Deformations with Non-Linear Constraints

TANG, Wing Shing

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Mechanical and Automation Engineering

The Chinese University of Hong Kong
June 2013

DECLARATION

I hereby declare that the Master of Philosophy thesis titled “Deformations with Non-Linear Constraints” represents my own work. I also declare that the work reported in this thesis has not been previously included in a thesis, dissertation or report submitted to this University or to any other institutions for a degree, diploma, or other qualification.

Tang Wing Shing

2013

ACKNOWLEDGMENTS

My sincere appreciation goes to Dr. Hui Kin-Chuen, my project supervisor, who guided my research towards the right direction and provided full academic and spiritual supports. He also gives me valuable suggestions and improvements on my work.

I would like to say thank you to my parents, who have been supporting me since I was born. Without their consideration and support, it would be impossible for me to take up the research work. Also, I would like to thank Miss Chan Hau Yan, who always cheers me up every time I am depressed.

This work is partially supported by a grant from the University Grant Council of the Hong Kong Special Administrative Region (No. 412508) and a Direct Grant (No. 2050492) from the Chinese University of Hong Kong.

ABSTRACT

To retain geometric features in the deformation of a parametric and feature-based model is a new challenge for CAD modeling. This thesis presents a constraints based deformation framework. This framework combines the advantage of free-form modeling with feature based modeling, and allows engineering design to be performed in a free-form manner.

The proposed method can be divided into three major steps. An object is deformed by common deformation techniques such as FFD and axial deformation. Parametric features are divided into systems of primitive constraints based on user specification. The targeting features are reconstructed by the use of incremental optimization technique.

An incremental constrained deformation is introduced. It is used to provide hints for the optimization. The optimization is to minimize the changes in the deformed model subjected to all the provided constraints. For a structural constraint specified with a group of constraints, it would be better to use a reference datum for all its component constraints. We show numerous results of constraints retained models using our framework.

摘要

於參數化和特徵模型的變形中保持幾何特徵是 CAD 建模中一項新的挑戰。這篇論文提出了一個以限制為基礎去進行變形的系統。此系統結合了自由曲面和特徵模型建模的好處，而且容許更自由的工程設計。

本方法可分為三個主要步驟。以常用的變形技術去改變一個模型的形狀，包括自由變形及軸向變形，然後參數特徵會根據用戶的要求去分拆為一系列基本的限制，最後目標特徵將會以逐步增量的優化技術去重建。

這篇論文提出了一個逐步增量的方法為優化中提供導引。這個優化是於維持所有提供的限制下盡量減少變形後模型的改變。另外，於一組的限制中以一個基准為參考，能使本系統更有效的運行。最後，我們也會展示一些使用本系統以限制為基礎去進行變形的結果。

Thesis/Assessment Committee

Professor Wang, Changling Charlie (Chair)

Professor Hui, Kin-chuen (Thesis Supervisor)

Professor Du, Ruxu (Committee Member)

Professor Yu, Kai-ming (External Examiner)

CONTENTS

1. INTRODUCTION

3D object are normally defined using a discrete set of parameters such as the vertices of a mesh, control points of parametric curves and surfaces. These parameters can also be used as handles for the interactive manipulation of the underlying shape. In computer-aided design (CAD), a feature usually refers to a region of a part with some geometric or topological properties. Feature based modeling systems describe object models with a number of parameters. Feature models are based on a dual representation scheme which includes a parametric representation and a geometric representation. The parametric representation describes the relation between the parameters and the geometry of the features. It is usually a CSG representation with geometric constraints. The geometric representation is a boundary representation (B-rep) which is generated by solving constraints in the parametric model.

On the other hand, free form deformation has various applications in modeling, animation, rendering or simulation. Current deformation techniques usually perform deformation on vertices, curves or surfaces of the object. These deformations will not maintain constraints specified in the modeling process. This may result in undesirable shapes when the deformation is to be applied to some parts of the object only. Previous research mainly focused on how to preserve some simple geometric feature (e.g. circular edge) of the object after deformation. The problem of maintaining the relation between geometric constraints and deformations is not well addressed.

1.1 Aims and Objectives

The objective of this research is to develop a system that allows geometric features to be retained after a deformation. In the proposed system, constraints can be maintained in a model deformed by different kinds of deformation techniques. It can be used to retain some functional features after deformations have been applied. For example, features of a component that have to match corresponding features of another component in an assembly may have chances to loss their functionalities in a deformation. The proposed system can be used to retain matching of the features in the assembly when the shapes of the components are adjusted in the design process.

The topology of the model will not be changed in the constraint based deformation. This means the number of vertices will not be changed in a deformation. In the proposed approach, all provided constraints are assumed to be valid, and there is no conflict between constraints. Since constraints that specified by users may consist of non-linear constraints, nonlinear optimization is used in the proposed system.

Moreover, in order to maintain angle constraints that exist in geometric feature of the model, dot product of two normalized vectors are always included in our constraint system. However, this kind of constraint is non-convex in nature. If there are non-convex constraints existed in a nonlinear optimization, the whole problem will become non-convex, and a non-convex optimization technique is required to solve it.

There are three steps in the proposed constraint based deformation technique. Firstly, an object is deformed by using some common deformation techniques such as FFD and axial deformation. This deformed model is called the *transitional model* and is used as the target shape in our optimization. Secondly, parametric features are grouped into systems of primitive constraints based on user specification. Finally, parametric features are reconstructed by the use of optimization technique.

Nonlinear and non-convex optimization may not always provide an optimal solution due to the algorithm is stuck at a locally infeasible point. The optimization

process and the formulation of the problem may have to be adjusted to assist the optimization system to find a solution and to reduce unpredictable errors.

1.2 Report Organization

The aim of this research is to establish a system that can preserve certain engineering feature constraints in a free-form deformation. The work reported in this thesis focuses on the development of the method concerned. There are two major issues addressed in this thesis. One issue is the formulation of the constraints for engineering features. The other one is the incremental optimization method. The content of the thesis are organized as follow:

Chapter 2 gives related background information on mesh editing techniques and the mathematical optimization method. A review on existing detail preserving techniques is presented. A review on linear and nonlinear optimization method is also included.

Chapter 3 describes the formulation of constraints. For every feature in parametric modeling, a set of constraints is required to maintain their feature characteristics. Most of these feature requirements can be classified into primitive constraints of points, lines or triangles. And constraints with references can be used to improve the performance of the system.

Chapter 4 describes the overall optimization process that implemented in this system. First it provides the objective function and the overall formulation of the optimization problem. And then it introduces an incremental optimization method to solve the problem when the optimization failed to find an optimal solution. When the optimization fails, the system will perform interpolation between the initial guess and a transitional model, and then generates an interpolated model for the next optimization.

Chapter 5 provides the results of the system with different applications. It provided some examples that maintain individual feature, pattern of features and relationship between features. Some implementation issues are also discussed in this chapter.

Chapter 6 discusses the effect of constraints with a common reference, different level of model detail and incremental method with different experiments. And some comparisons between our method and other deformation methods are also included.

Chapter 7 concludes the content of this thesis and discusses potential areas for further work and development.

2. BACKGROUND AND LITERATURE REVIEW

When applying deformation on the model, the mesh deformation techniques are required. Therefore in this chapter, the background information on mesh deformation techniques will be presented. Besides, mathematical optimization is required to retain the constraints on model after deformation, so a review on the mathematical optimization methods will also be presented. Linear and nonlinear optimization techniques are also included. Moreover, a review on existing detail preserving techniques is presented in order to find out the effectiveness of these algorithms and to see if there are rooms of improvement.

2.1 Mesh Editing Techniques

2.1.1 Mesh Deformation Techniques

Most existing CAD/CAM packages provide functions for constructing objects with feature based modeling technique [1]. For instance, different features require different techniques for their machining process [2][3]. Surface based mesh deformation methods have been widely used in mesh editing and animation. Space deformations (also called free-form deformations) are very popular in computer graphics. A space is defined with a lattice of control points. Deformation of the lattice points induces deformation to the mesh vertices of an object embedded in the space defined with the lattice [4][5][6]. Space deformation is easy to be implemented, and is highly efficient. In general, the computation cost of a deformation depends mainly on the complexity of the control lattice, but not the mesh vertices. However, FFD may deform object shapes in an unintended way, e.g. circular holes may be deformed into elliptical holes.

A recent related work used cages to control shape deformation [7]. Cage is an offset of the input mesh. Different coordinate functions are used to provide deformation on the entire space, including radial-basis function [8], mean-value coordinates [9], volume-preserving warps [10][11], locally rigid transformations [12] [13][14], harmonic coordinates [15] and Green coordinates [16]. However, they tend to treat the editing mesh as a homogenous surface such that high-level structure and features of the editing mesh cannot be retained.

There are also researches focusing on direct surface manipulation while preserving their geometric surface details. Several surface based approaches allow manual specification of varying surface stiffness [17][18], which allow some parts of a surface to behave more rigidly than others. However, no high level relationships can be specified, which makes these techniques difficult to use in engineering.

A more restricted space deformation was described for axis-aligned non-homogeneous scaling of structured man-made models [19]. By the measure of surface vulnerability to non-uniform scaling of the model, it classifies sensitive object features into two components, slippage and normal curvature. Slippage measures whether a local region on a surface remains on the surface after the transformation is applied. Normal curvature is used to distinguish between different degrees of vulnerability. The space deformation protects certain parts of the space that undergo similar transformation. Wang developed another content-aware mesh scaling technique [20]. It does not require an auxiliary regular grid, and directly deforms the mesh model according to its local sensitivity to geometric scaling. However, these content-aware mesh resizing techniques can only be applied to scaling, and cannot retain system of relationship specified on the model. For models composed of various types of joints, joint aware deformation, limb rigidity, local volume preservation and other physical constraints have been proposed [21-27].

2.1.2 Detail Preserving Techniques

Welch and Witkin introduced constraints in variational surface modeling and solved the constraint problem using Lagrange multipliers [28]. It presented the method to use B-spline control-points to modify surface shape and satisfying the user supplied geometric constraints. The constraints are grouped in two classes. The first class is called the finite-dimensional constraints that control surface shape at discrete points. Another class is called the transfinite constraints that control the surface shape along embedded curves or sub-regions of the surface. The research focused on linear constraint that can be applied on the surface.

Masuda et al. extended this method for form feature modeling [29]. It is a discrete framework for preserving the shapes of form-features using hard constraints in interactive shape deformation. Hard constraints mean constraints are precisely

satisfied. It constrains motion of form-features using linear constraint. It includes constraints on positions and rotations. Their framework was then further extended for handling non-manifold meshes and disconnected meshes [30]. It allows users to smoothly deformed disconnected meshes by propagating the rotations and translations through disconnected vertices. However, the relationships between features are not considered in these approaches.

Cabral et al. [31] propose an approach for modeling architectural scenes by reshaping and combining existing textured modals. For geometry, preserving angles such as floor orientation or vertical walls is important. It proposes a more lenient way to handle features by constraining the angles, but it also allow the user to changing the length of individual segment.

Gal introduced an approach called iWIRES [32] based on the argument that man-made models can be distilled using a few special 1D wires and their mutual relations. A small number of wires are used to preserve the defining characteristics of the entire object. By the use of analyze-and-edit approach, it performs a lightweight analysis of the input shape to extract a descriptive set of wires, and constructs a system of relationship based on a set of properties common to typical engineering models, such as linearity, planarity, parallelism and symmetry. By the editing of wires, it allows a wires-driven deformation on the model while maintaining the system of relationship of the wires. It helps the users to retain the original design intent and object characteristics of the model after deformation.

However, this wires-driven deformation only allows deformation to be applied through wires. It does not allow direct deformation on the object surface. Moreover, it only retains some primitive shape relationships between wires, and does not allow users to specify sets of complex constraints commonly found in engineering models.

Recently, direct modeling is adopted in some CAD/CAM software, such as Creo and Solid Edge. It allows users to design or modify engineering models without using history tree in transitional history-based modeling. It allows users to create and modify features by dragging geometries and dimension manipulators. However, it

cannot regenerate the features or the relationships between features if a free form deformation is applied to the model.

2.2 Optimization Techniques

2.2.1 Optimization Techniques

Optimization is a subject in applied mathematics with wide applications in engineering, science, finance, military and space technology. It is used for determining an optimal solution of a given problem. Although optimization is a classical problem, it did not become an independent subject until late 1940s, when G.B. Dantzig presented the simplex algorithm for linear programming [33]. The conjugate methods and quasi-Newton methods [34] were introduced in the 1950s. Nowadays, there are many optimization methods that can be used to solve complex and large scale optimization problems

The general form of an optimization problem:

$$\begin{aligned} &\min && f(x) \\ &\text{subject to} && x \in X \end{aligned} \tag{1}$$

where $x \in R^n$ is a decision variable, $f(x)$ is an objective function, $X \subset R^n$ is a constraint set or feasible region. If $X = R^n$, the optimization problem (1) is called an unconstrained optimization problem.

The constrained optimization problem can be written as follows:

$$\begin{aligned} &\min && f(x) \\ &\text{subject to} && b_i(x) = 0, i \in E \\ & && c_i(x) \geq 0, i \in I \end{aligned} \tag{2}$$

where E is the index set of equality constraints and I is the index set of inequality constraints, $b_i(x)$ and $c_i(x), (i = 0\ldots., m \in E \cup I)$ are constraint functions. When both objective function and constraint functions are linear functions, the problem is called linear programming. Otherwise, the problem is called nonlinear programming.

2.2.2 Linear Programming

2.2.2.1 Simplex Method

The simplex method is an algorithm for solving problems in linear programming. It operates on linear problems in standard form.

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & A x = b, x_i \geq 0 \end{aligned} \tag{3}$$

where $c, x \in R^n, b \in R^m$ with $x = (x_1, \ldots, x_n)$ are the variables of the problem, $c^T = (c_1, \ldots, c_n)$ are the coefficients of the objective function A which is a $p \times n$ matrix, and $b = (b_1, \ldots, b_p)$ are constants where $b_i \geq 0$. The feasible region of *x* is a convex polytope. If and only if the column vectors A_i are linearly independent for $x_i \neq 0$, $x = (x_1, \ldots, x_n)$ is an extreme point, and is referred to as a basic feasible solution.

It tests adjacent vertices of feasible set continuously, so at each vertex the objective function improves or remains unchanged. It takes around 2m to 3m number (where m is the number of equality constraints) of iterations to converge in expected polynomial time for some distributions of random inputs [35][36]. Moreover, its worst-case complexity is exponential [37].

2.2.2.2 Interior Point Method

A more efficient polynomial time algorithm was proposed by Karmarkar [38]. This algorithm achieves optimization by going through the interior of the inequality constraints in contrast to the simplex method that move along the boundary of the feasible region, and is called the interior point method. This approach is similar to the barrier function methods developed in 1960s which solve the problem by minimizing a set of unconstrained barrier functions. Interior point method received more attentions, because Karmarkar shown that it is much more effective than simplex method. Its complexity is polynomial for both average and worst case.

2.2.2.2.1 Primal-Dual Interior Point Method

Equation (3) can be called the primal problem. On the other hand, the dual problem can be formulated as

$$\max \quad b^T y \tag{4}$$

$$\text{subject to} \quad A^T y + s = c, s \geq 0$$

where $y \in R^n, s \in R^m$. Let $x^{\dot{c}}$ be a feasible solution to the primal problem and $(y^{\dot{c}}, s^{\dot{c}})$ be a feasible solution to dual problem. These solutions will be optimal to both problems if and only if they satisfy the complementary slackness conditions $x_i s_i = 0$ for $i = 1, \ldots, n$. The method try to move through the primal and dual solutions that improve the solutions, so that it satisfies the complementary slack conditions. In each iteration, the algorithm tries to find $x(\lambda), y(\lambda) \wedge s(\lambda)$ that satisfy the following condition:

$$Ax = b \tag{5}$$

$$A^T y + s = c \tag{6}$$

$$x_i s_i = \lambda, i = 1, .., n \tag{7}$$

$$x, s \geq 0 \tag{8}$$

In the optimization, the primal and dual equality constraints are both satisfied, and $x, s \geq 0$. Convergence can be found when the duality gap reaches zero or lies within the specified tolerances.

2.2.3 Nonlinear Programming

2.2.3.1 Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) was first proposed by Wilson [39] which can be used for solving nonlinear optimization problem. The basic idea of SQP is to model a nonlinear program at a given approximate solutions, say x_k , by a quadratic programming subproblem, and to use the solutions to construct a better approximation x_{k+1} . This process is iterated to generate a list of approximate solutions. Finally the algorithm tries to converge to an optimal solution. Consider the problem in (2), it can be derived by applying Newton's method to the corresponding optimality conditions. The Lagrangian for this problem is:

$$L(x, \lambda, \mu) = f(x) - \lambda^T b(x) - \mu^T c(x) \tag{9}$$

where λ and μ are the Lagrange multipliers, $b(x)$ are the inequality constraints functions and $c(x)$ are the equality constraints functions. By defining a

new search direction p_k for the solution to a quadratic programming subproblem at iteration x_k :

$$\min L(x_k,\lambda_k,\mu_k)+\nabla L(x_k,\lambda_k,\mu_k)^T p+\frac{1}{2}p^T\nabla^2_{xx}L(x_k,\lambda_k,\mu_k)p \quad (10)$$

$$\text{subject to } b(x_k)+\nabla b(x_k)^T p\geq 0$$

$$c(x_k)+\nabla c(x_k)^T p=0$$

2.2.3.2 Reduced Gradient Methods

Reduced Gradient method was proposed by Marguerite Frank and Phil Wolfe as an algorithm for quadratic programming [40]. It maintains the feasibility of the solution at every iteration. This approach has a number of advantages. If the approximation at an iteration is feasible, and the method may be stopped before the process converges, the current approximated solutions may still be useful. Besides, with the current value of the objective function, the identification of convergence can be simpler without making use of an auxiliary merit function. But the main disadvantage of this method is the computation cost for ensuring every constraint remains satisfied in each iteration. Since the constraints are nonlinear, this means a system of nonlinear equations needs to be solved at each trial point to satisfy the constraint requirement. For a system with a large number of nonlinear constraints, a huge computational cost will be needed to solve the nonlinear system, and the system may fail to find a solution. Moreover, variants of the reduced gradient methods have been developed to provide flexibility in allowing for some violations in the constraints during the optimization. These methods can be considered as a compromise between the reduced gradient method and sequential quadratic programming method. An example of this algorithm is described in the paper by Drud [41].

2.2.3.3 Interior Point Methods

Since Karmarkar developed interior point method in 1943 [38], it has become a major focus of theoretical research and practical experiments. It is efficient for linear, convex quadratic and general nonlinear problems. The method solves a sequence of subproblems with a barrier parameter that converges to zero. The solution to this system is used to update the primal and dual iterates simultaneously. Under certain conditions, and if the initial guess is sufficiently close to the solution, the algorithm

can achieve a quadratic rate of convergence. These methods provide an attractive alternative to the active set strategies for nonlinear optimization with a large number of nonlinear inequality constraints.

2.2.4 Optimization Solver

In the last 15 years, the quality of nonlinear optimization software has improved a lot. FILTERSQP [42] and SNOPT [43] both use the active-set sequential quadratic programming (SQP) methods. FILTERSQP use a trust region approach, together with filter globalization. SNOPT follow a line search approach and provided quasi-Newton approximation for the Hessian matrix. CONOPT [44] utilizes reduced Hessian and SQP methods. The MINOs [45] and LANCELOT [46] packages are capable of solving problems with large-scale optimization, and they implement the augmented Lagrangian methods. Moreover, most of the newly developed packages provide optimization based on the interior point methods. LOQO [47] adopts a line search primal-dual algorithm which can be considered as a modified version of interior methods for linear and quadratic programming. MOSEK [48] implements a primal-dual interior point method for convex optimization. BARNLP [49] adopts the line search interior point approaches, while IPOPT [50] implements a filer globalization and provides a feasibility restoration phase.

2.2.4.1 KNITRO

KNITRO [51] integrates the active set approach and the interior point approach for continuous, nonlinear optimization problem. The active set method uses a Sequential Linear-Quadratic Programming (SLQP) algorithm that uses linear programming subproblems to estimate the active set. The interior point method provides two different ways to solve the primal-dual Karush–Kuhn–Tucker (KKT) system which can be solved by using the direct linear algebra (Interior/Direct algorithm) or the projected conjugate gradient iteration (Interior/CG algorithm). The interior point methods can be combined with the active set methods to provide highly accurate active set and sensitivity information.

In order to maintain certain geometric feature, we need to constrain the point to point distance. However, these distance constraints are nonlinear. Moreover, we need dot product to constrain some angle features in model, and dot product between two

normalized vectors is non-convex. So because of the large number of convex and non-convex nonlinear constraints, this means that it is a large-scale continuous and nonlinear non-convex optimization problem. And interior point method is proved to be efficiency when solving this kind of optimization problem, so this system has implemented KNITRO as the solver.

KNITRO provides the Interior/Direct approach and Interior/CG approach. The Interior/Direct method computes new iterates by solving the primal-dual KKT matrix using direct linear algebra. The Interior/CG approach computes new iterates using a projected conjugate gradient iteration. This method differs from most interior point methods proposed in the literature. It computes steps by using a quadratic model and trust regions. This formulation allows great freedom in the choice of the Hessian and provides a mechanism for coping with Jacobian and Hessian singularities. And this method use requires much less memory during the process than the Interior/Direct approach. For the size of our problem, the Hessian matrix is large and dense and using most of the memory, so we choose the Interior/CG method that offer big saving in memory.

3. SPECIFICATION OF CONSTRAINTS

In this chapter, we will describe the formulation of constraints. For every feature in parametric modeling, a set of constraints is required to maintain their feature characteristics. Most of these feature requirements can be classified into primitive constraints of points, lines or triangles. Besides, the use of references in a structural constraint that is constructed from a group of constraints will be presented.

3.1 Constraints

In order to preserve engineering features in a given model, features to be preserved are identified by selecting some key feature vertices of the mesh. These selected features will then be grouped into systems of primitive constraints to be used in the subsequent optimization process. This optimization process minimizes the difference between the user-deformed model and the target model with the presence of constraints. Because of the nonlinear properties of the specified constraint, a nonlinear optimization technique has to be adopted.

For every feature in a parametric model, a set of constraints is required to maintain their feature characteristics. For example, an extrusion feature needs to have a cross-section profile as a base for the feature body, and the side faces of the feature have to be perpendicular to the plane of the cross-section profile. Most of these feature constraints can be classified into primitive constraints of point, line or triangles. The primitive constraints can be specified with point–point, point–line, line–line, line–triangle, triangle–triangle, and triangle–point relations. They can be angle or distance constraint. Different numbers of configurations may be specified between different primitives as shown in Figure 1. Different configurations mean that they are connected with different number of vertices. Besides, different constraint functions are obtained with different configuration.

Figure 1. Three configurations between two triangles. (a) No vertex is connected between two triangles. (b) One vertex is connected. (c) Two vertices are connected.

