Curvature tensor computation by piecewise surface interpolation

Xunnian Yang\textsuperscript{a}, Jianmin Zheng\textsuperscript{b}

\textsuperscript{a}Department of Mathematics, Zhejiang University, Hangzhou, China
\textsuperscript{b}School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore

Abstract

Estimating principal curvatures and principal directions of a smooth surface represented by a triangular mesh is an important step in many CAD or graphics related tasks. This paper presents a new method for curvature tensor estimation on a triangular mesh by replacing flat triangles with triangular parametric patches. An improved local interpolation scheme of cubic triangular Bézier patches to vertices and vertex normals of triangle meshes is developed. Piecewise parametric surfaces that have $C^0$ continuity across boundary curves of adjacent patches and $G^1$ continuity at the joint vertices are obtained by the interpolation scheme. A closed form expression of Taubin integral—a $3 \times 3$ symmetric matrix in integral formulation—is derived based on the piecewise parametric surfaces. Principal curvatures and principal directions are then computed from the Taubin integral. The proposed method does not need to parameterize data points or solve a linear system which is usually required by other surface fitting methods. Compared to several state-of-the-art curvature estimation methods, the proposed method can generate more accurate results for general surface meshes. The experiments have demonstrated its accuracy, robustness and effectiveness.

Keywords: Triangular meshes, curvature tensor, Taubin integral, piecewise surface interpolation, closed form formula

1. Introduction

Principal curvatures (or equivalently mean and Gaussian curvatures) and principal directions are fundamental differential invariants in differential geometry. They are often referred to as the tensor of curvature and can be used to characterize the local shape of a surface. In CAD and computer graphics, curvatures are frequently used for shape processing \cite{10}, segmentation \cite{21} and surface interrogation \cite{31, 26}, and principal direction vectors are essential quantities for surface remeshing \cite{2}, non-photorealistic rendering \cite{18, 19} or texture mapping \cite{15}. In computer vision and medical image analysis, these differential invariants are extensively used for recognition, registration, and free-form shape analysis \cite{22, 33}. Mathematically, the tensor of curvature is defined for twice differentiable surfaces. In practice geometric data are often available as polygonal (typically triangular) meshes. Therefore when we discuss the curvatures on triangular meshes, we mean the curvatures of smooth surfaces that the meshes represent. Since in general we do not have the analytic representation of the underlying smooth surfaces, the true surface curvature information cannot be computed directly. In this paper, we present a novel method for accurately estimating the tensor of curvature of a subadjacent, unknown, smooth surface from a triangular mesh.

1.1. Previous work

Many methods have been developed for estimating curvatures or curvature tensors on triangular meshes accurately or robustly. See for example references \cite{27, 13, 22, 39, 23} and references therein. When both principal curvatures and principal directions are estimated, the known techniques can roughly be classified into three categories:

1) The first category estimates principal curvatures and principal directions by first estimating normal curvatures along a few sampled curves or directions \cite{6, 25, 1, 3}. By using Euler’s theorem and Meusnier’s theorem, a set of equations describing the relation between the principal curvatures and the pre-estimated normal curvatures are built. Then the principal curvatures and directions can be found by solving the equations using a least squares approach. The advantage of approaches of this category is efficiency and simplicity. However, the estimation accuracy is usually not very high and depends heavily on the sampling frequency, mesh resolution and mesh accuracy. The approaches may also suffer from robustness problems due to insufficient sampling rates.

2) The second category is known as local surface fitting methods. The methods compute a simple analytical surface that locally approximates the mesh around a vertex and the principal curvatures and directions can then be computed by applying classic differential geometry methods to the obtained analytical surface \cite{14, 35, 4, 29}. Usually, 1, 2, or even 3-ring neighborhood of the mesh at a vertex is chosen as the local region and a lower order polynomial surface is typically used to fit to the points and/or normals of the chosen neighborhood. When the input data are accurate, local surface fitting often produces good estimation. However, surface fitting methods depend on which neighborhood to fit. This kind of methods may suffer heavy computational cost or numerical stability problem because a linear system has to be solved for each vertex.
3) The third category is discrete methods. Discrete methods attempt to calculate the curvature tensor directly from the local region of a mesh. Rusinkiewicz [30] proposed to compute curvature tensors using finite difference and least squares fitting. Cohen-Steiner and Morvan [7] proposed to compute average curvature tensor based on normal cycle theory. Another famous approach in this category is Taubin’s integral method that computes the curvature tensor via eigen-analysis of a $3 \times 3$ symmetric matrix [34]. The $3 \times 3$ symmetric matrix, which we call Taubin integral, is defined by an integral of a normal-curvature-weighted second order symmetric tangent tensor. Given a triangular mesh, the Taubin integral for each vertex is approximated by a weighted tensor sum over the neighborhood around the vertex. Some variants or extensions have also been proposed to compute the tensor matrix from the input data [17, 36, 20]. In general, discrete methods are very efficient and somewhat robust. However, the accuracy of estimation is adversely affected by obtuse triangles in the mesh and valences other than four or six [13].

The performance of a particular curvature estimation method may be affected by many factors such as the presence of noise, valences of the mesh, mesh resolution, and normal vectors [13, 22]. Nevertheless, the most important concerns about a curvature tensor estimation algorithm are its accuracy, robustness and efficiency. While many techniques have been developed to improve the robustness of curvature tensor estimation against data noise [30, 28, 5], they may generate inaccurate or blurred curvatures for graphics application. In other applications such as visualization of geometric details, edge detection, surface interrogation, etc., the accuracy of curvature tensor estimation plays an important role.

1.2. Our approach

Our work is inspired by Taubin’s integral method. Taubin’s method has a linear complexity, both in time and in space, as a function of the number of vertices of the mesh. All the computations are simple and direct with closed form expressions and the estimation results are reasonably accurate. However, it is noticed that in Taubin’s approach, the calculation of integral is discretized based on the vertex valence and approximated based on the geometry of a triangular mesh. Since the input triangular mesh is only a linear approximation of a subjacent, unknown, smooth surface, we believe that for a triangular mesh with normal vectors at vertices, each triangle can be replaced by a surface patch that can better approximate the underlying surface and the computation based on the appropriately chosen patch can yield more accurate curvature tensor estimation. This motivates us to estimate curvature tensors on triangular meshes by computing continuous Taubin integral and interpolating piecewise smooth surfaces to the meshes.

The main contributions of the paper are:

- An explicit, local scheme is presented for constructing piecewise cubic triangular Bézier patches to interpolate vertices and vertex normals of a triangle mesh. The free coefficients for every Bézier patch are carefully estimated for achieving high accuracy estimation of curvature tensors at vertices.
- A weighted sum of Taubin integrals is proposed, which leads to a robust scheme for curvature tensor estimation on meshes with data noise, and a unified algorithm is proposed to estimate curvature tensors for triangular meshes with accurate or noisy data.

