Chapter 1

Geodesic Algorithm for Measurements

When making a cloth, the body of the customer need to be measured first to determine the size of the cloth. In anthropometric data acquisition, two types of measurements are associated with cloth making, length and circumference. The length measurements are usually obtained by calculating the euclidean distance between two datum points which are the landmarks on the skin. This procedure requires the character remains standing or sitting posture, the correctness of the posture significantly affects the accuracy of the measurements. In computer animation, the "T-Pose" is the standard posture for character modelling because the body and limbs are stretched straight so the space between different body part are maximized. This provides many convenience for rigging and texturing. Unfortunately, the concept of "T-pose" are ambiguous because different studios have different definition of the "T-Pose".

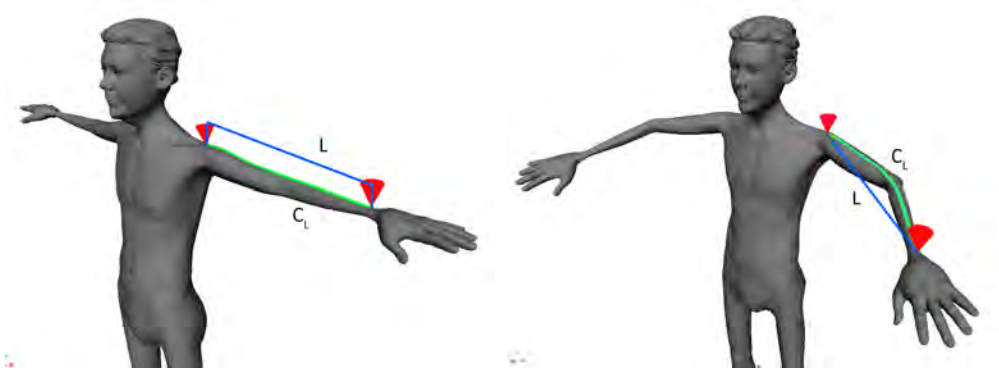
The changes of the posture might need different measuring method to cope with, For example, in order to measure the length of the arm, the character is required extend their arm straight, the length is the distance from

shoulder joint to the wrist joint. However, when measuring a character in a bent arm posture, the length of arm is acquired by adding the length of upper arm and lower arm. Another example is the acquisition of the height, for a character in a standing straight posture, the height can be obtained by calculating the Euclidean distance from top of the head to the bottom of the heel. However, when character bends or sits, the aforementioned measuring method no longer suitable for the circumstance. The height need to be measured separately from head, neck, torso and length of leg. With different posture, the datum point of measurements are also differs.

In tailoring, when measuring the human body for cloth making, one end of a measuring tape is held to one of two datum points and tension is applied to the tape ruler so that it follows the profile of our body as closely as possible when it reach the second of the datum point. In this thesis, geodesic is used for measuring characters. Comparing with the Euclidean distance calculation, geodesic distance calculates the distance between two points, moreover, similar as the tape ruler measuring techniques, geodesic path lies on the underlying surface. Therefore, by using geodesics, it can effectively measures the body of the character.

In computer graphic, if a character is rigged, that is, the character has skeleton to be associated with, the length of the body part, can be measured by acquiring the length of the skeleton associated with that part of body. However, the skeleton system only applies to the character that is used for dynamic purpose such as animation. For static purposed character such as the virtual character for still image, the measurements can only be acquired based on the datum points on the skin. In order to develop a measuring method for both static and dynamic purposed character, only the datum on the skin is considered.

When measuring character according to the datum points on surface, the Euclidean distance will change when surface is in different status. Fig-



(a) *Character in "T-pose" A* **(b)** *Character in "T-pose" B*

Figure 1.1: *Measurements in different posture*

ure.1.1 demonstrate different measurement caused by different posture. The red cone indicates the datum points on the shoulder and wrist of the character. The green curve indicates the geodesic generated by the method presented in this chapter between two datum points and blue line is the straight line connecting two datum points. L denotes the length of the blue line(the Euclidean distance between two datum points), C_L denote the arc length of geodesic. The character is in two most common "T-pose".

When the character in "T-pose" A, $L = 51.708$ and $C_L = 51.075$, when character in "T-pose" B, $L = 45.867$ and $C_L = 51.494$. This shows the geodesic distance between two datum points are more consistence in different posture than Euclidean distance when measuring length of the body part of character. Therefore in this thesis, the length measurement are acquired by measuring the geodesic between two datum points on the body of the character in order to cope with different postures of the character.

In order to calculate geodesic, a graph with positive edge weight is required because one of the fundamental requirement for geodesic computation is the connectivity between nodes in the graph. The discrete polyhedral surface is used as the most common shape representation in computer for a very long time and the topology of the discrete polyhedral surface suits this requirement very well. Therefore, in the past few decades, many algorithms

has been developed to compute the geodesic on discrete polyhedral surface.

Nowadays,with the easy access of 3D shape acquisition device, high-fidelity 3D scanned human body model has been widely accepted as simulation base model for clothing. However, The point cloud data from the 3D shape acquisition device does not have the connectivity between sampling points. This makes geodesic computation much more difficult.

This chapter presents a novel discrete geodesic computation scheme consists of three algorithms, accurate geodesic algorithm on polyhedral surface, approximate geodesic algorithm on polyhedral surface and approximate algorithm on point cloud model. Among these algorithms, the approximate algorithm for polyhedral surface is also able to achieve linear time complexity whilst maintain a bounded small error. Moreover, in order to measure the character represented by point cloud data, a geodesic algorithm for computing geodesics directly on point cloud model is also presented.

1.1 Introduction

For different shape representation, geodesic has different definitions.

Firstly, for parametric surface, the geodesic refers to a curve connecting two points on surface, such that the geodesic curvature at any point on the curve is zero (Polthier & Schmies 2006).

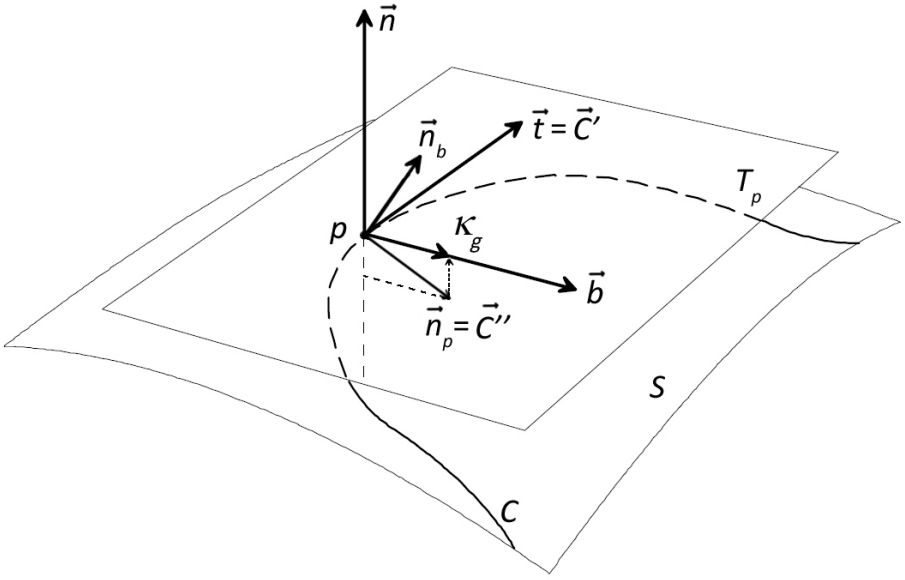


Figure 1.2: *Illustration of geodesic curvature*

Figure.1.2 demonstrate the geodesic curvature at a point on a curve, where S is a two-dimensional smooth surface. C is a curve on S, Tp is the tangent plane at the point $p \in S$, $\vec{n}$ is the normal at p, $\vec{C}'$ denote first order derivative of C and its the tangent vector at p. $\vec{b}$ is a vector perpendicular to $\vec{n}$ and $\vec{C}'$. $\vec{C}''$ denotes the second order derivative of C at p which is the geodesic curvature vector of C at p. $\vec{n_b}$ is the binormal unit vector that perpendicular to $\vec{C}'$ and $\vec{C}''$. κ_g is the projection of $\vec{C}''$ on $\vec{b}$.

Definition 1 *Let S be a two-dimensional parametric surface, a curve C is a geodesic if one of the equivalent properties holds(Polthier & Schmies 2006):*

1. *C is locally shortest curve.*
2. *$\vec{C}''$ is parallel to the $\vec{n}$.*
3. *C has vanishing geodesic curvature $\kappa_g = 0$.*

Secondly, in order to define geodesic on convex polyhedral surface, the

vertices of the polyhedral surface need to be categorized first. This can be done based on the total vertex angle. Polthier & Schmies (2006) defines total vertex angle as,

Definition 2 *Let S be a polyhedral surface and vertex $v \in S$. F_v denotes the one-ring neighbour faces of v, which can be written as $F_v = f_1, ..., f_i, ..., f_m$, θ_i is the interior angle of f_i at v. Then the total vertex angle $\theta(v)$ is given by,*

$$\theta(v) = \sum_{i=1}^{m} \theta_i(v)$$

All vertices of a polyhedral surface can be categorized based on the sign of the "vertex angle excess", which can be calculated by $2\pi - \theta(v)$. Figure.1.3 demonstrate three types of vertices on polyhedral surface.

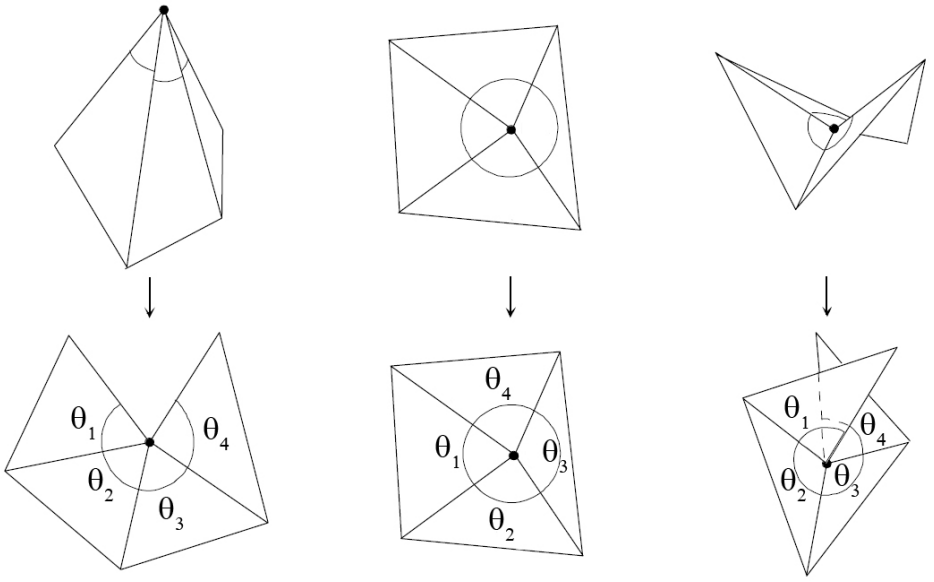


Figure 1.3: *Spherical Vertex, where $2\pi - \Sigma\theta > 0$ (left), Euclidean Vertex, where $2\pi - \Sigma\theta = 0$ (middle), Hyperbolic Vertex, where $2\pi - \Sigma\theta < 0$ (right).*

Therefore a geodesic on desecrate polyhedral surface can be defined as follow,

Definition 3 *Let S be a two-dimensional polygon surface, A piecewise curve C is a geodesic if one of the equivalent properties holds(Polthier & Schmies 2006): "A geodesic path P goes through an alternating sequence of hyperbolic vertices and (possibly empty) edge sequences such that the unfolded image of the path along any edge sequence is a straight line segment and the angle of the path passing through a vertex is greater that or equal to π. No edge can appear in more than one time in a edge sequence(Mitchell et al. 1987a)."*

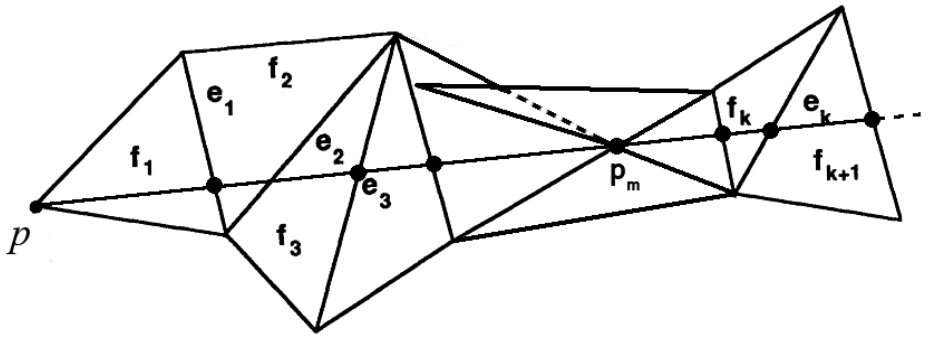


Figure 1.4: *Geodesic on a polyhedral surface. Where $f_1 \cdots f_k$ is the planar unfolding of S. $e_1 \cdots e_k$ is the edge sequence E the geodesic goes through. p_m is a hyperbolic vertex on the polygon surface.*

Figure.1.4 illustrate a geodesic starts at point p and goes through a sequence edge $e_1 ... e_k$.

According to Definition.4, given an initial piecewise curve that passes a series of vertices on the polyhedral surface, after defining the tangent plane at every point on this piecewise curve, by moving each point on the piecewise curve iteratively for a small step at direction of $\vec{b}$ indicated in Figure.1.2, the geodesic curvature κ_g can be diminished gradually. When this iteration converges, which means the geodesic curvature at every point on this piecewise curve has been diminished, this piecewise curve becomes a geodesic on the polyhedral surface.

In the following sections, a novel geodesic curvature flow based geodesic

computation scheme that is introduced in detail. Three algorithms has been developed to solve different problems,

1. Accurate geodesics on polyhedral surface
2. Linear time complexity approximate geodesics with an bounded error on polyhedral surface
3. Approximate geodesics on point clouds

1.2 Geodesic Curvature Flow

Based on the geodesic curvature flow defined in Definition.1, a scheme is proposed for computing geodesics over mesh and point clouds. This scheme employs iterative regression process to diminish the geodesic curvature, when the process converges, the curve becomes a geodesic.

Clairaut (1731); Serret (1851) introduced Frenet-Serret formulas to describe the kinematic properties of a particle moving on a continuous and differentiable curve in three-dimensional Euclidean space $\mathbb{R}^3$. In other words, this formulas describes the derivatives among the tangent vector, normal vector and binormal vector of a point on a continuous and differentiable curve. Frenet-Serret formulas is defined as follow,

Let C be a curve in Euclidean space that represents the position vector of the particle as a function of time, The FrenetSerret formulas apply to non-degenerate curves which have non-zero curvature.