Table 1. Types of constraint and numbers of geometry configuration between different primitives

Constraint type	Distance constraint		Angle constraint		No. of configuration
point – point	✔	Eq. 12			1
point – line	✔	Eq. 14			1
point – triangle	✔	Eq. 17			1
line – line	✔	Eq. 14 & Eq. 20	✔	Eq. 19	2
line – triangle	✔	Eq. 17 & Eq. 26	✔	Eq. 25	2
triangle – triangle	✔	Eq. 17 & Eq. 29	✔	Eq. 27	3

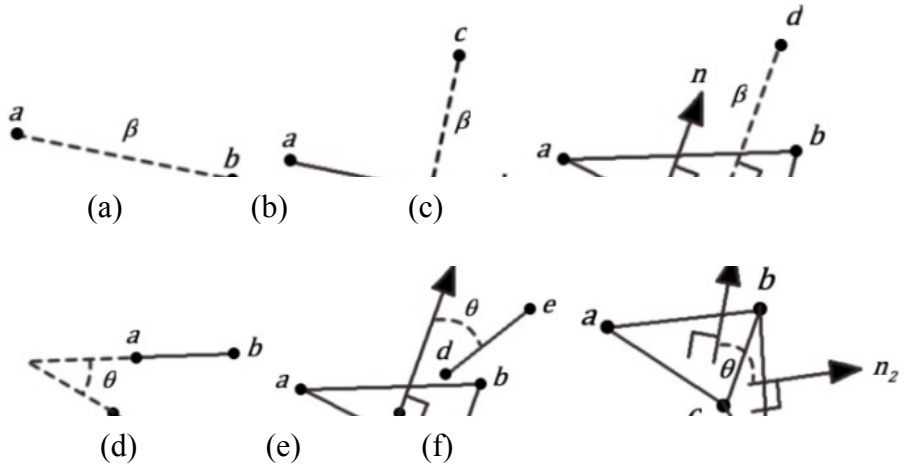


Figure 2. Dist
point distance (b) line - point distance (c) triangle - point distance (d) line - line angle (e) triangle - line angle (f) triangle - triangle angle

Given the distance β from point $\vec{a}(x_a, y_a, z_a)$ to point $\vec{b}(x_b, y_b, z_b)$:

$$\beta = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2} \tag{11}$$

Equation (11) can be simplified as:

$$(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2 - \beta^2 = 0 \tag{12}$$

Given the distance β from a point $\vec{c}$ to a line $(\vec{a}, \vec{b})$:

$$\beta = \frac{\left|(\vec{c} - \vec{a}) \times (\vec{c} - \vec{b})\right|}{\left|\vec{b} - \vec{a}\right|} \tag{13}$$

or

$$\left|(\vec{c} - \vec{a}) \times (\vec{c} - \vec{b})\right| - \beta \left|\vec{b} - \vec{a}\right| = 0 \tag{14}$$

Given the distance β from a point $\vec{d}$ to a triangle $(\vec{a}, \vec{b}, \vec{c})$ which lies on the plane with unit normal $\hat{n}$:

$$\hat{n} = \frac{(\vec{a} - \vec{c}) \times (\vec{b} - \vec{c})}{\left|(\vec{a} - \vec{c}) \times (\vec{b} - \vec{c})\right|} \tag{15}$$

$$\hat{n} \cdot (\vec{d} - \vec{c}) = \beta \tag{16}$$

or

$$\hat{n} \cdot (\vec{d} - \vec{c}) - \beta = 0 \tag{17}$$

Given the angle θ between line $(\vec{a}, \vec{b})$ and line $(\vec{c}, \vec{d})$:

$$\frac{(\vec{a} - \vec{b}) \cdot (\vec{c} - \vec{d})}{\left|\vec{a} - \vec{b}\right| \left|\vec{c} - \vec{d}\right|} = cos(\theta) \tag{18}$$

or

$$(\vec{a} - \vec{b}) \cdot (\vec{c} - \vec{d}) - \left|\vec{a} - \vec{b}\right| \left|\vec{c} - \vec{d}\right| cos(\theta) = 0 \tag{19}$$

From (21), the parallel property between two lines can be constrained by:

$$(\vec{a} - \vec{b}) \cdot (\vec{c} - \vec{d}) - \left|\vec{a} - \vec{b}\right| \left|\vec{c} - \vec{d}\right| = 0 \tag{20}$$

From (21), the prependicaulr property between two lines can be constrained by:

$$(\vec{a}-\vec{b})\cdot(\vec{c}-\vec{d})=0 \tag{21}$$

For the case that two lines share one point $\vec{b}$ (i.e. $\vec{b}=\vec{d}$), equation (19) becomes:

$$(\vec{a}-\vec{b})\cdot(\vec{c}-\vec{b})-|\vec{a}-\vec{b}||\vec{c}-\vec{b}|\cos(\theta)=0 \tag{22}$$

From (22), collinearity of three points can be constrained by:

$$\left(\frac{(\vec{a}-\vec{b})\cdot(\vec{c}-\vec{b})}{|\vec{a}-\vec{b}||\vec{c}-\vec{b}|}\right)^2=1 \tag{23}$$

or

$$((\vec{a}-\vec{b})\cdot(\vec{c}-\vec{b}))^2-(|\vec{a}-\vec{b}||\vec{c}-\vec{b}|)^2=0 \tag{24}$$

Given two parallel disconnected lines, if we need to constrain the distance between them, first we need to use equation (20) to constrain the parallel property. And then, the distance between them can be constrained by the distance from one of the lines to a point on the other line by the use of equation (14).

Given the angle θ between triangle $(\vec{a},\vec{b},\vec{c})$ and a line $(\vec{d},\vec{e})$ which lies on the plane with unit normal $\hat{n}$:

$$\hat{n}\cdot(\vec{d}-\vec{e})-(\vec{d}-\vec{e})\cos(\theta)=0 \tag{25}$$

In order to constrain the distance between a triangle (with unit normal $\hat{n}$) and a line $(\vec{d},\vec{e})$, the line is assumed to be parallel to the plane of the triangle, it means the angle between the line and the normal to the plane of the triangle is 90 degree. Equation (25) becomes:

$$\hat{n}\cdot(\vec{d}-\vec{e})=0 \tag{26}$$

And the distance between them can be constrained by the distance from one point of the lines to the plane of the triangle by the use of equation (17).

Given the angle θ between a triangle (with face normal $\vec{f}_1$) and another triangle (with face normal $\vec{f}_2$):

$$\frac{\vec{f}_1 \cdot \vec{f}_2}{\left|\vec{f}_1\right|\left|\vec{f}_2\right|} = cos\,(\theta) \tag{27}$$

This gives

$$\vec{f}_1 \cdot \vec{f}_2 - \left|\vec{f}_1\right|\left|\vec{f}_2\right| cos\,(\theta) = 0 \tag{28}$$

In order to constrain the distance between a triangle (with face normal $\vec{f}_1$) and another triangle (with face normal $\vec{f}_2$), these triangles are assumed to be parallel to each other, it means the angle between the face normal of the triangles is 0 or 180 degree. Equation (27) becomes:

$$\left(\frac{\vec{f}_1 \cdot \vec{f}_2}{\left|\vec{f}_1\right|\left|\vec{f}_2\right|}\right)^2 = 1 \tag{29}$$

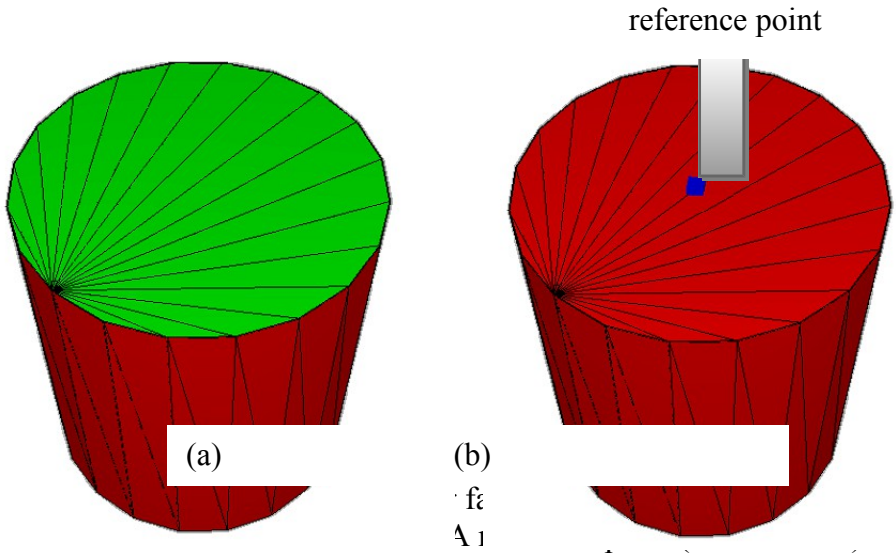
And then the distance between them can be constrained by the distance from the plane of the triangle to a point on the other triangle by the use of equation (17).

3.1.1 Constraints with Reference Points

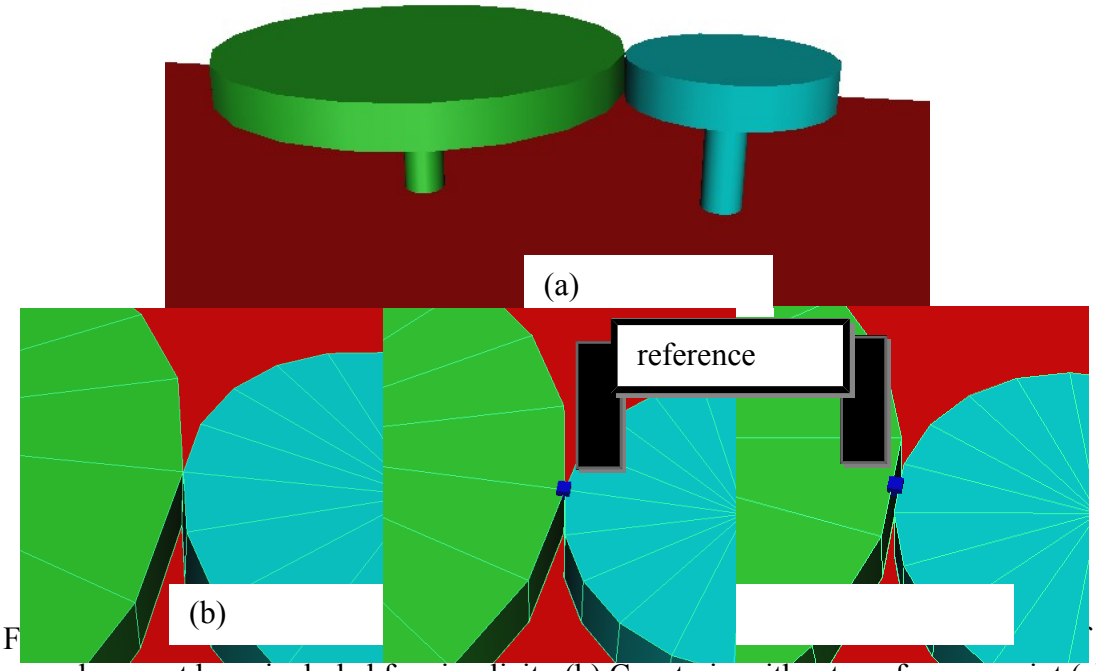
Reference points are required to be added in to the constraint system in order to provide constraint on some geometric features that cannot represent with mesh vertices only. For example, if we want to maintain a circular shape that existed on a surface of a mesh, but the mesh do not contain a vertex that can be specified as the centre of this circular feature, a reference point at the centre of this circular shape will be required for setting up the necessary constraints. An example is shown in Figure 3.

For some structural constraints, reference points are also required. Structural constraints refer to constraints that maintain the relationships between feature and feature, rather that the position of vertices on one single feature. For example, using a reference point to represent the contact point of two gear models will provide more freedom of movement for the gear model. This is because if we use mesh vertex as the contact point, the contact vertex may also associated with other constraint, such as

planar and circular constraints such that its movement may be limited. But if we use a reference point as the contact point, fewer constraints are associated with this reference point, freedom of movement for the gear model will be increased. An example is shown in Figure 4.



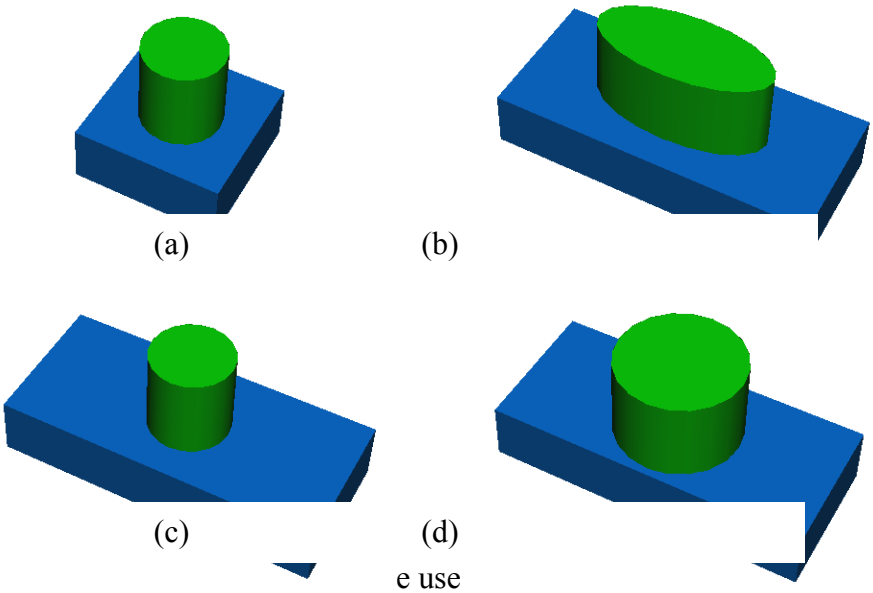
Figur fa ır)
that d A 1 dded
at the centre of the surface for the use of constraint establishment.



F
gears have not been included for simplicity (b) Constrain without a reference point (c) Constrain with a reference point (d) The gear can be rotated with the existence of the reference point.

3.1.2 Constraints with Reference Variables

Constraint can be constructed with a reference variable. A reference variable is an additional variable included in the constraint system in order to provide shared references between constraints. For example, a reference variable is added when a user wants to constrain the circular shape of a feature, but he/she do not want to constraint its radius to an exact value, so all the distances in the distance constraints that are used in this constraint group will be set to be the same as a shared reference variable. An example is shown in Figure 5.



(a) (b)

(c) (d)

Figure e use original undefo[illegible]etching the model (c) The exact dimensions of the cylinder has been retained, which mean the size of the cylinder is the same as that in Figure 5(a). (d) Using a reference variable for the distance constraints for the radius of the cylinder, the circularity of the shape is retained, but the size of the cylinder is scaled up as a result of the stretching operation.

This kind of reference variable can be used in all the angle and distance constraints. This means that it can be used to constrain the shape of a feature without providing the exact numerical dimensions. Using a reference variable in a distance constraint specified with equation (12), a reference variable R will replaced β which gives:

$$(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2 - R^2 = 0 \tag{30}$$

3.1.3 Reference Vector Constraints

A reference vector can be created form two vertices or cross product between two vectors which can be face normals or other reference vectors.

Given a vector $\hat{\boldsymbol{v}}$ formed from point $\vec{\boldsymbol{a}}$ to point $\vec{\boldsymbol{b}}$:

$$\hat{\boldsymbol{v}} = \frac{\vec{\boldsymbol{b}} - \vec{\boldsymbol{a}}}{\left|\vec{\boldsymbol{b}} - \vec{\boldsymbol{a}}\right|} \tag{31}$$

In order to prevent zero vector , the length of the vector should not become zero.

$$\left|\vec{\boldsymbol{b}} - \vec{\boldsymbol{a}}\right| \geq \boldsymbol{T} \tag{32}$$

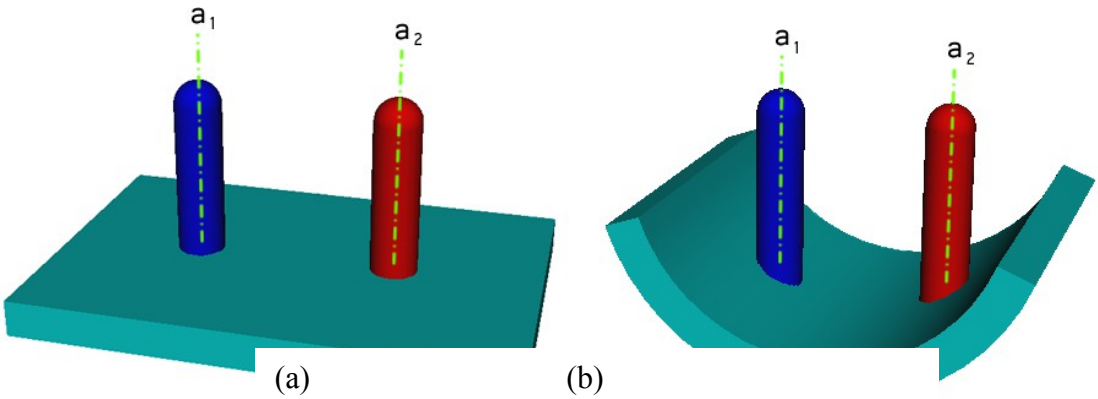
where $\boldsymbol{T}$ is the threshold.

Given a vector $\vec{\boldsymbol{v}}_a$ which is the cross product of the vectors $\vec{\boldsymbol{v}}_b$ and $\vec{\boldsymbol{v}}_c$:

$$\vec{\boldsymbol{v}}_a = \vec{\boldsymbol{v}}_b \times \vec{\boldsymbol{v}}_c \tag{33}$$

Vector dot product is used to constrain the angle between two vectors. Given the angle θ between unit vector $\hat{\boldsymbol{v}}_a$ and unit vector $\hat{\boldsymbol{v}}_b$:

$$\hat{\boldsymbol{v}}_a \cdot \hat{\boldsymbol{v}}_b = cos\,(\theta) \tag{34}$$



(a) (b)

s

the axes of two bars ( ), their axes will remain parallel in a deformation, and $\hat{\boldsymbol{a}}_1 \cdot \hat{\boldsymbol{a}}_2 = 1$
the result is shown in Figure 6(b).

Reference vector constraints are used for setting up constrains on the reference vector rather than mesh vertices. They are higher level constraints and are usually required when users want to provide constraints across features that cannot be specified with mesh vertices directly. An example is given in Figure 6.

3.1.4 Constraints with Reference Datum

For a group of constraints, it would be better to use a reference datum for all its component constraints. The use of reference datum speeds up and stabilized the optimization process. In the process, when constraints between adjacent primitives are being retained, the optimization process may terminate if one of the constraints is not satisfied. By using a common datum for a group of primitives, the chance of having one constraints being not satisfied is reduced. There are several types of constraints that can be implemented with this method such as planar, collinear and circular constraints.

3.1.4.1 Planer Constraint with References

For planer constraints on a set of triangles, by using a common reference for each of the triangle, the numerical error accumulated in the optimization process can be reduced.

For example, a planar surface can be constructed with a set of triangles sharing a common face normal $\hat{n}$. Given a plane with unit normal $\hat{n}(\hat{n}_x, \hat{n}_y, \hat{n}_z)$, this can be constrained as:

$$\hat{n}_x^{\,2}+\hat{n}_y^{\,2}+\hat{n}_z^{\,2}-1=0 \tag{35}$$

Given a planar surface with a unit normal $\hat{n}$, all points lying on the same plane can be constrained as following:

$$\hat{n}\cdot(\acute{v}_i-\acute{v}_c)=0 \tag{36}$$

where $\acute{v}_i$ is the *i*-th vertex lying on the plane and $\acute{v}_c$ is the common vertex lying on the same plane.

By using equation (36), the constrained points are to lie on the same plane. However, it does not prevent the triangle faces from overlapping. In order to eliminate overlapping triangles, the angle between the plane unit normal $\hat{n}$ and each triangle face normal $\acute{f}_i$ has to be less than 90 degrees.

$$\acute{f}_i\cdot\hat{n}\geq 0 \tag{37}$$

In order to apply appropriate constraints to all coplanar faces, all constraints are analyzed first, and then all coplanar triangles are grouped together. Similarly, the points on the same planes are grouped too. For a plane with *M* triangle faces and *N* vertices, to constraint its planarity, equation (35) specifies the normal at the reference vertex, $N-1$ equation (36) is used to specify the locations of the other vertices, and M equation (37) are required to prevent the overlapping of triangles.

By using the plane unit normals $\hat{n}$, the perpendicular property between the *i*-th and the *j*-th plane can be specified as following:

$$\hat{n}_i\cdot\hat{n}_j=0 \tag{38}$$

3.1.4.2 Collinear Constraint with References

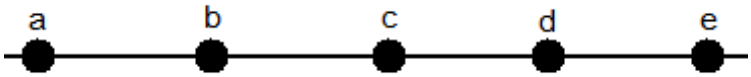


Figure 7. A straight line with 5 points

A collinear constraint is used to maintain the angle between two consecutive line segments at 180 degrees.

Instead of applying the collinear constraints on adjacent segments of a straight edge, constraints can be applied to each line segment relative to the line defined between the start point and the end point of the edge. It will improve the performance of the optimization.

The collinearity of three points can be constrained as follow (Figure 7):

$$\frac{(\acute{a}-\acute{b})\cdot(\acute{c}-\acute{b})}{|\acute{a}-\acute{b}||\acute{c}-\acute{b}|}=-1 \quad , \quad \frac{(\acute{b}-\acute{c})\cdot(\acute{d}-\acute{c})}{|\acute{b}-\acute{c}||\acute{d}-\acute{c}|}=-1 \quad , \ldots\ldots \tag{39}$$

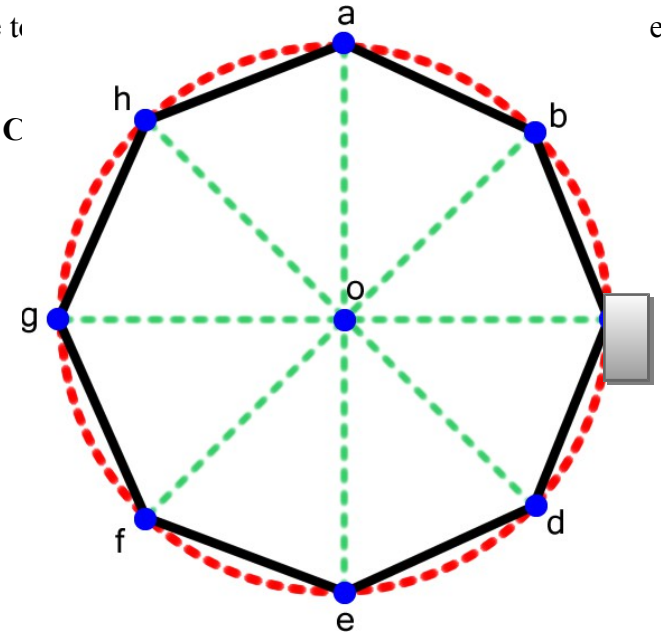
In this representation, the angle constraints are applied on adjacent line segments. Numerical error will accumulated while the optimization propagates along the straight edge.

A common reference can be used to reduce the amount of accumulated errors. In accordance, the straight line in Figure 7 can be constrained with the following:

$$\frac{(\acute{a}-\acute{e})\cdot(\acute{b}-\acute{a})}{|\acute{a}-\acute{e}||\acute{b}-\acute{a}|}=-1 \quad , \quad \frac{(\acute{a}-\acute{e})\cdot(\acute{c}-\acute{b})}{|\acute{a}-\acute{e}||\acute{c}-\acute{b}|}=-1 \quad , \ldots\ldots \tag{40}$$

In this representation, the angle constraints applied to each line segment are specified relative t end point.