1.3. Overview

Section 2 reviews pertinent background of surface curvatures. Section 3 derives a closed form formula for computing Taubin integral at a point where the surface is formed by joining several parametric patches. Section 4 describes how to construct appropriate parametric surfaces interpolating triangles with given normal vectors at vertices. Section 5 outlines the algorithm steps for computing principal curvatures and principal directions for a given triangular mesh. The experimental examples and comparisons are given in Section 6. Section 7 concludes the paper.

2. Surface curvature review

Curvatures and the tensor of curvature are well studied in classical differential geometry (see for example reference [8]). The curvature tensor is closely related to the surface normal and normal curvature evaluation.

Assume $p$ is a point on a twice differentiable parameterized surface $X(u, v)$. The normal vector of the surface at point $p$ can be obtained by $n = \frac{X_u \times X_v}{\|X_u \times X_v\|}$, where $X_u$ and $X_v$ are the partial derivatives of $X(u, v)$ with respect to $u$ and $v$, respectively. Let $r(s) = X(u(s), v(s))$ be an arc length parameterized curve on the surface, which passes through point $p$. The unit tangent vector of $r(s)$ at point $p$ can be computed by

$$T = \frac{dr(s)}{ds} = X_u \frac{du}{ds} + X_v \frac{dv}{ds}.$$  \hspace{1cm} (1)

From Equation (1), we can obtain the squared length of an infinitesimal arc of curve $r(s)$ near point $p$:

$$ds^2 = Edu^2 + 2Fdudv + Gdvd^2$$

where $E = X_u \cdot X_u$, $F = X_u \cdot X_v$, and $G = X_v \cdot X_v$.

With the surface normal and arc length defined, one may then compute the normal curvature $k_n(T)$ of the surface at point $p$ along direction $T$ as the curvature of a section curve which is the intersection of the original surface $X(u, v)$ and a plane through point $p$ spanned by vectors $n$ and $T$. Let $L = n \cdot X_u$, $M = n \cdot X_v$, and $N = n \cdot X_n$, the normal curvature along direction $T$ is computed by

$$k_n(T) = \frac{Ldu^2 + 2Mdudv + Ndv^2}{ Edu^2 + 2Fdudv + Gdvd^2}. \hspace{1cm} (2)$$
Except for umbilics at which the normal curvatures in all directions are equal, there exist two orthogonal directions \( T_1 \) and \( T_2 \) along which the normal curvatures achieve the maximum value \( k_1 \) and the minimum value \( k_2 \), respectively. These two directions and curvatures are the principal directions and principal curvatures of the surface at the point. They can be computed by Weingarten map in classical differential geometry.

Alternatively, Taubin [34] proposed to compute principal curvatures and principal directions using eigen-analysis of an integral matrix. Let \( T_\phi \) be some unit length tangent vector at \( p \) of the surface where \( \phi \) is the angle between \( T_\phi \) and a fixed direction on the tangent plane. Taubin defined the 3×3 symmetric matrix \( M_p \) by the integral formula of

\[
M_p = \frac{1}{2\pi} \int_{-\pi}^{\pi} k_n(T_\phi)T_\phi T_\phi^T d\phi,
\]

where upper \( t \) means the transpose of a vector. As \( T_\phi \) is perpendicular to the normal vector \( n \), \( n \) is an eigenvector of \( M_p \) associated with the eigenvalue 0. Furthermore, it is proved in [34] that the other two eigenvectors of \( M_p \) are the principal direction vectors \( T_1 \) and \( T_2 \) of the surface at point \( p \) and their corresponding eigenvalues \( m_1 \) and \( m_2 \) are linear combinations of the principal curvatures: \( m_1 = \frac{1}{2} k_1 + \frac{1}{2} k_2 \) and \( m_2 = \frac{1}{2} k_1 + \frac{1}{2} k_2 \). Thus the principal curvatures \( k_1 \) and \( k_2 \) can be computed from the eigenvalues: \( k_1 = 3m_1^2 - m_2^2 \) and \( k_2 = 3m_2^2 - m_1^2 \).

3. Closed form expression for Taubin integral

Note that Taubin integral is given in integral formulation and previous work computes it using discretization and approximation. In this section we derive a closed form formula for the evaluation of Taubin integral \( M_p \) of (3) at a point where several patches with independent parameterization join with a common tangent plane to approximate the subjacent surface. If all the patches are curvature continuous with the subjacent surface at the point, the curvature tensor derived from \( M_p \) is exactly the one of the subjacent surface. Otherwise, \( M_p \) will produce an approximate curvature tensor for the subjacent surface.

![Figure 1: Several patches surround a joint point with a common tangent plane.](image)

Suppose \( n \) parametric surface patches are assembled with position continuity between every two adjacent patches and a common tangent plane at a joint point \( p \), as illustrated in Figure 1. Denote the normal vector to the tangent plane by \( n \) and the tangent directions of the boundary lines of the surface patches at point \( p \) by \( T_i \) (\( i = 0, 1, \ldots, n \)). Assume the angle between vector \( T_i \) and a fixed direction on the tangent plane is \( \theta_i \), and the patch bounded by \( T_{i-1} \) and \( T_i \) is the \( i \)th surface patch \( X_i(u, v) \). Then Taubin integral at \( p \) can be computed by accumulating the integral for each surrounding surface patch:

\[
M_p = \sum_{i=1}^{n} M_p^i = \frac{1}{2\pi} \sum_{i=1}^{n} \int_{\phi_{i-1}}^{\phi_i} k_n(T_\phi)T_\phi T_\phi^T d\phi. \tag{4}
\]

Next we consider how to compute each \( M_p^i \) in Equation (4). We convert the computation from the tangent space to the parameter domain of the surface. Without loss of generality, we assume that the parametric domain of the \( i \)th surface patch \( X_i(u, v) \) is local rectangular at origin and \( p = X_i(0, 0) \). Noting that \( (du, dv) \) only represents a direction on the parametric domain, we can replace it by \( (\cos \theta_i, \sin \theta_i) \) where \( \theta_i \) is the angle between vector \( (du, dv) \) and the \( u \)-axis on the parameter domain. Thus the normal curvature \( k_n(T_\phi) \) at point \( p \) can be computed by

\[
k_n(T_\phi) = \frac{L_i \cos^2 \theta_i + 2M_i \cos \theta_i \sin \theta_i + N_i \sin^2 \theta_i}{E_i \cos^2 \theta_i + 2F_i \cos \theta_i \sin \theta_i + G_i \sin^2 \theta_i}, \tag{5}
\]

where \( E_i = X_{iu} \cdot X_{iu}, F_i = X_{iu} \cdot X_{iv}, G_i = X_{iv} \cdot X_{iv}, L_i = n \cdot X_{iu}, M_i = n \cdot X_{iv}, \) and \( N_i = n \cdot X_{iv} \). For notational simplicity, in the following we may drop subscript "i" within \( L_i, M_i, N_i, E_i, F_i, \) and \( G_i \), which are computed for the \( i \)th surface patch at the joint vertex.

