International Journal of the Physical Sciences Vol. 7(32), pp. 5248-5255, 23 August, 2012
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS12.189
ISSN 1992 - 1950 ©2012 Academic Journals

Full Length Research Paper

Incremental Fourier transform of triangular closed 2-manifolds

Mohamed H. MOUSA* and Mohamed K. HUSSEIN

Faculty of Computers and Informatics, Suez Canal University, Egypt.

Accepted 28 May, 2012

In this paper we present a technique for the calculation of the Fourier transform for functions defined on oriented closed 2-manifolds. The objects are given as oriented triangular meshes. Our focus in this paper is on the characteristic function of the model, that is, the function that is equal to one inside the model and zero outside. The advantage of our approach is that it provides an automatic, simple, and efficient method for computing the Fourier coefficients directly from the mesh representation. This avoids the approximation of the mesh by a grid of voxels which leads to a loss of details and error prone in calculation. The main idea is to distribute the calculation of the Fourier coefficients over the elementary shapes composing the mesh. Then we use the divergence theorem to simplify the computation by calculating the coefficients using integrations on simpler domains. The algorithm is simple and efficient, with many potential applications. Some examples are given to demonstrate the effectiveness of our approach.

Key words: Fourier coefficients, partial calculation, voxelization, triangular mesh, divergence theorem.

INTRODUCTION

3D scanning technology easily produces computer representations from real objects. The main purpose of a 3D scanner is usually to create point cloud of geometric samples on the surface of the subject. These points can then be used to extrapolate the shape of the subject (a process called surface reconstruction). Therefore, 3D object browsing is becoming more and more popular as it engages people with much richer experience than 2D images. The most common representation for these 3D objects is the polygonal mesh or triangular mesh. A triangle mesh is a type of polygon mesh in computer graphics. It comprises a set of triangles (typically in three dimensions) that are connected by their common edges or corners. Various methods of storing and working with a mesh in computer memory are possible. For example, VRML (Carey, 1998) file model, which is the surface of a virtual object or environment with a collection of 3D geometrical entities, such as vertices and polygons.

However, the acquired geometry often presents some noise that needs to be filtered out. More generally, it may be suitable to enhance some details while removing other ones, depending on their sizes (spatial frequencies). Taubin (1995) showed that the techniques of the classic spectral analysis can be successfully applied to geometry processing using the Fourier function basis. This Fourier function basis enables a given signal to be decomposed into a sum of sine waves of increasing frequencies. He used this analogy as a theoretical tool to design and analyze approximations of low-pass filters. Several variants of this approach were then suggested, as discussed later.

Intuitively, we may calculate the Fourier transform of a 3D object by first transforming the object into its volumetric representation and then voxelize this representation in a regular grid. Finally, we evaluate the required transform in the voxel space using one of the classical discrete Fourier transform algorithms (Tolimieri, et al., 1997; Frigo and Johnson, 2005). However, transforming a 3D model into its voxelized version allows finer details on the object to be lost. To minimize the loss of these finer details, the voxel grid dimension should

*Corresponding author. E-mail: mohamed_mousa@ci.suez.edu.eg.

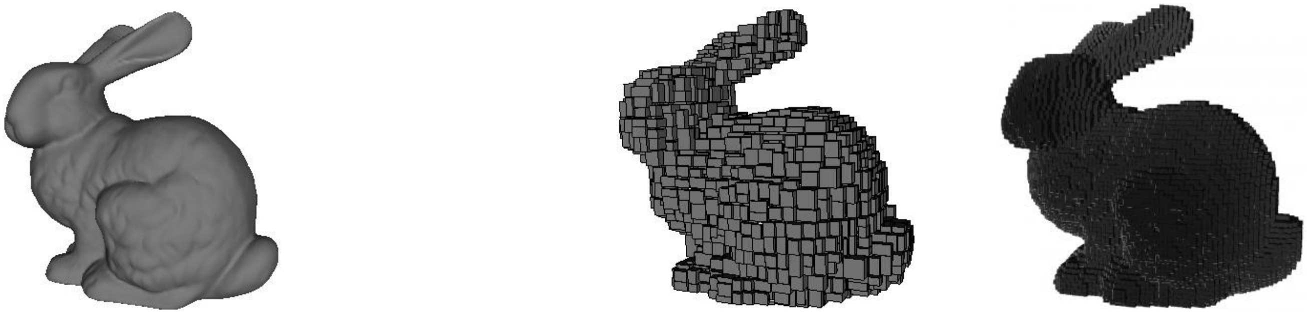

Figure 1. An example of object voxelization. The original object (left) loses some details according to the dimension of the voxelization (middle and right).

be large enough Vacavant and Coeurjolly, 2009).

