



HOKKAIDO UNIVERSITY

Title	Dimension-Driven Deformation and Adaptation of Finite Element Meshes for Efficient Computer Aided Engineering
Author(s)	前濱, 宏樹
Degree Grantor	北海道大学
Degree Name	博士(情報科学)
Dissertation Number	甲第12654号
Issue Date	2017-03-23
DOI	https://doi.org/10.14943/doctoral.k12654
Doc URL	https://hdl.handle.net/2115/65927
Type	doctoral thesis
File Information	Hiroki_Maehama.pdf



**Dimension-Driven Deformation and
Adaptation of Finite Element Meshes for
Efficient Computer Aided Engineering**

Hiroki Maehama

Graduate School of Information Science Technology

Hokkaido University

A thesis submitted for the degree of Doctor of Philosophy

Feb. 15th, 2017

Abstract

Currently, digital engineering becomes widely used for efficient product design. Especially, Computer Aided Design (CAD) is used for cost reduction of product shape design such as automobiles and air planes. In addition, Computer Aided Engineering (CAE) becomes absolutely imperative because it can reduce costs of building prototypes and development work periods of products. In CAD and CAE systems, “solid models” and “mesh models” are used for representing product shapes. Solid models enable us to define volumetric shapes of products and edit them easily. In addition, they can represent curved surfaces easily and precisely, and they can be easily transformed to drawing data. Therefore, solid models are widely used in CAD systems. On the other hand, mesh models generated from the solid models are often used in Finite Element Analysis (FEA) which is needed to evaluate performance of products in CAE. In FEA, accuracy and calculation time of analyses are strongly affected by several factors of mesh models, and they have to be set appropriately according to the purpose of the analysis.

In product shape design and analysis process using CAD/CAE, geometric operations for the products, e.g. changing dimension of the parts and object’s position and orientation in the assembly, and evaluation of product shapes by FEA are repeated until a product shape with the desired performance is obtained. In current process, geometric operations are performed using solid models and FEA is performed using the mesh models. In general, in order to obtain the mesh models, meshing is applied to solid models. However, for complex shapes, meshing process is time consuming tasks and mesh models generated by meshing may need to be modified manually in order to satisfy required quality. Moreover, in current process, because meshing is also repeated, it means that current process is inefficient. On the other hand, direct geometric operations for mesh models while keeping mesh properties can reduce the frequency of meshing. Therefore, methods for direct geometric operations of mesh models are required in order to realize an efficient product shape design process.

The purpose of this thesis is to develop certain geometric operations of finite element mesh models for efficient CAE process. To achieve this goal, a dimension-driven tetrahedral mesh deformation method is first proposed for efficient parameter survey. Second, a mesh adaptation method of tetrahedral meshes is proposed for efficient analysis of assembly models. Finally, the applicability of these two proposed methods to hexahedral mesh editing by a conversion method between tetrahedral meshes and hexahedral meshes [Meshkat 2000] is described.

This thesis mainly includes the following topics:

- (1) *Dimension-driven tetrahedral mesh deformation for parameter survey*: dimension-driven

mesh deformation is effective for finding optimal shape parameters because it can directly change the parameters of form features of product meshes. In this thesis, a new dimension-driven deformation method for tetrahedral meshes is proposed. The method consists of a mesh segmentation, dimension-driven shape deformation, and quality improvement. First, a surface segmentation method for tetrahedral meshes is proposed in order to extract dimensions of the input tetrahedral mesh. In the segmentation method, planar, cylindrical, conical, spherical, and torus surfaces with G^0 or G^1 continuities are sequentially extracted. For extracting each surface segment, region-growing based on principal directions, normal vectors, and surface fittings are performed. Secondly, a dimension-driven shape deformation method for tetrahedral meshes is proposed. In this method, vertices of tetrahedral meshes are moved using a space embedding method and surface information obtained by the segmentation. The proposed deformation method enables us to change several feature parameters of mesh models such as not only height of a boss and radius of a cylindrical hole, but also radius of a fillet, angle of a chamfer, and so on, which cannot be handled by existing methods [Takano 2010, Onodera 2008, Xian 2009]. Finally, we propose a quality improvement method based on Optimal Delaunay Triangulation (ODT) [Chen 2011], which improves element shape qualities, mesh densities, and shape approximation accuracy from the boundary to the inside of the tetrahedral mesh. Through some experiments, it is shown that the dimension-driven tetrahedral mesh deformation method enables us to change parameters of the form features of tetrahedral meshes, such as the fillet radius and chamfer angle, while preserving mesh qualities such as the mesh density, shape approximation accuracy, and element shape quality.

- (2) *Tetrahedral mesh adaptation for efficient finite element analysis of assembly models*: mesh adaptation can generate a conformal mesh from mesh models of multiple objects and space surrounding them (called object meshes and space meshes respectively in this thesis). It can provide the conformal mesh of each object pose by modifying the mesh connectivity and vertex positions on the contacting surfaces between the object meshes or the boundary surfaces of the object meshes and space meshes. Therefore, it is effective for some analysis of assembly models with movable parts such as structural analysis of assembly models and electro-magnetic field analysis of motors. However, the existing methods are inefficient because the mesh topology and geometry are globally adapted even if the differences in poses of the objects in motion are very small. In addition, the existing methods do not deal with contacts of the object meshes. In this thesis, a new efficient tetrahedral mesh adaptation method for moving objects with contact. In the proposed adaptation method, for efficient mesh adaptation, the mesh adaptation process is applied to only a set of space mesh elements around the moving object (deformed region). In addition, to keep mesh conformity on the contact regions between object meshes, the topology and geometry of surface triangular meshes of contacted object meshes are adapted by vertex repositioning and local topological operations. Moreover, in order to obtain high quality meshes, element

shape qualities of the deformed region and the contact regions are improved by a quality improvement method based on ODT smoothing with local topological operations. By the tetrahedral mesh adaptation method, even if the object meshes contact with each other, the conformal tetrahedral mesh of each motion step with moderate element shape qualities can be generated while avoiding drastic increase of elements on the contact region in at most 5s for a conformal tetrahedral mesh including about 160k tetrahedra in an experiment.

- (3) *Application of Deformation and Adaptation method to Hexahedral Meshes based on Tet-Hex Conversion*: hexahedral meshes are preferred over tetrahedral meshes because accurate and efficient FEA can be performed with the small number of elements by using hexahedral meshes. However, it is difficult to generate high quality hexahedral meshes for complex product shapes automatically. In addition, editing of hexahedral meshes such as deformation and quality improvement is difficult because local topological operations of hexahedral meshes are not established. Therefore, some conversion methods between tetrahedral meshes and hexahedral meshes have been proposed. In this thesis, in order to realize certain geometric operations of hexahedral meshes, the mesh deformation and adaptation methods developed in this research are combined with a conversion method between tetrahedral meshes and hexahedral meshes [Meshkat 2000], and the applicability of the proposed methods to hexahedral mesh editing are investigated. Although it is difficult to obtain all-hex meshes from deformed or adapted tetrahedral meshes by only performing the conversion method between tetrahedral meshes and hexahedral meshes, hex-dominant meshes after deformation and object motion can be generated by the combination of the proposed methods with the conversion method.

Contents

Dimension-Driven Deformation and Adaptation of Finite Element Meshes for Efficient Computer Aided Engineering

Abstract	i
Contents	v
List of Figures	xi
List of Tables	xvii
Chapter 1 Introduction	1
1.1 Background	1
1.2 Geometric operations of mesh models for product shape design	3
1.2.1 Dimension-driven mesh deformation	3
1.2.2 Mesh adaptation	4
1.3 Tetrahedral mesh, all-hex mesh, and hex-dominant mesh	6
1.4 Purpose	8
1.5 Organization of this thesis	8
Chapter 2 Dimension-Driven Tetrahedral Mesh Deformation for Parameter Survey	11
2.1 Functional requirements and organization of this chapter	11
2.2 Related work	12
2.2.1 Existing mesh segmentation methods	12
2.2.1.1 Mesh segmentation methods	12
2.2.1.2 Feature line extraction methods	13
2.2.1.3 Summary of existing methods	15
2.2.2 Existing mesh deformation methods	16
2.2.2.1 Dimension-driven mesh deformation methods	16
2.2.2.2 Other mesh deformation methods	18

2.2.2.3 Summary of existing methods	19
2.2.3 Existing mesh quality improvement methods	20
2.2.3.1 Mesh quality improvement methods	20
2.2.3.2 Degenerated or inverted element removal methods	22
2.2.3.3 Summary of existing methods	25
2.3 Overview of proposed method	28
2.4 Mesh segmentation based on normal tensor and region growing	29
2.4.1 Overview of the mesh segmentation	29
2.4.2 Sharp edge extraction and calculation of principal curvature and direction	30
2.4.3 Planar and quadric surface region extraction	32
2.4.3.1 Overview of planar and quadric surface region extraction	32
2.4.3.2 Initial seed region extraction	32
2.4.3.3 Region integration	34
2.4.4 Torus surface region extraction	37
2.4.5 Calculation of surface parameters	39
2.4.6 Boundary edge recognition	42
2.5 Dimension-driven shape deformation based on surface information and space embedding	43
2.5.1 Overview of dimension-driven shape deformation	43
2.5.1.1 Assumptions	43
2.5.1.2 Deformation types and user inputs	44
2.5.1.3 Algorithm	46
2.5.2 Surface region classification	47
2.5.3 Deformable region extraction	48
2.5.3.1 Overview of deformable region extraction	48
2.5.3.2 Deformable region extraction for change of the radius of the fillet	49
2.5.3.3 Deformable region extraction for change of the position	

of the local object	50
2.5.3.4 Deformable region extraction for the other changes	51
2.5.4 Vertex repositioning	52
2.5.4.1 Overview of vertex repositioning	52
2.5.4.2 Space embedding method	52
2.5.4.3 Parameterization of vertices	53
2.5.4.4 Mean Value Coordinates	54
2.5.4.5 Vertex repositioning according to vertex types	57
2.6 Quality improvement by Phased ODT smoothing	58
2.6.1 Overview of quality improvement	58
2.6.1.1 Optimal Delaunay Triangulation (ODT) smoothing	58
2.6.1.2 Overview of proposed method	62
2.6.1.3 Local topological operations	63
2.6.2 Degenerated or inverted element removal	65
2.6.3 Mesh density control	67
2.6.4 Shape approximation accuracy control	70
2.6.5 Phased ODT smoothing	70
2.6.5.1 Overview of Phased ODT smoothing	70
2.6.5.2 Boundary edge improvement	71
2.6.5.3 Surface triangle improvement	72
2.6.5.4 Tetrahedron improvement	74
2.7 Experimental results and evaluation	75
2.7.1 Overview of experiments	75
2.7.2 Mesh segmentation	75
2.7.3 Dimension-driven shape deformation	80
2.7.4 Mesh quality improvement	85
2.7.5 The evaluation of the generation time of tetrahedral meshes	94

2.8	Summary	96
Chapter 3 Tetrahedral Mesh Adaptation for Efficient Finite Element Analysis of Assembly Models		99
3.1	Functional requirements and organization of this chapter	99
3.2	Related work	100
3.2.1	Mesh adaptation method for object motion	100
3.2.2	Mesh merging method	101
3.2.3	Summary of related work	103
3.3	Overview of proposed method	104
3.4	Local region extraction using surface information and distance field	106
3.4.1	Grid generation	106
3.4.2	Contact region extraction	107
3.4.3	Deformed region extraction	109
3.5	Mesh adaptation using two local regions	110
3.5.1	Overview of the mesh adaptation	110
3.5.2	Space mesh element removal	110
3.5.3	Rigid transformation and deformed region adaptation	111
3.5.4	Contact region adaptation	112
3.5.4.1	Overview	112
3.5.4.2	Representation of boundary lines of the contact region	113
3.5.4.3	Corresponding vertex positions	113
3.5.4.4	Corresponding number of vertices	116
3.5.4.5	Corresponding connectivity	119
3.5.4.6	Quality improvement of the contact region	119
3.5.5	Tetrahedrization	120
3.6	Mesh quality improvement based on ODT	120
3.7	Experimental results and evaluation	121

3.7.1 Overview of experiments	121
3.7.2 Mesh adaptation for single moving object	122
3.7.3 Mesh adaptation for object motion with contact	126
3.7.4 Conformal mesh generation from a set of mesh models	127
3.8 Summary.....	129
Chapter 4 Application of Deformation and Adaptation methods to Hexahedral Meshes based on Tet-Hex Conversion	133
4.1 Requirements and organization of this chapter.....	133
4.2 Related work of indirect hexahedral mesh generation	134
4.3 Overview of the validation of the proposed method.....	138
4.4 Decomposition of hexahedra without inserting any vertices	139
4.5 Conversion between tetrahedral meshes and hexahedral meshes based on graph representation	142
4.5.1 Overview of the conversion between tetrahedral meshes and hexahedral meshes	142
4.5.2 Extended Region-Face-graph (RF-graph) representation	143
4.5.3 Local search using extended RF-graph	144
4.5.4 Greedy conversion between tetrahedral meshes and hexahedral meshes	147
4.6 Experimental results and evaluation	148
4.6.1 Overview of experiments	148
4.6.2 Conversions of deformed tetrahedral meshes	149
4.6.3 Conversions of tetrahedral meshes adapted to object motion	153
4.7 Summary.....	156
Chapter 5 Conclusions and Future Work	159
5.1 Conclusions	159
5.2 Future work	161
References	165
Acknowledgement	173

List of Figures

Fig. 1.1	Solid model and mesh model.....	1
Fig. 1.2	Geometric operations.....	2
Fig. 1.3	The current product shape design and analysis process.....	3
Fig. 1.4	The product shape design and analysis process based on geometric operations using mesh models.....	3
Fig. 1.5	Parameter survey process and making it efficient by dimension-driven mesh deformation.....	4
Fig. 1.6	Conformal mesh and non-conformal mesh.....	5
Fig. 1.7	The evaluation of assembly models with movable parts (a) by re-meshing (traditional process) and (b) by mesh adaptation (proposed process)	5
Fig. 1.8	The overview of simulation methods of a moving object in a finite space.....	6
Fig. 1.9	Organization of this thesis.....	10
Fig. 2.1	Functional requirements of the dimension-driven mesh deformation	12
Fig. 2.2	The repositioning of vertices belonging to two or more surfaces in [Onodera 2008]	18
Fig. 2.3	Importance of vertex insertion and vertex removal	22
Fig. 2.4	A fatal change of shape by NODT	27
Fig. 2.5	The overview of the proposed dimension-driven tetrahedral mesh deformation method	29
Fig. 2.6	The overview of the proposed segmentation method	30
Fig. 2.7	The sequential extraction of planar and quadric surface regions	32
Fig. 2.8	Feature of each surface	34
Fig. 2.9	Cylindrical surface fitting	35
Fig. 2.10	Conical surface fitting based on RANSAC.....	36
Fig. 2.11	Spherical surface fitting based on RANSAC	36

Fig. 2.12	Two neighboring torus surface regions with different signs of Gaussian curvatures	37
Fig. 2.13	False recognition surface regions.....	38
Fig. 2.14	Torus surface fitting based on RANSAC	38
Fig. 2.15	The pseudo code of the surface fitting based on LM method	40
Fig. 2.16	Control variables of each surface in LM method.....	40
Fig. 2.17	Deformation preserving the consistency of the curved surface	43
Fig. 2.18	The process of changing a dimension	46
Fig. 2.19	The overview of the dimension-driven shape deformation of tetrahedral meshes...	47
Fig. 2.20	Surface region classification for the deformation type (1)-(6).....	48
Fig. 2.21	Surface region classification for the deformation type (7).....	48
Fig. 2.22	Performing the deformation and quality improvement only on the <i>deformable region</i>	49
Fig. 2.23	Deformable region extraction for the deformation type (3) the radius of the fillet.....	50
Fig. 2.24	Deformable region extraction for the deformation type (7) the position of the local object.....	51
Fig. 2.25	Deformable region extraction for the deformation type (1) the depth of the hole.....	52
Fig. 2.26	The overview of the vertex repositioning	52
Fig. 2.27	The overview of the space embedding method.....	53
Fig. 2.28	Handle vertices and parameters of each vertex.....	54
Fig. 2.29	The angle α_j	55
Fig. 2.30	Three lengths of arcs and three angles	56
Fig. 2.31	Feature vertex movement.....	58
Fig. 2.32	Minimization of the error function by geometrical and topological operations.....	59
Fig. 2.33	Vertex repositioning near the boundary in ODT smoothing.....	61
Fig. 2.34	The overview of the proposed quality improvement method	63

Fig. 2.35	Local topological operations for vertex insertion	64
Fig. 2.36	Local topological operations for vertex removal	64
Fig. 2.37	Classification of edge for edge collapse preserving the shape of tetrahedral meshes	65
Fig. 2.38	Degenerated tetrahedra and removal operations	67
Fig. 2.39	Half edge collapse and inverted element removal	67
Fig. 2.40	The overview of the mesh density control	68
Fig. 2.41	The definition of geometric error and shape approximation accuracy control	70
Fig. 2.42	ODT smoothing on the circle or circular arc	71
Fig. 2.43	2D parameter space of each surface region for ODT smoothing	72
Fig. 2.44	The overview of ODT smoothing for curved surface regions.....	73
Fig. 2.45	Edge flipping in Phased ODT smoothing	74
Fig. 2.46	Flipping operation for the tetrahedron improvement of Phased ODT smoothing..	75
Fig. 2.47	The segmentation result of the first test model	77
Fig. 2.48	The segmentation result of the second test model	79
Fig. 2.49	Segmentation results of mechanical parts.....	80
Fig. 2.50	The various deformation results of a tetrahedral mesh using proposed method	81
Fig. 2.51	Changing a distance between two parallel planes of a connecting rod	82
Fig. 2.52	Changing the position of a cylindrical boss on a plane of a cover	84
Fig. 2.53	Result of the improvement of the deformed mesh (changing DP).....	86
Fig. 2.54	Result of the improvement of the deformed mesh (changing RC)	88
Fig. 2.55	Result of the changing RC by two steps using the proposed method	89
Fig. 2.56	Results of the quality improvement (mechanical parts).....	90
Fig. 2.57	The input mesh and resultant mesh of a cantilever beam	92
Fig. 2.58	The result of FEA using resultant mesh.....	92
Fig. 2.59	The input mesh and parameters of the plate with a cylindrical hole for FEA.....	93
Fig. 2.60	The enlarged views of three deformed meshes	93

Fig. 2.61	Distributions of tensile stresses on the minimum section	94
Fig. 2.62	The overview of the evaluation of the dimension-driven mesh deformation.....	95
Fig. 2.63	The comparison of mesh generation times.....	96
Fig. 3.1	The overview of the proposed tetrahedral mesh adaptation method.....	105
Fig. 3.2	Two regular grid and generation of the distance field grid	107
Fig. 3.3	The need of surface information for the contact region.....	108
Fig. 3.4	The overview of the contact region extraction.....	109
Fig. 3.5	The overview of the deformed region extraction.....	110
Fig. 3.6	The overview of the mesh adaptation process	111
Fig. 3.7	Space mesh element removal.....	111
Fig 3.8	Rigid transformation and deformed region adaptation	112
Fig. 3.9	The overview of contact region adaptation.....	113
Fig 3.10	Vertex insertion on the intersection points of boundary line segments.....	113
Fig. 3.11	Corresponding vertex positions (two meshes contact with each other on the bold line).....	114
Fig. 3.12	Making corresponding vertex pairs (two meshes contact with each other on the bold line).....	115
Fig 3.13	The movable check of vertices.....	116
Fig. 3.14	The overview of corresponding number of vertices (two meshes contact with each other on the bold line).....	117
Fig. 3.15	Virtual edge collapse for vertex removal	117
Fig. 3.16	Geometric error.....	118
Fig. 3.17	Four virtual split operations for vertex insertion.....	118
Fig 3.18	The corresponding connectivity of surface triangular meshes.....	119
Fig 3.19	The two patterns of regular grids of the target mesh density field.....	120
Fig. 3.20	Translation of a rotating rectangle	124
Fig. 3.21	The relation of the distance threshold to element shape qualities and time	125
Fig. 3.22	Translation of a cylinder toward a half tube	127

Fig. 3.23	The simple mechanical parts without space mesh	129
Fig. 4.1	The combination the proposed methods with a Tet-Hex conversion	134
Fig. 4.2	A famous conversion between a tetrahedron and four hexahedra.....	135
Fig. 4.3	Another type of conversions between tetrahedra and a hexahedra	135
Fig. 4.4	The overview of the validation	139
Fig. 4.5	Six decomposition types for a hexahedron	141
Fig. 4.6	Possibility of connection between decompositions of hexahedra.....	142
Fig. 4.7	The overview of the simplified indirect hexahedral mesh generation method	143
Fig. 4.8	Extended RF-graph of two tetrahedra composing a pyramid	143
Fig. 4.9	Extended RF-graphs of each decomposition type of a hexahedron	144
Fig. 4.10	Six search trees and sequences of two operations corresponding to the six extended RF-graphs	145
Fig. 4.11	Extended RF-graphs of prisms.....	147
Fig. 4.12	The order of vertices for the calculation of the scale Jacobian.....	148
Fig. 4.13	Input hexahedral mesh and tetrahedral mesh generated by the decomposition	149
Fig. 4.14	The deformed tetrahedral mesh by the proposed method	150
Fig. 4.15	Results of the Tet-Hex conversion of the deformed mesh	151
Fig. 4.16	The change of the volume ratio of hexahedra and the number of tetrahedra according to the angle threshold for QUAD operation.....	152
Fig. 4.17	Input conformal hexahedral mesh and tetrahedral mesh generated by the decomposition.....	153
Fig. 4.18	The conversion results from some conformal tetrahedral meshes generated by the proposed tetrahedral mesh adaptation method.....	154
Fig. 4.19	Resultant hexahedra with the minimum scaled Jacobian lower than a threshold .	155

List of Tables

Table 1.1	Comparison of element types of mesh models	7
Table 2.1	Existing mesh segmentation methods	16
Table 2.2	Existing feature line extraction methods	16
Table 2.3	Existing dimension-driven mesh deformation methods	20
Table 2.4	Existing other deformation method	20
Table 2.5	Existing method of degenerated element removal.....	27
Table 2.6	Existing method of inverted element removal.....	27
Table 2.7	Deformation type and user inputs.....	45
Table 2.8	The fillet radii in the first test model	77
Table 2.9	Calculation time of each process for the first test model (#Triangles: 20,032).....	78
Table 2.10	Calculation time of each process for the second test model (#Triangles: 10,122)	79
Table 2.11	The number of triangles and the total calculation time for each model.....	80
Table 2.12	Calculation time of changing a distance between two parallel planes	83
Table 2.13	Calculation time of changing the position of the cylindrical boss.....	85
Table 2.14	Mesh quality of each tetrahedral mesh (changing DP).....	87
Table 2.15	Mesh quality of each tetrahedral mesh (changing RC)	88
Table 2.16	Results of the quality improvement (mechanical parts)	91
Table 2.17	The data of the experiments for the evaluation of quality improvement.....	94
Table 2.18	Results of the quality improvement (mechanical parts)	96
Table 3.1	Existing mesh adaptation methods for object motion.....	104
Table 3.2	Existing mesh merging methods	104
Table 3.3	The data of each experiment.....	122
Table 3.4	The results obtained by different distance thresholds.....	125

Table 4.1	Summary of indirect hexahedral mesh generation methods.....	137
Table 4.2	Element shape qualities of the deformed tetrahedral mesh	150
Table 4.3	The volume and number of each polyhedron after the Tet-Hex conversion.....	152
Table 4.4	The minimum and average values of the minimum scaled Jacobian.....	152
Table 4.5	The conversion results of tetrahedral meshes adapted to an object motion.....	155

Chapter 1 Introduction

1.1 Background

Currently, for efficient product design and developing high quality products, digital engineering becomes widely used. Especially, Computer Aided Design (CAD) can be used for geometric modeling of products, drafting works, and searching for past drawing data or design standard. Therefore, CAD is needed for cost reduction of product shape design such as auto mobiles and air planes. In addition, Computer Aided Engineering (CAE) which simulates physical phenomena and analyzes performance of products on computers instead of prototype test becomes absolutely imperative because costs of building prototypes can be reduced by CAE. In CAD and CAE systems, “solid model” and “mesh model” (Fig. 1.1) are used for representation of product shapes. Boundary Representations (B-reps) is a famous shape representation as solid model (Fig. 1.1(a)). In B-reps, product shape is represented as a collection of connected surface topological elements. It enables us to define volumetric shapes of products and edit them easily. In addition, it can represent curved surfaces easily and precisely, and it can be easily transformed to drawing data. Therefore, it is widely used in CAD systems. On the other hand, the mesh model (Fig. 1.1(b)) is a shape representation where a product shape is represented by a set of polytopes such as tetrahedra and hexahedra. Mesh models are used in Finite Element Analysis (FEA) which is needed to evaluate performance of products in CAE. In FEA, accuracy and calculation time of analyses are strongly affected by following four factors of mesh models, and they have to be set appropriately according to the purpose of the analysis.

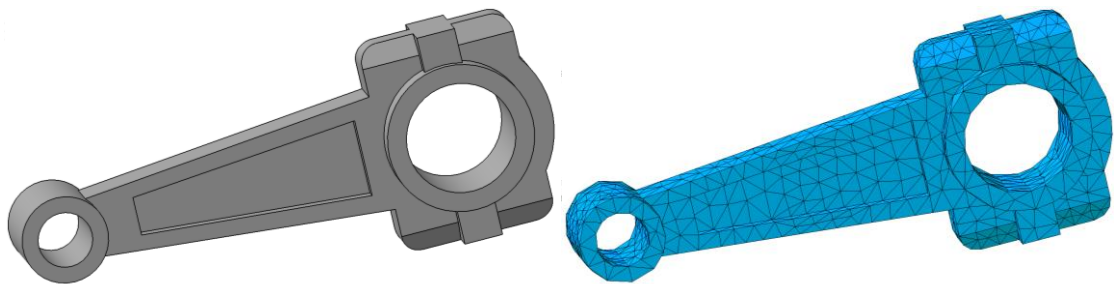


Fig. 1.1 Solid model and mesh model

- *Target of meshing*: there are surface meshes and volumetric meshes. Surface meshes represent only surfaces of objects. On the other hand, volumetric meshes which represent both of surfaces and insides of objects. In addition, mesh models of space surrounding objects are also needed in some analyses such as CFD and electro-magnetic analyses.
- *Conformity (topological condition)*: the mesh model is whether a conformal mesh or a non-conformal mesh.
- *Element type*: the mesh model is constructed by whether triangles, quadrangles, tetrahedra, or hexahedra.
- *Quality*: the mesh model has several properties such as element shape quality, mesh density, and shape approximation accuracy.

In product shape design and analysis process using CAD/CAE, geometric operations (Fig. 1.2) such as (a) changing dimension, (b) object motion, and (c) insertion/removal of form features and evaluation of product shapes by FEA are repeated until a product shape which has the desired performance is obtained. In current process, as shown in Fig 1.3, geometric operations are performed using solid models and FEA is performed using the mesh models. In this process, in order to obtain the mesh models, meshing is applied to solid models. However, for complex shapes, meshing process becomes time consuming and mesh models generated by meshing may need to be modified manually in order to satisfy the required quality. In current process, because meshing is also repeated, and current process has the potential to become more efficient by reducing the frequency of meshing.

On the other hand, as shown in Fig. 1.4, geometric operations of mesh models while keeping such properties enable us to perform an efficient product shape design process because it can reduce the frequency of meshing. Therefore, geometric operations of mesh models for product shape design are required. In the next section, two representative methods for geometric operations of mesh models, dimension-driven mesh deformation and mesh adaptation, are described.

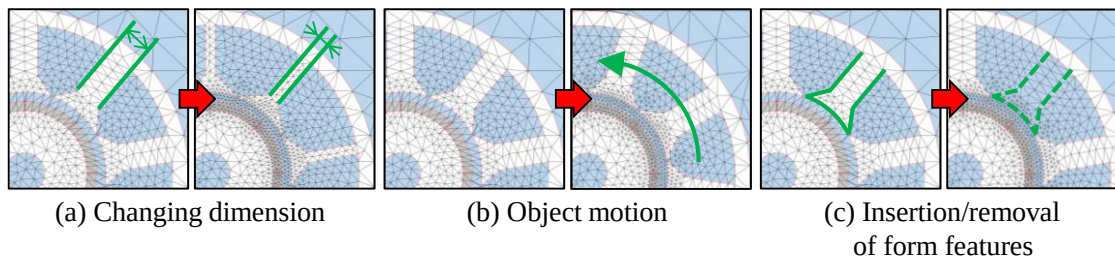


Fig. 1.2 Geometric operations

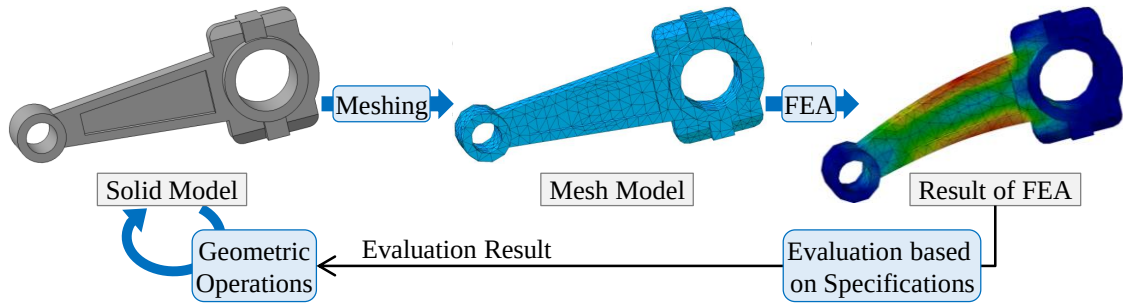


Fig. 1.3 The current product shape design and analysis process

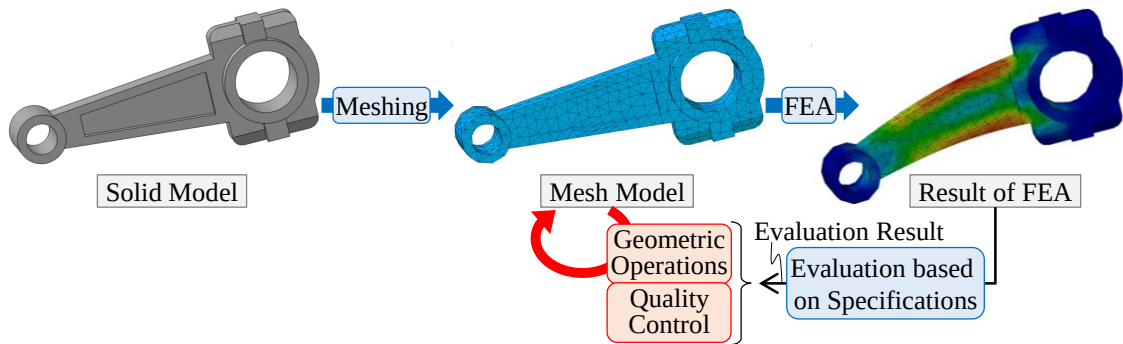


Fig. 1.4 The product shape design and analysis process based on geometric operations using mesh models

1.2 Geometric operations of mesh models for product shape design

1.2.1 Dimension-driven mesh deformation

In order to find optimal dimensions of product shapes, as shown in Fig. 1.5(a), parameter survey using CAD and CAE is performed. In the parameter survey process, product shapes are first modified by using solid models in the CAD system. Then, meshing is applied to the solid models to generate mesh models, and the product shapes are evaluated by FEA using the mesh models. The solid modification, meshing, and FEA are repeated until an optimal product shape is obtained. However, meshing is unstable and a time-consuming task. On the other hand, mesh deformations have the potential to achieve an efficient parameter survey process because it can change the shape of mesh models without meshing.

Currently, most of the 3D CAD systems employ feature-based parametric modeling. In other words, the parameters of the form features of the solid model often correspond to the dimensions of

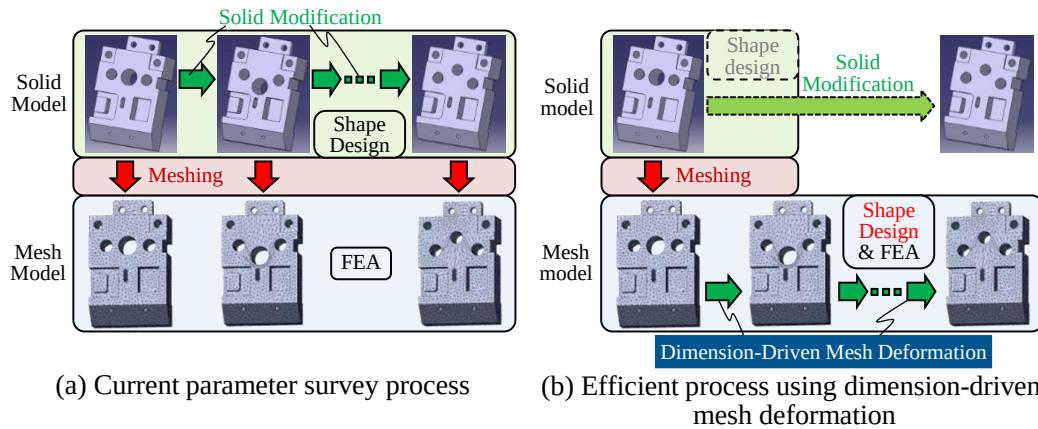


Fig. 1.5 Parameter survey process and making it efficient by dimension-driven mesh deformation

the product shape. Therefore, as shown in Fig. 1.5(b), dimension-driven mesh deformation which can directly change the parameters of the form features of mesh models is especially effective for the efficient parameter survey process because they enable us to modify the product shape without using the CAD systems and reduce the frequency of meshing in the parameter survey.

1.2.2 Mesh adaptation

In the evaluation of assembly models with movable parts (e.g. motor), object motion of movable parts is divided into small motion steps, and FEA is performed at each motion step. It means that many mesh models are needed. In some analysis of assembly models with movable parts such as electro-magnetic field analysis of motors and fluid-structure interaction (FSI) analysis, mesh models not only of objects (parts) but also of space (called “object mesh” and “space mesh” in this thesis) are needed. In order to represent multiple parts (objects) and the space surrounding them by mesh models, as shown in Fig. 1.6, there are following two types of mesh models.

- (a) Conformal mesh: the mesh models where the intersection of two connecting elements can be formed by a single face, edge, or vertex.
- (b) Non-conformal mesh: the mesh models which are not satisfy above condition.

As shown in Fig. 1.7(a), if the evaluation of assembly models with movable parts is performed based on re-meshing of conformal meshes, a large number of meshing is required. To solve this problem, as shown in Fig. 1.7(b), extended simulation methods and/or mesh handling techniques such as mesh adaptation which can provide the mesh of each object pose without meshing are expected.

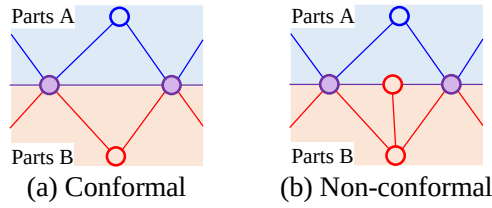


Fig. 1.6 Conformal mesh and non-conformal mesh

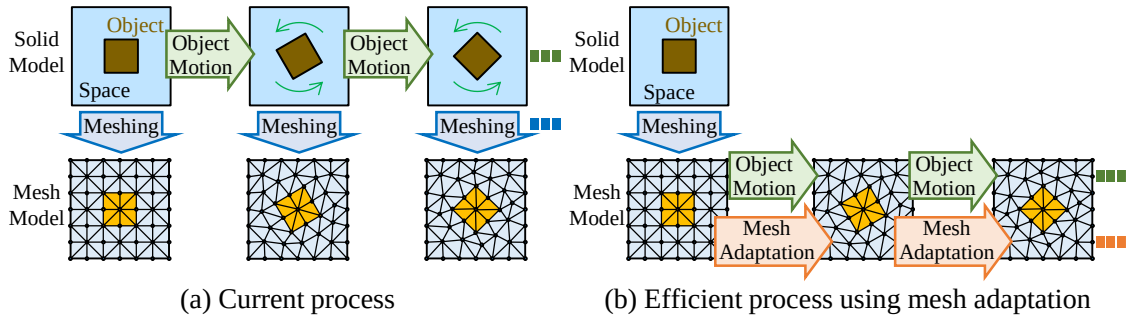


Fig. 1.7 The evaluation of assembly models with movable parts (a) by re-meshing (traditional process) and (b) by mesh adaptation (proposed process)

Figure 1.8 shows the overview of extended simulation methods and mesh handling techniques of a moving object in a finite space. This list is based on [Samaniego 2015].

- Domain decomposition (domain composition): A mesh model of a finite space is attached to the moving object and moving over a fixed space mesh. In order to obtain global solution, the information used in the analysis is exchanged between these two space meshes. Sliding mesh method (Fig. 1.8(a)) is an example of this class. The space mesh attached to the moving object is not connected with the fixed space mesh and does not overlap the fixed space mesh. During the simulation, information between them is transmitted over the interfaces. Similar example is shear-slip mesh update method (Fig. 1.8(b)) where a layer of elements absorbing the shear between two space meshes. Another example is chimera method (Fig. 1.8(c)) where the space mesh attached to the moving object overlap the fixed space mesh and information is transmitted over the overlapping region.
- Embedded boundary (Fig. 1.8(d)): A fixed space mesh which is independent of the moving object is generated. Then, a wet boundary of the moving object is embedded in the mesh. Some sample points (Lagrange points) are putted on the wet boundary and used for interpolation between the moving object and the fixed space mesh.
- Arbitrary Lagrangian-Eulerian (ALE, Fig. 1.8(e)): This class is the method takes

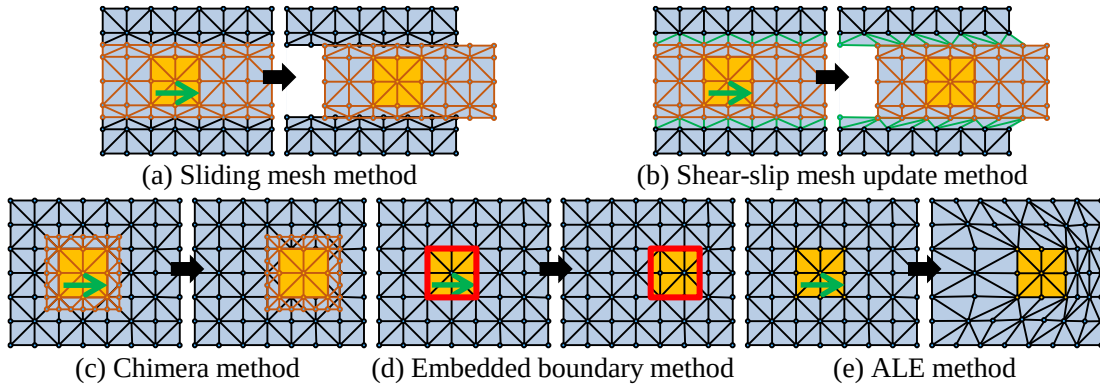


Fig. 1.8 The overview of simulation methods of a moving object in a finite space

advantage of Lagrangian and Eulerian descriptions. A conformal mesh is used and vertices of the space mesh are moved according to the object motion. Re-meshing is required when the elements are too distorted. Mesh adaptation methods are dealt with as this class in this thesis.

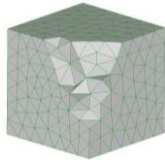
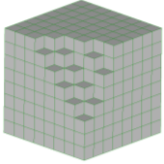
In general, ALE method enable us to perform FEA more accurately and efficiently than domain decomposition methods or embedded boundary methods. However, when the number of re-meshing becomes larger, the accuracy and efficiency of FEA become lower. Therefore, the method which can adapt conformal meshes to the large scale movement of objects with a small number of re-meshing is required. In this thesis, for conformal tetrahedral meshes, a mesh adaptation method including a quality improvement method is proposed.

1.3 Tetrahedral mesh, all-hex mesh, and hex-dominant mesh

3D mesh models are classified into surface mesh or volumetric mesh. In the surface mesh, only the surface of an object is represented by a set of polygons such as triangles or quadrangles. On the other hand, in the volumetric mesh, not only the surface but also the interior volume of an object is represented by a set of polyhedron such as tetrahedra, pentahedra, or hexahedra. In many analyses for a 3D shape such as structure analyses of a mechanical part, volumetric meshes are used. As shown in Table 1.1, volumetric meshes are classified into tetrahedral meshes, all-hex meshes, or hex-dominant meshes (i.e. mixed volume meshes). In this thesis, all-hex meshes and hex-dominant meshes are called “hexahedral meshes” in contrast with tetrahedral meshes.

In a tetrahedral mesh, an object is represented by a set of tetrahedra. Tetrahedral meshes can represent complex shapes easily. In addition, local topological operations such as vertex insertion or removal are established. Therefore, many editing methods such as deformation, adaptation, and quality improvement are proposed, and various automatic tetrahedral mesh generation methods are also proposed.

Table 1.1 Comparison of element types of mesh models

Element type	 Tetrahedral	 All-hex	Hex-dominant (Polyhedral mesh whose major element is hexahedron but other elements such as tetrahedron and prism are included)
Generation	Easy	Difficult	Easier than all-hex
Local topological operations	Established	Not established	Not established
Editing	Easy	Difficult	Difficult
Accuracy of FEA	Low	High	High*

*However this analysis needs special manner.

On the other hand, in the all-hex mesh, an object is decomposed by a set of only hexahedra. In compared with tetrahedral meshes, all-hex meshes enable us to perform more accurate analyses by smaller number of elements. However, local topological operations for all-hex meshes are not established. In addition, automatic generation of all-hex meshes for arbitrary shapes are difficult and not established.

In order to represent complex shapes by a set of hexahedra, in a hex-dominant mesh, all the possible hexahedra are generated inside an object and the remaining voids are filled by other elements such as tetrahedra, prisms, and pyramids. Although hex-dominant meshes enable us to perform accurate analyses, local topological operations are not established and special manners are needed in analyses. Therefore, hex-dominant meshes are not frequently used.

In general, in order to perform FEA accurately and efficiently, each element shape should be a regular polyhedron. When the ratio of distorted elements in a mesh model increases, the accuracy and efficiency of FEA decrease. Moreover, when degenerated elements such as sliver of tetrahedral mesh and inverted elements whose signed volume is a negative value are included in mesh models, numerical calculation in FEA may not be solved. Therefore, it is important to improve element shape qualities of mesh models. However, it is difficult to improve element shape qualities of hexahedral meshes because local topological operations of them are not established.

Therefore, in many application, tetrahedral meshes are used because they can be edited and generated more easily than hexahedral meshes. In this research, dimension-driven deformation and adaptation methods for tetrahedral meshes are first proposed. And then, using a conversion method between tetrahedral meshes and hexahedral meshes [Meshkat 2000], the applicability of the proposed methods to hexahedral meshes are investigated.

In this research, in order to evaluate the element shape qualities of tetrahedral meshes, stretch [Geuzaine 2009] is used. The stretch $Q(\tau)$ of tetrahedron τ is a normalized ratio between the radius of the inscribed sphere and that of the circumscribed sphere of τ , and it is calculated by Eq. (1.1):

$$Q(\tau) = \frac{6\sqrt{6}V(\tau)}{\left(\max_{e \in E_\tau} L(e)\right) A^{tet}(\tau)}, \quad (1.1)$$

where $V(\tau)$ is the signed volume of τ , $A^{tet}(\tau)$ the surface area of τ , E_τ a set of edges of τ , and $L(e)$ is the length of edge e . Stretch becomes 1 for regular tetrahedron, 0 for degenerated tetrahedron, and a negative value for inverted tetrahedron.

1.4 Purpose

The purpose of this research is to propose dimension-driven deformation and adaptation methods of finite element meshes for efficient CAE process. Concretely speaking, at first, the following two methods for tetrahedral meshes are proposed and evaluated.

- A dimension-driven tetrahedral mesh deformation method which can change several types of dimensions of product shapes like in CAD systems (Chapter 2).
- A mesh adaptation method for object motion with contact using conformal tetrahedral meshes consists of not only object meshes but also a space mesh around objects (Chapter 3).

After that, conversion methods between tetrahedral meshes and hexahedral meshes are described and the applicability of the proposed methods to hexahedral meshes is shown by combining them and the proposed methods (Chapter 4).

1.5 Organization of this thesis

The rest of this thesis is organized as follows, and the organization of this thesis is shown in Fig. 1.9.

In Chapter 2, a dimension-driven tetrahedral mesh deformation method for parameter survey is proposed. At first, three functional requirements: mesh segmentation, mesh deformation, and mesh quality improvement are described. Then, a dimension-driven tetrahedral mesh deformation method which has following three functions corresponding to these requirements is proposed.

- *Mesh segmentation based on normal tensor and region growing*: in this function, a planar, cylindrical, conical, spherical, or torus surface regions which are connected with others with G^0 or G^1 continuities are sequentially extracted. In addition, surface parameters of each surface region are accurately calculated by a non-linear least square surface fitting [Shakariji 1998].
- *Dimension-driven shape deformation based on surface information and space embedding*: this function enables us to change various feature parameters of tetrahedral meshes such as radius of the fillet, angle of the chamfer, and so on.
- *Quality improvement based on Phased ODT smoothing*: this function improves element

shape equalities by Optimal Delaunay Triangulation (ODT) smoothing [Chen 2011] from the boundary to the inside of tetrahedral meshes in a step-by-step manner. In addition, it can recover original mesh properties such as mesh density and shape approximation accuracy.

In Chapter 3, a tetrahedral mesh adaptation method for efficient finite element analysis of assembly models is proposed. The method can adapt tetrahedral meshes to object motion with contact. At first, functional requirements are described. Then a new tetrahedral mesh adaptation method for object motion which satisfies these requirements is proposed. The proposed mesh adaptation method has following four features.

- *Efficient adaptation*: for efficient mesh adaptation, the mesh adaptation process is applied to only a set of local mesh elements of space around the moving object.
- *Accurate contact detection*: surface parameters of mesh models are used so that elements in contact regions between objects can be detected accurately.
- *Generating high quality meshes*: in order to obtain a high quality mesh in each motion step, shape qualities of tetrahedral elements are improved and edge length is controlled by using a method based on ODT smoothing with local topological operations.
- *Keeping mesh conformity*: in order to keep mesh conformity on the contact regions between object meshes, the topology and geometry of surface triangular meshes of contacted object meshes are adapted by vertex repositioning and local topological operations.

In Chapter 4, requirements on hexahedral mesh generation are first described. Then, as indirect hexahedral mesh generation methods, conversion methods between tetrahedral meshes and hexahedral meshes are introduced. After that, by combining a conversion method [Meshkat 2000] and proposed mesh deformation method, the applicability of proposed methods to hexahedral meshes is shown and limitations are discussed.

In Chapter 5, conclusions and future work of this research are described.

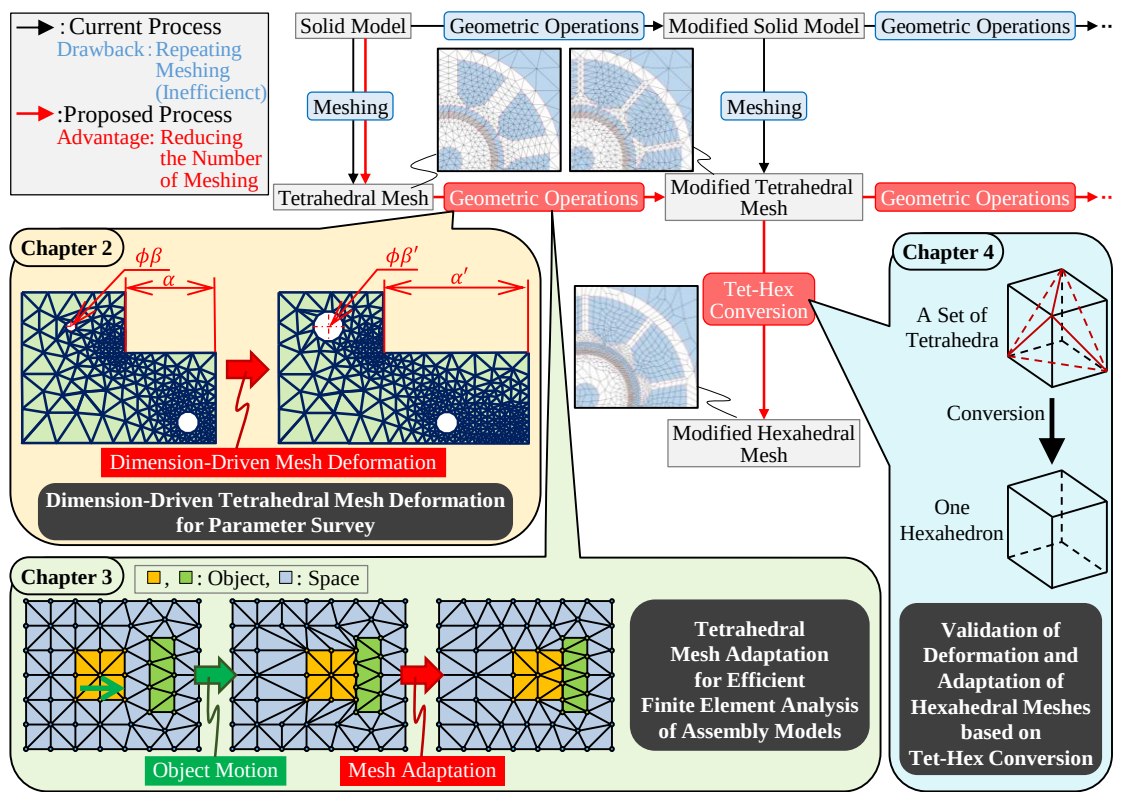


Fig. 1.9 Organization of this thesis

Chapter 2 Dimension-Driven Tetrahedral Mesh Deformation for Parameter Survey

2.1 Functional requirements and organization of this chapter

As shown in Fig 2.1, the following three functions are required in order to perform dimension-driven mesh deformations.

- (A) *Mesh Segmentation*: dimension-driven mesh deformations need to extract several kinds of dimensions of the product shape in order to change them. When a mesh model without its original solid model (e.g. past mesh models stored in a mesh database) is used, dimensions of the product shape should be estimated from the mesh model. Therefore, accurate mesh segmentation and extraction of dimensions from input mesh models are needed.
- (B) *Dimension-driven shape deformation*: It is important to deform surfaces of mesh models so that they will satisfy given dimensions, and vertices of the mesh models have to be moved according to the target dimension. In addition, it is desired to change rich types of dimensions of the mesh model in the deformation.
- (C) *Quality Improvement*: deformed meshes often include many distorted elements and may lose original mesh properties such as mesh density and shape approximation accuracy. The element shape qualities of deformed meshes have to be improved because distorted elements cause low accuracy and inefficiency of FEA. In addition, deformed meshes should keep the original mesh properties, because changes of them influence the accuracy of FEA and desired results may not be provided.

In this chapter, at first, existing methods of each function are introduced and their problems are discussed. Then, a new dimension-driven deformation method for tetrahedral meshes is proposed. The proposed method consists of mesh segmentation, mesh deformation, and quality improvement. In section 2.3, the overview of proposed method is described, and then, a mesh segmentation method based on normal tensor and region growing is proposed in section 2.4. In the mesh segmentation method, planar, cylindrical, conical, spherical, and torus surface regions and their surface parameters (surface information) that are connected with other regions with G^0 or G^1 continuities are sequentially extracted. After that, in section 2.5, a dimension-driven shape deformation method based on the surface information and space embedding is proposed. In the

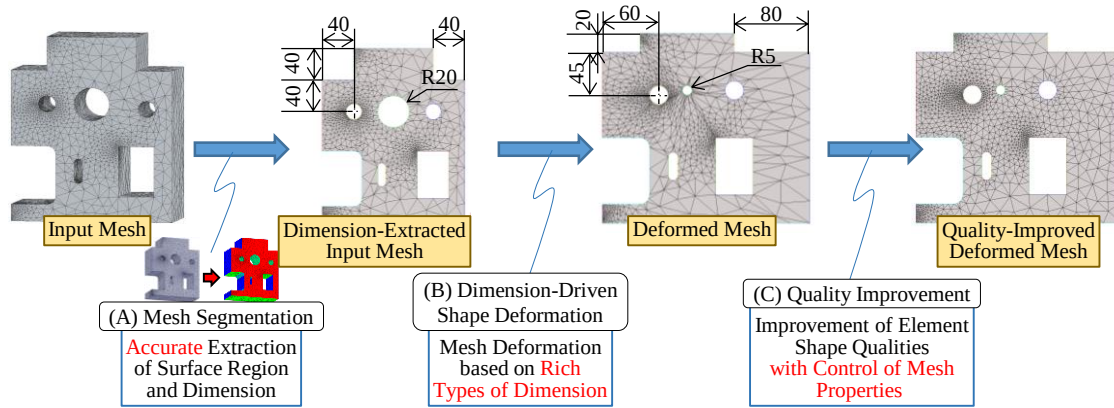


Fig. 2.1 Functional requirements of the dimension-driven mesh deformation

dimension-driven shape deformation method, the consistency of the curved surface of the input tetrahedral mesh is preserved during the deformation by the surface region classification and vertex repositioning according to the result of the classification. Moreover, a local region is extracted and the deformation is applied to only the local region for efficient deformation. The deformation method enables us to change various feature parameters of tetrahedral meshes such as the radius of the fillet and the angle of the chamfer. In section 2.6, a quality improvement method based on Optimal Delaunay Triangulation (ODT) [Chen 2004a] is proposed. The proposed quality improvement method has following four features. First, element shape qualities of all elements are improved by a sequential ODT-based method. Second, degenerated and inverted elements are removed by local topological operations. Third, the mesh density of the original mesh is preserved by the vertex insertion and removal according to the target mesh density field represented by a regular grid. Fourth, the shape approximation accuracy of the original mesh is kept in the resulting mesh by vertex insertion and vertex removal according to the acceptable shape error extracted from the original mesh. Finally, the effectiveness of the proposed methods is demonstrated in section 2.7, and this chapter is summarized in section 2.8.

2.2 Related work

2.2.1 Existing mesh segmentation methods

2.2.1.1 Mesh segmentation methods

Mesh segmentation is used in various applications e.g. mesh simplification, reverse engineering, texture mapping, and character animation. Dimension-driven mesh deformation also needs mesh segmentation to extract dimensions of the product shape. Mesh segmentation for dimension-driven mesh deformation has to divide the surface of the mesh model into each plane and curved surfaces e.g. cylinder, cone, sphere, and torus.

Geng et al. [2010] proposed a CAD mesh (triangular mesh) segmentation method based on hierarchical clustering and primitive fitting. In their method, using hierarchical clustering based on surface fitting, planar, spherical, torus, cylindrical, and conical surface regions (segments) are extracted, in that order. However, results may include some over or under segmentations.

Yan et al. [2012] proposed an extension of Variational Shape Approximation (VSA) [Cohen-Steiner 2004] for triangular meshes. In their method, at first, a function of error between surface region and fitting surface is minimized, where grouping proxy of each surface region and triangles of the input triangular mesh using distortion-minimizing flooding algorithm and fitting surface to surface region are performed iteratively. Then, region integration and region insertion based on the error function are performed. The first process (iteration of grouping and surface fitting) and second process (region integration and region insertion) are repeated until the values of the error functions of all surface regions become lower than a threshold. Finally, boundary line smoothing based on graph cut is performed. It can accurately divide the surface of mesh models into quadric surfaces, but cannot handle torus surfaces. In addition, the method is time-consuming, and needs other rough segmentation methods or simplification methods for efficiency.

Xiao et al. [2011] also proposed a CAD mesh (triangular mesh) segmentation method based on randomized Hough transformation and the mean shift technique. In their method, at first, the surface of the input CAD mesh model divided into a set of triangles belonging to planar, cylindrical, and conical surfaces or a set of triangles belonging to spherical and torus surfaces by clustering based on the shape and density of triangles. Then, planar, cylindrical, and conical surface regions are extracted from the first type of cluster (a set of triangles) by randomized Hough transformation. Finally, spherical and torus surface regions are extracted from the second type of cluster by the mean shift technique based on mean curvature of triangles. Although their segmentation method works very well, it uses a specific characteristic of the CAD mesh models, such as the shape and density of the triangles depending on the types of surfaces. Therefore, the surface of the mesh models for FEA (i.e. FEM meshes) cannot be divided accurately by their method.

Shu et al. [2011] proposed a triangular mesh segmentation method based on clustering using curvatures. In their method, some clusters of triangles are extended in order to maximize an objective function representing similarity of curvature. Although their method is very fast, the number of clusters needs to be specified by user and results may include some over or under segmentations.

2.2.1.2 Feature line extraction methods

If it is possible to obtain feature lines of the input product shape accurately, surface regions can be extracted by extracting sets of triangles which are bounded by feature lines. Therefore, feature line extraction methods may be useful for mesh segmentation.

Kim et al. [2009] proposed a feature line extraction method of triangular meshes based on the

k-means clustering and normal tensor. The k-means clustering is applied to vertices in the space of the normalized eigen values of the normal tensor. By the region growing using the result of the clustering, surface regions are extracted, and feature lines are extracted as boundary lines between surface regions. Their method can extract feature lines between surface regions which connect with each other with G^1 continuities. However, if the difference of the eigen values is small, their method cannot extract feature lines accurately. Although this problem can be solved by increasing number of clusters, the increase of clusters cause the increase of the calculation time and excess clusters cause over segmentations.

Tsuchie et al. [2014] proposed a feature line extraction method of triangular meshes based on principal curvature and principal directions calculated using a normal tensor framework. In their method, at first, the principal curvatures at each vertex are calculated using the normal tensor framework. Then, feature lines between a planar surface and other surfaces are extracted using difference of the principal curvatures. After that, using difference of principal directions, feature lines between two curved surfaces which connect smoothly with each other are extracted. Moreover, feature lines which cannot be extracted by difference of the principal curvatures and principal directions are extracted using difference of signs of the Gaussian curvature. However, if the principal curvatures are almost same and the difference of principal directions is small, feature lines cannot be extracted.

Zhihong et al. [2011] proposed a feature line extraction method based on the thresholding of the principal curvatures and the local modification of vertex classification by user. In their method, each vertex is classified into a boundary vertex or a non-boundary vertex by the thresholding of the principal curvatures. Then, the local regions including vertices which are not classified correctly are specified by user. After that, each vertex in the region is reclassified by its principal curvature. In this classification, all vertices in the local region are first labeled as non-boundary vertices, and then, each vertex whose principal curvatures are included in top 20-30 percent in the local region is labeled as a boundary vertex. Finally, by connecting neighboring boundary vertices, feature lines are extracted. Their method can extract feature lines according to requirements from user. However, many local regions have to be specified user in order to obtain feature lines for mesh segmentation. In addition, if boundary vertices cannot be extracted after reclassification using local regions, feature lines including them cannot be extracted.

Wang et al. [2012] proposed a feature line extraction method for triangular meshes which include some noises based on the normal tensor and *Neighbor Supporting*. Their method consists of the feature detection step and the feature refinement step. In the feature detection step, all feature vertices are extracted by modified tensor voting. Because the set of extracted feature vertices includes scan noise, in the feature refinement step, real feature lines are extracted using a novel salient measure via *Neighbor Supporting* (for a feature vertex, there will be a certain number of feature vertices along the principal direction, on the other hand, for a noisy vertex, very few or no feature vertices can be found by tracing its principal direction). Finally, as post-processing, weak

feature enhancing and optional branch pruning process are performed. Their method can extract weak feature lines from noisy triangular meshes. However, because feature vertices are extracted using eigen values of normal tensor, boundaries between two neighboring surface regions with G^1 continuity cannot be extracted as feature lines.

Jibin et al. [2012] also proposed a feature line extraction method using principal curvatures and principal directions at vertices. In their method, at first, the parabolic surface fitting is applied to 1-ring neighbors of each vertex in order to obtain principal curvatures and principal directions at each vertex. Then, feature vertices are extracted using principal curvatures, and feature lines are extracted by the seed growth (region growing using feature vertices). After that, if a feature line has less than 3 feature vertices, it is regarded a non-feature line. Finally, vertices are fitted into a Non-Uniform Rational Basis-Spline (NURBS) curve. Their method can extract feature lines as NURBS curves which are widely used in CAD systems. However, their method cannot feature lines between two surfaces whose curvatures are almost same.

2.2.1.3 Summary of existing methods

Comparisons of existing methods of mesh segmentation [Geng 2010, Xiao 2011, Yan 2012, Shu 2011] and feature line extraction [Kim 2009, Tsuchie 2014, Zhihong 2011, Wang 2012, Jibin 2012] are shown in Table 2.1 and Table 2.2, respectively. Existing mesh segmentation methods have any of the following problems.

- Neighboring surface regions with G^1 continuity cannot be segmented.
- The results may include over or under segmentations.
- Torus surfaces cannot be recognized.
- A specific characteristic of the CAD mesh models is needed.
- Some modification operations by user are needed.

On the other hand, existing feature line extraction methods cannot extract feature lines between two surfaces whose principal curvatures are approximately same and the difference of principal directions is small. Therefore, simple use of them for the mesh segmentation causes the under segmentation.

In this thesis, a new mesh segmentation method for dimension-driven mesh deformation is proposed. The proposed segmentation method is basis on the fact shown in [Tsuchie 2014] that using changes of the sign of curvatures and the principal directions is effective for extracting boundaries between two neighboring surface regions with G^1 continuities. In the proposed segmentation method, planar, cylindrical, conical, spherical, and torus surface regions are extracted in that order by region-growing based on normal vectors, principal directions, and surface fitting. Moreover, it can divide the surface of mesh models into each surface region which is connected with others with G^0 or G^1 continuities without over or under segmentations.

Table 2.1 Existing mesh segmentation methods

Method	Approach	Input Mesh	Recognized Surface	Problems
Geng 2010	Hierarchical clustering + Primitive fitting	CAD mesh (Triangular mesh)	Planar, quadric, and torus surface	Over or under segmentations
Yan 2012	VSA using Quadric Surface	Triangular mesh	Planar and quadric surface	Torus surface cannot be recognized.
Xiao 2012	Clustering + Randomized Hough transformation + Mean shift technique	CAD mesh (Triangular mesh)	Planar, quadric, and torus surface	A specific characteristic of the CAD mesh models is needed.
Shu 2011	Clustering based on curvature	Triangular mesh	Not recognizing	Not automatic Over or under Segmentations

Table 2.2 Existing feature line extraction methods

Method	Approach	Input	Advantage	Problems
Kim 2009	K-means clustering using eigen values of the normal tensor	Triangular mesh	Boundary lines between surface regions with G^1 continuities can be extracted	Unstable extraction of boundary lines between surface regions with G^1 continuities
Tsuchie 2014	Calculations of principal curvature and principal directions using a normal tensor framework	Triangular mesh	Boundary lines between surface regions with G^1 continuities can be extracted	Feature lines between two surfaces whose curvatures are approximately same cannot be extracted.
Zhihong 2011	Thresholding of the principal curvatures + Local modification of vertex classification by user	Triangular mesh	Feature lines are extracted based on requirements from user	Not automatic Feature lines between two surfaces whose curvatures are approximately same cannot be extracted.
Wang 2012	Feature detection based on the normal tensor + Feature refinement by Neighbor Supporting	Triangular mesh	Feature lines are extracted from noisy mesh	Boundary lines between surface regions with G^1 continuities cannot be extracted
Jibin 2012	Calculations of principal curvature and principal directions using parabolic surface fitting	CAD mesh (Triangular mesh)	Feature lines are extracted as NURBS curves	Feature lines between two surfaces whose curvatures are approximately same cannot be extracted.

2.2.2 Existing mesh deformation methods

2.2.2.1 Dimension-driven mesh deformation methods

For modification of mesh models of product shapes, many dimension-driven mesh deformation methods [Takano 2010, Onodera 2008, Ogawa 2008, Sawai 2010, Xian 2009] are proposed.