Let $s(t)$ represent the arc length which the particle has moved along C in time t. The curve C then can be parametrized by it arc length s. A point on curve C that has arc length s can be denoted by $C(s)$. Then the FrenetSerret frame is defined by three vectors,

The first order derivative of C, also known as the tangent unit vector $\vec{t}$

is defined as:

$$\vec{C'} = \vec{t} = \frac{\mathrm{d}C(s)}{\mathrm{d}s} \tag{1.1}$$

The second order derivative of C, also known as the principle normal unit vector $\vec{n_p}$ of the curve is defined as:

$$\vec{C''} = \vec{n_p} = \frac{\frac{\mathrm{d}\vec{t}}{\mathrm{d}s}}{\left\|\frac{\mathrm{d}\vec{t}}{\mathrm{d}s}\right\|} \tag{1.2}$$

The binormal unit vector of C is defined as:

$$\vec{n_b'} = \vec{n_p} \times \vec{t} \tag{1.3}$$

Note that $\vec{n_b'}$, $\vec{t}$ and $\vec{n_p}$ are perpendicular to each other, the plane defined by $\vec{t}$ and $\vec{n_p}$ is the osculating plane at point $C(s)$.

In order to calculate the gradient for the iteration, firstly, a natural coordinate system need to be defined by the matrix form of Frenet-Serret formulas(Kreyszig 1991),

$$\begin{bmatrix} \frac{d\vec{t}}{ds} \\ \frac{d\vec{n_p}}{ds} \\ \frac{d\vec{n_b}}{ds} \end{bmatrix} = \begin{bmatrix} 0 & \kappa & 0 \\ -\kappa & 0 & \tau \\ 0 & -\tau & 0 \end{bmatrix} \begin{bmatrix} \vec{t} \\ \vec{n_p} \\ \vec{n_b} \end{bmatrix} \tag{1.4}$$

where $\tau = \langle -\vec{n}, b' \rangle$ denotes the torsion which measures the turnaround of the binormal vector $\vec{n_b}$, "$\langle , \rangle$" is the dot production of two vectors, $\kappa = 1/r$ is the curvature where r is the radius of the osculating circle. s is the arc

length of the curve. Equation.1.4 can be written into following form,

$$\begin{cases} \frac{d\vec{t}}{ds} = \kappa\vec{n_p} \\ \frac{d\vec{n_p}}{ds} = -\kappa\vec{t} + \tau\vec{n_b} \\ \frac{d\vec{n_b}}{ds} = -\tau\vec{n_p} \end{cases} \tag{1.5}$$

Based on Definition.1, let $\vec{n}$ denotes the standard unit normal of surface S, κ_p denotes the geodesic curvature and τ denotes the geodesic torsion. Equation.1.5 can be written as,

$$\begin{cases} \frac{d\vec{t}}{ds} = \kappa\vec{n} + \kappa_g\vec{b} \\ \frac{d\vec{n_p}}{ds} = -\kappa_g\vec{t} + \tau_g\vec{n} \\ \frac{d\vec{n_b}}{ds} = -\tau\vec{n} - \kappa_g\vec{t} \end{cases} \tag{1.6}$$

where $\kappa = \left\langle \vec{C''}, \vec{n} \right\rangle$ is the normal curvature at $C(s)$, $\kappa_g = \left\langle \vec{C''}, \vec{b} \right\rangle$ is the geodesic curvature at $C(s)$. $\tau_g = \tau - \frac{d\theta}{ds}$ is the geodesic torsion, θ is the angle between n and C''. $C^{''}{}_s$ denote the second order derivative to arc length s.

Now, the geodesic curvature flow can be defined as follow,

Definition 4 *(Chopp 1993), Let $S \subset \mathbb{R}^3$ be a two-dimensional manifold embedded in three-dimensional space. $C(s)$ is a curve on S that parametrised by arc length s and it is moving with speed $F(\kappa_g)$ in the direction perpendicular to $C(s)$, κ_g is the geodesic curvature of Cs on S, $\vec{n}$ denotes the normal of S and it is continuous on S. Therefore, at every point on $C(s)$, based on the Frenet-serret frame, a natural coordinate system can be given by the vectors $\vec{C'_s}$, $\vec{t_s}$ and $\vec{C''_s}$. C'_s is the first order derivative of $C(s)$, Thus, for any point $C(s)$, the velocity can be defined by geodesic curvature is,,*

$$\kappa_g = \left\langle (\vec{n} \times C'_s), C''_s \right\rangle \tag{1.7}$$

therefore, the moving direction of the geodesic curvature flow $C(s,t)$ is given by,

$$k_g\vec{b} = C_s'' - \left\langle C_s'', \vec{n} \right\rangle \vec{n} \tag{1.8}$$

where $\left\langle C_s'', \vec{n} \right\rangle \vec{n}$ is the projection of C_s'' on $\vec{n}$, note that t is a time variable of the flow in which when $t = 0$ states the initial statue of the flow $C(s,t)$.

This flow is also known as the Euclidean curve shortening flow(Salden et al. 1999). Spira & Kimmel (2002); Wu & Tai (2010) proved that the flow of Equation.1.8 can minimize the geodesic curvature $C_{s,t}''$ pointwise on $C(s,t)$ by moving $C(s,t)$ in the gradient direction.

Polthier & Schmies (2006) proved that, on polyhedral surface, if a discrete geodesic does not pass any hyperbolic vertex, this geodesic is also a shortest geodesic. In this thesis, the concept of vanishing geodesic curvature is exploited from smooth surface geodesic to the discrete surface geodesic. In computer graphic, a polyhedral surface can be considered as the inscribed polygon approximation of a smooth surface. Therefore the a curve on the smooth surface can also be approximated by a piecewise curve consisting of a set of fixed number of sample points. However, there is no parametric form for the piecewise curve $C(s)$ nor for the polyhedron surface. Especially for the large model, that has millions of vertices, to generate parametric approximation surface for the polyhedron and the piecewise curve require large amount of computation resources. To avoid this excess calculation, the second order derivative of $C(s)$ is directly defined on the piecewise curve by a triangle $\triangle p_{i-1}, p_i, p_{i+1}$ as shown in Figure.1.5.

where C denotes a curve on smooth surface S and the $C(s)$ is the parametrized form of C by its arc length. p_{i-1},p_i and p_{i+1} are three sample points on C and p_i is a point on C with an arc length s to curve C. Thus $\widetilde{p_{i-1}p_ip_{i+1}}$ denotes the a section of the piecewise approximation of smooth

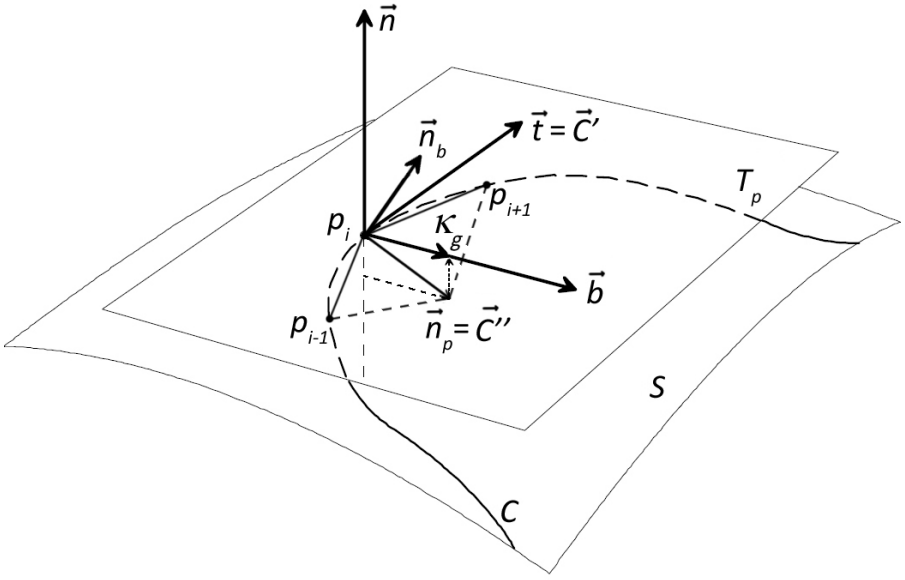


Figure 1.5: *Geodesic curvature on piecewise curve*

curve C. In this case, at p_i, the second order derivative of Cs at p_i can be estimated by Equation.1.9,

$$C_s'' \approx \overrightarrow{p_i p_{i+1}} + \overrightarrow{p_i p_{i-1}} \tag{1.9}$$

In order to define the natural coordinate system on a vertex of a polyhedral surface, the tangent plane of S at p_i need to be confirmed. Assume the polyhedral surface is a discrete approximation of smooth surface S, given a point p_i on $C(s)$, the tangent plane of the polyhedron at vertex p_i can be defined by the fitting plane of the one-ring neighbour vertices of p_i. According to Definition.1, it can be said that, if the projection of C_s'' onto this tangent plane at p_i vanishes, the geodesic curvature will also vanish at p_i accordingly. Furthermore, if the geodesic curvature at all the point on the piecewise approximation curve of $C(s)$ vanishes, $C(s,t)$ reaches a stable states, and becomes a geodesic.

Let P denotes all the points on the piecewise approximation curve of

$C(s)$, the geodesic curvature of $C(s)$ can be written as $C_s'' = KP$.

$$K = \begin{bmatrix}
0 & 0 & 0 & \cdots & & & & & & & \\
0 & 0 & 0 & \cdots & & & & & & & \\
0 & 0 & 0 & \cdots & & & & & & & \\
1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & 0 & & \cdots \\
0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & & \cdots \\
0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & & \cdots \\
 & & & 1 & 0 & 0 & -2 & 0 & 0 & 1 \; 0 & 0 \cdots \\
 & & & 0 & 1 & 0 & 0 & -2 & 0 & 0 \; 1 & 0 \cdots \\
 & & & 0 & 0 & 1 & 0 & 0 & -2 & 0 \; 0 & 1 \cdots \\
 & & & & & \cdots & & & \cdots & & \\
 & & & & & \cdots & & & \cdots & 0 \; 0 & 0 \\
 & & & & & \cdots & & & \cdots & 0 \; 0 & 0 \\
 & & & & & \cdots & & & \cdots & 0 \; 0 & 0
\end{bmatrix}_{3m \times 3m}$$

and

$$P = [p_{1x}, p_{1y}, p_{1z}, \cdots, p_{mx}, y_{my}, p_{mz}]^T$$

where K is a $3m \times 3m$ coefficient matrix and m denotes the number of the sample points on piecewise curve. Now, geodesic curvature flow $C(s,t)$ can be rewritten as,

$$C(s,t) = k_g \vec{b} = KP - \vec{n}\vec{n}^T KP \tag{1.10}$$

where $\vec{n}$ is a block diagonal matrix consisting of the normal vectors of the tangent plane at all the point in P,

$$\vec{n} = \begin{bmatrix} n_{1x} & & & \\ n_{1y} & & & \\ n_{1z} & & & \\ & n_{2x} & & \\ & n_{2y} & & \\ & n_{2z} & & \\ & & \ddots & \\ & & & n_{mx} \\ & & & n_{my} \\ & & & n_{mz} \end{bmatrix}_{3m \times m}$$

Let P_t denotes all the sample point on $C(s)$ at t time, μ denotes a iterative step length. when t evolves into $t+1$, all the points in P_t move towards the direction of $\vec{b}$. Therefore, the updated curve denoted as P_{t+1} can be expressed as,

$$P_{t+1} = P_t + \mu(KP_t - \vec{n}\vec{n}^T KP_t) \tag{1.11}$$

1.3 Geodesic on Mesh

1.3.1 Tangent Space Constraint

In order to calculate the geodesics on polyhedral surface using Equation.1.11, the normal vector needs to be estimated at every iteration for all the points on polyhedral surface corresponding to every sample point till Equation.1.11 converged.

In our experiment presented in this chapter, during the iteration, the updated sample points tend to deviate from the surfaces. To solve this problem, a constrain in the tangent space needs to be applied to the movement of the

sample points. This can be implemented as follows.

Firstly,P denotes all the sample points on curve C, all the normal vectors of the face that within one-ring neighbour of $p_i \in P$ is stored in to a matrix N,

$$N = \begin{bmatrix} n_{1_x} & n_{1_y} & n_{1_z} \\ & \vdots & \\ n_{m_x} & n_{m_y} & n_{m_z} \end{bmatrix}_{m \times 3}$$

Applying PCA(Jolliffe 2002) to N; the tangent space are defined by $\vec{n}_i$ which denotes the eigenvector associated with the smallest eigenvalue. After that, for each $p_i \in P$, let $p_{i,t}$ denotes the current position of p_i and $p_{i,t+1}$ denotes the p_i after it moves to the direction of its geodesic curvature vector. This update of location of p_i can be expressed by,

$$p_{i,t+1} = p_{i,t} + \mu \vec{n}_i \vec{n}_i^T \left(\overline{p_{i,t}} - p_{i,t}\right) \tag{1.12}$$

where, μ is an iterative step size and $\overline{p_{i,t}}$ denotes the projection of p_i on the tangent plane at the closest vertex to p_i on S. In essential, the tangent space constraint of Equation.1.12 moves a point within the tangent space and therefore it tends to preserve the characterization of the underlying surfaces. After that, a steady solution can be formed by combining Equation.1.12 with Equation.1.11,

$$P_{t+1} = P_t + \mu(\vec{n}\vec{n}^T\overline{P_t} - \vec{n}\vec{n}^T P_t + KP_t - \vec{n}\vec{n}^T K P_t) \tag{1.13}$$

where, $\overline{P_t}$ is a column vector consisting of the coordinates of all the projections of the sampling point of curve C on S.

1.3.2 Implicit Euler scheme

In order to solve Equation.1.13, the explicit method (Euler method) is a straightforward solution(Butcher 2008). However, because Euler method is a first-order method, its local error is proportional to the square of the step size, and its global error is proportional to the step size(Butcher 1987). In order to maintain the stability of the solution, integration must be proceed with a very small iterative step size. Moreover, this iteration performed for every geodesic on a mesh, therefore results in a very time-consuming process.

The basic idea of the implicit Euler method is to employ the implicit differencing, i.e, the right-hand side of Equation.1.13 is evaluated at the new location of $t+1$. This is called as the backward Euler scheme(Enns & McGuire 2000). Although the error of the backward Euler scheme is the same as explicit Euler method, the method could maintain its stability with any step size(Hairer 2010). By increasing the step size of the iteration, the implicit Euler method could achieve much higher efficiency(Butcher 2008).

Applying the backward scheme to Equation.1.13 yields,

$$(I + \mu(\vec{n}\vec{n}^T - K + \vec{n}\vec{n}^T K))P_{t+1} = P_t + \mu\vec{n}\vec{n}^T\overline{P_t} \tag{1.14}$$

where I is an identity matrix, μ denotes the step size. The resulting geodesic curvature flow of Equation.1.14 is remain stable even at $\mu \rightarrow \infty$(Hundsdorfer & Verwer 2003).

Equation.1.14 is the linear equation form of $AX = B$ where $A = I + \mu(\vec{n}\vec{n}^T - K + \vec{n}\vec{n}^T K)$, $B = p_t + \mu\vec{n}\vec{n}^T\overline{P_t}$. This equation can be solved efficiently by LU factorization with full pivoting presented by Trefethen & Bau (1997).

1.3.3 Least Squares scheme

Although the implicit Euler scheme provides an efficient method for solving Equation.1.14. The experiment shows that the geodesics result from Equation.1.14 still requires a few iterations in order to achieve convergence.