Related work

Mathematically, the Fourier exponentials are the eigen functions of the Laplacian operator applied on the rectangular domains (Zhang and Kaick, 2007). The discretization of the Laplacian operator plays a central role in geometry processing and has been extensively studied (Wardetzky and Mathur, 2008), motivated by the large number of its applications, such as parameterization, surface remeshing, compression, reconstruction and minimal surfaces.

The generation of the Laplacian depends only on the connectivity of the mesh. Spectral analysis of the Laplacian graph was first used by Taubin (1995) to approximate low pass filters. A basis of eigenfunctions of the Laplacian graph was used by Karni and Gotsman (2000) for geometry compression. Zhang (2004) studied several variants of the combinatorial Laplacian and its properties for spectral geometry processing and JPEG-like mesh compression. However, as pointed-out by Meyer and Desbrun (2003), the analogy between the Laplacian graph and the discrete cosine transform supposes a uniform sampling of the mesh. Moreover, different embeddings of the same graph yield the same eigenfunctions, and two different meshings of the same object yield different eigenfunctions.

More geometry can be injected into the definition of a discrete Laplacian through the ubiquitous cotan weights (Pinkall and Polthier, 1993; Meyer and Desbrun, 2003). These weights can also be derived from finite element modeling (FEM) as done by Wardetzky and Bergou (2007), and they converge to the continuous Laplacian under certain conditions as explained by Hildebrandt and Polthier (2006) and Arnold and Falk (2006).

To directly implement the spectral transform on manifolds, several methods consist in putting the input surface in one-to-one correspondence with a simpler domain or to partition it into a set of simpler domains (Lee and Sweldens, 1998; Pauly and Gross, 2001) on which it is easier to define a frequency space. Note that these methods generally need to resample the geometry. It is also possible to extract the frequencies from a progressive mesh (Lee and Sweldens, 1998) and avoid resampling the geometry by using irregular subdivision (Guskov and Sweldens, 1999).

Contribution

The main contribution of this paper is to use an efficient numerical mechanism to compute the Fourier coefficients of the indicator function representing the solid object. We calculate the Fourier coefficients from the mesh representation directly without any prior voxelization. Based on the linearity of the Fourier integrations, we distribute the calculation of the frequency coefficients over the elementary shapes composing the mesh, such as tetrahedrons, and then summing them up. The presented formula to calculate the coefficients values are mathematically exact, that is having an approximation error of zero. The proposed mechanism overcomes the current limits of calculating limited bandwidths. Moreover, the computational complexity is proportional to the number of elementary shapes, which is typically much smaller than the number of voxels in the equivalent volumetric representation. The result is general and has many potential applications.

The rest of the paper is organized as follows: The decomposition of the triangular mesh into a set of signed tetrahedrons is presented and also the application of the divergence theorem to distribute the calculation of the Fourier coefficients over the elementary tetrahedrons is discussed. Furthermore, some examples of applications that demonstrate the effectiveness of our approach are presented and then conclusion.

Volumetric decomposition

The mesh is represented by a set of vertices and polygons. Before we decompose the volume, we do

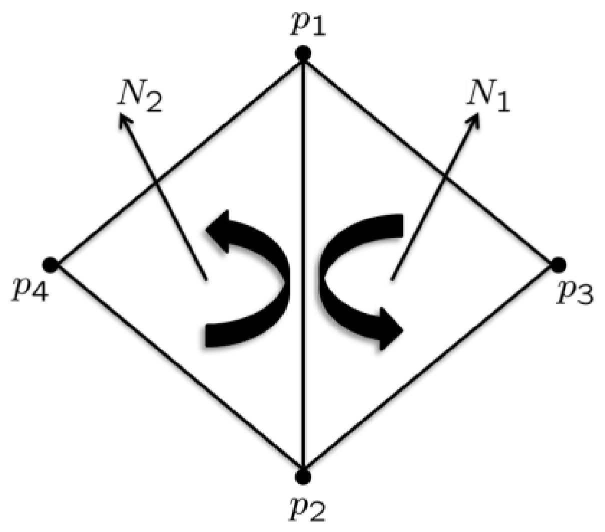

Figure 2. Normals and order of vertices around the faces. The shared edge p_1p_2 has different directions in each face.

