Practical Papercraft Models from Meshes

Student: Ying-Tsung Li

Advisor: Dr. Jung-Hong Chuang

Dr. Wen-Chieh Lin

Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

ABSTRACT

We present a new approach to generate papercraft models from meshes. The input mesh is approximated by a set of quadric surface proxies. Each quadric proxy is then cut and unfolded into a 2D papercraft pattern. Our method has the following advantages : First, we produce smoother papercraft models than previous methods. Second, the 2D patterns of papercraft models are easy to cut and glue. Finally, the 2D patterns of papercraft models are more meaningful due to the pre-defined cutting rules. We demonstrate this by physically assembling papercraft models from our algorithm.

Acknowledgments

I would like to thank my advisors, Professor Jung-Hong Chuang, and Wen-Chieh Lin for their guidance, inspirations, and encouragement. I also want to thank another faculty, Sai-Keung Wong. He give me many useful suggestions. I am grateful to Tan-Chi Ho for his comments and advices. Thanks to my colleagues in CGGM lab.: Chin-Hsian Chang, Kuang-Wei Fu, Hsin-Hsiao Lin, Yueh-Tse Chen, Ta-En Chen, Hong-Xuan Ji, Cheng-Min Liu, Jau-An Yang, Tsung-Shian Huang, and Chien-Kai Lo for their assistances and discussions. It is pleasure to be with all of you in the pass two years. Lastly, I would like to thank my parents for their love, and support.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Papercraft models are models that are assembled from a set of 2D patterns on the paper (Fig.1.1). The paper layout is usually generated by hand or interactively by various companies. This task is difficult for a normal user and we want to perform this automatically. There are two challenges to generate papercraft models automatically and practically. The first challenge is to segment a given model into parts that fit human's expectation and add cutting lines on each part to increase the developability. A meaningful part is always not a developable surface, so we need to apply cutting on it to increase the developability of this part and reduce the stretching when we unfold this patch. The second challenge is to unfold parts meaningfully. Some automatically cutting algorithm, like [14], will add the cut lines at the position that has the highest distortion. The cutting lines generated by those algorithm are irregular and are not easy to cut and glue. We want to cut patches meaningfully because the 2D patterns of man made papercraft model always follow some kinds of cutting rules.

To address the first issue, we use the quadric surface as the basic fitting primitive rather than simple developable surface like [20] [17]. Because quadric surfaces are more complex than developable surfaces, using quadric surface as the basic primitive can reduce the number of patches to approximate the input mesh. Thus, the assembled models which are fitted by quadric

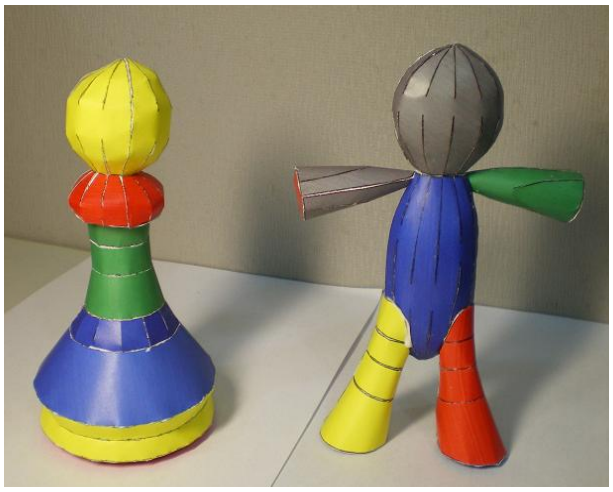


Figure 1.1: Papercraft models.

surfaces are smoother than models fitted by developable patches because the crease lines will appear on the patches boundary. For example, an ellipsoid can be consider as a single patch when quadric surfaces are the basic fitting primitives but it will have several patches when it is fitted by developable surfaces and it will not smooth between the patch boundary due to lack of G^1 continuity.

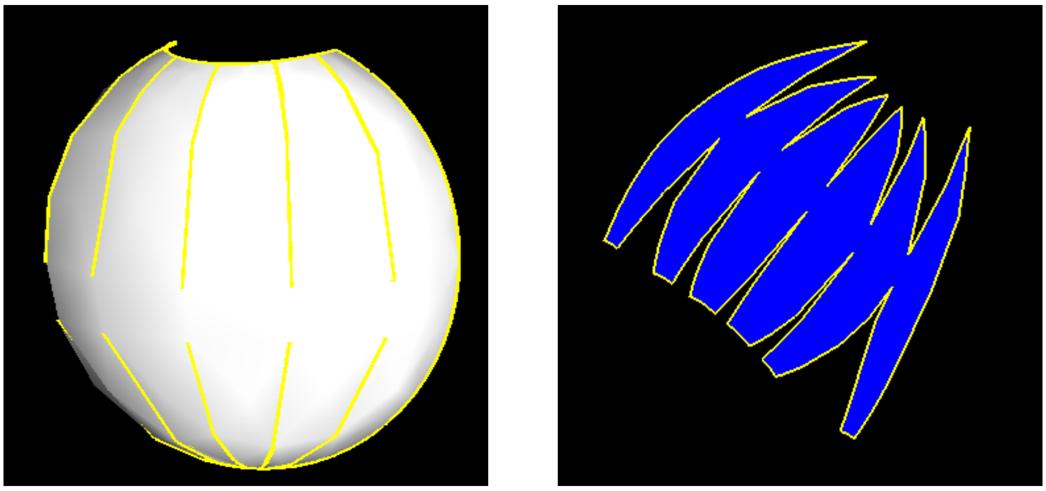
For the second issue, we cut the quadric surfaces using pre-defined cutting rules rather than automatically cutting algorithm like [14]. With pre-defined cutting rules, the unfolded 2D pattern will have the following advantages :

- User can figure out the assembled shape of the unfolded 2D pattern (Fig.1.2).
- The unfolded 2D pattern's boundary is smooth and is easy to cut and glue.

1.1 Contribution

The contributions of this thesis can be summarized as:

- The number of unfolded 2D patterns are fewer than previous papers.



(a) Half of an ellipsoid type model with the yellow lines as the cut lines.

(b) The unfolded 2D pattern of the half ellipsoid.

Figure 1.2: Model with pre-defined cutting rules and its unfolded 2D pattern.

- The boundary of unfolded 2D pattern are smooth and it is easy to cut and glue.
- The 2D pattern is correspond to a meaningful part of original 3D model.
- The 2D pattern's shape is meaningful that user can predict the assembled shape in 3D.
- The assembled papercraft model is smoother than previous paper because of using quadric surfaces as the basic fitting primitive rather than using simple developable surfaces.

1.2 Outline

The rest of the thesis is organized as follows: Chapter 2 gives the literature review, the background of mesh segmentation, shape approximation and related system for automatically generating papercraft models. Chapter 3 illustrate our proposed algorithm to generate a papercraft model from an input mesh. Chapter 4 shows our results from our approach. In the end, conclusions and future work are discussed in Chapter 5.

CHAPTER 2

Related work

The system we present shares common goals with a number of previously papers. Therefor, we introduce some related research.

2.1 Variational Shape Approximation

Cohen-Steiner et. al [6] propose a shape approximation algorithm based on clustering approach to optimally approximate a mesh surface by a specified number of planar faces. This optimization problem is solved as a discrete partition problem using the Lloyd algorithm [16], which is commonly used for solving the k-mean problem in data clustering. There are two iterative steps in this method: mesh partition and fitting a plane face, called a proxy, to each partitioned region. This method proves effective especially for extracting features and planar regions, but tends to produce an overly large number of planar proxies for a good approximation of a freeform surface. Because of its optimization nature, the method is often referred to as a variational method. Because this method only extracts the planar regions, the approximated models are piecewise linear in each patches. It does not fit our goal of generating a smooth papercraft models from meshes.

Wu and Kobbelt [27] extend the work in [6] by introducing spheres, cylinders and rolling ball patches as additional basic proxy types, so that a complex shape can be approximated to the same accuracy by a much fewer number of proxies, leading to a more compact representation. However, these newly added surface types mentioned are still rather restricted, even for CAD models and other man-made objects.

Simari et al. [24] use ellipsoids as the only type of proxies for approximating mesh surfaces, again using the Lloyd method with the error metric being a combination of Euclidean distance, angular distance and curvature distance. The segmentation boundaries are smoothed by a constrained relaxation of the boundary vertices.They also approximate the volume bounded by a mesh surface using a union of ellipsoids, where whole ellipsoids, rather than ellipsoidal surface patches, are used. This method is useful for collision detection but not useful for the purpose of papercrafting because the papercraft models do not consider the volume informations inside its surface.

Attene et al. [2] use hierarchical face clustering algorithm to approximate triangle meshes into a set of primitives include which planes, spheres and cylinders. This method can generate a binary tree of clusters which is fitted by one of the primitives in the bottom up scheme. The disadvantage of this method is that we can not get desire number of segment directly, we need to construct the binary tree first to eliminate the number of patches one by one. It takes too much time for processing large models.

Yan et al. [28] use the same framework as [6] [27] [24] but extens the types of proxy into the general quadric surfaces. Yan enhance the algorithm's performance by using Taubin's second order approximation to approximate the distance from some point to quadric surface [26]. The region boundaries are smoothed by graph cut method. We modified this method to fit the purpose of papercrafting by constraining some extreme cases that are nearly planar.

2.2 Paper Crafting

Papercraft models are the models which can be assembled from a set of 2D patterns on papers. The work for automatically generating papercraft models from meshes is to approximate the input mesh surface by a set of patches and these patches are unfolded into 2D paper. Because the paper can not stretch, crease and tear, these patches must be developable or nearly developable. There are three types of methods to achieve this goal:

Approximate by developable surfaces Elber [8] proposed a method for approximating NURBs by a set of developable strips. That work treats only a single surface patch, as opposed to models with complex geometry and topology. Elber also uses quite a loose upper bound on the Hausdorff distance as an error approximation. This bound can be far from the true Hausdorff distance, resulting in a much larger number of strips than necessary.

Massarwi et al. [17] approximate input mesh by a set of piecewise-developable surfaces which are generalized cylinder represented as a strip of triangles. They enhance Elber's method by computing more accurate Hausdorff distance between the input mesh and its piecewise-developable approximation. They also propose a framework to handle more complex meshes.

Shatz et al. [20] use conic surface as proxy surfaces, but additionally consider the error between the triangles and the conic surface. The proxy conic surfaces are treated as the final approximation surfaces and can be directly unfolded. Because of the error between the proxy conic and the origin meshes, sometimes the boundaries between the neighboring conic surfaces will be unstable and must be specially dealt with. They only apply additional optimization process on suce unstable boundaries, so seams will probably appear between neighboring conics.

Approximate by triangle strips Mitani and Suzuki [18] propose a algorithm that first segment the meshes into parts based on features. Then these parts are approximated with triangle strips. The same process repeat until all triangles are covered by some triangle strips. The triangle generated from their algorithm tend to have long boundaries which are not convenient for gluing. Another problem is that the method does not consider any error metric, the only way to control the error is the predefined width of the triangle strips, which is not flexible.