Experiments show that during the convergence of Equation.1.11 , all the sample points on the geodesic curve are updated within the tangent space except two endpoints p_1 and p_m of curve $C(t)$. Therefore, Equation.1.11 can be written in a least squares form in which two endpoints p_1 and p_m are the constrains(Jiang 1998),

$$\left(K - \vec{n}\vec{n}^T K\right) p = [p_{1_x}, p_{1_y}, p_{1_z}, 0, ..., 0, p_{m_x}, p_{m_y}, p_{m_z}]^T \tag{1.15}$$

where $K =$

$$\begin{bmatrix}
1 & 0 & 0 & 0 & \cdots & & & & & & \cdots & \\
0 & 1 & 0 & 0 & \cdots & & & & & & \cdots & \\
0 & 0 & 1 & 0 & \cdots & & & & & & \cdots & \\
1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & 0 & & \cdots & \\
0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & & \cdots & \\
0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & & \cdots & \\
 & & & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & 0\cdots \\
 & & & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0\cdots \\
 & & & 0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1\cdots \\
 & & & & & & \cdots & & & \cdots & & \\
0 & \cdots & & & & & & & & 1 & 0 & 0 \\
0 & \cdots & & & & & & & & 0 & 1 & 0 \\
0 & \cdots & & & & & & & & 0 & 0 & 1
\end{bmatrix}_{3m\times 3m}$$

where $\vec{n}$ denotes the normal of the tangent plane. Essentially, the system Equation.1.15 is the 1^{st} order approximation to the geodesic curvature flow.

A linear system can be constructed from Equation.1.15,

$$\begin{bmatrix} A_{0,0} & A_{0,1} & \cdots & A_{0,3m} \\ A_{1,0} & A_{1,1} & & \\ \vdots & & \ddots & \vdots \\ A_{3m,0} & & \cdots & A_{3m,3m} \end{bmatrix} p = \begin{bmatrix} p_{1_x} \\ p_{1_y} \\ p_{1_z} \\ 0 \\ \vdots \\ 0 \\ p_{m_x} \\ p_{m_y} \\ p_{m_z} \end{bmatrix} \tag{1.16}$$

where, $A = K - \vec{n}\vec{n}^T K$, m is the number of the sampling point on the piecewise curve C. Combining Equation.1.16 with tangent plane constrain defined by Equation.1.12, the Equation.1.16 can be written as,

$$\begin{bmatrix} A_{0,0} & A_{0,1} & \cdots & A_{0,3m} \\ A_{1,0} & A_{1,1} & & \\ \vdots & & \ddots & \vdots \\ A_{3m,0} & & \cdots & A_{3m,3m} \\ N^*_{0,0} & N^*_{0,1} & \cdots & N^*_{0,3m} \\ N^*_{1,0} & N^*_{1,1} & & \\ \vdots & & \ddots & \vdots \\ N^*_{3m,0} & & \cdots & N^*_{3m,3m} \end{bmatrix} p = \begin{bmatrix} p_{1_x} \\ p_{1_y} \\ p_{1_z} \\ 0 \\ \vdots \\ 0 \\ p_{m_x} \\ p_{m_y} \\ p_{m_z} \\ B^*_{0,0} \\ \vdots \\ B^*_{3m,0} \end{bmatrix} \tag{1.17}$$

where, $N^* = \vec{n}\vec{n}^T$, $B^* = \vec{n}\vec{n}^T\bar{P}$. $\bar{P}$ is a column vector consist of all the $\bar{p}_i$ which are the projections of the p_i on the tangent plane defined by $\vec{n}$

and the closest vertex on S to p_i. m denotes the number of sample point on curve. The coefficient matrix on the left hand side of the system has size of $(2 * 3m) \times 3m$ and the one on the right hand side of the system has the size of $(2 * 3m) \times 1$.

Note that Equation.1.17 cannot compute a geodesic curve from scratch but requires the initial location of the sample points for computing $\bar{p}$. The vertices near a geodesic path are usually taken as the initial sample points on the path. Such initial sample points are used for computing $\bar{p}$ and $\vec{n}$ in Equation.1.17, which is then solved to obtain the final sample locations. According to this scheme, the final location of a sample point is still constrained within the tangent space defined for its initial location. Note that the solution of Equation.1.17 cannot guarantee the updated sample points lying on the polyhedral surface since the tangent space and actual polyhedral surface space are two different surface. the actual polyhedral surface is reckoned as inscribed polygon of the surface that represented by tangent space. In the case that require the geodesic path on polyhedral surface, a projection method was developed to project the geodesic path on smooth surface to its approximate mesh.

1.3.4 Geodesic Path Projection

Although the sample point updated by Equation.1.13 only moves within the tangent space, the actual mesh which the geodesics are calculated on is not flat. Therefore, the updated points tend to derive from the mesh. This is showed in Figure.1.6,

In order to calculate geodesics on the polyhedral surface, the offset geodesic curve need to be projected on to the polyhedral surface. Polthier & Schmies (2006) defines geodesic as when at any point on the curve, geodesic curvature vanishes. On polyhedral surface, if a curve fits this defi-

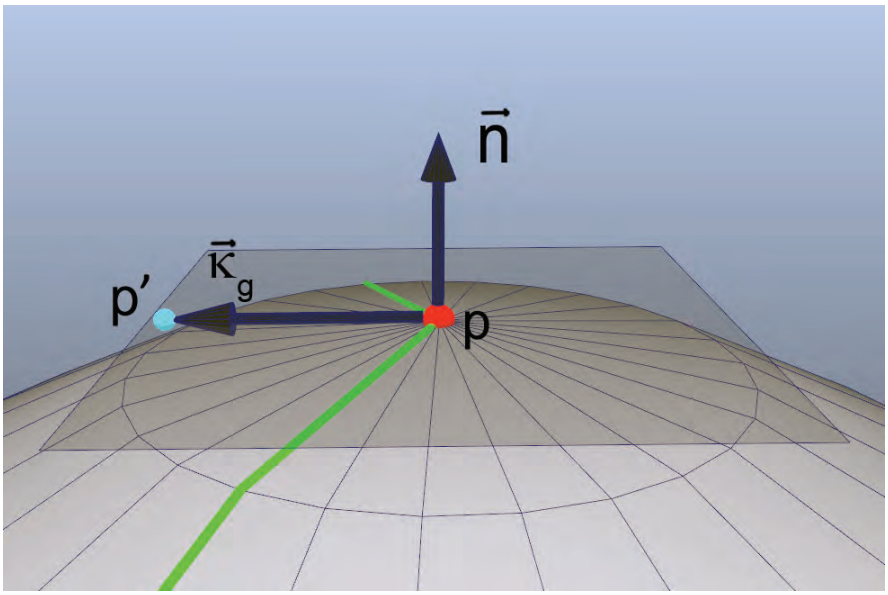


Figure 1.6: *Derived point caused by updating point. green piecewise curve is the curve before updated by Eq.1.11. gray plane is the tangent plane at point p (red point). the point p' (light blue point) is the updated point.)*

nition and does not pass any spherical vertex on the polyhedral surface, it is both straightest and locally shortest geodesic. This is shown in Figure.1.7.

In Figure.1.7 both the yellow line $\widetilde{p_1p_2p_3}$ and blue lines $\widetilde{p_1p_4p_3}$, $\widetilde{p_1p_5p_3}$ are the projections of their corresponding red geodesic path on sphere. The path $\widetilde{p_1p_2p_3}$ that goes through the spherical vertex p_2 is the straightest but not shortest geodesic path on the cubic. However, the blue lines that pass p_4 or p_5 is both the straightest and shortest path on cubic by the definition presented in (Polthier & Schmies 2006). Note that $\widetilde{p_1p_4p_3}$ and $\widetilde{p_1p_5p_3}$ are the projections of the great circle of the sphere on the direction of the curvature of their corresponding great circle.

In order to project the floating geodesic onto mesh, firstly, the direction of the projection need to be defined. According to the definition of geodesic curvature, geodesic curvature vector $\vec{\kappa_g}$ of a curve C at point p is the projected vector of the curvature vector $\vec{C''}$ at p onto the tangent plane T_p at point pP on surface S. This is shown in Figure.1.2. Therefore, the curvature κ at point p can be written as:

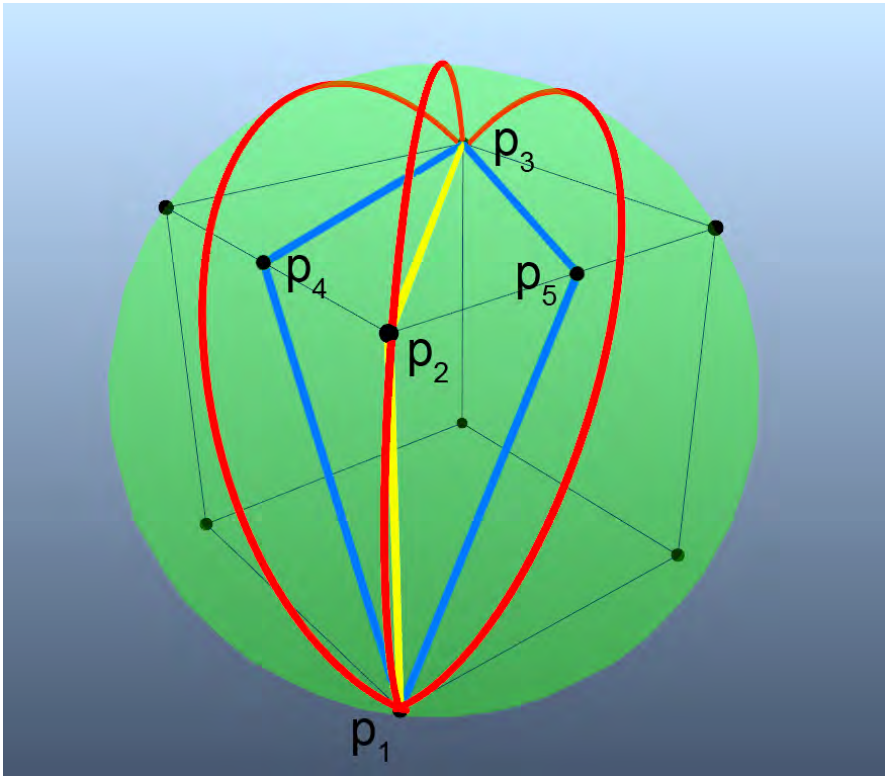


Figure 1.7: *Geodesic on smooth surface and its projection on the its inscribed polygon mesh. The green sphere contains its inscribed cubic. The red curves is the geodesic on sphere (great circle arc).* p_1,p_2 *and* p_3 *are points on sphere.* p_4 *and* p_5 *are the centre point on its edge.*

$$\vec{\kappa} = \vec{\kappa_g} + \vec{\kappa_n}$$

where $\vec{\kappa_g}$ is the component of $\vec{C''}$ along the $\vec{b}$, $\vec{\kappa_n}$ is the component of $\vec{C''}$ along the $\vec{n}$. $\vec{n}$ is the normal vector of point p on S. Let the magnitude of the vector $\vec{\kappa_g}$ is denoted by κ_g, then $\vec{\kappa_g} = \kappa_g\vec{b}$. the scalar κ_g is called geodesic curvature of C at p. Let κ be the magnitude of vector $\vec{C''}$. Then the curvature $\vec{C''}$ of curve C at p can be expressed by geodesic curvature k_g at p by:

$$\kappa_g = \kappa cos\theta$$

where θ is the angle between the osculating plane of C at point p and the tangent plane T_p. Therefore, is we move p along the direction of $\vec{C''}$, the osculating plane of C at point p will remain same so as the θ. Hence the geodesic curvature κ_g will remain identical. Therefore, for each sample point

on c the projection direction is its C''. However, since there is no guarantee that every successive sample point can projected into the successive faces on polygon, actually, in our experiments, in most cases, the successive sample point of current sample point will falls onto the face that outside of the one-ring face neighbour of the face contains current projected point. Here, by using osculating plane of C at p, we have developed a method connecting all the projected point of p. The procedure is described as follows.

The source point p_0 is firstly selected as the starting point that always lies on a vertex of the mesh. The successive 2 sample points p_1,p_2 of a floating geodesic form a plane cutting the mesh. This plane, $\Delta p_0 p_1 p_2$, is viewed as an approximation of the osculating plane Figure.1.8.

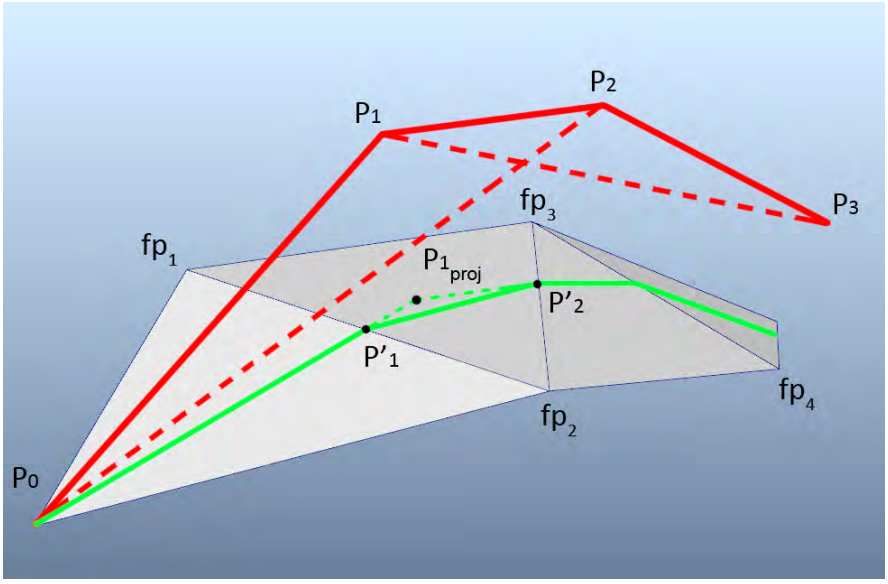


Figure 1.8: *Projection of a geodesic path.*

In Figure.1.8, the red line indicate the piecewise approximation of geodesic. p_0,p_1,p_2,p_3 is the sample point on the geodesic path $\widetilde{p_0 p_3}$. $\Delta p_0 p_1 p_2$ is the approximation of the osculating plane at p_1. fp_1 fp_4 are the vertices on polyhedral surface. $P_{1_{proj}}$ is the projection of p_1. p_0 denotes the source point of geodesic path, p'_1 and p'_2 are the point on projected geodesic path which depicted in green line.

Let $\vec{N_1}$ denotes the normal vector of osculating plane $\Delta p_0 p_1 p_2$ and $\vec{N_2}$ denotes the normal vector of face $\Delta p_0 f p_1 f p_2$. Firstly, starts from the source point p_0, the rectifying plane of projected path is approximated by $\Delta p_0 f p_1 f p_2$. As a result, the tangent vector of the projected geodesic at p_0 can be calculated by Equation.1.18.

$$\vec{v} = \vec{N_1} \times \vec{N_2} \tag{1.18}$$

where $\vec{v}$ is equivalent to $\vec{p_0 p_1'}$. Follow this direction, the $\vec{v}$ intersects with an edge within the one-ring face neighbour of p_0 . p_1' denotes this intersection on the edge $\overline{f p_1 f p_2}$. Then, p_1' is selected as the next starting point. A important property of manifold mesh is that, only one or two faces is incident to a edge. Therefore, the next projected face can be easily determined as the adjacent face $\Delta f p_1 f p_2 f p_3$ to the current face $\Delta p_0 f p_1 f p_2$ through edge $\overline{f p_1 f p_2}$.

