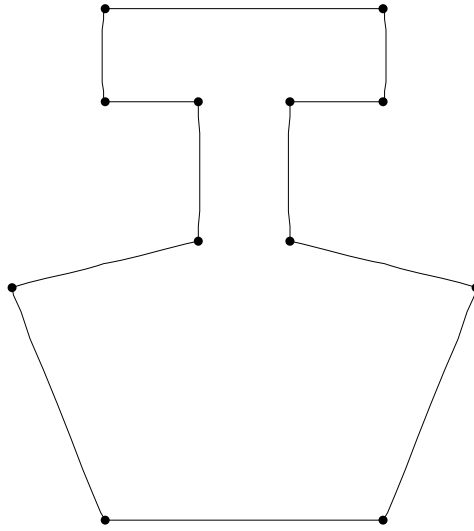


Figure 3.43 Global values of shape parameters $\alpha_i = 20$ and $\beta_i = 20$

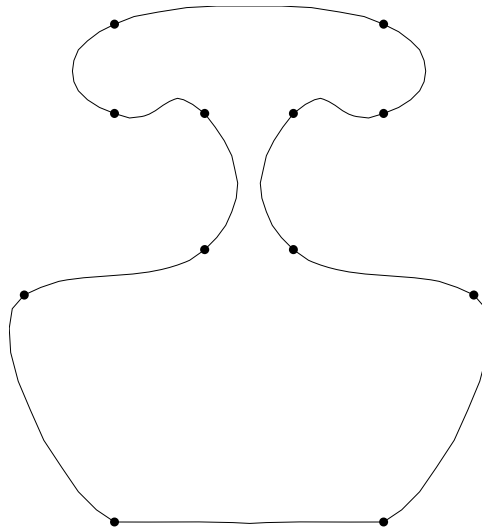


Figure 3.44 Interval tension at base $\alpha = 15$ and $\beta = 15$, for $default = 1$

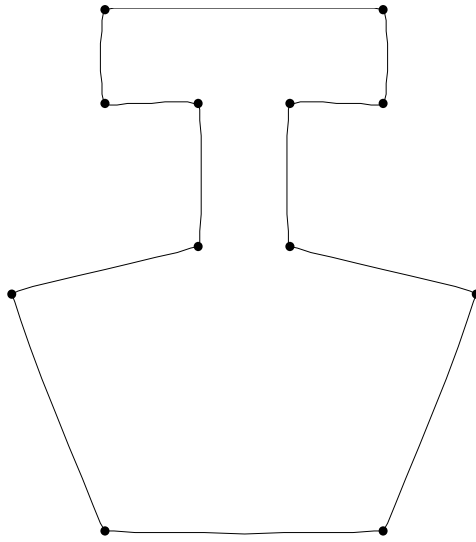


Figure 3.45 Global values of shape parameters $\alpha_i = -10$ and $\beta_i = -10$

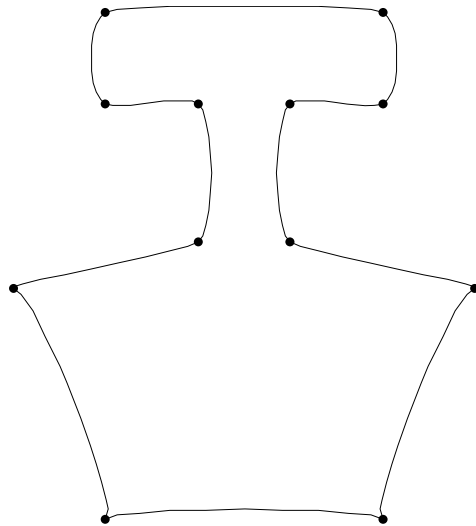


Figure 3.46 Interval tension at base $\alpha = -5$ and $\beta = -5$, for *default* = 3

3.5. Conclusion

Cubic spline method has been developed with the prospect of its applications in Computer Graphics and Geometric Modeling. The proposed methodology covered in this chapter includes the development of GC^2 interpolatory form of curve design and as well as freeform GC^2 curve design. The free form curve design is developed through the construction of local support B-Spline like basis functions. Our proposed approach incorporates all the good features of cubic splines and along with that it also includes two families of shape design, known as point and interval tension, which behave in well controlled and meaningful way. Further more the freeform curve design also enjoys the features of local control and as well as global control. However, we have not considered the use of shape parameters for control of convexity for the interpolatory case. Moreover, our approach also recovers C^2 continuity as special case. We have also studied a special class of our generic spline model, Timmer parametric cubic splines. We have designed its interpolant forms for C^1 and C^2 continuous shapes. Also we have studied the shape control parameters for this class.

CHAPTER 4

CURVE DESIGN

Data point approximation is a renowned problem of computer graphics, computer vision, image analysis, CAD/CAM etc. This kind of approximation is usually done by computing a curve close to the data point set [1,10,16,40,58]. The representation of planar object in terms of curve has many advantages, for example, scaling, shearing, translation, rotation and clipping operations can be performed. Further more we get the freedom to play with the shape of the object and we can tune it as desired and thus extending this research work for object designing.

Vectorization of raster graphics is one of the fundamental research areas of computer graphics, image processing and computer vision. A lot of research has been done in this area [2-5,8,17,18,20,22,53,54]. The application ranges from designing and reconstruction to recognition of objects. These objects vary from simple to complex geometrical shapes like space craft model, structural objects related to civil engineering, mechanical engineering objects, bio-medical equipment designing etc.

Most of the research has tackled this kind of problem by curve subdivision or curve segmentation [68,89]. Curve segmentation is advantageous in a way that it gives a rough geometry of the shape. Approaches used to achieve this task are polygonal approximations [5,8,22], circular arc approximations [2,18,25,26,31,35,38,39,47-49,52,63,65-67] and approximations using cubic or higher order splines [17,19,27,28,33,53,54,74].

Many approaches discussed use parametric piecewise-cubic functions, which are used throughout the computer graphics industry to represent curved shapes. For many applications, it is preferable to have such representation from a closely spaced set of points that approximate the desired curve.

A non-parametric dominant point detection algorithm was proposed in [22], it used these dominant points for polygonization of digital curves. The problem with polygonal approximation is that these approaches are rarely used for shape analysis [5,8].

Algorithm for conic approximation is proposed in [35,47,66,67]. The combination of line segments and circular arcs for object approximation can be seen in [25,26,63]. In [18] the authors have proposed a scheme to construct a curvature continuous conic spline. They presented the conic spline curve fitting and fairing algorithm using conic arc scaling. The smoothing is done by removing unwanted curvature extrema. Conic splines can also be used to fit a piecewise linear curve or another smooth curve [48]. Algorithms for data

fitting by arc spline curves is also presented in [38,49]. A method for segmentation of curves into line segments and circular arcs by using types of breakpoints is proposed in [2]. Advantage of this technique is that it is threshold free and transformation invariant. The authors have defined five categories of breakpoints. The line and conic segmentation and merging is based on these breakpoints. The computational complexity of the proposed algorithm is $O(n \log n)$. Short coming of arc splines is that they cannot be used for high quality shape modeling as desired smoothness cannot be achieved.

Least square fitting is mostly adopted in approximations which use splines and higher order polynomials. Usually the objective is to minimize the sum squared error measures [31,39,52,65]. This kind of fitting is largely dependent upon proper parameterization [74]. Another approach is based on active contour models known as snakes. Application of this approach is [19,27,28]. This technique is also based on parameterization. Enhancement to the scheme by adjusting both number and position of control points of the active spline curve is shown in [17]. Here the authors used curve approximation using iterative optimization with B-spline curve based on squared distance minimization.

Another way, other than parametric form, is to use implicit form of the polynomial. Curve reconstruction problem is solved by approximating the point clouds using implicit B-spline curve. Trust region algorithm is used in optimization theory as minimization heuristics [4]. Techniques described for fitting implicitly defined algebraic spline curves

and surfaces to scattered data by simultaneously approximating points and associated normal vectors are proposed by [37,50,51].

In our case the data point set represents planar object, the outline of which we want to capture. We present an iterative process to achieve our objective. The algorithms comprise of first finding the contour of the gray scaled bitmap image. Then the corners are detected using [1]. The first two steps are taken as preprocessing steps. We are using generic cubic spline curves, described in last chapter, for curve fitting. We have proposed two classes of algorithms. The first one involves random process and the second one is about the application of fuzzy inference rules. The first class includes further the description of four different variations.

In the first class of algorithms, for each iteration we are inserting random point(s) as knot in every piece if the distance of random point(s) and its corresponding contour point(s), d , is greater than ε . This is computationally much efficient as compared to computing least square distance or least mean square error for all data points. We stop the iteration when all d 's are less than ε . In the other class of algorithm, we take three random points and then compute their fuzzy membership for being a knot. The one with the highest fuzzy value is then taken as a knot. Again this algorithm is also iterative in nature and it stops when for all pieces the distance between fuzzy knot and corresponding contour point is less than ε . The second algorithm is little bit costlier in terms of time complexity as compared to other algorithms but it gives better curve approximation.

Our proposed algorithm for capturing the outline of digital images in general consists of the following steps;

- Finding Boundary of Object
- Detecting Corner from the Boundary
- Curve Fitting with Cubic Spline
- Inserting Knots for Breaking the Segment

The flow of the program is shown in (Figure 4.1). The steps are discussed in details in later sections.

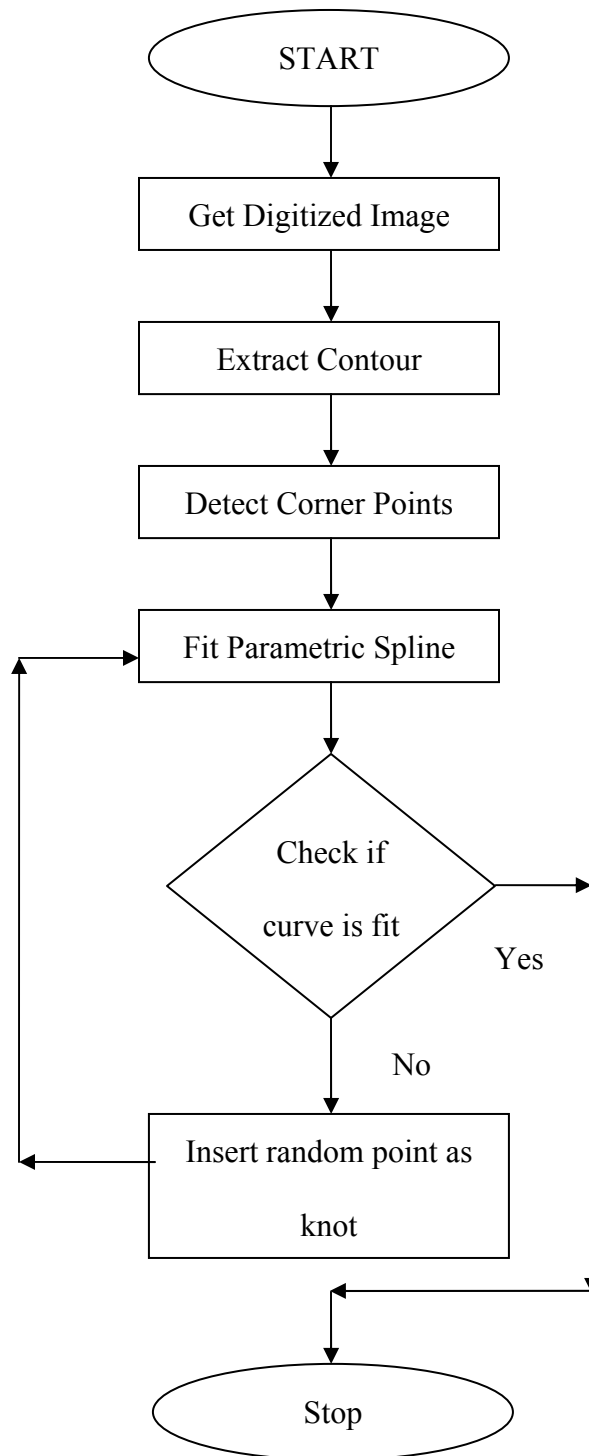


Figure 4.1 Outline capturing of the digital images

4.1 Preprocessing

This step consists of first finding the boundary of the planar object and then using the output to find the corner points or the significant points.

4.1.1 Finding Boundary of Planar Object

The image of the object can be acquired either by scanning or by making it in software like MS-Paint or Adobe Photoshop. In case of scanning the quality of scanned image is dependent upon factors such as paper quality and scanning resolution. The better the resolution and paper quality the better will be the image. On the other hand if software is used then the quality is dependent upon the format in which the image is stored. For example, .bmp files have more detail than .jpeg.

The aim of boundary detection is to produce an object's shape in graphical or non-scalar representation. Chain codes [34,36] are the most widely used representations. Other well known representations are syntactic techniques, boundary approximations and scale-space techniques. The benefit of using chain code is that it gives the direction of edges. The boundary points are selected as contour points based on their corner strength and fluctuations.

Chain codes were initially proposed by Freeman. The methodology adopted to detect the boundary is by encoding the shape boundary as a sequence of connected line segments of specified length and direction. The direction of a segment is coded using either 4-connected or 8-connected schemes. In both schemes initially a point is selected using either horizontal or vertical scan. After this, the 4-connected or 8-connected component algorithm is applied. Both algorithms work in intensive stack formulation. In case of 4-connected, four neighboring points are analyzed. These points are pixel positions that are right, left, above and below the current pixel. The second method is a little more complex. In this method the set of neighboring positions to be tested include the four diagonal pixels as well.

The point set obtained after this step is known as contour of the object. The bitmap image and its contour are shown in (Figures 4.2) and (Figure 4.3) respectively.

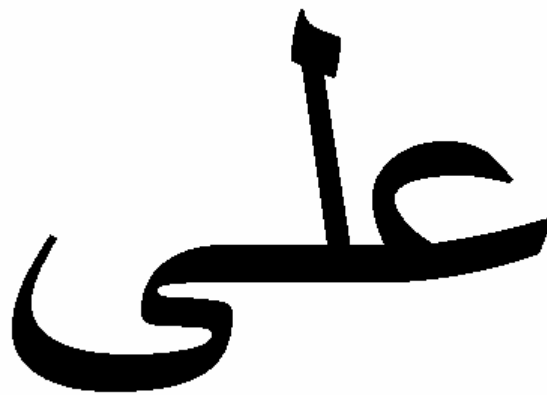


Figure 4.2 Bitmap image

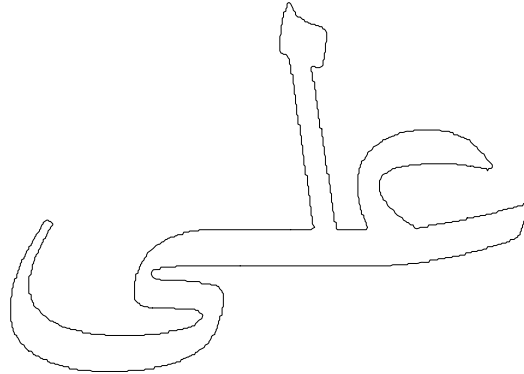


Figure 4.3 Contour of the bitmap image

4.1.2 Corner Detection

Accurate detection of corners in digital images accounts in the geometrical feature representation and analysis [1,22]. The corner detector used in this phase is described in detail in Chapter 2. However, for the ease of readers, we present the summary of proposed scheme.

The algorithm is composed of three phases. Slope analysis is done in the first phase. In this phase candidate points are chosen if slope is changing. These candidate points are then passed to the second phase where closed coordinate points are removed. The second phase acts as a preprocessing step for the third phase, which is the last phase of our algorithm and gives the valid corner points as result. This phase is used to remove the points which are lying in near proximities to each other and are selected in clusters because of jaggy nature of contour. The removal process is based upon angular

measurement. If the angle calculated exceeds a threshold then the candidate point is dropped from the list of valid corners.

In each phase the algorithm involves tuning parameters, which are used as thresholds. First phase includes Zero Count Threshold (ZCT). ZCT is used to differentiate a jaggy from a straight line. This is needed because we are selecting the two end points of straight line. Incase of jaggy, we do not select the end points as candidate points for being corners. Distance Threshold (DT) is used in second phase of the algorithm. It is used to remove all those candidate points which are in the limits of DT. Finally Tolerance Angle (TA) is applied in the third and final phase. It is calculated with the help of consecutive three candidate points. If these three points make an angle greater than TA then the middle point is removed from the list of valid corner points. The default value of ZCT is 7, DT is 5 and TA is 152° .

The advantage of this technique is that, we can apply this algorithm for smooth functions as well as irregular objects with jaggies. A distinct property of our algorithm is that the default values of tuning parameters work equally well for almost all the shapes, regardless of the object contour. Further, the change in default parameters does not produce much impact on the outcome. The corner points of the image are shown in (Figure 4.4).

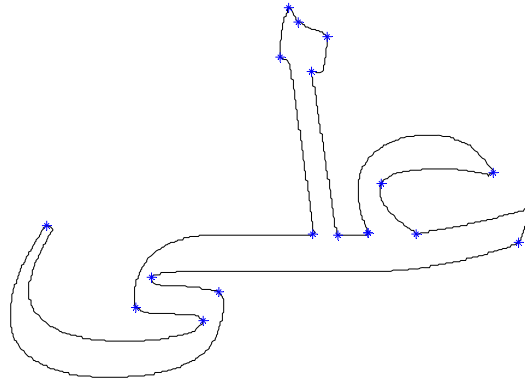


Figure 4.4 Corner Points of the Image

4.2 Curve Fitting with Cubic Spline Model

The motive of finding the corner points was to divide the contour into pieces. Each piece contains the data points in between two subsequent corners inclusive. This means that if there are m corner points cp_1, cp_2, \dots, cp_m then there will be m pieces p_1, p_2, \dots, p_m . We treat each piece separately and fit the spline [55] to it. First piece includes all the contour points in between cp_1 and cp_2 including them as well. Second piece contains all contour points in between cp_2 and cp_3 inclusive. Consequently, the m^{th} piece contains all contour points between cp_m and cp_1 . This is represented in (Figure 4.5).

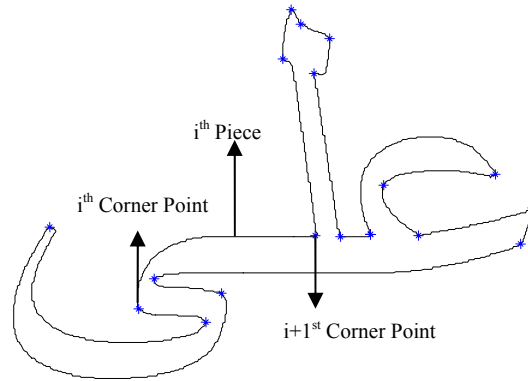


Figure 4.5 Contour Division into Pieces

After breaking the contour of the image into different pieces, we fit the spline curve to each piece. For this purpose we have used piecewise parametric rational cubic spline interpolant. Each parametric spline is GC^2 continuous.

4.2.1 Cubic Spline Interpolant Form

We have used the interpolant form of the cubic generic spline model that we have already explained in detail in (Section 3.1). However, for the ease of readers we will present the concepts again in brevity.

The cubic generic spline model is given as under,

$$P_i(t) = (1-\theta)^2 \{1 + \theta(2-\alpha_i)\} F_i + \alpha_i (1-\theta)^2 \theta V_i + \beta_i (1-\theta) \theta^2 W_i + \theta^2 \{1 + (1-\theta)(2-\beta_i)\} F_{i+1} \quad (4.1)$$

where,

$$\theta|_{[t_i, t_{i+1})}(t) = \frac{(t - t_i)}{h_i} \quad (4.2)$$

and

$$V_i = F_i + \frac{M_i h_i}{\alpha_i}, \quad W_i = F_{i+1} - \frac{M_{i+1} h_i}{\beta_i} \quad (4.3)$$

To get the control points $\{F_i, V_i, W_i, F_{i+1}\}$, we made use of a Bernstein-Bezier representation where we can impose the Hermite interpolation condition.

$$\begin{aligned} P(t_i) &= F_i, & P(t_{i+1}) &= F_{i+1} \\ P'(t_i) &= M_i, & P'(t_{i+1}) &= M_{i+1} \end{aligned} \quad (4.4)$$

F_i and F_{i+1} are corner points of i^{th} piece. M_i and M_{i+1} are the corresponding tangents at corner points.

To construct the parametric GC^2 cubic generic spline interpolant on the interval $[t_0, t_n]$ we have $F_i \in R^m$, $i = 0, \dots, n$ as interpolation data at knots t_i , $i = 0, \dots, n$. The derivatives $M_i \in R^m$ can be found out by the imposition of GC^2 constraints on the piecewise defined Hermite form of the spline model. The GC^2 constraint can be written as,

$$P_i''(t_i) = \frac{v_i}{\omega_i} P_{i-1}'(t_i) + \frac{\omega_{i-1}}{\omega_i} P_{i-1}''(t_i) \quad (4.5)$$

This gives us the tri-diagonal system of consistency equations,

$$\frac{2\omega_{i-1}M_{i-1}}{\omega_i h_{i-1}} + \left(\frac{4\omega_{i-1}}{\omega_i h_{i-1}} + \frac{v_i}{\omega_i} + \frac{4}{h_i} \right) M_i + \frac{2M_{i+1}}{h_i} = \frac{6\Delta_i}{h_i} + \frac{6\omega_{i-1}\Delta_{i-1}}{\omega_i h_{i-1}} \quad (4.6)$$

where,

$$\Delta_i = \frac{F_{i+1} - F_i}{h_i} \quad (4.7)$$

Multiply (Equation 4.6) by $\omega_i/2$ and then put the $c_i = \omega_i/h_i$, we get.

$$c_{i-1}M_{i-1} + \left\{ \frac{v_i}{2} + 2c_{i-1} + 2c_i \right\} M_i + 2c_i M_{i+1} = 3c_i \Delta_i + 3c_{i-1} \Delta_{i-1} \quad (4.8)$$

Dividing (Equation 4.8) by the co-efficient of M_i will give us unit diagonal form,

$$a_i M_{i-1} + M_i + c_i M_{i+1} = b_i, \quad i = 1, \dots, n-1 \quad (4.9)$$

Then (Equation 4.9) will give us diagonally dominant tri-diagonal system of linear equations in the unknowns M_i where, $i = 1, \dots, n-1$.

Spline fitting is shown in (Figure 4.6).



Figure 4.6 Spline fit over Object Contour

4.3 Knot Insertion

We have developed two classes of algorithms, namely *Randomized Knot Insertion* and *Fuzzy Random Knot Insertion*. Algorithms in both classes are iterative and random in nature and run piecewise on the contour of the object. Since the algorithms are random in nature therefore for each run, approximation takes different number of iterations for curve fitting.

4.3.1. Randomized Knot Insertion

The idea here is to fit a spline model to the object contour in such a way that the spline curve approximates the data points on the contour. The whole process is done in piecewise manner. The contour is divided into multiple segments. We consider all segments or pieces one by one, in each iteration. We have devised four algorithms based on this idea. All algorithms vary from each other in knot selection.

In our first algorithm, *Single Unconstrained Random Point Algorithm*, we pick a point on the spline curve at random considering it as candidate knot. Then we calculate the Euclidean distance of this candidate knot with the corresponding point on the contour. We insert the candidate knot into the appropriate position of corner points list if the distance is greater than a threshold ε , where $\varepsilon > 0$. The value of ε depends upon the user's choice of how close approximation he wants, the lesser the value the closer will be approximated curve fit. The formulation of the whole process is described as under.

Let's suppose that there are n points, P_1, P_2, \dots, P_n in j^{th} segment as shown in (Figure 4.7). To get a candidate random point r on the spline model in this segment, we used *rand()* function. This function generates random number whose value is uniformly distributed in the interval (0,1) and therefore we multiply the resultant value by 100 to get a representative value. Now let's suppose that the value generated by this function is x . Then the location of the candidate random point is given in (Equation 4.10).

$$LOC = \left\lceil \left(\frac{x}{100} \right) \times n \right\rceil \quad (4.10)$$

The ceiling function helps in avoiding selection of first corner point. This location exists in between the corner points of that particular segment. Once we have got the location of the candidate random point. We calculate the Euclidean distance between the corresponding points on the object contour represented by P_r and the point on the spline fit represented by C_r using (Equation 4.11).

$$|d| = \sqrt{(C_{r_x} - P_{r_x})^2 + (C_{r_y} - P_{r_y})^2} \quad (4.11)$$

The condition for valid knot selection is given as in (Equation 4.12);

$$|d| \geq \varepsilon, \quad 0 \leq \varepsilon \leq c \quad \text{where } c > 0 \quad (4.12)$$

The valid knot selection condition depicts that if this distance is greater than threshold ε then we select this candidate point as valid knot and insert it into the proper location in the corner points list. Where ε is defined by the user. Lower the value of ε the better approximation will be achieved. And subsequently if we increase the value of ε we get average approximation. Incase where we want to have optimal interpolation, that is, all spline computed points pass through the contour points then we put the value of ε as 0. This will increase the number of iterations for approximation and thus there will be

increase in total number of knots. In cases where interpolation is not required, a greater value of ε can be used to achieve the approximation. This threshold gives the user a freedom to use curve fit as per his desires.

We apply this process of finding valid knots to the whole list of pieces and if we are able to get at least one knot then a new spline fit is obtained and the whole process is repeated. On the other hand if the distance calculated is less than ε then we leave this candidate point and we check the next piece. We stop the process when we do not get any candidate point as valid knot.

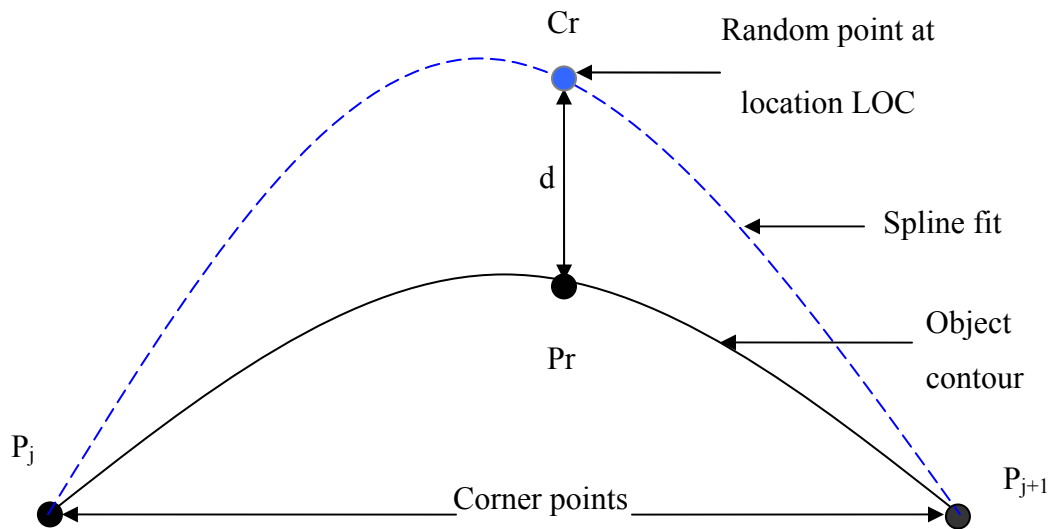


Figure 4.7 Calculation of random knot in Algorithm 4.1

(Algorithm 4.1) shows the steps of curve fitting.

Algorithm 4.1: Single Unconstrained Random Point

Step 1: For each piece do the following

Step 1.1: Pick a random point on the spline curve

Step 1.2: Calculate its Euclidean distance " d " with the corresponding point on the contour

Step 1.3: If $d > \varepsilon$

Step 1.3.1: Set the flag for another iteration, $flag = TRUE$

Step 1.3.2: With respect to its position, insert it into the list of corner points.

Step 2: If $flag = TRUE$ repeat step 1 otherwise stop the iteration.

The second approach, *Single Euclidean Distance Constraint Random Point Algorithm*, also works in similar fashion as that of first approach. The only difference comes in that before selecting the candidate knot as corner, it is also checked whether it lies in the proximity of either corner points of particular segment or not. The distance between the candidate point and both corner points is calculated using the Euclidean distance formula.

In short, if the Euclidean distance between candidate point and its respective point on the contour is more than the threshold ε and the candidate point is not near the corner points then this candidate point is taken as knot and it is then inserted in the appropriate position of corner points list. If the segment under consideration is j^{th} , then left corner point is denoted as P_j and the right corner point is denoted as P_{j+1} . Also suppose that random

candidate point is denoted as C_r . The distance formula for both sides can be given as in (Equation 4.13) and (Equation 4.14),

$$|d_l| = \sqrt{(C_{r_x} - P_{j_x})^2 + (C_{r_y} - P_{j_y})^2} \quad (4.13)$$

And,

$$|d_r| = \sqrt{(C_{r_x} - P_{j+1_x})^2 + (C_{r_y} - P_{j+1_y})^2} \quad (4.14)$$

The proximity criterion is again user's choice. The nearness threshold criterion is given as in (Equation 4.15);

$$\begin{aligned} |d_l| &< \tau \\ \text{or} \\ |d_r| &< \tau \end{aligned} \quad (4.15)$$

Where d_l is the Euclidean distance between the candidate point and left corner point of the segment. d_r is the Euclidean distance between the candidate point and the right corner point of the segment and τ is the threshold value which holds the nearness range. The description is shown in (Figure 4.8). The bold area in the figure shows restricted vicinity for a candidate point to be taken as knot.

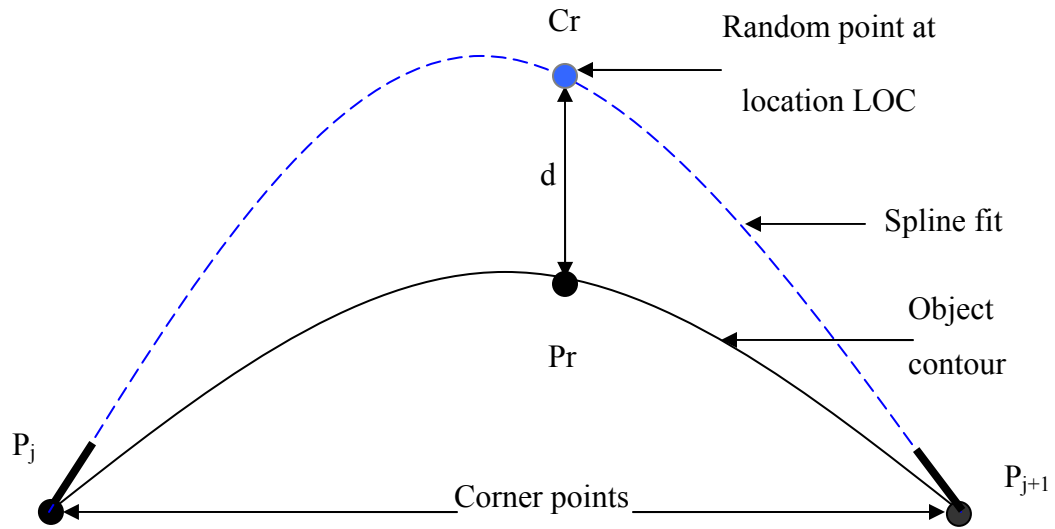


Figure 4.8 Calculation of random knot in Algorithm 4.2 and 4.3

The algorithm for this approach is outlined in (Algorithm 4.2).

Algorithm 4.2: Single Euclidean Distance Constraint Random Point

Step 1: For each piece do the following

Step 1.1: Pick a random point on the spline curve

Step 1.2: Calculate its Euclidean distance " d " with the corresponding point on the contour

Step 1.3: Calculate its Euclidean distance " d_l " with the left corner point.

Step 1.4: Calculate its Euclidean distance " d_r " with the right corner point.

Step 1.5: If $d > \varepsilon$ & $|d_l| > \tau$ & $|d_r| > \tau$

Step 1.5.1: Set the flag for another iteration, $flag = TRUE$

Step 1.5.2: With respect to its position, insert it into the list of corner points.

Step 2: If $flag = TRUE$ repeat step 1 otherwise stop the iteration.

The third, *Single Positional Distance Constraint Random Point Algorithm*, approach is a very slight variation of second one. Here we follow all the steps of second approach except for calculating the Euclidean distance for candidate knot proximity we calculate the positional distance. It is calculated based on index position information. That is if the candidate knot is near to either of the corners with respect to its position then we will not take such a point as knot. The closeness threshold τ in this case is taken as a percent value with respect to the total points on the piece.

Let's suppose that there are m points in a piece then, the actual index IDX to be considered as threshold vicinity can be expressed as in (Equation 4.16) and shown as bold shade in (Figure 4.8),

$$IDX = \left\lceil \frac{\tau}{100} \times m \right\rceil \quad (4.16)$$

The algorithm for third approach is written in (Algorithm 4.3).

Algorithm 4.3: Single Positional Distance Constraint Random Point

Step 1: For each piece do the following

Step 1.1: Pick a random point on the spline curve

Step 1.2: Calculate its Euclidean distance " d " with the corresponding point on the contour

Step 1.3: Calculate its positional distance " d_l " with the left corner point.

Step 1.4: Calculate its positional distance " d_r " with the right corner point.

Step 1.5: If $d > \varepsilon$ & $|d_l| > \tau$ & $|d_r| > \tau$

Step 1.5.1: Set the flag for another iteration, $flag = TRUE$

Step 1.5.2: With respect to its position, insert it into the list of corner points.

Step 2: If $flag = TRUE$ repeat step 1 otherwise stop the iteration.

The fourth algorithm, *Three Unconstraint Random Points Algorithm*, takes three random points as candidate knots and inserts them in list of valid corners or break points if their distances with respective points on the contour are less then the threshold. Rest of the process is followed as in case 1. This is demonstrated in (Figure 4.9).

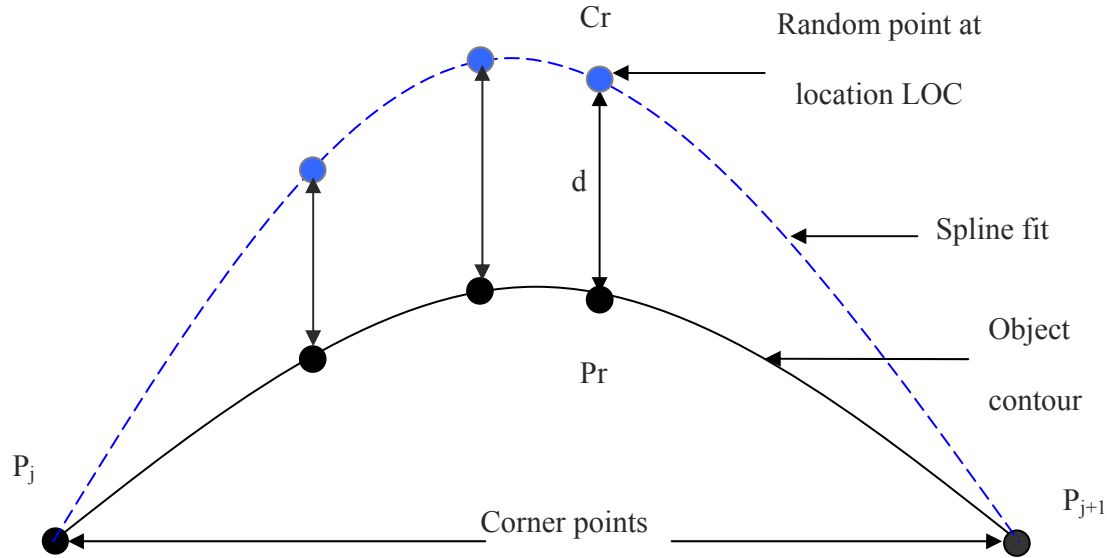


Figure 4.9 Calculation of random knot in Algorithm 4.4

The algorithm for fourth approach is written in (Algorithm 4.4).

Algorithm 4.4: Three Unconstraint Random Points

Step 1: For each piece do the following

Step 1.1: Pick three random point on the spline curve

Step 1.2: Calculate their Euclidean distances " d " with the corresponding points on the contour

Step 1.3: Check for each point separately if $d > \varepsilon$

Step 1.3.1: Set the flag for another iteration, $flag = TRUE$

Step 1.3.2: With respect to position of point under consideration, insert it into the list of corner points.

Step 2: If $flag = TRUE$ repeat step 1 otherwise stop the iteration.

The fifth algorithm, *Three Equal Spaces Segmented Random Points Algorithm*, is an enhancement of fourth approach. In this case instead of selecting three random points, first we are dividing the segment into three equally spaced sub-segments and then treating them as pieces, as shown in (Figure 4.10) with the help of vertical lines on the spline curves. We are then randomly selecting a candidate random knot for each of these sub-segments. These random candidate points are taken as valid control points incase their distances with corresponding points on the contour are greater than the threshold.

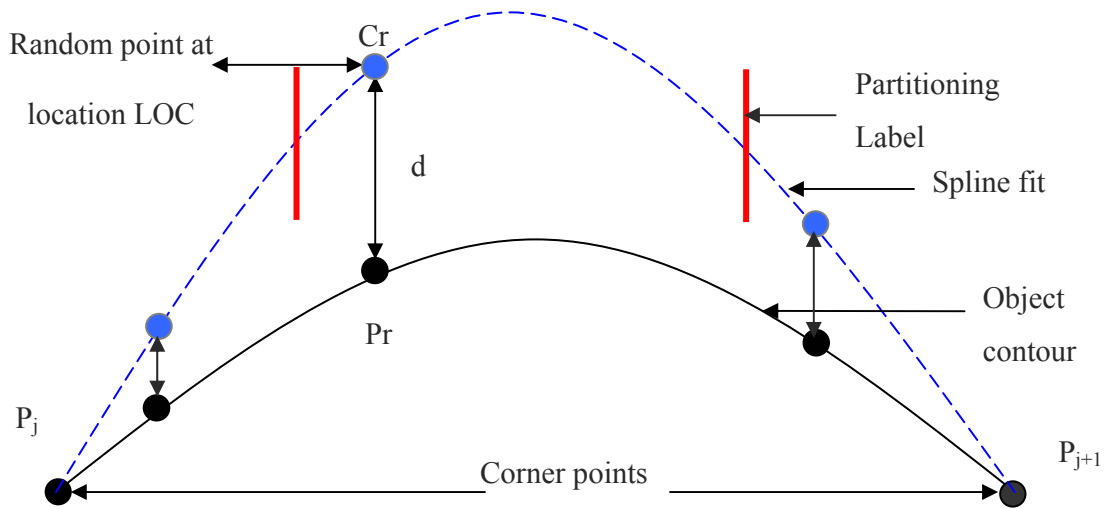


Figure 4.10 Calculation of random knot in Algorithm 4.5

The algorithm for fourth approach is written in (Algorithm 4.5).

Algorithm 4.5: Three Equal Spaces Segmented Random Points

Step 1: For each piece do the following

Step 2: Divide the piece under consideration into three equally spaced sub-segments

Step 3: Pick a random point on the spline curve for that sub-segment

Step 3.1: Calculate its Euclidean distance " d " with the corresponding point on the contour

Step 3.2: If $d > \varepsilon$

Step 3.2.1: Set the flag for another iteration, $flag = TRUE$

Step 3.2.2: With respect to its position, insert it into the list of corner points.

Step 4: If $flag = TRUE$ repeat step 1 otherwise stop the iteration.

4.3.2. Fuzzy Random Knot Insertion

The main idea behind this algorithm is to select a knot based upon its fuzzy membership value. This value defines the candidacy of a randomly selected point to be taken as a knot. Here we first take three points randomly on the spline curve as candidate knots. Next we delineate the criteria for assessing our problem in terms of fuzzy logic by defining our membership functions. After this, it is necessary to fuzzify all the input values. This is done to determine the degree to which the inputs belong to each of the appropriate fuzzy sets via membership functions. The resultant of this step is in fact the degree of membership in the qualifying linguistic set, which is between 0 and 1. These values are passed to the fuzzy operators so that we obtain one value which represents the

antecedent of our rule. Since we have different criteria to assess the fuzzy membership of a random point therefore we assign weights to each of these norms. In the final step we aggregate our rules in order to make a decision.

The fuzzy membership criteria are defined as under;

- Euclidean distance between the random point on spline curve and its corresponding point on the object contour.
- Positional distance between the random point and the left corner point on that piece.
- Positional distance between the random point and the right corner point on that piece.

The variables for fuzzy membership criteria are demonstrated in (Figure 4.11).

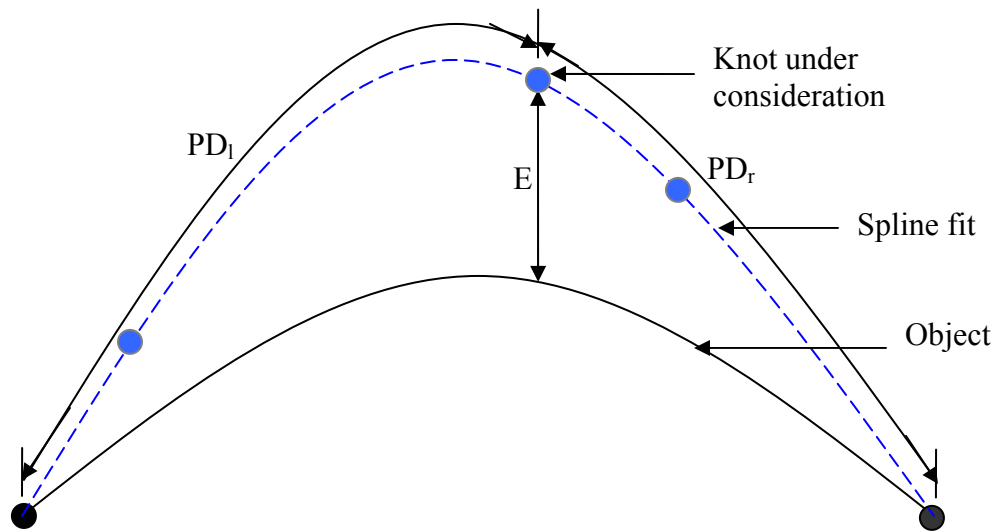


Figure 4.11 Fuzzy membership criteria

The fuzzification of the inputs is done as depicted in (Equation 4.17) and (Equation 4.19);

$$\text{Fuzzy Positional Distance} = \frac{PD_i}{\frac{|cp_{i+1} - cp_i|}{2}} \quad (4.17)$$

Where,

$$PD_i = \min(PD_l, PD_r) \quad (4.18)$$

PD_l and PD_r are the distances between random point and left corner point and right corner point respectively.

$$\text{Fuzzy Euclidean Distance} = \frac{E_j}{Sum_i} \quad (4.19)$$

Where,

E_j is the Euclidean distance $j=1,2,3$ and,

$$Sum_i = E_1 + E_2 + E_3 \quad (4.20)$$

Sum_i is the sum of Euclidean distances of all three random points in i^{th} piece.

Now we assume that the Euclidean distance factor produces more impact on the fuzzification process therefore we assign some weight w to it. Also we assign weight v

to positional distance factor. This implication method will ensure that in most of the cases a point with greater Euclidean distance value will be taken as knot. The relationship between w and v is shown in (Equation 4.21).

$$w > v \quad (4.21)$$

In the aggregation process we multiply these fuzzified values to get single representative result. This result helps us to choose just one point out of three points to take as knot. Although we get a point at this stage as a knot but we take it as candidate and before taking it as a knot we check if its Euclidean distance is within the range of ε .

The algorithm is as follows;

Algorithm 4.6: Fuzzy Random Knot Selection

Step 1: For each piece do the following

Step 1.1: Take three random points on the spline curve

Step 1.2: Calculate their respective Euclidean distances " E " with the corresponding points on the contour

Step 1.3: Calculate their respective positional distances with left and right corner points

Step 1.4: Calculate their fuzzy membership values and select one of them as knot k

Step 1.5: If $E_k > \varepsilon$, E_k is the Euclidean distance of selected point with corresponding point of the contour

Step 1.5.1: Set the flag for another iteration; $flag = TRUE$

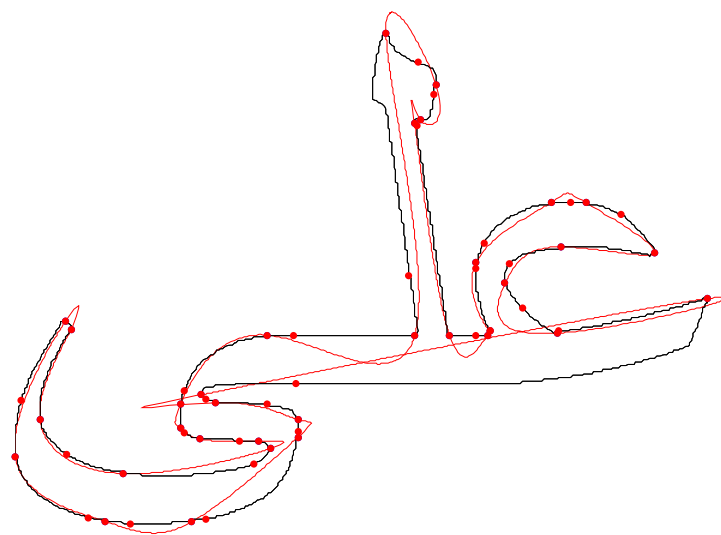
Step 1.5.2: With respect to its position, insert it into the list of corner points.

Step 2: If $flag = TRUE$ repeat the process otherwise stop the iteration.

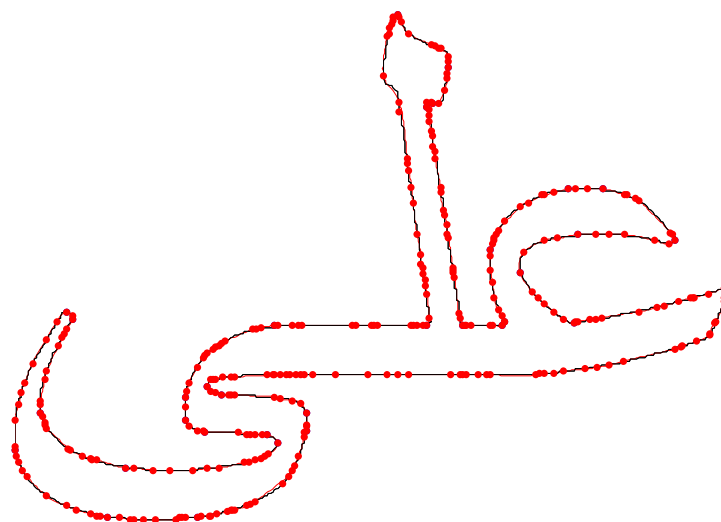
4.4 Results and Discussion

The evaluation results for all the algorithms as depicted in Table 4.1-Table 4.12 show that the number of knots inserted is independent of the total number of points on the contour and also they do not depend upon the complexity of the object shape. This is because of the true randomized nature of the algorithms. Also we can analyze that the total number of iterations are not greater in number thus showing the efficiency of the algorithm. The tradeoff is in terms of increased number of knots. A detailed analysis is presented after the compiled result set.

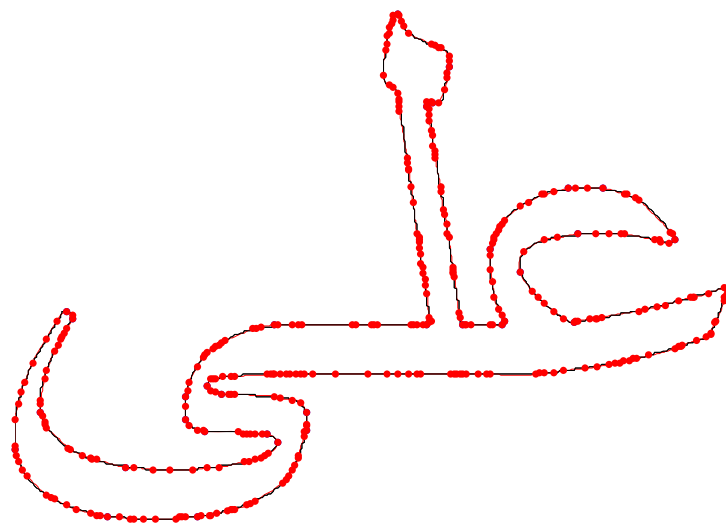
(Figure 4.12) to (Figure 4.14) shows the fitted curve over object contour, ‘Ali’, at different iterations for algorithm 4.1 at threshold values of 1,2 and 3 respectively.



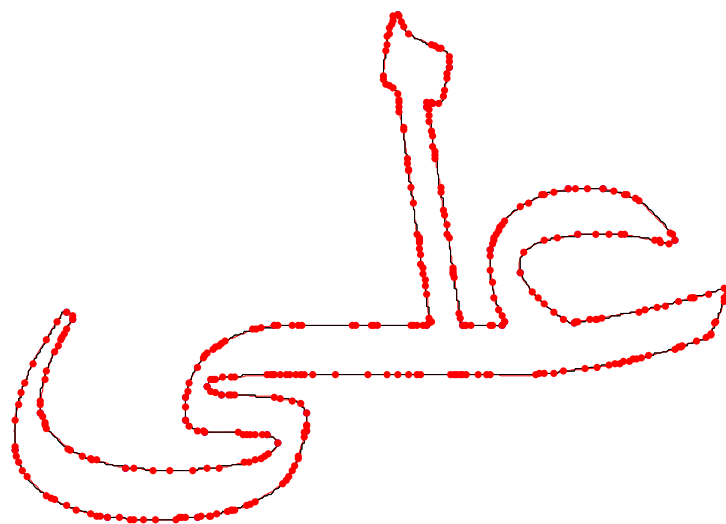
(a) At iteration = 1



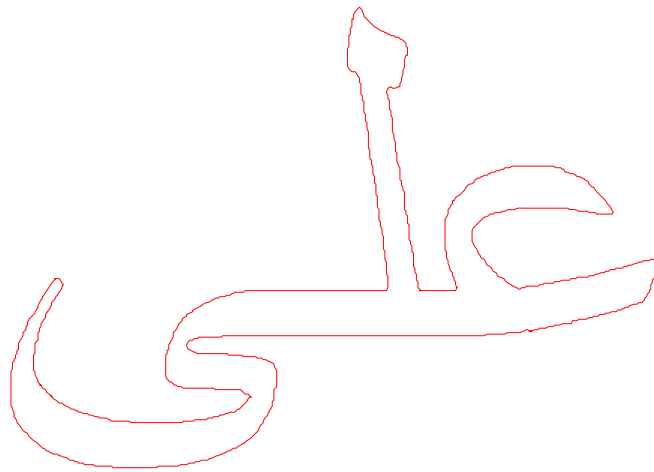
(b) At iteration = 10



(c) At last iteration = 20

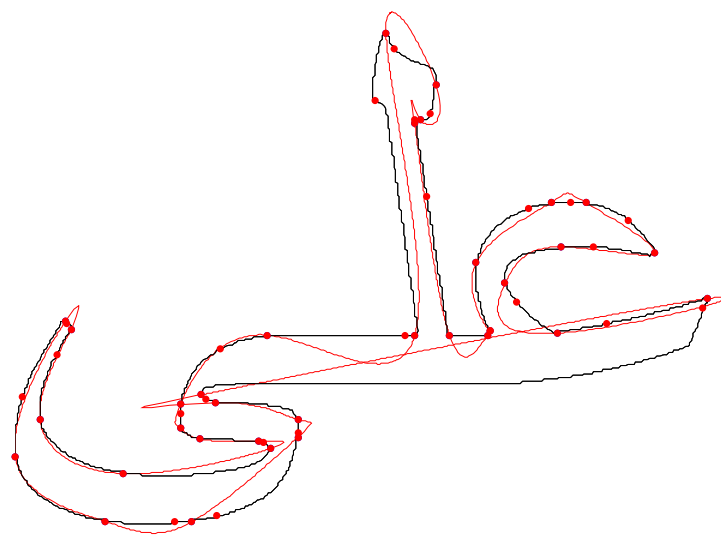


(d) At last iteration = 30

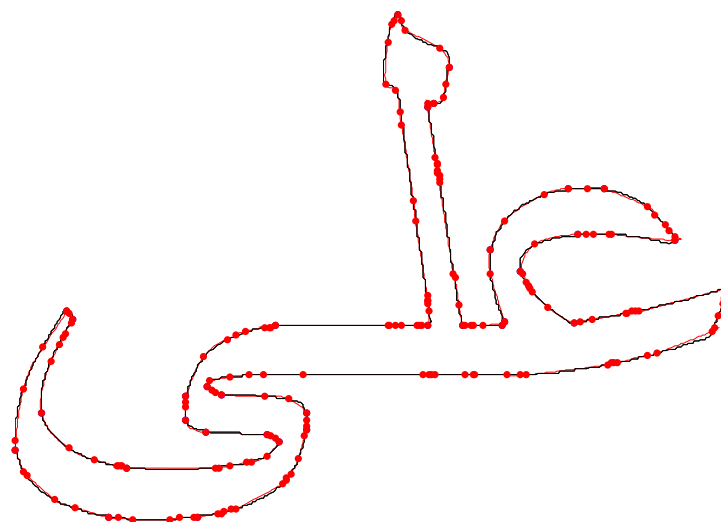


(e) Approximated spline for Arabic word “Ali”

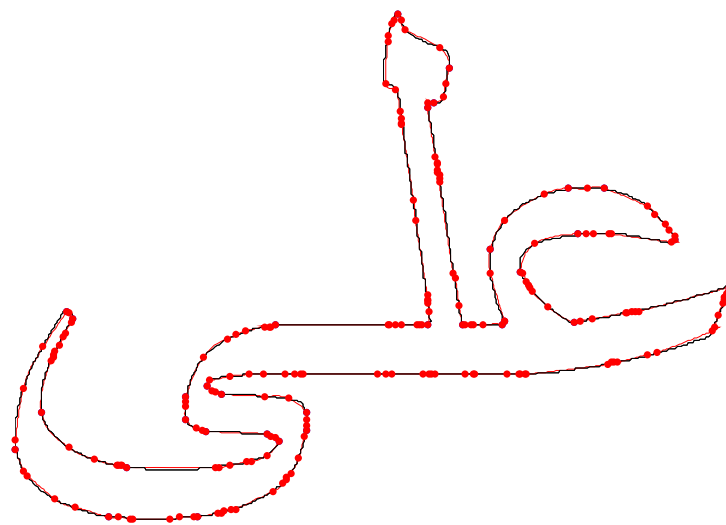
**Figure 4.12 Algorithm 4.1: Demonstration of spline fitting at each iteration using
threshold =1**



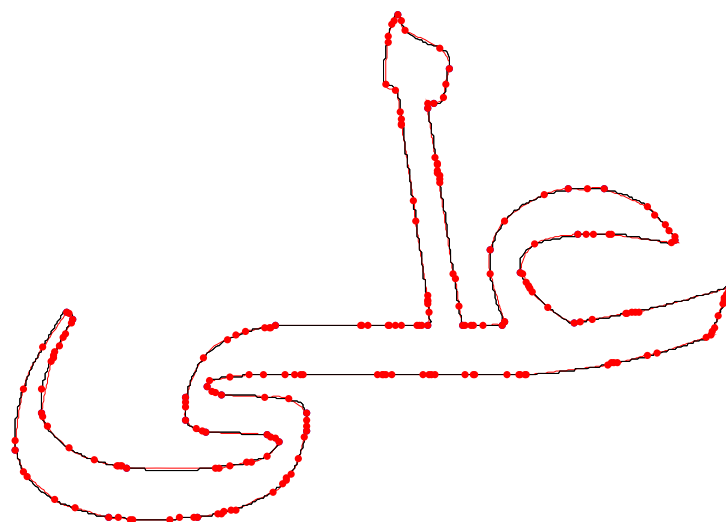
(a) At iteration = 1



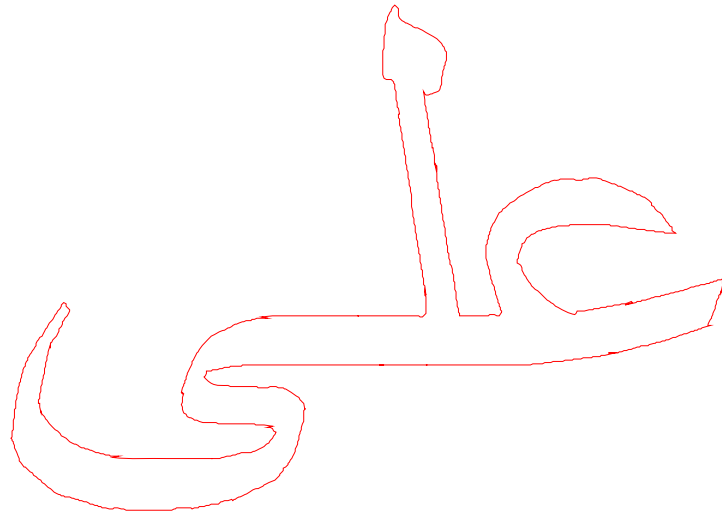
(b) At iteration = 10



(c) At last iteration = 20

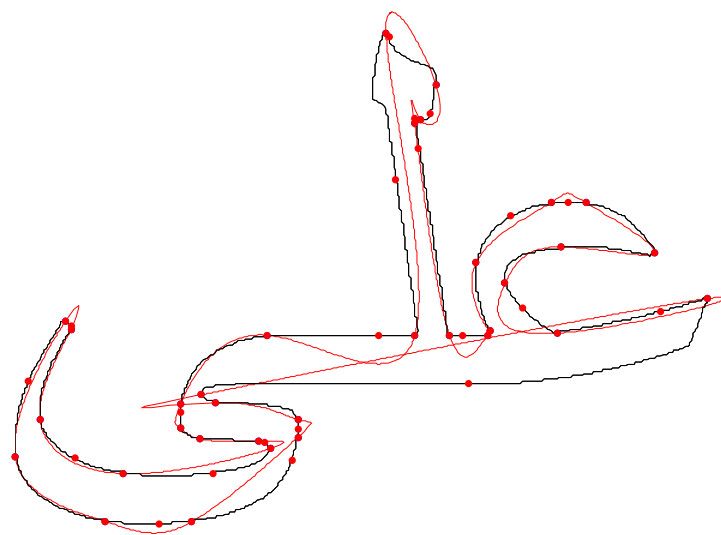


(d) At last iteration = 30

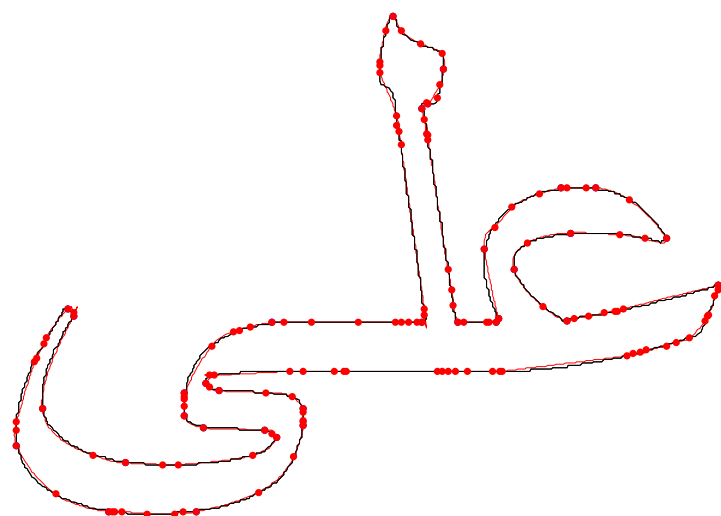


(e) Approximated spline for Arabic word “Ali”

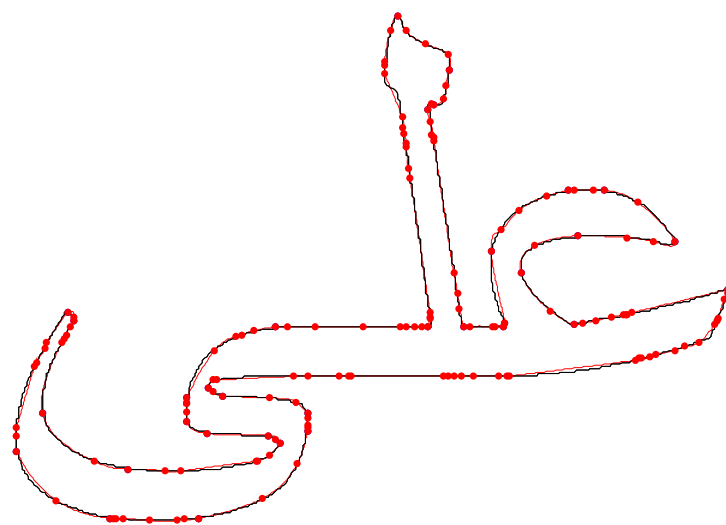
**Figure 4.13 Algorithm 4.1: Demonstration of spline fitting at each iteration using
threshold =2**



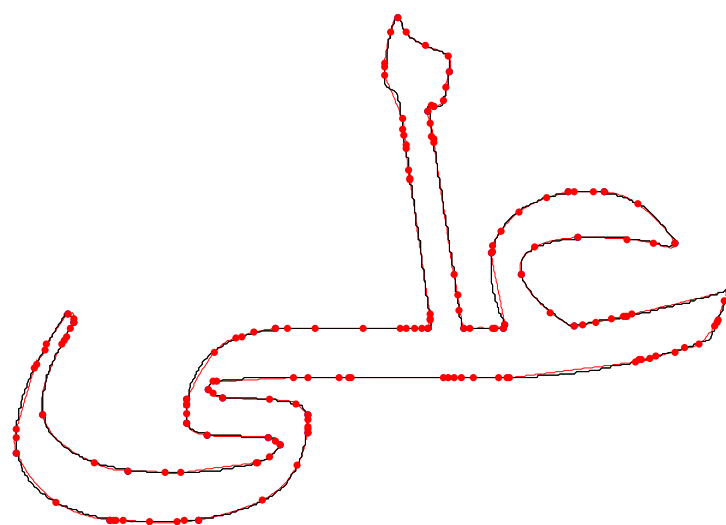
(a) At iteration = 1



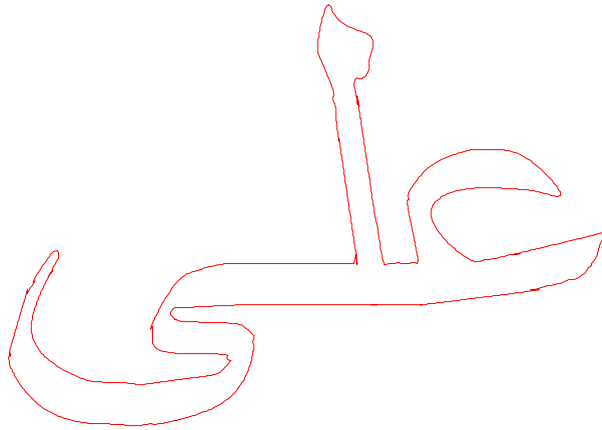
(b) At iteration = 10



(c) At last iteration = 20



(d) At last iteration = 30



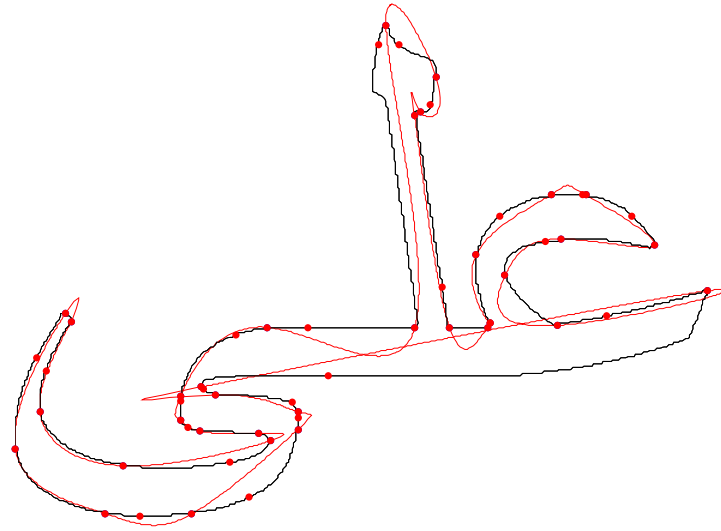
(e) Approximated spline for Arabic word “Ali”

Figure 4.14 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =3

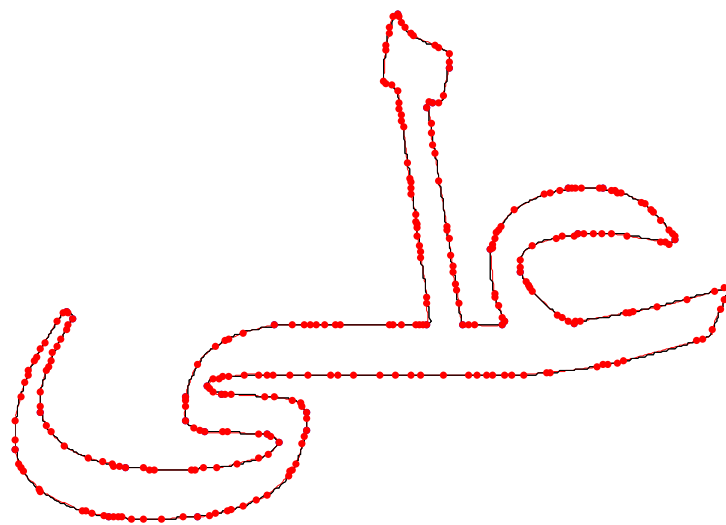
Table 4.1 Evaluation of algorithm 4.1 for Arabic word “Ali”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	652	30	531	1
1644	33	343	30	64	2
1644	33	197	30	24	3

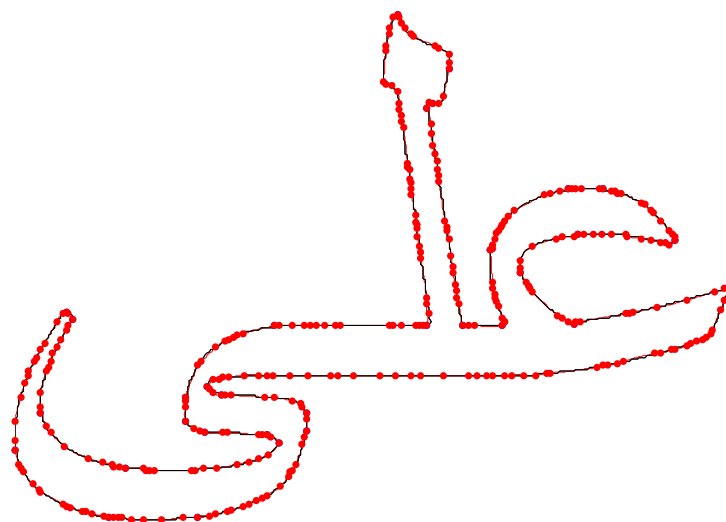
(Figure 4.15) to (Figure 4.17) shows the fitted curve over object contour at different iterations for algorithm 4.2 at threshold values of 1,2 and 3 respectively.