Approximate by nearly developable patches Julius et al. [11] propose using developability as error metric to segment the meshes. Their algorithm is based on region growing framework and use a Lloyd scheme. For each patch, a conic is used as a proxy surface. Because the angle between the conic's axis and normal on the surface is constant, the developable error metris is defined as

$$(N_C \cdots n_t - \cos\theta_C)^2,$$

where N_C is the axis of the conic, n_t is the normal of the triangle, θ is the constant angle. They additionally consider the compactness and boundary smoothness of the patch as error metrics and use the product of the three weighted error metrics as the region growing error metric. At each iteration, faces with the smallest error are inserted into the patches until all faces are covered, then the optimized proxy conics are computed to fit the patches, the process repeat until converge. Because the algorithm only segment the mesh, a parametrization method mustbe used to unfold patches into a plane. Because there are no isometric parametrization for general patches, distortion will still be introduced in the process.

2.3 Mesh Parameterization

Mesh parameterization is to find a parametric function that map the mesh surface onto the 2D parametric domain. There are two goals for mesh parameterization to achieve. The first one is to preserve area between 2D surface and the 3D surface. The second one is to preserve angle and is called conformal parametrization. The boundary type of the 2D surface also have two types : the restrict type and the free type. The restrict type sometimes use square as the boundary of 2D domain and solve the convex combination problem to get the parametric function [10] [9]. The free type will first fix some verties as the anchors and reconstruct the parametric function with the angle constrains [14] [21]. For the purpose of papercrafting, we need a conformal parametrization that also preserve the edge length between 3D and 2D. Following are the papers that are conformal parametrization.

Sheffer et al. [21] presented Angle-Based Flattening (ABF) method that defined an angle preservation metric by which the parameterization was computed in the angle space and then converted into a 2D coordinate. When converting the parameterization from the angle space into planar coordinates, natural boundaries were automatically formed. Sheffer et al. [22] presented ABF++ method that enhance the performance of ABF method based on numerical methods (sequential linear quadratic programming) and algebraic transforms of the initial problem. These transforms were combined with a multigrid optimization framework to further improve performances. Zayer et al. [29] use a linear approximate of the initial ABF equations. The time-space complexity and accuracy of the solution are to a great extent affected by the kind of approximation used. They reformulate the problem based on the notion of error of estimation. The error induced by this linearization is quadratic in terms of the error in angles and the validity of the approximation is further supported by numerical results.

Levy et al. [14] and Desbrun et al. [7] are parameterizations that also give free boundaries. It is difficult to add special constraints about the shearing (for shape) or stretching (for length) on boundaries in their linear systems. However, the positions of boundary vertices are more important than interior vertices since the shape and the area of a planar domain are essentially defined by the boundary vertices.

CHAPTER 3

Algorithm

This chapter describes algorithm of making papercraft models from meshes. We first give an overview of our proposed method (Subsection 3.1). Next, we describe the detail of each step of the algorithm, including Mesh Segmentation (Subsection 3.2), Fuzzy Segmentation (Subsection 3.3), Boundary Extraction (Subsection 3.4), Parametrization and Apply Cutting Rules (Subsection 3.5) and finally Unfold Quadric Patches (Subsection 3.6).

3.1 Overview

The aim of the algorithm is to segment the input mesh into a set of parts which includes several segments that can be well approximated by quadric surfaces and planes. Each part can be unfolded into 2D patterns which are meaningful and can be easily cut and glued. In order to achieve this goal, we do the following processes to our input meshes. First, we may segment the input mesh into parts manually or by some part-based segmentation algorithm and adapt quadric surface extraction method which is proposed by Yan et. al.[28] for further segmenting parts into quadric surface patches. Then, we use the idea of fuzzy region to avoid generating quadric

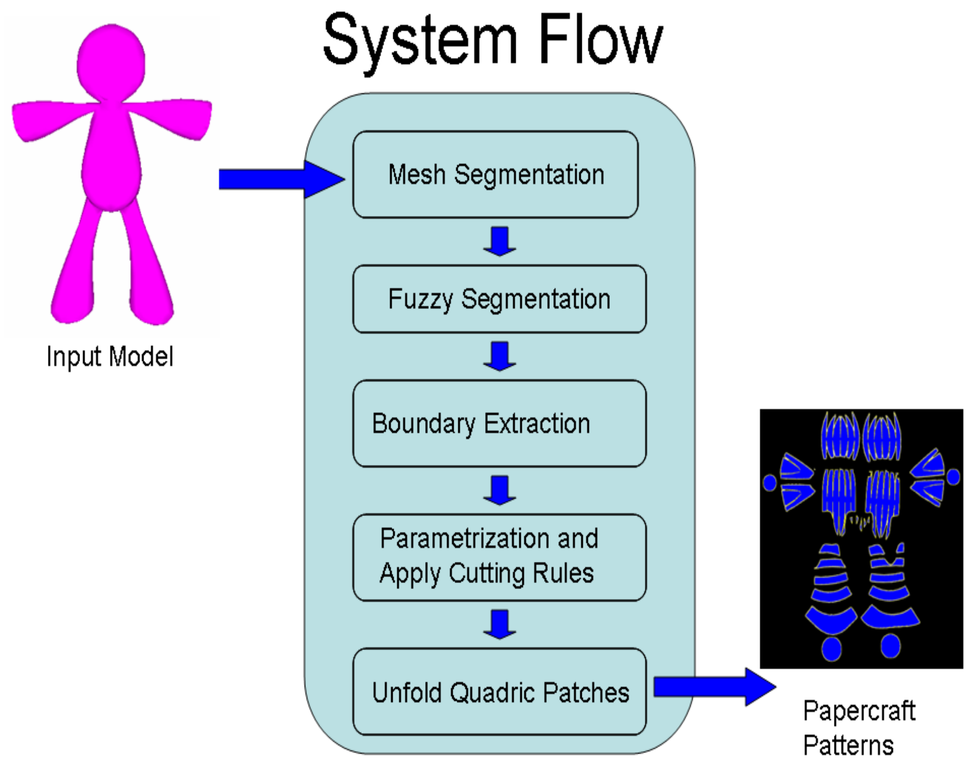


Figure 3.1: System flow of our papercrafting algorithm.

surface patches which have no intersection with others. The third step is to find the intersection curves between these quadric surfaces. In this step, we must avoid not to find curves which are not really needed for our algorithm by using the concept of Object-Oriented Bounding Box of the quadric surface. Next, we transform each quadric surface into its local parametric space and apply appropriate cutting rules on the parametric domain. For minimizing the number of quadric surface patches, we provide a function that is used to rotate the local parametric space to avoid some awful cutting results. Then, we triangulate each quadric patch on the parametric domain and restore from 2D to 3D by apply each patch's inverse parametrization. Finally, we use several conformal parametrization methods to unfold each quadric patch and use a metric to choose the best parametrization result as our final unfold 2D pattern. Fig.3.1 is the system flow of our algorithm.

3.2 Mesh Segmentation

In order to segment input meshes into a set of meaningful quadric patches, we do the following two steps : pre-segmentation and quadric surface extraction. The pre-segmentation step is optional and can be done manually or by some part-based segmentation algorithms like [12] [13] [5] [15]. The quadric surface extraction is based on Yan et. at.[28] and is modified to fit the purpose for papercrafting. We will introduce the variational framework first and demonstrate the error metric and fitting scheme that Yan used for quadric surfaces extraction. Then we will do some modifications to fit the purpose of papercrafting.

3.2.1 Variational Framework

Let $\mathcal{M}$ denote an input mesh surface, and $\mathcal{T}$ denote the set of triangles of $\mathcal{M}$. Suppose that $\mathcal{M}$ is partitioned into n non-overlapping regions, denoted as $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^{n}$, each region $\mathcal{R}_i$ containing a set of triangle $\mathcal{T}_i = \{t_k^i\}_{k=1}^{n_i}$ such $\cup_{i=1}^{n}\mathcal{T}_i = \mathcal{T}$. Each region $\mathcal{R}_i$ is approximated by a quadric proxy $\mathcal{P}_i$ (including the plane as a special case). A seed face $\mathcal{S}_i$ of $\mathcal{P}_i$ is the smallest error face in $\mathcal{T}_i$. In a variational framework the optimal partition $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^{n}$ is found by minimizing the following objective function :

$$E(\mathcal{R},\mathcal{P}) = \sum_{i=1}^{n} E'(\mathcal{R}_i,\mathcal{P}_i) = \sum_{i=1}^{n}\sum_{k=1}^{n_i} d(t_k^i,\mathcal{P}_i) \tag{3.1}$$

where $d(t_k^i,\mathcal{P}_i)$ measure the error between the triangle t_k^i and the proxy $\mathcal{P}_i$. Therefore, $E'(\mathcal{R}_i,\mathcal{P}_i)$ is the error between the region $\mathcal{R}_i$ and its approximating proxy $\mathcal{P}_i$. Lloyd's algorithm minimizes Eq.3.1 through iterative partition and fitting.

3.2.1.1 Error Metric for Proxies

In order to use quadric surface as the fitting proxy, they have to define the error metric quadric proxies. The error metric for quadric proxies extraction is defined as the L^2 distance. Because computing exact distance from a point to quadric surface is very slow, they decide to use Taubin's second order approximation of the Euclidean $\delta_d(p, Z(f))$ [26] from point p to the quadric surface $Z(f)$ as the approximate distance. Based on this distance, the approximated L^2 distance for a triangle t to quadric surface P_i is defined as :

$$d(t, P_i) = \frac{1}{m} \sum_{k=1}^{m} \delta_d(p_k, Z_i(f))^2 \cdots A$$

where $\{p_k\}_{k=1}^{m}$ are uniformly sampled points on the triangle t and A is the area of t which is a weighting factor to account for triangles of different size. The approximated L^2 distance between R_i and P_i is then defined as:

$$E'(R_i, P_i) = \sum_{t_j \in R_i} d(t_j, P_i) / \sum_{t_j \in R_i} A_j.$$

To have a uniform comparison, all mesh are scaled uniformly to fit in a rectangular box with the diagonal length being 1.

3.2.1.2 Quadric Surface fitting

Given a region R_i, we need to fit a quadric surface to R_i in L_2 metric. For performance concern, they use Taubin's method [25] based on a first-order approximation of L^2 metric for quadric surface fitting.

Let $f(x, y, z) = 0$ be a quadric surface. The squared distance from a point p to the implicit surface $Z(f) = \{(x, y, z) | f(x, y, z) = 0, x, y, z \in R\}$ is approximated as $d(p, Z(f))^2 \approx \frac{f(p)^2}{\|\nabla f(p)\|^2}$. The sum of approximated squared distance is following [28]

$$\frac{1}{A} \sum_{k=1}^{n_i} \int_{t_k} d(p, Z(f))^2 dp \approx \frac{\frac{1}{A} \sum_{k=1}^{n_i} \int_{t_k} f(p)^2 dp}{\frac{1}{A} \sum_{k=1}^{n_i} \int_{t_k} \|\nabla f(p)\|^2 dp} = \frac{s^t M_t s}{s^t N_t s}, \tag{3.2}$$

where M, N are coefficient metrics and $s =< C_0, C_1, ..., C_9 >^T$ and A is the sum of the areas of all triangles in R_i. The fitting problem is reduced to computing the eigenvector of $M - \lambda N_t$ associated with the minimum eigenvalue.