If the projection of p_1 does not falls into $\Delta f p_1 f p_2 f p_3$, the next intersected point on face $\Delta f p_1 f p_2 f p_3$ can be determined by performing Equation.1.18. If the projection of p_1 falls into the face $\Delta f p_1 f p_2 f p_3$ as shown in the Figure.1.8, $\vec{N_1}$ is updated by the normal vector of approximated osculating plane $\Delta p_1 p_2 p_3$ at point p_2. Starting from the projection $p_{1_{proj}}$ of p_1, perform Equation.1.18, the next intersection of $\vec{v}$ and $\overline{f p_2 f p_3}$ can be determined as p_2' shown in the Figure.1.8. The projection method is summarized in Algorithm.1

1.3.5 "Continuous Dijkstra" Propagation

As described in Equation.1.11, this algorithm needs an initial path to calculate the updated geodesic path. "continuous Dijkstra" strategy presented by Dijkstra (1959) is used to generate this initial path. When performing "continuous

Algorithm 1 Geodesic Projections on Mesh

procedure PROJECT GEODESIC PATH ON MESH(A mesh S, and a floating geodesic path $\widetilde{p_0 p_n}$)
  for $i = 1, i < n, i++$ **do**
    Select the successive vertices of p_i and build osculating plane $\triangle p_{i-1} p_i p_{i+1}$
    $\vec{N_1} \leftarrow$ normal vector of $\triangle p_{i-1} p_i p_{i+1}$
    $F_{end} \leftarrow$ Face contains projection of p_i on S
    select p_{i-1} as the starting point p_s
    while $F' \neq F_{end}$ **do**
      for $F' \in$ adjacent faces of p_s **do**
        $\vec{N_2} \leftarrow$ normal vector of current face
        $\vec{v_{tan}} \leftarrow p_s + \vec{N_1} \times \vec{N_2}$
        Compute the intersection point p' that an edge of current face F' intersects with $\vec{v_{tan}}$
        if p' is within the edge **then**
          Add p' into projection path
          Update p_s by p'
          Update F' by the adjacent face of F'
        end if
      end for
    end while
    Update p_s by p_i
  end for
end procedure

Dijkstra" strategy, all the points in propagation boundary are kept sorted by the distance back to the source point in incremental order in a priority queue. At each step, starts from the first element in the priority queue, the boundary is propagated outward. New point are inserted into the queue at the location where the order of the queue remains.

However, keeping the priority queue in order involves "comparison sorting algorithm" that involves a searching operation cost $log(n)$ time to n factorial possible orderings of a data set (Cormen et al. 2001). This process requires significant amount of time to complete especially when performing on large model. Because Equation.1.11 only requires an initial path that approximates to the shortest path only goes through vertices and edges. The vertices in the propagation boundary does not need to be kept in order. This algorithm is presented in more detail below.

Starting from the source p_s, firstly, the 1^{st} ring vertices of p_s are pushed into an array denoted as "wavefront" W that is equivalent to the propagation boundary. Each edge that connect p_s and the points in W is the geodesic path between them. This is due to with in a triangle, the shortest between two point is the edge that connects these two points. After that, the length of every edge is stored along with the path information to the vertices in W. Since this algorithm does not employ priority queue. To ensure the isometric propagation. A propagation radius limit r_{max} is introduced to constrain the propagation at every propagation step. At this moment, r_{max} is updated as the length of the longest edge that connected from p_s to W.

After the first W is formed by the one-ring neighbour of p_s, the first vertex $p_i \in W$ is selected and its one-ring neighbour excludes the vertices that has been visited are stored into an array N'. With a vertex $pn_i \in N'$, the next step is to determine its parent node that pn_i is connected to form its initial path at pn_i. This process is illustrate in Figure.1.9.

Where, pn_1 and pn_2 is the one-ring neighbour N' of p_i that have not

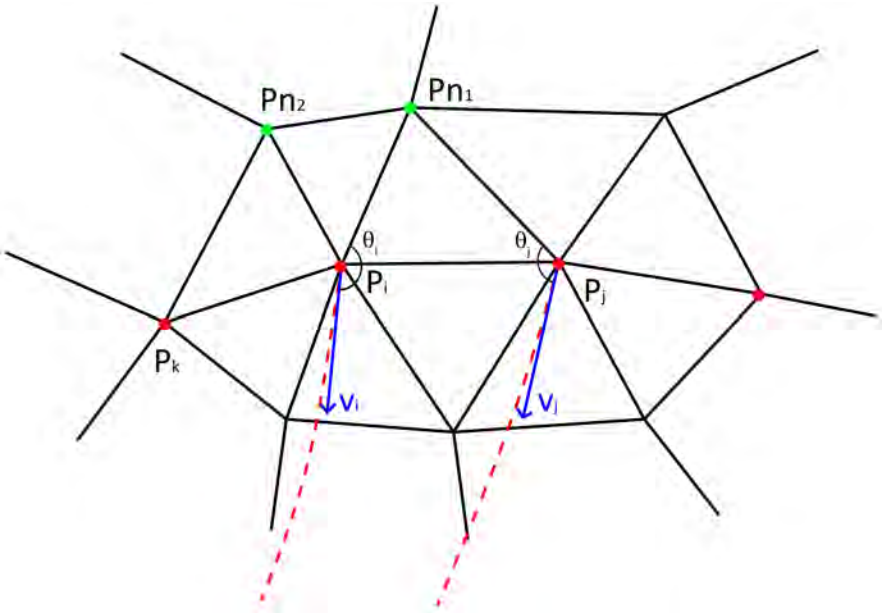


Figure 1.9: *Calculating geodesic path for a unvisited vertex. red point p_i,p_j and p_k are visited vertex in wavefront W. red dash line indicates geodesic path from source to p_i and p_j. vector $\vec{v_i}$ and $\vec{v_j}$ is the tangent vector at p_i and p_j respectively. pn_1 and pn_2 are one-ring neighbour of p_i that has not been visited*

been visited. For a random vertex $pn_1 \in N'$, the visited vertices in 1-ring neighbour of pn_i is selected. In Figure.1.9, p_i and p_j are selected. In the next step, for all the visited vertices that connect to pn_1, the angle θ between $\vec{v_i}$ and the edge that connects pn_1 and current selected visited vertex is calculated. $\vec{v_i}$ denotes the tangent vector at current visited vertex on its geodesic path. The parent node of p_i is select with the visited vertex with the largest θ.

In Figure.1.9, the included angle θ_i between $p_i\vec{pn}1$ and $\vec{v_1}$ is greater than the included angle θ_j between $p_j\vec{pn}1$ and $\vec{v_2}$. Therefore, it can be observed that, compared to the two curves of edge $\overline{p_ipn_1}$ plus geodesic path at p_i and ege $\overline{p_jpn1}$ plus geodesic path at p_j, the curve of edge $\overline{p_ipn_1}$ plus geodesic path at p_i are straighter than t he curve of edge $\overline{p_ipn_1}$ plus geodesic path at p_i. Thus, vertex pn_1 is pushed into the end of the geodesic path at vertex p_i as the initial path for pn_1. For every new point that pushed to the end of its parent point's geodesic path, the geodesic path information inherits from the

parent vertex to its child. As a result, the algorithm presented here does not require "backtracing" that are necessary for MMP, CH and FMM algorithm to retrieve the geodesic path.

The aforementioned method propagates geodesic information from the interior rings to external ones. However, without the help of priority queue, there is no guarantee to propagate geodesic information from the vertex with smaller geodesic distances to the source to vertex with larger distances. Therefore, isotropy need to be maintain during the propagation in order to ensure the correctness of the causality of the algorithm that presented in this chapter. Hence, a propagation radius limit r_{max} is employed here. After Pn_1 has selected p_i as its parent node, the estimate geodesic distance d_{est} from source to pn_1 can be approximated by the length of edge $\overline{pn_1p_i}$ plus geodesic distance from source to p_i. The Equation.1.11 is performed to calculate the geodesic path from source to pn_1 only if d_{est} is smaller than the current r_{max}. After that, p_i is stored into an array W' as the "new wavefront". After all the vertices in W has been popped out, W is replaced by W'.

However, if d_{est} is larger than the current r_{max}, then pn_1 is put into an array W_{next} as the "next wavefront" for the next propagation when both W and W' are empty. For every unvisited vertex that connected to the visited vertex in W, only the largest d_{est} is kept as d_{maxEst}. When both W and W' are empty, the W is replaced by W_{next} and r_{max} is replaced by d_{maxEst} as the new propagation limit. By using this method, within this limitation r_{max}, our method still propagates information from the interior rings to external ones. The r_{max} forms an isometric line on the polyhedral surface that constrains the propagation. Because r_{max} is updated every time when W is replaced by W_{next}, as a result, in between the updates of r_{max}, the length of all geodesic paths that are calculated are in between the r_{max} and d_{maxEst}. Therefore, the causality is guaranteed by r_{max}. This propagation algorithm is summarized in Algorithm.2.

Algorithm 2 Accurate Geodesics Algorithm

procedure INITIALIZATION(A mesh S, and a source p_s)
  for $p_i \in S$ **do**
    $p_i.geoDist \leftarrow \infty$
    $p_i.geoPath \leftarrow \emptyset$
  end for
end procedure

procedure CALCULATE ONE RING NEIGHBOUR OF p_s(W , r_{max})
  for $p_i \in W$ **do**
    $d_{est} \leftarrow Dist(p_i, p_s)$
    $p_i.geoDist \leftarrow d_{est}$
    $p_i.geoPath \leftarrow p_s + p_i$
    if $currentDist > r_{max}$ **then**
      $r_{max} \leftarrow d_{est}$
    end if
  end for
end procedure

function GETPARENTNODE(W,v)
  for $v' \in oneRingNeighbourOf(v)$ **do**
    $\theta = 0$
    if $v' \in W$ **then**
      $\vec{v_1} \leftarrow v - v'$
      $\vec{v_2} \leftarrow getTangent(v)$ ▷ Return the tangent vector of geodesic path from source to v' at point v'
      if $\theta < angleBetween(\vec{v_1}, \vec{v_2})$ **then**
        $\theta \leftarrow angleBetween(\vec{v_1}, \vec{v_2})$
        $parentNode \leftarrow v'$
      end if
    end if
  end for
  return $parentNode$
end function

Algorithm 2 accurate geodesics algorithm (continued)

procedure CONTINUES DIJKSTRA PROPAGATION(Calculate geodesic for every $p_i \in S$)
  while $W \neq \emptyset$ **do**
    for $p_i \in W$ **do**
      for $pc_j \in oneRingNeighbourOf(p_i)$ **do**
        if $pc_j \in W$ **OR** $pc_j.geoDist \neq \infty$ **then**
          continue
        else
          $parentNode \leftarrow$ GETPARENTNODE(W,pc_j)
          $d_{est} \leftarrow = distanceBetween(parentNode, pc_j) + parentNode.geoDist$
          if $d_{est} < rmax$ **then**
            Perform Equation.1.11 on $initialPath$ results geodesic path from source to pc_j
            Perform Algorithm.1 to project floating path $pc_j.geoPath$ onto S
            Update the neighbourhoods for the samples point on $pc_j.geoPath$ separately
            $W' \leftarrow pc_j$
          else
            Update d_{maxEst} if $d_{est} > d_{maxEst}$
            $W_{next} \leftarrow pc_j$
          end if
        end if
      end for
    end for
    $W \leftarrow W'$
    $W' \leftarrow \emptyset$
    if $W = \emptyset$ **then**
      $W \leftarrow W_{next}$
      $W_{next} \leftarrow \emptyset$
      $r_{max} \leftarrow d_{maxEst}$
    end if
  end while
end procedure

When solving Equation.1.11, the updated points on geodesic path always move away from the vertices on initial path. Therefore, in the process of Algorithm.2, in order to calculate the projections of sample point p on the floating geodesic path to mesh, exam every single faces on mesh for the projection is a very time consuming process. In order to maintain the efficiency of our algorithm, it is important to keep track on the corresponding vertices on the mesh where p are updated from. For every unvisited vertex, always the straightest path is selected as the parent and the updated sample point on geodesic path never move out of their original one-ring neighbourhoods. However, with the "wavefront" propagates, this assumption does not stands. Thus one-ring neighbourhood of every sample point need to be redetermined when a new point is added into the current geodesic. This is achieved by following procedure. Firstly, start from the 1-ring neighbours of a sample point q, the nearest vertex q_i is selected. Then within one-ring neighbours of q_i, the nearest vertex q_j to q is selected. This updating process is able to ensure the projection of updated sample point p on floating geodesic path always falls into the face that adjacent to the point that corresponds to p in its initial path.

Moreover, during the projection, the osculating plane is formed by three successive sample points on a geodesic. This osculating plane is used to cut the related faces consisting of the one-ring neighbours of these 3 sample points. The intersection of these two planes is the projection of the geodesic on the mesh. Note that with a manifold mesh, an edge can only be connected with one or two faces, therefore, with a face that has been determined to intersect with the osculating plane, its adjacent can be easily determined. As a result, to calculate the intersection points of the cutting plane with all the related edges costs a constant time.

A challenging issue of using Algorithm.2 to calculate geodesic is to solve the large sparse linear system of Equation.1.16. Moreover, followed by the propagation of "wavefront" the number of the point in the initial path

that need to be updated by Equation.1.11 is increasing. Therefore, the size of the matrix in Equation.1.16 is also increasing. This step is the most time consuming procedure in the algorithm presented here.

In order to improve the efficiency of this algorithm, an approximate algorithm is derived from Algorithm.2. In this algorithm, a small-size window is introduced to Equation.1.11 to avoid solving large sparse matrices. When an initial estimation of a geodesic path is obtained, as shown in Figure.1.9, a fixed-size window that covers the last w sample points is employed. Then Equation.1.11 is applied to w to calculate the local geodesic patch. Let $\widetilde{p}'_0 p'_{n-1}$ denotes the geodesic path from source to the parent of p' parent. Thus, only the section $\widetilde{p}'_{n-w} p'_n$ is updated to form the approximated geodesic path. This algorithm is summarized in Algorithm.3.

In Algorithm.3, because the window size w is a constant number, therefore, the matrix size in the linear system of Equation.1.11 does not change throughout the propagation of "wavefront", thus the time that solving the Equation.1.11 is also a constant. Therefore, by introducing the concept of window to each geodesic update, Algorithm.3 is able to achieve linear time complexity. Figure.1.10 demonstrate the geodesics result on four characters performed by Algorithm.3.

1.4 Geodesic on Point Cloud

Both Algorithm.2 and Algorithm.3 can be further extended for calculating the geodesics on point clouds easily.