From Equation (1), the matrix of \( T_\phi T_\phi^T \) can be formulated as

\[
T_\phi T_\phi^T = R_1^i \left( \frac{du}{ds} \right)^2 + 2R_2^i \frac{du}{ds} \frac{dv}{ds} + R_3^i \left( \frac{dv}{ds} \right)^2,
\]

where

\[
\begin{align*}
R_1^i &= X_{iu}^u X_{iu}, \\
R_2^i &= \frac{1}{2} (X_{iu}^u X_{iv} + X_{iv}^u X_{iu}), \\
R_3^i &= X_{iv}^u X_{iv}.
\end{align*}
\tag{6}
\]

In a similar way as we deal with the normal curvature, \( T_\phi T_\phi^T \) can be rewritten as

\[
T_\phi T_\phi^T = \frac{R_1^i \cos^2 \theta_i + 2R_2^i \cos \theta_i \sin \theta_i + R_3^i \sin^2 \theta_i}{E \cos^2 \theta + 2F \cos \theta \sin \theta + G \sin^2 \theta}. \tag{7}
\]

As to the angle differential \( d\phi \), it can be expressed by angle \( \theta \) and its differential too. Assume that \( T_{\theta_0} \) and \( T_{\theta_1} \) are two unit vectors on the tangent plane that pass through point \( p \) and the vectors in the parameter domain corresponding to \( T_{\theta_0} \) and \( T_{\theta_1} \) are \( (\cos \theta_0, \sin \theta_0) \) and \( (\cos \theta_1, \sin \theta_1) \), respectively. Then, from Equation (1), we have

\[
T_{\theta_0} = X_{iu} \cos \theta_0 + X_{iv} \sin \theta_0 \frac{d\theta_0}{ds},
\]

\[
T_{\theta_1} = X_{iu} \cos \theta_1 + X_{iv} \sin \theta_1 \frac{d\theta_1}{ds},
\]

where \( d\theta_0 = \left( (E \cos^2 \theta_0 + 2F \cos \theta_0 \sin \theta_0 + G \sin^2 \theta_0)^\frac{1}{2} \right. \) and \( d\theta_1 = \left( (E \cos^2 \theta_1 + 2F \cos \theta_1 \sin \theta_1 + G \sin^2 \theta_1)^\frac{1}{2} \right. \). From the expressions of \( T_{\theta_0} \) and \( T_{\theta_1} \), we have

\[
T_{\theta_0} \times T_{\theta_1} = \frac{\sin(\theta_1 - \theta_0)}{d\theta_0 d\theta_1} (X_{iu} \times X_{iv}).
\]
Without loss of generality, we assume the direction of $T_{\phi_1} \times T_{\phi_2}$ is identical to the surface normal $n$ at point $p$. Then $T_{\phi_1} \times T_{\phi_2} = \sin(\phi_1 - \phi_2)n$. Since $X_u \times X_v = \|X_u \times X_v\|n$, we obtain

$$\sin \Delta \phi = \|X_u \times X_v\| \frac{\sin \Delta \phi}{dx dy d\phi},$$

where $\Delta \phi = \phi_1 - \phi_2$ and $\Delta \theta = \theta_1 - \theta_2$. When $\Delta \phi$ and $\Delta \theta$ are small enough, we have

$$d\phi = \frac{\|X_u \times X_v\|}{E \cos^2 \theta + 2F \cos \theta \sin \theta + G \sin^2 \theta} \, d\theta. \quad (8)$$

Substituting (5), (7), and (8) into $M'_p$ of (4), and letting

$$D_i = \frac{1}{2\pi} \|X_u \times X_v\| = \frac{1}{2\pi} \sqrt{EG - F^2}$$

and

$$K(\theta) = \frac{L \cos^2 \theta + 2M \cos \theta \sin \theta + N \sin^2 \theta}{(E \cos^2 \theta + 2F \cos \theta \sin \theta + G \sin^2 \theta)^2},$$

give

$$M'_p = \frac{1}{2\pi} \int_0^{2\pi} k_\phi(T_\phi)T_\phi T_\phi d\phi = D_i \int_0^{2\pi} K(\theta) \times (\cos^2 \theta R'_1 + 2 \cos \theta \sin \theta R'_2 + \sin^2 \theta R'_3) d\theta = C'_1 R'_1 + C'_2 R'_2 + C'_3 R'_3 \quad (9)$$

where

$$C'_1 = D_i \int_0^{\pi} K(\theta) \cos^2 \theta d\theta,$$
$$C'_2 = D_i \int_0^{\pi} K(\theta) \cos \theta \sin \theta d\theta,$$
$$C'_3 = D_i \int_0^{\pi} K(\theta) \sin^2 \theta d\theta.$$

The above integrals can be computed directly (see Appendix A for detailed derivation). Let $e = \sqrt{E - F^2}$, $f = \sqrt{F - G}$ and $g = \sqrt{G - F^2}$, then the coefficients $C'_j (j = 1, 2, 3)$ are

$$\begin{cases}  
C'_1 &= \frac{a_1+ah_{11}+N_{11}}{2\sqrt{E - F^2}}, \\
C'_2 &= \frac{-a+b+ah_{11}+N_{11}}{2\sqrt{F - G}}, \\
C'_3 &= \frac{a_1+b+ah_{11}+N_{11}}{2\sqrt{G - F^2}} \end{cases} \quad (10)$$

where

$$a = L \cos^2 \theta + 2M \cos \theta \sin \theta + N \sin^2 \theta, \quad b = 2(2M \cos \theta - N \sin \theta), \quad a' = L(\cos^2 \theta - 1) + 2M \cos \theta + N \sin^2 \theta, \quad b' = 2(2Me - Lf), \quad a_{12} = \frac{\pi}{4} \arctan \frac{f}{\sqrt{E - F^2}}, \quad b_{12} = \frac{\pi}{4} \arctan \frac{a - b}{\sqrt{F - G}}, \quad a_{13} = \frac{3}{4} I_2 - \frac{1}{4} I_3, \quad b_{13} = \frac{1}{4} I_2 + \frac{3}{4} I_3.$$

To sum up, Taubin integral can be explicitly computed by

$$M_p = \sum_{i=1}^{n} (C'_1 R'_1 + C'_2 R'_2 + C'_3 R'_3) \quad (11)$$

where $C'_j$ and $R'_j$ are given by (10) and (6), respectively.