3.2.2 Modification for Papercrafting

By using Yan's method, we find that plane proxy is rarely happened due to small noises of mesh data. Mostly near planar regions are fitted as the quadric type of hyperboloid of two sheets which has small coefficients and is fitted to some region in the quadric proxy which is far from proxy's center. For papercrafting, we prefer to choose plane proxy than hyperboloid of two sheets proxy in such a region. In order to achieve this goal, we fit each region $\mathcal{R}_i$ as quadric proxy $\mathcal{P}_i$ and plane proxy $\mathcal{P}'_i$ at the same time. The fitting error is denote as E_i and E'_i. If E'_i is smaller than E_i or E'_i is larger than E_i but is smaller than user defined plane tolerance δ_{plane}, we choose $\mathcal{P}'_i$ as $\mathcal{R}_i$'s approximating proxy. Otherwise we choose $\mathcal{P}_i$ as $\mathcal{R}_i$'s approximating proxy.

3.3 Fuzzy Segmentation

There are some problems in the original quadric surface extraction method [28]. For example, the input mesh Fig.3.3(a) is compose of two ellipsoids like object. When we apply mesh segmentation, we can segment this mesh into two regions which have the quadric proxies of ellipsoid type Fig.3.3(b). Fig.3.3(c) is the quadric surface of these two quadric proxies which are drawn by MATLAB. From Fig.3.3(c), we can see that the two quadric surfaces are disconnected. This is a crucial problem for papercrafting and we want solve this problem by automatically connect these two quadric surface. We find out that this problem is due to lack of informations of triangles which are near the region boundary during the process of quadric surface fitting. When we apply quadric surface fitting, we only consider the triangles in R_i. Triangles which are near the ∂R_i and not belong to R_i also have useful information for fitting process because these triangles also have low error respect to R_i. Therefore, we define a fuzzy region $\mathcal{FR}_i$ which includes triangles nearby the region boundary for each region $\mathcal{R}_i$. Before we give the definition of $\mathcal{FR}_i$, we first define the triangle to triangle distance D_{ij} as follow (Fig.3.2):

$$D_{ij} = \begin{cases} undefined & , t_i \textit{ is not adjacency to } t_j \\ \overline{ab} + \overline{cb} & , t_i \textit{ is adjacency to } t_j \end{cases}$$

, where a, c are the center respect to triangle t_i and t_j and b is the middle of the shared edge of t_i and t_j. Then, we define σ as the average triangle to triangle distance of whole mesh.

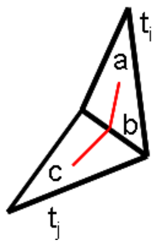


Figure 3.2: Triangle to triangle distance : a,c are the center respect to triangle t_i and triangle t_j, and b is the middle of the shared edge between t_j and t_j. $D_{ij} = |\overline{ab}| + |\overline{bc}|$.

The fuzzy region $\mathcal{FR}_i$ is defined as :

$$FR_i = \{t | \forall t \in T, t \notin R_i \text{ and the distance of } t \text{ to } \partial R_i \text{ is less than } \kappa\sigma\},$$

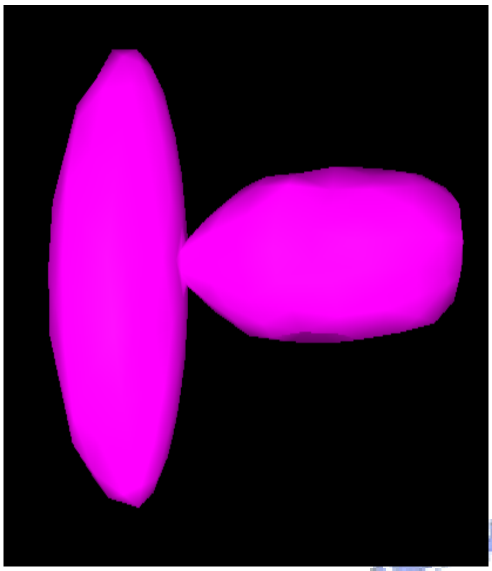
where κ is the user defined parameter that can control the size of fuzzy region. Now, we adapt the original fitting method by adding weighting for each triangle and incorporating triangles in fuzzy region. The original quadric surface fitting equation 3.2 is modified as follow:

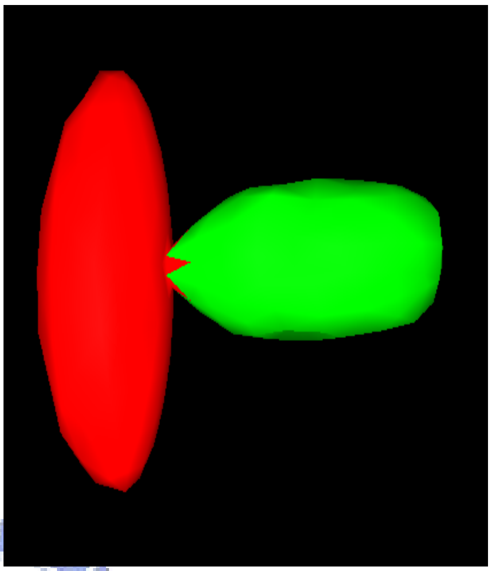
$$\frac{1}{A}\sum_{k=1}^{n_i} w_i \int_{t_k} d(p, Z(f))^2 dp \approx \frac{\frac{1}{A}\sum_{k=1}^{n_i} w_i \int_{t_k} f(p)^2 dp}{\frac{1}{A}\sum_{k=1}^{n_i} w_i \int_{t_k} \|\nabla f(p)\|^2 dp} = \frac{s^t M_t s}{s^t N_t s},$$

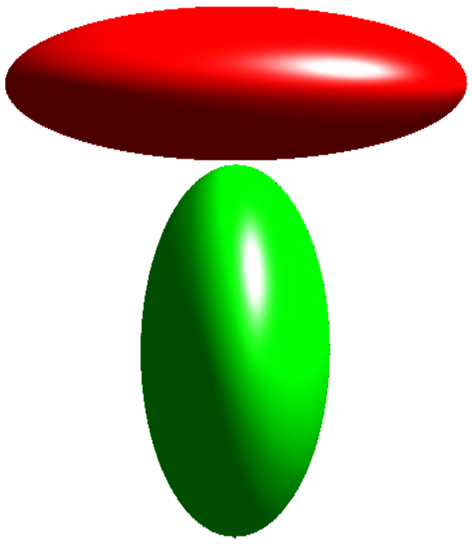
where M_t, N_t are coefficient matrices, n_i is the number of triangles of $|\mathcal{R}_i \cup \mathcal{FR}_i|$, A is the sum of the areas of all triangles in $\mathcal{R}_i \cup \mathcal{FR}_i$, and w_i is the additional weighting for each triangle in $\mathcal{R}_i \cup \mathcal{FR}_i$. The weighting w_i for each triangle t_i is defined as :

$$w_i = \{ \begin{array}{ll} 1.0 & , t_i \in \mathcal{R}_i \\ 1.5 \times (0.5)^{\lfloor \frac{dist(t_i, \partial R_i)}{\sigma} \rfloor} & , t_i \in \mathcal{FR}_i \end{array}$$

Through the new fitting scheme, the resulting quadric proxies will not be disconnected Fig. 3.3(d). The disadvantage of using fuzzy region is increasing fitting error, because the resulting quadric proxy P_i is not only best fit to the region R_i but also include the nearby region $\mathcal{FR}_i$. The quadric proxy with fuzzy region looks fatter than the quadric proxy without fuzzy region. Figure3.5 shows the fuzzy regions which are marked as the white faces with different κ value. Figure 3.5 shows the fitting results with different κ values.


(a) Original input mesh.


(b) Mesh that is segmented into two non-overlapping regions.


(c) Without fuzzy segmentation, quadric proxies are disconnected.

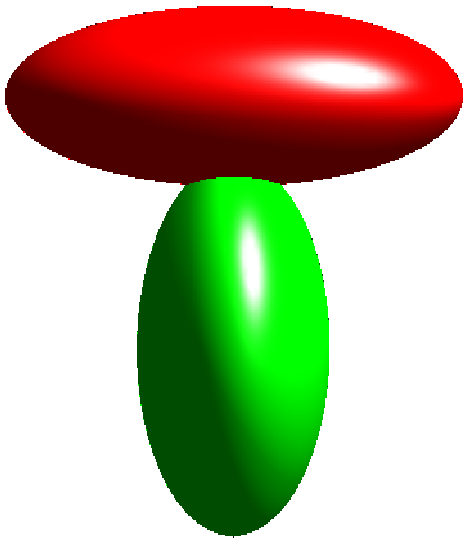
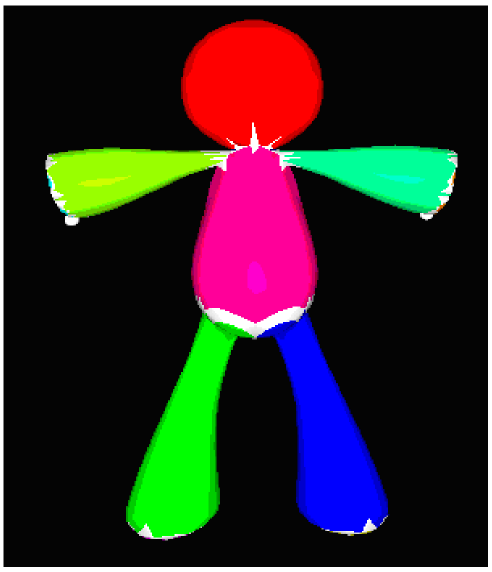

(d) With fuzzy segmentation, quadric proxies are connected.

Figure 3.3: Example that is used to illustrate the effect of fuzzy segment.


(a) $\kappa = 1$

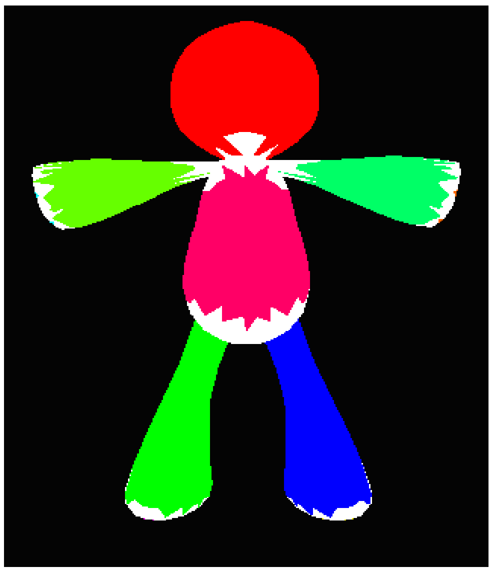
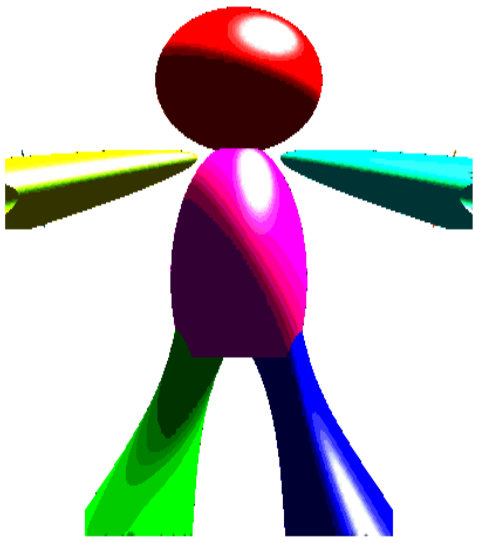

(b) $\kappa = 3$

Figure 3.4: Demonstrate fuzzy regions with different κ. The white faces are the fuzzy regions of the mesh.


