Chapter 1

Character Measurements Extraction

In cloth production, customer needs to be measured first in order to determine the size of cloth. In anthropometric data acquisition, two types of measurements are associated with cloth making, length measurements and circumference(girth) measurements. Both measurements are obtained by calculating the length of tape ruler between two datum points on skin or surrounding a body part. Table 1.1 lists out several major measurements that are associated with cloth making.

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

In order to extract measurements listed in Table 1.1, datum points need to be defined on character body. Figure 1.1 illustrates the datum points for general measuring on a female figure based on the tailoring rule (Xiong 2008).

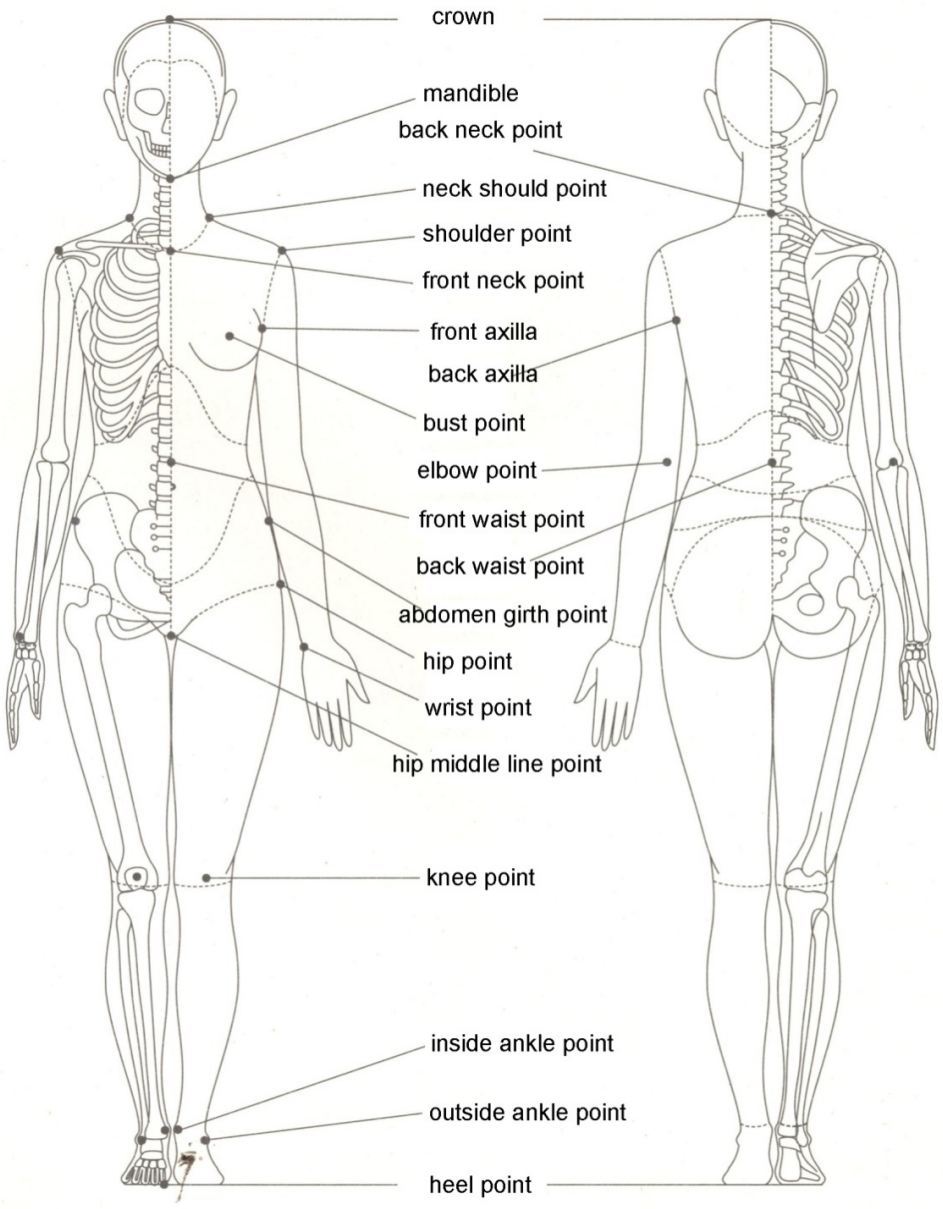


Figure 1.1: *Datum points on a female body(Xiong 2008)*

When extracting circumference from a subject, tape ruler is applied to the cross-section of a body part. After tension is applied to the ruler, it forms a convex hull of the cross-section of that body part. Therefore, in this thesis, the circumference is measured by applying convex hull algorithm(Graham 1972; De Berg et al. 2008) to the cross-section of a body part, the arc length of the convex hull is used as circumference of the body part. Figure 1.2 demonstrates two circumference measurements on a character.

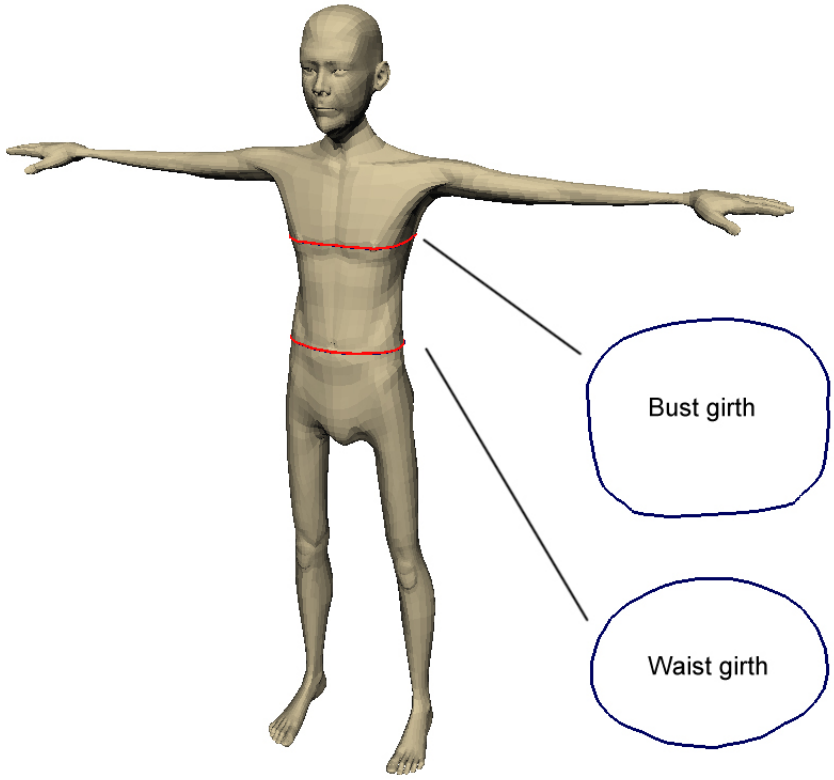


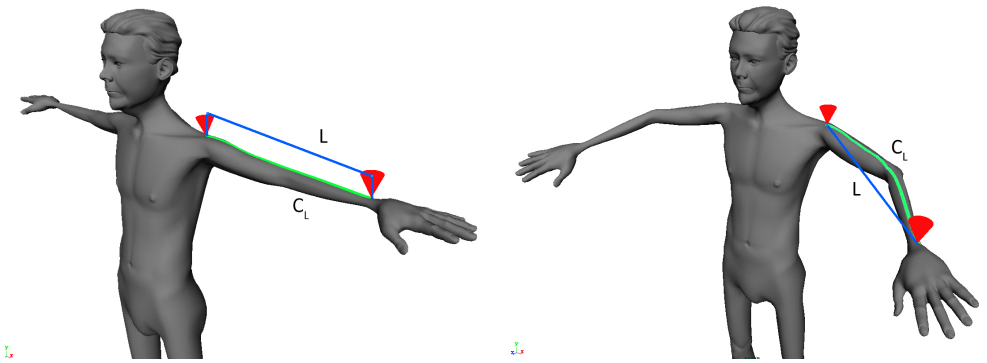
Figure 1.2: *Two circumference measurements on character, two blue closed curves indicate the convex hull of its corresponding measurements*

For extracting length measurements, subjects are usually required to stay in a predefined posture through out the measuring process. For example, when measuring arm length, subjects are required to stretch straight their arm in order to apply ruler from shoulder point to wrist point. The correctness of

the posture significantly affects the accuracy of measurements. When arm is bent, the distance between shoulder point and wrist point will be shortened. In computer animation, "T-Pose" is the standard posture for character modelling because the body and limbs are stretched straight so the space between different body parts is maximized. This provides many convenience for rigging or texturing. Unfortunately, the concept of "T-pose" are ambiguous because different studios have different definition of "T-Pose".

The changes of posture might need different measuring method to cope with, for example, 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 character height, for a character in a standing straight posture, the height can be obtained by calculating the Euclidean distance from top of head to the bottom of heel. However, when character bows 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 used for measuring are also differs.

In tailoring, when measuring customer for making a cloth, one end of a tape ruler is held to one of two datum points and tension is applied to the tape ruler so that it follows the profile of the body as closely as possible when it reaches the second datum point. At this point, due to the applied tension on tape ruler, the length of ruler reaches shortest following the body profile between these two datum points. Geodesic path is the shortest path between two points on a curved surface. Therefore, in this thesis, geodesic is used for measuring length of body parts of characters. Using geodesic to extract length measurement can also copes with different character postures. For example, Figure 1.3 demonstrates 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



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

Figure 1.3: *Measurements in different postures*

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". In Figure 1.3, 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. The discrete polyhedral surface, as the most common shape representation in computer graphic, has this connectivity. Therefore, in the past few decades, many algorithms have been developed to compute the geodesic on discrete polyhedral surfaces. However, current geodesic computation on a high resolution polygon model is still a very time consuming process. In many applications, such as surface flattening(Zigelman et al. 2002) and remeshing(Peyré & Cohen 2006), the efficiency of geodesic algorithm has become the bottle neck of its implementations.

This chapter presents a novel discrete geodesic computation scheme to improve the efficiency of geodesic computation on a discrete polyhedral surface. It consists of two algorithms for accurate geodesic path computation and high efficiency approximate geodesic computation. Among these algorithms, the approximate algorithm for polyhedral surface achieves linear time complexity whilst maintain a small bounded error.

1.1 Introduction

For different shape representation, geodesic has different definitions. Firstly, for parametric surface, 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).

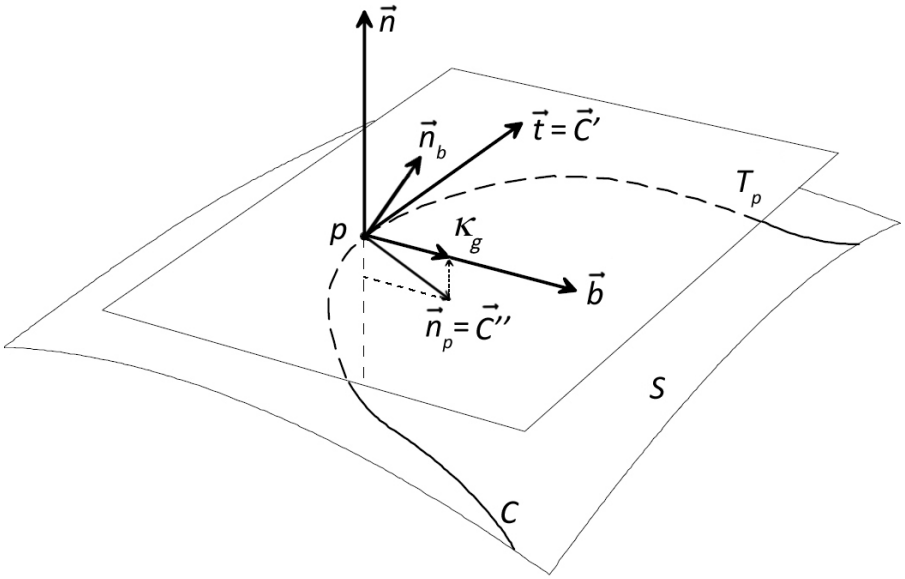


Figure 1.4: *Illustration of geodesic curvature*

Figure 1.4 demonstrates geodesic curvature at a point on a curve, where

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

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

1. *C is locally shortest curve(C has shortest arc length).*
2. *$\vec{C}''_s$ 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.