Followed by the fast development of 3D shape acquisition device, 3D body scanner is becoming more and more popular in today's modelling applications. The most common shape representation that outputs by 3D shape acquisition device are point cloud. Therefore, it is important to be able to

Algorithm 3 Approximate Geodesics Algorithm

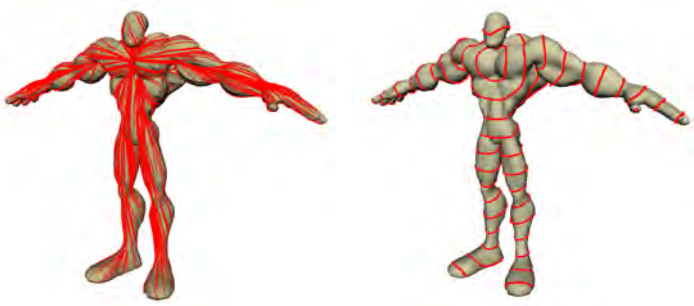
procedure INITIALIZATION(A mesh S, and a source p_s)
  for $p_i \in S$ **do**
    $p_i.geoDist \leftarrow \infty$
    $p_i.geoPath \leftarrow \emptyset$
  end for
end procedure

procedure CALCULATE ONE RING NEIGHBOUR OF p_s(W , r_{max})
  for $p_i \in W$ **do**
    $d_{est} \leftarrow Dist(p_i, p_s)$
    $p_i.geoDist \leftarrow currentDist$
    $p_i.geoPath \leftarrow p_s + p_i$
    if $currentDist > r_{max}$ **then**
      $r_{max} \leftarrow d_{est}$
    end if
  end for
end procedure

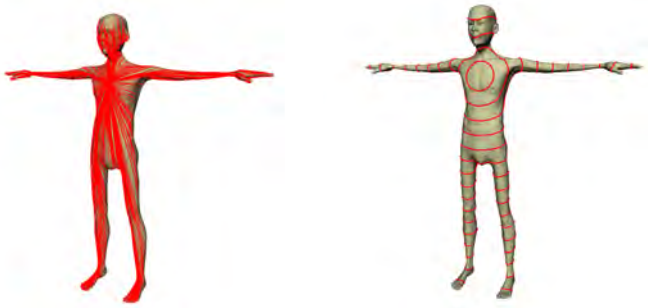
function GETPARENTNODE(W,$v_{unvisited}$)
  for $v' \in oneRingNeighbourOf(v_{unvisited})$ **do**
    $\theta = 0$
    if $v' \in W$ **then**
      $\vec{v_1} \leftarrow v_{unvisited} - v'$
      $\vec{v_2} \leftarrow getTangent(v_{unvisited})$ ▷ Return the tangent vector of geodesic path from source to v' at point v'
      if $\theta < angleBetween(\vec{v_1}, \vec{v_2})$ **then**
        $\theta \leftarrow angleBetween(\vec{v_1}, \vec{v_2})$
        $parentNode \leftarrow v'$
      end if
    end if
  end for
  return $parentNode$
end function

Algorithm 3 Approximate Geodesics Algorithm (continued)

procedure CONTINUES DIJKSTRA PROPAGATION(Calculate geodesic for every $p_i \in S$)
  while $W \neq \emptyset$ **do**
    for $p_i \in W$ **do**
      for $pc_i \in oneRingNeighbourOf(p_i)$ **do**
        if $pc_j \in W$ **OR** $pc_j.geoDist \neq \infty$ **then**
          continue
        else
          $parentNode \leftarrow$ GETPARENTNODE(W,pc_j)
          $d_{est} \leftarrow = distanceBetween(parentNode, pc_j) + parentNode.geoDist$
          if $d_{est} < rmax$ **then**
            Perform Equation.1.11 on the $\widetilde{p_{n-w}p_n}$ part of $initialPath$ results geodesic path from p_{n-w} to pc_j
            Combine $\widetilde{p}_0 p_{n-w-1}$ with $\widetilde{p}_{n-w} pc_j$ to form the approximate geodesic from source to pc_j
            Perform Algorithm.1 to project floating path $pc_j.geoPath$ onto S
            Update the neighbourhoods for the samples point on $pc_j.geoPath$ from $n - w^{th}$ to n^{th} point separately
            $W' \leftarrow pc_j$
          else
            Update d_{maxEst} if $d_{est} > d_{maxEst}$
            $W_{next} \leftarrow parentNode$
          end if
        end if
      end for
    end for
    $W \leftarrow W'$
    $W' \leftarrow \emptyset$
    if $W = \emptyset$ **then**
      $W \leftarrow W_{next}$
      $W_{next}\emptyset$
      $r_{max} \leftarrow d_{maxEst}$
    end if
  end while
end procedure



(a) *Geodesics on Character A* **(b)** *Isolines on Character A*



(c) *Geodesics on Character B* **(d)** *Isolines on Character B*

(e) *Geodesics on Character C* **(f)** *Isolines on Character C*

(g) *Geodesics on Character D* **(h)** *Isolines on Character D*

Figure 1.10: *Geodesics on character*

directly extract measurements data from point cloth model.

Despite the form of geometrical representation of the data, to calculate a geodesic, a source point must be specified in advance. The challenges of handling point clouds data is that, "Continue Dijkstra" propagation expand the "wavefront" from inward towards outward based on the connectivity between points. On polyhedral surface, this connectivity is represented by edges that connect vertices. However, on point clouds data, all the points are scattered and unordered. Therefore the "Continue Dijkstra" propagation cannot be performed on point clouds data directly.

In order to facilitate the connectivity of the points, a regular grid that covers the point cloud data set is employed. Within each cell that belongs to the grid, the mean point of the points that contained by this cell is regarded at the destination for all the points within this cell. This regular grid distributes destinations evenly over the point cloud data. Here, the concept of destination represents a location in a cell that from source point, all the geodesic that connected to the points in this cell are end at. Therefore, one destination represented all the points contained by a cell, the number of destinations n is much smaller than the actual number of scattered points in point cloud data. Moreover, because all the cell in the grid are identical in term of size and shape, it is very easy to index a cell and its neighbour by using Equation.1.19.

$$index = \frac{(c_x - min_x)}{\Delta c} + x_{seg} \times \frac{c_y - min_y}{\Delta c} + x_{seg} \times y_{seg} \times \frac{c_z - min_z}{\Delta c} \quad (1.19)$$

where c_x,c_y,c_z are the 3D coordinate of the centre point c of cell C, min_y,min_y,min_z are the 3D coordinate of the minimum point of the bounding box of the point cloud data. δc is the interval of regular grid. Note that, to build a whole grid requires an additional $O(n^3)$ space. However, because each cell can be indexed directly by the space it covers, therefore no search-

ing method is required to determine the 1-ring neighbour of a given cell. In the implementation of this algorithm, to optimize the memory consumption, an array C_{im} with $O(n^3)$ space cost only stores the index of non-empty cell, the actual data of the non-empty cell are stored into an dense array C with $O(n)$ space cost. Because the number of non-empty cells are much less than the number of the cell in a whole regular grid. The actual memory cost is close to $O(n)$. This process is summarized below.

Algorithm 4 Build Regular Grid for Point Cloud

procedure BUILD REGULAR GRID FOR POINT CLOUD(A point cloud data set S, a source p_s within the cell c_s,the intervals $\triangle x$ of a grid, and number of neighbours, d)
    Build up a searching tree for ANN performing on S
    Select the minimum point p_{min} of the bounding box $BBox$ of S. c is the cell that contains the source point p_s.
    while $c \in BBox$ **do**
        Get the closest point $p \in S$ to the centre of c
        if $p \in c$ **then**
            Store c into regular grid G
            Use ANN fix radius search to get all the points $C \in S$ that for every point $p_c \in C$, their distance to centre of c smaller than $\triangle x$
            Compute mean of C as the destination of c
        end if
        Move c orthogonally by $\triangle x$
    end while
end procedure

After regular grid G has been built, the "Continue Dijkstra" is employed here to form the initial path for solving Equation.1.11. This process is illustrate below.

In Algorithm.5, in order to determine the neighbourhood for each sample point on a geodesic path, ANN searching method(Arya et al. 1998) is used on the scattered points. Assume that there are d nearest scattered points to some sample point on a geodesic. The normal vector of the tangent plane is calculated by performing PCA(Jolliffe 2002) to the d scattered points that within a fixed radius. In addition,causality of the propagation can also benefit

Algorithm 5 Approximate Geodesics Algorithm on Point Cloud

procedure INITIALIZATION(A Point Cloud data set S, a Grid G and a source p_s and the cell c_s contains p_s)
    for $c_i \in G$ **do**
        $c_i.geoDist \leftarrow \infty$
        $C_i.geoPath \leftarrow \emptyset$
    end for
end procedure

procedure CALCULATE ONE RING NEIGHBOUR OF c_s(W , r_{max})
    for $c_i \in W$ **do**
        $d_{est} \leftarrow Distance(c_i.centre, c_s.centre)$
        $c_i.geoDist \leftarrow currentDist$
        $c_i.geoPath \leftarrow c_s + c_i$
        if $currentDist > r_{max}$ **then**
            $r_{max} \leftarrow d_{est}$
        end if
    end for
end procedure

function GETPARENTNODE(W,$c_{unvisited}$)
    for $c' \in oneRingNeighbourOf(c_{unvisited})$ **do**
        $\theta = 0$
        if $c' \in W$ **then**
            $\vec{v_1} \leftarrow c_{unvisited} - c'$
            $\vec{v_2} \leftarrow getTangent(c_{unvisited})$ ▷ Return the tangent vector of geodesic path from source to c' at point c'
            if $\theta < angleBetween(\vec{v_1}, \vec{v_2})$ **then**
                $\theta \leftarrow angleBetween(\vec{v_1}, \vec{v_2})$
                $parentNode \leftarrow c'$
            end if
        end if
    end for
    return $parentNode$
end function

Algorithm 5 Approximate Geodesics Algorithm on Point Cloud(continued)

procedure CONTINUES DIJKSTRA PROPAGATION(Calculate geodesic for every $c_i \in G$)
  while $W \neq \emptyset$ **do**
    for $c_i \in W$ **do**
      for $c_i' \in oneRingNeighbourOf(c_i)$ **do**
        if $c_i \in W$ **OR** $c_i'.geoDist \neq \infty$ **then**
          continue
        else
          $parentNode \leftarrow$ GETPARENTNODE(W,c_i')
          $d_{est} \leftarrow = distanceBetween(parentNode, c_i') + parentNode.geoDist$
          if $d_{est} < rmax$ **then**
            Perform Equation.1.11 on $initialPath$ results geodesic path from source to c_i'
            Update the neighbourhoods for the samples point on $c_i'.geoPath$ separately
            $W' \leftarrow c_i'$
          else
            Update d_{maxEst} if $d_{est} > d_{maxEst}$
            $W_{next} \leftarrow parentNode$
          end if
        end if
      end for
    end for
    $W \leftarrow W'$
    $W' \leftarrow \emptyset$
    if $W = \emptyset$ **then**
      $W \leftarrow W_{next}$
      $W_{next}\emptyset$
      $r_{max} \leftarrow d_{maxEst}$
    end if
  end while
end procedure

from the isotropicity of the regular grid. The propagation process is illustrated in Figure.1.11.

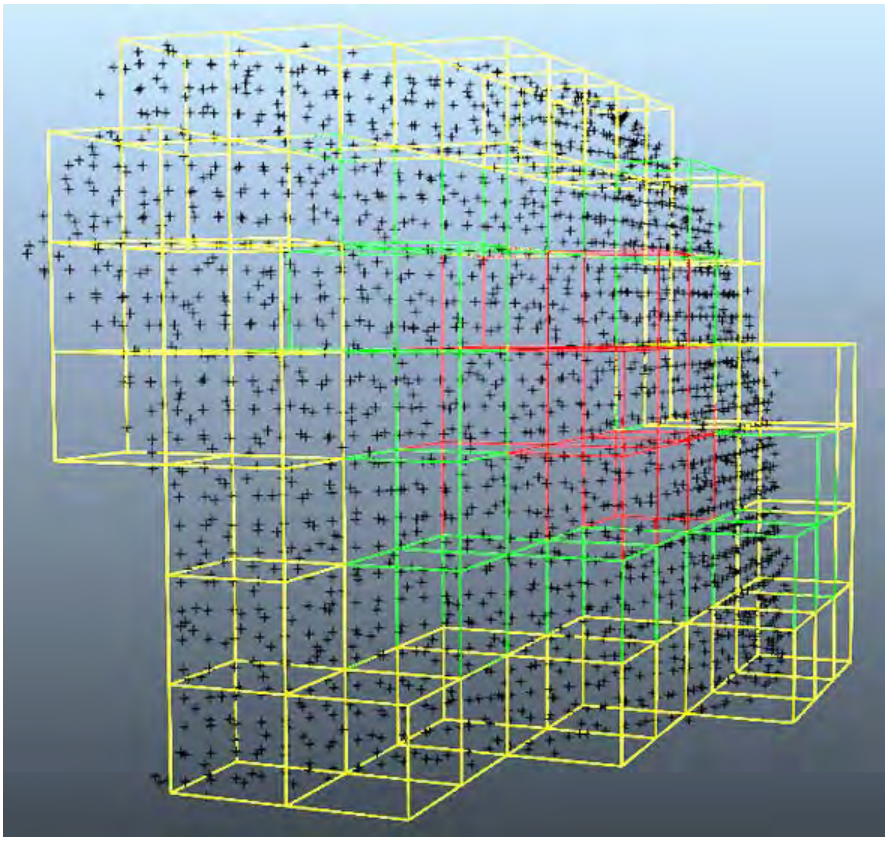
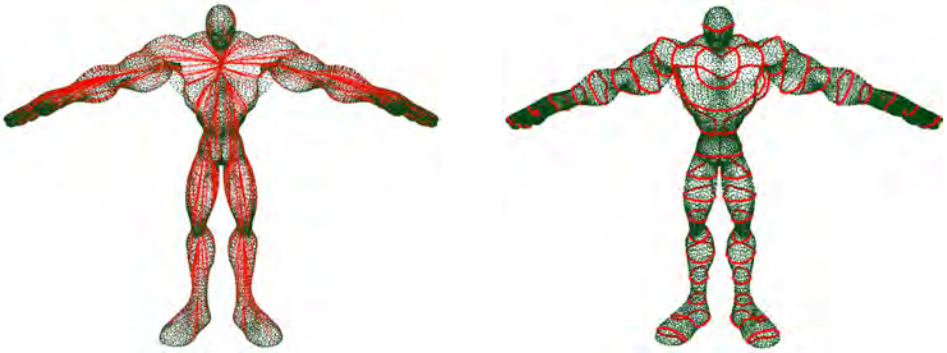


Figure 1.11: *Regular grid covering a point cloud. Propagation is from the red cells to the yellow ones. In practice, we store the nonempty cells in an array. No need to store the whole grid.*



(a) *Geodesics on point cloud data of Character A.*

(b) *Isolines on point cloud data of Character A.*

Figure 1.12: *Geodeiscs and Isolines on the point cloud model of Character A*

1.5 Performance Analysis

The performance of the proposed algorithms is analysed in two categorise, the accuracy of geodesic paths and the efficiency of the algorithm. Assume the given mesh contains n vertices, e edges and f faces.

1.5.1 Efficiency

MMP algorithm(Mitchell et al. 1987b; Surazhsky et al. 2005) defines the "Exact geodesic algorithm" as an algorithm that is able to produce geodesics that on a flattened planner, the geodesic path is a straight line. Because the geodesics results from Alg.2 is done by minimizing the geodesic curvature on the smooth surface that the polyhedral surface is approximated from, therefore when Equation.1.11 converged, the geodesic curvature reach to the minimum(Butcher 2008; Hairer 2010). However, this curve is on the smooth with minimum geodesic curvature, (Polthier & Schmies 2006) point out that, if the projection of this curve does not pass any hyperbolic vertex on the polyhe-

dron, the projected curve is a straight and shortest geodesic on the polyhedron. Therefore, in this thesis, Algorithm.2 is denoted as "accurate geodesic algorithm".