(a) With $\kappa = 1$, the arm is disconnect to body.

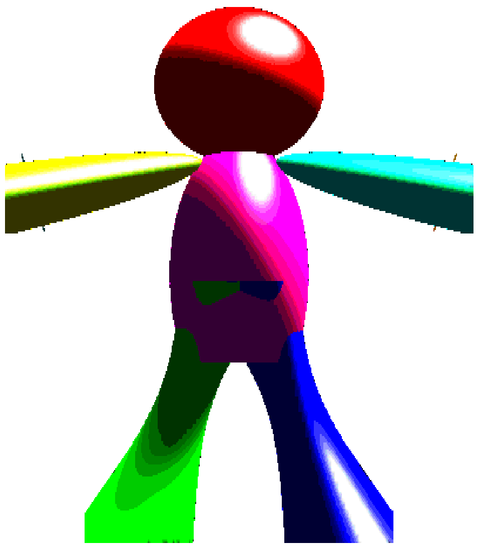

(b) With $\kappa = 3$, the arm is connect to body but the legs become fatter.

Figure 3.5: The fuzzy segmentation results with different κ.

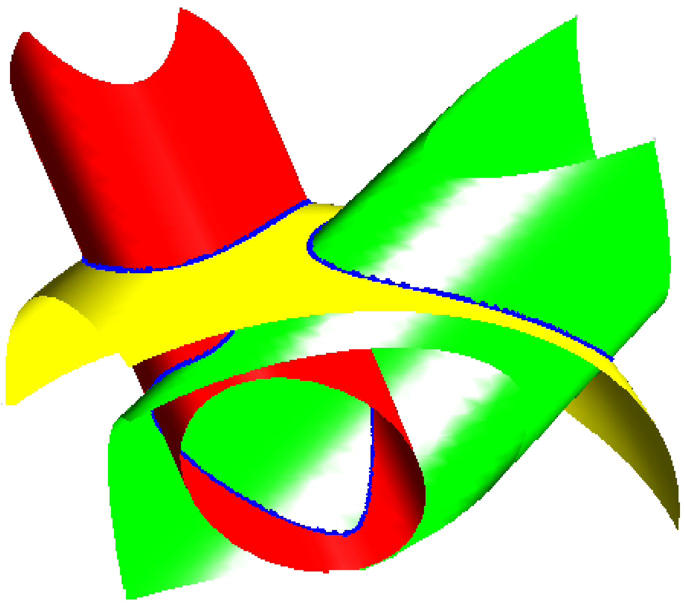


Figure 3.6: Three quadric surfaces and their intersection curves. The blue lines are the intersection curve.

3.4 Boundary Extraction

After the previous steps, the mesh has been segmented into a set of non-overlapping regions. Each region $\mathcal{R}_i$ has its own quadric proxy $\mathcal{P}_i$. Define $\mathcal{N}_i$ be the set of regions which is adjacency to $\mathcal{R}_i$. The problem of finding $\mathcal{P}_i$'s real boundary becomes finding the quadric surface intersection between $\mathcal{P}_i$ to the proxy of every region in $\mathcal{N}_i$ (Fig.3.6). Because finding quadric surface intersection exactly is very hard, we use a marching like method to get an acceptable result. This method has two steps and can apply to two adjacency regions. Firstly, we need to find a starting point that is on the intersection curve of these two quadric surface(Sec.3.4.1). Secondly, we trace from the starting point along the curve's tangent direction until reaching starting point again (Sec.3.4.2). For convenience, we denote $\mathcal{R}_i$ and $\mathcal{R}_j$ are the regions which we want to find their intersection curve.$\mathcal{B}$ is the boundary that $\mathcal{R}_i$ and $\mathcal{R}_j$ shared.

3.4.1 Find Starting Point

We pick an arbitrary vertex on the boundary as our initial guess point p. Then, we use an iterative algorithm to pull p to the intersection curve (Algo.3.1). The function "Project" projects a point p onto the quadric surface $\mathcal{P}$. Fig.3.7 is an example of this procedure.

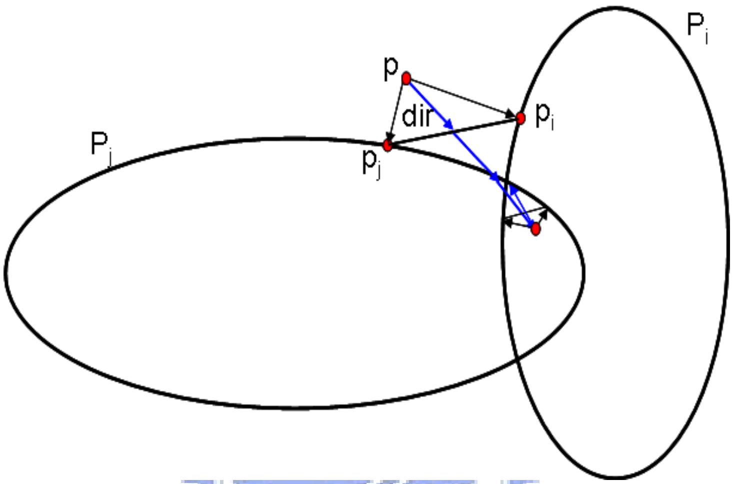


Figure 3.7: An example to illustrate Algo.3.1.

At the end of the find starting point algorithm, we need to check whether the starting point we found is correct or not. A correct starting point means the distance d of of $|\overline{pp_i}| + |\overline{pp_j}|$ are less than some user defined threshold. One of the incorrect case may occur when the starting point reaches a position that is nearly collinear to the two projected points p_i and p_j. In this case, the starting point may oscillate on the line of $\overline{p_ip_j}$ and will not come closer to the intersection curve. In order to avoid this problem, we change the quadric surface P_i to its pencil quadric $Q = P_i + \lambda P_j$ and find the intersection curve between the quadric surface Q and P_j. In our implementation, λ starts from 1 until we find a correct intersection point.

3.4.2 Tracing Along Tangent Direction

After finding a correct starting point on the intersection curve, we can trace the entire curve along the curve's tangent direction $\overrightarrow{t}$. Let $G_i(x, y, z) = \nabla f_i(x, y, z)$ where $f_i(x, y, z)$ is the

Algorithm 3.1: The algorithm of finding first intersection point

```
//step 1: Choose an arbitrary vertex on boundary
$p$ = any vertex $\in \mathcal{B}$


//step 2: Iterative optimizes $p$ to the intersection curve
for( $i = 0$ ; $i < num_iter$ ; $i$++ ) {
    // Compute the search direction
    $p_i$ = Project($\mathcal{P}_i$, $p$)
    $p_j$ = Project($\mathcal{P}_j$, $p$)
    $dir = p - \frac{p_i + p_j}{2}$
    // Walk from $p$ along direction $dir$ until crossing both quadric surface
    $sign_i = \mathcal{P}_i(p)$
    $sign_j = \mathcal{P}_j(p)$
    $p' = p$
    do {
        $p' = p' + dir$
        $sign'_i = \mathcal{P}_i(p')$
        $sign'_j = \mathcal{P}_j(p')$
    }while($sign_i \times sign'_i > 0$ || $sign_j \times sign'_j > 0$)
    $p = p'$
}
```

implicit function of quadric surface $\mathcal{P}_i$ and $N_i(x, y, z) = \frac{G_i(x,y,z)}{|G_i(x,y,z)|}$. The curve's tangent direction $\overrightarrow{t}$ at p is derived as $N_i(p) \times N_j(p)$ which is perpendicular out of the paper (Fig.3.8). With the point p and the tangent direction $\overrightarrow{t}$ at p, we can shoot from p to next intersection point p' with constant shooting distance. Due to numerical error of computer program, we must do some correction to p after every shooting step to avoid p go away from real intersection curve too far. This correction is done by projecting p to quadric surfaces $\mathcal{P}_i$ and $\mathcal{P}_j$ and get two projected points p_1 and p_2. Then, we correct p by $p = \frac{p+p1+p2}{3}$. The tracing process terminates until the distance of p and the first intersection point is less than the constant shooting distance.

After the entire intersection curve is found out, we still need to check whether this curve is the desired one. For some quadric surfaces pair, they may have more than one intersection curves. We want the intersection curve that is near the region boundary R_i and R_j. In order to check this, we compute the object-oriented bounding box of R_i and R_j and scale the axis length by 1.2 first. Then, we compute the length L_i and L_j that the intersection curve in the obb respect to R_i and R_j. Next, we compute the ratio r_i and r_j which are the percentage of L_i and L_j to the total length of intersection curve. We define a probability threshold δ_r and use this threshold to check whether the intersection curve is desired or not. We want the curve that both r_i and r_j are larger than δ_r.

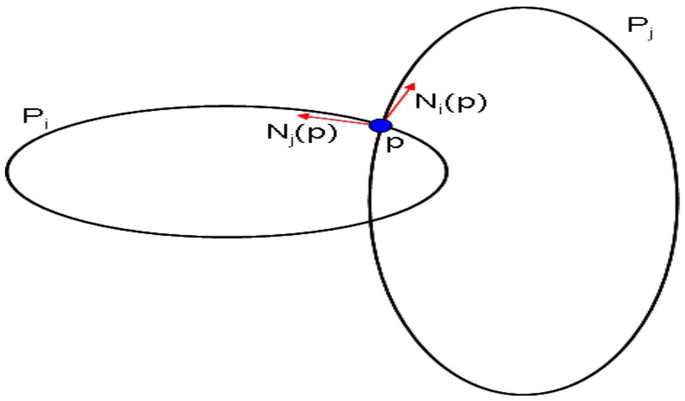


Figure 3.8: The tangent direction of the intersection curve at p is $N_i(p) \times N_j(p)$ which is perpendicular out the paper.

3.5 Parametrization and Apply Cutting Rule

Because the quadric surface are not all developable, we need to add some cutting lines on the surface to increase the developability. For the purpose of practical papercrafting ,we want the cutting lines are added in some pre-defined rules that fulfill human's expectation. The cutting rules are defined on the 2D parametric space. In order to apply cutting rules on quadric proxy, we do not add cutting lines on 3D surface directly. We need to transform each proxy into its local parametric space. There are four steps to achieve this goal. First, we need to classify the type of the quadric proxy (Subsection 3.5.1). Next, we transform this quadric proxy to its canonical form and parametrize it by predefined parametric function(Subsection 3.5.2). Finally, we apply cutting rules on the 2D patches (subsection .3.5.3).