### 3.1. Taubin integral for one parametric surface

In a similar way to compute Taubin integral at the joint point of several surface patches, the above derivation can be adapted to derive a closed form expression for Taubin integral of a parametric surface. Now there is only one parametric surface $X(u,v)$. Assume that $p$ is a point on the surface. To compute Taubin integral of the surface at $p$, unlike matrix $M'_p$ in Equation (9) which is an integral on interval $[\phi_{-1}, \phi_1]$, the integral now should be computed on interval $[0, 2\pi]$. Thus

$$M_p^X = \frac{1}{2\pi} \int_0^{2\pi} k_\phi(T_\phi)T_\phi T_\phi d\phi = \frac{\sqrt{E - F^2}}{2\pi} \int_0^{2\pi} K(\theta) \times (\cos^2 \theta R'_1 + 2 \cos \theta \sin \theta R'_2 + \sin^2 \theta R'_3) d\theta = C_1 R'_1 + C_2 R'_2 + C_3 R'_3 \quad (12)$$

where $R_1 = X_uX'_u, R_2 = X_uX'_v + X_vX'_u, R_3 = X_vX'_v$ and

$$C_1 = \frac{3L^2 - 6MF + N(E + G - 2F^2)}{8(E - F^2)},$$
$$C_2 = \frac{-3LF + 2M + 2(EG + 2F^2) - 3NEF}{4(E - F^2)^2},$$
$$C_3 = \frac{L(G + 2F^2) - 6MEF + 3NE^2}{8(E - F^2)^2}.$$

By the integral matrix in Equation (12) and Taubin’s formula for eigen decomposition of the matrix, principal curvatures and principal directions of a parametric surface can be obtained in a new way other than the classical Weingarten transformation.

### 4. Piecewise surface interpolation

Interpolation of a triangular mesh by piecewise polynomial patches has been studied extensively. Walton and Meek [38] and Hahnmann and Bonneau [16] proposed to interpolate meshes by $G^1$ smooth patches. Vlachos et al. [37] used $G^0$ surface patches and continuous normal patches for visually smooth rendering of surfaces. However, these methods may not be applicable for our purpose as they usually need a lot of computational costs or suffer low approximation accuracy for meshes sampled on a known surface. For accurate and efficient curvature tensor estimation purpose, we interpolate surfaces to a triangular mesh satisfying the following requirements:

(a) The normal curvatures of any interpolating surface at mesh vertices should be as accurate as possible.
(b) The interpolating surfaces should not deviate from the triangular mesh too much when the triangle vertices lie on some sharp edges.
(c) Each surface patch should be constructed independently and explicitly based on vertices and vertex normals of a triangle.

We follow the idea of the curved PN triangles method [37] and construct a cubic triangular Bézier patch for each triangle with given corner normals. Cubic triangular Bézier patches are the lowest degree surface patches that permit inflections along boundary curves and interpolate vertices and normals of a given
triangle. While the original curved PN triangles method mainly focuses on contour smoothing of a triangular mesh, it in general does not satisfy requirement (a) stated above. We propose a new cubic Bézier surface interpolation method that satisfies all the three requirements very well. Figure 2 is an example showing different results of piecewise surface interpolation by the curved PN triangles method and our proposed method.

![Figure 2: Interpolation of a mesh by cubic triangular Bézier patches: (a) input triangular mesh; (b) the curved PN triangles method; (c) the proposed method.](image)

A cubic triangular Bézier patch is represented by

\[
X(u, v) = \sum_{i+j+k=3} b_{ijk} \frac{3!}{i! j! k!} u^i v^j (1-u-v)^k
\]

where \( u \geq 0, v \geq 0, 1-u-v \geq 0 \), and \( b_{ijk} \) are the control points (c.f. [11]). The control points are connected to form a control net, as illustrated in Figure 3. Similar to the curved PN triangles method, we classify the control points into three groups:

1. **Vertex control points**: \( b_{003}, b_{100}, b_{030} \)
2. **Boundary control points**: \( b_{102}, b_{210}, b_{021}, b_{102}, b_{201} \)
3. **Center control point**: \( b_{111} \)

Given an input triangle \( \triangle p_1 p_2 p_3 \) together with the unit normals \( n_1, n_2, n_3 \) at three respective corner vertices, we determine the vertex control points first, followed by the boundary control points, and finally the center control point.

- **Vertex control points**. Based on the properties of triangular Bézier patches, the three vertex control points are obtained immediately:
  \[ b_{003} = p_1, \ b_{100} = p_2, \ b_{030} = p_3. \]

- **Boundary control points**. Let us consider boundary control points \( b_{102} \) and \( b_{201} \) corresponding to edge \( p_1 p_2 \). The other boundary control points can be dealt with similarly. The four control points \( b_{003} (= p_1), b_{102}, b_{201}, \) and \( b_{300} (= p_2) \) define a cubic Bézier curve that is one boundary of the triangular Bézier patch \( X(u, v) \). Since we require that a triangular patch be constructed independently from a triangle and two adjacent triangular patches share a common boundary curve, points \( b_{102} \) and \( b_{201} \) should be constructed only based on \( p_1, n_1, \) and \( p_2, n_2 \).

Thus the problem now becomes a point-normal interpolation problem [32]. That is, given two points \( p_1, p_2 \) and two unit normals \( n_1, n_2 \) at these two points, we want to find a spatial cubic Bézier curve that interpolates \( p_1, p_2 \) and is also tangential to their tangent planes at the respective points.

Obviously, point \( b_{102} \) should be on the tangent plane of the triangular patch at \( p_1 \) with normal \( n_1 \), and point \( b_{201} \) should be on the tangent plane at \( p_2 \) with normal \( n_2 \). It is thus suggested to compute \( b_{102} \) and \( b_{201} \) by finding two "good" locations on edge \( p_1 p_2 \) and projecting them onto the two tangent planes, respectively. Specifically, the two locations are represented by \((1-s_1)p_1 + s_1 p_2\) and \(s_2 p_1 + (1-s_2) p_2\) with some parameters \( s_1 \) and \( s_2 \), and then we have

\[
\begin{align*}
\mathbf{b}_{102} &= (1-s_1)\mathbf{p}_1 + s_1 \mathbf{p}_2 - \omega_1 \mathbf{n}_1 \\
\mathbf{b}_{201} &= s_2 \mathbf{p}_1 + (1-s_2) \mathbf{p}_2 - \omega_2 \mathbf{n}_2
\end{align*}
\]

where \( \omega_1 = s_1 (p_2 - p_1) \cdot n_1 \) and \( \omega_2 = s_2 (p_1 - p_2) \cdot n_2 \). In the curved PN triangles method [37], both \( s_1 \) and \( s_2 \) were set to \( \frac{1}{2} \), but this setting cannot make the constructed curve reproduce the curvature of the circular arc even if the boundary data can define a circular arc. In the following we discuss how to appropriately choose \( s_1 \) and \( s_2 \) by taking consideration of inflection case of the curve or the circular arc precision at the ends.