Takano et al. [2010] proposed a dimension-driven mesh deformation method for tetrahedral meshes based on space embedding. In their method, a segmentation method using the region growing based on normal vectors of triangles is first applied to surfaces of the input tetrahedral mesh. Then, for surface regions extracted by the segmentation, prism deformation handles are created using Variational Shape Approximation (VSA) [Cohen-Steiner 2004]. After that, surface regions are classified into one of three types using deformation types and surface regions specified by the user. After the target dimension is inputted, surface vertices are repositioned using barycentric coordinates for prism handles, and inner vertices follow surface vertices by Mean Value

Coordinates (MVCs) [Floater 2003] for polyhedra consisting of their 1-ring neighbors. Their method can change the three types of dimensions: distance between two parallel planes (DP, e.g. height of the boss), radius of the cylinder (RC, e.g. radius of the circular hole), and position of the local object on the plane (PO) of the tetrahedral meshes. Their method can deform tetrahedral meshes consisting only of planar and cylindrical surfaces. Moreover, if the dimension associated by two connecting surfaces with G^1 continuity, it cannot be changed.

Onodera et al. [2008] proposed a parametric morphing method for tetrahedral meshes based on surface fitting and Laplacian smoothing [Field 1988]. In their method, as preprocessing, a segmentation method using the region growing based on curvatures and surface fitting is applied to the surface of the input tetrahedral mesh. In the deformation step, each triangle is first labeled by the fitted surfaces. Then, the fitted surfaces are deformed so as to satisfy the target dimension, and vertices of triangles labeled by the fitted surfaces are moved onto the deformed surfaces. Finally, the quality of surface triangular mesh is improved by Laplacian smoothing, and inner vertices are also repositioned by Laplacian smoothing. Their method can change DP, RC, PO, and the radius of the spherical surfaces for the mesh models. However, vertices belonging to two or more surfaces are moved to the target position by iterative manner as shown in Fig. 2.2. Therefore, the vertex repositioning may take a long processing time.

Ogawa et al. [2008] proposed a 3D triangular mesh deformation method using some constraints. In their method, positions of vertices after deformation are decided by the least square method in order to minimize an objective function consisting of constraints for mean curvatures of vertices, positions of vertices, and shape of form features. Their method can interactively and intuitively deform the 3D triangular meshes of assembly models including some non-manifold regions. In addition, the method can deform tetrahedral mesh models by combining it with their related work [Masuda 2008] based on the space embedding method using MVCs [Floater 2003].

Sawai et al. [2010] proposed a tetrahedral mesh deformation method using CAD data. Their method can deform tetrahedral meshes by specifying the parameters of the form features, and copy form features. In addition, their method can deform loft surfaces, and modified solid models can be easily generated from resulting mesh models through the relation between them. However, as a preprocessing, the solid model in the CAD system has to be divided into each form feature, and a data base of the form features needs to be created. Moreover, the conformity of the boundary between the copied mesh of the form feature and the base mesh is not kept. In their method, Femap [Femap] which is a pre-post software of FEA is used, and the detail of the deformation process is not described.

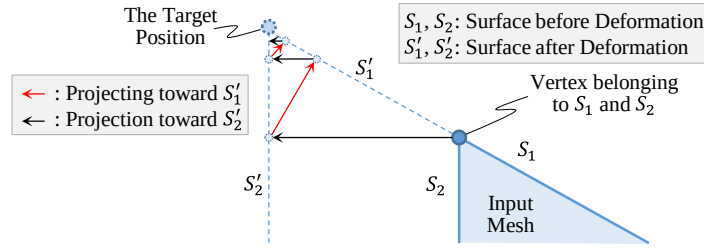


Fig. 2.2 The repositioning of vertices belonging to two or more surfaces in [Onodera 2008]

Xian et al. [2009] proposed a 3D triangular mesh editing method using some cages (deformation handles). In their method, the input triangular mesh is manually divided into a base mesh and some triangular meshes of form features. Then, deformation handles for each form feature are created. In the deformation step, each vertex is repositioned by Green Coordinates [Lipman 2008] for deformation handles while satisfying constraints of each form feature. Their method can deform mesh models while preserving rectangular parallelepipeds and cylinders. The method also can handle assembly models and can be applied to large mesh models. However, their method can deform only triangular meshes consisting of planar and cylindrical surfaces.

2.2.2.2 Other mesh deformation methods

Staten et al. [2011] surveyed inner vertex repositioning methods based on following six approach: Smoothing, Weighted Residuals, Simplex-Linear Transformation, Simplex-Natural Neighbor Transformations, Finite Element-based Mesh Warping (FEMWARP), and Log Barrier-based Mesh Warping (LBWARP). In their paper, these six methods are applied to four tetrahedral meshes and six all-hex meshes and the results are compared. As a results of comparison, FEMWARP provided the best results in terms of the element quality for all test cases and can be implemented easily. In FEMWARP, position of each inner vertex is represented as an affine combination of its neighbors, and the weights form an element stiffness matrix $\mathbf{A}$. Then, $\mathbf{A}$ is divided into two sub matrices: $\mathbf{A}_I$ whose rows and columns are indexed by inner vertices, and $\mathbf{A}_B$ whose rows are indexed by inner vertices and columns are indexed by boundary vertices. Let $\mathbf{X}_I$ is a matrix whose each row represents coordinates of each inner vertex, and $\mathbf{X}_B$ is a matrix whose each row represents known coordinates of each boundary vertex. After repositioning of boundary vertices, the positions of inner vertices are calculated by solving a linear system of equations $\mathbf{A}_I \mathbf{X}_I = -\mathbf{A}_B \mathbf{X}_B$. In our method, the positions of inner vertices of tetrahedral meshes are obtained by similar manner.

Based on the survey by Staten et al. [2011], Sieger et al. [2013] proposed a new mesh deformation method based on the triharmonic Radial Basis Function (RBF). In their method, at first, boundary lines of the underlying deformed solid model are uniformly sampled by vertices so that the number of vertices on the boundary lines will be the same as one on the boundary lines of the input mesh model. Then, surface vertices are repositioned by the triharmonic RBF interpolation

using boundary vertices in the 2D parameter space. Finally, inner vertices are repositioned by the triharmonic RBF interpolation using surface vertices. In comparison with FEMWARP which needs to be modified according to the element type, because their method is based on the triharmonic RBF interpolation, not only tetrahedral meshes but also all-hex meshes can be deformed by the same manner. In addition, the element shape qualities of mesh models deformed by their method are better than ones obtained by FEMWARP. However, in some test cases, the calculation time of their method became ten times as long as that of FEMWARP. They also propose a new deformation method [Sieger 2014] which use the Moving Least Square technique instead of the RBF interpolation. In comparison with the method using RBF interpolation, flexibility and scalability of method are increased. However, the method based on MLS needs the selection of several parameters such as the constraint weights, the number of sample points, and the support radii of the basis function.

2.2.2.3 Summary of existing methods

Summary of existing dimension-driven mesh deformation methods [Takano 2010, Onodera 2008, Ogawa 2008, Sawai 2010, Xian 2009] and other mesh deformation methods [Staten 2011, Sieger 2013, Sieger 2014] are shown in Table 2.3 and Table 2.4, respectively. In existing dimension-driven mesh deformation methods, except for the use of CAD data, tetrahedral meshes whose surface consists of planar, cylindrical, conical, and spherical surfaces can be deformed. In this thesis, a new dimension-driven deformation method for tetrahedral meshes using a space embedding method and surface information obtained from segmentation is proposed. The proposed method enables us to change more types of parameters of form features (e.g. radius of the fillet, angle of the chamfer, and length of the chamfer) of tetrahedral meshes. In addition, the proposed method can handle tetrahedral meshes whose surface includes torus surfaces. Moreover, parameters of form features with subsidiary surface such as fillets or chamfers cannot be changed by existing methods. On the other hand, the proposed method enable us to change them by a surface classification.

In addition, in all existing methods mentioned above, all vertex positions of the input mesh model are recalculated even if the deformation changed the shape of only one local form feature. It may take a large calculation time. In the proposed method in this thesis, a local region for deformation is extracted and the deformation and the quality improvement are applied only to the extracted region for efficiency.

Moreover, the resulting deformed meshes of existing methods often include many distorted elements, and lose original mesh properties such as mesh density and shape approximation accuracy. The distorted elements and the loss of original mesh properties cause low accuracy and inefficiency of FEA. Therefore, the quality improvement for deformed meshes is needed.

Table 2.3 Existing dimension-driven mesh deformation methods

Method	Approach	Target surface class	Types of deformation
Takano 2010	Space embedding using boundary coordinates and MVCs	Planar and cylindrical	Height of the boss Radius of the hole Position of the form feature
Onodera 2008	Surface fitting + Laplacian smoothing	Planar, cylindrical, conical, and spherical	Height of the boss Radius of the hole Radius of the fillet (cylindrical surface only) Radius of the sphere Position of the form feature
Ogawa 2008 Ogawa 2008	Three constraints + Space embedding using MVCs	No mentioned	Height of the boss Position of the form feature (Other is not mentioned.)
Sawai 2010	The use of CAD data	No mentioned	No mentioned
Xian 2009	Space embedding using Green coordinates	Planar and cylindrical	Height of the boss Radius of the hole

Table 2.4 Existing other deformation method

Paper	Method	Advantage	Problems
Staten 2011	Smoothing	High element shape quality for small deformations	Inverted elements are often generated.
	Weighted Residuals	High element shape quality for simple shapes	Taking long time for complex shapes
	Simplex-Linear	Validate element shape quality and fast processing independently of the complexity of the shape and scale of deformation	Other repositioning technique is needed for surface vertices (in, Weighted Residuals is used.)
	Simplex-Natural Neighbor	Higher element shape quality than Simplex-Linear	Having the same problem as Simplex-Linear Complex implementation
	FEMWARP	Simple implementation and high element shape quality	Taking long time for large deformation
	LBWARP	high element shape quality	Having the same problem as FEMWARP The 1-ring neighbors of each vertex must form a convex hull
Sieger 2013	RBF	Simple implementation and high element shape quality	Time-consuming (ten times as long as FEMWARP)
Sieger 2014	MLS	High element shape quality	Complex implementation (The selection of several parameters is needed.)

2.2.3 Existing mesh quality improvement methods

2.2.3.1 Mesh quality improvement methods

There are many methods of quality improvement for tetrahedral meshes, and the methods based on Optimal Delaunay Triangulation (ODT) [Chen 2004a] and Centroidal Voronoi Tessellation (CVT) [Du 2003] are especially effective.

ODT smoothing [Chen 2004b, Chen 2011] is one of the ODT-based methods. In ODT smoothing, vertex repositioning and edge (face) flipping are repeated to minimize a specific energy. In the repositioning step, interior vertices are moved to the average of circumcenters of their neighboring elements. However, because boundary vertices are fixed, the element shape qualities near the boundary of tetrahedral meshes are not improved.

In order to solve this problem for tetrahedral meshes, Tournois et al. [2009a] proposed Natural Optimal Delaunay Triangulation (NODT). They added a special term into the repositioning equation of ODT smoothing for boundary (surface) vertices. In NODT, if the surface vertices move to the outside of the given tetrahedral mesh, the vertices are repositioned to the closest position on the surface of the tetrahedral mesh. Therefore, the element shape qualities near the surface of tetrahedral meshes may not be completely improved.

Gao et al. [2012] proposed Boundary-Optimized Delaunay Triangulation (B-ODT) in order to solve the same boundary problem. In B-ODT, boundary vertices move under the tangent plane constraint and the feature preserving constraint. B-ODT provides better results than NODT.

The methods based on CVT [Du 2003] improve element shape quality of tetrahedral meshes by iterative generation of Voronoi Diagram and Delaunay Triangulation (Tetrahedrization) using the centroids of the Voronoi regions. In general, ODT smoothing converges faster than the CVT-based method and resultant element shape qualities of ODT are higher than that of CVT.

For the quality improvement of tetrahedral meshes deformed by dimension-driven mesh deformation methods, vertex insertion and vertex removal are important. For example, as shown in Fig. 2.3, when a product shape is expanded drastically, the tetrahedral mesh with high element shape qualities cannot be obtained by using only vertices of the original (before deformation) tetrahedral mesh. Because ODT smoothing and the CVT-based methods improve qualities of element shapes of tetrahedral meshes without vertex insertion and vertex removal, they are not appropriate for tetrahedral meshes deformed by dimension-driven mesh deformation methods. In order to improve the deformed tetrahedral meshes, Onodera et al. [2014] proposed a new quality improvement method. In their method, before the deformation, a field of element shape qualities is generated using barycenters of tetrahedral elements. After the deformation, tetrahedral elements whose shape qualities do not satisfy thresholds obtained from the element shape quality field are removed from current mesh model. Then, the advancing front mesh generation is applied to the hole created by the element removal. The element shape qualities of original tetrahedral meshes are kept in deformed tetrahedral meshes by their method. However, their method does not handle original mesh properties such as the mesh density and the shape approximation accuracy explicitly.

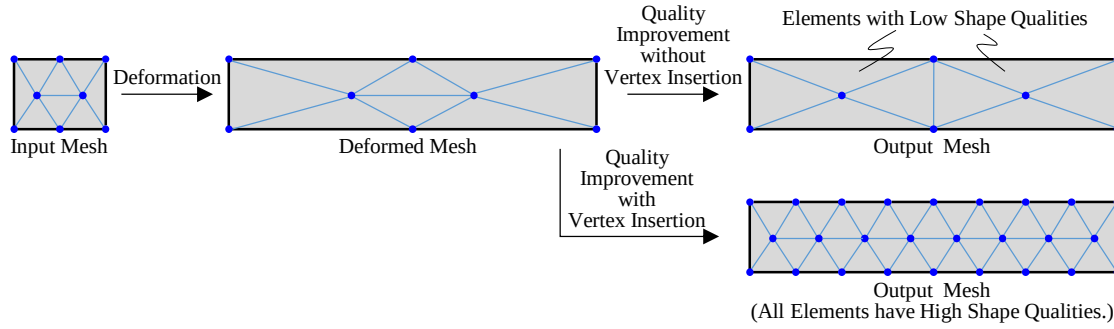


Fig. 2.3 Importance of vertex insertion and vertex removal

2.2.3.2 Degenerated or inverted element removal methods

Mesh models sometimes include degenerated elements, such as slivers. When only using ODT smoothing, they are not often removed. For improving their quality, explicit perturbation and local topological modification are used in NODT [Tournois 2009a] and B-ODT [Gao 2012], respectively. Moreover, many researches for improving or removing the degenerated elements based on vertex repositioning or changing connectivity have been conducted [Chew 1997, Cheng 2000, Edelsbrunner 2002, Li 2000, Tournois 2009b, Li 2014, Faraj 2016, Houlin 2014, Gao 2012, Li 2003]. In addition, also inverted elements are sometimes included in mesh models. In order to remove inverted elements from mesh models, many mesh untangling methods are proposed [Takano 2010, Knupp 2001, Bhowmick 2010, Sastry 2012].

Sliver Removal (Degenerated Element Removal)

There are many degenerated element removal methods, and removing sliver is especially desired and conducted. They are roughly classified into the following four types: Delaunay refinement-based [Chew 1997, Labelle 2007], weighted Delaunay triangulation-based [Cheng 2000, Edelsbrunner 2002], vertex perturbation-based [Li 2000, Tournois 2009b], and combination of local topological operations-based [Faraj 2016, Houlin 2014, Gao 2012, Li 2003].

Chew [1997] proposed a sliver removal method based on the Delaunay refinement. In the removal method, vertices are randomly inserted near the circumcenter of slivers in order to perform flipping operations which makes mesh models Delaunay triangulations. Chew's method theoretically avoided the common problem of Delaunay refinement-based methods that new slivers may be generated by the vertex insertion and flipping operations. However, a sliver whose four vertices are lying on the boundary (surface) cannot be removed. In addition, the method handles only tetrahedral meshes with the uniform density.

Labelle [2006] also proposed a sliver removal method based on the Delaunay refinement. In the removal method, the positions of added vertices are determined using two types of lattice. Except

for the regions near the boundary, all dihedral angles of tetrahedra become between 30 degree and 150 degree by their method. However, the input tetrahedral mesh is assumed that it has only a few tetrahedra with the low element shape qualities, and if many tetrahedra of the input tetrahedral mesh have low element shape qualities, many new vertices may be required.

Cheng et al. [2000] proposed a sliver exudation method based on weighted Delaunay triangulation. Edelsbrunner et al. [2002] presented an experimental study of the sliver exudation. In their method, a Delaunay triangulation is turned into a weighted Delaunay triangulation by assigning weights to vertices. Flipping operations are introduced by the weight of each vertex and minimum dihedral angle is enlarged. For mesh models of the infinite domain, all slivers can be removed theoretically without vertex repositioning, insertion, or removal. However, flipping operations cannot be performed near the boundary. Therefore, some slivers remain near the boundary.

Li [2000] proposed a sliver removal method based on random perturbation of vertices. Inspired by Li's method, Tournois [2009b] proposed a perturbation method based on the gradient of circumradius and volume of tetrahedra for sliver removal. In Tournois's method, vertex of each sliver is moved so that the circumradius of the degenerated tetrahedron will increase or the volume of the tetrahedron will decrease. Then, flipping operations are performed in order to maintain the valid Delaunay Triangulation. If slivers are not removed by two perturbation, Li's method is performed. Tournois's method is combined with NODT and the result has the larger minimum dihedral angle than that of the results of [Cheng 2000] and [Li 2000]. However, a sliver whose four vertices are lying on the boundary (surface) cannot be removed.

Li et al. [2013] combined a vertex perturbation-based method and a Delaunay refinement-based method in their meshing method. In their method, inner slivers are removed by vertex perturbations using directions of the circumradius gradient ascent and the volume gradient ascent. On the other hand, slivers near the boundary are removed by a Delaunay refinement based on vectors of these perturbation directions. The number of vertices added by their method for sliver removal is smaller than the one by [Chew 1997]. Because their method is based on vertex perturbations and a Delaunay refinement, the input tetrahedral meshes should be Delaunay triangulations.

Faraj et al. [2016] proposed a re-meshing method including a sliver removal. In their method, slivers are removed by two types of flipping operations. These flipping operations are included in ODT smoothing which is used in this thesis, and degenerated elements near the boundary cannot be removed by only flipping operations.

Houlin et al. [2014] proposed a mesh optimization method including sliver removal. In their method, a flipping operation and the combination of local topological operations are used for sliver removal. In the combination of local topological operations, two vertices are inserted to an edge and two newly two edges generated by the vertex insertion are collapsed. By the combination of local topological operations, slivers near the boundary can be removed. However, only slivers can

be removed and other types of degenerated elements have to be removed by other quality improvement method.

Gao et al. proposed a sliver removal method based on local topological operations in B-ODT [Gao 2012]. In their method, a slivers is turned into eight tetrahedra by inserting two vertices to two edges. In addition, a cap or a spade which is one of degenerated tetrahedra is turned into three or two tetrahedra by inserting a vertex to a triangle or an edge, respectively. The tetrahedra generated by the local topological operations have at least one short edge, and are improved by B-ODT. However, the number of tetrahedra becomes large and all degenerated tetrahedra may not be removed after B-ODT.

Li et al. [2003] also proposed a simple degenerated element removal method by the combination of local topological operations. In their method, a sliver is turned into four triangles by inserting two vertices to two edges and collapsing a new short edge (i.e. double split collapse). In addition, a cap or a spade is turned into two triangles by inserting a vertex to an edge and collapsing a new short edge (i.e. face collapse). Their method can remove slivers near the boundary. In addition, the increase of the number of tetrahedra is smaller than the one by [Gao 2012]. In a mesh adaptation method [Compère 2010], this method is adopted. The proposed method in this thesis also adopts their method.

As another type of approach, Guo et al. [2016] proposed a meshing method including a sliver removal. In their method, two vertices of the longest edge of a sliver are first removed, and the hole created by the removal is then re-meshed by their meshing method. In [Guo 2016], the minimum dihedral angle of results of their method is larger than that of the results of [Cheng 2000] and [Tournis 2009b]. However, the experiment is performed using a mesh model generated by their meshing method as an initial mesh model. Therefore, the number of slivers is smaller than that of deformed meshes obtained in the problem setting of this thesis and a large number of re-meshing may be needed for the deformed meshes.

Mesh Untangling (Inverted Element Removal)

Takano et al. [2010] proposed a mesh untangling method for tetrahedral meshes deformed by their dimension-driven mesh deformation method. In their method, a vertex of the inverted tetrahedron are moved into an inverted-free space so as to optimize the element shape quality of tetrahedron. First, by solving a linear programming problem, the existence or nonexistence of the inverted-free space is checked. If the inverted-free space exists, the vertex is repositioned to the solution of the linear programming problem. After that, by vertex repositioning based on solving the constrained nonlinear optimization problem, the element shape quality of the tetrahedron is improved. In their method, only one vertex of each inverted tetrahedron is moved. Therefore, any inverted-free space may be found, and if the inverted-free space cannot be found, the inverted tetrahedron cannot be removed.

Knupp [2001] proposed a mesh untangling method for hexahedral and tetrahedral meshes. In

their method, a global single objective function based on the difference between the absolute and signed volumes of each tetrahedron is minimized by inner vertex repositioning using a conjugate gradient method. In their method, the acceptable volume of tetrahedra can be controlled, but if it is too large, the optimization problem cannot be solved. On the other hand, if the acceptable volume is too small, degenerated tetrahedra are generated.

Bhowmick et al. [2010] proposed a mesh untangle and quality improvement method based on Fruchterman-Reingold (FR) graph layout algorithm [Fruchterman 1991]. In FR algorithm, graph nodes and edges are placed so that (1) the number of edges crossing each other will be minimized, and (2) the standard deviation of the edge length will be reduced. Graph nodes are considered as charged objects and graph edges are considered as springs. Therefore, two types of forces work in the nodes: (1) attraction forces due to the spring only between connected nodes, and (2) repulsion force due to mutually charged objects between all nodes. Each node is repositioned by these forces. In [Bhowmick 2010], FR algorithm is modified by weight assignment and boundary modification so that it can be applied to a mesh model as a graph. In comparison with [Knupp 2001], their method is substantially faster and element shape qualities of results are better. However, it is difficult to preserve the shape of input mesh models.

Sastry et al. [2012] proposed a mesh untangling and quality improvement method based on a log-barrier method. In their method, inner vertices are repositioned based on the solution of a constrained optimization problem. The constrained optimization problem using an objective function based on an element shape quality measure is solved by an *interior point method*. In the interior point method, the constraint is added as a logarithmic barrier term to the objective function, and the modified objective function is maximized by an iterative unconstrained optimization method. In each iteration step, the weight of the logarithmic barrier term decreases, and when the weight becomes approximately 0, the objective function is maximized. In comparison with an iterative stiffening method [Shontz 2012], although the number of iterations becomes larger, their method can modify tetrahedral meshes with inverted elements more stably. However, in their method, only inner vertices are repositioned. Therefore, inverted tetrahedra whose triangle is on the surface and inverted cannot be removed.

Although, these methods mentioned above may be useful for improving or removing the degenerated and inverted elements, in this research, a simple method for removing the degenerated and inverted elements caused by the mesh deformation is adopted. The method is based on the local topological operations.

2.2.3.3 Summary of existing methods

The quality improvement methods based on ODT and CVT are mentioned above, and in general, the ODT-based methods prefer over the CVT-based methods. In original ODT smoothing, the vertices on the boundary of the mesh model are fixed, therefore it cannot improve the shape qualities of the boundary elements of mesh models (triangles of tetrahedral meshes) well.

Tournois's method (NODT) and Gao's method (B-ODT) solved this problem and proposed an extension method for improving qualities of the boundary element shapes of mesh models in the framework of the ODT. In the proposed method in this thesis, in order to improve the boundary elements (e.g. surface triangles for the tetrahedral mesh), the ODT is sequentially applied to the boundary of surface regions, the surface, and the inner volume while fixing each boundary. This strategy will provide better surface mesh quality because the ODT energy for the surface is minimized independent of the volume. In addition, this method is applicable to 3D surface mesh models.

In addition, when using only ODT smoothing, degenerated and inverted elements are not often removed. As shown in Table 2.5 and Table 2.6, many methods which improve or remove the degenerated and inverted elements have been proposed [Chew 1997, Cheng 2000, Edelsbrunner 2002, Li 2000, Tournois 2009b, Li 2014, Faraj 2016, Houlin 2014, Gao 2012, Li 2003, Takano 2010, Bhowmick 2010, Knupp 2001, Sastry 2012]. Although they may be useful for improving or removing the degenerated and inverted elements, a simple method [Li 2003] for removing them by the combination of some local topological operations is adopted for degenerated element removal in this thesis, and a simple method by a local topological operation is proposed for inverted element removal.

In addition to element shape qualities, mesh density, and shape approximation accuracy of the original tetrahedral mesh should be preserved after mesh deformation. Because ODT smoothing and the CVT-based methods improve qualities of element shapes of tetrahedral meshes without vertex insertion and vertex removal, they cannot recover mesh density and shape approximation accuracy of the original tetrahedral mesh. Although Onodera's method [Onodera 2014] is developed in order to improve deformed tetrahedral meshes, their method also does not deal with mesh density and shape approximation accuracy. In the proposed method in this thesis, ODT smoothing is combined with insertion and removal of vertices in order to improve element shape qualities of deformed tetrahedral meshes and preserve mesh density and approximation accuracy of the original tetrahedral mesh.

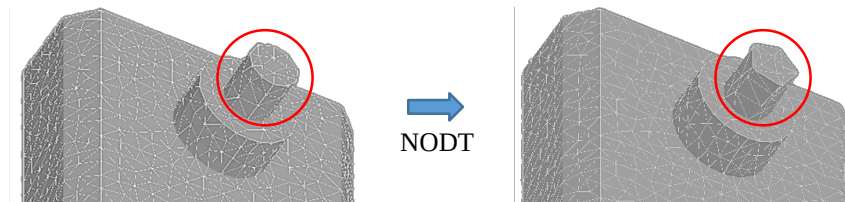
In fact, NODT is a process included in a refinement method (i.e. it can control the mesh density and the shape approximation accuracy), and is combined with a sliver removal method. Therefore, NODT may be useful for improving tetrahedral meshes which are deformed by dimension-driven mesh deformation methods. However, as shown in Fig 2.4, the projection of the surface vertices which move to outside of the given tetrahedral mesh may cause a fatal change of shape on the surface whose curvature is very large. In addition, although their method includes improvement of degenerated elements, inverted elements are not improved and may be generated in the projection process.

Table 2.5 Existing method of degenerated element removal

Method	Type	Increase of elements	Boundary Elements	Input mesh
Chew 1997	Sliver removal (Delaunay refinement)	May be large	Can NOT be removed (when all vertices on the boundary)	Should be Delaunay
Labelle 2006	Sliver removal (Delaunay refinement)	May be large	Can NOT be removed	Should be Delaunay
Cheng 2000 Edelsbrunner 2002	Sliver removal (Weighted Delaunay)	None	Can NOT be removed	Should be Delaunay
Li 2000	Sliver removal (Vertex perturbation)	None	Can NOT be removed (when all vertices on the boundary)	Should be Delaunay
Tournois 2009b	Sliver removal (Vertex perturbation)	None	Can NOT be removed (when all vertices on the boundary)	Should be Delaunay
Li 2013	Sliver removal (Vertex perturbation & Delaunay refinement)	None	Can be removed	Should be Delaunay
Fraj 2016	Sliver removal (Local topological operations)	None	Can NOT be removed	Not important
Houlin 2014	Sliver removal (Local topological operations)	May be large	Can be removed	Including ONLY sliver as degenerated elements
Gao 2012	Sliver removal (Local topological operations)	May be large	Can be removed	Not important
Li 2003 Compère 2010	Sliver removal (Local topological operations)	May be large but smaller than Gao 2012	Can be removed	Not important
Guo 2016	Sliver removal (Local re-meshing)	May be large	Can be removed	Not important (Should be high quality)

Table 2.6 Existing method of inverted element removal

Method	Approach	Advantage	Drawback
Takano 2010	Search for an inverted-free space by solving a linear programming problem & inner vertex repositioning to the inverted-free space	Only vertices whose positions need to be changed are moved to improve neighboring tetrahedra. (Efficiency)	Inverted-free space may not be found & some boundary inverted tetrahedra cannot be removed.
Knupp 2001	Inner vertex repositioning minimizing objective function based on signed volumes of tetrahedra	For a single free vertex, the objective function is convex.	Element shape qualities are low after untangling & some boundary inverted tetrahedra cannot be removed.
Bhowmick 2010	Vertex repositioning based on FR graph layout algorithm	Faster and element shape quality of result is better than [Knupp 2001]	Shape of mesh models may not be preserved.
Sastry 2012	Inner vertex repositioning maximizing objective function based on element shape quality of tetrahedra by interior point method	Stable removal of inner inverted tetrahedra.	Some boundary inverted tetrahedra cannot be removed.

**Fig. 2.4 A fatal change of shape by NODT**

2.3 Overview of proposed method

Based on the existing methods and their problems mentioned above, in this chapter, a novel method of the dimension-driven deformation of tetrahedral meshes is proposed. As shown in Fig. 2.5, the proposed method consists of surface segmentation, dimension-driven shape deformation, and quality improvement. In the proposed method, tetrahedral meshes are handled and it is assumed that the surface of the input tetrahedral mesh consist of planar, cylindrical, conical, spherical, and torus surfaces.

The mesh segmentation (A1) is basis on the fact shown in [Tsuchie 2014] that using changes of the sign of curvatures and the principal directions is effective for extracting boundaries between two neighboring surface regions with G^1 continuities. In the mesh segmentation, planar, cylindrical, conical, spherical, and torus surface regions that are connected with other regions with G^0 or G^1 continuities are sequentially extracted. In order to extract each surface region, region-growing based on principal directions, normal vectors, and surface fittings are applied to the surface of the input tetrahedral mesh.

In the dimension-driven shape deformation (A2), in order to keep the consistency of the curved surface of the input tetrahedral mesh, each surface region is first classified into four types using deformation types and surface regions specified by the user, and the surface information obtained from the segmentation results. Then, a local region for efficient deformation is extracted using the type of deformation and types of surface regions. Finally, each vertex of the extracted local region is moved according to the types of surface regions using the space embedding method and the surface information. The deformation method enables us to change various feature parameters of tetrahedral meshes such as the radius of the fillet and the angle of the chamfer.

As the quality improvement method (A3), Phased ODT smoothing is proposed. Phased ODT smoothing improves element shape qualities from the boundary to the inside of the mesh model in a step-by-step manner. Moreover, it recovers original mesh properties such as mesh density and shape approximation accuracy. In addition, a simple method based on local topological operations for removing degenerated and inverted elements is included in the Phased ODT smoothing.

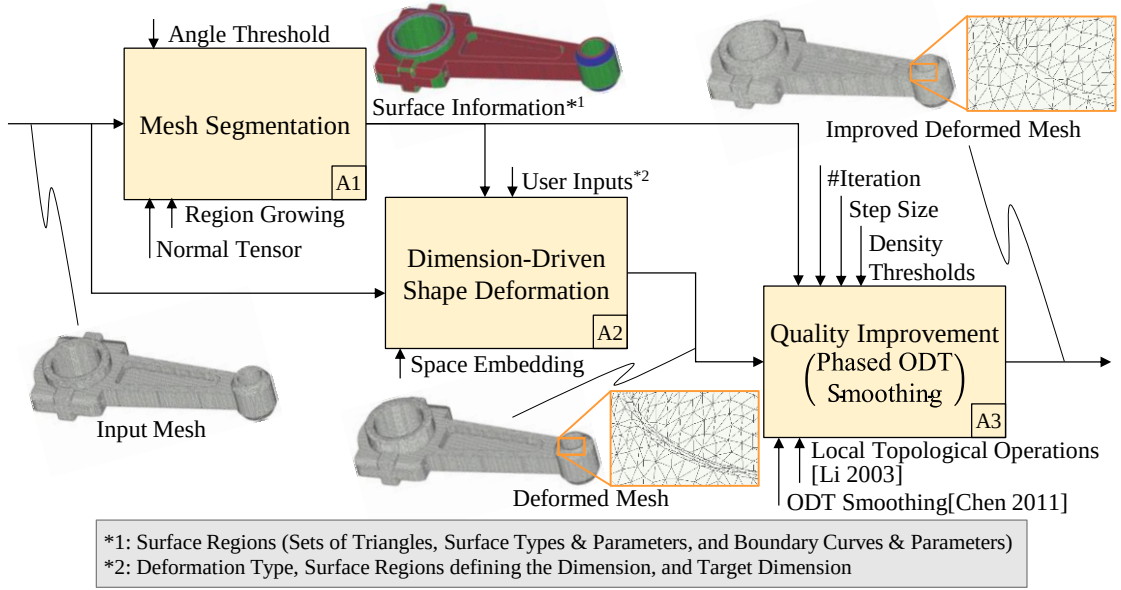


Fig. 2.5 The overview of the proposed dimension-driven tetrahedral mesh deformation method

2.4 Mesh segmentation based on normal tensor and region growing

2.4.1 Overview of the mesh segmentation

In the proposed segmentation method, the surface of the input tetrahedral mesh is divided into planar, quadric (cylindrical, conical, and spherical), and torus surface regions. Using change of the sign of curvatures and the principal directions is effective for extracting boundary lines between two neighboring surface regions with G^0 or G^1 continuities [Tsuchie 2014]. Moreover accurate surface fitting and an evaluation of the distance between the fitting surface and mesh vertex are useful for accurate segmentation because vertices of tetrahedral meshes for FEA exist exactly on the underlying curved surfaces.

The overview of the proposed segmentation method is shown in Fig. 2.6. In the segmentation method, first, sharp edges are extracted by thresholding of the dihedral angle (A11). Second, principal curvatures and principal directions at each vertex are calculated by a normal tensor framework [Tsuchie 2014] (A12). Third, planar, cylindrical, conical, and spherical surface regions are extracted in that order by region-growing based on principal directions, normal vectors, and surface fittings [Schnabel 2007] (A13). Then, region growing based on the sign of the Gaussian curvature is applied to the rest of the surface regions to extract torus surface regions, and false recognition surface regions on the torus surface regions are removed (A14). After that, the least

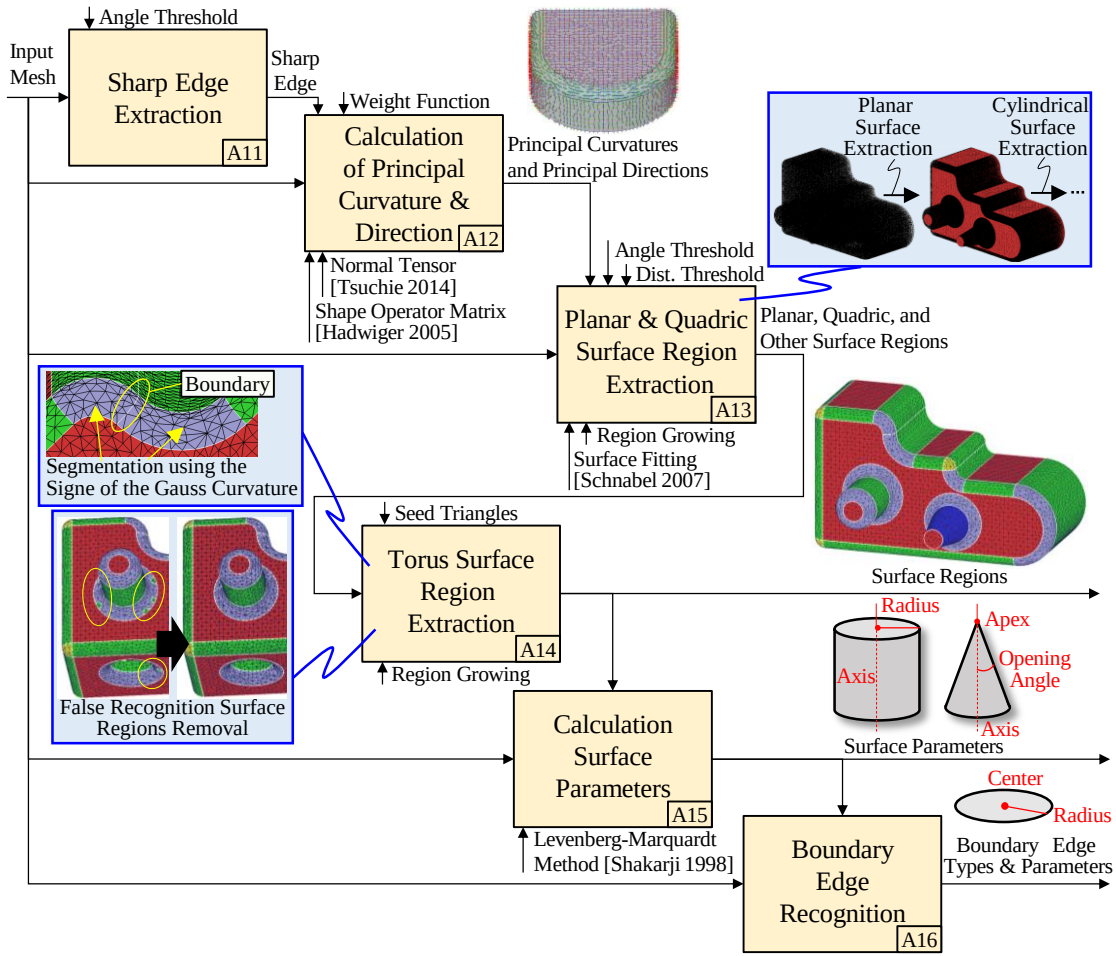


Fig. 2.6 The overview of the proposed segmentation method

square fitting based on the Levenberg-Marquardt method [Shakarji 1998] is applied to each curved surface region for more accurate fitting (A15). Finally, the type of each boundary edge is recognized, and the parameters of each boundary edge are calculated (A16).

In the segmentation method, it is assumed that the surface of the input tetrahedral mesh has enough triangles in each surface region, and each boundary line between two neighboring surface regions can be represented by a sequence of edges of the tetrahedral mesh.

2.4.2 Sharp edge extraction and calculation of principal curvature and direction

In the proposed segmentation method, the principal and Gaussian curvatures and the principal directions at each triangle are calculated based on the normal tensor framework [Tsuchie 2014] which calculates the curvatures at mesh vertices. In the curvature calculation method, curvatures at vertices on the sharp edges cannot be calculated accurately. Therefore, as a preprocessing, sharp

edges are extracted by thresholding of dihedral angles (A11).

After that, in the curvature calculation (A12), the normal tensor $\mathbf{N}_i$ of each vertex i except for vertices on the sharp edges is first calculated as shown in Eq. (2.1):

$$\mathbf{N}_i = \sum_{t \in N_i^{tgl}} \mathbf{n}_t^{tgl} (\mathbf{n}_t^{tgl})^T, \quad (2.1)$$

where N_i^{tgl} is a set of the neighboring surface triangles of i , t a triangle, and $\mathbf{n}_t^{tgl}$ is the normal vector of t . Secondly, the eigenvalue analysis is applied to the normal tensor of each vertex. The eigenvectors $\mathbf{e}_{i,m}$ ($m = 1, 2, 3$) of the eigenvalues $\mu_{i,1} \geq \mu_{i,2} \geq \mu_{i,3}$ of the normal tensor correspond to the normal vector, the principal directions of the maximum and minimum principal curvatures at the vertex i . Then, the maximum and minimum principal curvatures are calculated as eigenvalues of the 2×2 Shape Operator matrix [Hadwiger 2005] which is defined by the eigenvectors of the normal tensor as Eq. (2.2):

$$\mathbf{A}_i^{Sh} = (\mathbf{e}_{i,2} \mathbf{e}_{i,3})^T (\nabla \mathbf{e}_{i,1}^T) (\mathbf{e}_{i,2} \mathbf{e}_{i,3}), \quad (2.2)$$

where $\nabla \mathbf{e}_{i,1}$ is the gradient of the normal vector $\mathbf{n}_i^{vtx}$ ($= \mathbf{e}_{i,1}$) at vertex i and it defined as Eq. (2.3):

$$\nabla \mathbf{e}_{i,1}^T = \nabla \mathbf{n}_i^{vtxT} = \mathbf{A}_i^{Ro} \mathbf{A}_i^{GrT} (\mathbf{A}_i^{Gr} \mathbf{A}_i^{GrT})^{-1}, \quad (2.3)$$

where

- $\mathbf{A}_i^{Ro} = [\mathbf{a}_1^{Ro} \cdots \mathbf{a}_{|N_i^{vtx}|}^{Ro}]$,
- $\mathbf{a}_j^{Ro} = \mathbf{n}_j^{vtx} - \mathbf{n}_i^{vtx}$,
- $\mathbf{A}_i^{Gr} = [\mathbf{a}_1^{Gr} \cdots \mathbf{a}_{|N_i^{vtx}|}^{Gr}]$,
- $\mathbf{a}_j^{Gr} = \mathbf{a}_{ji}^{Dis} - (\mathbf{n}_i^{vtx} \cdot \mathbf{a}_{ji}^{Dis}) \mathbf{n}_i^{vtx}$,
- $\mathbf{a}_{ji}^{Dis} = \mathbf{x}_j - \mathbf{x}_i$, and
- $\mathbf{x}_i, \mathbf{x}_j$: positions of vertices i and j , respectively.

If the inverse matrix of $\mathbf{A}_i^{Gr} \mathbf{A}_i^{GrT}$ cannot be calculated, $\mathbf{A}_i^{Ro}$ and $\mathbf{A}_i^{Gr}$ are re-defined by Eq. (2.4) and (2.5), respectively.

$$\mathbf{A}_i^{Ro} = [\mathbf{a}_1^{Ro} \cdots \mathbf{a}_{|N_i^{vtx}|}^{Ro} \mathbf{0}], \quad (2.4)$$

$$\mathbf{A}_i^{Gr} = [\mathbf{a}_1^{Gr} \cdots \mathbf{a}_{|N_i^{vtx}|}^{Gr} \mathbf{n}_i^{vtx}], \quad (2.5)$$

Finally, curvatures and principal directions at each triangle are defined as the average of those of its three vertices.

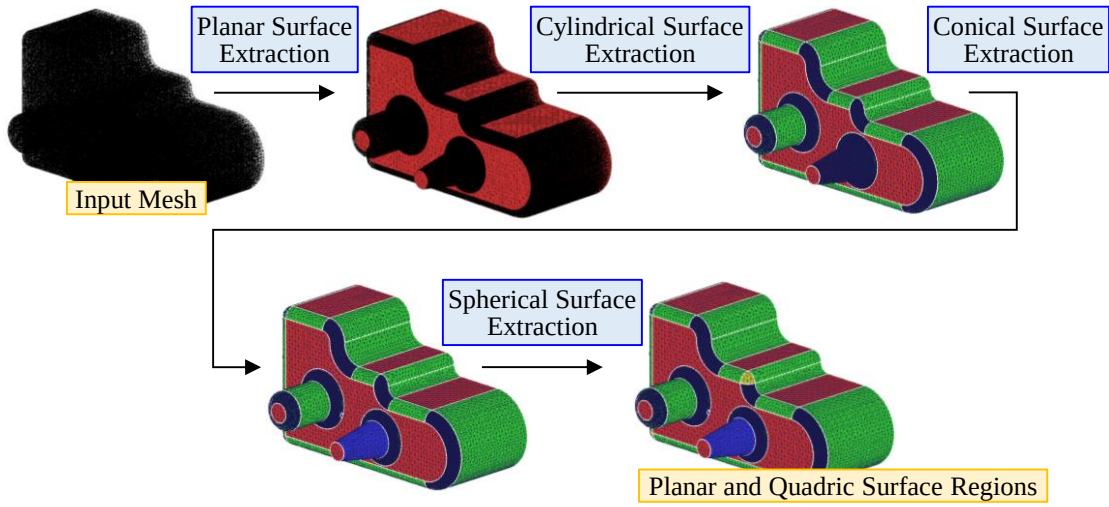


Fig. 2.7 The sequential extraction of planar and quadric surface regions

2.4.3 Planar and quadric surface region extraction

2.4.3.1 Overview of planar and quadric surface region extraction

As shown in Fig. 2.7, planar, cylindrical, conical and spherical surface regions are sequentially extracted. The planar and quadric surface region extraction consists of the following two steps.

- Step 1 *Initial seed region extraction*: the initial seed regions for surface fitting and region growing performed in the Step 2 are extracted by using features of each surface.
- Step 2 *Region integration*: surface regions are integrated and expand by iteration of surface fitting [Schnabel 2007] and region growing based on the distances between the fitting surfaces and vertices.

In this subsection, the detail of each step is described.

2.4.3.2 Initial seed region extraction

In order to find initial seed regions, the following features of each surface are used (Fig. 2.8).

- (a) The variance of unit normal vectors becomes 0 on a plane.
- (b) The variance of principal directions of the minimum curvature becomes 0 on a cylinder.
- (c) Principal directions of the minimum curvature are parallel to generatrices on a cone.
- (d) There is no consistency in the principal directions on a sphere.

Let a variance σ_ϵ^2 of vectors $\boldsymbol{v}$ of elements ϵ such as vertices or triangles is defined as Eq. (2.6):

$$\sigma_\epsilon^2 = \left(\sum_{\eta \in N_\epsilon} \|\mathbf{v}_\eta - \overline{\mathbf{v}_\epsilon}\|^2 \right) / |N_\epsilon|, \quad (2.6)$$

where N_ϵ is a set of neighboring elements of ϵ (including ϵ), $\overline{\mathbf{v}_\epsilon}$ normalized sum of the vectors of elements η .

Planar initial seed region extraction:

Planar initial seed regions are extracted by region-growing. In this region-growing, the seed triangle is first determined as a triangle which has two or more vertices whose variances of unit normal vectors are 0. Then, neighboring triangles are added to the initial seed regions iteratively if they have two or more vertices whose unit normal vector variances are 0.

Cylindrical and conical initial seed region extraction:

Cylindrical or conical initial seed regions are also extracted by region-growing. In this region-growing, the seed triangle is first determined as a triangle whose principal direction variance is less than a threshold. Then, neighbor triangles are added to the initial seed regions if they satisfy following two conditions for two neighboring triangles.

- The angle between principal directions at them is less than a threshold.
- They have the same signs of maximum principal curvatures.

In the conical initial seed region extraction, an angle threshold smaller than one in the cylindrical initial region extraction is used. For example, if the angle threshold for cylindrical surface was 1.0~1.5 degree, the angle threshold for conical surface was set to 0.4~0.8 degree in experiments in this thesis.

Spherical initial seed region extraction:

A triangle whose maximum difference of principal direction between its vertices is larger than the threshold are extracted as a spherical initial seed region. In this thesis, the threshold was set to 60 degree.

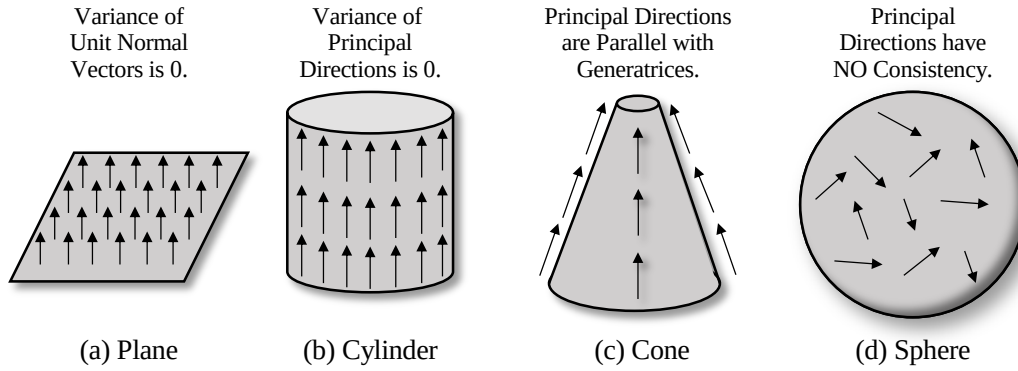


Fig. 2.8 Feature of each surface

2.4.3.3 Region integration

In this step, surface fitting is first applied to each seed region. Secondly, a set of triangles whose distance from the fitting surface is smaller than a given threshold are extracted by region-growing. To obtain a more accurate segmentation result, the surface fitting and the region-growing are iteratively performed. The surface fitting methods are described as follows.

Planer surface region:

A plane is defined by the average of normal vectors of triangles and barycenter of the vertices in the surface region.

Cylindrical surface region (Fig. 2.9):

As shown in Fig. 2.9(a), to find the axis vector, normal vectors of vertices are first mapped into a Gaussian sphere, and then plane fitting is applied to the mapped normal vectors. The axis vector is obtained as the normal vector of the fitting plane. After that, as shown in Fig. 2.9(b), the vertices are projected onto a plane P_{cyl} whose normal vector is the axis vector. Finally, as shown in Fig. 2.9(c) the radius and a point on the axis are calculated by least square circle fitting for the projected vertices in the surface region.

After the region growing, extracted surface regions are checked whether they are cylindrical surface regions or not in order not to recognize conical surface regions as cylindrical surface regions. In this classification, normal vectors of the vertices included in the surface region are mapped into a Gaussian sphere, and plane fitting is applied to images of the normal vectors. If the fitting plane passes the center of the Gaussian sphere, the surface seed region is recognized as a cylindrical surface region. Otherwise, the surface region is rejected.

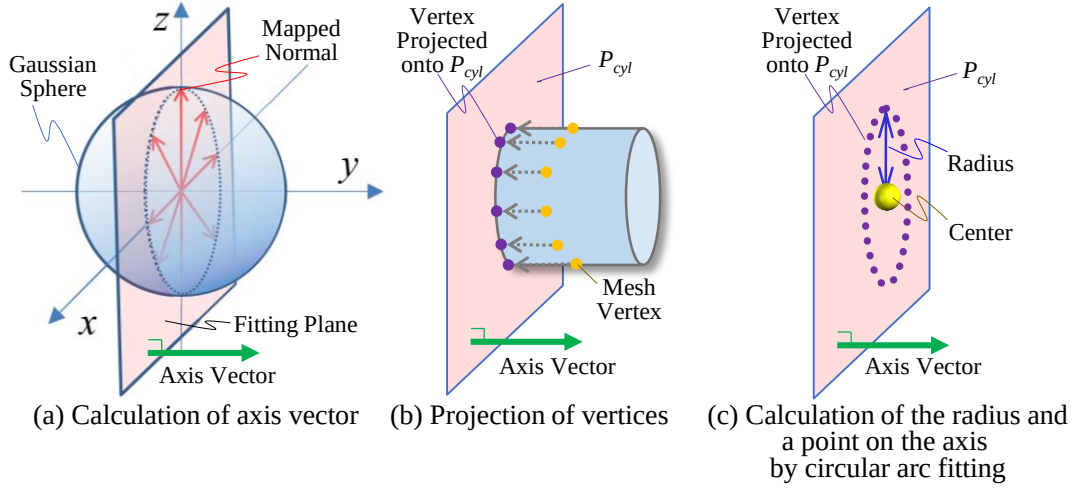


Fig. 2.9 Cylindrical surface fitting

Conical surface region:

After the conical initial seed region extraction, a conical surface region is divided by some strip of initial seed regions. Therefore, the surface fitting becomes unstable if only one initial seed region is used. To solve this problem for the initial fitting, the fitting based on Random Sample Consensus (RANSAC) [Schnabel 2007] using some initial seed regions is performed. After that, a cone is fitted by RANSAC using the vertices in the surface region. The conical surface fitting based on RANSAC consists of the following six steps (Fig. 2.10).

Step 1: Three vertices (i , j and k) are randomly selected. Their positions are described by $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{x}_k$, respectively.

Step 2: The apex position $\mathbf{p}_{cone}$ of the cone is estimated as an intersection of three planes which are defined by normal vectors and positions of the selected vertices (Fig. 2.10(a)).

Step 3: The axis vector $\mathbf{a}_{cone}$ is estimated as the normal vector of a plane defined by three points

$$\left\{ \mathbf{p}_{cone} + \frac{\mathbf{x}_i - \mathbf{p}_{cone}}{\|\mathbf{x}_i - \mathbf{p}_{cone}\|}, \mathbf{p}_{cone} + \frac{\mathbf{x}_j - \mathbf{p}_{cone}}{\|\mathbf{x}_j - \mathbf{p}_{cone}\|}, \mathbf{p}_{cone} + \frac{\mathbf{x}_k - \mathbf{p}_{cone}}{\|\mathbf{x}_k - \mathbf{p}_{cone}\|} \right\} \quad (\text{Fig. 2.10(b)}).$$

Step 4: As shown in Fig. 2.10(c), the opening angle φ_{cone} is calculated by Eq. (2.7):

$$\sum_{l \in \{i, j, k\}} \frac{\cos^{-1}\{(\mathbf{x}_l - \mathbf{p}_{cone}) \cdot \mathbf{a}_{cone}\}}{3}. \quad (2.7)$$

Step 5: The number of vertices whose distances from the fitting conical surface are smaller than the threshold is counted as the consensus of the model.

Step 6: The above five steps are iteratively performed. Next, the conical surface containing the largest number of vertices is adopted. However, in order to avoid inappropriate fitting, the conical surface whose opening angle is larger than a threshold (80 degree w used in this

thesis) is rejected.

Spherical surface region:

For the initial fitting, a center of the sphere is found as an intersection point of straight lines defined by the normal vectors and the positions of the three vertices. After that, a sphere is fitted by RANSAC using the vertices in the surface region. The spherical surface fitting based on RANSAC consists of the following four steps (Fig. 2.11).

- Step 1: Three vertices are randomly selected.
- Step 2: The center of the sphere is estimated as an intersection of three straight lines which are defined by normal vectors and positions of the selected vertices.
- Step 3: The radius is estimated as the average of distances between the center of the sphere and the selected vertices.
- Step 4: The above three steps are iteratively performed. Next, the spherical surface containing the largest number of vertices is adopted.

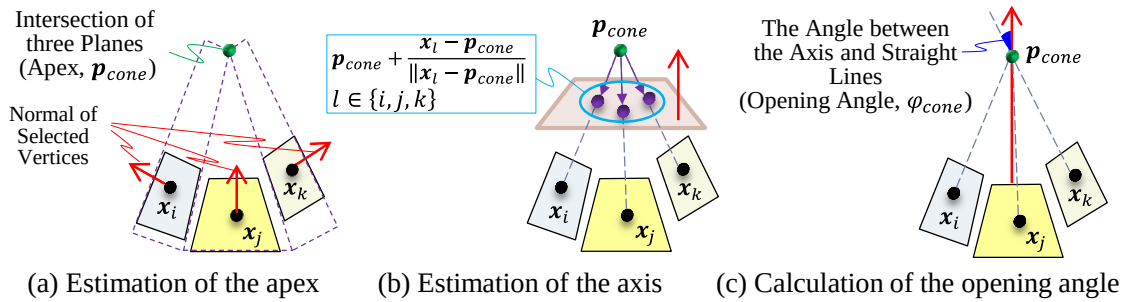


Fig. 2.10 Conical surface fitting based on RANSAC

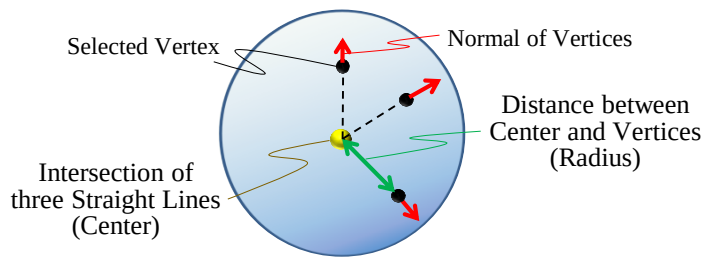


Fig. 2.11 Spherical surface fitting based on RANSAC

2.4.4 Torus surface region extraction

The remaining surface regions after the planar and quadric surface region extraction are torus surface regions in the problem setting of this thesis. Because the signs of Gaussian curvatures at two neighboring torus surface regions are different in most cases (see Fig. 2.12), the boundary edges between two neighboring torus surface regions are extracted by region growing based on the signs of the Gaussian curvatures.

The resulting torus surface regions may be over-segmented (Fig. 2.13(a)), and sometimes small incorrect cylindrical surface regions are extracted on the torus (Fig. 2.13(b)). Therefore, under the assumption that the input tetrahedral mesh has enough triangles in each surface region, the following two types of surface regions are re-recognized as a part of the neighboring torus surface region, having the longest boundary edge with a small over-segmented surface region or an incorrect cylindrical surface region.

Type 1: All vertices of the surface region exist on the boundary edges (Fig. 2.13(a)).

Type 2: The surface region shares a boundary edge having an acute angle with one surface region (Fig. 2.13(b)).

After that, for more accurate segmentation, the boundary edges of the torus surface regions are re-estimated using boundary triangles of the neighbor surface region. In this reevaluation, at first, the torus fitting is applied to the torus surface region. Then, by comparing distances from the fitting torus surface and the other fitting surface of neighbor surface region, boundary triangles of the neighbor surface region are added to the region having the small distance. In this process, a RANSAC-based torus surface fitting by the following six steps (Fig. 2.14) is used.

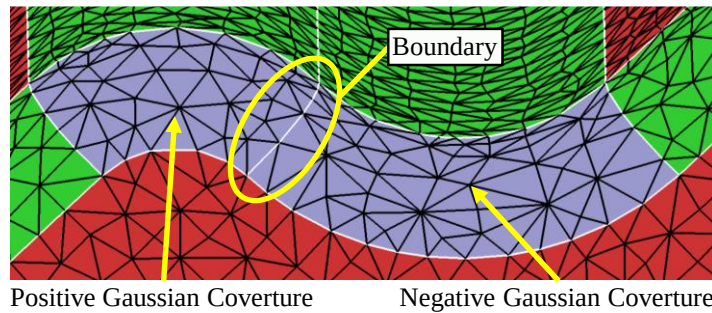
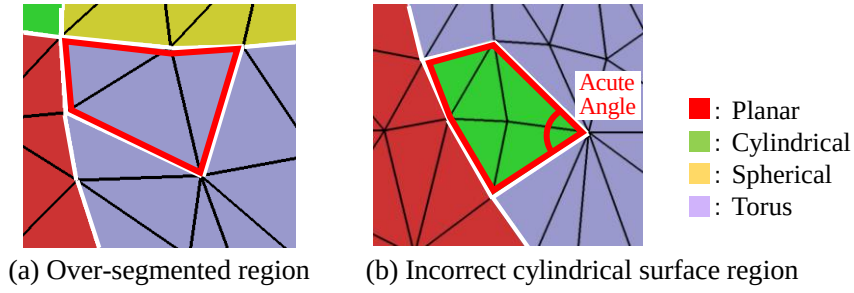
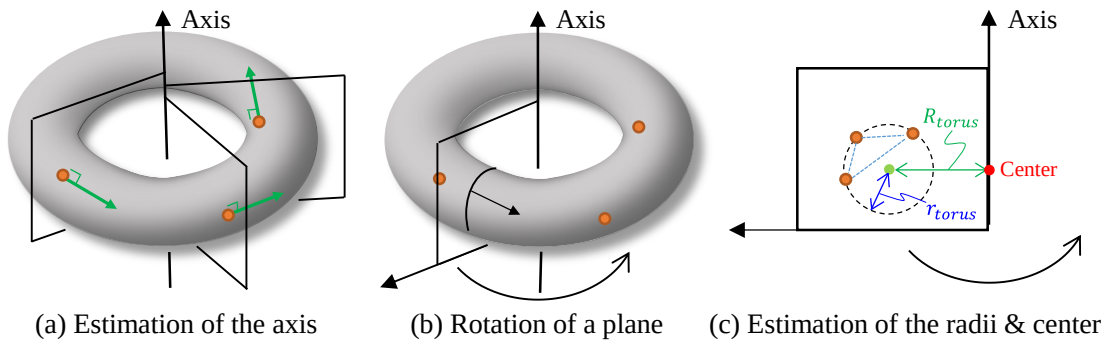


Fig. 2.12 Two neighboring torus surface regions with different signs of Gaussian curvatures

**Fig. 2.13 False recognition surface regions****Fig. 2.14 Torus surface fitting based on RANSAC**

- Step 1: Three vertices are randomly selected.
- Step 2: The axis of the torus is estimated as an intersection of planes which are defined by principal directions and positions of the selected vertices (Fig. 2.14(a)).
- Step 3: A plane that includes an axis is rotated around the axis, and the selected vertices are plotted on the plane (Fig. 2.14(b)).
- Step 4: The circumcenter of the plotted vertices are calculated. The minor radius r_{torus} and major radius R_{torus} of the torus are calculated as the circumradius and the distance between the circumcenter and the axis, respectively. Moreover, the center of the torus is extracted as a point on the axis closest to the circumcenter (Fig. 2.14(c)).
- Step 5: The number of vertices whose distances from the fitting torus surface are smaller than the threshold is counted.
- Step 6: The above five steps are iteratively performed. Next, the torus containing the largest number of vertices is adopted.

Finally, some small spherical surface regions such as junctions of fillet surfaces are extracted as torus regions because of the lack of triangles. In general, a torus surface region cannot connect with only three cylindrical surface regions with G^1 continuities. Therefore, if a torus surface region connects with only three cylindrical surface regions with G^1 continuities, it is re-recognized as a

spherical surface region.

2.4.5 Calculation of surface parameters

Obtained parameters of conical, spherical, and torus surface regions are not accurate because they are calculated by only rough surface fitting based on RANSAC. In order to obtain the accurate parameters of each surface region, the non-linear least square fitting based on the Levenberg-Marquardt (LM) method [Shakarji 1998] is applied to each conical, spherical, and torus surface region. LM method is used to find the local minimum of an objective function of a non-linear least square problem. LM method behaves like the Gauss-Newton method when the solution is close to the minimum of the objective function, and behaves like the gradient descent method when the solution is far from the minimum of the objective function. In the surface fitting based on LM method, a vector $\mathbf{p}^{LM}$ consisting of control variables (i.e. surface parameters) is found by minimizing an objective function. As shown in Eq. (2.8), the objective function $J^{LM}(\mathbf{p}^{LM})$ is the sum of square distances $d_i^{fit}(\mathbf{p}^{LM})$ between the fitted surface and each vertex i of the surface region:

$$J^{LM}(\mathbf{p}^{LM}) = \sum_{i \in V_S} d_i^{fit}(\mathbf{p}^{LM}), \quad (2.8)$$

where V_S is a set of vertices on the surface region. The pseudo code of the surface fitting based on LM method is shown in Fig. 2.15.

The rest of this subsection, the vector $\mathbf{p}^{LM}$ (see Fig. 2.16), the normalization of $\mathbf{p}^{LM}$, the distance equation $d_i^{fit}(\mathbf{p}^{LM})$, and the objective function $J^{LM}(\mathbf{p}^{LM})$ for each surface are described.