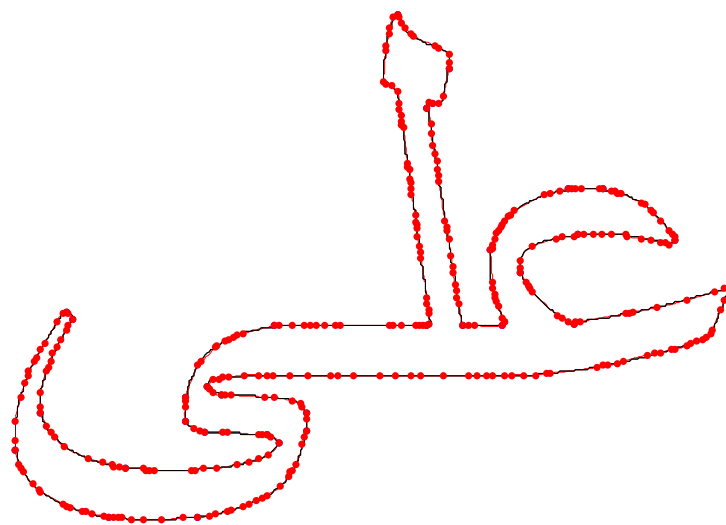
(a) At iteration = 1



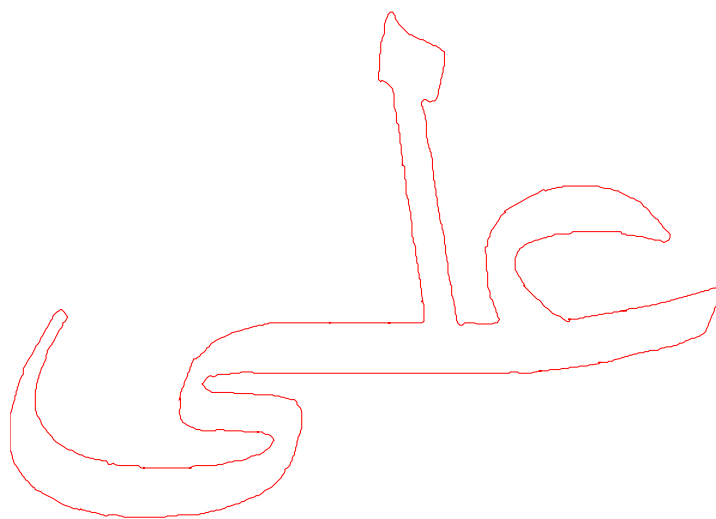
(b) At iteration = 10



(c) At last iteration = 20

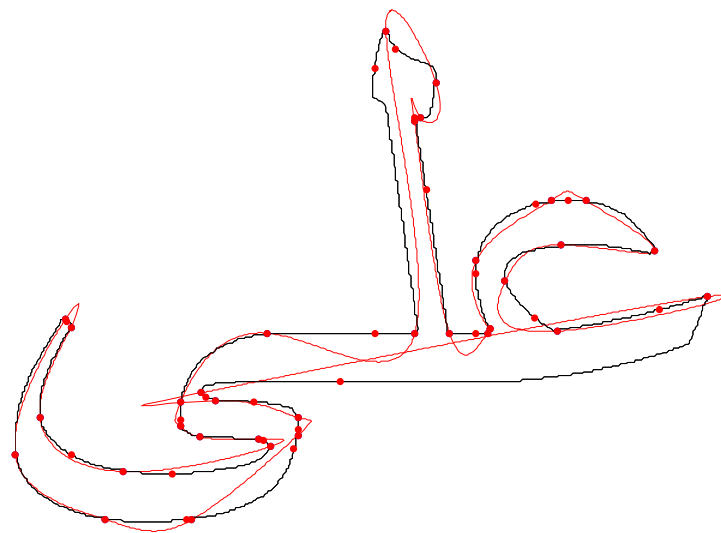


(d) At last iteration = 30

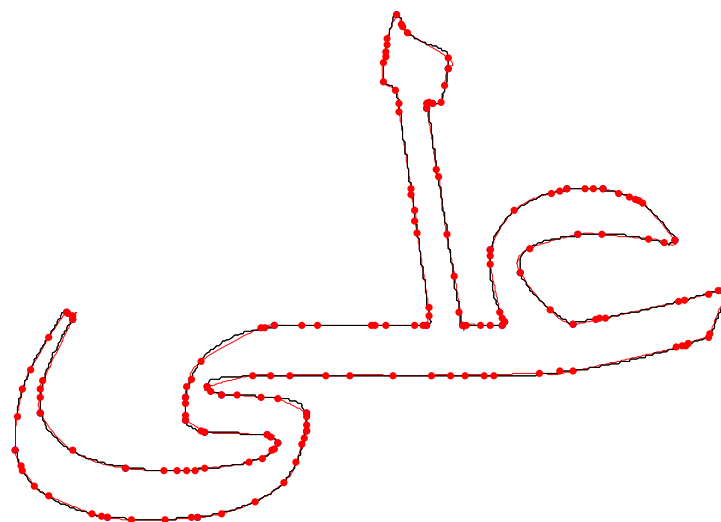


(e) Approximated spline for Arabic word “Ali”

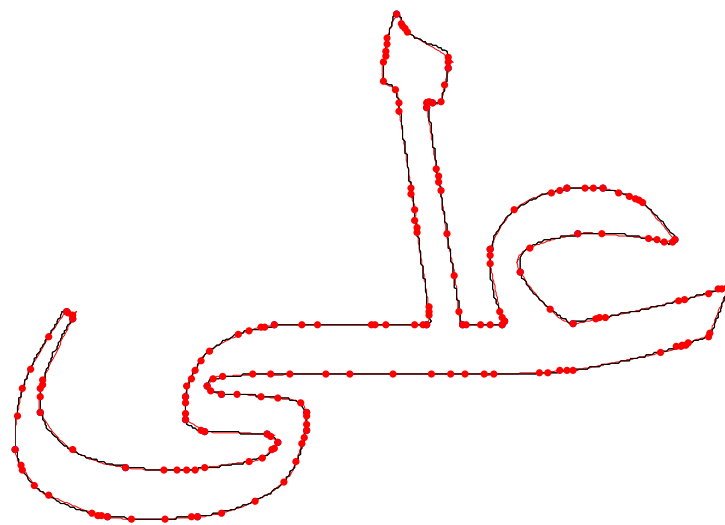
**Figure 4.15 Algorithm 4.2: Demonstration of spline fitting at each iteration using
threshold =1**



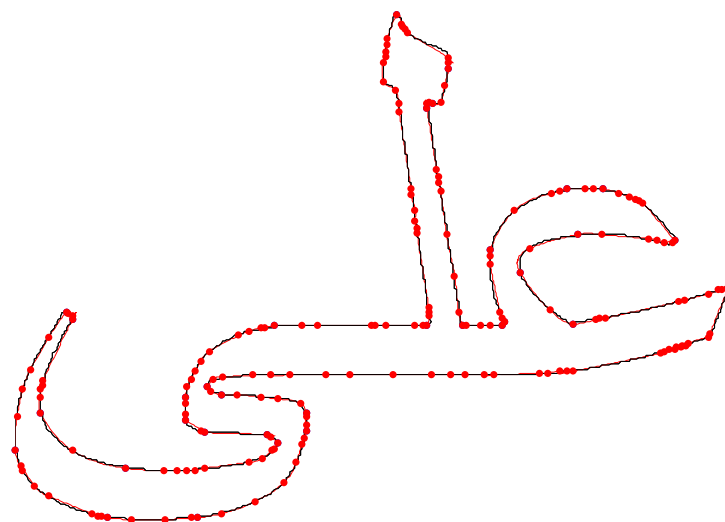
(a) At iteration = 1



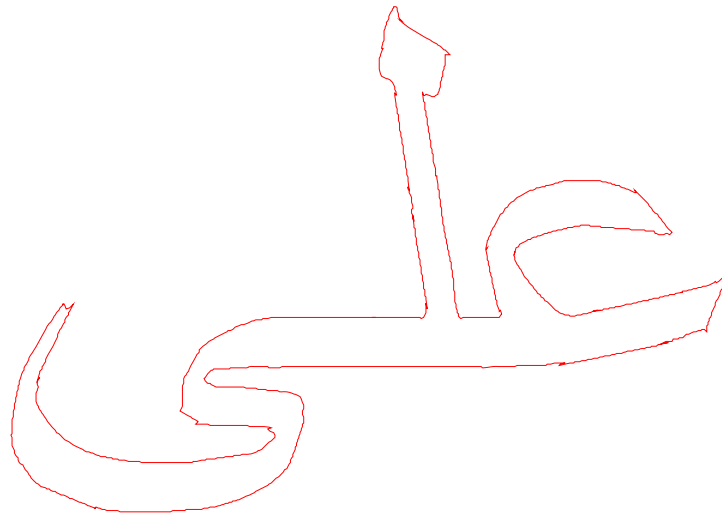
(b) At iteration = 10



(c) At last iteration = 20

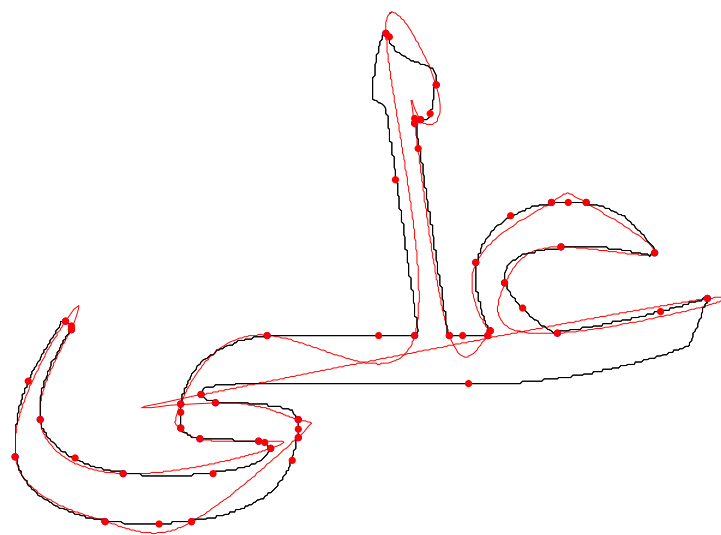


(d) At last iteration = 30

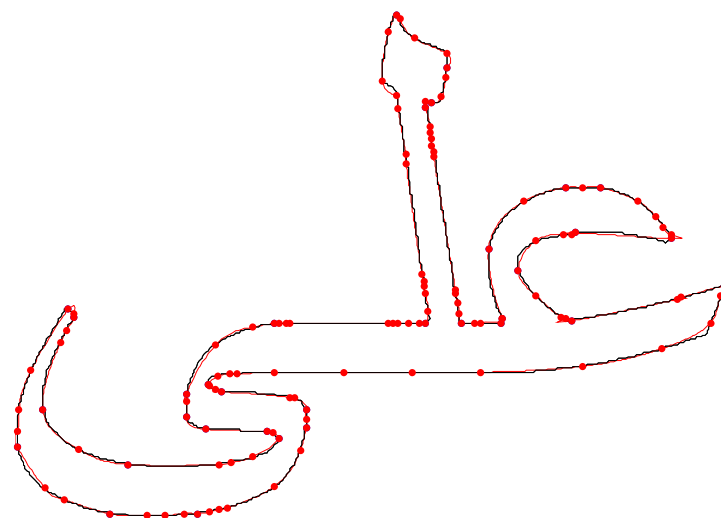


(e) Approximated spline for Arabic word “Ali”

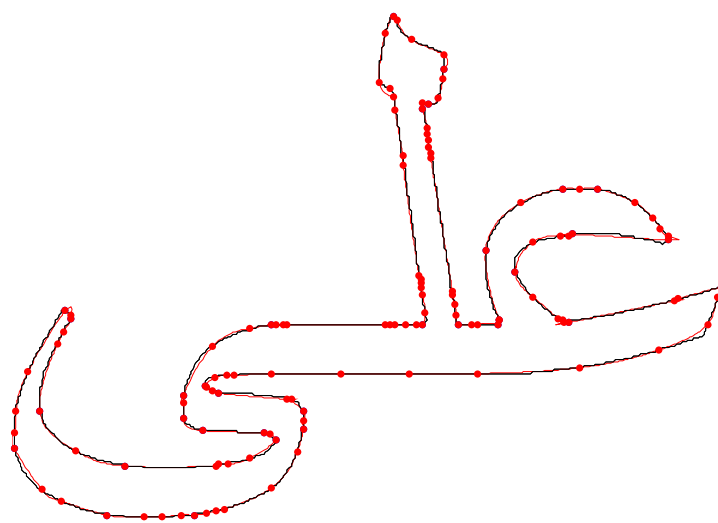
**Figure 4.16 Algorithm 4.2: Demonstration of spline fitting at each iteration using
threshold =2**



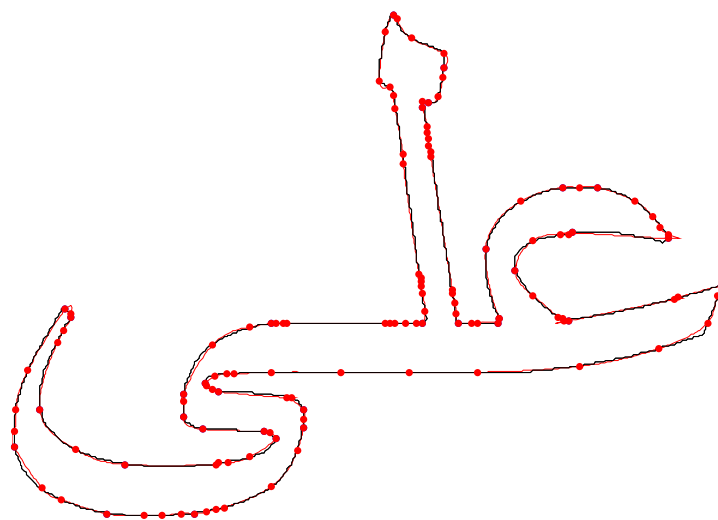
(a) At iteration = 1



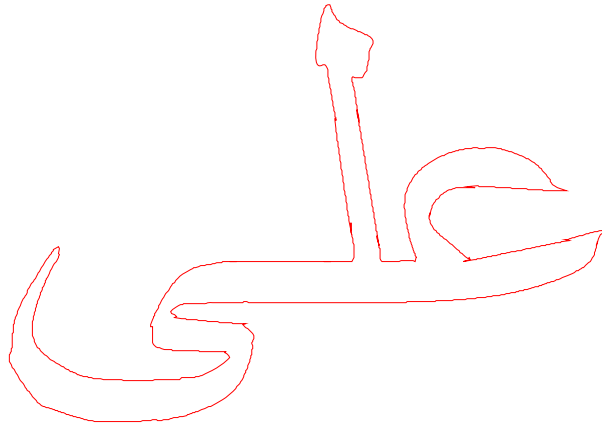
(b) At iteration = 10



(c) At last iteration = 20



(d) At last iteration = 30



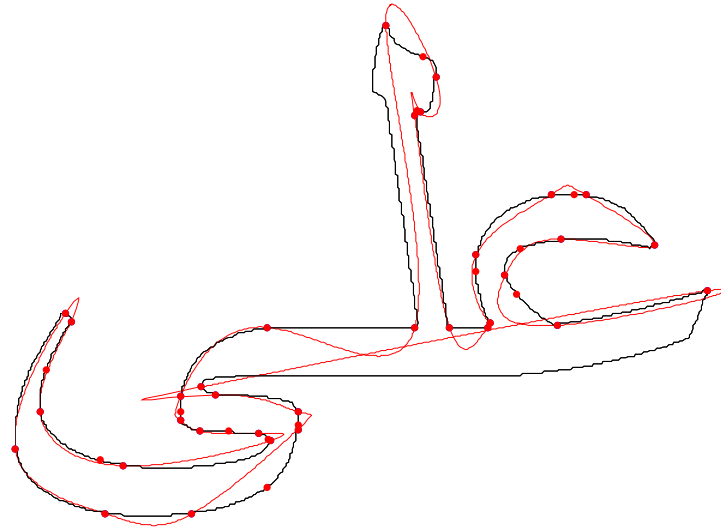
(e) Approximated spline for Arabic word “Ali”

Figure 4.17 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =3

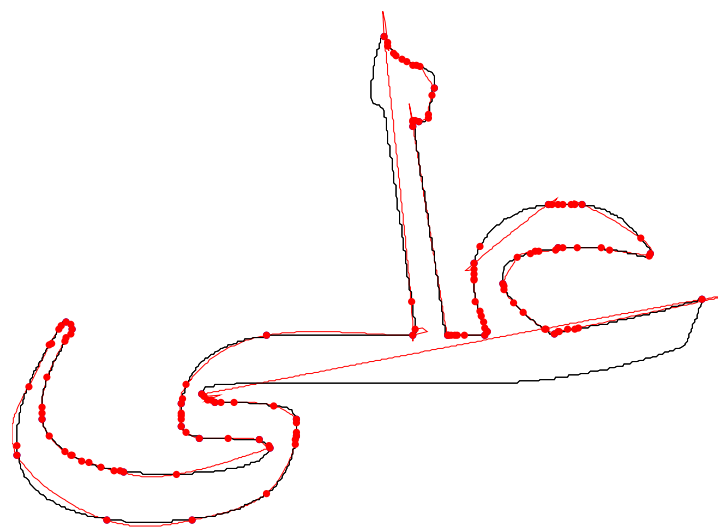
Table 4.2 Evaluation of algorithm 4.2 for Arabic word “Ali”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	308	30	112	1
1644	33	160	30	29	2
1644	33	96	30	22	3

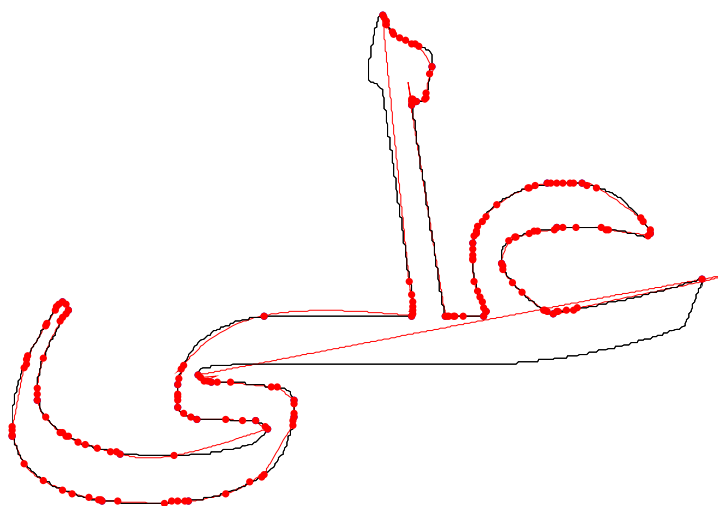
(Figure 4.18) to (Figure 4.20) shows the fitted curve over object contour at different iterations for algorithm 4.3 at threshold values of 1,2 and 3 respectively.



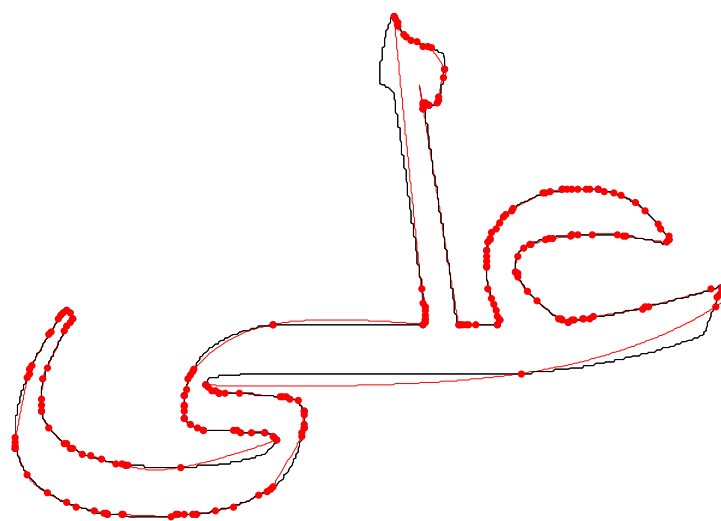
(a) At iteration = 1



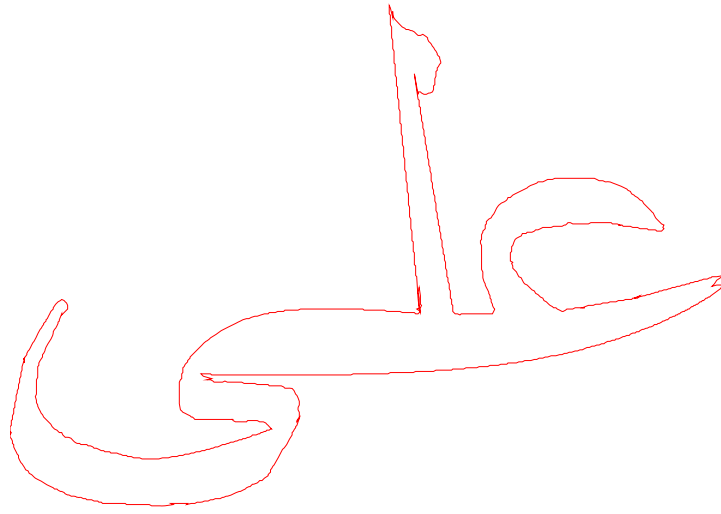
(b) At iteration = 10



(c) At last iteration = 20

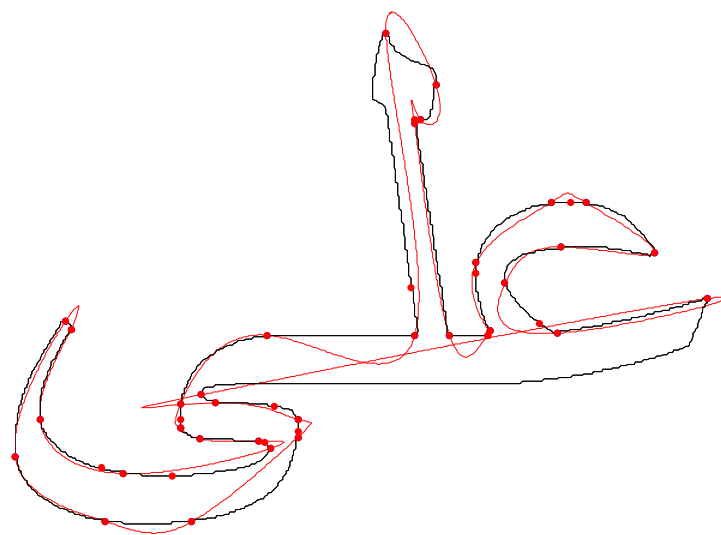


(d) At last iteration = 30

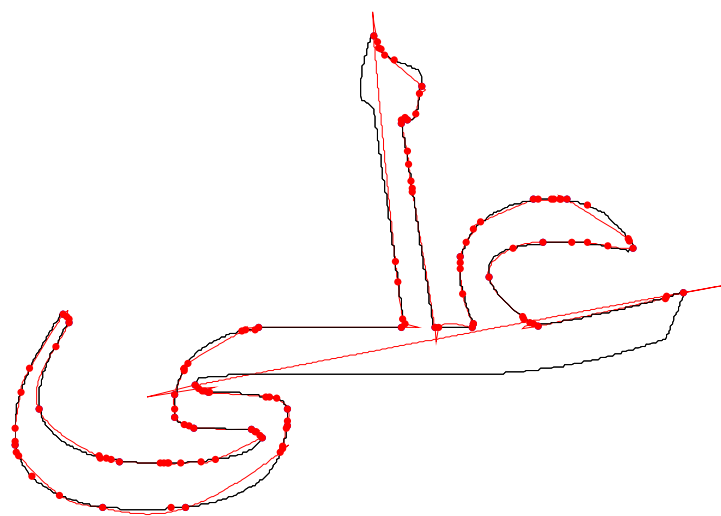


(e) Approximated spline for Arabic word “Ali”

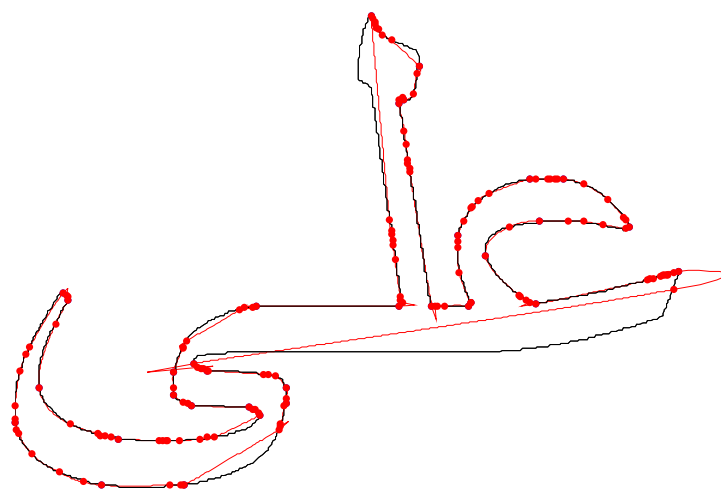
**Figure 4.18 Algorithm 4.3: Demonstration of spline fitting at each iteration using
threshold =1**



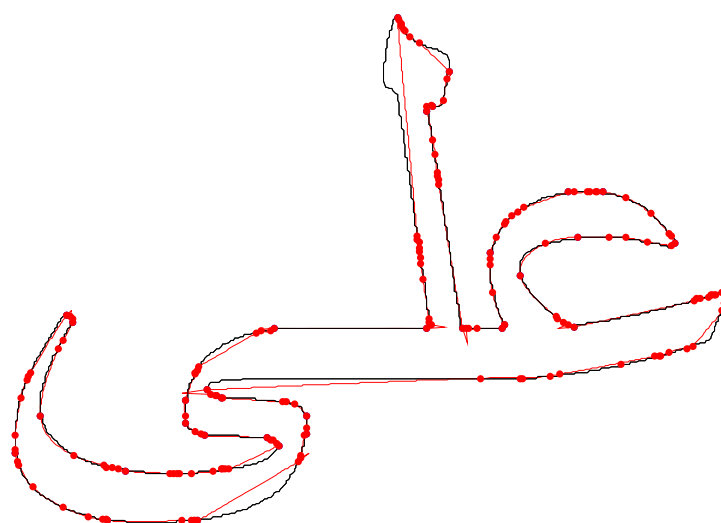
(a) At iteration = 1



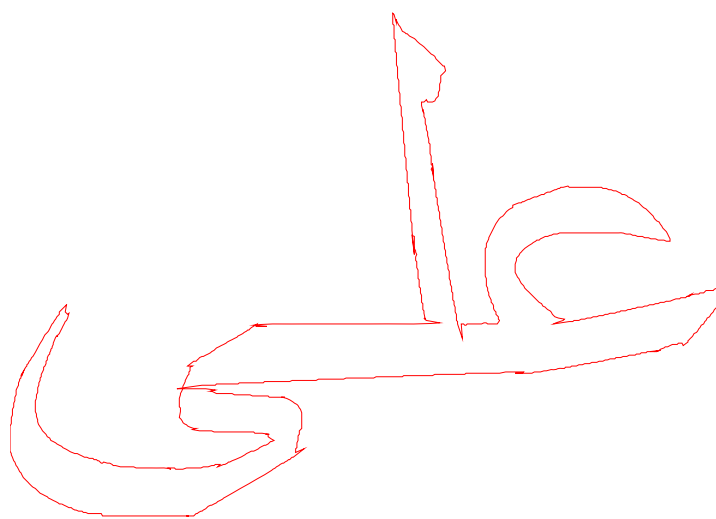
(b) At iteration = 10



(c) At last iteration = 20

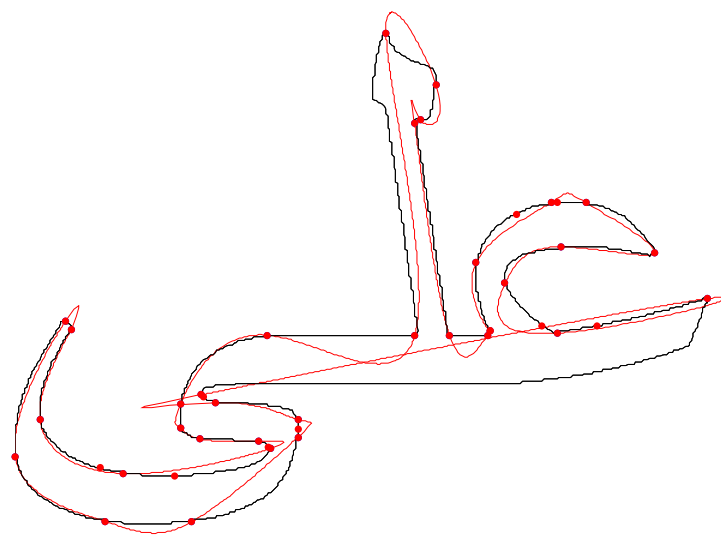


(d) At last iteration = 30

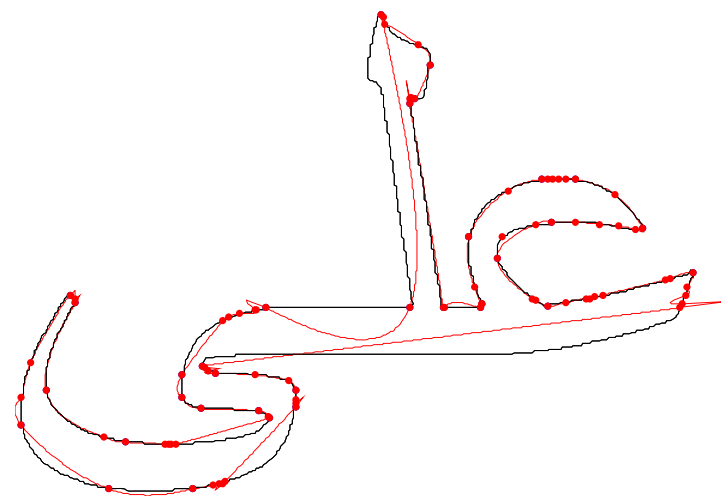


(e) Approximated spline for Arabic word “Ali”

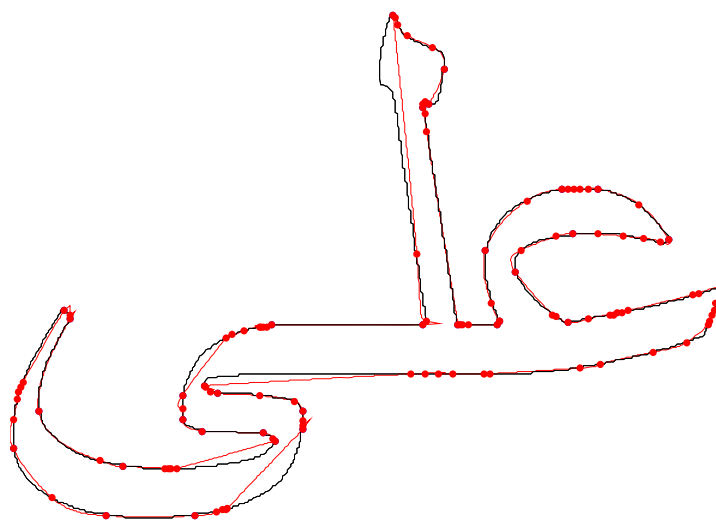
**Figure 4.19 Algorithm 4.3: Demonstration of spline fitting at each iteration using
threshold =2**



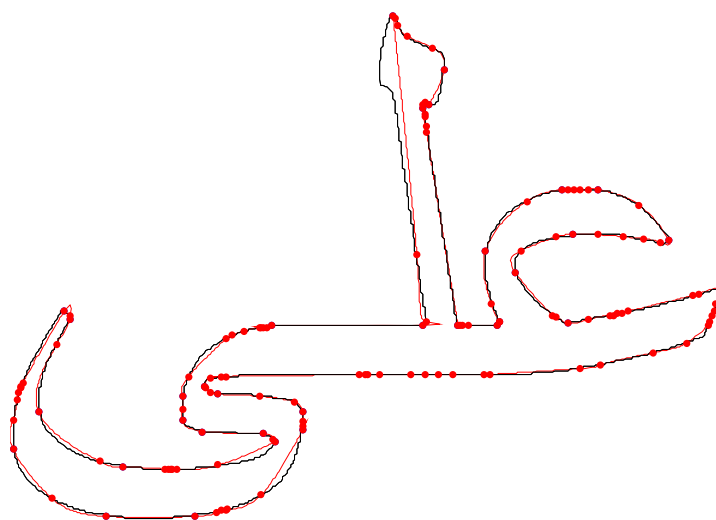
(a) At iteration = 1



(b) At iteration = 10



(c) At last iteration = 20



(d) At last iteration = 30



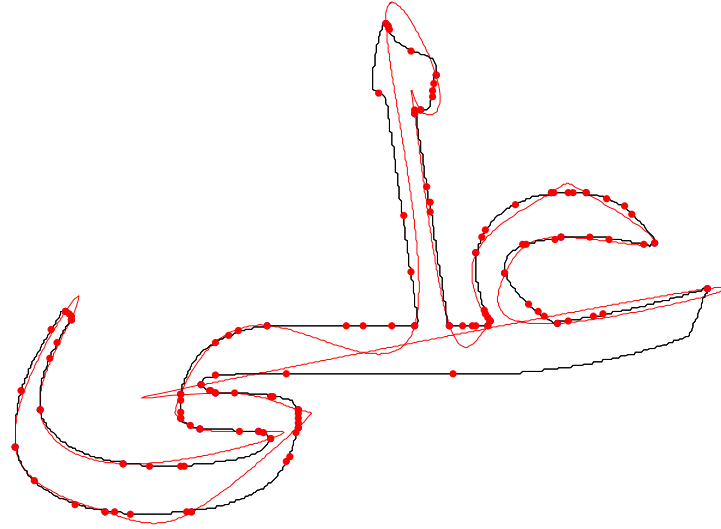
(e) Approximated spline for Arabic word “Ali”

Figure 4.20 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =3

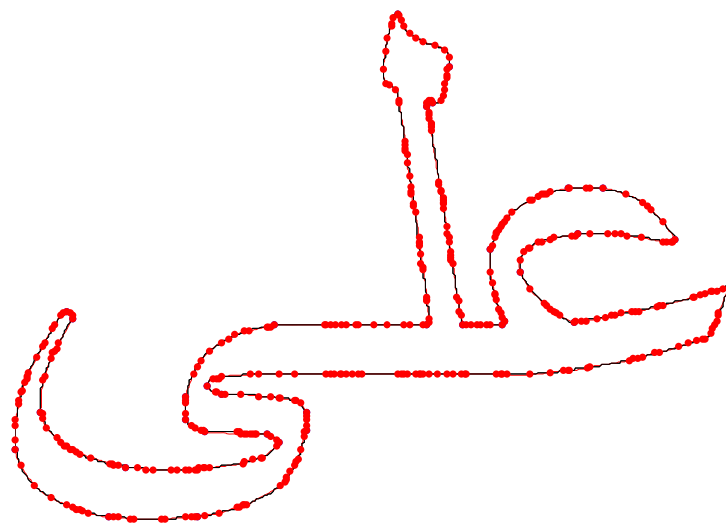
Table 4.3 Evaluation of algorithm 4.3 for Arabic word “Ali”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	183	30	28	1
1644	33	144	30	26	2
1644	33	98	30	19	3

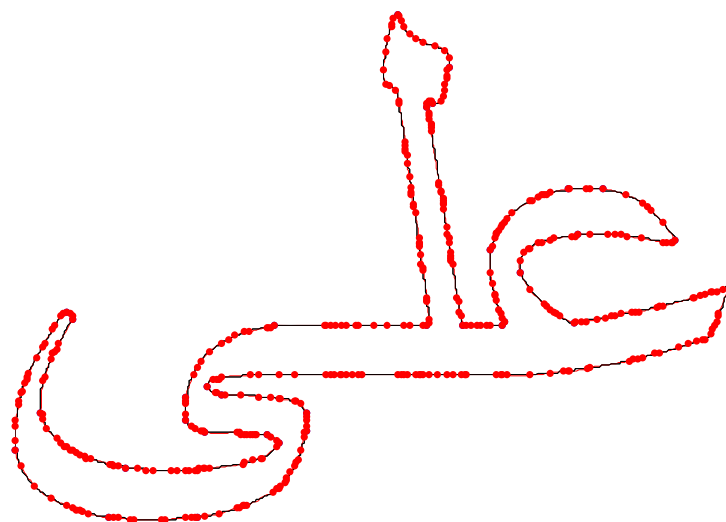
(Figure 4.21) to (Figure 4.23) shows the fitted curve over object contour at different iterations for algorithm 4.4 at threshold values of 1,2 and 3 respectively.



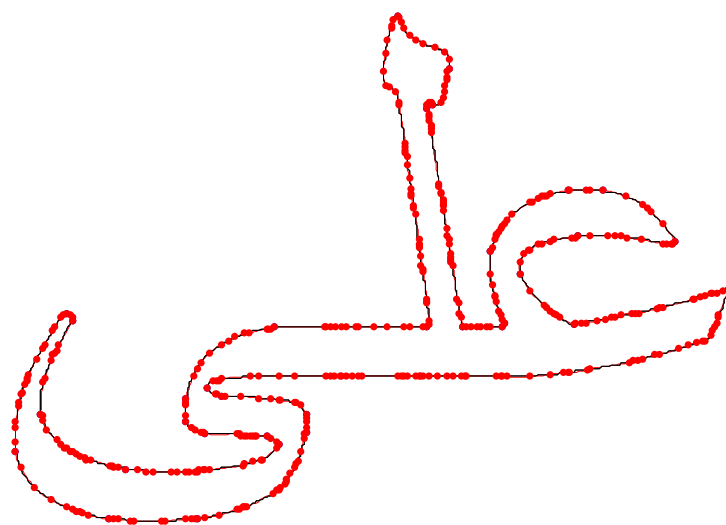
(a) At iteration = 1



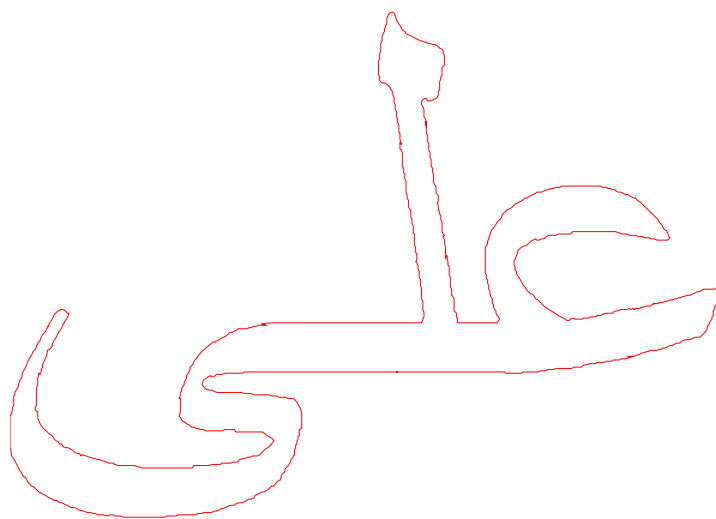
(b) At iteration = 10



(c) At last iteration = 20

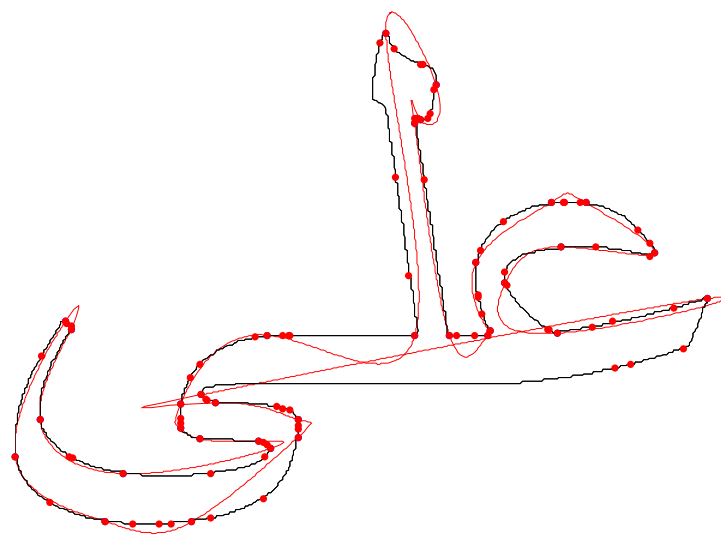


(d) At last iteration = 30

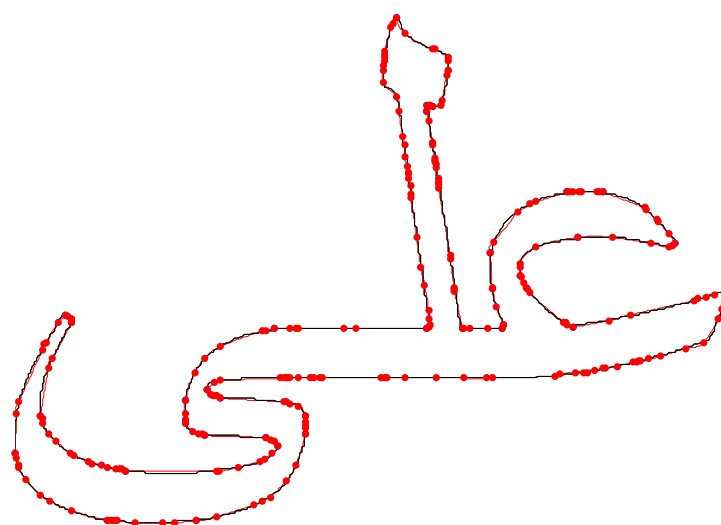


(e) Approximated spline for Arabic word “Ali”

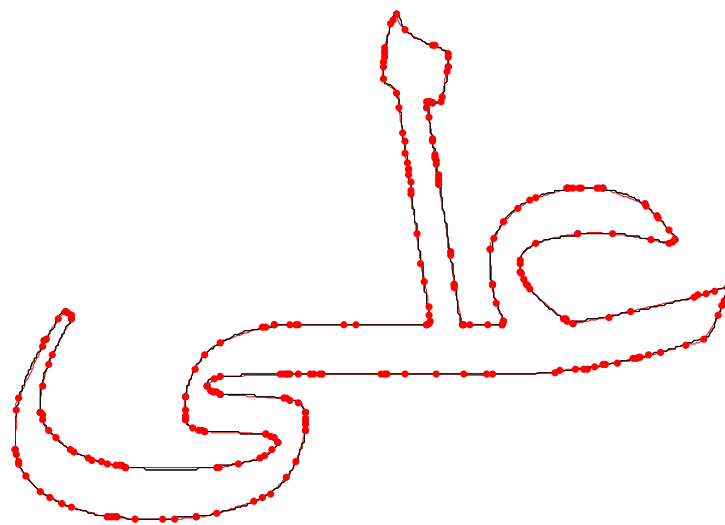
**Figure 4.21 Algorithm 4.4: Demonstration of spline fitting at each iteration using
threshold =1**



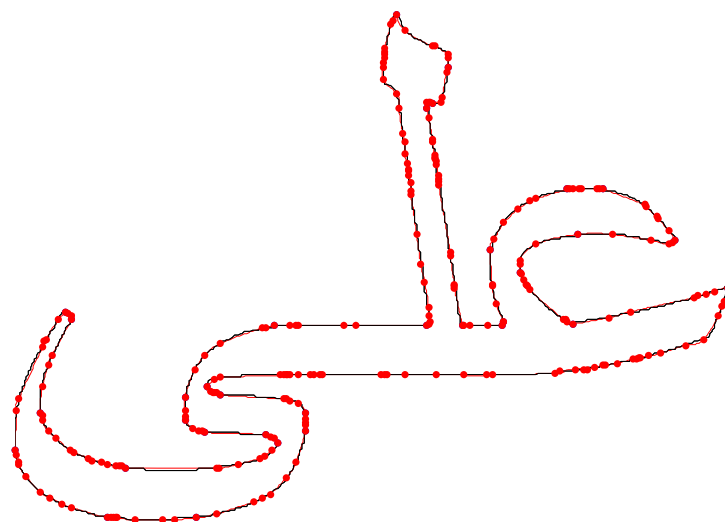
(a) At iteration = 1



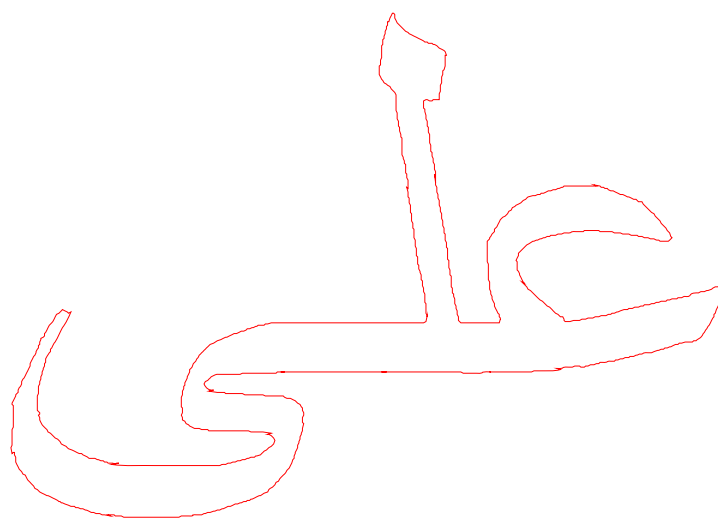
(b) At iteration = 10



(c) At last iteration = 20

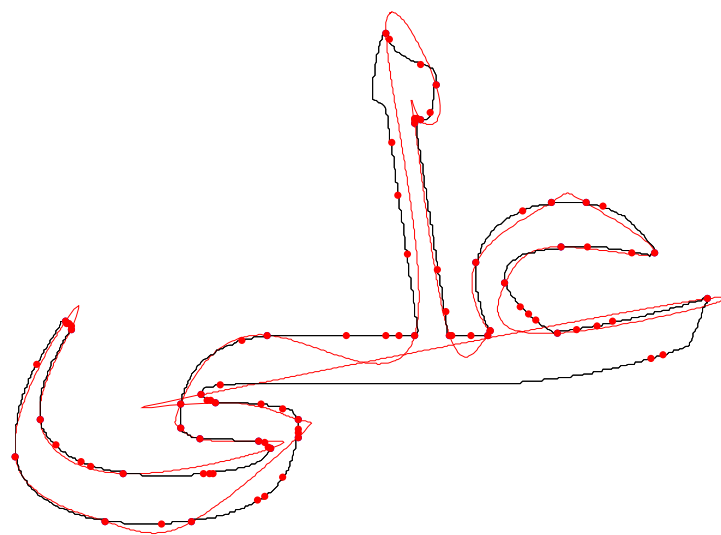


(d) At last iteration = 30

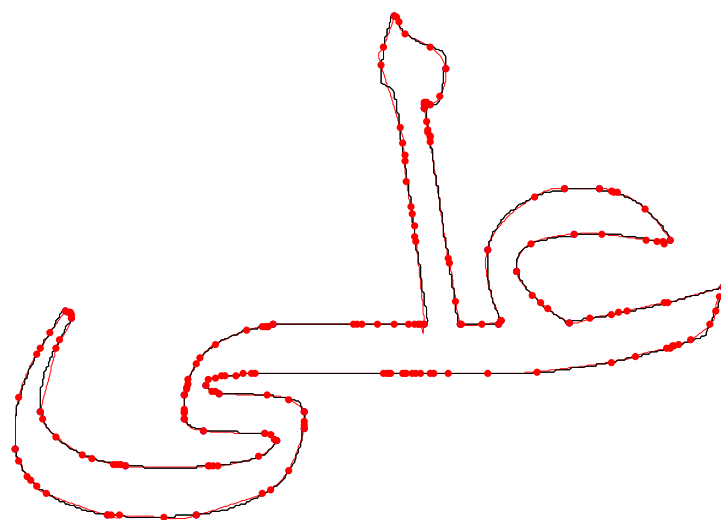


(e) Approximated spline for Arabic word “Ali”

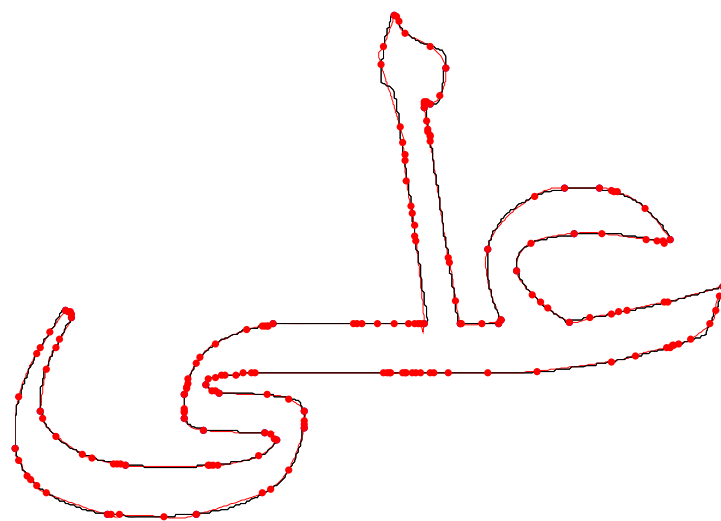
**Figure 4.22 Algorithm 4.4: Demonstration of spline fitting at each iteration using
threshold =2**



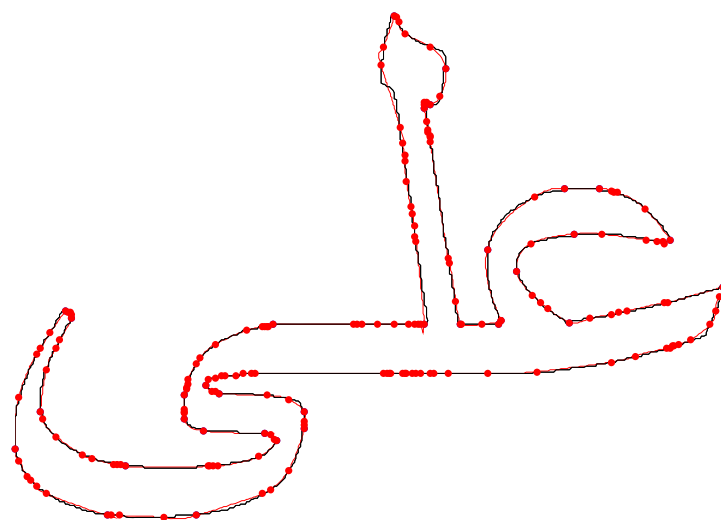
(a) At iteration = 1



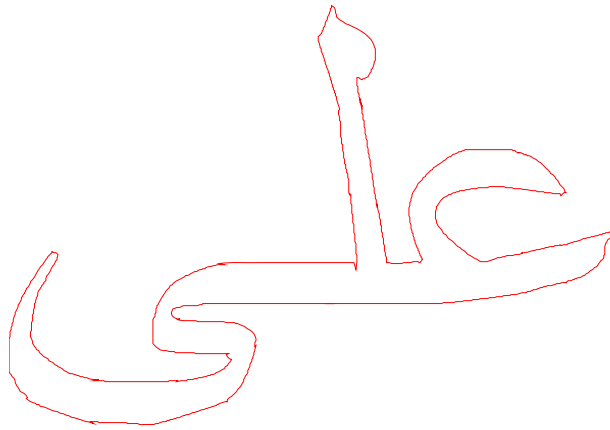
(b) At iteration = 10



(c) At last iteration = 20



(d) At last iteration = 30



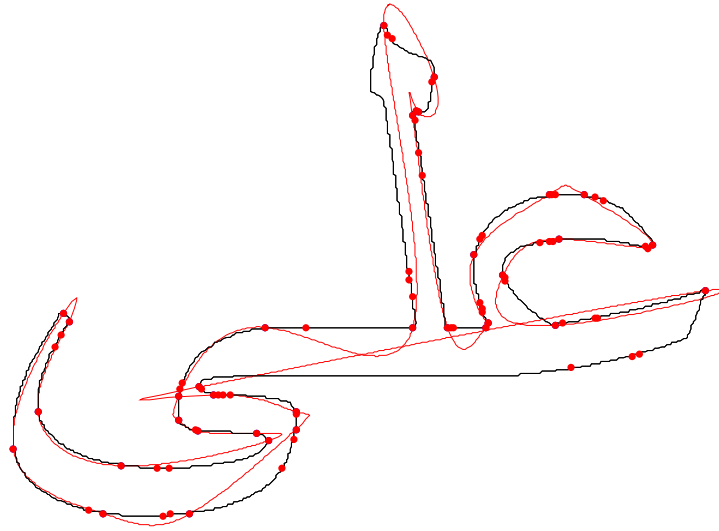
(e) Approximated spline for Arabic word “Ali”

Figure 4.23 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =3

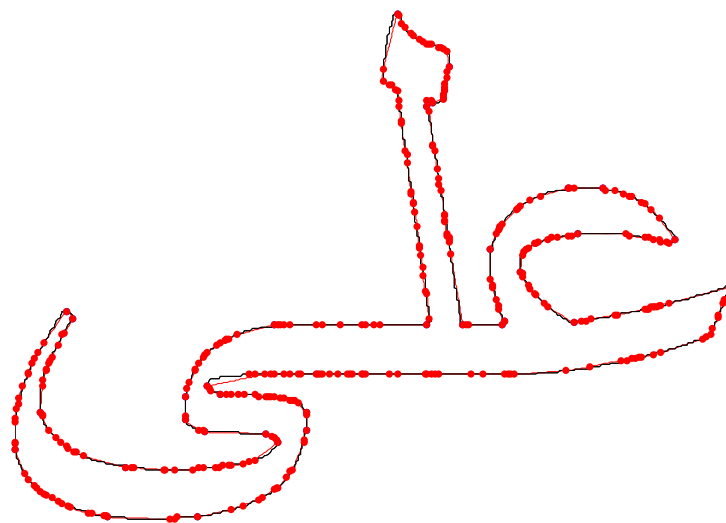
Table 4.4 Evaluation of algorithm 4.4 for Arabic word “Ali”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	1334	30	5219	1
1644	33	664	30	229	2
1644	33	422	30	65	3

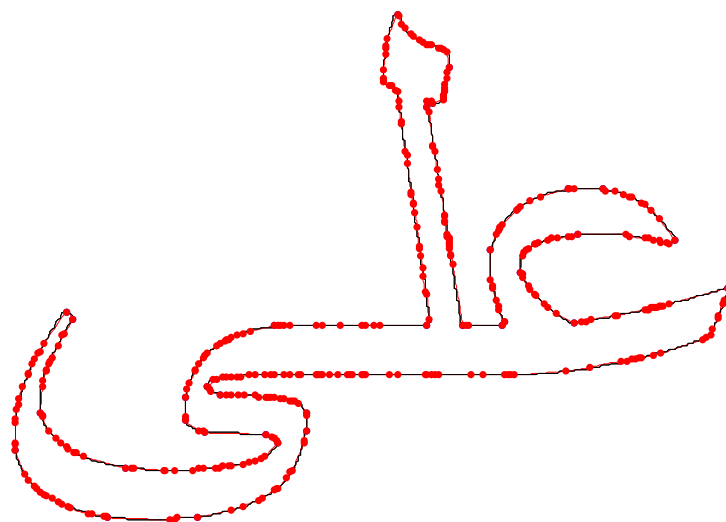
(Figure 4.24) to (Figure 4.26) shows the fitted curve over object contour at different iterations for algorithm 4.5 at threshold values of 1,2 and 3 respectively.



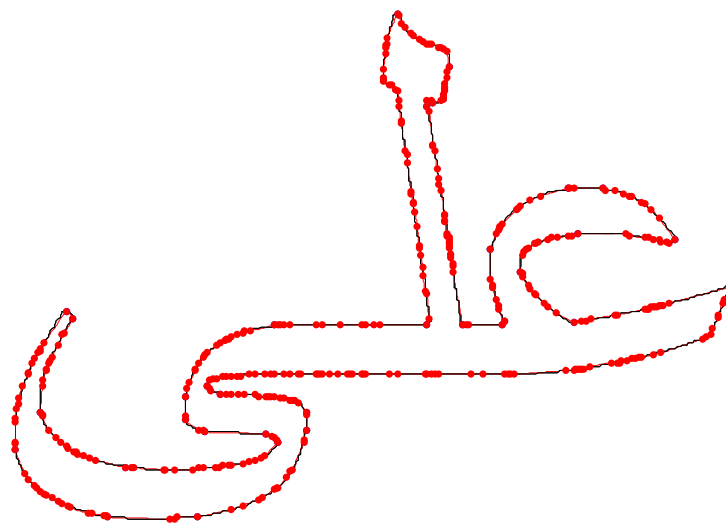
(a) At iteration = 1



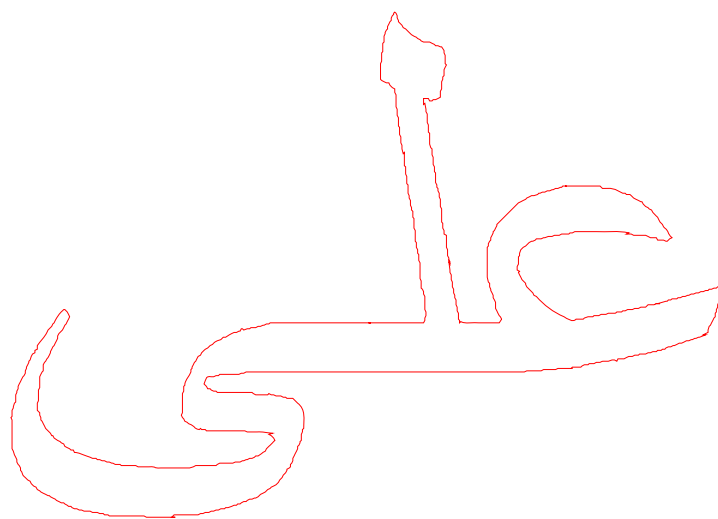
(b) At iteration = 10



(c) At last iteration = 20

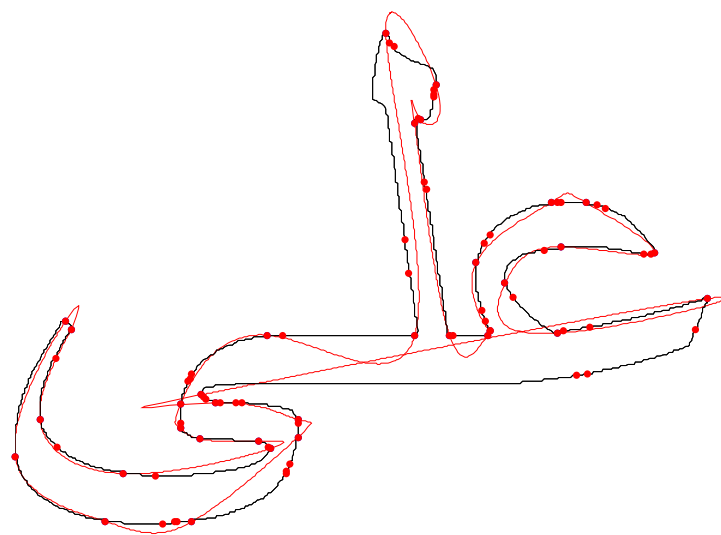


(d) At last iteration = 30

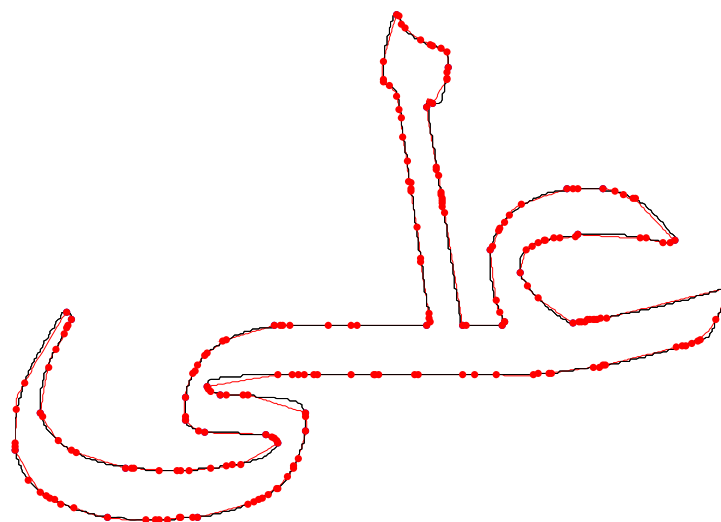


(e) Approximated spline for Arabic word “Ali”

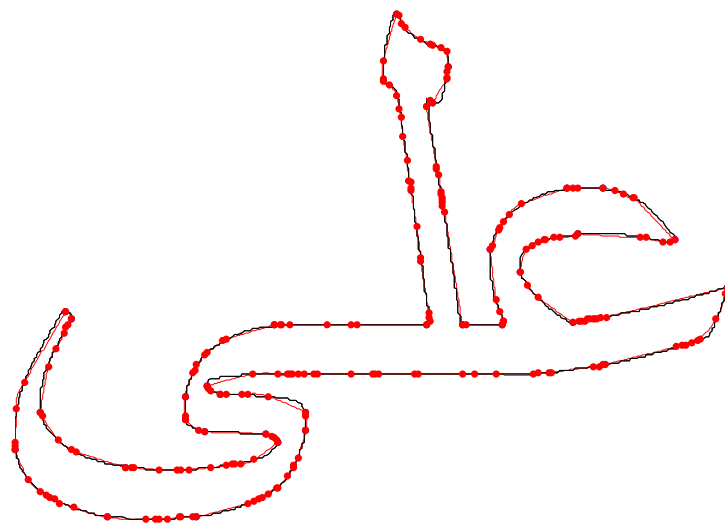
**Figure 4.24 Algorithm 4.5: Demonstration of spline fitting at each iteration using
threshold =1**



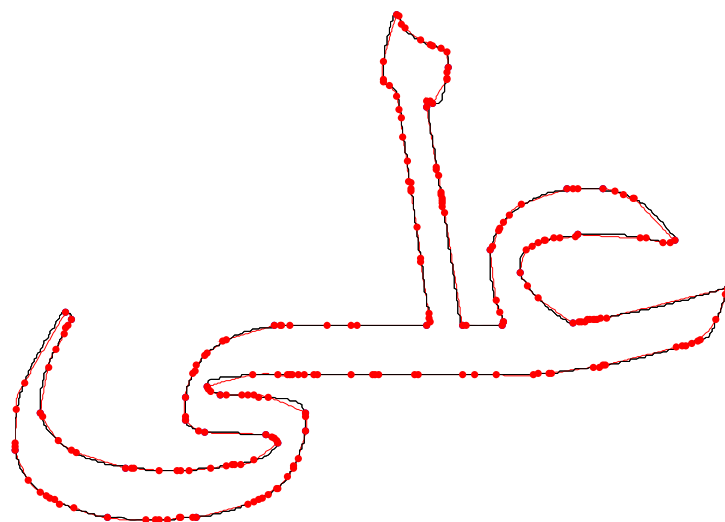
(a) At iteration = 1



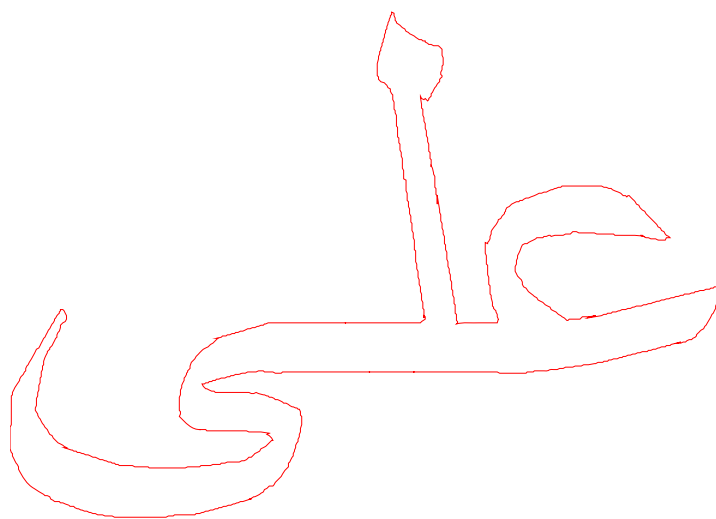
(b) At iteration = 10



(c) At last iteration = 20

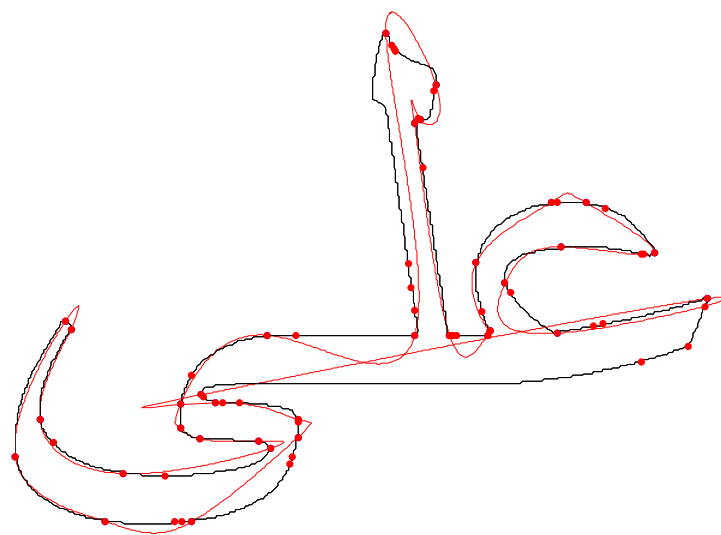


(d) At last iteration = 30

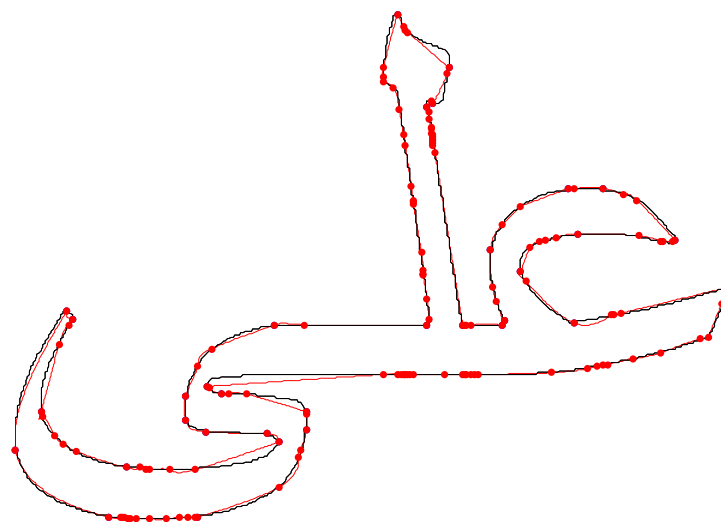


(e) Approximated spline for Arabic word “Ali”

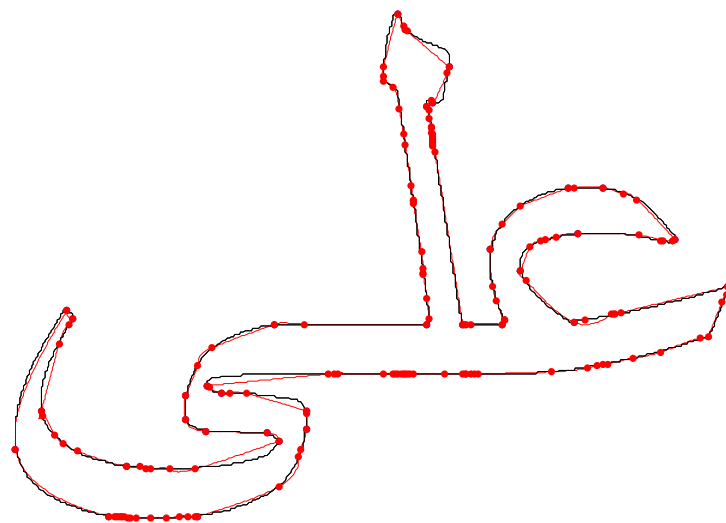
**Figure 4.25 Algorithm 4.5: Demonstration of spline fitting at each iteration using
threshold =2**



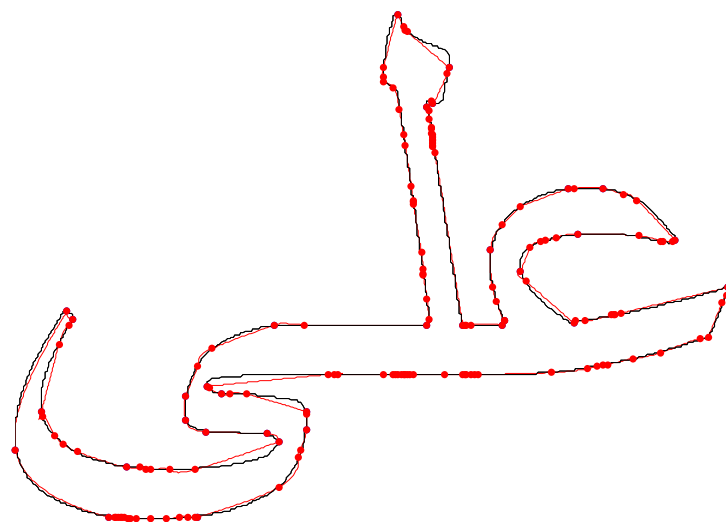
(a) At iteration = 1



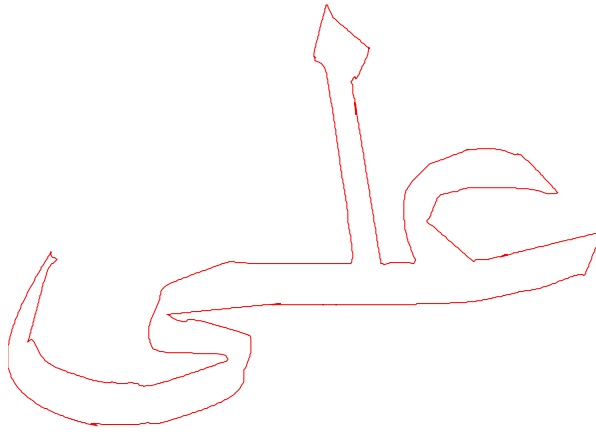
(b) At iteration = 10



(c) At last iteration = 20



(d) At last iteration = 30



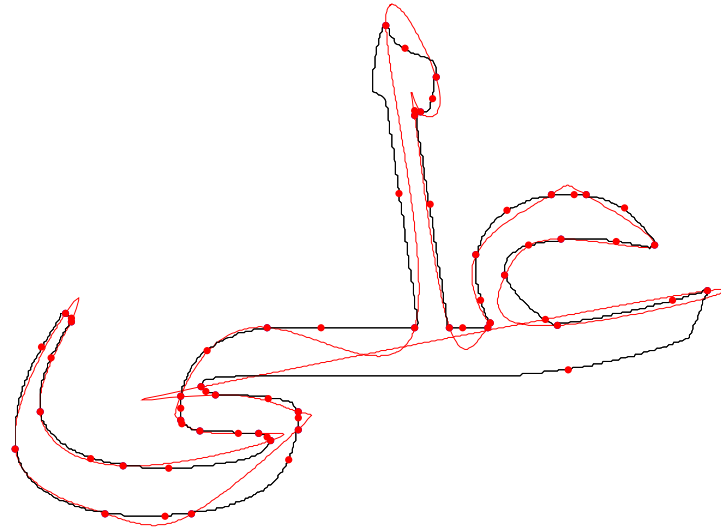
(e) Approximated spline for Arabic word “Ali”

Figure 4.26 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =3

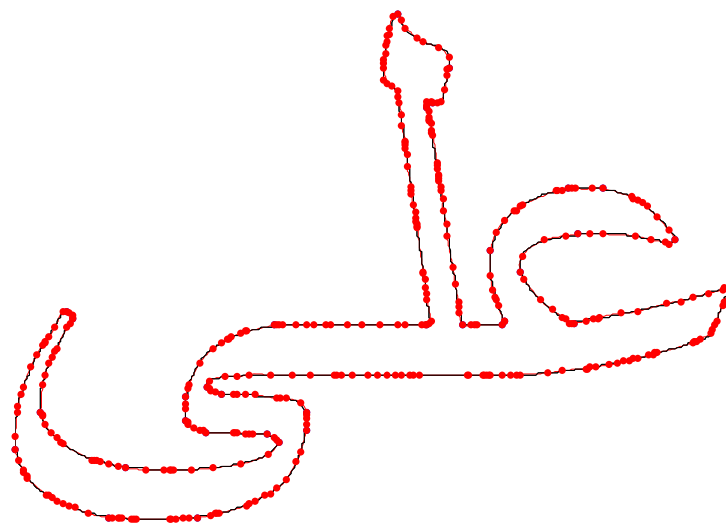
Table 4.5 Evaluation of algorithm 4.5 for Arabic word “Ali”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	1002	30	1653	1
1644	33	431	30	325	2
1644	33	260	30	88	3

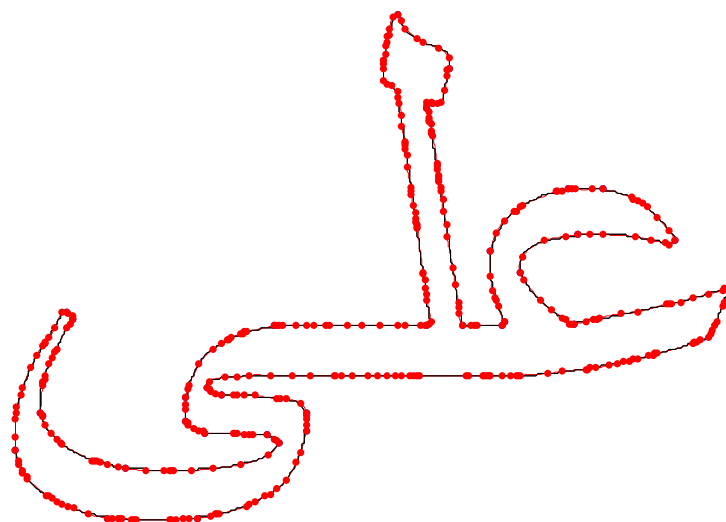
(Figure 4.27) to (Figure 4.29) shows the fitted curve over object contour at different iterations for algorithm 4.6 at threshold values of 1,2 and 3 respectively.