![Figure 4: Computing the control points for a boundary curve.](image)
For two points \( \mathbf{p}_1 \) and \( \mathbf{p}_2 \) with respective unit normal vectors \( \mathbf{n}_1 \) and \( \mathbf{n}_2 \), it is always possible to find a cylinder (not necessarily circular cylinder) that meets the interpolation constraints. In fact, a cylinder can be formed by all the straight lines going through points on \( X(u, 0) \) along a generatrix direction \( \mathbf{n}_0 \). The generatrix direction \( \mathbf{n}_0 \) can be computed by \( \mathbf{n}_0 = \frac{\mathbf{b}_0 - \mathbf{b}_1}{\|\mathbf{b}_0 - \mathbf{b}_1\|} \) when \( \mathbf{n}_1 \times \mathbf{n}_2 \neq 0 \) or \( \mathbf{n}_0 = \frac{\mathbf{b}_0 - \mathbf{p}_2 + \mathbf{p}_1}{\|\mathbf{b}_0 - \mathbf{p}_2 + \mathbf{p}_1\|} \) when \( \mathbf{n}_1 \times \mathbf{n}_2 = 0 \). It can be verified that the normal vector of the cylinder at point \( \mathbf{p}_1 \) is \( \mathbf{n}_1 \).

Since both of the interpolating surface \( X(u, v) \) and the cylinder pass through curve \( X(u, 0) \) and have the same normal vector at \( \mathbf{p}_1 \), the normal curvatures of these two surfaces along the tangent direction of \( X(u, 0) \) at \( \mathbf{p}_1 \) are the same based on Meusnier’s theorem. We determine \( s_1 \) and \( s_2 \) based on the shape or precision of a directrix which lies in a plane with normal \( \mathbf{n}_0 \).

Let \( \mathbf{n}_0 \) be the plane that passes through point \( \mathbf{p}_1 \) with normal \( \mathbf{n}_0 \). After projecting vector \( \mathbf{p}_2 - \mathbf{p}_1 \) onto plane \( \mathbf{n}_0 \), we obtain a unit vector \( \mathbf{v} = \frac{\mathbf{p}_2 - \mathbf{p}_1}{\|\mathbf{p}_2 - \mathbf{p}_1\|} \), where \( \mathbf{v} = (\mathbf{p}_2 - \mathbf{p}_1) \cdot -\mathbf{n}_0 \). With reference to Figure 4, the angle between \( \mathbf{v} \) and the tangent plane at \( \mathbf{p}_1 \) with normal \( \mathbf{n}_1 \) is denoted by \( \alpha \) and the angle between \( \mathbf{v} \) and the tangent plane at \( \mathbf{p}_2 \) with normal \( \mathbf{n}_2 \) by \( \beta \). The signs of \( \alpha \) and \( \beta \) reflect whether the edge \( \mathbf{p}_1 \mathbf{p}_2 \) is lying below or above the tangent plane at \( \mathbf{p}_1 \) and the tangent plane at \( \mathbf{p}_2 \). For example, if edge \( \mathbf{p}_1 \mathbf{p}_2 \) lies below the tangent plane at \( \mathbf{p}_1 \) with normal \( \mathbf{n}_1 \), we have \( \alpha > 0 \); Otherwise, \( \alpha < 0 \). The angles \( \alpha \) and \( \beta \) are computed by

\[
\sin \alpha = -\mathbf{v} \cdot \mathbf{n}_1, \quad \sin \beta = \mathbf{v} \cdot \mathbf{n}_2.
\]

If \( \sin \alpha \sin \beta < 0 \), the projection of a cubic Bézier curve meeting the constraints will have inflections. If the projection curve is inflectional, we choose parameters \( s_1 = s_2 = \frac{1}{2} \); Otherwise, the parameters are chosen to achieve local circular precision at ends. Let \( \eta = \min[|\alpha + \beta|, \pi/3] \), we choose \( s_1 = s_2 = s_0 \), where

\[
s_0 = \frac{1}{3} \left( 1 + \frac{3}{\cos^2 \eta} - 1 \right).
\]

when \( \sin \alpha \sin \beta \geq 0 \) holds. We assume \( |\alpha + \beta|/2 \leq \pi/3 \) for general edges and it is guaranteed \( s_0 < 1 \) when \( \eta \leq \pi/3 \). If \( \alpha \) and \( \beta \) are equal and be less than \( \frac{\pi}{2} \), the curvatures at two ends of the projection curve are equal to the curvature of a circular arc that interpolates the two ends and tangents of the projection curve. If \( \alpha \) or \( \beta \) approaches \( \frac{\pi}{2} \), vertex \( \mathbf{p}_1 \) or \( \mathbf{p}_2 \) is probably lying on a sharp edge or corner, and a bounded value for \( s_1 \) or \( s_2 \) can help to generate an interpolating surface close to the mesh. It also yields large values of curvature at the vertices lying on sharp edges or corners.

We also observe that it might be more appropriate to set \( s_1 \) and/or \( s_2 \) to be negative sometimes. From Equation (14), we can find that the tangent vector of the cubic Bézier curve defined by \( \mathbf{b}_{003}, \mathbf{b}_{012}, \mathbf{b}_{020}, \mathbf{b}_{030} \) and \( \mathbf{b}_{800} \) at \( \mathbf{p}_3 \) is \( 3s_T T_{1proj} \) with \( T_{1proj} = \mathbf{p}_2 - \mathbf{p}_1 - \mathbf{p}_1 - \mathbf{n}_1 \). Let \( \mathbf{n}_f \) and \( \mathbf{n}_i \) be the unit normal vectors of the two triangles sharing edge \( \mathbf{p}_1 \mathbf{p}_2 \). We define the normal vector of the edge to be

\[
\mathbf{n}_v = \frac{\mathbf{n}_f + \mathbf{n}_i}{\|\mathbf{n}_f + \mathbf{n}_i\|}.
\]
edges. If \( D_1H_1 = D_2H_2 = D_3H_3 = 0 \) hold for every interpolating Bézier patch, any two adjacent triangular Bézier patches have the same tangent plane at the midpoint of their common boundary. Since there is only one unknown scalar parameter \( h \) within the three equations, we compute \( h \) by solving equation \( D_1H_1 + D_2H_2 + D_3H_3 = 0 \). With some calculations, we obtain

\[
h = \frac{-2(E_1H_1 + E_2H_2 + E_3H_3)}{n \cdot (H_1 + H_2 + H_3)}
\]

where

\[
E_1 = \frac{b_{201} + 2b_{212} + b_{221}}{4} - \frac{b_{000} + 3b_{210} + 3b_{120} + b_{030}}{8},
\]

\[
E_2 = \frac{b_{102} + 2b_{120} + b_{120}}{4} - \frac{b_{000} + 3b_{201} + 3b_{012} + b_{003}}{8},
\]

\[
E_3 = \frac{b_{212} + 2b_{210} + b_{210}}{4} - \frac{b_{000} + 3b_{012} + 3b_{201} + b_{300}}{8}.
\]

In practice, this solution still makes those interpolating Bézier patches approximately have continuous tangent planes across common boundaries.