Definition 2 *(Polthier & Schmies 2006) Let S be a polyhedral surface and a 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 1.5 demonstrates three types of vertices on polyhedral surface.

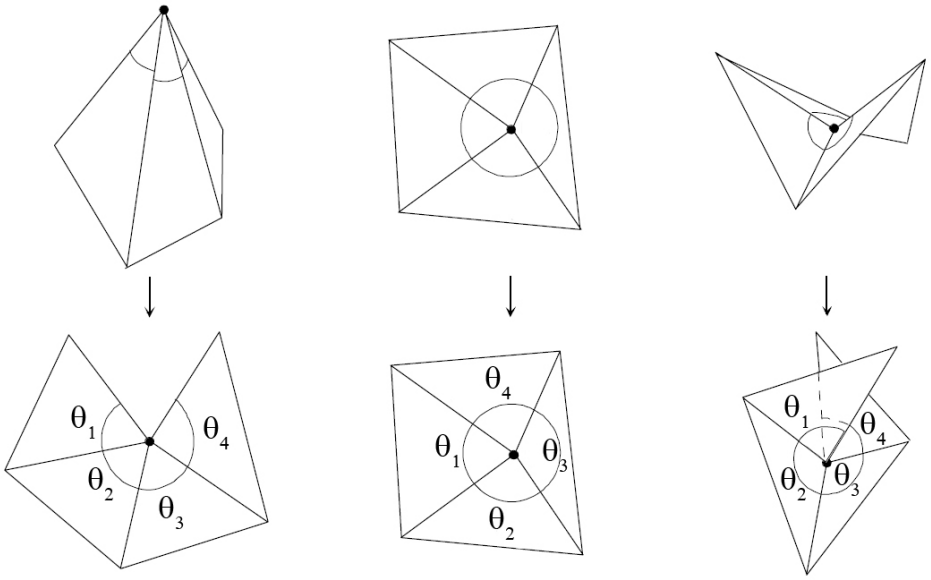


Figure 1.5: *Spherical Vertex(left), where* $2\pi - \Sigma\theta > 0$*; Euclidean Vertex(middle), where* $2\pi - \Sigma\theta = 0$*; Hyperbolic Vertex(right), where* $2\pi - \Sigma\theta < 0$*.*

Therefore a geodesic on 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 more than once in an edge sequence"(Mitchell et al. 1987a).*

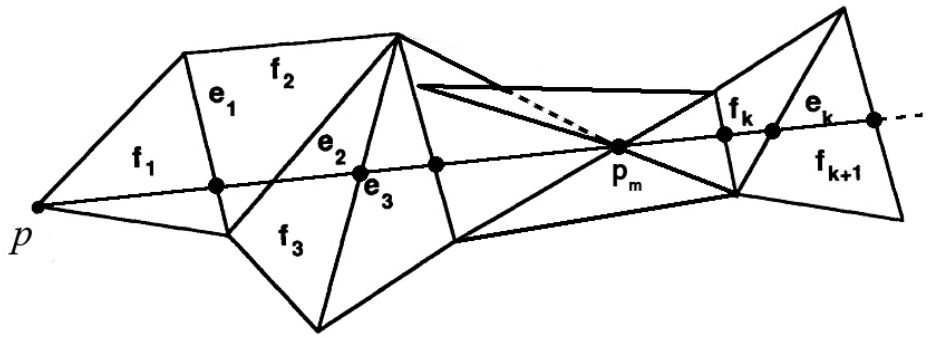


Figure 1.6: *Geodesic on a polyhedral surface, where $f_1 \cdots f_k$ is the planar unfolding of S. $e_1 \cdots e_k$ is the edge sequence E the geodesic goes through, p_m is a hyperbolic vertex on the polygon surface.*

Figure 1.6 illustrates a geodesic starts at point p and goes through a sequence edge $e_1 ... e_k$. After each all the that a geodesic goes through are flattened, its path forms a straight line.

According to the theory of Chopp (1993), 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 $\vec{b}$ which is indicated in Figure 1.4, the geodesic curvature κ_g can be eliminated gradually. When this iteration converges, which means the geodesic curvature at every point on this piecewise curve has been eliminated, this piecewise curve becomes a geodesic on the polyhedral surface.

In the following sections, a novel geodesic curvature flow based geodesic computation scheme is introduced in detail. Two algorithms have been developed to solve different problems,

1. Accurate geodesics computation on polyhedral surface
2. Linear time complexity approximate geodesics computation with a bounded error on polyhedral surface

1.2 Geodesic Curvature Flow

Based on geodesic curvature defined in Definition 1, an iterative scheme is proposed for computing geodesics over meshes. This scheme employs iterative regression to eliminate 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. This curve C 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:

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

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

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

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 $\tau = \langle -\vec{n}, \vec{n_b} \rangle$ 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 standard unit normal of surface S, κ_p denotes geodesic curvature and τ denotes geodesic torsion. The Frenet–Serret formula can be defined on a tangent plane T_p by,

Equation 1.5 can be written as,

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

where $\kappa_n = \left\langle \vec{C_s''}, \vec{n} \right\rangle$ is the normal curvature at $C(s)$, $\kappa_g = \left\langle \vec{C_s''}, \vec{b} \right\rangle$ 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, 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 is 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 speed 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 velocity of the geodesic curvature flow $C(s,t)$ can be given by,

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

where $\left\langle \vec{C_s''}, \vec{n} \right\rangle \vec{n}$ is the projection of $\vec{C_s''}$ on $\vec{n}$, note that t is a time variable of the flow in which $t = 0$ states the initial status of 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 minimizes geodesic curvature $C_{s,t}''$ point-wise on $C(s,t)$ by moving $C(s,t)$ in 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 computation to the discrete surface geodesic computation. In computer graphic, a polyhedral surface can be considered as the inscribed polygon approximation of a smooth surface.

Therefore a curve on smooth surface can also be approximated by a piecewise curve consists 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 high resolution model, which usually has millions of vertices, creating parametric approximation surface for such a polyhedron requires a large amount of computational resources. In order to avoid this excess calculation, the second order derivative of $C(s)$ is directly defined on the piecewise curve by $\widetilde{p_{i-1}p_ip_{i+1}}$ as shown in Figure 1.7, 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. Thus $\widetilde{p_{i-1}p_ip_{i+1}}$ denotes a section of the piecewise approximation of smooth curve C. In this case, at p_i, the second order derivative of Cs at p_i can be estimated by Equation 1.9,

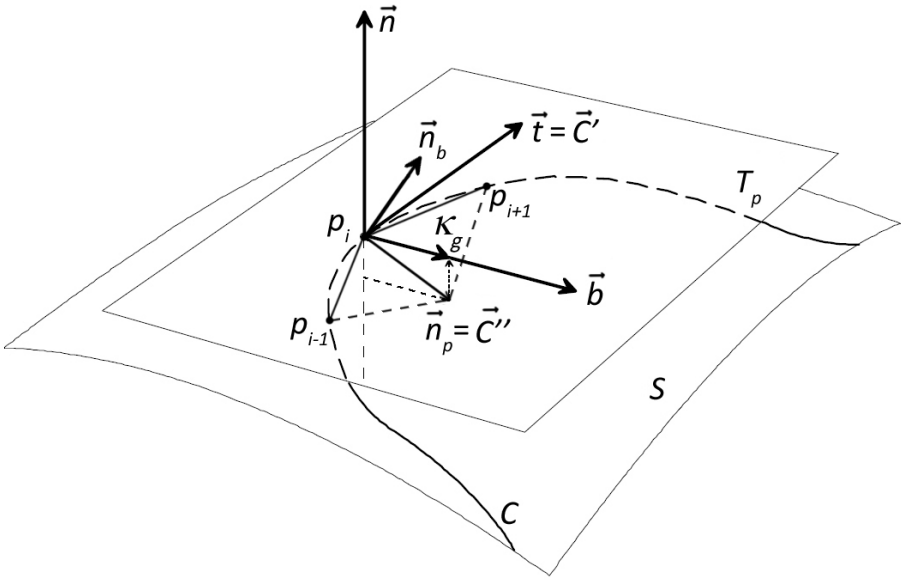


Figure 1.7: *Geodesic curvature on a piecewise curve*

$$\begin{aligned}\vec{C_s''} &\approx \overrightarrow{p_i p_{i+1}} + \overrightarrow{p_i p_{i-1}} \\ &\approx (p_{i+1} - p_i) + (p_{i-1} - p_i) \\ &\approx p_{i-1} - 2p_i + p_{i+1}\end{aligned} \tag{1.9}$$

In order to define the natural coordinate system on a vertex of a polyhedral surface, the tangent plane of S at p_i needs to be confirmed. Assume that the polyhedral surface is a discrete approximation of a 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 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 $\vec{C_s''} = KP$.

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

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 evolved from the coefficients in Equation 1.9, m denotes the number of sample points on piecewise curve. Note that, elements of the first and the last three rows of K are set to 0. This is because by using Equation 1.9 to approximate $\vec{C}_s''$ on a piece-wise curve $C(s)$, three points on $C(s)$ is required. However, on the first and the last point on $C(s)$, only one sample exist within its one-ring neighbour. Moreover, during geodesic curvature flow calculation, the position of source point and destination point should not be moved. Therefore, in K, the first and last three rows which correspond to the calculation of first and last point on $C(s)$ are set to 0 explicitly to avoid the movement on both source and destination point of $C(s)$.

Next, constructing a matrix $\vec{n}$ which is a block diagonal matrix consists of normal vectors of the tangent plane at every point in P.

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

Now, geodesic curvature flow $C(s,t)$ can be rewritten as,

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

Let P_t denotes all the sample points 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,

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

where μ is an iterative step size which heavily affects the stability of curve updating. In order to maintain the stability, μ is usually set to a small value, in our experiment, $\mu = 0.0001$. However, setting μ into a small value requires extensive increment on number of iterations required for eliminating geodesic curvature.

1.3 Geodesic on Mesh

1.3.1 Tangent Space Constraint

In order to calculate geodesics on a 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.

Because Equation 1.11 updates points on $C(s)$ in the direction that is on tangent plane of surface S at the initial position of p. If S is a non-flat surface, moving a point on tangent plane will cause this point to deviate from S. In order to solve this problem, a constrain in the tangent space needs to be applied to the movement of sample points. This can be implemented as follow.

Firstly,let P denotes all the sample points on curve C, all the normal vectors of one-ring neighbour faces of the closest vertex to p_i on S 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, then the tangent space can be defined by the eigenvector which associated with the largest 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

tangent plane at the closest vertex to p_i on S. Every time when p is updated by Equation 1.12, it is always projected back to tangent plane of the closest vertex to it. In essential, the tangent space constraint of Equation 1.12 always 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.

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

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 minimize κ_g by using 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).

In order to apply backward Euler scheme to Equation 1.13, firstly, move all parts which contains P_t on the right hand side of Equation 1.13 to the left hand side. Equation 1.13 then can be written as,

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

New, let the P_t in parts which was moved from right hand side of Equation 1.13 are evaluated by P_{t+1}. Then applying the backward scheme to Equation 1.13 yields,

$$(I + \mu(\vec{n}\vec{n}^T - K + \vec{n}\vec{n}^T K))P_{t+1} = P_t + \mu\vec{n}\vec{n}^T\overline{P_t} \tag{1.15}$$

where I is an identity matrix, μ denotes the step size. The resulting geodesic curvature flow of Equation 1.15 remains stable even at $\mu \to \infty$ (Hundsdorfer & Verwer 2003). Moreover, Equation 1.15 is the linear equation form of $AX = B$ where $A = I + \mu(\vec{n}\vec{n}^T - K + \vec{n}\vec{n}^T K)$, and $B = P_t + \mu\vec{n}\vec{n}^T\overline{P_t}$. 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 1.15. The experiment shows that the geodesics result from Equation 1.15 still requires a few iterations in order to achieve convergence. Because during the convergence of Equation 1.11 , all the sample points on 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),

$$\left(K - \vec{n}\vec{n}^T K\right) p = \left[p_{1_x}, p_{1_y}, p_{1_z}, 0, ..., 0, p_{m_x}, p_{m_y}, p_{m_z}\right]^T \tag{1.16}$$

where,

$$K = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & & & & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & & & & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & & & & 0 \\
1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & \cdots & & & 0 \\
0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & \cdots & & 0 \\
0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & \cdots & 0 \\
 & & & \ddots & & & \ddots & & & \ddots & & \\
0 & \cdots & & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & 0 \\
0 & \cdots & & & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 \\
0 & \cdots & & & & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 \\
0 & \cdots & & & & & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & \cdots & & & & & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & \cdots & & & & & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

$$\vec{n} = \begin{bmatrix} 0 & & & & \\ 0 & & & & \\ 0 & & & & \\ & n_{2x} & & & \\ & n_{2y} & & & \\ & n_{2z} & & & \\ & & \ddots & & \\ & & & n_{m-1x} & \\ & & & n_{m-1y} & \\ & & & n_{m-1z} & \\ & & & & 0 \\ & & & & 0 \\ & & & & 0 \end{bmatrix}_{3m \times m}$$

where K is a $3m \times 3m$ size matrix, $\vec{n}$ denotes the normal of the tangent plane. Essentially, Equation 1.16 is the 1^{st} order approximation to the geodesic curvature flow.