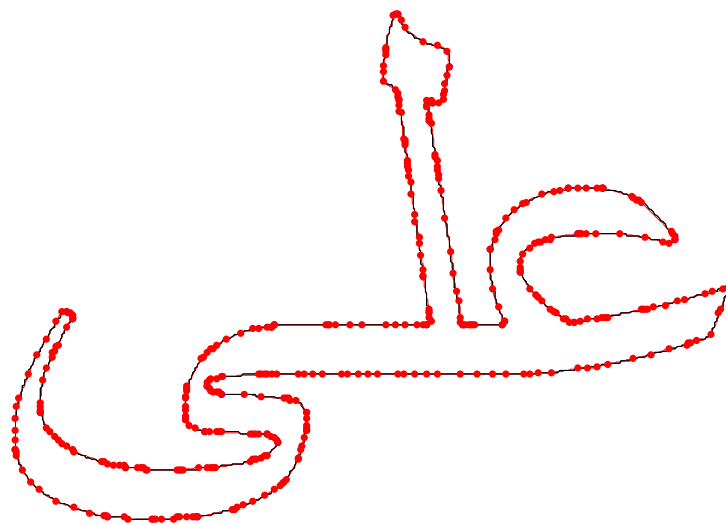
(a) At iteration = 1



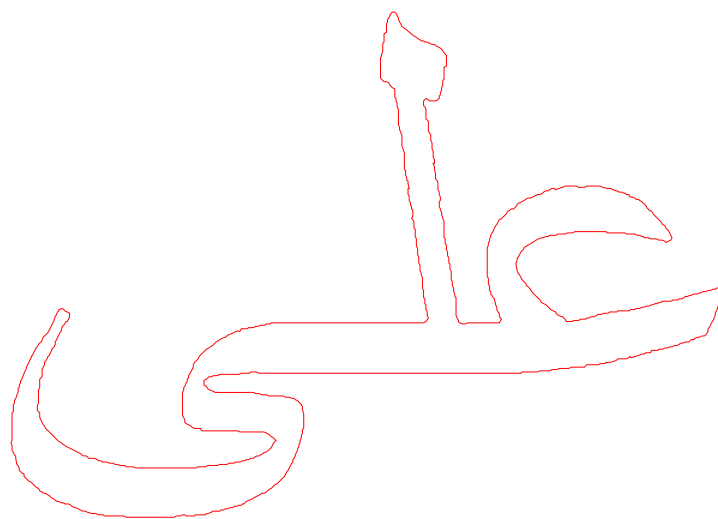
(b) At iteration = 10



(c) At last iteration = 15

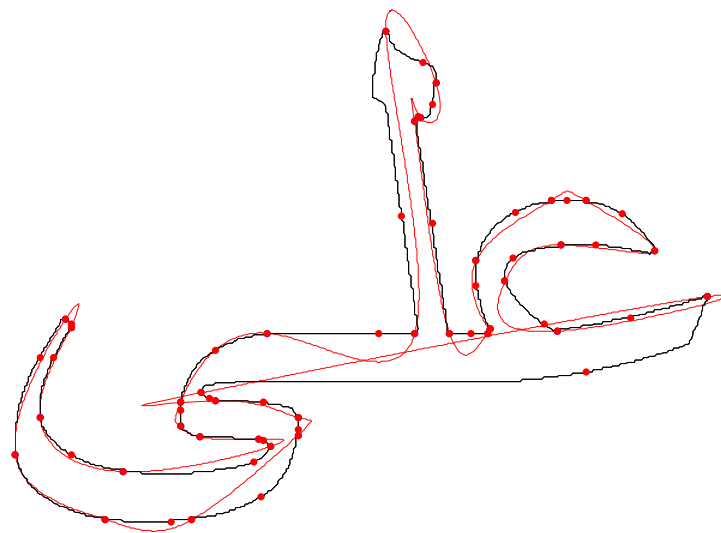


(d) At last iteration = 18

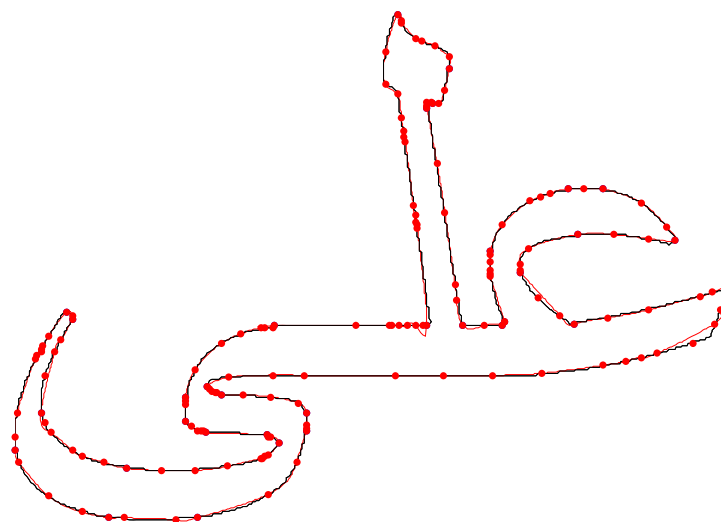


(e) Approximated spline for Arabic word “Ali”

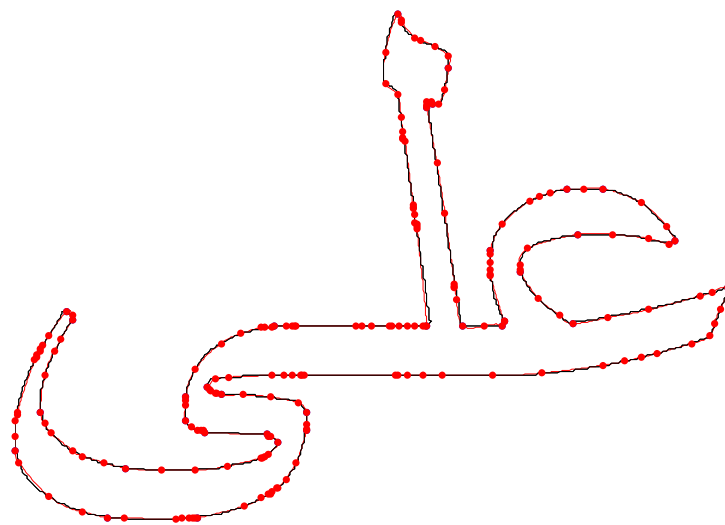
**Figure 4.27 Algorithm 4.6: Demonstration of spline fitting at each iteration using
threshold =1**



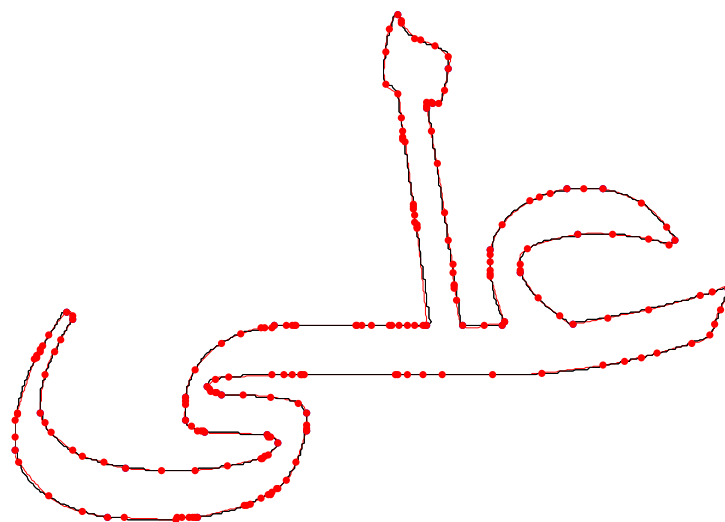
(a) At iteration = 1



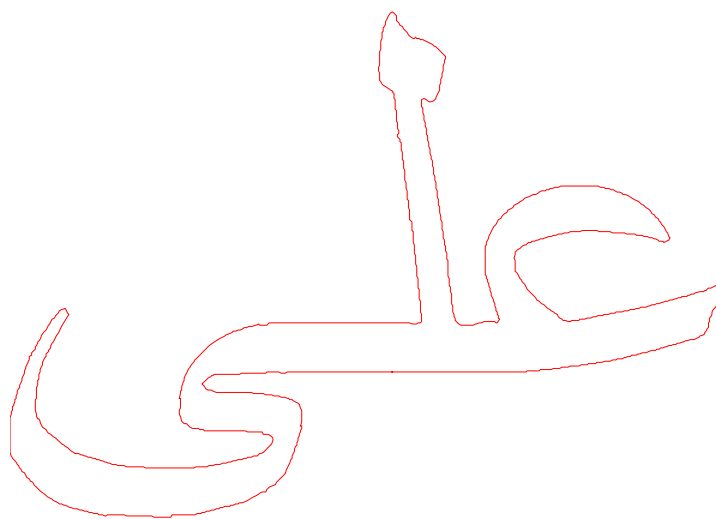
(b) At iteration = 5



(c) At last iteration = 10

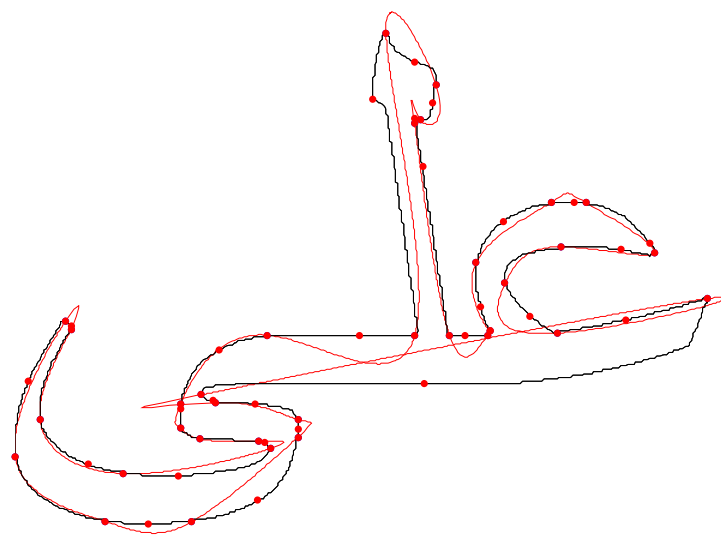


(d) At last iteration = 14

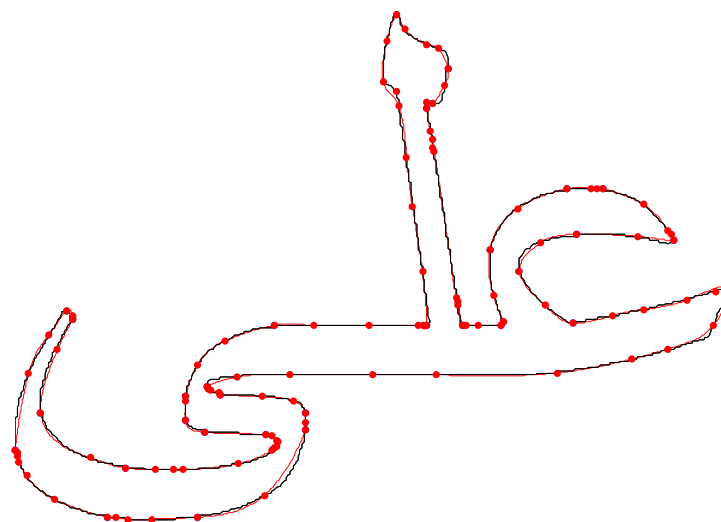


(e) Approximated spline for Arabic word “Ali”

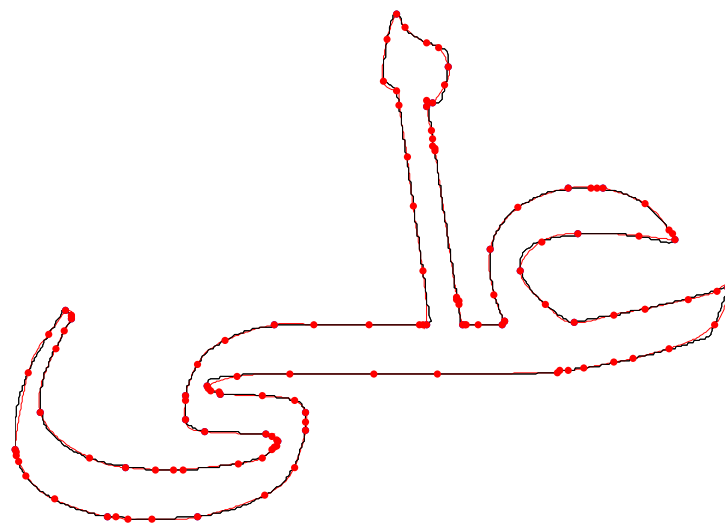
**Figure 4.28 Algorithm 4.6: Demonstration of spline fitting at each iteration using
threshold =2**



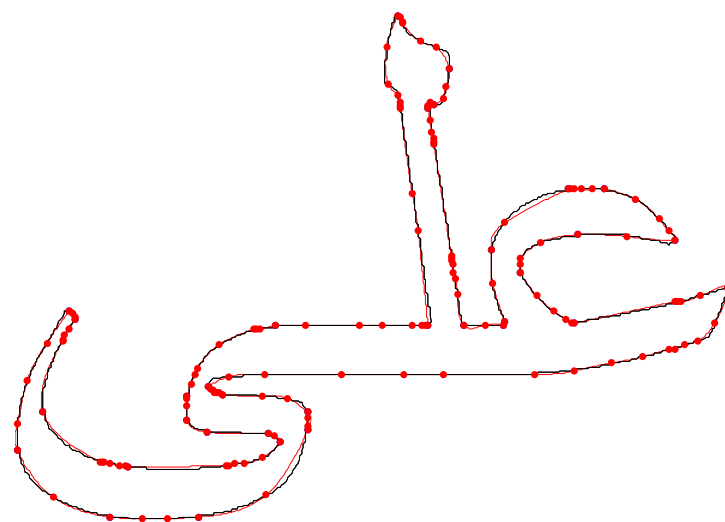
(a) At iteration = 1



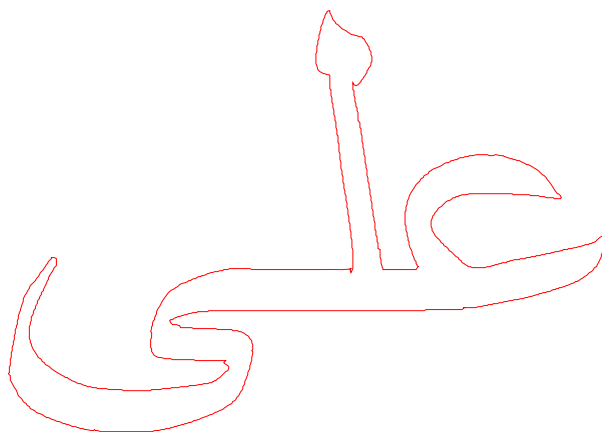
(b) At iteration = 5



(c) At last iteration = 10



(d) At last iteration = 12



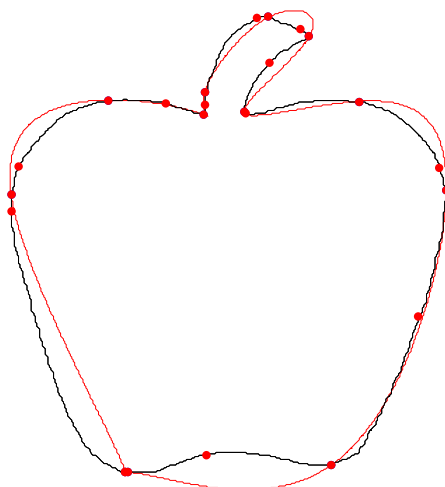
(e) Approximated spline for Arabic word “Ali”

Figure 4.29 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =3

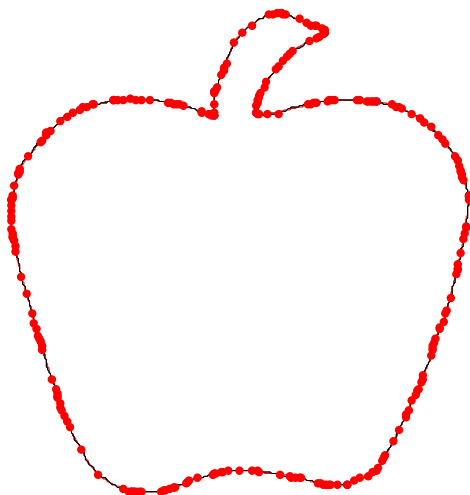
Table 4.6 Evaluation of algorithm 4.6 for Arabic word “Ali”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	344	14	81	1
1644	33	173	15	21	2
1644	33	91	10	9	3

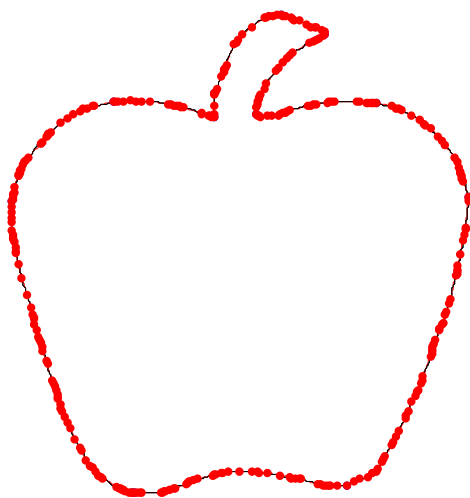
(Figure 4.30) to (Figure 4.32) shows the fitted curve over object contour at different iterations for algorithm 4.1 at threshold values of 1,2 and 3 respectively.



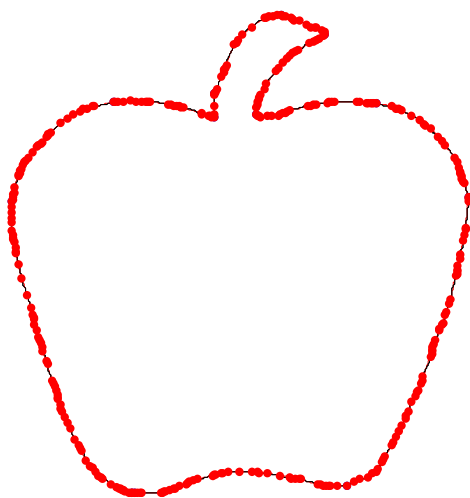
(a) At iteration = 1



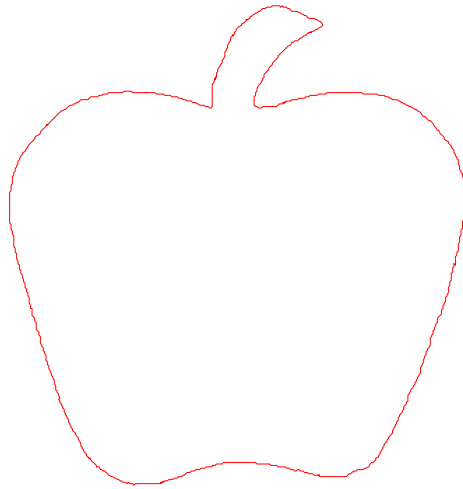
(b) At iteration = 10



(c) At iteration = 20

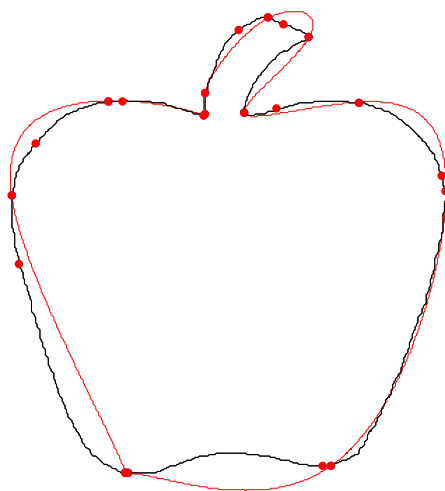


(d) At iteration = 30

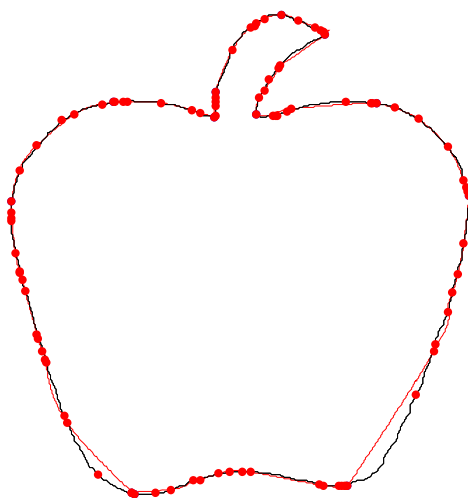


(e) Approximated spline for object “Apple”

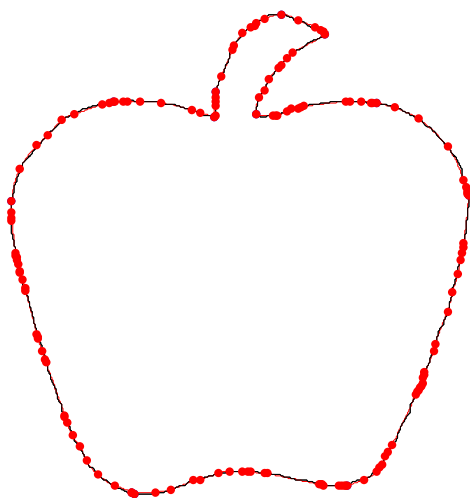
**Figure 4.30 Algorithm 4.1: Demonstration of spline fitting at each iteration using
threshold =1**



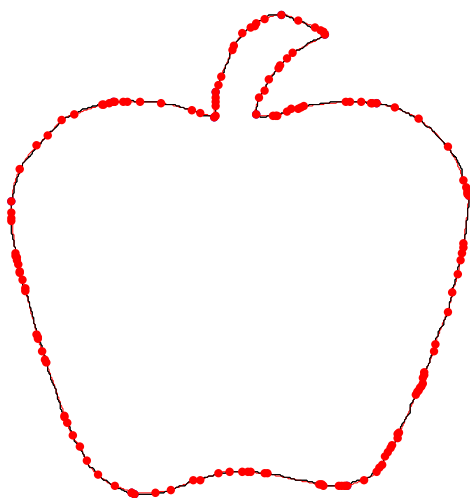
(a) At iteration = 1



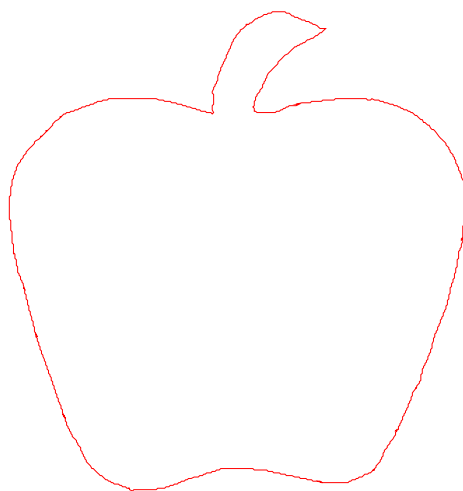
(b) At iteration = 10



(c) At iteration = 20

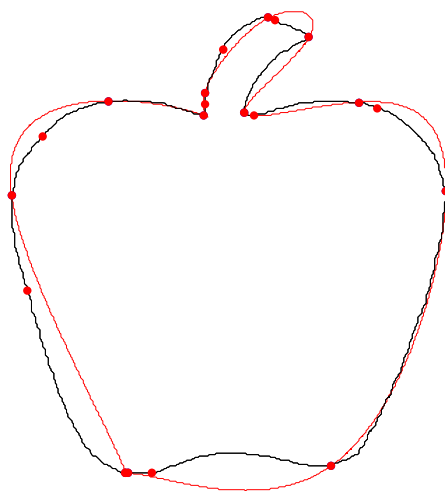


(d) At iteration = 30

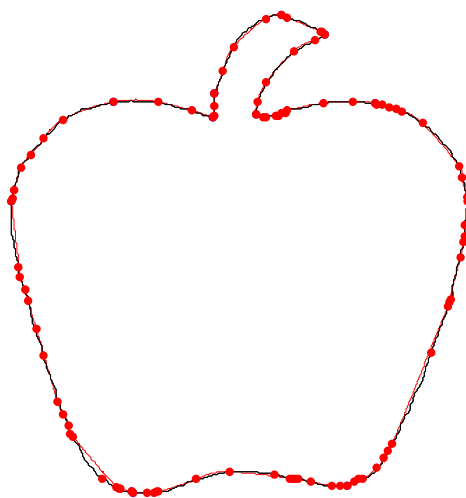


(e) Approximated spline for object “Apple”

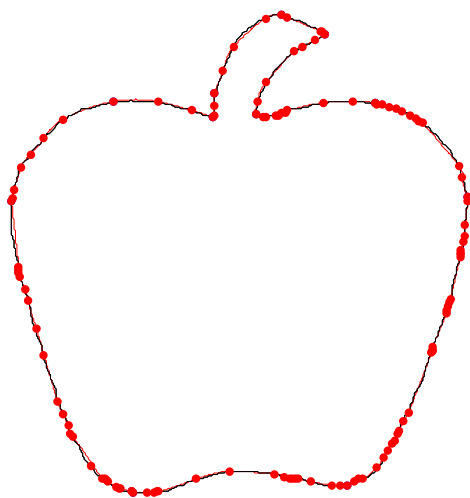
**Figure 4.31 Algorithm 4.1: Demonstration of spline fitting at each iteration using
threshold =2**



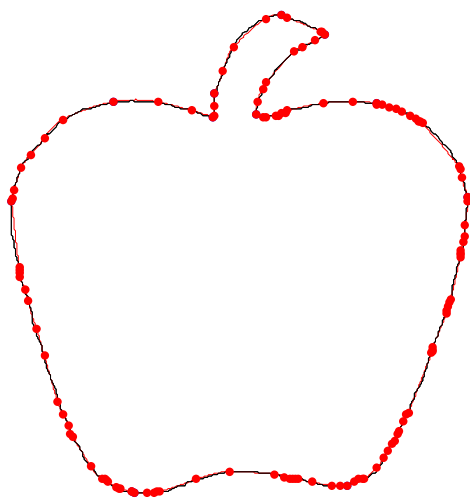
(a) At iteration = 1



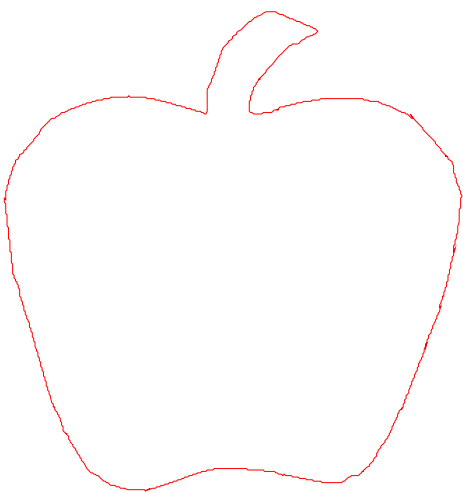
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 30



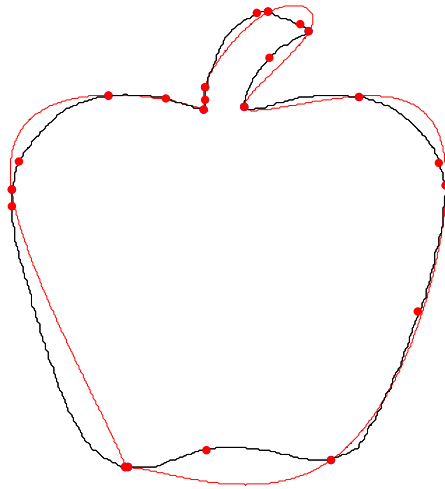
(e) Approximated spline for object “Apple”

Figure 4.32 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =3

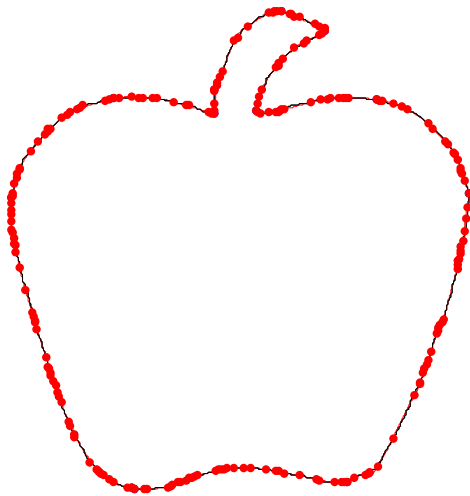
Table 4.7 Evaluation of algorithm 4.1 for object “Apple”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	544	30	202	1
1242	13	219	30	50	2
1242	13	151	30	21	3

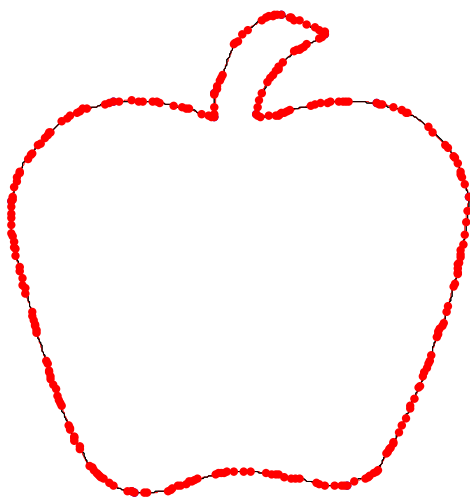
(Figure 4.33) to (Figure 4.35) shows the fitted curve over object contour at different iterations for algorithm 4.2 at threshold values of 1,2 and 3 respectively.



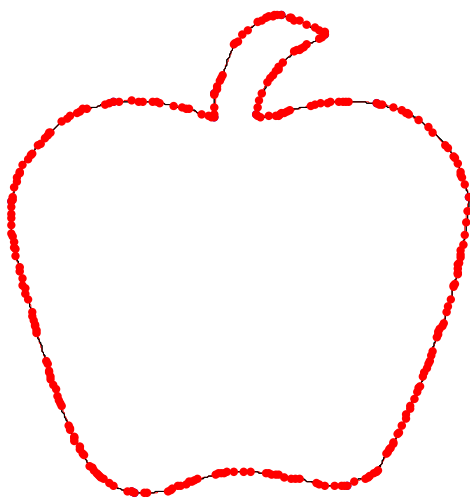
(a) At iteration = 1



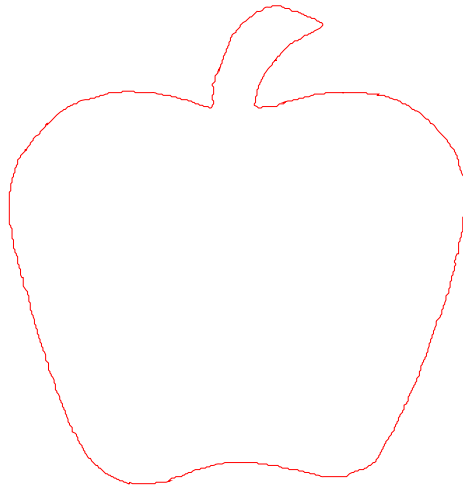
(b) At iteration = 10



(c) At iteration = 20

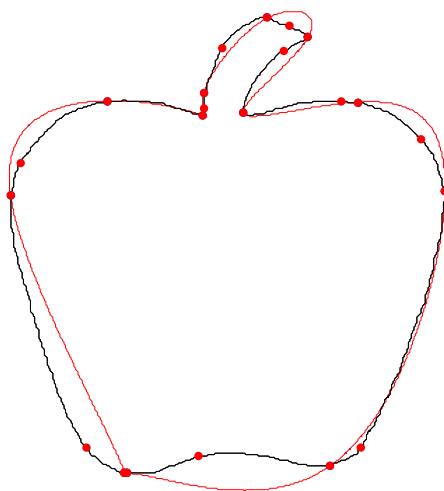


(d) At iteration = 30

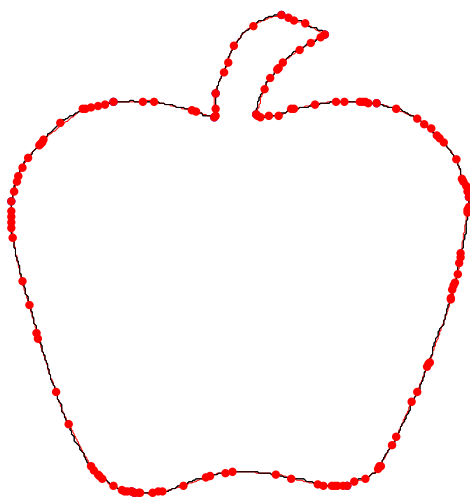


(e) Approximated spline for object “Apple”

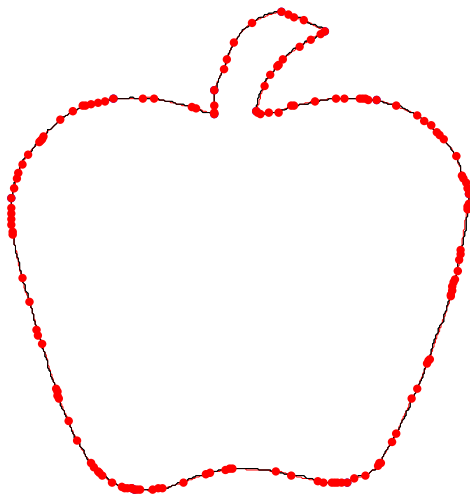
**Figure 4.33 Algorithm 4.2: Demonstration of spline fitting at each iteration using
threshold =1**



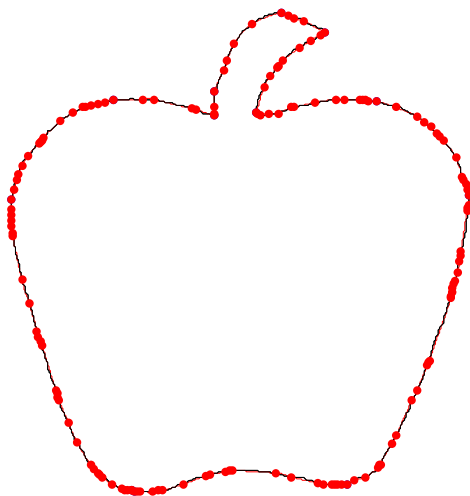
(a) At iteration = 1



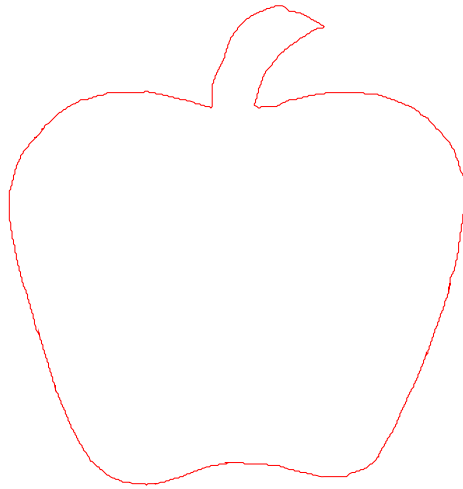
(b) At iteration = 10



(c) At iteration = 20

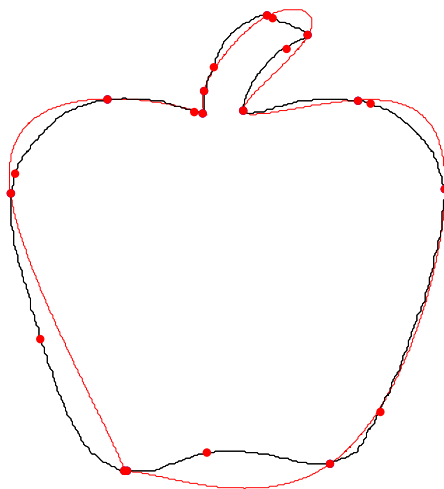


(d) At iteration = 30

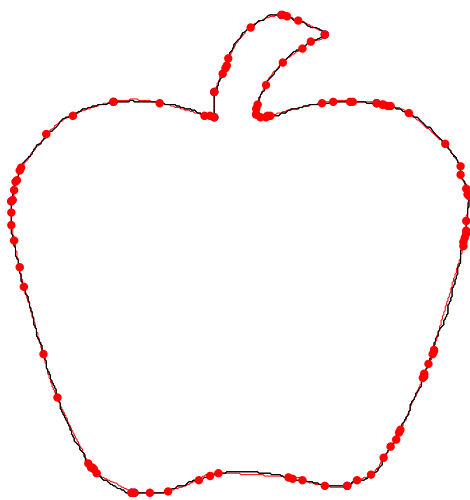


(e) Approximated spline for object “Apple”

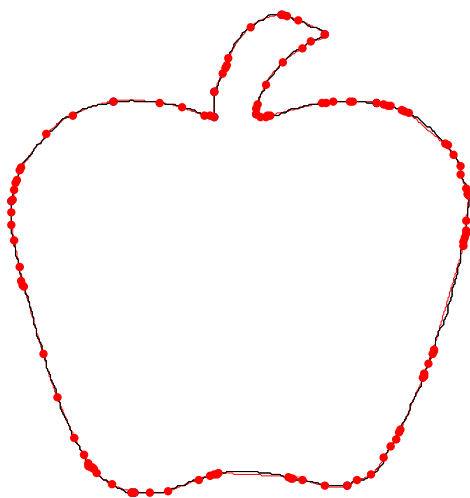
**Figure 4.34 Algorithm 4.2: Demonstration of spline fitting at each iteration using
threshold =2**



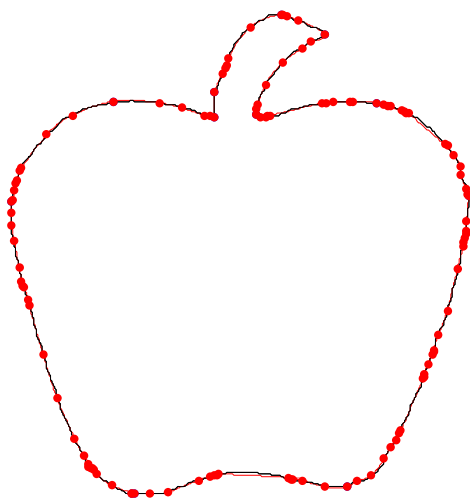
(a) At iteration = 1



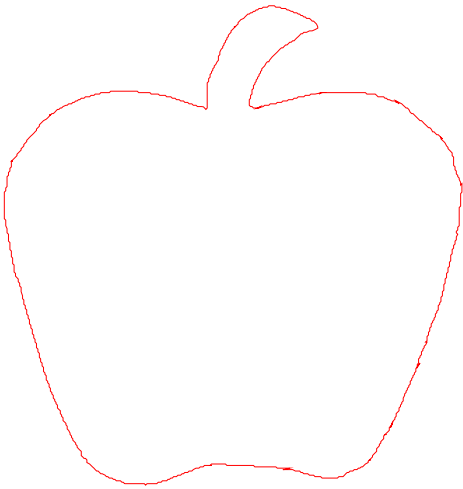
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 30



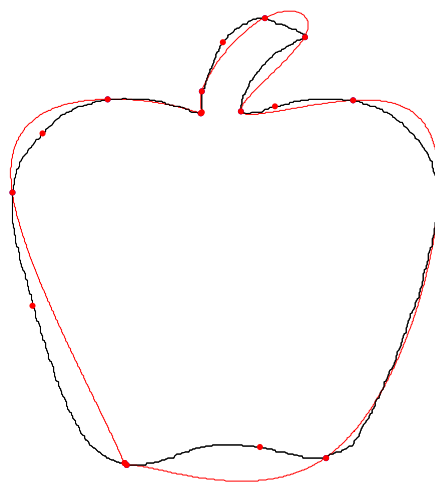
(e) Approximated spline for object “Apple”

Figure 4.35 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =3

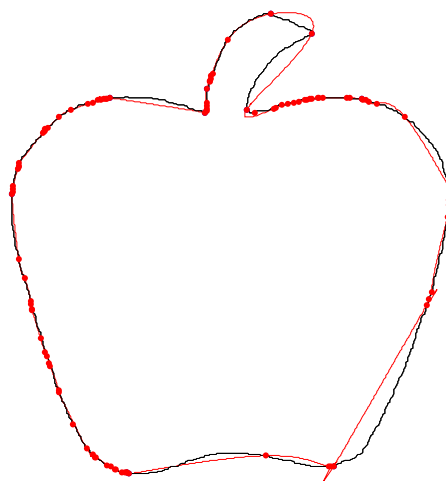
Table 4.8 Evaluation of algorithm 4.2 for object “Apple”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	271	30	57	1
1242	13	151	30	21	2
1242	13	105	30	16	3

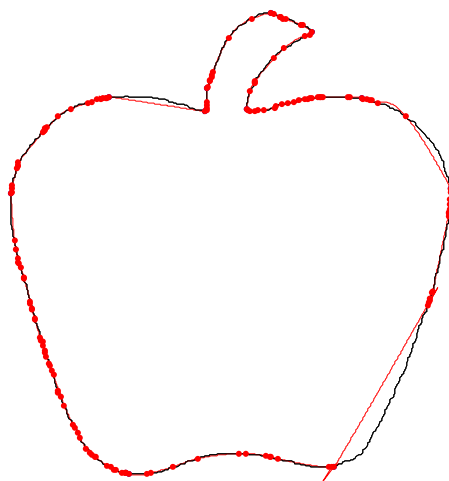
(Figure 4.36) to (Figure 4.38) shows the fitted curve over object contour at different iterations for algorithm 4.1 at threshold values of 1,2 and 3 respectively.



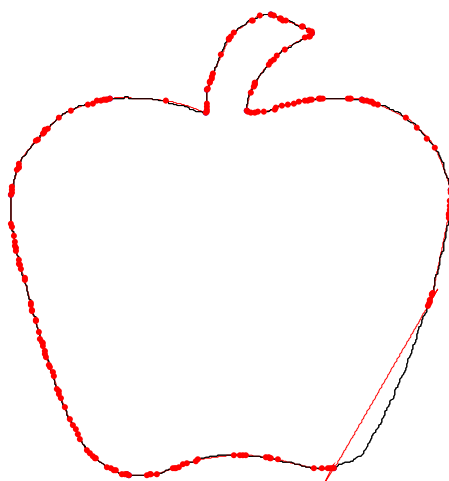
(a) At iteration = 1



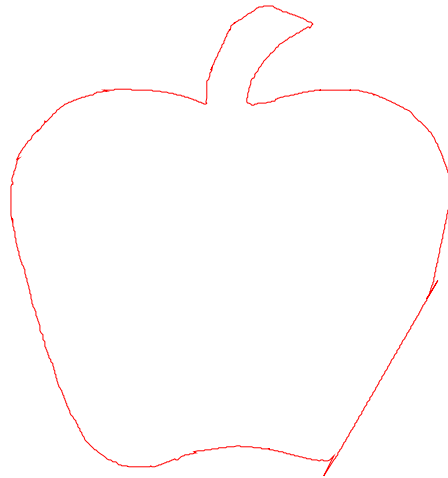
(b) At iteration = 10



(c) At iteration = 20

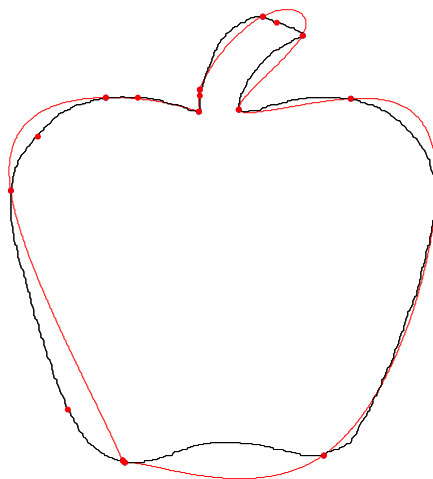


(d) At iteration = 30

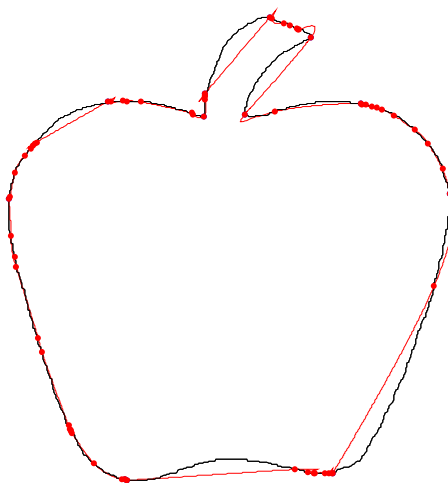


(e) Approximated spline for object “Apple”

**Figure 4.36 Algorithm 4.3: Demonstration of spline fitting at each iteration using
threshold =1**

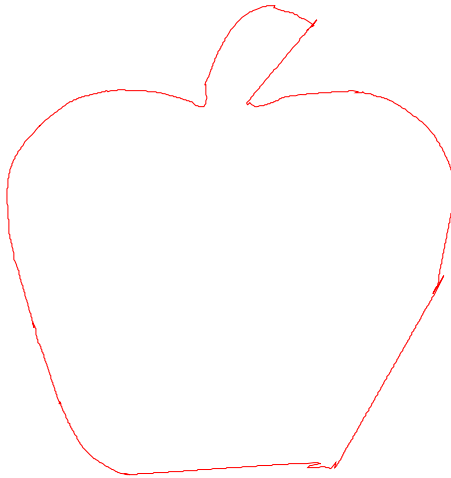


(a) At iteration = 1



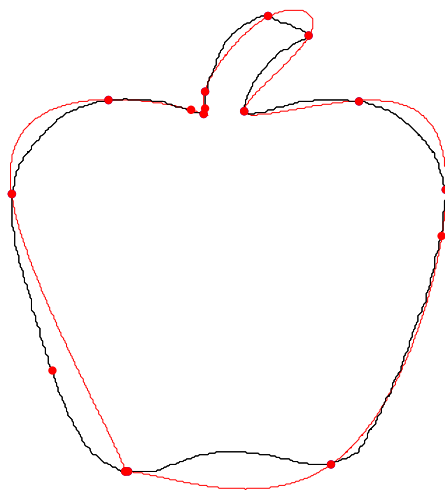
(b) At iteration = 10

(d) At iteration = 30

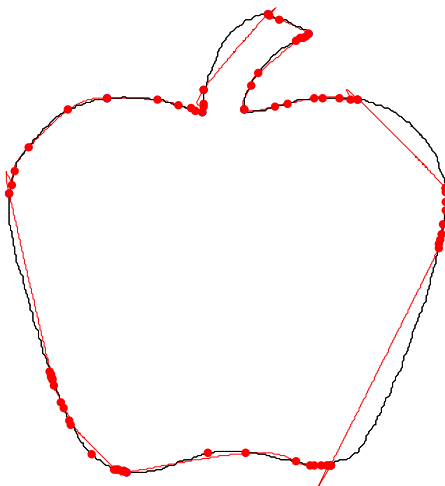


(e) Approximated spline for object “Apple”

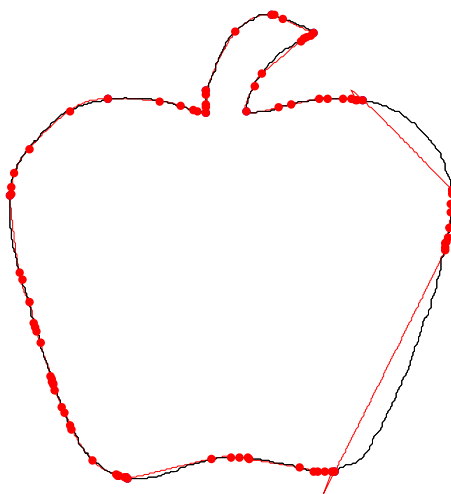
**Figure 4.37 Algorithm 4.3: Demonstration of spline fitting at each iteration using
threshold =2**



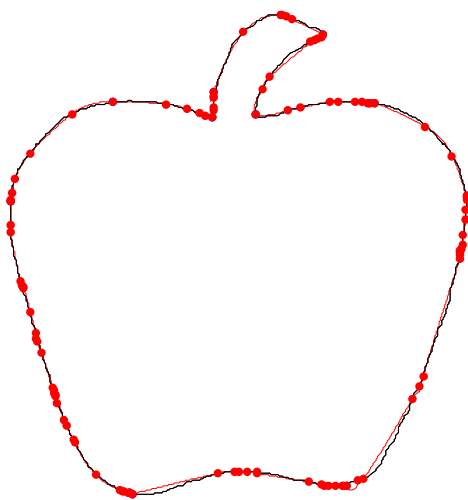
(a) At iteration = 1



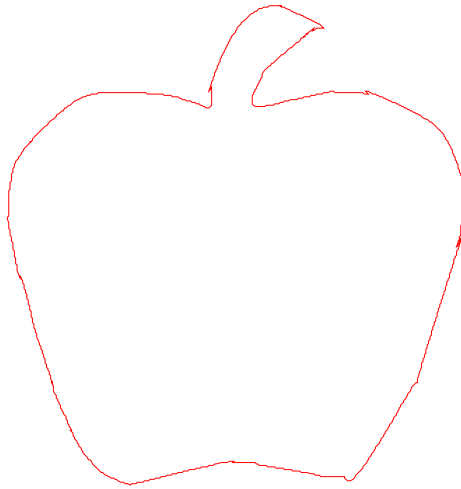
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 30



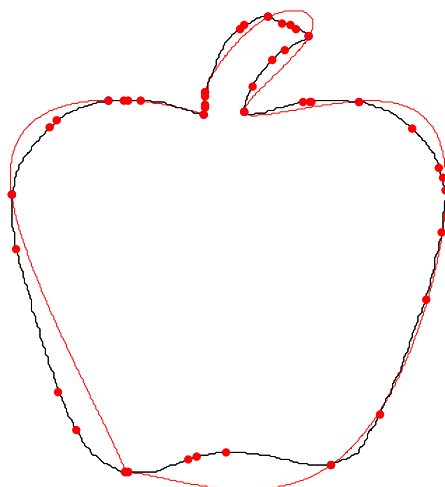
(e) Approximated spline for object “Apple”

Figure 4.38 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =3

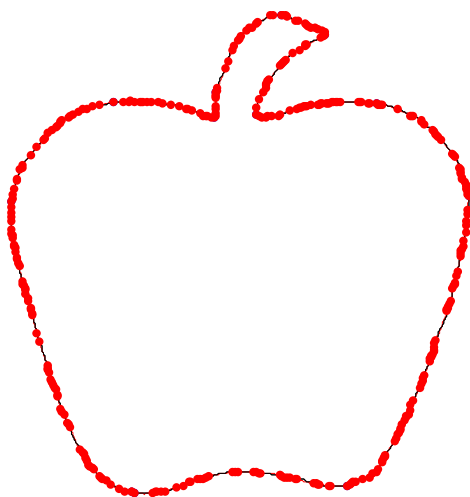
Table 4.9 Evaluation of algorithm 4.3 for object “Apple”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	189	30	26	1
1242	13	107	30	17	2
1242	13	89	30	15	3

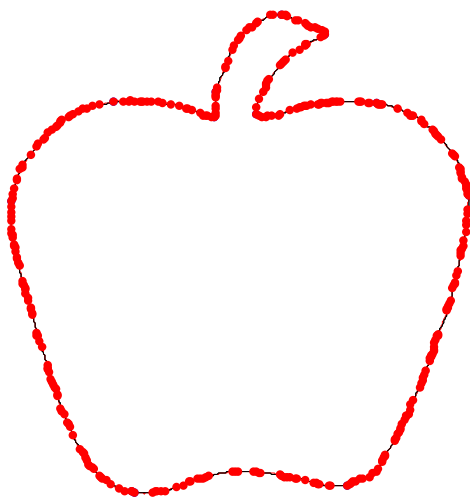
(Figure 4.39) to (Figure 4.41) shows the fitted curve over object contour at different iterations for algorithm 4.1 at threshold values of 1,2 and 3 respectively.



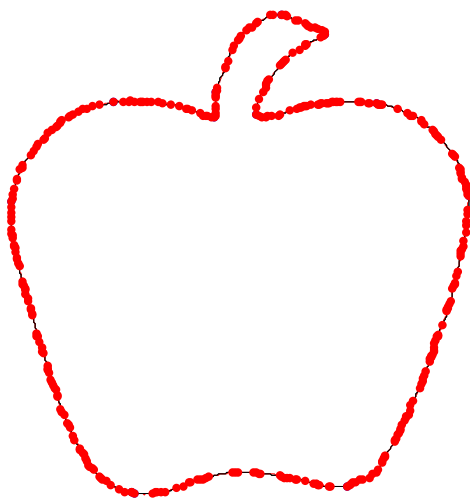
(a) At iteration = 1



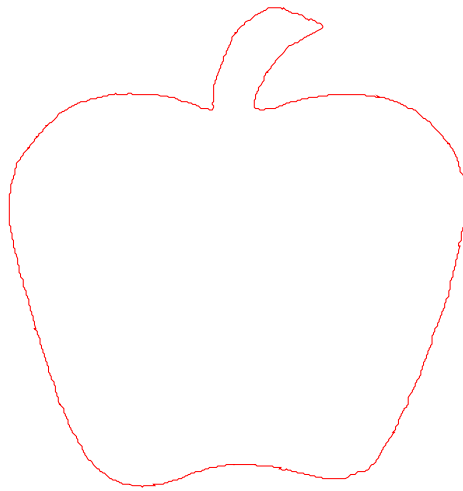
(b) At iteration = 10



(c) At iteration = 20

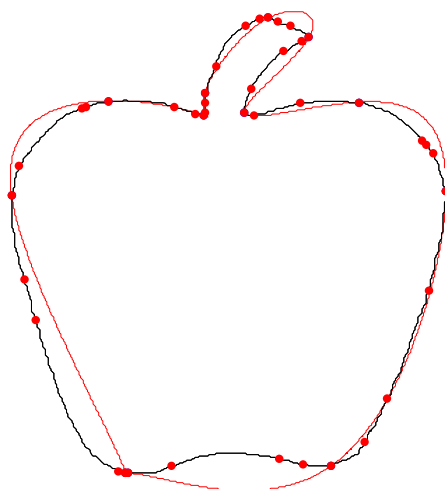


(d) At iteration = 30

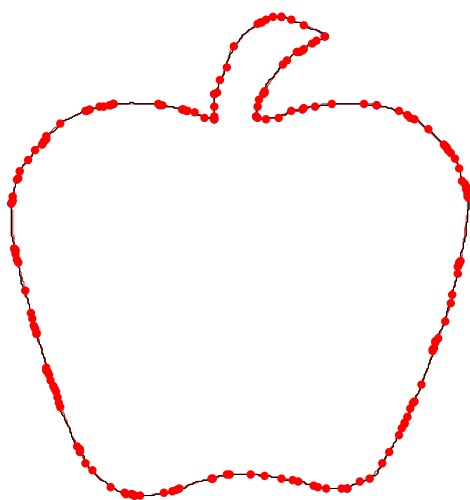


(e) Approximated spline for object “Apple”

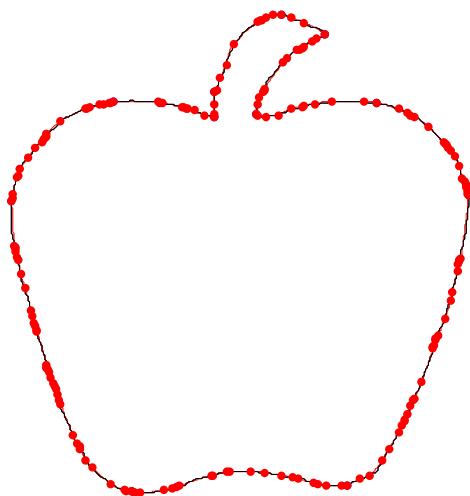
**Figure 4.39 Algorithm 4.4: Demonstration of spline fitting at each iteration using
threshold =1**



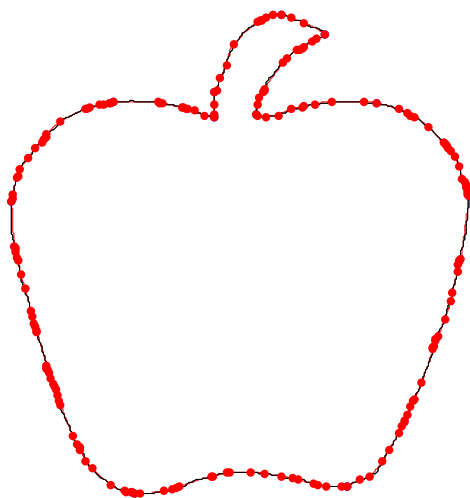
(a) At iteration = 1



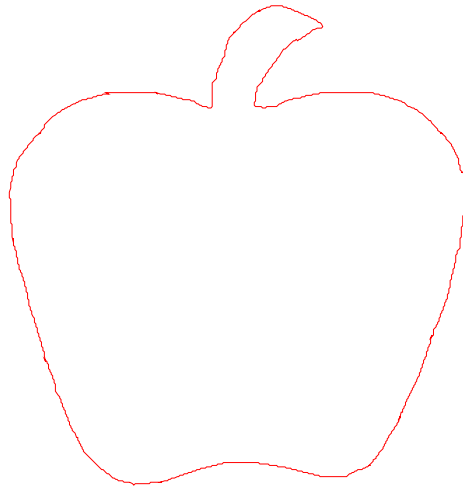
(b) At iteration = 10



(c) At iteration = 20

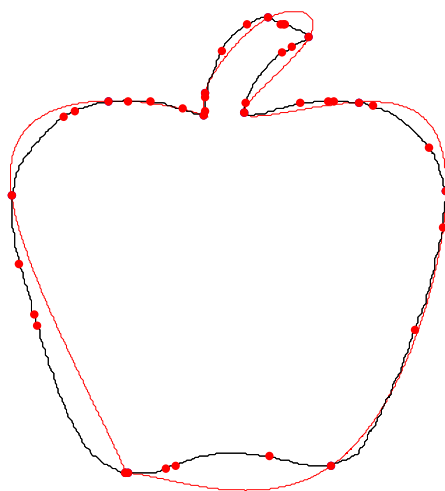


(d) At iteration = 30

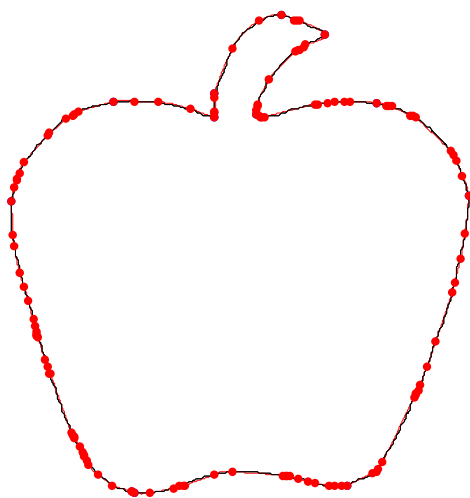


(e) Approximated spline for object “Apple”

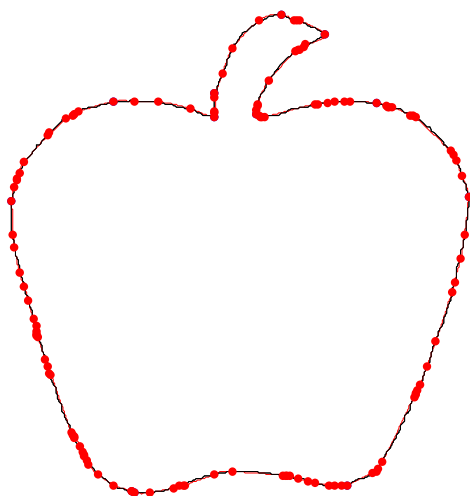
**Figure 4.40 Algorithm 4.4: Demonstration of spline fitting at each iteration using
threshold =2**



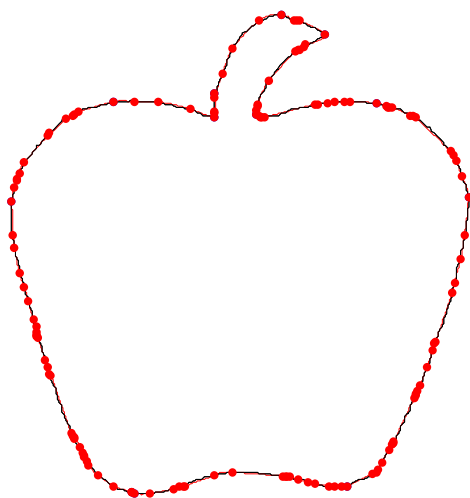
(a) At iteration = 1



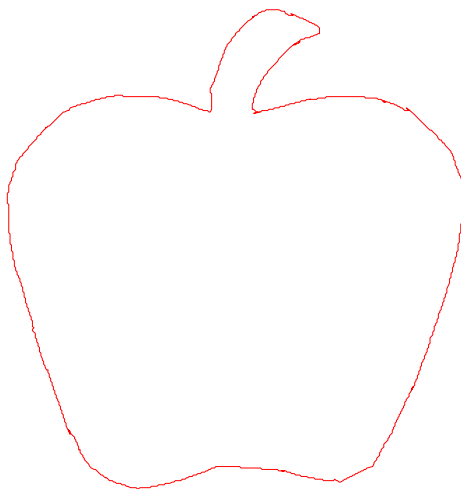
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 30



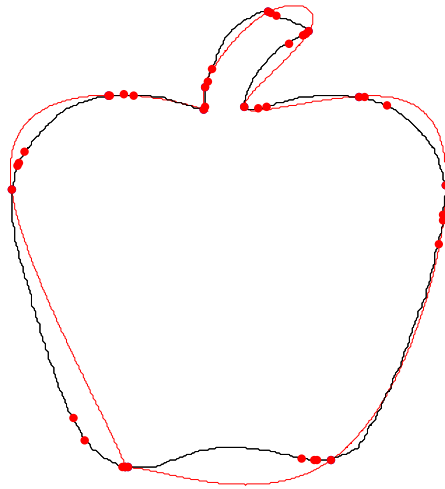
(e) Approximated spline for object “Apple”

Figure 4.41 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =3

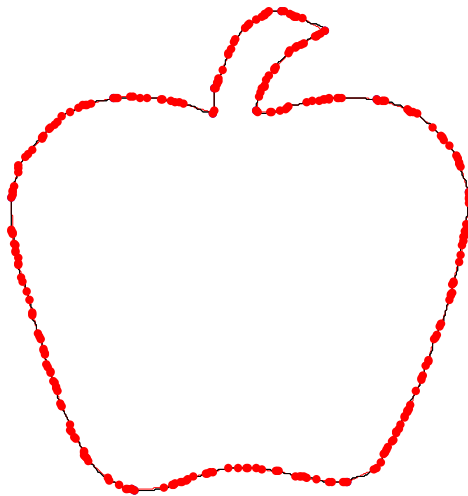
Table 4.10 Evaluation of algorithm 4.4 for object “Apple”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	1196	30	1299	1
1242	13	398	30	112	2
1242	13	305	30	53	3

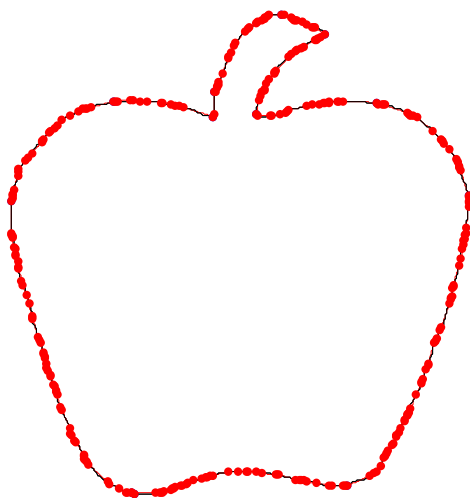
(Figure 4.42) to (Figure 4.44) shows the fitted curve over object contour at different iterations for algorithm 4.1 at threshold values of 1,2 and 3 respectively.



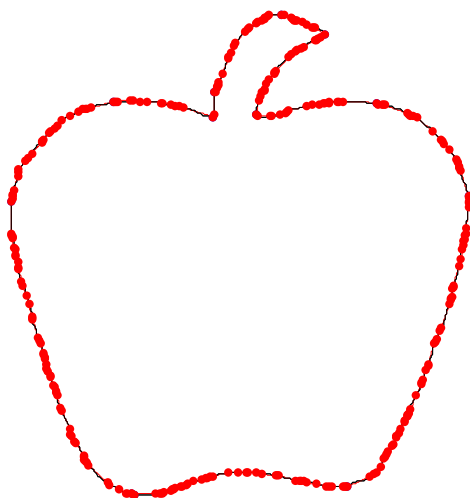
(a) At iteration = 1



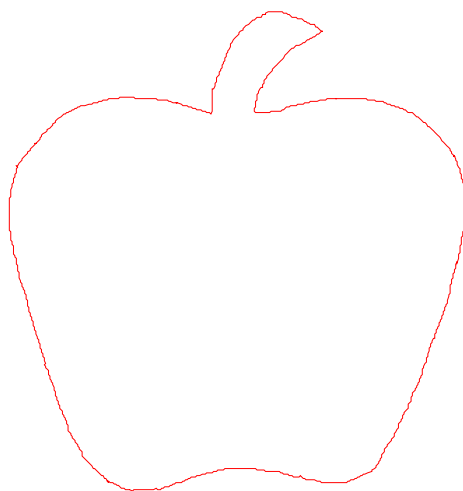
(b) At iteration = 10



(c) At iteration = 20

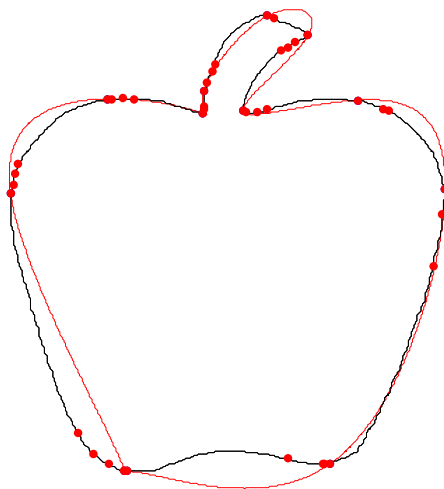


(d) At iteration = 30

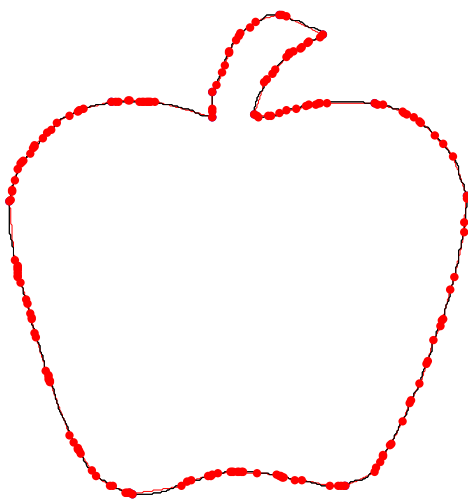


(e) Approximated spline for object “Apple”

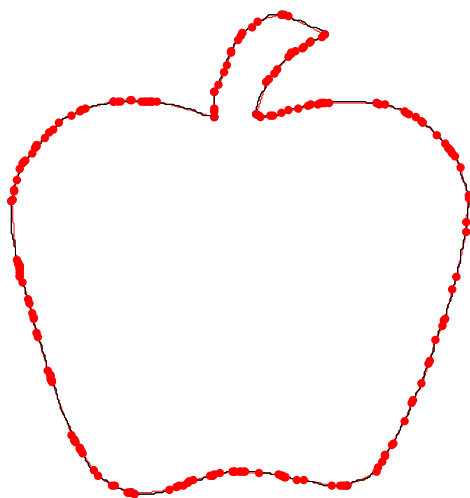
**Figure 4.42 Algorithm 4.5: Demonstration of spline fitting at each iteration using
threshold =1**



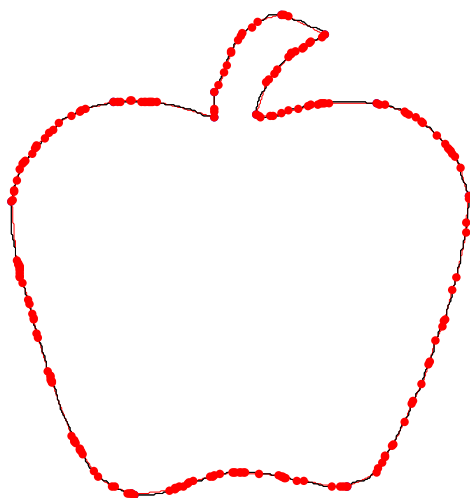
(a) At iteration = 1



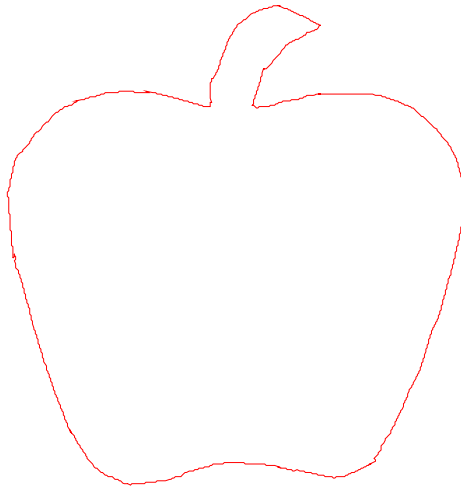
(b) At iteration = 10



(c) At iteration = 20

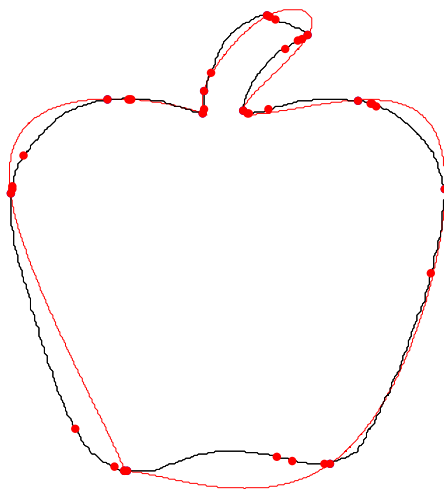


(d) At iteration = 30

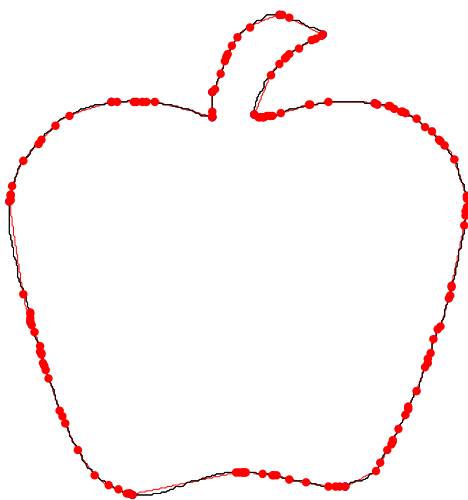


(e) Approximated spline for object “Apple”

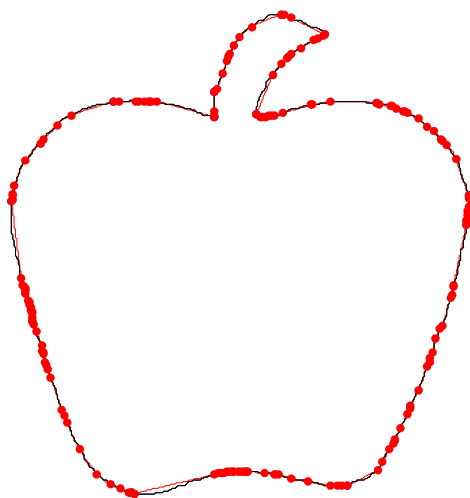
**Figure 4.43 Algorithm 4.5: Demonstration of spline fitting at each iteration using
threshold =2**



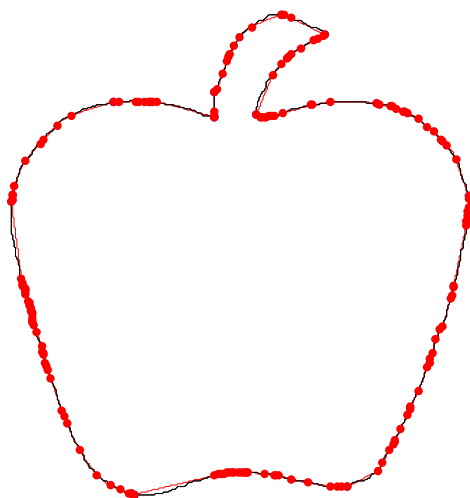
(a) At iteration = 1



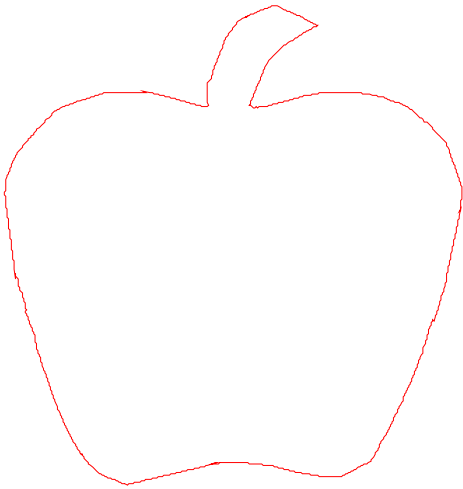
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 30



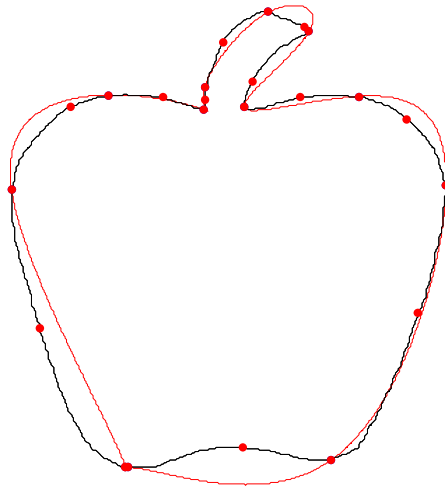
(e) Approximated spline for object “Apple”

Figure 4.44 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =3

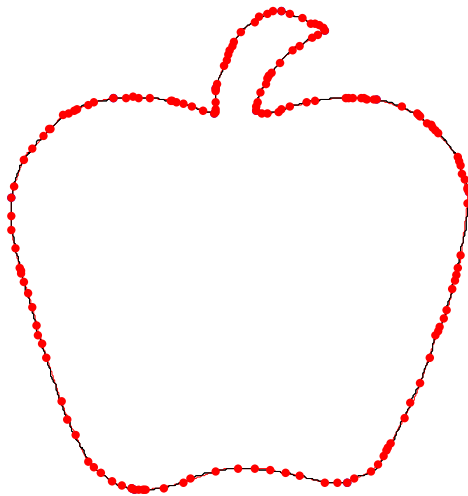
Table 4.11 Evaluation of algorithm 4.5 for object “Apple”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	821	30	955	1
1242	13	443	30	142	2
1242	13	268	30	42	3

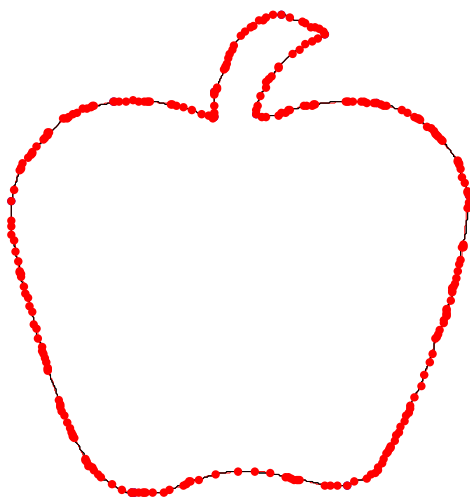
(Figure 4.45) to (Figure 4.47) shows the fitted curve over object contour at different iterations for algorithm 4.1 at threshold values of 1,2 and 3 respectively.