3.5.1 Classify quadric type

The quadric surfaces can be defined as the implicit function

$$f(x) = x^T Ax + B^T x + c = 0 \tag{3.3}$$

where A is a 3×3 nonzero symmetric matrix, B is a 3×1 vector, and c is a scalar.

The surface type of each quadric surface can be characterized by analyzing the eigenvalue of matrix A. Since the matrix A is symmetric, it has an eigendecomposition

$$A = RDR^T \tag{3.4}$$

where $R = [v_0|v_1|v_2]$ is a rotation matrix whose columns v_i are linearly independent eigenvectors of A_i, and where $D = Diag(d_0, d_1, d_2)$ is a diagonal matrix of eigenvalues of A. The eigenvector v_i corresponds to the eigenvalue d_i.
Define $y = R^T x$ and $e = R^T b$. Equation 3.3 may be written as

$$y^T Dy + e^T y + c = 0 \tag{3.5}$$

Let y have components labeled y_i and let e have components labeled e_i for $0 \leq i \leq 2$. Equation 3.5 becomes

$$d_0 y_0^2 + d_1 y_1^2 + d_2 y_2^2 + e_0 y_0 + e_1 y_1 + e_2 y_2 + c = 0 \tag{3.6}$$

If any of the d_i are not zero, we can complete the square on the d_i terms:

$$d_i y_i^2 + e_i y_i = d_i (y_i + \frac{e_i}{2d_i})^2 - \frac{e_i^2}{4d_i} \tag{3.7}$$

This is the basis for the classification, but requires us to analyze the signs of the d_i. When a value d_i is zero, we will then have to analyze the sign of the corresponding e_i value. In preparation for the classification, define

$$r = -c + \sum_{i=0, d_i \neq 0}^{2} \frac{e_i^2}{4d_i} \tag{3.8}$$

The classifications are depend on the signs of the d_i, e_i, and r. We assume that the eigenvalues are to be ordered as $d_0 \leq d_1 \leq d_2$ so that the classification can be summarized as table 3.1, 3.2, 3.3 and 3.4. For more details, the reader is referred elsewhere Boehm et al.[3] and Schneider et al.[19].

Table 3.1: Three nonzero eigenvalues

Value of r	Relation of d_i	Surface type
$r > 0$	$0 < d_0 \leq d_1 \leq d_2$	ellipsoid
	$d_0 < 0 < d_1 \leq d_2$	hyperboloid of one sheet
	$d_0 \leq d_1 < 0 < d_2$	hyperboloid of two sheets
	$d_0 \leq d_1 \leq d_2 < 0$	no solution
$r < 0$	$0 < d_0 \leq d_1 \leq d_2$	no solution
	$d_0 < 0 < d_1 \leq d_2$	hyperboloid of two sheet
	$d_0 \leq d_1 < 0 < d_2$	hyperboloid of one sheets
	$d_0 \leq d_1 \leq d_2 < 0$	ellipsoid
$r = 0$	$0 < d_0 \leq d_1 \leq d_2$	point
	$d_0 < 0 < d_1 \leq d_2$	elliptic cone
	$d_0 \leq d_1 < 0 < d_2$	elliptic cone
	$d_0 \leq d_1 \leq d_2 < 0$	point

Table 3.2: Two nonzero eigenvalues

Relation of d_i	Value of e_i	Value of r	Surface type
$d_0 = 0 < d_1 \leq d_2$	$e_0 \neq 0$		elliptic paraboloid
	$e_0 = 0$	$r > 0$	elliptic cylinder
		$r = 0$	line
		$r < 0$	no solution
$d_0 < 0 = d_1 < d_2$	$e_1 \neq 0$		hyperbolic paraboloid
	$e_1 = 0$	$r \neq 0$	hyperbolic paraboloid
		$r = 0$	two planes
$d_0 \leq d_1 < d_2 = 0$	$e_2 \neq 0$		elliptic paraboloid
	$e_2 = 0$	$r > 0$	no solution
		$r = 0$	line
		$r < 0$	elliptic cylinder

Table 3.3: One nonzero eigenvalue

Relation of d_i	Value of e_i	Value of r	Surface type
$d_0 = d_1 = 0 < d_2$	$e_0 \neq 0$ or $e_1 \neq 0$		parabolic cylinder
	$e_0 = e_1 = 0$	$r > 0$	two plane
		$r = 0$	double plane
		$r < 0$	no solution
$d_0 < 0 = d_1 = d_2$	$e_1 \neq 0$ or $e_2 \neq 0$		parabolic cylinder
	$e_1 = e_2 = 0$	$r > 0$	no solution
		$r = 0$	double planes
		$r < 0$	two plane

Table 3.4: Zero nonzero eigenvalue

Value of e_i	Surface type
$e_0 \neq 0$ or $e_1 \neq 0$ or $e_2 \neq 0$	plane
$e_0 = e_1 = e_2 = 0$	no solution

3.5.2 Transform to canonical form

In order to transform quadric proxy into canonical form, we use R as the rotation matrix that can transform original quadric proxy to axis-aligned proxy. By equation 3.7, we can get a translation vector $T = [T_0, T_1, T_2], T_i = -\frac{e_i}{2d_i}$ which can move each axis-aligned quadric proxy's center to origin.The detailed method is referred [4]. After we transform each quadric proxy to its canonical form, we can use the predefined parametric function to parametrize it. We choose the parametric function according to its quadric type. The quadric types and the corresponding parametric functions are in the table 3.5.2.

Quadric Type	Parametric function
Parabolic Cylinder	$u = \frac{z}{e_2}$ $v = y$
Elliptic Cylinder	$u = \tan^{-1}(\frac{y\times d_0}{x\times d_1})$ $v = z$
Hyperbolic Cylinder	$u = \cos^{-1}(\frac{y\times d_0}{x\times d_1})$ $v = z$
Elliptic Paraboloid	$u = \tan^{-1}(\frac{y\times d_0}{x\times d_1})$ $v = \frac{z}{e_2}$
Hyperbolic Paraboloid	$u = \cos^{-1}(\frac{y\times d_0}{x\times d_1})$ $v = \frac{z}{e_2}$
Elliptic Cone	$u = tan^{-1}(\frac{d_0\times y}{d_1\times x})$ $v = \frac{z}{d_2}$
Hyperboloid One Sheet	$u = \tan^{-1}(\frac{y\times d_0}{x\times d_1})$ $v = \tan^{-1}(\frac{z}{d_2})$
Hyperboloid Two Sheets	$u = \sin^{-1}(\frac{y\times d_0}{x\times d_1})$ $v = \tan^{-1}(\frac{z}{d_2})$
Ellipsoid	$u = \tan^{-1}(\frac{y\times d_0}{x\times d_1})$ $v = \sin^{-1}(\frac{z}{d_2})$
Plane	$u = x$ $v = y$

Table 3.5: Quadric type and the corresponding parametric function, where $v = [x, y, z]$ is a point on the quadric surface.

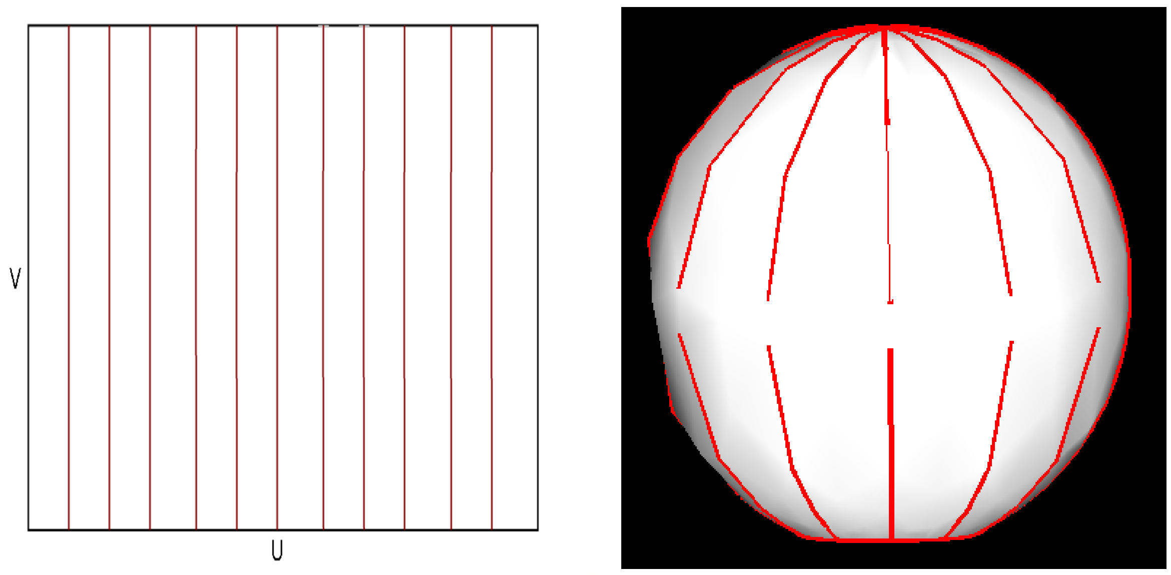
3.5.3 Apply cutting rules

Once we parametrize each quadric proxy into its local parametric space. We add cut lines on the 2D parametric space according to its quadric type. There are three category of quadric surfaces.

Developable quadric This category includes plane, elliptic cylinder, parabolic cylinder, hyperbolic cylinder and elliptic cone. For this type of quadric, we do not apply any cutting on the 2D pattern.

Ellipsoid We use cut lines that cut this ellipsoid like we peel the banana (Fig.3.9(b)). The cutlines are added vertically for every 30 degrees which is starts from -180 to 180 (3.9(a)). For each vertical cut line, we analyze the intersections between this line and the curves on the 2D parametric space. We keep the 10 percents of length between every two adjacency intersection points to avoid small pieces (Fig.3.10).

Others For quadric types of hyperboloid one sheet, hyperboloid two sheets, elliptic paraboloid and hyperboloid paraboloid, we add horizontal cut lines from the bottom to top and all cut lines are evenly spacing on the v-axis (Fig.3.11). We define a minimum vertical spacing δ_{vs} to control the number of horizontal cut lines N_H. Let H be the height of the bounding box for the quadric proxy in its local parametric space, N_H is defined as the largest positive integer such that $\delta_{vs} \times N_H < H$. Fig.3.12 is an example of cutting hyperbolic paraboloid.



(a) The cutting rule for the quadric type of ellipsoid. The vertical cut lines are added for every 30 degrees in u axis.

(b) An ellipsoid with ellipsoid's cutting rule, the red lines indicate the cut lines.

Figure 3.9: Cutting rule for ellipsoid type and the example quadric that apply the rule.