The time complexity of Algorithm.2 depends on the number of the sample points on a geodesic. Because the coefficient matrix A on the left hand side of Equation.1.16 is not a square matrix, it is necessary to left-multiply A^T on both side of the Equation.1.16 to form the square matrix for linear system solver. In the implementation of this algorithm, LU solver(Bunch & Hopcroft 1974) is used for solving the linear system. Let m denotes the order of the matrix in Equation.1.16, according to Bunch & Hopcroft (1974); Coppersmith & Winograd (1987), the LU solver has a time complexity if $O(m^{2.379})$. Therefore, for a single source to all destination geodesic computation, the upper bound of the time complexity can be estimated by $O(m^{2.379}n)$, where n denotes the number of vertices on mesh. Additionally, the computation of the projection of each 3 successive sample points on a geodesic path onto the mesh faces costs a constant time to denotes at h, the projection of the whole geodesic path costs $O(kh)$ where k denotes the number of sample point on a geodesic path. Consequently, projecting all the geodesics onto mesh costs $O(khm^{2.379}n)$.

For Algorithm.3, due to the fixed size window, the size of the matrices in Equation.1.16 is constant. Therefore, solving Equation.1.16 costs a constant time c. The total time complexity can be written as $O(cn)$. Because the window size is w, a new geodesic usually shares a segment with an existing geodesic. The projection of a geodesic path also shares a segment with an existing geodesic projection on a mesh. Projecting one geodesic path therefore costs $O(wh)$, and projecting all the geodesics costs $O(nwh)$, where w and h are constants. This conclusion is important, as it shows Algorithm.3 is a linear-time algorithm.

For Algorithm.5, it employs ANN search on the point cloud data set.

Let the number of the points in the point cloud data is N and the number of the non-empty cells in the regular grid is n. The ANN search for a given query point costs $O(\log n)$ time (Arya et al. 1998). Assume that there are at most d nearest neighbours for one query point by the ANN search. As a result, the ANN searching costs $O(d \log N)$ time on a point cloud. Thus, each geodesic thus costs $O(d \log Nn)$ time. Moreover, the time complexity for a single source geodesic computation is $O(d \log Nn^2)$.

Among the above presented algorithms, the most noteworthy achievement is the ability of computing geodesic paths in linear time in Alg.3 with a bounded error. This is especially significant for tracing a large number of geodesic paths and large models with over one million vertices, which is more and more common in various applications due to the technological advancements in high performance hardware and cheap storage. Larger models offer much better resolution and more detailed structural information, making many previously impossible operations possible today.

The aforementioned algorithms is implemented on a Intel Xeon 3.33GHz PC with 24GB RAM running Windows 7 (64-bit) operating system. For time comparison, several popular existing algorithms are used for comparison including the MMP(Surazhsky et al. 2005), improved CH algorithms(Xin & Wang 2009) and FMM(Kimmel & Sethian 1998). In order to compare the time consumption for acquire the geodesic, "backtracing" is added into MMP, CH/ICH and FMM. The algorithm used for "backtracin" is presented in Surazhsky et al. (2005).

The source codes of MMP and ICH algorithms are available on the Internet[1]. These algorithms are performed respectively on the Stanford Bunny's model with 8 different resolutions as shown in Table.1.1. The running times are plotted in Figure.1.13.

[1]MMP at http://code.google.com/p/geodesic/downloads/detail?name=geodesic_cpp_03_02_2008.zip&can=2&q=;
Improved CH at https://sites.google.com/site/xinshiqing/knowledge-share.

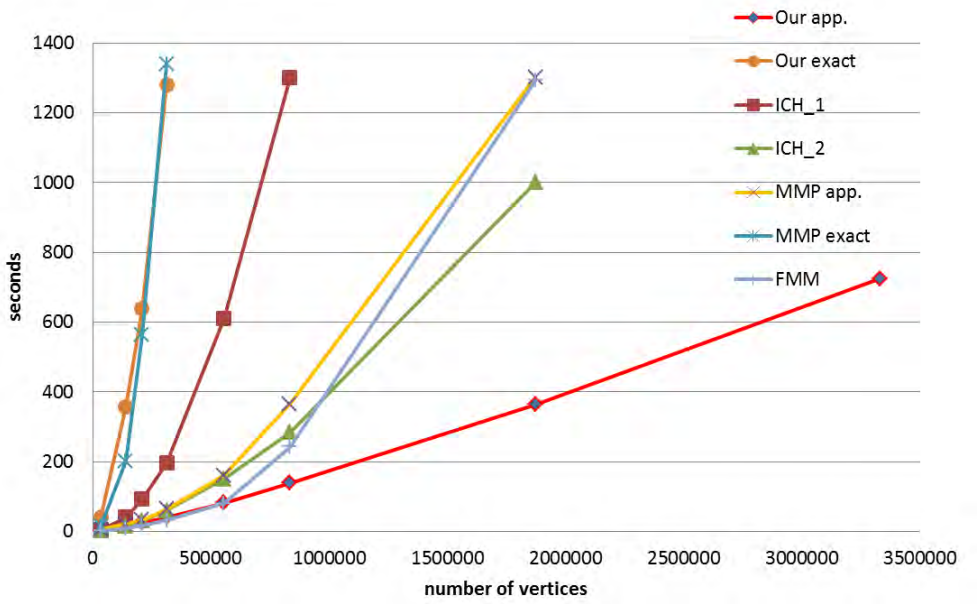


Figure 1.13: *The comparison of running times. The times of the ICH and MMP algorithms include the cost of "backtracing". The window size of our Algorithm.3, w=30.*

#Vertices	34834	139122	208573	312861
#Faces	69451	277804	416706	625059
#Vertices	556051	833855	1875843	3334537
#Faces	1111216	1666824	3750354	6667296

Table 1.1: *The resolution of the bunny models used in the experiment in Figure.1.13*

For human model that used in this thesis for dressing, Algorithm.3 also shows great advantage in terms of efficiency.

Furthermore, Algorithm.3, MMP approximation and ICH_2) are also performed on many large models for further comparison. We believe that these experiments provide some insight into the linear time complexity of our 1st order approximation algorithm against the MMP and ICH algorithms. The detail of this experiment please see in Appendix.**??**.

1.5.2 Accuracy

The accuracy of Algorithm.3 depends on the following parameters, the edge length e, the window size w, and the number of samples on each geodesic m. The basic assumption held by Algorithm.3 is that when computing a new geodesic, there exists one of the previous geodesics that is accurate enough and close to the desired one. As a result, we may crop a small patch at the end of the geodesic estimation by a w-sized window and apply the geodesic curvature flow of Equation.1.11 to this patch instead of the whole geodesic.

Let C_g denotes a geodesic on polyhedron that does not pass through any vertex p of the polyhedron unless p is the source p_s or a destination p_d. It can be observed that the C goes across a set of faces. Unfolding this set of faces into a plane, the C should be the straight line linking the p_s and p_d. For a new vertex q, we may combine the edge $\overline{p_d q}$ with the C as the estimation of the geodesic $p_s q$. Without loss of generality, let a window cover this set of faces and the number of faces be w. The error estimation for any geodesic computed by Algorithm.3 is illustrated in Figure.1.14,

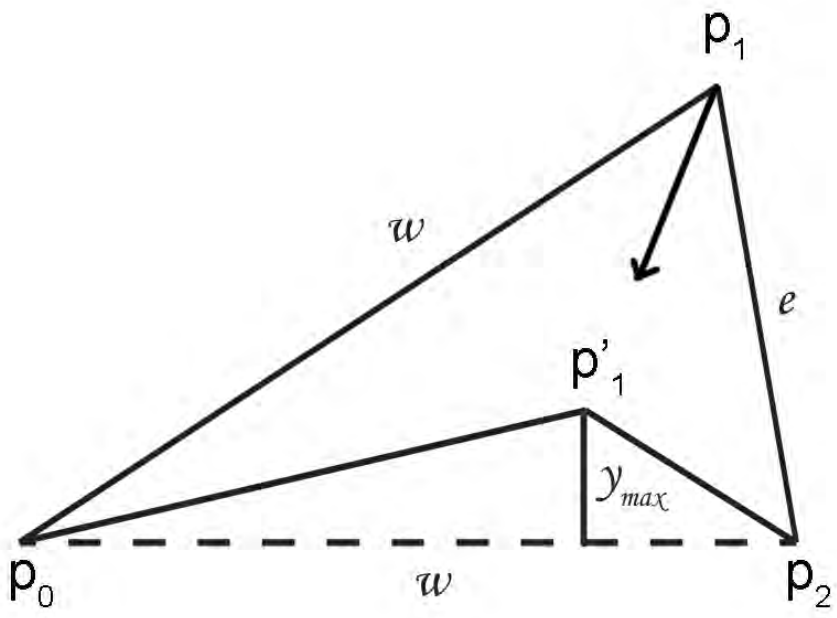


Figure 1.14: *The estimation and desired geodesics. The $\overline{p_0p_2}$ is the desired geodesic while $\overline{p_0p_1}$ plus $\overline{p_1p_2}$ being the estimation.*

Let p_0, p_1,p_2 are three point on a plane as shown in Figure.1.14. The initial path for solving Equation.1.11 is $\widetilde{p_0p_1p_2}$ where p_0 is source point and p_2 is destination point. Obliviously, the true geodesic from p_0 to p_2 is $\overline{p_0p_2}$. Let $\overline{p_0p_1}$ is an established geodesic path, p_2 is an unvisited vertex. the length of edge $\overline{p_0p_1}$ is always smaller or equal than the length of $\overline{p_0p_2}$ since the propagation only performed on outward direction. Therefore, in the worst situation, as in Figure.1.14, $\overline{p_0p_2}$ and $\overline{p_0p_1}$ are equal. p_1' is the updated point of Equation.1.11, Let the distance from p_1' to edge $\overline{p_0p_2}$ is y_{max}, $\|p_1p_2\| = e$, $\|p_0p_1\| = \|p_0p_2\| = w$, $\|p_0p_1'\| + \|p_1'p_2\| = L$. Therefore, the based on the Heron's formula, the area of $\triangle p_0p_1'p_2$ can be written as,

$$T_{\triangle p_0p_1'p_2} = \frac{\sqrt{(L^2 - w^2)w^2}}{4} = \frac{wy_{max}}{2} \tag{1.20}$$

therefore,

$$L = \sqrt{4y_{max}^2 + w^2} \tag{1.21}$$

Therefore, let err denote the difference between true geodesic length and the solution of Equation.1.11, where err can be written as,

$$err = L - \|p_0 p_2\| = \sqrt{4y_{max}^2 + w^2} - w \tag{1.22}$$

The average error over the w-sized window can be estimated as,

$$err_{ave} = \frac{\sqrt{4y_{max}^2 + w^2} - w}{w} \tag{1.23}$$

For a geodesic with m sample points, the upper bound of the error is therefore estimated as,

$$err = m\left(\sqrt{1 + 4(\frac{y_{max}}{we})^2} - 1\right) < 2m\frac{y_{max}}{w} \sim O(\varepsilon m) \tag{1.24}$$

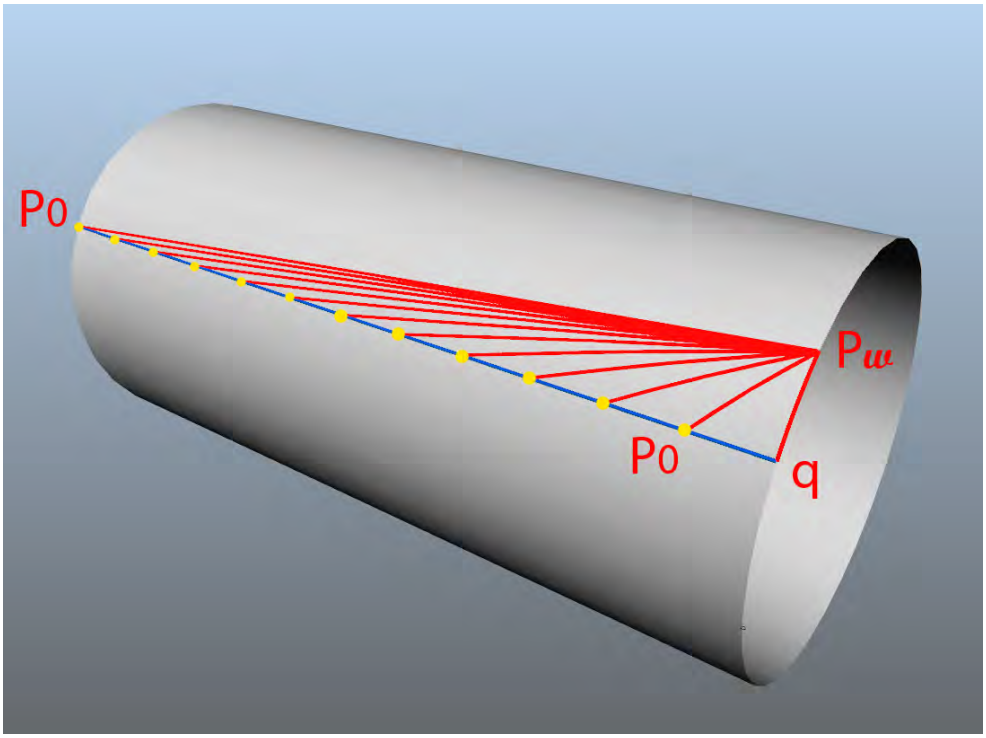
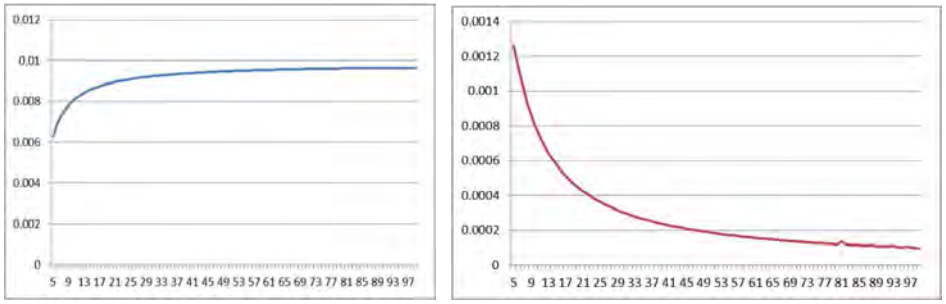


Figure 1.15: *Error estimation of Algorithm.3 with different window size*



(a) *The convergence of y_{max} with varying window size*

(b) *The convergence of ϵ with varying window size*

Figure 1.16: *Error of Algorithm.3*

where $\varepsilon = \frac{y_{max}}{w} \ll 1$ and y_{max} denotes the maximum offset distance of sample points to the ground truth. To have an insight into the y_{max}, we performed the Algorithm.3 on a cylinder surface with the edge length e = 1 and the window size w varying from 5 to 100 in Figure.1.15. The numerical results are shown in Figure.1.16a. It can be noted that when w = 5, the y_{max} tends to zero. But, the larger the window, the bigger the y_{max}. This can be explained that on the plane shown in Figure.1.14, the curvature of the geodesic estimation $\widetilde{p_0p_1p_2}$ is becoming small when increasing the window size w. The system of Equation.1.11 is appropriate to deal with high curvature areas. On the other hand, the window size w is usually expected to be as large as possible. In Figure.1.15, the p_0 is viewed as the start point of the window. If it is not the source, there must exist a geodesic from the source to the p_0. It is ideal that the desired geodesic from the source to the p_2 passes through the geodesic from the source to the p_0. Thus, it is natural to enlarge the window size as much as possible. Figure.1.16a and Figure.1.16b shows that both the y_{max} and the ratio ϵ convergence to small values. This means that the large w does not decrease the error significantly. The choice of the window size w should take into account the time of solving Equation.1.11 rather than the computational error.