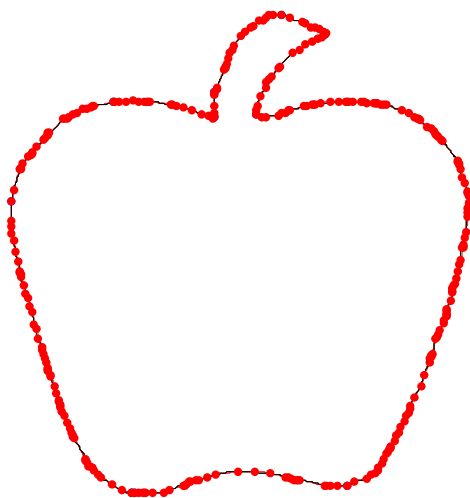
(a) At iteration = 1



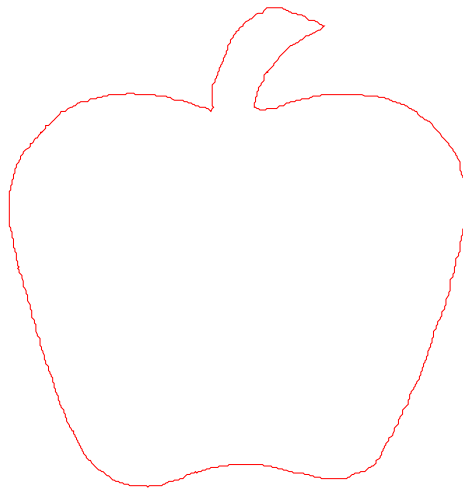
(b) At iteration = 5



(c) At iteration = 10

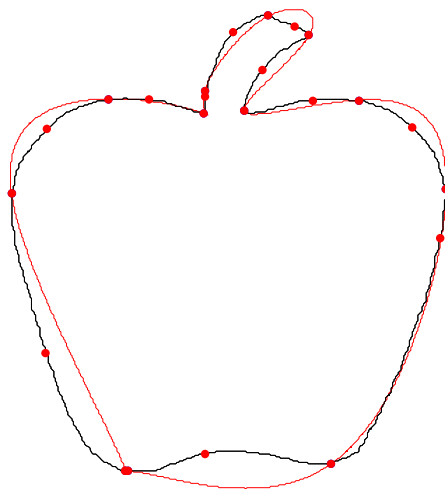


(d) At iteration = 15

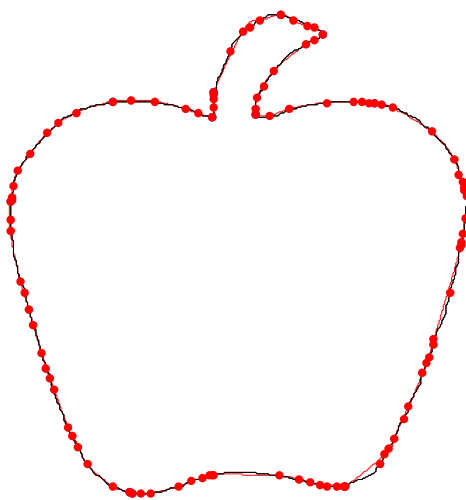


(e) Approximated spline for object “Apple”

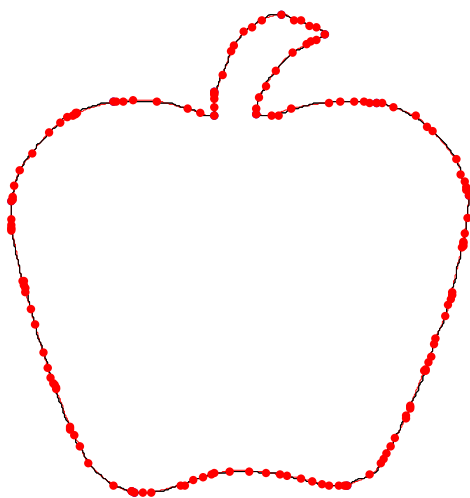
**Figure 4.45 Algorithm 4.6: Demonstration of spline fitting at each iteration using
threshold =1**



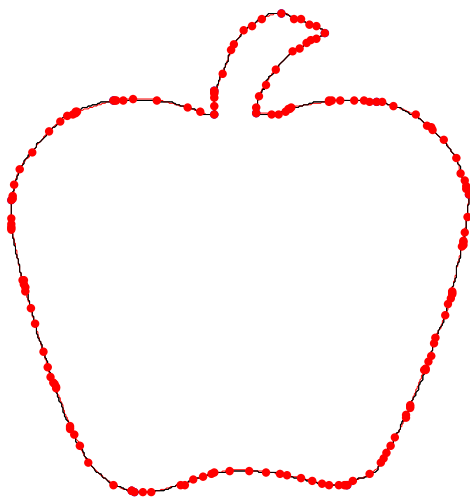
(a) At iteration = 1



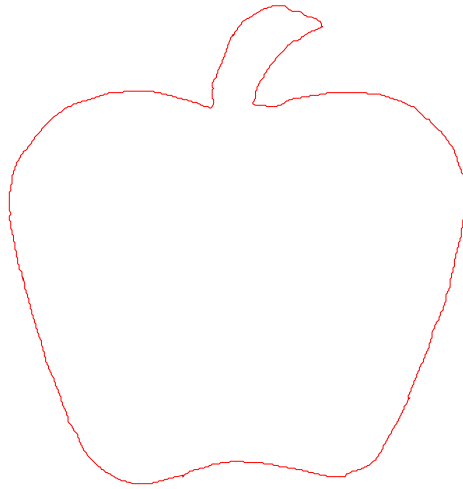
(b) At iteration = 5



(c) At iteration = 10

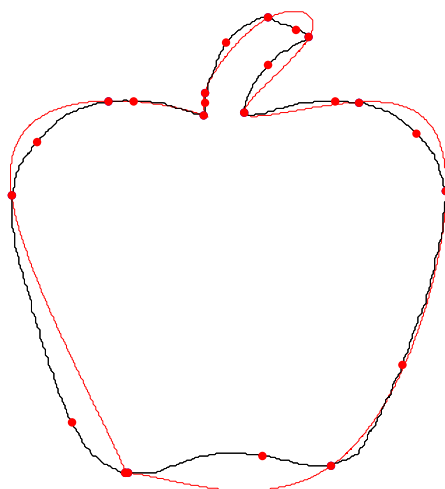


(d) At iteration = 15

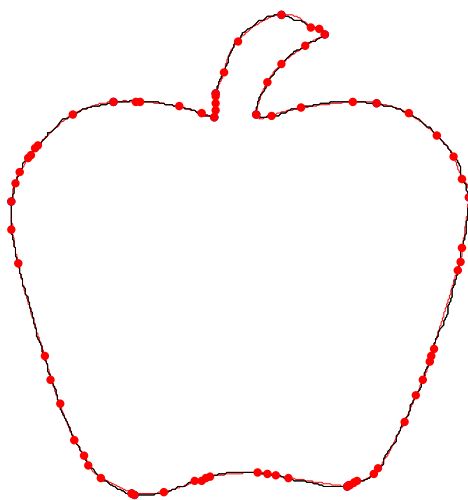


(e) Approximated spline for object “Apple”

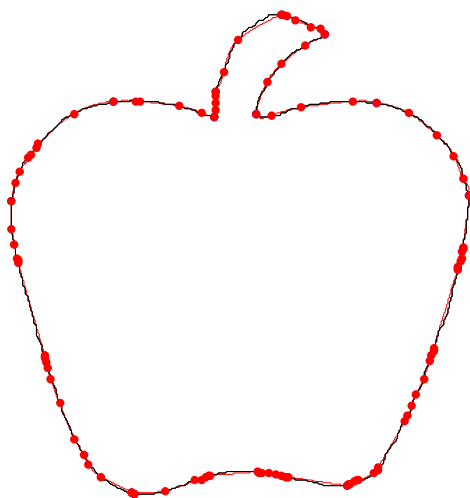
**Figure 4.46 Algorithm 4.6: Demonstration of spline fitting at each iteration using
threshold =2**



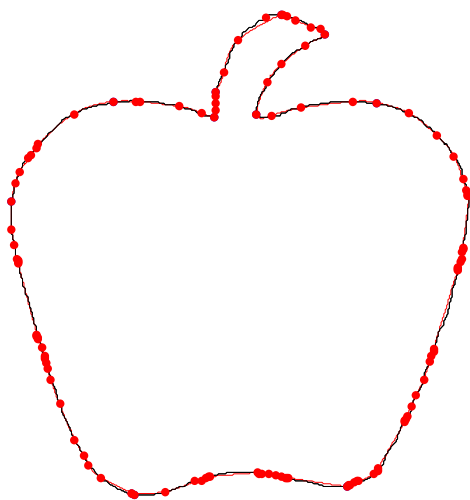
(a) At iteration = 1



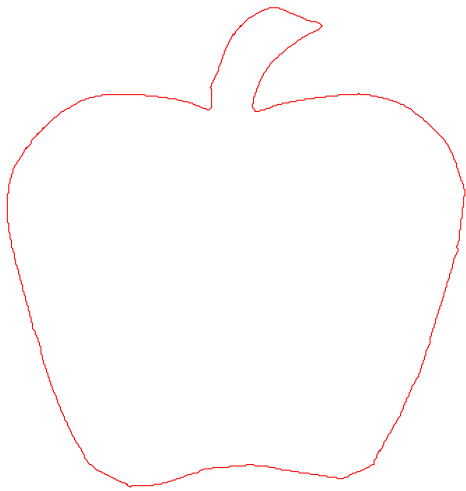
(b) At iteration = 5



(c) At iteration = 10



(d) At iteration = 15



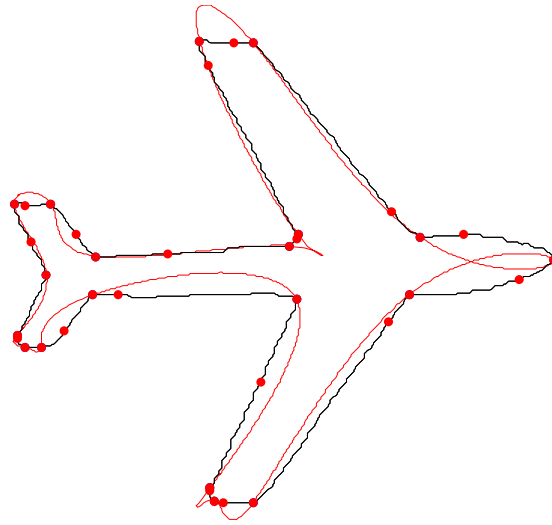
(e) Approximated spline for object “Apple”

Figure 4.47 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =3

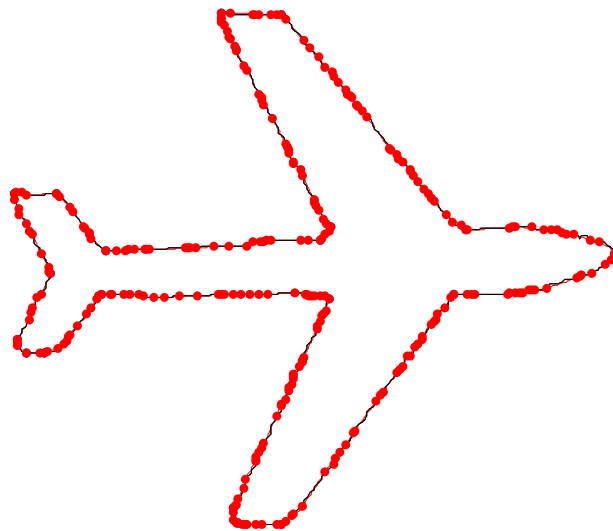
Table 4.12 Evaluation of algorithm 4.6 for object “Apple”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	289	16	35	1
1242	13	159	15	8	2
1242	13	101	16	9	3

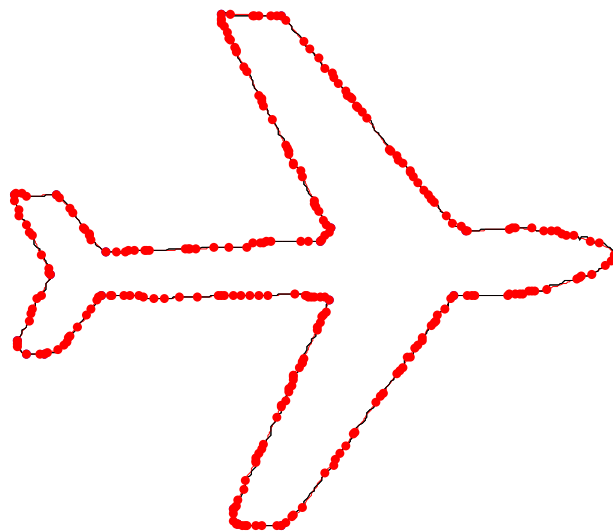
(Figure 4.48) to (Figure 4.50) shows the fitted curve over object contour at different iterations for algorithm 4.1 at threshold values of 1,2 and 3 respectively.



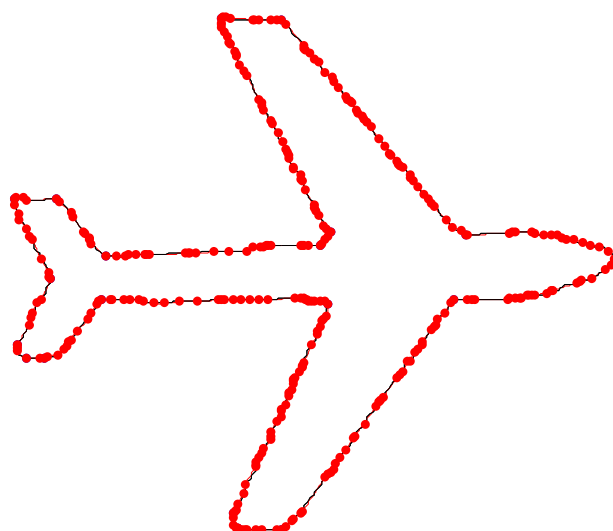
(a) At iteration = 1



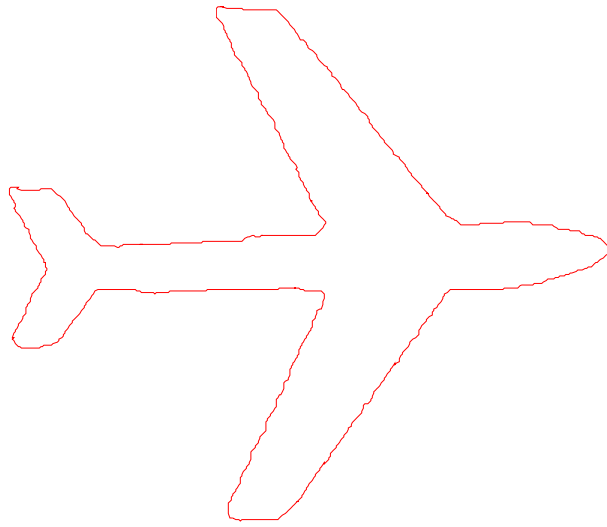
(b) At iteration = 10



(c) At iteration = 20

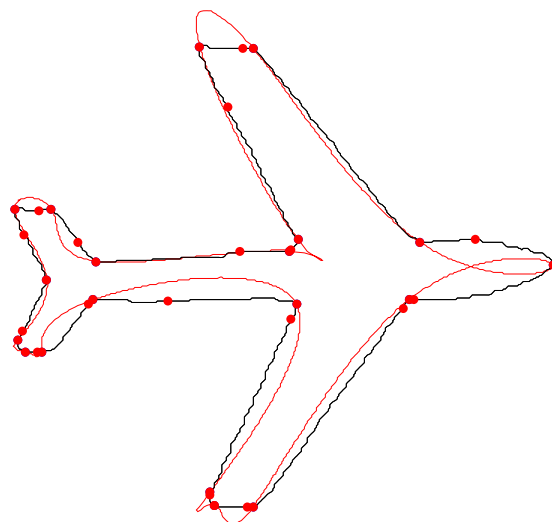


(d) At iteration = 30

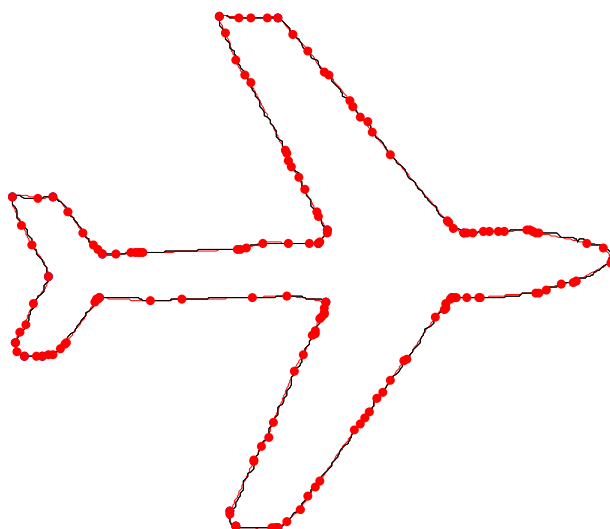


(e) Approximated spline for object “Plane”

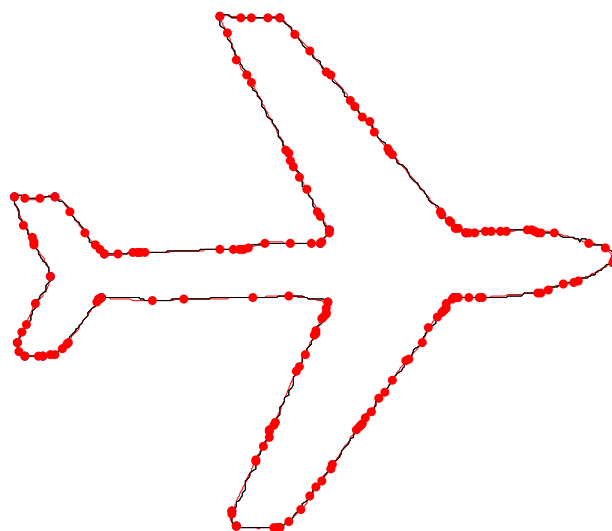
**Figure 4.48 Algorithm 4.1: Demonstration of spline fitting at each iteration using
threshold =1**



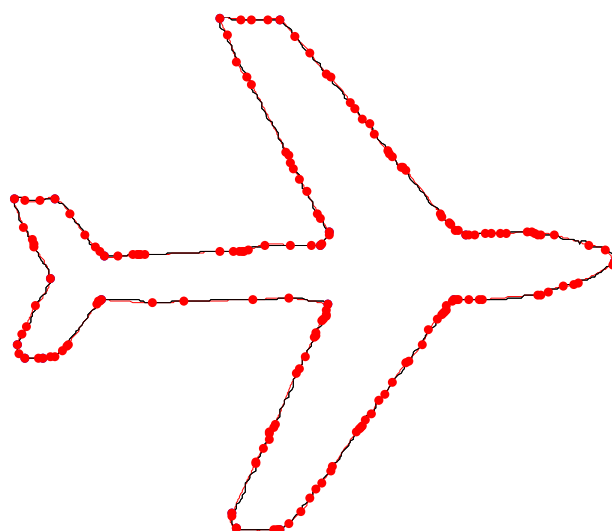
(a) At iteration = 1



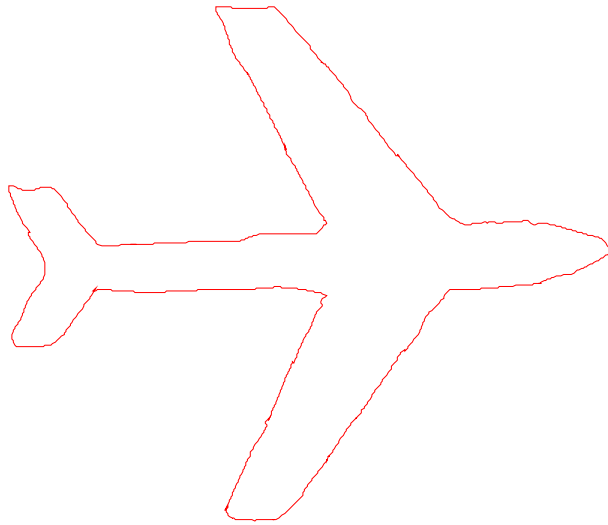
(b) At iteration = 10



(c) At iteration = 20

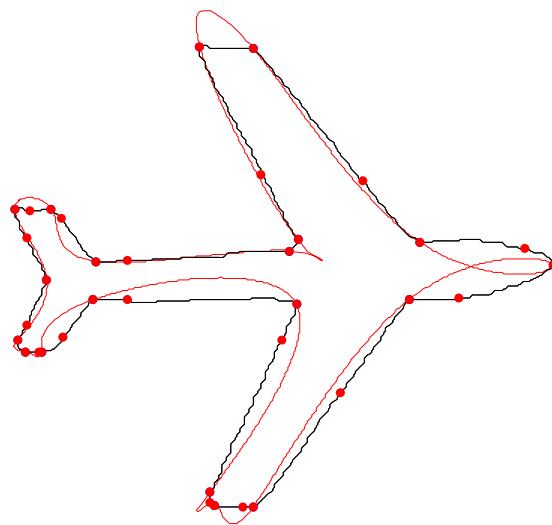


(d) At iteration = 30

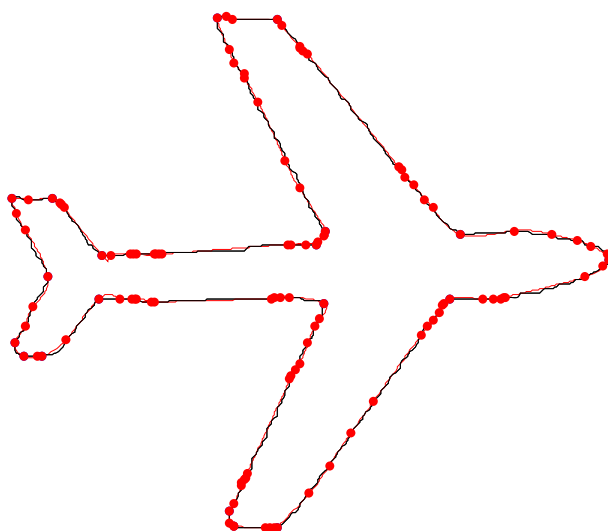


(e) Approximated spline for object “Plane”

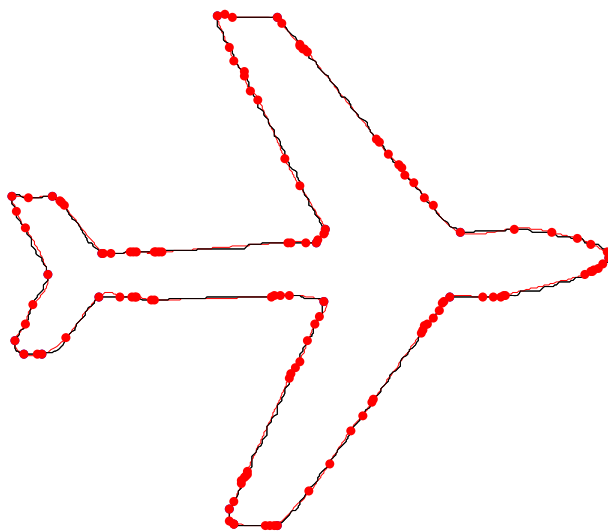
**Figure 4.49 Algorithm 4.1: Demonstration of spline fitting at each iteration using
threshold =2**



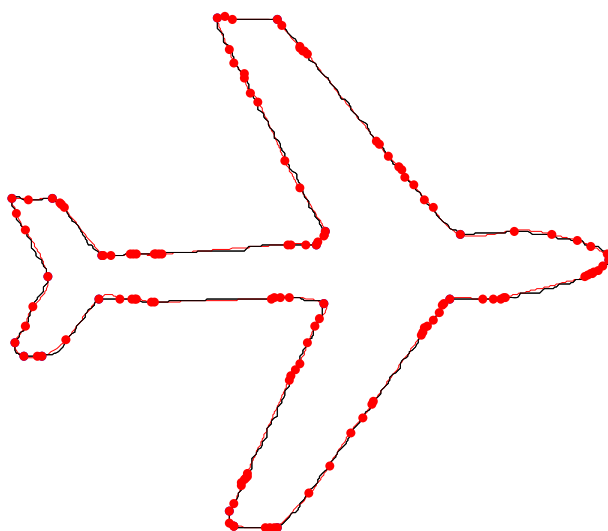
(a) At iteration = 1



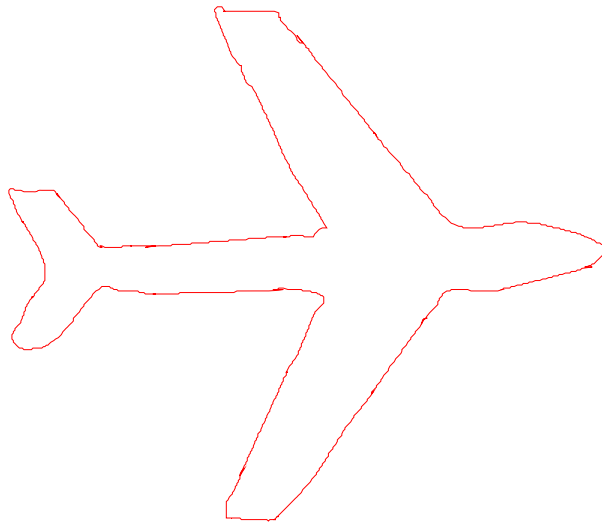
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 25



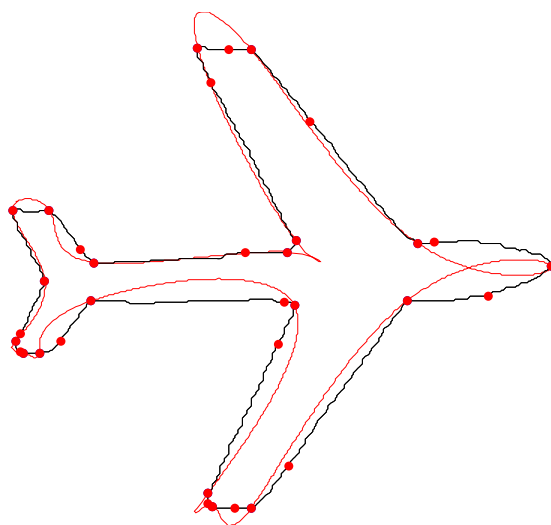
(e) Approximated spline for object “Plane”

Figure 4.50 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =3

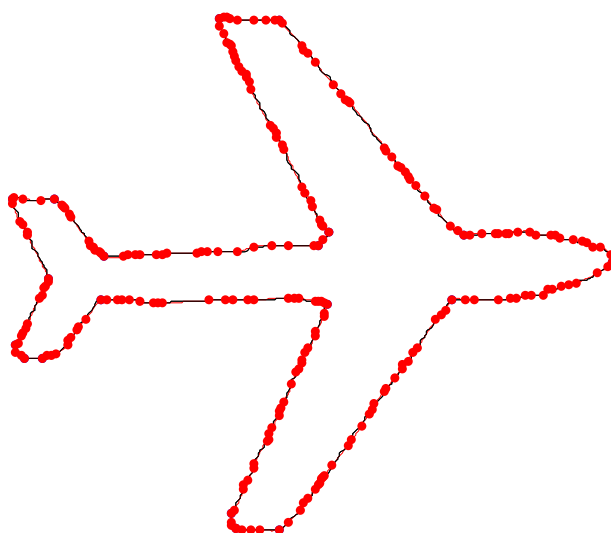
Table 4.13 Evaluation of algorithm 4.1 for object “Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	531	30	257	1
1293	27	254	30	39	2
1293	27	174	30	23	3

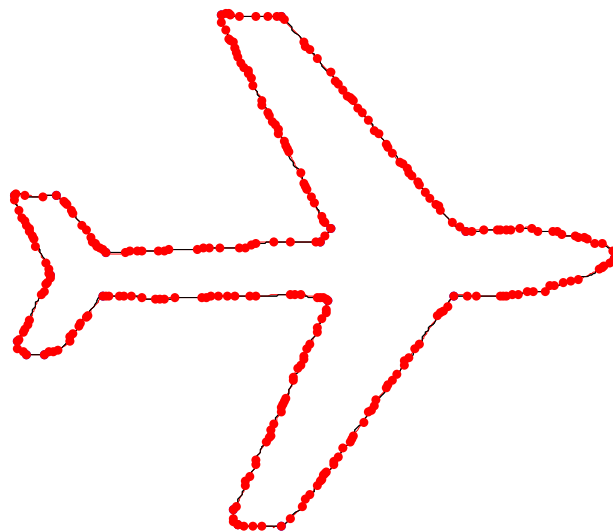
(Figure 4.51) to (Figure 4.53) shows the fitted curve over object contour at different iterations for algorithm 4.2 at threshold values of 1,2 and 3 respectively.



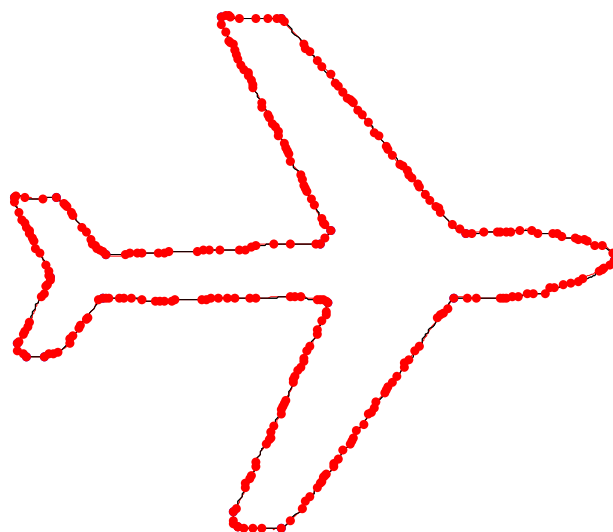
(a) At iteration = 1



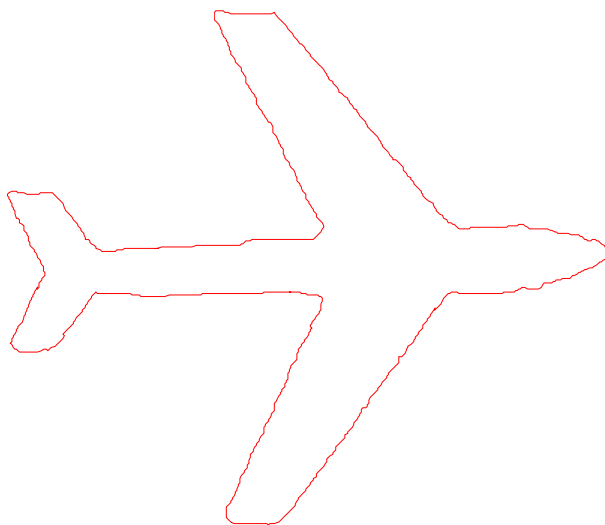
(b) At iteration = 10



(c) At iteration = 20

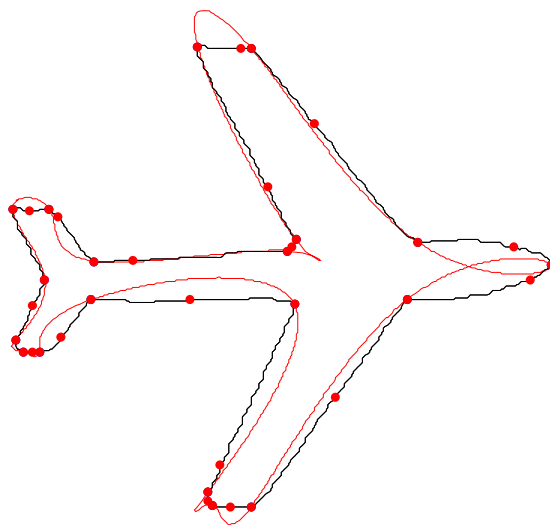


(d) At iteration = 30

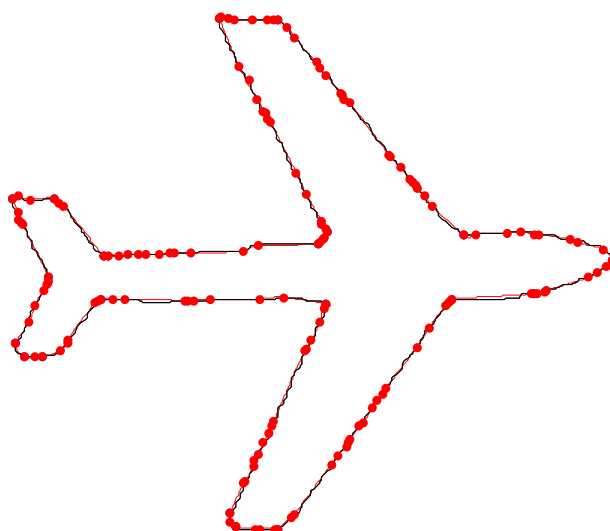


(e) Approximated spline for object “Plane”

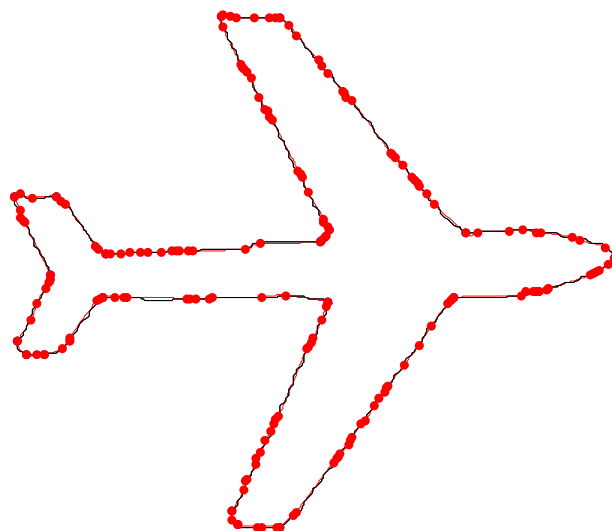
**Figure 4.51 Algorithm 4.2: Demonstration of spline fitting at each iteration using
threshold =1**



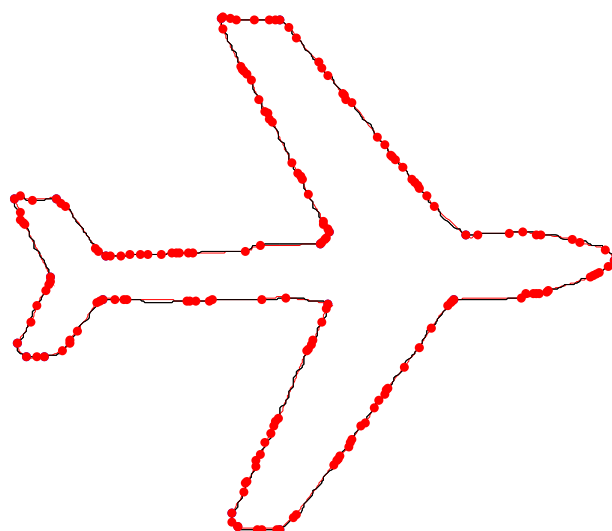
(a) At iteration = 1



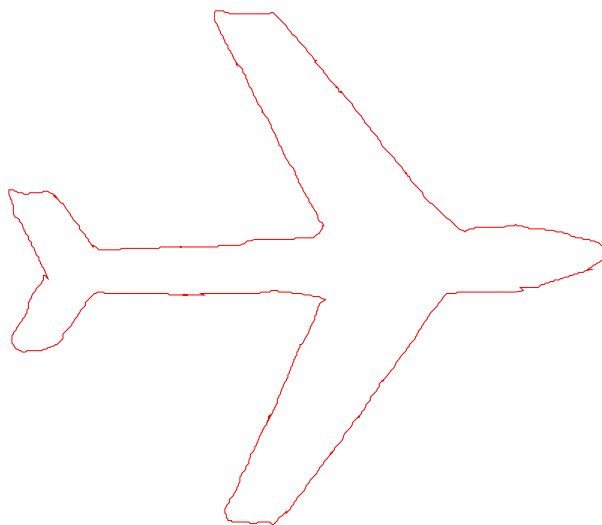
(b) At iteration = 10



(c) At iteration = 20

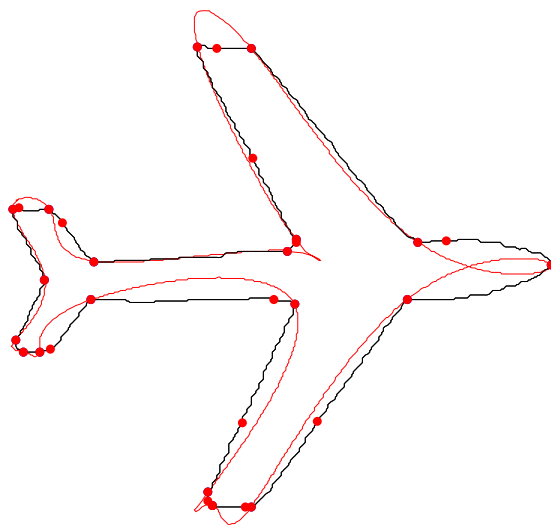


(d) At iteration = 30

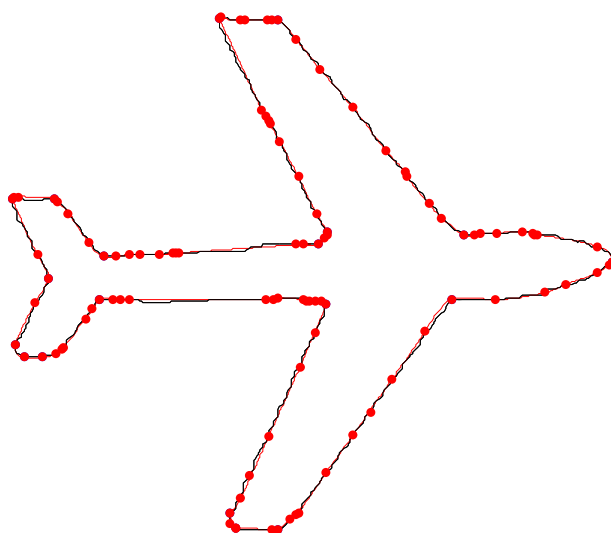


(e) Approximated spline for object “Plane”

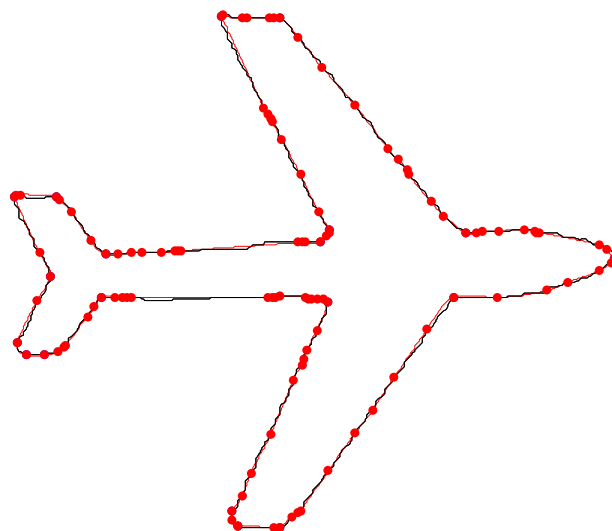
**Figure 4.52 Algorithm 4.2: Demonstration of spline fitting at each iteration using
threshold =2**



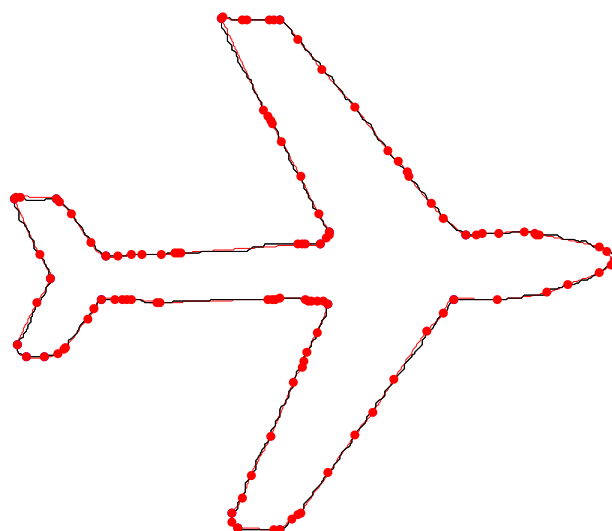
(a) At iteration = 1



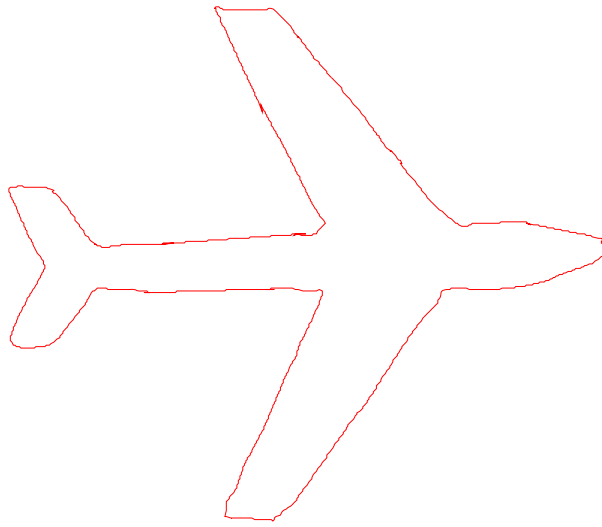
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 30



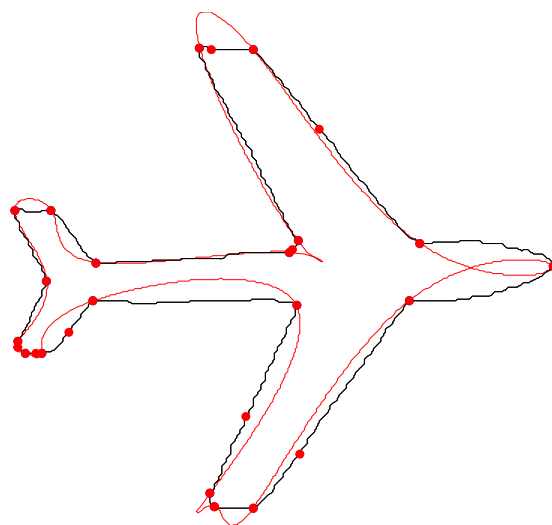
(e) Approximated spline for object “Plane”

Figure 4.53 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =3

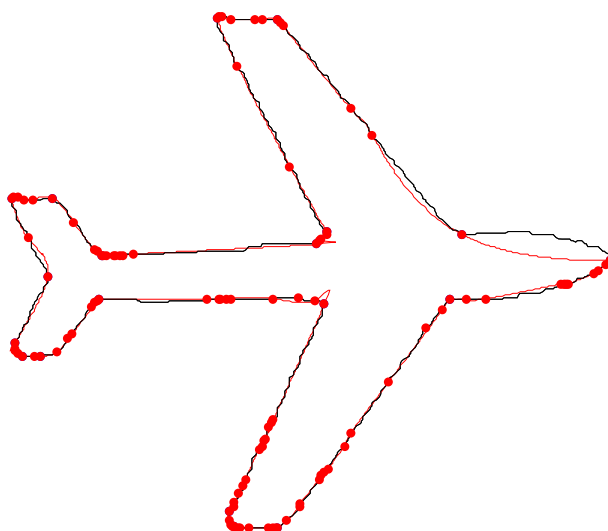
Table 4.14 Evaluation of algorithm 4.2 for object “Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	285	30	78	1
1293	27	150	30	27	2
1293	27	86	30	18	3

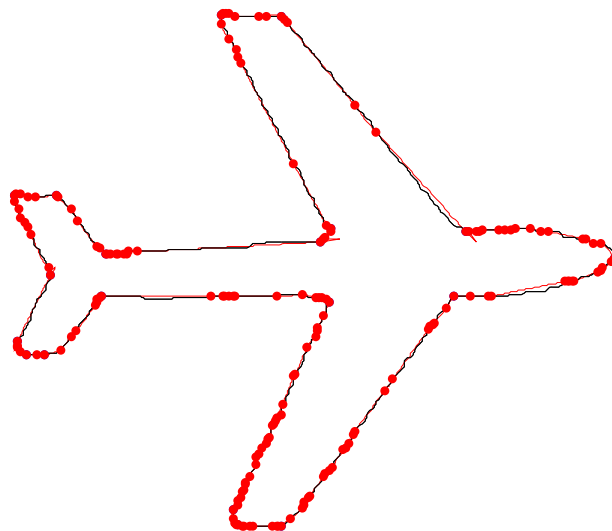
(Figure 4.54) to (Figure 4.56) shows the fitted curve over object contour at different iterations for algorithm 4.3 at threshold values of 1,2 and 3 respectively.



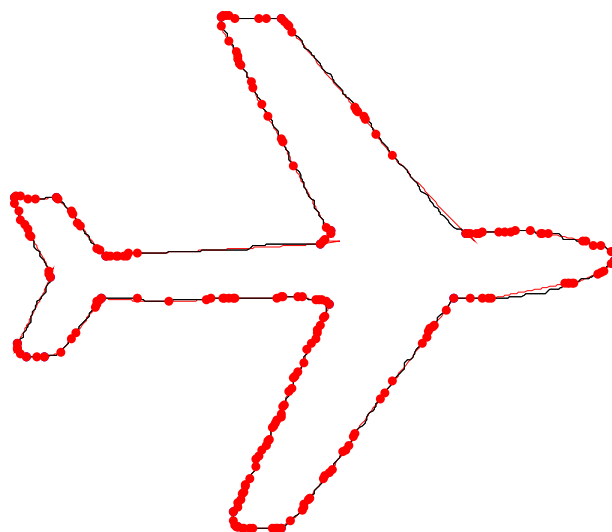
(a) At iteration = 1



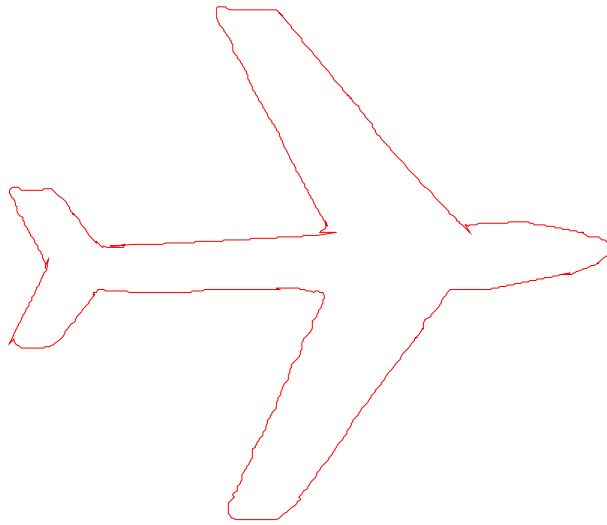
(b) At iteration = 10



(c) At iteration = 20

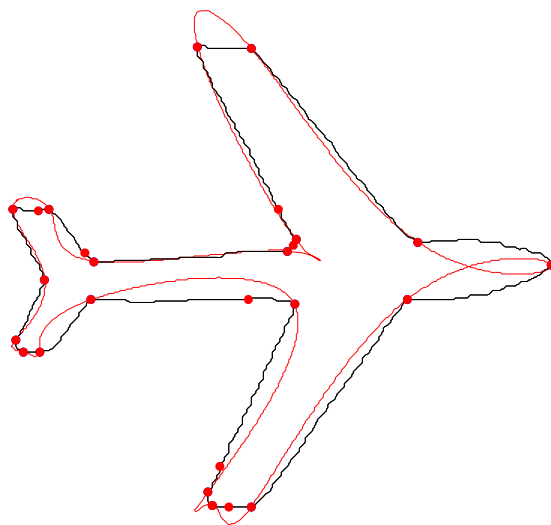


(d) At iteration = 30

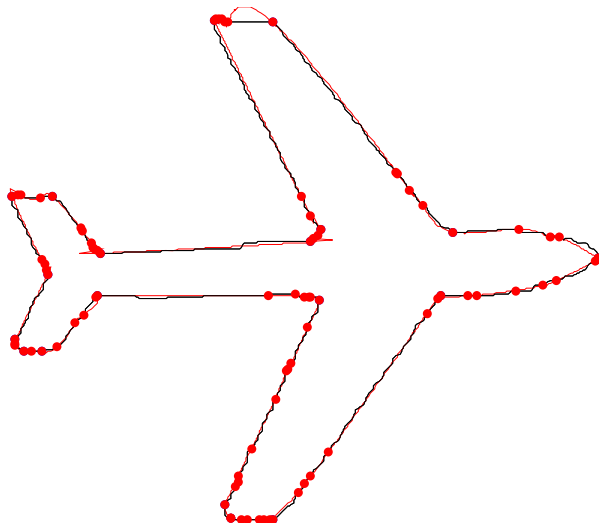


(e) Approximated spline for object “Plane”

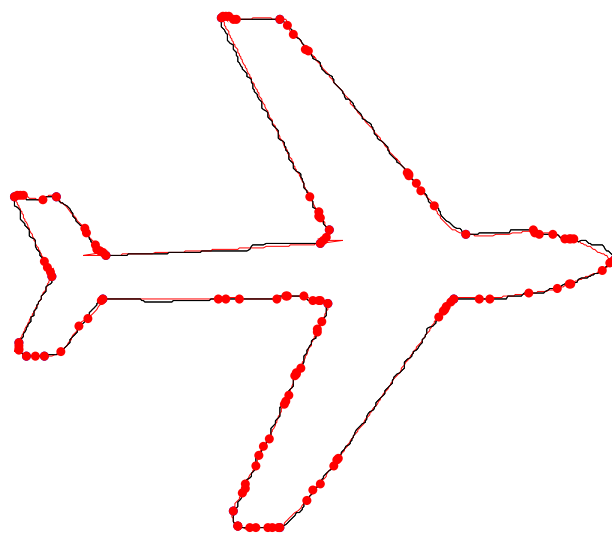
**Figure 4.54 Algorithm 4.3: Demonstration of spline fitting at each iteration using
threshold =1**



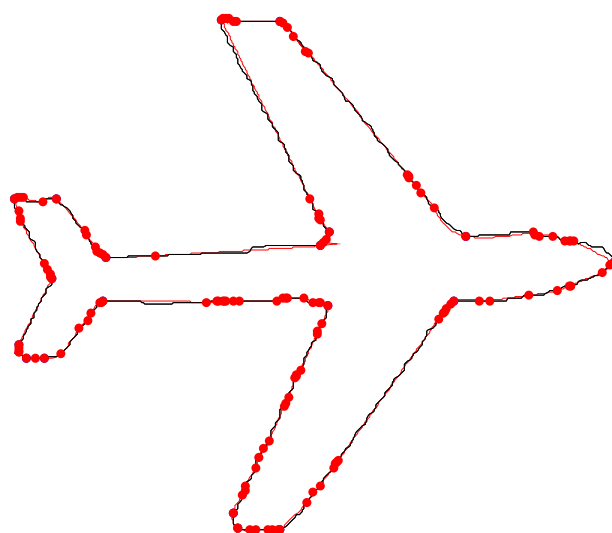
(a) At iteration = 1



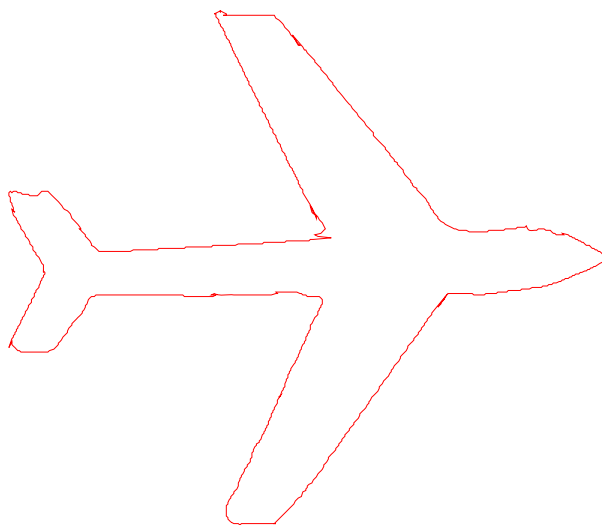
(b) At iteration = 10



(c) At iteration = 20

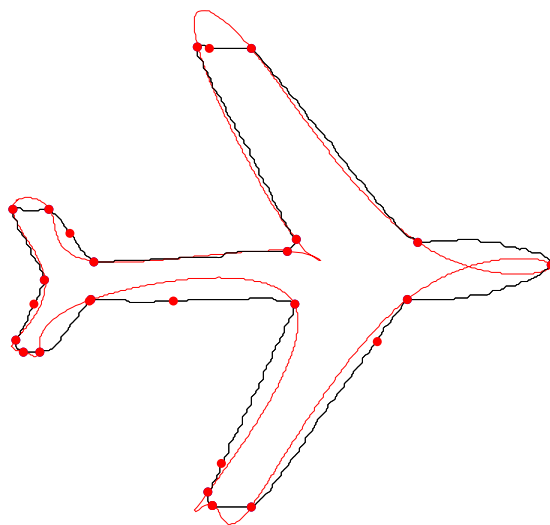


(d) At iteration = 30

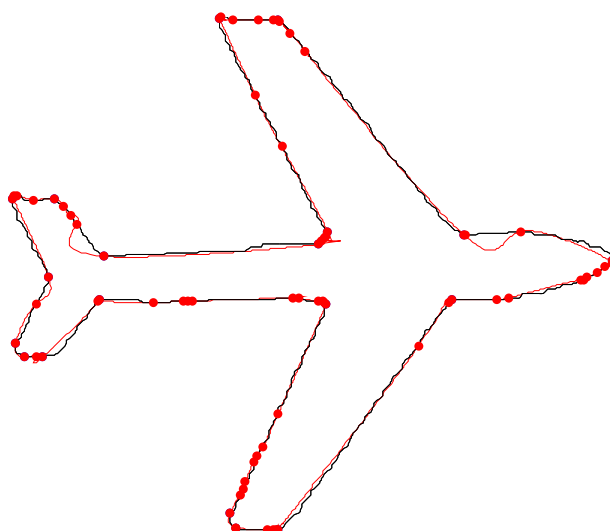


(e) Approximated spline for object “Plane”

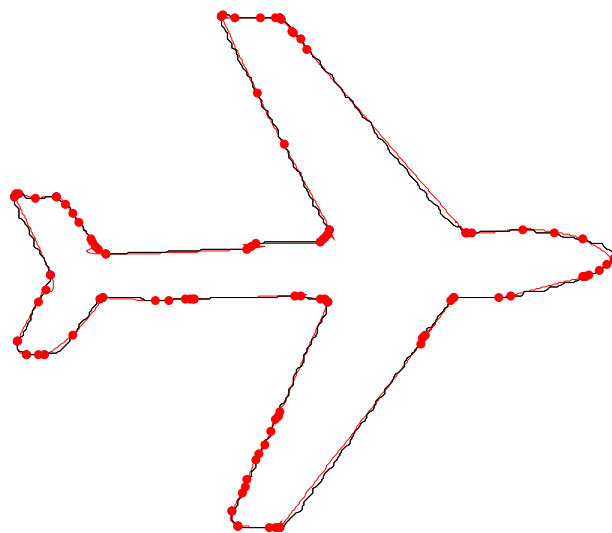
**Figure 4.55 Algorithm 4.3: Demonstration of spline fitting at each iteration using
threshold =2**



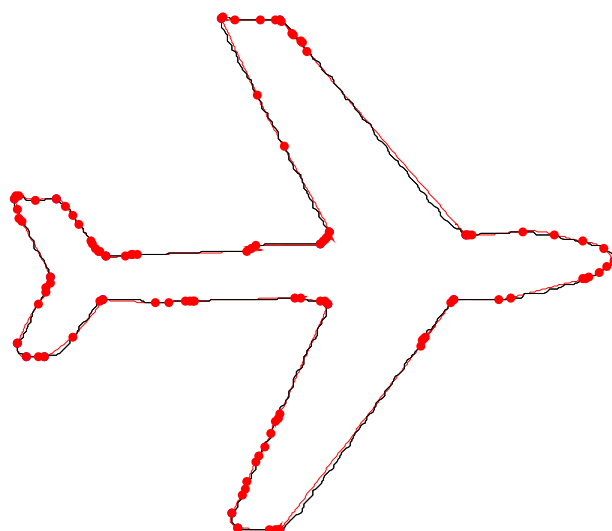
(a) At iteration = 1



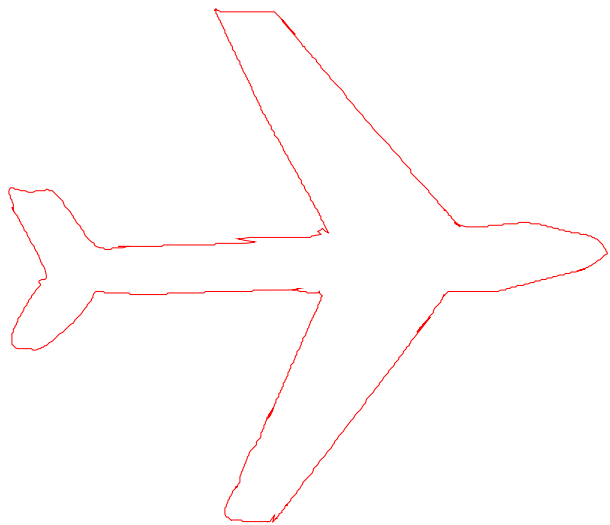
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 30



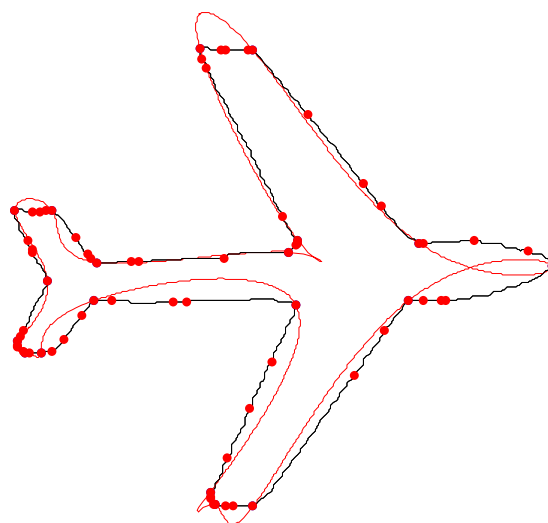
(e) Approximated spline for object “Plane”

Figure 4.56 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =3

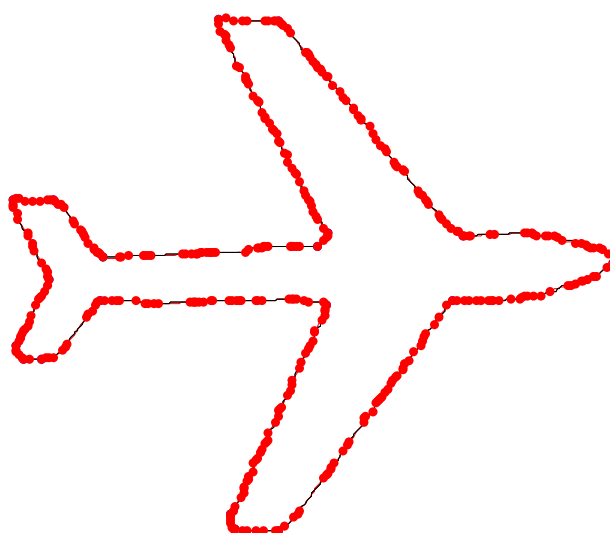
Table 4.15 Evaluation of algorithm 4.3 for object “Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	190	30	24	1
1293	27	110	30	17	2
1293	27	80	30	16	3

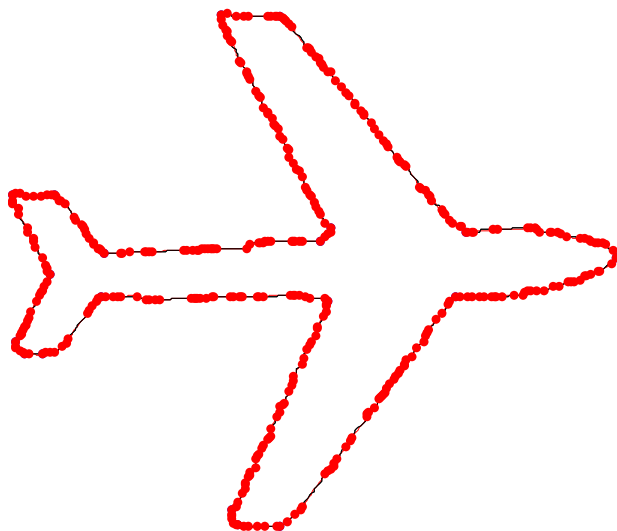
(Figure 4.57) to (Figure 4.59) shows the fitted curve over object contour at different iterations for algorithm 4.4 at threshold values of 1,2 and 3 respectively.