Here, define a matrix A which contains all the geodesic curvature of sample points on curve $C(s)$.

$$A = K - \vec{n}\vec{n}^T K \tag{1.17}$$

Then a linear system can be constructed from Equation 1.16,

$$AP = [p_{1_x}, p_{1_y}, p_{1_z}, 0, ..., 0, p_{m_x}, p_{m_y}, p_{m_z}]^T \tag{1.18}$$

where, m is the number of sample points on piecewise curve C. Next, define a matrix N^* where $N^* = \vec{n}\vec{n}^T$ and a matrix B, where $B^* = \vec{n}\vec{n}^T\bar{P}$. Combining Equation 1.18 with tangent plane constrain defined by Equation

1.12, the Equation 1.18 can be written as,

$$\begin{bmatrix} A_{0,0} & A_{0,1} & \cdots & A_{0,3m} \\ A_{1,0} & A_{1,1} & & \\ \vdots & & \ddots & \vdots \\ A_{3m,0} & & \cdots & A_{3m,3m} \\ N^*_{0,0} & N^*_{0,1} & \cdots & N^*_{0,3m} \\ N^*_{1,0} & N^*_{1,1} & & \\ \vdots & & \ddots & \vdots \\ N^*_{3m,0} & & \cdots & N^*_{3m,3m} \end{bmatrix} P = \begin{bmatrix} p_{1_x} \\ p_{1_y} \\ p_{1_z} \\ 0 \\ \vdots \\ 0 \\ p_{m_x} \\ p_{m_y} \\ p_{m_z} \\ B^*_{0,0} \\ \vdots \\ B^*_{3m,0} \end{bmatrix} \tag{1.19}$$

where, $A_{i,j}$, $N^*_{i,j}$ and $B_{i,j}$ denote the corresponding element in row i and column j in matrix A,N^* and B accordingly. $\bar{P}$ is a column vector consists of all the projections of p_i on 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 a 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 1.19 cannot computes 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 1.19, which is then solved to obtain the final locations. According to this scheme, the final location of a sample point is still constrained within the tangent space defined by its initial location. Note that the solution of Equation 1.19 cannot guarantees 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 is 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 from smooth surface to its approximate mesh.

1.3.4 Geodesic Path Projection

Although sample point updated by Equation 1.13 only moves within tangent space, the actual mesh is not flat. Therefore, all updated points tend to deviate from mesh. This is showed in Figure 1.8,

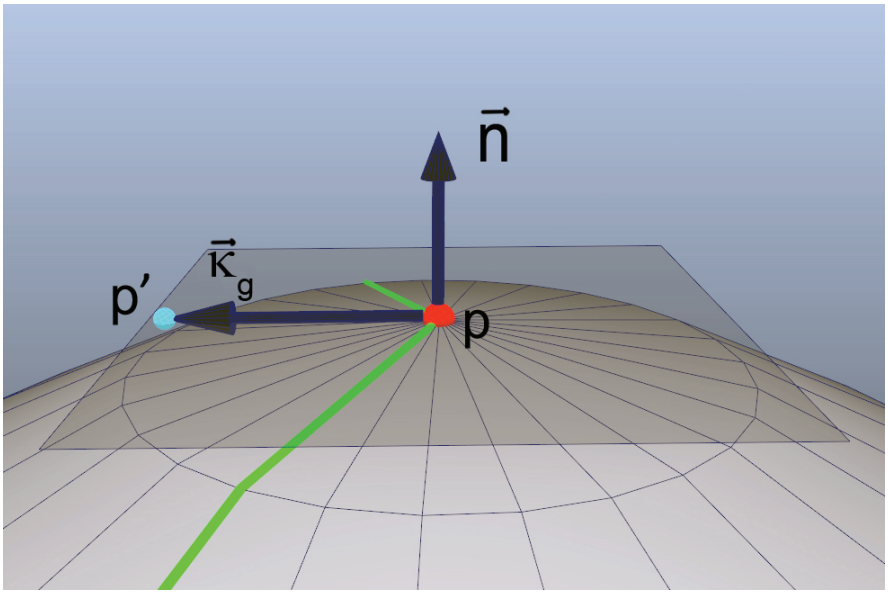


Figure 1.8: *Derived point caused by updating its location within tangent space, green piecewise curve is the curve before it is updated by Equation 1.11, gray plane is the tangent plane at point p (red point), the point p' (light blue point) is the updated point.)*

In order to calculate geodesics on the polyhedral surface, the offset geodesic curve needs to be projected back onto 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-

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 1.9.

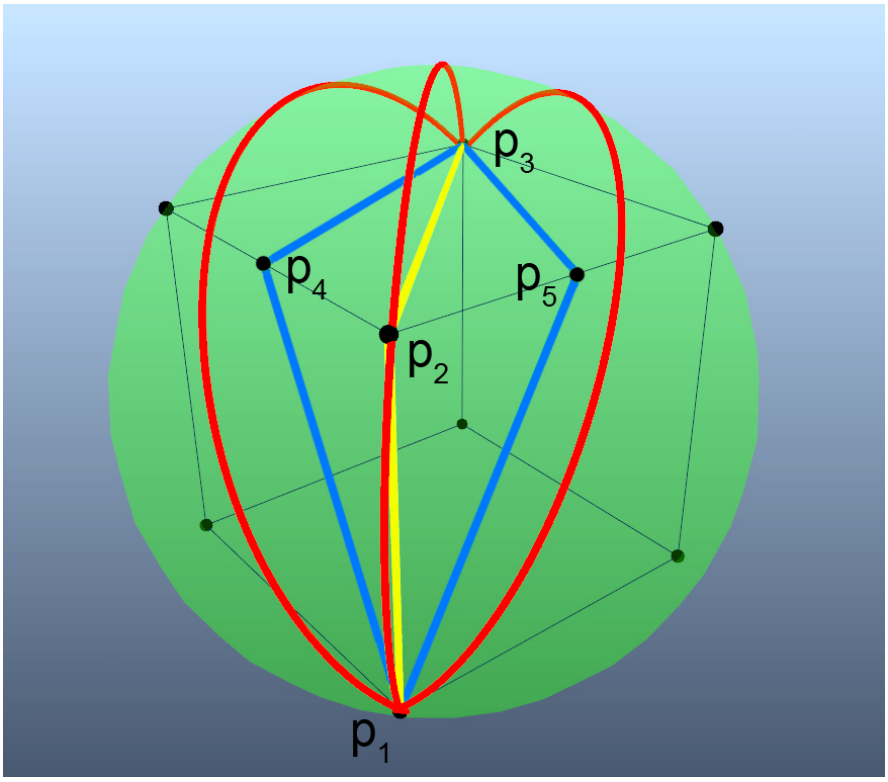


Figure 1.9: *Geodesic on smooth surface and its projection on its inscribed polygon mesh, the green sphere contains its inscribed cubic, the red curves is the geodesic on sphere (great circle arc),* p_1,p_2 *and* p_3 *are points on sphere,* p_4 *and* p_5 *are the centre point on its edge.*

In Figure 1.9 both the yellow line $\widetilde{p_1p_2p_3}$ and the blue lines $\widetilde{p_1p_4p_3}$, $\widetilde{p_1p_5p_3}$ are projections of their corresponding red geodesic path on cube. Path $\widetilde{p_1p_2p_3}$ that goes through the spherical vertex p_2 is the straightest but not the shortest geodesic path on cube. However, the blue lines that pass p_4 or p_5 is both the straightest and shortest path on cube according to the definition presented in (Polthier & Schmies 2006). Note that $\widetilde{p_1p_4p_3}$ and $\widetilde{p_1p_5p_3}$ are projections of the great circle on sphere in the direction of the curvature of their corresponding great circle.

Here, in order to project floating geodesics onto mesh, firstly, the direction of projection needs 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 p on surface S. This is shown in Figure 1.4. Therefore, the curvature κ at point p can be written as:

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

where $\vec{\kappa_g}$ is the component of $\vec{C''}$ along $\vec{b}$, $\vec{\kappa_n}$ is the component of $\vec{C''}$ along $\vec{n}$. $\vec{n}$ is the normal vector of point p on S. Let the magnitude of vector $\vec{\kappa_g}$ be 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 κ_g at p by:

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

where θ is the angle between osculating plane of C at point p and tangent plane T_p. Therefore, if p moves along the direction of $\vec{C''}$, value of θ will not be changes and osculating plane of C at point p stays the same. Hence 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 be 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 one-ring neighbour face 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.

Firstly, select p_0 as the starting point that always lies on a vertex of mesh. With two successive sample points p_1 and p_2 on the floating geodesic, these three points form a plane which cuts mesh S. This plane, $\Delta p_0 p_1 p_2$, is viewed as the approximation of the osculating plane at p_1, this is shown in Figure 1.10.

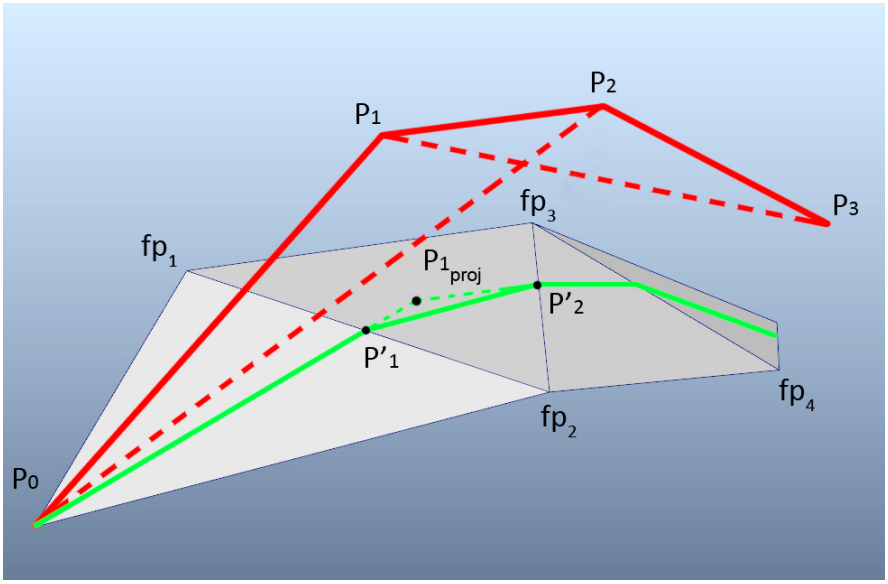


Figure 1.10: *Projection of a geodesic path.*

In Figure 1.10, red line indicate the piecewise approximation of a geodesic. p_0,p_1,p_2,p_3 are the sample points on geodesic path $\widetilde{p_0p_3}$. $\Delta p_0p_1p_2$ is the approximation of the osculating plane at p_1. fp_1 fp_4 are the vertices on polyhedral surface. P_{1proj} is the projection of p_1. p_0 which 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 $\Delta p_0p_1p_2$ and $\vec{N_2}$ denotes the normal vector of face $\Delta p_0fp_1fp_2$. Firstly, starts from the source point p_0, the rectifying plane of the projected path is approximated by $\Delta p_0fp_1fp_2$. As a result, the tangent vector of projected geodesic at p_0 can be calculated by Equation 1.20.

$$\vec{v} = \vec{N_1} \times \vec{N_2} \tag{1.20}$$

where $\vec{v}$ is equivalent to $\vec{p_0p'_1}$. Following this direction, $\vec{v}$ intersects with an edge within the one-ring neighbour faces of p_0. p'_1 denotes the intersection on edge $\overline{fp_1fp_2}$. Then, p'_1 is selected as the next starting point. Am impor-

tant property of manifold mesh is that, only one or two faces are incident to an 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 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 intersecting point on face $\Delta f p_1 f p_2 f p_3$ can be determined by Equation 1.20. If the projection of p_1 falls into the face $\Delta f p_1 f p_2 f p_3$ as shown in Figure 1.10, $\vec{N_1}$ is updated by the normal vector of approximated osculating plane $\Delta p_1 p_2 p_3$ at point p_2. Starting from projection $p_{1_{proj}}$ of p_1, perform Equation 1.20, the next intersection of $\vec{v}$ and $\overline{f p_2 f p_3}$ can be determined as p_2' shown in Figure 1.10. The projection method is summarized in Algorithm 1

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 $i = 1, i < m, i++$ **do** (m denote the number of sample points on a floating geodesic)
    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
        $\vec{t} \leftarrow p_s + \vec{N_1} \times \vec{N_2}$
        Compute the intersection point p' that an edge of current face F' intersects with $\vec{t}$
        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

1.3.5 "Continuous Dijkstra" Propagation

As described in Equation 1.11, this algorithm needs an initial path to calculate a updated geodesic path. "continuous Dijkstra" strategy presented by Dijkstra (1959) is used to generate this initial path. When performing "continuous Dijkstra" strategy, all points in the propagation boundary are kept sorted by the distance back to the source point in an incremental order in a priority queue. At each step, starting from the first element in the priority queue, the boundary is propagated outward. New points are inserted into the queue at the location where the order of queue remains increment.