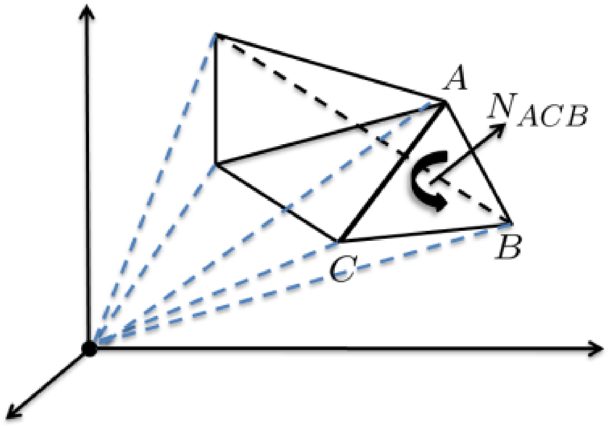

Figure 3. The decomposition of the 3D volume into a set of signed sub-volumes. Each triangle is connected to the origin to form a tetrahedron.

some preprocessing on the model and make sure that all the polygons are triangles. Such preprocessing is called triangulation and is commonly used in mesh coding, mesh signal processing, and mesh editing. The order of the vertices around the triangles should be consistent. That is, for two neighboring triangles, if the common edge has different directions, then the normals of the two triangles are consistent. The direction of the normal for a triangle can be determined by the order of the vertices and the right-hand rule, as shown in figure 2.

The consistency condition is very easy to satisfy. In Figure 2, p_1p_2 is the common edge of the triangles$p_1p_2p_3$ and $p_1p_4p_2$. In the triangle $p_1p_2p_3$, the direction of the shared edge is from p_1 to p_2, and in the triangle $p_1p_4p_2$, the direction is from p_2 to p_1, thus N_1 and N_2 are consistent.

In our case, the elementary decomposition unit is a tetrahedron. For each triangle, we connect each of its vertices with the origin and form a tetrahedron, as shown in figure 3.

We define the signed volume for each elementary tetrahedron as: The magnitude of its value is the volume of the tetrahedron, and the sign of the value is determined by checking if the origin is at the same side as the normal with respect to the triangle. In Figure 3, triangle ACB has a normal $\boldsymbol{N_{ACB}}$. As the origin O is at the opposite side of $\boldsymbol{N}_{ACB}$, the sign of this tetrahedron is positive. The sign can also be calculated by inner product $\overrightarrow{OA} \cdot N_{ACB}$.

Given a triangular mesh M, the enclosed volume V of this mesh is given by the following expression:

$$V = \sum_{t \in T} s_t V_t \qquad (1)$$

where T is the set of triangles of M, s_t is the sign of the tetrahedron corresponding to t and V_t is its volume.

The previous volumetric decomposition is valid for any mesh whether its shape is convex or concave. Moreover, the adoption of the signed volume calculation makes Equation (1) independent of the choice of the origin. That is, if we choose any point $p \in \mathbf{R}^3$ instead of the origin, Equation (1) is still valid.

Partial Fourier coefficients

The Fourier transform is a mathematical operation with many applications in physics and engineering that expresses a mathematical function of time as a function of frequency, known as its frequency spectrum. Given a solid model, we are interested in the characteristic function of the given model. This is a function defined in 3D whose value is equal to one inside the solid and zero outside. Our goal is to compute the Fourier coefficients of this characteristic function. Specifically, if M is a solid model and χ_M is its characteristic function, we would like to compute the coefficients.

$$\begin{aligned}\hat{\chi}_M(\omega) &= \int_{\mathbb{R}^3} \chi_M(x,y,z) e^{-2\pi i p .\omega} dp \\ &= \int_{V_M} e^{-2\pi i p .\omega} dp \end{aligned} \qquad (2)$$

where $p = (x,y,z)$ and $\omega = (l,m,n)$, which are three dimensional vectors. Note that since the function χ_M is