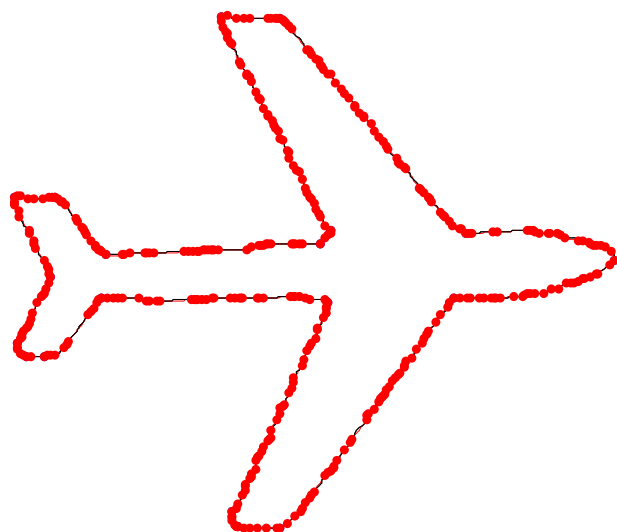
(a) At iteration = 1



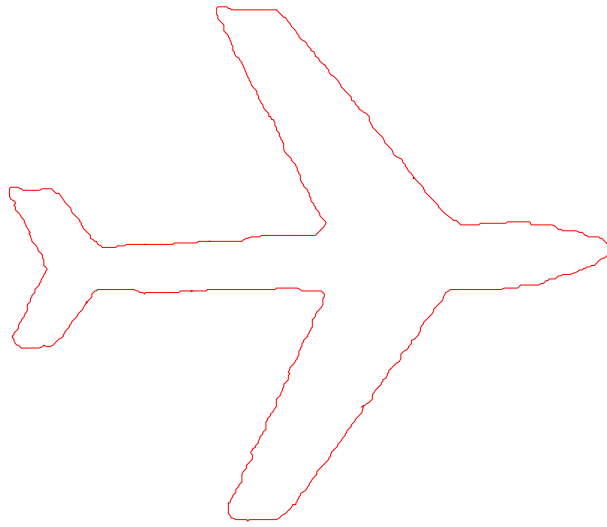
(b) At iteration = 10



(c) At iteration = 20

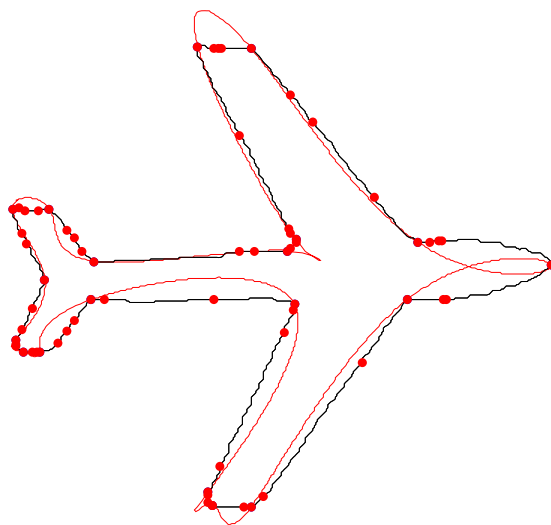


(d) At iteration = 30

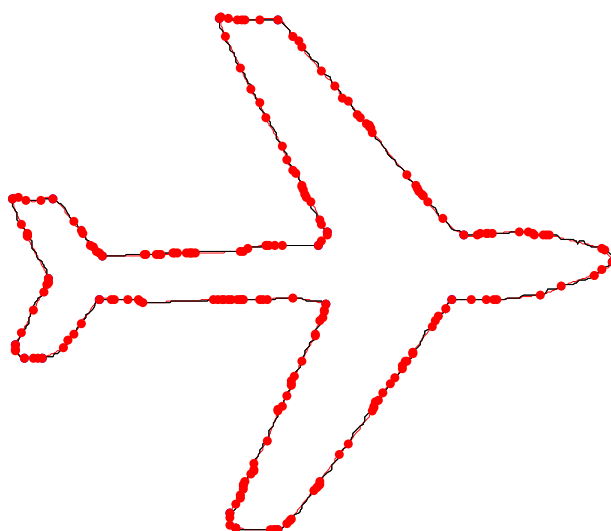


(e) Approximated spline for object “Plane”

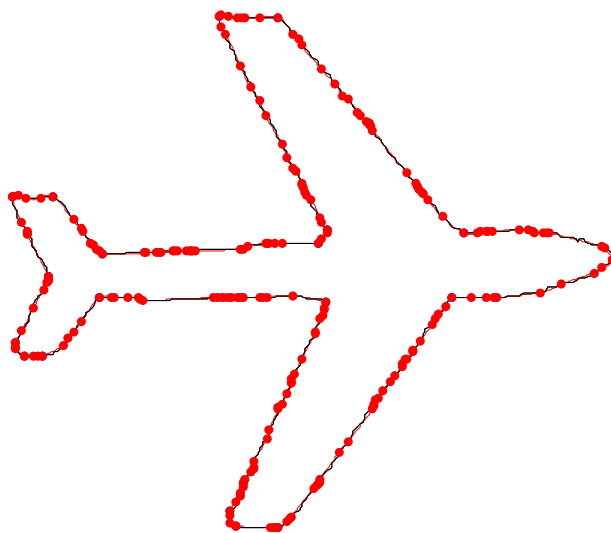
**Figure 4.57 Algorithm 4.4: Demonstration of spline fitting at each iteration using
threshold =1**



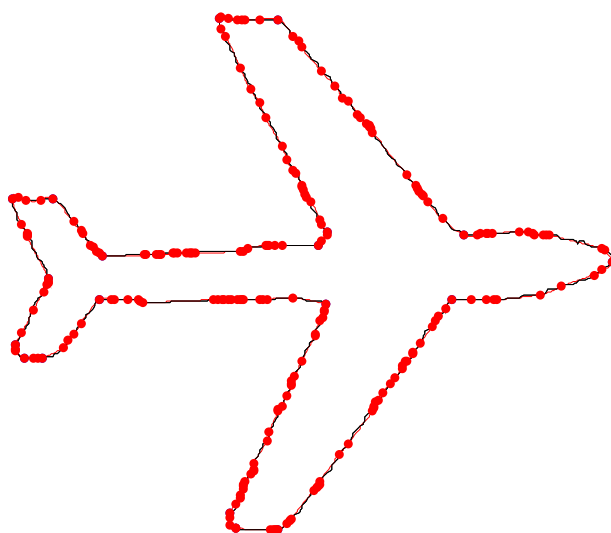
(a) At iteration = 1



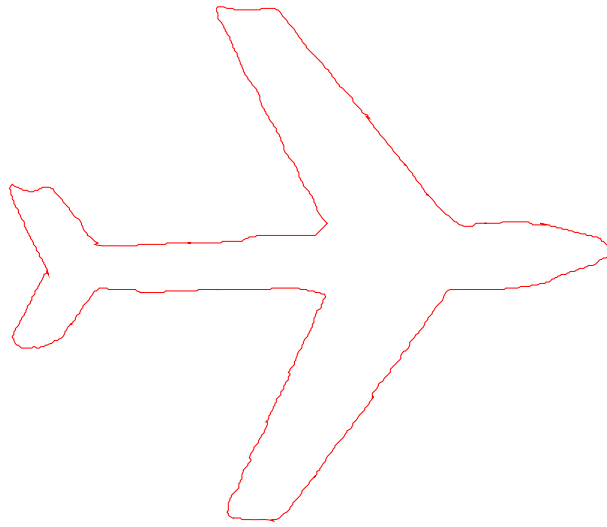
(b) At iteration = 10



(c) At iteration = 20

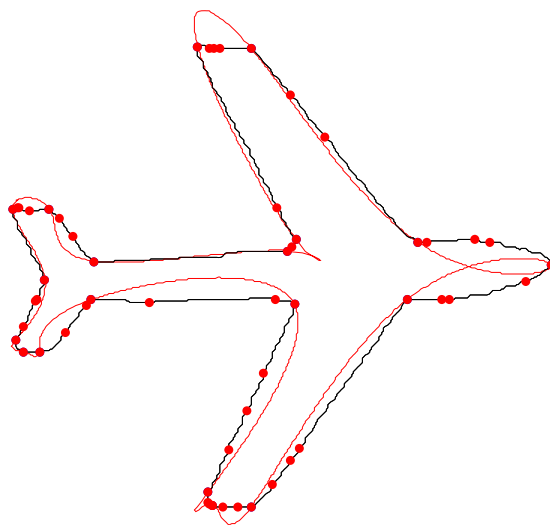


(d) At iteration = 30

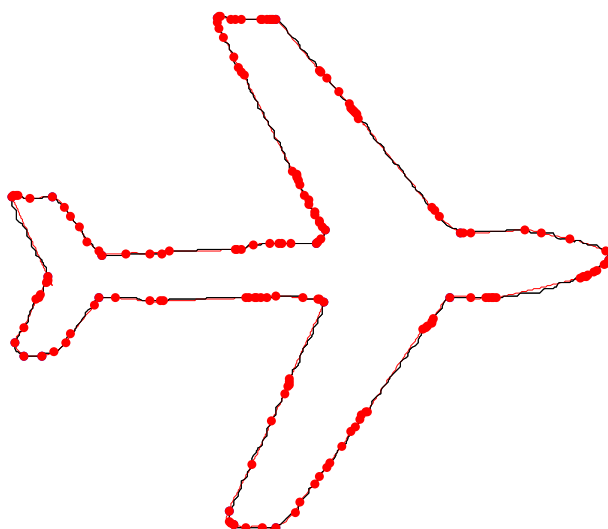


(e) Approximated spline for object “Plane”

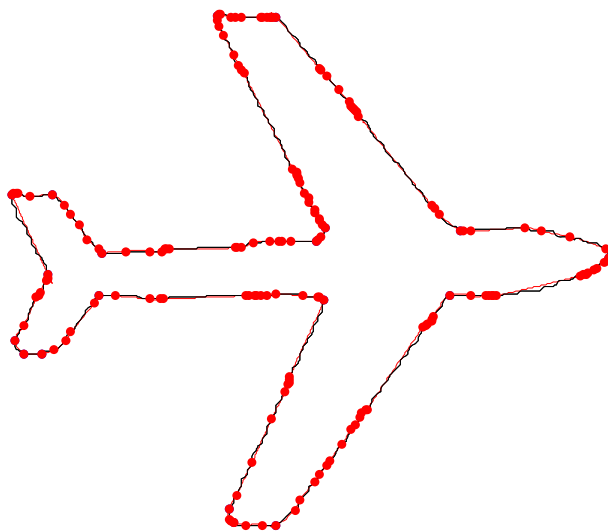
**Figure 4.58 Algorithm 4.4: Demonstration of spline fitting at each iteration using
threshold =2**



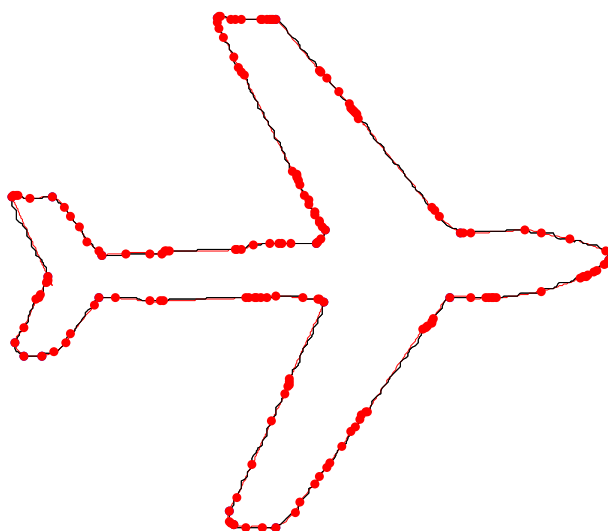
(a) At iteration = 1



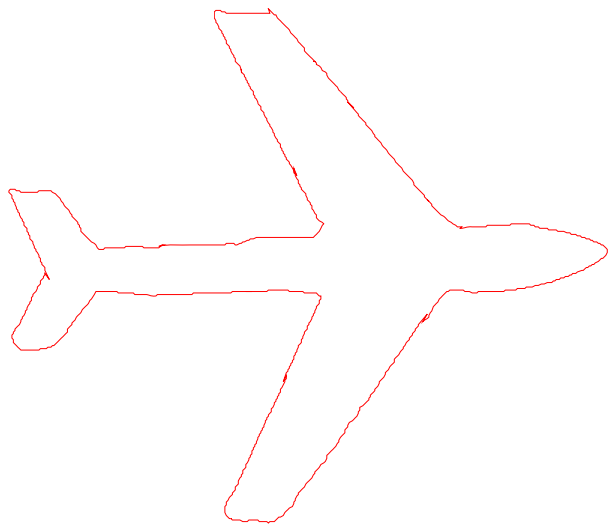
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 30



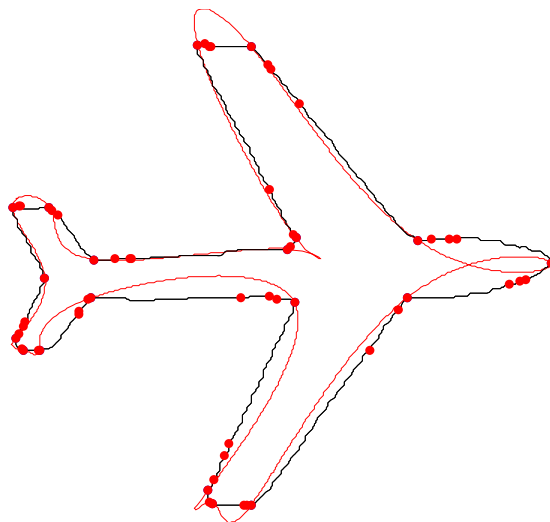
(e) Approximated spline for object “Plane”

Figure 4.59 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =3

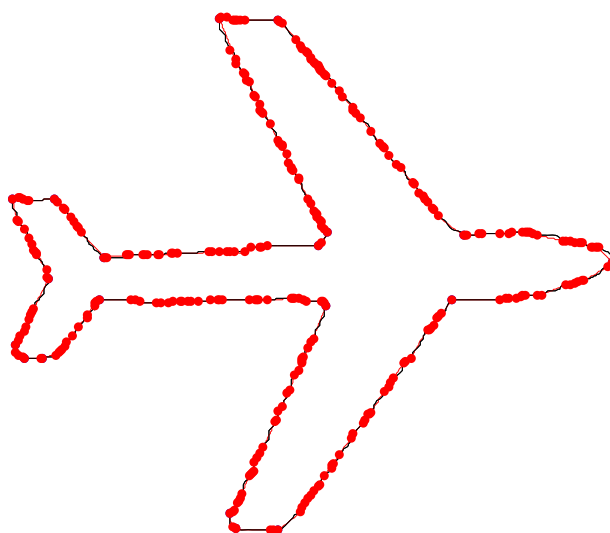
Table 4.16 Evaluation of algorithm 4.4 for object “Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	971	30	1441	1
1293	27	572	30	293	2
1293	27	507	30	192	3

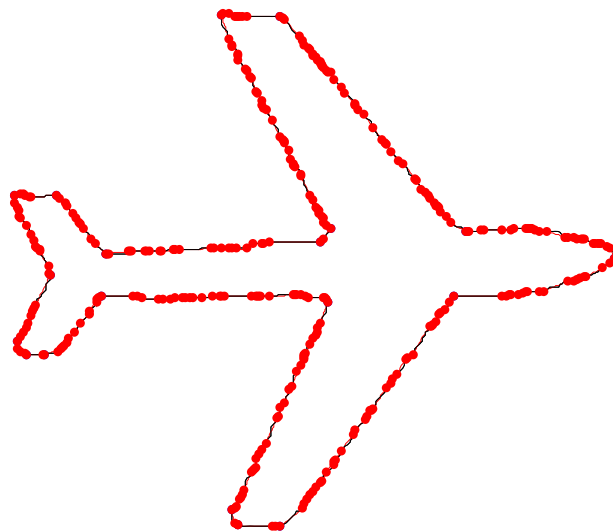
(Figure 4.60) to (Figure 4.62) shows the fitted curve over object contour at different iterations for algorithm 4.5 at threshold values of 1,2 and 3 respectively.



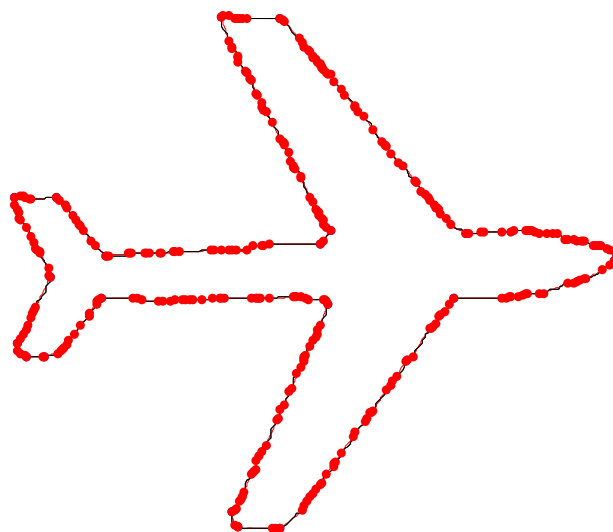
(a) At iteration = 1



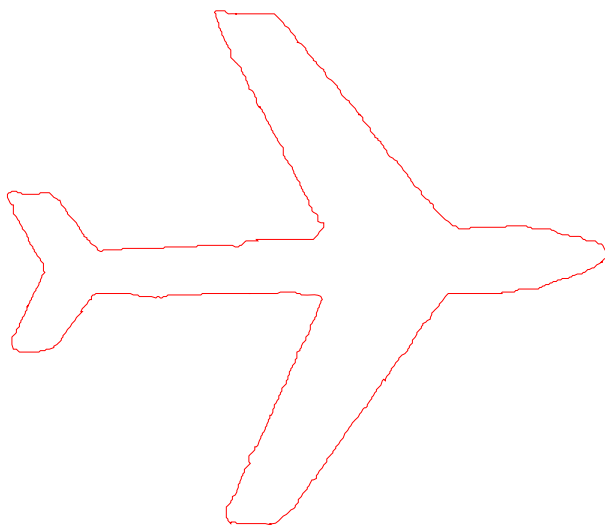
(b) At iteration = 10



(c) At iteration = 20

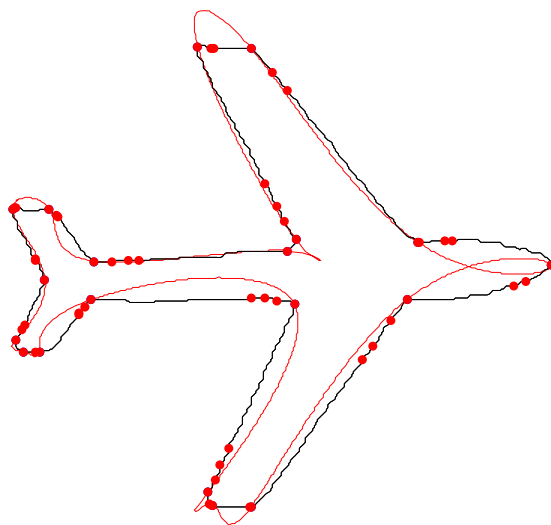


(d) At iteration = 30

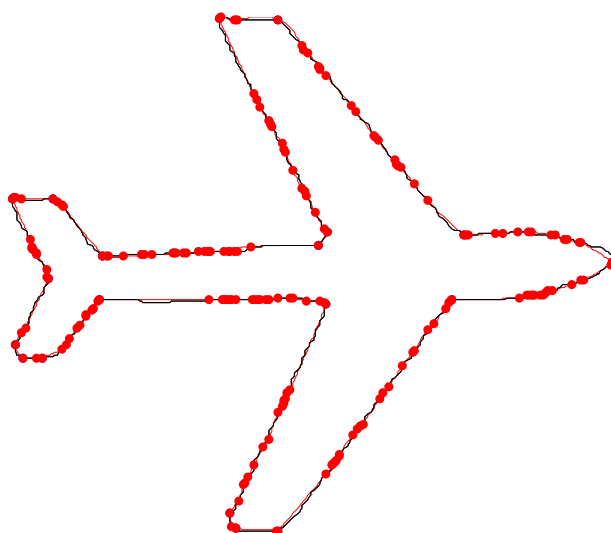


(e) Approximated spline for object “Plane”

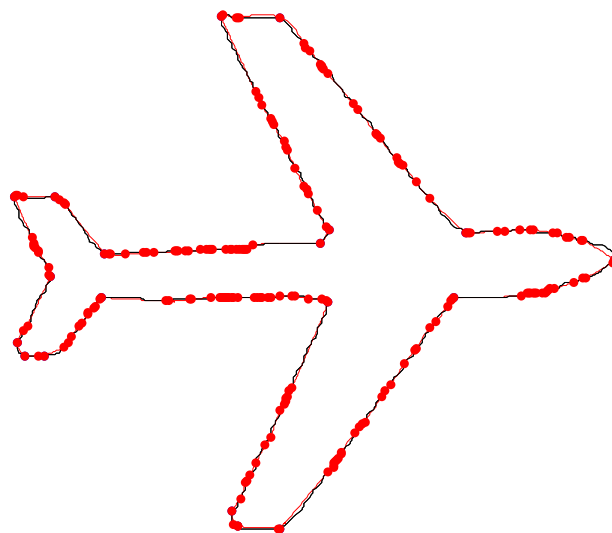
**Figure 4.60 Algorithm 4.5: Demonstration of spline fitting at each iteration using
threshold =1**



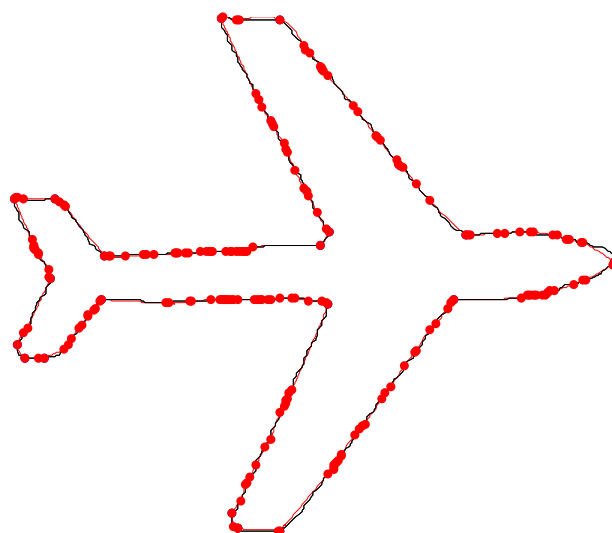
(a) At iteration = 1



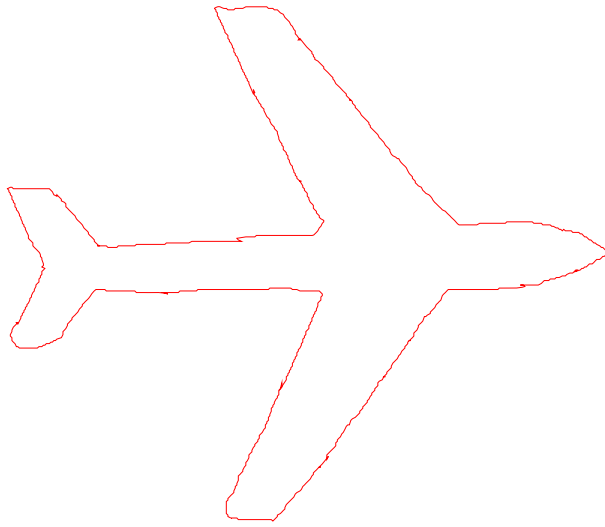
(b) At iteration = 10



(c) At iteration = 20

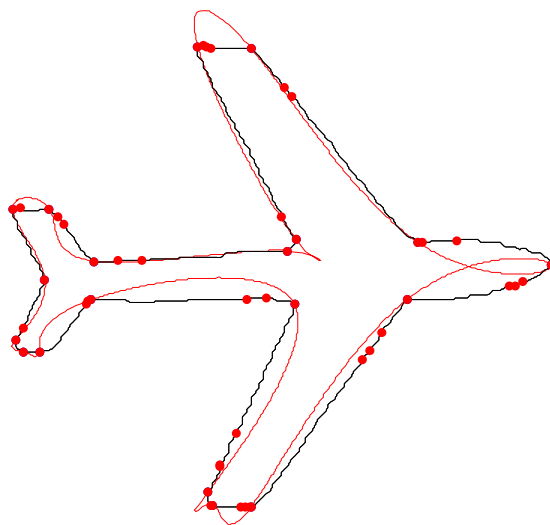


(d) At iteration = 30

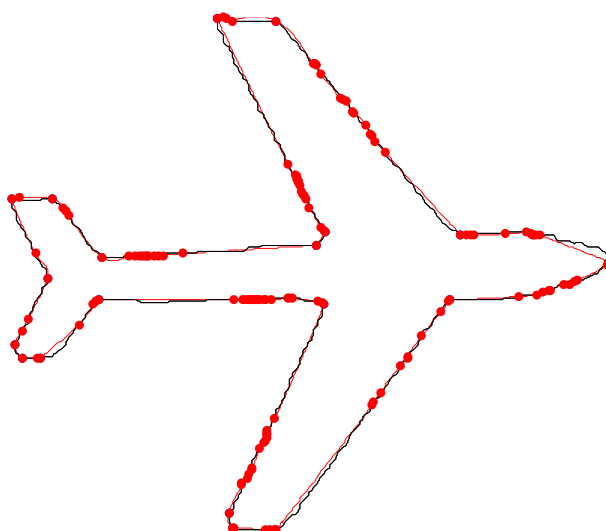


(e) Approximated spline for object “Plane”

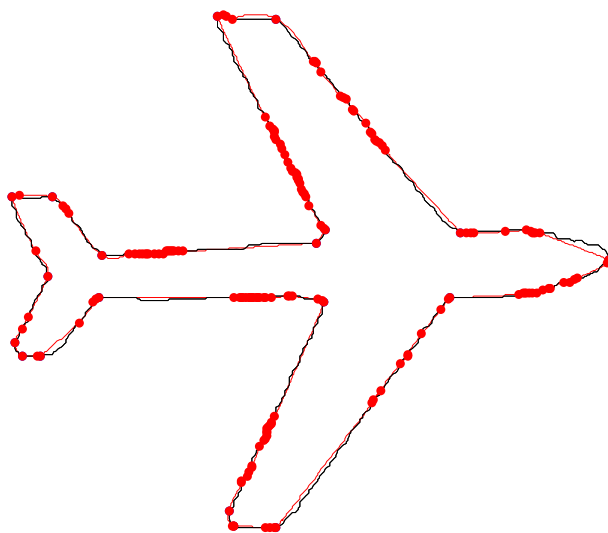
**Figure 4.61 Algorithm 4.5: Demonstration of spline fitting at each iteration using
threshold =2**



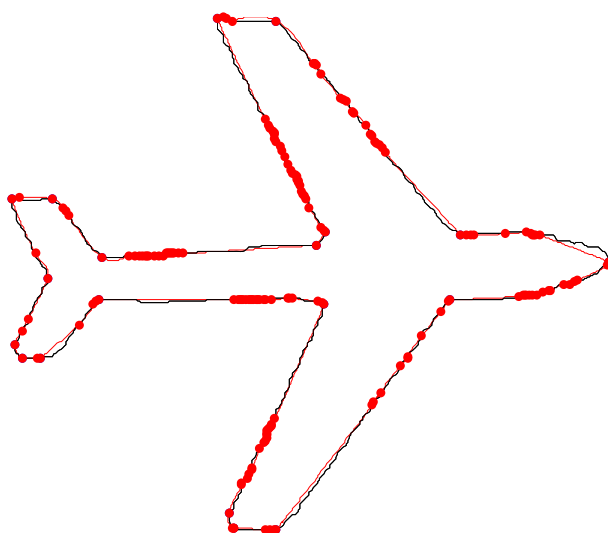
(a) At iteration = 1



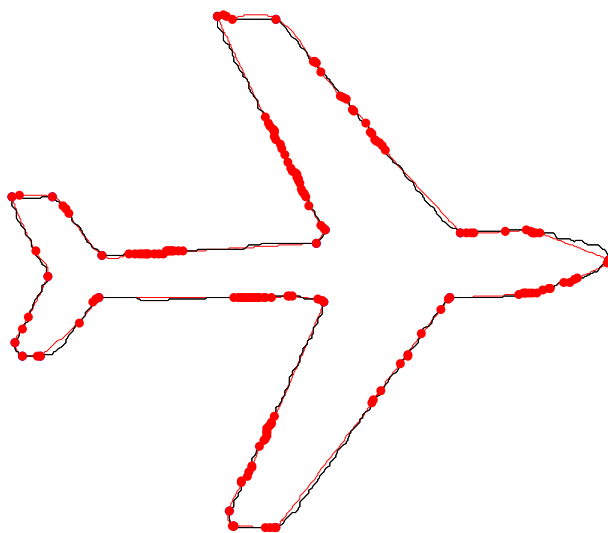
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 30



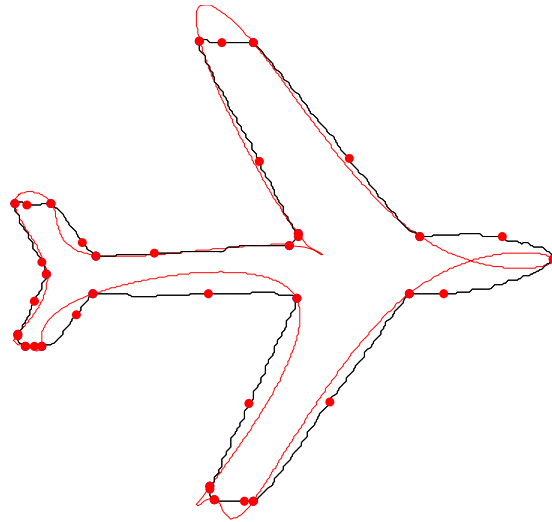
(e) Approximated spline for object “Plane”

Figure 4.62 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =3

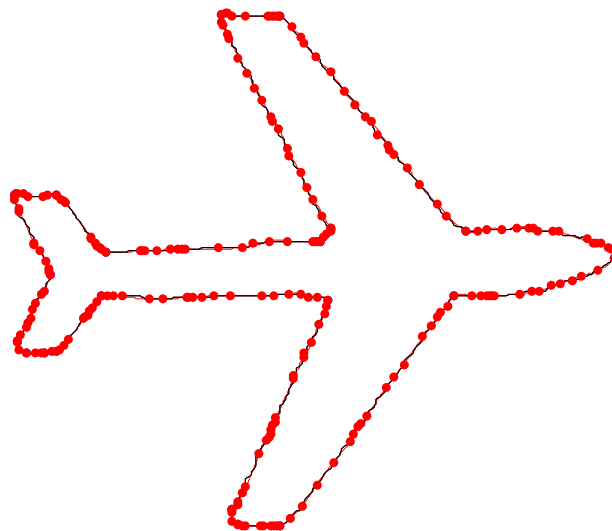
Table 4.17 Evaluation of algorithm 4.5 for object “Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	863	30	1257	1
1293	27	390	30	130	2
1293	27	395	30	98	3

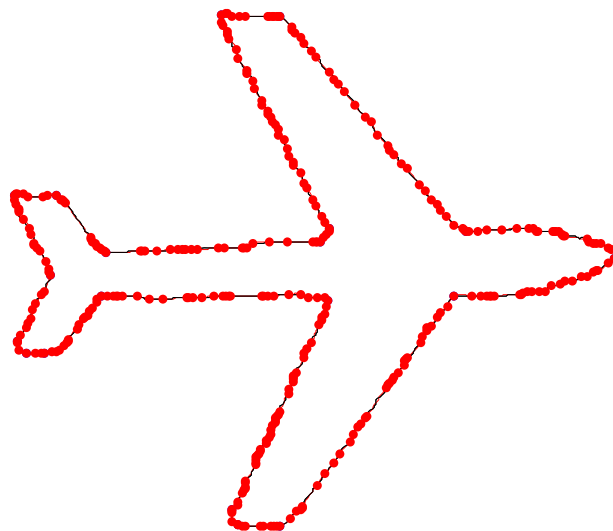
(Figure 4.63) to (Figure 4.65) shows the fitted curve over object contour at different iterations for algorithm 4.6 at threshold values of 1,2 and 3 respectively.



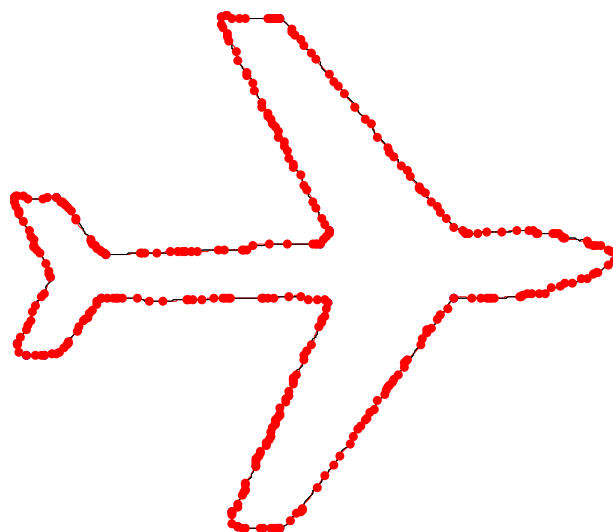
(a) At iteration = 1



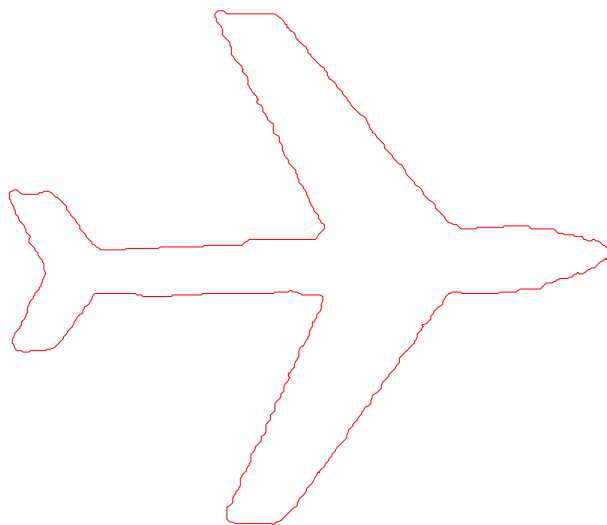
(b) At iteration = 5



(c) At iteration = 10

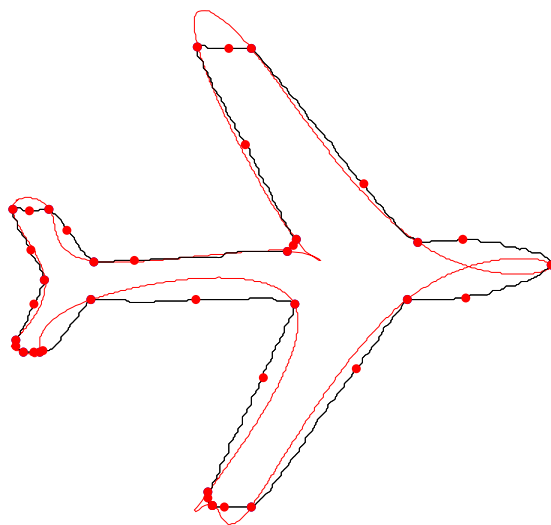


(d) At iteration = 15

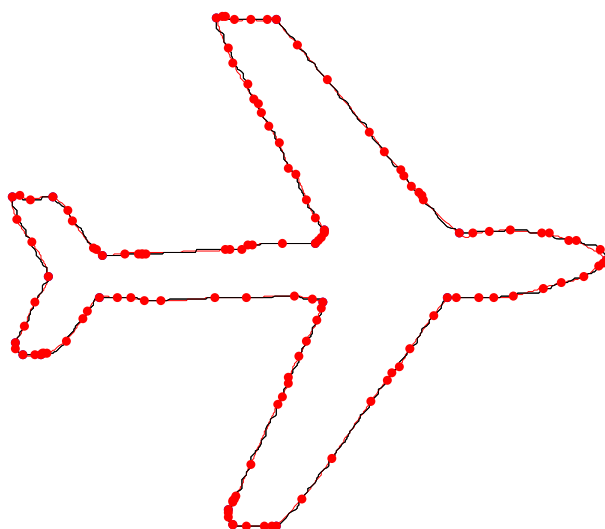


(e) Approximated spline for object “Plane”

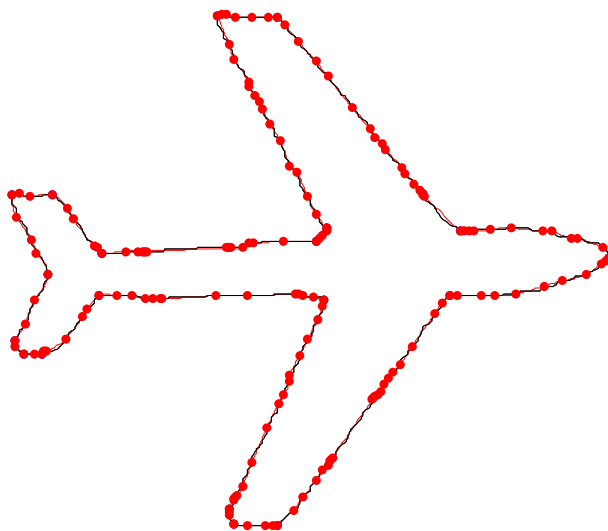
**Figure 4.63 Algorithm 4.6: Demonstration of spline fitting at each iteration using
threshold =1**



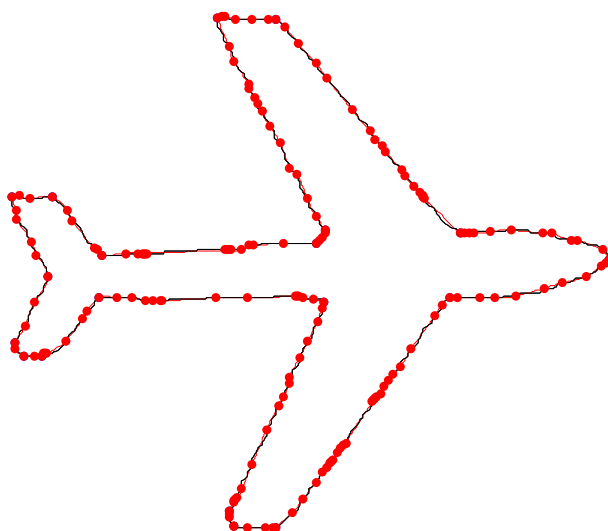
(a) At iteration = 1



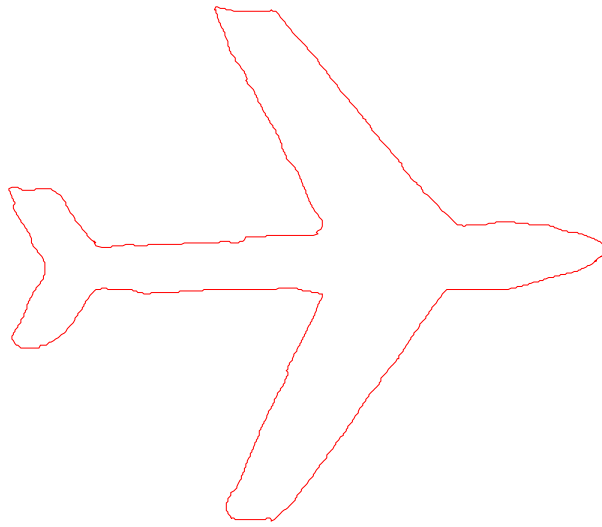
(b) At iteration = 5



(c) At iteration = 10

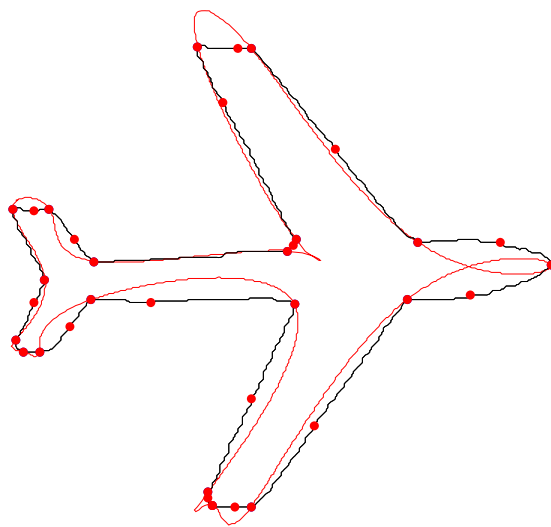


(d) At iteration = 15

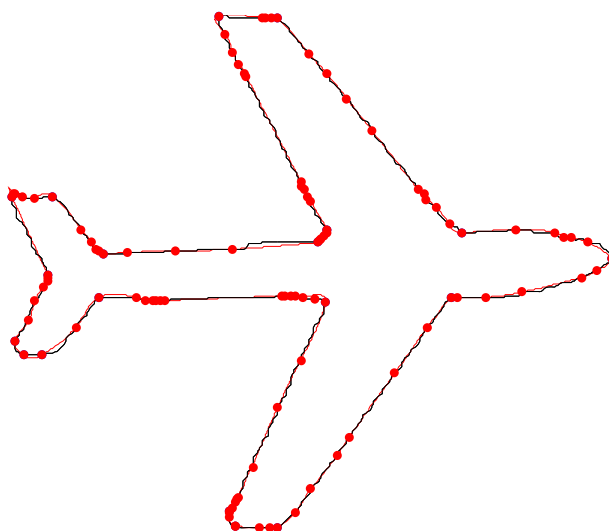


(e) Approximated spline for object “Plane”

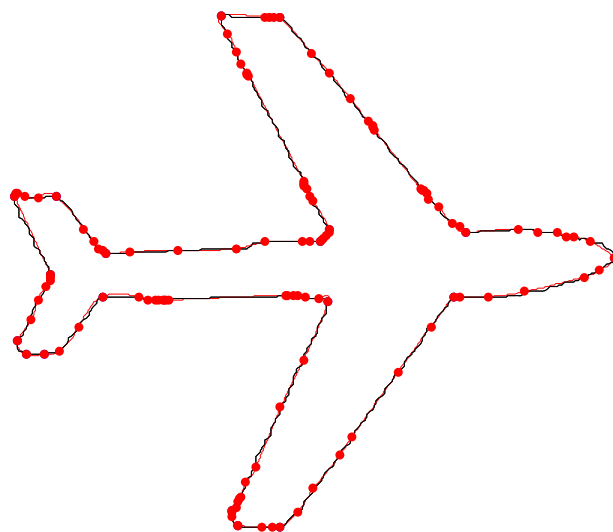
**Figure 4.64 Algorithm 4.6: Demonstration of spline fitting at each iteration using
threshold =2**



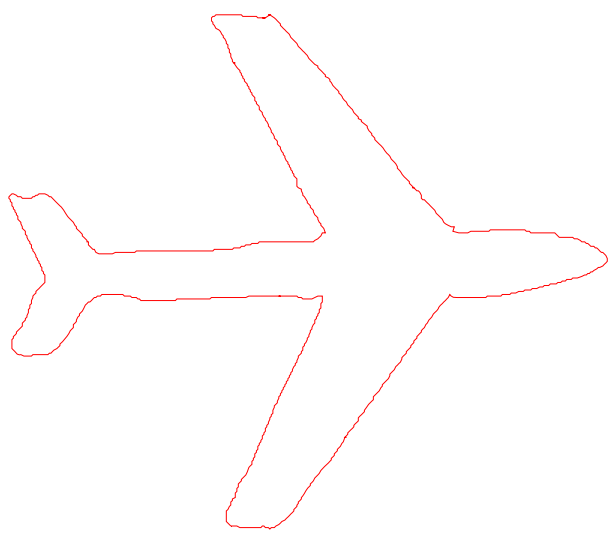
(a) At iteration = 1



(b) At iteration = 5



(c) At iteration = 10



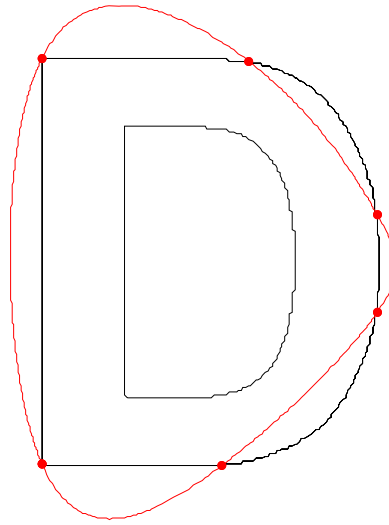
(d) Approximated spline for object “Plane”

Figure 4.65 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =3

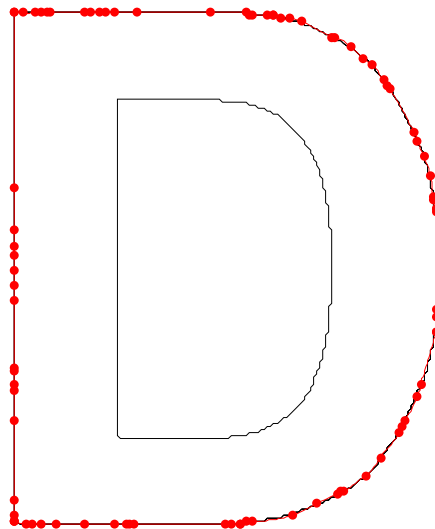
Table 4.18 Evaluation of algorithm 4.6 for object “Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	332	30	64	1
1293	27	142	30	12	2
1293	27	92	30	5	3

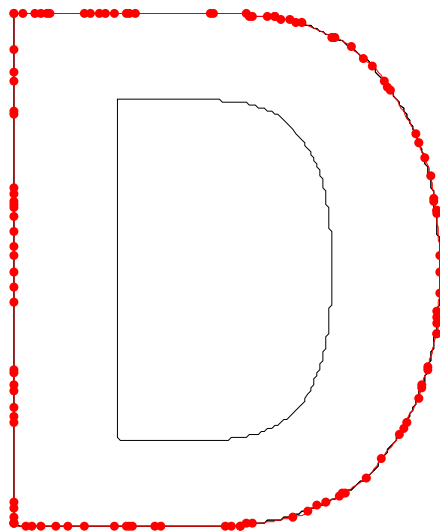
(Figure 4.66) to (Figure 4.68) shows the fitted curve over object contour at different iterations for algorithm 4.1 at threshold values of 1,2 and 3 respectively.



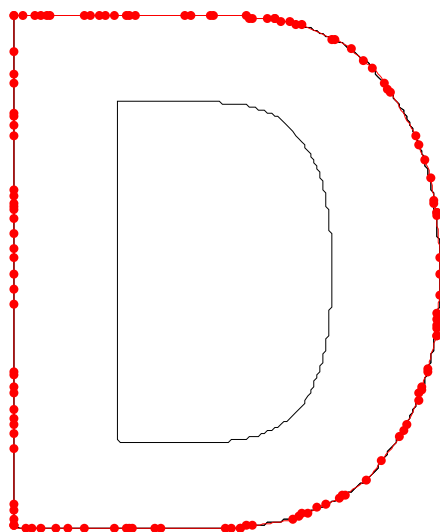
(a) At iteration = 1



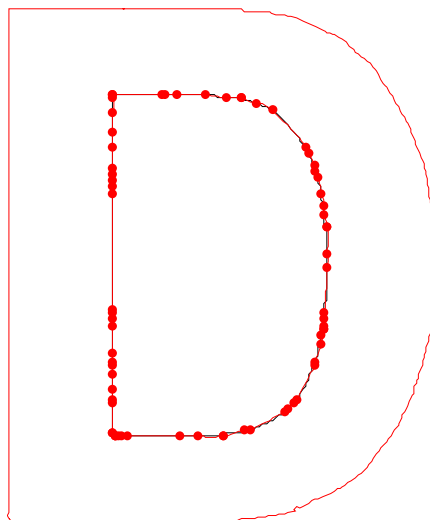
(b) At iteration = 10



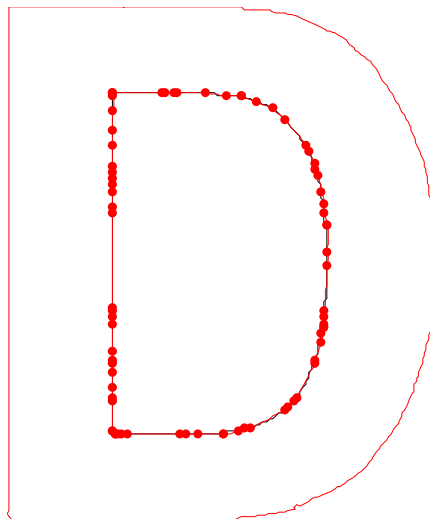
(c) At iteration = 20



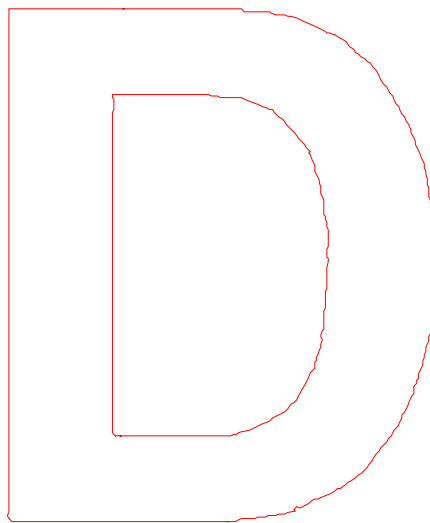
(d) At iteration = 30



(e) At iteration = 40

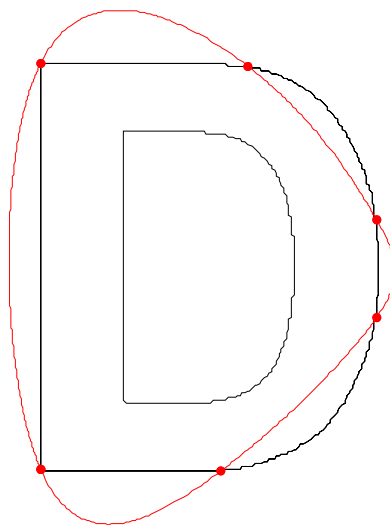


(f) At iteration = 50

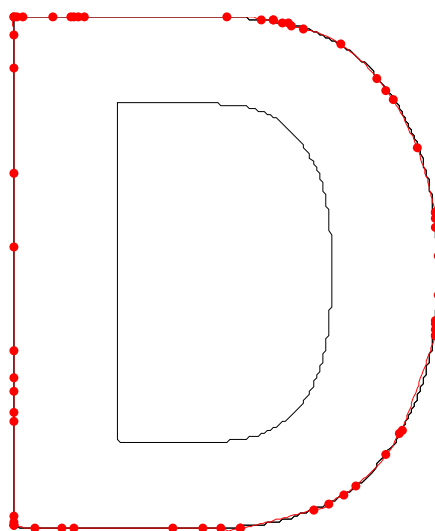


(g) Approximated spline for English alphabet “D”

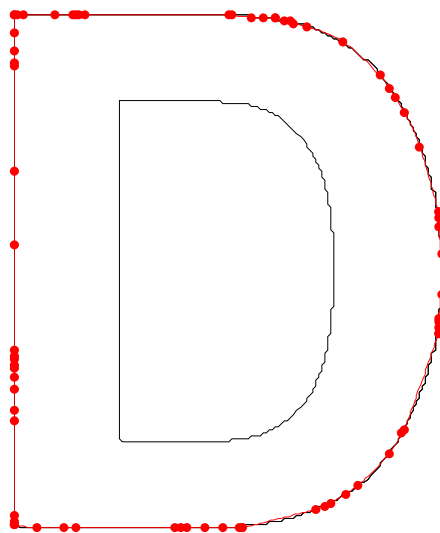
**Figure 4.66 Algorithm 4.1: Demonstration of spline fitting at each iteration using
threshold =1**



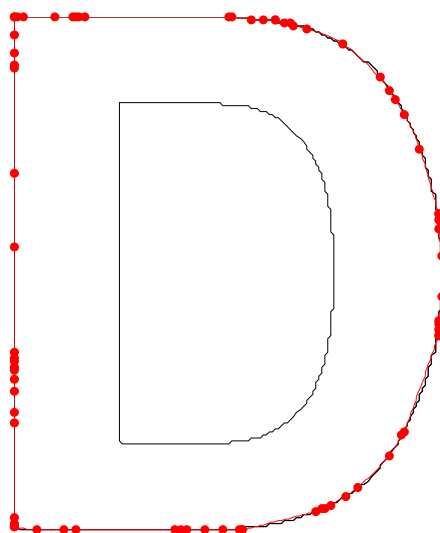
(a) At iteration = 1



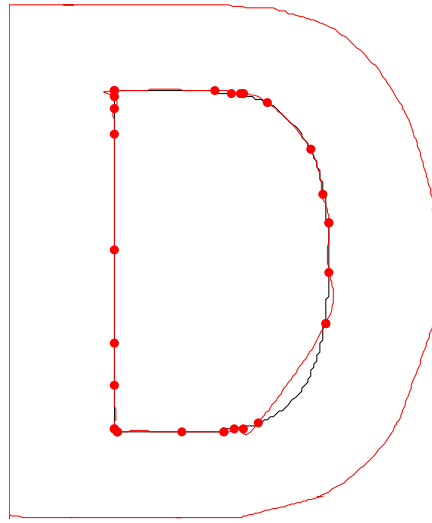
(b) At iteration = 10



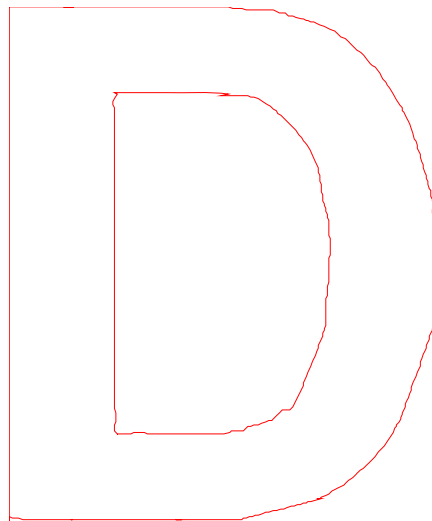
(c) At iteration = 20



(d) At iteration = 30

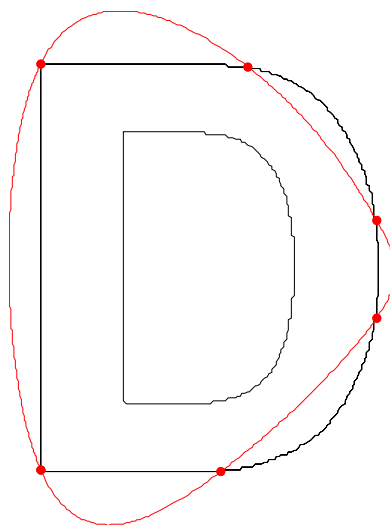


(e) At iteration = 35

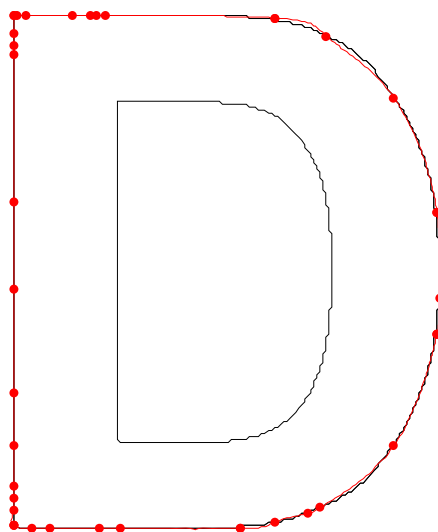


(f) Approximated spline for English alphabet “D”

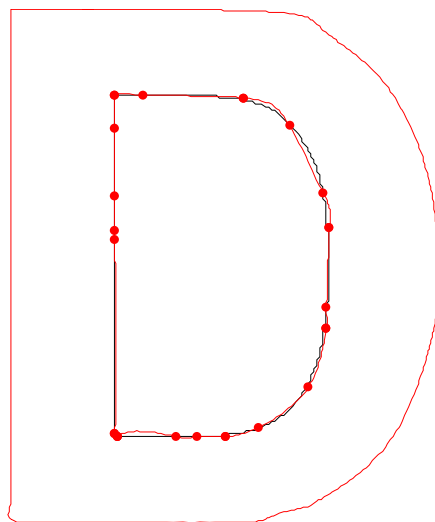
Figure 4.67 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =2



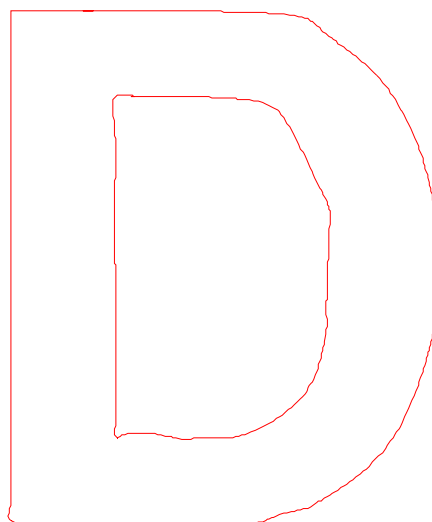
(a) At iteration = 1



(b) At iteration = 10



(c) At iteration = 15



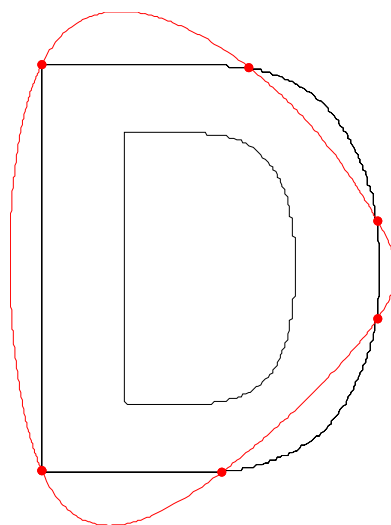
(d) Approximated spline for English alphabet “D”

Figure 4.68 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =3

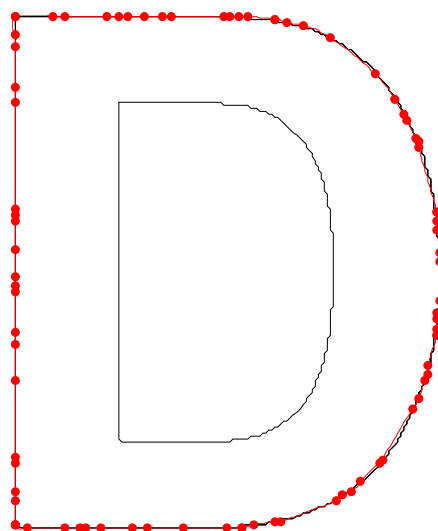
Table 4.19 Evaluation of algorithm 4.1 for English alphabet “D”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	314	55	1.85	1
849	15	155	39	1.09	2
849	15	47	19	0.59	3

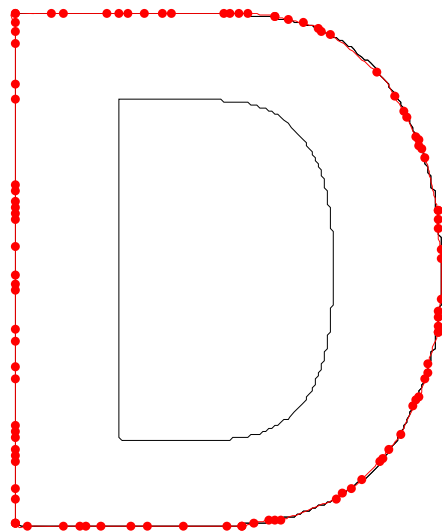
(Figure 4.69) to (Figure 4.71) shows the fitted curve over object contour at different iterations for algorithm 4.2 at threshold values of 1,2 and 3 respectively.



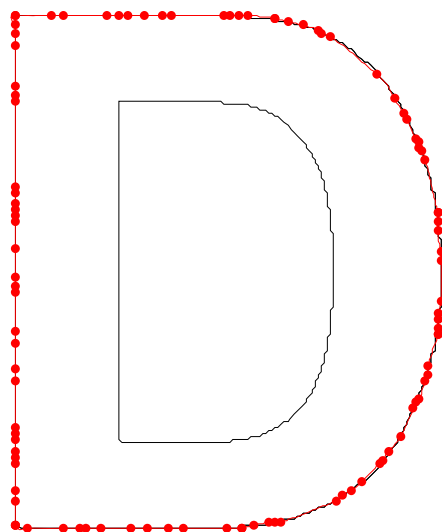
(a) At iteration = 1



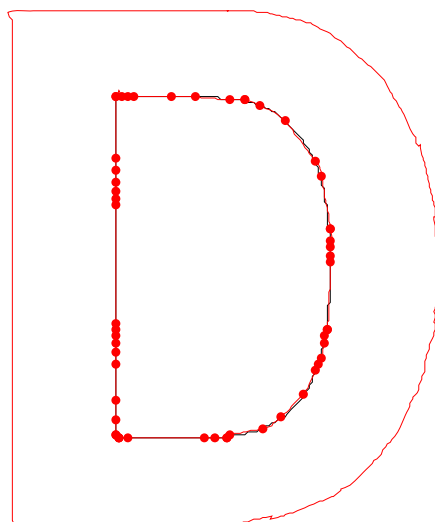
(b) At iteration = 10



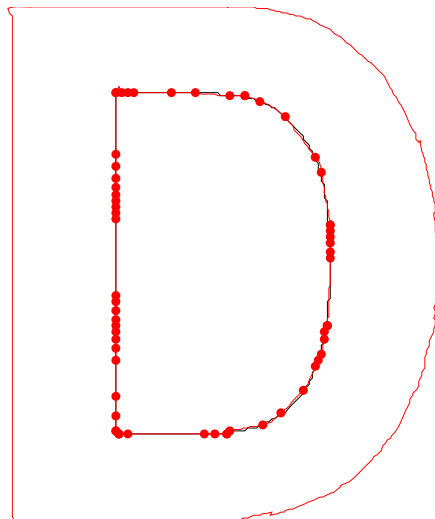
(c) At iteration = 20



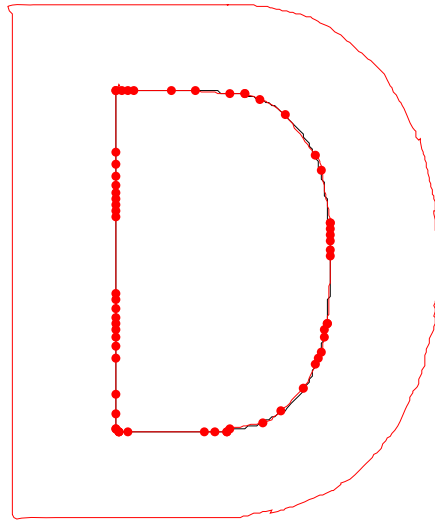
(d) At iteration = 30



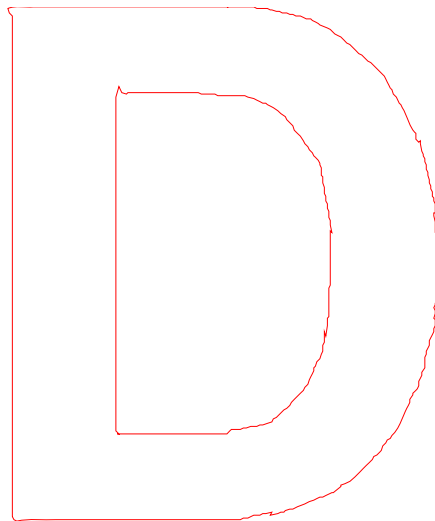
(e) At iteration = 40



(f) At iteration = 50

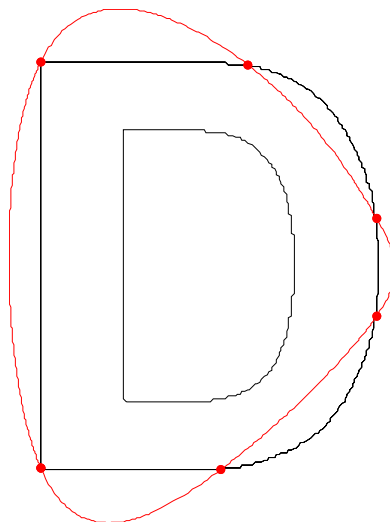


(g) At iteration = 60

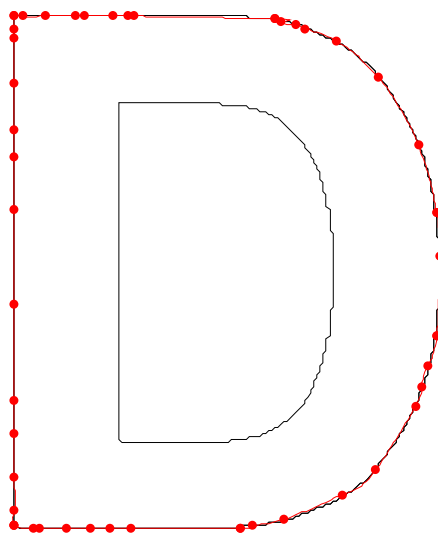


(h) Approximated spline for English alphabet “D”

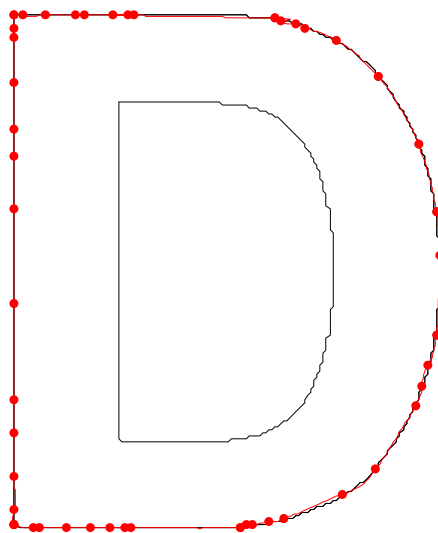
Figure 4.69 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =1



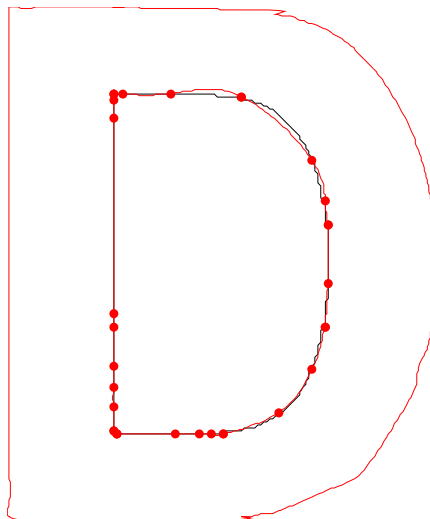
(a) At iteration = 1



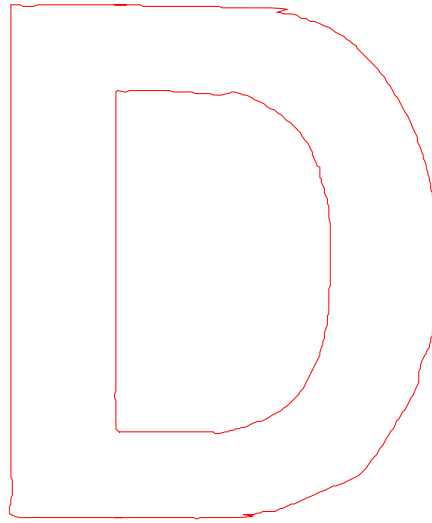
(b) At iteration = 10



(c) At iteration = 20

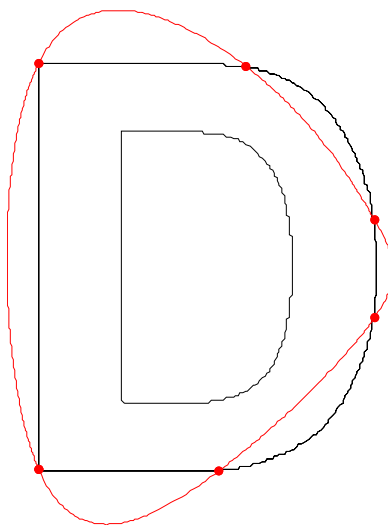


(d) At iteration = 30

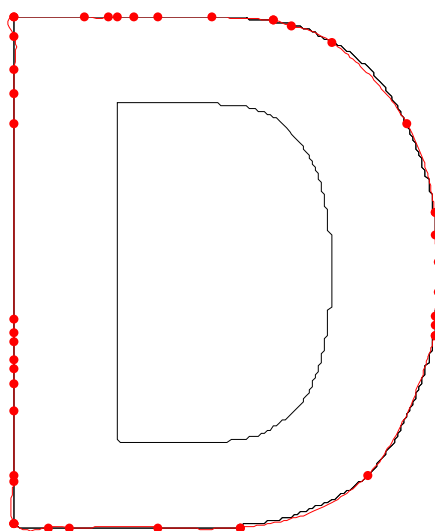


(e) Approximated spline for English alphabet “D”

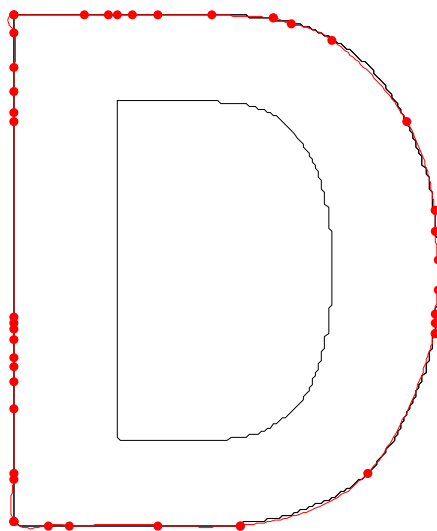
**Figure 4.70 Algorithm 4.2: Demonstration of spline fitting at each iteration using
threshold =2**



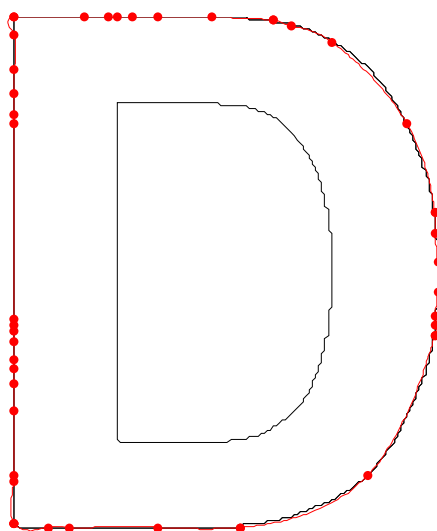
(a) At iteration = 1



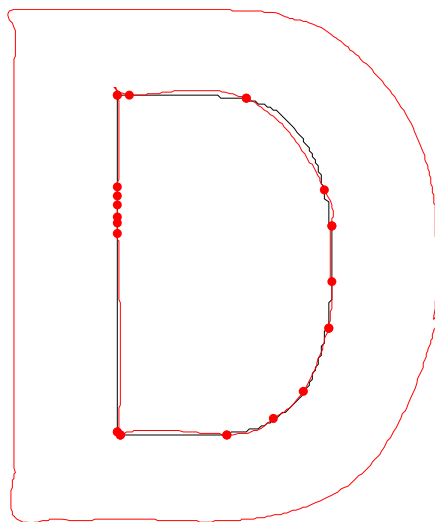
(b) At iteration = 10



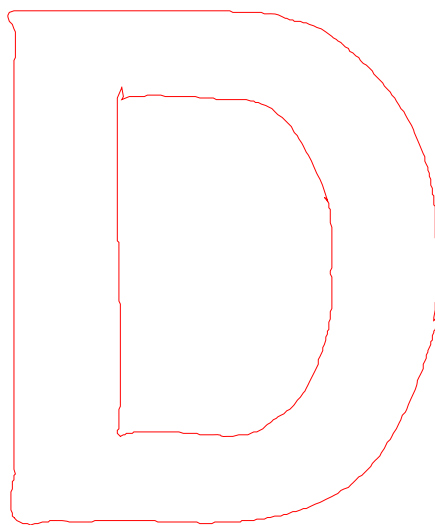
(c) At iteration = 20



(d) At iteration = 30



(e) At iteration = 40



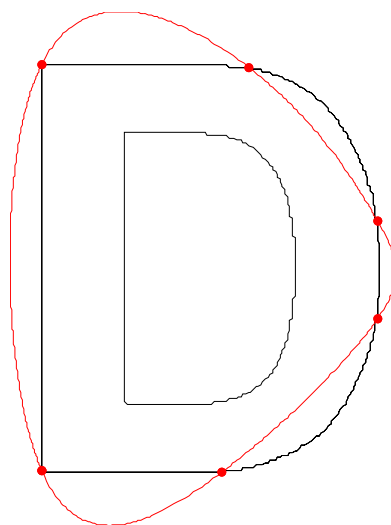
(e) Approximated spline for English alphabet “D”

Figure 4.71 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =3

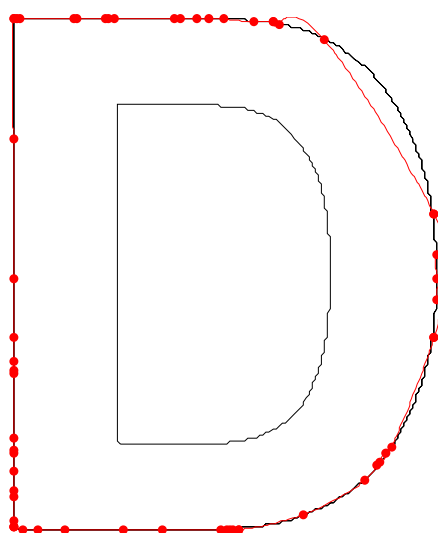
Table 4.20 Evaluation of algorithm 4.2 for English alphabet “D”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	137	60	0.92	1
849	15	68	35	0.67	2
849	15	50	57	0.73	3

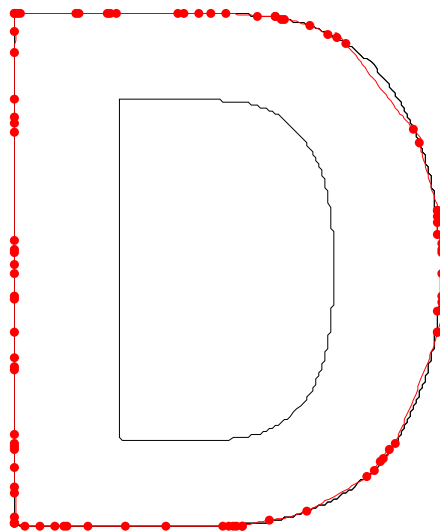
(Figure 4.72) to (Figure 4.74) shows the fitted curve over object contour at different iterations for algorithm 4.3 at threshold values of 1,2 and 3 respectively.



