4.6 Implementation and Software

Our algorithms are implemented in `Java` on the Eclipse Platform. See 3.7 for the hardware configuration. The code for $2D$ meshing is available for download at `http://cs.nyu.edu/exact/papers/cxy/`, and the code for $3D$ meshing is available for download at `http://cs.nyu.edu/exact/papers/cxyz/`.

Note that this implementation is based on machine arithmetic. This is valid since all arithmetic operations use only ring operations and divide by 2 and no under/overflows occur. The limitation of machine precision is that, for high degree polynomials, the code might fail because of under/overflows. Cxy algorithm has been transformed to `C++` based exact computational library `Core Library` by Shuxing Lu, and improved by Narayan Kamath. We will convert other `Java` codes to `C++` for distribution with our open source `Core Library`.

We use the default `Java` heap memory 256MB (some runs result in OutOfMemoryError (OME)). We implemented four algorithms: PV, Balanced Cxyz, Balanced Cxyz with epsilon precision, and Rectangular Cxyz. These are abbreviated as PV, Cxyz, Cxyze, and Rect-n (where n is the maximum aspect ratio). We did not implement Snyder's algorithm in $3D$ since it is relatively more complicated.

4.7 Experimental Results

We report some encouraging experimental results. Table 4.1 lists 11 examples of our tests. Table 4.2 compares the number of boxes and the running time among Cxyz, PV, and Rect-n ($n = 2, 4, 8, 16, 32$). The percentages represents the relative running times, using Cxyz as 100%. Figure 1.1, Figure 4.28, Figure 4.29, Figure 4.30, Figure 4.31, Figure 4.32 and Figure 4.33 illustrates the meshes for Eg.1 to Eg.7 in Table 1 respec-

tively, using Cxyze, PV, Cxyz and Rect-n, where n is selected in a way that Rect-n is the fastest among all Rect algorithms.

Table 4.1: Equations and input boxes of examples

#	Curve name	Equation $f(x, y, z) = 0$	Original Box
Eg1	tangle cube	$x^4 - 5x^2 + y^4 - 5y^2 + z^4 - 5z^2 + 10$	$[(-8, -8, -8), (8, 8, 8)]$
Eg2	chair	$(x^2 + y^2 + z^2 - 23.75)^2 - 0.8((z - 5)^2 - 2x^2)((z + 5)^2 - 2y^2)$	$[(-8, -8, -8), (8, 8, 8)]$
Eg3	quartic cylinder	$y^2x^2 + y^2z^2 + 0.01x^2 + 0.01z^2 - 0.01$	$[(-8, -8, -8), (8, 8, 8)]$
Eg4	quartic cylinder	$y^2(x - 1)^2 + y^2(z - 1)^2 + 0.01(x - 1)^2 + 0.01(z - 1)^2 - 0.2002$	$[(-5, -5, -5), (7, 7, 7)]$
Eg5	quartic cylinder	$y^2(x - 1)^2 + y^2(z - 1)^2 + 0.01(x - 1)^2 + 0.01(z - 1)^2 - 1.0002$	$[(-12, -12, -12), (14, 14, 14)]$
Eg6	shrek	$-x^4 - y^4 - z^4 + 4(x^2 + y^2z^2 + y^2 + z^2x^2 + z^2 + x^2y^2) - 20.7846xyz - 10$	$[(-8, -8, -8), (8, 8, 8)]$
Eg7	tritrumpet	$8z^2 + 6xy^2 - 2x^3 + 3x^2 + 3y^2 - 0.9$	$[(-8, -8, -8), (8, 8, 8)]$
Eg8a	eclipse	$x^2 + 10^2y^2 + 10^2z^2 - 1$	$[(-8, -8, -8), (8, 8, 8)]$
Eg8b(n)	eclipse	$x^2 + 10^ny^2 + 10^nz^2 - 1$	$[(-7, -7, -7), (8, 8, 8)]$