Characters		A #V:21250 #F:42946	B #V:15368 #F:30644	C #V:15266 #F:30440	D #V:14876 #F:29704
Ours Alg.2 (offset)	ave abs	0.00309	0.00098	0.00786	0.005218
	ave rel	0.5054%	0.1911%	1.3454%	0.4852%
Ours Alg.2 (proj)	ave abs	0.00401	0.00759	0.00716	0.01095
	ave rel	1.2298%	1.0439%	1.3053%	1.4867%
MMP app.	ave abs	0.00416	0.00991	0.00752	0.00917
	ave rel	1.4183%	1.3665%	1.3283%	1.0886%
ICH_2	ave abs	0.00344	0.003751	0.00795	0.01198
	ave rel	0.7867%	0.5886%	1.3311%	1.5128%

Table 1.2: *The average absolute and relative errors of Algorithm.3, MMP app. and ICH_2. The window size of our Algorithm.3,* $w = 30$.

In order to evaluate the accuracy of Algorithm.3 the MMP approximation and ICH_2 algorithms are also performed on four characters.

MMP exact algorithm is performed on these four characters and its solution are considered as the ground truth for the experiment and then calculated the absolute errors by the differences between the exact geodesic distances and approximate ones and the relative errors by the ratios of the absolute errors over the exact distances.

Note that for Algorithm.3 two results are kept as one is the offset geodesic paths and the other is the projections of the offset geodesics onto the meshes. Table.1.2 gives the average absolute and relative errors of these 3 algorithms. It can be noted that the offset solution of Algorithm.3 obviously outperforms the others, including the ICH_2, even though ICH_2 is regarded as the exact algorithm. However, the projections of the offset geodesics to the meshes have no distinct difference compared to MMP approximation and ICH_2 algorithms.

Moreover, to further validate the error estimation for Algorithm.3, MMP exact algorithm is performed on the lowest resolution bunny model in Figure.1.1 and use the length of geodesics result from MMP exact algorithm as ground truth.

The ratio ε in Equation.1.24 is usually a very small number. Figure.1.17 shows the histogram of the ratio ε over the bunny model, where the horizontal axis indicate the value of ε and vertical axis indicate the number of point in a geodesic.

Figure.1.18 further shows the distribution of the real absolute errors and the estimated ones in the bunny test. It can be noted that the error estimation of Equation.1.11 can accurately reflect the upper error bound of Algorithm.3

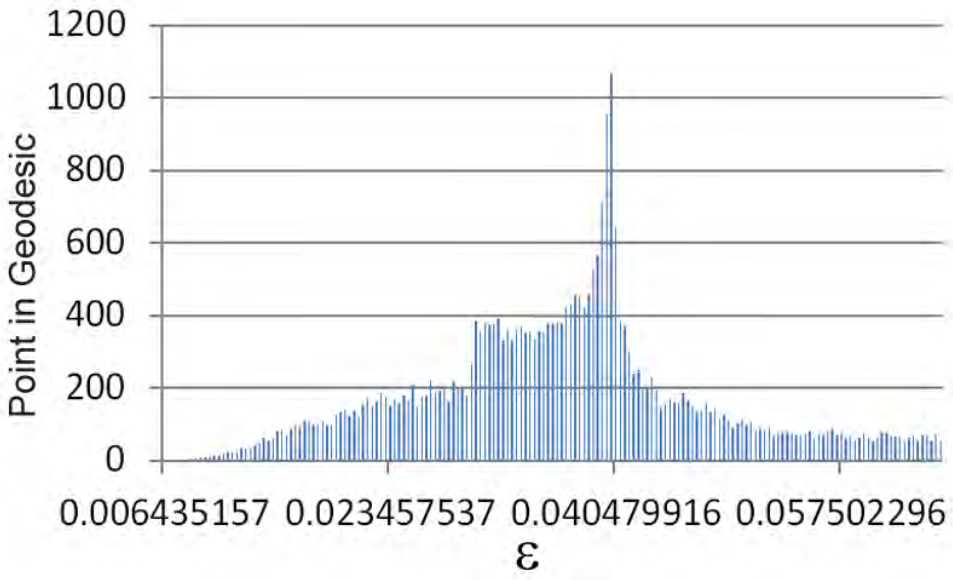


Figure 1.17: *Histogram of the ratio ε*

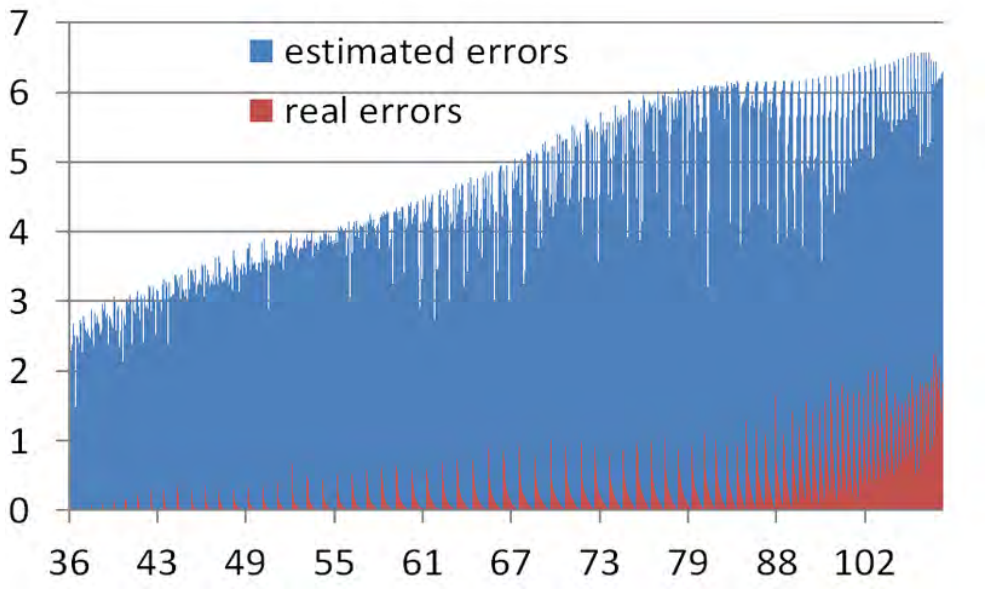


Figure 1.18: *The distribution of the estimated errors and real errors. The window size of our Alg.3, w = 30. The x-axis indicates the number of the faces covered by a geodesic path.*

Thirdly, the Algorithm.5 is further preformed on three point cloud models, which are, Stanford bunny, Buddha and sculpture, respectively for geodesic computation. The experimental results are shown in Table.1.3. Destinations are the means of the patches that are cropped by cells separately here. Obviously, they are fewer than the scattered points. Although the ANN searching leads to a $O(n \log n)$ searching time and requires extra space to store the searching tree in theory, it does not result in large time and space complexities in practice. Figure.1.19 shows shortest paths and isolines on the point clouds in Table.1.3.

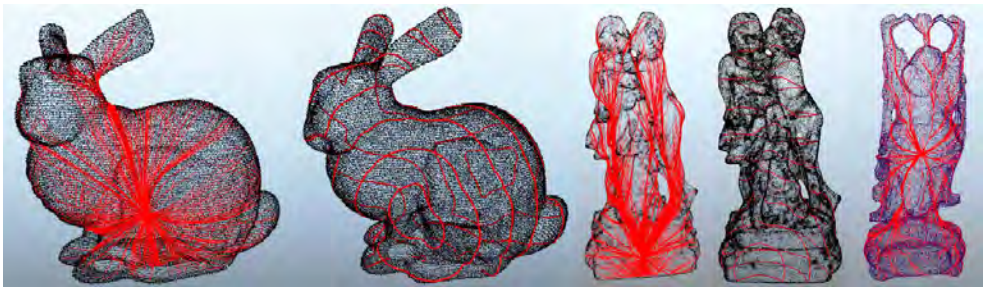
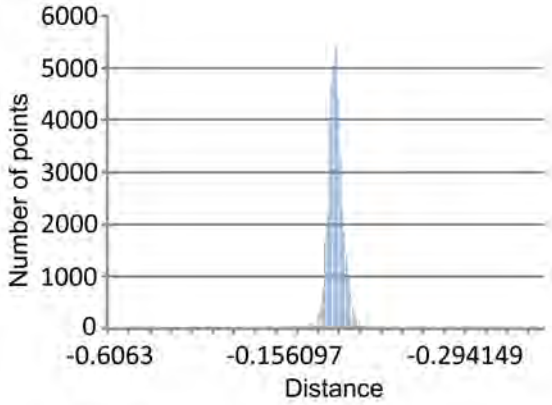


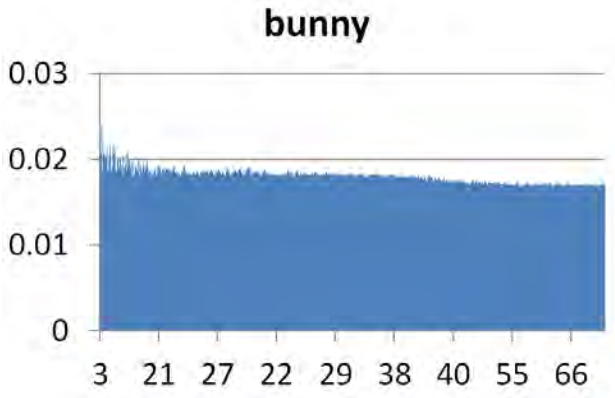
Figure 1.19: *Models used for tests in Table.1.3*

	Character A	Bunny	Sculpture	Buddha
#Points	134175	69668	406224	841151
#Destinations	7904	3748	10878	30177
Time(sec)	53.677	32.924	73.019	226.772
Memory(Mb)*	17.381	7.668	24.336	101.221

Table 1.3: *Performance of Algorithm.5. *Memory indicates the peak memory cost used in performing Algorithm.5, excluding the storage of the model itself.*



(a) *The histogram of the offset distances of the sample points to the implicit surface of the point cloud bunny*



(b) *The distribution of the average relative errors of our Algorithm.5 on bunny. The x-axis indicates the number of the sample point in a geodesic path*

Figure 1.20: *Accuracy of Algorithm.5.*

The accuracy of the resulting geodesic paths can be evaluated by the

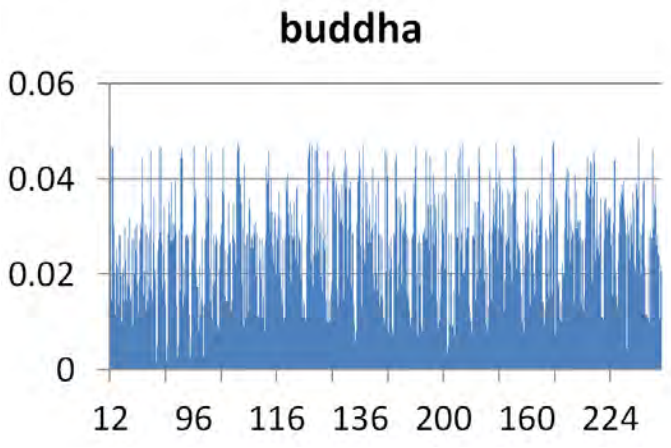


Figure 1.21: *The distribution of the average relative errors of Algorithm.5 on buddha. The x-axis indicates the number of the sample point in a geodesic path*

offset distances of the sample points on geodesics to the implicit surface of a point cloud. Figure.1.20b shows the histogram of the offset distances of the sample points to the reconstructed polyhedral surface of the bunny model.

According to the Figure.1.20b, Algorithm.5 can guarantee the resulting geodesic paths close to the implicit surfaces of point clouds. Moreover, MMP exact algorithm is preformed respectively on the reconstructed Bunny and Buddha meshes and the results geodesics are regarded as the ground truth for this experiment. Figure.1.20a and Figure.1.20b show the distribution of the average relative errors of the Algorithm.5 on the two point clouds. It can be noted that the error on Buddha is obviously higher than that on Bunny. This is due to the fact that the model of Buddha' has more details than the model of Bunny. Further increasing the number of cells in the grid can decrease error.

1.6 Geodesic in body measurement

During the process of measuring, two types of measurements is used for describing the body dimension of the character, circumference and length.

	Name	Measuring Method
Circumference	Bust Girth	The horizontal girth around the bust point
	Chest Girth	The horizontal girth passed over the shoulder blades, under the axillae, and across the chest
	Waist Girth	The horizontal girth go through front waist point and back waist point
	Middle Hip Girth	The horizontal girth around the abdomen girth point
	Hip Girth	The horizontal girth around the hip point
	Neck girth	The horizontal girth go through the neck shoulder point
	Cuff Girth	The girth around the wrist point
Length	Height	The distance from the back neck point to the heel point
	Back Length	The distance from the back neck point to the back waist point
	Sleeve Length	The distance from the neck point to the wrist point
	Arm Hole Length	The distance from the front axilla point go through the shoulder point to the back axilla point
	Sleeve Top	The shortest distance from the shoulder point to the line which go through two axilla point on the flattened sleeve pattern
	Waist length	The distance between the waist line and the hip line
	Crotch Depth	The distance from the centre of the front waist line to through crotch point to the centre of the back waist line
	Inside-Leg Length	The distance from the crotch point to the inside ankle point

Table 1.4: *The definitions of the measurements and their associated datum points (Armstrong 2000; EN:13402 2001)*

Table.1.4 listed out the major measurements are taken from the body of the character. and illustrate the datum points for general measuring on a female figure.