5. The algorithm

This section describes our algorithm for curvature tensor estimation on triangular meshes by piecewise surface interpolation. The algorithm estimates the curvature tensors at mesh vertices by the closed form Taubin integral given in Section 3 and the carefully constructed surfaces described in Section 4. This usually gives quite accurate curvature tensors when the sampled points and normals are accurate. However, the estimation may suffer from noise when the mesh vertices or normals are not accurate.

To enhance the robustness of estimation for a mesh containing noise, we propose to compute the curvature tensor at a vertex by incorporating the Taubin integrals at the center of triangular patches incident to the vertex. Specifically, as in Section 3, we assume \( p \) is the joint vertex of surface patches \( X_i(u, v) \) with \( p = X_i(0, 0) \) for \( i = 1, 2, \ldots, n \). Let \( M_p \) be the integral matrix (11) and \( M_p^{sk} \) be the integral matrix (12) of surface \( X_i(u, v) \) at point \( p_i = X_i(u_i, v_i) \) with \( u_i = v_i = 1/3 \). We define a weighted sum of Taubin integrals at vertex \( p \) as

\[
\overline{M}_p = (1 - \rho)M_p + \rho \sum_{i=1}^{n} \omega_i M_p^{sk},
\]

where \( 0 \leq \rho \leq 1 \) is a parameter specified by users, and \( \omega_i \) are the weights satisfying \( \omega_i \geq 0 \) and \( \sum_{i=1}^{n} \omega_i = 1 \). In general, \( \omega_i \) are chosen to be proportional to the areas of triangles incident to the joint vertex. To further improve the accuracy of (17), we follow the technique in [30] to rotate the tangent plane to surface \( X_i(u, v) \) through \( p_i \) and compute updated \( X_u \) and \( X_v \) for matrices \( R_1, R_2 \) and \( R_3 \) in \( M_p^{sk} ).

For a triangular mesh consisting of vertex set \( V = \{ v_i \} \), edge set \( E = \{(v_i, v_j)\} \) and face set \( F = \{(v_i, v_j, v_k)\} \), and a user specified parameter \( \rho \), we estimate curvature tensors at mesh vertices as follows:

1. for each vertex \( v_i \), compute unit normal vector \( n_i \) at the vertex as a weighted sum of normals to the surrounding facets with weights given in [24];
2. for each edge \( (v_i, v_j) \) with unit normal vectors at two end vertices, construct a cubic curve by Equations (14) and (15);
3. for each triangle \( (v_i, v_j, v_k) \) construct a cubic interpolating triangular Bézier patch by copying the control points of three cubic curves interpolating the edges and computing the center control point by Equation (16);
4. for each vertex \( v_i \),
   - if \( \rho < 1 \), compute the sum of Taubin integrals of surrounding Bézier patches at the vertex using Equations (9) and (11);
   - if \( \rho > 0 \), compute Taubin integrals at centers of surrounding Bézier patches using Equation (12) and a weighted sum of Taubin integrals using Equation (17);
   - compute the principal curvatures and principal directions from Taubin integral using closed form expressions, just as Taubin’s approach [34].

Note that the combination of the closed form Taubin integral and the curved PN triangles can also estimate curvature...
tensors for triangular meshes. Figure 6(a) shows a triangulated torus model with 1800 randomly sampled points and exact curvature tensors at the points. When we interpolate the torus using PN triangles and compute curvature tensors for every vertex by computing Taubin integral, the average and maximum errors of mean curvatures are 0.099527 and 0.464881, respectively; see Figure 6(b) for the plot of mean curvature errors. The mean and maximum deviations of principal directions by PN triangles method are 0.749599 and 8.909531 degrees, respectively. By our proposed surface interpolation scheme, the average and maximum errors of mean curvatures reduce to 0.008991 and 0.083860 while the mean and maximum deviations of principal directions become 0.221218 and 1.741704 degrees, respectively; see Figure 6(c) for the plot of mean curvature errors using this new surface interpolation scheme.

6. Experimental results

This section evaluates our proposed algorithm with several models. We first test the accuracy of the proposed curvature tensor estimation method using two models generated from two analytical surfaces. Then we apply the proposed method to a subdivision surface and a denoised mesh to test its use in checking the smoothness of discrete models. Finally we apply the proposed method for visualizing the detail or estimating smooth curvatures of a real scan-reconstructed mesh.

We compare the proposed method with several state-of-the-art methods which use continuous surface patches or discrete techniques for curvature tensor estimation in CAD or graphics. The vertex normal (VN) triangle based method [23] is the latest one that computes curvatures using curved triangles. The cubic fitting algorithm [14] is a representative for analytic fitting methods, which can achieve very high accuracy. The discrete methods which are widely used in graphics include Taubin’s method [34], the finite difference method proposed by Rusinkiewicz [30] and the tensor averaging method [7]. They are very competitive in speed for curvature tensor estimation. All these algorithms were implemented using C++ on a PC with Intel(R) Core(TM)2 CPU, T9900@3.06GHz 3.07GHz and 4G RAM.

The two analytical surfaces are defined as follows. The first surface is a ring-shaped surface:

\[
\begin{align*}
    x(u, v) &= (1.5 + 0.3 \cos(v)) \cos(u) \\
    y(u, v) &= (1.5 + 0.3 \cos(v)) \sin(u) \\
    z(u, v) &= 0.6 \sin(v)
\end{align*}
\]

where \((u, v) \in [0, 2\pi] \times [0, 2\pi]\). The ring-shaped surface is generated by sweeping an ellipse along a circle. The second surface is a surface patch which was introduced by Goldfeather and Interrante in [14]. The equations of G&I surface are

\[
\begin{align*}
    x(u, v) &= f(u) \cos(v) \\
    y(u, v) &= f(u) \sin(v) \\
    z(u, v) &= u + 0.2 \sin(2x) + 0.15 \cos(3xy)
\end{align*}
\]

where \(f(u) = -2u^4 + 2u^2 + u/6 + 0.3\) and \((u, v) \in [-0.9, 1] \times [0, 2\pi]\). These two surfaces serve as the ground truth. Both are free of umbilics and thus the principal curvature and principal directions can be computed accurately for each point on the surfaces. We tessellate each surface into a low resolution mesh and a high resolution mesh with different numbers of sampled points on the surface. Meanwhile, the exact curvature tensors of the surface at all sampled points are evaluated for comparison. Figure 7(a) illustrates two meshes tessellated from the ring-shaped surface, and the two meshes have 1800 or 7200 vertices, respectively. Figure 7(b) illustrates the two meshes of G&I surface which have 2550 and 10100 vertices, respectively.