3.1.4.3 Circular C



mesh

Figure 8. A circular mesh

In order to confine a set of points to lie on a circle, circular constraints are to be adopted. A circular constraint is specified with distance constraints and angle constraints. If we want to maintain the circular property of the point $\acute{a}$, $\acute{b}$, $\acute{c}$, $\acute{d}$, $\acute{e}$, $\acute{f}$, $\acute{g}$, $\acute{h}$ in Figures 8, we will need to apply the circular constraint on these points. Distance constraints are used to maintain the distance between the origin $\acute{o}$ to all the points in the group. Angle constraints are applied to adjacent radial line segments (e.g. $\acute{o}\acute{a}$ and $\acute{o}\acute{b}$, $\acute{o}\acute{b}$ and $\acute{o}\acute{c}$, etc). The constraints will be specified as following:

$$\frac{(\acute{a}-\acute{o})\cdot(\acute{b}-\acute{o})}{|\acute{a}-\acute{o}||\acute{b}-\acute{o}|}=\cos(angle\ between\ \acute{o}\acute{a}\wedge\acute{o}\acute{b}) \tag{41}$$

$$\frac{(\acute{b}-\acute{o})\cdot(\acute{c}-\acute{o})}{|\acute{b}-\acute{o}||\acute{c}-\acute{o}|}=\cos(angle\ between\ \acute{o}\acute{b}\wedge\acute{o}\acute{c})$$

$$\frac{(\acute{c}-\acute{o})\cdot(\acute{d}-\acute{o})}{|\acute{c}-\acute{o}||\acute{d}-\acute{o}|}=\cos(angle\ between\ \acute{o}\acute{c}\wedge\acute{o}\acute{d})$$

……

Instead of applying angle constraints on adjacent line segments, angle constraints would better be specified relative to a reference line. In Figure 8, we use line $\acute{o}\acute{a}$ as the reference line, and the angle constraints can be formulated as following:

$$\frac{(\acute{a}-\acute{o})\cdot(\acute{b}-\acute{o})}{|\acute{a}-\acute{o}||\acute{b}-\acute{o}|}=\cos(angle\ between\ \acute{o}\acute{a}\wedge\acute{o}\acute{b}) \tag{42}$$

$$\frac{(\acute{a}-\acute{o})\cdot(\acute{c}-\acute{o})}{|\acute{a}-\acute{o}||\acute{c}-\acute{o}|}=\cos(angle\ between\ \acute{o}\acute{a}\wedge\acute{o}\acute{c})$$

$$\frac{(\acute{a}-\acute{o})\cdot(\acute{d}-\acute{o})}{|\acute{a}-\acute{o}||\acute{d}-\acute{o}|}=\cos(angle\ between\ \acute{o}\acute{a}\wedge\acute{o}\acute{d})$$

……

3.2 Redundant Constraints

The constraint system should not contain any redundant constraints. This is because redundant constraints will lead to degeneracy such that extra iterations is required in the optimization process, or in some cases, the optimization process may fail to locate a solution. In general, users should plan the specification of constraints carefully, and clearly identify that there will not be geometrical or numerical redundancies. For example, if we are specifying the circular shape of the mesh in Figure 8, only 7 angle constraints, 8 radius distance constraints and planar constraints are needed to constrain this circular face. If one more angle constraint is added to the system, it will become a redundant constraint and cause degeneracy in the optimization process.

4. CONSTRAINED OPTIMZATION

In this chapter, the overall optimization process implemented in the experimental system will be presented. First it provides the objective function and the overall formulation of the optimization problem. And then it introduces an incremental optimization method to solve the problem when the optimization fails to find an optimal solution. When the optimization fails, the system will perform interpolation between the initial guess and a transitional model, and then generates an interpolated model for the next optimization.

4.1 Objective Function

With the above primitive constraints, geometric features commonly found in engineering components can be specified. In order to retain the specified constraints after a deformation, an optimization process is performed. This optimization provides a better reconstruction result than reconstructing the constraints one by one, because the optimization tries to preserve the deformed shape rather than only considering individual constraint. It also allows the deformation to have more effect on the constrained model.

By minimizing the least squared distance between the deformed coordinates and the coordinates generated in the optimization process, the deformation can be retained as much as possible in the existence of constraints. The function can be written as:

$$\boldsymbol{f}(x',y',z') = \sum_{i=1}^{n}((x_i - x'_i)^2 + (y_i - y'_i)^2 + (z_i - z'_i)^2) \tag{43}$$

where $\vec{\boldsymbol{v}}_{\boldsymbol{i}}(x_i, y_i, z_i)$ is the *i*-th vertex of the transitional model and $\vec{\boldsymbol{v}}'_{\boldsymbol{i}}(x'_i, y'_i, z'_i)$ is the *i*-th vertex of the optimized constrained deformed model. Reference points are not included in the objective function, because they are used in the constraints only. Since the objective function is nonlinear, a nonlinear solver is required to minimize this objective function. For a general optimization problem, we need to provide an objective function, constraints and initial guess. The initial guess is used as the input for the optimization. Therefore, the optimization problem can be written as:

$$min\left[f(x,y,z)\right] subjected\ to \sum_{i=1}^{a}\sum_{j=1}^{b} A_{i,j}(x,y,z) \tag{44}$$

where $A_{i,j}(x,y,z)$ are the constraint functions, a is the number of constraint type and b is the number of constraints of the same type. $A_{i,j}(x,y,z)$ can be a combination of equation (12), (14), (17), (19-22), (24-26) and (28- 42).

The overall procedure is listed as follow: First, the user specifies the constraints in the undeformed model (Figure 9), and then the users perform deformation on the model to produce a transitional model. Finally, the system performs an optimization that minimizes the displacement on the vertices of the transitional model using the original undefor

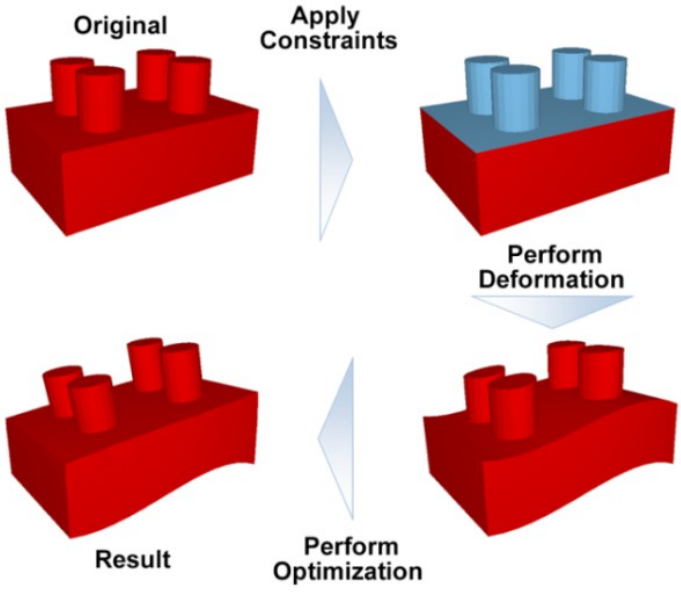


Figure 9. Preserving constraints on a Lego model. By applying constraints on the four cylinders and the plane (highlighted), the shape of the cylinder can be retained

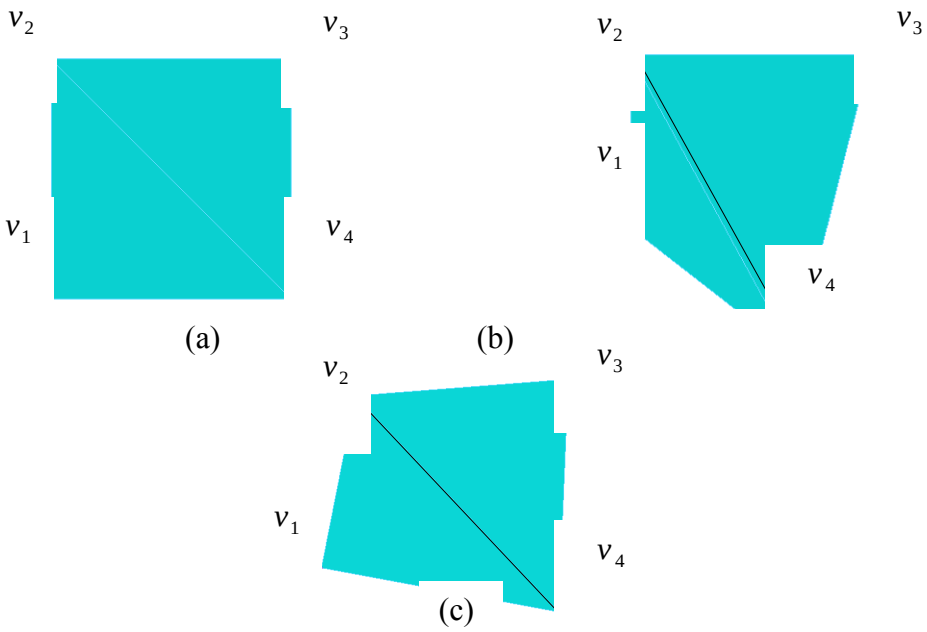


Figure 10. A plane model example

Here we will give an example on the formulation of the optimization problem. In Figure 10(a), there is a plane with 4 vertices. We want to maintain the distance between vertex $\acute{v}_1(x_1, y_1, z_1)$ and vertex $\acute{v}_4(x_4, y_4, z_4)$ and the distance between vertex $\acute{v}_3(x_3, y_3, z_3)$ and $\acute{v}_4$ with distance β (which is a constant). We also want to maintain the right angle at vertex $\acute{v}_1$ with line $v_1'v_2$ and line $v_1'v_4$. And then we deformed the model as shown in Figure 10(b), and use it as the transitional model. After that, we perform the constrained optimization with it.

By equation (43), the objective function for this problem will become:

$$\boldsymbol{f}(x',y',z') = \sum_{i=1}^{4}((x_i - x'_i)^2 + (y_i - y'_i)^2 + (z_i - z'_i)^2) \tag{45}$$

where $\vec{\boldsymbol{v}}_{\boldsymbol{i}}(x_i, y_i, z_i)$ is the i-th vertex of the transitional model (which are all constants) and $\vec{\boldsymbol{v}}'_{\boldsymbol{i}}(x'_i, y'_i, z'_i)$ is the i-th vertex of the optimized constrained deformed model and they are the variables in the optimzaimation problem.

By equation (14), the distance constraint of line $v_1'v_4$ will become:

$$(x'_1 - x'_4)^2 + (y'_1 - y'_4)^2 + (z'_1 - z'_4)^2 - \beta^2 = 0 \tag{46}$$

And the distance constraint of line $v_3'v_4$ will become:

$$(x'_3 - x'_4)^2 + (y'_3 - y'_4)^2 + (z'_3 - z'_4)^2 - \beta^2 = 0 \tag{47}$$

By equation (22), the perpendicular property of line $v_1'v_2$ and $v_1'v_4$ will become:

$$(\vec{\boldsymbol{v}}_2 - \vec{\boldsymbol{v}}_1)\cdot(\vec{\boldsymbol{v}}_4 - \vec{\boldsymbol{v}}_1) = 0 \tag{48}$$

And then:

$$(x'_2 - x'_1)(x'_4 - x'_1) + (y'_2 - y'_1)(y'_4 - y'_1) + (z'_2 - z'_1)(z'_4 - z'_1) = 0 \tag{49}$$

Rewriting the constraints as constraint functions c_1, c_2, c_3, the system becomes:

$$c_1 = ¿ \quad (x'_1 - x'_4)^2 + (y'_1 - y'_4)^2 + (z'_1 - z'_4)^2 - \beta^2 \tag{50}$$

$$c_2 = ¿ \quad (x'_3 - x'_4)^2 + (y'_3 - y'_4)^2 + (z'_3 - z'_4)^2 - \beta^2 \tag{51}$$

$$c_3 = ¿ \quad (x'_2 - x'_1)(x'_4 - x'_1) + (y'_2 - y'_1)(y'_4 - y'_1) + (z'_2 - z'_1)(z'_4 - z'_1) \tag{52}$$

Besides the objective function and the constraint functions, we also need to provide the Jacobian matrix and Hessians of the objective and constraint functions for the solver during the optimization.

The gradients (first derivatives) of the objective and constraint functions are given by:

$$\nabla f = \begin{matrix} -2(x_1 - x'_1) \\ -2(y_1 - y'_1) \\ -2(z_1 - z'_1) \\ -2(x_2 - x'_2) \\ -2(y_2 - y'_2) \\ -2(z_2 - z'_2) \\ -2(x_3 - x'_3) \\ -2(y_3 - y'_3) \\ -2(z_3 - z'_3) \\ -2(x_4 - x'_4) \\ -2(y_4 - y'_4) \\ -2(z_4 - z'_4) \end{matrix}, \quad \nabla c_1 = \begin{bmatrix} 2(x'_1 - x'_4) \\ 2(y'_1 - y'_4) \\ 2(z'_1 - z'_4) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -2(x'_1 - x'_4) \\ -2(y'_1 - y'_4) \\ -2(z'_1 - z'_4) \end{bmatrix}, \quad \nabla c_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2(x'_3 - x'_4) \\ 2(y'_3 - y'_4) \\ 2(z'_3 - z'_4) \\ -2(x'_3 - x'_4) \\ -2(y'_3 - y'_4) \\ -2(z'_3 - z'_4) \end{bmatrix}$$

$$, \quad \nabla c_3 = \begin{bmatrix} 2x'_1 - x'_2 - x'_4 \\ 2y'_1 - y'_2 - y'_4 \\ 2z'_1 - z'_2 - z'_4 \\ x'_4 - x'_1 \\ y'_4 - y'_1 \\ z'_4 - z'_1 \\ 0 \\ 0 \\ 0 \\ x'_2 - x'_1 \\ y'_2 - y'_1 \\ z'_2 - z'_1 \end{bmatrix}$$

The constraint Jacobian matrix J is the matrix whose rows store the transpose of the constraint gradients:

$$J = \begin{bmatrix} \nabla c_1^T \\ \nabla c_2^T \\ \nabla c_3^T \end{bmatrix} = ¿$$

$$\begin{bmatrix} 2(x'_1 - x'_4) & 2(y'_1 - y'_4) & 2(z'_1 - z'_4) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 2x'_1 - x'_2 - x'_4 & 2y'_1 - y'_2 - y'_4 & 2z'_1 - z'_2 - z'_4 & x'_4 - x'_1 & y'_4 - y'_1 & z'_4 - z'_1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 & -2(x'_1-x'_4) & -2(y'_1-y'_4) & -2(z'_1-z'_4) \\ 2(x'_3-x'_4) & 2(y'_3-y'_4) & 2(z'_3-z'_4) & -2(x'_3-x'_4) & -2(y'_3-y'_4) & -2(z'_3-z'_4) \\ 0 & 0 & 0 & x'_2-x'_1 & y'_2-y'_1 & z'_2-z'_1 \end{bmatrix}$$

The Hessian of the Lagrangian matrix is defined as:

$$H = \nabla^2 f + \sum_{i=0}^{m-1} \lambda_i \nabla^2 c_i \tag{53}$$

where λ is the vector of Lagrange multipliers (dual variables) which is associated with the second derivatives of the constraint functions. The Hessians (second derivatives) of the objective and constraint functions are given by:

$$\nabla^2 f = \begin{bmatrix} 2&0&0&0&0&0&0&0&0&0&0&0 \\ 0&2&0&0&0&0&0&0&0&0&0&0 \\ 0&0&2&0&0&0&0&0&0&0&0&0 \\ 0&0&0&2&0&0&0&0&0&0&0&0 \\ 0&0&0&0&2&0&0&0&0&0&0&0 \\ 0&0&0&0&0&2&0&0&0&0&0&0 \\ 0&0&0&0&0&0&2&0&0&0&0&0 \\ 0&0&0&0&0&0&0&2&0&0&0&0 \\ 0&0&0&0&0&0&0&0&2&0&0&0 \\ 0&0&0&0&0&0&0&0&0&2&0&0 \\ 0&0&0&0&0&0&0&0&0&0&2&0 \\ 0&0&0&0&0&0&0&0&0&0&0&2 \end{bmatrix},$$

$$\nabla^2 c_1 = \begin{bmatrix} 2&0&0&0&0&0&0&-2&0&0 \\ 0&2&0&0&0&0&0&0&-2&0 \\ 0&0&2&0&0&0&0&0&0&-2 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ -2&0&0&0&0&0&0&2&0&0 \\ 0&-2&0&0&0&0&0&0&2&0 \\ 0&0&-2&0&0&0&0&0&0&2 \end{bmatrix},$$

$$\nabla^2 c_2 = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&2&0&0&-2&0&0 \\ 0&0&0&0&0&0&0&2&0&0&-2&0 \\ 0&0&0&0&0&0&0&0&2&0&0&-2 \\ 0&0&0&0&0&0&-2&0&0&2&0&0 \\ 0&0&0&0&0&0&0&-2&0&0&2&0 \\ 0&0&0&0&0&0&0&0&-2&0&0&2 \end{bmatrix},$$

$$\nabla^2 c_3 = \begin{bmatrix} 2&0&0&-1&0&0&0&0&0&-1&0&0 \\ 0&2&0&0&-1&0&0&0&0&0&-1&0 \\ 0&0&2&0&0&-1&0&0&0&0&0&-1 \\ -1&0&0&0&0&0&0&0&0&1&0&0 \\ 0&-1&0&0&0&0&0&0&0&0&1&0 \\ 0&0&-1&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0 \\ -1&0&0&1&0&0&0&0&0&0&0&0 \\ 0&-1&0&0&1&0&0&0&0&0&0&0 \\ 0&0&-1&0&0&1&0&0&0&0&0&0 \end{bmatrix}$$

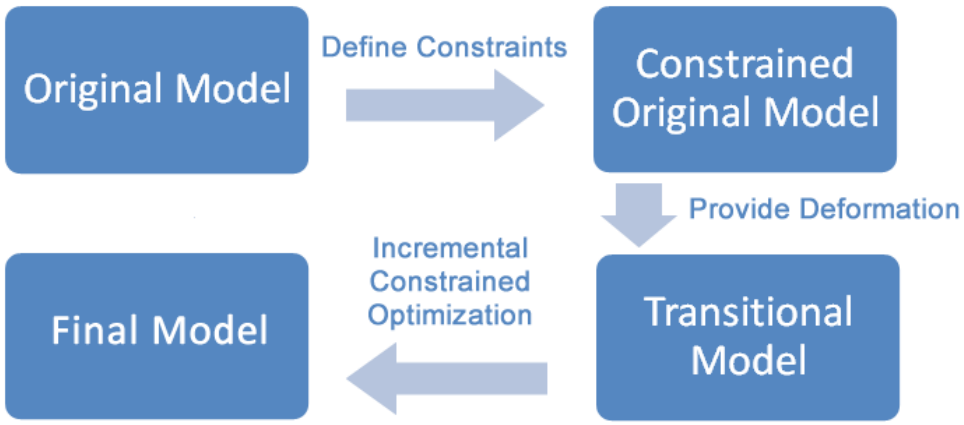
Scaling the constraint matrices by their corresponding Lagrange multipliers and summing, we get:

$H = ¿$

$$\begin{bmatrix} 2+2\lambda_1+2\lambda_3 & 0 & 0 & -\lambda_3 & 0 & 0 & 0 & 0 & 0 & -2\lambda_1-\lambda_3 & 0 & \\ 0 & 2+2\lambda_1+2\lambda_3 & 0 & 0 & -\lambda_3 & 0 & 0 & 0 & 0 & 0 & -2\lambda_1-\lambda_3 & \\ 0 & 0 & 2+2\lambda_1+2\lambda_3 & 0 & 0 & -\lambda_3 & 0 & 0 & 0 & 0 & 0 & -2 \\ -\lambda_3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & \lambda_3 & 0 & \\ 0 & -\lambda_3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & \lambda_3 & \\ 0 & 0 & -\lambda_3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 2+2\lambda_2 & 0 & 0 & -2\lambda_2 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2+2\lambda_2 & 0 & 0 & -2\lambda_2 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2+2\lambda_2 & 0 & 0 & - \\ -2\lambda_1-\lambda_3 & 0 & 0 & \lambda_3 & 0 & 0 & -2\lambda_2 & 0 & 0 & 2+2\lambda_1+2\lambda_2 & 0 & \\ 0 & -2\lambda_1-\lambda_3 & 0 & 0 & \lambda_3 & 0 & 0 & -2\lambda_2 & 0 & 0 & 2+2\lambda_1+2\lambda_2 & \\ 0 & 0 & -2\lambda_1-\lambda_3 & 0 & 0 & \lambda_3 & 0 & 0 & -2\lambda_2 & 0 & 0 & 2+2 \end{bmatrix}$$

After the optimization, the result model is shown in Figure 10(c).

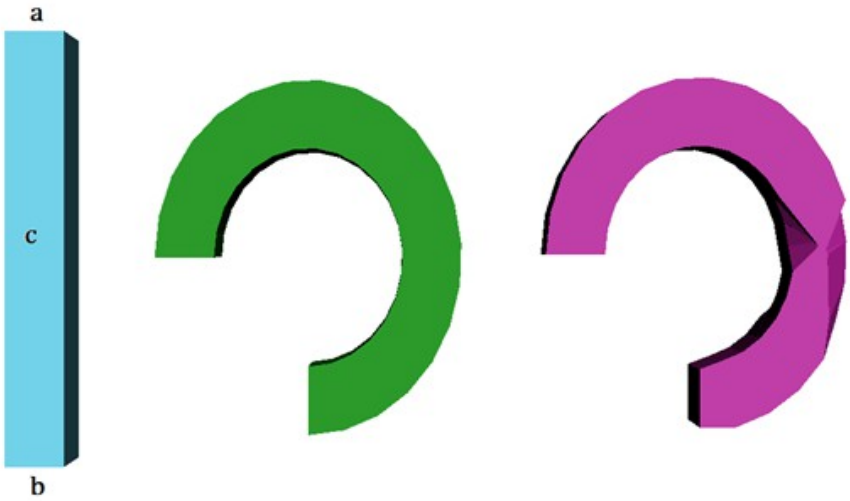
4.2 Incremental Constrained Deformation



F

Because of the nonlinear properties of the specified constraint, a nonlinear optimization technique has to be adopted. In general, a local optimum point can be obtained while a global optimum point may not be achieved. Local optimum point means the specified constraints are satisfied, but the archived optimum point is only a local minimum or maximum point. In other words, even the constraints are satisfied, the user provided deformation may not be fully retained after the optimization. An example is shown in Figure 12 in which some faces collapse although their planarities are maintained. In Figure 12, the straight bar on the left is the original model. The face at the two ends of the bar *a* and *b* are constrained as square with fixed edge length. The side *c* is constrained as planar surface. Then, the model obtained with an unconstrained deformation is shown in the middle of the figure. The bar model on the right is the result of the optimization. As shown in the figure, some faces of the deformed model are collapsed although they lie on the same plane. This is a result of the local solution obtained from the nonlinear optimization.