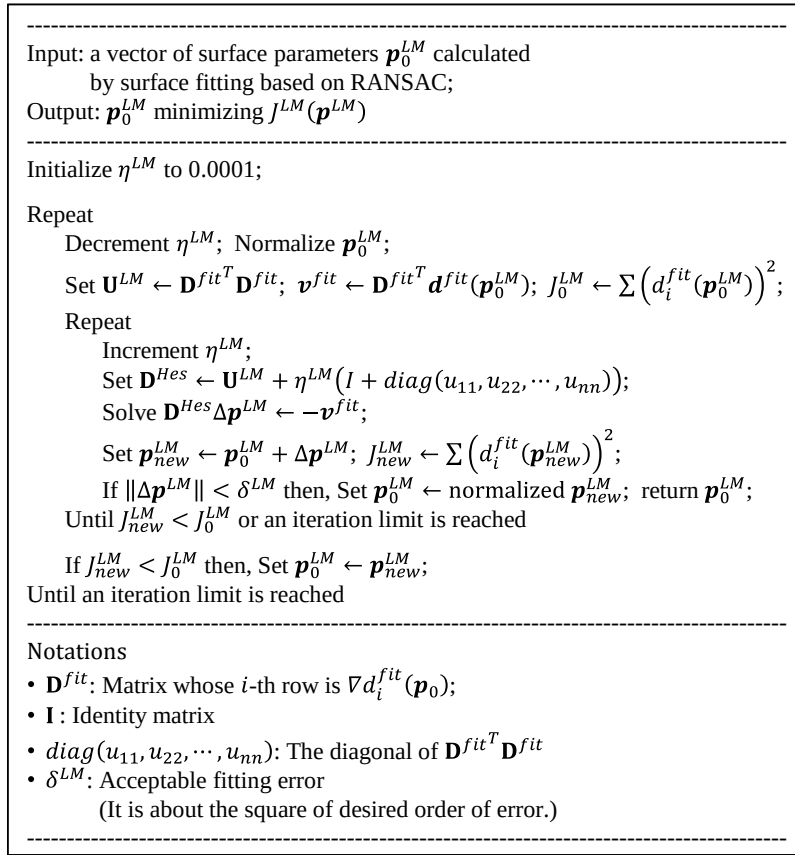


Fig. 2.15 The pseudo code of the surface fitting based on LM method

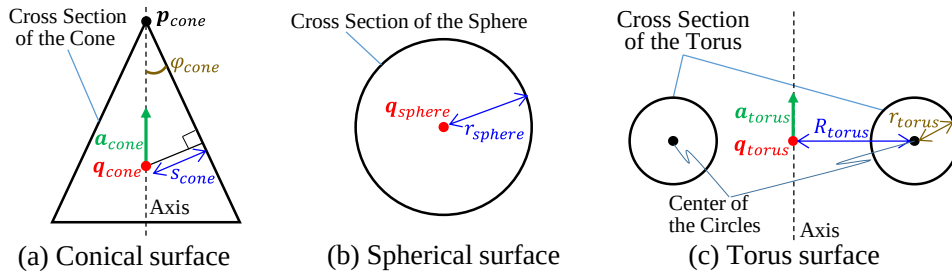


Fig. 2.16 Control variables of each surface in LM method

Conical surface region:

$\mathbf{p}^{LM} := (\mathbf{q}_{cone}, \mathbf{a}_{cone}, s_{cone}, \varphi_{cone})$ (Fig. 2.16(a)).

$\mathbf{q}_{cone}$: a point on the axis (not the apex),

$\mathbf{a}_{cone}$: the axis vector pointing toward the apex from $\mathbf{q}_{cone}$,

s_{cone} : the orthogonal distance between $\mathbf{q}_{cone}$ and the cone,

φ_{cone} : the opening angle of the cone.

Normalization of $\mathbf{p}^{LM}$: $\mathbf{a}_{cone} \leftarrow \mathbf{a}_{cone} / \|\mathbf{a}_{cone}\|$,
 $\mathbf{q}_{cone} \leftarrow$ (a point on the axis closest to origin),
 $\varphi_{cone} \leftarrow \varphi_{cone} \pmod{2\pi}$,
 if $\varphi_{cone} > \pi$ then $[\varphi_{cone} \leftarrow \varphi_{cone} \pmod{\pi}; \mathbf{a}_{cone} \leftarrow -\mathbf{a}_{cone}]$,
 if $\varphi_{cone} > \frac{\pi}{2}$ then $[\varphi_{cone} \leftarrow \pi - \varphi_{cone}]$,
 if $s_{cone} < 0$ then $[s_{cone} \leftarrow -s_{cone}; \mathbf{a}_{cone} \leftarrow -\mathbf{a}_{cone}]$.

$d_i^{fit}(\mathbf{p}^{LM}) := f_i^{LM} \cos \varphi_{cone} + g_i^{LM} \sin \varphi_{cone} - s_{cone}$.
 f_i^{LM} : the distance between the axis and a vertex i ,
 g_i^{LM} : the distance between a vertex i and the plane defined by $\mathbf{q}_{cone}$ and $\mathbf{a}_{cone}$.

$$J^{LM}(\mathbf{p}^{LM}) := \sum_{i \in V_S} (f_i^{LM} \cos \varphi_{cone} + g_i^{LM} \sin \varphi_{cone} - s_{cone})^2.$$

Spherical surface region:

$\mathbf{p}^{LM} := (\mathbf{q}_{sphere}, r_{sphere})$ (Fig. 2.16(b)).
 $\mathbf{q}_{sphere}$: the center of the sphere, r_{sphere} : the radius of the sphere.

Normalization of $\mathbf{p}^{LM}$: None.

$d_i^{fit}(\mathbf{p}^{LM}) := \|\mathbf{x}_i - \mathbf{q}_{sphere}\| - r_{sphere}$.
 $\mathbf{x}_i$: the position of a vertex i .

$$J^{LM}(\mathbf{p}^{LM}) := \sum_{i \in V_S} (\|\mathbf{x}_i - \mathbf{q}_{sphere}\| - r_{sphere})^2.$$

Torus surface region:

$\mathbf{p}^{LM} := (\mathbf{q}_{torus}, \mathbf{a}_{torus}, R_{torus}, r_{torus})$ (Fig. 2.16(c)).
 $\mathbf{q}_{torus}$: the center of the torus, $\mathbf{a}_{torus}$: the axis vector of the torus,
 R_{torus} : the major radius, r_{torus} : the minor radius.

Normalization of $\mathbf{p}^{LM}$: $\mathbf{a}_{torus} \leftarrow \mathbf{a}_{torus} / \|\mathbf{a}_{torus}\|$.

$d_i^{fit}(\mathbf{p}^{LM}) := \sqrt{g_i^{LM^2} + (f_i^{LM} - R_{torus})^2} - r_{torus}$.
 f_i^{LM} : the distance between the axis and a vertex i ,
 g_i^{LM} : the distance between a vertex i and the plane defined by $\mathbf{q}_{torus}$ and $\mathbf{a}_{torus}$.

$$J^{LM}(\mathbf{p}^{LM}) := \sum_{i \in V_S} \left(\sqrt{g_i^{LM^2} + (f_i^{LM} - R_{torus})^2} - r_{torus} \right)^2.$$

2.4.6 Boundary edge recognition

For the following deformation (A2) and quality improvement (A3), the boundary edge is segmented. At first, the vertices having more than two edges on the boundary edge are identified as a feature vertex. Secondly, a boundary edge segment is defined as the connected sequence of edges on the boundary edge between two feature vertices or on the loop boundary edge. Then, each boundary edge segment is classified into a straight line, circle, circular arc, ellipse, elliptic arc, parabola, or hyperbolic curve by following steps.

The classification of a boundary edge segment between two feature vertices:

- Step 1: A straight line is fitted to the boundary edge segment. If the max distance between fitted line and vertices is lower than a threshold, the boundary edge segment is recognized as a straight line segment.
- Step 2: A plane is fitted to the boundary edge segment. If the maximum distance between fitted plane and vertices is lower than a threshold, the boundary edge segment is recognized as a plane curve segment. Otherwise, the boundary edge segment is recognized as a non-plane curve segment (*unclassified curve segment*).
- Step 3: The plane curve segment is classified into a circular arc, elliptic arc, parabola, or hyperbolic curve segment by following two types of classifications.
 - (a) For the plane curve segment between a torus surface region and others, a circular arc is fitted by RANSAC. If the maximum distance between the fitted circular arc and vertices is lower than a threshold, the boundary edge segment is recognized as a circular arc segment. Otherwise, it is recognized as an *unclassified curve segment*.
 - (b) The other plane curve segments are classified into circular arc, elliptic arc, parabola, or hyperbolic curve segments according to the types of surface region intersections.

The classification of a loop boundary edge segment:

- Step 1: A plane is fitted to the boundary edge segment. If the maximum distance between fitted plane and vertices is lower than a threshold, the boundary edge segment is recognized as a plane curve segment. Otherwise, the boundary edge segment is recognized as a non-plane curve segment (*unclassified curve segment*).
- Step 2: The plane curve segment is classified into a circle or ellipse segment by following two types of classifications.
 - (a) For the plane curve segment between a torus surface region and others, a circle is fitted by RANSAC. If the maximum distance between the fitted circle and vertices is lower than a threshold, the boundary edge segment is recognized as a circle segment.
 - (b) The other plane curve segments are classified into circle or ellipse segments

according to the types of surface region intersections.

Finally, the parameters of each boundary edge segment are calculated by curve fitting.

2.5 Dimension-driven shape deformation based on surface information and space embedding

2.5.1 Overview of dimension-driven shape deformation

2.5.1.1 Assumptions

In this section, a dimension-driven shape deformation method for tetrahedral meshes is proposed. In the proposed deformation method, as shown in Fig. 2.17, it is assumed that changes of dimensions are performed without self-intersections and degenerations of surface regions while preserving the consistency of the curved surface. In order to keep the consistency of the curved surfaces during the deformation, surface regions extracted by segmentation are classified into the following four regions in the proposed deformation method.

- (a) *Control region*: a surface region defining the dimension and changed to satisfy the target dimension.
- (b) *Following region*: a surface region corresponding to subsidiary surfaces such as fillets or chamfers of the *control region*.
- (c) *Deformed region*: a surface region deformed by the change of the *control region* and *following regions*.
- (d) *Fixed region*: a surface region unchanged before and after deformation.

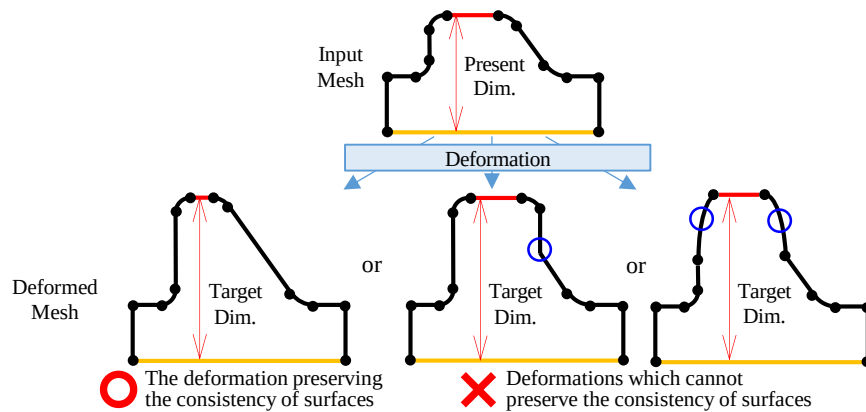


Fig. 2.17 Deformation preserving the consistency of the curved surface

In this thesis, surface information describes the following parameters, the adjacent boundary edge segments, and the adjacent surface regions of each surface region. In addition, the boundary edge information describes the following parameters and adjacent surface regions of each boundary edge segment.

Parameters of each surface region:

- Planar surface region: a point on the plane, and the normal vector.
- Cylindrical surface region: the axis vector, a point on the axis, the radius, and a label describing whether the surface region is convex or concave.
- Conical surface region: the axis vector, the apex, the opening angle, and a label describing whether the surface region is convex or concave.
- Spherical surface region: the center, the radius, and a label describing whether the surface region is convex or concave.
- Torus surface region: the center, the axis vector, the major radius, the minor radius, projected vertices onto the core curve, and a label describing whether the core curve is outside or inside of the given tetrahedral mesh.

Parameters of each boundary edge segment:

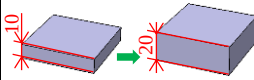
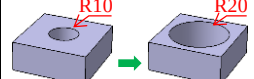
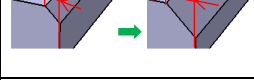
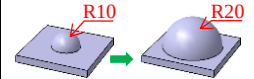
- Straight line segment: the end points (two feature vertices).
- Circle segment: the center, the radius, and the normal vector of a plane including the circle segment.
- Circular arc segment: the end points (two feature vertices), the center, the radius, and the normal vector of a plane including the circular arc.
- Other segment (ellipse, ellipse arc, parabola, and hyperbolic curve segment): the normal vector of a plane including the segment.

2.5.1.2 Deformation types and user inputs

The proposed deformation method enables us to change the following seven types of dimensions of mesh models (see Table 2.7): (1) the height of the boss and the depth of the hole; (2) the radius of the cylindrical boss and the circular hole; (3) the radius of the fillet; (4) the angle of the chamfer and the taper; (5) the length of the chamfer; (6) the radius of the spherical surface; and (7) the position of the local object on planes (such as bosses and ribs). However, the dimensions (3) (4) (5) cannot be changed if a surface region defining the dimension connects with another fillet or chamfer surfaces. In addition, topology of overall of mesh models cannot be changed by the proposed method e.g. a blind hole cannot be a through hole by changing its depth.

Note that these deformation types are defined for explanation. Indeed, in the proposed method, surface parameters are modified as shown in the third column of Table 2.7. Therefore, types of form features are not recognized in the proposed method.

Table 2.7 Deformation type and user inputs

Dimension (deformation type)		Modified surface/object parameter	User inputs { CR: Control Region RD: Reference for dimension TD: Target dimension
(1) Height of the boss and depth of the hole		Position of the planar surface	CR: a planar surface region RD: a planar surface region parallel to CR TD: the distance between CR and RD
(2) Radius of the cylindrical boss and the circular hole		Radius of the cylindrical surface	CR: a cylindrical surface region RD: none TD: the radius
(3) Radius of the fillet		Radius of the cylindrical or the torus surface	CR: a cylindrical or a torus surface region of the fillet RD: none TD: the radius
(4) Angle of the chamfer and the taper		Angle of the planar surfaces or the apex angle of the conical surface	CR: a planar or a conical surface region of the chamfer or the taper RD: a surface region neighboring CR TD: the angle between CR and RD
(5) Length of the chamfer		Position of the planar or the conical surface	CR: a planar or a conical surface region of the chamfer or the taper RD: a surface region neighboring CR TD: the distance from an edge between CR and RD
(6) Radius of the sphere		Radius of the spherical surface	CR: a spherical surface region RD: none TD: the radius
(7) Position of the local object (boss, hole, rib, and slot)		Position of the representative point on the local coordinate system	CR: a surface region on the local object RD: planar surface regions and a local coordinate system TD: the position on the local coordinate system

As shown in Table 2.7, the user inputs of the proposed method are a control region, reference of dimension, and the target dimension. If the deformation type is (7) the position of the local object, the number of surface regions for the reference of the dimension is also inputted, and two or three vertices are specified by user in order to define the local coordinate system.

As shown in Fig. 2.18, changing a dimension is performed by following five steps.

- The deformation type, a *control region*, and the reference for the dimension are specified by user.
- Surface regions are classified into *control regions*, *following regions*, *deformed regions*, and *fixed regions* based on user inputs.
- A *deformable region* which is a set of tetrahedra deformed according to the target dimension is extracted and the current dimension is calculated.
- Target dimension is specified by user.
- Vertices included in *deformable region* moved according to the target dimension.

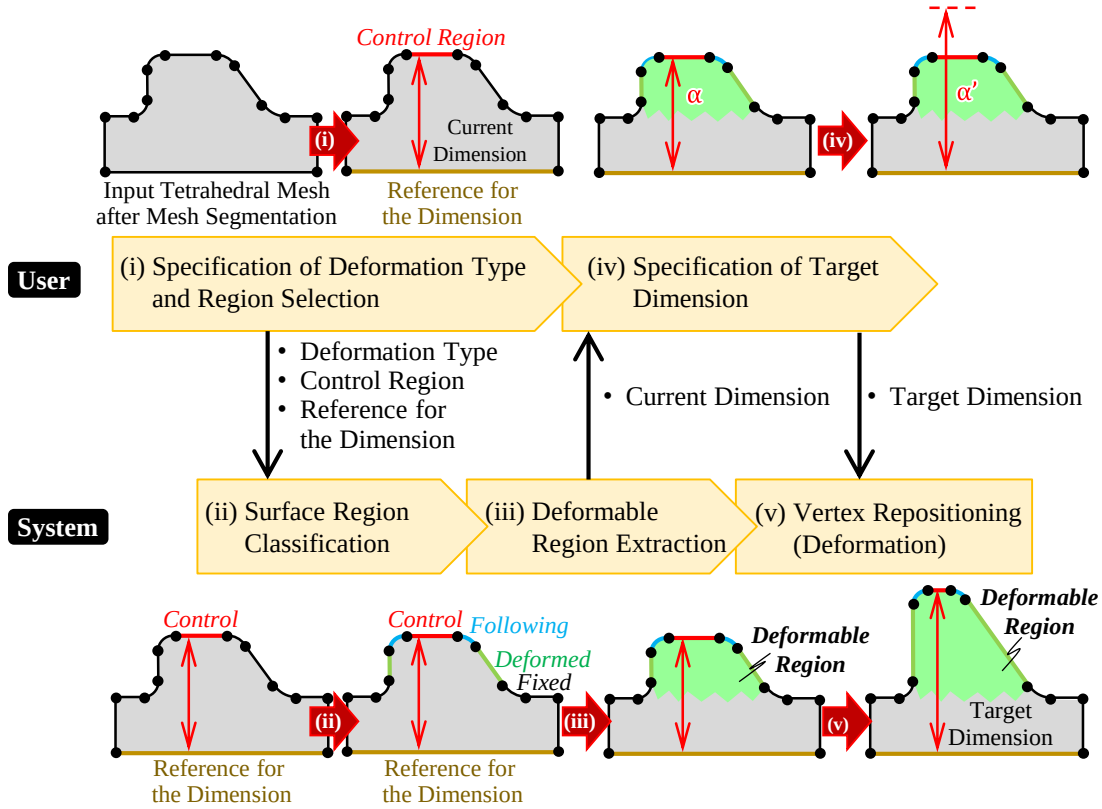


Fig. 2.18 The process of changing a dimension

2.5.1.3 Algorithm

The algorithm is shown in Fig. 2.19. In the proposed deformation method, at first, surface regions extracted by segmentation are classified into *control regions*, *following regions*, *deformed regions*, and *fixed regions* through user inputs (A21). Then, the *deformable region*, the local region of the given tetrahedral mesh (i.e. a set of tetrahedra) where the deformation and the quality improvement are performed, is extracted using the type of deformation and types of regions (A22). Finally, vertices are repositioned for the deformation (A23). In the vertex repositioning, at first, the positions of vertices in the *deformable region* are parameterized for the space embedding method. After that, vertices included in the *control region* and *following regions* are repositioned in order to satisfy the target dimension. Finally, vertices included in the *deformable region* are moved while preserving the consistency of the curved surface.

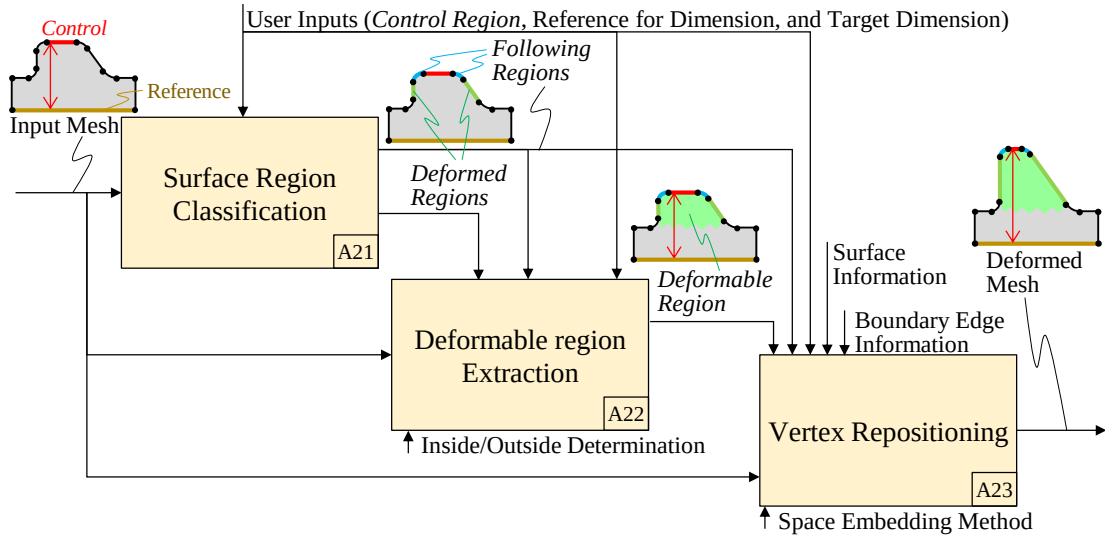


Fig 2.19 The overview of the dimension-driven shape deformation of tetrahedral meshes

2.5.2 Surface region classification

After user's specification of the deformation type and the *control region*, the other surface regions are classified into *following regions*, *deformed regions*, or *fixed regions*. As shown in Fig. 2.20, for the deformation type (1)-(6) (i.e. all deformations except for the position of the local object), the surface region classification is performed by following steps. First, cylindrical and torus surface regions are classified into the *following regions* as fillet surfaces if they connect with the *control region* with G^1 continuity, and their neighbor spherical surface regions are also classified into the *following regions* as fillet surfaces if their radii are equal to the radius of the fillet surfaces (Fig. 2.20(b)). Second, planar and conical surface regions whose area ratios to the area of the *control region* are smaller than a given threshold are classified into the *following regions* as chamfer surfaces (Fig. 2.20(c)). Then, neighbor surface regions of the *control region* and *following regions* are classified into the *deformed regions*. The others are classified into the *fixed regions* (Fig. 2.20(d)). Finally, if the classification result are incorrect, *following regions* can be re-selected manually, and the *deformed regions* and the *fixed regions* are re-recognized based on them.

On the other hand, as shown in Fig. 2.21, for the deformation type (7) the position of the local object, at first, surface regions specified by user as the reference of the dimension are classified into *deformed regions* (Fig. 2.21(a)). Then, if the number of deformed region is two and there are cylindrical surface regions connecting with both of deformed regions with G^1 continuity, the cylindrical regions are classified into the *deformed regions* as fillet surfaces (Fig. 2.21(b)). Finally, surface regions between the *control region* specified by user and the *deformed regions* are classified into *control regions* (Fig. 2.21(c)).

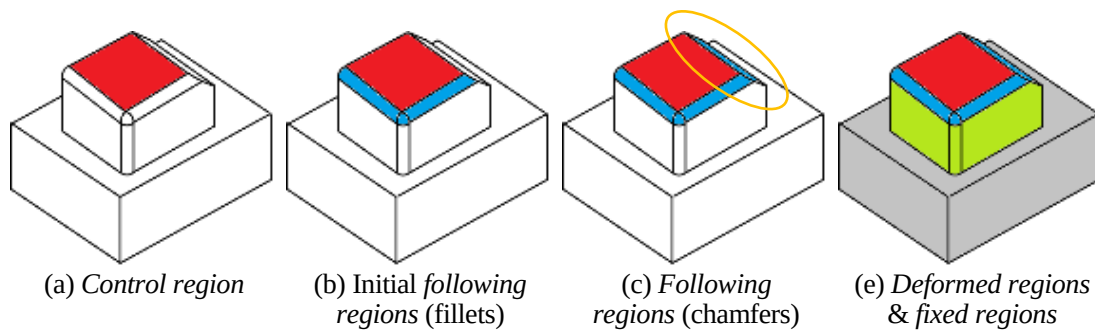


Fig. 2.20 Surface region classification for the deformation type (1)-(6)

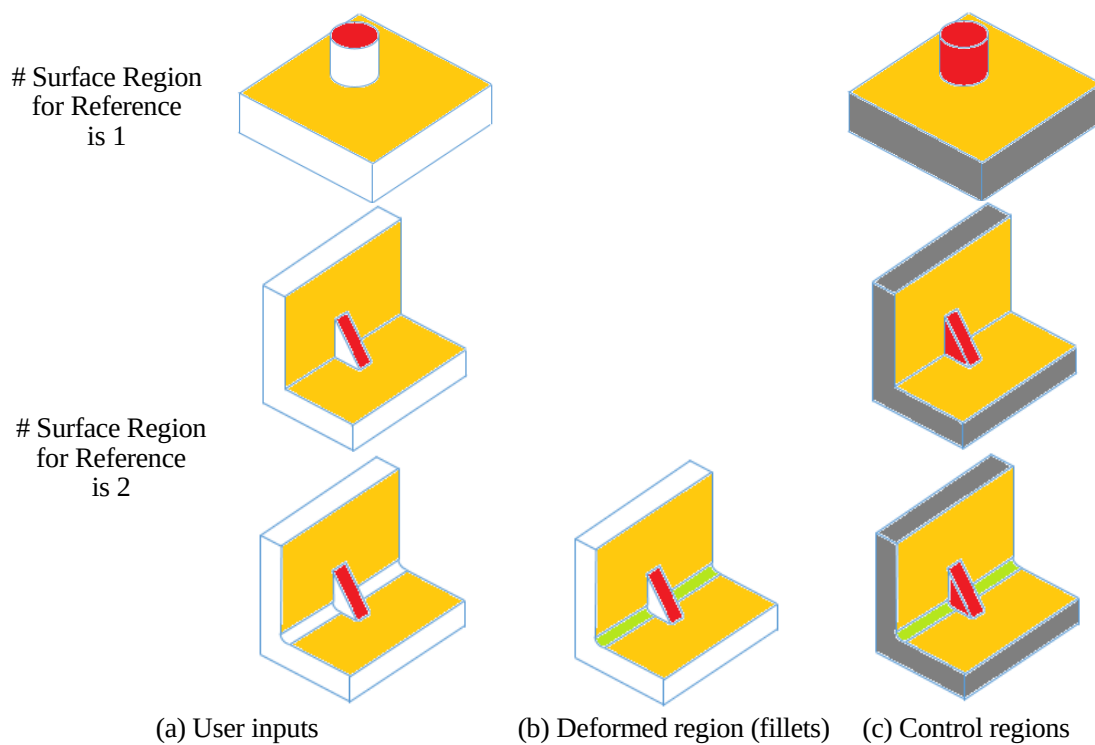


Fig. 2.21 Surface region classification for the deformation type (7)

2.5.3 Deformable region extraction

2.5.3.1 Overview of deformable region extraction

For more efficient deformation and quality improvement, deformation and quality improvement should be performed locally rather than globally. In the proposed method in this thesis, under the precondition that changes of dimensions are performed without self-intersections and degeneracies

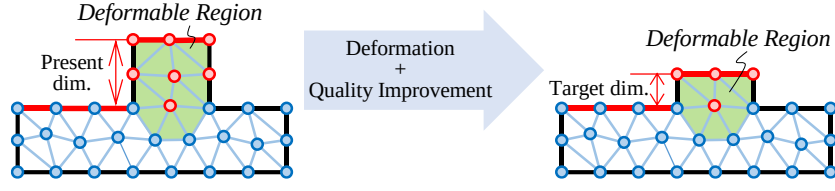


Fig. 2.22 Performing the deformation and quality improvement only on the *deformable region*

of surface regions, a set of tetrahedra which may change during the deformation are extracted as the *deformable region*. As shown in Fig. 2.22, the vertex repositioning of the deformation and the quality improvement described in section 2.6 are applied only to the *deformable region*. The procedure of the deformable region extraction has three types corresponding to the deformation type (3) the radius of the fillet, (7) the position of the local object, and the others. In this subsection, the extraction of the *deformable region* according to the deformation type is described.

2.5.3.2 Deformable region extraction for change of the radius of the fillet

At first, as shown in Fig. 2.23(a), two *deformed regions* S_{D1} and S_{D2} connecting with the *control region* with G^1 continuity are extracted. Secondly, the fillet surface F_{max} (and/or F_{min}) which has the largest (and/or smallest) radius without degenerating S_{D1} or S_{D2} is extracted (Fig. 2.23(b)). Thirdly, a set of vertices V_{in} of the *fixed regions* in a space surrounded by F_{max} (and/or F_{min}) and the curved surfaces of S_{D1} and S_{D2} is extracted. Then, as shown in Fig. 2.23(c), the fillet surface F_+ (and/or F_-) whose radius is the closest to current radius among fillet surfaces contacting with vertices included in V_{in} are extracted (F_+ is extracted by V_{in} obtained using F_{max} and F_- is extracted by V_{in} obtained using F_{min}). After that, a set of vertices V_{df} included in a space surrounded by F_+ (and/or F_-) and the curved surfaces of S_{D1} and S_{D2} is extracted (Fig. 2.23(d)). Finally, a set of tetrahedra with vertices included in V_{df} is extracted as the deformable region K_{df} (Fig. 2.23(e)).

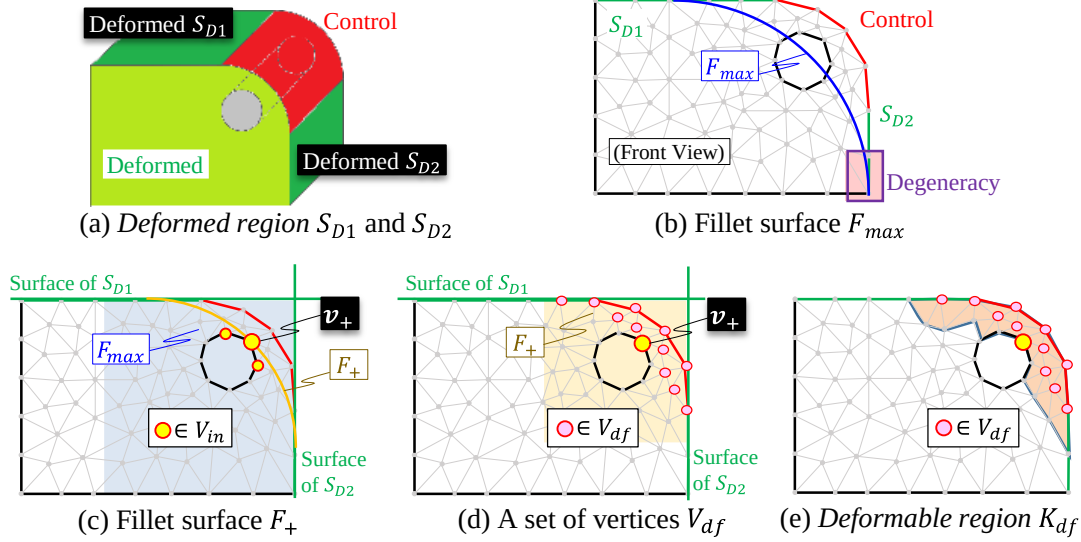


Fig. 2.23 Deformable region extraction for the deformation type (3) the radius of the fillet

2.5.3.3 Deformable region extraction for change of the position of the local object

Let O_m is the local object and F_{pl} is the plane where O_m is located. In the extraction of *deformable region*, as shown in Fig. 2.24(a), vertices of the *fixed regions* whose distance from F_{pl} is smaller than the height (or the depth) of O_m is first projected onto F_{pl} (a set of projected points is described by P_f). Then, O_m is also projected onto F_{pl} and the bounding box O_b (a square whose edge is parallel to the axis of the local coordinate system) of the projected O_m is extracted. After that, as shown in Fig. 2.24(b), a 2D regular grid whose axis is parallel to the local coordinate system and size of cells is a half of the edge length of O_b is generated on F_{pl} . After that, cells which include points of P_f are classified into *occupied* (Fig. 2.24(c)), and each cell which does not has any *occupied* cells between the barycenter of O_b and itself is extracted as a *visible* cell using ray tracing by Digital Differential Analyzer [Fujimoto 1986] (Fig. 2.24(d)). After the extraction of a set of *visible* cells, a set of grid points P_G whose all incident cells are *visible* cells is extracted, and the cells whose all grid points are included in P_G are classified into *movable* cells (Fig. 2.24(e)). Finally, a set of tetrahedra with vertices included in a space which is defined by a set of *movable* cells and the height (or the depth) of O_m is extracted as the *deformable region* K_{df} (Fig. 2.24(f)).

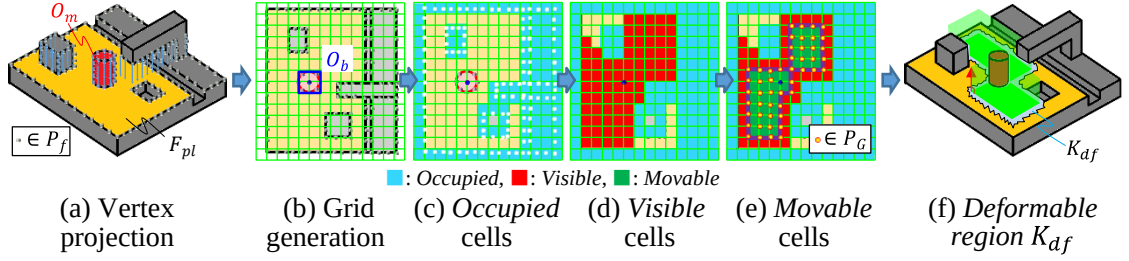


Fig. 2.24 Deformable region extraction for the deformation type (7) the position of the local object

2.5.3.4 Deformable region extraction for the other changes

Figure 2.25 shows the overview of the deformable region extraction for the deformation type (1) the depth of the hole. At first, a set of vertices V_{in} of the *fixed regions* which is included in a space defined by sweeping the *control region* and the *following regions* along the moving direction of the vertices. Then, as shown in Fig. 2.25(b), for upside and downside of the *control region*, vertices which is the closest to the *control region* in the V_{in} are extracted as *dimension range definition vertices* v_+ and v_- . After that, a set of vertices V_{df} included in a space which is defined by re-sweeping the *control region* and the *following regions* until contacting with v_+ and v_- is extracted (Fig. 2.25(c)). Finally, a set of tetrahedra with vertices included in V_{df} is extracted as the *deformable region* K_{df} (Fig. 2.25(d)).

In the deformable region extraction for the deformation type (4) the angle of the chamfer, the *control region* is swept along the normal of a surface region specified as the reference of the dimension instead of the moving direction of the vertices. On the other hand, in the deformable region extraction for the deformation type (2) the radius of the cylindrical boss and (6) the radius of the spherical surface, the *control region* is not swept but offset.

In addition, in the deformable region extraction for the case where the boundary shape between the *deformed regions* and a set of the *control region* and the *following regions* is changed by the deformation such as a change of the height of the cylindrical boss with the taper, the extraction of V_{in} and V_{df} is performed while considering the change of the boundary shape.

Moreover, the *deformable region* is expanded using neighbors of tetrahedra included in K_{df} , if only a few inner vertices which are not on the boundary between the *deformable region* and the other are included in the *deformable region*.

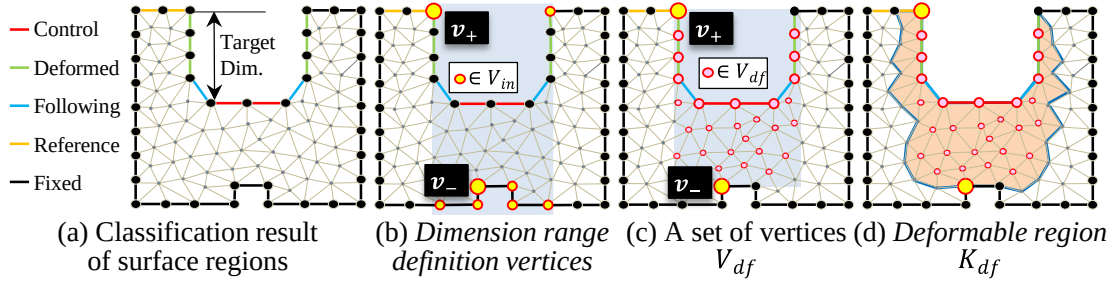


Fig. 2.25 Deformable region extraction for the deformation type (1) the depth of the hole

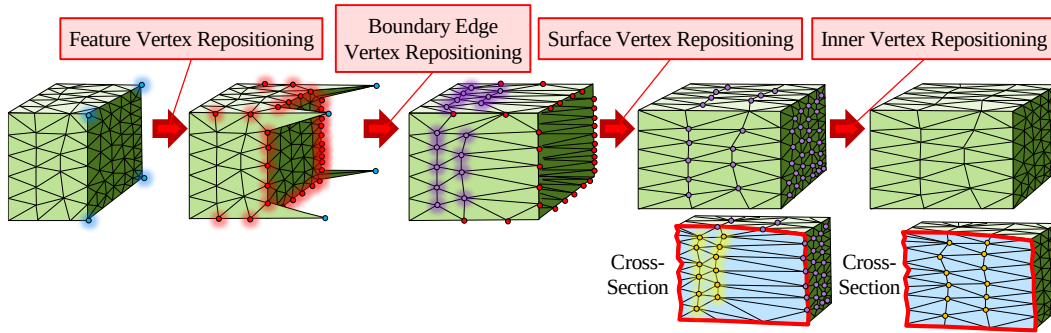


Fig. 2.26 The overview of the vertex repositioning

2.5.4 Vertex repositioning

2.5.4.1 Overview of vertex repositioning

In the vertex repositioning, vertices are moved by the following two methods.

Method 1: Movement onto boundary edges or surfaces updated to satisfy the target dimension.

Method 2: Space embedding method using parameterization of vertex positions.

As shown in Fig. 2.26, feature vertices that are intersection points of boundary edge segments are first moved by Method 1. Second, boundary edge vertices that are vertices on a boundary edge segment are moved by a combination of Methods 1 and 2. Then, surface vertices are also moved by a combination of Methods 1 and 2. Finally, inner vertices are moved by Method 2.

2.5.4.2 Space embedding method

Space embedding methods are useful for mesh deformation [Takano 2010, Xian 2009]. Figure 2.27 show the overview of the space embedding method. At first, the position $\mathbf{x}_i$ of each vertex i is represented using parameters ω_j for the positions $\mathbf{x}_j$ of the handle vertices $j \in H_i$ (H_i is a set of handle vertices of the vertex i) as shown in Eq. (2.9):

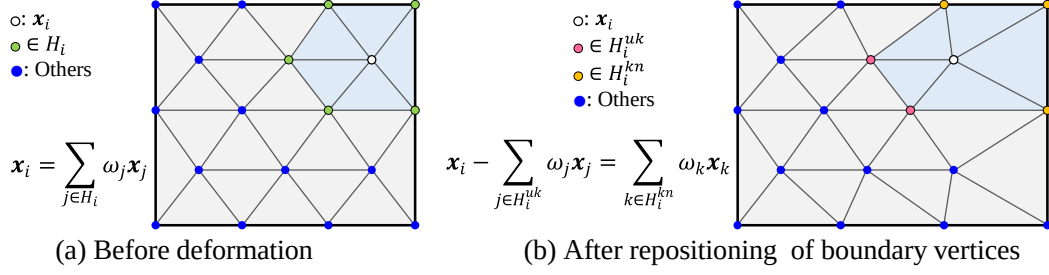


Fig. 2.27 The overview of the space embedding method

$$\mathbf{x}_i = \sum_{j \in H_i} \omega_j \mathbf{x}_j. \quad (2.9)$$

In the proposed deformation method, feature vertices, boundary edge vertices, surface vertices, and inner vertices are moved in that order. In each vertex movement step after the first one, the regions for vertex movement have a fixed boundary. Therefore, H_i can be divided into two sets of vertices whose positions are knowns H_i^{kn} and unknowns H_i^{uk} , and Eq. (2.9) can be represented by Eq. (2.10):

$$\mathbf{x}_i - \sum_{j \in H_i^{uk}} \omega_j \mathbf{x}_j = \sum_{k \in H_i^{kn}} \omega_k \mathbf{x}_k. \quad (2.10)$$

Then, using ω_j and ω_k which are preliminarily calculated, positions of unknown vertices are determined as a solution to a system of linear equations.

2.5.4.3 Parameterization of vertices

The handle vertices and the parameters of each vertex are defined as follows (see Fig. 2.28).

- *Boundary edge vertex*: The handle vertices are two end points of the boundary edge segment. The parameters are their internal division ratios.
- *Surface vertex*: The handle vertices are surface vertices in the 1-ring neighborhood. The parameters are Mean Value Coordinates (MVCs) [Floater 2003] for a polygon consisting of these vertices. The MVCs of each vertex is calculated on the development plane for the cylindrical or conical surface regions.
- *Inner vertex*: The handle vertices are vertices in the 1-ring neighborhood. The parameters are MVCs [Ju 2005] for a polyhedron consisting of these vertices.

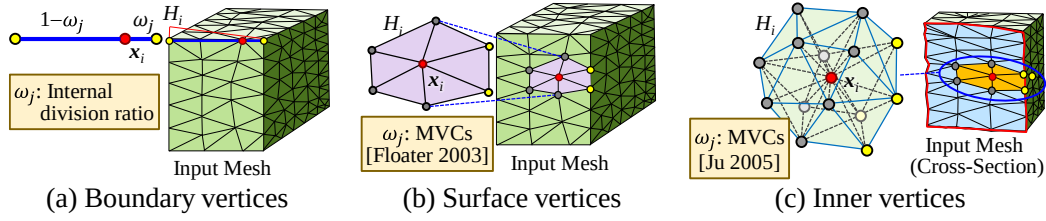


Fig. 2.28 Handle vertices and parameters of each vertex

2.5.4.4 Mean Value Coordinates

In the proposed deformation method, surface vertices and inner vertices are parameterized using Mean Value Coordinates (MVCs) which is proposed by Floater et al. [2003] for the smooth interpolation of the inside of a polygon. Floater et al. [2005] generalized MVCs from closed 2D polygons to closed 3D polyhedral, and Ju et al. [2005] also extended MVCs for a robust interpolation of the inside of closed 3D triangular meshes. In the field of computer graphics, MVCs is widely used such as shading, parameterization, and deformation of mesh models. MVCs is calculated as follows.

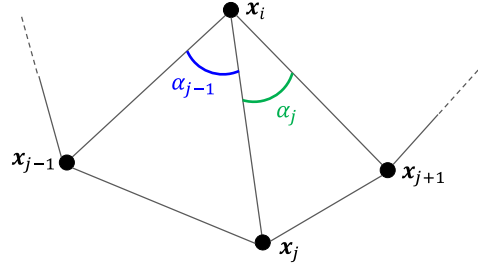
MVCs in the 2D space [Floater 2003]:

For a point i located inside of any closed polygon whose number of vertices is m , the weight ω'_j of each vertex j ($j = 1, 2, \dots, m$) of the closed polygon is defined as Eq. (2.11):

$$\omega'_j = \frac{\tan\left(\frac{\alpha_{j-1}}{2}\right) + \tan\left(\frac{\alpha_j}{2}\right)}{\|x_j - x_i\|}, \quad (2.11)$$

where x_j and x_i is positions of j and i respectively, and α_j is the angle formed by $x_j - x_i$ and $x_{j+1} - x_i$ (see Fig. 2.29). The weight ω_j in the MVCs of i for the polygon is calculated by Eq. (2.12):

$$\omega_j = \frac{\omega'_j}{\sum_{j=1}^m \omega'_j}. \quad (2.12)$$

Fig. 2.29 The angle α_j

MVCs in the 3D space [Floater 2005, Ju 2005]:

Let T_{cl} is a closed triangular mesh, j ($j = 1, 2, \dots, m$) each vertex of T_{cl} , and i is a point located inside of T_{cl} . The weight ω_j in the MVCs of i for T_{con} is obtained by mapping T_{cl} onto a unit sphere whose center is i (the mapped T_{cl} is described by $\tilde{T}_{cl}$). Because T_{cl} is a closed triangular mesh, $\tilde{T}_{cl}$ covers whole of the unit sphere. Hence, since the integral of unit normal vectors on the whole of a sphere becomes 0, Eq. (2.13) can be obtained:

$$\int_{\tilde{T}_{cl}} \text{sign}(\tilde{\mathbf{n}}_j \cdot (\mathbf{x}_j - \mathbf{x}_i)) \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\tilde{T}_{cl} = 0, \quad (2.13)$$

where $\mathbf{x}_j$ and $\mathbf{x}_i$ is positions of j and i respectively, $\text{sign}(\cdot)$ the sign of the scalar value in the bracket, and $\tilde{\mathbf{n}}_j$ is the normal vector of j on the unit sphere. Then, Eq. (2.14) is obtained by solving Eq. (2.13) in terms of $\mathbf{x}_i$:

$$\mathbf{x}_i = \frac{\int_{\tilde{T}_{cl}} \text{sign}(\tilde{\mathbf{n}}_j \cdot (\mathbf{x}_j - \mathbf{x}_i)) \frac{\mathbf{x}_j}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\tilde{T}_{cl}}{\int_{\tilde{T}_{cl}} \text{sign}(\tilde{\mathbf{n}}_j \cdot (\mathbf{x}_j - \mathbf{x}_i)) \frac{1}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\tilde{T}_{cl}}. \quad (2.14)$$

Because T_{con} consists of triangles, Eq. (2.15) is obtained from Eq. (2.14):

$$\mathbf{x}_i = \frac{\sum_{\tilde{t} \in \tilde{T}_{cl}} \left\{ \int_{\tilde{t}} \text{sign}(\tilde{\mathbf{n}}_j \cdot (\mathbf{x}_j - \mathbf{x}_i)) \frac{\mathbf{x}_j}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\tilde{t} \right\}}{\int_{\tilde{T}_{cl}} \text{sign}(\tilde{\mathbf{n}}_j \cdot (\mathbf{x}_j - \mathbf{x}_i)) \frac{1}{\|\mathbf{x}_j - \mathbf{x}_i\|} d\tilde{T}_{cl}}, \quad (2.15)$$

where $\tilde{t}$ is a spherical triangle which is obtained by mapping a triangle $t \in T_{cl}$ onto the unit sphere.

Let (j_1, j_2, j_3) is vertices of t , $(\tilde{j}_1, \tilde{j}_2, \tilde{j}_3)$ vertices of $\tilde{t}$, (L_1^s, L_2^s, L_3^s) the length of the arcs of $\tilde{t}$, and $(\vartheta_1, \vartheta_2, \vartheta_3)$ is three angles of $\tilde{t}$ (see Fig. 2.30). The weight $\tilde{\omega}'_{j_l}$ of $\tilde{j}_l$ ($l = 1, 2, 3$) for the vertex i on $\tilde{t}$ is obtained by Eq. (2.16):

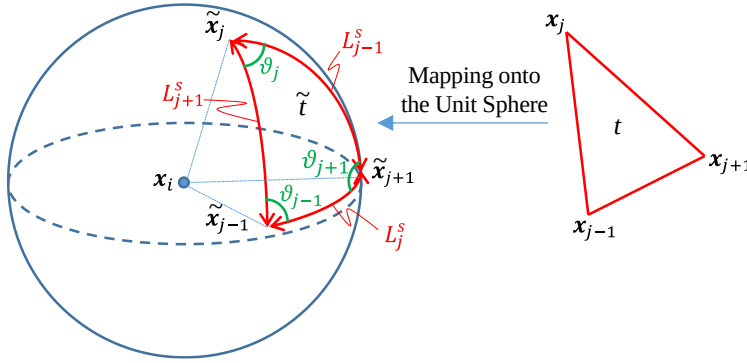


Fig. 2.30 Three lengths of arcs and three angles

$$\tilde{\omega}'_{\tilde{j}_i} = \frac{L_i^s - L_{i-1}^s \cos(\vartheta_{i+1}) - L_{i+1}^s \cos(\vartheta_{i-1})}{2 \sin(\vartheta_{i+1}) \sin(\vartheta_{i-1}) \|\mathbf{x}_j - \mathbf{x}_i\|}, \quad (2.16)$$

where

$$\cos(\vartheta_l) = \frac{2 \sin(h) \sin(h - L_l^s)}{\sin(L_{l+1}^s) \sin(L_{l-1}^s)} - 1, \quad (2.17)$$

$$h = \frac{L_1^s + L_2^s + L_3^s}{2}. \quad (2.18)$$

As shown in Eq. (2.19), the weight of ω'_j each vertex j of T_{cl} is the sum of the weight $\tilde{\omega}'_{\tilde{j}}$ of each mapped vertex $\tilde{j}$ corresponding to j :

$$\omega'_j = \sum_{\tilde{i} \in \tilde{T}_j} \tilde{\omega}'_{\tilde{j}}. \quad (2.19)$$

where $\tilde{T}_j$ is a set of spherical triangles obtained by mapping triangles including j onto the unit sphere.

Finally, as shown in Eq. (2.20), the weight ω_j in the MVCs of i for T_{cl} is calculated by normalizing ω'_j :

$$\omega_j = \frac{\omega'_j}{\sum_{j \in V^{T_{cl}}} \omega'_j} \quad (2.20)$$

where $V^{T_{cl}}$ is a set of vertices of T_{cl} .

In the proposed deformation method, MVCs in the 2D space is used for surface vertex repositioning, and MVCs in the 3D space is used for inner vertex repositioning. For developable surface regions (i.e. planar, cylindrical, and conical surface regions), the calculation of MVCs in the 2D space is performed on the development planes.

2.5.4.5 Vertex repositioning according to vertex types

In order to satisfy the target dimension, the boundary edge and surface information are first updated. Then, feature vertices, boundary edge vertices, surface vertices and inner vertices are sequentially moved by Methods 1 and 2 mentioned in sub-subsection 2.5.4.1. The movement procedure each vertex is described as follows.

Step 1: Feature Vertex Movement

All feature vertices are moved by Method 1. For the deformation type (7) the position of the local, feature vertices simply follow the representative point.

For the other deformation types, the movement procedure is shown in Fig. 2.31. First, each feature vertex is classified into a type-A or type-B vertex (Fig. 2.31(a)). The vertices shared by two types of regions (*control region* and *following regions*) or three types of regions (*control region*, *following regions*, and *deformed regions*) are identified as type-A vertices. Also, vertices shared by two types of regions (*control region* and *deformed region* or *following region* and *deformation region*) are identified as type-B vertices. Second, each type-A vertex forms a group with type-B vertices connected with the type-A vertex by a boundary edge segment of a *following region* (Fig. 2.31(b)). Then, the movement direction of each type-A vertex is determined as the average of the ones along the boundary edge segments between two *deformed regions*, which is incident to the type-B vertex in the group (Fig. 2.31(c)). Finally, each feature point is moved onto the updated surface along the direction (Fig. 2.31(d)).

Step 2: Boundary Edge Vertex Movement

If the boundary edge segment is a straight line segment, vertices are moved by Method 2. Otherwise, the vertices are moved by Method 1.

Step 3: Surface Vertex Movement

For developable surface regions (i.e. planar, cylindrical, or conical surface regions), vertices are first moved by Method 2. Then, vertices move to the closest points on the surface. For the others, vertices are moved by Method 1.

Step 4: Inner Vertex Movement

Inner vertices are moved by Method 2.

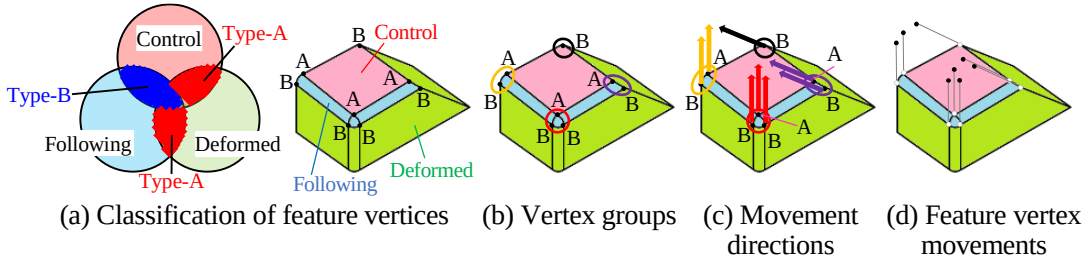


Fig. 2.31 Feature vertex movement

2.6 Quality improvement by Phased ODT smoothing

2.6.1 Overview of quality improvement

2.6.1.1 Optimal Delaunay Triangulation (ODT) smoothing

In this section, a quality improved method based on Optimal Delaunay Triangulation (ODT) smoothing [Chen 2011] is proposed. ODT smoothing is one of the mesh improvement methods which minimizes an error function defined by Eq. (2.21):

$$E_{ODT}(M) = \int_{\Omega} |\tilde{u}_M(\mathbf{x}) - u(\mathbf{x})| \rho_M(\mathbf{x}) d\mathbf{x}, \quad (2.21)$$

where $u(\mathbf{x})$ is $\|\mathbf{x}\|^2$, $\rho_M(\mathbf{x})$ a given density function defined on a convex domain $\Omega \subset \mathbb{R}^n$, M a simplicial subdivision (mesh model) of Ω , and $\tilde{u}_M(\mathbf{x})$ the piecewise linear approximation of $u(\mathbf{x})$ based on M . As shown in Fig. 2.32(a), $E_{ODT}(M)$ represents the difference volume between a paraboloid $u(\mathbf{x})$ and its piecewise linear approximation $\tilde{u}_M(\mathbf{x})$. When the position of vertices of M are fixed, the minimizer of $E_{ODT}(M)$ is a Delaunay triangulation (see Fig. 2.32(b)). A sequence of flipping operations are useful to obtain a Delaunay triangulation from an arbitrary triangulation. Therefore, for fixed vertices, $E_{ODT}(M)$ can be minimized by a sequence of flipping operations.

On the other hand, as shown in Fig. 2.32(c), ODT is a Delaunay triangulation minimizing $E_{ODT}(M)$ under the condition where the connectivity between vertices of M is fixed. ODT is defined as follows.

Definition [Chen 2004a]

Let $\mathcal{M}^N$ describes the set of all simplicial subdivisions of Ω with at most m vertices. Given a continuous function $u(\mathbf{x})$ on Ω and $1 \leq q \leq \infty$, a simplicial subdivision $M_{ODT} \in \mathcal{M}^m$ satisfying Eq. (2.22) is called *Optimal Delaunay Triangulation* (ODT) with respect to $u(\mathbf{x})$ and q .

$$\left(\int_{\Omega} |u(\mathbf{x}) - \tilde{u}_{M_{ODT}}(\mathbf{x})|^q d\mathbf{x} \right)^{1/q} = \inf_{M \in \mathcal{M}^m} \left(\int_{\Omega} |u(\mathbf{x}) - \tilde{u}_M(\mathbf{x})|^q d\mathbf{x} \right)^{1/q}. \quad (2.22)$$

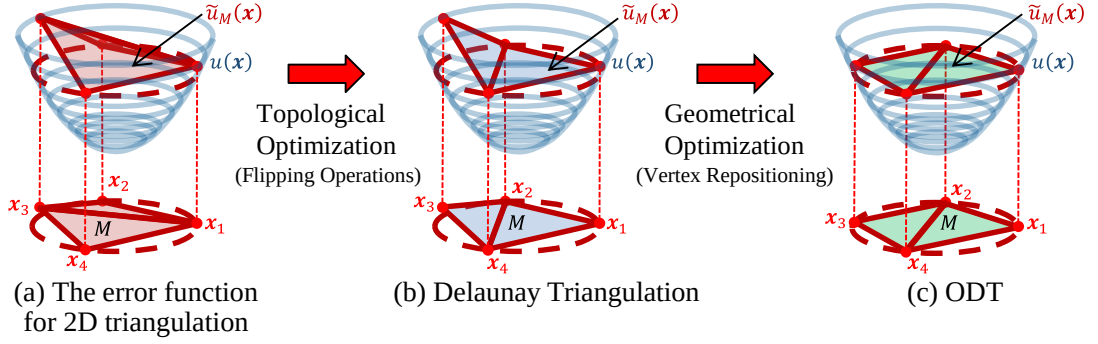


Fig. 2.32 Minimization of the error function by geometrical and topological operations

$E_{ODT}(M)$ is obtained when $u(\mathbf{x}) = \|\mathbf{x}\|^2$ and $q = 1$ in Eq. (2.22). In ODT smoothing, vertex repositioning and flipping operations are repeated in order to minimize $E_{ODT}(M)$.

For explanation of vertex repositioning minimizing $E_{ODT}(M)$, following notation are used.

- d_M : the dimension of the space where M exists,
- V^Ω : a set of inner vertices of M (its number of vertices is described by m),
- $\mathbf{x}_i$: the position of vertex i ,
- τ : a simplex (e.g. triangle for 2D, tetrahedron for 3D),
- N_i^{sim} : a set of neighboring simplexes of vertex i ,
- ψ_i : the hat function of $\mathbf{x}_i$ (a piecewise linear function which is 1 for $\mathbf{x}_i$ and 0 for the position of the other vertices),
- $|\cdot|$: the Lebesgue measure in $\mathbb{R}^{d_M}$ (e.g. area for 2D, volume for 3D).

For the uniform density $\rho_M(\mathbf{x}) = 1$, $E_{ODT}(V^\Omega)$ which is the discretized representation of $E_{ODT}(M)$ is obtained by Eq. (2.23):

$$E_{ODT}(V^\Omega) = \frac{1}{d_M + 1} \sum_{i=1}^m \left(\sum_{\tau \in N_i^{sim}} |\tau| \right) \mathbf{x}_i^2 - \int_{\Omega} \|\mathbf{x}\|^2 d\mathbf{x}. \quad (2.23)$$

In order to minimize $E_{ODT}(V^\Omega)$ with fixed connectivity of M , as shown in Eq. (2.24), an iterative optimization method is used:

$$V^{\Omega^{k+1}} = V^{\Omega^k} - \mathbf{A}_{opt}^{-1} \nabla E_{ODT}(V^{\Omega^k}), \quad (2.24)$$

where V^{Ω^k} is a set of vertices after k steps, $\mathbf{A}_{opt}$ is a scalar value or a regular matrix specified in each optimization method (e.g. in the gradient descent method, a scalar value is used), and;

$$\nabla E_{ODT}(V^\Omega) = \left(\frac{\partial E_{ODT}}{\partial \mathbf{x}_1}, \dots, \frac{\partial E_{ODT}}{\partial \mathbf{x}_m} \right)^T, \quad (2.25)$$

For convergence of the optimization method, ∇E_{ODT} should be $\mathbf{0}$. In other words Eq. (2.26) is satisfied for all vertices.

$$\frac{\partial E_{ODT}}{\partial \mathbf{x}_i} = 0, \quad (2.26)$$

ODT smoothing is an optimization method to find a vertex distribution satisfying Eq. (2.26). For the uniform density $\rho_M(\mathbf{x}) = 1$, Eq. (2.27) is obtained from Eq. (2.23) and Eq. (2.26):

$$\begin{aligned} \frac{\partial E_{ODT}}{\partial \mathbf{x}_i} &= \frac{1}{d_M + 1} \left[2\mathbf{x}_i \sum_{\tau \in N_i^{sim}} |\tau| + \sum_{\tau \in N_i^{sim}} \sum_{\mathbf{x}_k \in \tau, \mathbf{x}_k \neq \mathbf{x}_i} \|\mathbf{x}_k\|^2 \nabla_{\mathbf{x}_i} |\tau| \right] \\ &= \frac{1}{d_M + 1} \sum_{\tau \in N_i^{sim}} \sum_{\mathbf{x}_k \in \tau, \mathbf{x}_k \neq \mathbf{x}_i} \|\mathbf{x}_k - \mathbf{x}_i\|^2 \nabla_{\mathbf{x}_i} |\tau| \\ &= \frac{2}{d_M + 1} \sum_{\tau \in N_i^{sim}} (\mathbf{x}_i - \mathbf{c}_\tau) |\tau|, \end{aligned} \quad (2.27)$$

where $\mathbf{c}_\tau$ is the circumcenter of a simplex τ . Then, Eq. (2.28) is obtained from Eq. (2.26) and Eq. (2.27):

$$\mathbf{x}_i = \frac{\sum_{\tau \in N_i^{sim}} |\tau| \mathbf{c}_\tau}{\sum_{\tau \in N_i^{sim}} |\tau|}. \quad (2.28)$$

Hence, in an ODT, each inner vertex i is located at the weighted average of circumcenters of its neighboring simplexes.

For general density $\rho_M(\mathbf{x})$, Eq. (2.29) and Eq. (2.30) are obtained:

$$E_{ODT}(V^\Omega, \rho_M) = \sum_{i=1}^m \left(\sum_{\tau \in N_i^{sim}} |\tau| \rho_M \psi_i \right) \mathbf{x}_i^2 - \int_\Omega \|\mathbf{x}\|^2 \rho_M(\mathbf{x}) d\mathbf{x}. \quad (2.29)$$

$$\begin{aligned} \frac{\partial E_{ODT}(V^\Omega, \rho_M)}{\partial \mathbf{x}_i} &\approx 2\mathbf{x}_i \sum_{\tau \in N_i^{sim}} |\tau| \rho_M \psi_i + \sum_{\tau \in N_i^{sim}} \sum_{\mathbf{x}_k \in \tau, \mathbf{x}_k \neq \mathbf{x}_i} \|\mathbf{x}_k\|^2 \nabla_{\mathbf{x}_i} |\tau| \rho_M \psi_i \\ &= \frac{2}{d + 2} \sum_{\tau \in N_i^{sim}} (\mathbf{x}_i - \mathbf{c}_\tau) \rho_\tau |\tau|. \end{aligned} \quad (2.30)$$

where ρ_τ is a density defined on a simplex τ . In ODT smoothing, each inner vertex i is repositioned to $\mathbf{x}_i^{new}$ based on Eq. (2.30) as shown in Eq. (2.31):

$$\mathbf{x}_i^{new} = (1 - \alpha_{ODT}) \mathbf{x}_i + \alpha_{ODT} \frac{\sum_{\tau \in N_i^{sim}} |\tau| \rho_\tau \mathbf{c}_\tau}{\sum_{\tau \in N_i^{sim}} |\tau| \rho_\tau}, \quad (2.31)$$

where $|\cdot|_{\rho_\tau}$ is the Lebesgue measure under the given density ρ_τ . The step size α_{ODT} is used so that

inverted simplexes are not generated by the repositioning. The initial setting of α_{ODT} is 1, and if inverted simplexes are generated, α_{ODT} is reduced by half.

In the vertex repositioning by Eq. (2.31), as shown in Fig. 2.33(a), distorted simplexes can be generated near the boundary of the mesh model where flipping operations cannot be performed. The reason of this problem is that the objective function is not directly related to shape qualities such as the aspect ratio and the dihedral angle. Although the objective function should be changed near the boundary, in order to keep the simple form and the geometric interpretation of the smoothing, this problem is solved by using the following rule in ODT smoothing (see Fig 2.33(b)).

If a simplex τ includes at least one vertex on the boundary, the barycenter of τ is used as $\mathbf{c}_\tau$ instead of the circumcenter of τ .

By this modification, the polytope formed by $\mathbf{c}_\tau$ is located inside of the convex domain Ω , and the contribution of the distorted simplex near the boundary for the new vertex position becomes smaller.

ODT smoothing is efficient and effective for improving element shape qualities. However, in ODT smoothing, it is assumed that vertices on the boundary are well distributed. Hence, ODT smoothing allows only interior vertices to move, and it cannot improve element shape qualities near the boundary of mesh models. In this thesis, in order to solve this problem, a new quality improvement method based on ODT smoothing which improves element shape qualities of tetrahedral meshes from the boundary to the inside is proposed.

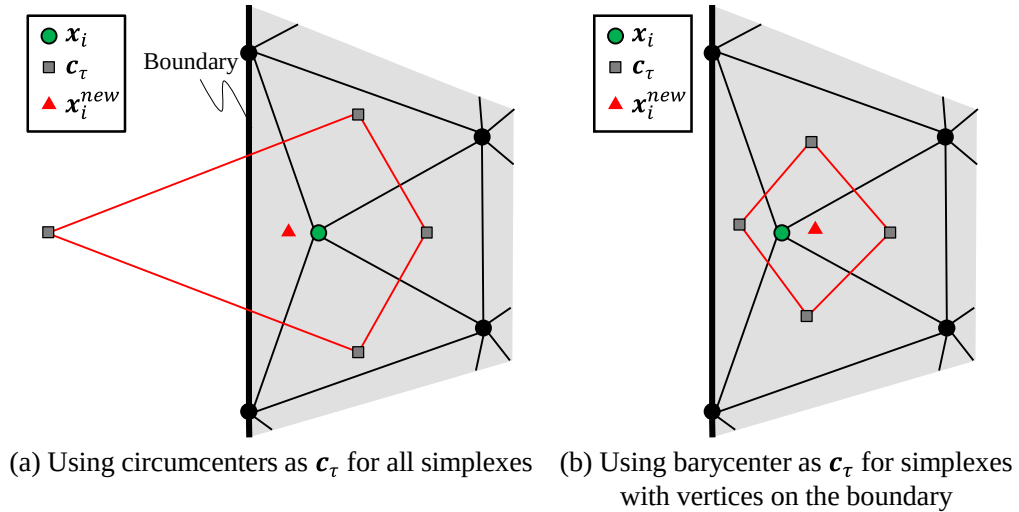


Fig. 2.33 Vertex repositioning near the boundary in ODT smoothing

2.6.1.2 Overview of proposed method

The overview of the proposed quality improvement method is shown in Fig. 2.34. The proposed quality improvement method contains following four features.