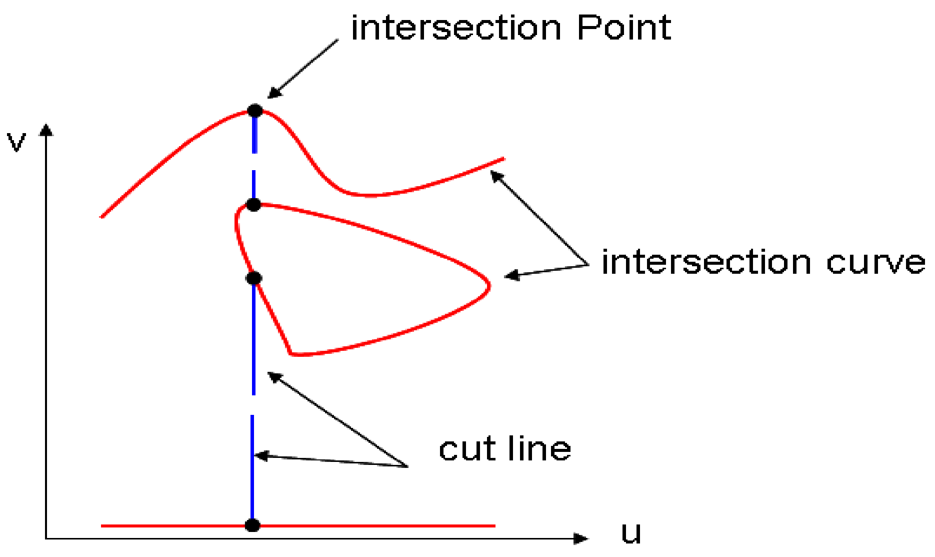


Figure 3.10: A example of vertical cut line that preserve 10 percent length between every two intersection points.

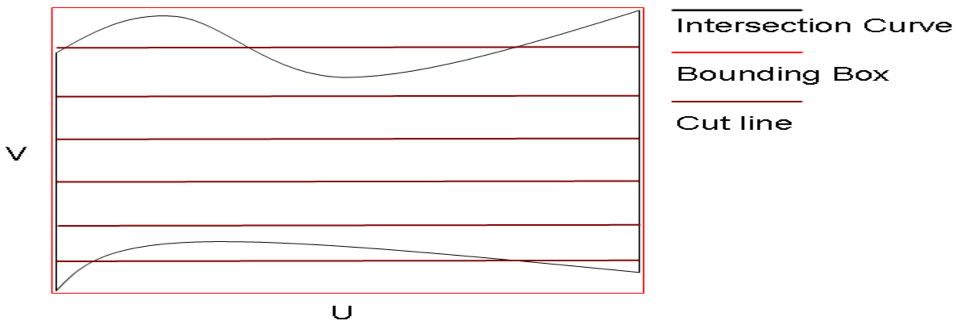
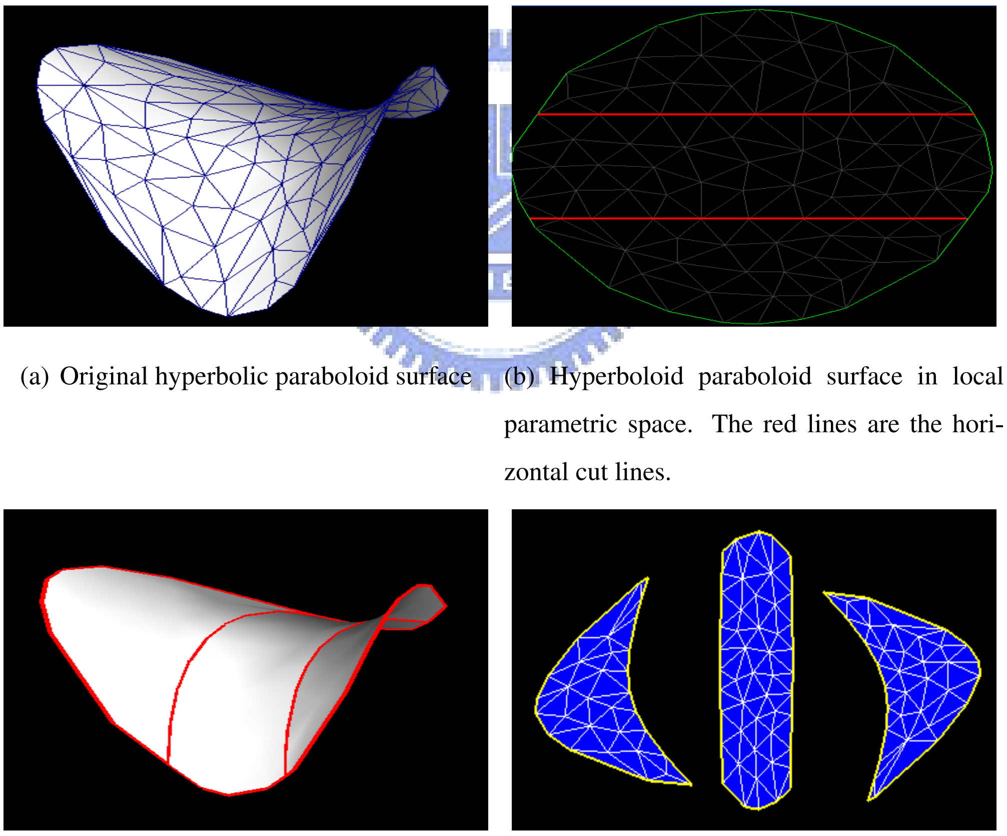


Figure 3.11: The cutting rule for hyperboloid one sheet, hyperboloid two sheets, elliptic paraboloid and hyperboloid paraboloid types.



(a) Original hyperbolic paraboloid surface

(b) Hyperboloid paraboloid surface in local parametric space. The red lines are the horizontal cut lines.

(c) The cut lines on the hyperboloid paraboloid surface in 3D.

(d) The unfolded patterns.

Figure 3.12: Example of cutting hyperbolic paraboloid surface.

3.6 Unfold Quadric Patches

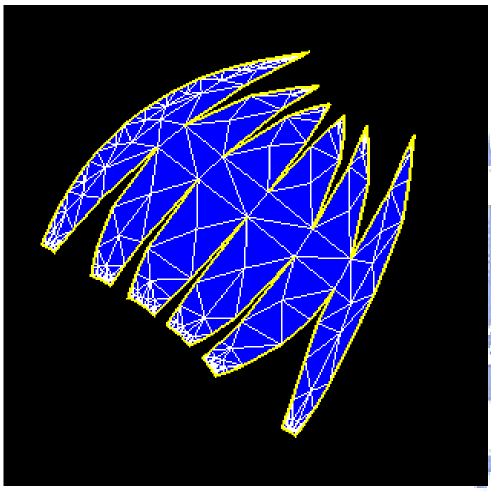
After adding cut lines on the 2D patches generated from Section 3.5, we use Triangle(.[23]) to triangular each 2D patch. For the resulting 2D mesh, we reconstruct the 3D mesh by projecting 2D mesh's vertices onto the original quadric surface using the inverse parametric function(Table 3.6). Thus, we have a set of 3D patches that represent the quadric proxy. For each 3D patch, we unfold it using several conformal parametrization methods and define an metric D to choose the best one as our papercraft pattern.

The metric D are used to ensure that the 2D unfolded pattern can be cut and glued correctly, so we measure the parametrization quality by computing the boundary edges' length difference between the 3D reconstructed mesh and the 2D unfolded mesh. Before we apply this metric on the 2D unfolded mesh, we need to scale the 2D mesh's boundary length to match with the 3D mesh's boundary length. The metric D can be defined as follow :

$$D(\mathcal{M}) = \sum_{edge\ e\in\partial\mathcal{M}} (||e_{2D}| - |e_{3D}||)$$

where $\mathcal{M}$ is the mesh topology and e_{2D} and e_{3D} are the edge in the 2D unfolded mesh and in the 3D reconstructed mesh.

In our current implementation, we use LSCM[14] and ABF[21] as the possible parametrization candidates. Fig.3.13 shows the parametrization results of the model in fig.3.9(b) using LSCM and ABF respectively. From fig. 3.13, we can see that the parametrization result of LSCM in this case has serious problems. The error metric D that computes the parametrization results of ABF and LSCM are 0.0640588 and 0.313379. Because ABF method gives smaller error than LSCM method, we choose the ABF parametrization result as our patch unfolded pattern in this case.


(a) Parametrize with ABF (Error = 0.0640588)

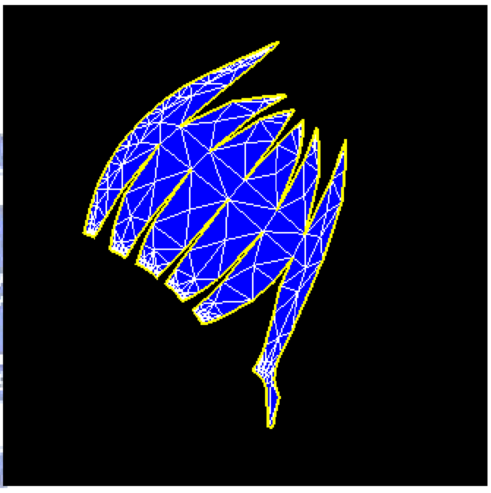

(b) Parametrize with LSCM (Error = 0.313379)

Figure 3.13: Parametrize the model in 3.9(b) with different parametrization methods

Quadric Type	Inverse Parametric function
Parabolic Cylinder	$x = d_0 \times \sqrt{2 \times u}$ $y = v$ $z = u$
Elliptic Cylinder	$x = d_0 \times cos(u)$ $y = d_1 \times sin(u)$ $z = v$
Hyperbolic Cylinder	$x = d_0 \times tan(u)$ $y = d_1 \times sec(u)$ $z = v$
Elliptic Paraboloid	$x = d_0 \times \sqrt{v} \times cos(u)$ $y = d_1 \times \sqrt{v} \times sin(u)$ $z = e_2 \times v$
Hyperbolic Paraboloid	$x = d_0 \times \sqrt{v} \times tan(u)$ $y = d_1 \times \sqrt{v} \times sec(u)$ $z = e_2 \times v$
Elliptic Cone	$x = d_0 \times cos(u) \times v$ $y = d_1 \times sin(u) \times v$ $z = d_2 \times v$
Hyperboloid One Sheet	$x = d_0 \times cos(u) \times sec(v)$ $y = d_1 \times sin(u) \times sec(v)$ $z = d_2 \times tan(v)$
Hyperboloid Two Sheets	$x = d_0 \times sec(u) \times sec(v)$ $y = d_1 \times tan(u) \times sec(v)$ $z = d_2 \times tan(v)$
Ellipsoid	$x = d_0 \times cos(u) \times cos(v)$ $y = d_1 \times sin(u) \times cos(v)$ $z = d_2 \times sin(v)$
Plane	$x = u$ $y = v$ $z = 0$

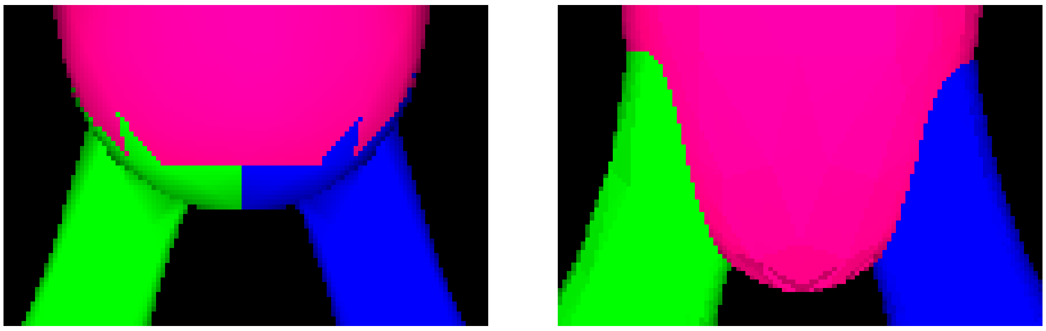
Table 3.6: Quadric type and the corresponding inverse parametric function, where $p = [u, v]$ is a point on the parametric space.