Figure 12.



In addition, incremental constrained deformation is implemented to provide hints for the optimization. First, the system tries to minimize the displacement between the vertices of the transitional model and the original model using the original undeformed model as the initial guess. If the optimization fails or the collapsing faces are found, then the system will perform interpolation between the initial guess and transitional model, and then generate an interpolated model. Hence, the optimization problem becomes the minimization of the displacement from the interpolated model using the original undeformed model as the initial guess. The interpolation is described as following:

$$\acute{v}_i = \alpha\left(\acute{v}_t - \acute{v}_o\right) + \acute{v}_o \tag{54}$$

where $\acute{v}_o$ is the vertex on the original model, $\acute{v}_t$ is the vertex on the transitional model, $\acute{v}_i$ is the vertex on the interpolated model and α is the factor for the interpolation.

The factor for the first interpolation is set to be 0.5. If the optimization is completed with this factor, then the next interpolation will use 1 as the factor. In this way, the optimization problem becomes the minimization of the displacement in the newly interpolated model using the previous optimal result as the initial guess. However, if the optimization fails or collapsing faces are found, the factor will be set to 0.25, and the system will repeat the optimization with the newly interpolated model. Based on this algorithm, the optimization is performed incrementally to give a constraints retained model. For the case that the interpolation factor is decreased to a user specified level while no optimal result is obtained, the system will use the

optimization result obtained in a previous succeeded incremental optimization. The pseudocode for this incremental method is shown in Table 2.

Table 2. The Pseudocode for the incremental method

```
Define variable OS for the optimization status
Define variable TS for the temination status
Define variable SV for the starting value
Define variable CV for the current interpolated value
Define variable CS for the current step size
Define variable US for the user-defined termination step size
Define variable UIM for the user input model
Define variable TDM for the transitional model
Define variable IM for the interpolated model
Define variable COM for the current optimized model
Define variable POM for the previous optimized model

Initialize TS to false
Initialize SV to 0.0
Initialize CS to 1.0
Initialize CV to 0.0
Initialize US to 0.03125
Copy UIM to POM
While TS is Equal to false
{
    Set CV to the Sum of SV and CS
    Intepolate from UIM to TDM with CV
    Copy the result of Intepolation to IM
    Pass UIM, IM and constraints to the optimizer
    COM and OS is Returned from the optimizer
    If OS is true And CV is 1.0
    {
        Set TS to true
    }
    Else If OS is true And CV is Not Equal to 1.0
    {
        Set SV to CV
        Copy COM to POM
    }
    Else If OS is false
    {
        Set TS to false
        Divide CS by 2.0
        If CS is Smaller than or Equal to US
        {
            Copy POM to COM
            Set TS to true
        }
    }
}
Return COM
```

4.3 The Scaling Problem

Moreover, optimization may fail when the constraints are badly scaled. The differences in magnitudes of the first derivatives of the constraint equations may be very large such that errors in the calculation accumulate. This may finally causes the solver to fail in obtaining a solution. When the optimization process starts, scaling of constraints are performed by the nonlinear optimization solver. The scales are determined base on the provided initial guess for the optimization. The scaling is performed by multiplying different constraint functions with a scale factor such that the order of magnitude of the first derivatives of the functions lies within a range determined by the solver. This makes sure that each constraint functions are comparable in the optimization process. The initial guess of the optimization problem is hence very important to the scaling of the problem. Besides, a well scaled problem requires less computation time and can give more accurate result. Since the scaling of all variables in the optimization may also affect the convergence rate of the constraints problem, normalization is performed on the coordinates of the given vertices.

5. CASE STUDIES

The proposed techniques are implemented on Win7 system using C++ language. We performed all the experiments on a Quad-core 3.00GHz PC with 8G. Tests are performed to study its capability in retaining geometric constraints on models. Test cases are selected to evaluate the ability of the system in maintaining individual feature, pattern of features and relationship between features. Constraints, reference points, reference variables and reference vectors in some example are provided. Beside, the original model, transitional model and the final deformed model for every example will be presented. Deformation techniques used in following examples are free-form deformation. Some implementation issues are also discussed in this chapter.

5.1 Maintain Individual Engineering Features

In the following section, several constraints retained deformation results are illustrated. The results show that the system retains individual features.

By providing the dimensions of the cylindrical features on the model in Figure 13(a), the dimension of the cylindrical features can be retained after various deformations and the cylinders remain perpendicular to the top plane, the result is shown in Figure 14. First, the triangles on the upper plane are constrained with coplanar constraints as shown in Figure 13(b). In order to illustrate the use of reference point on the model, the top face of each cylinder is modelled with an irregular mesh such that no mesh vertex can be used as the centre point. A reference point is inserted at the centre as shown in Figure 13(c). Third, 16 radius distance constraints and 15 angle constraints (Figure 13(d)) are applied on each cylinder. Then, all triangles on the top face of each cylinder are constrained with coplanar constraints as shown Figure 13(e). In Figure 13(f), each pair of perpendicular faces is constrained with a perpendicular constraint. Notice that only one pair of perpendicular faces is highlighted in the figure, there are 16 pairs of perpendicular faces on each cylinder. In Figure 13(g), the triangles of each vertical stripe of the cylinder are constrained with coplanar constraint. Hence, there are 16 groups of coplanar constraints on the vertical side of each cylinder. In Figure 13(h), one triangle on the vertical side of each cylinder is constrained to be perpendicular to one triangle of the top plane. Finally, one more perpendicular constraint is added between the vertical side of each cylinder and the top plane as shown Figure 13(i). With these perpendicular constraints (in

Figure 13(h) and Figure 13(i)) on each cylinder, they are already enough to keep each cylinder perpendicular to the top plane, so no more perpendicular constraint is needed. There will be redundant constraints if more than one perpendicular constraint (with the same property) is specified on each cylinder.

The techniques used in the specification of constraints on the Lego model are applied to a gamepad model. In Figure 15, constraints are applied to the target features of the original gamepad model, and by providing the transitional model, the target features can be retained on the final model.

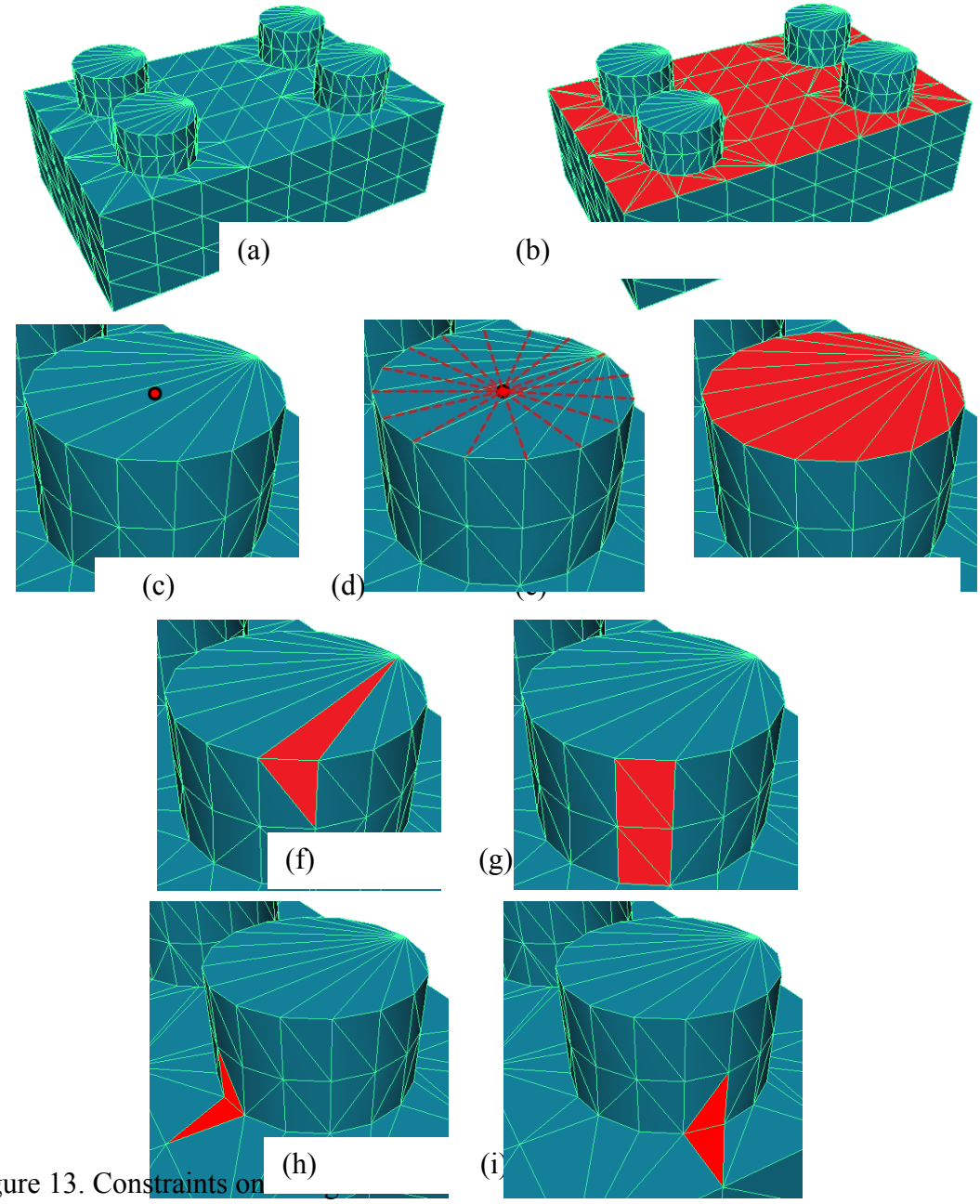


Figure 13. Constraints on

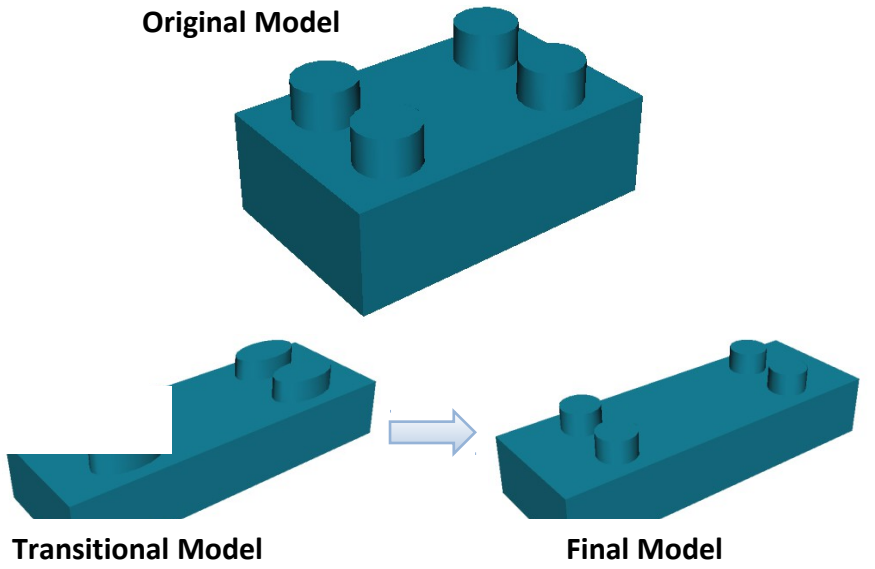
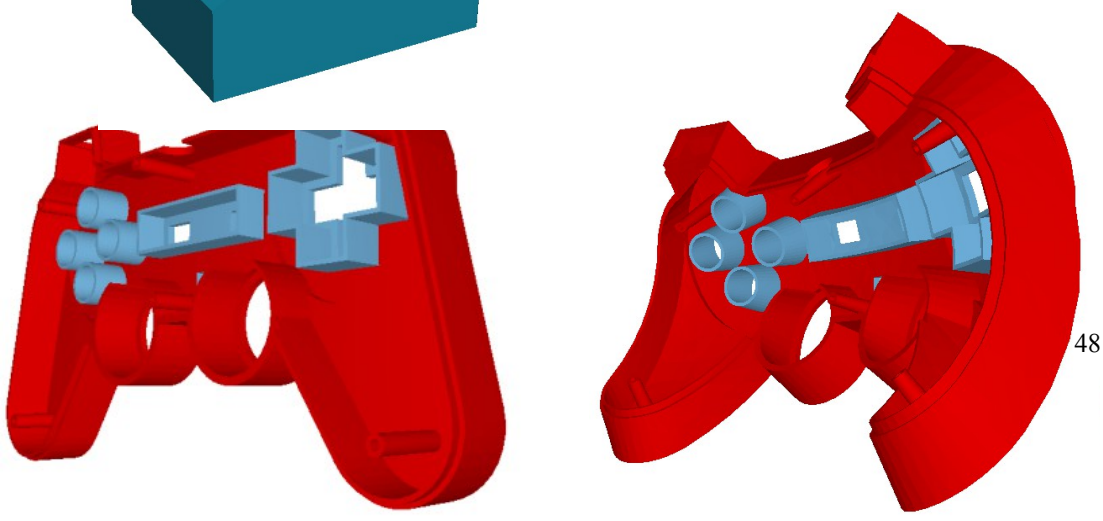


Figure 1 on an obj
models a



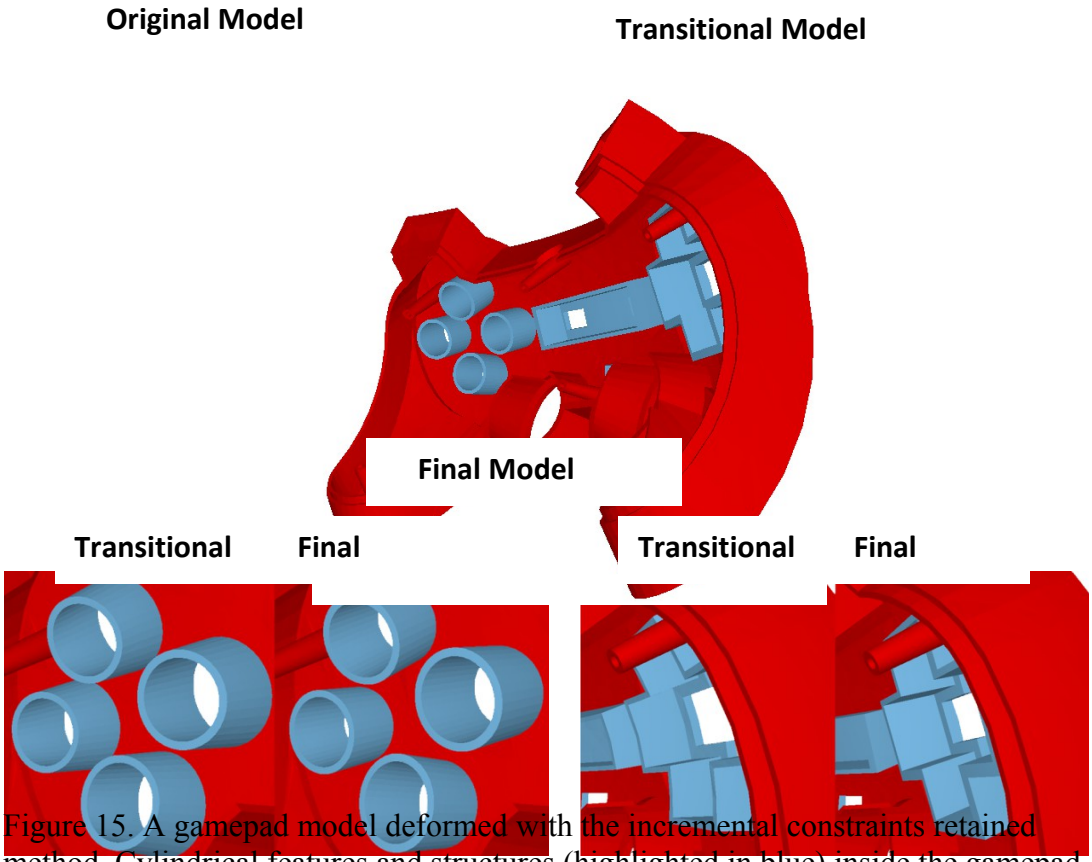


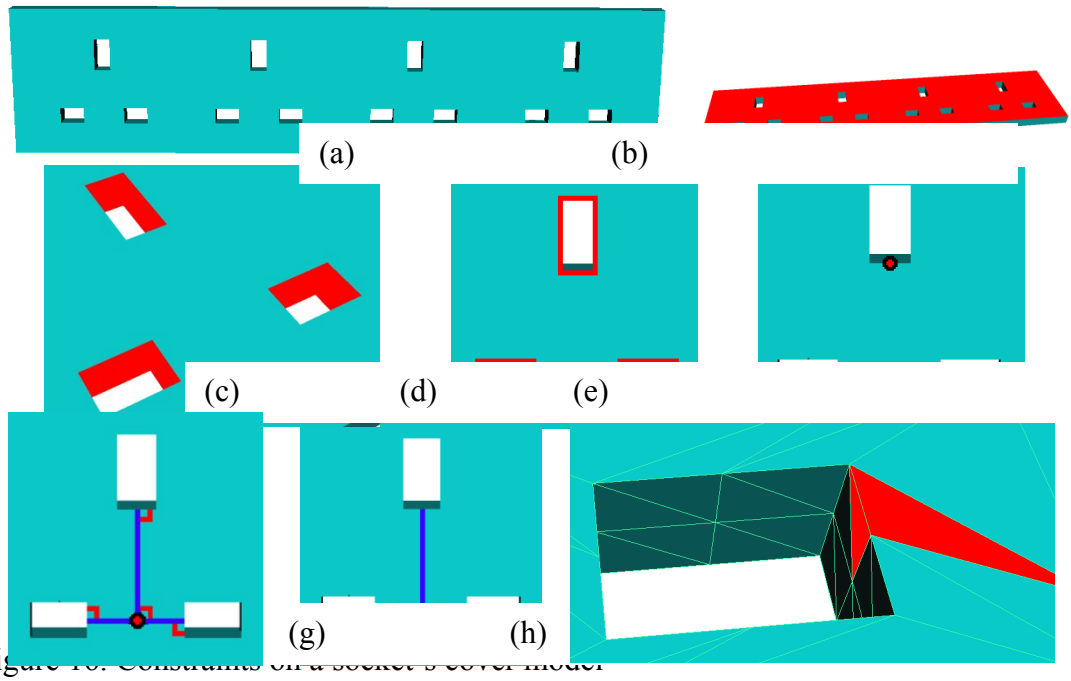
Figure 15. A gamepad model deformed with the incremental constraints retained method. Cylindrical features and structures (highlighted in blue) inside the gamepad model are retained after the deformation.

5.2 Maintain Pattern between Engineering Features

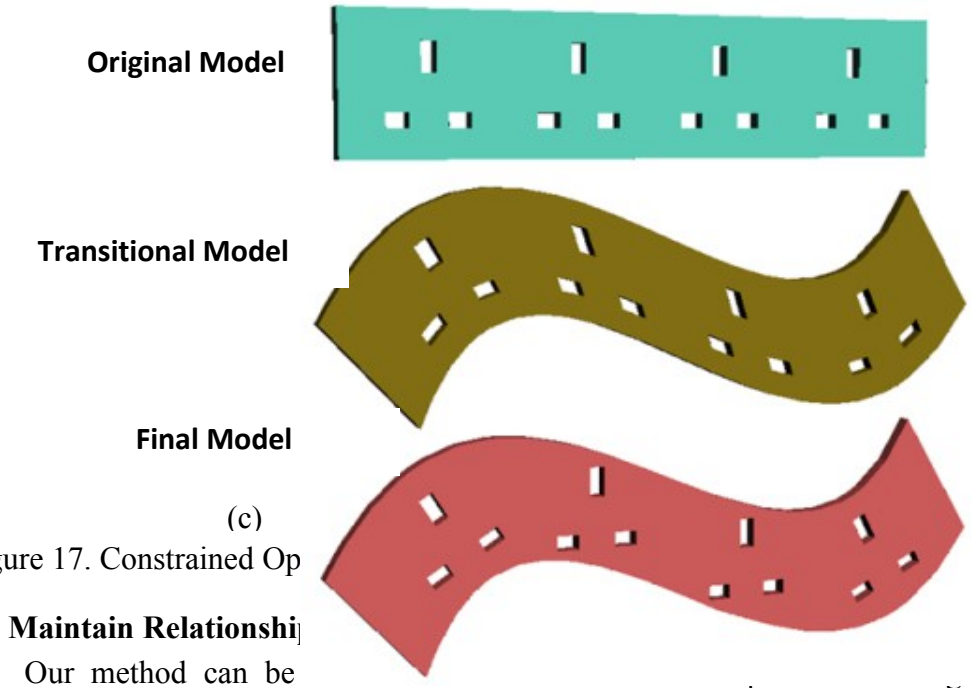
Our method can retain the pattern of engineering features in a deformation. By using a combination of parallel, perpendicular, planar, collinear and circular constraints, pattern between engineering features can be retained.

In Figure 16, constraints are applied to the structures of the socket's cover on the original model, and by providing the transitional model, the target features can be retained on the final model shown in Figure 17. The original model of the socket's cover is shown in Figure 16(a). In Figure 16(b), all the triangles on the top face are constrained with coplanar constraints. In Figure 16(c), all the triangles on the inner vertical wall of the hole are constrained with coplanar constraints too. In Figure 16(d), the angles and lengths of the rectangles for each socket are constrained with distance and angle constraints. In Figure 16(e), a reference point is added at the mid-point on the side of the socket's hole as indicated in the figure. Distance and angle constraints are also required to maintain the reference points to lie at the mid-point of the

corresponding edge. For each socket, another reference point is added at the mid-point between two edge reference points. The angles between the edges and the lines connecting the reference points are constrained as shown in Figure 16(f). Distance constraints are applied as shown in Figure 16(g). Finally, each pair of perpendicular triangles is constrained with perpendicular constraints as shown in Figure 16(h). For each hole, 4 pairs of these constraints are required to constrain the shape of hole without resulting in redundant constraints.



Fi



(c)

Figure 17. Constrained Op

5.3 Maintain Relationshi

Our method can be eering features, e.g. the relationship between components in a mechanism is constrained such that the functions of the mechanism is retained after being deformed. The method is tested on two gear train examples. Our target is to maintain the gear train to be functional after the deformation and optimization. The original undeformed model of a simple gear train is shown in Figure 18. The gear train is constructed with 3 gears of different diameters. Figure 19 illustrates how to constrain the shape of one gear

model, and how to make these gears connecting together. First, 3 reference points ($\acute{v}_1, \acute{v}_2$ and $\acute{v}_3$) are inserted at the centre of the three circular planes. These reference points are used to construct radius distance constraint and angle constraints for the circular shape of the gear planes (Figure 19(b)). Second, 3 reference vectors ($f\acute{n}_1, f\acute{n}_2$ and $f\acute{n}_3$) are inserted to represent the normal to these three circular planes (Figure 19(c)). Third, $\acute{v}_1, \acute{v}_2$ and $\acute{v}_3$ are constrained to be collinear. The angle between $f\acute{n}_1$ and $f\acute{n}_2$ is zero, and the angle between $f\acute{n}_1$ and $f\acute{n}_3$ is 180 degrees. The constrained angle between $f\acute{n}_3$ and $b\acute{v}_1$ is zero degrees (Figure 19(d)). This constraint is applied for the whole lower cylindrical features. Then, 2 reference points ($\acute{v}_7$ and $\acute{v}_8$) are added at the contact point of the two gears for upper and low plane (Figure 19(e)). Reference vector are created from the centre reference points ($\acute{v}_1, \acute{v}_3, \acute{v}_4$ and $\acute{v}_6$) to the reference gear connecting points ($\acute{v}_7$ and $\acute{v}_8$) (Figure 19(f)). After that, $c\acute{v}_1$ is the reference vector that is the cross product of $f\acute{n}_1$ and $r\acute{v}_1$ (Figure 19(g)). By constraining angle between $c\acute{v}_1$ and $c\acute{v}_2$ to be zero, the axes of two gears are confined to lie on the same plane. Finally, in order to maintain the gear ratio between each pair of gears, a constraint is constructed between the radiuses of each pair of gears such that the ratio of the gears' radiuses is determined by their gear ratio.