However, keeping the priority queue involves "comparison sorting algorithm" which is a searching algorithm which costs $log(n)$ 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 a high resolution model. Because Equation 1.11 only requires an initial path that approximates to the shortest path. Vertices in the propagation boundary do not need to be kept in such an order. This algorithm is presented in 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 connects p_s and a point in W is the geodesic path between them. This is due to within a triangle, the shortest path between two points is the edge that connects these two points. Next, the length of every edge is stored along with the path information to vertices in W. Since this algorithm does not employ priority queue,in order to maintain isometric propagation, a propagation radius limitation r_{max} is introduced to constrain the propagation at every step. At this moment, r_{max} is updated as the length of the longest edge that connects p_s and W.

After the first W is formed by one-ring neighbour vertices of p_s, the first vertex $p_i \in W$ is selected and its one-ring neighbour vertices exclud-

ing vertices that have 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 in order to form its initial path from p_s to pn_i. The algorithm presented in this chapter calculates geodesics by eliminating geodesic curvature at each sampling point on curves. When performing propagation outwards, each unvisited vertex needs to be connected to a visited vertex to form a new initial path for geodesic curvature flow calculation. For each visited vertex, the path from source vertex to this vertex is already a straightest geodesic path resulted by solving Equation 1.19. For each unvisited vertex, in order to form an initial path that is close to its "ideal" straightest geodesic path, the section from this unvisited vertex to its adjacent visited vertex should follows the direction of geodesic path from source to this visited vertex. Therefore, each unvisited vertex always selects a geodesic path which can form the straightest initial path within its one-ring-visited vertices. This process is illustrated in Figure 1.11.

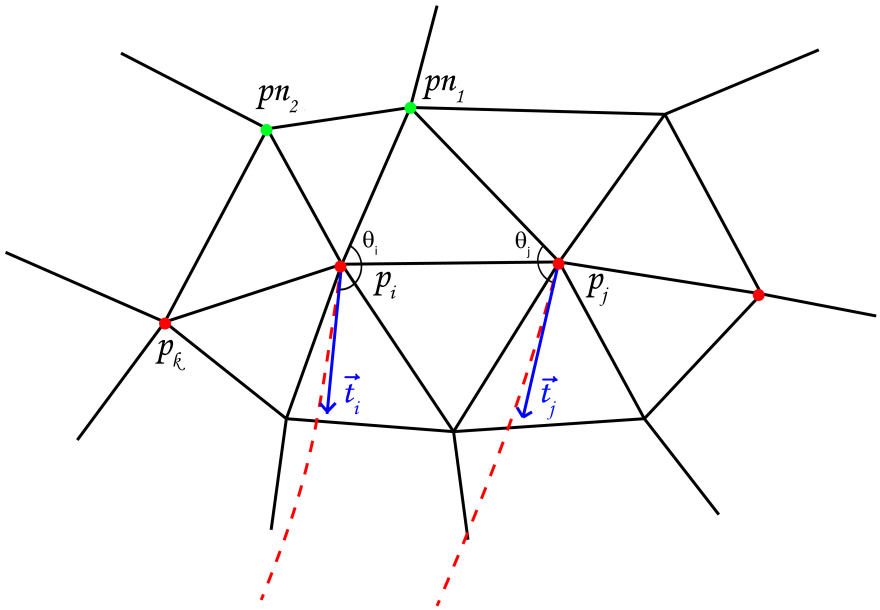


Figure 1.11: *Calculating geodesic path for a unvisited vertex, red point p_i,p_j and p_k are visited vertex in wavefront W, red dash line indicates geodesic path from source to p_i and p_j, vector $\vec{t_i}$ and $\vec{t_j}$ is the tangent vector at p_i and p_j respectively, pn_1 and pn_2 are one-ring neighbour of p_i that has not been visited.*

Where, pn_1 and pn_2 is in the one-ring neighbour N' of p_i that have not been visited. For a random vertex $pn_1 \in N'$, the visited vertices in one-ring neighbour of pn_i are selected which in the example demonstrated in Figure 1.11, 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 $\vec{t_i}$ and the edge that connects pn_1 and current selected visited vertex is calculated. $\vec{t_i}$ denotes the tangent vector at current visited vertex on its geodesic path. The parent node of pn_1 is then selected from the visited vertex with the largest θ.

In Figure 1.11, the included angle θ_i between $\overline{p_i pn1}$ and $\vec{t_i}$ is greater than the included angle θ_j between $\overline{p_j pn1}$ and $\vec{v_j}$. Therefore, it can be observed that, comparing two curves of edge $\overline{p_i pn_1}$ plus geodesic path at p_i and edge $\overline{p_j pn1}$ plus geodesic path at p_j, the curve of edge $\overline{p_i pn_1}$ plus geodesic path at p_i are straighter than the curve of edge $\overline{p_i pn_1}$ 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 is pushed to the end of its parent point's geodesic path, it contains the entire information of the parent vertex. As a result, the algorithm presented here does not require "backtracing" that are necessary for MMP, CH and FMM algorithms in order to retrieve the geodesic path. The aforementioned method propagates the geodesic information from the interior ring to the external one. However, without the help of priority queue, propagating the geodesic information from the vertex with smaller geodesic distances to the vertex with larger distances can not be guaranteed. Therefore, in order to maintain the isotropy of the propagation, a propagation radius limit r_{max} is employed. 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 $\overline{pn_1 p_i}$ plus geodesic distance from source to p_i. Then 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 stored into an array W_{next} as the "next wavefront" for next propagation. For every unvisited vertex that connects 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}, the presented 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 at every step. 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 the geodesic paths 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.

When solving Equation 1.11, the updated points on geodesic path always move away from the vertices on initial path. Therefore, in Algorithm 2, in order to calculate the projections of the sample point p on floating geodesic path, examining every single face on the mesh is a very time consuming process. In order to maintain the efficiency of the presented algorithm, it is important to keep track on the corresponding vertex on the mesh where p is updated from. Based on the observation from our experiments, for every unvisited vertex, 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, after the "wavefront" propagates for more than once, this assumption does not stand any more. Thus one-ring neighbourhood of every sample point needs to be redetermined after the initial path is updated by Equation 1.11. This is achieved by the following procedure. Firstly, starting from one-ring neighbour vertices 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 as the "new" initial vertex of point q. This updating process is able to ensure the projection of the updated sample point q on floating geodesic path always falls into the one-ring neighbour faces that is adjacent to q_j.

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

During the projection, the osculating plane is formed by three successive sample points on a geodesic. This osculating plane is used to cuts the related faces which are one-ring neighbours of these three 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 face can easily be determined. As a result, calculating the intersections between the cutting plane and all the related edges costs a constant time.

A challenging issue of using Algorithm 2 to calculate geodesic is to solve large sparse linear system of Equation 1.18. Moreover, the size of matrix in Equation 1.18 is determined by the number of sample points on a path. Therefore, followed by the propagation of "wavefront", the size of matrix in Equation 1.18 increases. This increment on the matrix size of Equation 1.18 results an very expensive computational cost for calculating geodesics on a mesh. In fact, solving Equation 1.18 is the most time consuming process in the entire geodesic computation.

In order to improve the efficiency of Algorithm 2, an approximate algorithm is derived from it. In this algorithm, a small-size window is introduced to Equation 1.11 to avoid solving large sparse linear system. When an initial estimation of a geodesic path is obtained, as shown in Figure 1.11, a fixed-size window that covers the last w sample points is employed. Then Equation 1.11 is applied to w to calculate a 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, window size w is a constant number, therefore, the matrix size in the linear system of Equation 1.11 is also constant throughout the propagation. Because for each unvisited vertex on mesh, the time used for

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,$v_{unvisited}$)(W denotes propagation boundary)
  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

solving Equation 1.11 is constant and the time used for calculating geodesic is also same. Therefore, by introducing the concept of window to each geodesic update, Algorithm 3 is able to achieve linear time complexity. However, when window size is fixed, only a section of the initial path is updated by Equation 1.11 which means the geodesic curvature at points fall into this section is eliminated, the rest sample point on initial path are not updated. Combining the updated section with the rest initial path resulting a path which is not a true geodesic path but an approximation of the "true" geodesic path. Moreover, larger the w is set, closer the resulting path is to the true geodesic path and higher the accuracy of approximation is but longer the computation for solving Equation 1.11 is. Hence, in the experiment section of this chapter, an experiment for using different window sizes on a model is designed to illustrate the relationship between window size and computational cost as well as the relationship between window size and approximation accuracy.

In real world, when using tape ruler to measure the length of a body part, it follows the profile of body in a convex-hull like manner that it only makes contact with subject body on the convex part. This is demonstrated in Figure 1.12.

Figure 1.12: *Tape measuring on human body*

The shape of upper leg is a convex shape, therefore, the tape ruler follows its profile till knee cap. Because the knee cap is much higher than the surface of lower leg, the lower section of leg between ankle to knee cap is a concave shape, therefore, tape ruler from knee to ankle forms a straight line.

However, geodesic path only travels on curved surface no matter the shape of surface is concave or convex, therefore, a difference exists in the measurement data acquired by tape ruler measuring and the one acquired by geodesic path measuring. This difference is become larger on muscular character whose body surface is more uneven than a skinny character.

In order to improve the measuring quality of the presented geodesic path measuring method, a path correction procedure is designed to straighten the section of a geodesic path where it travels through a concave part of character body. Firstly, all the vertices normal of character model are set outward first. Because at each sample point on a geodesic path, the geodesic curvature is eliminated by Equation 1.19, the normal vector at each sample point collinear to the normal vector of surface. Based on the direction of the normal vector at a sample point on a geodesic path and its corresponding surface normal vector. A geodesic path can be segmented into several sections. This is shown in Figure 1.13.