CHAPTER 4

Result

All experiments are performed on Intel core 2 duo CPU E6750 2.66GHz. We use the PMLAB library to maintain the half-edge structure of input models and use Triangle library [23] to generate quality triangulation of our parametrized 2D quadric proxies. The parametrization routines of LSCM [14] and ABF [21] are provided by INRIA's Graphite software [1]. Graphite is a research platform for computer graphics, 3D modeling and numerical geometry.

The first test case is the "small people" model. Figure 4.4(a) shows the original model. This model is modified from a man made model that remove some non-manifold vertices and small saliency features. The model is segmented into 9 parts by the our quadric surface extraction algorithm (Figure 4.4(b) and Figure 4.4(c)) is the approximated model using quadric proxies. We set κ to 3 to avoid generating quadric surface fitting result that the arms are disconnect to the body (Fig.4.3). The plane tolerance δ_{plane} is set to 0.005 to classify the four ends of limbs to plane. The minimum vertical spacing δ_{vs} is set to 0.4. Figure 4.4(d) shows the unfolded pattern of the quadric proxies and Figure 4.4(e) is the assembled papercraft model for the input mesh. From figure 4.4(d), we can see that the number of patches are 21. This is a reasonable number that user can afford. It takes about five to six hours to make the papercraft model.



(a) Original model

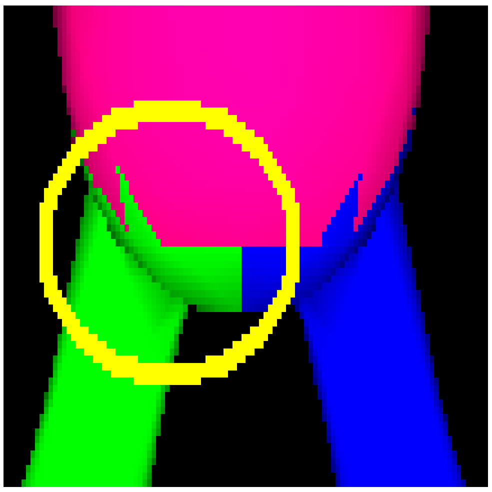
(b) Approximated model

Figure 4.1: The problem on the bottom of the body in ”small people” model.

There is a large distortion at the bottom of the body (Fig.4.1(a)). This problem is that there are only few triangles at the body’s bottom of the original model. When we fit the body part as an ellipsoid shape, the bottom of the original model was replaced as the quadric surface’s shape (Fig.4.1(b)). This kind of problems are also happened for legs (Fig.4.2(a) and Fig.4.2(b)). The boundary between body and legs looks quite different in original model and the approximated model because the boundary of approximated model is computed by quadric surface intersection.

Another test case is the ”chess” model. Figure 4.5(a) shows the original model. This model is a CAD-like model that is assembled by the CSG (constructive solid geometry) operations of several quadric primitives. The model is segmented into 9 parts by the our quadric surface extraction algorithm (Figure 4.5(b) and Figure 4.5(c)). The plane tolerance δ_{plane} is set to 0.00005 to classify the of bottom of the chess to plane. The minimum vertical spacing δ_{vs} is set to 0.4 to avoid generating too thin strip for the quadric type of hyperboloid of one sheet because users are hard to glue such a pattern to others. From Figure 4.5(d), there are 15 patches for the entire papercraft model. Figure 4.5(e) is the assembled papercraft model for the input mesh. It takes about four to five hours to make the papercraft model.

In this model, the largest error is happened at the pink part of fig.4.4(b) and fig.4.4(c). The number of triangles in this part and in its fuzzy region are nearly equal. So, the fuzzy segmentation of this part change the shape and type of its quadric surface heavily.


(a) Original model

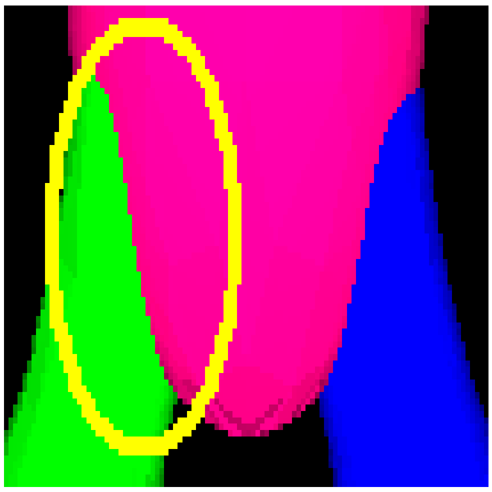

(b) Approximated model

Figure 4.2: The different between body and legs in original model and approximated model.

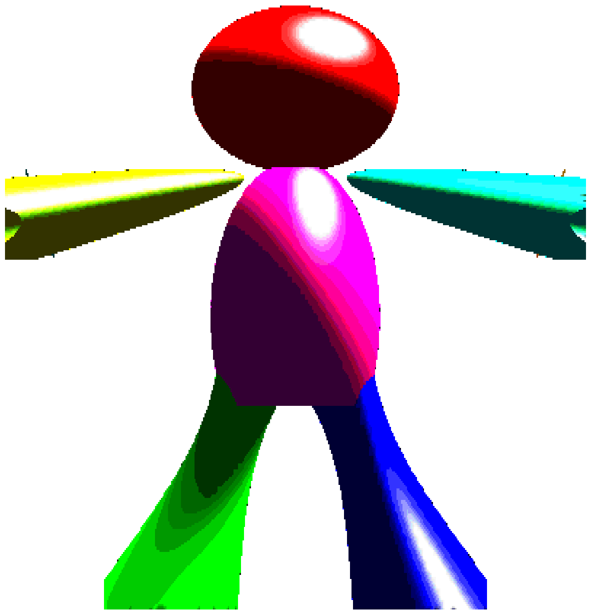
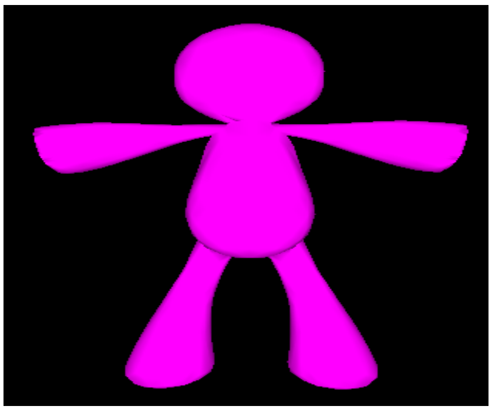
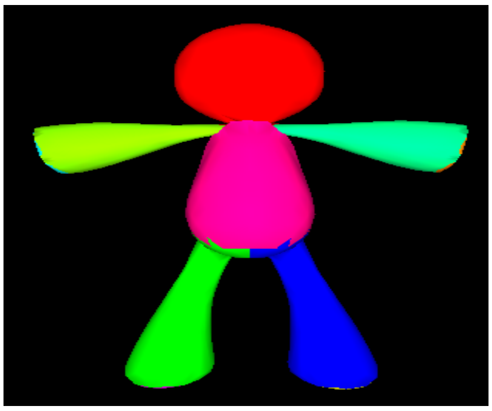
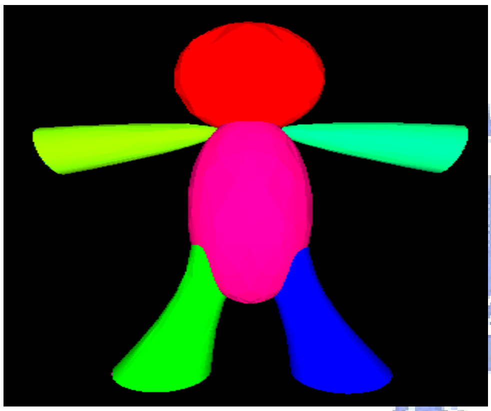
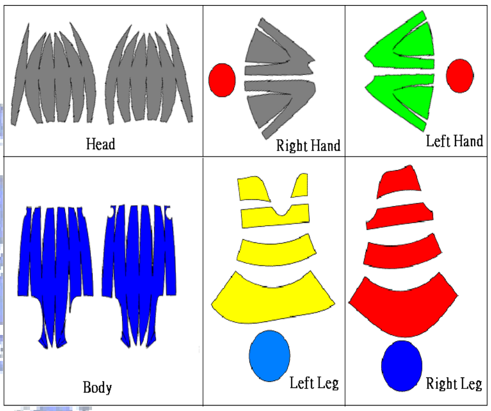


Figure 4.3: Quadric surface fitting result that the arms are disconnect to body.