- All element shapes are sequentially improved from the boundary to the inside (boundary edges, surface triangles, and tetrahedra in that order) by ODT smoothing (called *Phased ODT smoothing*, Fig. 2.34(A)).
- Degenerated and inverted tetrahedra are removed by the local topological operations (Fig. 2.34 (B)).
- To recover mesh density of the original tetrahedral mesh in the deformed mesh, the target mesh density field represented by a regular grid is generated, and vertex insertion and vertex removal according to the target mesh density field are performed during ODT smoothing (Fig. 2.34(C)).
- To recover shape approximation accuracy of the original tetrahedral mesh, acceptable geometric error is calculated from the original mesh model, and vertex insertion is applied during ODT smoothing depending on the acceptable geometric error (Fig. 2.34(C)).

In the proposed quality improvement method, as a preprocessing, the density information and the acceptable geometric error are extracted from the original tetrahedral mesh before the deformation. After the deformation, the target mesh density field is extracted using the density information of the original tetrahedral mesh. Then, a modified ODT smoothing is applied to the boundary edges, surface triangles, and tetrahedra, in that order. In the modified ODT smoothing, the degenerated and inverted elements removal, the mesh density control, the shape approximation accuracy control, and ODT smoothing are performed in that order. The modified ODT smoothing in each phase is repeated until no more the mesh density control and the shape approximation accuracy control are performed. In each step of the modified ODT smoothing, two or three times ODT smoothing is enough for the improvement of element shape qualities.

In the next sub-subsection, local topological operations used for the vertex insertion and the vertex removal is described. After that, in the following sections, methods for degenerated and inverted tetrahedra removal are first presented. Second, the mesh density control including the extraction of the density information of the original tetrahedral mesh is described. Then, the shape approximation accuracy control including definition of geometric error is outlined. Finally, ways of applying ODT smoothing to the boundary edges, the surface triangles, and the tetrahedra are shown.

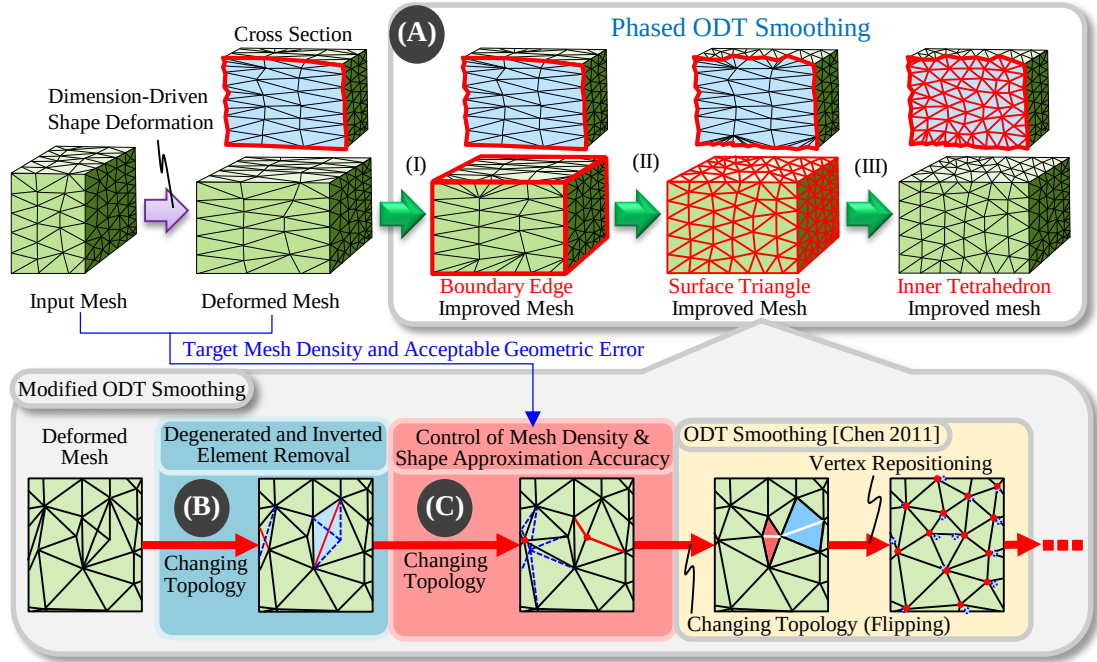


Fig. 2.34 The overview of the proposed quality improvement method

2.6.1.3 Local topological operations

In order to control the mesh density and shape approximation accuracy, vertices are inserted and removed. In the proposed quality improvement method, edge split is used for the vertex insertion and edge collapse is used for the vertex removal in order to keep the topological consistency of the tetrahedral mesh.

Edge split (Fig. 2.35(a)):

In edge split, an edge is divided into two edges and a vertex is inserted. In comparison with face split (Fig. 2.35(b)), edge length can be controlled more easily. Therefore, in the proposed quality improved method, edge split is used for the vertex insertion. In the proposed quality improved method, if the edge e is on the curved surface region, the new vertex is moved onto the fitted surface after the edge split of e .

Edge collapse (Fig. 2.36(a)):

In edge collapse, an edge is collapsed and its two end points are merged into a new vertex. Although local re-meshing (Fig. 2.36(b)) can be used for vertex removal, it is difficult to remove vertices on the surface while preserving the shape of tetrahedral meshes. In addition, the result of re-meshing can be obtained by combination of edge collapse and edge split. Therefore, in the proposed quality improved method, edge collapse is used for the vertex removal.

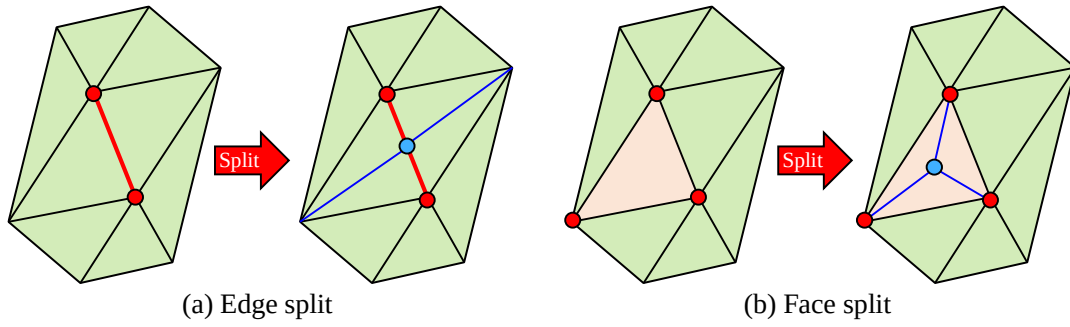


Fig. 2.35 Local topological operations for vertex insertion

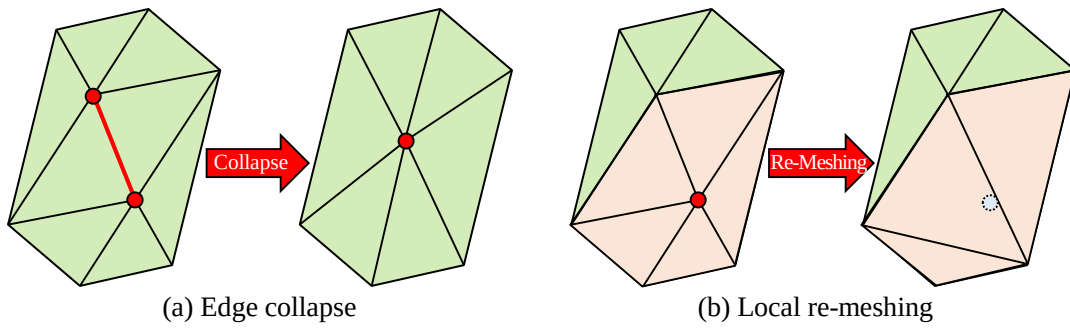


Fig. 2.36 Local topological operations for vertex removal

In order to preserve the shape of tetrahedral meshes, edge collapse is not applied to the following edges (see Fig. 2.37);

- (A) the edge connecting two feature vertices,
- (B) the edge which is not lying on the boundary edge segments and connects two boundary edge vertices, and
- (C) interior edge connecting two surface vertices.

In addition, the position of new vertex after edge collapse is decided according to the following rules (each edge is shown in Fig. 2.37);

- (D) if an endpoint of the edge is a feature vertex, the new vertex position is the one of the feature vertex,
- (E) if an endpoint of the edge which is not lying on the boundary edge segments is a boundary edge vertex, the new vertex position is the one of the boundary edge vertex,
- (F) if an endpoint of the interior edge is a surface vertex, the new vertex position is the one of the surface vertex, and
- (G) if other edges are collapsed, their endpoints merge into their midpoints.

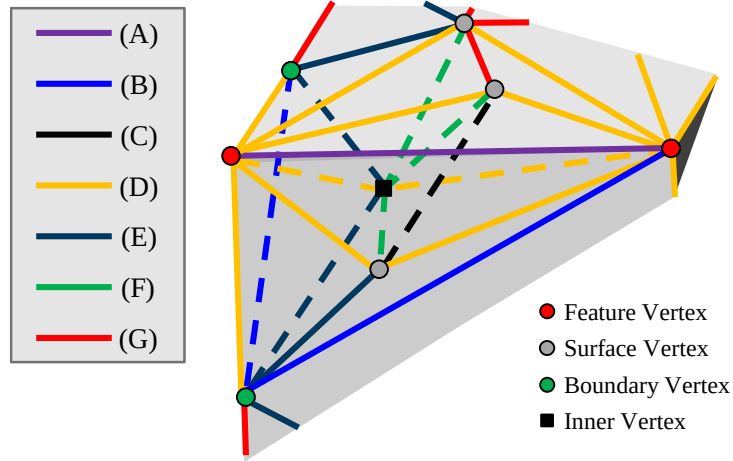


Fig. 2.37 Classification of edge for edge collapse preserving the shape of tetrahedral meshes

In addition, if the edge e is on the curved surface region, the new vertex is moved onto the fitted surface after the edge collapse of e .

2.6.2 Degenerated or inverted element removal

After deformation, mesh models may have some degenerated tetrahedra and inverted tetrahedral. In ODT smoothing, vertices are moved to the weighted average of the circumcenter of their incident elements in order to improve element shape quality. However, if degenerated or inverted elements are included in the mesh model, the circumcenter of degenerated or inverted elements prevent the improvement and ODT smoothing does not work well. Hence, if only ODT smoothing is used, it is difficult to remove all of degenerated or inverted tetrahedra. Therefore, degenerated or inverted element removal methods are needed.

In the problem setting of this thesis, the mesh properties do not have to be considered in this process because they are improved in the later smoothing process. Therefore, degenerated or inverted tetrahedra removal methods based on local topological operations can be used. As mentioned in subsection 2.2.3, although many methods which improve or remove the degenerated and inverted elements have been proposed and they may be effective for deformed meshes, a simple method based on local topological operations is introduced in this thesis.

In the proposed quality improvement method, degenerated and inverted tetrahedra are found by stretch (Eq.1.1) mentioned in section 1.3:

$$Q(\tau) = \frac{6\sqrt{6}V(\tau)}{\left(\max_{e \in E_\tau} L(e)\right) A^{tet}(\tau)}, \quad (1.1)$$

where

- $V(\tau)$: the signed volume of τ ,
- $A^{tet}(\tau)$: the surface area of τ ,
- E_τ : a set of edges of τ , and
- $L(e)$ is the length of edge e .

Stretch becomes 1 for regular tetrahedron, 0 for degenerated tetrahedron, and a negative value for inverted tetrahedron.

Degenerated element removal:

In the proposed quality improvement method, degenerated elements, e.g. sliver, cap, and spade are removed by the combination of edge split and edge collapse (Fig. 2.38). These elements have four vertices near one plane. In the sliver and cap, their projected vertices on to a plane form a quadrangle and a triangle having one interior vertex, respectively (Fig. 2.38(a) and (b)). Spade is a tetrahedron whose three vertices are near a line (Fig. 2.38(c)).

For slivers, double split collapse operation [Li 2003] is performed. As shown in Fig. 2.38(a), edge split is first applied to two edges which cross each other. Then, a new short edge is collapsed. As a result, a sliver becomes four triangles. For caps and spades, split collapse operation [Li 2003] is performed. As shown in Fig. 2.38(b) and (c), edge split is first applied to an edge which is shared by the largest triangle and the smallest triangle of the tetrahedron. Then, a new short edge is collapsed. By this operation, a cap or a spade becomes two triangles. In edge collapse, if an endpoint of the edge is a surface vertex, the new vertex position is the one of the surface vertex in order to preserve shape of tetrahedral meshes.

The degenerated element removal is performed by the following four steps.

- Step 1 Tetrahedra whose stretches are lower than a threshold γ_{st} are inserted to a min-heap based on their stretch. (In this thesis, 0.05 which is the recommended lower limit is used as γ_{st} .)
- Step 2 A tetrahedron which has the minimum stretch in the min-heap is popped from the min-heap and it classified into a sliver, cap, or spade by its shape.
- Step 3 Above removal operation is applied to the popped tetrahedron, and check stretches of tetrahedra which are affected by the removal operation. If the checked tetrahedron does not exist in the min-heap and its stretch is lower than γ_{st} , it is inserted to the min-heap.
- Step 4 Step 2 and 3 are repeated until stretches of all tetrahedra become larger than γ_{st} .

Inverted element removal:

Because the mesh properties do not have to be considered as mentioned above, half edge collapse (Fig. 2.39) is adopted for removing inverted tetrahedra. In this operation, all tetrahedra which have a negative value of stretch are found, and half edge collapse is applied to an edge of the tetrahedron

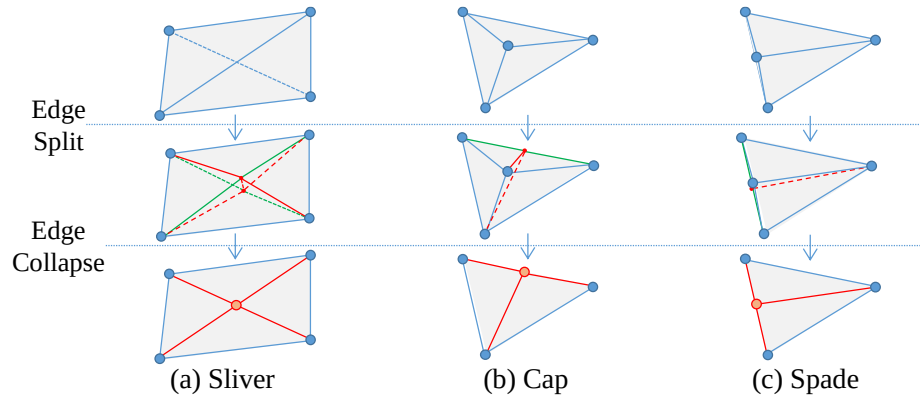


Fig. 2.38 Degenerated tetrahedra and removal operations

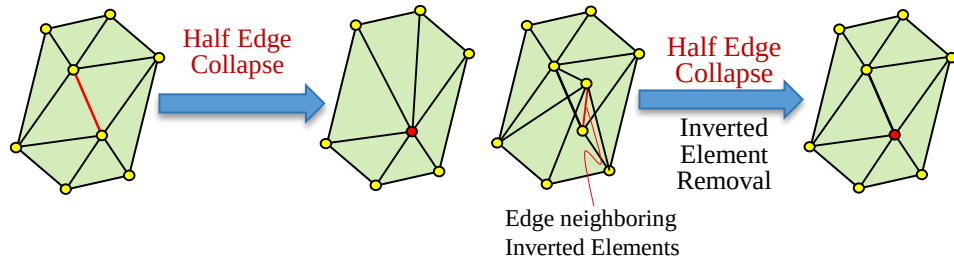


Fig. 2.39 Half edge collapse and inverted element removal

which has the minimum stretch until the minimum stretch becomes more than 0. The collapsed edge and the new vertex position are selected so that minimum stretch of tetrahedra neighboring the new vertex is maximized after edge collapse.

2.6.3 Mesh density control

In this research, the element size is discussed using edge length because tetrahedral elements with high shape quality are approximately regular tetrahedra and their six edge lengths are almost same with each other. Therefore, for the control the mesh density, vertices are inserted to edges or edges are collapsed in the proposed quality improvement method.

In the proposed quality improvement method, vertices are moved by ODT smoothing, and the other elements (edges, triangles, and tetrahedra) are divided or collapsed for control the mesh density and shape approximation accuracy. Hence, the mesh density of the original mesh model cannot be represented using elements of mesh models. Therefore, a regular grid is generated in order to represent mesh density of the original mesh model in the deformed mesh models.

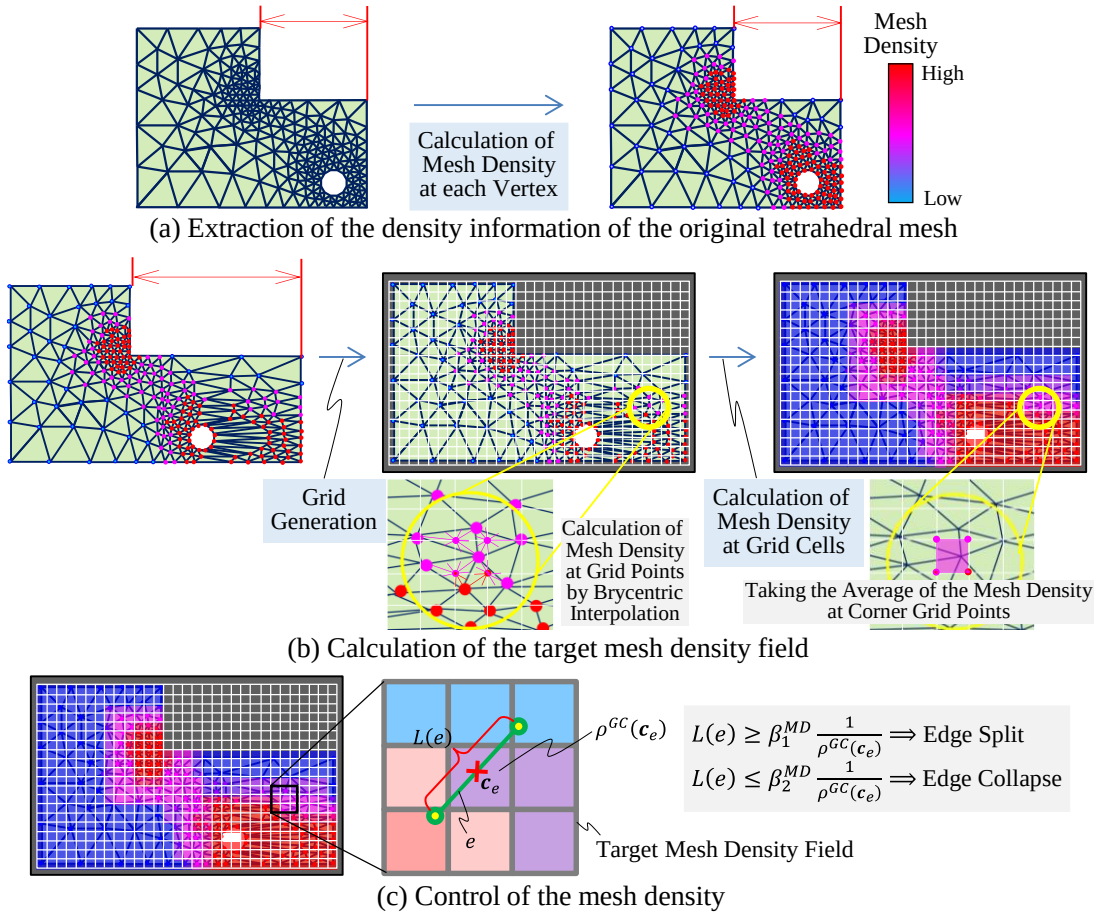


Fig. 2.40 The overview of the mesh density control

The overview of the mesh density control is shown in Fig. 2.40. Before the deformation, in order to recover the mesh density of the original tetrahedral mesh in the deformed meshes, the density information of the original tetrahedral mesh is extracted. After the deformation, the target mesh density field for the deformed mesh model is represented by using a regular grid and the density information of the original mesh model. In Phased ODT smoothing, edge split and edge collapse according to the target mesh density field are performed.

Extraction of the density information of the original tetrahedral mesh (Fig. 2.40(a)):

Before the deformation, at each vertex i of the original tetrahedral mesh, the mesh density ρ_i^v is calculated as the average of reciprocals of its incident edge lengths as shown in Eq. (2.32):

$$\rho_i^v = \frac{1}{|N_i^{edg}|} \sum_{e \in N_i^{edg}} \frac{1}{L(e)}. \quad (2.32)$$

where N_i^{edg} is a set of incident edges e of i and $L(e)$ is the length of e .

Calculation of the target mesh density field (Fig. 2.40(b)):

After the deformation, a regular grid covering the deformed mesh is generated. Then, the mesh density is given to each cell of the grid. In order to give the mesh density to cells of the grid, as shown in Eq. (2.33), the mesh density $\rho^{GP}(\mathbf{p}^g)$ of each grid point g is estimated using the barycentric interpolation:

$$\rho^{GP}(\mathbf{p}^g) = \frac{1}{V(\tau_g)} \sum_{i \in N_{\tau_g}^{vtx}} V(\tau_g^i) \rho_i^v. \quad (2.33)$$

where τ_g is a tetrahedron which includes g , τ_g^i a tetrahedron which is obtained by substituting the vertex i with g on τ_g , $V(\cdot)$ the volume of the tetrahedra (τ_g or τ_g^i), and ρ_i^v is at vertex i . Finally, mesh density of each grid cell is calculated by taking the average of the nonzero mesh densities of its corner grid points.

Control of the mesh density (Fig. 2.40(c)):

To recover mesh density of the original mesh model, if $L(e) \geq \beta_1^{MD} \frac{1}{\rho^{GC}(\mathbf{c}_e)}$, edge e is split, and if $L(e) \leq \beta_2^{MD} \frac{1}{\rho^{GC}(\mathbf{c}_e)}$, edge e is collapsed where $\rho^{GC}(\mathbf{c}_e)$ is the mesh density of the grid cell of the target mesh density field which includes the midpoint $\mathbf{c}_e$ of the edge e . β_1^{MD} and β_2^{MD} are constants and called *density thresholds* in this thesis.

The control of mesh density is performed by the following seven steps.

- Step 1 Edges satisfying $L(e) \geq \beta_1^{MD} \frac{1}{\rho^{GC}(\mathbf{c}_e)}$ are inserted to a max-heap based on their length.
- Step 2 The longest edge in the heap is popped and split.
- Step 3 Step 2 is repeated until the max-heap becomes empty.
- Step 4 Edges satisfying $L(e) \leq \beta_2^{MD} \frac{1}{\rho^{GC}(\mathbf{c}_e)}$ are inserted to a min-heap based on their length.
- Step 5 The shortest edge in the heap is popped.
- Step 6 The popped edge is collapsed and check length of edges which are affected by the edge collapse. If the checked edge does not exist in the min-heap and satisfies $L(e) \leq \beta_2^{MD} \frac{1}{\rho^{GC}(\mathbf{c}_e)}$, it is inserted to the heap.
- Step 7 Step 5 and 6 are repeated until min-heap becomes empty.

The size of each cell and *density thresholds* can be decided based on the objective of analyses. If uniform tetrahedral meshes are desired, using a half of the average edge lengths of the original tetrahedral mesh as the size of each cell may be effective. Because vertices are inserted to midpoint of edges by edge split, the edge length is reduced by a half after the edge split. On the other hand, new vertices are created on the midpoint of collapsed edges after edge collapse, a half of the length of the collapsed edge is added to the lengths of neighbors in the worst case. Therefore, for the convergence, β_1^{MD} should be larger than twice of β_2^{MD} , and $1.0 + (\beta_2^{MD}/2)$ should be smaller than β_1^{MD} . In this thesis, for uniform tetrahedral meshes, β_1^{MD} and β_2^{MD} are set to about 1.5 and 0.67, respectively.

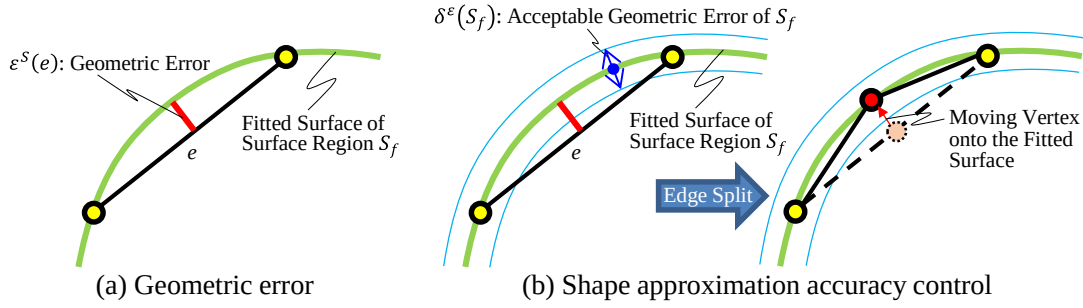


Fig. 2.41 The definition of geometric error and shape approximation accuracy control

2.6.4 Shape approximation accuracy control

In this research, as shown in Fig. 2.41(a), the geometric error $\varepsilon^S(e)$ of each surface edge e of the tetrahedral mesh is defined as the shortest distance from the midpoint of the edge to the fitted surface of the curved surface region S_f (i.e. cylindrical, conical, spherical, and torus surface region). In each curved surface region of the original mesh model, acceptable geometric error $\delta^\varepsilon(S_f)$ is calculated as the maximum geometric error of the region before the deformation.

In order to recover the shape approximation accuracy of the original mesh model, edge split is applied to surface mesh edges $e \in E_S$ (E_S : a set of edges on the surface region S_f) which have a larger geometric error than the acceptable geometric error $\delta^\varepsilon(S_f)$ of the curved surface. As shown in Fig. 2.41(b), in the edge split, a vertex is first inserted to the midpoint of the edge, and then the vertex is moved onto the fitted surfaces. Thus, the shape approximation accuracy of the deformed mesh becomes similar to that of the original mesh model.

2.6.5 Phased ODT smoothing

2.6.5.1 Overview of Phased ODT smoothing

In Phased ODT smoothing, ODT smoothing is applied to the boundary edges, the surface triangles, and the tetrahedra of the tetrahedral mesh in that order (see Fig. 2.34(A)). For the explanation, the vertex repositioning (Eq. (2.31)) of ODT smoothing is shown again:

$$\mathbf{x}_i^{new} = (1 - \alpha_{ODT})\mathbf{x}_i + \alpha_{ODT} \frac{\sum_{\tau \in N_i^{sim}} |\tau|_{\rho_\tau} \mathbf{c}_\tau}{\sum_{\tau \in N_i^{sim}} |\tau|_{\rho_\tau}}, \quad (2.31)$$

where

- $\mathbf{x}_i^{new}$: the position of vertex i after repositioning,
- α_{ODT} : a step size,
- $\mathbf{x}_i$: the position of vertex i before repositioning,

- τ : an element (an edge, a triangle, or a tetrahedron),
- $\mathbf{c}_\tau$: the circumcenter or barycenter of τ ,
- N_i^{sim} : a set of incident elements of i , and
- $|\cdot|_{\rho_\tau}$: the Lebesgue measure under the given density ρ_τ .

If $|\tau|_{\rho_\tau} = |\tau| \cdot \frac{1}{|\tau|} = 1$, Eq. (2.34) is obtained from Eq. (2.31):

$$\mathbf{x}_i^{new} = (1 - \alpha_{ODT})\mathbf{x}_i + \frac{\alpha_{ODT}}{|N_i^{sim}|} \sum_{\tau \in N_i^{sim}} \mathbf{c}_\tau, \quad (2.34)$$

where $|N_i^{sim}|$ is the number of incident elements of i . In Eq. (2.31), if $|\tau|_{\rho_\tau} = |\tau|$, when $|\tau|$ becomes larger, the contribution of τ becomes larger. It means that the vertex tends to approach elements with large Lebesgue measure, and the vertex repositioning has a large influence for the mesh density. On the other hand, in Eq. (2.34), the contribution of each element is not depend on its Lebesgue measure. Therefore, the influence of the vertex repositioning for the mesh density becomes smaller. In Phased ODT smoothing, Eq. (2.34) is used.

The following sub-subsections, ways of applying ODT smoothing to the boundary edges, the surface triangles, and the tetrahedra are shown.

2.6.5.2 Boundary edge improvement

For straight line segments, the vertex repositioning by ODT smoothing is performed by using Eq. (2.34), where midpoints of edges are used as $\mathbf{c}_\tau$ (stated exactly, this smoothing is called CPT smoothing). On the other hand, the circles or circular arcs shrink when their vertices are moved by Eq. (2.34). Therefore, for the circles or circular arcs, the vertex repositioning is performed on the 1D parameter space based on the central angle. As shown in Fig. 2.42, the position of each vertex i on the circle (or circular arc) is parameterized by its central angle θ_i , and the new central angle is calculated by Eq. (2.34). The vertex moves onto the circle or the circular arc corresponding to the new calculated central angle θ_i^{new} . In this phase, flipping operations are not performed.

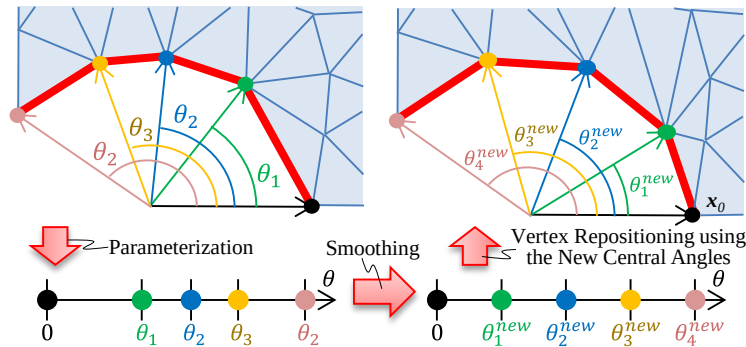


Fig. 2.42 ODT smoothing on the circle or circular arc

2.6.5.3 Surface triangle improvement

In the surface triangle improvement, for planar surface regions, 2D ODT smoothing is done by Eq. (2.34) where circumcenters of surface triangles are used as $\mathbf{c}_\tau$. In order to avoid the generation of distorted triangles near the region boundary, barycenters are used as $\mathbf{c}_\tau$ in Eq. (2.34) for triangles that have at least one vertex on the region boundary instead of their circumcenters.

On the other hand, curved surface regions (i.e. cylindrical, conical, spherical, and torus surface regions) are shrunk by the vertex repositioning using Eq. (2.34). Therefore, for curved surface regions, the vertex repositioning is performed on the following 2D parameter space.

- *Cylindrical and conical surface region improvement* (Fig. 2.43(a)(b)): because cylindrical and conical surfaces are developable surfaces, ODT smoothing is performed on the development plane consisting of a 1-ring neighborhood of each vertex.
- *Spherical surface region improvement* (Fig. 2.43(c)): first, for each vertex, a spherical-coordinate system containing the vertex on its equator is defined, and the 1-ring neighborhoods are parameterized in the 2D parameter space defined by the polar angle θ^{sp} and the azimuth angle φ^{sp} .
- *Torus Surface Region Improvement* (Fig. 2.43(d)): for each vertex, vertices in the 1-ring neighborhood are parameterized in the 2D parameter space defined by the poloidal angle φ^{tori} and the toroidal angle θ^{tori} .

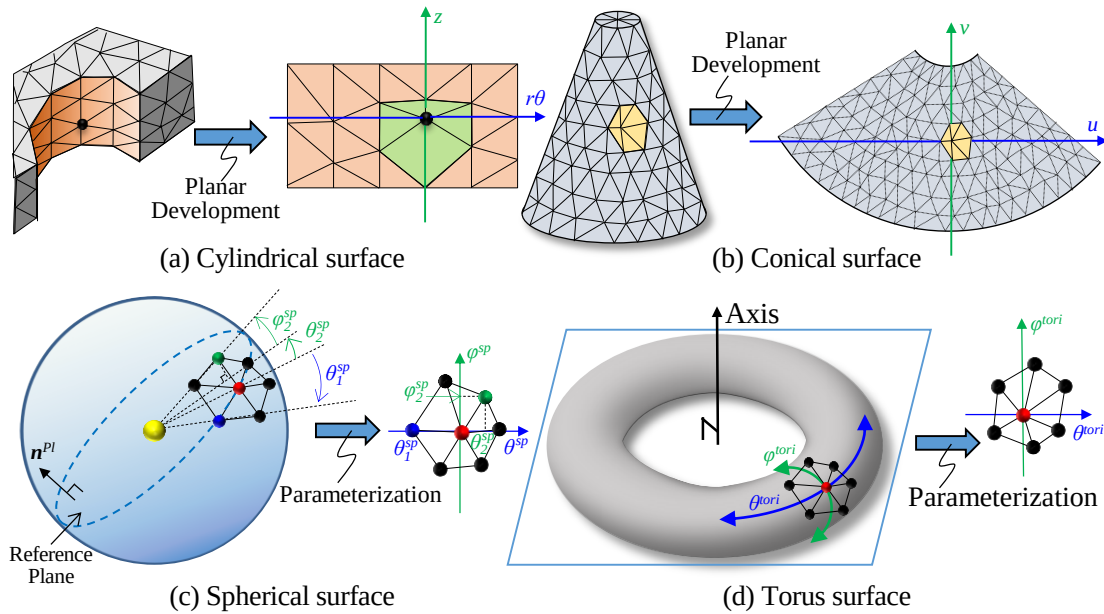


Fig. 2.43 2D parameter space of each surface region for ODT smoothing

The vertex repositioning of ODT smoothing on the 2D parameter planes for each surface region is performed by the following three steps (see Fig 2.44).

- Step 1 Each vertex i and a set of circumcenters (or barycenters) $P^{c_\tau} = \{c_\tau | \tau \in N_i^{stgl}\}$ (where N_i^{stgl} is a set of surface triangles τ neighboring i) are mapped onto the 2D parameter plane. The position of mapped i and mapped c_τ are denoted by $\tilde{x}_i$ and $\tilde{c}_\tau$.
- Step 2 The new position $\tilde{x}_i^{new}$ of mapped i is calculated by Eq. (2.34) using $\tilde{x}_i$ and $\tilde{c}_\tau$ instead of x_i and c_τ .
- Step 3 The new position of vertex i is derived from $\tilde{x}_i^{new}$ by inverse mapping of the first step.

In the surface triangles improvement, edge flipping is applied to obtain a Delaunay triangulation. In Phased ODT smoothing, to preserve the topological consistency of tetrahedra, the edge flipping is performed by the combination of edge split and edge collapse, as shown in Fig. 2.45. In the edge flipping, the edge is first split. Then a new edge whose two vertices are on the surface is collapsed. If degenerated or inverted elements would occur by these operations, flipping is not applied to such edges. Hence, some triangles do not satisfy the Delaunay condition (any vertices which are not included in the triangle are not the circumscribe circle of the triangle) because some edges cannot be flipped. Therefore, the barycenter is used as c_τ in Eq. (2.34) instead of their circumcenters for triangles which do not satisfy the Delaunay condition.

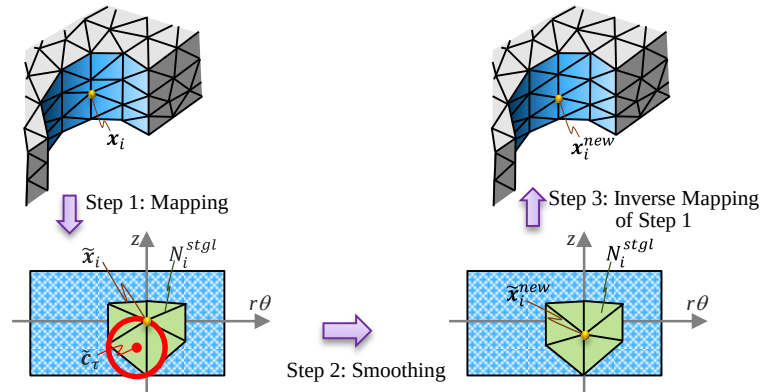


Fig. 2.44 The overview of ODT smoothing for curved surface regions

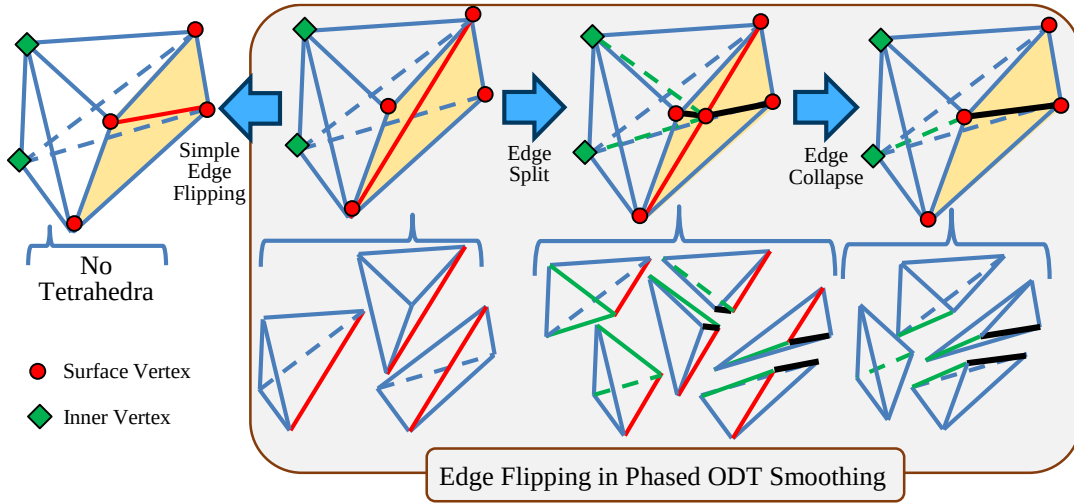


Fig. 2.45 Edge flipping in Phased ODT smoothing

2.6.5.4 Tetrahedron improvement

In the tetrahedron improvement, original 3D ODT smoothing is used. Therefore, inner vertices are moved by using Eq. (2.34) where circumcenters of tetrahedra are used as $\mathbf{c}_\tau$. In order to avoid the generation of distorted tetrahedra near the surface, barycenters are used as $\mathbf{c}_\tau$ in Eq. (2.34) for tetrahedra that have at least one surface vertex instead of their circumcenters. Also, if the tetrahedra do not satisfy the Delaunay condition (any vertices which are not included in the tetrahedron are not the circumscribe sphere of the tetrahedron), their barycenters are used as $\mathbf{c}_\tau$ in the Eq. (2.34) instead of their circumcenters.

In the tetrahedra improvement, *Flipping 2-3*, *Flipping 3-2*, and *Flipping 4-4* (Fig. 2.46) are applied so that the tetrahedra satisfy the Delaunay condition. In *Flipping 2-3*, two neighboring tetrahedra sharing a triangle become three neighboring tetrahedra sharing an edge. *Flipping 3-2* is an inverse operation of *Flipping 2-3*. In *Flipping 4-4*, an edge shared by four tetrahedra which form an octahedron is swapped. These flipping operations are applied to triangles or edges which satisfy the following three conditions.

- Some neighboring tetrahedra do not satisfy the Delaunay condition.
- All resulting tetrahedra satisfy the Delaunay condition after the flipping operation.
- After the flipping operation, the minimum quality of the resulting tetrahedra is higher than the one before the flipping operation.

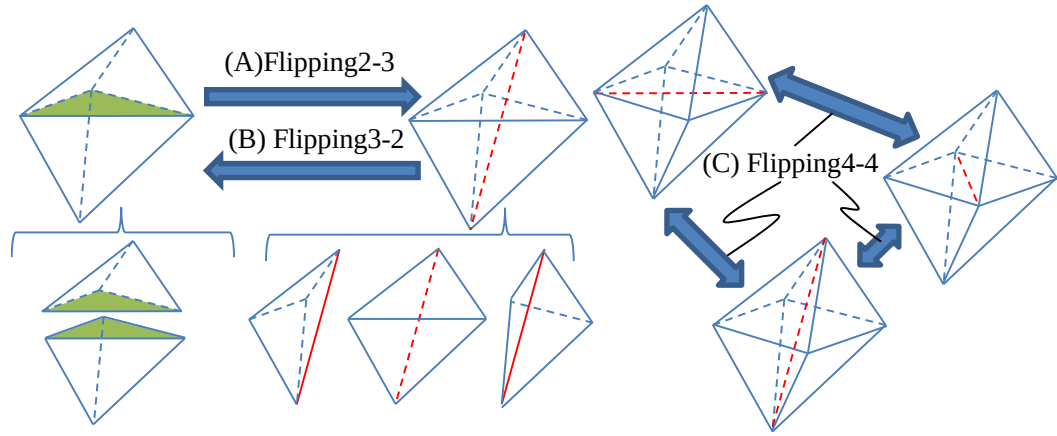


Fig. 2.46 Flipping operation for the tetrahedron improvement of Phased ODT smoothing

2.7 Experimental results and evaluation

2.7.1 Overview of experiments

All the experiments were processed on a personal computer having following spec.

- OS: Windows 7 Professional 64bit.
- CPU: Intel Core i7-5960X 3.00GHz.
- RAM: 64.0 GB.
- Development environment: Microsoft Visual Studio 2010.
- Programing language: C++.
- Graphic API: Open GL.

In this section, the effectiveness of each function is demonstrated using some tetrahedral meshes. Unless otherwise noted, the input tetrahedral meshes are generated by the Octree meshing tool of [CATIA V5R18]. In addition, in order to solve systems of linear equations in the space embedding method mentioned in sub-subsection 2.5.4.2, Eigen 3.1.2 [2012] which is a template library for linear algebra is used.

2.7.2 Mesh segmentation

Input tetrahedral meshes and segmentation results by the proposed mesh segmentation method in this thesis are shown in Fig. 2.47-2.49. In these figures, each surface region is color coded: planar, cylindrical, conical, spherical, and torus surface regions are represented by red, green, blue, yellow, and light blue, respectively. In Fig. 2.47 and Fig. 2.48, the input meshes includes many fillet surfaces which connects with others with G^0 and G^1 continuities. In Fig. 2.49, some segmentation results of mechanical parts are shown.

As shown in Fig. 2.47(a), some boundary lines (A) between a conical surface region and a torus surface region, (B) between a spherical surface region and a torus surface region, and (C) between two torus surface regions with G^0 and G^1 continuities are included in the surface of the input mesh and it is difficult to extract them by existing methods as mentioned in subsection 2.2.1. However, as shown in Fig. 2.47(b), the surface of the tetrahedral mesh was correctly segmented and surface regions are classified into correct surface types by the proposed mesh segmentation method. In addition, as shown in Fig. 2.47(c), each boundary edge segment between surface regions was extracted and recognized correctly. Table 2.8 shows all radii of the torus fillet surface regions of the original solid model and calculation results by the proposed method. Although calculation errors became large for some surface regions because Levenberg-Marquardt (LM) method could not converge, radii of most torus fillet surface regions were calculated in accuracy of 10^{-5} . The cause why the LM method could not converge is inaccurate initial parameters (the vector $\mathbf{p}^{LM}$ in subsection 2.4.5) obtained by the surface fitting based on RANSAC, and the determination of initial parameters is an issue in the future. Table 2.9 shows the calculation time of each process. The number of triangles of the input tetrahedral mesh was 20,032 and the total calculation time was 58.2s. In this experiment, the conical surface region extraction took the longest time and it was 15.7s. In order to segment the surface of the input tetrahedral mesh which includes two or more conical surface regions, the initial seed regions used in the conical surface fitting based on RANSAC are randomly selected and incorrect combinations of the initial seed regions can be selected again and again. Therefore, the selection of the initial seed regions should be improved in the future works.

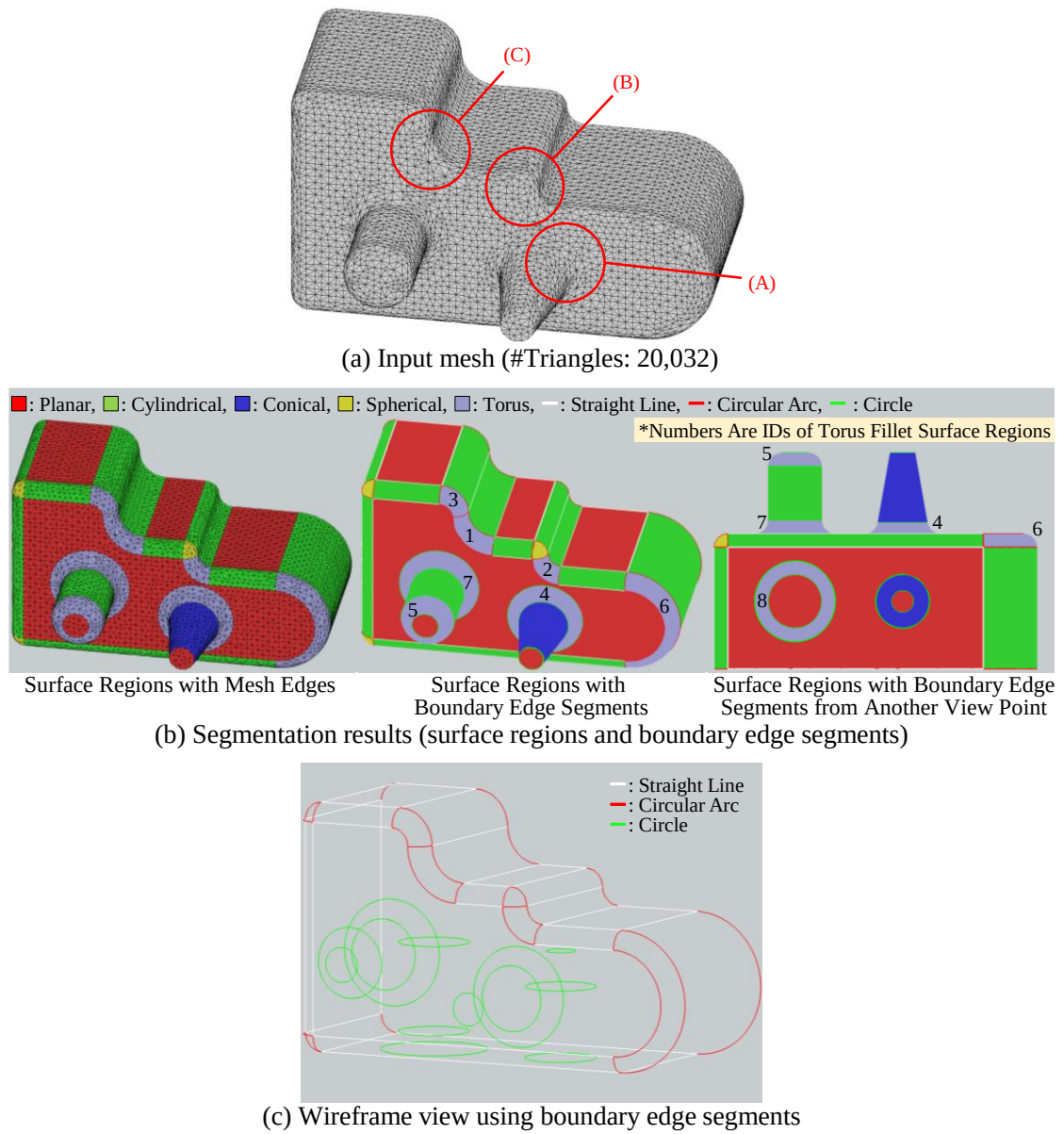


Fig. 2.47 The segmentation result of the first test model

Table 2.8 The fillet radii in the first test model

Fillet surface (see Fig.2.47(b))	The major radius in the original solid model	The major radius calculated by the proposed method	Calculation error of major radius	The minor radius in the original solid model	The minor radius calculated by the proposed method	Calculation error of minor radius
1	15.0	14.93755	-0.06245	5.0	5.026767	0.026767
2	10.0	10.83004	0.83004	5.0	4.913348	-0.08665
3	5.0	5.168836	0.168836	5.0	5.010056	0.010056
4	14.1955	14.19554	0.00004	5.0	5.00006	0.00006
5	5.0	4.99999	-0.00001	5.0	5.0	0
6	15.0	14.99999	-0.00001	5.0	5.0	0
7	15.0	14.99999	-0.00001	5.0	5.0	0
8	15.0	15.00004	0.00004	5.0	5.000045	0.000045

Table 2.9 Calculation time of each process for the first test model (#Triangles: 20,032)

Process		Calculation time [s]
Sharp edge extraction		0.031
Calculation of principal curvature and direction		0.14
Planar and quadric surface region extraction	Total	16.6
	Planar	0.078
	Cylindrical	0.23
	Conical	15.7
	Spherical	0.58
Torus surface region extraction		12.5
Calculation of surface parameters		28.8
Boundary edge recognition		0.044
Total		58.2

As shown in Fig 2.48(a), the second test model also includes some boundaries which cannot be extracted by existing methods. For example, each spherical surface region connects with a torus surface whose sign of the Gaussian curvature is same as that of the spherical surface in Fig. 2.48(a). As shown in Fig. 2.48(b) and (c), each surface region and its boundary were extracted accurately by the proposed method. The calculation time is shown in Table 2.10. The number of triangles of the input mesh was 10,122 and the total calculation time was 21.9s. The calculation of surface parameters took the longest time because the surface fitting based on RANSAC was repeated until the LM method converged or the iteration time reached a threshold (in this experiment, 100) in order to obtain the initial parameters of the LM method. Therefore, for more speeding up of the proposed method, the determination of initial parameters of the LM method is important.

Results of some mechanical parts are shown in Fig. 2.49 and the number of surface triangles of each tetrahedral mesh and each calculation time are shown in Table 2.11. These results shows that the surfaces of tetrahedral meshes were correctly segmented and surface regions are classified into correct surface types.

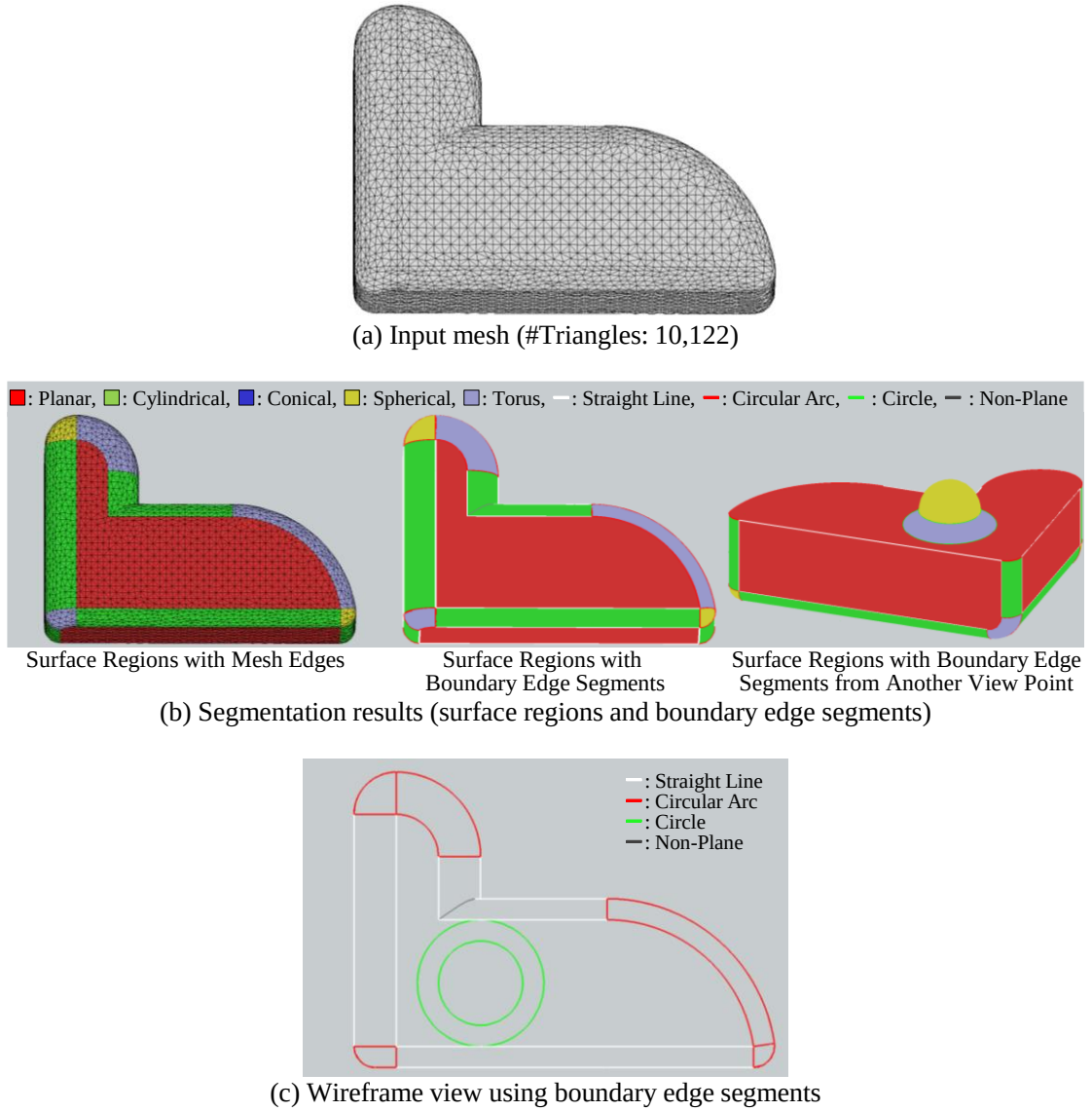


Fig. 2.48 The segmentation result of the second test model

Table 2.10 Calculation time of each process for the second test model (#Triangles: 10,122)

Process		Calculation time [s]
Sharp edge extraction		0.016
Calculation of principal curvature and direction		0.062
Planar and quadric surface region extraction	Total	3.05
	Planar	0.047
	Cylindrical	0.078
	Conical	1.98
	Spherical	0.95
Torus surface region extraction		4.7
Calculation of surface parameters		14.0
Boundary edge recognition		0.036
Total		21.9

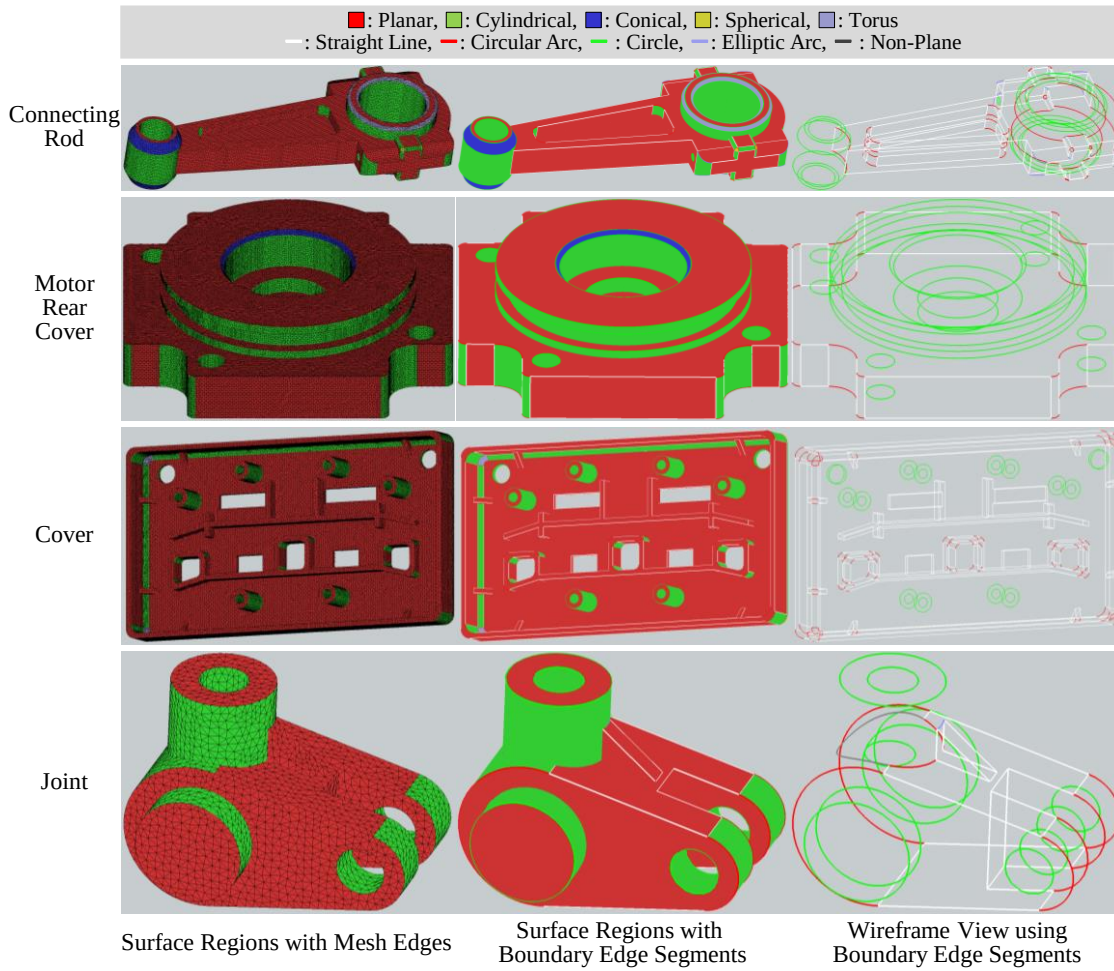


Fig. 2.49 Segmentation results of mechanical parts

Table 2.11 The number of triangles and the total calculation time for each model

Model	#Triangles	Total calculation time [s]
Connecting rod	78,384	140.6
Motor rear cover	169,812	50.4
Cover	136,372	82.7
Joint	7,928	0.59

2.7.3 Dimension-driven shape deformation

In this subsection, the results of various deformation of a tetrahedral mesh using the proposed dimension-driven shape deformation method are first shown in Fig. 2.50. After that, as shown in Fig. 2.51 and Fig.2.52, some mechanical parts are deformed by the proposed method.

Figure 2.50 shows the input mesh (a) and the deformation result of the height of the boss with chamfer and fillets (b), the radius of the cylindrical boss with fillets (c), the radius of the fillet (d),

the angle of the chamfer (e), the length of the chamfer (f), and each cross-section. In Fig. 2.50, each surface region is color coded: *control*, *following*, *deformed*, the reference of the dimension, and *fixed* regions are represented by red, blue, green, dark yellow, and light gray, respectively. Because various types of deformation were applied to only one tetrahedral mesh and the whole of the tetrahedral mesh was deformed, *deformable regions* were not extracted and the deformation process was applied to all vertices in this experiment. It is shown that the proposed method enables us to change the parameters of form features of tetrahedral meshes like 3D CAD system. For the tetrahedral mesh including 22,156 vertices, the parameterization of the vertex position took 1.36s (for the surface: 0.046s, for the inside: 1.3s), and vertex repositioning for the deformation took 0.063s at most. These results suggest that once the vertex positions are parameterized, one can interactively change various parameters of the form features of the tetrahedral mesh.

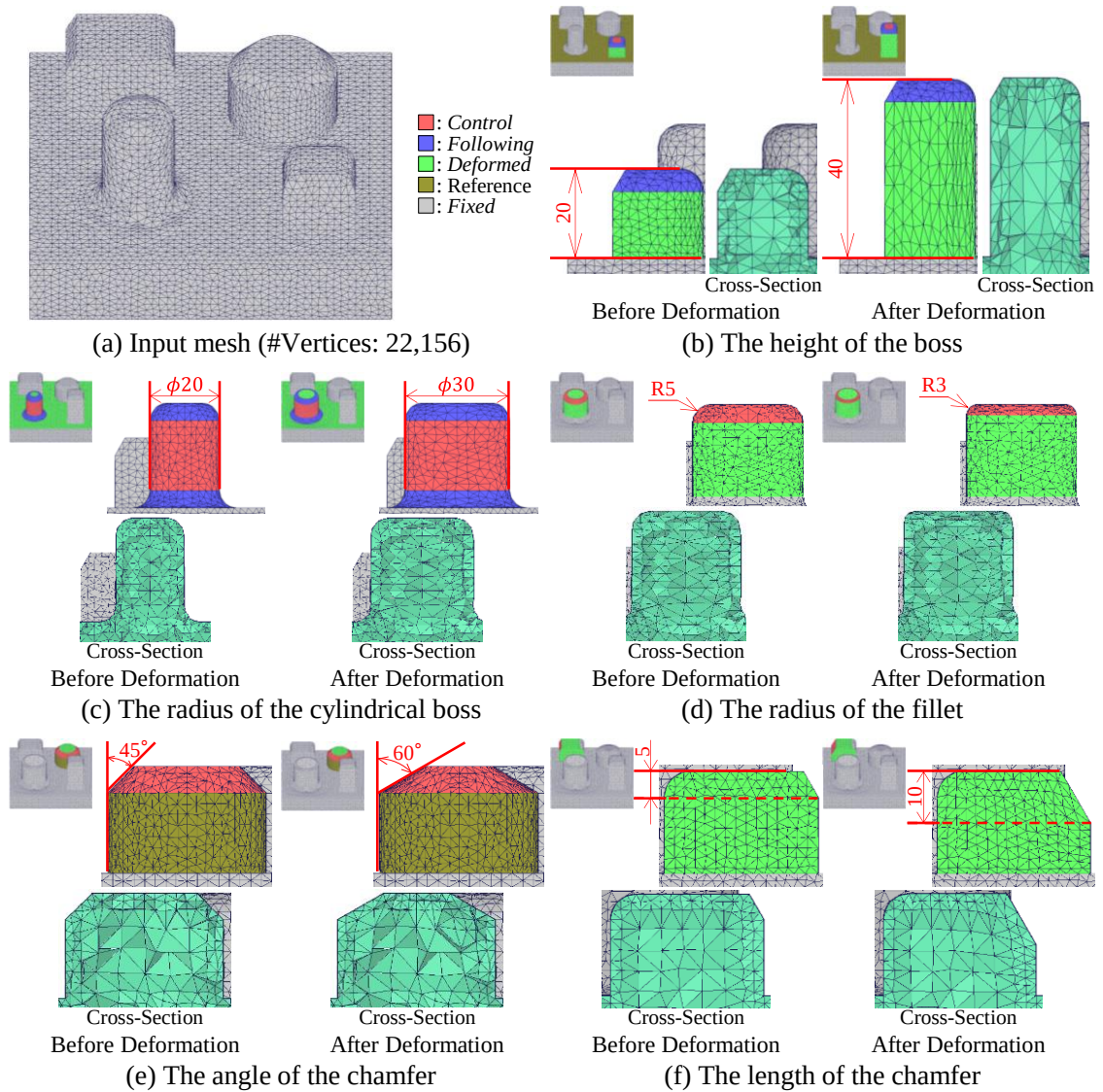


Fig. 2.50 The various deformation results of a tetrahedral mesh using proposed method

The input mesh (#Vertices: 94,706 and #Tetrahedra: 429,706) of a connecting rod and the deformed mesh by changing a distance between two parallel planes (i.e. the deformation type (1) mentioned in sub-subsection 2.5.1.2) are shown in Fig. 2.51(a) and (b), respectively. In Fig. 2.51, each surface triangle is color coded: triangles included in *control region*, *deformable region*, the reference of the dimension, and *fixed regions* are represented by red, green, dark yellow, and light gray, respectively. The distance between two parallel planes was changed from 25 to 40 by the proposed method. In Fig. 2.51, the cross-section of each tetrahedral mesh is also shown, and relative positions of inner vertices were preserved as much as possible after the deformation. The *deformable region* includes 5,609 vertices and 26,548 tetrahedra, which are 6 percent of all vertices and tetrahedra included in the input mesh. Table 2.12 shows the calculation time of each process. If the deformation process was applied to all vertices, the parameterization of the vertex position took 15.0s (for the surface: 0.27s, for the inside: 14.7s), and vertex repositioning for the deformation took 0.26s. On the other hand, if the *deformable region* was extracted and the vertex repositioning was applied to only the *deformable region*, the parameterization of the vertex position took 0.10s (for the surface: 0.022s, for the inside: 0.075s), and vertex repositioning for the deformation took 0.0074s. In this experiment, it is shown that the tetrahedral mesh of mechanical part can be deformed interactively by the proposed method.