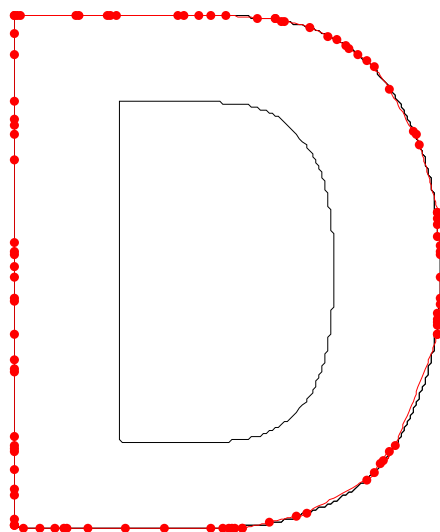
(a) At iteration = 1



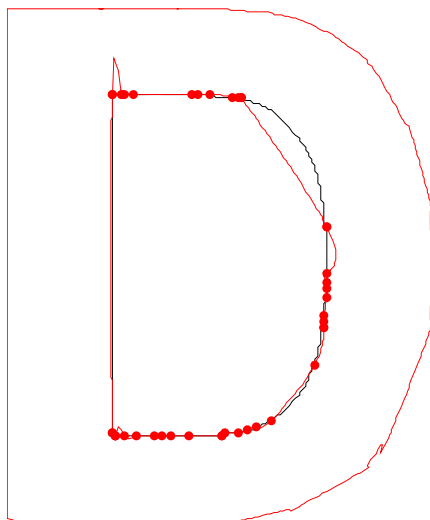
(b) At iteration = 10



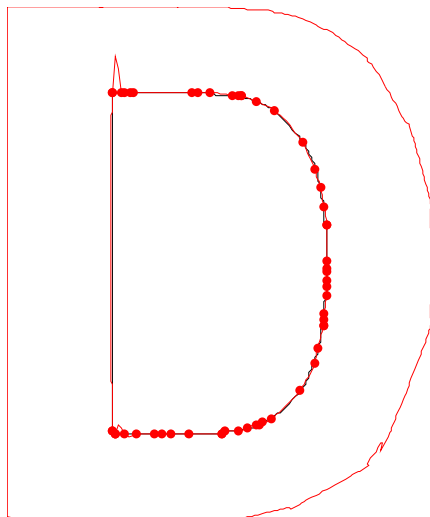
(c) At iteration = 20



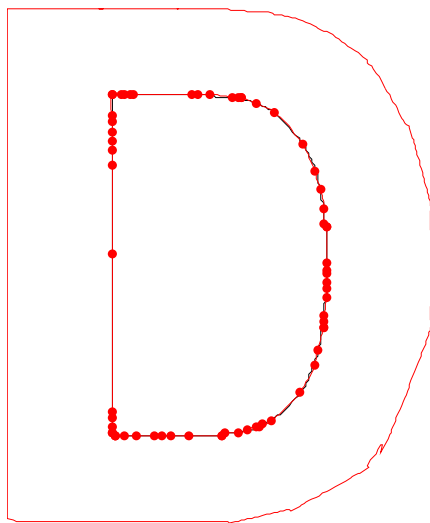
(d) At iteration = 30



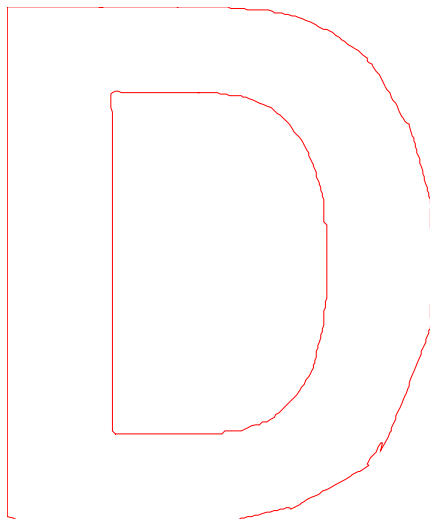
(e) At iteration = 40



(f) At iteration = 50

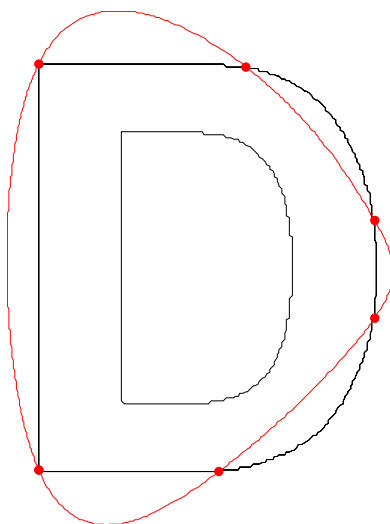


(g) At iteration = 60

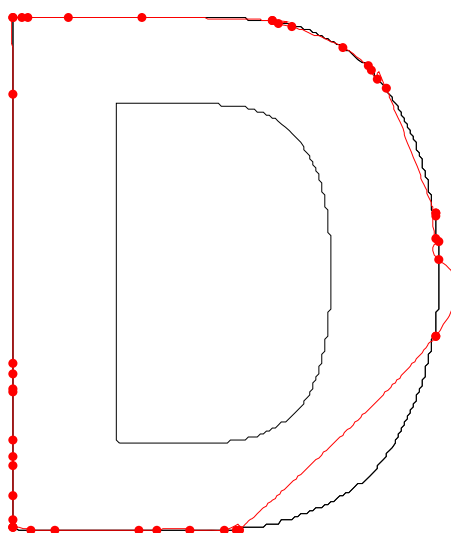


(h) Approximated spline for English alphabet “D”

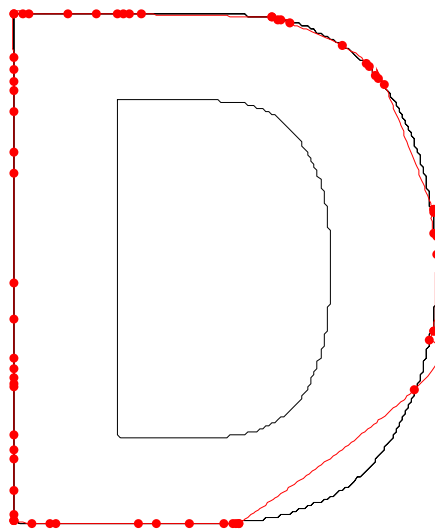
Figure 4.72 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =1



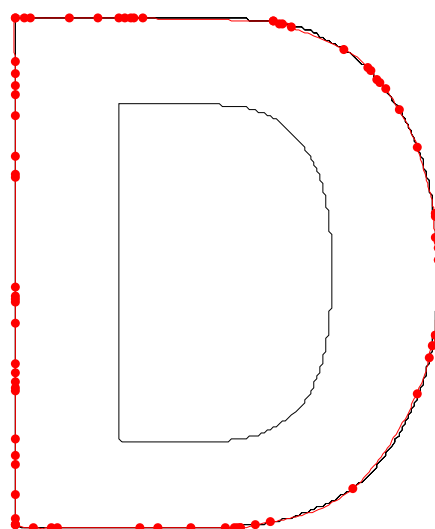
(a) At iteration = 1



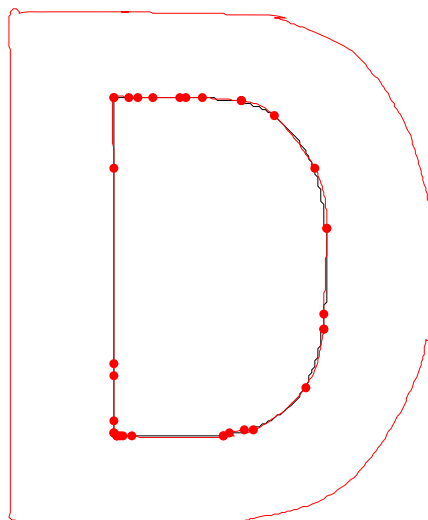
(b) At iteration = 10



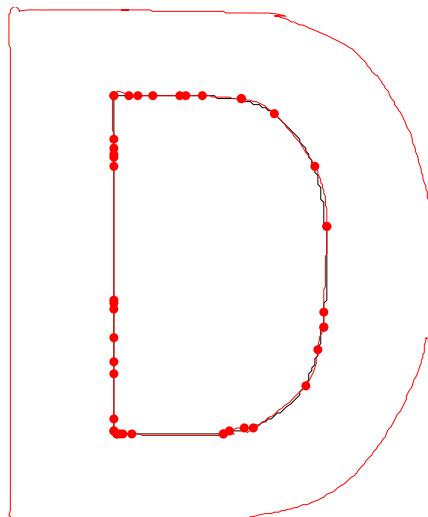
(c) At iteration = 20



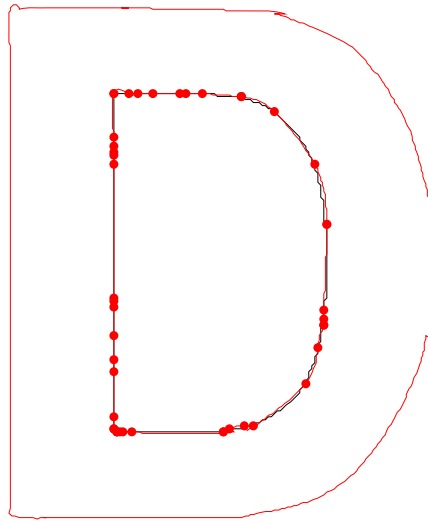
(d) At iteration = 30



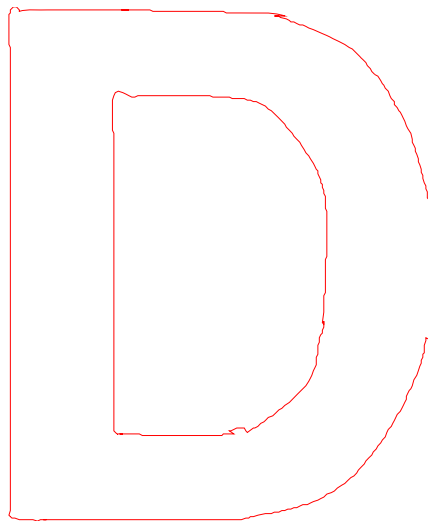
(e) At iteration = 40



(f) At iteration = 50

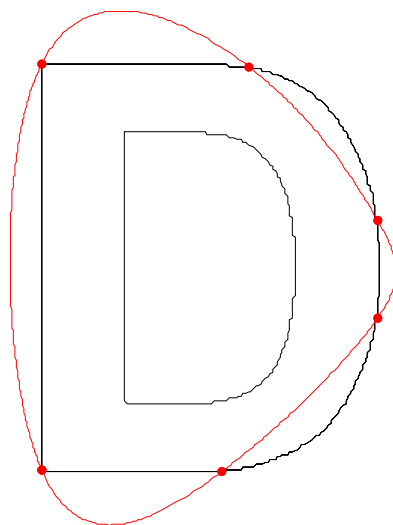


(g) At iteration = 60

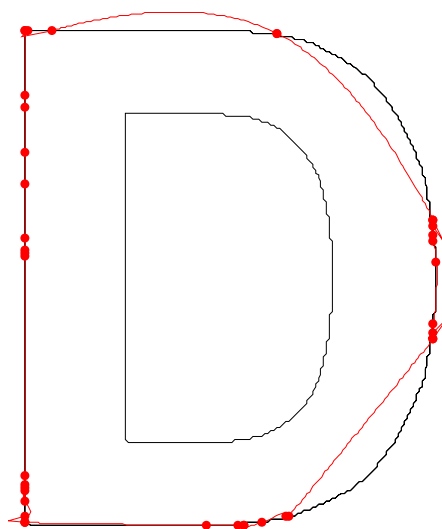


(h) Approximated spline for English alphabet “D”

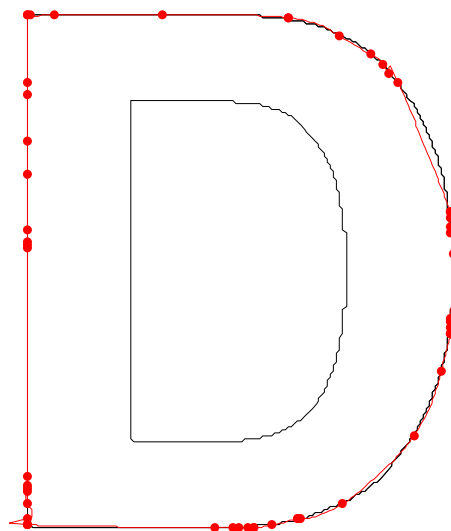
Figure 4.73 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =2



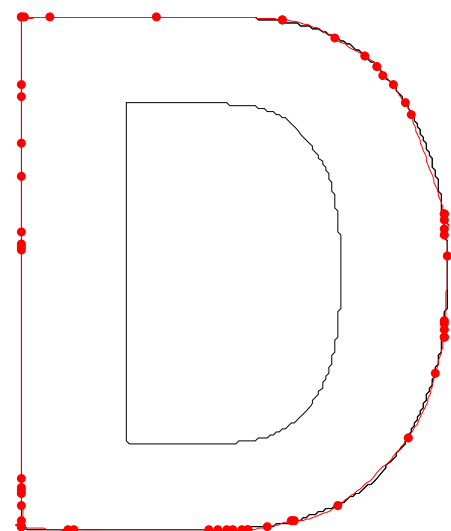
(a) At iteration = 1



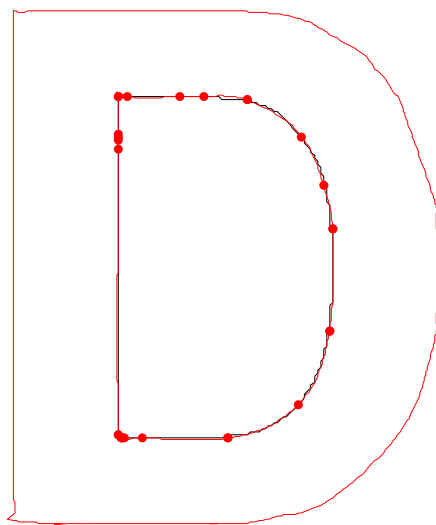
(b) At iteration = 10



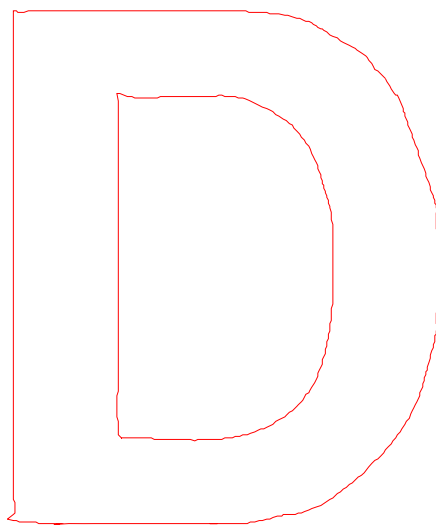
(c) At iteration = 20



(d) At iteration = 30



(e) At iteration = 40



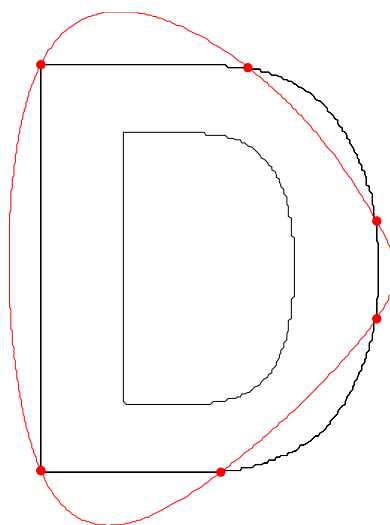
(f) Approximated spline for English alphabet “D”

Figure 4.74 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =3

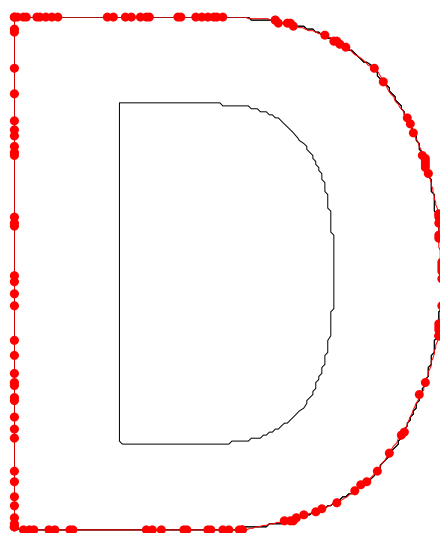
Table 4.21 Evaluation of algorithm 4.3 for English alphabet “D”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	141	60	0.93	1
849	15	94	60	0.87	2
849	15	60	43	0.78	3

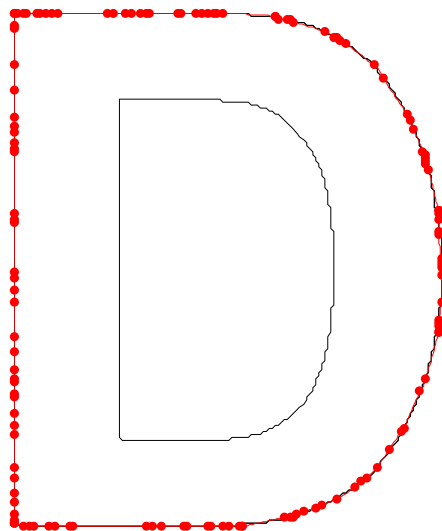
(Figure 4.75) to (Figure 4.77) shows the fitted curve over object contour at different iterations for algorithm 4.4 at threshold values of 1,2 and 3 respectively.



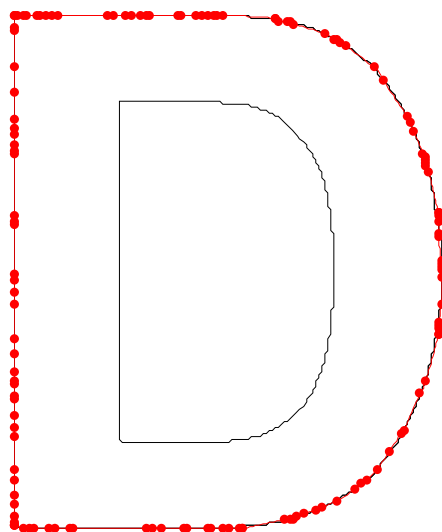
(a) At iteration = 1



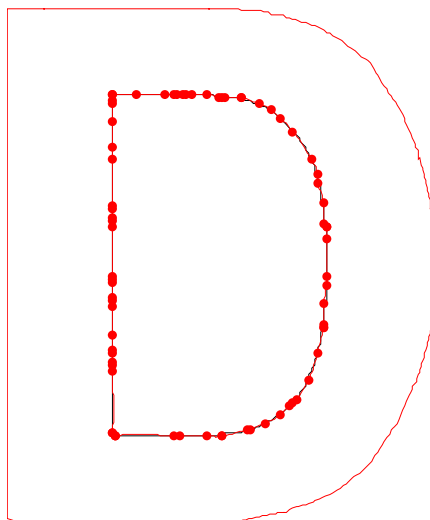
(b) At iteration = 10



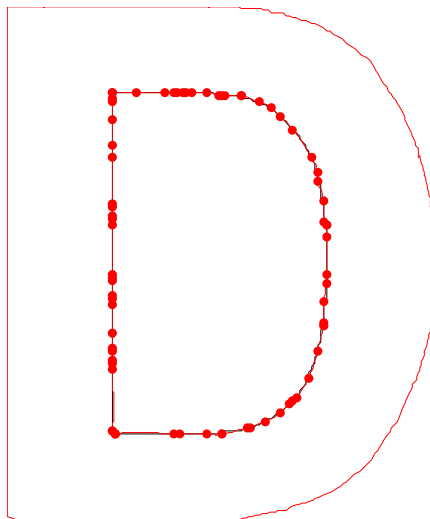
(c) At iteration = 20



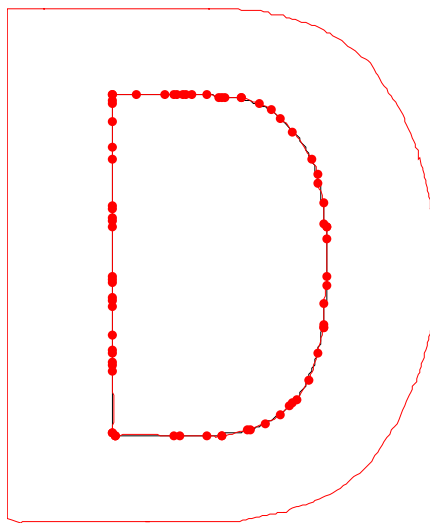
(d) At iteration = 30



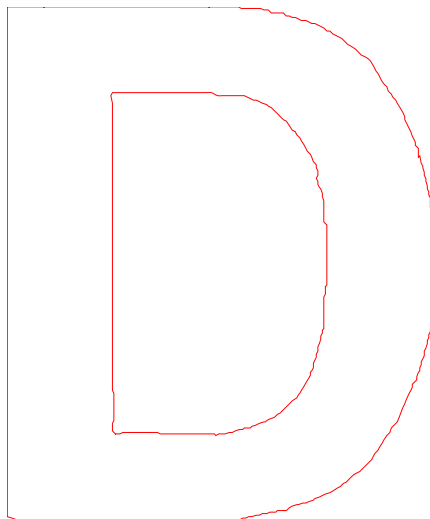
(e) At iteration = 40



(f) At iteration = 50

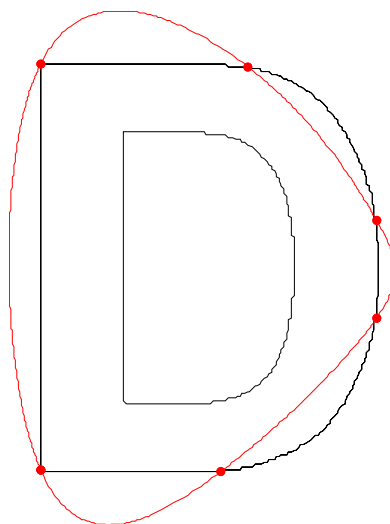


(g) At iteration = 60

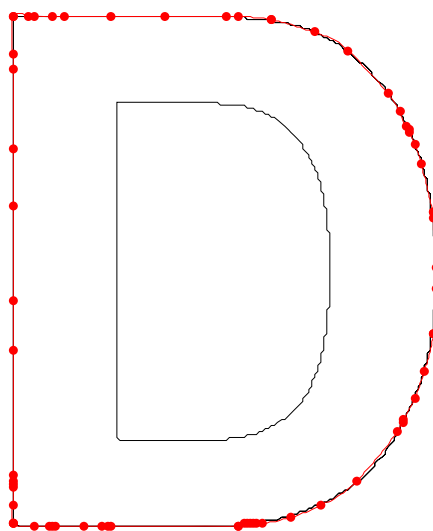


(e) Approximated spline for English alphabet “D”

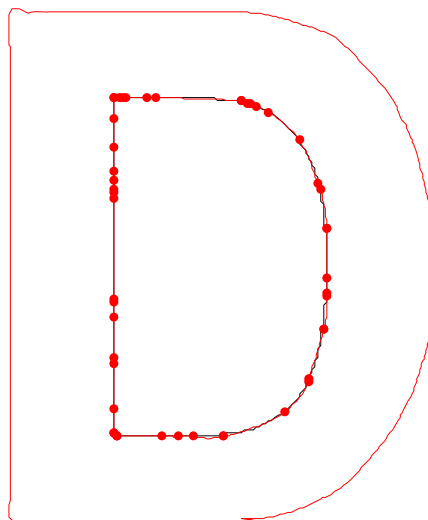
**Figure 4.75 Algorithm 4.4: Demonstration of spline fitting at each iteration using
threshold =1**



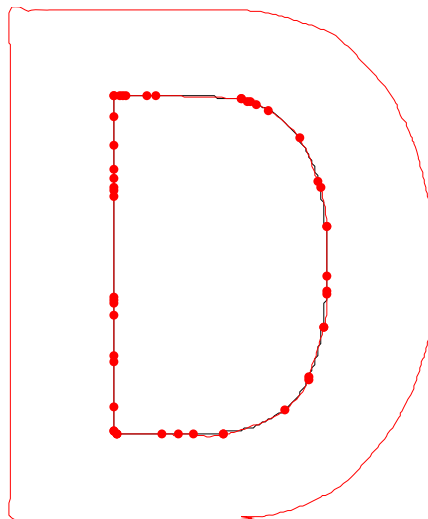
(a) At iteration = 1



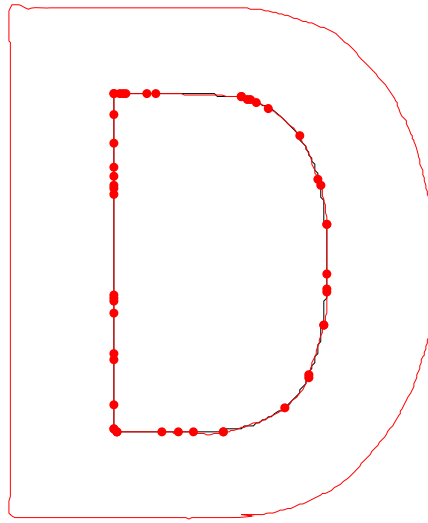
(b) At iteration = 10



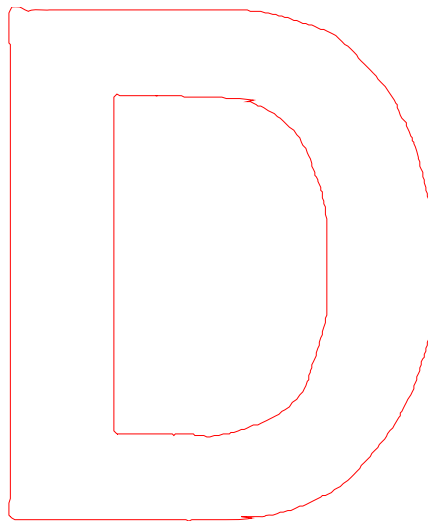
(c) At iteration = 20



(d) At iteration = 30

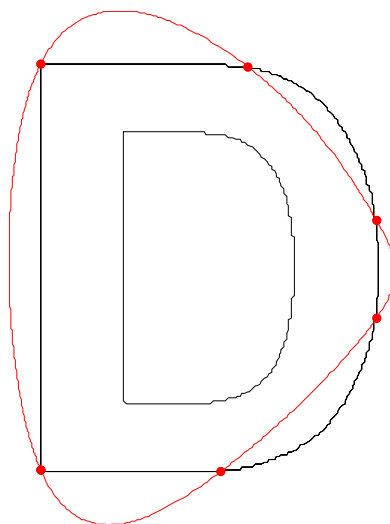


(e) At iteration = 40

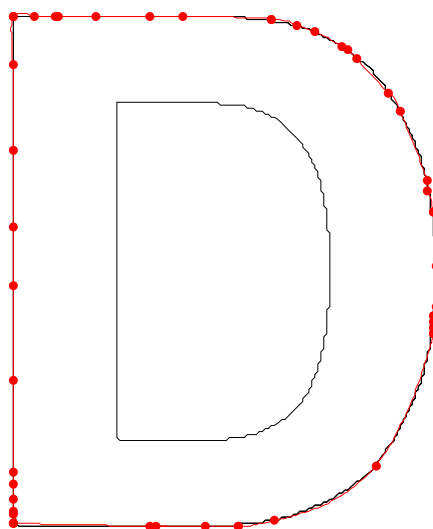


(f) Approximated spline for English alphabet “D”

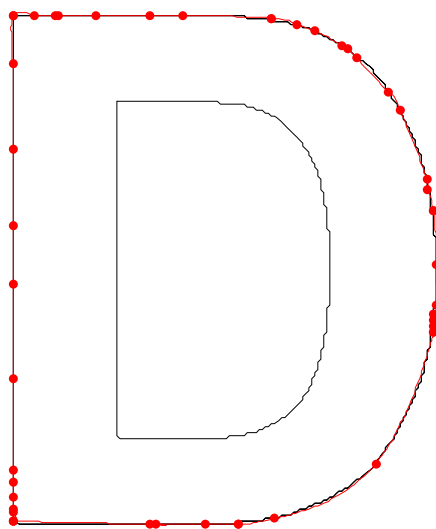
Figure 4.76 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =2



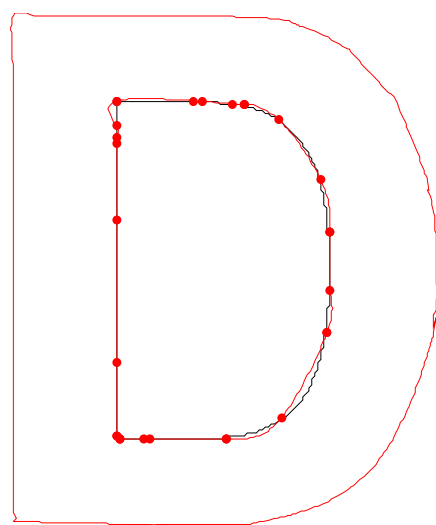
(a) At iteration = 1



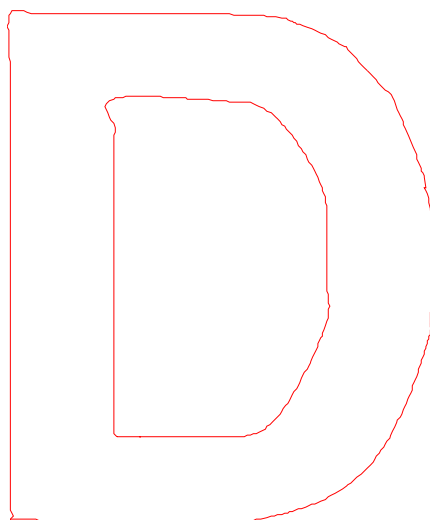
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 25



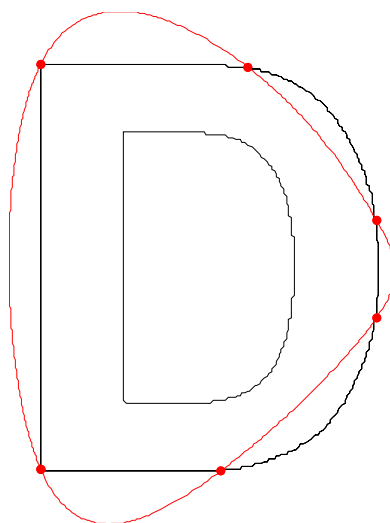
(e) Approximated spline for English alphabet “D”

Figure 4.77 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =3

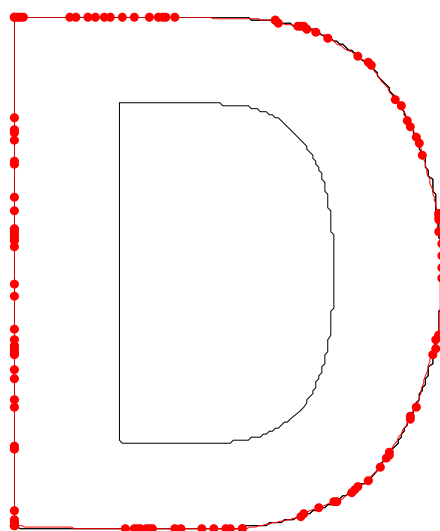
Table 4.22 Evaluation of algorithm 4.4 for English alphabet “D”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	612	60	14.17	1
849	15	216	43	1.2	2
849	15	125	26	1.53	3

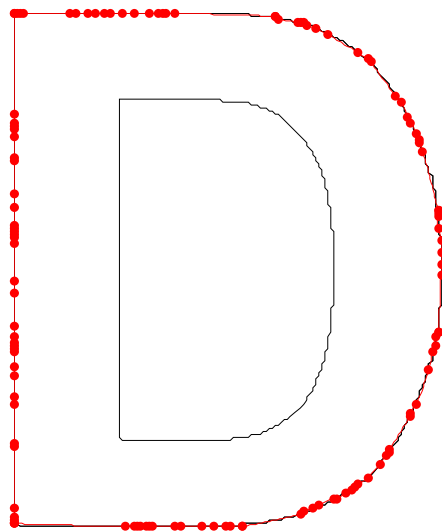
(Figure 4.78) to (Figure 4.80) shows the fitted curve over object contour at different iterations for algorithm 4.5 at threshold values of 1,2 and 3 respectively.



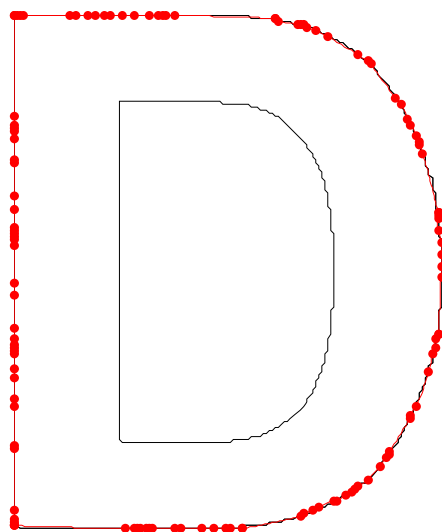
(a) At iteration = 1



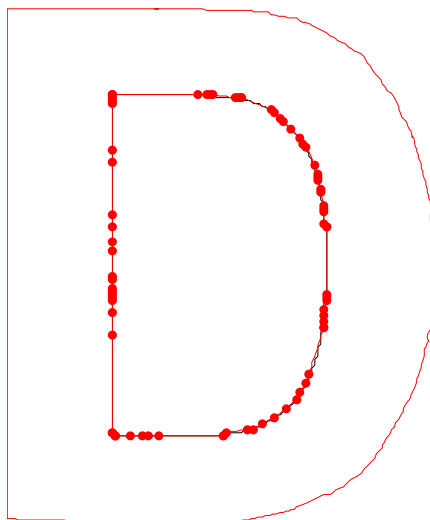
(b) At iteration = 10



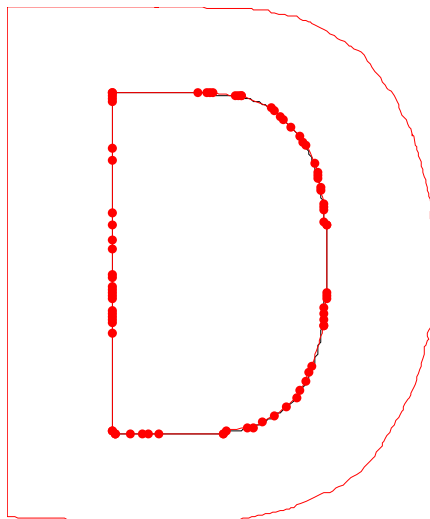
(c) At iteration = 20



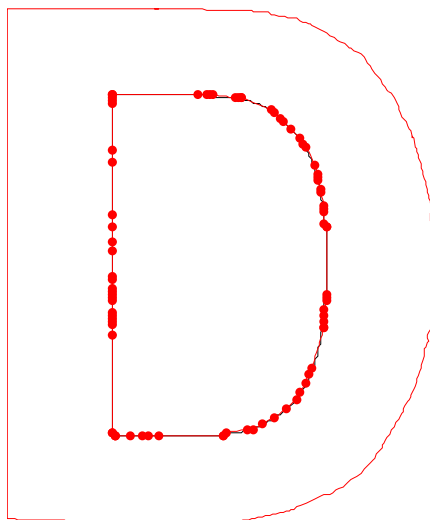
(d) At iteration = 30



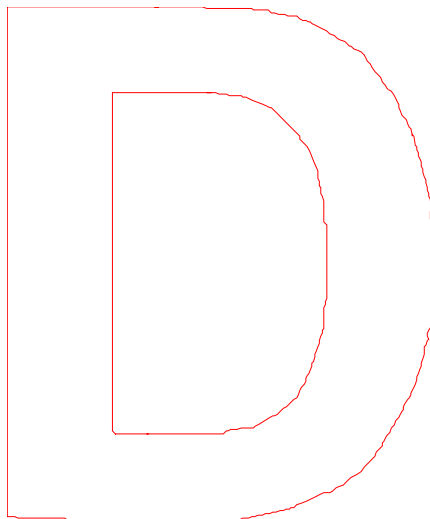
(e) At iteration = 40



(f) At iteration = 50

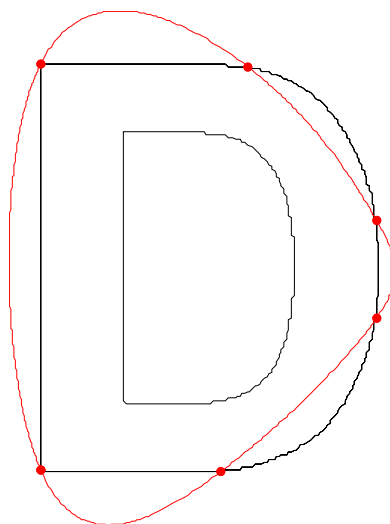


(g) At iteration = 60

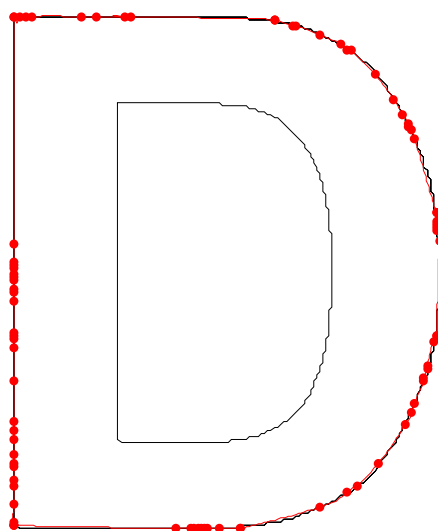


(h) Approximated spline for English alphabet “D”

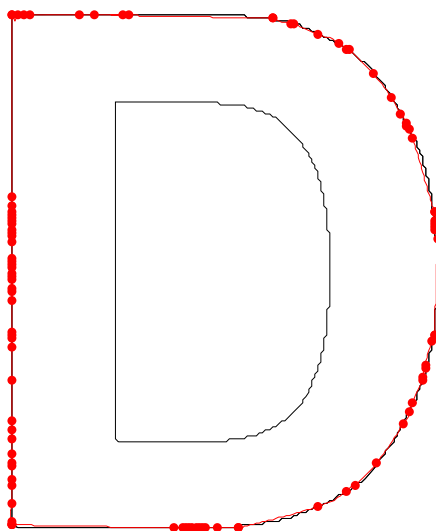
Figure 4.78 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =1



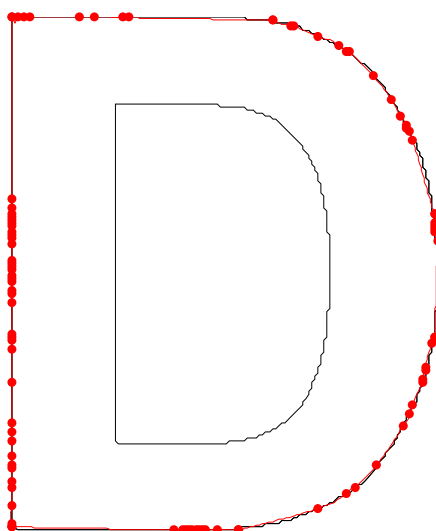
(a) At iteration = 1



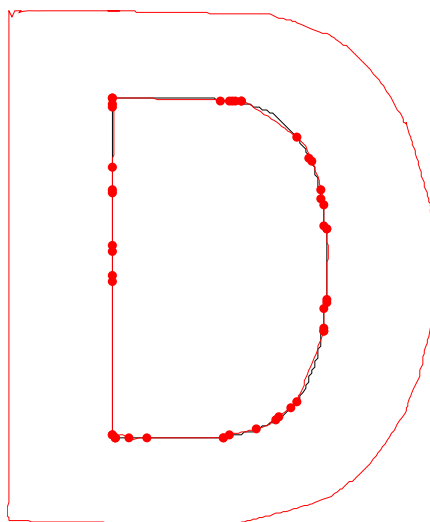
(b) At iteration = 10



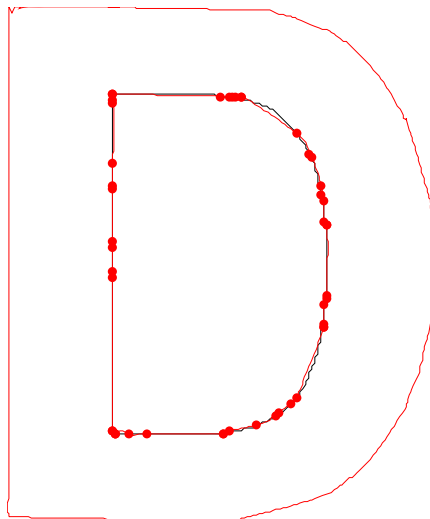
(c) At iteration = 20



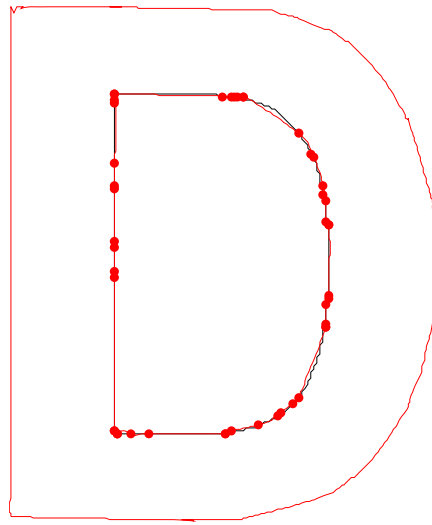
(d) At iteration = 30



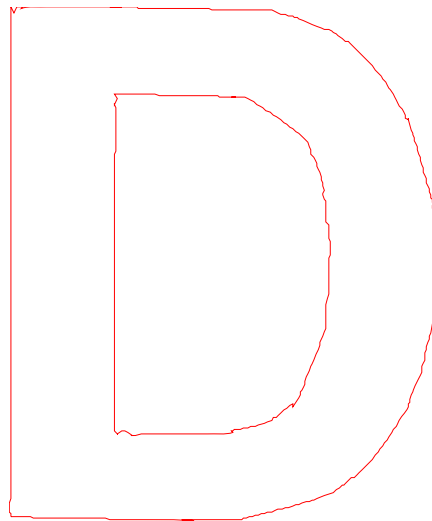
(e) At iteration = 40



(f) At iteration = 50

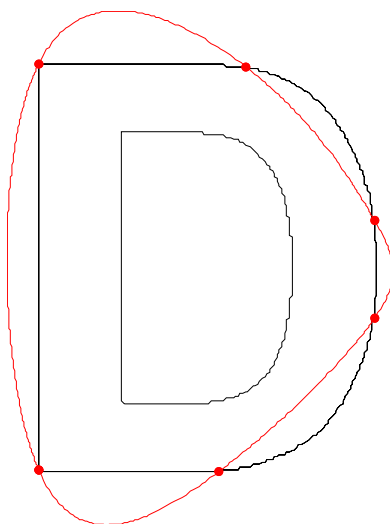


(g) At iteration = 60

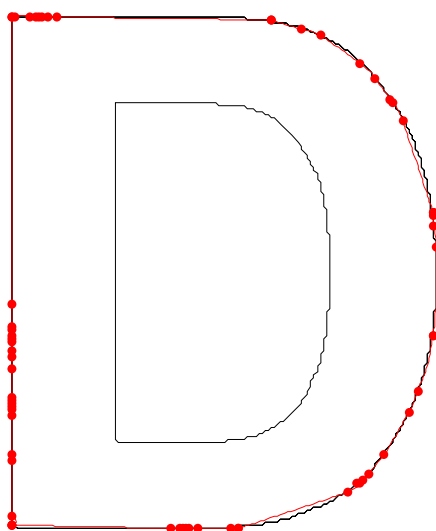


(h) Approximated spline for English alphabet “D”

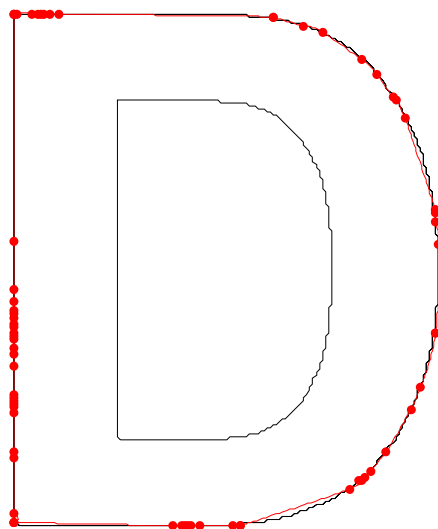
Figure 4.79 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =2



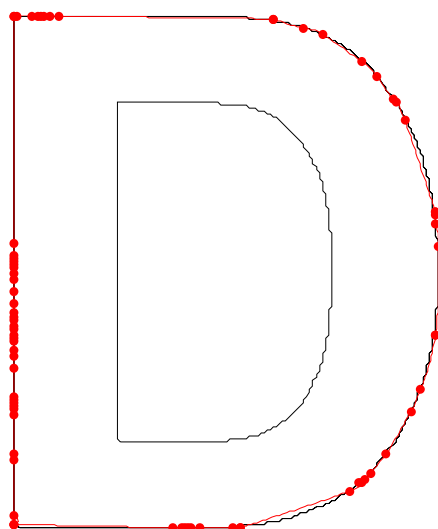
(a) At iteration = 1



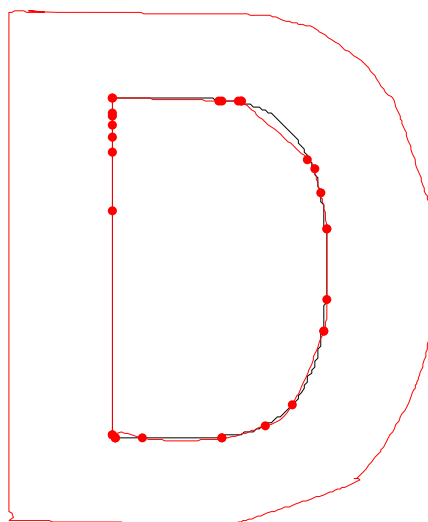
(b) At iteration = 10



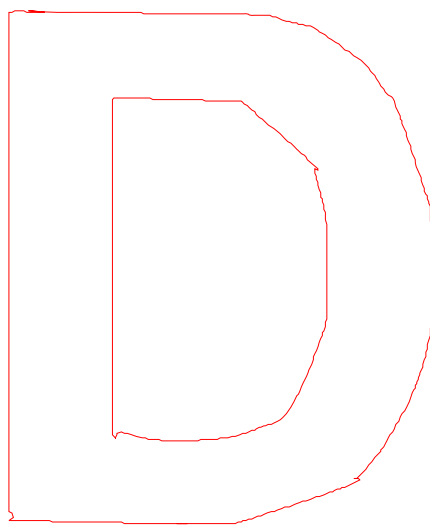
(c) At iteration = 20



(d) At iteration = 30



(e) At iteration = 35



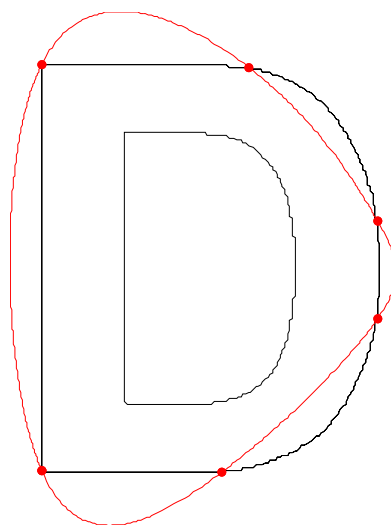
(f) Approximated spline for English alphabet “D”

Figure 4.80 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =3

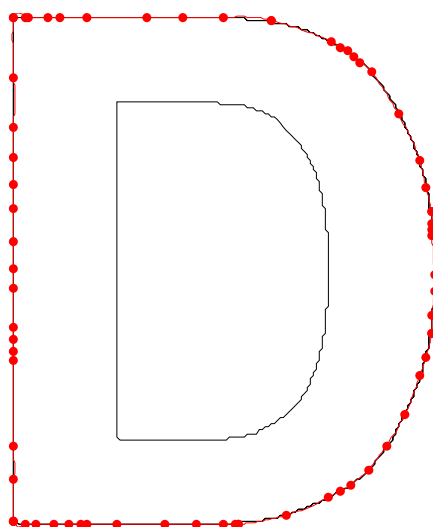
Table 4.23 Evaluation of algorithm 4.5 for English alphabet “D”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	618	60	9.43	1
849	15	324	60	2.18	2
849	15	220	36	1.422	3

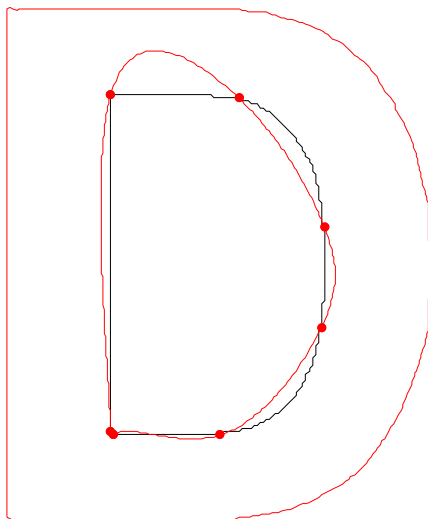
(Figure 4.81) to (Figure 4.83) shows the fitted curve over object contour at different iterations for algorithm 4.6 at threshold values of 1,2 and 3 respectively.



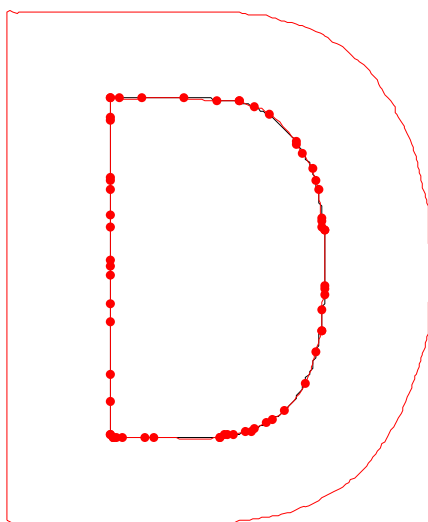
(a) At iteration = 1



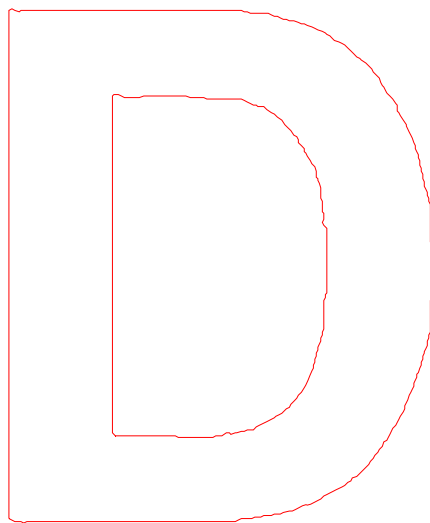
(b) At iteration = 5



(c) At iteration = 10

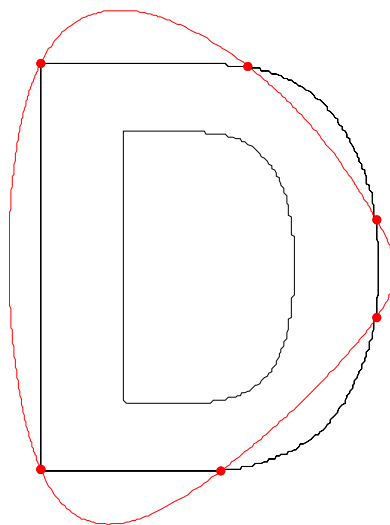


(d) At iteration = 15

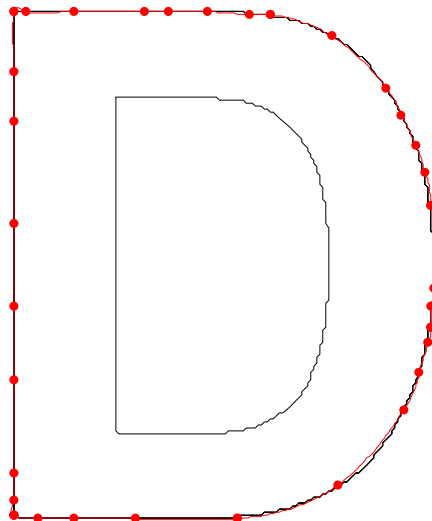


(e) Approximated spline for English alphabet “D”

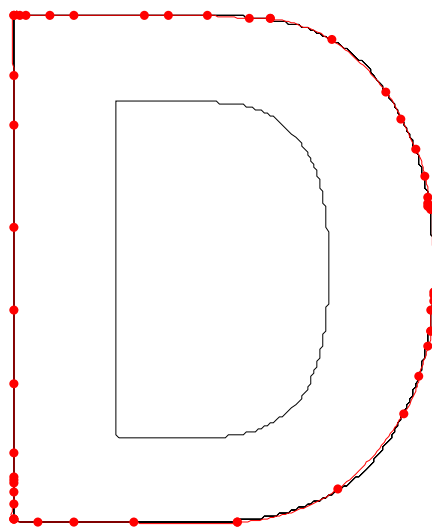
**Figure 4.81 Algorithm 4.6: Demonstration of spline fitting at each iteration using
threshold =1**



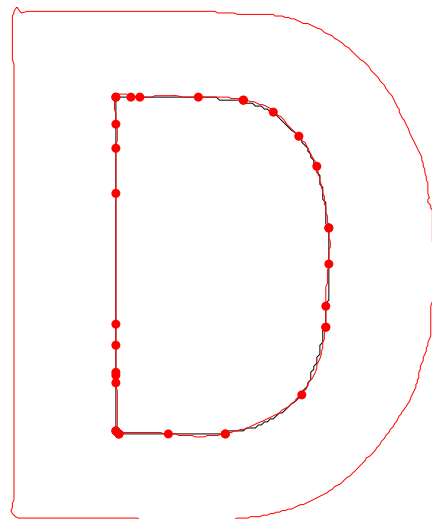
(a) At iteration = 1



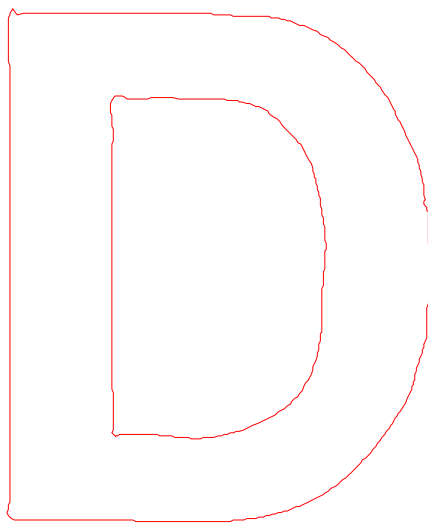
(b) At iteration = 5



(c) At iteration = 10

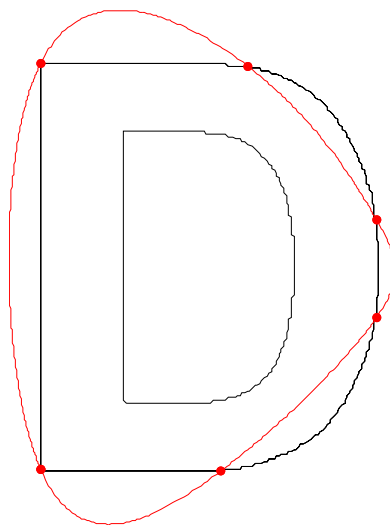


(d) At iteration = 15

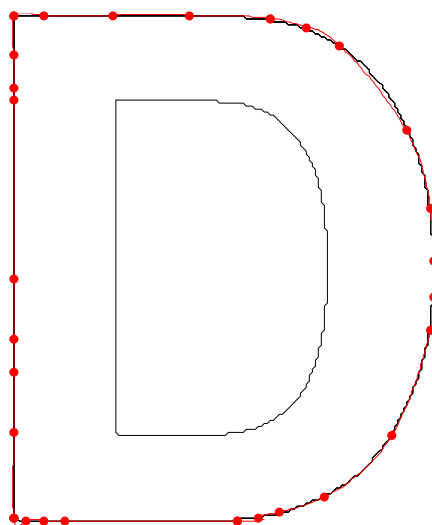


(e) Approximated spline for English alphabet “D”

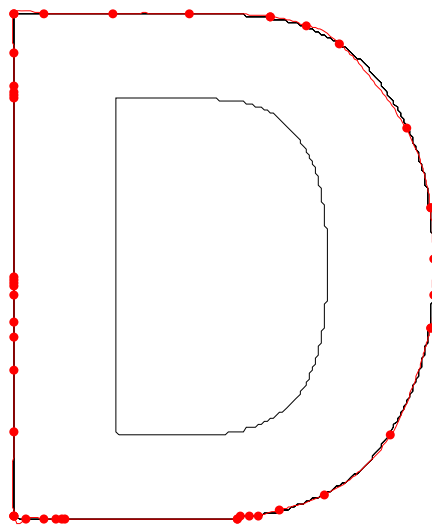
**Figure 4.82 Algorithm 4.6: Demonstration of spline fitting at each iteration using
threshold =2**



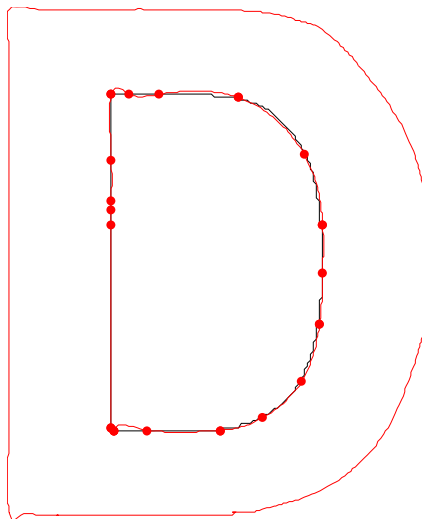
(a) At iteration = 1



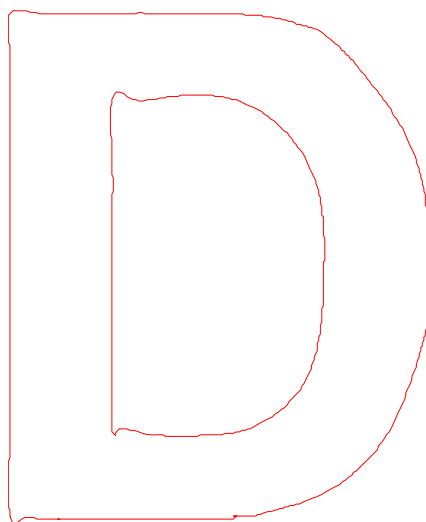
(b) At iteration = 5



(c) At iteration = 10



(d) At iteration = 15



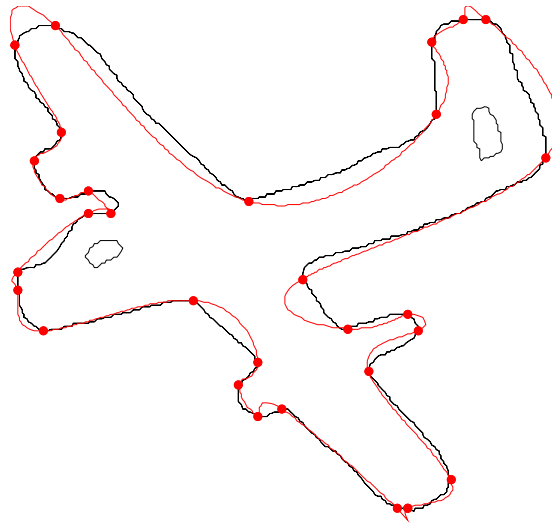
(e) Approximated spline for English alphabet “D”

Figure 4.83 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =3

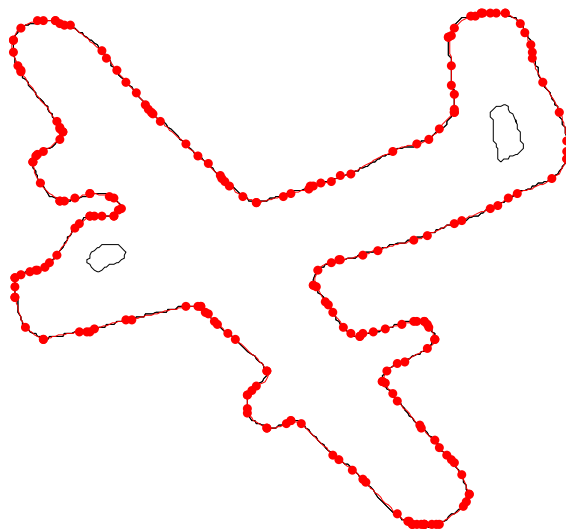
Table 4.24 Evaluation of algorithm 4.6 for English alphabet “D”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	160	18	0.92	1
849	15	77	19	0.67	2
849	15	61	17	0.67	3

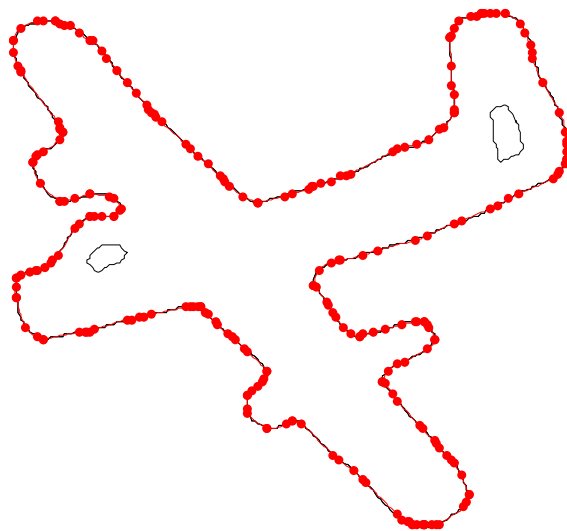
(Figure 4.84) to (Figure 4.86) shows the fitted curve over object contour at different iterations for algorithm 4.1 at threshold values of 1,2 and 3 respectively.