(1) Cxyz is at least as good as PV, and is significantly faster than PV in most examples. In Eg8b(4), Cxyz is 7.5 times faster than PV. In Eg8b(6), Cxyz spends 1.3 seconds to construct the mesh, compared to PV which spends more than 70 seconds, and runs out of memory. Rect is the fastest in both Eg8b(4) and Eg8b(6): Rect-2 spends 141 milliseconds for Eg8b(4), and 172 milliseconds for Eg8b(6). Note that the only exception is Eg8a, Cxyz and PV produce the same number of boxes, and spend the same amount of time. In Eg8b(2), we use the same function as Eg8a, but with an asymmetric original box. Cxyz is twice as fast as PV. Also note that in the Eg3, Cxyz and PV also produce the same number of boxes, but Cxyz is faster than PV because the computational cost for the C_1 predicate is bigger than the C_{xyz} predicate.

(2) Rect can be significantly faster than Cxyz, but the performance of Rect is inconsistent. In Eg3, Rect-32 takes 11.8% of Cxyz's running time; and in Eg8b(6), Rect-2 takes 12.8% of Cxyz's running time. The input surface for these examples are very long and thin, in which Rect algorithm can take advantage of various aspect ratios. The results also show that although Rect produces less boxes than Cxyz in all examples but Eg8b(2), the running time of Rect is not always faster than the Cxyz (especially when the input surface is squarish, like Eg2). This is because Rect needs to spend more time

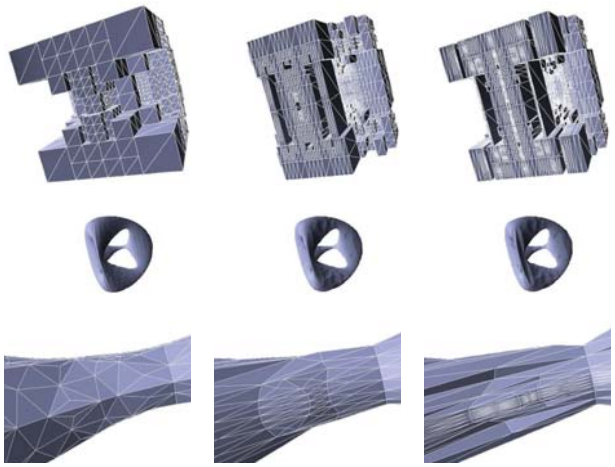
to check the criteria before splitting a box, and needs to process each box in three directions in Rect.

(3) Increasing the maximum aspect ratio n in Rect does not necessarily improve the performance of the algorithm. In Eg3, increasing the maximum aspect ratio directly improves the performance of Rect; but in Eg8b(6), it causes an opposite effect. This is because increasing the maximum aspect ratio might cause the boxes to "over split" in one direction, which is also the reason for the inconsistency of Rect. Another example for over-splitting in Rect is Eg2, where Rect-n spends more time than Cxyz. Figure 4.27 shows the resulting boxes, meshes, and details by running Cxyz, Rect-8, and Rect-32 on Eg2.

(4) Figure 4.34 illustrates an example that our algorithms preserve the topology: the first row of Figure 4.34 shows the approximations of Eg4 using Rect-n ($n = 2, 4, 8, 16, 32$) algorithm. It is not clear that the topology of the resulting meshes is the same by looking at the squared area. By zooming in the squared area (see the second row of Figure 4.34), We could see that the topology is preserved in the squared area of the meshes.