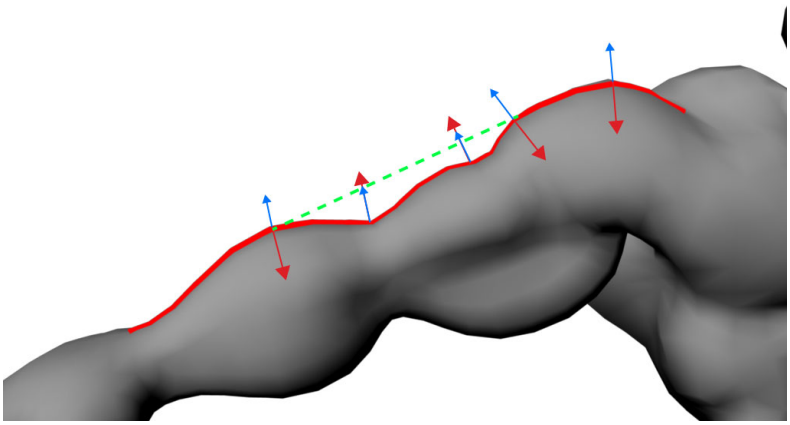


Figure 1.13: *Curve normal and vertex normal on a geodesic path on the arm of character A*

In, Figure 1.13 red path is the geodesic path used for measuring arm length. Red arrows demonstrate curve normal at sample points on path, blue arrows indicate vertex normal on the character model. Green dashline indicates the path where tape ruler would went through based on our observation. Because red path is a geodesic path, at each sample point, its curve normal collinear with the vertex normal at corresponding location on mesh. Note that, on the convex part of mesh, the curve normal and vertex normal of mesh are in opposite direction, whilst on the concave part of mesh, the curve normal and vertex normal of mesh are in the same direction. The presented path shape correction procedure iteratively moves each sampling point which is in the same direction as it corresponding vertex normal to its curve normal direction till all the sample points on a curve either in opposite direction with its vertex normal or has zero curvature. Here, defining concave point as a sample point on a geodesic path whose curve normal is in the same direction with its vertex normal on mesh. The geodesic path shape correction procedure is illustrated in Figure 1.14.

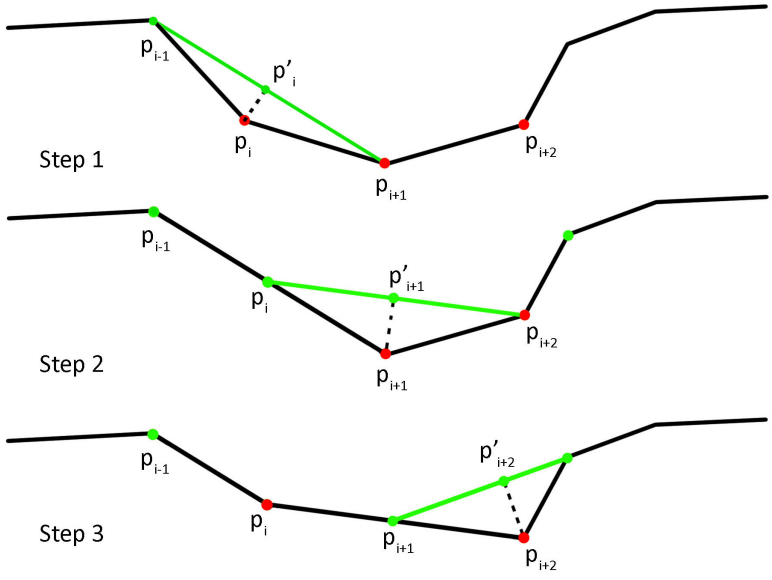


Figure 1.14: *Demonstration of path correction process*

Where black piece-wise curve is a geodesic path on a character, green

point indicates a non-concave point, red point indicates a concave point. The procedure works in following manner, firstly, select a concave point p_i. Then project p_i onto a line defined by two adjacent points of p_i, where in Figure 1.14 are p_{i-1} and p_{i+1}. Update p_i by its projection p'_i. Select next concave point p_{i+1}, project p_i onto a line defined by p_i and p_{i+2}. Update p_{i+1} by its projection p'_{i+1}. Continue this process till all concave points are become non-concave. Figure 1.15 demonstrates the result of path shape correction process on Character A.

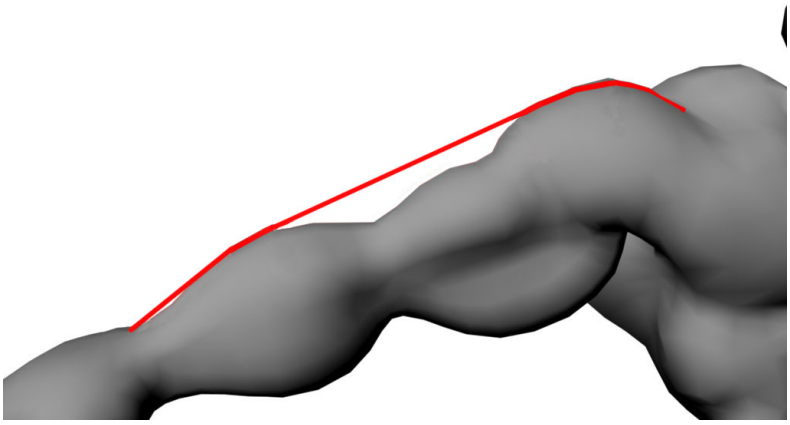


Figure 1.15: *A straightened path for measuring arm length.*

Moreover, for characters with bent arm where the geodesic path goes through inside of elbow, shown in Figure 1.16, perform this shape correction process will straighten the entire geodesic path. In order to force the corrected path still follows the trend of a bent arm, user can select a section of on a geodesic path where this correction procedure will ignores, such as area around elbow, therefore, at the the selected area, the path still follows the shape of character model meanwhile the concave part in the rest of geodesic path will be straightened, Figure 1.17 demonstrates the result on a muscular character.

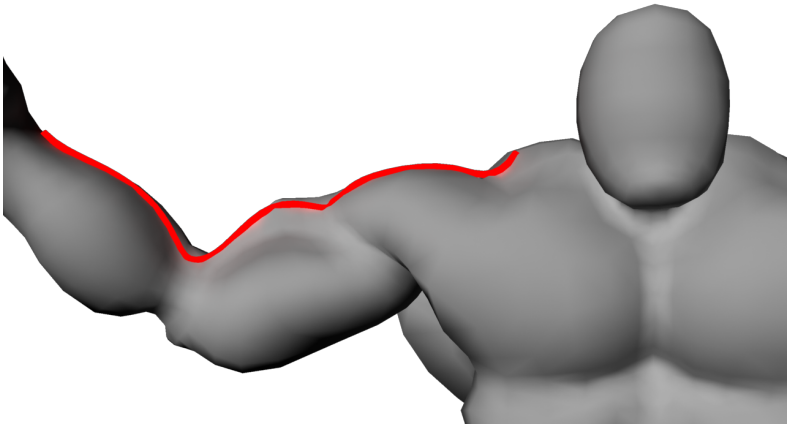


Figure 1.16: *Geodesic path on a bent arm of Character A*

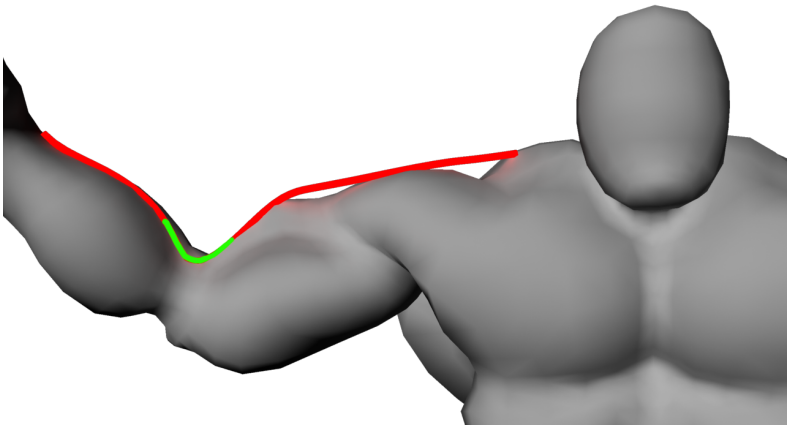


Figure 1.17: *Shape corrected path with a section selected by user (green section) to be ignored by shape correction process*

1.4 Performance Analysis

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

1.4.1 Efficiency

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 1.18 is not a square matrix, it is necessary to left-multiply A^T on both side of the Equation 1.18 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 1.18, according to Bunch & Hopcroft (1974); Coppersmith & Winograd (1987), the LU solver has a time complexity of $O(m^{2.379})$. 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, computation of the projection for each three successive sample points of a geodesic onto a mesh costs a constant time, denotes at h, the projection of the entire geodesic costs $O(kh)$ where k denotes the number of sample points 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 matrices in Equation 1.18 is constant. Therefore, solving Equation 1.18 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.

Among the above presented algorithms, the most noteworthy achievement is the ability of computing geodesic paths in a linear time in Algorithm 3 with a bounded error. This is especially significant for tracing a large number

of geodesic paths on high resolution 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 an 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 MMP (Surazhsky et al. 2005), Improved CH algorithms (Xin & Wang 2009) and FMM (Kimmel & Sethian 1998). In order to compare the time consumption for acquiring geodesic path, "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 author's project page[1]. These algorithms are performed respectively on the Stanford Bunny's model with 8 different resolutions as shown in Table 1.2. The running times are plotted in Figure 1.18. It can be seen that the Algorithm 3 shows great advantage over others when the resolution of the model increases.

[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`

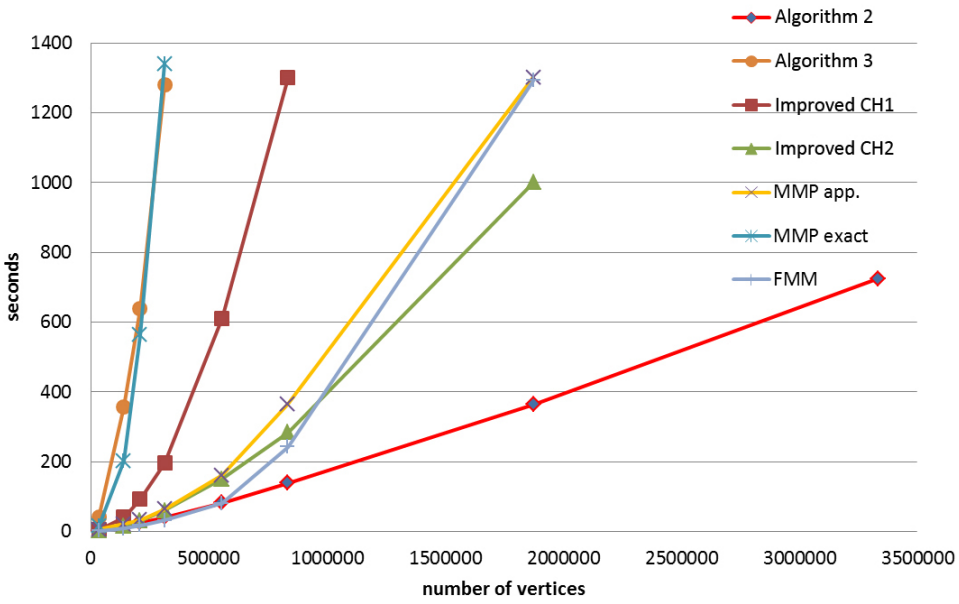


Figure 1.18: *The comparison of running time. The time of the ICH and MMP algorithms includes the cost of "backtracing", for Algorithm 3, w=30.*

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

Table 1.2: *The resolution of the bunny models used in the experiment in Figure 1.18*

1.4.2 Accuracy

On a non-convex surface, geodesic path between two points is not unique (Thielhelm et al. 2012). In order to set up a ground truth for comparing geodesic calculation accuracy. The experiment is carried out on a sphere. This is due to all great-circles between two points on a sphere are geodesics and geodesic distance between them is the length of a great-circle, this distance can also be called as orthodromic distance. The orthodromic distance

between two points on a sphere can be calculated using Equation 1.21 (Aleksandrov & Zalgaller 1967).

$$d = r \times arccos(\vec{n_1} \cdot \vec{n_2}) \tag{1.21}$$

where r denotes the radius of the sphere, $\vec{n_1}$ and $\vec{n_2}$ are to normals at two points on the sphere.

Both MMP algorithm(Mitchell et al. 1987b; Surazhsky et al. 2005) and Algorithm 2 are performed on a unit sphere and the length of each geodesic path is compared against the orthodromic distance between the source point and destination point. Figure 1.19 illustrates geodesics from one vertex to all the other vertices on a unit sphere.