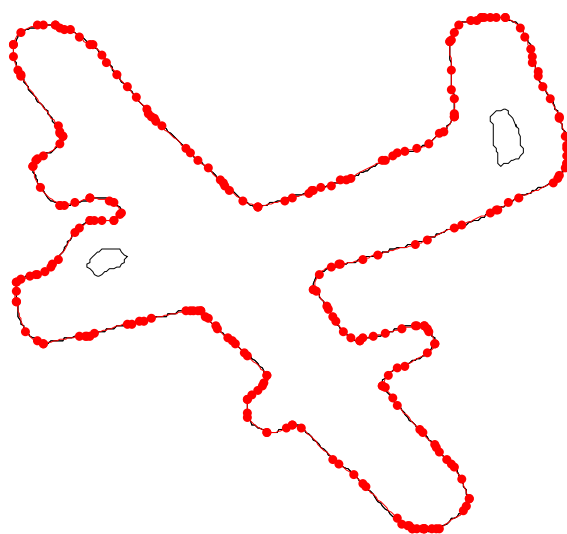
(a) At iteration = 1



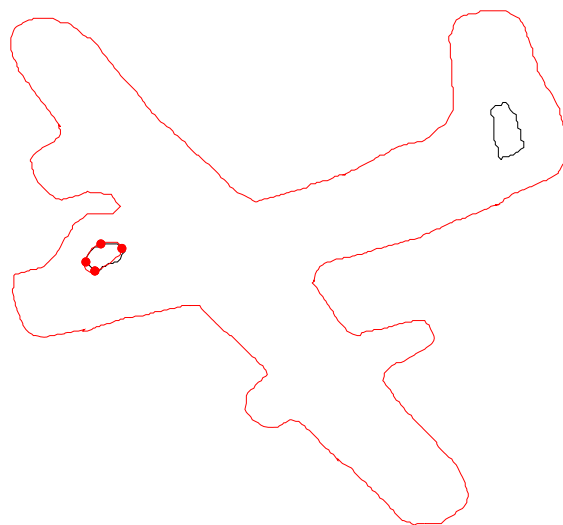
(b) At iteration = 10



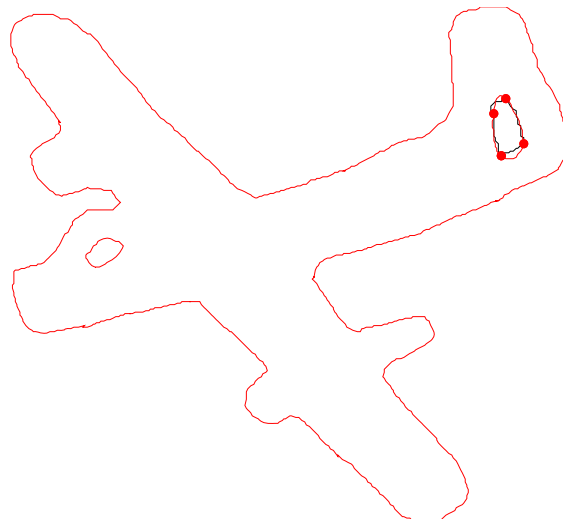
(c) At iteration = 20



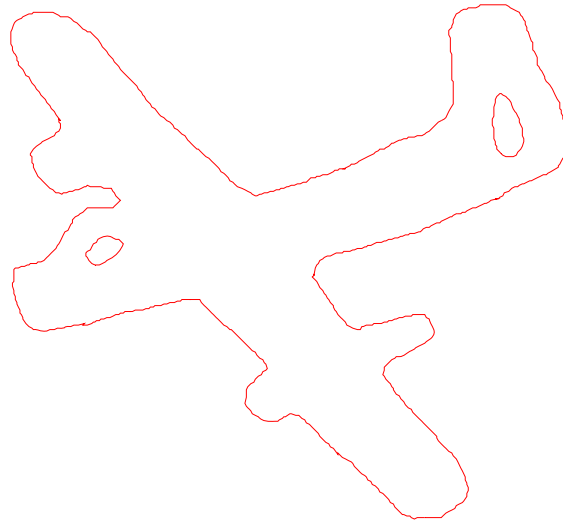
(d) At iteration = 30



(e) At iteration = 31

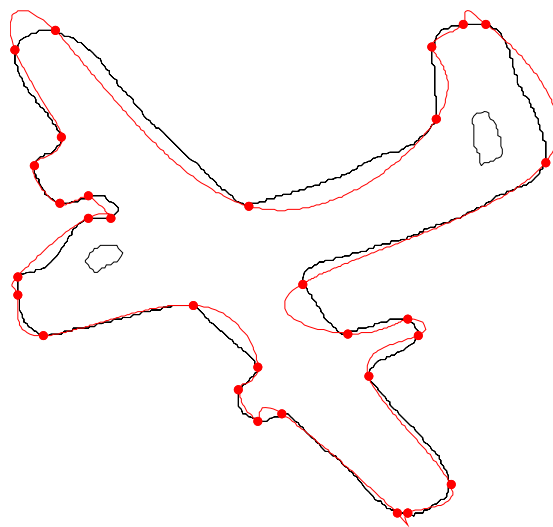


(f) At iteration = 33

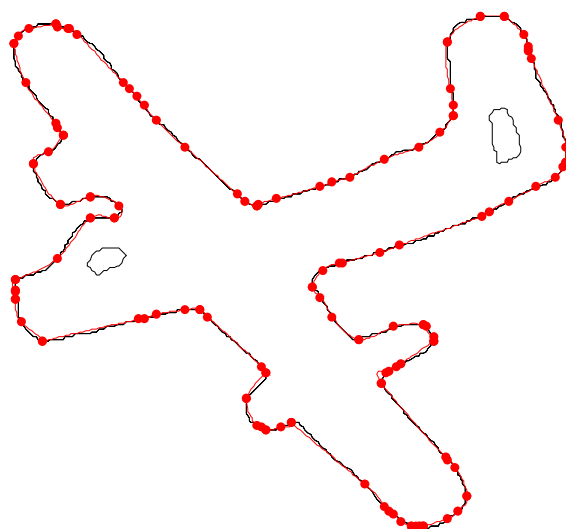


(g) Approximated spline for Object “Mult_Seg_Plane”

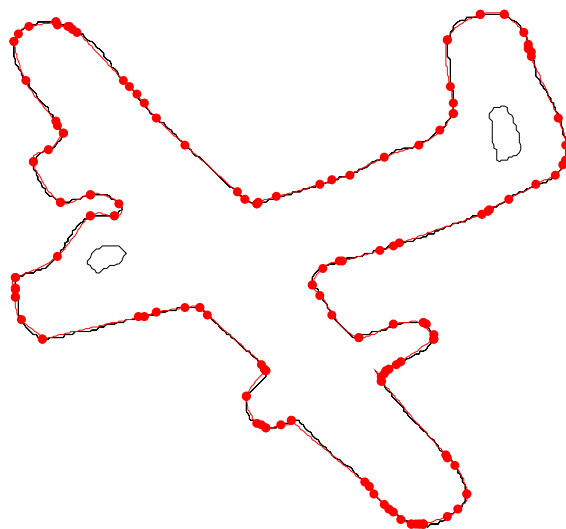
**Figure 4.84 Algorithm 4.1: Demonstration of spline fitting at each iteration using
threshold =1**



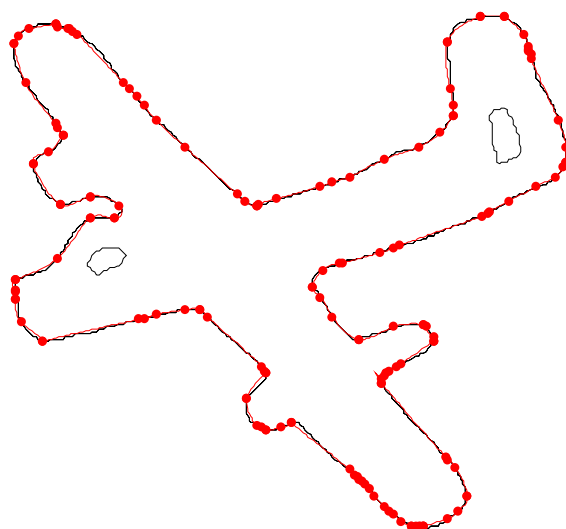
(a) At iteration = 1



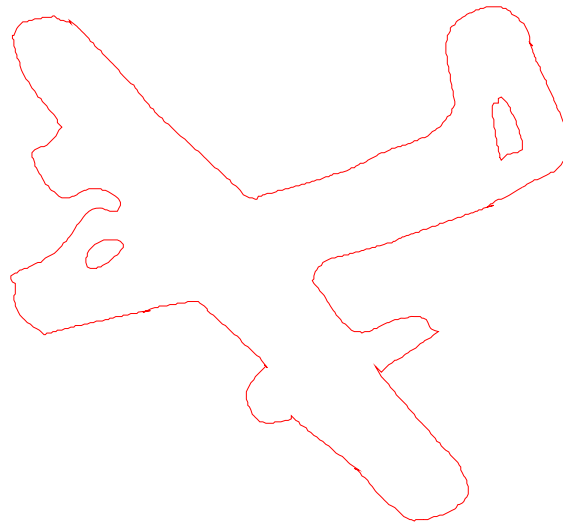
(b) At iteration = 10



(c) At iteration = 20

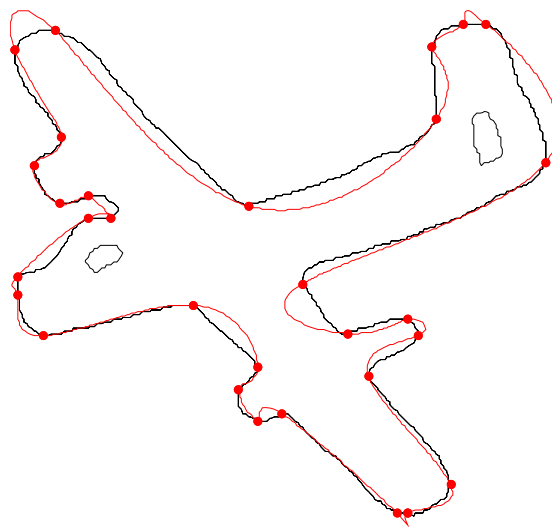


(d) At iteration = 30

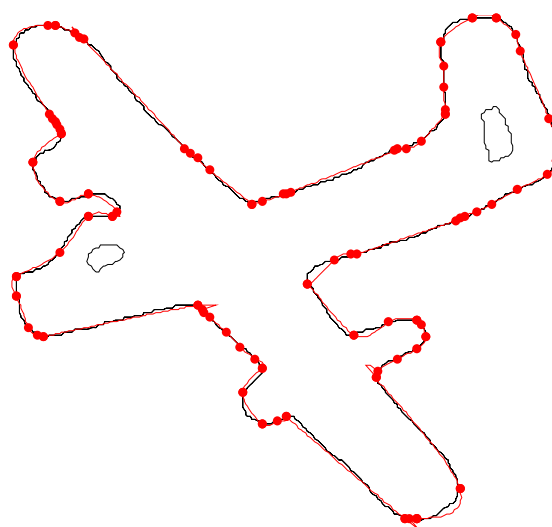


(e) Approximated spline for Object “Mult_Seg_Plane”

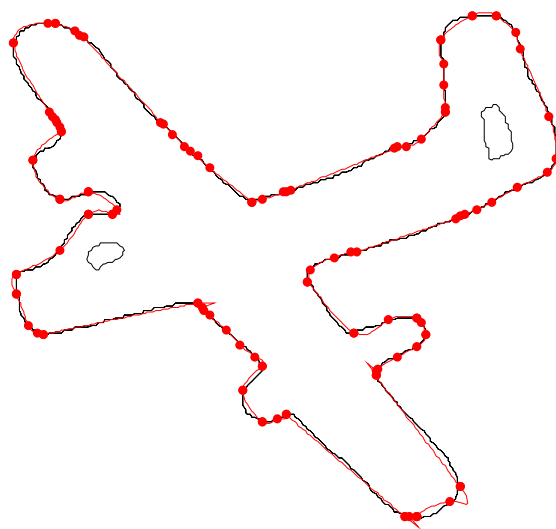
**Figure 4.85 Algorithm 4.1: Demonstration of spline fitting at each iteration using
threshold =2**



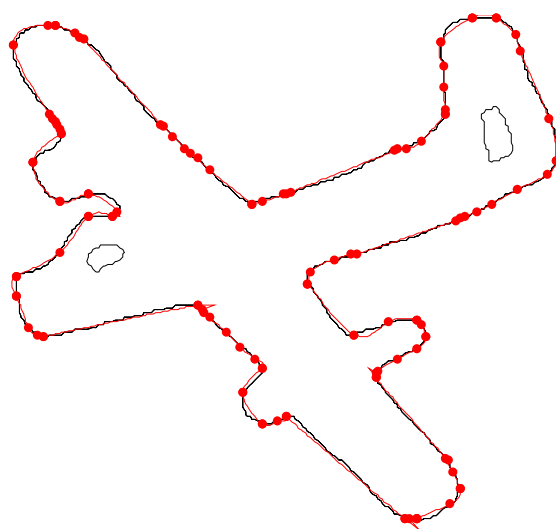
(a) At iteration = 1



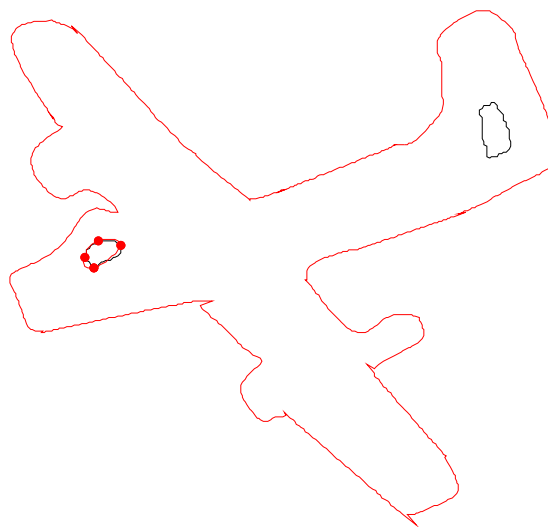
(b) At iteration = 10



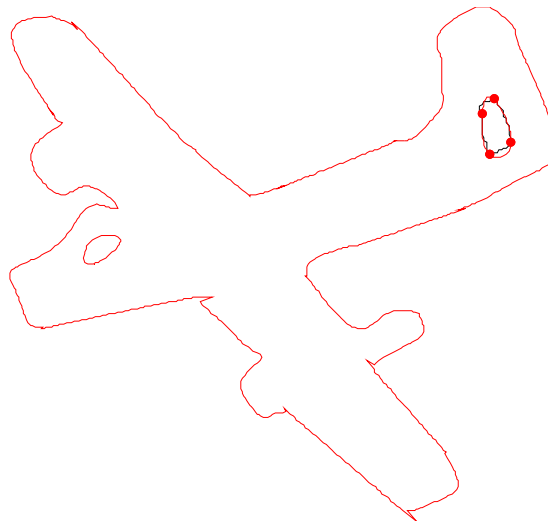
(c) At iteration = 20



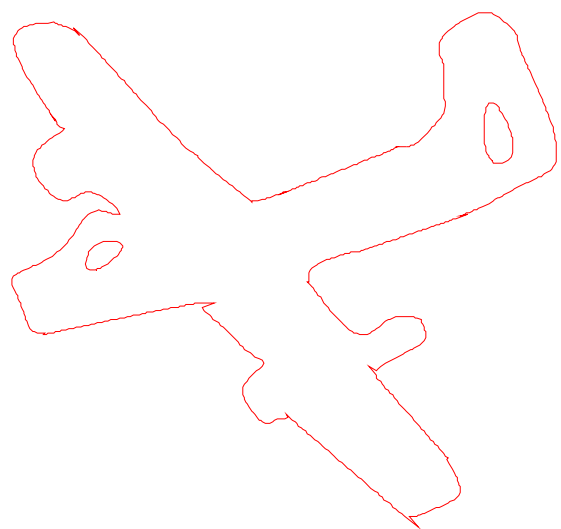
(d) At iteration = 30



(e) At iteration = 31



(f) At iteration = 32



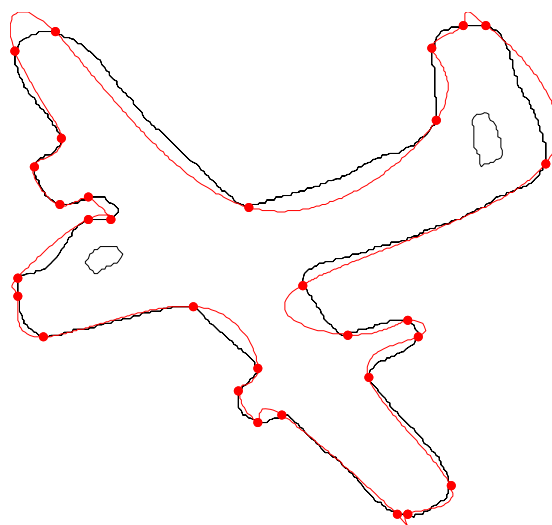
(g) Approximated spline for Object “Mult_Seg_Plane”

Figure 4.86 Algorithm 4.1: Demonstration of spline fitting at each iteration using threshold =3

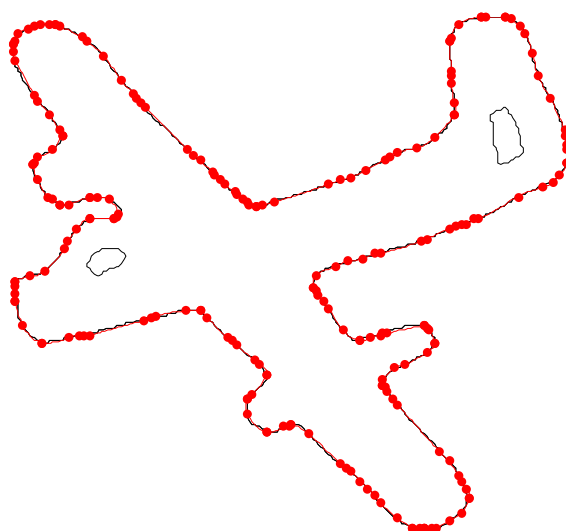
Table 4.25 Evaluation of algorithm 4.1 for Object “Mult_Seg_Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	415	33	10.96	1
1005	41	224	34	2.45	2
1005	41	182	32	2.18	3

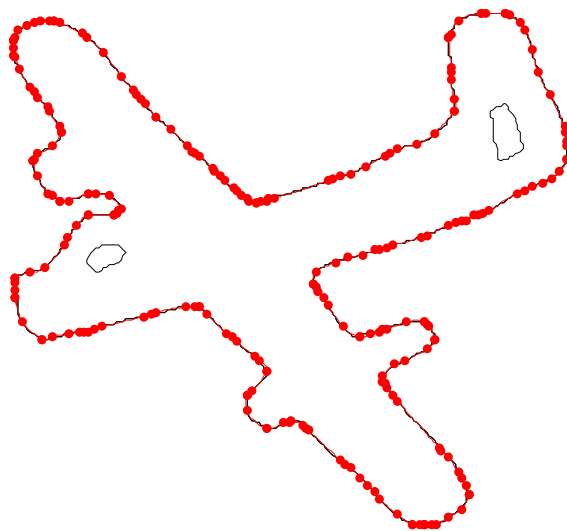
(Figure 4.87) to (Figure 4.89) shows the fitted curve over object contour at different iterations for algorithm 4.2 at threshold values of 1,2 and 3 respectively.



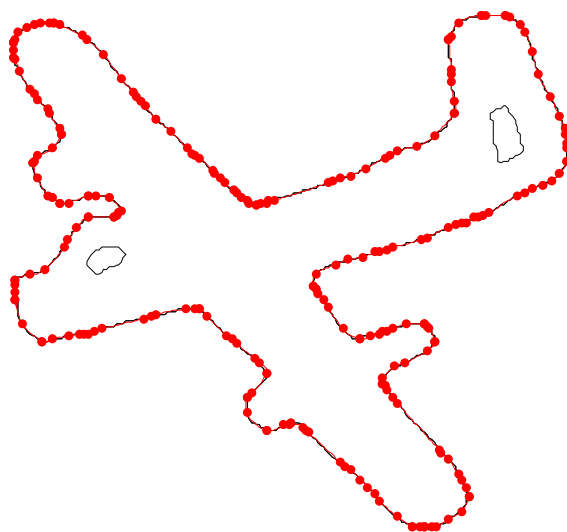
(a) At iteration = 1



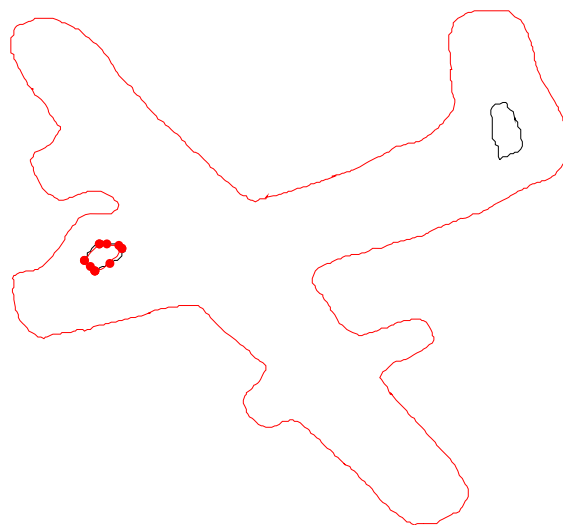
(b) At iteration = 10



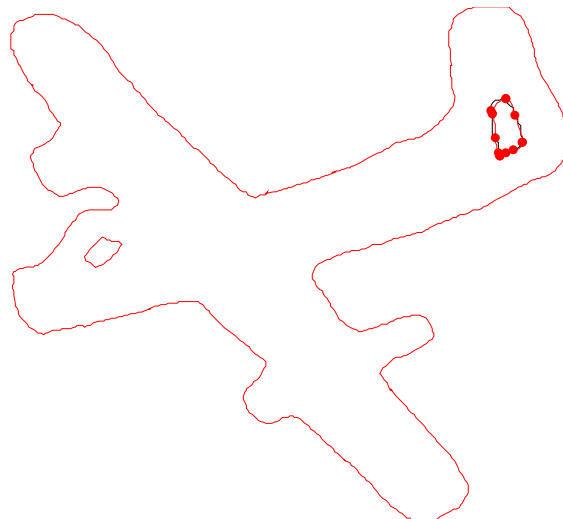
(c) At iteration = 20



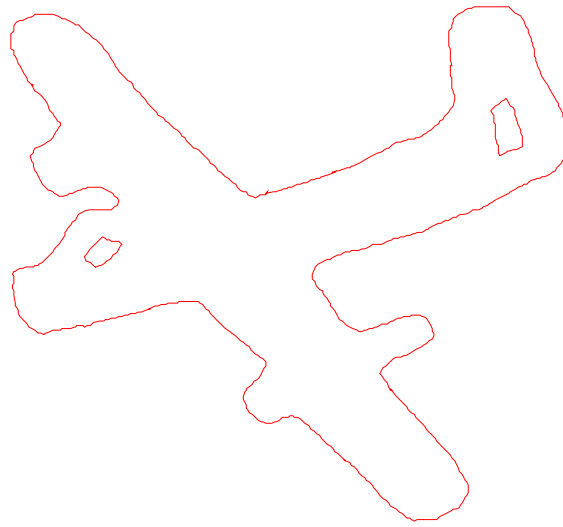
(d) At iteration = 30



(e) At iteration = 35

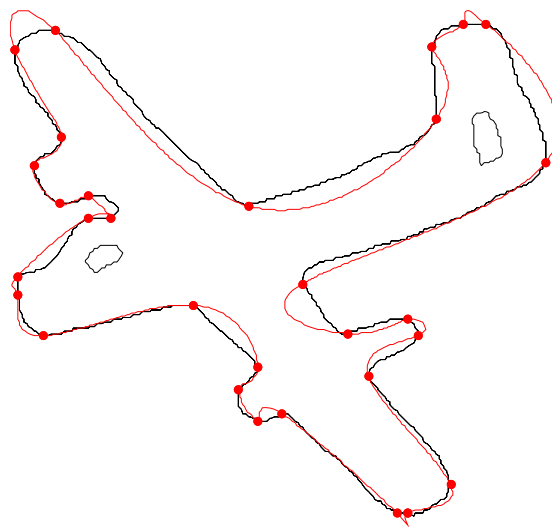


(f) At iteration = 40

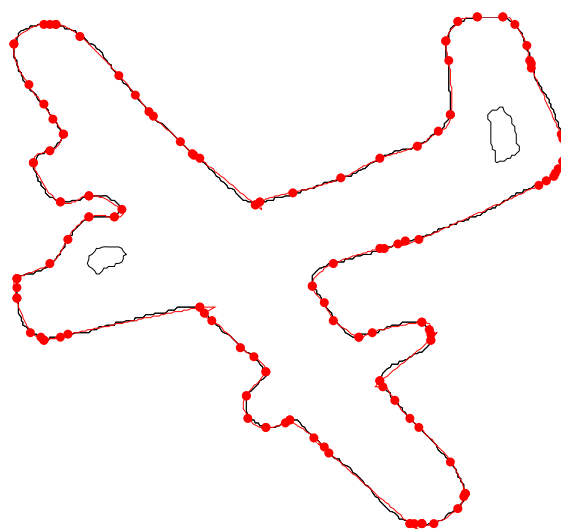


(g) Approximated spline for Object “Mult_Seg_Plane”

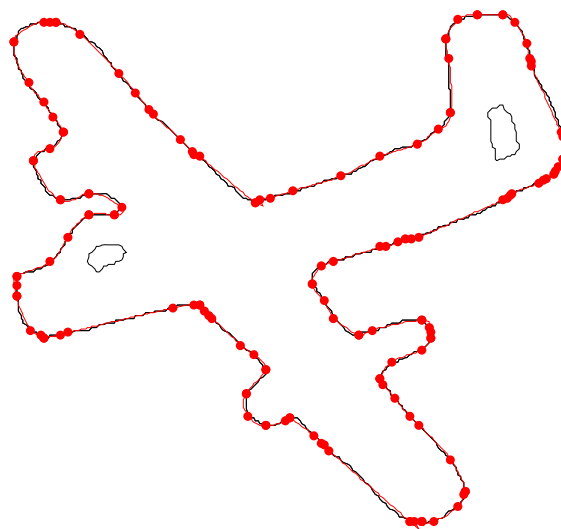
**Figure 4.87 Algorithm 4.2: Demonstration of spline fitting at each iteration using
threshold =1**



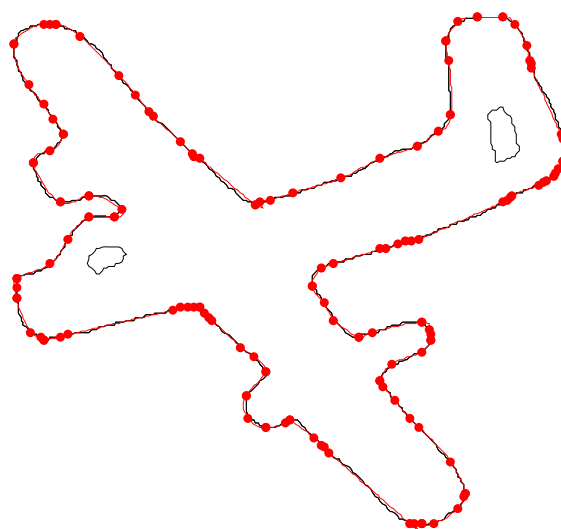
(a) At iteration = 1



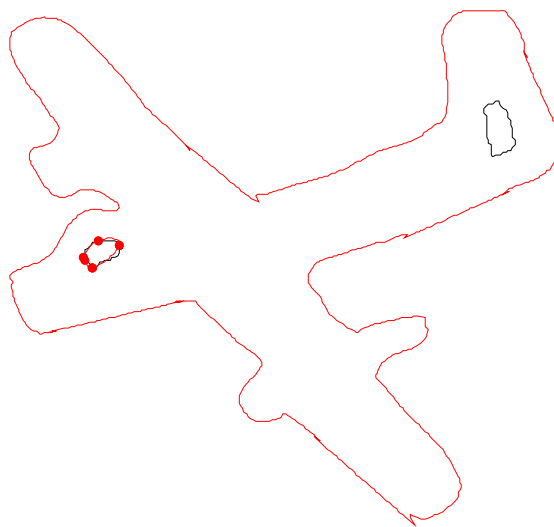
(b) At iteration = 10



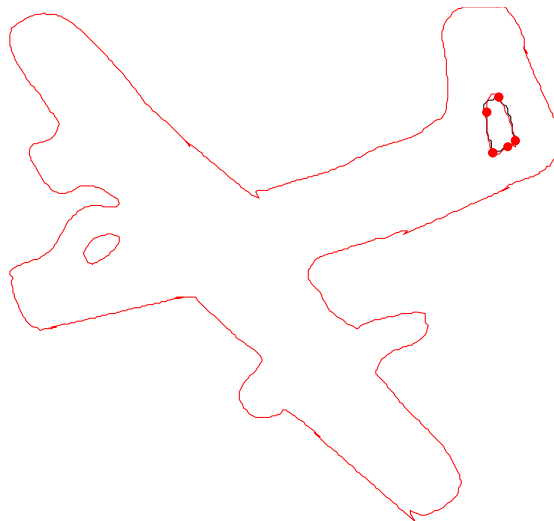
(c) At iteration = 20



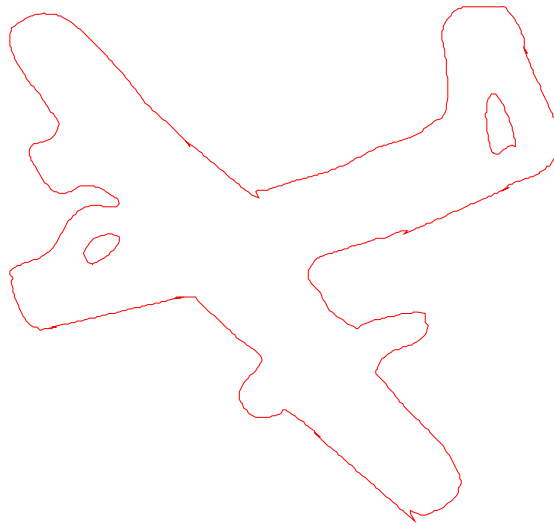
(d) At iteration = 30



(e) At iteration = 32

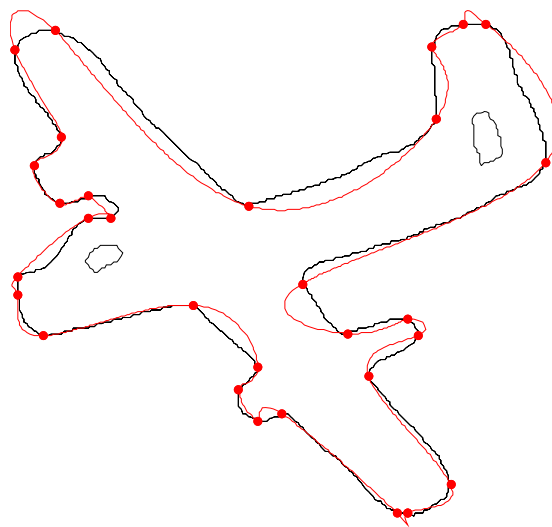


(f) At iteration = 34

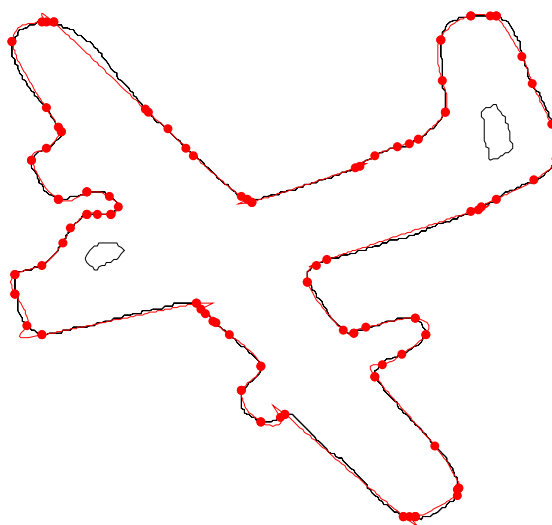


(g) Approximated spline for Object “Mult_Seg_Plane”

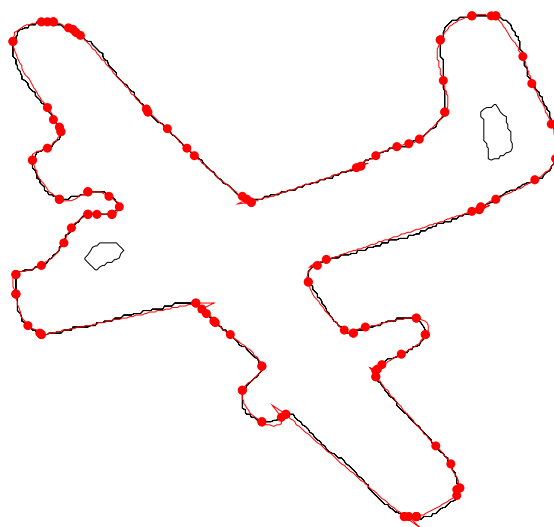
**Figure 4.88 Algorithm 4.2: Demonstration of spline fitting at each iteration using
threshold =2**



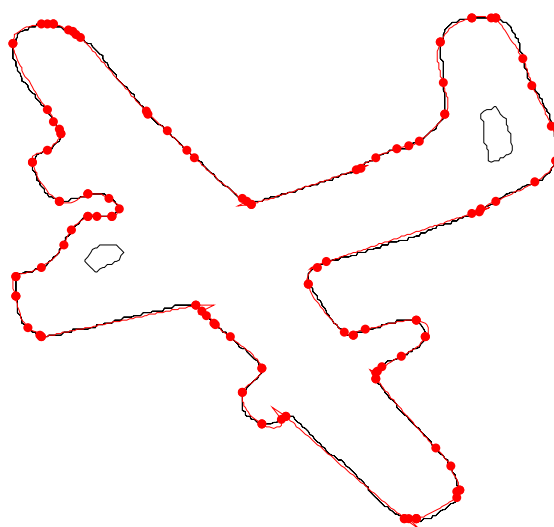
(a) At iteration = 1



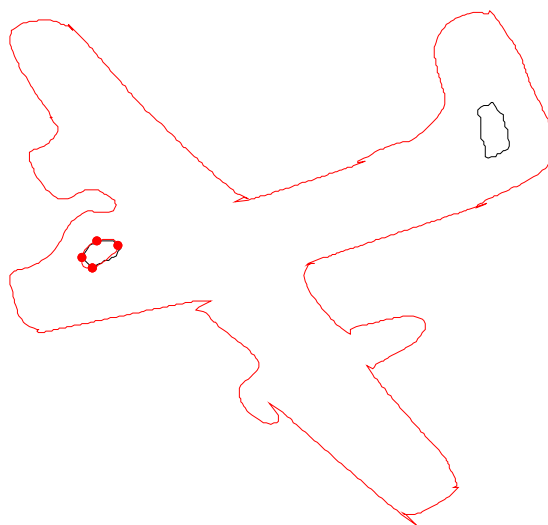
(b) At iteration = 10



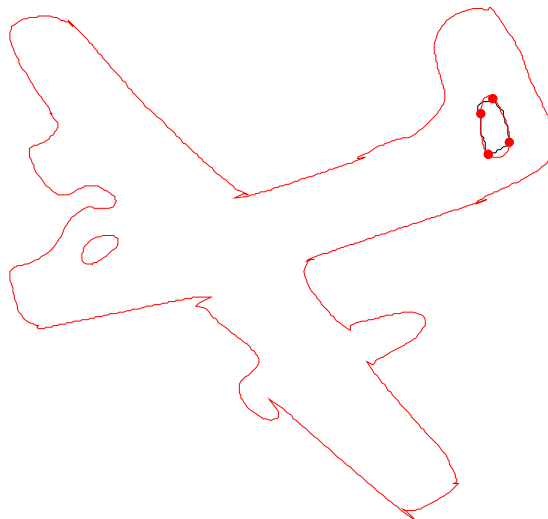
(c) At iteration = 20



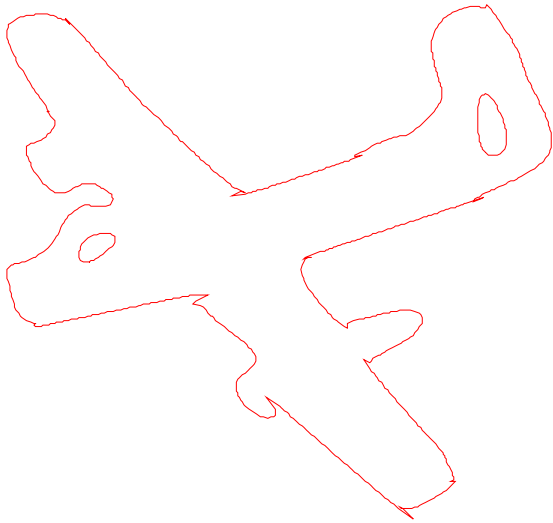
(d) At iteration = 30



(e) At iteration = 31



(f) At iteration = 32



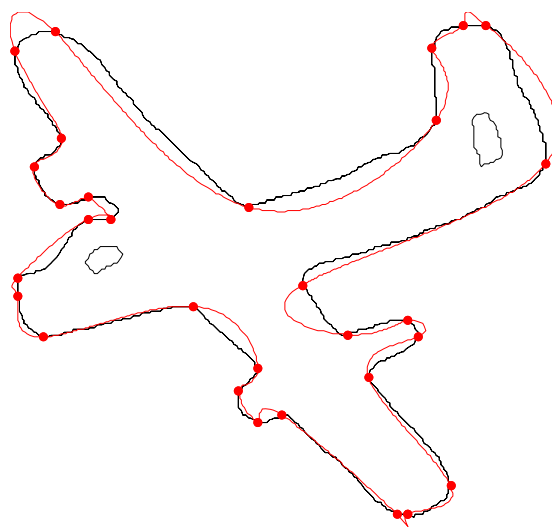
(g) Approximated spline for Object “Mult_Seg_Plane”

Figure 4.89 Algorithm 4.2: Demonstration of spline fitting at each iteration using threshold =3

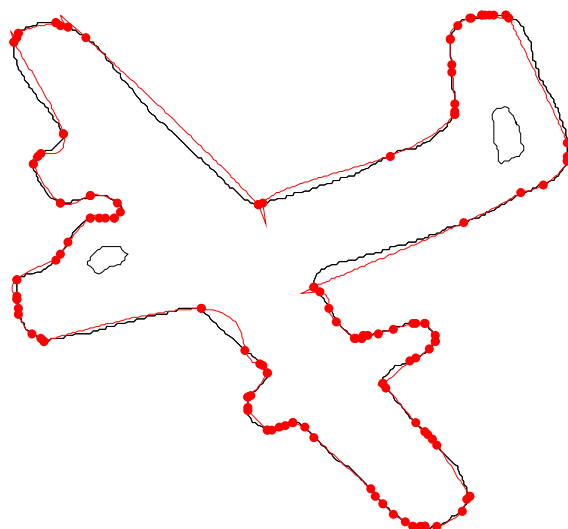
Table 4.26 Evaluation of algorithm 4.2 for Object “Mult_Seg_Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	195	44	2.1	1
1005	41	90	34	1.21	2
1005	41	60	32	1.59	3

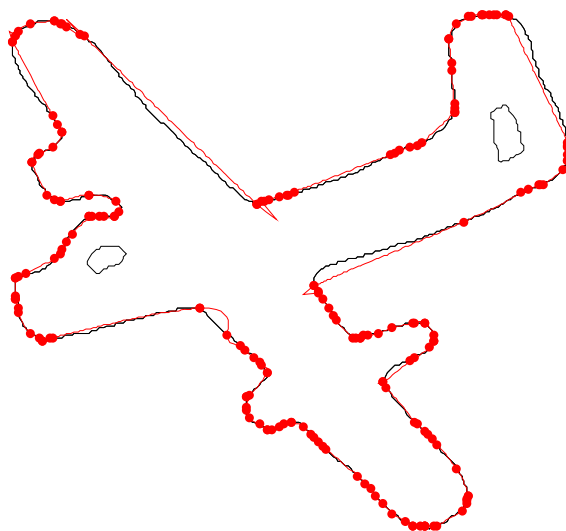
(Figure 4.90) to (Figure 4.93) shows the fitted curve over object contour at different iterations for algorithm 4.3 at threshold values of 1,2 and 3 respectively.



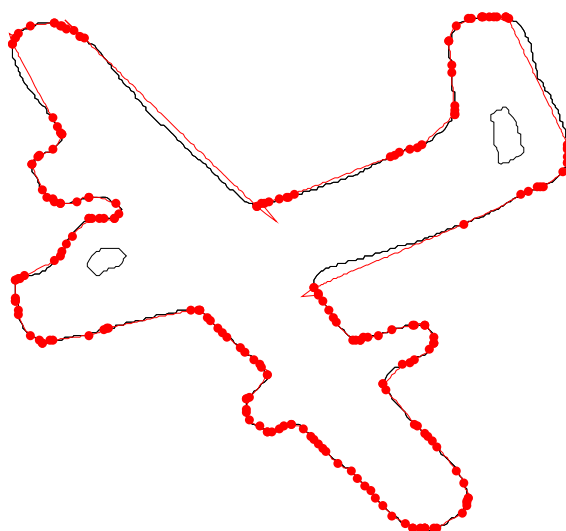
(a) At iteration = 1



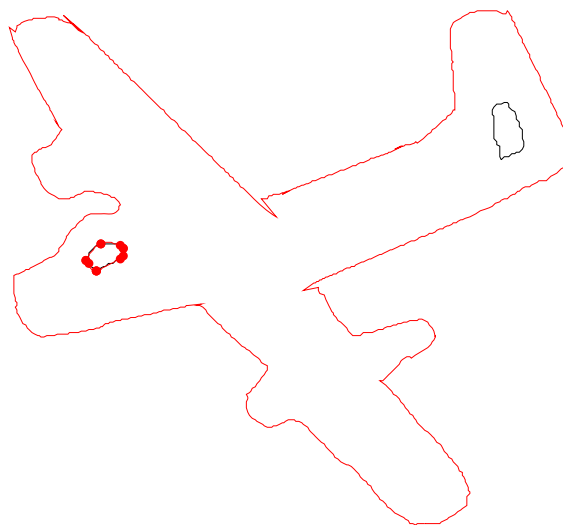
(b) At iteration = 10



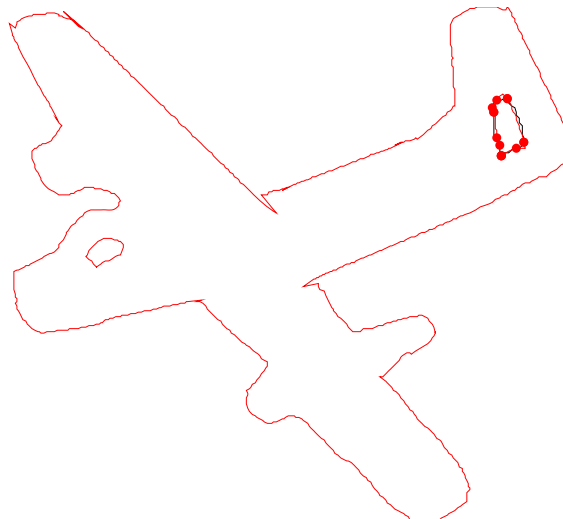
(c) At iteration = 20



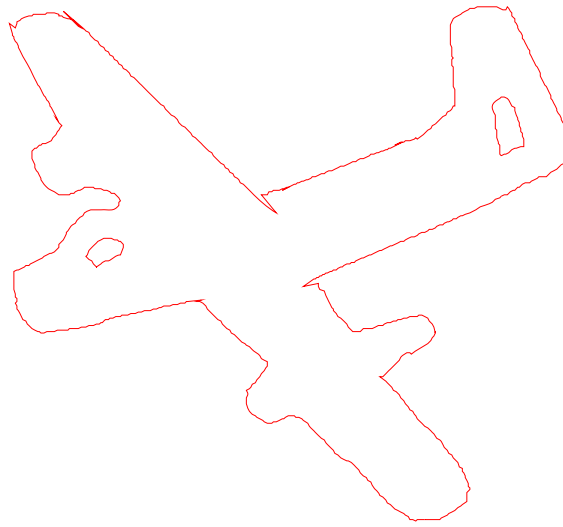
(d) At iteration = 30



(e) At iteration = 35

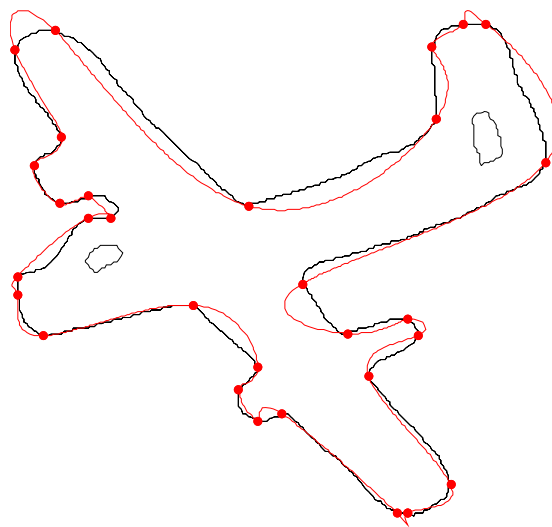


(f) At iteration = 40

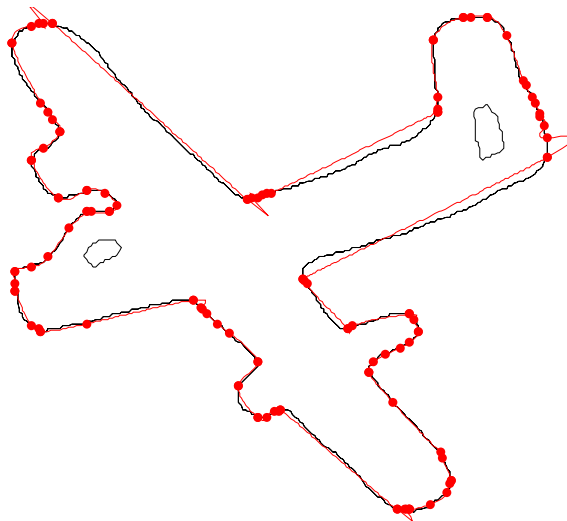


(g) Approximated spline for Object “Mult_Seg_Plane”

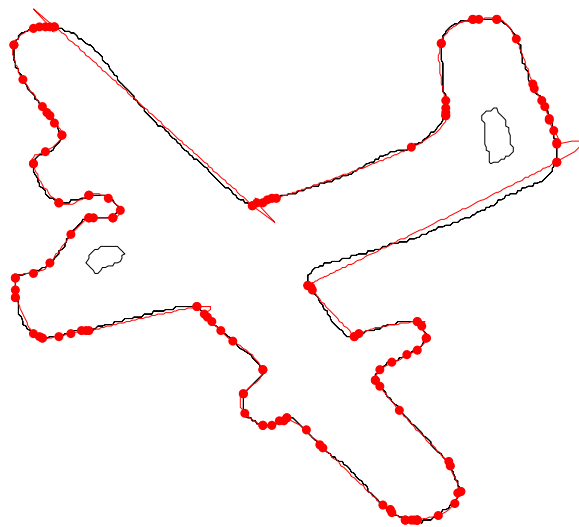
**Figure 4.90 Algorithm 4.3: Demonstration of spline fitting at each iteration using
threshold =1**



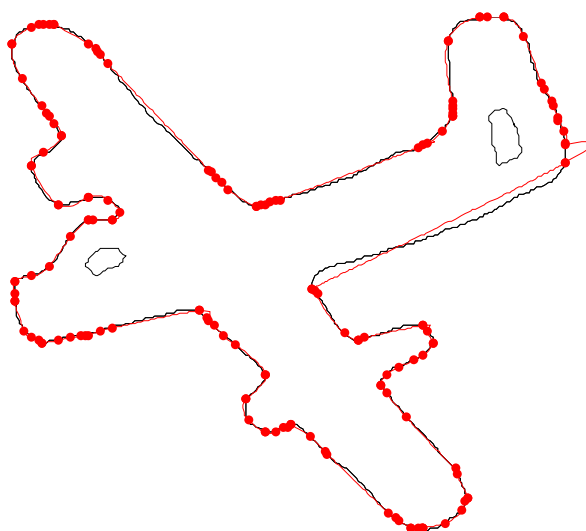
(a) At iteration = 1



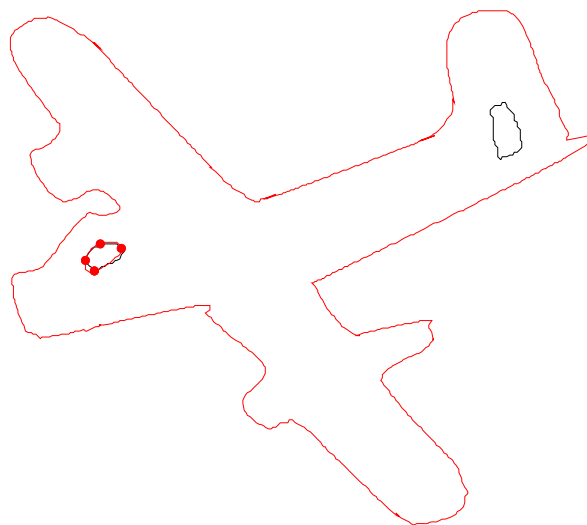
(b) At iteration = 10



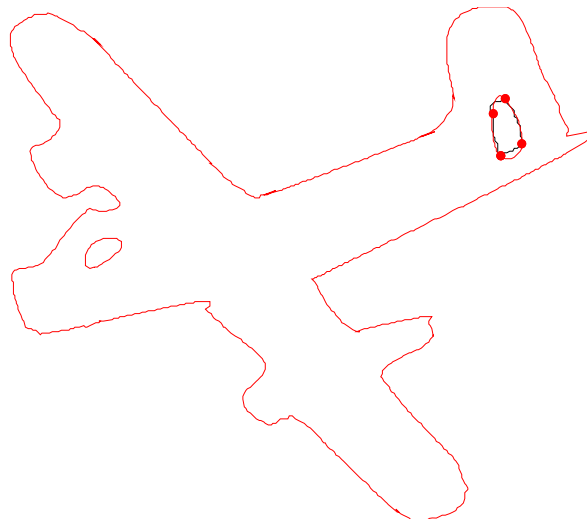
(c) At iteration = 20



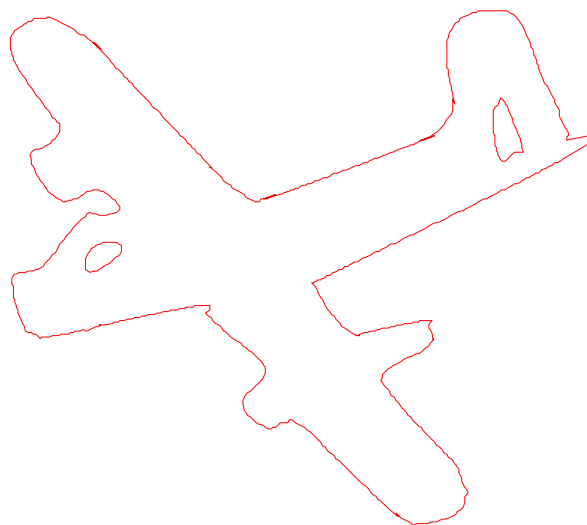
(d) At iteration = 30



(e) At iteration = 31

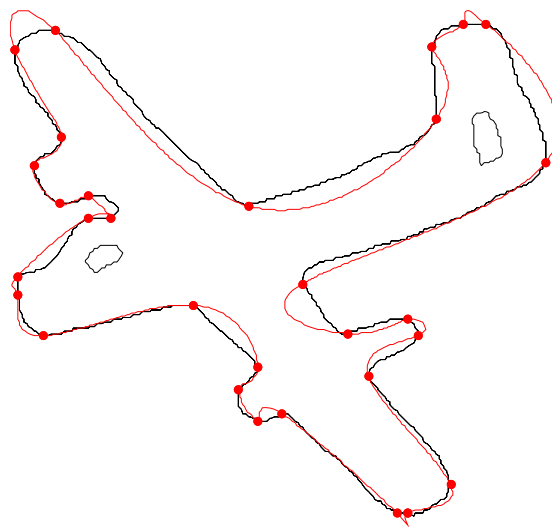


(f) At iteration = 32

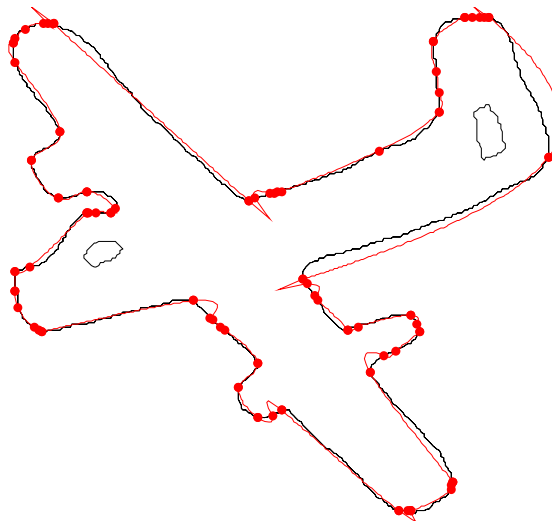


(g) Approximated spline for Object “Mult_Seg_Plane”

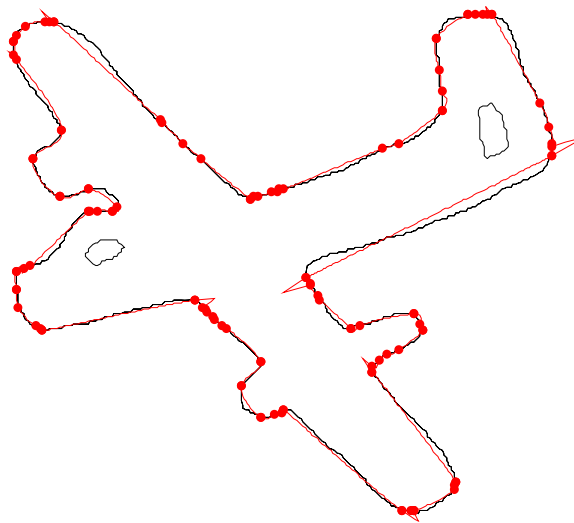
**Figure 4.91 Algorithm 4.3: Demonstration of spline fitting at each iteration using
threshold =2**



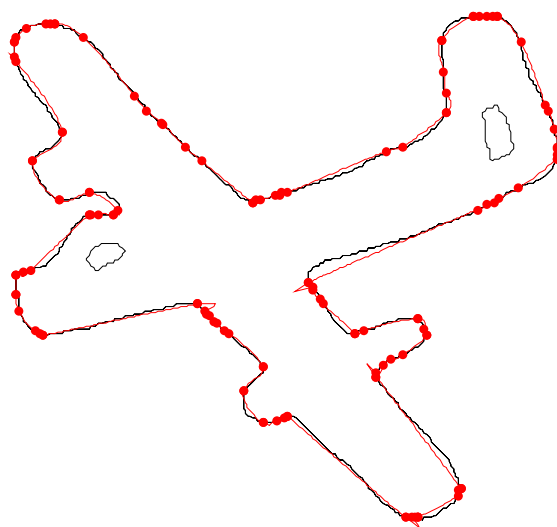
(a) At iteration = 1



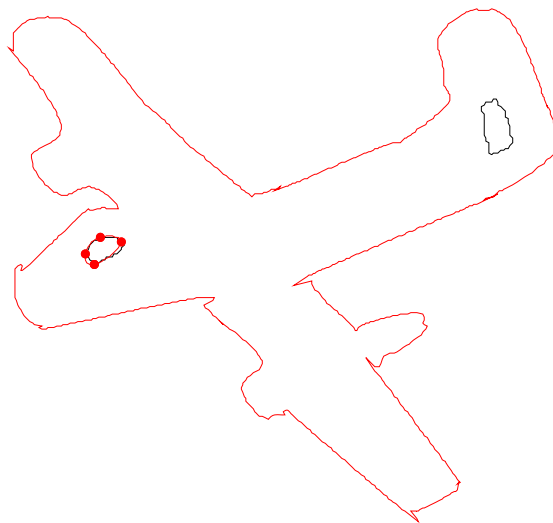
(b) At iteration = 10



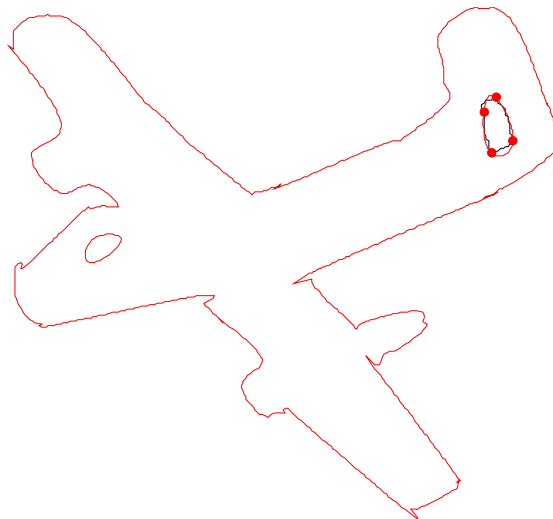
(c) At iteration = 20



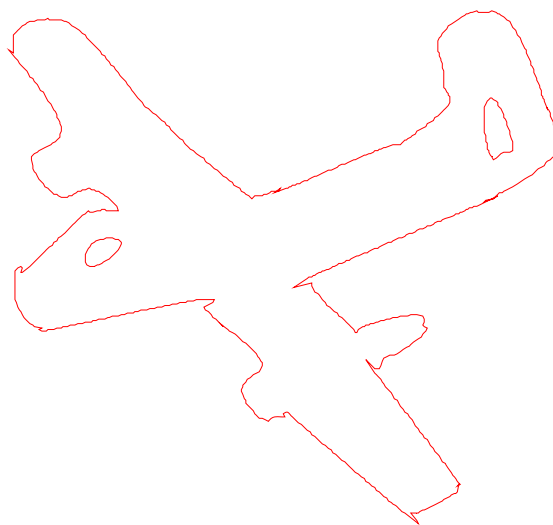
(d) At iteration = 30



(e) At iteration = 31



(f) At iteration = 32



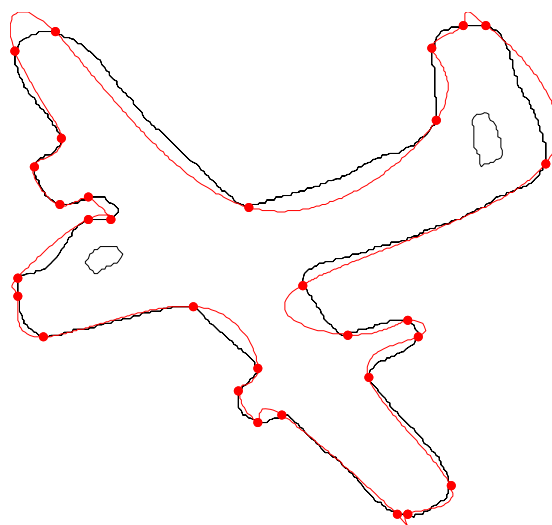
(g) Approximated spline for Object “Mult_Seg_Plane”

Figure 4.92 Algorithm 4.3: Demonstration of spline fitting at each iteration using threshold =3

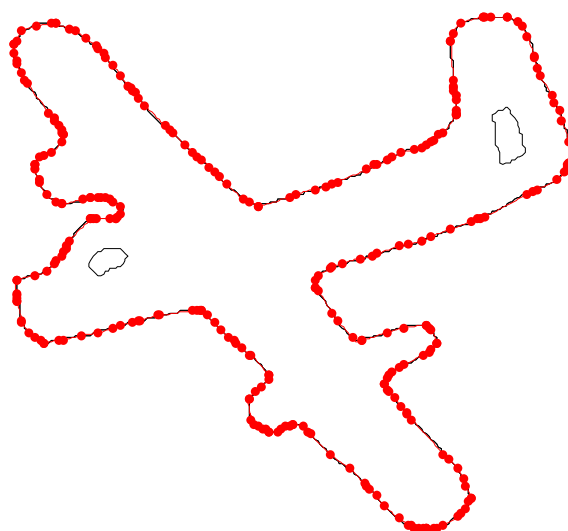
Table 4.27 Evaluation of algorithm 4.3 for Object “Mult_Seg_Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	165	44	1.68	1
1005	41	97	33	1.79	2
1005	41	70	33	1.74	3

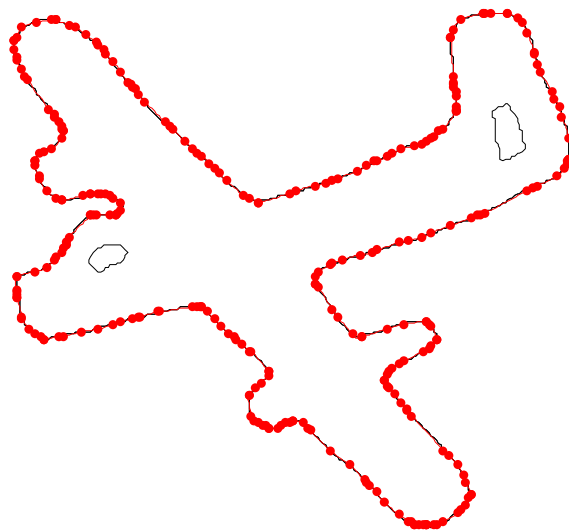
(Figure 4.93) to (Figure 4.95) shows the fitted curve over object contour at different iterations for algorithm 4.4 at threshold values of 1,2 and 3 respectively.



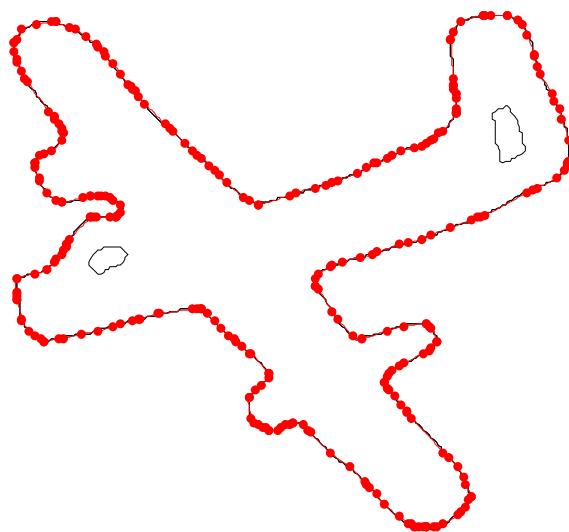
(a) At iteration = 1



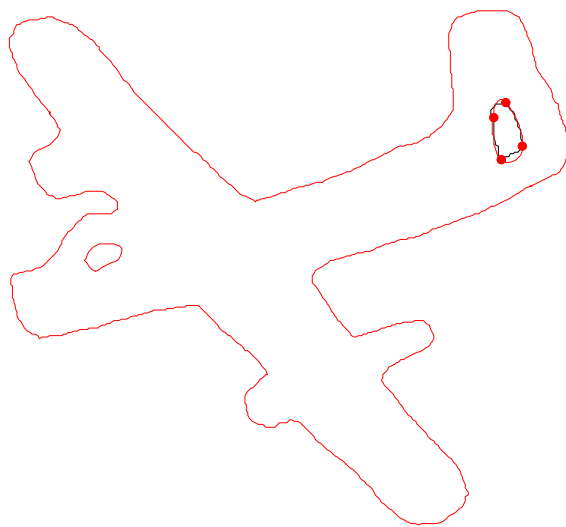
(b) At iteration = 10



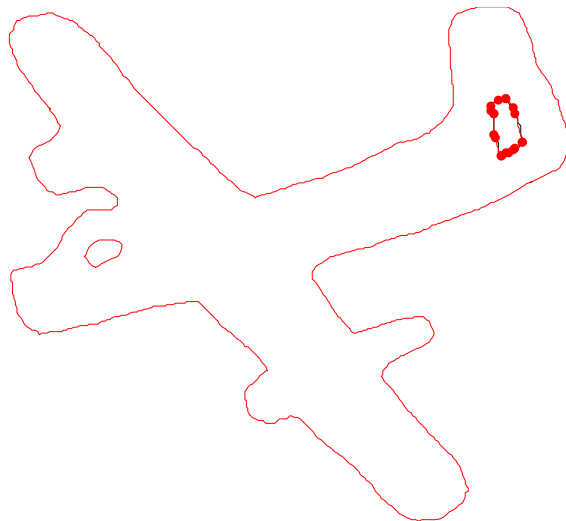
(c) At iteration = 20



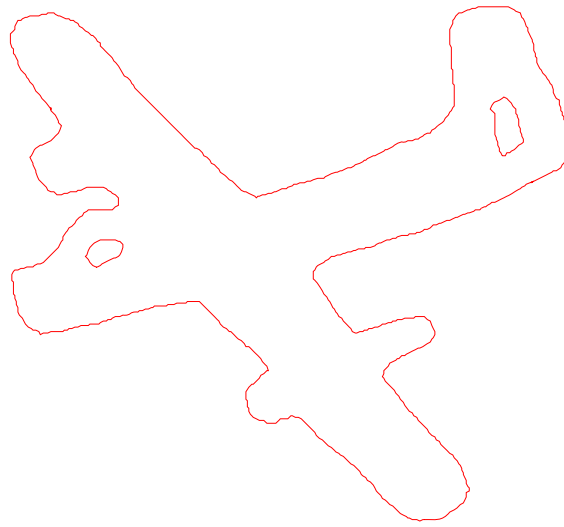
(d) At iteration = 30



(e) At iteration = 35

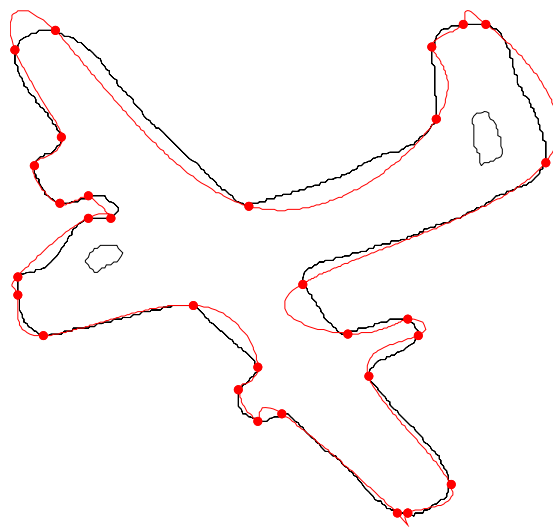


(f) At iteration = 40

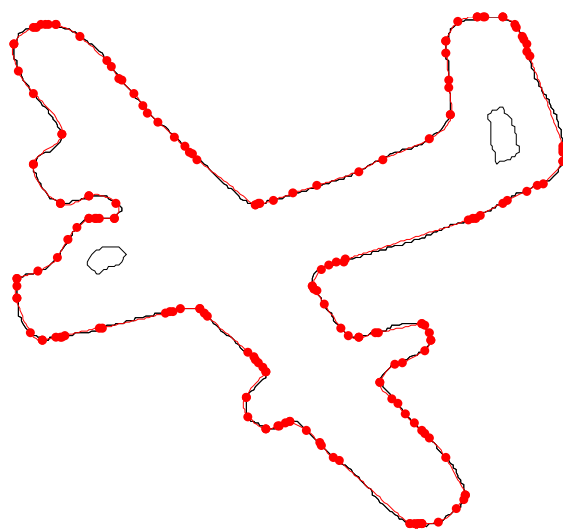


(g) Approximated spline for Object “Mult_Seg_Plane”

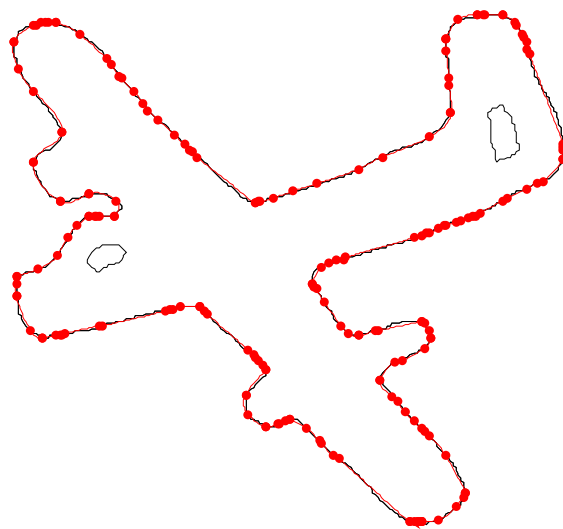
**Figure 4.93 Algorithm 4.4: Demonstration of spline fitting at each iteration using
threshold =1**



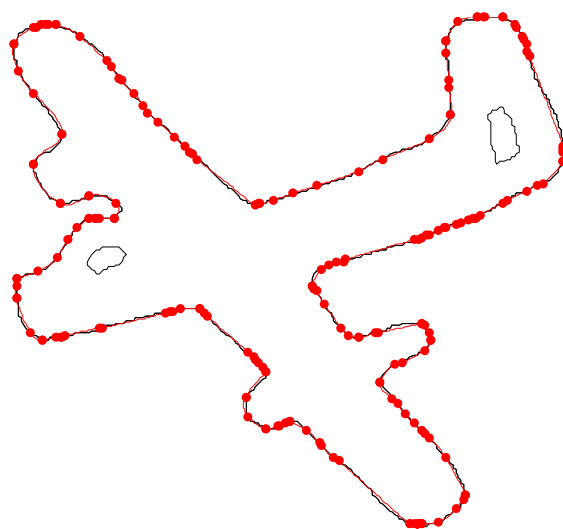
(a) At iteration = 1



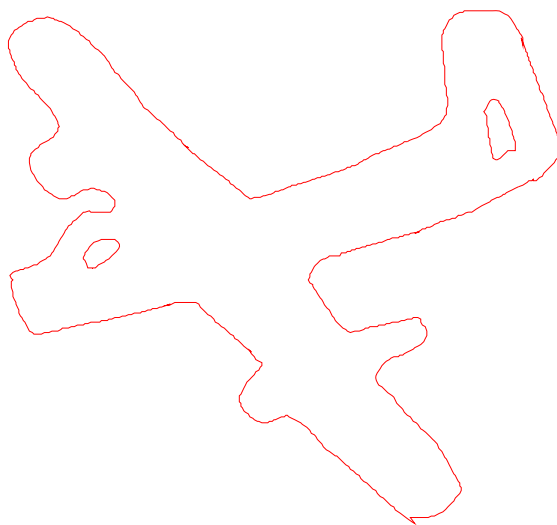
(b) At iteration = 10



(c) At iteration = 20

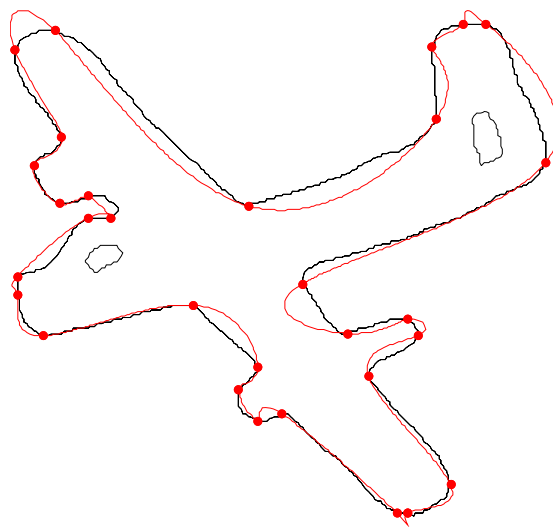


(d) At iteration = 30

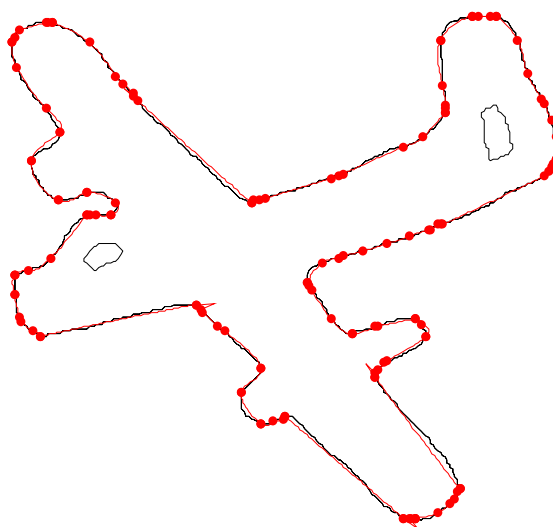


(e) Approximated spline for Object “Mult_Seg_Plane”

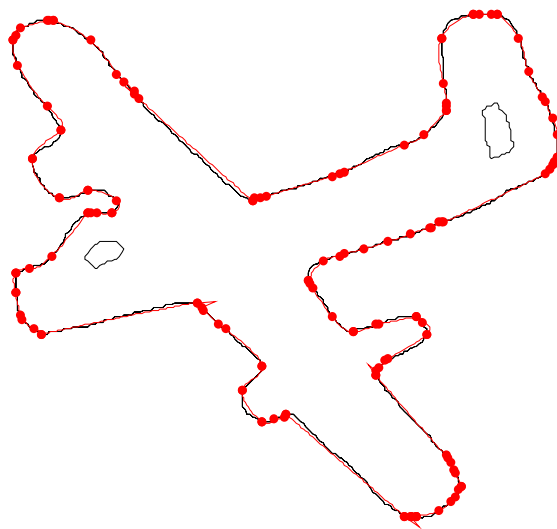
**Figure 4.94 Algorithm 4.4: Demonstration of spline fitting at each iteration using
threshold =2**



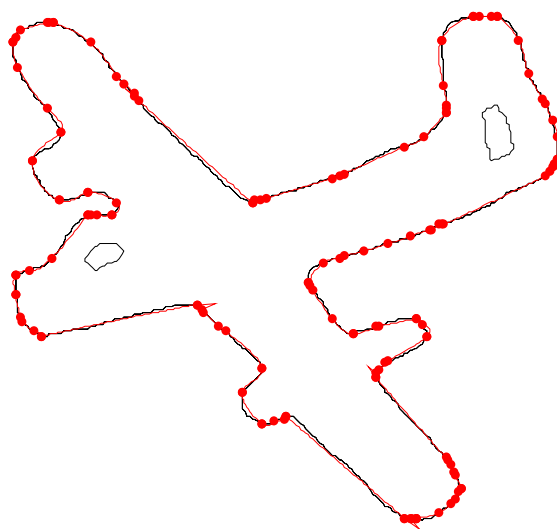
(a) At iteration = 1



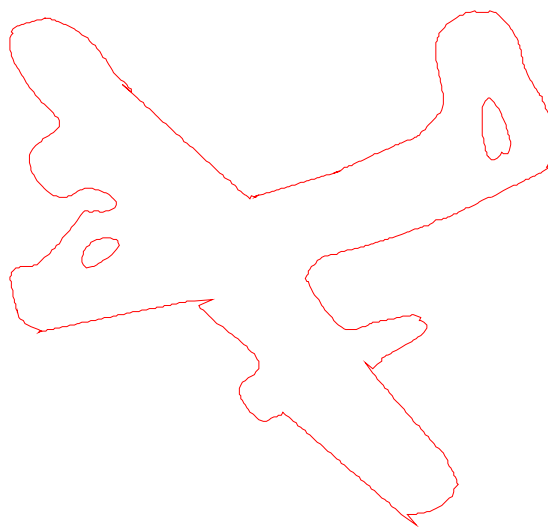
(b) At iteration = 10



(c) At iteration = 20



(d) At iteration = 30



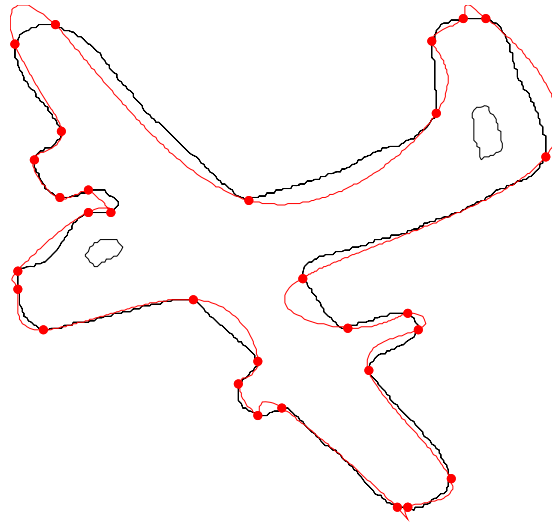
(e) Approximated spline for Object “Mult_Seg_Plane”

Figure 4.95 Algorithm 4.4: Demonstration of spline fitting at each iteration using threshold =3

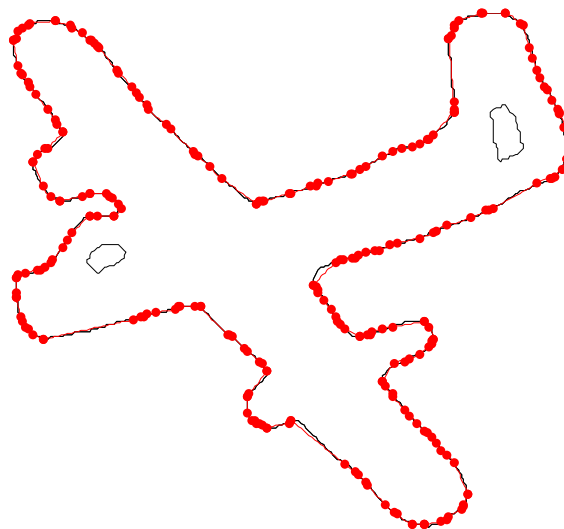
Table 4.28 Evaluation of algorithm 4.4 for Object “Mult_Seg_Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	774	49	49.87	1
1005	41	403	34	9.85	2
1005	41	316	33	6.21	3

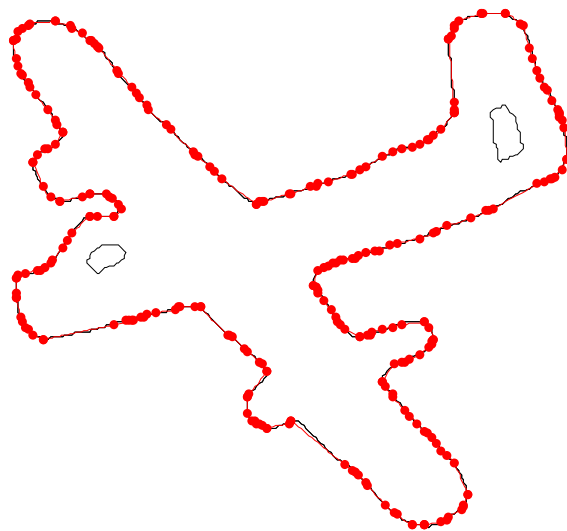
(Figure 4.96) to (Figure 4.98) shows the fitted curve over object contour at different iterations for algorithm 4.5 at threshold values of 1,2 and 3 respectively.