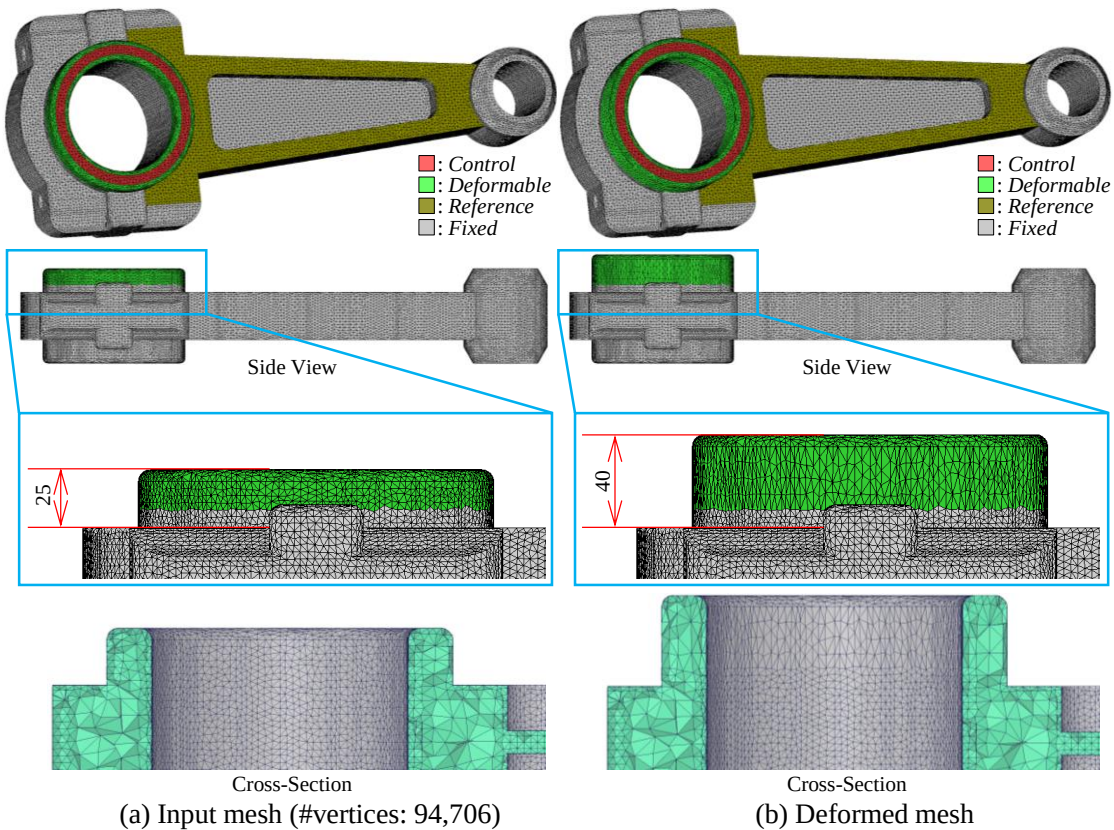


Fig. 2.51 Changing a distance between two parallel planes of a connecting rod

Table 2.12 Calculation time of changing a distance between two parallel planes

Process		Calculation time [s]	
		Using all vertices (#Vertices: 94,706)	Using only vertices in the <i>deformable region</i> (#Vertices: 5,609)
Surface region classification (and calculation of current dimension)		0.00040	0.00043
Deformable region extraction		-	0.063
Parameterization of vertices	Total	15.0	0.10
	Surface vertices	0.27	0.022
	Inner vertices	14.7	0.075
Vertex repositioning		0.26	0.0074
Total		15.2	0.17

Figure 2.52 shows the input mesh (a), the deformed mesh by the proposed method (b) and the cross-section of each tetrahedral mesh. In this experiment, the position of a cylindrical boss was changed by the proposed method. The cylindrical boss was moved 5 toward the right side along the u axis and 10 toward the upper side along v axis. In this experiment, the *deformable region* was expanded, and as shown in cross-section, inner vertices followed vertices of the cylindrical boss by the space embedding method. In Fig. 2.52, surface triangles were represented using the same colors with those in Fig. 2.51. The numbers of vertices and tetrahedra of the input mesh were 128,163 and 524,117 respectively, and the numbers of vertices and tetrahedra of the *deformable region* were 5,437 (4 percent of all vertices) and 28,090 (5 percent of all tetrahedra) respectively. As shown in Table 2.13, by local deformation using the *deformable region*, calculation time become much shorter. When the deformation process was applied to whole of the tetrahedral mesh, it took 5.41s. On the other hand, when it was applied only to the *deformable region*, the calculation time became 0.21s.

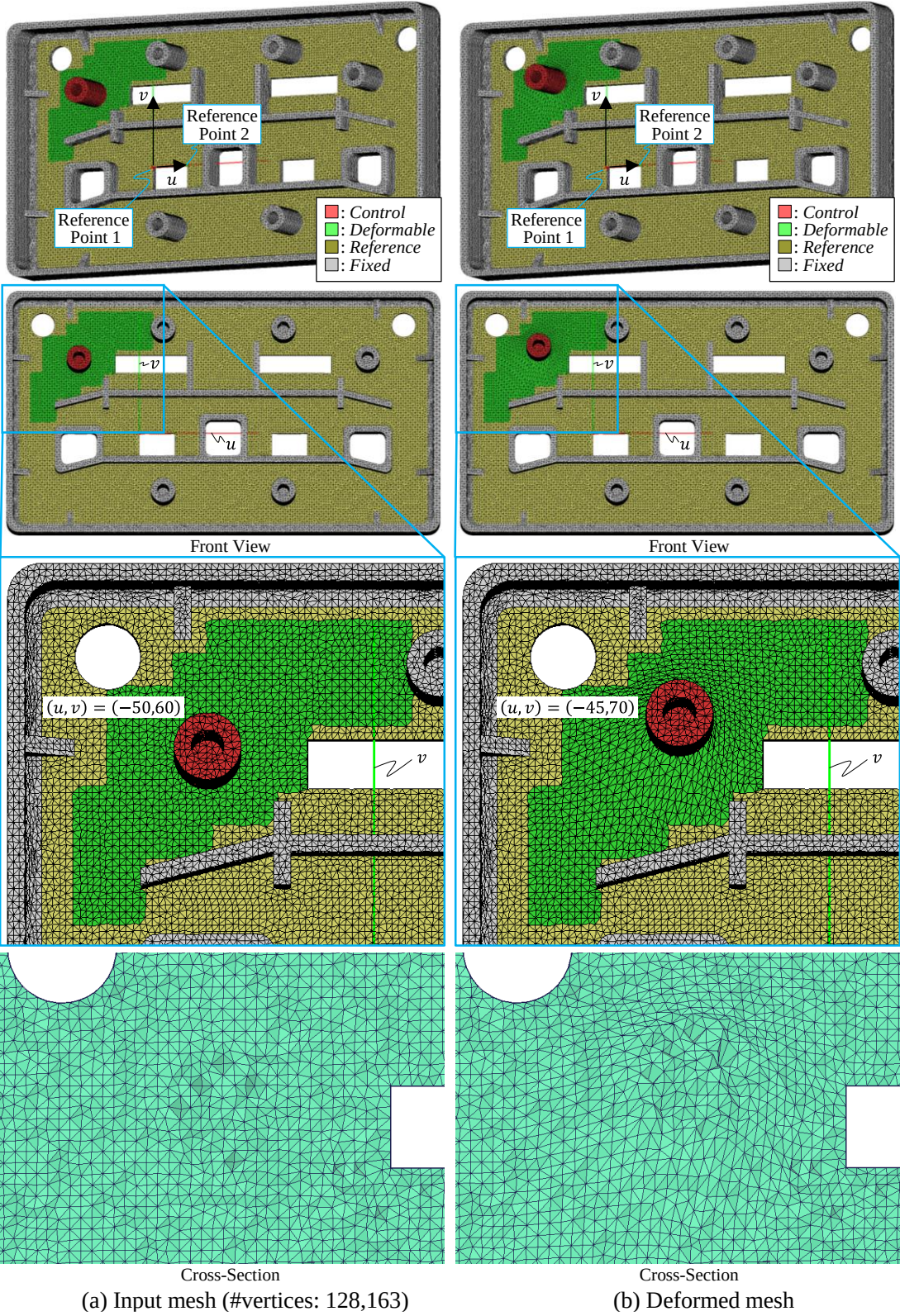


Fig. 2.52 Changing the position of a cylindrical boss on a plane of a cover

Table 2.13 Calculation time of changing the position of the cylindrical boss

Process		Calculation time [s]	
		Using all vertices (#Vertices: 128,163)	Using only vertices in the deformable region (#Vertices: 5,437)
Surface region classification		0.00024	0.00029
Deformable region extraction		-	0.047
Parameterization of vertices	Total	5.23	0.14
	Surface vertices	0.50	0.031
	Inner vertices	4.73	0.094
Vertex repositioning		0.19	0.019
Total		5.41	0.21

2.7.4 Mesh quality improvement

For the first and second examples, the effects of element shape quality improvement and recovering the original mesh properties are shown using two simple deformed meshes and compared with the one of original ODT smoothing [Chen 2011]. Then, the proposed method is applied to the deformation results mentioned in subsection 2.7.3. Finally, finite element analyses (FEAs) are performed using resultant mesh of the deformation, and results of FEAs are evaluated.

At first, the effectiveness of the proposed method to improve deformed non-uniform density tetrahedral meshes is shown. In order to evaluate mesh density of each mesh model, a mesh density error $\varepsilon_\rho(e)$ at each edge e is calculated by Eq. (2.35):

$$\varepsilon_\rho(e) = \frac{\left| \frac{1}{L(e)} - \rho^{GC}(\mathbf{c}_e) \right|}{\rho^{GC}(\mathbf{c}_e)}, \quad (2.35)$$

where $\rho^{GC}(\mathbf{c}_e)$ is the mesh density of the grid cell of the target mesh density field which includes the midpoint $\mathbf{c}_e$ of the edge e and $L(e)$ is the length of e .

For this experiment, the input tetrahedral mesh is generated by using NETGEN [Schöberl 1997] in LISA 8.00 [2012]. Figure. 2.53 shows the input mesh (a), the deformed mesh by changing a distance between two parallel planes (DP) using the proposed deformation method (b), the improved mesh using the original ODT smoothing (c), and the improved mesh using the proposed quality improvement method (d). Cross-section and stretch histograms are also shown in Fig. 2.53, and Table 2.14 shows the stretches and density errors in the each tetrahedral mesh. In the deformed mesh, tetrahedra were enlarged by the deformation and the average of mesh density errors became 0.26. By using the original ODT smoothing, the average of the mesh density errors became 0.24 and could not be improved significantly. On the other hand, the proposed method can reduce the mesh density error to 0.15, and as shown in cross-section in Fig. 2.53, the mesh density of the improved mesh was close to that of the input mesh. By the proposed method, the minimum stretch became 0.26, it was larger than that of the improved mesh using the original ODT smoothing. In addition, by the proposed method, the average of stretches became 0.73 and the ratio of tetrahedra having high stretches was

increased as shown in the stretch histogram. These results show that the proposed quality improvement method could improve the element shape qualities of the deformed meshes while recovering the mesh density of the input mesh in the deformed mesh. In the proposed method, because the insertion of many vertices was needed, edge split was repeated many times and the calculation time became 3630.2s (i.e. longer than one hour). Techniques of parallel processing for the vertex insertion and removal can be effective for solving this problem.

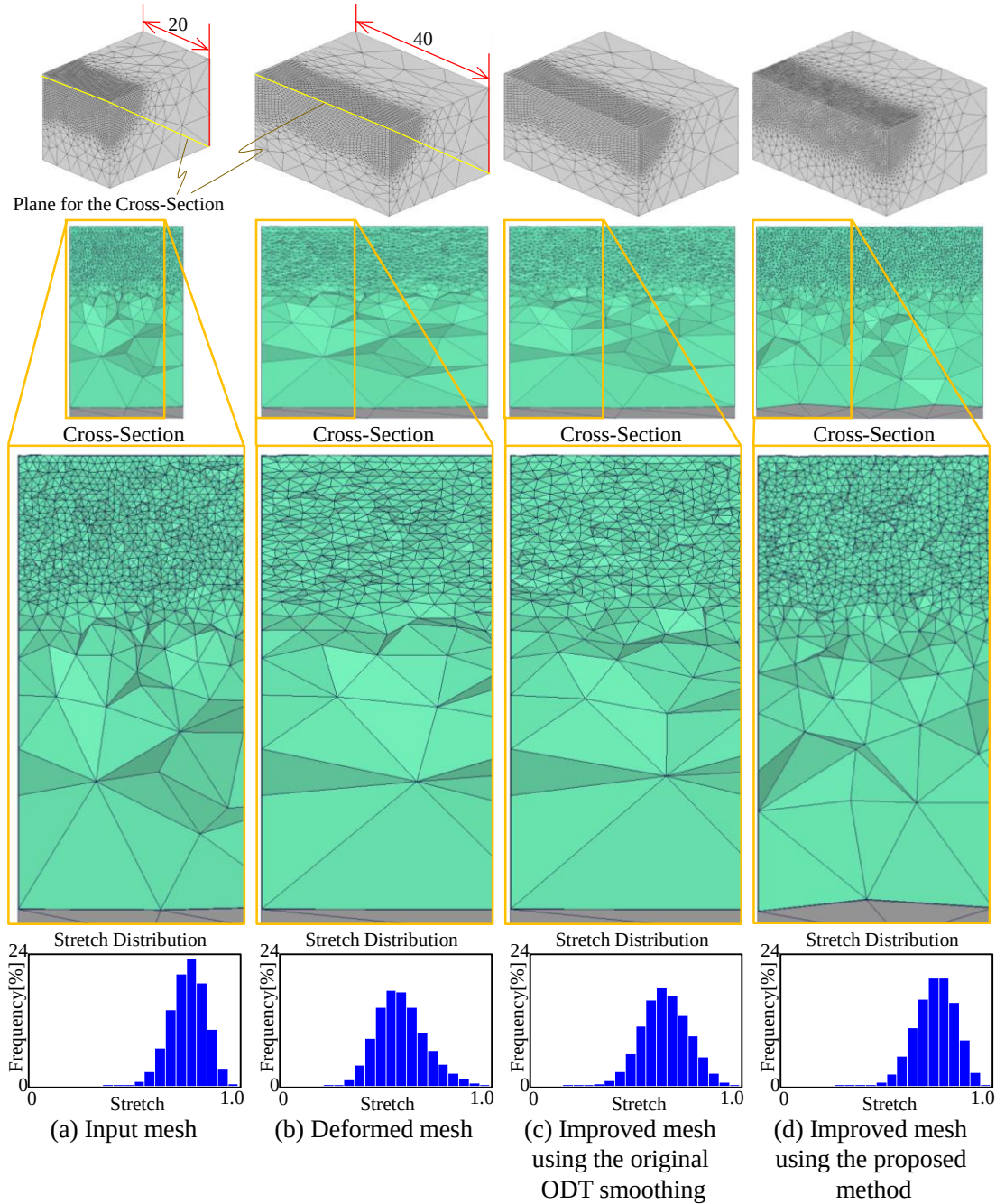


Fig. 2.53 Result of the improvement of the deformed mesh (changing DP)

Table 2.14 Mesh quality of each tetrahedral mesh (changing DP)

Mesh model	#Tetrahedra	Stretch		Density error	
		Average	Minimum	Average	Maximum
(a) Input mesh	72,575	0.69	0.49	-	-
(b) Deformed mesh		0.57	0.23	0.26	0.83
(c) Improved mesh (ODT)	72,444	0.64	0.16	0.24	1.28
(d) Improved mesh (proposed method)	144,028	0.73	0.26	0.15	1.62

Figure 2.54 shows the input mesh (a), the deformed mesh by changing a radius of a cylindrical hole (RC) (b), the improved mesh using the original ODT smoothing (c), the improved mesh using the proposed quality improvement method (d), and each stretch histogram. Table 2.15 shows the number of tetrahedra, the stretches, and geometric errors in each tetrahedral mesh. The calculation times of the original ODT smoothing and the proposed quality improvement method were 0.15s and 3.12s, respectively. Although the calculation time of the proposed method was longer than that of the original ODT smoothing, these results show that the proposed method could provide a deformed mesh which has shape approximation accuracy similar to that of the input mesh and good element shape qualities.

In this experiment, the radius cannot be enlarged more than 29 without inverted elements at one time even if the quality improvement were performed. However, if the radius was enlarged by two steps, in other words, if the improved mesh shown in Fig. 2.54(d) was used as the input of the proposed method, the radius could be enlarged more. The result of changing radius to 40 after the proposed quality improvement is shown in Fig. 2.55, and the minimum stretch and the maximum geometric errors in the improved mesh were 0.18 and 0.17, respectively. Therefore, for more stable deformation, the method which automatically divides a large scale deformation into some small scale deformations with the quality improvement may be effective.

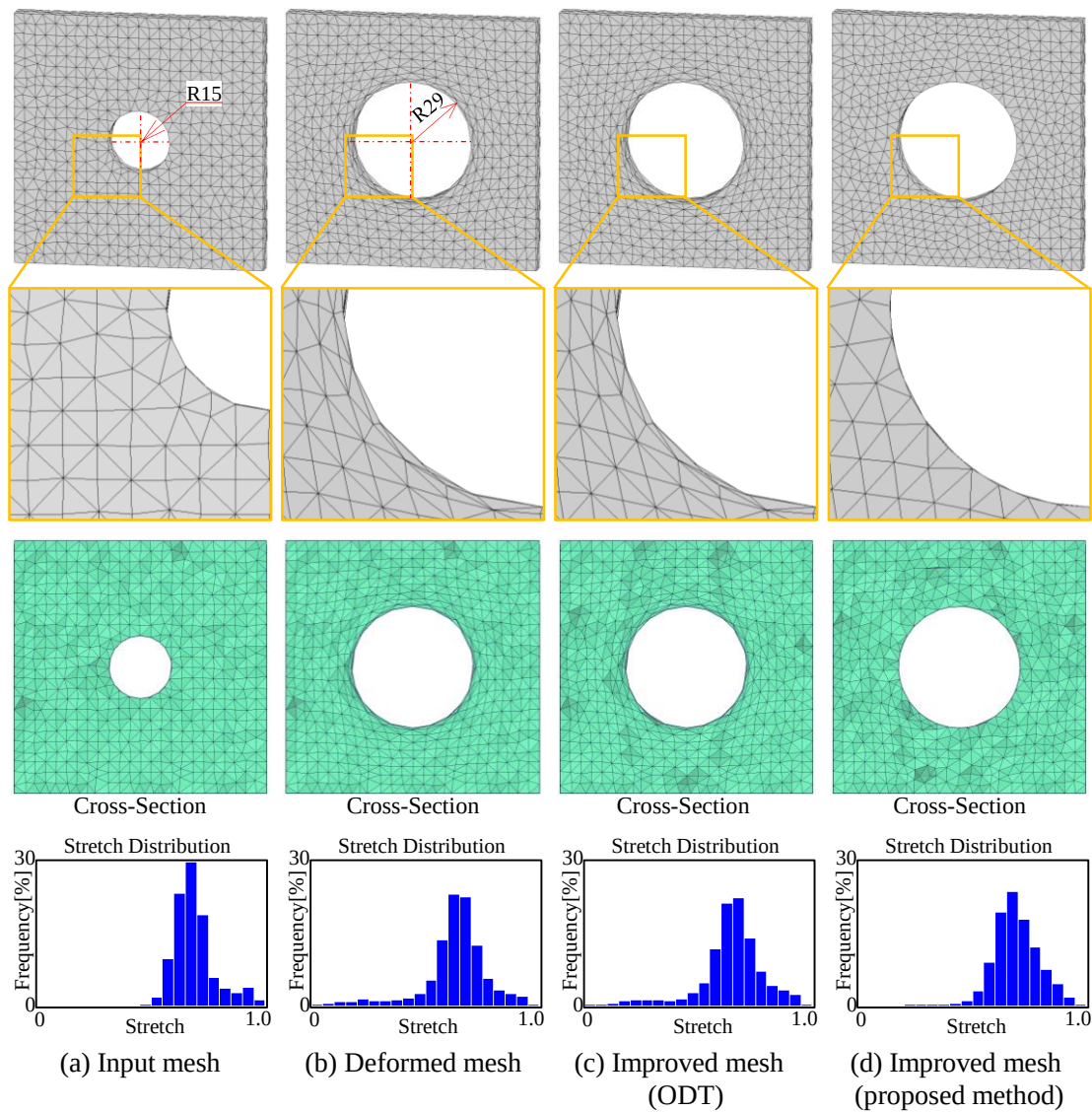


Fig. 2.54 Result of the improvement of the deformed mesh (changing RC)

Table 2.15 Mesh quality of each tetrahedral mesh (changing RC)

Mesh model	#Tetrahedra	Stretch		Geometric error	
		Average	Minimum	Average	Maximum
(a) Input mesh	6,138	0.69	0.49	0.071	0.32
(b) Deformed mesh		0.64	0.016	0.14	0.62
(c) Improved mesh (ODT)	6,149	0.65	0.016	0.14	0.62
(d) Improved mesh (proposed method)	5,774	0.69	0.23	0.039	0.27

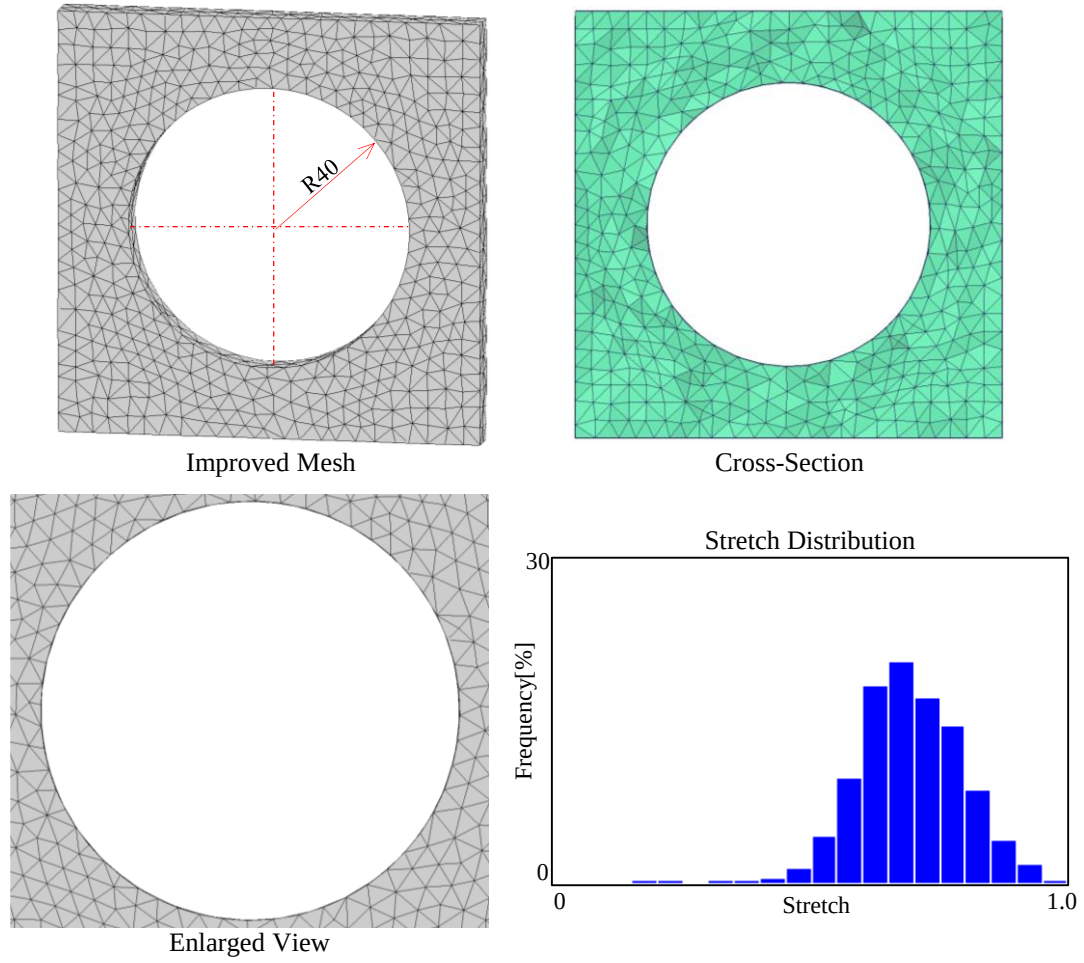
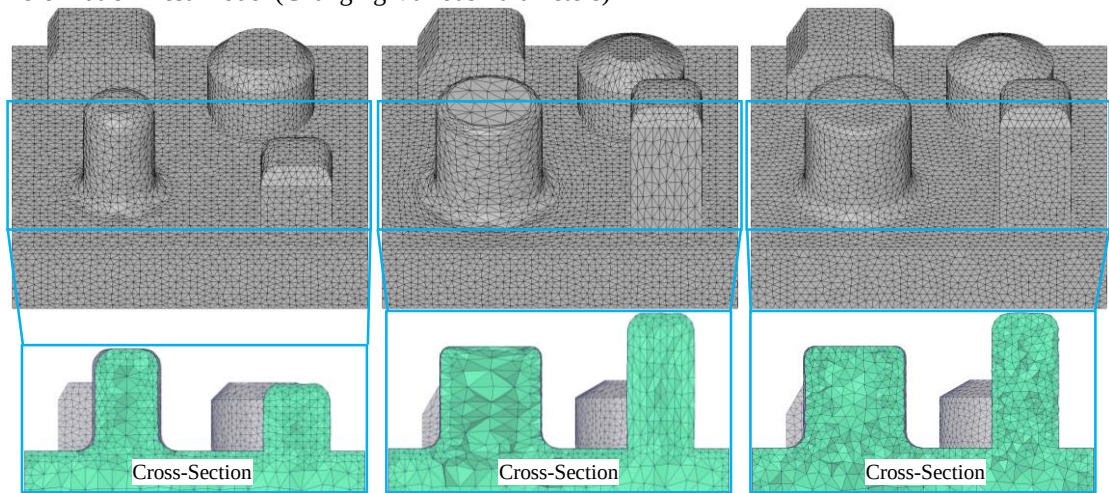


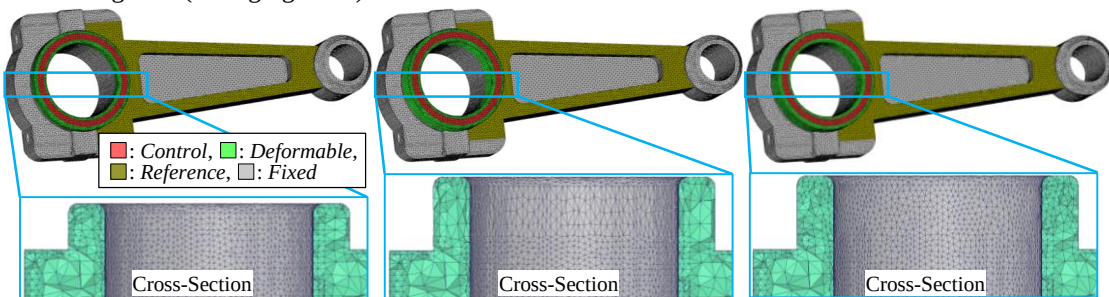
Fig. 2.55 Result of the changing RC by two steps using the proposed method

Then, the deformed meshes shown in subsection 2.7.3 were improved by the proposed quality improvement method. In Fig. 2.56 shows input meshes (a), deformed meshes (b), improved meshes (c), and each stretch histograms. Table 2.16 summarizes the result of the quality improvement. In the all of examples, the average and minimum stretches were improved even if surfaces of tetrahedral meshes include some conical, spherical, and torus surface regions. In addition, as shown in each cross-section, tetrahedra enlarged by the deformation were divided in to small tetrahedra which have high element shape qualities. In particular, although the deformed mesh of the third example (*Cover*) included many inverted and degenerated elements and the number of tetrahedra whose stretches were lower than the recommended lower limit (0.05) was 53 (#Inverted elements: 39 and #Degenerated elements: 14), such elements were removed by the proposed quality improvement method.

Deformation Test Model (Changing Various Parameters)



Connecting Rod (Changing a DP)



Cover (Changing the Position of a Boss)

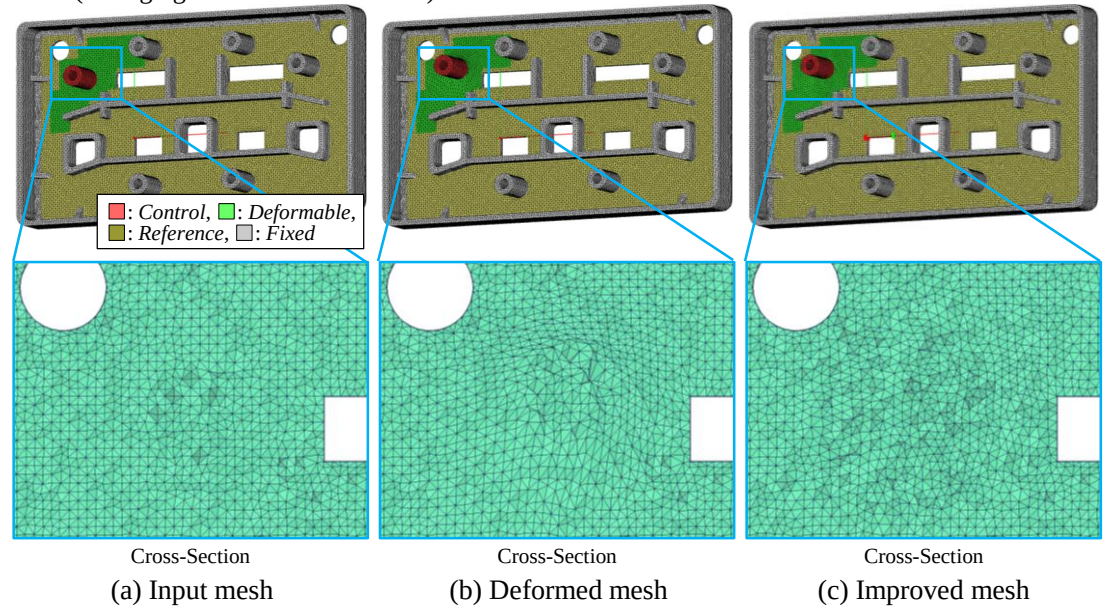


Fig. 2.56 Results of the quality improvement (mechanical parts)

Table 2.16 Results of the quality improvement (mechanical parts)

Mesh model	#Tetrahedra (#Inverted tetrahedra)			Stretch						Calculation time [s]
				Minimum			Average			
	(a) Input	(b) Deformed	(c) Improved	(a) Input	(b) Deformed	(c) Improved	(a) Input	(b) Deformed	(c) Improved	
Deformation test model (Fig. 2.50)	97,469 (0)		91,565 (0)	0.35	0.074	0.10	0.66	0.64	0.74	133.9
Connecting rod (Fig. 2.51)	429,702 (0)		452,033 (0)	0.29	0.22	0.24	0.65	0.65	0.66	697.7
Cover (Fig. 2.52)	524,117 (0)	524,117 (39)	525,333 (0)	0.31	−0.31	0.19	0.67	0.67	0.68	257.5

Finally, the resultant meshes of the method are evaluated by results of FEAs. At first, in order to show that resultant meshes of the method can be used for FEAs, a tetrahedral mesh of a cantilever beam is deformed, and the result of FEA using it is compared with the theoretical solution. Then, in order to check the effectiveness of the quality improvement, distributions of tensile stresses of a thin plate with a cylindrical hole are calculated through FEA using some resultant tetrahedral meshes with non-uniform density and compared with the theoretical solution. Each FEA was performed by LISA 8.00.

The input mesh of a cantilever beam (#Vertices: 290 and # Tetrahedra: 715) is shown in Fig. 2.57(a). The width of the beam was changed from 10 mm to 20 mm by the proposed method, and the resultant mesh (#Vertices: 552 and # Tetrahedra: 1,772) is shown in Fig. 2.57(b). In this experiment, 100 N was added to the top of the beam, and as the Young's modulus and the Poisson's ratio, 206 GPa and 0.3 were used, respectively. Therefore, the theoretical solution of the maximum displacement magnitude was 0.328 mm. The FEA result is shown in Fig. 2.58. The maximum displacement magnitude given by the FEA was 0.326 mm. Therefore, the error was about 0.002 mm (0.57%). On the other hand, the result of FEA using a tetrahedral mesh generated by meshing of NETGEN in LISA 8.00 was 0.325 mm. It means that resultant meshes of the proposed method can be used in FEA, and the accuracy of FEA using the resultant mesh is almost same as that using the mesh generated through meshing. In order to avoid shear locking problems, before each FEA, all elements of the tetrahedral mesh was converted from first order elements to second order elements.

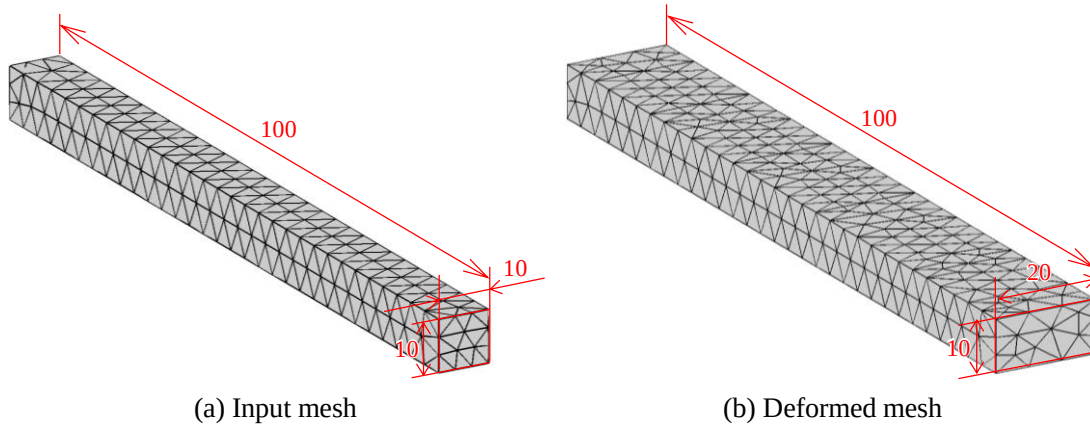


Fig. 2.57 The input mesh and resultant mesh of a cantilever beam

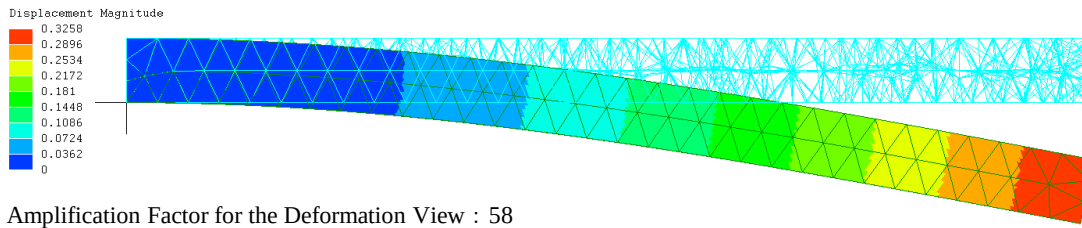


Fig. 2.58 The result of FEA using resultant mesh

The input mesh of the quarter of a thin plate with a cylindrical hole (#Vertices: 1,741 and #Tetrahedra: 7,797) is shown in Fig. 2.59(a). As shown in Fig. 2.59(b), in this experiment, the radius of the cylindrical hole was changed from 5 mm to 10 mm by the dimension-driven shape deformation with or without the quality improvement, and the plate was pulled by 100 N. In order to check the effectiveness, as shown in Fig. 2.60, three deformed meshes were generated. One was the deformed mesh where any qualities were not improved (a), another was the deformed mesh where only element shape quality was improved (b), and the other was the deformed mesh where not only element shape qualities but also mesh density and shape approximation accuracy were improved (c). The comparison of distributions of tensile stresses on the minimum section obtained by FEAs using resultant meshes with the theoretical solution is shown in Fig. 2.61 and the data of this experiment is shown in Table 2.17. All results obtained by FEAs using resultant meshes were close to the theoretical solution [Howland 1930]. In addition, the maximum error in the result of FEA using the improved mesh (c) was smaller than those of FEAs using the others. These results show that the quality improvement could improve the results of FEA.

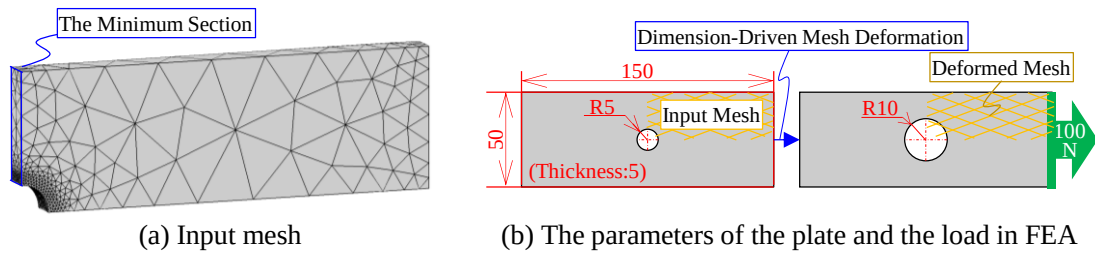


Fig. 2.59 The input mesh and parameters of the plate with a cylindrical hole for FEA

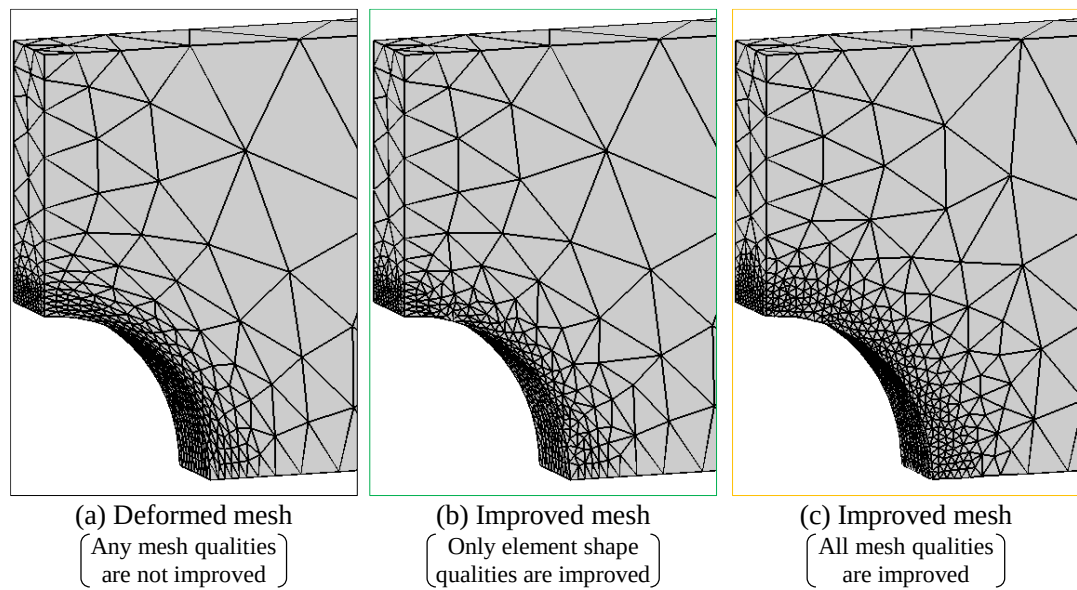
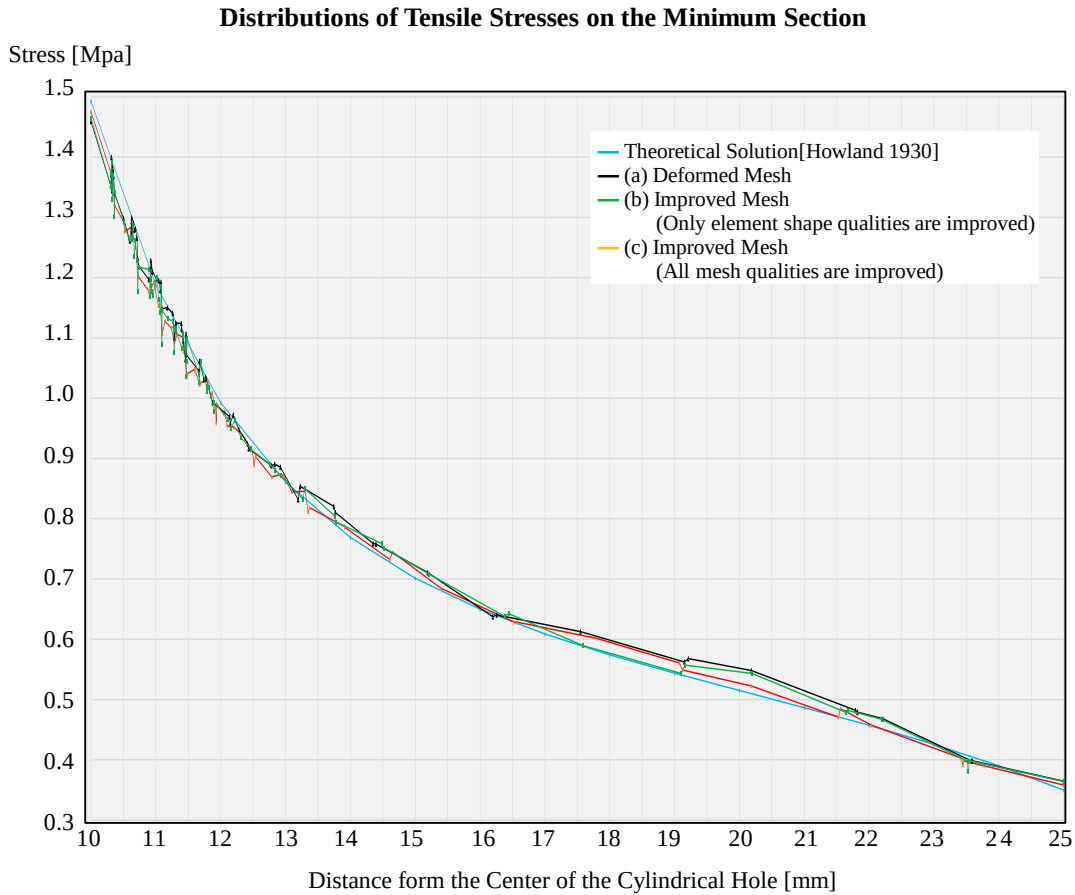


Fig. 2.60 The enlarged views of three deformed meshes

**Fig. 2.61 Distributions of tensile stresses on the minimum section****Table 2.17 The data of the experiments for the evaluation of quality improvement**

Mesh model	#Vertices	#Tetrahedra	Stretch		Error of tensile stress [%]	
			Minimum	Average	Maximum	Average
Input mesh	1,741	7,797	0.41	0.74	-	-
(a) Deformed mesh			0.069	0.52	7.6	2.0
(b) Improved mesh (Only element shape qualities are improved)	1,781	8,005	0.10	0.61	8.5	2.2
(c) Improved mesh (All mesh qualities are improved)	3,282	15,279	0.21	0.72	6.9	1.9

2.7.5 The evaluation of the generation time of tetrahedral meshes

Finally, in order to show the effectiveness of the dimension-driven mesh deformation method, the generation time of tetrahedral meshes using the deformation method was compared with that using meshing. The overview of this experiment is shown in Fig. 2.62. At first, the height of a cylindrical boss was changed from 35 to 25 in the CAD system using the solid model. Then, a tetrahedral mesh was generated by the meshing of the changed solid model and the time of the meshing was

measured. On the other hand, a tetrahedral mesh was generated by the meshing of the original solid model. After that, the height of a cylindrical boss was changed from 35 to 25 by the dimension-driven mesh deformation method and the time of the deformation with the quality improvement was measured. For the generation of a tetrahedral mesh whose element size is 1.6, the meshing took about 227s and the deformation took about 64s. It means that tetrahedral meshes could be generated more efficiently by the dimension-driven mesh deformation than by meshing. Figure 2.63 and Table 2.18 shows the processing times of the meshing and the deformation in other settings of the element size. In addition, table 2.18 shows numbers of elements of resultant meshes of them. These results show that the times of the generation of tetrahedral meshes by the dimension-driven mesh deformation were about one third of those of the meshing. In this experiment, as the meshing tool, NETGEN in LISA 8.00 was used.

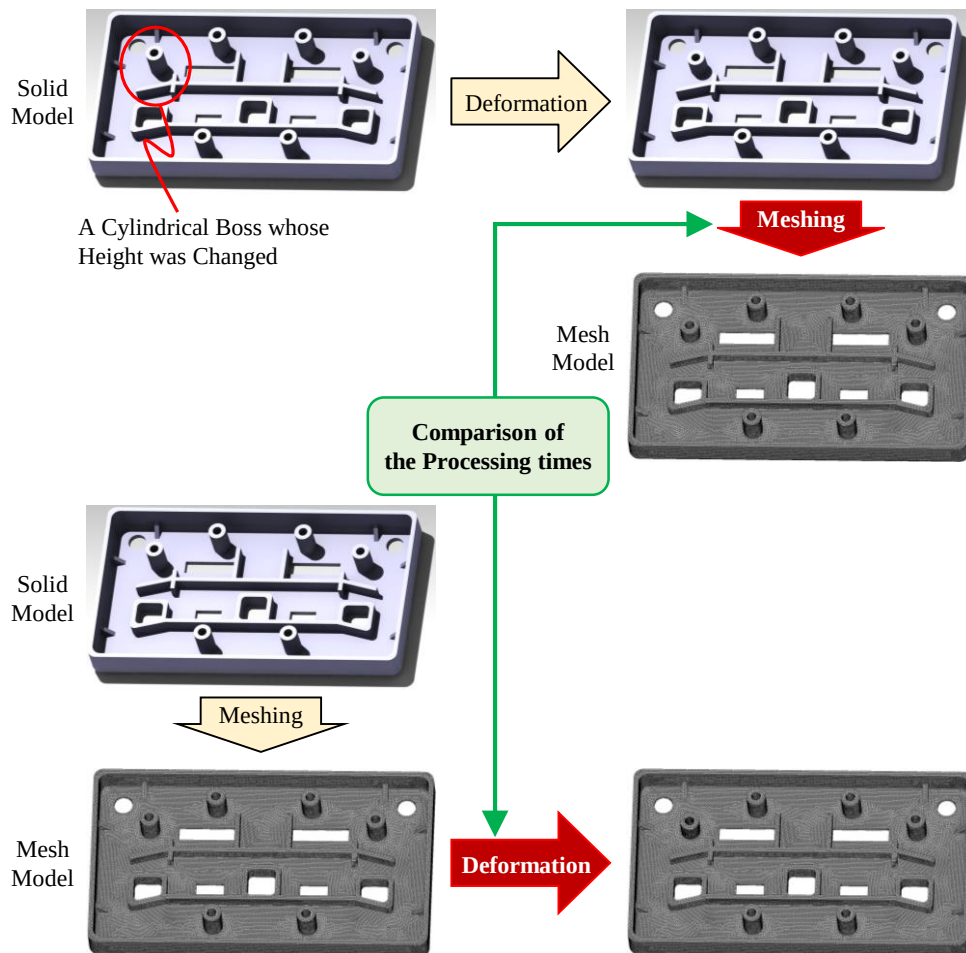


Fig. 2.62 The overview of the evaluation of the dimension-driven mesh deformation

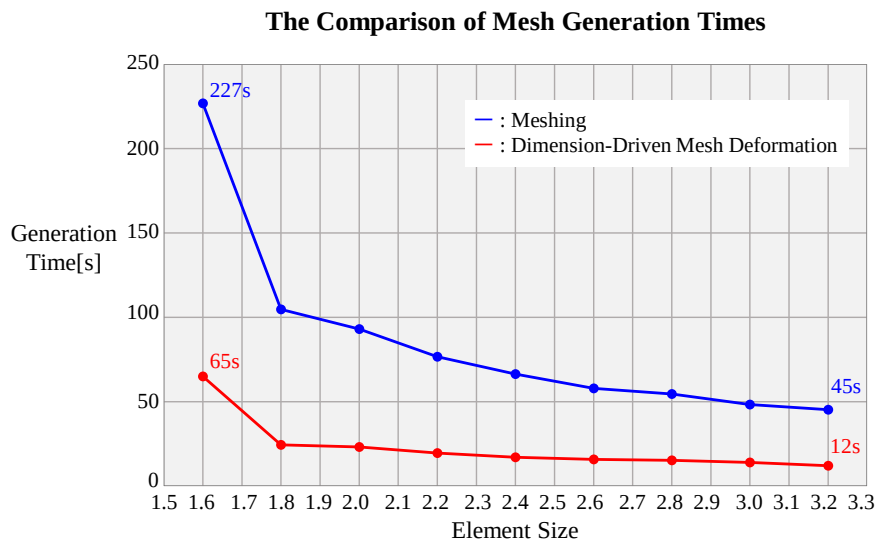


Fig. 2.63 The comparison of mesh generation times

Table 2.18 Results of the quality improvement (mechanical parts)

		Element size								
		1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0	3.2
Meshing	Calculation time	226.8	104.7	93.1	76.7	66.4	57.9	54.6	48.4	45.4
	#Vertices	222,807	107,983	95,664	81,282	69,972	63,297	57,190	52,218	46,992
	#Tetrahedra	991,433	372,858	343,764	288,654	250,368	228,288	206,149	188,101	167,737
Dimension-Driven Mesh Deformation	Calculation time	65.0	24.4	23.2	19.6	17.1	15.9	15.3	14.0	12.1
	#Vertices	221,575	108,835	95,628	82,025	70,821	64,259	58,004	52,749	47,305
	#Tetrahedra	984,293	377,618	343,495	292,860	255,337	233,301	210,424	191,207	170,197

2.8 Summary

In this chapter, a dimension-driven tetrahedral mesh deformation method was proposed. The proposed method consists of the following three functions.

- *Mesh segmentation based on normal tensor and region growing*: in this function, the surfaces of mesh models are divided into planar, quadric, and torus surface regions with G^0 or G^1 continuities by a step-by-step manner based on region-growing.
- *Dimension-driven shape deformation based on surface information and space embedding*: in this function, the surface regions of the input tetrahedral mesh are classified into four types of regions, and vertices are moved according to the region types.
- *Quality improvement based on Phased ODT smoothing*: in this function, in order to improve all elements of deformed meshes, ODT smoothing [Chen 2011] is applied to deformed meshes in three phases: boundary edge improvement, surface triangle improvement, and tetrahedron improvement. In addition, degenerated and inverted

elements are removed by edge split and edge collapse. Moreover, edge split and edge collapse based on the target mesh density field and the acceptable geometric error are combined with ODT smoothing in order to recover the original mesh properties.

In addition, experimental results shows the following facts.

- The proposed segmentation method could accurately divide the surface of tetrahedral meshes into planar, cylindrical, conical, spherical, and torus surface regions even if the surface regions connect with each other with G^0 or G^1 continuities. Although its speeding up may be needed, surface parameters of most surface regions could be extracted in accuracy of 10^{-5} .
- The proposed dimension-driven shape deformation method enables us to change parameters of the form features of tetrahedral meshes, such as the fillet radius and chamfer angle.
- The proposed quality improvement method could recover the mesh density and shape approximation accuracy of original tetrahedral mesh (before deformation) in deformed meshes. In addition, the average and distribution of element shape qualities were significantly improved. Moreover inverted and degenerated elements generated by the deformation were removed by the proposed method. As a problem of the proposed method, the processing time tends to become longer, and some speeding-up techniques such as a parallel processing will be needed for more efficiency.

The proposed method enables us to change parameters of the form features of tetrahedral meshes, such as the fillet radius and chamfer angle, while preserving mesh qualities such as the mesh density, shape approximation accuracy, and element shape quality.

Chapter 3 Tetrahedral Mesh Adaptation for Efficient Finite Element Analysis of Assembly Models

3.1 Functional requirements and organization of this chapter

For efficient generation of conformal tetrahedral finite element meshes of assembly models with moving parts, there are the following four functional requirements of mesh adaptation methods.

- *Efficient adaptation*: for efficient finite element analyses (FEA), tetrahedral meshes should be generated by the mesh adaptation method more efficiently than by re-meshing. In addition, the input original mesh model should be maintain as much as possible so that the interpolation for FEA between mesh models of each motion step is not performed as much as possible.
- *Keeping mesh conformity*: conformal meshes are preferred over non-conformal meshes because of the accuracy of FEA using them. Therefore, after each motion step, the conformity between the space mesh and the moving object mesh has to be kept. In addition, if a moving object contacts with other object meshes, keeping conformity between contacting object meshes is required.
- *Accurate contact detection*: if a moving object mesh is contact with other object meshes, the accurate detection of the contact is needed. In addition, in order to keep conformity between contacting object meshes, contact regions has to be extracted accurately from each contacting object mesh.
- *Generating high quality meshes*: adapted meshes often have many distorted elements and lose original mesh density. The element shape qualities of deformed meshes have to be improved because distorted elements cause low accuracy and inefficiency of FEA. In addition, deformed meshes should have the original mesh density, because change in the mesh density influences the accuracy of FEA and desired results may not be provided.

In this chapter, at first, existing mesh adaptation methods for moving objects and existing mesh adaptation methods for contacting objects (i.e. mesh merging methods) are introduced and their problems are discussed. Then, a new tetrahedral mesh adaptation method is proposed. The proposed method consists of local region extraction, mesh adaptation, and quality improvement. In section 3.3, the overview of the proposed method is described, and then (in section 3.4), the local region extraction using surface information and distance field is mentioned. For efficient mesh

adaptation, the mesh adaptation process is applied to only a set of space mesh elements (*deformed region*) around the moving object. In section 3.4, the way how to extract the deformed region is described. Moreover, contacts are detected and contact regions are extracted accurately using surface parameters. After that, in section 3.5, a mesh adaptation method using the two local regions is proposed. As mentioned above, for efficient mesh adaptation, the mesh adaptation process based on a space embedding method is applied to only the deformed region. In addition, in order to keep mesh conformity on the contact regions between object meshes, the geometry and topology of surface triangular meshes of contacting object meshes are adapted by vertex repositioning and local topological operations. Next, in section 3.6, the quality improvement method based on ODT smoothing [Chen 2011] for adapted tetrahedral meshes is proposed. In order to obtain a high quality mesh in each motion step, shape qualities of tetrahedral elements are improved and edge length is controlled by using a method based on ODT smoothing with local topological operations. In section 3.7, the effectiveness of the proposed method is demonstrated through application of the method to some tetrahedral meshes. Finally, this chapter is summarized in section 3.8.

3.2 Related work

3.2.1 Mesh adaptation method for object motion

There are many tetrahedral mesh adaptation methods for conformal mesh generation of moving objects and space. It is known that linear elasticity-based mesh adaptation methods are more robust and can produce higher quality meshes than Laplacian-based methods and spring analogy methods. In linear elasticity mesh adaptation methods, the movement of space mesh vertices is obtained by solving an elasticity-like equation.

Dobrzynski et al. [2008] a linear elasticity-based mesh adaptation method with a quality improvement for tetrahedral meshes. In the quality improvement, edge length is controlled by edge split using anisotropic Delaunay kernel and edge collapse while keeping element shape qualities by edge flipping and vertex repositioning.

Compère et al. [2010] also proposed a linear elasticity-based mesh adaptation method of tetrahedral meshes. In their method, edge length is controlled by edge split and edge collapse, and element shape qualities are improved by flipping operations. In addition, degenerated elements are removed by the combinations of edge split and edge collapse [Li 2003].

Alauzet [2014] proposed a linear elasticity-based mesh adaptation method where a give tetrahedral mesh is adapted to object motion while keeping the number of vertices. In Alauzet's method, trajectories of space mesh vertices are calculated at each motion step. If inverted elements are predicted by analyzing the trajectories, the time step (displacement of the moving object) is reduced to half. In addition, element shape qualities are improved by flipping operations and vertex repositioning. Barral et al. [2014] extended the method proposed by Alauzet. In their method, explicit Inverse Distance Weighted (IDW) interpolation is used instead of the linear elasticity mesh

adaptation. In comparison with the method proposed by Alauzet, the number of required motion steps is reduced.

By these methods, the conformal mesh of each object pose can be generated with few or no re-meshing process. These mesh adaptation methods based on linear elasticity or IWD need to adjust some parameters appropriately.

In [Sieger 2014] mentioned in sub-subsection 2.2.2.2, the applicability of their deformation method to a moving object in a space mesh is shown. Their deformation method based on Moving Least Square (MLS) technique can provide tetrahedral mesh whose elements shape qualities higher than those of tetrahedral meshes provided by Radial Basis Function-based deformation method [Sieger 2013], and any inverted elements are not generated in the experiment in [Sieger 2014]. However, the method based on MLS also needs the selection of several parameters such as the constraint weights, the number of sample points, and the support radii of the basis function.

For more efficiency and a higher quality of output meshes, Chen et al. [2015] proposed a parallel re-meshing method to remove elements with bad shape generated by mesh adaptation. In their experiment, the sequential re-meshing process took about 4,500 seconds and it was about 25 percent of the total simulation time. On the other hand, their parallel re-meshing process took about only 640 seconds and the re-meshing time became less than 5 percent of the total simulation time on 256 computer cores. Although it may need to be improved because it only achieves a 7.1 times speedup on 256 computer cores, the sum of re-meshing time and mesh adaptation (deformation) time became about 8 percent of the total simulation time by using their method (cf. when sequential re-meshing process was used, it was about 28 percent of the total simulation time).

By using these methods, the conformal mesh of each object pose can be generated more efficiently than re-meshing of the whole of mesh models. However, the mesh topology and geometry are globally adapted even if the differences in poses of the objects in motion are very small. In addition, these methods do not deal with contacts of the object meshes.

3.2.2 Mesh merging method

In order to create a conformal mesh of two or more contacting object meshes, surface meshes between object meshes have to be identical to each other.

Silva et al. [2010, 2014] proposed a mesh merging method for simplicial meshes (i.e. 2D triangular meshes and tetrahedral meshes) based on the weighted Delaunay triangulation. In their method, the overlapping regions of two mesh models are first represented by weighted Delaunay triangulations, and for each region, a lifted polytope is defined by its vertices and their weights. Then, weights of vertices in one region are adjusted so that lifted polytopes of two regions become identical. After that, a conformal weighted Delaunay triangulation (mesh model) is generated using only vertices whose lifted points are on the resultant lifted polytope (i.e. vertices whose lifted vertices are not inside of the resultant lifted polytope). In their method, explicitly maintaining

connectivity is not needed, and their method can handle large and complex tetrahedral meshes with arbitrary topology. In addition, high element shape qualities can be preserved by using an intermediate triangulation (called *fringe* in [Silva 2010]) even if two mesh models have different refinement levels. However, the additional operations for weights are needed for the concave shape and the operations can be painful tasks. In addition, because a weighted Delaunay triangulation is performed in their method, their merging method can be more inefficient than a simple meshing process (e.g. Delaunay triangulation) for large data sets.

Juntunen [2013] proposed a post-refinement method for conformal tetrahedral mesh generation from two tetrahedral meshes which are generated individually and contact with each other by planes. In Juntunen's method, vertices on a tetrahedral mesh are added to the other tetrahedral mesh and intersection points are inserted to both tetrahedral meshes as new vertices in order to make vertices identical in both tetrahedral meshes.

A similar procedure is performed in a conformal tetrahedral mesh generation method proposed by Song et al [2013]. In their method, a tetrahedral mesh near the surface is generated by advancing front method and a tetrahedral mesh of the remaining inner part is created by Delaunay triangulation. Next, the two tetrahedral meshes are merged. In their method, the geometry of the contacting boundary between two tetrahedral meshes is identical, but redundant vertices are added to the surface of the tetrahedral mesh generated by Delaunay triangulation. Therefore, in the mesh merging process, the redundant vertices are added to the tetrahedral mesh generated by the advancing front method.

Wang et al. [2013] proposed a method for linking two dissimilar structured hexahedral meshes. In their methods, vertices on the contact region are first repositioned so that distorted small elements are not generated by the later process. Then, contact regions between two hexahedral meshes are triangulated by insertion of vertices to positions of vertices of a hexahedral mesh, intersecting points of edges of two hexahedral meshes, and centers of quadrangles.

These three methods based on vertex insertion are very simple and robust. However, they have the potential for a drastic increase in the number of vertices on the contact regions. In addition, many distorted elements may be generated by their mesh merging processes.

On the other hand, Staten et al. [2008] proposed a mesh merging method for two hexahedral meshes in order to create a conformal mesh of assembly models. In their method, surface meshes of contacting objects are modified by some hexahedral dual operations in order to be the same. The quality of the resultant conformal hexahedral meshes is high and the increase in number of elements is moderate. However, the method is not completely automatic.

Chen et al. [2016] proposed a new chord-matching criteria which is used in the mesh merging method based on hexahedral dual operations [Staten 2008]. In their new criteria, instead of a spatial (geometrical) threshold used in the original method, topological properties of each chord (a sequence of quads on the contact surface) is used. Topological properties can provide more accurate and reliable matching chord sets without any adjustment of thresholds. In addition, they proposed a new

mesh merging method using the proposed criteria. Their method includes a mesh quality prediction strategy in order to speed up the sheet extraction for dual operations, and can marge hexahedral meshes with more complex contact surfaces automatically and efficiently. However, the topology change is expanded a large region including a region unrelated to contacts. In addition, element shape qualities of resultant conformal hexahedral meshes became low in some of their test cases.

3.2.3 Summary of related work

Table 3.1 and 3.2 shows the existing methods of mesh adaptation for object motion and mesh merging, respectively.

Most existing methods of mesh adaptation for object motion [Dobrzynski 2008, Compère 2010, Alauzet 2013] are based on the linear elasticity technique because linear elasticity-based mesh adaptation methods are more robust and can produce higher quality meshes than Laplacian-based methods and spring analogy methods. In [Barral 2014], it is shown that the IDW-based mesh adaptation method can provide better result than a linear elasticity-based mesh adaptation method. On the other hand, Sieger et al. [2014] shows that their deformation method based on MLS can be used for the adaptation of object motion in a space mesh without any inverted elements. By these methods, the conformal mesh of each object pose can be generated more efficiently than re-meshing. However, mesh adaptation methods based on linear elasticity, IWD, or Moving Least Square technique need to adjust some parameters appropriately. In addition, there are two problems. First, the mesh topology and geometry are globally adapted even if the differences in poses of the objects in motion are very small. Second, these methods do not deal with contacts of the object meshes.

Many mesh merging methods are also proposed [Silva 2010, Silva 2014, Juntunen 2013, Song 2013, Wang 2013, Staten 2008, Chen 2016]. However, existing mesh segmentation methods have any of the following problems.

- The merging method may be more inefficient than a simple meshing process.
- The merging method needs a special operation for concavities.
- The number of vertices on the contact regions can be increased drastically.
- Some or many distorted elements are generated.
- The merging method is not completely automatic.
- The topological change is expanded a large region including a region unrelated to contacts.