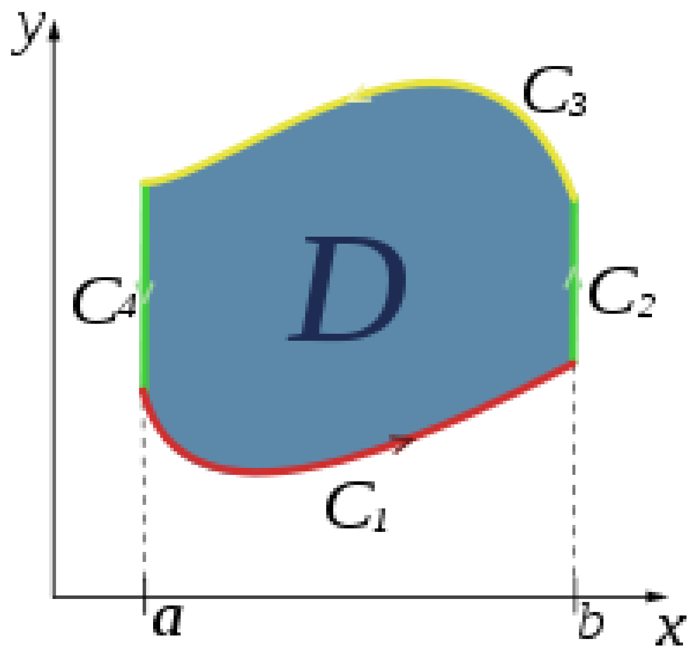

Figure 4. A normal integration manifold for the special case $n = 2$.

equal to one inside the model and zero outside, integrating the complex exponentials against the characteristic function is equivalent to computing the integral of these functions over the volume of the solid model. Putting the previous note in combination with Equation (1):

$$\hat{\chi}_M(\omega) = \int_{V_M} e^{-2\pi i p.\omega} dp$$
$$= \sum_{t\in T} s_t \int_{V_t} e^{-2\pi i p.\omega} dp \qquad (3)$$

Note that the last integration of Equation (3) is equivalent to:

$$\int_{V_t} e^{-2\pi i p.\omega} dp = \int_{\mathbb{R}^3} \chi_{V_t} e^{-2\pi i p.\omega} dp$$
$$= \hat{\chi}_{V_t}(\omega) \qquad (4)$$

where χ_{V_t} is the partial indicator function defined on the tetrahedron V_t. and $\hat{\chi}_{V_t}(\omega)$ is its corresponding partial Fourier coefficient. Therefore, Equation (3) becomes:

$$\hat{\chi}_M(\omega) = \sum_{t\in T} s_t \, \hat{\chi}_{V_t}(\omega) \qquad (5)$$

That is to calculate the Fourier coefficient $\hat{\chi}_M(\omega)$ it is sufficient to calculate the sum of the corresponding partial coefficients $\hat{\chi}_{V_t}(\omega)$ defined over the tetrahedrons V_t. Now we will state the Stokes' theorem and show how to use this theorem to simplify the calculation of the partial Fourier coefficients of $\hat{\chi}_{V_t}(\omega)$.

Divergence theorem

In differential geometry, Stokes' theorem (also called the generalized Stokes' theorem) is a statement about the integration of differential forms on manifolds, which simplifies and generalizes several theorems from vector calculus. It provides a method for expressing the integral of a function over the interior of a volume as an integral over the volume's boundary. The general form of the Stokes' theorem reads: If ω is an $(n-1)$- form with compact support on Ω and $\partial\Omega$ denotes the boundary of Ω with its induced orientation, then:

$$\int_{\Omega} d\omega = \oint_{\partial\Omega} \omega \qquad (6)$$

Here d is the exterior derivative, which is defined using the manifold structure only. On the right hand side of Equation, a circle is used within the integral sign to stress the fact that the $(n-1)$-manifold is closed. The theorem is often used in situations where Ω is an embedded oriented sub-manifold of some bigger manifold on which the form ω is defined (Figure 4).