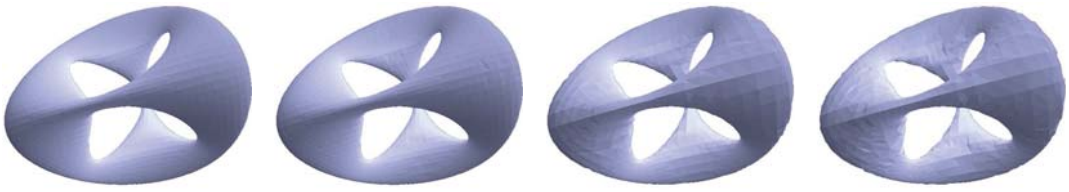
Table 4.2: Cxyz vs. PV vs. Rect-n

Box/Time (ms)/%	Cxyz	PV	Rect-2	Rect-4	Rect-8	Rect-16	Rect-32
Eg1	2584/391	5104/718/184%	1096/579/148%	1304/656/168%	1710/781/200%	2081/922/236%	2653/1125/288%
Eg2	26104/4516	106072/15765/349%	13400/7360/163%	19847/10672/236%	25513/13656/302%	30880/16797/372%	36931/20360/451%
Eg3	35792/3437	35792/3843/112%	12056/2812/82%	6264/1625/47%	3328/953/28%	2000/578/17%	1088/407/12%
Eg4	80662/10282	$OME_{>90sec.}$	43977/17875/174%	32836/13313/129%	27577/10766/105%	29143/11797/115%	26700/10594/103%
Eg5	134163/17187	$OME_{>90sec.}$	64617/35156/205%	37237/14703/86%	30730/12188/71%	27612/11187/65%	26221/10532/61%
Eg6	31144/4046	99436/11985/296%	13688/5421/134%	16348/6922/171%	19332/8422/208%	21698/10328/255%	23827/11469/283%
Eg7	1688/328	2920/421/128%	796/359/109%	836/390/119%	1028/422/129%	1244/453/138%	1652/578/176%
Eg8a	400/94	400/94/100%	176/125/133%	200/140/149%	232/156/166%	272/156/166%	320/172/183%
Eg8b(2)	274/125	2164/250/200%	149/109/87%	154/109/87%	197/125/100%	225/140/112%	279/140/112%
Eg8b(4)	1247/203	22121/1531/754%	345/141/69%	418/141/69%	484/156/77%	551/172/85%	658/203/100%
Eg8b(6)	15226/1343	$OME_{>70sec.}$	696/172/13%	733/187/14%	886/203/15%	952/203/15%	1129/219/16%



(a) Cxyz (b) Rect-8 (c) Rect-32

Figure 4.27: Boxes, meshes, and details of Eg2 using Cxyz, Rect-8 and Rect-32. Note that the triangles are elongated as the maximum aspect ratio increases.

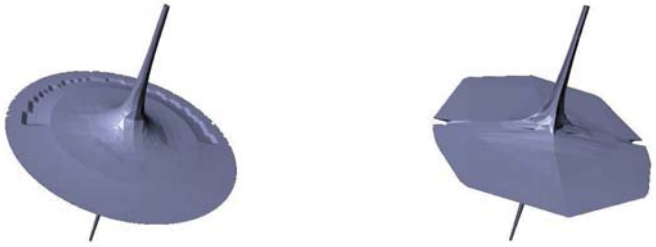


(a) Cxyze (b) PV (c) Cxyz (d) Rect-2

Figure 4.28: Approximation of Eg2: chair $f(x, y, z) = (x^2 + y^2 + z^2 - 23.75)^2 - 0.8((z-5)^2 - 2x^2)((z+5)^2 - 2y^2) = 0$.

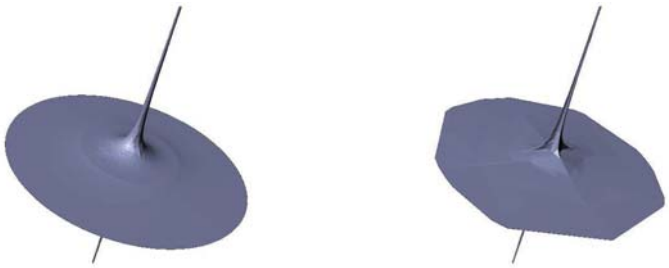
(a) Cxyze (b) PV (c) Cxyz (d) Rect-32

Figure 4.29: Approximation of Eg3: quartic cylinder $f(x, y, z) = y^2x^2 + y^2z^2 + 0.01x^2 + 0.01z^2 - 0.01 = 0$.