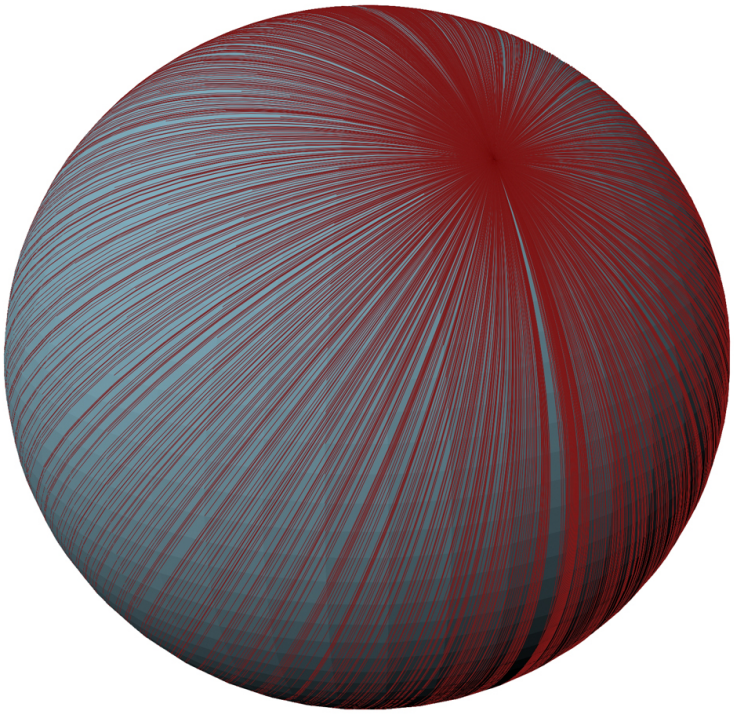


Figure 1.19: *Geodesics on a sphere.*

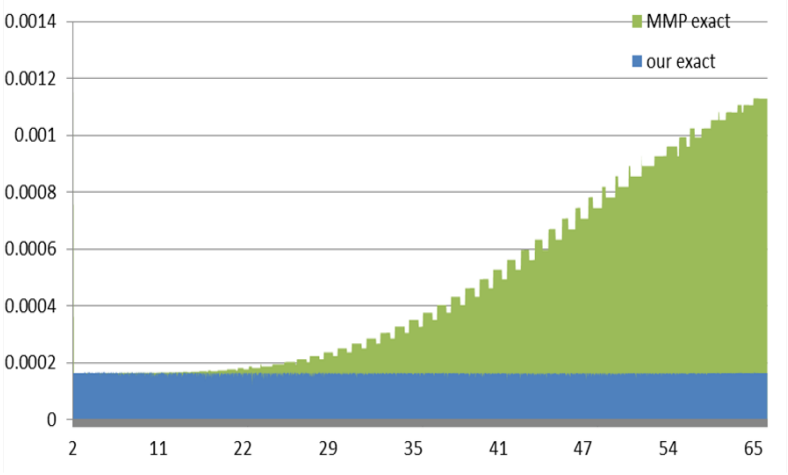


Figure 1.20: *Distribution of the relative errors of MMP and Algorithm 2*

Figure 1.20 demonstrates the relative error of MMP algorithm(Mitchell et al. 1987b; Surazhsky et al. 2005) and Algorithm 2. Moreover, this experiment is also performed to examine the accuracy of MMP approximate algorithm and Algorithm 3, the result is shown in Figure 1.21.

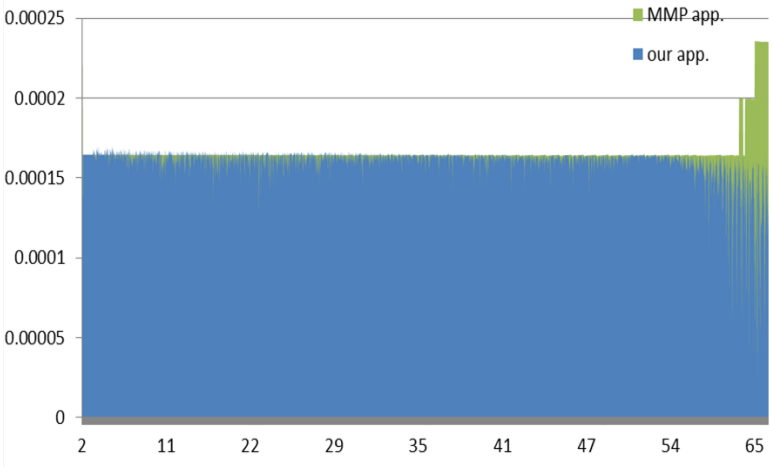


Figure 1.21: *Distribution of the relative errors of MMP approximate algorithm and Algorithm 3*

The accuracy of Algorithm 3 depends on the following parameters, the edge length e, the window size w, and the number of sample points on each geodesic. Let m denotes the number of sample points on a geodesic. 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 Equation 1.11 to this patch instead of the whole path.

Let C_g denotes a geodesic path on a polyhedral surface. C_g does not pass through any vertex p of the polyhedron unless p is the source point or the destination. It can be observed that the C_g goes across a set of faces. Unfolding this set of faces into a plane, C_g should becomes a straight line linking the source point p_s to the destination point p_d. For a new vertex q, the geodesic $\widetilde{p_sq}$ can be estimated by combining the edge $\overline{p_dq}$ with C_g. Without losing the generality, let a window covers a section of $\widetilde{p_sq}$ from q, w denotes the number of the sample points covered by this window. The error estimation for any geodesic computed by Algorithm 3 is illustrated in Figure 1.22,

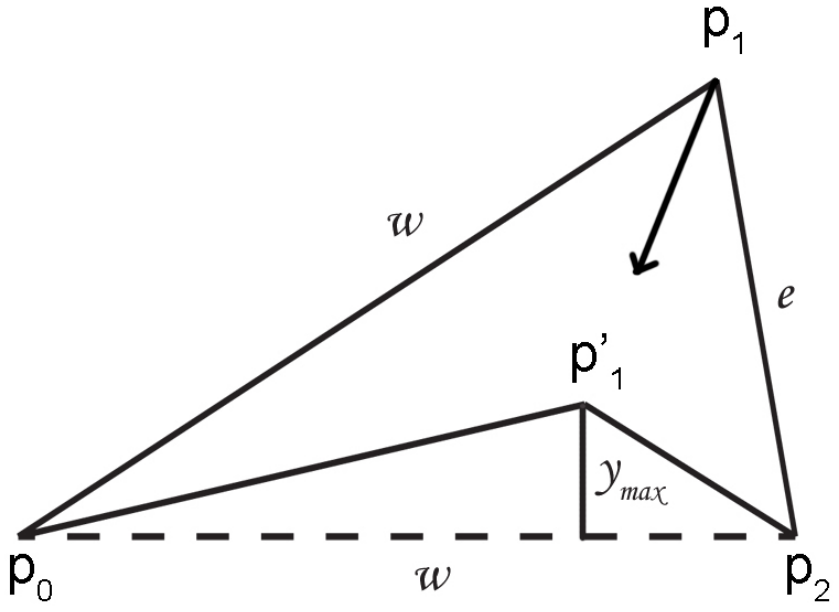


Figure 1.22: *Error estimation of Algorithm 3, the $\overline{p_0p_2}$ is the desired geodesic while $\overline{p_0p_1}$ plus $\overline{p_1p_2}$ being the estimation.*

Let p_0, p_1,p_2 are three points on a plane as shown in Figure 1.22. The initial path for solving Equation 1.11 is $\widetilde{p_0p_1p_2}$ where p_0 is the source point

and p_2 is the destination point. Obliviously, the true geodesic from p_0 to p_2 is $\overline{p_0p_2}$. Let $\overline{p_0p_1}$ be an established geodesic path, p_2 is an unvisited vertex. The length of edge $\overline{p_0p_1}$ is always smaller than or equals to the length of $\overline{p_0p_2}$ because the propagation only performs on outward direction. Therefore, in the worst situation, which is demonstrated in Figure 1.22, $\overline{p_0p_2}$ and $\overline{p_0p_1}$ are equal. p_1' is the updated point calculated by 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, based on Heron's formula (Aleksandrov & Zalgaller 1967), the area of $\triangle p_0p_1'p_2$ can be written as,

$$T_{\triangle p_0p_1'p_2} = \frac{\sqrt{(L^2 - w^2)w^2}}{4} = \frac{wy_{max}}{2} \tag{1.22}$$

therefore,

$$L = \sqrt{4y_{max}^2 + w^2} \tag{1.23}$$

Let err denotes the difference between true geodesic path length and the solution of Equation 1.11, where err can be written as,

$$err = L - \|p_0p_2\| = \sqrt{4y_{max}^2 + w^2} - w \tag{1.24}$$

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

$$err_{ave} = \frac{\sqrt{4y_{max}^2 + w^2} - w}{w} \tag{1.25}$$

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

therefore estimated by,

$$err = m\left(\sqrt{1+4(\frac{y_{max}}{w})^2}-1\right) < 2m\frac{y_{max}}{w} \sim O(\varepsilon m) \tag{1.26}$$

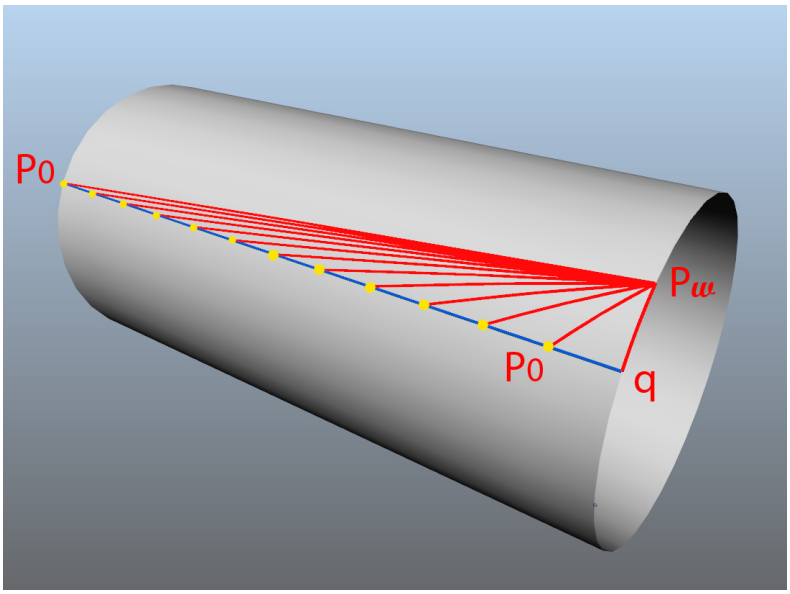
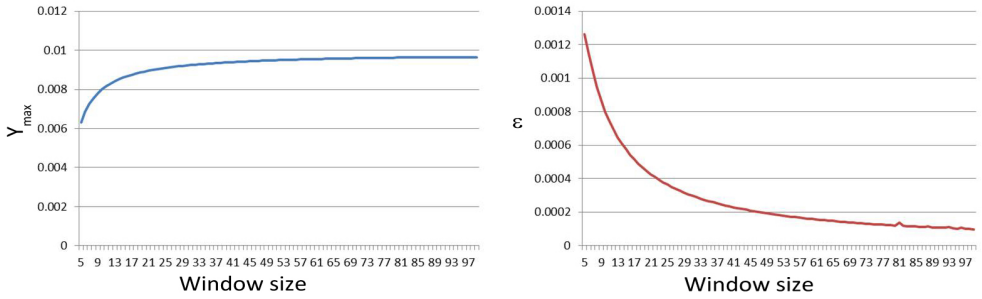


Figure 1.23: *Error estimation of the approximate algorithm with different window size*

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 y_{max}, Algorithm 3 is performed on a cylindrical surface, see Figure 1.23. $\overline{p_0q}$ is parallel to the axis of the cylindrical surface, and the true geodesic between p_0 and q is $\overline{p_0q}$. Let the length of the edge between each yellow point is e = 1 and the window size w varying from 5 to 100, The numerical results are shown in Figure 1.24a. It can be noted that when w = 5, the y_{max} tends to be zero. But, the larger the window, the bigger the y_{max}. This can be explained that on the plane shown in Figure 1.22, the curvature of the geodesic estimation $\widetilde{p_0p_1p_2}$ is becoming smaller when window size w is increased. 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 1.23, the p_0 is viewed as the start point of the window. If it is not the source, there must exist a geodesic path from source to p_0. It is ideal that the desired geodesic from source to p_2 passes through the geodesic from source to p_0. Thus, it is natural to enlarge the window size as much as possible. Figure 1.24a and Figure 1.24b shows that both the y_{max} and ratio ϵ converge to small values. This means that enlarge w does not reduce 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.