(a) original model


(b) segmented model


(c) approximate by quadric patches



(d) part layout

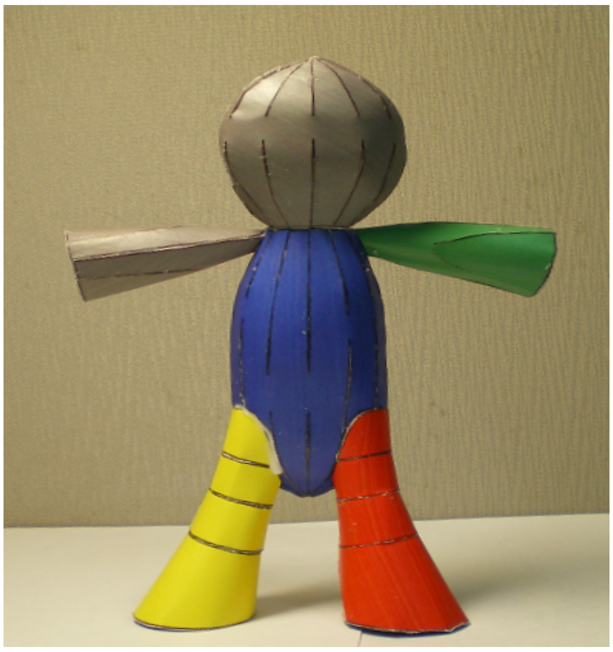
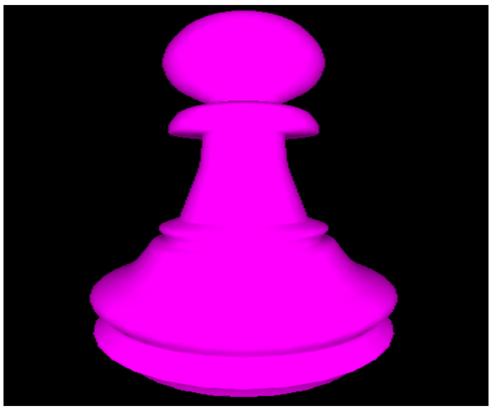
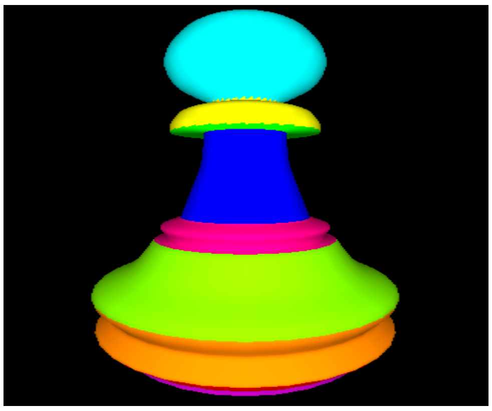
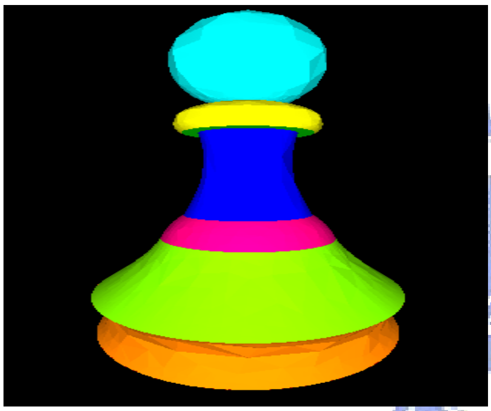
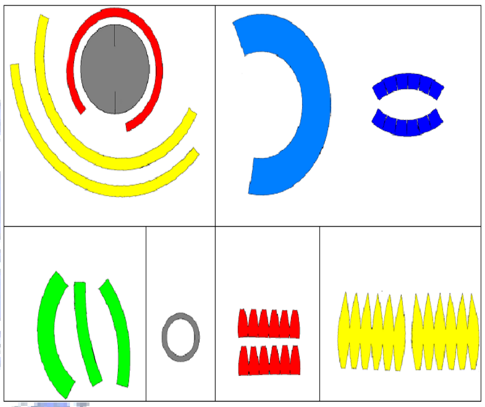

(e) papercraft model

Figure 4.4: The small people model.


(a) original model


(b) segmented model


(c) approximate by quadric patches


(d) part layout

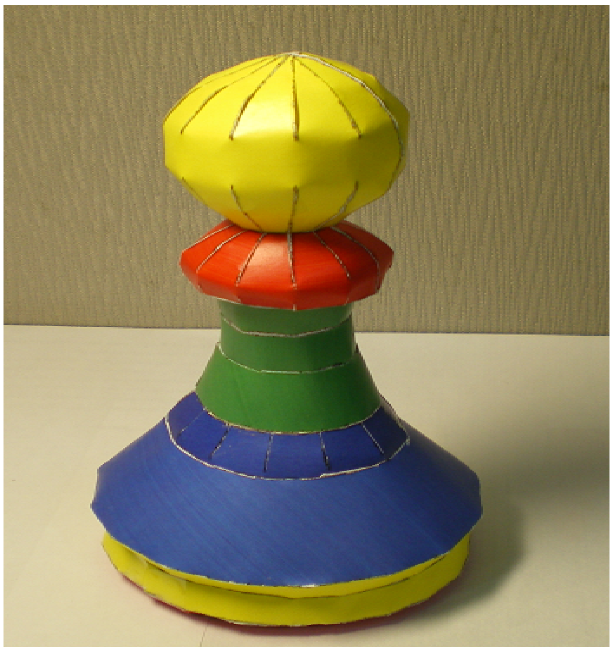

(e) papercraft model

Figure 4.5: The chess model.

CHAPTER 5

Summary

In this chapter, we give brief conclusion of the thesis, and suggest some direction of the future work.

5.1 Conclusion

We propose a method that can automatically generate papercraft models from meshes. This method use quadric surface as the basic fitting primitives to reduce the number of final 2D paper patterns and the assembled model become smoother than previous methods. In order to obtain meaningful 2D patterns, we use the pre-defined cutting rules for each quadric type. By using pre-defined cutting rules, user can predict the assembled shape of the 2D paper pattern. We also purpose the fuzzy segmentation method that can automatically fix the problem that segmentation using primitives fitting will sometimes generate disconnect surfaces. Our system also provides parameters to control the complexity of the resulting 2D patterns. The advantages of our system are listed below :

- Generate lesser patches than previous paper because we use a more complex primitives to approximate the input meshes.

- Our papercraft models are smoother than models that generated from the methods[17] [20] [18] (Fig.5.1).
- The boundary of 2D pattern is smooth and is easy to cut and glue.
- The 3D shape of 2D pattern is predictable for user due to the pre-defined cutting rule. The user can guess out that the 2D pattern of head of ”small people” model is an ellipsoid shape.
- Each 2D pattern is correspond to a meaningful part of original 3D mesh.

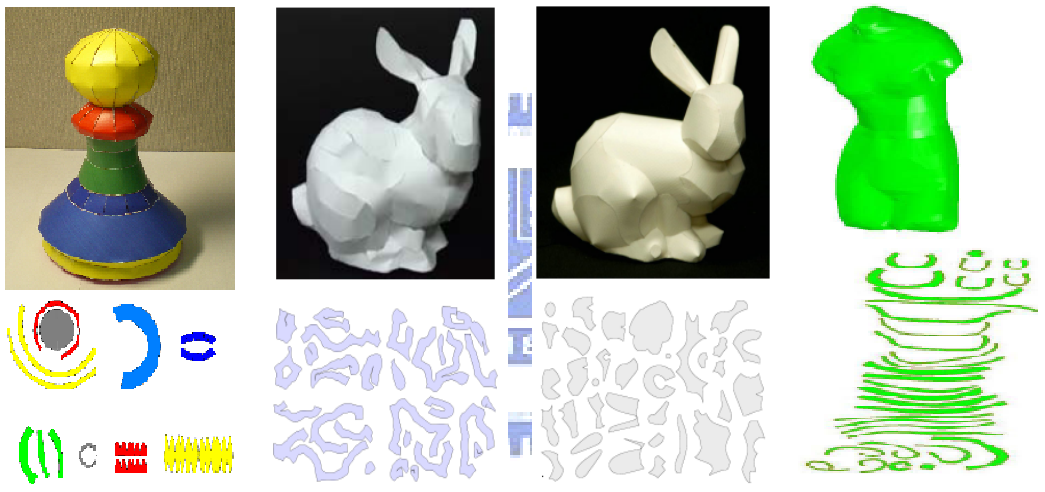

(a) Our result. (b) Jun et al. [18] (c) Shatz et al. [20] (d) Massarwi et al. [17]

Figure 5.1: The comparison of previous work [18] [20] [17] with our results.

5.2 Future Work

Currently, the stage of finding intersection curves between quadric surfaces are not stable. The process of tracing the entire intersection curve will diverge and go to infinite due to numerical error. We still look for a robust method to solve this problem.

The fuzzy region scheme still need to improve. We need a more efficient method to automatically connect those disconnect quadric surfaces without generating too many distortions.

Unfolding by several conformal parametrization methods and choose the best result does not really solve the problem that the boundary length of 2D pattern is not match to the length of 3D surface. We need to find parametrization method than is conformal and can preserve the boundary length exactly.

Bibliography

[1] Graphite. URL `http://alice.loria.fr/index.php?option=com_content&view=article&id=22`.

[2] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *Vis. Comput.*, 22(3):181–193, 2006.

[3] W. Boehm and H. Prautzsch. *Geometric concepts for geometric design*. A. K. Peters, Ltd., 1994.

[4] K. R. Borg and A. F. Bielajew. Quadplot: A programme to plot quadric surfaces. 1995.

[5] Z.-Q. Cheng, K. Xu, B. Li, Y. Wang, G. Dang, and S. Jin. A mesh meaningful segmentation algorithm using skeleton and minima-rule. In *ISVC (2)*, pages 671–680, 2007.

[6] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 905–914. ACM, 2004.

[7] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21:209–218, 2002.

[8] G. Elber. Model fabrication using surface layout projection. *Computer-Aided Design*, 27: 283–291, 1995.

[9] M. S. Floater. Mean value coordinates. *Comput. Aided Geom. Des.*, 20(1):19–27, 2003.

[10] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Comput. Aided Geom. Des.*, 14(3):231–250, 1997.

[11] D. Julius, V. Kraevoy, and A. Sheffer. D-charts: Quasi-developable mesh segmentation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, volume 24, pages 581–590. Eurographics, 2005.

[12] S. Katz, G. Leifman, and A. Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):649–658, 2005.

[13] T.-Y. Lee, Y.-S. Wang, and T.-G. Chen. Segmenting a deforming mesh into near-rigid components. *Vis. Comput.*, 22(9):729–739, 2006.

[14] B. Levy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph.*, 21(3):362–371, 2002.

[15] R. Liu and H. Zhang. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum (Special Issue of Eurographics 2007)*, 26:385–394, 2007.

[16] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[17] F. Massarwi, C. Gotsman, and G. Elber. Papercraft models using generalized cylinders. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 148–157. IEEE Computer Society, 2007. ISBN 0-7695-3009-5.

[18] J. Mitani and H. Suzuki. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Trans. Graph.*, 23(3):259–263, 2004.

[19] P. J. Schneider and D. Eberly. *Geometric Tools for Computer Graphics*. Elsevier Science Inc., 2002.

[20] I. Shatz, A. Tal, and G. Leifman. Paper craft models from meshes. *Vis. Comput.*, 22(9):825–834, 2006.

[21] A. Sheffer and E. de Sturler. Parameterization of faceted surfaces for meshing using angle-based flattening, 2001.

[22] A. Sheffer, B. Levy, M. Mogilnitsky, and A. Bogomyakov. Abf++: fast and robust angle based flattening. *ACM Trans. Graph.*, 24(2):311–330, 2005.

[23] J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148, pages 203–222. Springer-Verlag, May 1996.

[24] P. D. Simari and K. Singh. Extraction and remeshing of ellipsoidal representations from mesh data. In *GI '05: Proceedings of Graphics Interface 2005*, pages 161–168, 2005.

[25] G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(11):1115–1138, 1991.

[26] G. Taubin. An improved algorithm for algebraic curve and surface fitting. pages 658–665, 1993.

[27] J. Wu Leif Kobbelt. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum*, 24:277–284(8), September 2005.

[28] D.-M. Yan, Y. Liu, and W. Wang. Quadric surface extraction by variational shape approximation. In *GMP*, pages 73–86, 2006.

[29] R. Zayer, B. Levy, and H.-P. Seidel. Linear angle based parameterization. In *ACM/EG Symposium on Geometry Processing conference proceedings*, 2007.