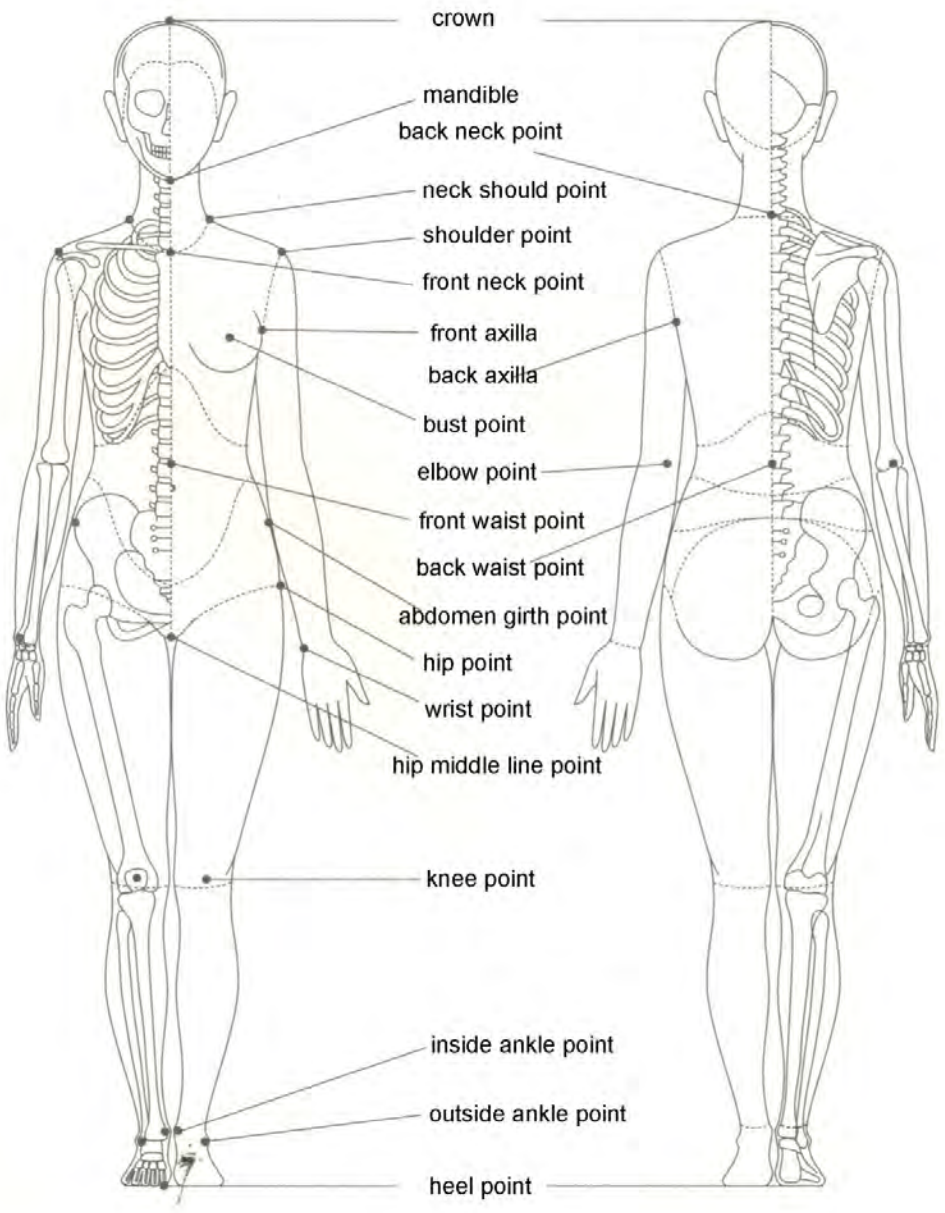


Figure 1.22: *Datum point on a female body(Xiong 2008)*

In this thesis, the circumference is measured by applying convex hull

algorithm(Graham 1972; De Berg et al. 2008) to the intersection of a cutting plane and the body part that are measured, then the arc length of the curve that connect all the intersect point is the circumference of the desired body part. Figure.1.23 demonstrate the bust girth and waist girth that are required for fitting a shirt on to the character.

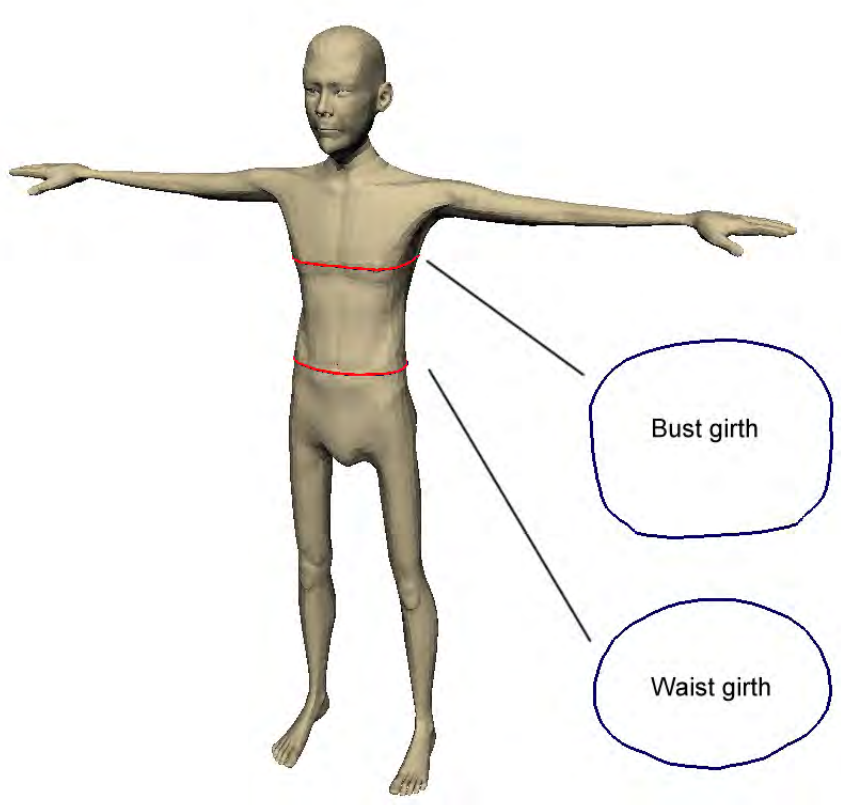


Figure 1.23: *Two circumferences measurement on character, two blue close curve indicate the convex hull of its corresponding measurements*

For length measurement, the geodesic algorithm presented in this chapter is applied to the body of the character. With each required measurement, two datum that associated with the measurement is used as the source point and the destination point for the geodesic algorithm respectively. Figure.1.24 demonstrate the length measurements that required for fitting a shirt on to the character.

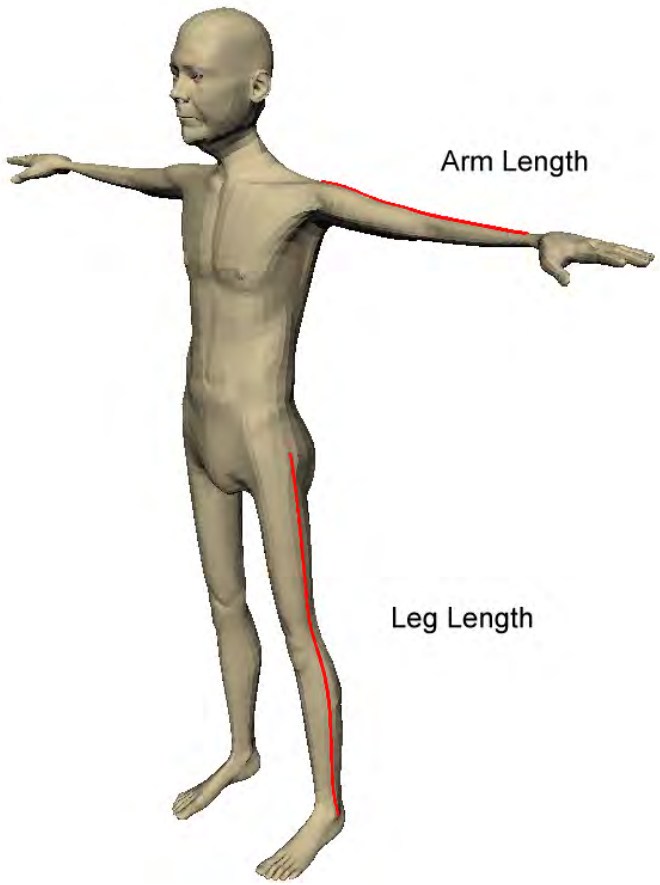


Figure 1.24: *Two length measurement on character, the "Arm Length" is the length of the geodesic path from "Shoulder Point" to "Wrist Point", the "Leg Length" is the length of the geodesic path from "Hip Point" to "Outside Ankle Point", for the datum points, see Figure.1.22.*

1.7 Conclusions

The standard posture used for modelling character differs from person to person. Traditional anthropomorphic data acquisition method requires character to stay in a standard posture in order to extract the correct measurement data. Therefore, when applying traditional anthropomorphic measuring method, different posture of the character will results different measurements. Hence

the cloth adjusted based on those measurement will be ill-fitted. On the contrary, geodesics are more close to the circumstance of tape measuring in real world, it also result less variation when posture of the character changes.

This chapter proposed three algorithm respectively for accurate and approximate geodesic computation on triangulated manifolds as well as approximate geodesic computation on point cloud data set. The most important contribution is that the proposed approximation algorithm can reach linear time complexity with an bounded error on triangulated manifolds. Numerical comparisons with existing algorithms (i.e. MMP, ICH_1 and ICH_2) have further demonstrated the advantages of our algorithms in terms of both speed and accuracy. By integrating Algorithm.3 into the measuring system, the time consumed on solve the geodesic path between multiple pair of source and destination has largely reduced, moreover, the accuracy of the solution is maintained. In the final chapter , more experiments will be presented to illustrate the advance of the algorithm presented in this chapter.

Bibliography

L. Aleksandrov, et al. (2005). 'Determining approximate shortest paths on weighted polyhedral surfaces'. *J. ACM* **52**(1):25–53.

H. J. Armstrong (2000). *Patternmaking: for fashion design*. Pearson Prentice Hall.

S. Arya, et al. (1998). 'An optimal algorithm for approximate nearest neighbor searching fixed dimensions'. *J. ACM* **45**(6):891–923.

A. Bartesaghi & G. Sapiro (2001). 'A system for the generation of curves on 3D brain images'. *Human Brain Mapping* **14**(1):1–15.

P. Bose, et al. (2011). 'A survey of geodesic paths on 3D surfaces'. *Computational Geometry* **44**(9):486 – 498.

J. R. Bunch & J. E. Hopcroft (1974). 'Triangular Factorization and Inversion by Fast Matrix Multiplication'. *Mathematics of Computation* **28**:231–231.

J. Butcher (2008). *Numerical Methods for Ordinary Differential Equations*. Wiley.

J. C. Butcher (1987). *The numerical analysis of ordinary differential equations: Runge-Kutta and general linear methods*. Wiley-Interscience, New York, NY, USA.

J. Chen & Y. Han (1990). 'Shortest paths on a polyhedron'. In *Proceedings of the sixth annual symposium on Computational geometry*, SCG '90, pp. 360–369, New York, NY, USA. ACM.

S. J. A. Chopp, David L. (1993). 'Flow under curvature: Singularity formation, minimal surfaces, and geodesics.'. *Experimental Mathematics* **2**(4):235–255.

Clairaut (1731). *Recherches sur les courbes a double courbure [microform]*. Nyon, Didot, Quillau Paris.

D. Coppersmith & S. Winograd (1987). 'Matrix multiplication via arithmetic progressions'. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pp. 1–6, New York, NY, USA. ACM.

T. Cormen, et al. (2001). *Introduction To Algorithms*. MIT Press.

J. daniel Boissonnat & F. Cazals (2001). 'Coarse-to-fine surface simplification with geometric guarantees'. *Computer Graphics Forum* **20**:490–499.

M. de Berg, et al. (2008). *Computational Geometry: Algorithms and Applications*. Springer.

E. W. Dijkstra (1959). 'A note on two problems in connexion with graphs'. *Numerische Mathematik* **1**:269–271.

EN:13402 (2001). *Size designation of clothes*.

R. Enns & G. McGuire (2000). *Nonlinear Physics With Maple for Scientists and Engineers*. Springer.

R. L. Graham (1972). 'An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set'. *Inf. Process. Lett.* **1**(4):132–133.

G. Hairer (2010). *Solving Ordinary Differential Equations II*. Springer Berlin Heidelberg.

J. Hershberger & S. Suri (1999). 'An Optimal Algorithm for Euclidean Shortest Paths in the Plane'. *SIAM J. Comput.* **28**(6):2215–2256.

M. Hofer & H. Pottmann (2004). 'Energy-minimizing splines in manifolds'. *ACM Trans. Graph.* **23**(3):284–293.

W. Hundsdorfer & J. Verwer (2003). *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*. Springer Series in Computational Mathematics. Springer.

B. Jiang (1998). *The Least-Squares Finite Element Method: Theory and Applications in Computational Fluid Dynamics and Electromagnetics*. Lecture Notes in Mathematics. Springer.

I. Jolliffe (2002). *Principal Component Analysis*. Springer Series in Statistics. Springer.

T. Kanai & H. Suzuki (2000). 'Approximate Shortest Path on Polyhedral Surface Based on Selective Refinement of the Discrete Graph and Its Applications'. In *Proceedings of the Geometric Modeling and Processing 2000*, GMP '00, pp. 241–, Washington, DC, USA. IEEE Computer Society.

S. Katz & A. Tal (2003). 'Hierarchical mesh decomposition using fuzzy clustering and cuts'. *ACM Trans. Graph.* **22**(3):954–961.

R. Kimmel & J. A. Sethian (1998). 'Computing geodesic paths on manifolds'. *Proceedings of the National Academy of Sciences* **95**(15):8431–8435.

E. Kreyszig (1991). *Differential Geometry*. Differential Geometry. Dover Publications.

F. Mémoli & G. Sapiro (2001). 'Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces: 730'. *J. Comput. Phys.* **173**(2):764–.

J. S. B. Mitchell, et al. (1987a). 'The discrete geodesic problem'. *SIAM J. Comput.* **16**(4):647–668.

J. S. B. Mitchell, et al. (1987b). 'The discrete geodesic problem'. *SIAM J. Comput.* **16**(4):647–668.

G. Peyré & L. D. Cohen (2006). 'Geodesic Remeshing Using Front Propagation'. *Int. J. Comput. Vision* **69**(1):145–156.

K. Polthier & M. Schmies (2006). 'Straightest geodesics on polyhedral surfaces'. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, pp. 30–38, New York, NY, USA. ACM.

H. Pottmann, et al. (2010). 'Geodesic patterns'. *ACM Trans. Graph.* **29**:43:1–43:10.

M. R. Ruggeri, et al. (2006). 'Approximating geodesics on point set surfaces'. In *Proceedings of the 3rd Eurographics / IEEE VGTC conference on Point-Based Graphics*, SPBG'06, pp. 85–94, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

A. Salden, et al. (1999). 'Linearised Euclidean Shortening Flow of Curve Geometry'. *International Journal of Computer Vision* **34**(1):29–67.

Y. Schreiber & M. Sharir (2006). 'An optimal-time algorithm for shortest paths on a convex polytope in three dimensions'. In *Proceedings of the twenty-second annual symposium on Computational geometry*, SCG '06, pp. 30–39, New York, NY, USA. ACM.

J.-A. Serret (1851). 'Sur quelques formules relatives la thorie des courbes double courbure.'. *Journal de Mathmatiques Pures et Appliques* pp. 193–207.

M. Sharir(& A. Schorr (1984). 'On shortest paths in polyhedral spaces'. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, STOC '84, pp. 144–153, New York, NY, USA. ACM.

A. Spira & R. Kimmel (2002). 'Geodesic Curvature Flow on Parametric Surfaces'. In *In Curve and Surface Design: Saint-Malo 2002*, pp. 365–373.

V. Surazhsky, et al. (2005). 'Fast exact and approximate geodesics on meshes'. *ACM Trans. Graph.* **24**(3):553–560.

H. Thielhelm, et al. (2012). 'Connecting geodesics on smooth surfaces'. *The Visual Computer* **28**(6-8):529–539.

L. Trefethen & D. Bau (1997). *Numerical Linear Algebra*. Miscellaneous Bks. Society for Industrial and Applied Mathematics.

B. A. Wandell, et al. (2000). 'Visualization and Measurement of the Cortical Surface'. *J. Cognitive Neuroscience* **12**(5):739–752.

C. Wu & X. Tai (2010). 'A Level Set Formulation of Geodesic Curvature Flow on Simplicial Surfaces'. *Visualization and Computer Graphics, IEEE Transactions on* **16**(4):647–662.

S.-Q. Xin & G.-J. Wang (2009). 'Improving Chen and Han's algorithm on the discrete geodesic problem'. *ACM Trans. Graph.* **28**(4):104:1–104:8.

N. Xiong (2008). *World Classical fashion Design and Pattern*. Jiangxi Art Press.