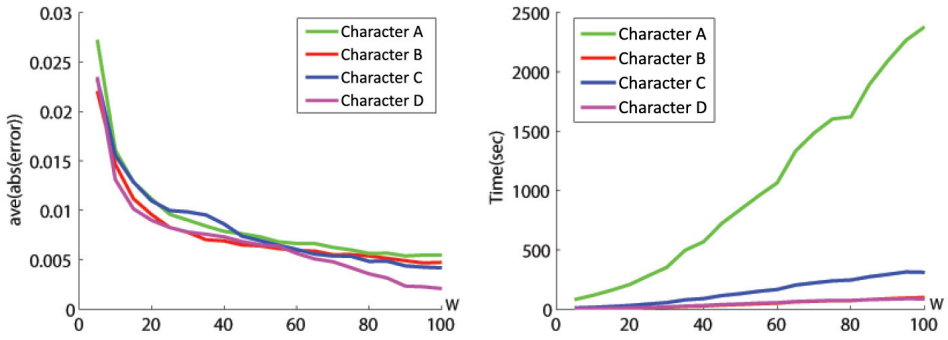


(a) *The convergence of y_{max} with varying window size*

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

Figure 1.24: *The convergenecy of error to the window size of the approximate algorithm*

In order to investigated how the running time and accuracy of Algorithm 3 vary with the window size w on a more complex model, MMP algorithm is performed on four character models to obtain the ground truth. The window size w is varied between 5 and 100. Experimental results on four character models are reported in Figure 1.25.



(a) *Average absolute errors on four models.*

(b) *Running times on the same four models.*

Figure 1.25: *Performance of Algorithm 3 with varying window size.*

According to the results, a general trend is that the approximation error decreases with an increasing window size. It drops quickly when the window size is small. However, when the window size reaches 20, the speed of improvement in accuracy starts to slow down as shown in Figure 1.25a. Meanwhile, the running time starts to increase more rapidly, as shown in Figure 1.25b. Therefore, in the character measurements extraction process, the window size w is empirically set to 30.

Further 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 solutions are considered as the ground truth for this experiment. The absolute errors is the differences between exact geodesic distances and approximate ones and the relative errors is the ratios of absolute errors over exact distances.

Characters		A #V:127490 #F:254976	B #V:367942 #F:735656	C #V:365476 #F:730760	D #V:356544 #F:712992
Algorithm 3 (floating)	ave abs	0.00309	0.00098	0.00786	0.005218
	ave rel	0.5054%	0.1911%	1.3454%	0.4852%
Algorithm 3 (projected)	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 1.3: *The average absolute and relative errors of Algorithm 3, MMP app. and ICH_2. For Algorithm 3,* $w = 30$.

Note that, for Algorithm 3, two results are kept as one is the offset geodesics and the other is the projections of the offset geodesics onto the mesh. Table 1.3 gives the average absolute and average relative error of these three algorithms.

Moreover, in order to further validate the error estimation for Algorithm 3, MMP exact algorithm is performed on the lowest resolution bunny model in Table 1.2 and length of the geodesics result from MMP exact algorithm is used as the ground truth.

The ratio ε in Equation 1.26 is usually a very small number. Figure 1.26 shows the histogram of the ratio ε over the bunny model, where horizontal axis indicates the value of ε and vertical axis indicates the number of sample points in a geodesic. Figure 1.26 indicates that on the bunny model, for all geodesics, ε remains a small value.

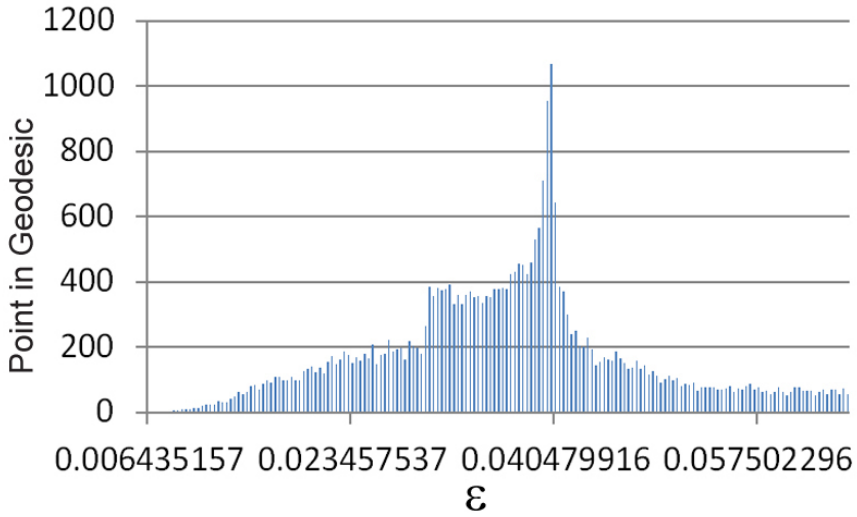


Figure 1.26: *Histogram of the ratio ε on the bunny model*

Figure 1.27 further shows the distribution of real absolute errors and the estimated ones in the bunny model test. It can be noted that the error estimation of Equation 1.26 can accurately reflects the upper error bound of Algorithm 3

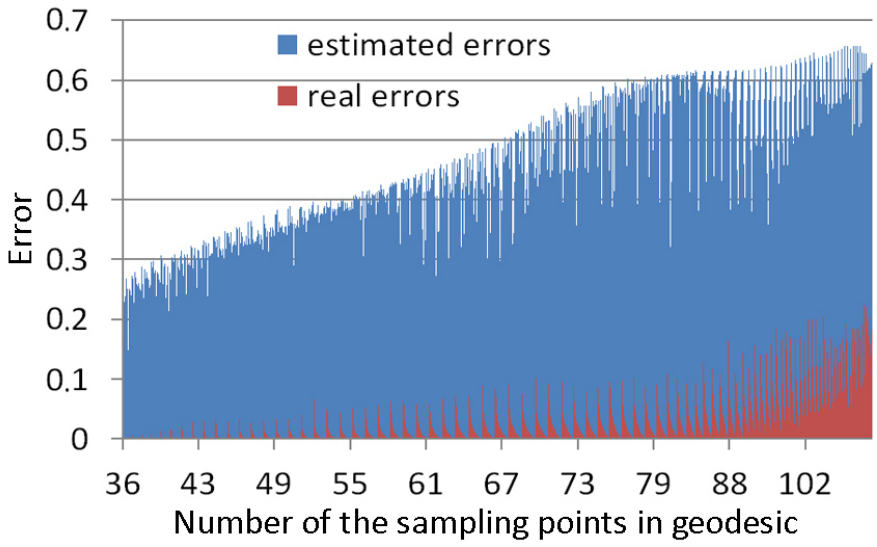


Figure 1.27: *The distribution of estimated errors and real errors, the window size of Algorithm 3 is set to 30, the real error is bounded by the estimated error for all the geodesics on the bunny model.*

1.5 Results and Conclusions

The proposed measurement method is performed on four different characters in the same hardware environment where experiments in previous section are carried out. For length measurements extraction, Algorithm 3 is utilised for geodesic path calculation. Also, MMP and ICH2 are used in order to compare the measuring efficiency. The experimental results are shown in Table 1.4.

	Character A #V:127490 #F:254976	Character B #V:367942 #F:735656	Character C #V:365476 #F:730760	Character D #V:356544 #F:712992
Algorithm 3	63.59s	194.13s	196.94s	191.55s
MMP	69.78s	479.03s	481.61s	471.10
ICH_2	65.52s	465.19s	466.84s	461.91s

Table 1.4: *Measuring time on different characters.*

Figure 1.28 to 1.32 visualize resulting measurements on four characters. Figure 1.29 demonstrates length measurements with geodesic path shape correction. In these figures, red curves indicate geodesic path used for extracting length measurements, blue curves indicate convex-hull used for extracting circumference measurements. Table 1.5 to 1.8 lists out resulting measurement data.

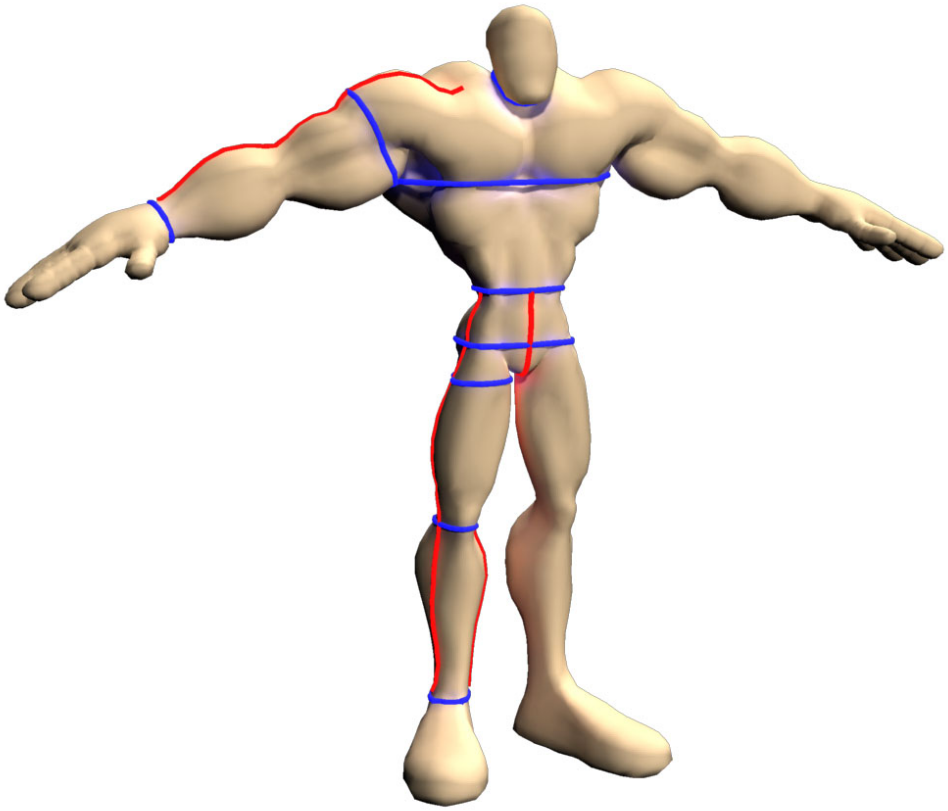


Figure 1.28: *Measurements on Character A*

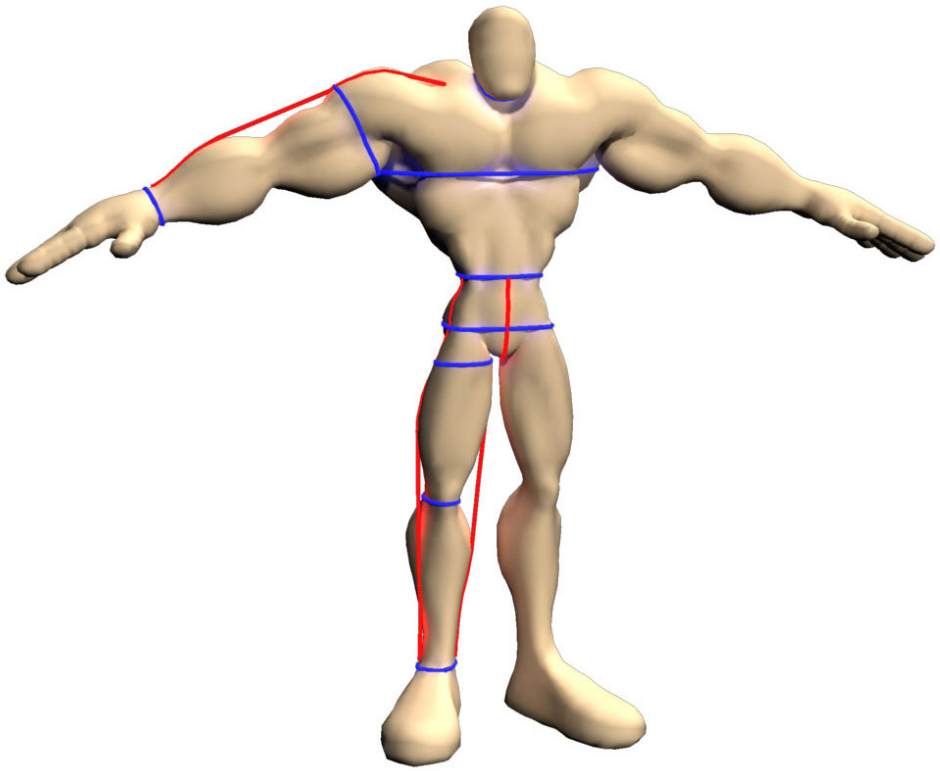


Figure 1.29: *Measurements with geodesic shape correction on Character A*

Bust Girth	211.9015	Sleeve length	122.8928
Waist Girth	73.8146	Sleeve length (with correction)	120.1646
Hip Girth	83.2698	Arm hole	157.3058
Neck Girth	67.3815	Back length	77.6125
Cuff Girth	59.3813	Outside Leg length	129.1429
Shoulder width	89.9452	Outside Leg length (with correction)	126.2821
Back width	74.8823	Inside Leg length	109.7143
Front width	88.7650	Inside Leg length (with correction)	107.2786
Ankle Girth	47.4285	Thigh Girth	65.5714