In this paper, we consider a specific instance of Stokes' theorem known as the divergence theorem or Gauss's theorem. Specifically, if $M \subset \mathbf{R}^3$ is a three dimensional solid and $\vec{F} = (F_x, F_y, F_z): \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ is a vector-valued function, the divergence theorem expresses the volume integral as a surface integral as follows:

$$\int_M \nabla \cdot \vec{F}(p) dv = \int_{\partial M} \langle \vec{F}(p), \vec{n}(p) \rangle da \qquad (7)$$

Where $\nabla \cdot \vec{F} = \left(\frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \frac{\partial}{\partial z}\right) \cdot \vec{F}$ is the divergence of $\vec{F}$ and $\vec{n}(p)$ is the surface normal at the point p.

Fourier transform decomposition

Our objective in this section is to simplify the integration of $\hat{\chi}_{V_t}(\omega) = \int_{V_t} e^{-2\pi i p.\omega} dp$ into a surface integration over the boundary of the tetrahedron V_t. Using the divergence theorem, we know that if $\vec{F}_\omega : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ is a vector-valued function such that:

$$(\nabla \cdot \vec{F}_\omega)(p) = e^{-2\pi i p . \omega} \tag{8}$$

The volume integral can be expressed as the surface integral as follows:

$$\hat{\chi}_{V_t}(\omega) = \int_{p \in V_t} e^{-2\pi i p . \omega} dp$$

$$= \int_{\partial V_t} \langle \vec{F}(p), \vec{n}(p) \rangle da \tag{7}$$

where $\vec{n}(p)$ is the unit normal at p around the boundary ∂V_t.

In order to be able to evaluate the above summation explicitly, we need to choose a vector-valued function $\vec{F}_\omega$ whose divergences are equal to the complex exponentials $e^{-2\pi i p . \omega}$. In this paper, we use the function $\vec{F}_\omega$ that does not depend on the alignment of the coordinate axis:

$$\vec{F}_\omega(x, y, z) = \frac{1}{2\pi} \begin{pmatrix} \frac{il}{||\omega||^2} e^{-2\pi i p . \omega} \\ \frac{im}{||\omega||^2} e^{-2\pi i p . \omega} \\ \frac{in}{||\omega||^2} e^{-2\pi i p . \omega} \end{pmatrix} \tag{10}$$

Partial coefficient evaluation

Each tetrahedron V_t consists of four vertices; the origin $v_0 = (0,0,0)$ and the three vertices of the triangle $t;\ v_1 = (x_1, y_1, z_1),\quad v_2 = (x_2, y_2, z_2)$ and $v_3 = (x_3, y_3, z_3)$. These four points are sufficient to calculate the magnitudes and the directions of the four normals; $\vec{\mathbf{n}}_{012}$, $\vec{\mathbf{n}}_{023}$, $\vec{\mathbf{n}}_{031}$ and $\vec{\mathbf{n}}_{123}$ around the four triangular faces of the tetrahedron V_t. For example, the normal $\vec{\mathbf{n}}_{012}$ can be calculated by $\frac{\overrightarrow{v_0 v_1} \times \overrightarrow{v_0 v_2}}{|\overrightarrow{v_0 v_1} \times \overrightarrow{v_0 v_2}|}$. Using the divergence theorem and the proposed $\vec{F}_\omega$, the partial coefficient $\hat{\chi}_{V_t}(\omega)$ expressed in Equation (4) is simplified to:

$$\hat{\chi}_{V_t}(\omega) = \int_{\partial V_t} \langle \vec{F}_\omega(p), \vec{n}(p) \rangle da \tag{11}$$

Since the tetrahedron V_t.has four triangular faces, we get the following sub-integrals:

$$\hat{\chi}_{V_t}(\omega) = \int \vec{F}_\omega \cdot \vec{\mathrm{n}}_{012} da + \int \vec{F}_\omega \cdot \vec{\mathrm{n}}_{023} da + \int \vec{F}_\omega \cdot \vec{\mathrm{n}}_{031} da + \int \vec{F}_\omega \cdot \vec{\mathrm{n}}_{123} da$$

(12) (6)

Evaluating these bounded integrals along the faces of the tetrahedron V_t yields the following results (Brandolini and Colzani, 1997; Li and Xu, 2009):

$$\hat{\chi}_{V_t}(\omega) = V_t(Q_0 + Q_1 + Q_2 + Q_3) \tag{13}$$

where V_t is the volume of the tetrahedron and is defined as follows,

$$V_t = \frac{1}{6}(-x_3 y_2 z_1 + x_2 y_3 z_1 + x_3 y_1 z_2 - x_1 y_3 z_2 - x_2 y_1 z_3 + x_1 y_2 z_3) \tag{14}$$

and Q_i is as follows:

$$Q_i = \frac{i e^{-2\pi i p . \omega}}{\prod_{\substack{k \in [0,3] \\ k \neq i}} (l(x_i - x_k) + m(y_i - y_k) + n(z_i - z_k))} \tag{15}$$