Table 3.1 Existing mesh adaptation methods for object motion

Method	Approach		Problems
	Vertex repositioning	Quality improvement	
Dobrzynski 2008	Linear elasticity	• Element shape quality improvement based on edge flipping and vertex repositioning • Edge length control based on edge split using anisotropic Delaunay kernel and edge collapse	• Inefficient (global adaptation) • Inapplicability to the contacts of the objects • Need of appropriate adjustment of some parameters
Compère 2010	Linear elasticity	• Element shape quality improvement based on flipping operations and vertex repositioning • Edge length control based on edge split and edge collapse • Degenerated element removal based on local topological operations [Li 2003]	
Alauzet 2014	Linear elasticity	• Element shape quality improvement based on flipping operations, vertex repositioning, and displacement control using vertex trajectory	
Barral 2014	IDW	• Element shape quality improvement based on flipping operations, vertex repositioning, and displacement control using vertex trajectory	
Sieger 2014	MLS	• None	

Table 3.2 Existing mesh merging methods

Method	Input mesh	Approach	Advantage	Problem
Silva 2010 Silva 2014	Triangular meshes or Tetrahedral meshes	Weighted Delaunay triangulation	• Applicability to the large and complex mesh models with arbitrary topology • Implicit connectivity maintenance	• Inefficient • Need of a special operation for concavity
Juntunen 2013	Triangular meshes or Tetrahedral meshes	Vertex insertion	• Applicability to the large and complex mesh models with arbitrary topology • Robust and simple operations	• Possibility of drastic increase of vertices • Existence of distorted elements
Song 2013	Triangular meshes or Tetrahedral meshes	Vertex insertion	• Applicability to the large and complex mesh models with arbitrary topology • Robust and simple operations	• Possibility of drastic increase of vertices • Existence of distorted elements
Wang 2013	Quadrilateral meshes or Hexahedral meshes	Vertex insertion Vertex repositioning	• Applicability to the large and complex mesh models with arbitrary topology • Robust and simple operations	• Possibility of drastic increase of vertices
Staten 2008	Hexahedral meshes	Hexahedral dual operations	• Moderate increase of vertices • High quality elements	• Not completely automatic • Expansion of topological change
Chen 2016	Hexahedral meshes	Hexahedral dual operations	• Moderate increase of vertices • High quality elements • More efficiently than [Staten 2008] • Applicability to more complex contact surfaces in comparison with [Staten 2008]	• Expansion of topological change • Existence of distorted elements

3.3 Overview of proposed method

Based on the existing methods and their problems mentioned in section 3.2, in this chapter, a new tetrahedral mesh adaptation method for object motion with contact is proposed. In the proposed method, for efficient mesh adaptation, the mesh adaptation and quality improvement process is applied to only a set of space mesh elements around the moving object. In addition, in the mesh

adaptation process, because Mean Value Coordinates (MVC) [Ju 2005] which is used for vertex repositioning of some mesh deformation methods (e.g. [Takano 2010, Sun 2016]) does not need to adjust any parameters unlike linear elasticity and Inverse Distance Weighted (IDW), the vertex repositioning based on MVCs is used instead of linear elasticity and IWD. Moreover, the proposed method includes an automatic adaptation method for keeping mesh conformity on the contact regions between object meshes while avoiding rapid increase of vertices. In the proposed method, geometry and topology of surface triangular meshes of contacted object meshes are adapted by vertex repositioning and local topological operations.

Inputs of the proposed method are a conformal tetrahedral mesh composed of space mesh M^S and object meshes $\{M_i^O\}$ and a sequence of 4×4 homogeneous rigid transformation matrices for the object motion $(\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_n)$, which guarantees that objects do not penetrate others. In the proposed method, it is assumed that the contact region consists of planar, cylindrical, conical, spherical, or torus surfaces which are used often in the design of many mechanical parts. In addition, it is assumed that contact regions of two object meshes are the same surface type and surface parameters. In this chapter, the moving object mesh is described by M_m^O and an object mesh which contacts with M_m^O is described by M_c^O .

The overview of the proposed tetrahedral mesh adaptation method is shown in the Fig. 3.1. Our method consists of the following three steps: local region extraction (A1), mesh adaptation (A2) and quality improvement (A3).

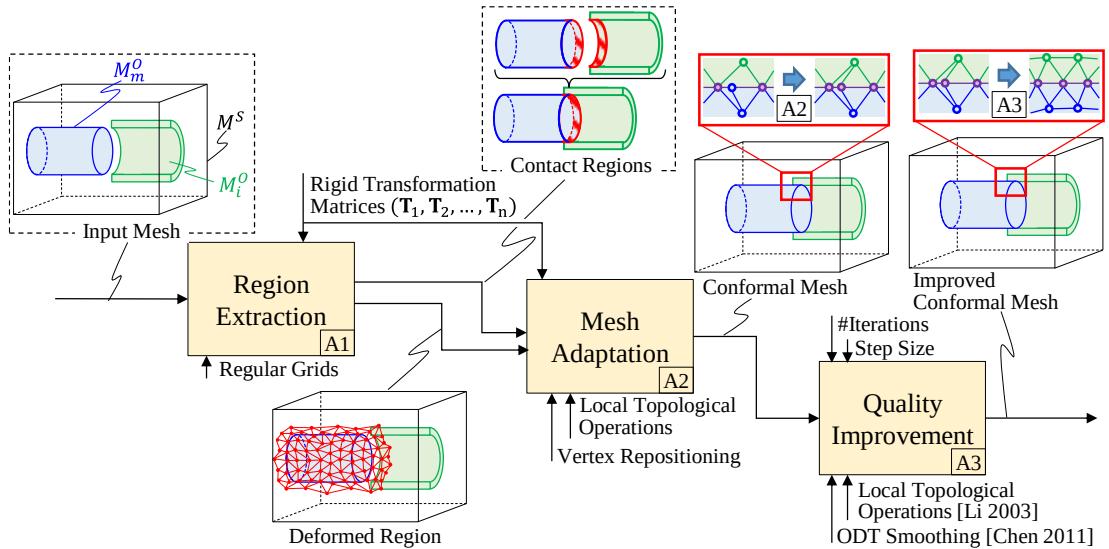


Fig. 3.1 The overview of the proposed tetrahedral mesh adaptation method

In the local region extraction process (A1), two regions, called the “contact region” and “deformed region,” are extracted in each motion step. A contact region is a set of vertices of object meshes, which are on the overlapping area of surface of two object meshes. On the other hand, a deformed region is a set of tetrahedral elements of M^S around the moving object mesh M_m^O . In the proposed method, for efficient extraction of these two regions, two regular grids (G_d and G_v) are used. In the next section, the way how to generate these two grids and extract these two region is described.

In the mesh adaptation process (A2), the moving object mesh M_m^O is moved and the space mesh M^S is deformed according to the motion of M_m^O . For efficiency, vertices of M^S only in the deformed region are repositioned. In the vertex repositioning, because MVC does not need to adjust any parameters, a space embedding method using MVCs is adopted instead of linear elasticity and IWD. If M_m^O contacts with other object meshes, in order to keep the mesh conformity, the contact regions are adapted by vertex repositioning and local topological operations.

In the quality improvement process (A3), inverted and degenerated elements are first removed by local topological operations [Li 2003]. In addition, the mesh density is controlled by vertex insertion and vertex removal. Moreover, element shape qualities of the deformed region are improved by ODT smoothing [Chen 2011].

3.4 Local region extraction using surface information and distance field

3.4.1 Grid generation

As preprocessing, for efficient search of vertices in M_c^O and M^S near M_m^O , two regular grids (*distance field grid* G_d and *vertex search grid* G_v) are generated (see Fig. 3.2(a)). The G_d represents a distance field from the surface of M_m^O , and moves together with M_m^O . On the other hand, G_v stores vertices of the given tetrahedral mesh in each cell and is fixed.

In the generation of G_d , at first, a regular grid which covers the whole of the given tetrahedral mesh is generated. Then, the distance between each cell and the surface of M_m^O is calculated by following the simplified fast marching method [Sethian 2001] as shown in Fig. 3.2(b).

- Step 1 Cells including vertices of M_m^O are identified as “trial cells” and pushed to a trial cell list and their distance values are set to 0. On the other hand, other cells are labeled as “far cells” and their distance values are set to infinity.
- Step 2 A cell c_{min} whose distance value d_{min} is the minimum in the trial cell list is popped and it is labeled as a “known cell.” Its neighboring far cells are labeled as “trial cells” and pushed to the trial cell list, and their distance values are set to $d_{min} + 1$. This step is repeated until the distance value of a popped cell is larger than a threshold δ^{dist} .

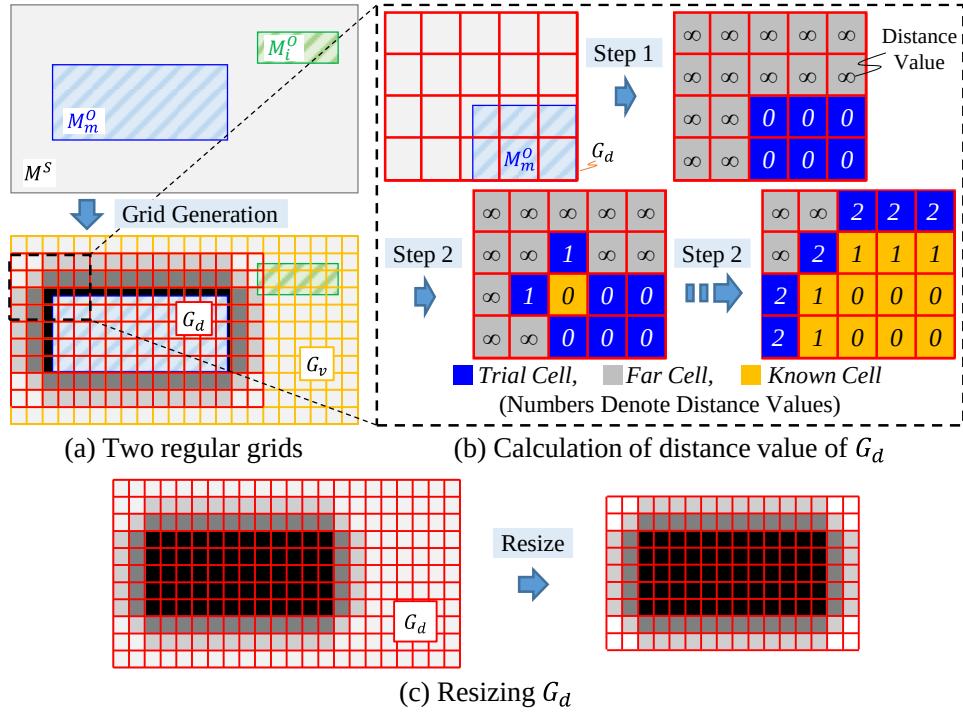


Fig. 3.2 Two regular grid and generation of the distance field grid

Finally, as shown in Fig. 3.2(c), for efficient calculation, G_d is resized so that G_d consists of cells included in an Axis Aligned Bounding Box (AABB) which includes cells whose distance values are smaller than δ^{dist} .

On the other hand, in the generation of G_v , at first, a regular grid which covers the whole of the given tetrahedral mesh is generated. After that, each vertex of the given tetrahedral mesh is stored in each cell which includes it.

The size of cells of both grids should be large enough compared to the maximum displacement of vertices of M_m^O .

3.4.2 Contact region extraction

If the contact detection and the contact region extraction are performed by using only object meshes, as shown in Fig. 3.3(a), it is difficult to find all vertices on the contact regions. On the other hand, if surface types and surface parameters of the object meshes are used for the detection of contacts and the extraction of contact regions, as shown in Fig. 3.3(b), all vertices on the contact regions can be extracted easily. Therefore, in the proposed method, a mesh segmentation method proposed in section 2.4 is applied to each object mesh as a preprocessing, and the surface information mentioned in sub-subsection 2.5.1.1 is used for the contact region extraction.

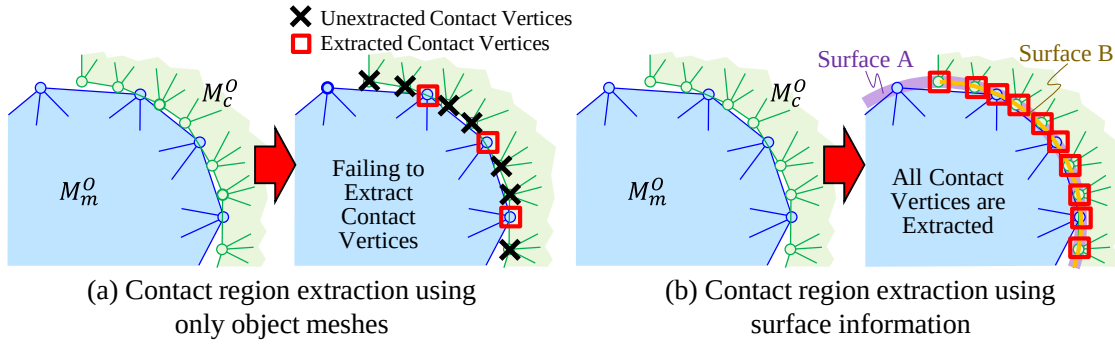


Fig. 3.3 The need of surface information for the contact region

In each motion step, if M_m^O contacts with other object meshes, the contact regions are extracted. In this process, M_c^O is first detected using G_d . Then, surface regions overlapping with other surface regions, boundary line segments and sets of vertices on the overlapping area are extracted using surface information. As shown in Fig. 3.4, this process consists of the following three steps.

- Step 1 The rigid transformation $\mathbf{T}_i$ is first applied to M_m^O and G_d (see Fig. 3.4(a)). At the same time, the surface information of M_m^O is also updated based on $\mathbf{T}_i$. All object meshes which have vertices included in cells of G_d whose distance value is 0 are detected as M_c^O which contact with M_m^O .
- Step 2 Surface regions of M_m^O and M_c^O with the same surface type and surface parameters are extracted as contact surface regions S_m^O and S_c^O , respectively (see Fig. 3.4(b)). Boundary line segments B_O of the overlapping area are extracted using boundary information of S_m^O and S_c^O .
- Step 3 Two sets of vertices $V_m^O \in M_m^O$ and $V_c^O \in M_c^O$ in the overlapping area are extracted as contact regions. As shown in Fig. 3.4(c), for accurate extraction of the vertices, the inclusion check of each vertex using B_O is performed on the following 2D space defined by surface parameters of S_m^O and S_c^O .
- If S_m^O and S_c^O are cylindrical or conical surface regions: the development planes for the cylindrical or conical surfaces. (e.g. for cylindrical surface, the 2D space is defined by height z along their axis $\mathbf{a}_{cyl}$ and width obtained by length of arc $r\theta$ as shown in Fig. 3.4(c).)
 - If S_m^O and S_c^O are spherical surface regions: the 2D parameter space defined by the polar angle θ^{sp} and the azimuth angle φ^{sp} .
 - If S_m^O and S_c^O are torus surface regions: the 2D parameter space defined by the poloidal angle φ^{tori} and the toroidal angle θ^{tori} .

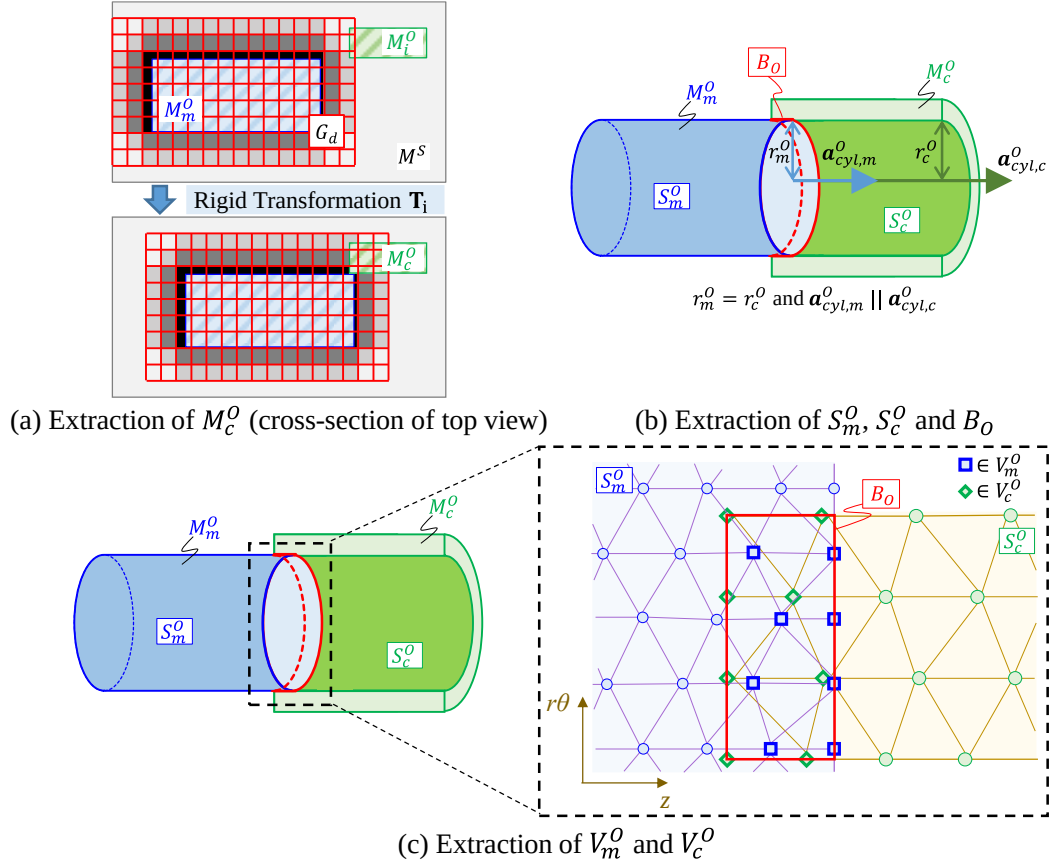


Fig. 3.4 The overview of the contact region extraction

3.4.3 Deformed region extraction

In each motion step, for efficient mesh adaptation, a set of tetrahedral elements of M^S near M_m^O is extracted as the deformed region using G_d and G_v . The overview of the deformed region extraction is shown in Fig. 3.5. In this process, at first, the position and surface information of M_m^O and the position of G_d are returned to those before the contact region extraction by performing the inverse transformation of $\mathbf{T}_i$. Then, a subset of cells C_{v1} of G_v whose center points are included in the AABB of G_d is extracted (Fig. 3.5(a)). After that, distance values are assigned to each cell $c_{(i,j,k)} \in C_{v1}$ from each cell of G_d which includes the barycenter of $c_{(i,j,k)}$. Next, a set of cells C_{v2} of G_v whose distance values are lower than a threshold is extracted (Fig. 3.5(b)). Finally, a set of tetrahedral elements of M^S with vertices included in cells $c_{(i,j,k)} \in C_{v2}$ is extracted as the deformed region K_{Df} (Fig. 3.5(c)).

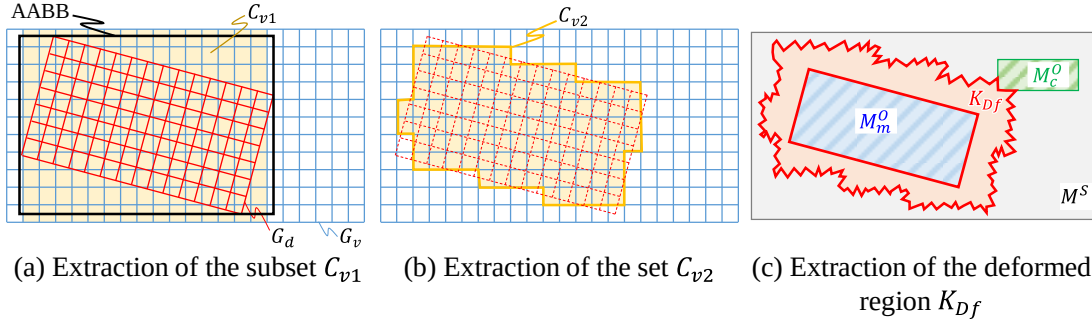


Fig. 3.5 The overview of the deformed region extraction

3.5 Mesh adaptation using two local regions

3.5.1 Overview of the mesh adaptation

The overview of the mesh adaptation process (Fig. 3.1, A2) is shown in Fig. 3.6. This process consists of the following five steps: (i) space mesh element removal, (ii) rigid transformation, (iii) deformed region adaptation, (iv) contact region adaptation, and (v) re-triangulation. If M_m^O contacts with other object meshes, all steps are performed. Otherwise, only (ii) rigid transformation and (iii) deformed region adaptation are performed. In the following subsections, the detail of each step is described.

3.5.2 Space mesh element removal

In the proposed method, if M_m^O contacts with other object meshes, in order to avoid degenerated elements, tetrahedral elements of M^S near the contact region are removed (see Fig. 3.7). In this process, at first, tetrahedral elements of M^S with vertices included in the contact region (i.e. $v_{cr} \in V_m^O \cup V_c^O$) or their neighboring vertices v_{ne} in M_m^O and M_c^O are removed, and a hole H^{rem} is created. Here, vertices of M^S on the surface of H^{rem} except for vertices of M_m^O and M_c^O are denoted by v_H . Then, because v_H have the possibility of penetrating M_m^O after the object motion, tetrahedral elements of M^S with v_H whose distances from the surface of M_m^O are smaller than a threshold δ^H are removed. The second type of removal is repeated until the distances between all v_H and the surface of M_m^O are larger than a threshold. The distance threshold δ^H is determined based on the maximum displacement of vertices of M_m^O .

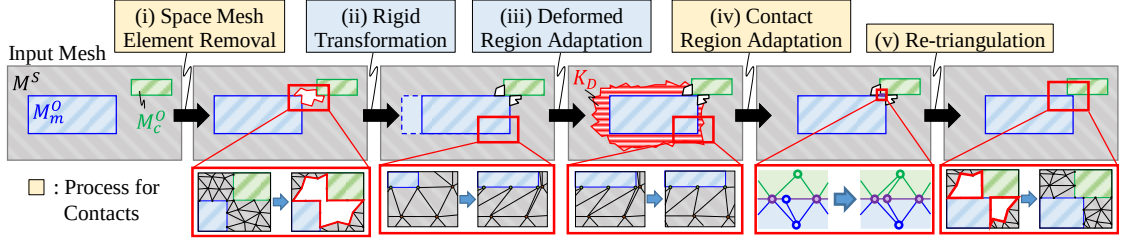


Fig. 3.6 The overview of the mesh adaptation process

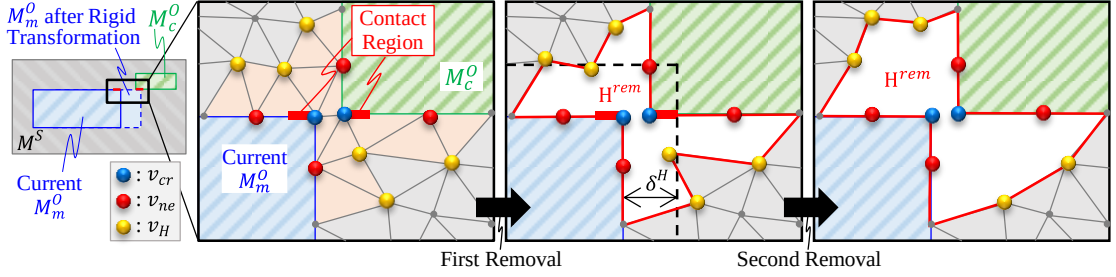


Fig. 3.7 Space mesh element removal

3.5.3 Rigid transformation and deformed region adaptation

In the proposed method, M_m^O is moved by the rigid transformation and inner vertices V^{idf} of K_{Df} are moved according to the motion of M_m^O . In the existing methods of mesh adaptation, linear elasticity and IWD are used for adaptation of the position of vertices in M^S according to the motion of the objects. However, some parameters need to be adjusted appropriately. In the proposed method in this thesis, for simple calculation, a space embedding method based on Mean Value Coordinates (MVC) [Ju 2005] similar to section 2.5 which does not need to adjust any parameters is adopted.

In this space embedding method, for each vertex $i \in V^{idf}$, vertices $j \in N_i^{vtx}$ (N_i^{vtx} is a set of neighboring vertices of i) are used as handle vertices, and MVCs for a polyhedron consisting of these handle vertices are used as parameter ω_j . Therefore, vertices $i \in V^{idf}$ are moved by the following three steps.

Step 1 The position $\mathbf{x}_i$ of each vertex $i \in V^{idf}$ is represented by a linear combination of its neighboring vertices $j \in N_i^{vtx}$ as shown in Eq. (3.1) (see Fig. 3.8(a)):

$$\mathbf{x}_i = \sum_{j \in N_i^{vtx}} \omega_j \mathbf{x}_j. \quad (3.1)$$

Step 2 A rigid transformation matrix $\mathbf{T}_i$ is applied to the M_m^O and G_d , and the surface information of M_m^O is updated based on $\mathbf{T}_i$.

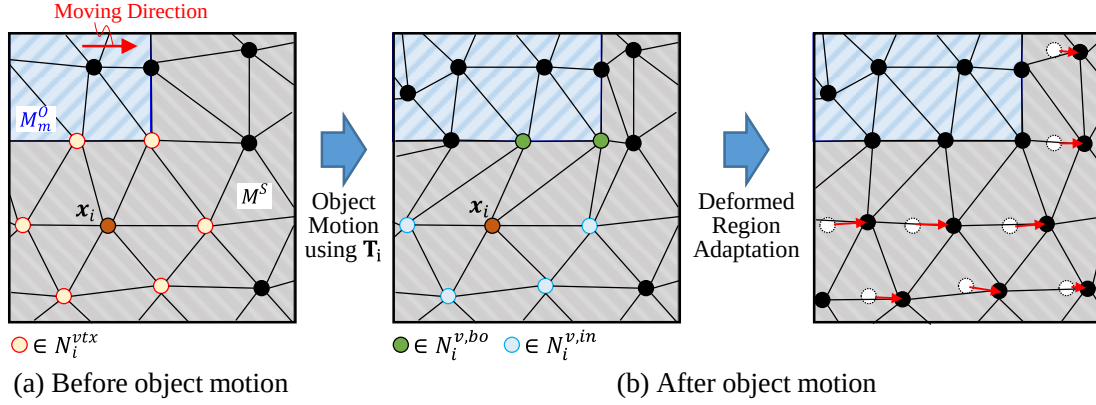


Fig 3.8 Rigid transformation and deformed region adaptation

Step 3 The position of each vertex $i \in V^{idf}$ is derived as a solution of the following system of linear equations:

$$\mathbf{x}_i - \sum_{j \in N_i^{v,in}} \omega_j \mathbf{x}_j = \sum_{k \in N_i^{v,bo}} \omega_k \mathbf{x}_k, \quad (3.2)$$

where $N_i^{v,bo}$ is a set of vertices of fixed boundary of K_{df} and M_m^O in N_i^{vtx} , whose positions are known after movement of M_m^O . $N_i^{v,in}$ is $N_i^{vtx} - N_i^{v,bo}$ (see Fig. 3.8(b)). The derived positions $\mathbf{x}_i$ preserve relative location to neighbors before the object motion as much as possible.

3.5.4 Contact region adaptation

3.5.4.1 Overview

In the proposed method, to maintain mesh conformity on the contact regions between object meshes without rapid increase of vertices, the geometry and topology of surface triangular meshes in the contact regions of M_m^O and M_c^O are adapted by vertex repositioning and local topological operations.

The overview the contact region adaptation is shown in Fig. 3.8. At first, the boundary lines B_O of the overlapping area is represented by edges of M_m^O and M_c^O (Fig. 3.9(A)). Secondly, positions of V_m^O and V_c^O are corresponded (Fig. 3.9(B)). After that, numbers of V_m^O and V_c^O are corresponded (Fig. 3.9(C)). Then, connectivity of surface triangular meshes on the contact regions of M_m^O and M_c^O is made identical (Fig. 3.9(D)). Finally, the element shape qualities of M_m^O and M_c^O are improved (Fig. 3.9(E)).

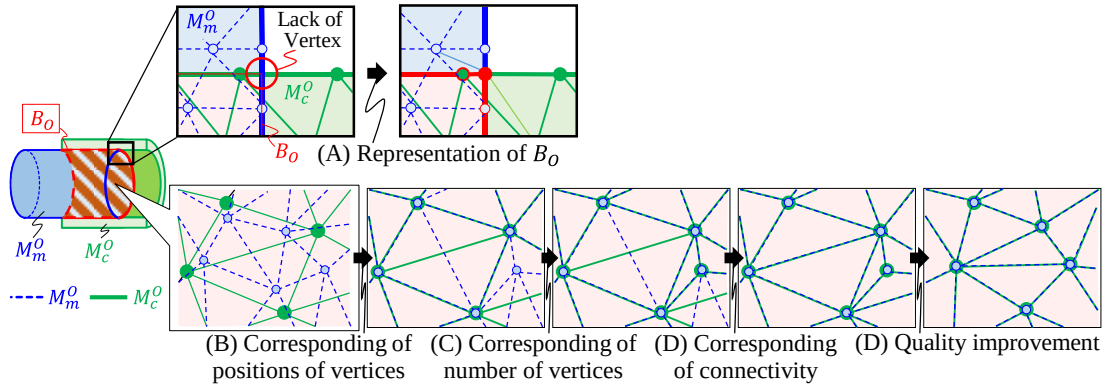


Fig. 3.9 The overview of contact region adaptation

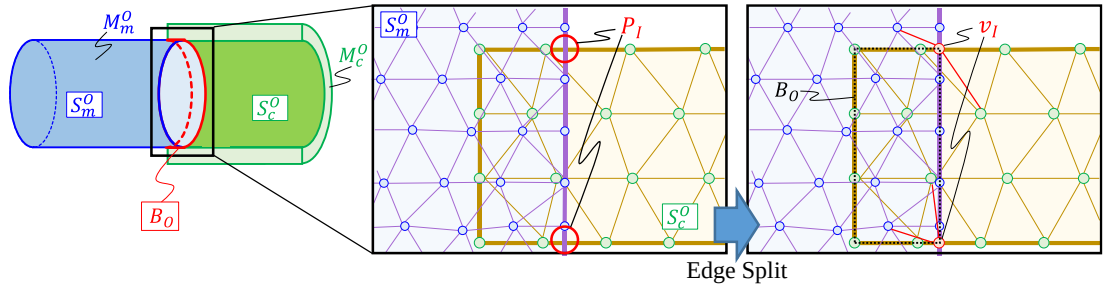


Fig 3.10 Vertex insertion on the intersection points of boundary line segments

3.5.4.2 Representation of boundary lines of the contact region

In order to represent B_O accurately, the intersection points P_I of boundary line segments of the contact regions are needed as vertices of M_m^O and M_c^O . However, there may be no vertices on the location of the intersection points. Therefore, as shown in Fig. 3.10, vertices v_I located on the intersecting points are inserted to M_m^O and M_c^O by edge split. In order to extract intersection points P_I accurately, boundary line segments of S_m^O and S_c^O are first projected to the 2D space which is used in the contact region extraction (subsection 3.4.2) and then P_I are extracted. The insertion of vertices v_I is also performed on the 2D space.

In the following sub-subsections the added vertices v_I are handled as feature vertices. In other words, their position is fixed and preserved after local topological operations such as edge collapse.

3.5.4.3 Corresponding vertex positions

After the modification of boundary lines, as shown in Fig. 3.11, the positions of vertices in V_m^O and V_c^O are corresponded by vertex repositioning. At first, corresponding vertex pairs are made from

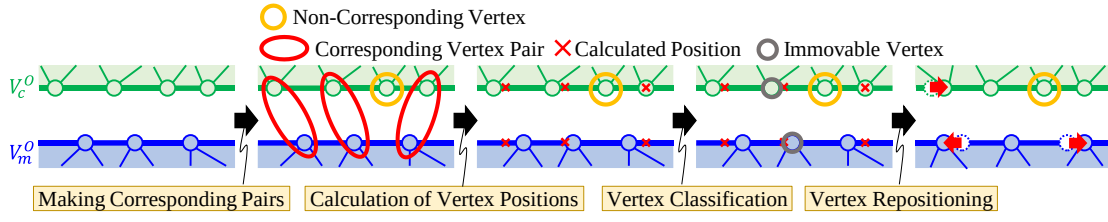


**Fig. 3.11 Corresponding vertex positions
(two meshes contact with each other on the bold line)**

V_m^O and V_c^O . Then, the position of each vertex is calculated for the vertex repositioning. After that, vertices are classified into “movable” or “immovable” vertices. Finally, all vertices in V_m^O and V_c^O except for immovable vertices move to positions calculated in the vertex position calculation process. The vertex position calculation, the vertex classification, and the vertex repositioning are repeated until no vertices move.

Making corresponding pairs:

In the proposed method, it is assumed that the input tetrahedral mesh has high element shape qualities and good distribution of vertices. Therefore, as shown in Fig. 3.12, in the making corresponding vertex pairs, each vertex is made a pair with its closest vertex of the other object mesh so that the distance between positions before and after the vertex repositioning becomes smaller and element shape qualities are preserved as much as possible. In this process, each vertex $v_{cr} \in V_m^O \cup V_c^O$ is first labeled as a “non-corresponding vertex”. Secondary, if a non-corresponding vertex $v_c^n \in V_c^O$ is the closest non-corresponding vertex of a non-corresponding vertex $v_m^n \in V_m^O$, if v_m^n is the closest non-corresponding vertex of v_c^n , and if their distance is smaller than a threshold δ^{pair} , they become “corresponding vertices” of each other. The threshold δ^{pair} is determined for each vertex as the longest length of its incident edges. The closest vertex search and labeling of pairs are repeated until corresponding vertex pairs are not made any more.

Calculation of vertex positions:

After the making corresponding vertex pairs, the new positions of vertices are calculated. In this process, in order to keep the original position of each vertex as much as possible, it is assumed that a new position of each vertex i and its corresponding vertex c_i is on a straight line which passes i and c_i as shown in Eq. (3.3):

$$\mathbf{x}'_i(\alpha_i) = \mathbf{x}'_{c_i}(\alpha_i) = \alpha_i(\mathbf{x}_{c_i} - \mathbf{x}_i) + \mathbf{x}_i, \quad (3.3)$$

where $\mathbf{x}'_i$ and $\mathbf{x}'_{c_i}$ are new positions of i and c_i respectively, $\mathbf{x}_i$ and $\mathbf{x}_{c_i}$ are original positions of i and c_i respectively, and α_i is a parameter determining the amount of movement. α_i of each vertex $i \in V_m^O \cup V_c^O$ is calculated in order to minimize an energy E_{dif} shown in Eq. (3.4):

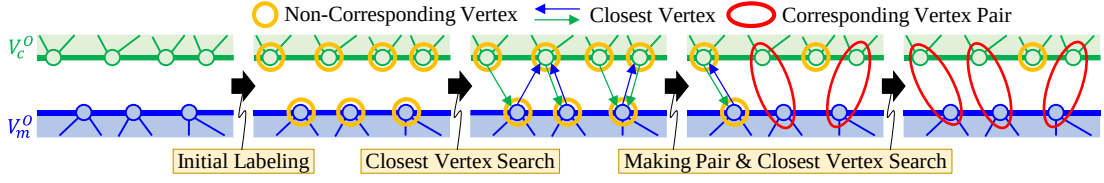


Fig. 3.12 Making corresponding vertex pairs
(two meshes contact with each other on the bold line)

$$E_{dif} = \sum_{i \in V_m^O \cup V_c^O} \sum_{j \in N_i^{svtx}} \left\| (x_j - x_i) - (x'_j(\alpha_j) - x'_i(\alpha_i)) \right\|^2, \quad (3.4)$$

where N_i^{svtx} is a set of neighboring surface vertices of i . E_{dif} represents the change of difference between surface vertices neighboring each other. If j is a fixed vertex such as a non-corresponding vertex or a feature vertex, x_j is used as $x'_j(\alpha_j)$ in Eq. (3.4). In addition, if only one of i and c_i (or j and c_j) is a boundary edge vertex, the position of the boundary edge vertex is used as the new position x'_i (or x'_j) in Eq. (3.4). The new position x'_i of each vertex $i \in V_m^O \cup V_c^O$ is determined using α_i which is obtained by solving a system of linear equations introduced by Eq. (3.4).

Vertex classification:

Then, vertices are classified into “movable” or “immovable” vertices. The movable vertices guarantee that their movement does not generate inverted elements. In the vertex classification process, signed volumes of tetrahedral elements are used for finding “movable” vertices. If signed volumes of neighboring tetrahedral elements of vertex i are always positive under the possible movements of the vertices of the tetrahedron, the vertex i can move without causing any inverted elements. As shown in Fig. 3.13, for each tetrahedron with vertices in V_m^O or V_c^O , its signed volumes for at most 15 patterns of the vertex movement are calculated, and if the signed volume becomes a negative value, vertices repositioned in the pattern and their corresponding vertices are labeled as “immovable.” In the vertex classification process, all vertices in the V_m^O and V_c^O are first labeled as “movable,” and then the labeling of immovable vertices is applied to all tetrahedral elements including vertices in the V_m^O and V_c^O .

Vertex repositioning:

Finally, each vertex $i \in V_m^O \cup V_c^O$ except for immovable vertices is moved to the new position obtained by using α_i .

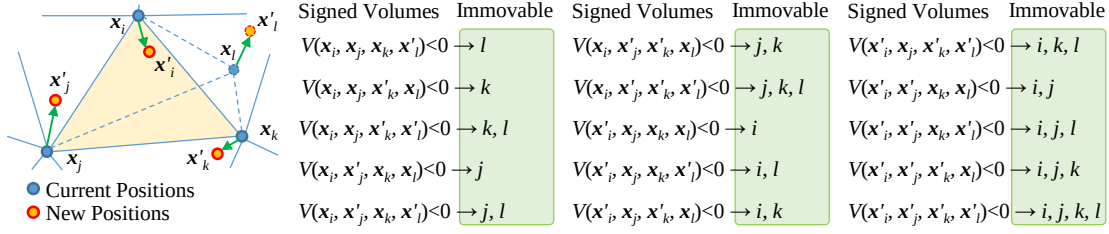


Fig 3.13 The movable check of vertices

3.5.4.4 Corresponding number of vertices

After the corresponding position of vertices, the numbers of vertices in V_m^O and V_c^O are corresponded by vertex removal and vertex insertion. Let V_u^O is a set of non-corresponding vertices and immovable vertices. As shown in Fig 3.14, each vertex $i \in V_u^O$ is first removed by edge collapse based on element shape qualities of neighboring tetrahedra. Then, if i cannot be removed without inverted elements, i is inserted to *the other object mesh* by edge split or face split based on element shape qualities. In this sub-subsection, “the other object mesh” means M_c^O if $i \in V_m^O$ and M_m^O if $i \in V_c^O$.

In this sub-subsection, as mentioned in section 1.3, stretch [Geuzaine 2009] is used as an element shape quality measure. For explanation, Eq. (1.1) is re-written here:

$$Q(\tau) = \frac{6\sqrt{6}V(\tau)}{\left(\max_{e \in E_\tau} L(e)\right) A^{tet}(\tau)}, \quad (1.1)$$

where τ is a tetrahedron, $Q(\tau)$ is stretch, $V(\tau)$ the signed volume of τ , $A^{tet}(\tau)$ the surface area of τ , E_τ a set of edges of τ , and $L(e)$ is the length of edge e . Stretch becomes 1 for regular tetrahedron, 0 for degenerated tetrahedron, and a negative value for inverted tetrahedron.

Vertex removal:

In order to correspond the numbers of vertices in V_m^O and V_c^O , each vertex $i \in V_u^O$ is removed by edge collapse. In this process, a set of tetrahedra K_u with vertices included in V_u^O is first extracted. Then, a vertex i neighboring a tetrahedron with the minimum stretch of K_u is preferentially removed in order to remove tetrahedra with low element shape qualities preferentially. As shown in Fig. 3.15, in the removal of each vertex i , for each incident surface edge $e = \{i, j\}$, the following three virtual operations are tested and the minimum stretch Q_{min}^R of tetrahedra with the new vertex k (and j 's corresponding vertex c_j) is calculated in each case.

- (A) A half edge collapse where the position of the new vertex k is x_i is applied to e . If j has a corresponding vertex c_j , c_j is moved to x_i .
- (B) A half edge collapse where the new vertex position is x_j is applied to e .

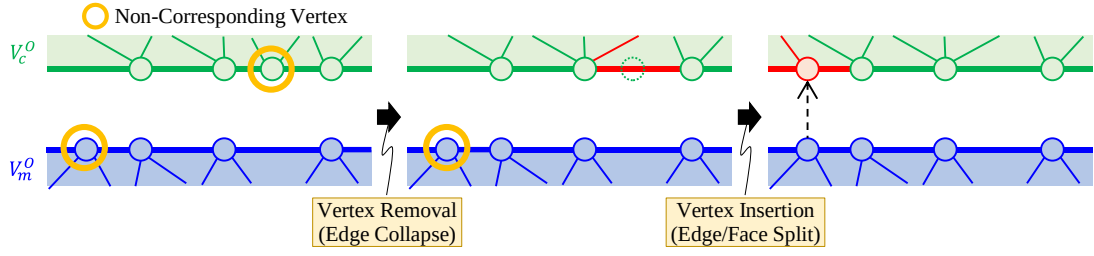


Fig. 3.14 The overview of corresponding number of vertices
(two meshes contact with each other on the bold line)

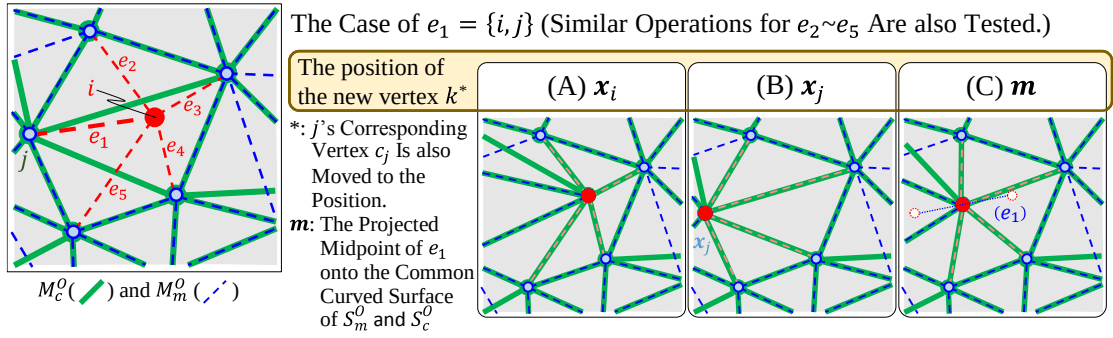


Fig. 3.15 Virtual edge collapse for vertex removal

- (C) An edge collapse where the position of the new vertex k is the midpoint of e is applied to e , and the new vertex k is moved onto the common curved surface of S_m^o and S_c^o (in Fig. 3.15, the position described by m). If j has a corresponding vertex c_j , c_j is moved to x_k .

After that, the edge collapse with the largest Q_{min}^R is adopted. If the maximum Q_{min}^R is a negative value, the vertex removal is not performed.

In addition, if only one of i and j is a boundary edge vertex, only the edge collapse where the position of the new vertex k is the position of the boundary edge vertex is tested for e .

Moreover, in order to keep the shape approximation accuracies of object meshes, if the geometric error (see Fig. 3.16) becomes larger than the acceptable geometric error δ^{geo} by the edge collapse, the edge collapse is not performed and other edge collapse with second largest Q_{min}^R is adopted. δ^{geo} is preliminarily calculated as the largest geometric error of each surface region of input object meshes.

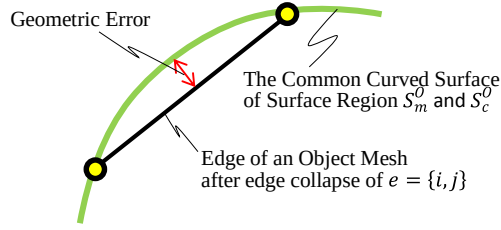


Fig. 3.16 Geometric error

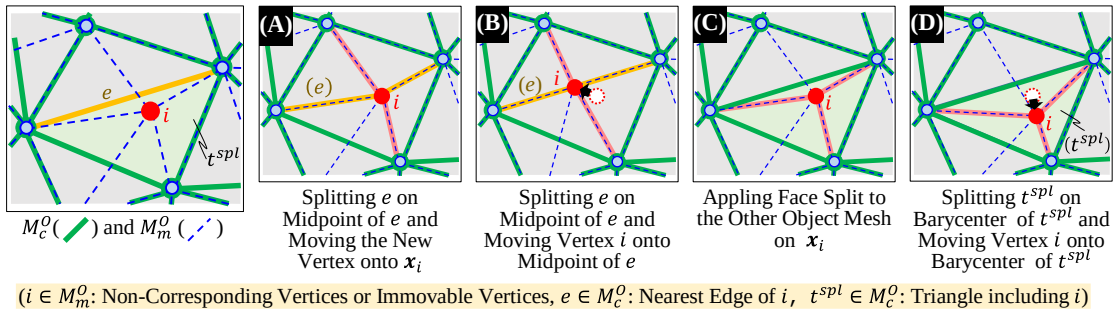


Fig. 3.17 Four virtual split operations for vertex insertion

Vertex insertion:

If a vertex $i \in V_u^O$ cannot be removed without inverted elements, the vertex i is inserted to the other object mesh by edge split or face split. In this process, as shown in Fig. 3.17, the following four operations are first performed virtually. Then, the operation after which the minimum stretch of tetrahedra neighboring the new inserted vertex and vertex i is highest is adopted.

- (A) The nearest edge e of x_i on the other object mesh is split and a new vertex is inserted to the midpoint of e . Then, the new vertex is moved to x_i .
- (B) The nearest edge e of x_i on the other object mesh is split and a new vertex is inserted to the midpoint of e . Then, the vertex i is moved to the midpoint of e .
- (C) Face split is applied to the other object mesh and a new vertex is inserted to x_i .
- (D) The vertex i is projected to the other object mesh. Then, Face split is applied to a triangle t^{split} which includes the projected i on the other object mesh and a new vertex is inserted to the barycenter of t^{split} . Finally, the vertex i is moved to the barycenter of t^{split} .

Note that stretches are calculated after moving the new inserted vertex and i onto the common curved surface of S_m^O and S_c^O if S_m^O and S_c^O are curved surface regions.

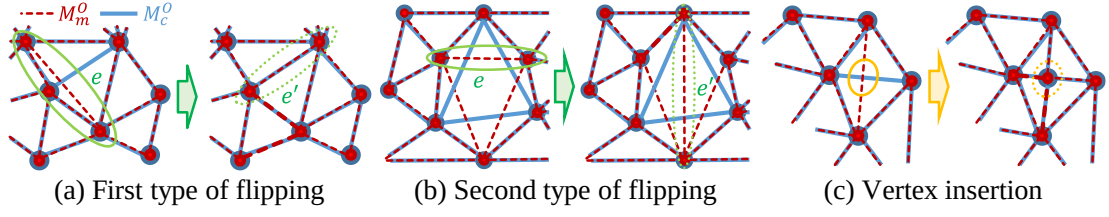


Fig 3.18 The corresponding connectivity of surface triangular meshes

3.5.4.5 Corresponding connectivity

After the corresponding positions and number of vertices, the connectivity of surface triangle meshes T_m^O and T_c^O on the contact regions of M_m^O and M_c^O are made identical to each other by edge flipping and edge split. In order to keep the topological consistency of tetrahedra, the edge flipping consisting of edge split and edge collapse is used (see sub-subsection 2.6.5.3 and Fig. 2.45). In this thesis, an edge whose two endpoint positions are same as those of an edge e is called a “corresponding edge” of e , and if the corresponding edge of e does not exist, e is called a “non-corresponding edge.” In addition, an edge obtained by flipping of e is denoted by e' , and edges which can be flipped are called “flip-able” edges.

In this process, at first, as shown in Fig. 3.18(a), for non-corresponding flip-able edge e , if e' is a corresponding edge of a non-corresponding edge, e is flipped. If there are no such edges but the connectivity of T_m^O and T_c^O are not the same, as shown in Fig. 3.18(b), other non-corresponding flip-able edges are flipped. These two flipping processes are repeated until any non-corresponding edge cannot be flipped or the iteration times reached a threshold. Finally, as shown in Fig. 3.18(c), if any non-corresponding edge cannot be flipped on T_m^O and T_c^O , vertices located on the intersection points between two non-corresponding edges are inserted to M_m^O and M_c^O by edge split.

3.5.4.6 Quality improvement of the contact region

After that, in order to improve element shape qualities, phased ODT smoothing mentioned in section 2.6 is applied to the set of tetrahedral elements for which one or more vertices are included in V_m^O or V_c^O . In this Phased ODT smoothing, as shown in Fig. 3.19, the mesh density is controlled by one of the following two patterns. In the first pattern (Fig 3.19(a)), the distance field grid G_d or vertex search grid G_v are used as the regular grid of target mesh density field. On the other hand, in the second pattern (Fig 3.19(b)), the regular grid of target density field is generated for each object mesh, and moved together with each object mesh. In the second pattern, if the midpoint of an edge is included two cells (cell of the grid for M_m^O , and cell of the grid for M_c^O), the length of the edge is evaluated by using the ranges of target edge length obtained by one of the grids. In this thesis, if a space mesh surrounding object meshes exists, mesh density controlled by the first pattern. On the

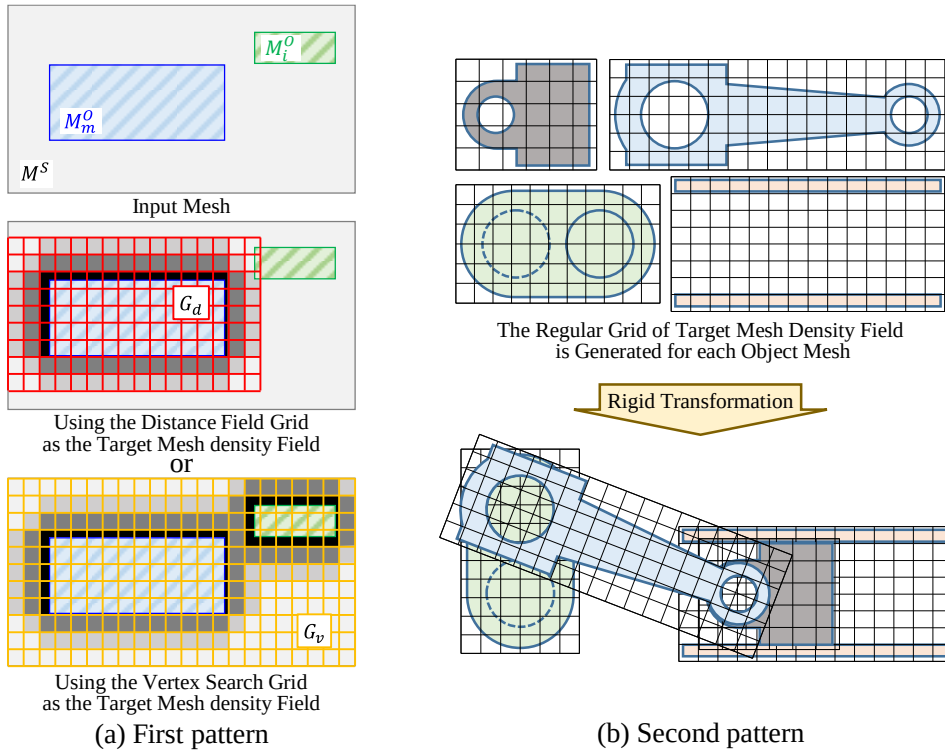


Fig 3.19 The two patterns of regular grids of the target mesh density field

other hand, if any space mesh do not exist, in other words, if a conformal tetrahedral mesh of an assembly model is generated from object meshes which are generated individually without any space mesh, mesh density controlled by second pattern because the distance field grid G_d and vertex search grid G_v are not needed in the situation.

3.5.5 Tetrahedrization

After mesh adaptation of the contact region, if M_m^O contacts with other object meshes, the hole H^{rem} generated by the space mesh element removal mentioned subsection 3.5.2 is tetrahedrized by the constrained Delaunay Triangulation. A set of tetrahedral elements generated by this triangulation is denoted by K_H in this thesis.

3.6 Mesh quality improvement based on ODT

After mesh adaptation, many tetrahedral elements with bad shapes are included in the resulting conformal mesh. Therefore, element shape qualities are improved by a quality improvement based on Optimal Delaunay Triangulation (ODT) [Chen 2004]. In this section, as mentioned in section 1.3, stretch [Geuzaine 2009] is used as an element shape quality measure. Stretch becomes 1 for regular tetrahedron, 0 for degenerated tetrahedron, and a negative value for inverted tetrahedron.

In addition, in order to obtain output meshes having desirable mesh density, the edge length of M^S is controlled by using G_d . In the proposed method, the range of target edge lengths is defined by minimum and maximum edge length δ_{min}^{length} and δ_{max}^{length} at each cell in the G_d . If keeping the edge length of the original input mesh is required, δ_{min}^{length} and δ_{max}^{length} are determined according to the original edge length. For example, for each cell, $\delta_{min}^{length} = \beta_1^{EL} \delta_{ave}^{length}$ and $\delta_{max}^{length} = \beta_2^{EL} \delta_{ave}^{length}$ where δ_{ave}^{length} is the average length of original input mesh edges whose midpoints are included in the cell can be used. If meshes which become coarser at the portion far from the object are required, δ_{min}^{length} and δ_{max}^{length} are assigned according to the distance from the surface of object meshes.

In our method, at first, a set of tetrahedral elements K_Q of M^S which satisfy any of the following conditions is extracted from K_{Df} and K_H .

- Condition 1 The stretch is smaller than a threshold δ_1^{st} .
- Condition 2 The minimum or maximum edge length is out of the range of target edge lengths of cells including their midpoints.

Then, a quality improvement consisting of the following four steps is applied to K_Q .

- Step 1 Tetrahedral elements whose stretches are smaller than a threshold $\delta_2^{st} < \delta_1^{st}$ (i.e. inverted and degenerated elements) are removed by half edge collapse, double split collapse operation and split collapse operation [Li 2003].
- Step 2 Each edge shorter than the threshold δ_{min}^{length} of a cell including the midpoint of the edge is collapsed by edge collapse.
- Step 3 Each edge longer than the threshold δ_{max}^{length} of a cell including the midpoint of the edge is divided into two edges by edge split.
- Step 4 ODT smoothing [Chen 2011] is applied to K_Q .

3.7 Experimental results and evaluation

3.7.1 Overview of experiments

In this subsection, first of all, an application of the proposed mesh adaptation method to the motion of a single object mesh in a space mesh are shown. Then, in order to show that the proposed method can deal with object motion with contact, an experiment was carried out using object meshes of a moving cylinder and a half tube whose radii are the same. In this experiment, the cylinder was translated toward a half tube and contacted with the half tube. Finally, it is shown that the contact region adaptation of our method enables us to generate conformal tetrahedral meshes of moving assembly models from a set of meshes generated individually without space mesh. Table 3.3 shows the data of each experiment.

Table 3.3 The data of each experiment

Figure	#Motion steps	#Vertices		#Tetrahedra		Minimum stretch		Average stretch		Generation time of all meshes [s]
		Input	Final output	Input	Final output	Input	Min. of outputs	Input	Min. of outputs	
Fig. 3.20	360	37,629	36,970	210,465	205,525	0.28	0.13	0.75	0.75	49
Fig. 3.22	45	31,530	34,810	160,791	177,309	0.15	0.056	0.71	0.70	202
Fig. 3.23	36	6,694	6,150	25,320	24,777	0.21	0.504	0.67	0.67	165

All the experiments were processed on a personal computer having following spec.

- OS: Windows 7 Professional 64bit.
- CPU: Intel Core i7-5960X 3.00GHz.
- RAM: 64.0 GB.
- Development environment: Microsoft Visual Studio 2010.
- Programing language: C++.
- Graphic API: Open GL.
- Others: Octree meshing tool of [CATIA V5R18] for the input mesh generation, Tetgen 1.4.3 (tetrahedral mesh generator) [Si 2006] for tetrahedrization mentioned in subsection 3.5.5, and Eigen 3.1.2 (template library for linear algebra) [2012] for the space embedding method mentioned in subsection 3.5.3 and the corresponding vertex positions mentioned in sub-subsection 3.5.4.3.

3.7.2 Mesh adaptation for single moving object

The input mesh model (#Vertices: 37,629 and #Tetrahedra: 210,465) is shown in Fig. 3.20(a). The input tetrahedral mesh was generated by the following three steps: first, $20 \times 20 \times 50$ cubes were aligned, and a material id was assigned to $3 \times 5 \times 1$ cubes which became the red rectangle shown in Fig. 3.20(a). Secondly, a vertex was inserted to the center of each cube, and the set of cubes was tetrahedrized by dividing each quadrangle into two triangles and connecting the inserted centers with vertices of cubes. Finally, a random perturbation of vertices and ODT smoothing were applied to the tetrahedral mesh.

In this experiment, the red rectangle moved 0.1 toward the right side along z axis and rotated by 1 degree around an axis which passes a barycenter of the rectangle and is parallel to x axis in each motion step. The total number of motion steps is 360. The minimum average stretch of all output meshes was almost the same as the average stretch of the input mesh. The minimum stretch of tetrahedral mesh in each motion step and the cross-section of output meshes are shown in Fig. 3.20(b) and (c), respectively. The minimum stretch of all output meshes is 0.13, and it is larger than the recommended lower limit (0.05) of stretch. The generation of all meshes took 49s. Therefore, in each motion step, the conformal tetrahedral mesh without any inverted elements could be generated within only 0.14s. As shown in Fig. 3.20(c), space mesh could be adapted depending on the motion

of the rectangle, and the adaptation was done only around the rectangle.

The distance threshold is 3 (Manhattan distance based on the distance field grid), and 0.82 which is about 6 times of the maximum displacement of vertices of the moving object mesh M_m^O was used as the grid size in above experiment. In the proposed mesh adaptation method, it is expected that larger deformed region provides higher element shape qualities of the resultant meshes and longer processing time. In order to verify the relation of the size of the deformed region to element shape qualities of resultant meshes or processing time, the results obtained by different distance thresholds for the deformed region extraction are shown in Table 3.4. In addition, the relation of the distance threshold to (a) the minimum value of the minimum stretches of the deformed region of each step, (b) the minimum value of the average stretches of the deformed region of each step, (c) the average value of the average stretches of the deformed region of each step, and (d) the generation time of all meshes are shown in Fig. 3.21. These results shows that larger deformed region could not always introduce higher minimum stretch. On the other hand, the average stretch could be improved when larger distance threshold was used. In addition, processing time became longer when larger distance threshold was used. However, when the distance threshold became larger, although the increase of processing time by the increase of the distance threshold became larger, that of the average stretch became smaller.

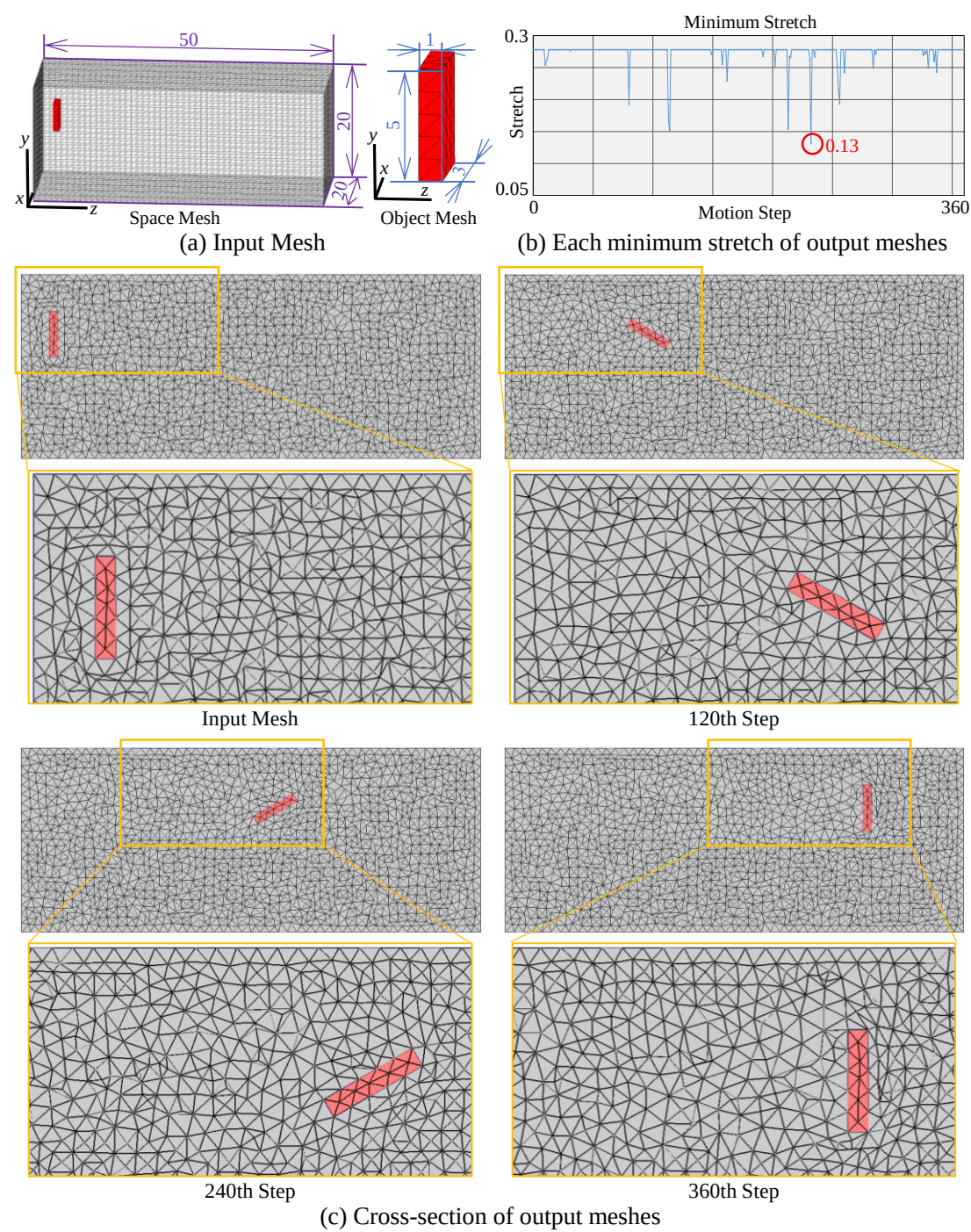
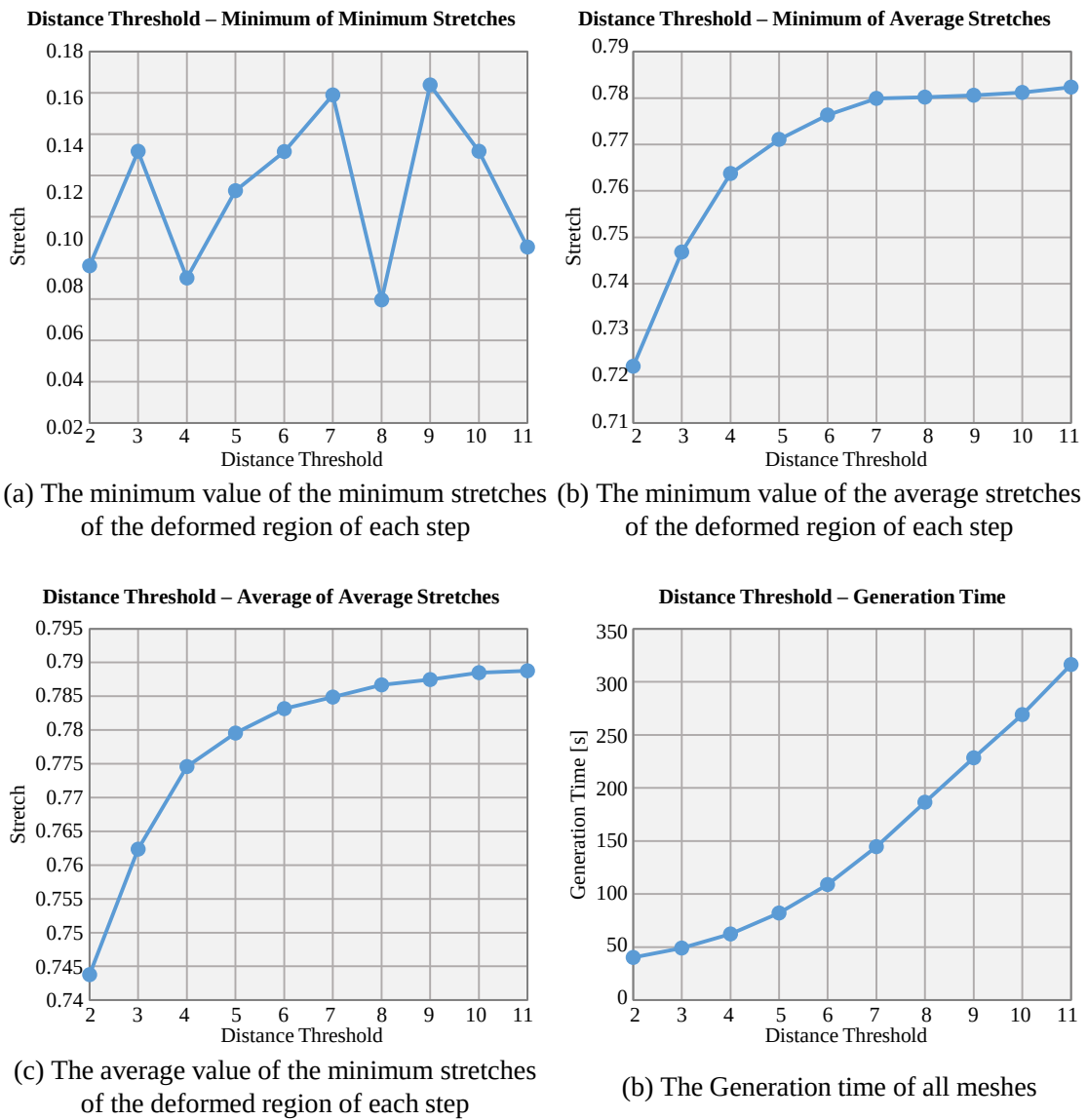


Fig. 3.20 Translation of a rotating rectangle

Table 3.4 The results obtained by different distance thresholds

Distance threshold	The minimum of minimum stretches	The minimum of average stretches	The average of average stretches	Generation time of all meshes [s]
2	0.0761	0.7222	0.7438	40.09
3	0.1317	0.7468	0.7624	49.10
4	0.0702	0.7637	0.7747	62.26
5	0.1126	0.7711	0.7796	81.99
6	0.1315	0.7764	0.7832	108.84
7	0.1590	0.7799	0.7849	144.47
8	0.0596	0.7802	0.7867	186.48
9	0.1638	0.7805	0.7875	228.33
10	0.1316	0.7812	0.7885	268.96
11	0.0852	0.7823	0.7888	316.11

**Fig. 3.21 The relation of the distance threshold to element shape qualities and time**

3.7.3 Mesh adaptation for object motion with contact

The other example (#Vertices: 31,530 and #Tetrahedra: 160,791 in the input mesh) is shown in Fig. 3.22(a), where an object contacts with another. The input mesh was generated by the following three steps: first, the space mesh was generated by the Octree meshing tool of [CATIA V5R18]. Then, meshes of the red cylinder and the green half tube were generated by Tetgen [Si 2006] in the space mesh. Finally, some random vertex insertions and phased ODT smoothing were applied to the tetrahedral mesh. The space mesh generation took 7s, the generation of object meshes took 3s, and the quality improvement took 367s.