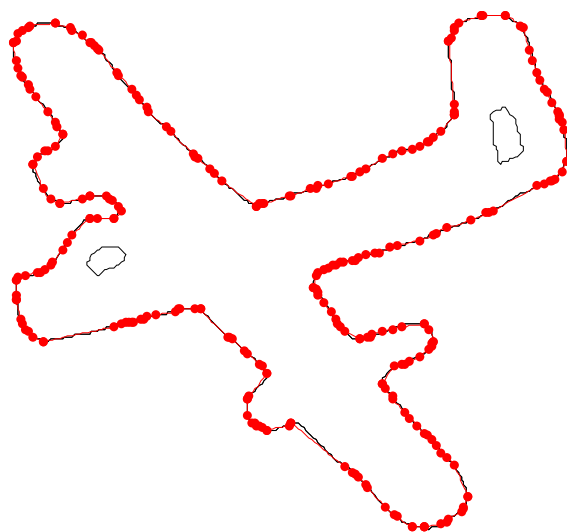
(a) At iteration = 1



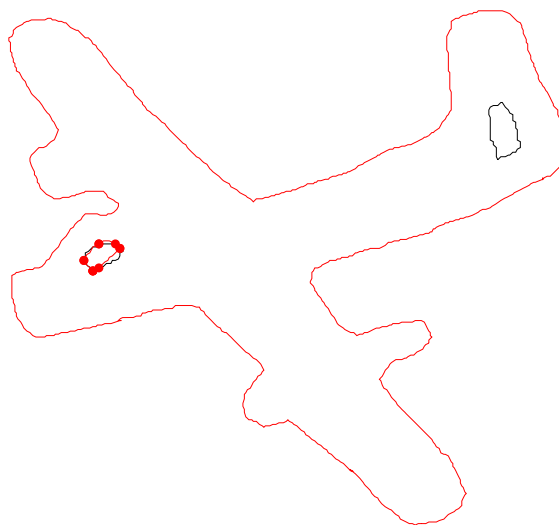
(b) At iteration = 10



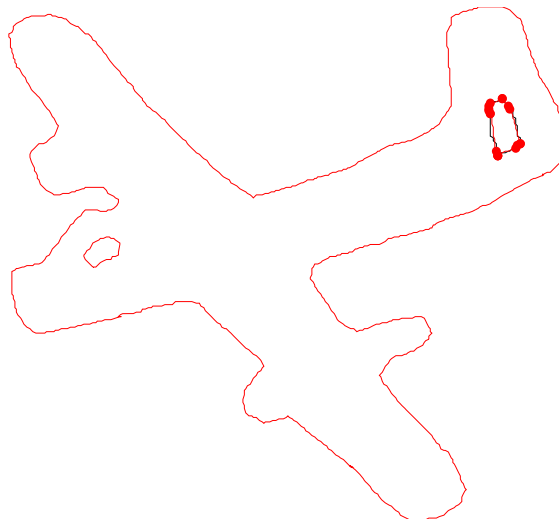
(c) At iteration = 20



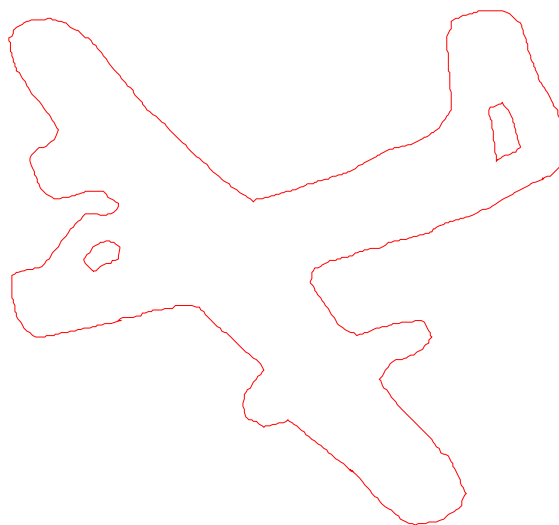
(d) At iteration = 30



(e) At iteration = 35

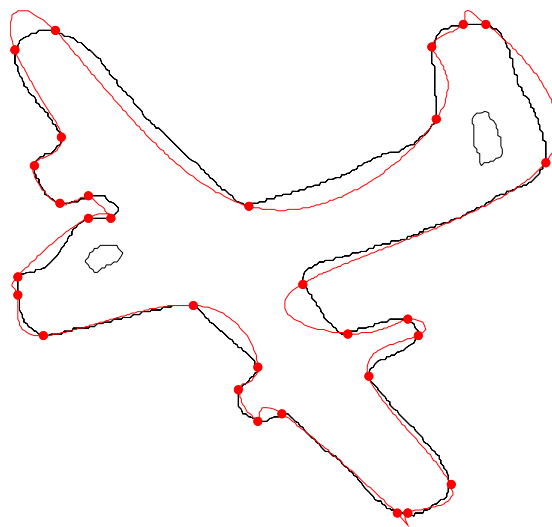


(f) At iteration = 45

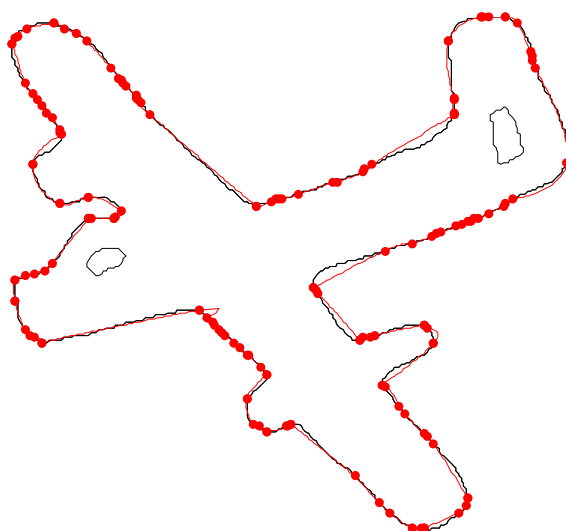


(g) Approximated spline for Object “Mult_Seg_Plane”

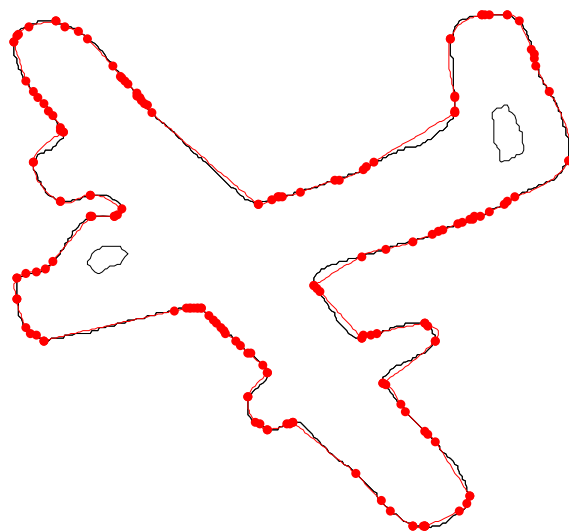
**Figure 4.96 Algorithm 4.5: Demonstration of spline fitting at each iteration using
threshold =1**



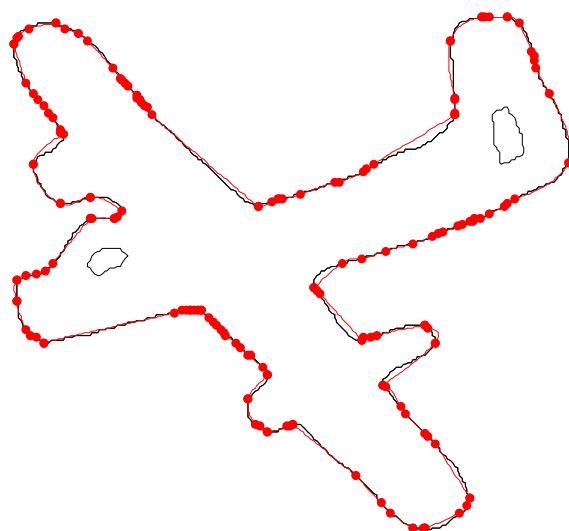
(a) At iteration = 1



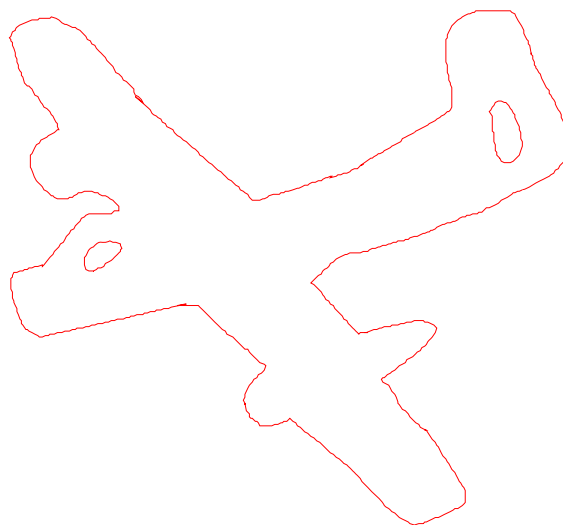
(b) At iteration = 10



(c) At iteration = 20

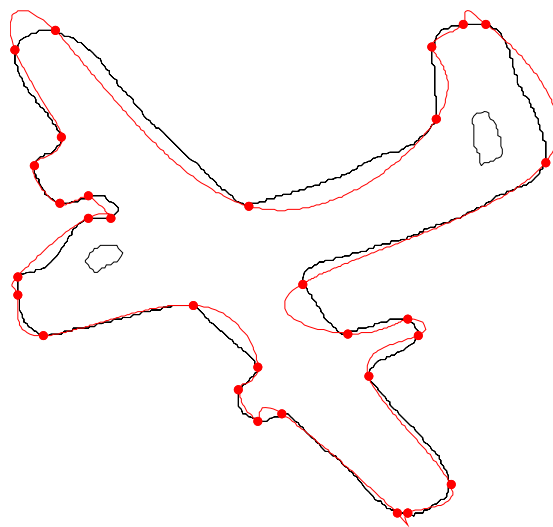


(d) At iteration = 30

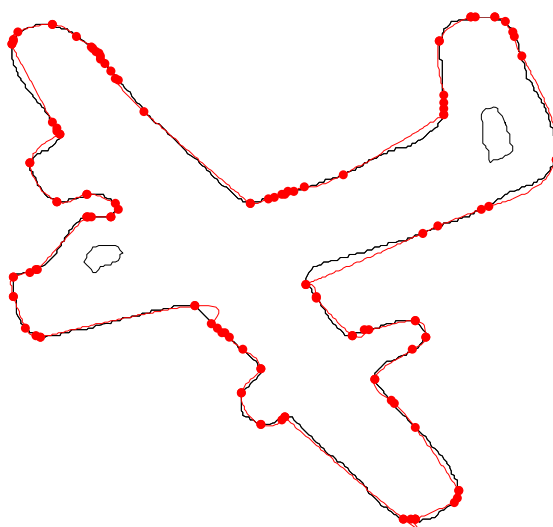


(e) Approximated spline for Object “Mult_Seg_Plane”

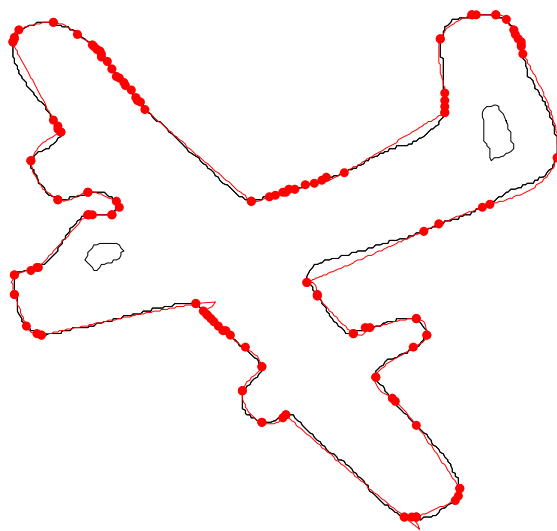
**Figure 4.97 Algorithm 4.5: Demonstration of spline fitting at each iteration using
threshold =2**



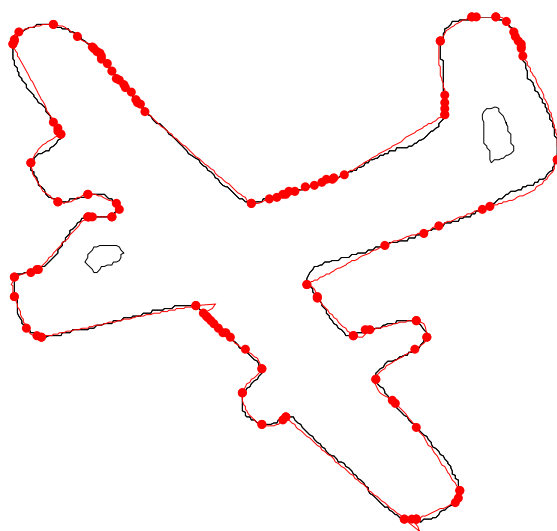
(a) At iteration = 1



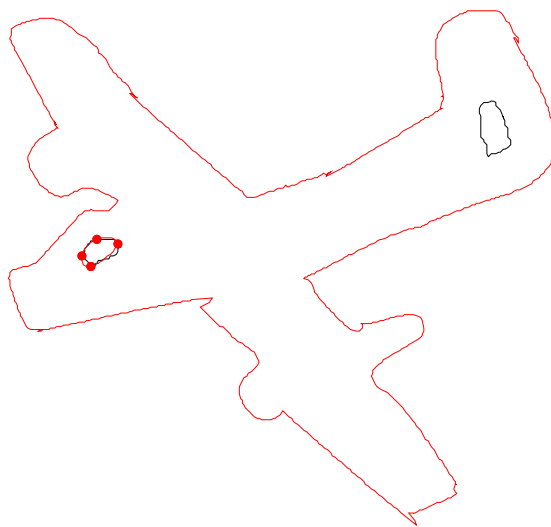
(b) At iteration = 10



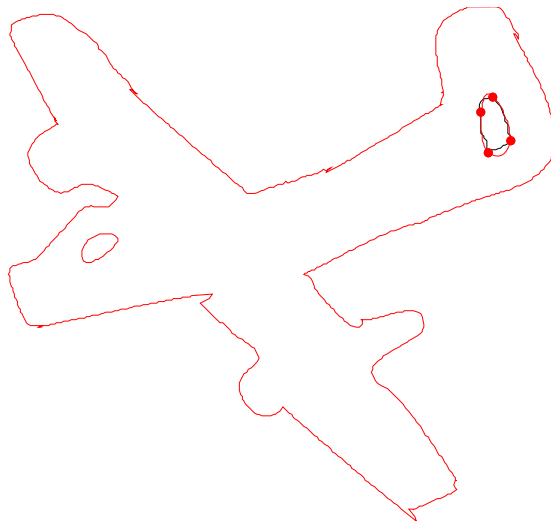
(c) At iteration = 20



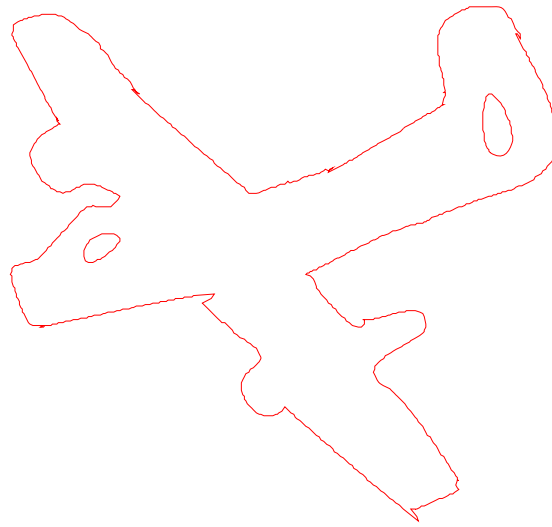
(d) At iteration = 30



(e) At iteration = 31



(f) At iteration = 32



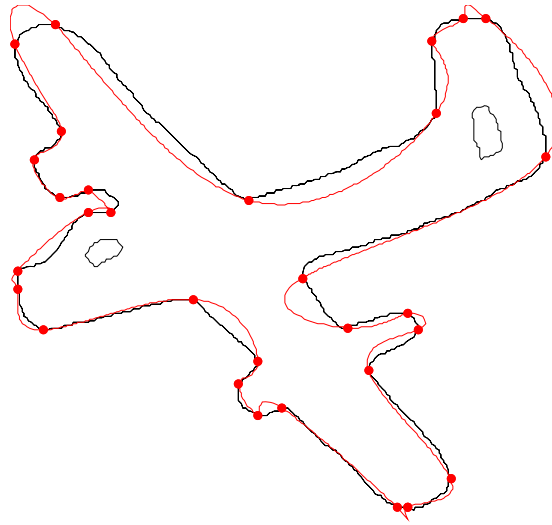
(g) Approximated spline for Object “Mult_Seg_Plane”

Figure 4.98 Algorithm 4.5: Demonstration of spline fitting at each iteration using threshold =3

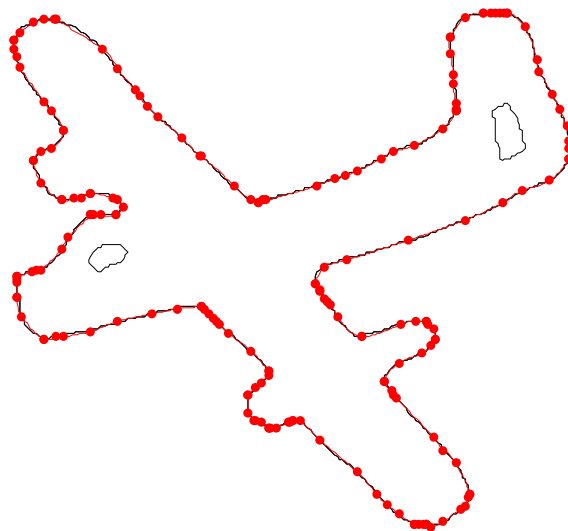
Table 4.29 Evaluation of algorithm 4.5 for Object “Mult_Seg_Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	591	45	26.84	1
1005	41	282	33	4.84	2
1005	41	218	32	3.51	3

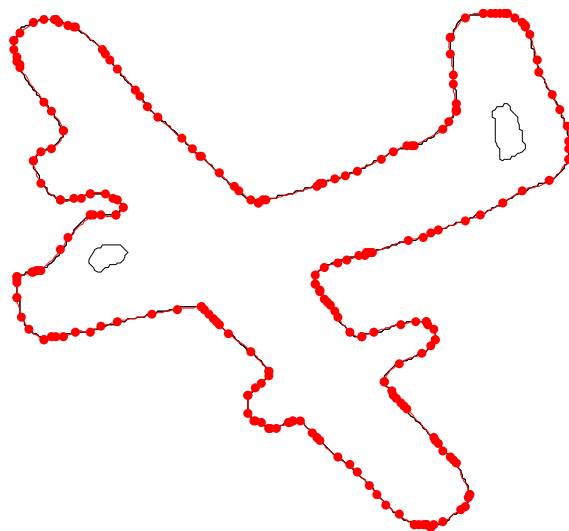
(Figure 4.99) to (Figure 4.101) shows the fitted curve over object contour at different iterations for algorithm 4.6 at threshold values of 1,2 and 3 respectively.



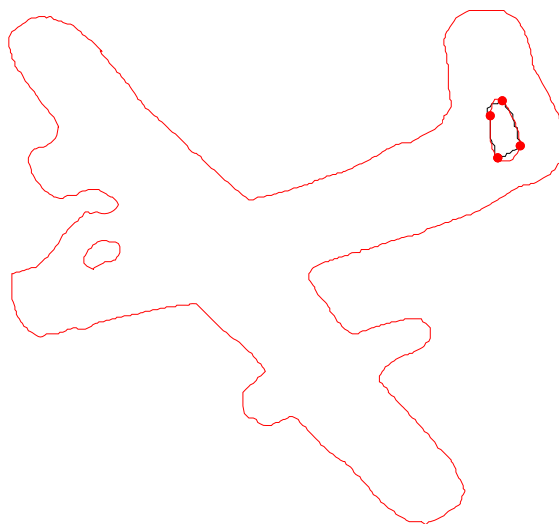
(a) At iteration = 1



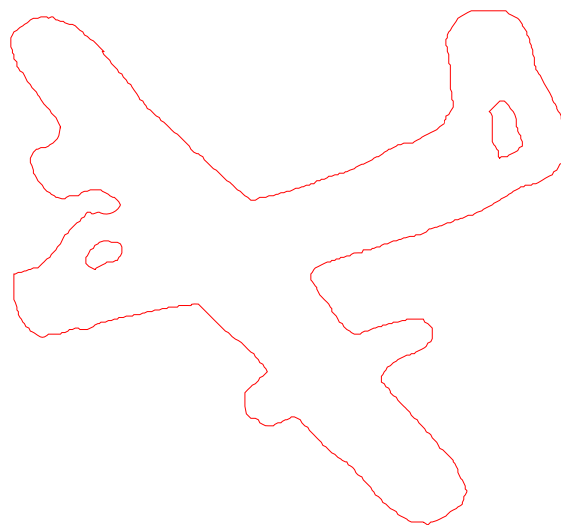
(b) At iteration = 5



(c) At iteration = 10

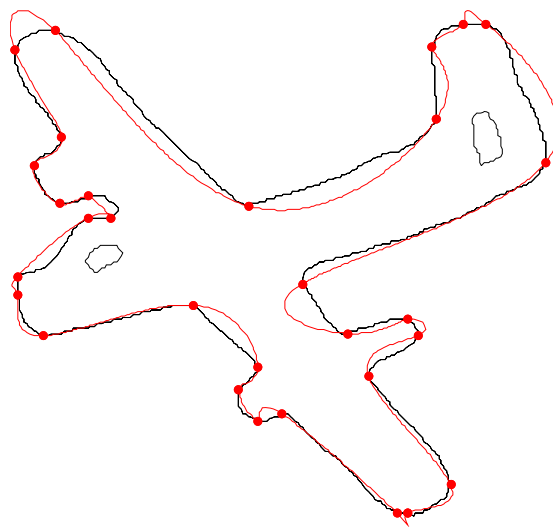


(d) At iteration = 15

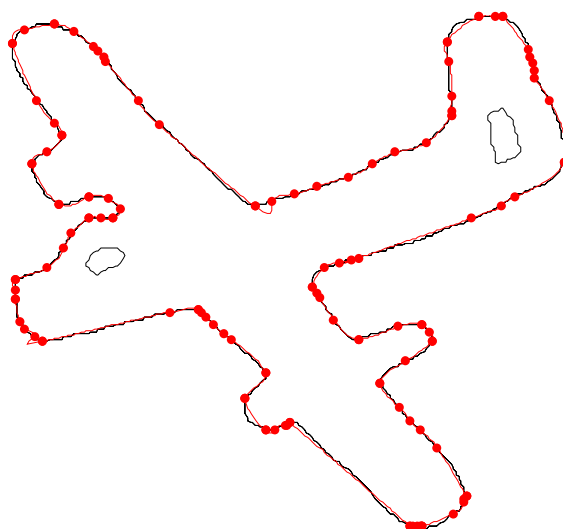


(e) Approximated spline for Object “Mult_Seg_Plane”

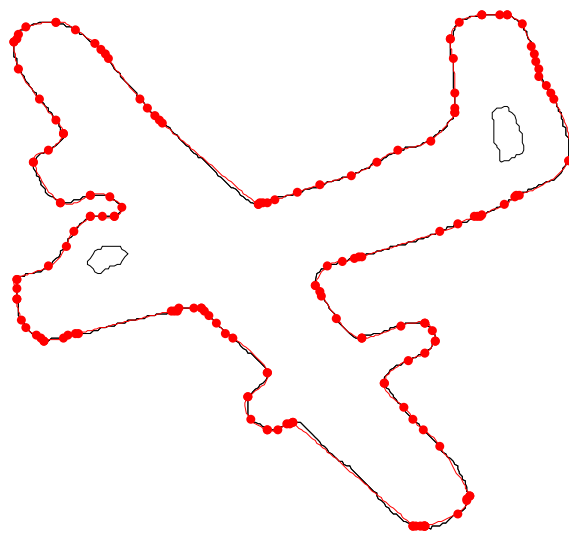
**Figure 4.99 Algorithm 4.6: Demonstration of spline fitting at each iteration using
threshold =1**



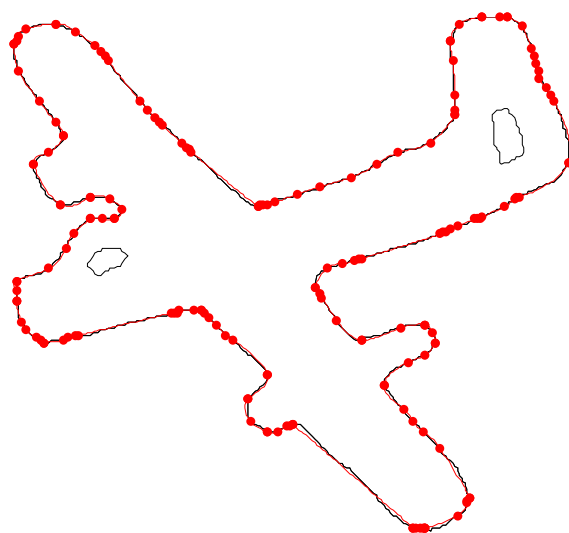
(a) At iteration = 1



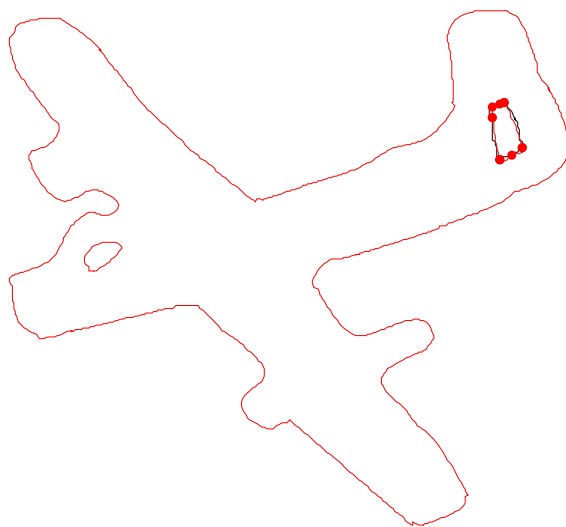
(b) At iteration = 5



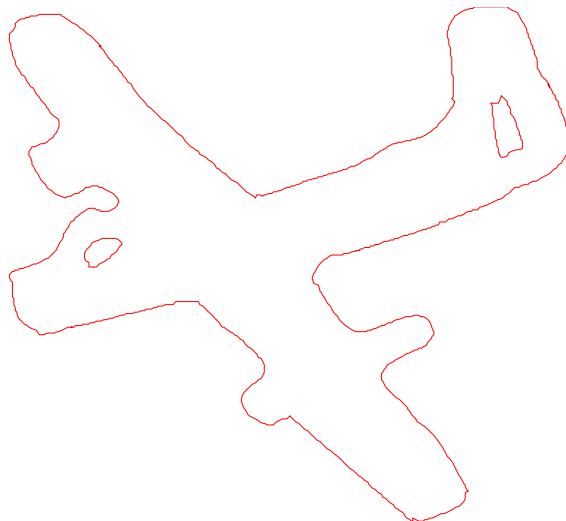
(c) At iteration = 10



(d) At iteration = 15

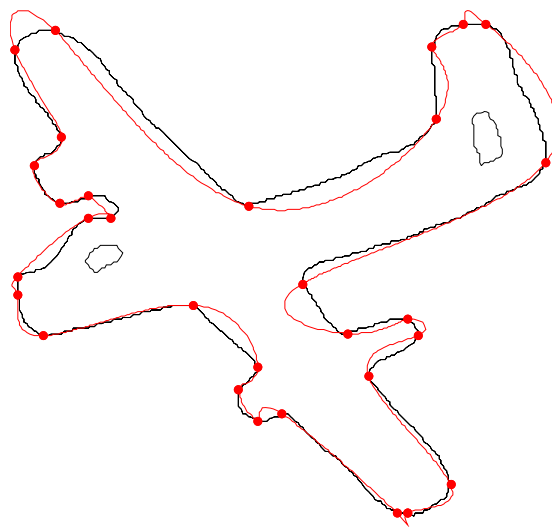


(e) At iteration = 20

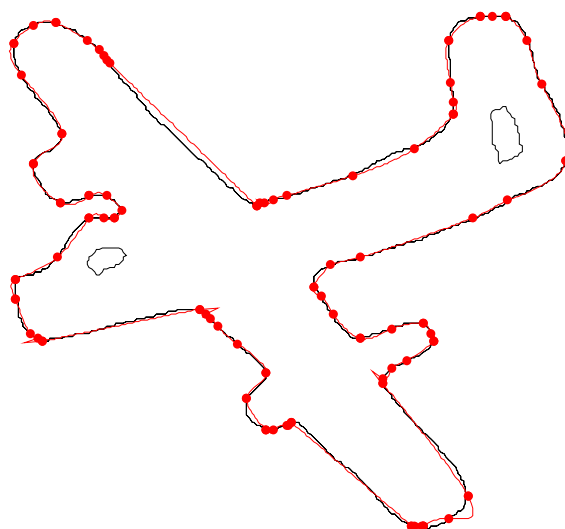


(f) Approximated spline for Object “Mult_Seg_Plane”

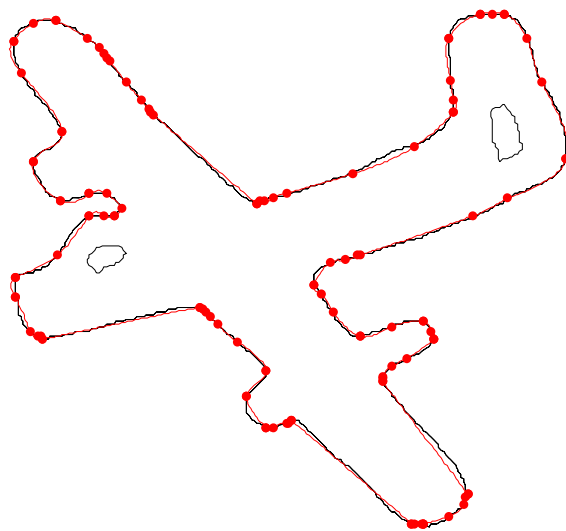
**Figure 4.100 Algorithm 4.6: Demonstration of spline fitting at each iteration using
threshold =2**



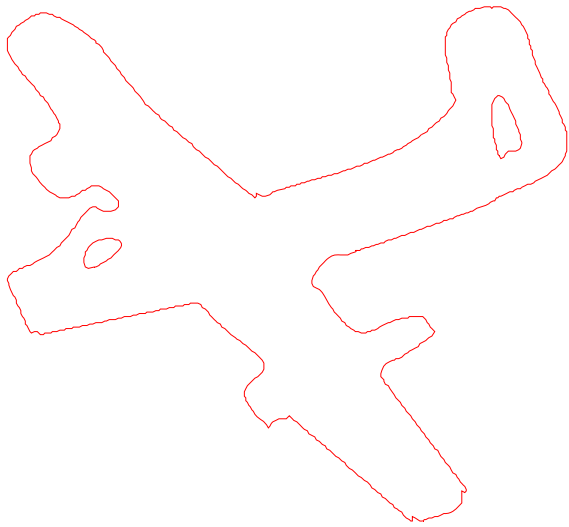
(a) At iteration = 1



(b) At iteration = 5



(c) At iteration = 10



(d) Approximated spline for Object “Mult_Seg_Plane”

Figure 4.101 Algorithm 4.6: Demonstration of spline fitting at each iteration using threshold =3

Table 4.30 Evaluation of algorithm 4.6 for Object “Mult_Seg_Plane”

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	197	18	2.01	1
1005	41	113	20	1.29	2
1005	41	58	13	1.43	3

(Table 4.31) to (Table 4.60) show the cumulative results for all algorithms at threshold values of 1,2 and 3.

Table 4.31 Evaluation of algorithm 4.1 for object ‘Ali’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	652	30	531	1
1644	33	343	30	64	2
1644	33	197	30	24	3

Table 4.32 Evaluation of algorithm 4.2 for object ‘Ali’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	308	30	112	1
1644	33	160	30	29	2
1644	33	96	30	22	3

Table 4.33 Evaluation of algorithm 4.3 for object ‘Ali’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	183	30	28	1
1644	33	144	30	26	2
1644	33	98	30	19	3

Table 4.34 Evaluation of algorithm 4.4 for object ‘Ali’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	1334	30	5219	1
1644	33	664	30	229	2
1644	33	422	30	65	3

Table 4.35 Evaluation of algorithm 4.5 for object ‘Ali’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	1002	30	1653	1
1644	33	431	30	325	2
1644	33	260	30	88	3

Table 4.36 Evaluation of algorithm 4.6 for object ‘Ali’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1644	33	344	14	81	1
1644	33	173	15	21	2
1644	33	91	10	9	3

Table 4.37 Evaluation of algorithm 4.1 for object ‘Apple’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	544	30	202	1
1242	13	219	30	50	2
1242	13	151	30	21	3

Table 4.38 Evaluation of algorithm 4.2 for object ‘Apple’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	271	30	57	1
1242	13	151	30	21	2
1242	13	105	30	16	3

Table 4.39 Evaluation of algorithm 4.3 for object ‘Apple’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	189	30	26	1
1242	13	107	30	17	2
1242	13	89	30	15	3

Table 4.40 Evaluation of algorithm 4.4 for object ‘Apple’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	1196	30	1299	1
1242	13	398	30	112	2
1242	13	305	30	53	3

Table 4.41 Evaluation of algorithm 4.5 for object ‘Apple’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	821	30	955	1
1242	13	443	30	142	2
1242	13	268	30	42	3

Table 4.42 Evaluation of algorithm 4.6 for object ‘Apple’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1242	13	289	16	35	1
1242	13	159	15	8	2
1242	13	101	16	9	3

Table 4.43 Evaluation of algorithm 4.1 for object ‘Plane’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	531	30	257	1
1293	27	254	30	39	2
1293	27	174	30	23	3

Table 4.44 Evaluation of algorithm 4.2 for object ‘Plane’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	285	30	78	1
1293	27	150	30	27	2
1293	27	86	30	18	3

Table 4.45 Evaluation of algorithm 4.3 for object ‘Plane’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	190	30	24	1
1293	27	110	30	17	2
1293	27	80	30	16	3

Table 4.46 Evaluation of algorithm 4.4 for object ‘Plane’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	971	30	1441	1
1293	27	572	30	293	2
1293	27	507	30	192	3

Table 4.47 Evaluation of algorithm 4.5 for object ‘Plane’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	863	30	1257	1
1293	27	390	30	130	2
1293	27	395	30	98	3

Table 4.48 Evaluation of algorithm 4.6 for object ‘Plane’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1293	27	332	30	64	1
1293	27	142	30	12	2
1293	27	92	30	5	3

Table 4.49 Evaluation of algorithm 4.1 for English character ‘D’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	314	55	1.85	1
849	15	155	39	1.09	2
849	15	47	19	0.59	3

Table 4.50 Evaluation of algorithm 4.2 for English character ‘D’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	137	60	0.92	1
849	15	68	35	0.67	2
849	15	50	57	0.73	3

Table 4.51 Evaluation of algorithm 4.3 for English character ‘D’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	141	60	0.93	1
849	15	94	60	0.87	2
849	15	60	43	0.78	3

Table 4.52 Evaluation of algorithm 4.4 for English character ‘D’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	612	60	14.17	1
849	15	216	43	1.2	2
849	15	125	26	1.53	3

Table 4.53 Evaluation of algorithm 4.5 for English character ‘D’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	618	60	9.43	1
849	15	324	60	2.18	2
849	15	220	36	1.42	3

Table 4.54 Evaluation of algorithm 4.6 for English character ‘D’

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
849	15	160	18	0.92	1
849	15	77	19	0.67	2
849	15	61	17	0.67	3

Table 4.55 Evaluation of algorithm 4.1 for object Mult_Seg_Plane

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	415	33	10.96	1
1005	41	224	34	2.45	2
1005	41	182	32	2.18	3

Table 4.56 Evaluation of algorithm 4.2 for object Mult_Seg_Plane

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	195	44	2.1	1
1005	41	90	34	1.21	2
1005	41	60	32	1.59	3

Table 4.57 Evaluation of algorithm 4.3 for object Mult_Seg_Plane

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	165	44	1.68	1
1005	41	97	33	1.79	2
1005	41	70	33	1.7	3

Table 4.58 Evaluation of algorithm 4.4 for object Mult_Seg_Plane

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	774	49	49	1
1005	41	403	34	9.85	2
1005	41	316	33	6.21	3

Table 4.59 Evaluation of algorithm 4.5 for object Mult_Seg_Plane

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	591	45	26	1
1005	41	282	33	4.84	2
1005	41	218	32	3.51	3

Table 4.60 Evaluation of algorithm 4.6 for object Mult_Seg_Plane

Total number of points on contour	Number of corner points	Number of knots inserted	Total number of iterations	Execution time	Threshold values
1005	41	197	18	2.01	1
1005	41	113	20	1.29	2
1005	41	58	13	1.43	3

Table 4.61 Evaluation of space efficiency in terms of reduction of dataset

Algorithm	Threshold	Object 'Ali' % Reduction in Dataset	Object 'Apple' % Reduction in Dataset	Object 'Mult_Seg_Plane' % Reduction in Dataset	Object 'D' % Reduction in Dataset	Object 'Plane' % Reduction in Dataset	Average % Reduction	Algorithmic Average % Reduction
1	1	60.34	56.19	58.70	63.01	58.93	59.43	76.36
	2	79.13	82.36	81.79	81.74	80.35	81.07	
	3	88.01	87.84	85.97	94.46	86.54	88.56	
2	1	81.26	78.18	80.59	83.86	77.95	80.37	87.90
	2	90.26	87.84	91.04	91.99	88.39	89.90	
	3	94.16	91.54	94.02	94.11	93.34	93.43	

Algorithm	Threshold	Object 'Ali' % Reduction in Dataset	Object 'Apple' % Reduction in Dataset	Object 'Mult_Seg_Plane' % Reduction in Dataset	Object 'D' % Reduction in Dataset	Object 'Plane' % Reduction in Dataset	Average % Reduction	Algorithmic Average % Reduction
3	1	88.86	84.78	83.58	83.39	85.30	85.18	89.73
	2	91.24	91.38	90.34	88.92	91.49	90.67	
	3	94.03	92.83	93.03	92.93	93.81	93.33	
4	1	18.85	3.70	22.98	27.91	24.90	19.67	52.27
	2	59.61	67.95	59.90	76.32	55.76	63.91	
	3	74.33	75.44	68.55	87.04	60.78	73.23	

Algorithm	Threshold	Object 'Ali' % Reduction in Dataset	Object 'Apple' % Reduction in Dataset	Object 'Mult_Seg_Plane' % Reduction in Dataset	Object 'D' % Reduction in Dataset	Object 'Plane' % Reduction in Dataset	Average % Reduction	Algorithmic Average % Reduction
5	1	39.05	33.89	41.19	27.20	33.25	34.92	60.17
	2	73.78	64.33	71.94	61.83	69.83	68.34	
	3	84.18	78.42	78.30	75.85	69.45	77.24	
6	1	79.07	76.73	80.39	82.92	74.32	78.68	87.31
	2	89.47	88.24	88.75	92.69	89.01	89.63	
	3	94.46	91.86	94.22	94.58	92.88	93.60	

The ideal objective in reengineering of objects is to get an accurate representation in least possible time with least possible data set. This defines the time and space complexity of an algorithm. However in practical situations, either of the two conditions is usually targeted.

Our algorithms are more time efficient as compared to space efficiency. We have observed that except for Algorithm 4.3, ‘Single Positional Distance Constraint Random Point’, all of the algorithms converge to the object contour in 15 iterations on the average. We can say that the rest of the iterations are taken as tuning iterations for achieving better reengineered result. In these tuning iterations we observe a very slight improvement on curve fitting outcome. Also from the result set we can observe that Algorithm 4.6, ‘Fuzzy Random Knot Selection’, performs the best in terms of convergence in total number of iterations. The specialty of this algorithm is that it does not need extra tuning iterations, thus improving overall time efficiency. Most importantly it is to be noted that our algorithms are linear in time.

It is analyzed from Table 4.61 that Algorithm 4.3, ‘Single Positional Distance Constraint Random Point’, performs best by reducing the dataset representation by 89%. Also Algorithm 4.2, ‘Single Euclidean Distance Constraint Random Point’ and Algorithm 4.6, ‘Fuzzy Random Knot Selection’, reduce the dataset representation by 87% and 87%. On the other hand Algorithm 4.4, ‘Three Unconstraint Random Points’ and Algorithm 4.5,

‘Three Equal Spaces Segmented Random Points’, perform the worst. The dataset reduction is very low.

A very important factor involved in our algorithms is that one can tune up the accuracy and space efficiency by adjusting the threshold value. In case accuracy is desired, then the threshold value can be decreased while on the other hand threshold value is increased to get the object representation in the least possible dataset. We have observed, Table 4.31 – Table 4.60, that Algorithm 4.3, ‘Single Positional Distance Constraint Random Point’, performs most uniformly. Such that the threshold value does not produce much effect on the space complexity however the accuracy is increased when the threshold value is decreased. The rest of the algorithms show a great deal of improvement in terms of space efficiency incase the threshold value is increased. Moreover we observed that overall there is no effect on the number of iterations for convergence.

Furthermore, it is also analyzed that the number of knots inserted is independent of the total number of points on the contour and also they do not depend upon the complexity of the object shape. This is because of the true randomize nature of the algorithms.

The close inspection and analysis of results depicted in Table 4.31 – Table 4.60, show that Algorithm 4.3, ‘Single Positional Distance Constraint Random Point’ and Algorithm 4.6, ‘Fuzzy Random Knot Selection’, give us better results in terms of both time and space complexity. Algorithm 4.3 is better in terms of space complexity and Algorithm 4.6

is better in terms of time complexity. Moreover Algorithm 4.6 is far beyond Algorithm 4.3 in case of space complexity. So, overall we can say that Algorithm 4.6 works best among all spline approximation algorithms in all the cases regardless of the threshold values. It not only converges in least iterations but also it selects near to the least number of knots and produces a very close fitted curve on the object contour.

4.5 Conclusion

A lot of applications in computer graphics, image processing, computer vision and CAD/CAM require the data points to be approximated by computing curves. These applications include outline capturing of bitmap images or fonts, designing of objects, data compression, regression analysis etc. The proposed work presented in this chapter, is concerned with efficient techniques of curve fitting to a large amount of digital planar data using cubic splines. We have developed enhanced techniques for knot insertion to obtain an approximation in lesser operations. Moreover, these algorithms are quite economical in terms of computation cost as they use a cubic function in their description and the randomized nature of algorithms is also simple. Our approaches can be used to visualize large set of data in much smaller data set. Further they can be used in object designing area. On the storage side, they require very few points to store and recover a planar image, especially the fuzzy approach is far better than any other approach in terms of time and space complexity.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1. Conclusion

The research work was substantially aimed to come up with an efficient strategy for object designing using smooth cubic splines. During the course of this thesis we also developed some efficient schemes of spline approximation which can also be used in object designing. To achieve the objective, several areas were analyzed and studied in depth, which are as under;

- Corner detection.
- Introduction of smoothness in parametric cubic spline model.
- Designing of interpolant form of smooth parametric cubic spline model.
- Designing of local support basis form of smooth parametric cubic spline model.

- Introduction of shape control parameters in the spline model.
- Designing of objects using smooth parametric cubic spline model.
- Approximating the object contour using the smooth parametric cubic spline model.

5.1.1. Corner Detection

A rough shape of digital object can be represented with the help of corner points. Thus, corners of a digital image store vital information for shape analysis. We have developed an efficient scheme for detecting corners in digital images. Our scheme's time complexity is linear. Moreover we have demonstrated that our algorithm neither detects wrong corners nor it leaves true corners undetected. Although our algorithm is not transformation invariant but we have shown that rotation does not produce much effect on shape descriptors. Further, the most important aspect of our algorithms is that our default tuning parameters work in the same way for all the objects. We have also shown that the tuning parameters do not have much effect when changed. That is, it is not required to change the tuning parameters as the input object is changed.

5.1.2. Spline Modeling

Splines are used to produce smooth curves. They can be applied to set up paths for object motions or to provide a representation for an existing object or drawing also they can be used to design object shapes. The important aspects to address for spline modeling and usage are to decide if it will be an interpolating or approximating spline. Interpolating splines mean that the curve is passing through the designated set of data points also

known as control points. Where as, approximating splines do not necessarily pass through the given set of control points. Further, degree of spline plays a very important role in object designing and approximation. The most often used class of splines is cubic, which offer a reasonable compromise between flexibility and speed of computation. Compared to higher-order polynomials, cubic splines require less calculations and memory and they are more stable. Compared to lower-order polynomials, cubic splines are more flexible for modeling arbitrary curve shapes.

We have formulated the mathematical notion of the interpolant and as well as local support basis form of cubic generic spline model. The interpolant form, as the name suggests is used for interpolation of given data set where as the local support basis form is used for approximation purposes. We have induced the GC^2 continuity in its description. This spline formulation recovers general cubic Bézier, Ball and Timmer curves as special cases. Also we have introduced the shape parameters for control over design. These shape parameters can be used to produce local or global tension. Moreover we have studied Timmer parametric cubic splines in detail and we have further proposed two interpolating schemes for curve rendering using C^1 and C^2 continuity.

5.1.3. Introduction of Smoothness in Curve Design

To attain a better shape it is required that the spline should permit the mixing of sharp and smooth sections within the same description. Continuity condition provides the

solution for this requirement. To achieve shapes with cusps, a zero order continuity condition can be used. For smooth shapes higher order continuity conditions are satisfied.

We have introduced GC^2 continuity in the generic cubic spline description. The benefit is that it can be used as C^2 continuous constraint with some change in parameter values.

5.1.4. Introduction of Shape Parameters

Shape parameters allow the designer to play with the shape of the object without ever changing the set of data points. These parameters in fact add a layer of shape control. Generally there are two kinds of shape parameters associated with curves. The first one is known as point tension. As the name suggests, it is associated with producing affect only on the neighborhood of a specific point. Where as the second type of shape parameter is concerned with producing the affect on an interval and thus known as interval tension parameter. We have successfully introduced the concept of shape parameters in the generic cubic curve design theory. We have also demonstrated the affect of changing the point and tension parameters. Further it is up to the designer to use these parameters globally or locally. Global change will impact the whole shape of the object where as local change will render the change in a specified interval or point.

5.1.5. Development of Interpolant Form of Spline Model

As we have already explained that interpolation is used when spline curve is suppose to pass through the given set of data points. Further it is used when the control points which

are describing the contour of the object are smooth with no sharp edges. We have devised an interpolant form of generic cubic spline model. For interpolation curve scheme, the data point set is transformed into Hermite form and the implementation of the tri-diagonal system is proposed for computation of tangents. We have used Type 1 natural end conditions for the formulation of tri-diagonal system of linear equations. The tangents are calculated at joining points of the segments in such a way that they follow GC^2 continuity.

5.1.6. Development of Local Support Basis Form

Local support basis formulation is used for spline approximation. This is less restrictive as compared to the interpolant form of spline model. In this case, as we have discussed earlier, the spline curve does not necessarily pass through the data points describing the contour of object. This approach is useful when the object to be approximated is not smooth. We have introduced the local support basis in the spline by transforming it in to piece wise Bernstein-Bézier representation. The freeform curve method is computed by the generation of Bezier points through B-Spline representation. The rationale of this conversion is to get all the desired properties of B-Spline like basis function.

5.1.7. Planar Object Approximation and Designing

One way to tackle object designing is by using spline approximation. Here it is required to find a close spline fit to the object contour. Once we get the approximated spline fit of the object we can apply all transformations on it. Thus allowing a designer to play with

the shape of the object or use it for designing purposes. Several steps are required to get approximated spline fit. Among them are, boundary extraction, corner detection and knot insertion or breaking the segment. We have used chain codes for boundary extraction. Most importantly we have proposed two classes of knot insertion algorithms. Both classes involve random process. Further we have also described our fuzzy criteria for selecting a random point amongst three using our proposed fuzzy membership function. We have shown that our proposed approaches are much efficient as far as the time complexity is concerned. We have also demonstrated that our approaches converge to the solution in very less iterations. However, these approaches are not good in terms of space complexity. Thus we claim that these approaches are very useful in on-line applications where space complexity is not much of concern.

5.2. Future Work

5.2.1. Designing of tuning parameter independent corner detector

In ideal situation it is required to have a corner detector which is independent of all kinds of tuning parameters and also which is transformation invariant. Even though our proposed corner detector's tuning parameters do not produce much difference when changed but still their presence invalidate an ideal condition.

5.2.2. Enhancement of curve fitting technique in terms of time and space complexity

A universal approach is required for curve fitting which could be applied both at online and offline applications. Thus it is required to find a solution which is efficient in terms of both time and space complexity.

5.2.3. Extension of concepts to 3D geometry

In this thesis we have explored the planar objects in detail. All our algorithms and techniques revolve around 2D objects. It will be interesting to find if these approaches are also applicable to 3D geometry.

References:

- [1] M. Sarfraz, A. Rasheed and Z. Muzaffar, "A Novel Linear Time Corner Detection Algorithm", Proceedings of International Conference on Computer Graphics, Imaging and Vision CGIV05, IEEE Computer Society, 27-29 July 2005, Beijing, China, pp. 191-196.
- [2] Wu-Chih Hu, "Multiprimitive Segmentation Based on Meaningful Breakpoints for Fitting Digital Planar Curves with Line Segments and Conic Arcs", Image and Vision Computing Vol. 23, Issue 9, September 2005, pp. 783-789.
- [3] Hiroyuki Kano, Hiroaki Nakata, Clyde F. Martin, "Optimal Curve Fitting and Smoothing Using Normalized Uniform B-Splines: A Tool for Studying Complex Systems", Applied Mathematics and Computation 169, 2005, pp. 96-128.
- [4] Zhouwang Yang, Jiansong Deng, Falai Chen, "Fitting Unorganized Point Clouds with Active Implicit B-Spline Curves", Proceedings of Special Issues of Pacific Graphics, Visual Computing Vol. 21, Issue 8-10, September 2005, pp. 831-839.
- [5] Guillaume Lavoue, Florent Dupont, Atilla Baskurt, "A New Subdivision Based Approach for Piecewise Smooth Approximation of 3D Polygonal Curves", Pattern Recognition Vol. 38, Issue 8, 2005, pp. 1139-1151.
- [6] Gobithassan Rudrudamy, "Designing Geometrically Continuous Curves Using Timmer Parametric Cubic", Master's Thesis, Department of Mathematics, FST, College University of Science and Technology Malaysia, March 2004.
- [7] M. Sarfraz, "Weighted Nu Spline with Local Support Basis Functions", Computer and Graphics, Vol. 28, 2004, pp. 539-549.

- [8] Sarfraz M., Asim M.R., Masood A., “Piecewise Polygonal Approximation of Digital Curves”, IEEE Computer Society in the proceedings of 8th International Conference on Information Visualization - IV 2004, London, England 14-16 July 2004, pp. 991-996.
- [9] Sarfraz, M., Asim, M. S., and Masood, A., “Capturing Outlines using Cubic Bezier Curves”, The Proceedings of The International Conference on Information & Communication Technologies: from Theory to Applications - ICTTA'04, Omayyad Palace, Damascus, Syria, IEEE Computer Society Press, USA, ISBN: 0-7803-8482-2, 2004.
- [10] Muhammad Sarfraz, “Some Algorithms for Curve Design and Automatic Outline Capturing of Images”, International Journal of Image and Graphics, Vol. 4, Issue 2, 2004, pp. 301-324.
- [11] Sarfraz M., Asim M.R., Masood A., “Web Based System for Capturing Outlines of 2D shapes”, International Conference on Information and Computer Science, KFUPM Dhahran, Saudi Arabia 28-30 Nov, 2004-ICICS' 2004, pp. 575-586.
- [12] Sarfraz M., Asim M.R., Masood A., “Capturing Outlines with Cubic Bezier Curves”, The Proceedings of The International Conference on Information and Communication Technologies: From Theory to Applications – ICTTA'04, IEEE-Computer Society, Syria, April 2004.
- [13] Sarfraz, M., Asim, M. R. and Masood, A. “A New Approach to Corner Detection”, Kluwer in The Book Series: Computational Imaging and Vision, Kluwer, ISBN: 1-4020-1817-7, 2004, pp. 181 – 197.

- [14] D.S Guru, R. Dinesh and P. Nagabhushan, "Boundary Based Corner Detection and Localization Using New 'Cornerity' Index: A Robust Approach", Proceedings of the First Canadian Conference on Computer and Robot Vision (CRV'04), IEEE, 2004.
- [15] Lu Sun, Y.Y. Tang, Xinge You, "Corner Detection for Object Recognition by Using Wavelet Transform", Proceedings of the Third International Conference on Machine Learning and Cybernetics, IEEE, Shanghai, 26-29 August 2004.
- [16] M. Sarfraz, M.A. Khan, "An Automatic Algorithm for Approximating Boundary of Bitmap Characters", Future Generation Computer Systems, Vol. 20, Issue 8, November 2004, pp 1327-1336.
- [17] Huaiping Yang, Wenping Wang, Jianguang Sun, "Control Point Adjustment for B-Spline Curve Approximation", Computer Aided Design Vol. 36, Issue 7, 2004, pp. 639-652.
- [18] Xummian Yang, "Curve Fitting and Fairing Using Conic Splines", Computer Aided Design Vol. 36, Issue 5, 2004, pp. 461-472.
- [19] C.C. Leung, C.H. Chan, F.H.Y. Chan and W.K. Tsui, "B-Spline Snakes in Two Stages", In Proceedings of 17th International Conference on Pattern Recognition ICPR'04, 2004, pp. 568-571.
- [20] Wenping Wang, Helmut Pottmann, Yan Liu, "Fitting B-Spline Curves to Point Clouds by Squared Distance Minimization", HKU CS Tech Report TR-2004-11.
- [21] Terézia P. V., "Curve and Surface Modeling with Spline Functions of Mixed Type", PhD Thesis, Budapest University of Technology and Economics, 2003.

- [22] Marji M., Siy P., "A New Algorithm for Dominant Points Detection and Polygonization of Digital Curves", *Pattern Recognition*, Vol. 36, Issue 10, 2003, pp. 2239-2251.
- [23] C.Urdiales, C. Trazegnies, A. Bandera and F. Sandoval, "Corner Detection Based on Adaptively Filtered Curvature Function", *Electronic Letters*, Vol. 39, Issue 5, 6th March 2003, pp. 426-428.
- [24] Wen-Yen Wu, "An Adaptive Method for Detecting Dominant Points", *Journal of Pattern Recognition*, Vol. 36, Issue 10, 2003, pp. 2231-2237.
- [25] J. H. Horng, "An Adaptive Smoothing Approach for Fitting Digital Planar Curves with Line Segments and Circular Arcs", *Pattern Recognition Letters* Vol. 24, Issue 1-3, January 2003, pp. 565-577.
- [26] B. Sarkar, L. K. Singh, D. Sarkar, "Approximation of Digital Curves with Line Segments and Circular Arcs Using Genetic Algorithms", *Pattern Recognition Letters*, Vol. 24, Issue 15, November 2003, pp. 2585-2595.
- [27] Pottmann, H., Hofer, M., "Geometry of the Squared Distance Function to Curves and Surfaces", In Hege, H., Polthier, K. (Eds) *Visualization and Mathematics III*, Springer, Berlin Heidelberg New York, 2003, pp. 223-244.
- [28] Pottmann, H., Leopoldseder, S., Hofer, M., "Approximation with Active B-Spline Curves and Surfaces", In *Proceedings of 10th Pacific Conference on Computer Graphics*, 2002, pp. 8-25.
- [29] M. Sarfraz & M.F.A. Razzak, "An algorithm for Automatic Capturing of the Font Outlines", *Computers & Graphics* Vol. 26, Issue 5, 2002, pp. 795-804.

- [30] Meek, D.S., "Coaxing a Planar Curve to Comply", Journal of Computational and Applied Mathematics, Vol. 140, Issue 1-2, March 2002, pp. 599-618.
- [31] Meek D.S., Walton D.J., "Planar G^2 Hermite Interpolation with Some Fair C-Shaped Curves", Journal of Computational Applied Mathematics, Vol. 139, Issue 1, February 2002, pp. 141-161.
- [32] M. Sarfraz & Arshad Raza, "Visualization of Data Using Genetic Algorithm", The Proceedings of the Fourth KFUPM Workshop on Information and Computer Science: Internet Computing, WICS, 2002.
- [33] M. Sarfraz, "Fitting Curve to Planar Digital Data", Proceedings of the Sixth International Conference on Information Visualization, IEEE, 2002, pp.633-638.
- [34] Z.J. Hou, G.W. Wei, "A New Approach to Edge Detection", Pattern Recognition, Vol. 35, Issue 7, July 2002, pp. 1559-1570.
- [35] G.Y. Wang, Z. Houkes, B.Zheng, X. Li, "A Note on Conic Fitting by the Gradient Weighted Least Squares Estimation: Refined Eigenvector Solution", Pattern Recognition Letters Vol. 23, Issue 14, 2002, pp. 1695-1703.
- [36] P. Reche, C. Urdiales, A. Bandera, C. Trazegnies and F. Sandoval, "Corner Detection by Means of Contour Local Vectors", Electronic Letters Vol. 38, Issue 14, July 2002, pp. 699-701.
- [37] Juttler, B., Felis, A., "A Least Square Fitting of Algebraic Spline Surfaces", Advance Computer Mathematics 17, 2002, pp. 135-152.
- [38] Piegl L.A., Tiller W., "Data Approximation Using Bi-arcs", Engineering with Computers, Vol. 18, Issue 1, 2002, pp. 59-65.

- [39] Bae S.H., Choi B.K., “BURBS Surface Fitting Using Orthogonal Coordinate Transform for Rapid Product Development”, *Computer Aided Design*, Vol. 34, Issue 10, 2002, pp. 683-690.
- [40] M. Sarfraz & M. A. Khan, “Automatic Outline Capture of Arabic Fonts”, *Information Sciences – Informatics and Computer Science: An International Journal*, Vol. 140, Issue 3, February 2002, pp. 269-281.
- [41] Murtaza Ali Khan, “An Efficient Font Design Method”, Master’s Thesis, KFUPM, January 2001.
- [42] M. Sarfraz & Arshad Raza, “Visualization of Data Using Genetic Algorithm”. *Soft Computing and Industry, Recent Applications*, Eds.: R. Roy, M. Koppen, S. Ovaska, T. Furuhashi, and F. Hoffmann, ISBN: 1-85233-539-4, 2002, pp. 535-544.
- [43] M. Sarfraz & Arshad Raza, “Genetic Algorithm and Data Visualization”, *The 6th Online World Conference on Soft Computing in Industrial Applications, WSC6: Multimedia and Internet Session*, 2001.
- [44] M. Sarfraz & Arshad Raza, “Capturing Outline of Fonts Using Genetic Algorithm and Splines”, *The proceedings of IEEE International Conference on Information Visualization-IV 2001-UK*, IEEE Computer Society Press, 25-27 July 2001, pp. 738-743.
- [45] Arshad Raza, “Genetic Algorithm for Visualization. Master’s Thesis”, KFUPM, 2001.
- [46] Muhammad Faisal Abdul Razzak, “A Web Based Automatic Outline Capturing of Images”, Master’s Thesis, KFUPM, December 2001.

- [47] S.J. Ahn, W. Tauh, H.J. Warnecke, "Least Squares Orthogonal Distances Fitting of Circle, Sphere, Ellipse, Hyperbola and Parabola", Pattern Recognition Vol. 34 Issue 12, December 2001, pp. 2283-2303.
- [48] Ahn Y.J., "Conic Approximation of Planar Curves", Computer Aided Design Vol. 33, Issue 12, 2001, pp. 867-872.
- [49] Yang X.N. Wang G.Z, "Planar Point Set Fairing and Fitting by Arc Splines", computer Aided Design 33(1), 2001, pp. 35-43.
- [50] Morse, B.S., Yoo, T.S., Chen, D.T., Rheingans, P., Subramanian, K.R., "Interpolating Implicit Surfaces from Scattered Surface Data using Compactly Supported Radial Basis Functions", In Shape Modeling International 01 Proceedings of the international Conference on Shape Modeling and Applications, IEEE Computer Society, Washington DC, 2001, pp. 89-98.
- [51] Carr, J.C., Beatson, R.K, Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R., "Reconstruction and Representation of 3D Objects with Radial Basis Functions", In Proceedings of SIGGRAPH, ACM Press New York 2001, 2001, pp. 67-76.
- [52] Park H. "Choosing Nodes and Knots in Closed B-Spline Curve Interpolation to a Point Data", Computer Aided Design Vol. 33, Issue 13, 2001, pp. 967-974.
- [53] M. Sarfraz, M.N. Haque, & M. A. Khan, "Capturing Outlines of 2D Images", The Proceedings of International Conference on Imaging Science, Systems and Technology (CISST 2000), Las Vegas, Nevada, CSREA Press, USA, 2000, pp. 87-93.

- [54] M. Sarfraz & M. A. Khan, "Towards Automation of Capturing Outlines of Arabic Fonts", Proceedings of the Third KFUPM Workshop on Information and Computer Science: Software Development for the New Millennium (WICS'2000), Saudi Arabia, 2000, pp. 83-98.
- [55] M.Sarfraz & M. Abdul Raheem, "Curve Designing Using a Rational Cubic Spline with Point and Interval Shape Control", The Proceedings of IEEE International Conference on Information Visualization-IV'2000-UK, IEEE Computer Society Press, USA, 2000, pp. 63-68.
- [56] Qi Duan, T.S. Chen, K. Djidjeli & W.G. Price, Twizell E.H., "A Method of Shape Control of Curve Design", Proceedings of Geometric Modeling and Processing, Digital Object Identifier, IEEE, Hong Kong 2000, pp. 184-189.
- [57] Bandera, A. Urdiales, C. Arrebola, F. and Sandoval, F., "Corner Detection by Means of Adaptively Estimated Curvature Function", Electronic Letters, Vol. 36, Issue 2, January 2000, pp. 124-126.
- [58] A. Ardeshtir Goshtasby, "Grouping and Parameterizing Irregularly Spaced Points for Curve Fitting", ACM Transactions on Graphics (TOG), Vol. 19, Issue 3, July 2000, pp. 185-203.
- [59] Foley, Van Dam, Feiner & Hughes, "Computer Graphics: Principles and Practice", Addison Wesley Publishing Company, 1999.
- [60] Dmitry Chetverikov & Zsolt Szabo, "A Simple and Efficient Algorithm for Detection of High Curvature Points in Planar Curves", Proceedings of 23rd workshop of the Australian Pattern Recognition Group, 1999, pp. 175-184.

- [61] Tsai D.M., H.T. Hou and H.J. Su, "Boundary Based Corner Detection Using Eigenvalues of Covariance Matrices", Pattern Recognition Letters, Vol. 20, Issue 1, 1999, pp. 31-40.
- [62] T. M. Cronin, "A Boundary Concavity Code to Support Dominant Point Detection", Pattern Recognition Letters, Volume 20, Issue 6, June 1999, pp. 617-634.
- [63] H.T. Sheu, W.C. Hu, "Multi-primitive Segmentation of Planar Curves – A Two Level Breakpoint Classification and Tuning Approach", IEEE Transactions on Pattern Analysis and Machine Intelligence Vol. 21, Issue 8, IEEE Computer Society, 1999, pp 791-797.
- [64] Zheng Z., Wang H. Teoh E., "Analysis of Gray Level Corner Detection", Pattern Recognition Letters, Vol. 20, 1999, pp. 149-162.
- [65] Walton D.J. Meek D.S., "Planar G^2 Curve Design with Spiral Segments", Computer Aided Design Vol. 34, Issue 13, 1998, pp. 1037-1046.
- [66] Z.Zhang, "Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting", Image and Vision Computing Journal, Vol. 15, Issue 1, 1997, pp 59-76.
- [67] Q. Zhu, L. Peng, "A New Approach to Conic Section Approximation of Object Boundaries", Image and Vision Computing Vol. 17, Issue 9, July 1999, pp. 645-658.
- [68] Chang H. & Yan H., "Vectorization of Hand-Drawn Image Using Piecewise Cubic Bézier Curves Fitting", Pattern Recognition, Elsevier Science, Vol. 31, Issue 11, 1998, pp. 1747-1755.

- [69] M. Sarfraz, "A Rational Spline with Tension: Some CAGD Perspectives", Proceedings of International Conference on Information Visualization, IV, IEEE Computer Society, 29-31 July, 1998.
- [70] M. Sarfraz, "Designing of Curves and Surfaces using Cubic Splines with Geometric Characterization", The Proceedings of International Conference on Information Visualization-IV'97, UK, IEEE Computer Society Press, 1997, pp. 82-94.
- [71] K. HSollig, J. Koch., "Geometric Hermite Interpolation with Maximal Order and Smoothness", Computer Aided Geometric Design, Vol. 13, Issue 8, 1996, pp. 681-695.
- [72] M. Sarfraz, "A Mathematical Model for Computer Graphics", Journal of Scientific Research, Pakistan, Vol. 25, Issue 1&2, 1996.
- [73] M. Sarfraz, "Designing of 3D Rectangular Objects", Lecture Notes in Computer Science 1024: Image Analysis Applications and Computer Graphics, Third International Computer Science Conference, ICSC'95, Springer-Verlag, 1995, pp. 411-418.
- [74] Ma, W.Y., Ruth, J.P., "Parameterization of Randomly Measured Points for Least Squares Fitting of B-Spline Curves and Surfaces", Computer Aided Geometric Design Vol. 27, Issue 9, 1995, pp. 663-675.
- [75] G.E. Farin, "Curves and Surfaces for Computer Aided Geometric Design", Academic press, New York, 1994.
- [76] M. Sarfraz, "Cubic Spline Curves with Shape Control", Computer and Graphics, Vol. 18, Issue 5, Ingenta, 1994, pp.707-713.

- [77] Chang S.P, J.H. Horng, "Corner Point Detection Using Nest Moving Average", Pattern Recognition Journal, Vol. 27, Issue 11, 1994, pp. 1533-1537.
- [78] M. Sarfraz, " C^2 Rational B-spline Surfaces with Tension Control", New Advances in CAD and Computer Graphics, 1993, pp.314-320.
- [79] Koichi Itoh & Yoshio Ohno, "A Curve Fitting Algorithm for Character Fonts", Electronic Publishing, Vol. 6, Issue 3, September 1993, 195-198.
- [80] D. Sarkar, "A Simple Algorithm for Detection of Significant Vertices for Polygonal Approximation of Chain-Code Curves", Pattern Recognition Letters, Vol. 14, Issue 12, 1993, pp. 959-964.
- [81] C. Arcelli, G. Ramella, "Finding Contour-Based Abstractions of Planner Patterns", Pattern Recognition Letters, Vol. 26, Issue 10, 1993, pp. 1563-1577.
- [82] B. K. Ray, K. S. Ray, "Detection of Significant Points and Polygonal Approximation of Digitized Curves", Pattern Recognition Letters, Vol. 13, Issue 6, June 1992, pp. 443-452.
- [83] B. K. Ray, K. S. Ray, "An Algorithm for Detecting Dominant Points and Polygonal Approximation of Digitized Curves", Pattern Recognition Letters, Vol. 13, 1992, pp. 840-856.
- [84] N. Ansari, K. W. Huang, "Non-Parametric Dominant Point Detection", Pattern Recognition Vol. 24, Issue 9, 1991, pp. 849-862.
- [85] H. C. Liu and M. D. Srinath, "Corner Detection from Chain-Code", Pattern Recognition, Vol. 23, Issue 1-2, 1990, pp. 51-68.

- [86] C. The, R. Chin, "On the Detection of Dominant Points on Digital Curves", IEEE Transaction on Pattern Analysis and Machine Intelligence Vol. 2, Issue 8, Aug 1989, pp. 859-872.
- [87] Thomas A. Foley, "Interpolation with Interval and Point Tension Controls Using Cubic Weighted V-Splines", ACM Transactions on Mathematical Software, Vol. 13, Issue 1, March 1987, pp 68-96.
- [88] Brain A. Barsky & John C. Beatty, "Local Control of Bias and Tension in Beta-Splines", ACM Transactions on Graphics, Vol. 2, Issue 2, April 1983.
- [89] Michael Plass & Maureen Stone, "Curve-Fitting with Piecewise Parametric Cubics", Computer Graphics, Vol. 17, Issue 3, July 1983, pp. 229-239.
- [90] Timmer, H.G., "Alternative Representation of Parametric Cubic Curves and Surfaces", Computer Aided Design, Vol. 12, Issue 1, 1980, pp. 25-28.

Vita

- Aiman Rasheed.
- Born in Tripoli, Libya on May 08, 1977.
- Received Bachelor of Science degree in Computer Science from University of Karachi, Karachi, Pakistan in January 2001.
- Co-operative Teacher in University of Karachi.
- Worked as a Software Engineer in VTR Services (PVT) Ltd.
- Joined KFUPM as Research Assistant in September 2002.

Publications

- **“Security of XML Documents”**, e-Business WorldExpo hosted by WowGao on June 16 and 17, 2004, Toronto, Canada.
- **“Methods of Protecting Stack Overflow Vulnerability”**, The Proceedings of ICICS, 2004, Dhahran, Saudi Arabia.
- **“A Novel Linear Time Corner Detection Algorithm”**, International Conference on Computer Graphics, Imaging and Vision cgiv05, 27-28 and 29 July 2005, Beijing, China.