Figures 20-22 illustrates the transitional model and final model. It can be observed that when the base plane of the model is deformed, the angle of the tooth-bearing faces is changed to maintain the contact relation between the gears. The gear teeth are not included in the examples. This simplified model reduces the computation requirement because of the smaller number of faces and hence smaller number of constraints. In the final gear models, the diameters and angle of bearing-faces of each gear can be obtained. And the user can use these data to design the gear teeth for manufacturing. An example is shown in Figure 23 that the gear teeth are regenerated on the gear train based on the requirement.

A more complex gear train model example is given in Figure 24. The original model is bended downward and used as transitional model in Figure 25. The final model is shown in Figure 26 after the constrained optimization is finished.

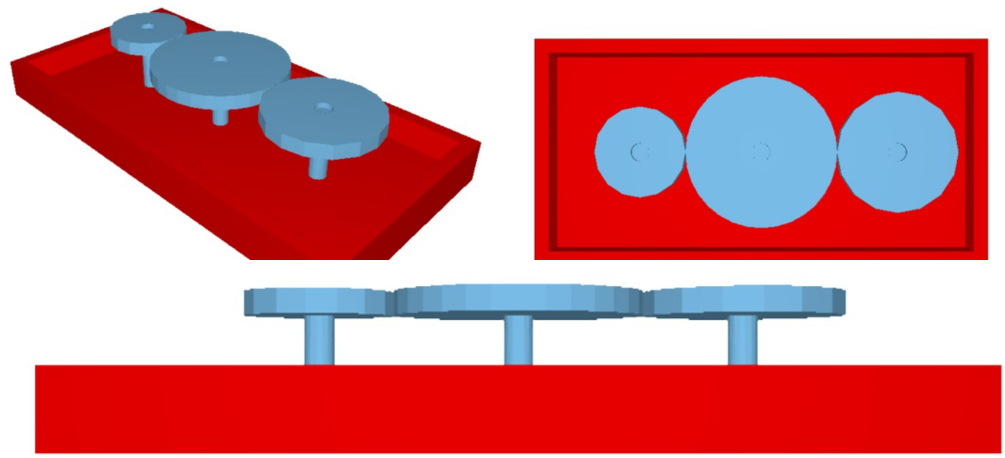


Figure [illegible]: The original model for this gear train model.

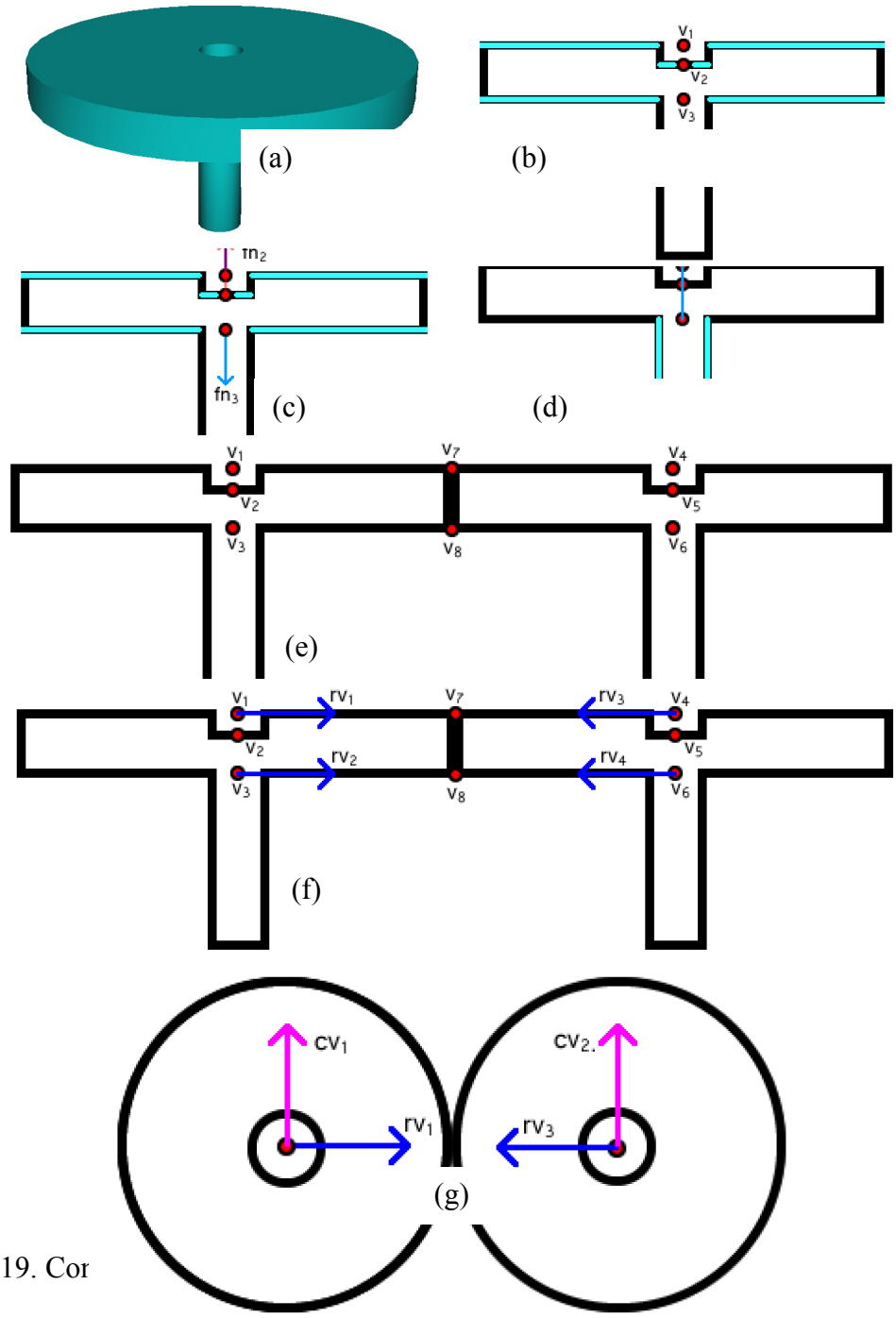
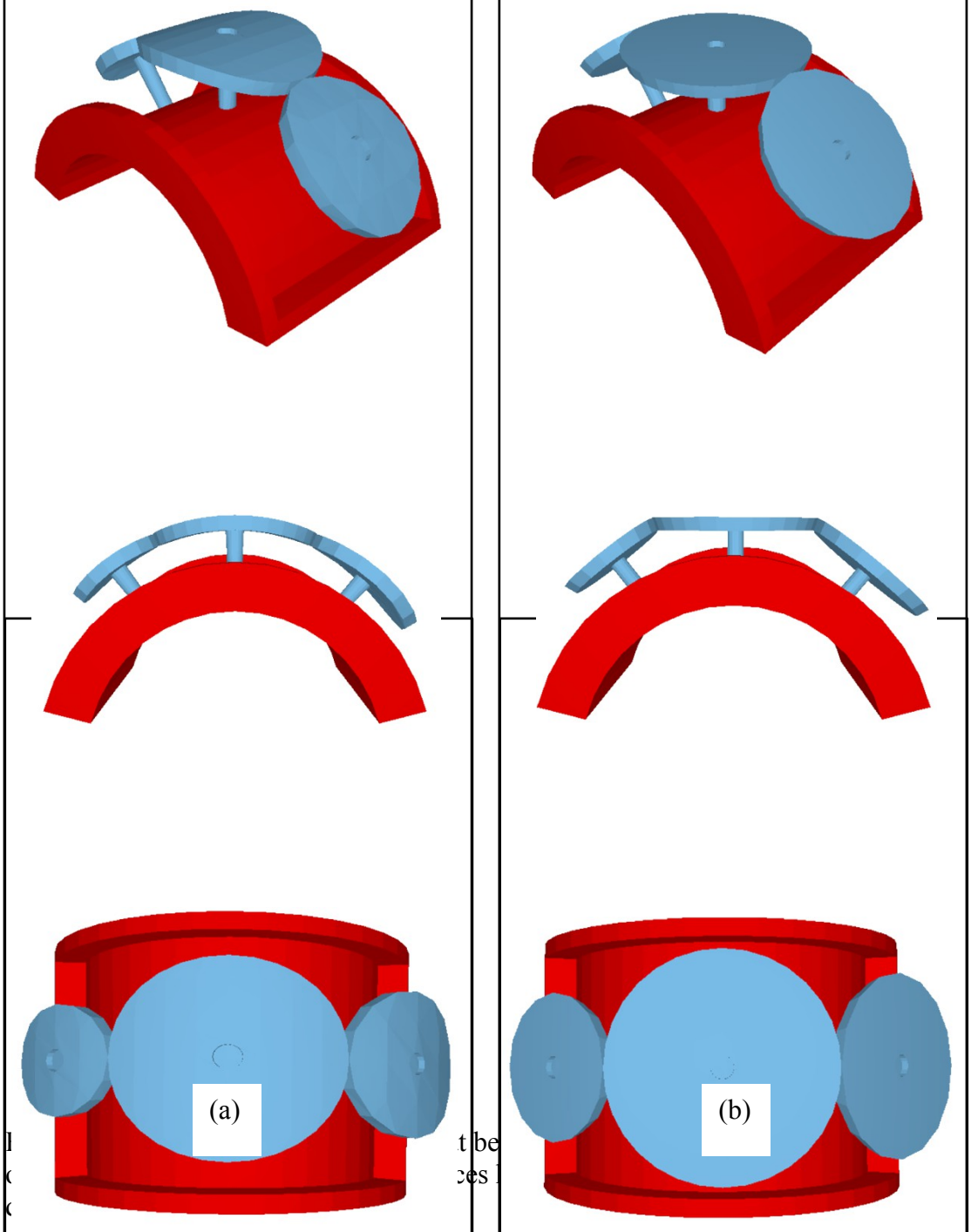


Figure 19. Cor

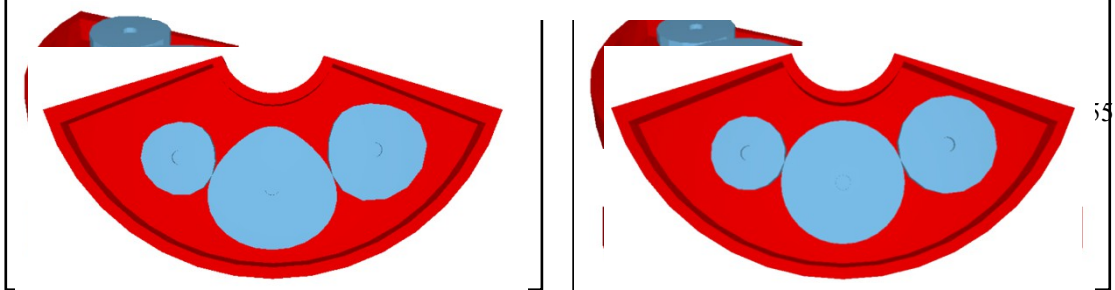
Transitional Model

Final Model



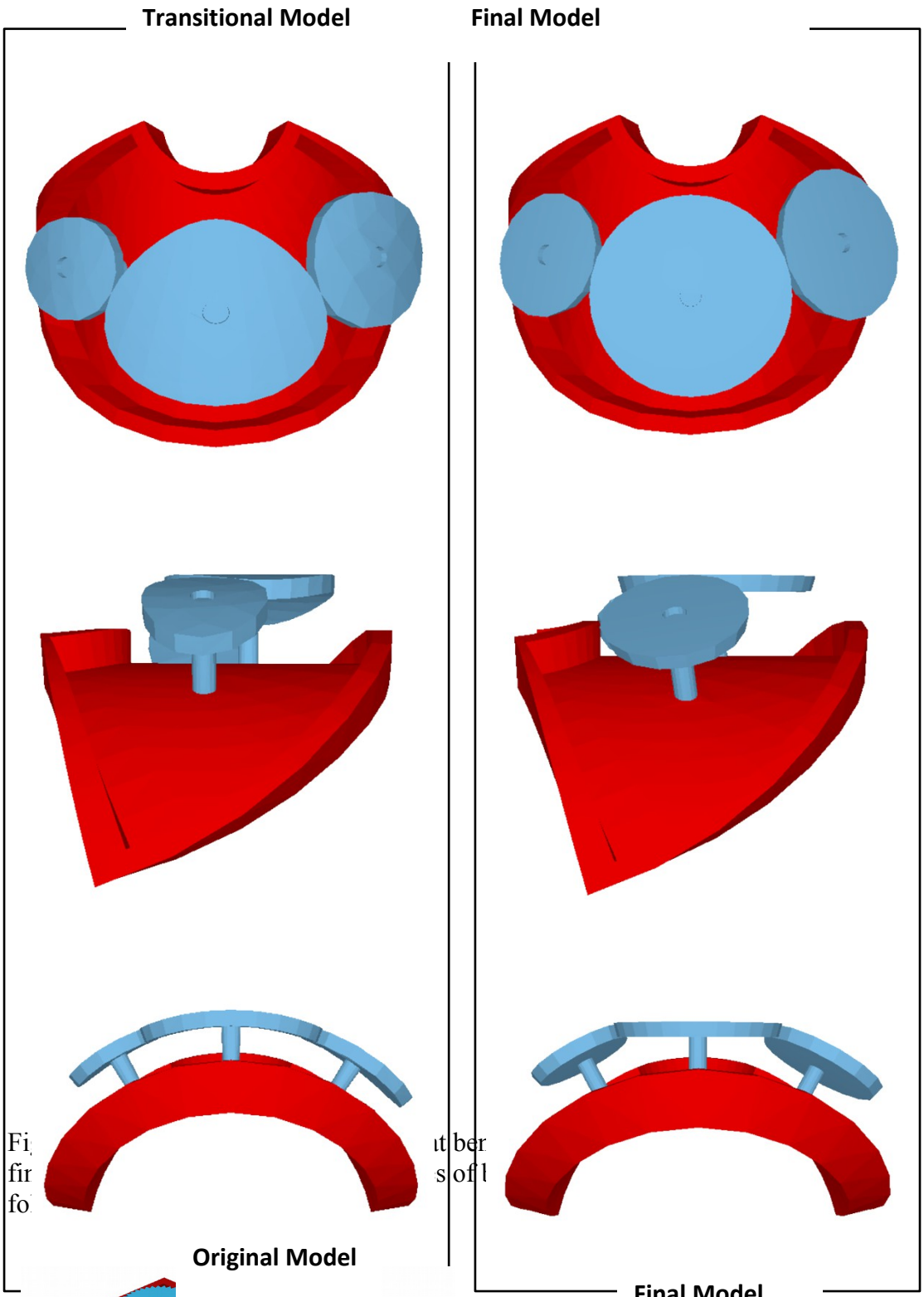
Transitional Model

Final Model



(a) (b)

Figure 21. (a) Th itional model that bended sideway (b) T al model of our method



Fi ıt ber
fir s of l
fo

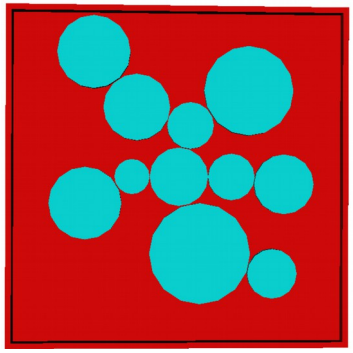
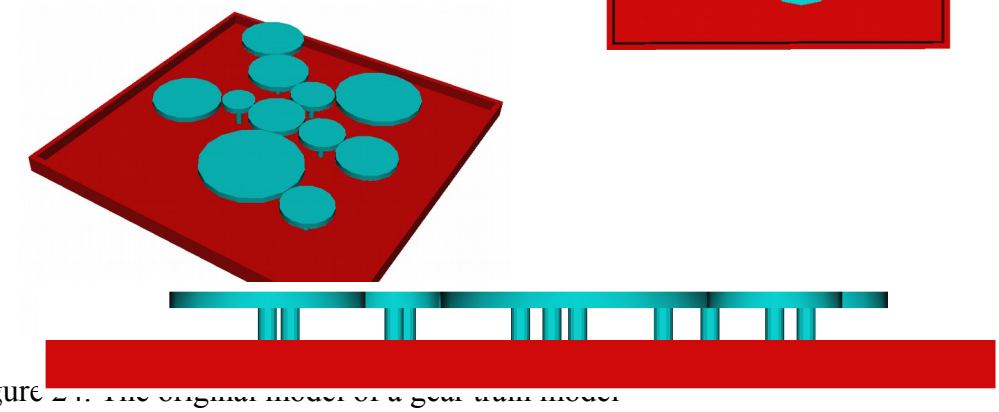


Figure 23. The gear train model with gear teeth



Figure

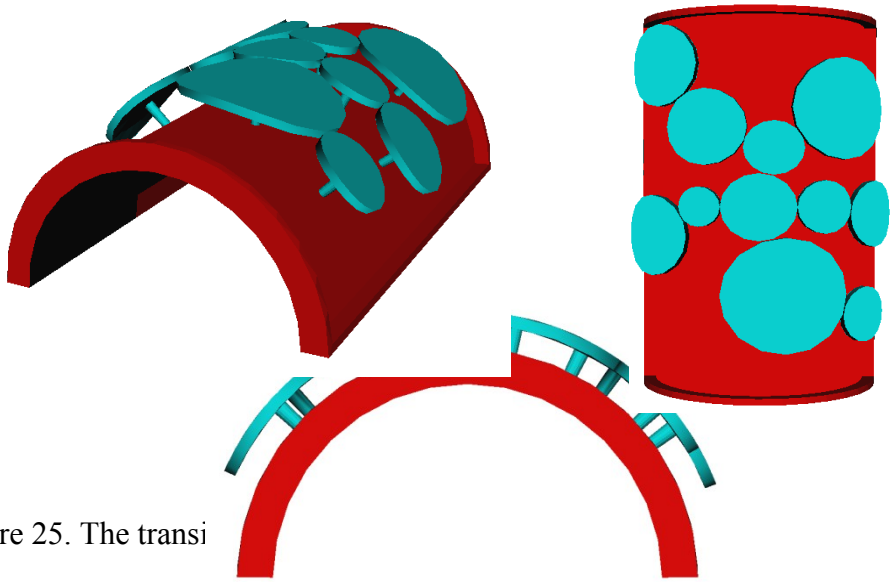


Figure 25. The transi

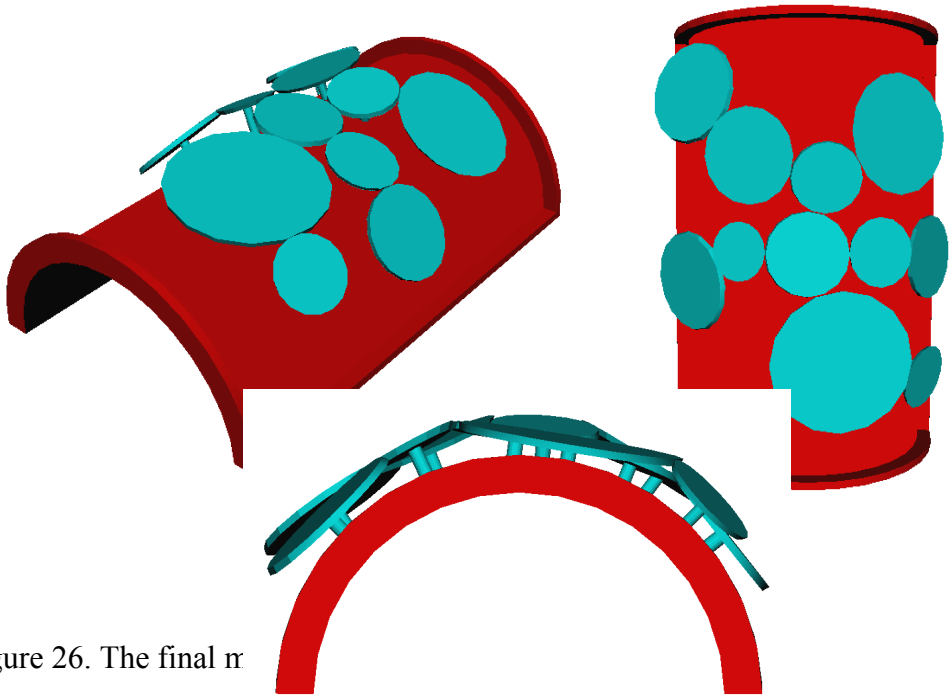


Figure 26. The final m

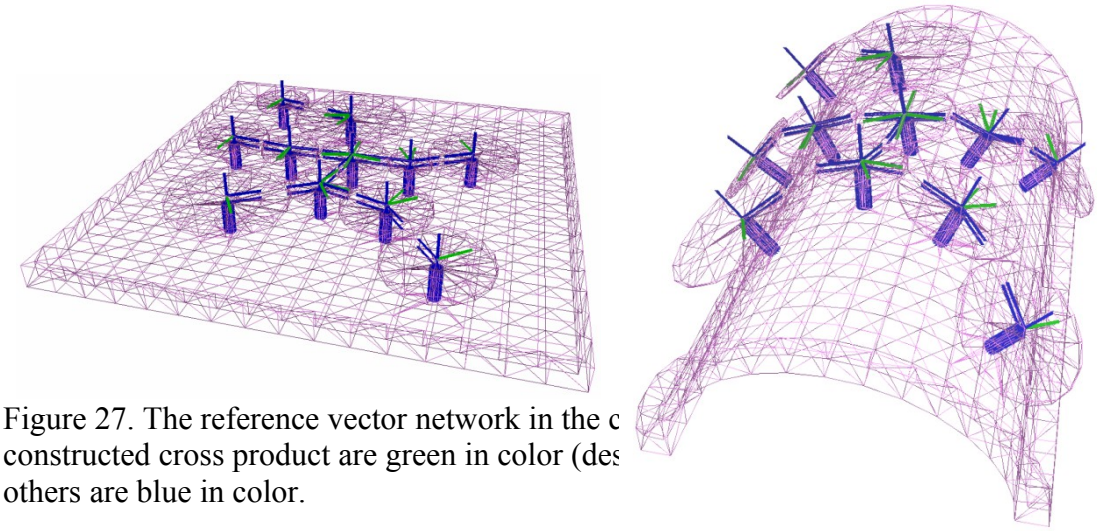


Figure 27. The reference vector network in the c
constructed cross product are green in color (des
others are blue in color.