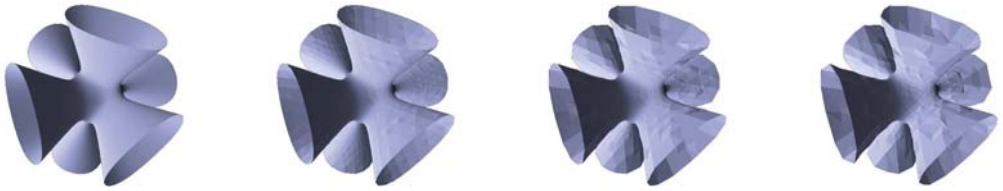
(a) Cxyz (b) Rect-32

Figure 4.30: Approximation of Eg4: quartic cylinder 1 $f(x,y,z) = y^2(x-1)^2 + y^2(z-1)^2 + 0.01(x-1)^2 + 0.01(z-1)^2 - 0.2002 = 0$.



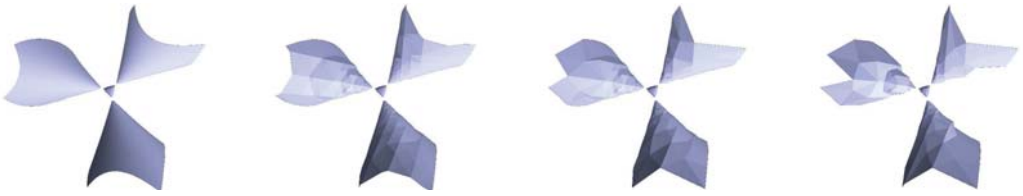
(a) Cxyz (b) Rect-32

Figure 4.31: Approximation of Eg5: quartic cylinder 2 $f(x,y,z) = y^2(x-1)^2 + y^2(z-1)^2 + 0.01(x-1)^2 + 0.01(z-1)^2 - 0.1002 = 0$.



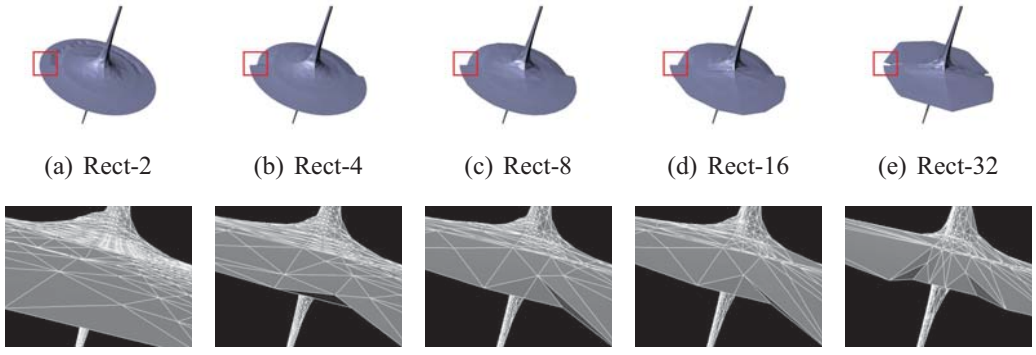
(a) Cxyze (b) PV (c) Cxyz (d) Rect-2

Figure 4.32: Approximation of Eg6: shrek $f(x,y,z) = -x^4 - y^4 - z^4 + 4(x^2 + y^2z^2 + y^2 + z^2x^2 + z^2 + x^2y^2) - 20.7846xyz - 10 = 0$.



(a) Cxyze (b) PV (c) Cxyz (d) Rect-2

Figure 4.33: Approximation of Eg7: tritrumpet $f(x, y, z) = 8z^2 + 6xy^2 - 2x^3 + 3x^2 + 3y^2 - 0.9 = 0$.



(a) Rect-2 (b) Rect-4 (c) Rect-8 (d) Rect-16 (e) Rect-32

(a) Rect-2 (b) Rect-4 (c) Rect-8 (d) Rect-16 (e) Rect-32

Figure 4.34: First row(a)-(e): Approximations of a quartic cylinder 1 $f(x, y, z) = y^2(x-1)^2 + y^2(z-1)^2 + 0.01(x-1)^2 + 0.01(z-1)^2 - 0.2002 = 0$ using Rect-n $(n = 2, 4, 8, 16, 32)$. Second row(a)-(e): Topology preservation in the squared area of the approximations.