We test the accuracy of our proposed algorithm and other five methods using low resolution tessellated meshes and high resolution tessellated meshes shown in Figure 7 with exact sampled points and normals. To test the robustness of these methods, we add noise to the four meshes by moving each vertex along its normal direction by a random distance within 2% and 5% of the mean edge length. The vertex normals for the noisy meshes are then computed as the weighted sum of the 1-ring facet normals using the method presented in [24]. The accuracy of the estimated curvature tensors by each method is measured by the root mean square (RMS) error, which is the square root of the average of square errors of all major and minor curvatures against the ground truth.

From Table 1 and Table 2 we can see that our proposed method with \(\rho = 0\) can achieve the highest accuracy for the
Table 1: RMS errors of the estimated curvatures for the ring model

<table>
<thead>
<tr>
<th>Method</th>
<th>ring (LR) (#vertices=1800)</th>
<th>ring (HR) (#vertices=7200)</th>
<th>ring (HR) (2% noise)</th>
<th>ring (HR) (5% noise)</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposed ($\rho = 0$)</td>
<td>0.044276</td>
<td>0.017946</td>
<td>1.241357</td>
<td>3.127922</td>
</tr>
<tr>
<td>proposed ($\rho = 1$)</td>
<td>0.159244</td>
<td>0.069801</td>
<td>0.239621</td>
<td>0.536465</td>
</tr>
<tr>
<td>VN triangle</td>
<td>0.172309</td>
<td>0.062181</td>
<td>0.245828</td>
<td>0.575565</td>
</tr>
<tr>
<td>cubic fitting</td>
<td>0.135025</td>
<td>0.031862</td>
<td>0.222330</td>
<td>0.371346</td>
</tr>
<tr>
<td>finite difference</td>
<td>0.162506</td>
<td>0.070257</td>
<td>0.165780</td>
<td>0.308891</td>
</tr>
<tr>
<td>Tensor averaging</td>
<td>0.759555</td>
<td>0.748065</td>
<td>0.823753</td>
<td>1.163509</td>
</tr>
<tr>
<td>Taubin’s method</td>
<td>0.997030</td>
<td>0.827438</td>
<td>0.928336</td>
<td>1.307278</td>
</tr>
</tbody>
</table>

Table 2: RMS errors of the estimated curvatures for the G&I surface

<table>
<thead>
<tr>
<th>Method</th>
<th>G&amp;I (LR) (#vertices=2550)</th>
<th>G&amp;I (HR) (#vertices=10100)</th>
<th>G&amp;I (HR) (2% noise)</th>
<th>G&amp;I (HR) (5% noise)</th>
</tr>
</thead>
<tbody>
<tr>
<td>proposed ($\rho = 0$)</td>
<td>0.064131</td>
<td>0.025054</td>
<td>2.327218</td>
<td>6.206422</td>
</tr>
<tr>
<td>proposed ($\rho = 1$)</td>
<td>0.189813</td>
<td>0.080634</td>
<td>0.349591</td>
<td>0.875071</td>
</tr>
<tr>
<td>VN triangle</td>
<td>0.224454</td>
<td>0.082247</td>
<td>0.350479</td>
<td>1.067194</td>
</tr>
<tr>
<td>cubic fitting</td>
<td>0.122904</td>
<td>0.029810</td>
<td>0.231335</td>
<td>0.535465</td>
</tr>
<tr>
<td>finite difference</td>
<td>0.178445</td>
<td>0.074829</td>
<td>0.228927</td>
<td>0.513157</td>
</tr>
<tr>
<td>Tensor averaging</td>
<td>0.907732</td>
<td>0.901229</td>
<td>1.064728</td>
<td>1.742106</td>
</tr>
<tr>
<td>Taubin’s method</td>
<td>0.864383</td>
<td>0.882754</td>
<td>1.193384</td>
<td>2.277044</td>
</tr>
</tbody>
</table>

two surfaces among all the six methods. Moreover, similar to the cubic fitting method, the finite difference method and the VN triangle method, our method with either $\rho = 0$ or $\rho = 1$ can raise the accuracy when the mesh resolution increases. However, our method with $\rho = 0$ is sensitive to noise. This is not surprising because a surface mesh with noisy vertices or disturbed normals may represent a different underlying surface. An accurate method should then be sensitive to the change of the vertices or the normals of the mesh. On the other hand, the cubic fitting method, the finite difference method and our method with $\rho = 1$ are robust against noise due to the use of the least squares fitting or averaging strategy though the accuracy of the results may not be the highest.

We next show the use of the proposed method for smoothness check of surfaces. We apply the curvature tensor estimation methods to a Doo-Sabin subdivision mesh (see Figure 8(a)). An initial polygonal mesh is subdivided six times by Doo-Sabin subdivision, and then each face in the refined mesh is split into triangles such that the vertices have random valences. Note that this triangulated mesh is very close to the limit surface of Doo-Sabin subdivision though the vertices of the mesh do not necessarily lie on the limit surface. Thus accurately estimated curvature tensors of the triangulated mesh are assumed to have similar behavior as those of the limit surface. A Doo-Sabin subdivision surface is a generalized biquadratic B-spline surface, and it is composed of a set of parametric surface patches which have tangent plane continuity across the joint edges or the extraordinary points [9]. Figures 9(a) and (b) display the mean curvatures and the Gaussian curvatures estimated by our proposed method with $\rho = 0$, respectively. In these two figures, the joint lines and individual surface patches are clearly visible. Figure 9(h) shows that the Gaussian curvatures estimated by Taubin’s method are also sensitive to the boundaries of surface patches. However, the accuracy of Taubin’s method is influenced by vertex valences. Figures 9(d), (e), (f), (g) and (c) show the plots of Gaussian curvatures estimated by the VN triangle method, the cubic fitting method, the finite difference method, the tensor averaging method and our method with $\rho = 1$, where the curvatures around the patch edges are seen to be blurred due to the process of the least squares fitting or averaging.