Table 1.5: *Measurements for characters A.*

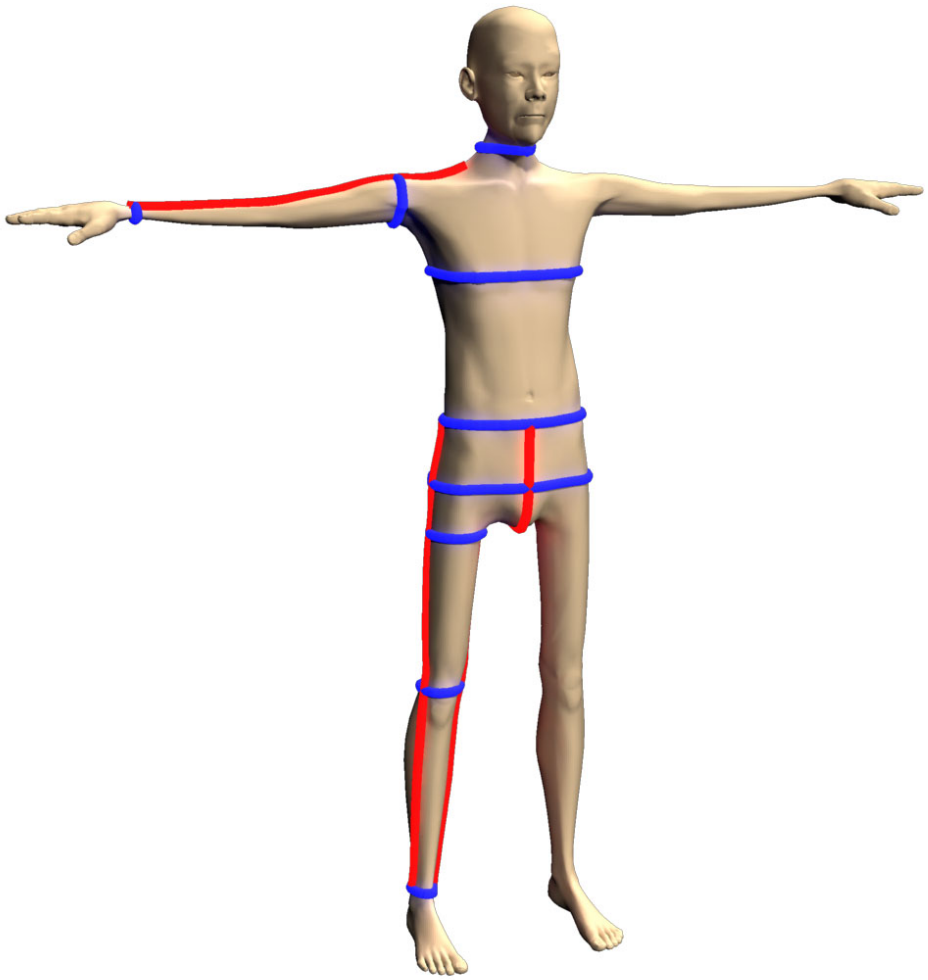


Figure 1.30: *Measurements on Character B*

Bust Girth	79.0833	Sleeve length	57.3615
Waist Girth	70.2424	Arm hole	29.2281
Hip Girth	87.2813	Back length	44.3503
Neck Girth	43.6637	Outside Leg length	89.1428
Cuff Girth	8.3771	Inside Leg length	74.2857
Shoulder width	40.0348	Thigh Girth	33.7142
Back width	38.6953	Ankle Girth	16.5741
Front width	45.7562		

Table 1.6: *Measurements for characters B.*

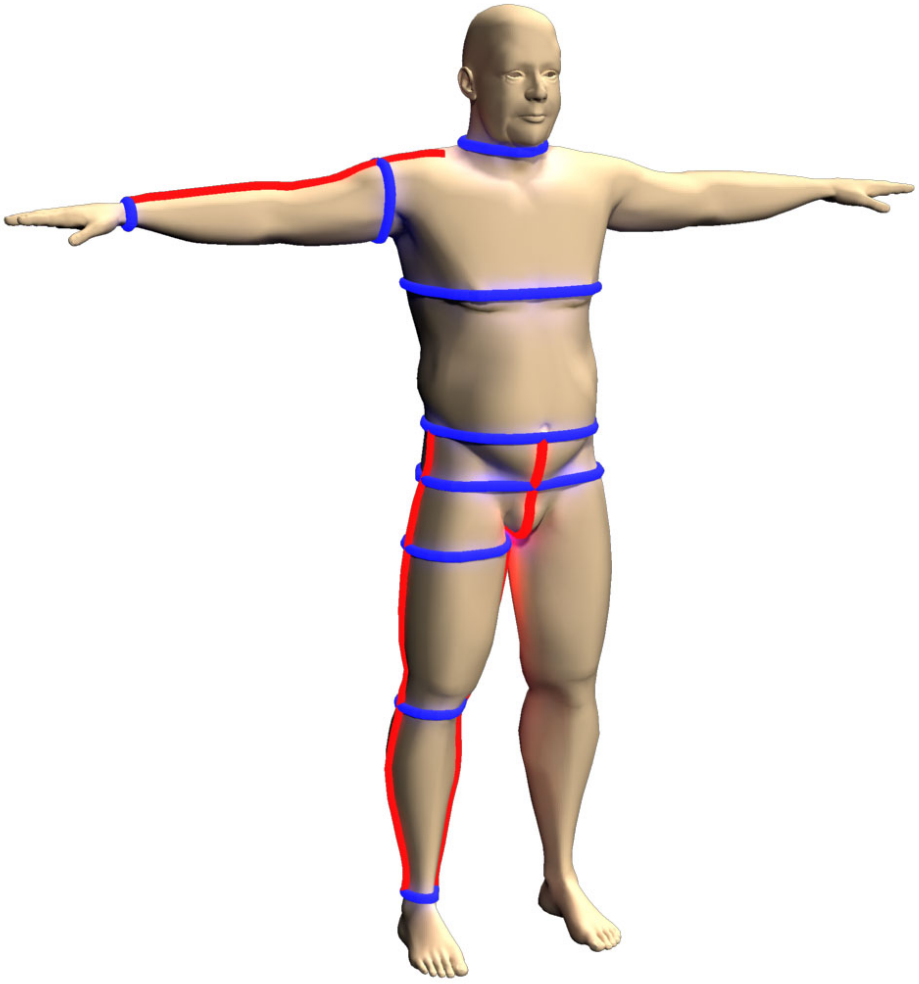


Figure 1.31: *Measurements on Character C*

Bust Girth	111.9165	Sleeve length	58.3748
Waist Girth	96.6514	Arm hole	42.9157
Hip Girth	107.5671	Back length	45.1839
Neck Girth	68.2416	Outside Leg length	85.0919
Cuff Girth	15.3368	Inside Leg length	71.4285
Shoulder width	51.3362	Thigh Girth	57.1428
Back width	50.3038	Ankle Girth	20.0174
Front width	59.4946		

Table 1.7: *Measurements for characters C.*

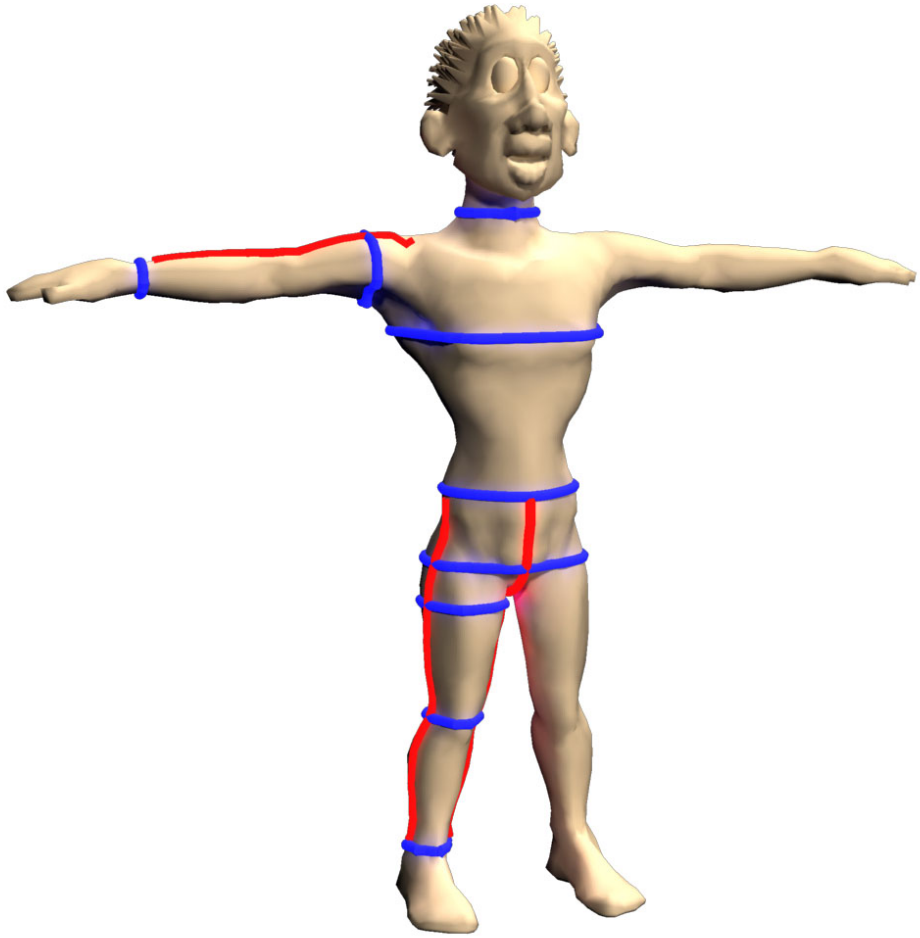


Figure 1.32: *Measurements on Character D*

Bust Girth	95.8160	Sleeve length	43.0211
Waist Girth	70.7769	Arm hole	37.9510
Hip Girth	73.1573	Back length	36.4173
Neck Girth	59.6871	Outside Leg length	51.4826
Cuff Girth	14.5720	Inside Leg length	41.3055
Shoulder width	34.2857	Thigh Girth	38.5871
Back width	32.2461	Ankle Girth	21.7142
Front width	38.1302		

Table 1.8: *Measurements for characters D.*

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

This chapter proposed a character body measurements extraction method for animation character based on convex-hull and geodesic path computation. Calculating geodesic using current method is a time-consuming task especially on high resolution character models. In order to improve the efficiency of geodesic computation, two algorithms respectively for accurate(Algorithm 2) and high efficiency approximate geodesic computation(Algorithm 3) on triangulated manifolds are proposed in this chapter. For Algorithm 3, due to the fixed window size, it is able to achieve linear time complexity with a bounded error on triangulated manifolds. Numerical comparisons with existing algorithms (i.e. MMP, ICH_1 and ICH_2) have further demonstrated the advantages of Algorithms 3 in terms of both speed and accuracy. Experiments have shown that by integrating Algorithm 3 into the measuring system, the time consumed for measuring a character has been largely reduced.

Bibliography

A. Aleksandrov & V. Zalgaller (1967). *Intrinsic geometry of surfaces*. Translations of mathematical monographs. American Mathematical Society.

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

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.

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.

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.

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.

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.

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.

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

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

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.

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.

G. Zigelman, et al. (2002). 'Texture mapping using surface flattening via multidimensional scaling'. *Visualization and Computer Graphics, IEEE Transactions on* **8**(2):198–207.