The method is also tested on a phone casing assembly. Our target is to ensure that the two components of the phone casing can be assembled when the shape of the phone is adjusted. The original undeformed model of the phone casing is shown in Figure 28(a). This phone casing assembly contains two parts. They are the upper-half casing and lower-half casing. Different views of the original model are shown in Figure 29. After a deformation, the transitional model of the phone casing is shown in Figure 28(b). Different views of this transitional model are shown in Figure 31. The final model is shown in Figure 28(c) and Figure 32.

In this example, the eight supporting cylinders on the lower-half casing are constrained to maintain their radiuses and heights. The top faces of each of the four cylinders on both sides are coplanar. This is required for housing other components of the phone properly. The eight circular holes on the upper-half casing are also constrained with the same radius.

Figure 33 shows the matching features of the original model in the cross sectional views of the assembly. After the deformations, the two parts cannot be assembled as shown in Figure 34. In order to ensure that the two components can be assembled, Two reference vectors ($\bar{i}$, $\bar{j}$) are provided in the coordinate space and they are perpendicular to each other as shown in Figure 30. These two vectors are constant vectors with fixed directions. All the face normals of the triangles on plane C are constrained to be parallel to $\bar{i}$. And all the face normals of the triangles on plane A are constrained to be parallel to $-\bar{i}$. This is because these triangle normals are pointing in the opposite direction of $\bar{i}$. And perpendicular constraints are applied to the triangles on plane A and plane B. Also, plane A and plane C are constrained to be coplanar. Same forms of constraints are applied to the other faces on the matching features, but with different reference vectors for constraining the plane normal directions in the original model. In Figure 35, these two constrained parts can be assembled successfully.

Moreover, Figure 36 shows the cross sectional views of the screw holes in the original model. After the deformations, these two parts cannot be assembled as shown in Figure 37. Therefore, these screw holes are constrained with the same radiuses and the upper and lower cylinders are constrained to be concentric. In the final model, these components can be assembled properly.

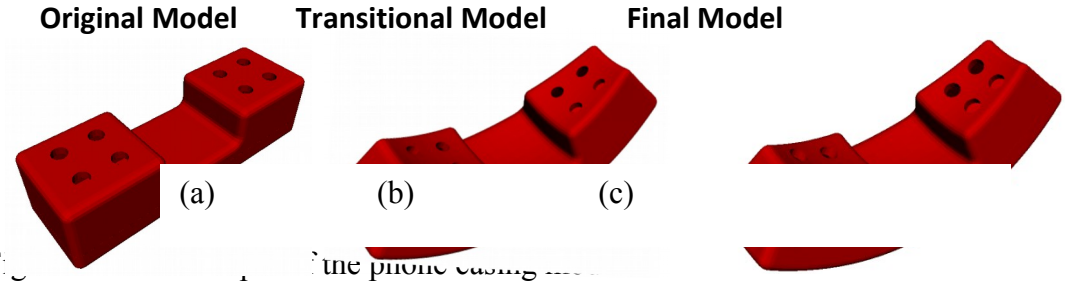


Figure 28. Different views of the phone casing model

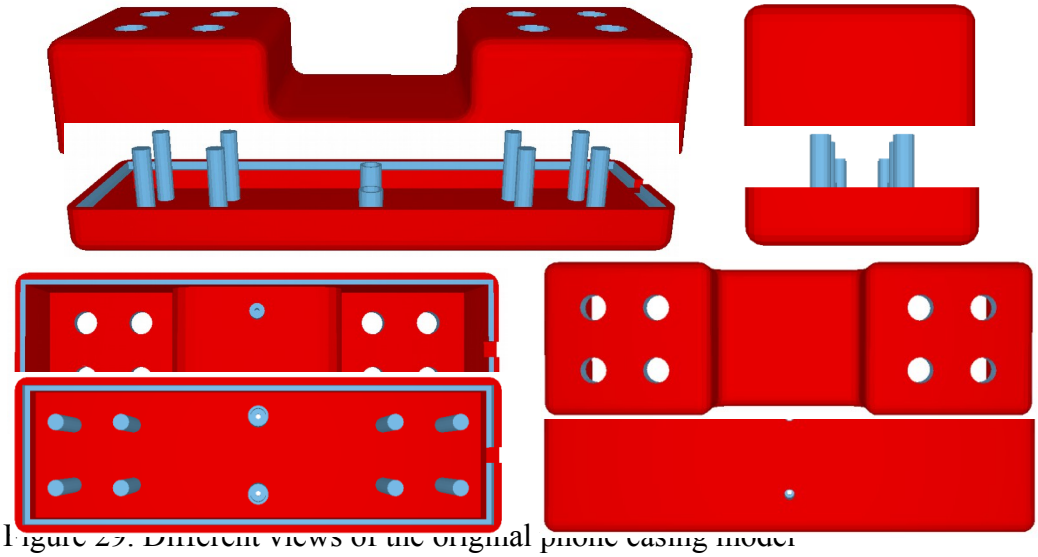
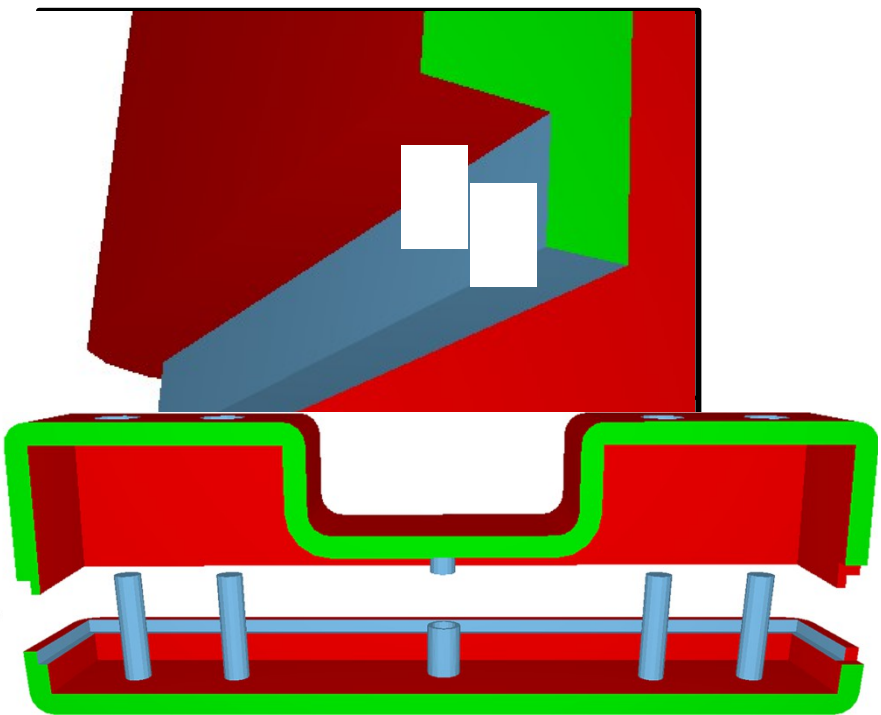


Figure 29. Different views of the original phone casing model



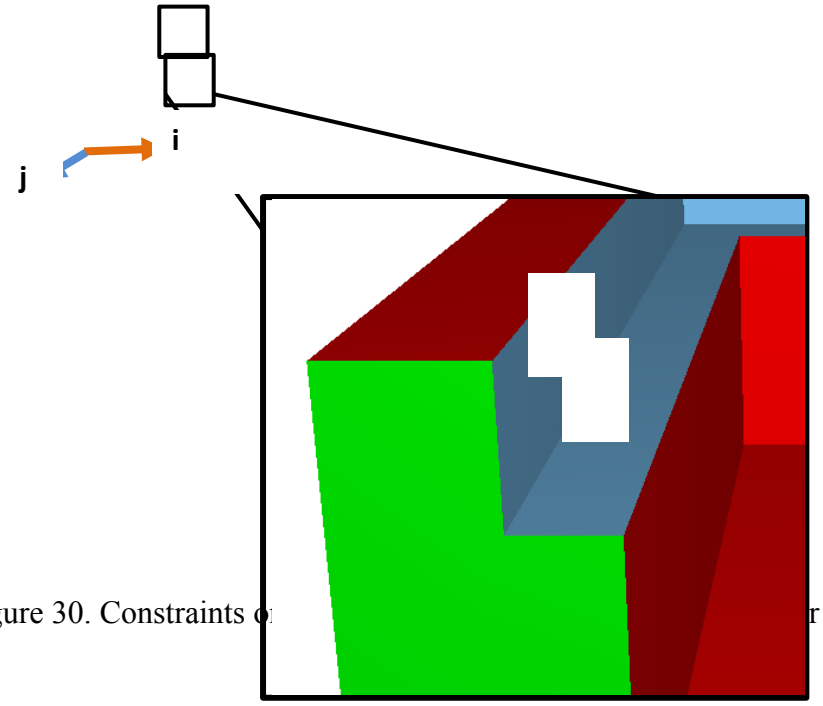
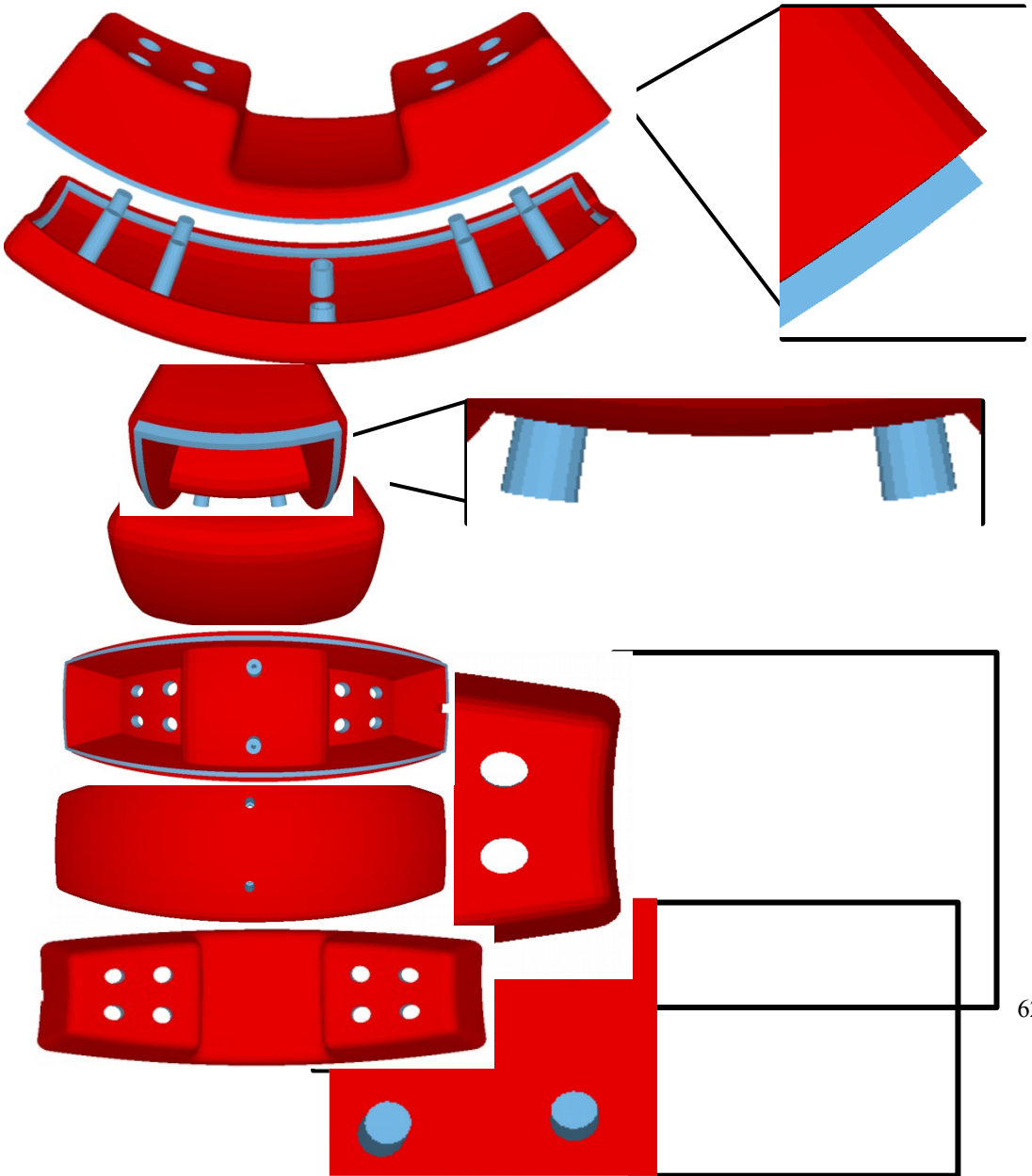
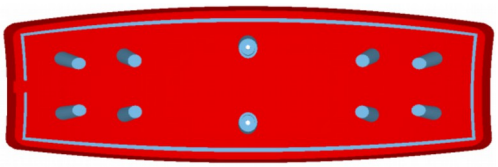


Figure 30. Constraints o r half





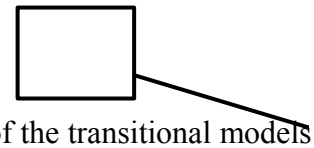


Figure 31. Different views of the transitional models for the phone casing

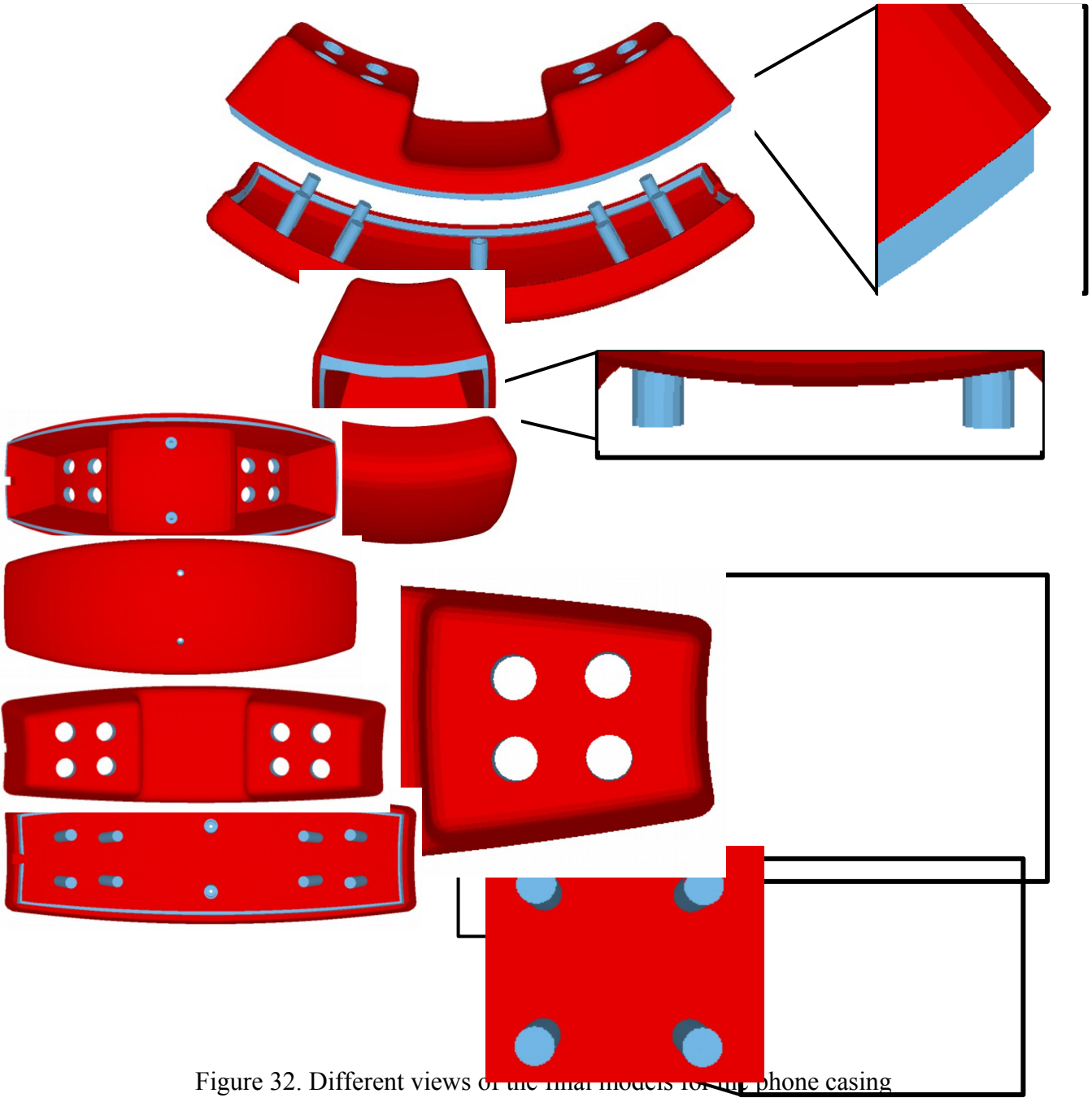


Figure 32. Different views of the final models for the phone casing

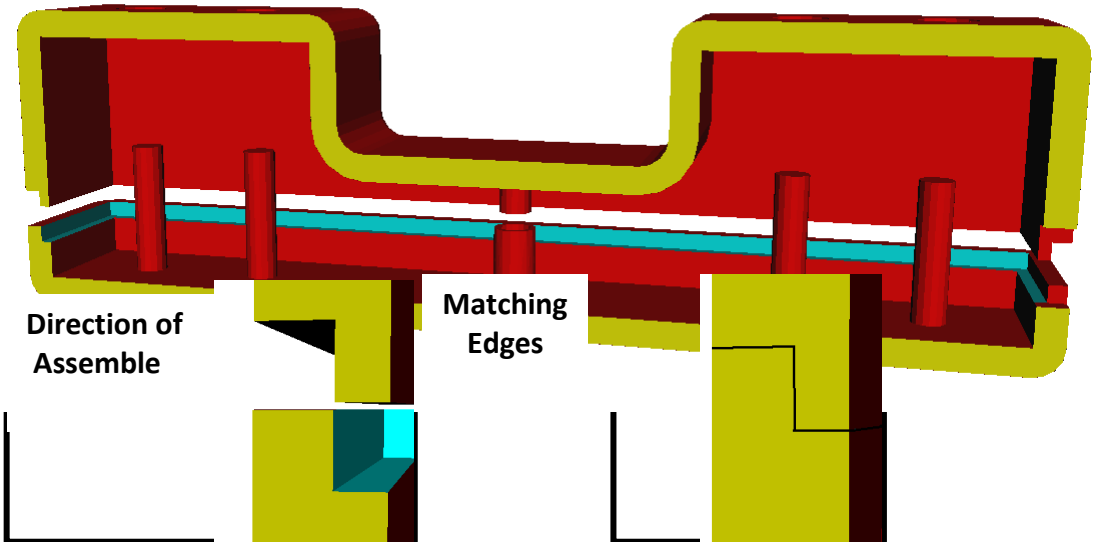
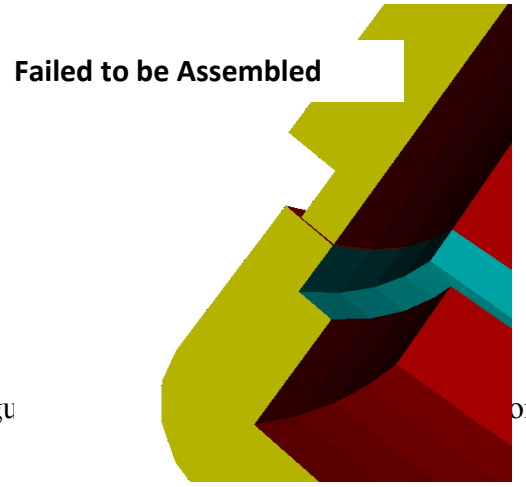


Figure 33. Matching features (highlighted in blue) in the undeformed phone casing model



Figu onal models of the phone casing

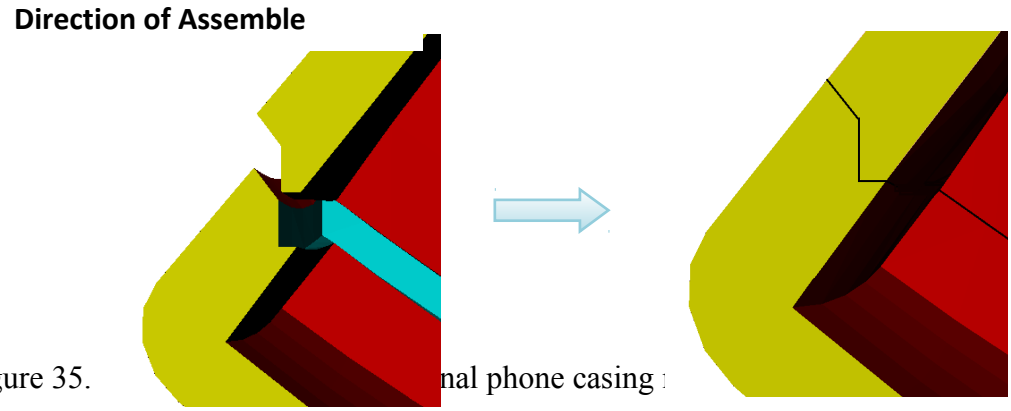


Figure 35. nal phone casing

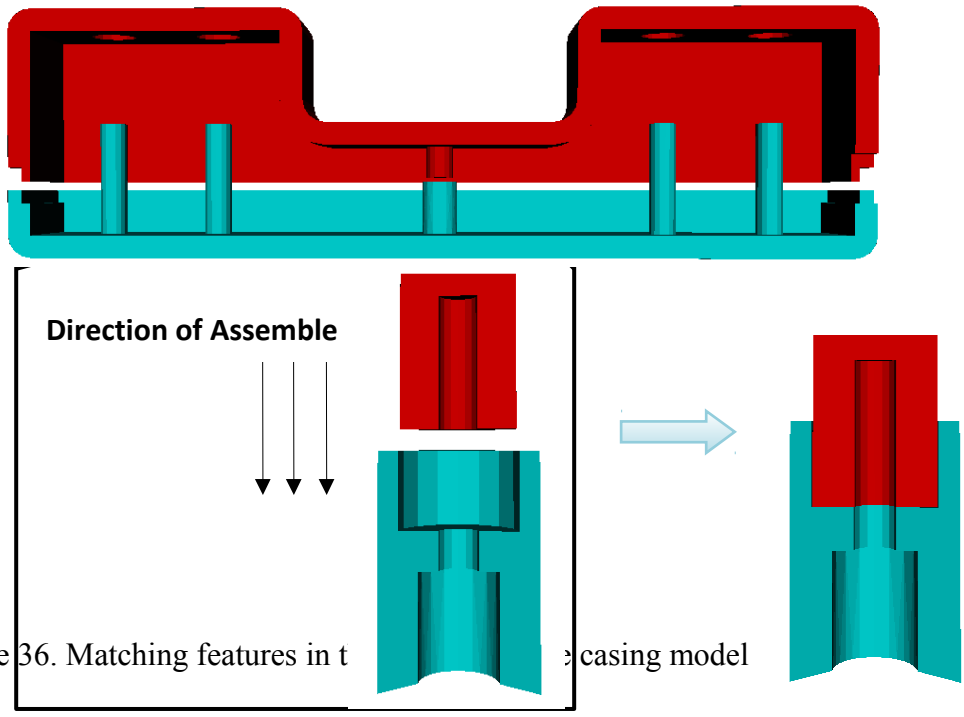


Figure 36. Matching features in t e casing model

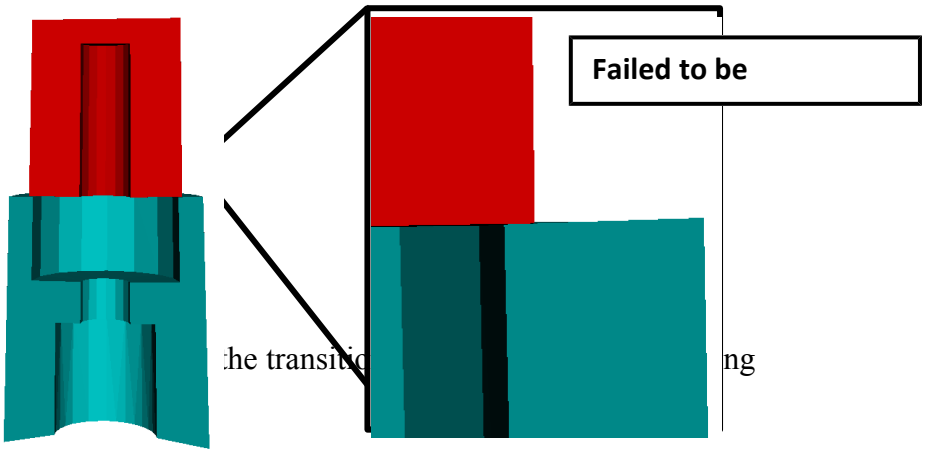


Figure 37. A m :he transiti ng

5.4 Implementation Issue

The experimental system for incremental constraint optimization is implemented on Win7 system using C++ language. KNITRO is used for the optimization process. The KNITRO system has an application programming interface (API) to handle the objective function, constraint functions and all the related variables. It requires the user to provide the type of optimization, formulation of objective function, formulation of constraints functions, bounds of constraints, bound of variables and the initial guess to define the optimization problem.