In this experiments, the red cylinder moved 1 toward the green half tube along its axis in each motion step. The total number of motion steps is 45. The minimum average stretch of all output meshes was almost the same as the average stretch of the input mesh. The minimum stretch of tetrahedral mesh in each motion step and the every 5 step output meshes after 30 steps are shown in Fig. 3.22(b) and (c), respectively. The minimum stretch of all output meshes is 0.056, and it is larger than the recommended lower limit of stretch. As a result, the total calculation time was 202.6s. In each motion step, the conformal tetrahedral mesh without any inverted elements could be generated in about 4.502s. As shown in Fig. 3.22(c), all edges on the boundary between two object meshes were shared by two object meshes, and all triangles between two object meshes were also shared by two tetrahedral elements of two object meshes. The final output mesh has 34,810 vertices, which means that vertices increased by about 10 percent. These results show that the tetrahedral mesh was adapted to the object motion with contact by the proposed method.

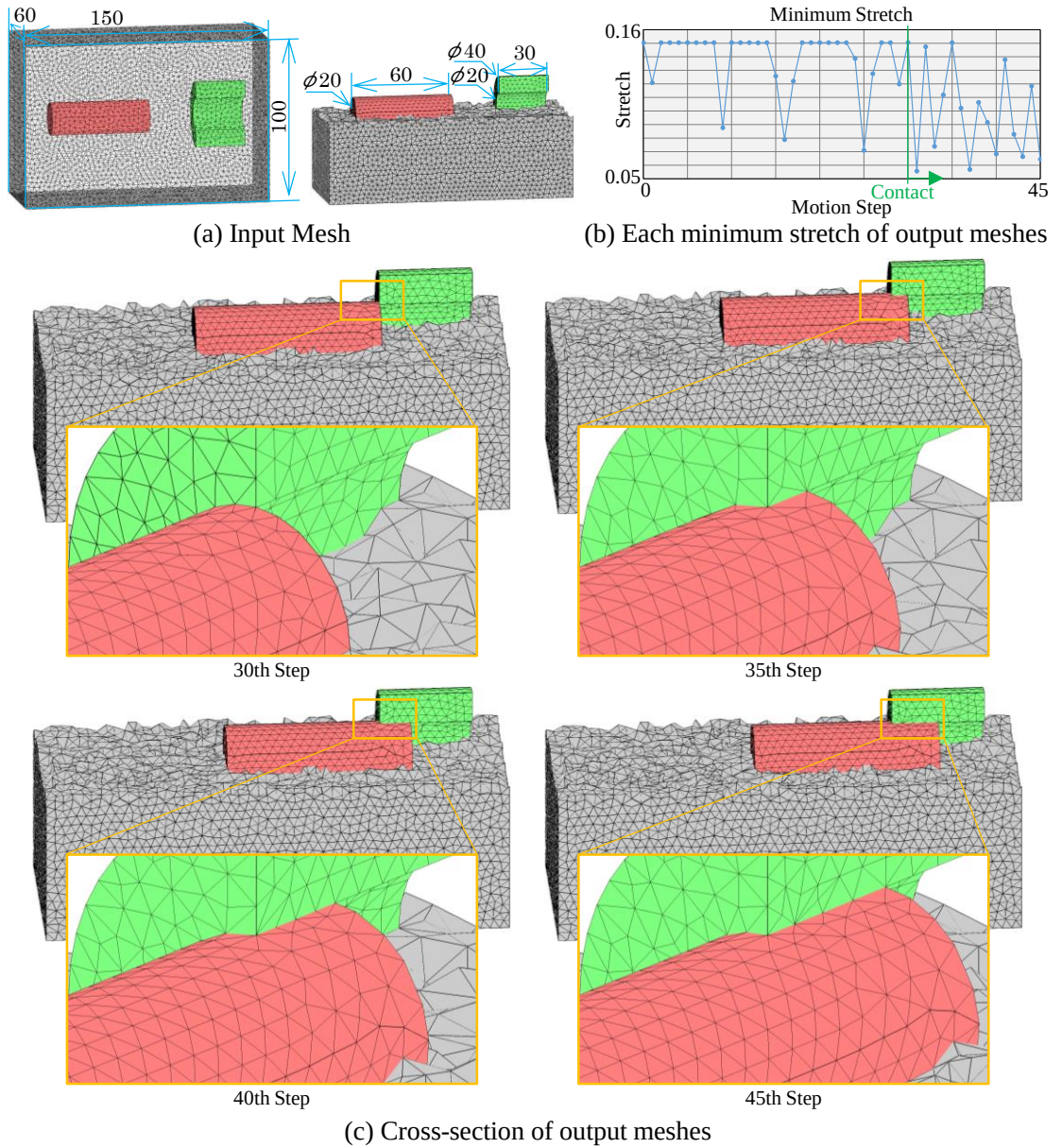


Fig. 3.22 Translation of a cylinder toward a half tube

3.7.4 Conformal mesh generation from a set of mesh models

Finally, it is shown that conformal tetrahedral meshes of moving assembly models can be generated by the proposed method from object meshes generated individually without space mesh. The input simple assembly mesh (#Vertices: 6,694 and #Tetrahedra: 25,320) whose interfaces are non-conforming is shown in Fig. 3.23. They were generated by the Octree meshing tool of [CATIA V5R18] and improved by phased ODT smoothing. The total mesh generation time was about 5s.

The green part (*crank shaft*) is rotated 10 degrees in each motion step around an axis (shown by a black arrow in Fig. 3.23(a)) and the blue part (*piston head*) can only translate along the orange arrow. The total number of motion steps is 36. The average stretch of the input mesh was 0.64 and the minimum average stretch of all output meshes was 0.67. The minimum stretch of each motion step is shown in Fig. 3.23(b). The minimum stretch of all output meshes is 0.0504, and it is larger than the recommended lower limit of stretch. The output meshes of the first step, 12th step, 24th step, and 36th step are shown in Fig. 3.23(c). As shown in Fig. 3.23(c), triangles between two object meshes were shared by at most two tetrahedral elements of two object meshes. The generation of all meshes took 165.4s and the average time of all motion steps is 4.59s. The final output mesh has 6,150 vertices and the maximum number of vertices of all output meshes was 6,338. The result shows the proposed method can be used for generation of conformal meshes from a set of meshes which are generated individually while avoiding the drastic increase of elements. In addition, it was also shown that any inverted and degenerated elements are not included in output meshes of the proposed method.

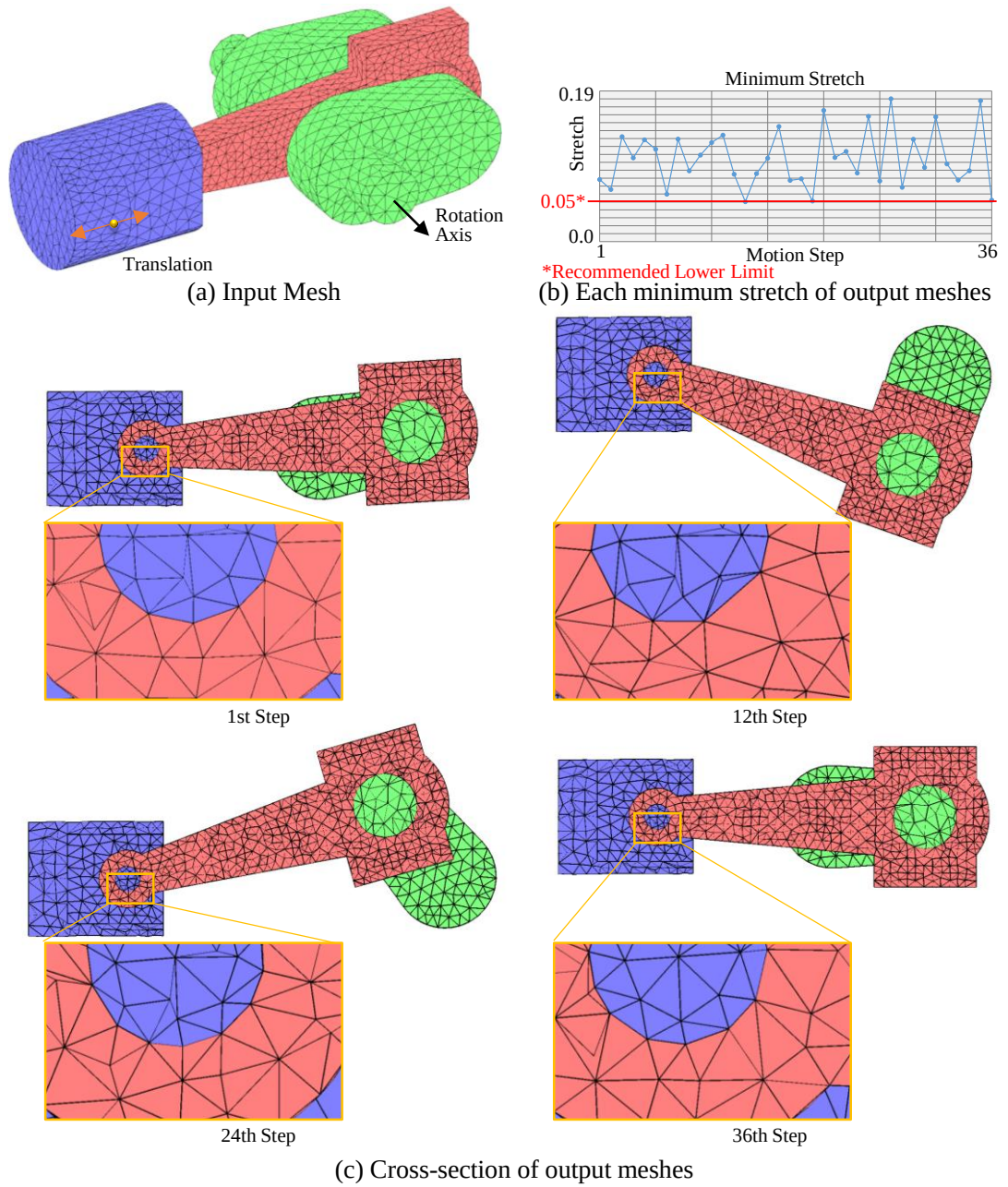


Fig. 3.23 The simple mechanical parts without space mesh

3.8 Summary

In this chapter, a mesh adaptation method was proposed. The proposed method consists of the following three processes.

- *Local region extraction using surface information and distance field:* for efficient mesh adaptation and handling contacts between objects, the contact region and deformed

region are extracted in each motion step. In order to extract two regions, as pre-processing the distance field grid and the vertex search grid are first generated, and surface information is extracted by the segmentation method proposed in the section 2.4. By using these two regular grid, a set of tetrahedron of the space mesh near the moving object mesh is extracted as the deformed region. In addition, a set of vertices of object meshes, which are on the overlapping area of surface of two object meshes is accurately extracted using the surface information and the regular grids.

- *Mesh adaptation using two local regions:* in order to avoid degenerated elements, if the moving object mesh contacts other object meshes, tetrahedra of the space mesh near the contact region are first removed. Then, a rigid transformation is applied to the moving object mesh and the distance field grid, and the deformed region is adapted to the object motion by the space embedding method. In addition, for keeping mesh conformity on the contact regions between object meshes, the topology and geometry of surface triangular meshes of contacting object meshes were adapted by vertex repositioning and local topological operations. Finally, if the moving object mesh contacts other object meshes, the constraint Delaunay triangulation is applied to the hole created by the removal of tetrahedra of the space mesh near the moving object mesh.
- *Mesh quality improvement based on ODT:* in order to obtain a high quality tetrahedral mesh of each motion step, a quality improvement method based on ODT were performed. In the quality improvement, elements which need to be improved or removed are first extracted. Then, degenerated and inverted elements are removed by edge split and edge collapse. After that, each edge length is controlled by regular grids and local topological operations. Finally element shape qualities are improved by ODT smoothing.

In addition, the effectiveness of the proposed mesh adaptation method was demonstrated through three simple experiments.

- The proposed method was first applied to a tetrahedral mesh with the motion of a single object mesh in a space mesh, and it was shown that space mesh could be modified depending on the motion of the object mesh, and the adaptation was done only around the moving object mesh.
- Then, the proposed method was applied to a mesh which consisted of a moving cylinder, a fixed half tube, and a space mesh around them in order to show that the proposed method can deal with contact between object meshes. As a result, in each motion step, a conformal tetrahedral mesh without any inverted and degenerated elements could be generated.
- In addition, it was shown that the proposed method can be used for generation of conformal tetrahedral meshes of moving assembly models from a set of meshes which are generated individually. As a result, in each motion step, a conformal tetrahedral mesh could be generated while avoiding the drastic increase of elements.

In the experiments, even if the object meshes contact with each other, the conformal tetrahedral meshes of each motion step were generated without drastic increase of elements on the contact region and generation of inverted and degenerated elements. In addition, the processing time of each motion step is at most 5s for the conformal tetrahedral mesh including about 160k tetrahedra.

Chapter 4 Application of Deformation and Adaptation methods to Hexahedral Meshes based on Tet-Hex Conversion

4.1 Requirements and organization of this chapter

As mentioned in section 1.3, hexahedral meshes are preferred over tetrahedral meshes because hexahedral meshes can provide more accurate results of FEA by more small number of elements in comparison with tetrahedral meshes. Therefore, many hexahedral mesh generation methods have been proposed [Stephenson 1992, Blacker 1993, Tautges 1996, Owen 2000, Meshkat 2000, Yamakawa 2003, Bottella 2016,]. However, the generation of all-hex meshes is a challenging problem, and domains with complex shapes cannot be decomposed into all-hex meshes automatically.

In order to obtain hexahedral meshes of complex solid models, there are two approaches. One is deformation and adaptation methods reusing hexahedral meshes which are generated previously [Staten 2008, Staten 2011, Sieger 2013, Sieger 2014, Chen 2016]. The other is hex-dominant mesh generation methods [Owen 2000, Meshkat 2000, Yamakawa 2003, Bottella 2016]. Although all-hex meshes can be obtained by deformation and adaptation methods, direct editing of all-hex meshes while keeping mesh qualities such as element shape quality and mesh density is difficult and there exist many limitations for represented shapes yet. On the other hand, although hex-dominant mesh generation methods cannot generate all-hex meshes in many cases, complex solid models can be decomposed by polyhedra. However, as mentioned in section 1.1, repeating meshing of modified solid models where parameters of form features are changed or poses of objects in a fluid are changed is inefficiency.

As mentioned in section 1.4, the objective of this thesis is an efficient conformal mesh generation for modified solid models where parameters of form features are changed or poses of objects in a fluid are changed. As one step to archive this objective, in this chapter, as shown in Fig. 4.1, the proposed tetrahedral mesh deformation and adaptation methods in this thesis are combined with a conversion between tetrahedra and hexahedra (called “Tet-Hex conversion” in this thesis) in a hex-dominant mesh generation method, and the effectiveness of the combination is verified.

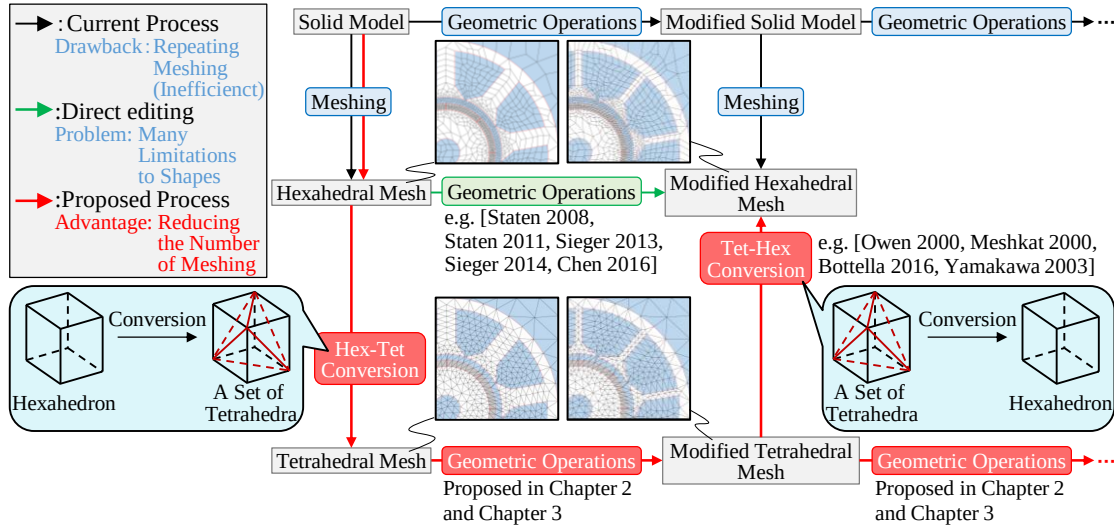


Fig. 4.1 The combination the proposed methods with a Tet-Hex conversion

There are the following requirements for the Tet-Hex conversion in the problem setting of this thesis.

- Arbitrary tetrahedral meshes can be converted into hexahedral meshes.
- Hexahedral meshes with moderate element shape qualities should be obtained.
- The number of elements should be decreased by the conversion for efficient FEA.
- The number of tetrahedra which become a hexahedron should be constant in order to reflect results of the mesh density control of the proposed methods in resultant hexahedral meshes.

In this chapter, at first, existing hex-dominant mesh generation methods based on conversions between tetrahedra and hexahedra (called indirect hexahedral mesh generation methods) are introduced in section 4.2. Then, in section 4.3, the overview of the validation of the combination of the proposed methods mentioned in Chapter 2 and Chapter 3 with an indirect hexahedral mesh generation method [Meshkat 2000] is shown. In section 4.4, the conversion of hexahedral meshes into tetrahedral meshes is described. In section 4.5, key techniques of the indirect hexahedral mesh generation method are described. After that, in section 4.6, the effectiveness of the combination of the proposed methods with [Meshkat 2000] is verified through application of the method to some tetrahedral meshes. Finally, this chapter is summarized in section 4.7.

4.2 Related work of indirect hexahedral mesh generation

Because it is difficult to generate all-hex meshes of arbitrary domains, conversions between tetrahedra and hexahedra are used in some hexahedral mesh generation methods (called indirect hexahedral mesh generation methods). Figure 4.2 shows a well-known conversion between a

tetrahedron and four hexahedra. By using the conversion, all-hex meshes can be obtained from arbitrary tetrahedral meshes stably, and it is known that the resultant hexahedra have moderate element shape qualities if the element shape quality of the tetrahedron is high. However, the number of elements is drastically increased through the Tet-Hex conversion, and the resultant hexahedral meshes includes hexahedra with low shape qualities. On the other hand, using the conversion between some tetrahedra and a hexahedron as shown in Fig. 4.3, the number of the resultant hexahedra is decreased. Although some tetrahedra may be remained in the resultant hexahedral mesh (i.e. all-hex meshes may not be generated by this Tet-Hex conversion), the element shape quality of the resultant hexahedron can be adjusted according to the way of combination of tetrahedra. Therefore, the second type of conversions is used in many indirect hexahedral mesh generation methods. In this subsection, some indirect hexahedral mesh generation methods based on the second type of conversions are introduced.

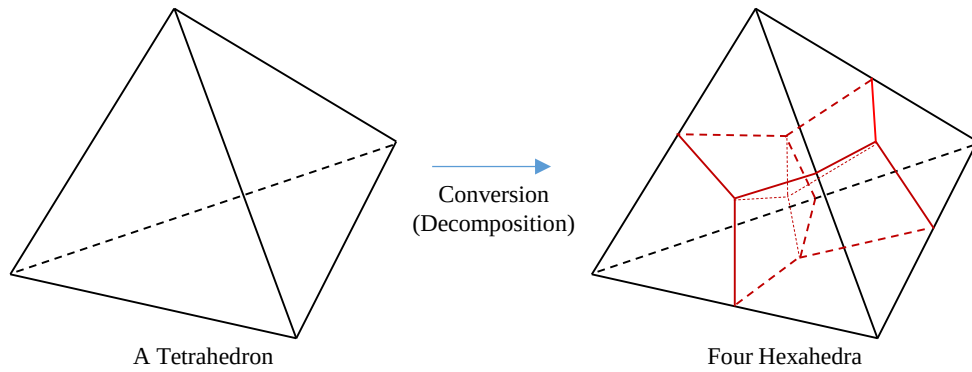


Fig. 4.2 A famous conversion between a tetrahedron and four hexahedra

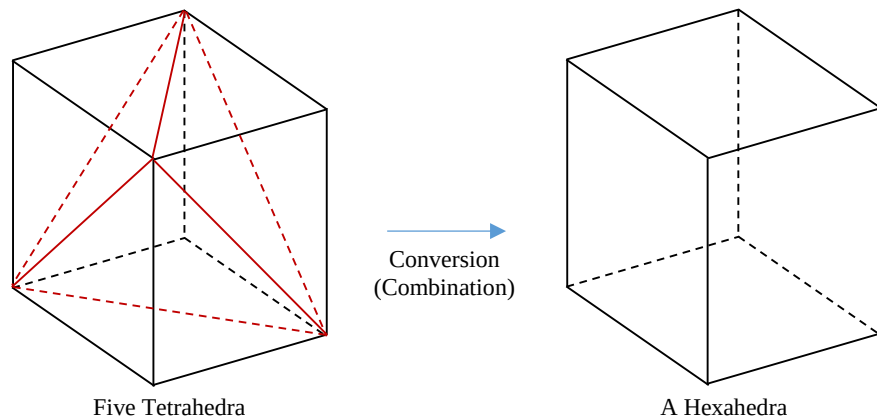


Fig. 4.3 Another type of conversions between tetrahedra and a hexahedra

Owen et al. [2000] proposed an indirect hexahedral mesh generation method based on conversions between tetrahedra and hexahedron with an advancing front technique. In their method, a surface quadrangle mesh of the input domain is first generated. Then each quadrangle of the quadrangle mesh is divided into two triangles. After that, using the triangle mesh, a tetrahedral mesh is generated by constrained Delaunay meshing. Finally, the tetrahedral mesh is converted into a hexahedral mesh by an advancing front manner while modifying inner tetrahedral mesh by local topological operations and vertex repositioning for the conversion. Their method always provides a valid hex-dominant mesh which does not have any non-conforming face during the conversion. In addition, the output hex-dominant mesh has well-aligned layers of elements parallel to the boundary. However, although quadrangles are needed as the initial front, such ideal initial front cannot be obtained in the problem setting of this thesis. Moreover, element shape qualities of remaining tetrahedra tend to become lower.

Meshkat and Talmor [2000] also proposed an indirect hexahedral mesh generation method based on conversions between five or six tetrahedra and a hexahedron using an extended Region-Face-graph (RF-graph). In the extended RF-graph, a tetrahedron is represented by a node and a triangle or a pair of triangles are represented by a graph edge. If a triangle is shared by two tetrahedra, corresponding graph edge is represented by a solid line. On the other hand, if two triangles of different two tetrahedra form a quadrangle, corresponding graph edge is represented by a dashed line. In their method, candidates of hexahedra formed by five or six tetrahedra are first searched from the input tetrahedral mesh using the extended RF-graph. Then the conversion between the input tetrahedral mesh and a hex-dominant mesh is performed by a greedy algorithm using the extracted candidates. Their method can be applied to arbitrary tetrahedral meshes. However, the result of their method depends on the input tetrahedral mesh, and the ratio of hexahedral elements may be low. In addition, the existence of the sliver between quadrangles is not considered, and the output hexahedral mesh includes some non-conformal contact such as a contact between a hexahedron and two tetrahedra. Bottella et al. [2016] extended the conversion in [Meshkat 2000] in order to deal with slivers by some additional subgraphs. In addition, they solved a problem of non-conformity between polyhedral elements in [Meshkat 2000] by a vertex insertion.

Yamakawa and Shimada [2003] proposed an indirect hexahedral mesh generation method based on conversions between five or six tetrahedra and a hexahedron based on searches for eight vertices composing a hexahedron. The inputs of their method are a domain boundary, a desired directionality, and a desired edge length. In their method, in order to obtain well-aligned vertices, body centered cubic (BCC) structured cells are first packed on the boundary and inside of the input domain, and then vertices are created at the center of the cells. After that, using the vertices, a tetrahedral mesh is generated by an advancing front method [Yamakawa 2000] with local transformations [Joe 1995]. Finally, the tetrahedral mesh is converted into a hex-dominant mesh. In the conversion, a list of combinations of eight vertices which form a hexahedron is first created. Then, the entities of list are sorted by the element shape qualities of hexahedra predicted by the entities. Given a combination of eight vertices from the list, tetrahedra whose four vertices are

included in the given combination are removed and a hexahedron is created while avoiding overlap of elements and non-conformities. The conversion between some tetrahedra and a hexahedron is repeated until the list becomes empty. Finally, a similar conversion between tetrahedra and a prism is repeated, and the hex-dominant mesh is generated. Their method can generate hex-dominant meshes with high element shape qualities.

Table 4.1 shows the summary of the indirect hexahedral mesh generation methods. In this chapter, the Tet-Hex conversion in [Meshkat 2000] is adopted because of the following reasons.

- The Tet-Hex conversion can be applied to arbitrary tetrahedral meshes.
- Hexahedral meshes with moderate element shape qualities can be obtained.
- The number of elements are decreased by the conversion.
- The number of tetrahedra which become a hexahedron is constant (five or six).

Table 4.1 Summary of indirect hexahedral mesh generation methods

Method	Approach	Limitation of input tetrahedral meshes	#Tetrahedra converted into a hexahedron
Owen 2000	Advancing front conversion with vertex repositioning and local topological operations	Surface triangular mesh can be converted into quadrangular mesh easily.	Non-constant
Meshkat 2000	Search for RF-graph represented hexahedra	None	Constant (5 or 6)
Bottela 2016	Search for RF-graph represented hexahedra	None	Constant (5, 6, or 7)
Yamakawa 2003	Searching vertices composing hexahedra	None	Non-constant

4.3 Overview of the validation of the proposed method

In this chapter, a validation of the combination of two proposed methods with an indirect hexahedral mesh generation method based on the conversion between tetrahedra and hexahedra [Meshkat 2000] is performed. The overview is shown in Fig. 4.4. At first, a tetrahedral mesh is generated from an original hexahedral mesh by a decomposition of hexahedra. Then, proposed tetrahedral mesh deformation and adaptation methods are performed. Finally, the Tet-Hex conversion in [Meshkat 2000] is applied to the resultant tetrahedral meshes of the second step, and element shape qualities of resultant hexahedra and the ratio between the sum of volume of hexahedra and that of all elements are evaluated. In the variation in this thesis, all-hex meshes generated by regular grids are used as input meshes.

The advantage of hexahedral meshes over tetrahedral meshes is fewer elements required for FEA with the same accuracy. Therefore, avoiding the drastic increase of number of elements through the conversion of hexahedral meshes to tetrahedral meshes is required. In addition, in the conversion of tetrahedral meshes to hexahedral meshes [Meshkat 2000], five or six tetrahedra become one hexahedron without removing any vertices. In order to avoid the drastic increase of number of elements through the conversion of hexahedral meshes to tetrahedral meshes, decompositions of a hexahedron into five or six tetrahedra without inserting any vertices are described in the next section.

As geometric operations, the dimension-driven tetrahedral mesh deformation method described in Chapter 2 and the mesh adaptation method described in Chapter 3 are used.

As mentioned in section 4.2, many conversion methods of tetrahedral meshes to hexahedral meshes are proposed as a key technique in indirect hexahedral mesh generation methods. In this thesis, a conversion method based on RF-graph [Meshkat 2000] is adopted. The details of the application of the method in this research are described in section 4.5.

For the evaluation of the combination of the deformation method and adaptation method with the Tet-Hex conversion in [Meshkat 2000], the Tet-Hex conversion is applied to resultant meshes of the deformation method and the adaptation method, and ratios of volume and number of the hexahedral elements to that of all elements are checked and evaluated in section 4.6.

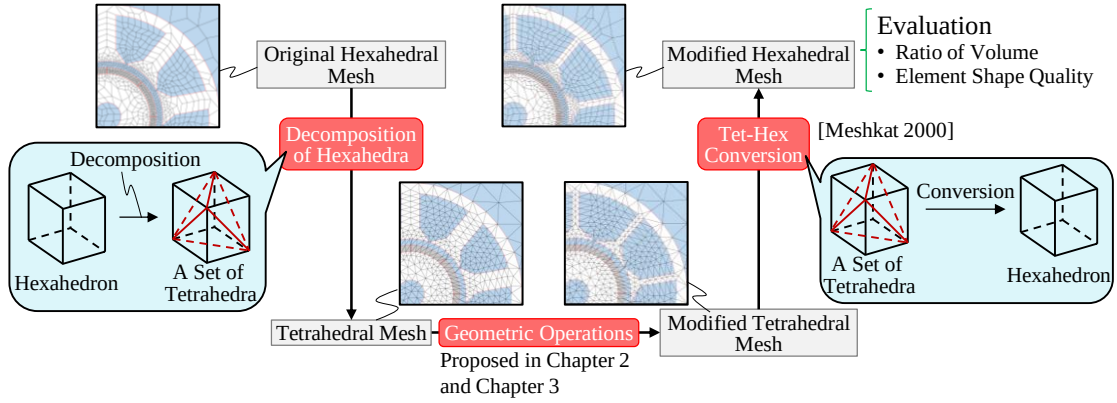


Fig. 4.4 The overview of the validation

4.4 Decomposition of hexahedra without inserting any vertices

In order to obtain tetrahedral meshes from hexahedral meshes, each hexahedron is divided into a set of tetrahedra. In the conversion between tetrahedral meshes and hexahedral meshes [Meshkat 2000], five or six tetrahedra become one hexahedron without removing any vertices. In order to convert tetrahedral meshes into hexahedral meshes as easily as possible, each hexahedron divided into five or six tetrahedra without inserting any vertices. As shown in Fig. 4.5, the following six types of the decomposition and their symmetry (the decomposition which use the other diagonal line of each quadrangle) satisfy this condition.

- (a) A hexahedron is divided into four tetrahedra at two pairs of opposite corners and one tetrahedron at the center (called Type A^+ , and its symmetry described by Type A^- in this thesis).
- (b) In decomposition Type A^+ , Flipping 2-3 mentioned in sub-subsection 2.6.5.4 are applied to the center tetrahedron and one of the corner tetrahedra (called Type B^+ , and its symmetry described by Type B^- in this thesis). In Fig. 4.5(b), Flipping 2-3 is applied to two tetrahedra $\tau^{cen} = \{5,4,2,7\}$ and $\tau^{cor} = \{5,2,6,7\}$ of the decomposition in Fig. 4.5(a).
- (c) A hexahedron divided into six tetrahedra shearing an edge which is a diagonal line of the hexahedron (called Type C^+ , and its symmetry described by Type C^- in this thesis).
- (d) In decomposition Type C^+ , edge flipping is applied to a pair of triangles which forms a quadrangle of the hexahedron (called Type D^+ , and its symmetry described by Type D^- in this thesis). In Fig. 4.5(d), edge flipping is applied to a pair triangle $t^{rb} = \{1,3,2\}$ and $t^{lb} = \{1,4,3\}$ of the decomposition in Fig. 4.5(c).
- (e) In decomposition Type C^+ , edge flipping is applied to two pairs of triangles which form two opposite quadrangles of the hexahedron (called Type E^+ , and its symmetry described by Type E^- in this thesis). In Fig. 4.5(e), edge flipping is applied to two pairs

of triangles (t^{rb}, t^{lb}) and $(t^{rt} = \{5,6,7\}, t^{lt} = \{5,7,8\})$ of the decomposition in Fig. 4.5(c).

- (f) In decomposition Type B^+ , edge flipping is applied to a pair of triangles which forms a quadrangle of the hexahedron and neighboring tetrahedra form a pyramid (called Type F^+ , and its symmetry described by Type F^- in this thesis). In Fig. 4.5(f), edge flipping is applied to a pair triangle t^{rt} and t^{lt} of the decomposition in Fig. 4.5(b).

As described in section 4.3, input meshes are all-hex meshes generated by regular grids in the validation. Therefore, as shown in Fig. 4.6(a), if a hexahedral mesh is converted into a tetrahedral mesh by using the decomposition Type A^+ , Type B^+ , Type D^+ , or Type F^+ , its symmetry or other decomposition types are needed in order to avoid topological conflicts. On the other hand, as shown in Fig. 4.6(b), hexahedral meshes can be converted into tetrahedral meshes by using only Type C^+ , Type E^+ , or their symmetries. For a cube, all resultant six tetrahedra of Type C^+ become the same shape, and if a hexahedral mesh consists of well aligned hexahedra, the number of neighboring tetrahedra of each inner edge of the resultant tetrahedral mesh becomes four or six. These facts may be useful for an extension of the proposed method in future work. Therefore, in this thesis, the decomposition Type C^+ is adopted.

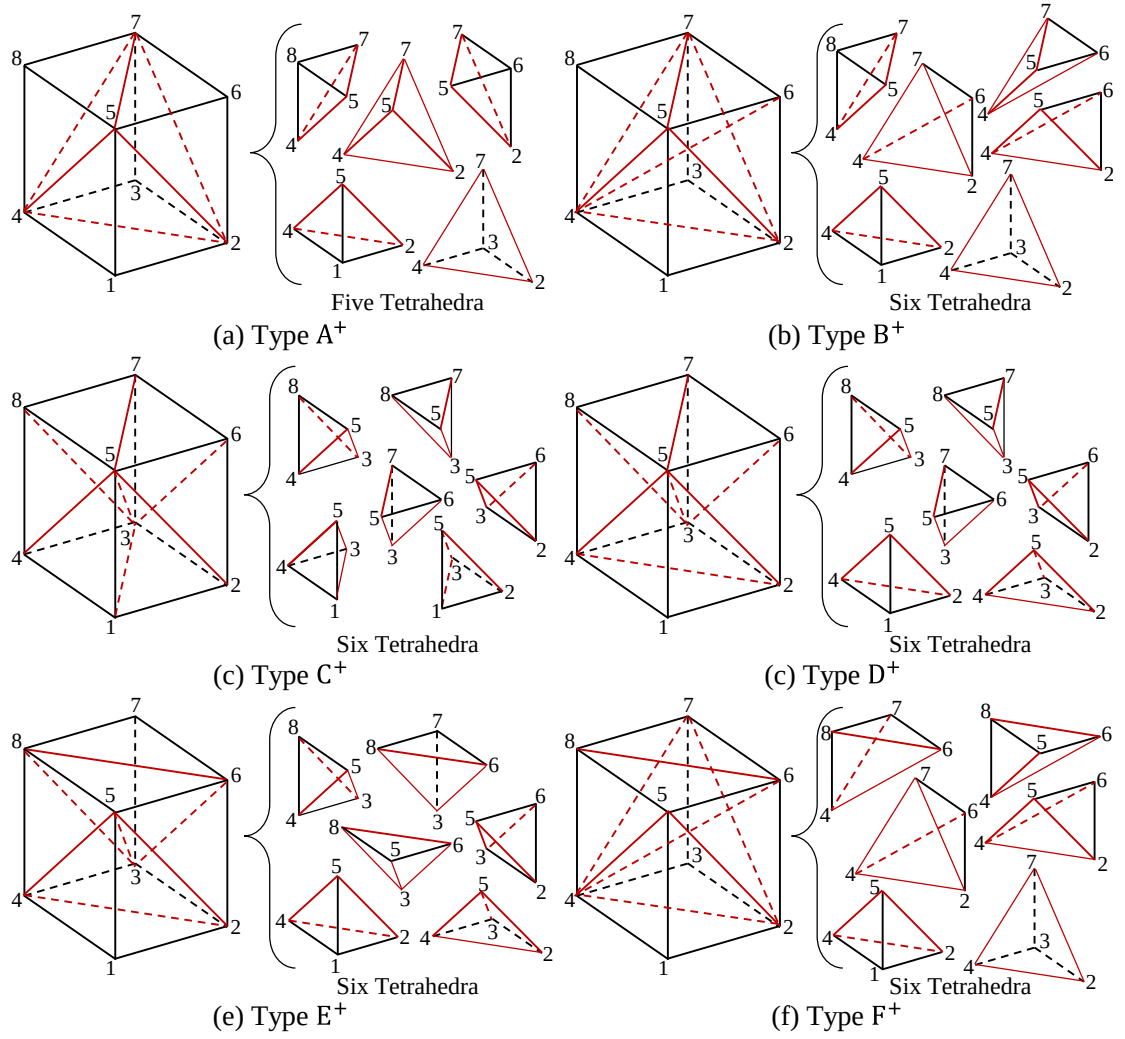


Fig. 4.5 Six decomposition types for a hexahedron

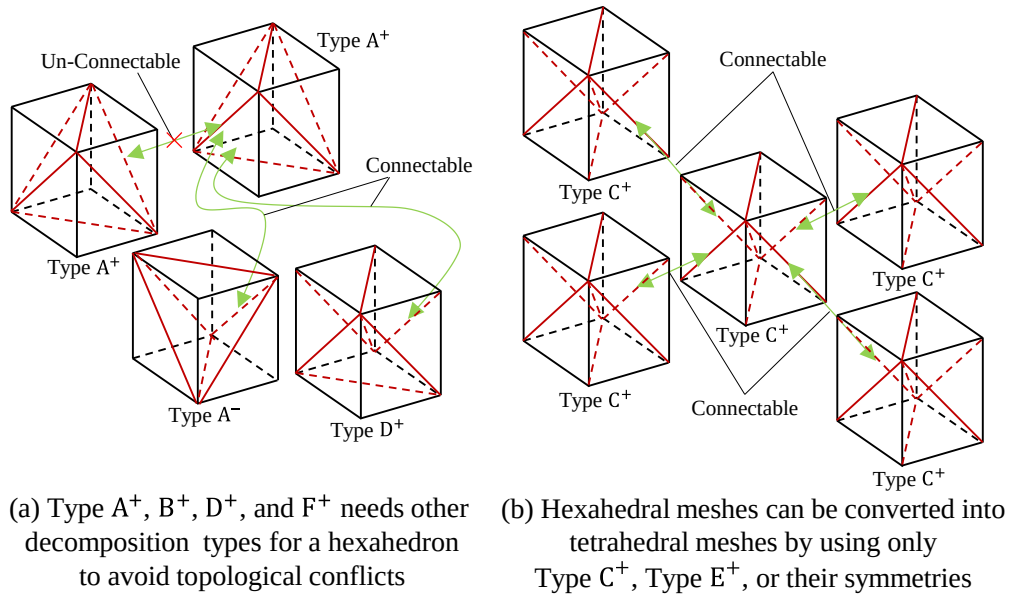


Fig. 4.6 Possibility of connection between decompositions of hexahedra

4.5 Conversion between tetrahedral meshes and hexahedral meshes based on graph representation

4.5.1 Overview of the conversion between tetrahedral meshes and hexahedral meshes

In the validation, an indirect hexahedral mesh generation method [Meshkat 2000] is simplified and used. The overview of the simplified conversion between tetrahedral meshes and hexahedral meshes is shown in Fig. 4.7. The method consists of the following two processes: One is Local Search (A1) where candidates of hexahedra formed by five or six tetrahedra are searched from the input tetrahedral mesh using the extended Region-Face-graph (RF-graph). The other is Greedy Conversion (A2) where the conversion between the tetrahedral mesh and a hex-dominant mesh is performed by a greedy algorithm using the extracted candidates.

In this section, the idea of the extended RF-graph and the representation of each decomposition type by extended RF-graph are first described. Then, the local search (A1) which searches for candidates of hexahedra from the input tetrahedral mesh is mentioned. Finally, the greedy conversion (A2) between the tetrahedral mesh and hexahedral mesh is described.

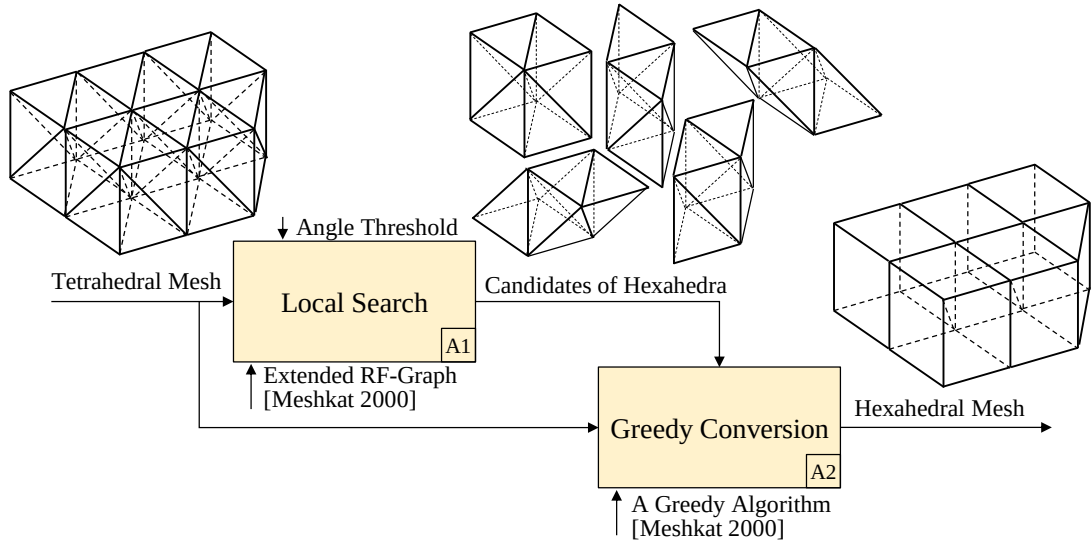


Fig. 4.7 The overview of the simplified indirect hexahedral mesh generation method

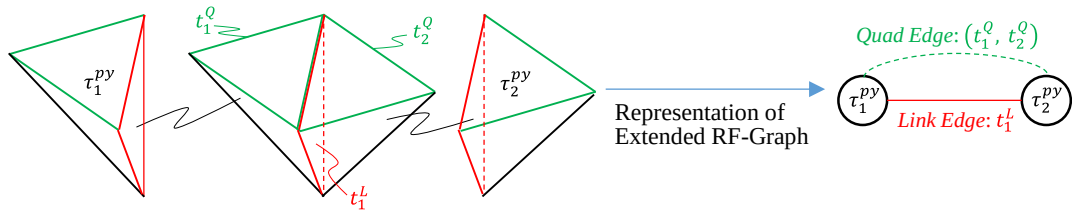


Fig. 4.8 Extended RF-graph of two tetrahedra composing a pyramid

4.5.2 Extended Region-Face-graph (RF-graph) representation

The Tet-Hex conversion is based on extended Region-Face-graph (RF-graph) [Meshkat 2000]. An example of the extended RF-graph is shown in Fig. 4.8. In the extended RF-graph, each tetrahedron is represented by a graph node, and two types of graph edges are used. One corresponds to a triangle shared by two tetrahedra and it is represented by solid line as shown in Fig. 4.8. This type of graph edges is called “*link edge*” in this thesis. The other corresponds to a pair of triangles of different two tetrahedra, which forms a quadrangle. The latter graph edge is represented by a dashed line as shown in Fig. 4.8, and called “*quad edge*” in this thesis.

The extended RF-graph of each decomposition type of a hexahedron is shown in Fig. 4.9. In the conversion method, candidates of hexahedra formed by five or six tetrahedra are extracted by the latter local search process using these graph.

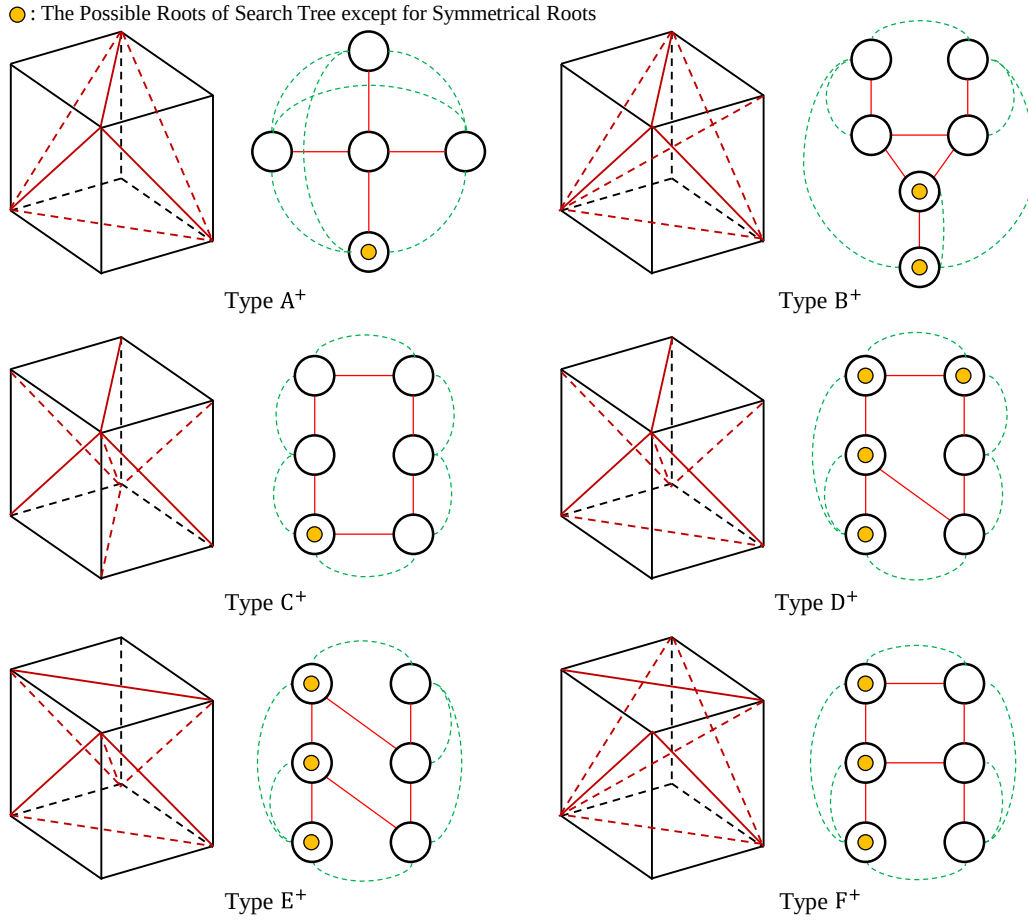


Fig. 4.9 Extended RF-graphs of each decomposition type of a hexahedron

4.5.3 Local search using extended RF-graph

In the local search, extended RF-graphs shown in Fig. 4.9 are searched by a sequence of two operations: “LINK” and “QUAD.” In the $\text{LINK}(\tau_1, \tau_2)$ operation, a *link edge* between two nodes corresponding to tetrahedra τ_1 and τ_2 is searched. In other words, the $\text{LINK}(\tau_1, \tau_2)$ operation becomes “Success” if τ_1 and τ_2 have a common triangle, and otherwise, it becomes “Failure.” On the other hand, the $\text{QUAD}(\tau_1, \tau_2)$ searches for a *quad edge* between two nodes corresponding to tetrahedra τ_1 and τ_2 . Therefore, if there is a pair of triangles of τ_1 and τ_2 forms a quadrangle and the cosine of their dihedral angle is larger than a threshold δ^{angle} , the $\text{QUAD}(\tau_1, \tau_2)$ operation becomes “Success,” and otherwise, the operation becomes “Failure.”

In the conversion in this thesis, all possible six RF-graphs are searched from each tetrahedron of input tetrahedral meshes. Therefore, only one search tree is used for the search of each extended RF-graph. The six search trees and sequences of the two operations corresponding to the six extended RF-graph are shown in Fig. 4.10.

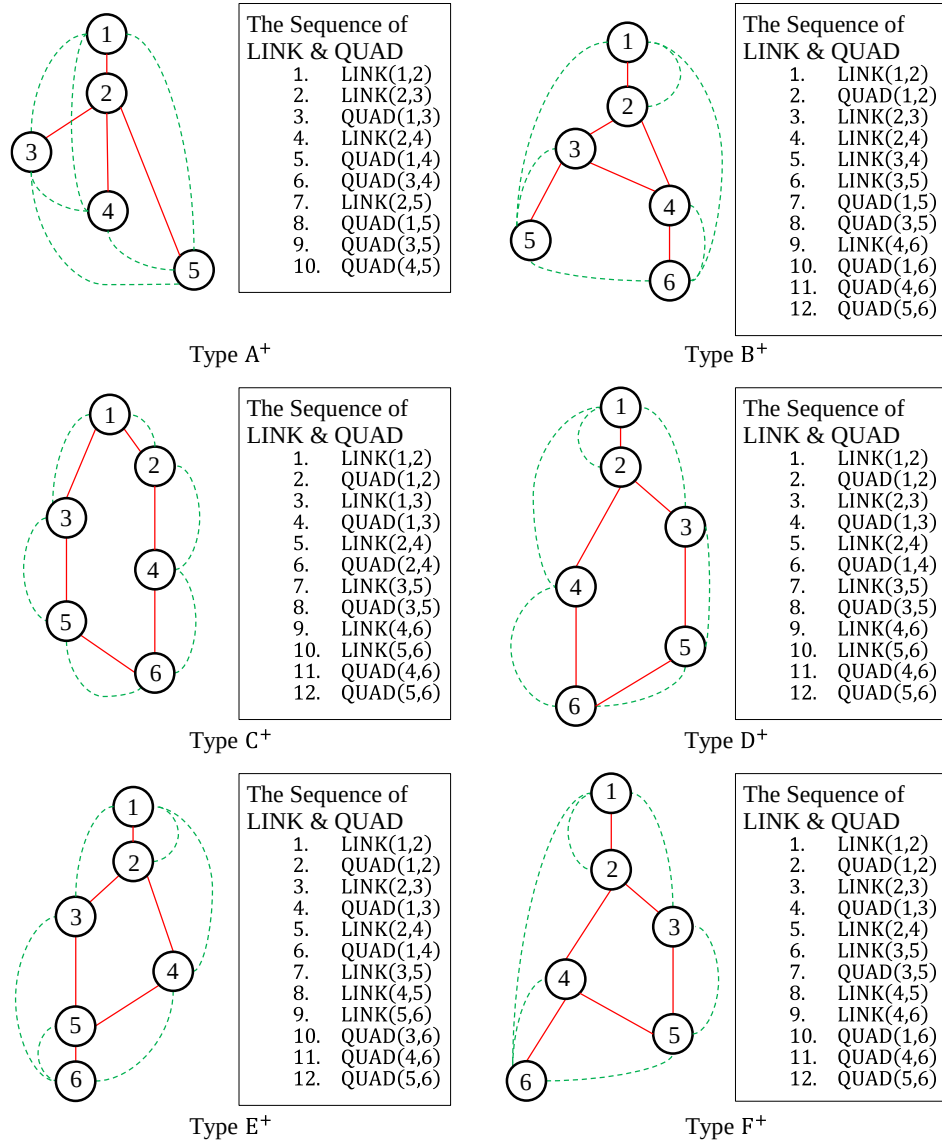


Fig. 4.10 Six search trees and sequences of two operations corresponding to the six extended RF-graphs

In the local search, all possible extended RF-graphs are searched by these search trees where each tetrahedron of the input tetrahedral mesh is used as the roots. In each search process, four or six LINK operations and six QUAD operations are performed sequentially. For any two tetrahedra neighboring each other, the LINK operation becomes *Success*. Therefore, in the search process, the search of a neighboring tetrahedron is performed as a LINK operation. If a LINK (QUAD) operation becomes *Failure*, the search process is returned to an earlier LINK operation (i.e. the search of a neighboring tetrahedron). If all operations become *Success*, a candidate of a hexahedron corresponding to the extracted RF-graph is inserted to a max-heap based on a quality Q_{can} whose

detail is described later. The search process is continued until all possible operations for the search tree are tested. For example, a hexahedron decomposed by Type E^+ is found by the following ten steps:

- Step 1 Given a tetrahedron τ_1 corresponding to the node 1, a tetrahedron τ_2 neighboring τ_1 is extracted. The tetrahedron τ_2 corresponds to the node 2. If all neighboring tetrahedra have been used as the tetrahedron τ_2 corresponding to the node 2 of the current search tree where τ_1 is used as its root, the search process by the current search tree using τ_1 as its root is ended.
- Step 2 QUAD(1,2) is performed. If the QUAD operation becomes *Success*, the search process is proceeded to Step 3. Otherwise, the search process goes back to Step 1.
- Step 3 A neighboring tetrahedron τ_3 of τ_2 except for τ_1 is extracted. The tetrahedron τ_3 corresponds to the node 3. If τ_3 cannot be found, the search process goes back to Step 1.
- Step 4 QUAD(1,3) is performed. If the QUAD operation becomes *Success*, the search process is proceeded to Step 5. Otherwise, the search process goes back to Step 3.
- Step 5 A neighboring tetrahedron τ_4 of τ_2 except for τ_1 and τ_3 is extracted. The tetrahedron τ_4 corresponds to the node 4. If τ_4 cannot be found, the search process goes back to Step 3.
- Step 6 QUAD(1,4) is performed. If the QUAD operation becomes *Success*, the search process is proceeded to Step 7. Otherwise, the search process goes back to Step 5.
- Step 7 A neighboring tetrahedron τ_5 of τ_3 except for τ_1 , τ_2 and τ_4 is extracted. The tetrahedron τ_5 corresponds to the node 5. If τ_5 cannot be found, the search process goes back to Step 5.
- Step 8 LINK(4,5) is performed. If the LINK operation becomes *Success*, the search process is proceeded to Step 9. Otherwise, the search process goes back to Step 7.
- Step 9 A neighboring tetrahedron τ_6 of τ_5 except for τ_1 , τ_2 , τ_3 and τ_4 is extracted. The tetrahedron τ_6 corresponds to the node 6. If τ_6 cannot be found, the search process goes back to Step 7.
- Step 10 QUAD(3,6), QUAD(4,6), and QUAD(5,6) are performed. If and only if all of the operations becomes *Success*, the combination $\{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6\}$ is inserted to the max-heap, and the search process goes back to Step 9. Otherwise, the search process goes back to Step 9.

In addition, graphs corresponding to a prism (see Fig.4.11) and a pyramid (see Fig. 4.8) are also searched and inserted to the max-heap. In order to adopt candidates of high quality hexahedra preferentially, the priority Q_{can} is given depend on the types of the candidates, the dihedral angles between triangles forming quadrangle faces, and the interior angles of the resultant quadrangles. The priority Q_{can} is defined by Eq. (4.1):

$$Q_{can} = \frac{1}{2|Q^{dec}|} \sum_{q_i \in Q^{dec}} \cos \theta^{da}(q_i) + \frac{1}{2|Q^{dec}|} \sum_{q_i \in Q^{dec}} \min_{j \in V_{q_i}} \{\sin \theta^{fa}(q_i, j)\} + \gamma_{type}, \quad (4.1)$$

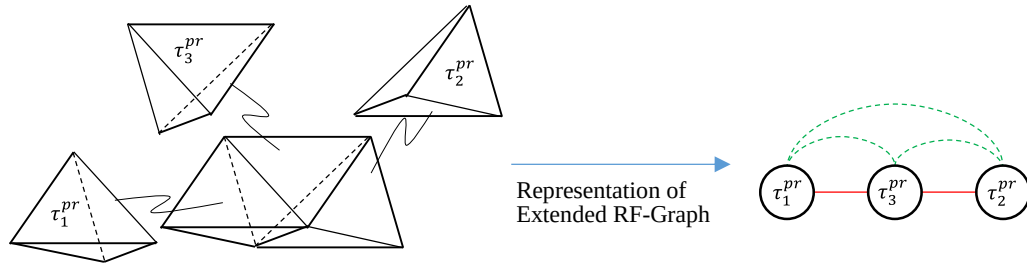


Fig. 4.11 Extended RF-graphs of prisms

where Q^{dec} is a set of quadrangles included in each candidate, q_i a quadrangle, $\theta^{da}(q_i)$ a dihedral angle of triangles composing q_i , which corresponds to “*dangle*” in [Meshkat 2000], V_{q_i} a set of vertices j included in q_i , $\theta^{fa}(q_i, j)$ an interior angle of q_i corresponding to j , which corresponds to “*fangle*” in [Meshkat 2000], and γ_{type} is a weight depend on the resultant polyhedron and it takes 10 for a candidate of a hexahedron, 4 for that of a prism, and 0 for that of a pyramid [Bottela 2016].

4.5.4 Greedy conversion between tetrahedral meshes and hexahedral meshes

After the extraction of all candidates of hexahedra formed by five or six tetrahedra, the input tetrahedral mesh is converted into a hexahedral mesh by adopting some candidates. Finding an optimal set of candidates in a 3D space is a classical combinatorial optimization problem which is a NP-Complete problem (see [Bottela 2016] for the detail). Therefore, a “good” solution which may not be the optimal solution but is not far away from the optimal solution is found heuristic approaches are used in [Meshkat 2000, Bottela 2016]. In [Bottela 2016], five types of heuristic approaches are compared, and the simplest greedy approach (called GMax in [Bottela 2016]) is outperformed the others in terms of the ratio between the total volume of hexahedra and that of all elements. In this thesis, based on their experimental results, the combination of the candidates is founded by the simplest greedy approach.

The greedy conversion is performed by the following four steps:

- Step 1 A candidate is popped from the max-heap.
- Step 2 Tetrahedra included in the popped candidate are checked whether they are included in candidates which are adopted previously or are not included in the candidates.
- Step 3 If all of them are not included in the candidates, the popped candidate is adopted. Otherwise, the popped candidate is rejected.
- Step 4 The above steps are repeated until the max-heap becomes empty.

4.6 Experimental results and evaluation

4.6.1 Overview of experiments

In order to show and discuss the applicability of the combination of the proposed mesh deformation and adaptation methods with the Tet-Hex conversion method, tetrahedral meshes which are generated by the decomposition of hexahedral meshes and edited by the proposed deformation and adaptation methods are converted into hexahedral meshes by the Tet-Hex conversion, and the results are evaluated by element shape qualities and the volume ratio of each element type. At first, a deformed tetrahedral mesh is converted into hexahedral meshes using some values of δ^{angle} (an angle threshold for the local search described in subsection 4.5.3). After that, some conformal tetrahedral meshes which consist of a single moving object mesh and a space mesh and are obtained by the proposed mesh adaptation method are converted and evaluated.

In order to evaluate the resultant hexahedral meshes, the scaled Jacobian [Knupp 2000] defined as Eq. (4.2) is used:

$$Q_{SJ}(i, \tau) = \frac{(\mathbf{x}_{j_1} - \mathbf{x}_i) \cdot (\mathbf{x}_{j_2} - \mathbf{x}_i) \times (\mathbf{x}_{j_3} - \mathbf{x}_i)}{\|\mathbf{x}_{j_1} - \mathbf{x}_i\| \|\mathbf{x}_{j_2} - \mathbf{x}_i\| \|\mathbf{x}_{j_3} - \mathbf{x}_i\|}, \quad (4.2)$$

where τ is a polyhedron (e.g. a tetrahedron, a prism, or a hexahedron), $\mathbf{x}_i$ the position of a vertex i of τ , and $\mathbf{x}_{j_k}$ ($k = 1, 2, 3$) are positions of other vertices j_k of τ , which are ordered so that $Q_{SJ}(i, \tau) = 1.0$ if τ is a rectangular parallelepiped (see Fig. 4.12). The scaled Jacobian $Q_{SJ}(i, \tau)$ takes 0 when τ is flat at $\mathbf{x}_i$, and takes a negative value for an concave element. Because a scaled Jacobian represents a quality evaluated at only a vertex and is not sufficient to evaluate the shape quality of the element, in general, the minimum scaled Jacobian of all vertices included in an element is used as a shape quality of the element. Although the minimum scaled Jacobian cannot be defined for pyramids because one vertex of a pyramid has four incident triangles, the other element types can be evaluated by it. The minimum scaled Jacobian takes a value included in ranges from -1.0 to 1.0 for a hexahedron, from -0.866 to 0.866 for a prism, and from -0.707 to 0.707 for a tetrahedron.

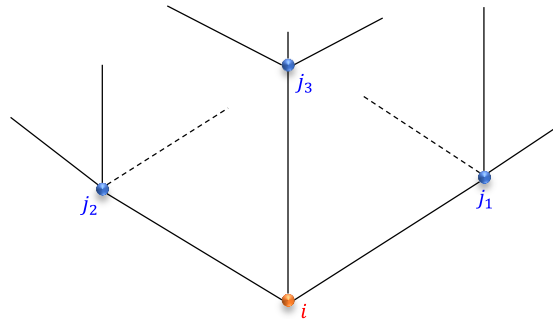


Fig. 4.12 The order of vertices for the calculation of the scale Jacobian

All the experiments were processed on a personal computer having following spec.

- OS: Windows 7 Professional 64bit.
- CPU: Intel Core i7-5960X 3.00GHz.
- RAM: 64.0 GB.
- Development environment: Microsoft Visual Studio 2010.
- Programing language: C++.
- Graphic API: Open GL.

4.6.2 Conversions of deformed tetrahedral meshes

The input all-hex mesh (#vertices: 132, #hexahedra: 60) is shown in Fig. 4.13(a), and the tetrahedral mesh (#vertices: 132, #tetrahedra: 360) generated from the input mesh using the decomposition Type C^+ is shown in Fig. 4.13(b). As shown in Fig. 4.14, the distance between two parallel planes were changed from 6 to 24 by the proposed dimension-driven tetrahedral mesh deformation method. The element shape qualities of the deformed mesh are shown in Table 4.2. The minimum stretch of the deformed mesh was 0.35, and the minimum scaled Jacobian of all tetrahedra was 0.17. As shown in theses quality measures, any degenerated elements were not included in the deformed mesh.