Figure 8(b) is another visually smooth surface mesh obtained by bilateral mesh denoising [12]. To check the smoothness of the denoised mesh, we compute the mean curvature and Gaussian curvature of the mesh by our proposed method with $\rho = 0$. From the curvature plots in Figures 10(a) and (b) we learn that the mesh is not as smooth as it looks. It still contains some
stairs or artifacts even after bilateral denoising. The curvatures estimated by the proposed method with $\rho = 1$, the VN triangle method, the cubic fitting method, the finite difference method all look smooth; see Figures 10(c), (d), (e) and (f), respectively. These methods have hidden the artifacts of the surface mesh. The plots of the mean curvatures estimated by the tensor averaging approach and Taubin’s method are given in Figures 10(g) and (h), which illustrate some curvature discontinuity of the denoised surface. However, the artifacts are not clear enough due to the limited accuracy of curvature estimation by these methods.

Our proposed curvature tensor estimation algorithm provides a tradeoff between accuracy and robustness, which is controlled by parameter $\rho$. A small value of $\rho$ yields accurate but noise sensitive curvature tensors while a large value of $\rho$ helps to estimate curvature tensors robustly against noise. Figure 11(a) illustrates a triangular mesh reconstructed from real scanned data. From the figure we can see that the original surface contains low magnitude noise. The accurate mean curvature plot in Figure 11(b) corresponding to $\rho = 0$ clearly depicts the noise distribution on the surface. When the parameter value is changed to $\rho = 0.5$ and $\rho = 1.0$, the curvature plots become smoother and smoother; see Figures 11(c) and (d) for the results. It is also observed that the principal directions become smoother when the value of parameter $\rho$ increases.

Finally, we compare the computational efficiency of our proposed algorithm and all other methods by reporting the running time for all the test models in Table 3. From the table, it can be seen that Taubin’s method is the fastest and the cubic fitting algorithm is the slowest. Our proposed method has similar time cost as the finite difference algorithm. Due to its high accuracy and considerable efficiency, our proposed method can be used for curvature tensor computation for triangular meshes that have a large number of vertices.
7. Conclusions and discussion

This paper has derived closed form Taubin integral for piecewise smooth surfaces or a single parametric surface, which leads to a new way for curvature tensor estimation for triangular meshes. The paper has also presented an improved local surface interpolation scheme for triangular meshes. Compared to other surface fitting methods, the proposed surface interpolation scheme does not need to solve any linear systems. Our proposed curvature tensor estimation method usually generates more accurate results for general surface meshes than the state-of-the-art methods. Also by using weighted sums of Taubin integrals, the proposed algorithm can robustly estimate curvature tensors for noisy meshes. The tradeoff between accuracy and robustness of curvature tensor estimation is controlled using a single parameter.

Accurate curvature tensor estimation can be applied when the input vertices or normals are accurate or have low magnitude noise. Our proposed method can be used to visualize the joint lines between surface patches or the distribution of noise. Although the paper only discusses triangular meshes, the extension of the proposed method to general polygonal meshes is straightforward. One can replace piecewise triangular Bézier surface interpolation to other types of surface interpolation, and the curvature tensors at selected points can be computed by the closed form Taubin integral and the consequent eigen-analysis of the integral matrix.

Acknowledgment

We thank the editor and anonymous reviewers for their helpful comments. This work is supported by NNSF of China grants (60970077, 11290142) and Multi-plAtform Game Innovation Centre (MAGIC) in Nanyang Technological University. MAGIC is funded by the Interactive Digital Media Programme Office (IDMPO) hosted by the Media Development Authority of Singapore.
We then rewrite

\[ \rho = \sqrt{0} \]

can be further expressed as

\[ 1 = \frac{\rho^2}{\sqrt{E^2 - F^2}} \]

To compute \( C_1^i \), we use substitution \( t = \tan \theta \) and obtain

\[
\begin{align*}
C_1^i &= D_i \int_0^{\pi/2} L \cos^2 \theta + M \sin 2\theta + N \sin^2 \theta \cos^2 \theta d\theta \\
&= D_i \int_0^{\pi/2} \frac{L + 2M \tan \theta + N \tan^2 \theta}{E + 2F \tan \theta + G \tan^2 \theta} d\theta \\
&= D_i \int_0^{\infty} \frac{L + 2M t + N t^2}{E + 2F t + G t^2} dt.
\end{align*}
\]

We then rewrite \( E + 2F \tan \theta + G \tan^2 \theta \) as

\[
\frac{E}{\sqrt{E^2 - F^2}} = \frac{E \sqrt{E^2 - F^2}}{E^2 - F^2} = \frac{1}{1 + \frac{G}{E} t + \frac{F}{\sqrt{E^2 - F^2}} t^2} = \frac{1}{1 + \frac{G}{E} t + f^2 t^2}.
\]

By using substitution \( \tau = gt + f \), \( C_1^i \) can be further expressed as

\[
C_1^i = \frac{D_i G^3}{G^3} \int_f^\infty \frac{L g^2 - 2M f g + N(f^2 - 1)}{(1 + f^2 t)^3} dt.
\]

Let \( I_n = \int_f^\infty \frac{1}{(1 + f^2 t)^n} dt \) and \( J_n = \int_f^\infty \frac{t}{(1 + f^2 t)^n} dt \). \( I_n \) and \( J_n \) can be computed recursively or explicitly: \( I_1 = \frac{\pi}{2} - \arctan f \), \( I_{n+1} = \frac{2n-1}{2n} I_n - \frac{1}{2n(1 + f^2)} f \) for \( n > 0 \), and \( J_n = \frac{1}{2(n-1)(1 + f^2)^n} \) for \( n > 1 \). Thus after simplification, the formula for \( C_1^i \) given in Section 3 can be obtained.

In a similar way, the formula for \( C_2^i \) can be derived with the substitution \( t = \tan \theta \).
For $C_i^j$, we replace $\theta$ by $\frac{\pi}{2} - \varphi$. Then

\[
C_i^j = D_i \int_0^{\frac{\pi}{2}} \frac{L \cos^2 \theta + M \sin 2\theta + N \sin^2 \theta}{(E \cos^2 \theta + F \sin 2\theta + G \sin^2 \theta)^\frac{3}{2}} \sin^2 \theta d\theta = D_i \int_0^{\frac{\pi}{2}} \frac{N \cos^2 \varphi + M \sin 2\varphi + L \sin^2 \varphi}{(G \cos^2 \varphi + F \sin 2\varphi + E \sin^2 \varphi)^\frac{3}{2}} \cos^2 \varphi d\varphi.
\]

Therefore the formula for $C_i^j$ can be obtained from the one for $C_i^1$ by interchanging $L$ and $N$, $E$ and $G$.

References