Moreover, it also requires the Jacobian matrix of the constraints and the Hessian Matrix of the Lagrangian function in sparse form to reduce memory usage. The KNITO system provides some build-in functions to approximate these two matrixes. However, in order to obtain more accurate Jacobian and Hessian matrixes that is required for speeding up the optimization; the current experimental system provides exact Jacobian and Hessian matrixes for each optimization problem. In the optimization process, the system will automatically compute the position of the nonzero elements in the Jacobian Matrix and Hessian Matrix, so that a sparse matrix is obtained for the computation. During the optimization, we need to provide the value of the nonzero elements in these matrixes that are constructed with the current value of the variables in each iteration.

In order to further improve the speed of process, the vertices that do not relate to any applied constraint are not included in the optimization. That is, the locations of the unconstrained vertices are directly obtained from the coordinates of the

corresponding vertices on the transitional model and are not determined in the optimization process.

6. TESTS AND RESULTS

In this chapter, experiments on the effects of constraints with a common reference, different level of model detail and incremental method are discussed. And some comparisons between our method and other deformation methods are also included.

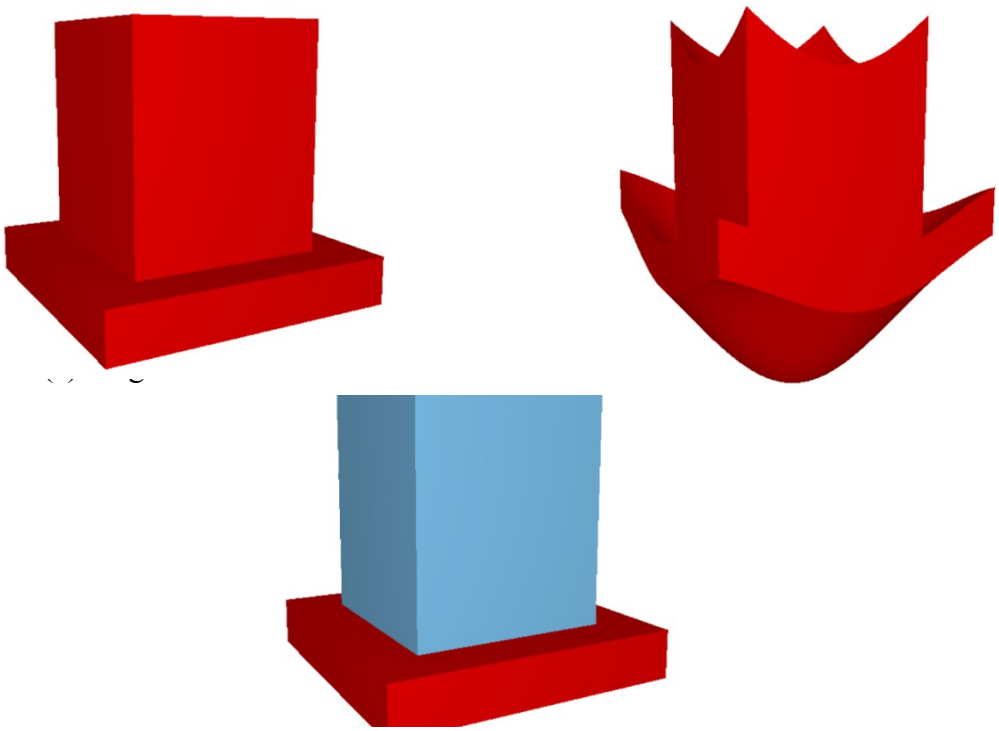
6.1 Constraints with References

In this section we will discuss the effect of using referenced constraint rather than local constraints. We performed an experiment with the original model shown in Figure 38(a), and we want it to be deformed into the target model shown in Figure 38(b). And we also want to retain the cubical feature shown in Figure 38(c).

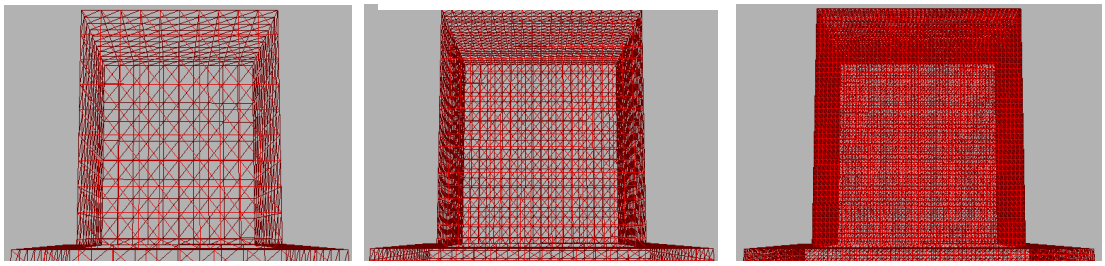
By applying coplanar constraints on faces of a plane, perpendicular constraints between planes and collinear constraint along the edges of a plane, we perform optimization at different resolutions. They are 10x10x10, 20x20x20 and 50x50x50 for the cubical feature shown in Figure 38(d-f). For the 10x10x10 case, it took 152.4 seconds to complete the process. And for the 20x20x20 case, it took 645.7 seconds to complete. For the 50x50x50 case, the solver failed to locate an optimal solution, and terminated at a point that the collinear constraints were still not satisfied. Hence, by increasing the number of constraints of the same type, the optimization will have higher chance to fail. Details of the test are given in Table 3.

Model	Num. of vertices	Num. of faces	Total Num. of constraints	Time (s)
Cube in Fig 38d	770	1536	1546	152.4
Cube in Fig 38e	3018	6032	5991	645
Cube in Fig 38f	17979	35952	31963	Failed

Table 3. Time required for constrained optimization without the use of reference



(c) Constrained Part



(d) 10x10x10 for the cubic (e) 20x20x20 for the cubic (f) 50x50x50 for the cubic

Figure 38: Constrained Optimization for a cube model

It is expected that the numerical errors arising from a pair of local constraint may accumulate during the optimization. When these numerical errors accumulated to certain level, the solver may iterate into an infeasible region, and the iteration does not converge.

Based on this observation, we performed tests that use a global reference for setting up constraints on the triangles. Instead of confining adjacent triangle normal to be the same, all the triangle normals in same plane are confined to be the same as one reference normal vector. The models and the deformations in Figure 38 are performed

with these new set of constraints. The statistics of the experiment results are shown in Table 4.

By replacing the old collinear constraints with these new collinear constraints, and performing the experiment again, the optimization can be completed successfully, and the optimal solution is located. This shows that the new collinear constraints provide a significant effect on the optimization process.

Model	Num. of vertices	Num. of faces	Total Num. of constraints	Time (s)
Cube in Fig. 38d	770	1536	1495	101.2
Cube in Fig. 38e	3018	6032	5743	354.6
Cube in Fig. 38f	17979	35952	31457	946.2

Table 4. Time required for constrained optimization with the use of reference

Hence, the use of reference is preferred in setting up structural constraint that is built from a group of basic constraints. Structural constraints that can be implemented with this method include planar, collinear and circular constraints.

6.2 Level Of Detail

An experiment is performed on a simple gear train model (Figure 38) to the model is to be deformed into the shape of transitional model shown in Figure 20. Tests are conducted with different resolutions of the gear bodies and the statistics of the experiment results are shown in Table 5.

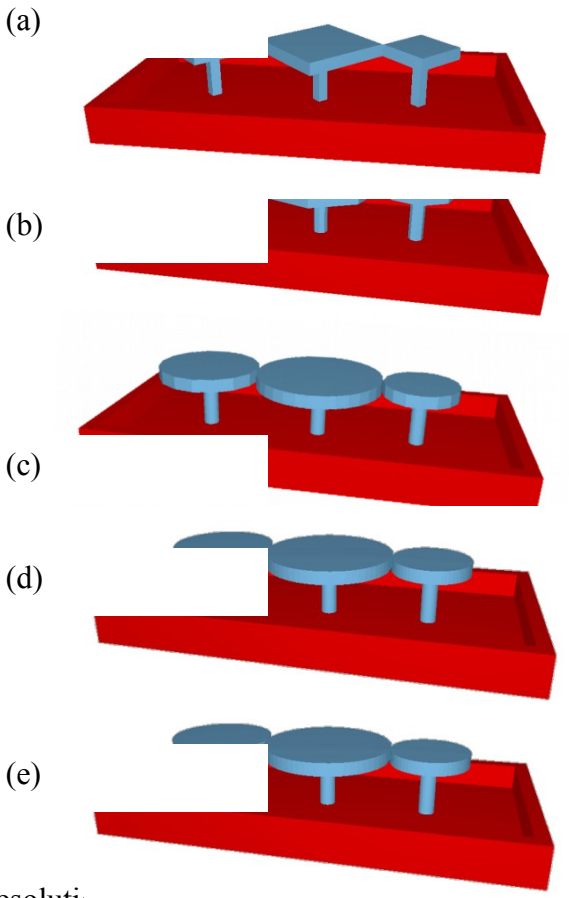
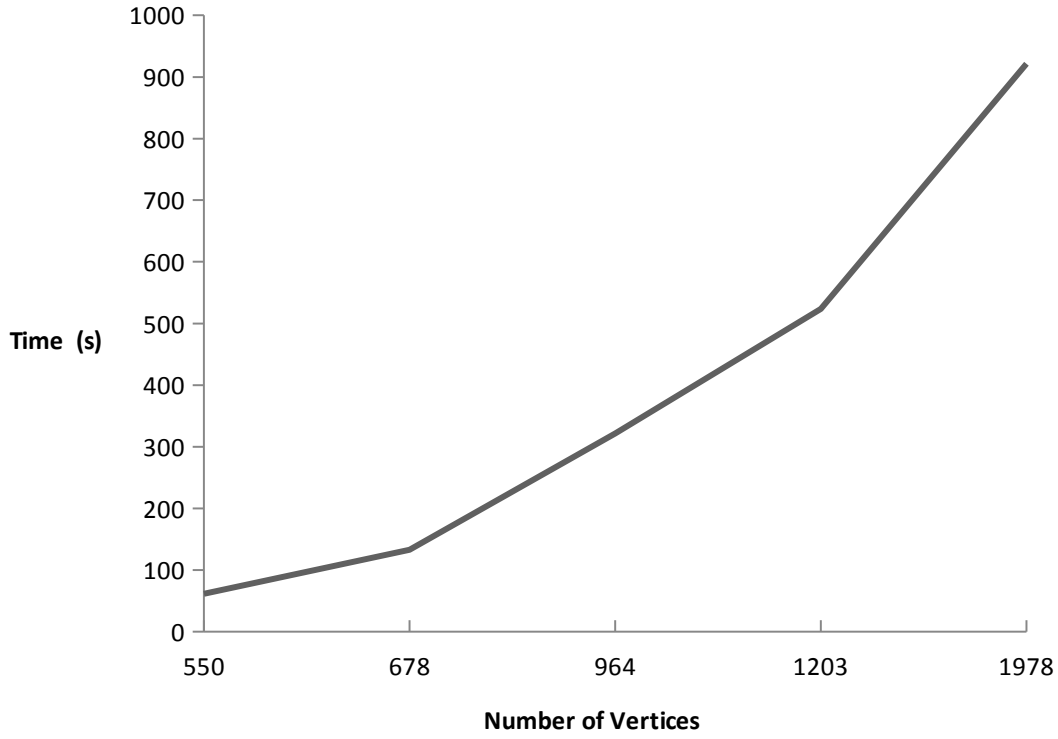


Figure 39. Different resolutions for a gear train model

Table 5. Time required for constrained optimization with different level of resolutions on the gear train model

Model	Num. of vertices	Num. of faces	Total Num. of constraints	Time (s)
Gear train in Fig. 39a	550	925	524	61.5
Gear train in Fig. 39b	678	1378	1029	132.6
Gear train in Fig. 39c	964	1911	1924	321.4
Gear train in Fig. 39d	1203	2710	2142	523.6
Gear train in Fig. 39e	1978	3421	2810	921.6

Graph 1. Number of Vertices and Computational Time



From Table 4, Table 5 and Graph 1, it can be observed that a higher resolution model will require more computation time for the optimization. This is because more

constraints and more variables are included in the problem setting of the higher resolution case. For some engineering feature with standard set of constraints pattern such as gear model, we can just simplify the model to one with a lower resolution, and which is then used in the deformation. After the optimization, we can extract useful information from this simplified optimization. In the gear train model, the angle of tooth-bearing and radius of the gears can still be extracted from the lower resolution model which is sufficient for defining the shape of the gear. This operation will improve the optimization speed of the system and lower the chance of optimization failure.

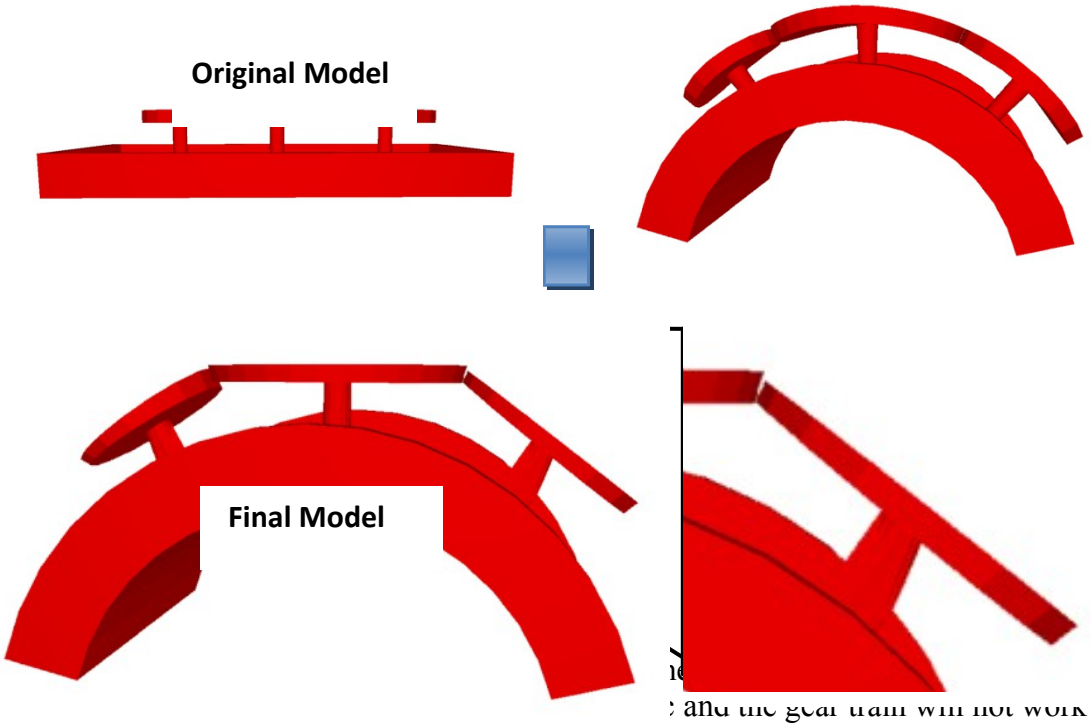
6.3 Incremental Method

Incremental method has a major role in our method. It helps the system by providing guidance for the optimization. When the optimization failed at some point, the system will automatically generate a new interpolated model between the original model and target model. When the system uses this interpolated model as current target model, the provided initial guess is closer to the current target model; hence, the possible solution will become closer to the initial guess. In this situation, the optimization will has higher chance to get to the optimal solution. Several tests are performed to demonstrate the effect of incremental deformation. Figure 40 shows an unsuccessful optimization result without using the incremental method. The experiment is performed again with the incremental method, and the successful results are shown in Figures 41-42.

Table 6 summarizes the statistics for the demonstrated models. The result shows that the computational times increase with the number of vertices, faces and constraints. Besides, the more the difference between the original model and transitional model, the more the computation time it takes.

Table 6. Number of incremental iteration and the time required for the constrained optimization of different models

Model	Num. of vertices	Num. of faces	Num. of distance constraints	Num. of angle constraints	Num. of planar constraints	Total Num. of constraints	Num. of incremental iteration	Time (s)
Lego in Fig. 9	712	1420	80	160	882	1122	1	25.2
Lego in Fig. 14a	353	702	64	132	406	602	1	20.1
Lego in Fig. 14b	353	702	64	132	406	602	1	22.4
Lego in Fig. 14c	353	702	64	132	406	602	1	24.9
Gamepad in Fig. 15	7922	15891	322	1030	5546	6898	3	526.2
Socket in Fig. 17	866	1776	40	259	1428	1727	1	63.2
Gear train in Fig. 20	964	1911	112	216	1598	1926	3	546.1
Gear train in Fig. 21	964	1911	112	216	1598	1926	3	677.0
Gear train in Fig. 22	964	1911	112	216	1598	1926	3	691.3
Gear train in Fig. 24	1613	3222	902	1007	1645	3554	3	1134.1
Phone case in Fig. 28	4772	9576	928	1671	2852	5451	3	925.6



properly.

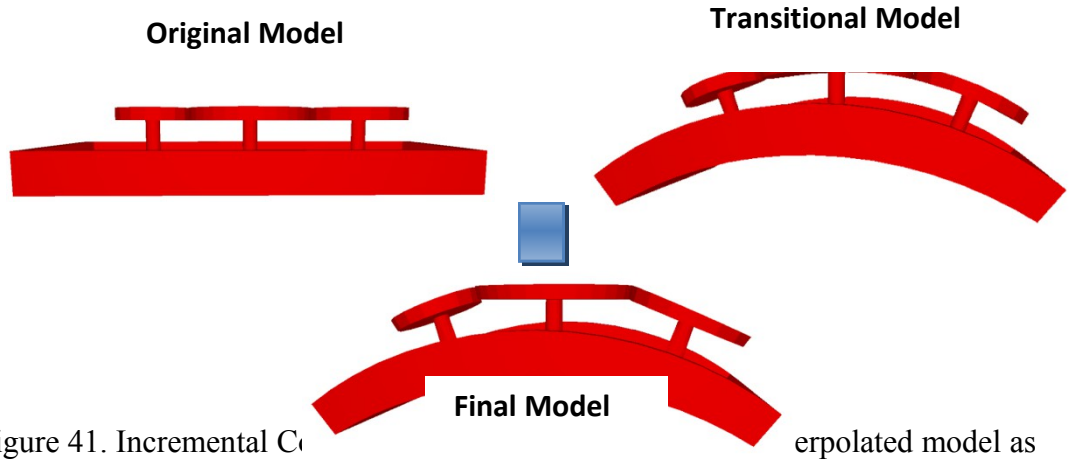


Figure 41. Incremental C erpolated model as new target model

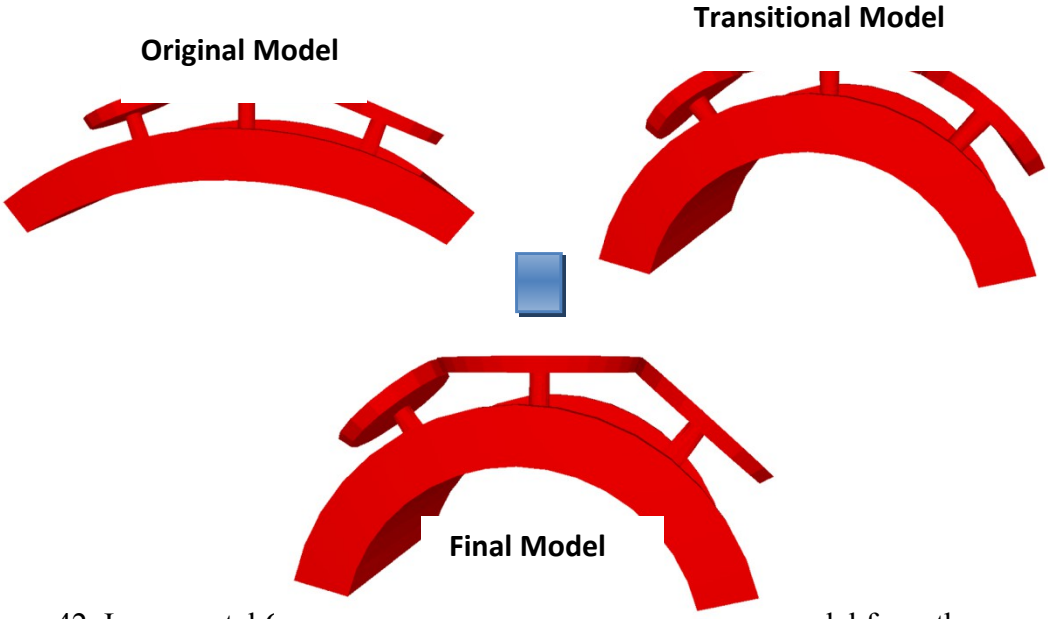


Figure 42. Incremental C odel from the incremental previous optimization as initial guess for the optimization.