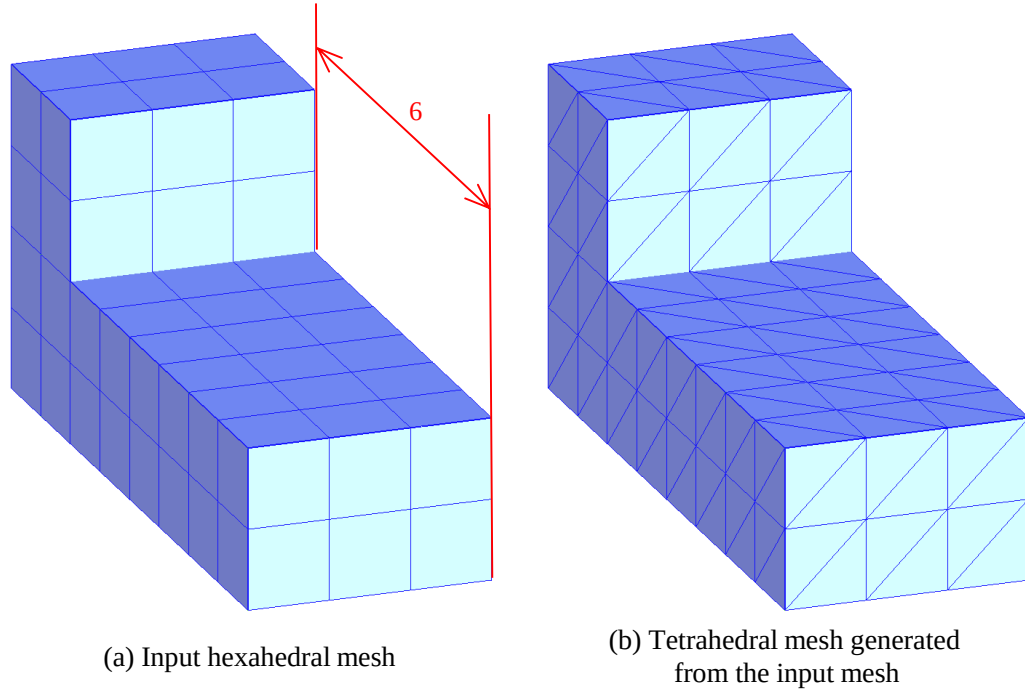


Fig. 4.13 Input hexahedral mesh and tetrahedral mesh generated by the decomposition

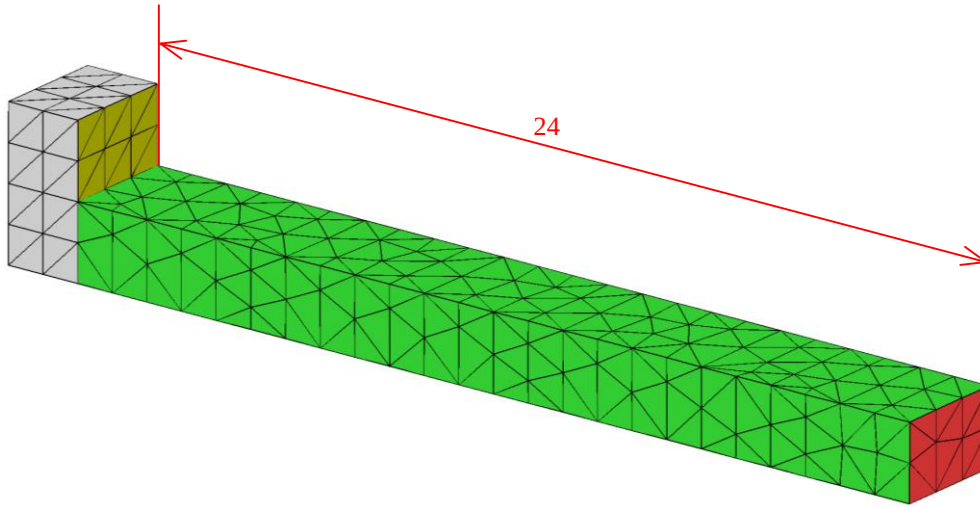


Fig. 4.14 The deformed tetrahedral mesh by the proposed method

Table 4.2 Element shape qualities of the deformed tetrahedral mesh

#Vertices	#Tetrahedra	Total Volume	Stretch		The minimum scaled Jacobian	
			Minimum	Average	Minimum	Average
373	1143	168	0.35	0.66	0.17	0.41

The results of the Tet-Hex conversion where the angle threshold δ^{angle} was set from $\cos 0^\circ$ to $\cos 90^\circ$ in 10 degree increments are shown in Fig. 4.15. In Fig. 4.15, each polyhedra is color coded: hexahedra, prisms, pyramids, and tetrahedra are represented by red, green, blue, and gray, respectively. It is shown that the ratio of hexahedra became larger when δ^{angle} became smaller.

For the evaluation, the total volumes and numbers of each type of polyhedra are summarized in Table 4.3, and the minimum scaled Jacobian of each type of polyhedra is shown in Table 4.4. Because the outside of the *deformable region* were not changed, 24 hexahedra were easily re-composed from tetrahedra through the Tet-Hex conversion. Therefore, the volume and number of hexahedra without the 24 hexahedra are also shown in Table 4.3. When δ^{angle} is 0, the ratio of the sum of volumes of hexahedra became about 77 percent of the total volume of all elements. The change of the volume ratio of hexahedra on the *deformable region* and the decrease of the number of tetrahedra according to δ^{angle} are shown in Fig. 4.16(a) and (b). These graph show that the increase of hexahedra and the decrease of tetrahedra according to δ^{angle} became small when δ^{angle} was close to 0. In addition, as shown in Table 4.4, because any other geometrical thresholding is not performed, elements which have a negative value of the minimum scaled Jacobian increased when δ^{angle} became smaller. Although the deformed tetrahedral mesh has high element shape qualities, it was impossible to obtain a high quality hexahedral mesh by only adjusting δ^{angle} . Because the conversion between deformed tetrahedral meshes and hexahedral meshes by only adjusting δ^{angle} will fall into a trade-off problem between element shape qualities

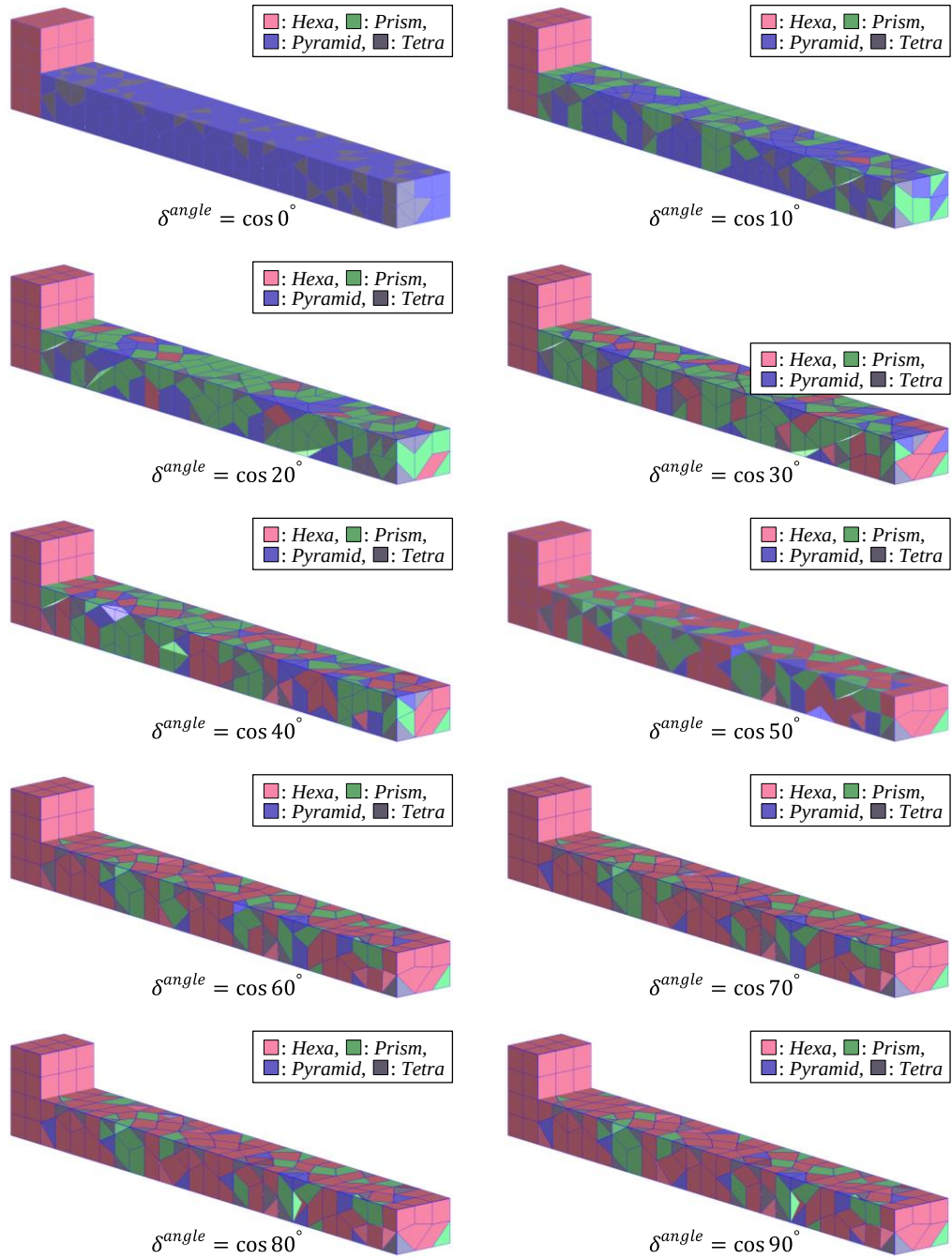


Fig. 4.15 Results of the Tet-Hex conversion of the deformed mesh

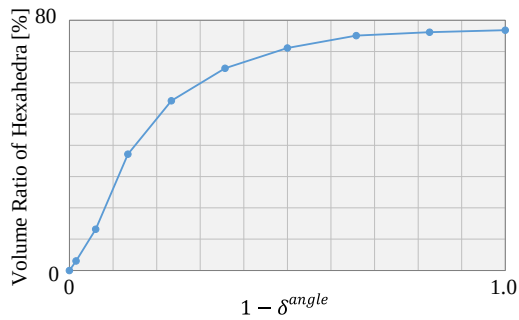
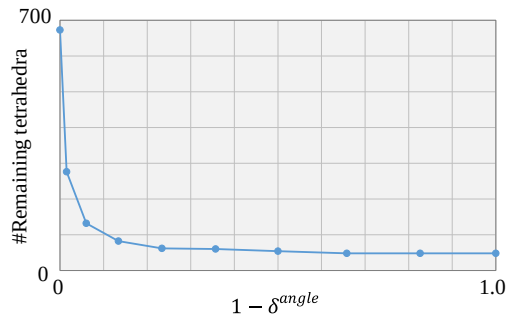
and the ratio of hexahedra in the resultant hexahedral mesh, some editing of tetrahedral meshes such as quality improvement where the Tet-Hex conversion is taken into account will be needed.

Table 4.3 The volume and number of each polyhedron after the Tet-Hex conversion

Angle threshold δ^{angle} for QUAD operation	Volume (Ratio [%])					#Elements (Ratio [%])				
	Total	Hexa	Prism	Pyramid	Tetra	Total	Hexa	Prism	Pyramid	Tetra
$\cos 0^\circ$	168	24 (14) /0 (0)	0 (0)	44.25 (26)	99.75 (59)	860	24 (2.8) /0 (0)	0 (0)	163 (19)	673 (78)
$\cos 10^\circ$		28.38 (17) /4.38 (3.0)	43.67 (26)	54.89 (33)	41.06 (24)	605	28 (4.6) /4 (0.69)	97 (16)	204 (34)	276 (46)
$\cos 20^\circ$		42.98 (26) /18.98 (13)	69.53 (41)	36.55 (22)	18.94 (11)	470	45 (9.6) /21 (4.7)	155 (33)	138 (30)	132 (28)
$\cos 30^\circ$		77.57 (46) /53.57 (37)	59.66 (36)	20.29 (12)	10.47 (6.2)	382	79 (21) /55 (15)	145 (38)	76 (20)	82 (21)
$\cos 40^\circ$		102.15 (61) /78.15 (54)	42.697 (25)	15.95 (9.5)	7.20 (4.2)	334	109 (33) /85 (27)	102 (31)	61 (18)	62 (19)
$\cos 50^\circ$		117.08 (70) /93.08 (65)	37.96 (23)	6.15 (3.7)	6.81 (4.1)	306	127 (42) /103 (37)	90 (29)	29 (9.4)	60 (20)
$\cos 60^\circ$		126.42 (75) /102.42 (71)	29.93 (18)	5.92 (3.5)	5.73 (3.4)	292	137 (47) /113 (42)	74 (25)	27 (9.2)	54 (18)
$\cos 70^\circ$		132.12 (79) /108.12 (75)	23.52 (14)	7.35 (4.4)	5.01 (2.98)	284	143 (50) /119 (46)	60 (21)	33 (12)	48 (17)
$\cos 80^\circ$		133.73 (80) /109.73 (76)	23.15 (14)	6.15 (3.7)	4.97 (2.96)	279	146 (52) /122 (48)	58 (21)	27 (9.7)	48 (17)
$\cos 90^\circ$		134.67 (80) /110.66 (77)	22.42 (13)	5.93 (3.5)	4.99 (2.97)	278	147 (53) /123 (48)	57 (21)	26 (9.4)	48 (17)

Table 4.4 The minimum and average values of the minimum scaled Jacobian

Angle threshold δ^{angle} for QUAD operation	Minimum					Average				
	All	Hexa	Prism	Pyramid	Tetra	All	Hexa	Prism	Pyramid	Tetra
$\cos 0^\circ$	0.17	1.0	-	-	0.17	0.44	1.0	-	-	0.42
$\cos 10^\circ$	0.0	0.24	0.0	-	0.17	0.47	0.92	0.48	-	0.43
$\cos 20^\circ$	0.0	0.06	0.0	-	0.18	0.49	0.73	0.49	-	0.42
$\cos 30^\circ$	-0.12	-0.12	0.0	-	0.22	0.49	0.60	0.47	-	0.41
$\cos 40^\circ$	-0.12	-0.12	0.0	-	0.18	0.49	0.59	0.46	-	0.38
$\cos 50^\circ$	-0.32	-0.32	0.0	-	0.18	0.49	0.57	0.46	-	0.37
$\cos 60^\circ$	-0.51	-0.51	0.0	-	0.18	0.50	0.58	0.46	-	0.36
$\cos 70^\circ$	-0.32	-0.32	0.0	-	0.18	0.50	0.56	0.46	-	0.36
$\cos 80^\circ$	-0.38	-0.38	0.0	-	0.18	0.49	0.55	0.45	-	0.36
$\cos 90^\circ$	-0.38	-0.38	0.0	-	0.18	0.49	0.55	0.45	-	0.36

(a) Volume ratio of hexahedra - $(1 - \delta^{angle})$ (b) #Remaining tetrahedra - $(1 - \delta^{angle})$ **Fig. 4.16** The change of the volume ratio of hexahedra and the number of tetrahedra according to the angle threshold for QUAD operation

4.6.3 Conversions of tetrahedral meshes adapted to object motion

The Tet-Hex conversion is also applied to some tetrahedral mesh generated by the proposed tetrahedral mesh adaptation method. The input all-hex mesh (#vertices: 4,096, #hexahedra: 3,375) is shown in Fig. 4.17(a), and the tetrahedral mesh (#vertices: 4,096, #tetrahedra: 20,250) generated from the input mesh using the decomposition Type C^+ is shown in Fig. 4.17(b). The swastika object mesh located the center of the cubic space mesh were rotated by 0.1 degree around an axis which passes the center of the cube and is parallel to y axis in each motion step. The total number of motion steps is 900. The Tet-Hex conversion was applied to four tetrahedral meshes which are generated at 53th, 450th, and 900th motion step respectively, and the result shown in Fig. 4.18(a), (b), (c), and (d) respectively. The results are summarized in Table 4.5. These experiments are performed by using $\delta^{angle} = \cos 60^\circ$ based on subsection 4.6.2.

In Fig. 4.18(a), the Tet-Hex conversion is applied to a tetrahedral mesh generated by the proposed tetrahedral mesh adaptation method at 53th motion step. In the proposed mesh adaptation method, any topological changes are not performed from the initial motion step until this motion step. Therefore, the topology of the tetrahedral mesh was the same as that of the tetrahedral mesh shown in Fig. 4.17(b). Although some distorted hexahedra can be included in the resultant mesh, the tetrahedral mesh can be converted into an all-hex mesh. As shown in Table 4.5, the minimum value of the minimum scaled Jacobian was 0.27 before the conversion and 0.88 after the conversion, respectively.

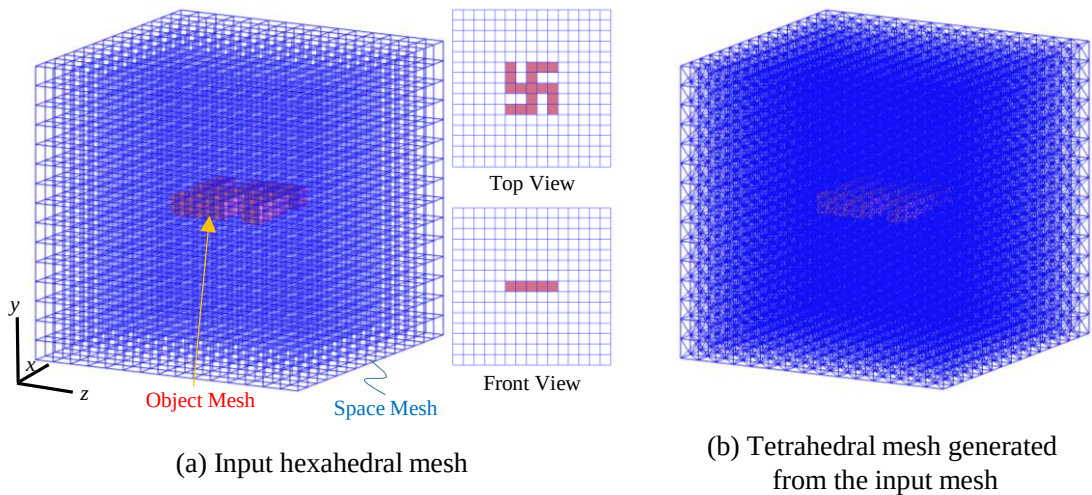


Fig. 4.17 Input conformal hexahedral mesh and tetrahedral mesh generated by the decomposition

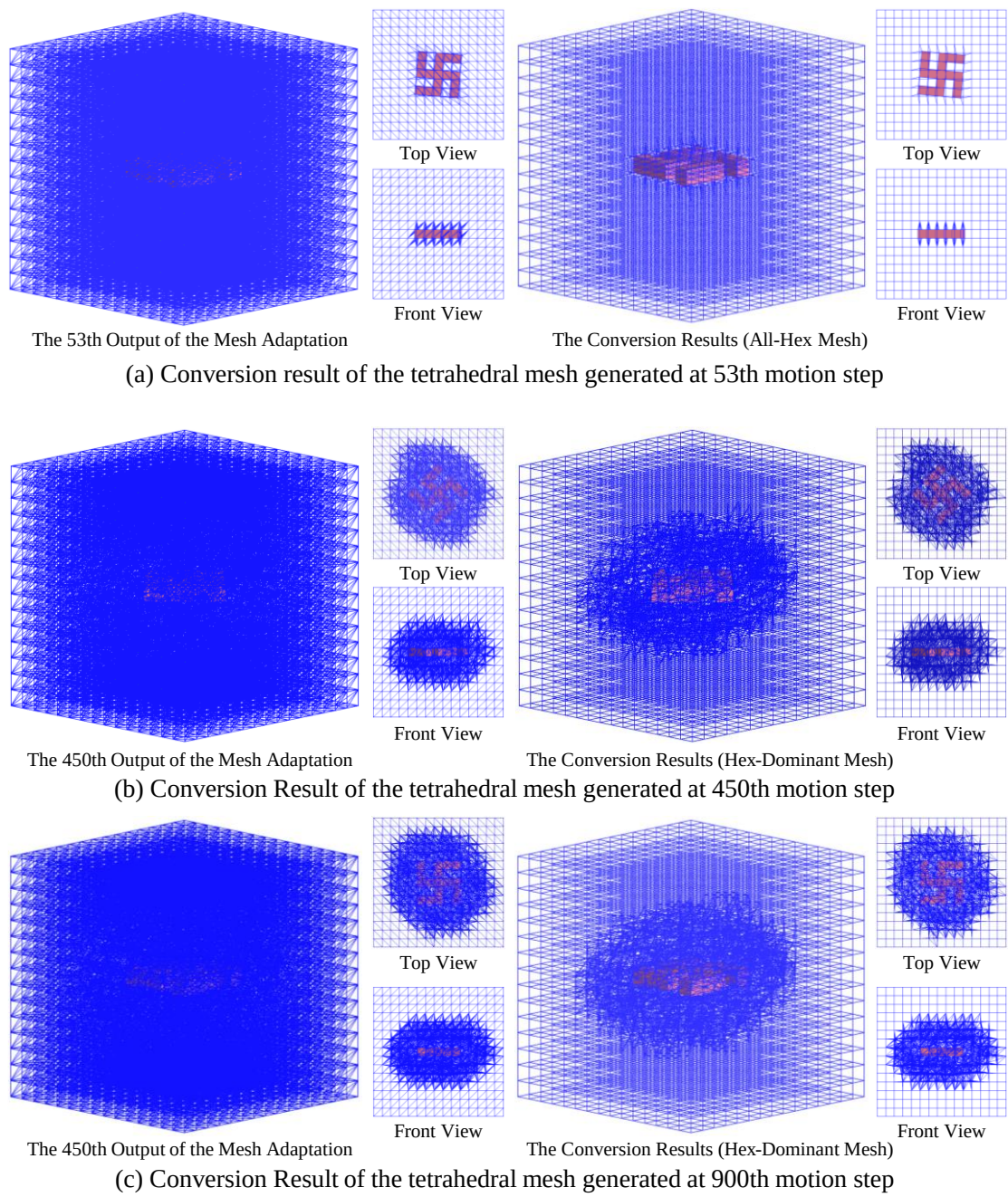


Fig. 4.18 The conversion results from some conformal tetrahedral meshes generated by the proposed tetrahedral mesh adaptation method

Table 4.5 The conversion results of tetrahedral meshes adapted to an object motion

Mesh models	#elements of input tetrahedral mesh		The minimum scaled Jacobian				#elements of output hexahedral mesh			
			Before the conversion		After the conversion					
	#vertices	#tetrahedra	Minimum	Average	Minimum	Average	#hexahedra	#prism	#pyramid	#tetrahedra
(a) 53th	4,096	20,250	0.27	0.41	0.88	0.998	3,375	0	0	0
(b) 450th	4,434	21,629	0.03	0.399	−0.66	0.79	3,262	432	302	293
(c) 900th	4,521	21,999	0.02	0.40	−0.89	0.76	3,268	513	345	350

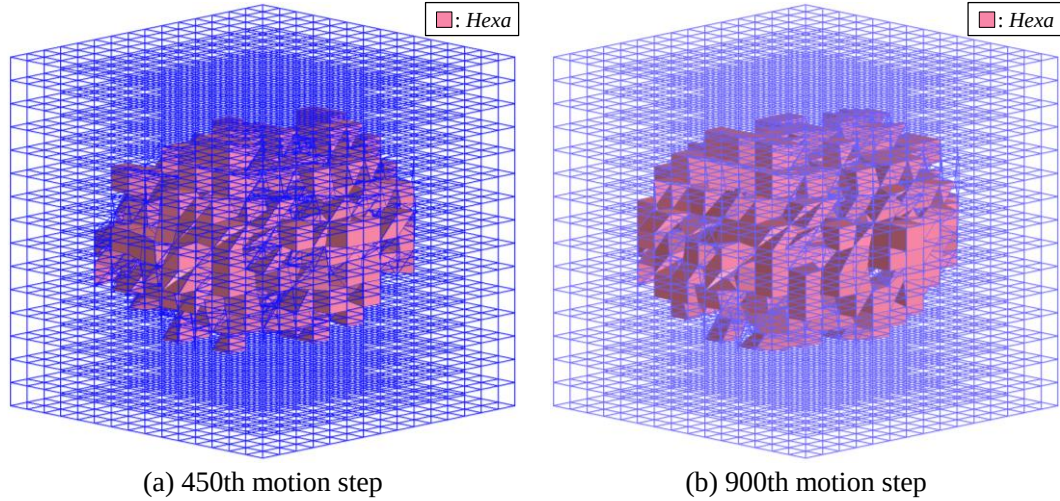


Fig. 4.19 Resultant hexahedra with the minimum scaled Jacobian lower than a threshold

The proposed tetrahedral mesh adaptation method changed only the deformed region of tetrahedral meshes. Therefore, tetrahedra located the outside of the deformed region can be combined into the original hexahedra easily, and the average element shape qualities of resultant hexahedral meshes were increased by them. In Fig. 4.19(a) and (b), hexahedra whose the minimum scaled Jacobian is smaller than 0.8 in the resultant hexahedral mesh shown in Fig. 4.18 (b) and (c) are represented by red. In these regions of the resultant hexahedral mesh converted from the tetrahedral mesh of 450th motion step and 900th motion step included 483 and 582 hexahedra respectively. If these numbers are the number of all hexahedra included in the deformed region, the ratio of hexahedra in the deformed region became about 32% and 33%, respectively. Because a hexahedron is created from five or six tetrahedra, and two prisms or three pyramid can become a hexahedron, the ratio of the number of hexahedra are not increased so much, and the tendency is shown in another existing method [Yamakawa 2003] (in their experiments, the ratio of the number of the resultant hexahedra is between 30% and 40%). Therefore, the ratios of the number of resultant hexahedra in the deformed region were reasonable.

Although element shape qualities of resultant hexahedral meshes and the conformity between hexahedra were not considered in this experiments, it is shown that a lot of hexahedra could be created in the tetrahedral meshes edited by the proposed deformation and adaptation method

through the simple greedy-based Tet-Hex conversion.

4.7 Summary

In this chapter, in order to realize certain geometric operations of hexahedral meshes, an application of deformation and adaptation methods to hexahedral meshes based on Tet-Hex conversion was conducted. For the validation, the proposed dimension-driven tetrahedral mesh deformation method and tetrahedral mesh adaptation method were combined with an indirect hexahedral mesh generation method based on the conversion between tetrahedra and hexahedra [Meshkat 2000]. In order to combine them, the decomposition types for a hexahedron are discussed and the conversion method is introduced. The two topics are summarized as follows.

- *Decomposition of hexahedra without inserting any vertices*: in order to obtain tetrahedral meshes from hexahedral meshes, each hexahedron is divided into a set of tetrahedra. In section 4.4, the decomposition types of a hexahedron, where any vertices are not inserted to the hexahedron were introduced. In addition, a decomposition type was adopted for the tetrahedral mesh generation in terms of avoiding topological conflicts and extensibility of the conversion method.
- *Conversion between tetrahedral meshes and hexahedral meshes based on graph representation*: in the indirect hexahedral mesh generation method proposed by [Meshkat 2000], in order to search for tetrahedra which form a hexahedron, each decomposition type for a hexahedron is represented by extended RF-graphs. In the method, all candidates of hexahedra are extracted by sequences of two types search operations. After the extraction, the input tetrahedral mesh is converted into a hexahedral mesh by adopting candidates using a simple greedy algorithm.

Through some experiments, the following results are obtained.

- A lot of hexahedra were generated from the tetrahedral mesh deformed by the proposed dimension-driven mesh deformation method. In addition, the influence of a threshold for the ratio of resultant hexahedra was investigated. As a result, for realizing a compatibility of the high ratio of resultant hexahedra and high element shape qualities, editing of tetrahedral meshes taking into account the Tet-Hex conversion will be needed.
- Although the conformity between hexahedra and element shape qualities of resultant hexahedral meshes were not considered, the tetrahedral mesh generated by the proposed tetrahedral mesh adaptation method is converted into a hexahedral mesh. If any topological changes are performed in the adaptation method, the all-hex mesh also could be generated. On the other hand, in the deformed region of the mesh adaptation, where many topological operations were performed, the moderate numbers of hexahedra were obtained.

Through these experiments, it was shown that many hexahedra could be generated in the

tetrahedral meshes edited by the proposed deformation and adaptation method through the simple greedy-based Tet-Hex conversion. As a future work, some editing methods of tetrahedral meshes taking into account the Tet-Hex conversion will be needed in order to obtain high quality hexahedral meshes or all-hex meshes from edited tetrahedral meshes.

Chapter 5 Conclusions and Future Work

5.1 Conclusions

In this thesis, the following dimension-driven deformation and adaptation methods of finite element meshes for efficient CAE process were proposed.

- Dimension-driven tetrahedral mesh deformation method which can change several types of dimensions of product shapes like in 3D CAD systems.
- Adaptation method for object motion with contact using conformal tetrahedral meshes where not only objects but also space around objects is decomposed.

In addition, a conversion method between tetrahedral meshes and hexahedral meshes were introduced, and the applicability and limitations of the above proposed methods to hexahedral meshes were shown by combining them. The conclusions of each chapter was summarized as follows:

Chapter 2: Dimension-driven tetrahedral mesh deformation for parameter survey.

A dimension-driven tetrahedral mesh deformation method which consists of the following three functions was proposed and evaluated.

- *Mesh segmentation based on normal tensor and region growing:* in this function, the surfaces of mesh models are divided into planar, quadric, and torus surface regions with G^0 or G^1 continuities by a step-by-step manner based on region-growing. The experimental results showed that the proposed method could accurately divide the surface of tetrahedral meshes into each surface regions even if the surface regions connect with each other with G^0 or G^1 continuities. Although its speeding up may be needed, surface parameters of most surface regions could be extracted in accuracy of 10^{-5} .
- *Dimension-driven shape deformation based on surface information and space embedding:* in this function, the surface regions of the input tetrahedral mesh are classified into four types of regions, and vertices are moved according to the region types. In the experimental results, the proposed method enables us to change parameters of the form features of tetrahedral meshes, such as the fillet radius and chamfer angle.
- *Quality improvement based on Phased ODT smoothing:* in this function, in order to improve all elements of deformed meshes, ODT smoothing [Chen 2011] is applied to deformed meshes in three phases: boundary edge improvement, surface triangle

improvement, and tetrahedron improvement. In addition, degenerated and inverted elements are removed by edge split and edge collapse. Moreover, edge split and edge collapse based on the target mesh density field and the acceptable geometric error are combined with ODT smoothing in order to recover the original (before deformation) mesh properties. The following three features of the proposed method were shown in through the experiments: First, the proposed method could recover the mesh density and shape approximation accuracy of original tetrahedral mesh in deformed meshes. Second, the average and distribution of element shape qualities were significantly improved. Third, inverted and degenerated elements generated by the deformation were removed by the proposed method. As a problem of the proposed method, the calculation time tends to become longer, and some speeding-up techniques such as a parallel processing will be needed for more efficiency.

The proposed dimension-driven tetrahedral mesh deformation method enables us to change parameters of the form features of tetrahedral meshes, such as the fillet radius and chamfer angle, while preserving mesh qualities such as the mesh density, shape approximation accuracy, and element shape quality.

Chapter 3: Tetrahedral mesh adaptation for efficient finite element analysis of assembly models

In Chapter 3, a tetrahedral mesh adaptation method was proposed. The proposed method consists of the following three processes.

- *Local region extraction using surface information and distance field*: for efficient mesh adaptation and handling contacts between objects, the contact region and deformed region are extracted in each motion step. In order to extract two regions, as pre-processing the distance field grid and the vertex search grid are first generated, and surface information is extracted by a segmentation method. By using these two regular grid and surface information, a set of tetrahedron of the space mesh near the moving object mesh and a set of vertices on the overlapping area of surface of two object meshes are extracted as the deformed region and the contact region, respectively.
- *Mesh adaptation using two local regions*: for avoiding degenerated elements, tetrahedra of the space mesh near the contact region are first removed. Then, a rigid transformation is performed, and the deformed region is adapted to the object motion by the space embedding method. In addition, for keeping mesh conformity on the contact regions between object meshes, the topology and geometry of surface triangular meshes of contacting object meshes were adapted by vertex repositioning and local topological operations. Finally, the constraint Delaunay triangulation is applied to the hole created by the removal of tetrahedra of the space mesh near the moving object mesh.
- *Mesh quality improvement based on ODT*: in order to obtain a high quality tetrahedral mesh of each motion step, a quality improvement method based on ODT were performed. In the quality improvement, elements which need to be improved or removed are first

extracted. Then, degenerated and inverted elements are removed by edge split and edge collapse. After that, each edge length is controlled by regular grids and local topological operations. Finally element shape qualities are improved by ODT smoothing.

Through three simple experiments, the effectiveness of the proposed mesh adaptation method was demonstrated: First, in the proposed method, the space mesh could be modified depending on the motion of the object mesh, and the adaptation was done only around the moving object mesh. Second, it was shown that a conformal tetrahedral mesh without any inverted and degenerated elements could be generated in each motion step even if the moving object mesh contacts with the other object mesh. Finally, it was shown that the proposed method can be used for generation of conformal tetrahedral meshes of moving assembly models from a set of meshes which are generated individually.

By the proposed tetrahedral mesh adaptation method, even if the object meshes contact with each other, the conformal tetrahedral mesh of each motion step with moderate element shape qualities can be generated while avoiding drastic increase of elements on the contact region. In addition, the processing time of each motion step was at most 5s for conformal tetrahedral mesh including about 160k tetrahedra in the experiments.

Chapter 4: Application of deformation and adaptation methods to hexahedral meshes based on Tet-Hex conversion.

In order to realize certain geometric operations of hexahedral meshes, the proposed dimension-driven tetrahedral mesh deformation method and tetrahedral mesh adaptation method were combined with an indirect hexahedral mesh generation method based on the conversion between tetrahedra and hexahedra (Tet-Hex conversion) [Meshkat 2000]. In Chapter 4, the overview of the combination of the proposed methods with the Tet-Hex conversion was shown after the discussion of existing methods. In addition, in order to convert hexahedral meshes into tetrahedral meshes based on the concept of [Meshkat 2000], six decomposition types for a hexahedron were described, and after that the method proposed in [Meshkat 2000] were mentioned. In their method, five or six tetrahedra were converted into a hexahedron based of an extended RF-graph and a greedy algorithm.

Through the experiments, it was shown that hexahedral meshes after deformation and object motion can be generated by the combination of the proposed methods with the Tet-Hex conversion in [Meshkat 2000]. However, it is difficult to obtain high quality hexahedral meshes or all-hex meshes from deformed or adapted tetrahedral meshes by only performing the Tet-Hex conversion.

5.2 Future work

Although the proposed method in this thesis have many advantages described above, some problems are also remained. In this section, the problems of the proposed methods and some strategies for the solution of the problems are discussed.

Chapter 2: Dimension-driven tetrahedral mesh deformation for parameter survey.

The proposed dimension-driven tetrahedral mesh deformation method consists of three functions, and each function has the following problems.

- The mesh segmentation function may take a long processing time for the extraction of conical surface regions and the calculation of surface parameters. In the former problem, a random selection of initial seed regions for a conical surface fitting based on RANSAC [Schnabel 2007] took a lot of time. In order to solve this problem, adding some constraints such as the distance between the initial seed regions to the selection may be effective. In the latter problems, in order to obtain the initial parameters of the LM method, the surface fitting based on RANSAC was repeated until the LM method converged or the iteration time reached a threshold. For solving this problem, the threshold for the convergence of LM method may need to be adjusted appropriately. Moreover, the way of the calculation of the initial parameters should be changed from the RANSAC to other surface fitting technique.
- The dimension-driven shape deformation method may generate inverted elements which cannot be removed the quality improvement function. In order to avoid inverted elements, the effectiveness of dividing the deformation into some small steps was shown an experiment. Therefore, a function which divides the deformation into some small steps automatically will be useful for more stable deformation. In addition, changing feature parameters must be performed without self-intersections and degenerations of surface regions because the deformation method does not deal with topological changes of product shapes. Therefore, in order to change several feature parameters more freely, some techniques where surface regions are created or removed according to the deformation are needed.
- The quality improvement function took long processing time if many vertices need to be inserted to the deformed mesh for recovering the original mesh density. Therefore, some speeding-up techniques such as a parallel processing of the vertex insertion will be needed for more efficiency.

Chapter 3: Tetrahedral mesh adaptation for efficient finite element analysis of assembly models

Although benchmarks of some parameters such as a distance threshold for extraction of deformed regions and a displacement of a moving object mesh in each motion step are needed for the proposed tetrahedral mesh adaptation method, the verification for appropriate parameters was not performed enough in this thesis. Therefore, the verification of the influence of each parameter is needed in the future work.

In addition, it is not theoretically guaranteed that the minimum stretch which is an element shape quality measure become larger than the recommended lower limit of stretch in the mesh adaptation process and the quality improvement process. For more stable mesh adaptation, other techniques

for improvement of the minimum stretch may need to be introduced into the proposed method.

Moreover, the contact region adaptation has room for improvement. In current algorithm, the corresponding connectivity of surface triangular mesh on the contact region includes a random manner and does not use any criteria. Therefore, it can be made more efficient by using methods for obtaining an edge flipping sequence for triangular mesh matching [Hanke 1996, Espinas 2013].

Chapter 4: Application of deformation and adaptation methods to hexahedral meshes based on Tet-Hex conversion.

In this thesis, only validation of the combination of two proposed methods with an indirect hexahedral mesh generation method based on the conversion between tetrahedra and hexahedra [Meshkat 2000] is performed. In the experiment, many tetrahedra were remained in the output hexahedral meshes. In order to solve this problem, the condition where a set of tetrahedra can be converted into hexahedra easily should be taken into account and created in the quality improvement. For example, each tetrahedron should be a trirectangular tetrahedron whose three face angles are right angles at one vertex, vertices should be aligned along three orthogonal axes, the shape of each tetrahedron should be the same as that of tetrahedra obtained by decomposition Type C^+ for a cube, or the number of adjacent tetrahedra of each inner edge should be four or six. In addition, the extended conversion method in [Bottella 2016] will be more effective, and combination of other conversion methods [Owen 2000, Yamakawa 2003] with [Meshkat 2000] can be useful for solving the problem.

In addition, for more efficiency, extraction of a small region for the conversion between tetrahedral meshes and hexahedral meshes and processing to only the small region may be useful.

References

- [Alauzet 2014] F. Alauzet: “A changing-topology moving mesh technique for large displacements,” *Engineering with Computers*, 30(2), pp. 175-200, 2014.
- [Barral 2014] N. Barral, E. Luke, and F. Alauzet: “Two mesh deformation methods coupled with a changing-connectivity moving mesh method for CFD applications,” *Proc. of the 23rd International Meshing Roundtable*, pp. 213-227, 2014.
- [Bhowmick 2010] S. Bhowmick and S. M. Shontz: “Towards high-quality, untangled meshes via a force-directed graph embedding approach,” *Procedia Computer Science*, 1(1), pp. 357-366, 2010.
- [Blacker 1993] T.D. Blacker and R.J. Meyers: “Seams and wedges in plastering: A 3-D hexahedral mesh generation algorithm,” *Engineering with Computers*, 9(2), pp. 83-93, 1993.
- [Bottella 2016] A. Bottella, B. Lévy, and G. Caumon: “Indirect unstructured hex-dominant mesh generation using tetrahedra recombination,” *Computational Geosciences*, 20(3), pp.437-451, 2016.
- [CATIA V5R18] CATIA V5R18, Dassault Systems,
<http://www.3ds.com/products-services/catia/>
- [Chen 2004a] L. Chen and J. Xu: “Optimal Delaunay triangulation,” *Journal of Computational Mathematics*, 22(2), pp. 299-308, 2004.
- [Chen 2004b] L. Chen: “Mesh smoothing schemes based on optimal Delaunay triangulations,” *Proc. of the 13th International Meshing Roundtable*, pp. 109-120, 2004.
- [Chen 2011] L. Chen and M. Holst: “Efficient mesh optimization schemes based on Optimal Delaunay Triangulations,” *Computer Methods in Applied Mechanics and Engineering*, 200(9-12), pp. 967-984, 2011.
- [Chen 2015] J. Chen, S. Li, J. Zheng, and Y. Zheng: “Parallel local remeshing for moving body applications,” *Proc. of the 24th International Meshing Roundtable (Session 7A Research Notes III: Parallel Meshing 7A-3)*, 2015.
- [Chen 2016] J. Chen, S. Gao, and H. Zhu: “An improved hexahedral mesh matching algorithm,” *Engineering with Computers* 32(2), pp. 207-230, 2016.
- [Cheng 2000] S.W. Cheng, T.K. Dey, H. Edelsbrunner, M.A. Facello, and S.H. Teng: “Sliver Exudation,” *J. ACM*, 47, pp. 883-904, 2000.
- [Chew 1997] L.P. Chew: “Guaranteed quality Delaunay meshing in 3d (short version),”

- Proc. of the 13th annual Symposium on Computational Geometry, pp. 391-393, 1997.
- [Cohen-Steiner 2004] D. Cohen-Steiner, P. Alliez, and M. Desbrun: "Variational shape approximation," ACM Transactions on Graphics (TOG) - Proc. of ACM SIGGRAPH 2004, 23(3), pp. 905-914, 2004.
- [Compère 2010] G. Compère, J.F. Remacle, J. Jansson, and J. Hoffman: "A mesh adaptation framework for dealing with large deforming meshes," Numerical Methods in Engineering, 82(7), pp. 843-867, 2010.
- [Du 2003] Q. Du and D. Wang: "Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations," International Journal for Numerical Methods in Engineering, 56(9), pp. 1355-1373, 2003.
- [Dobrzynski 2008] C. Dobrzynski and P. Frey: "Anisotropic Delaunay mesh adaptation for unsteady simulation," Proc. of the 17th International Meshing Roundtable, pp. 177-194, 2008.
- [Edelsbrunner 2002] H. Edelsbrunner and D. Guoy: "An experimental study of sliver exudation," Engineering with Computers, 18(3), pp. 229-240, 2002.
- [Eigen 2012] Eigen 3.1.2, http://eigen.tuxfamily.org/index.php?title=Main_Page, last updated on 30th November 2016.
- [Espinas 2013] J. Espinas, R. Chaine, and P.M. Gandoin: "Practical reduction of edge flip sequences in two-dimensional triangulations," arXiv preprint arXiv:1310.2586, 2013.
- [Faraj 2016] N. Faraj, J.M. Thiery, and T. Boubekeur: "Multi-material adaptive volume remesher," Computers and Graphics, 58, pp. 150-160, 2016.
- [Femap] Femap, NST Co., Ltd., <http://www.cae-nst.co.jp/products/femap/index.html>
- [Field 1988] D.A. Field: "Laplacian smoothing and Delaunay triangulation," Communication in Applied Numerical Methods, 4(6), pp. 709-712, 1988.
- [Floater 2003] M.S. Floater: "Mean value coordinates," Computer Aided Geometric Design, 20(1), pp. 19-27, 2003.
- [Floater 2005] M.S. Floater, G. Kós, and M. Reimers: "Mean value coordinates in 3d," Computer Aided Geometric Design, 22(7), pp. 623-631, 2005.
- [Fruchterman 1991] T. Fruchterman and E. Reingold: "Graph drawing by force-directed placement," Software-Practice and Experience, 21(11) pp.1129-1164, 1991.
- [Fujimoto 1986] A. Fujimoto, T. Tanaka, and K. Iwata: "ARTS: Accelerated Ray Tracing System," IEEE - CGA, 6(4), pp.16-26, 1986.
- [Gao 2012] Z. Gao, Z. Yu, and M. Holst: "Quality tetrahedral mesh smoothing via boundary-optimized Delaunay triangulation," Computer Aided Geometric Design, 29(9), pp. 707-721, 2012.
- [Geng 2010] C. Geng, H. Suzuki, D.M. Yan, T. Michikawa, Y. Sato, M. Hashima, and E. Ohta: "A thin-plate CAD mesh model splitting approach based on fitting primitives," EG UK Theory and Practice of Computer Graphics, pp. 45-50,

-
- 2010.
- [Geuzaine 2009] C. Geuzaine and J.F. Remacle: “Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities,” *International Journal for Numerical Methods in Engineering* 79(11), pp. 1309-1331, 2009.
- [Guo 2016] J. Guo, D.M. Yan, L. Chen, X. Zhang, O. Deussen, and P. Wonka: “Tetrahedral meshing via maximal Poisson-disk sampling,” *Geometric Modeling and Processing* 2016, 43, pp. 186-199, 2016.
- [Hadwiger 2005] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. Gross: “Realtime ray-casting and advanced shading of discrete isosurfaces,” *Computer Graphics Forum*, 24(3), pp. 303-312, 2005.
- [Hanke 1996] S. Hanke, T. Ottmann, and S. Schuierer: “The edge-flipping distance of triangulations,” *Journal of Universal Computer Science*, 2(8), pp. 570-579, 1996.
- [Houlin 2014] L. Houlin, L. Mingzhen, T. Minggao, D. Liang and R. Yun: “The research on tetrahedral mesh optimization algorithm of the double channel pump,” *Applied Mathematics & Information Sciences*, 8(3), pp. 1041-1047, 2014.
- [Howland 1930] R. C. J. Howland: “On the Stresses in the Neighbourhood of a Circular Hole in a Strip under Tension,” *Phil. Trans. Roy. Soc. A*, 229 (670-680), pp. 49-86, 1930.
- [Jibin 2012] Z. Jibin, L. Weijun, and X. Renbo: “A method of feature line extraction of triangle mesh surface model,” *Proc. of the IEEE International Conference of Information and Automation (ICIA)*, pp. 602-606, 2012.
- [Joe 1995] B. Joe: “Construction of three-dimensional improved-quality triangulations using local transformations,” *SIAM Journal on Scientific Computing*, 16 (6), pp.1292-1307.
- [Ju 2005] T. Ju, S. Schaefer, and J. Warren: “Mean value coordinates for closed triangular meshes,” *ACM Transactions on Graphics (TOG) - Proc. of ACM SIGGRAPH 2005*, 24(3), pp. 561-566 (2005).
- [Juntunen 2013] M. Juntunen and S. Korotov: “Conforming post-refinement of non-matching tetrahedral meshes,” *Proc. of the MASCOT13 IMACS/ISGG Workshop*, 2013. (to appear)
- [Kim 2009] H.S. Kim, H.K. Choi, and K.H. Lee: “Feature detection of triangular meshes based on tensor voting theory,” *Computer-Aided Design*, 41(1), pp. 47-58, 2009.
- [Knupp 2000] P.M. Knupp: “Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities Part II—a framework for volume mesh optimization and the condition number of the Jacobian matrix,” *International Journal for Numerical Methods in Engineering*, 48(8), pp. 1165-1185, 2000.

- [Knupp 2001] P.M. Knupp: "Hexahedral and tetrahedral mesh untangling," Engineering with Computers, 17(3), pp. 261-268, 2001.
- [Labelle 2006] F. Labelle: "Sliver removal by lattice refinement," Proc. of the twenty-second annual symposium on Computational geometry, pp. 347-356, 2006.
- [Li 2000] X. Li: "Sliver-free 3-dimensional Delaunay mesh generation," PhD Thesis, University of Illinois, 2000.
- [Li 2003] X. Li: "Mesh modification procedure for general 3-d non-manifold domains," Ph.D. Thesis, Renselear Polytechnic Institute, 2003.
- [Li 2013] Y. Li, Z. Jing, and W. Hailing: "Tetrahedron Model Reconstruction and Optimization," 2013 Seventh International Conference on Internet Computing for Engineering and Science. IEEE, pp. 20-22, 2013.
- [Lipman 2008] Y. Lipman, D. Levin, and D. Cohen-Or: "Green coordinates," ACM Transactions on Graphics (TOG) - Proc. of ACM SIGGRAPH 2008, 27(3), Article No. 78, 2008.
- [LISA 2012] LISA, Sonnenhof Holdings, <http://www.lisafea.com/index.html>, 2012.
- [Masuda 2008] H. Masuda and K. Ogawa: "Interactive deformation of 3D mesh models," Computer-Aided Design and Applications, 5(1-4), pp.47-57(2008).
- [Meshkat 2000] S. Meshkat and D. Talmor: "Generating a mixed mesh of hexahedra, pentahedra and tetrahedra from an underlying tetrahedral mesh," International Journal for Numerical Methods in Engineering, 49(12), pp.17-30, 2000.
- [Ogawa 2008] K. Ogawa and H. Masuda: "A constraint-based mesh deformation technique for assembly models," Transactions of the Japan Society of Mechanical Engineers, Series C, 74(739), pp.739-745, 2008.(in Japanese)
- [Onodera 2008] M. Onodera, I. Nishigaki, Y. Hiro, and C. Kongo: "Development of parametric mesh morphing technique," Transactions of the Japan Society of Mechanical Engineers, Series C, 74(742), pp.1594-1601, 2008.(in Japanese)
- [Onodera 2014] M. Onodera, Y. Hiro, and C. Kongo: "Development of elements quality distribution sustainable mesh quality improvement technique," Transactions of the Japan Society of Mechanical Engineers, Series C, 79(806), pp.3642-3651, 2014.(in Japanese)
- [Owen 2000] S.J. Owen and S. Saigal: "H-Morph: an indirect approach to advancing front hex meshing," Numerical Methods in Engineering, 49(1-2), pp. 289-312, 2000.
- [Samaniego 2015] C. Samaniego, G. Houzeaux, E. Samaniego, and M. Vázquez: "Parallel embedded boundary methods for fluid and rigid-body interaction," Computer Methods in Applied Mechanics and Engineering, 290(15), pp. 387-419, 2015.
- [Sastry 2012] S.P. Sastry, S.M. Shontz, S.A. Vavasis: "A log-barrier method for mesh

- quality improvement and untangling,” Proc. of 20th International Meshing Roundtable, pp. 329-346, 2012.
- [Sawai 2010] Y. Sawai and H. Aoyama: “Efficient design method based on evaluation using CAE system,” Journal of Advanced Mechanical Design, Systems, and Manufacturing, 4(5: Special Issue on Advanced Manufacturing Technology (LEM21)), pp. 827-837, 2010.
- [Schnabel 2007] R. Schnabel, R. Wahl, and W. Klein: “Efficient RANSAC for point-cloud shape detection connectivity oblivious fashion,” Computer Graphics Forum, 26(2), pp. 214-226, 2007.
- [Schöberl 1997] J. Schöberl: “NETGEN An advancing front 2D/3D-mesh generator based on abstract rules,” Computing and Visualization in Science, 1(1), pp. 41-52, 1997.
- [Sethian 2001] J.A. Sethian: “Evolution, implementation, and application of level set and fast marching methods for advancing fronts,” Computational Physics, 169(2), pp. 503-555, 2001.
- [Shakarji 1998] C.M. Shakarji: “Least-squares fitting algorithms of the NIST algorithm testing system,” Journal of Research of the National Institute of Standards and Technology, 103(6), pp. 633-641, 1998.
- [Shontz 2012] S. Shontz and S. Vavasis, “A robust solution procedure for hyperelastic solids with large boundary deformation,” Engineering with Computers, 28(2), pp. 135-147, 2012.
- [Shu 2011] Z.Y. Shu, G.Z. Wang, and L.Z. Fan: “Fast mesh segmentation for CAD models,” Key Engineering Materials, 460-461, pp. 780-785, 2011.
- [Si 2006] H. Si, TetGen, <http://wias-berlin.de/software/tetgen/>
- [Sieger 2013] D. Sieger, S. Menzel, and M. Botsch: “High quality mesh morphing using triharmonic radial basis functions,” Proc. of the 21st International Meshing Roundtable, pp. 1-15, 2013.
- [Sieger 2014] D. Sieger, S. Menzel, and M. Botsch: “Constrained space deformation for design optimization,” Procedia Engineering, 23rd International Meshing Roundtable (IMR23), 82, pp. 114-126, 2014.
- [Silva 2010] L.F. Silva: “Merging meshes using dynamic regular triangulation,” Master Thesis, Universidade Federal do Rio Grande do Sul, 2010.
- [Silva 2014] L.F. Silva, L.F. Scheidegger, T. Etienne, J.L.D. Comba, L.G. Nonato, and C.T. Silva: “A weighted Delaunay triangulation framework for merging triangulation in connectivity oblivious fashion,” Computer Graphics Forum, 33(6), pp. 18-30, 2014.
- [Song 2013] S. Song, M. Wan, S. Wang, D. Wang, and Z. Zou: “Robust and quality boundary constrained tetrahedral mesh generation,” Communications in Computational Physics, 14(5), pp. 1304-1321, 2013.
- [Staten 2008] M.L. Staten, J.F. Shepherd, and K. Shimada: “Mesh matching – creating

- conforming interfaces between hexahedral meshes,” Proc. of the 17th International Meshing Roundtable, pp. 467-484, 2008.
- [Staten 2011] M.L. Staten, S.J. Owen, S.M. Shontz, A.G. Salinger, and T.S. Coffey: “A comparison of mesh morphing methods for 3D shape optimization,” Proc. of the 20th International Meshing Roundtable, pp. 293-311, 2011.
- [Stephenson 1992] M.B. Stephenson, S.A. Canann, and T.D. Blacker: “Plastering: A new approach to automated, 3D hexahedral mesh generation. Progress report 1.” Sandia National Laboratories, Report SAND89-2192, 1992.
- [Sun 2016] S. Sun, S. Lv, Y. Yuan, and M. Yuan: “Mesh deformation method based on mean value coordinates interpolation,” *Acta Mechanica Solida Sinica*, 29(1), pp. 1-12, 2016.
- [Takano 2010] Y. Takano, H. Date, and M. Onosato: “Dimension-driven deformation and untangling of tetrahedral meshes,” Proc. of Asian Conference on Design and Digital Engineering 2010 (ACDDE 2010), pp. 483-491, 2010.
- [Tautges 1996] T.J. Tautges, T. Blacker, and S.A. Mitchell: “The whisker weaving algorithm: A connectivitybased method for constructing all-hexahedral finite element meshes,” *Numerical Methods in Engineering*, 39(19), pp. 3327-3349, 1996.
- [Tournois 2009a] J. Tournois, C. Wormser, P. Alliez, and M. Desburn: “Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation,” *ACM Transactions on Graphics (TOG) - Proc. of ACM SIGGRAPH 2009*, 28(3), Article No. 75, 2009.
- [Tournois 2009b] J. Tournois, R. Srinivasan, and P. Alliez: “Perturbing slivers in 3D Delaunay meshes,” Proc. of the 18th International Meshing Roundtable, pp. 157-173, 2009.
- [Tsuchie 2014] S. Tsuchie and M. Higashi: “Extraction of surface-feature lines on meshes using normal tensor framework,” *Computer-Aided Design and Applications*, 11(2), pp. 172-181, 2014.
- [Wang 2012] X. Wang, J.J. Cao, X.P. Liu, X.Q. Shi, and Y.Z. Sun: “Feature detection of triangular meshes via neighbor supporting,” *Journal of Zhejiang University Science C*, 13(6), pp. 440-451, 2012.
- [Wang 2013] Y. Wang, N. Qin, G. Carnie, and S. Shahpar: “Zipper layer method for linking two dissimilar structured meshes,” *Computational Physics*, 255, pp. 130-148, 2013.
- [Xian 2009] C. Xian, S. Gao, H. Lin, and D. Xiao: “FEA-mesh editing with feature constrained,” Proc. of 11th International Conference on Computer-Aided Design and Computer Graphics, CAD/Graphics 2009, pp. 340-343, 2009.
- [Xiao 2011] D. Xiao, H. Lin, C. Xian, and S. Gao: “CAD mesh model segmentation by clustering,” *Computers and Graphics*, 35(3), pp. 685-691, 2011.
- [Yamakawa 2000] S. Yamakawa and K. Shimada: “High Quality Anisotropic Tetrahedral Mesh

- Generation Via Ellipsoidal Bubble Packing,” Proc. of the 9th International Meshing Roundtable, pp. 263-273, 2000.
- [Yamakawa 2003] S. Yamakawa and K. Shimada: “Fully-automated hex-dominant mesh generation with directionality control via packing rectangular solid cells,” Numerical Methods in Engineering, 57(15), pp. 2099-2129, 2003.
- [Yan 2012] D.M. Yan and W. Wang: “Variational mesh segmentation via quadric surface fitting,” Computer-Aided Design, 44(11), pp. 1072-1082, 2012.
- [Zhihong 2011] M. Zhihong, K. Lee, and C. Guo: “Interactive feature extraction on 3D meshes,” Computer-Aided Design and Applications, 8(5) pp. 785-793, 2011.

Acknowledgement

Time passed quickly, my nine years university life comes to an end. I had many experiences in this nine years and I have been supported by many people.

First of all, I would like to express my sincere gratitude to my supervisor Associate Professor Hiroaki Date who gave insightful and incisive comments through face-to-face discussions almost every week. I received many invaluable and technical advices for my research from him each time I consulted with him about my research. In addition, he checked my papers or presentations for journals and international conferences thoroughly, and guided me appropriately using his experiences and knowledges. Without his guidance, this thesis also would not have materialized.

I would like to thank Professor Satoshi Kanai, he also gave a lot of invaluable comments to me and guided the way of presentations. He also supported my research with his intelligence and gave encouragement when I reached an impasse or I was trouble about my research. Moreover, I received opportunities for visiting other countries and laboratories.

I would like to appreciate Professor Masahiko Onosato for judging my doctoral thesis and providing a lot of valuable advises and incisive comments.

In addition, I would like to thank past and present members of Digital Geometry Processing Laboratory (System Design & Modeling Laboratory). Specially, I appreciate Mr. Shinsuke Hatakeda who is an undergraduate student at that time helping me out with the research of hexahedral mesh modeling by surveying papers and cording a base program of the Tet-Hex conversion. In addition, I would like to thank Midori Takahashi. She supported my overseas internship, helped us in the laboratory life, and proceeded some office processing such as preparation of business trips.

I also thank my family, without their constant encouragement and support, I could not have finished my PhD research.

Research Assistant project in Graduate School of Information Science and Technology, Hokkaido University, which provide me financial supports. In addition, this work was supported by JSPD KAKENHI Grant Number 15J01584

Publication List

Journal Papers:

- (1) Hiroki Maehama, Hiroaki Date, and Satoshi Kanai: “Quality Improvement of Deformed Mesh Models for FEA by Density and Geometry-recovering Phased ODT Smoothing,” Computer-Aided Design and Applications. Vol. 11, No. 6, pp. 622-631 (2014).
- (2) Hiroki Maehama, Hiroaki Date, and Satoshi Kanai: “Dimension-driven Deformation of Tetrahedral Meshes of Mechanical Parts for Parameter Survey Analysis,” Journal of the Japan Society for Precision Engineering, Vol. 81, No. 4, pp. 368-375 (2015).

Peer-reviewed Proceedings of International Conference:

- (3) Hiroki Maehama, Hiroaki Date, and Satoshi Kanai: “Quality Improvement of Finite Element Mesh Models Modified by Mesh Deformation,” Proc. of the 2012 Asian Conference on Design and Digital Engineering (ACDDE 2012), Niseko, Japan, December 6-8th, 10064 (2012).
- (4) Hiroki Maehama, Hiroaki Date, and Satoshi Kanai: “Quality Improvement of Deformed Mesh Models for FEA by Density and Geometry-recovering Phased ODT Smoothing,” Proc. of the CAD’13 – the 10th Annual International CAD Conference and Exhibition, Bergamo, Italy, June 17-20th, 89 (2013).
- (5) Hiroki Maehama, Hiroaki Date, and Satoshi Kanai: “Segmentation, Dimension-Driven Deformation, and Quality Improvement of Tetrahedral Mesh Models for Finite Element Analysis,” Proc. of the 15th International Conference on Precision Engineering (ICPE2014), Kanazawa, Japan, July 23-25th, D30 (2014).

Conference:

- (6) Hiroki Maehama, Hiroaki Date, and Satoshi Kanai: “Quality Improvement of Dimension-Driven Deformed Mesh Models by Density and Geometry-recovering Phased ODT Smoothing,” Proceedings of 2012 Japan Society for Precision Engineering Hokkaido branch meeting, pp.41-42, Muroran, September, 2012 (in Japanese).
- (7) Hiroki Maehama, Hiroaki Date, and Satoshi Kanai: “Quality Improvement of Dimension Driven Deformed Mesh Models by Phased Optimal Delaunay Triangulation Smoothing,” Proceedings of 2012 Japan Society for Precision Engineering Fall Conference, pp.259-260, Kita-KyuSyu, September, 2012 (in Japanese).
- (8) Hiroki Maehama, Hiroaki Date, and Satoshi Kanai: “Quality Improvement of Dimension Driven Deformed Tetrahedral Mesh Models by Phased Optimal Delaunay Triangulation

- Smoothing - Modification of degenerated or inverted elements by local topological operations -,” Proceedings of 2012 Japan Society for Precision Engineering Spring Conference, pp.847-848, Tokyo, March, 2013 (in Japanese).
- (9) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Finite Element Mesh Segmentation based on Normal Tensor and Region Growing,” Proceedings of 2013 Japan Society for Precision Engineering Hokkaido branch meeting, C-18, Kitami, August, 2013 (in Japanese).
- (10) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Quality Improvement of Dimension Driven Deformed Tetrahedral Mesh Models by Phased Optimal Delaunay Triangulation Smoothing - FEM Mesh Segmentation and Quality Improvement of arbitrary surface mesh -,” Proceedings of 2013 Japan Society for Precision Engineering Fall Conference, pp.845-846, Suita, September, 2013 (in Japanese).
- (11) Hiroki Maehama, Hiroaki Date, and Satoshi Kanai: “Dimension-Driven Deformation of Tetrahedral Mesh Models for FEA by Surface Information and Space Embedding,” Proceedings of 2014 Japan Society for Precision Engineering Spring Conference, pp.803-804, Tokyo, March, 2014 (in Japanese).
- (12) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Speeding up dimension-driven mesh deformation by localization of a deformation process,” Proceedings of 2014 Japan Society for Precision Engineering Hokkaido branch meeting, pp.21-22, Sapporo, August, 2014 (in Japanese).
- (13) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Dimension-Driven Deformation of FEM Tetrahedral Mesh Models for Efficient CAE - Speeding up mesh deformation and quality improvement by localization of a deformed region -,” Proceedings of 2014 Japan Society for Precision Engineering Fall Conference, pp.845-846, Tottori, September, 2014 (in Japanese).
- (14) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Efficient Generation of FEM Tetrahedral Mesh Models for Moving Objects and Space,” Proceedings of 2015 Japan Society for Precision Engineering Spring Conference, pp.433-434, Tokyo, March, 2015 (in Japanese).
- (15) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Mesh Adaptation for Efficient High Quality Tetrahedral Mesh Generation for Object Motion with Contact,” Proceedings of 2015 Japan Society for Precision Engineering Hokkaido branch meeting, pp.85-86, Chitose, August, 2015 (in Japanese).
- (16) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Efficient Generation of FEM Tetrahedral Mesh Models for Moving Objects and Space - Mesh adaptation for object motion with contact -,” Proceedings of 2015 Japan Society for Precision Engineering Fall Conference, pp.37-38, Sendai, September, 2015 (in Japanese).
- (17) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “High Quality Tetrahedral Mesh Generation for Efficient Analysis of Assembly Models (Tetrahedral Mesh Adaptation for Object Motion),” Proceedings of 2015 Japan Society for Design Engineering Fall Conference, pp.37-40, Sapporo, October, 2015 (in Japanese).
- (18) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Mesh Adaptation for Object Motion with

- Contact,” the 24th International Meshing Roundtable, Austin, TX, USA, October 11-14th, Technical Posters 4 (2015).
- (19) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Efficient Generation of FEM Tetrahedral Mesh Models for Assembly Models,” Proceedings of 2016 Japan Society for Precision Engineering Spring Conference, pp.205-206, Noda, March, 2016 (in Japanese).
- (20) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Finite Element Tetrahedral Mesh Generation for Object Motion with Contact - Quality Improvement near the Contact Region-,” Proceedings of 2016 Japan Society for Precision Engineering Hokkaido branch meeting, Asahikawa, August, 2016, (in Japanese).
- (21) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Conformal Mesh Generation for Assembly Models from Tetrahedral Meshes of Multiple Parts,” Proceedings of 2016 Japan Society for Design Engineering Hokkaido meeting, pp.13-16, Sapporo, September, 2016 (in Japanese).
- (22) Hiroki Maehama, Hiroaki Date, Satoshi Kanai: “Efficient Conformal Tetrahedral Finite Element Mesh Generation for Moving Objects with Contact by Mesh Adaptation,” the 2016 Asian Conference on Design and Digital Engineering (ACDDE 2016), Jeju, Korea, October 25-28th, 13-5 (2016).