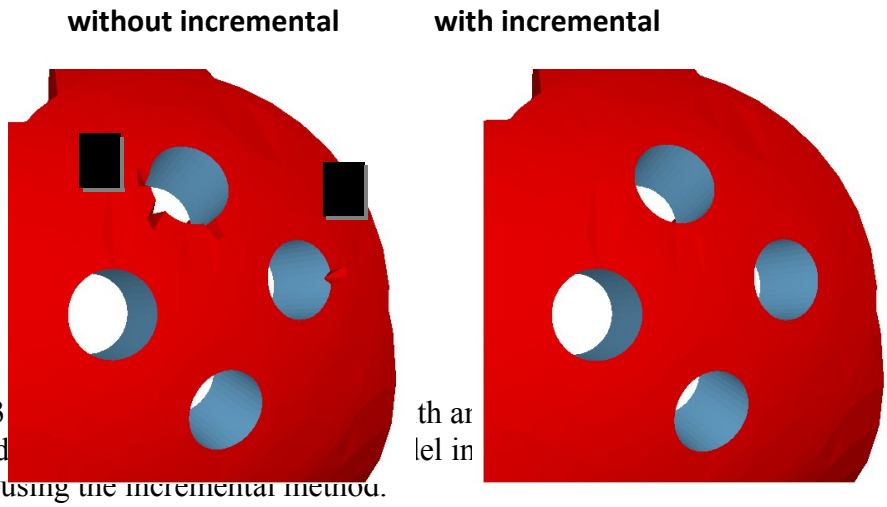


Figure 43 th ar s retained d lel in obtained using the incremental method.

6.4 Comparison

In this section, we give some comparisons between our method and other deformation methods. In Figure 44, we give a comparison on the stretching effect of a model. The original model is shown in Figure 44(a). By applying simple space deformation (2x2x2 FFD), the result in Figure 44(b) showed that the shape and dimensions of the cylinder cannot be retained. In Figure 44(c), by applying the linear rotation-invariant coordinates on mesh representation presented in [12] which is a detail preserving technique, the result showed that the shape and dimensions of cylinder cannot be preserved too. In Figure 44(d), the result of our method showed that it preserves the exact shape and dimensions of the cylinder while the block is stretched.

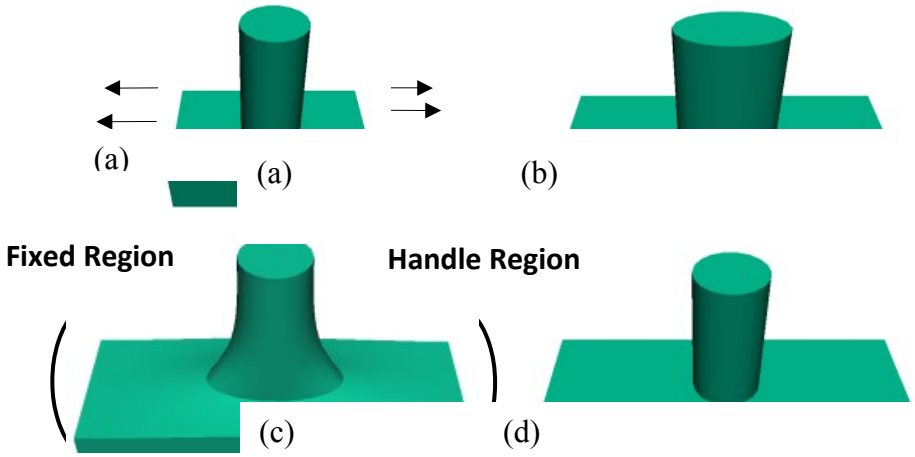


Figure 4((c)) The original model and the stretching direction (b) The result of 2x2x2 FFD (c) The result of applying the linear rotation-invariant coordinates on mesh representation (d) The result of our method

In Figure 45, a comparison on the bending effect on a cylinder of a model is presented. The original model is shown in Figure 45(a). By applying simple space deformation (2x2x2 FFD), the result in Figure 45(b) shows that the shape and dimensions of the cylinder cannot be retained. In Figure 45(c), by applying linear rotation-invariant coordinates on mesh representation, fixing the surrounding faces of the block, and using the top circular face of the cylinder as the handle, the result showed that the dimensions of the cylinder cannot be preserved. In Figure 45(d), by applying the same detail preserving technique and fixing all faces of the block, and using the top circular face of the cylinder as the handle, the result showed that the shape and dimensions of cylinder cannot be preserved too. In Figure 45(e), the result of our method showed that it generates the elliptic intersection between the cylinder and the base, and maintains the dimensions of the cylinder. Moreover, the iWIRES method does not allow bending on the constrained features.

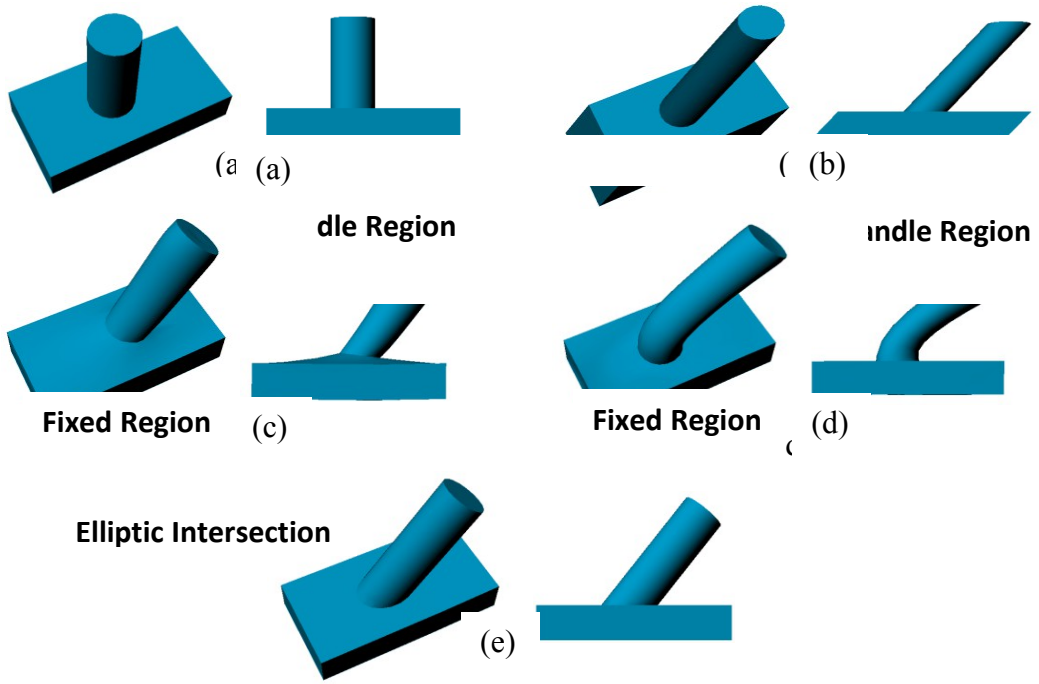


Figure 45. Comparison between our m and other deformation methods. (a) The original model (b) The result of simple space deformation (2x2x2 FFD) (c) The result of including linear rotation-invariant coordinates in the mesh representation (d) Another result of using linear rotation-invariant coordinates in the mesh representation (e) The result of our method

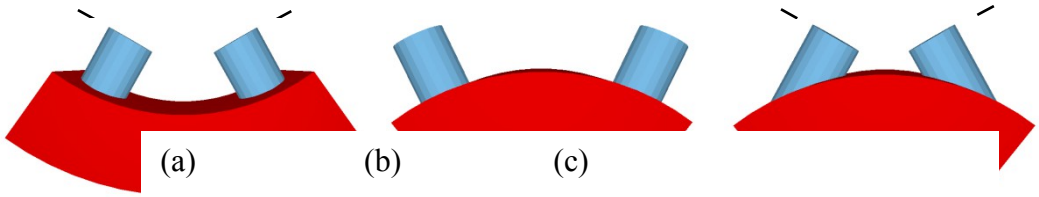
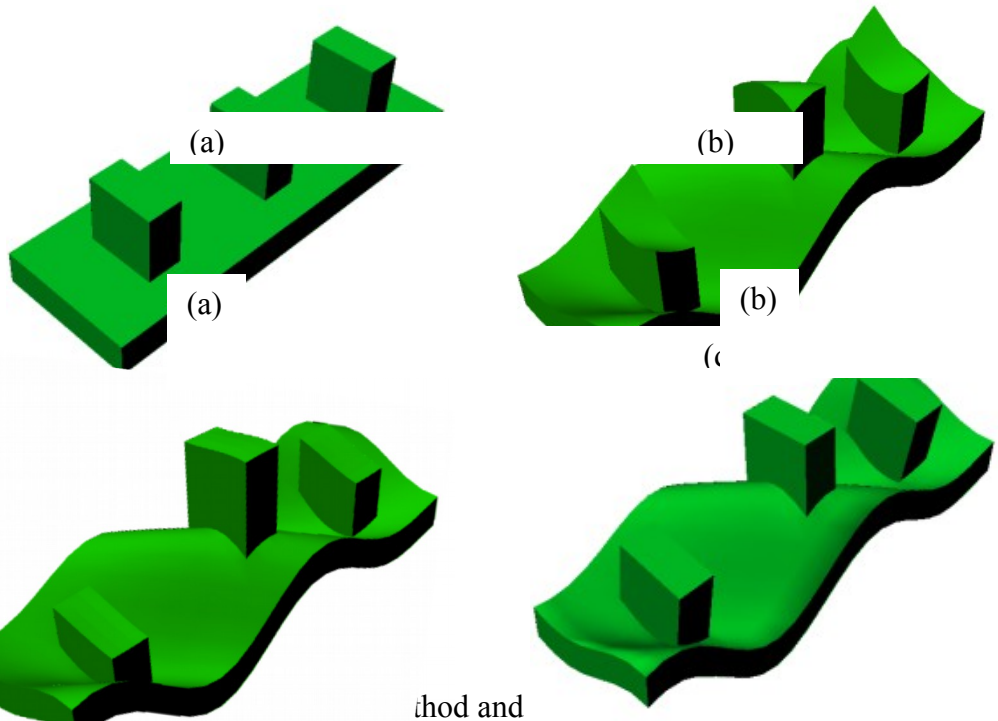


Figure 46. Comparison between our method and iWIRES. (a) The original model (b) The result of iWIRES (e) The result of our method

In Figure 46, a comparison on the ability to retain the angle constraint between the top faces of two cylinders is presented. The original model is shown in Figure 46(a). By bending the object downward, the result of iWIRES is shown in Figure 46(b). It shows that the angle between the top face of the two cylinders cannot be retained, because iWIRES cannot provide angle constraints between faces and all its constraints are constructed through the model edges. In Figure 46(c), the result of our method shows that it retains the shape of the cylinders and it c preserves the angle between the top faces of the two cylinders.



Fi thod and
original space deformation (20x2 FFD) (c) The
result of applying th variant coordinates on me presentation (d)
The result of our me

In Figure 47, a comparison with the free form deformed model is shown. The original model is shown in Figure 47(a). By applying simple space deformation (20x20x20 FFD), the result in Figure 47(b) shows that the shape and dimensions of the cylinder cannot be retained. In Figure 47(c), by fixing the vertices of the waving lower block, preserving the surface detail of those three blocks, and applying linear rotation-invariant coordinates on mesh representation, the result shows that the shape and dimensions of those three blocks cannot be preserved. In Figure 47(d), the result of our method shows that it preserves the exact dimensions of the rectangular blocks and retains the waving effect of the base block. Besides, the iWIRES method does not allow free form deformation on the constrained features.

7. FURTHER WORK AND CONCLUSIONS

7.1 Recommendation for Further Work

Laplacian operator is a different surface representation that can be used to preserve surface details in a deformation process. In contrast to the traditional global Cartesian coordinates which only provides the spatial locations of points, Laplacian operator carries information about the local shape of surface, the size and orientation of local details. Defining operations on mesh surfaces try to preserve such a

differential representation results in detail preserving operations. If the Laplacian operator is adopted in the proposed system, the shape or surface detail after a deformation can be retained. This may also be used to obtain a smoother transitional surface between the constrained part and unconstrained part of a model, and which is useful for free-form mesh editing.

In the current system, incremental deformation method has been adopted in the optimization. In each iteration, if the optimization fails, the system will try to generate a new interpolated model which will be used as the target model for the next optimization. However, a failed optimization usually takes longer computation time than a successful optimization. So it would be better if we can find a predefined step size for the optimization that can make sure the optimization will not fail. It means the system will not need to spend times on some optimizations that will fail. This will improve the computational speed of the method, and the qualities of the final models. The predefined step size may be related to the amount of constraints violations on the given target model. By generating an interpolated model with increasing step size, while checking if constraint violations exist on this interpolated model based on the provide constraints, the process can be repeated until the amount of constraint violation reach a pre-defined level. This interpolated model can then be used to perform constraint optimization. The result can be used as the initial guess for the next optimization. In each iteration, the allowed number of constraint violation is increased to allow the system to generate models that are closer to the original transitional model. The optimization process is repeated with increasing number of allowed constraint violation until the interpolated model is to the same as the transitional model. The last incremental optimization result will then give a model closest to the original transitional model and with all constraints satisfied.

Because of the nonlinear property of the constraints adopted in the system, the speed of the current method is relatively slower than common shape preserving deformation techniques that usually assume linear constraints in their problems. In order to improve the computation speed of the system, parallel processing for nonlinear optimization may be useful in this case. This will require breaking down the problem into different parts, and then solving each part with its own optimization process. It will improve the computational speed of the method.

Although this system provides constraints on planes to prevent triangles from overlapping on the same plane, self-intersections of the mesh have not been considered. More constraints are required to avoid self-intersection in the deformation. It is expected that by extending the existing method to 3D, a general approach for preventing self-intersections in a deformation may be obtained.

7.2 Conclusions

In this thesis, we have proposed a framework for retaining feature in a deformation. Firstly, a deformed object is obtained with common deformation techniques such as FFD and axial deformation. Secondly, features defined with parametric expressions are grouped into systems of primitive constraints based on user specification. Finally, features are reconstructed by the use of optimization technique.

Primitive constraints can be distance and angle constraints between points, lines and faces. Besides, a structural constraint constructed from a group of constraints would be better specified relative to a reference datum for all its component constraints. Reference vectors and reference points are used as tools to specific more complex constraints.

Incremental deformation is adopted to eliminate possible collapsing faces and optimization failures. Features are reconstructed by the use of non-linear optimization technique. Besides, this work can be implemented along with common deformation techniques. Examples have been given on maintaining individual engineering features, pattern between engineering features and relationship between engineering features. Finally, analyses have been performed on the factors affecting the performances of our constrained deformation approach.

REFERENCES

[1] Venkat, A. and Sam, A.: Feature-based modeling approaches for integrated manufacturing: state-of-the art survey and future research directions. *International Journal of Computer Integrated Manufacturing*, 8 (6), 411–440, 1995.

[2] Huikang K. Miao, Nandakumar Sridharan and Jami J. Shah: CAD-CAM integration using machining features. *International Journal of Computer Integrated Manufacturing*, 15 (4), 296–318, 2002.

[3] Houa, M. and Faddis, T. N.: Automatic tool path generation of a feature-based CAD/CAPP/CAM integrated system. *International Journal of Computer Integrated Manufacturing*, 19 (4), 350–358, 2006.

[4] Sederberg, T. W., and Parry, S. R.: Free-form deformation of solid geometric models. In *Proc. of ACM SIGGRAPH*, 151–160, 1986.

[5] Coquillart, S.: Extended free-form deformation: A sculpturing tool for 3D geometric modeling. In *Proc. of ACM SIGGRAPH*, 187–196, 1990.

[6] Milliron, T., Jensen, R. J., Barzel, R., and Finkelstein, A.: A framework for geometric warps and deformations. *ACM Trans. Graph*, 21 (1), 20–51, 2002.

[7] Floater, M. S.: Mean value coordinates. *Computer Aided Geometric Design*, 20, 1, 19–27, 2003.

[8] Botsch, M., and Kobbelt, L.: Real-Time Shape Editing using Radial Basis Functions. In *Proc. of Eurographics*, 483–491, 2005.

[9] Ju, T., Schaefer, S., and Warren, J.: Mean value coordinates for closed triangular meshes. *ACM Trans. Graph*, 24 (3), 561–566, 2005.

[10] Angelidis, A., Cani, M.-P., Wyvill, G., and King, S.: Swirling-sweepers: Constant-volume modeling. In *Proc. of Pacific Graphics*, 10–15, 2004.

[11] Von Funck, W., Theisel, H., and Seidel, H.-P.: Vector field based shape deformations. *ACM Trans. Graph*, 25, 3, 2006.

[12] Lipman, Y., Sorkine, O., Levin, D., and Cohen-Or, D.: Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph*. 24 (3), 479–487, 2005.

[13] Sumner, R. W., Schmid, J., and Pauly, M.: Embedded deformation for shape manipulation. *ACM Trans. Graph*, 26, 3, 2007.

[14] Botsch, M., Pauly, M., Wicke, M., and Gross, M.: Adaptive space deformations based on rigid cells. In *Proc. Of Eurographics*, 339–347, 2007.

[15] Joshi, P., Meyer, M., DeRose, T., Green, B., and Sanocki, T.: Harmonic coordinates for character articulation. *ACM Trans. Graph*, 26 (3), 71, 2007.

[16] Lipman, Y., Levin, D., and Cohen-Or, D.: Green coordinates. *ACM Trans. Graph*, 27, 3, 2008.

[17] Botsch, M., Pauly, M., Gross, M., and Kobbelt, L.: PriMo: Coupled prisms for intuitive surface modeling. In *Proc. of Sym. on Geometry Processing*, 11–20, 2006.

[18] Popa, T., Julius, D., and Sheffer, A.: Interactive and linear material aware deformations. In *Proc. of Shape Modeling International*, 13 (1), 73–100, 2007.

[19] Kraevoy, V., Sheffer, A., Cohen-Or, D., and Shamir, A.: Non-homogeneous resizing of complex models. *ACM Trans. Graph*, 27 (5), #111, 2008.

[20] Wang, K. P., and Zhang, C. M.: Content-aware model resizing based on surface deformation. *Computers & Graphics*, 33, 433–438, 2009.

[21] Botsch, M., and Kobbelt, L.: Multiresolution surface representation based on displacement volumes. In *Proc. of Eurographics*, 483–491, 2003.

[22] Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., and Shum, H.-Y.: Large mesh deformation using the volumetric graph Laplacian. *ACM Trans. Graph*, 24 (3), 496–503, 2005.
[23] Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.-Y., Teng, S., Bao, H., Guo, B., and Shum, H.-Y.: Subspace gradient domain mesh deformation. *ACM Trans. Graph*, 25 (3), 1126–1134, 2006.
[24] Shi, X., Zhou, K., Tong, Y., Desbrun, M., Bao, H., and Guo, B.: Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Trans. Graph*, 26, 3, 2007.
[25] Au, O. K.-C., Fu, H., Tai, C.-L., and Cohen-Or, D.: Handle-aware isolines for scalable shape editing. *ACM Trans. Graph*, 26, 3, 83, 2007.
[26] Lipman, Y., Cohen-Or, D., Gal, R., and Levin, D.: Volume and shape preservation via moving frame manipulation. *ACM Trans. Graph*, 26, 1, 2007.
[27] Xu, W., Wang, J., Yin, K., Zhou, K., Van De Panne, M., Chen, F., Guo, B.: Joint-aware manipulation of deformable models. In *Proc. of ACM SIGGRAPH*, 1–9, 2009.
[28] Welch, W., and Witkin, A.: Variational surface modeling. In *Proc. of SIGGRAPH*, 26, 157–166, 1992.
[29] Masuda, H., Yoshioka, Y., and Furukawa, Y.: Preserving form features in interactive mesh deformation. *Computer Aided Design*, 39 (5), 361–368, 2007.
[30] Masuda, H., and Ogawa, K.: Application of interactive deformation to assembled mesh models for CAE analysis. In *ASME Int. Design Engineering Technical Conferences*, 2007.
[31] Cabral, M., Lefebvre, S., Dachsbacher, C., and Drettakis, G. Structure preserving reshape for textured architectural scenes. In *Proc. of Eurographics*, 469–480, 2009.
[32] Gal, R., Sorkine, O., Mitra, N. J., and Cohen-Or, D.: iWIRES: An analyze-and-edit approach to shape manipulation. *ACM SIGGRAPH Trans. Graph*, 28, 3, #33, 1–10, 2009.
[33] Dantzig, G. B.: Linear Programming and Extensions. Princeton, *NJ: Princeton University Press*, 1949.
[34] Davidon, W. C.: Variable metric method for minimization, Technical Report ANL–5990 (Revised). Argonne National Laboratory, Argonne, IL, 1959.
[35] Nocedal, J. and Wright, S. J.: Numerical Optimization1999. *New York: Springer-Verlag*, 1999.
[36] Forsgren, A., Gill, P. E., Wright, M. H.: Interior methods for nonlinear optimization. *SIAM Review*, 44(4), 525–597, 2002.
[37] Klee, V., Minty, G. J., and Shisha, O.: How Good is the Simplex Algorithm? In Inequalities 3. *New York: Academic Press*, 159-175, 1972.
[38] Karmarkar, N.: A New Polynomial-time Algorithm for Linear Programming. *Combinatorica* 4, 373 P. E. 395, 1984.
[39] Wilson, R: A Simplicial Method for Convex Programming. PhD thesis, Harvard University, 1963.
[40] Frank, M., Wolfe, P.: An algorithm for quadratic programming. *Naval Res. Logist. Quar.*, vol. 3, 95–110, 1956.
[41] Drud, A.: CONOPT – A Large-Scale GRG Code. *ORSA Journal on Computing* 6, 207–216, 1992.
[42] Fletcher, R., Leyffer, S.: User manual for filtersqp. Technical Report NA/181,Dundee, Scotland, 1998.

[43] Gill, P. E., Murray, W.: SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–006, 2002.
[44] Drud, A.: CONOPT: A GRG code for large sparse dynamic nonlinear optimization. *Mathematical Programming* 31,153–191, 1985.
[45] B. A. Murtagh and M. A. Saunders. MINOS 5.4 user's guide. Technical report, SOL, 83-20R, Systems Optimization Laboratory, Stanford University, 1983. Revised, 1995.
[46] Conn, A. R., Gould, G. I. M., and Toint, P. L. : LANCELOT: a Fortran package for Large-scale Nonlinear Optimization (Release A). *Springer Series in Computational Mathematics*. Springer Verlag, Heidelberg, Berlin, New York, 1992.
[47] Vanderbei, R.J., Shanno, D. F.: An interior point algorithm for nonconvex nonlinear programming. Computational Optimization and Applications, 13:231–252, 1999.
[48] Andersen, E.D., Andersen, K.D.: The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In H. Frenk, K. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization*, pages 197-232, Dordrecht, The Netherlands, Kluwer Academic Publishers, 2000.
[49] Betts, J., Eldersveld, S. K., Frank, P. D., Lewis, J. G.: An interior-point nonlinear programming algorithm for large scale optimization. Technical report MCT TECH-003, Mathematics and Computing Technology, The Boeing Company, P.O. Box 3707, Seattle WA 98124-2207, 2000.
[50] Wächter, A., Biegler, L. T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Technical Report RC 23149, IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, March, 2004.
[51] Byrd, R., Nocedal, J., Waltz, R.: Knitro: An integrated package for nonlinear optimization, Technical Report 18, Optimization Technology Center, Evanston, IL, 2005.