We should note that computing the Fourier coefficient of the characteristic function χ_{V_t} using Stokes' theorem defined in Equations (6) to (15), all the Fourier coefficients of the characteristic function except the constant order term, $l = m = n = 0$. Thus, the obtained function is well defined up to an additive constant. This constant order term can be calculated using Equation (2) as follows: (8)

$$\hat{\chi}_M(0,0,0) = \int_{p \in M} dp \tag{16}$$

which is equal to the volume of the mesh.

RESULTS

Our algorithm is implemented in C++, using the Computational Geometry Algorithms Library (CGAL, 2011). The experiments were run on a PC with a dual-core 2.8 GHz processor and 4 GB of memory. The input solids are given as oriented triangulated meshes. If the meshed are not triangulated a preprocessing step is required to convert the polygonal faces to triangles. We compute the required Fourier coefficients of the indicator function of the input object using Equations (5), (13) to (15). Our algorithm evaluates the resulting Fourier transform in continuous form. There is no discretization alias since we can evaluate a Fourier transform coefficient from the continuous form directly. The computational complexity of the Fourier transform of the

Table 1. The average times, in seconds, for the calculation of the Fourier coefficients of the input objects as a function of the number of triangles T and bandwidthB. The first column of s corresponds to our approach and the second column corresponds to the approach described by Frigo and Johnson (2005) including the voxelization preprocessing time.

$\#^T$	$B = 32$		$B = 64$		$B = 128$	
	s	s	s	s	s	s
	(Ours)	**Frigo and Johnson (2005)**	**(Ours)**	**Frigo and Johnson (2005)**	**(Ours)**	**Frigo and Johnson (2005)**
1000	0.01	0.07	0.04	0.1	0.15	0.5
10000	0.1	0.6	0.35	1.1	1.5	4
100000	1	5	3.8	10	16	30

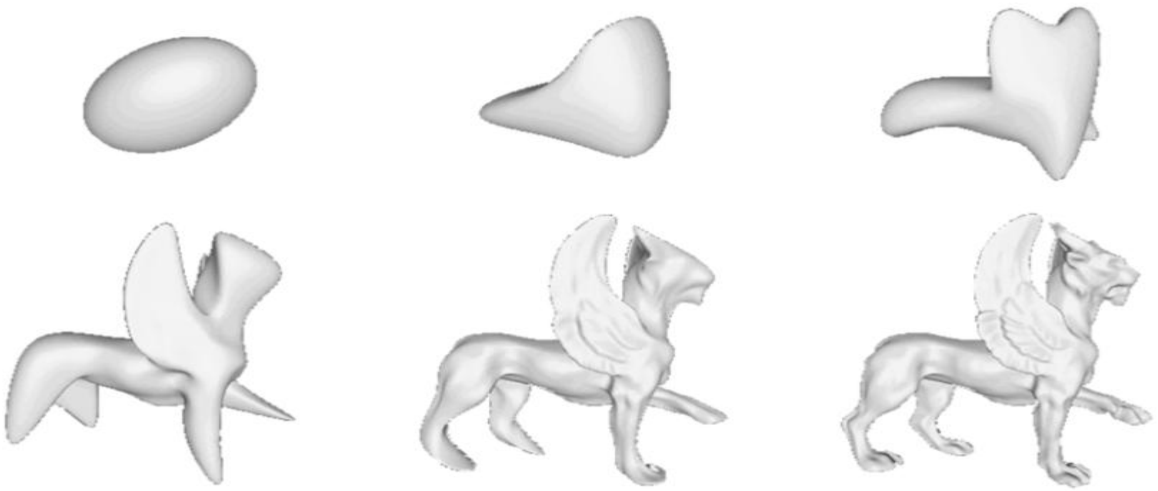

Figure 5. An object constructed using a successive calculation of the frequency components using our approach. The number of calculated frequency components for each version is 2, 8, 14, 45, 64 and 96 respectively.

input surface is described in Table 1. shows the timing (in seconds) of calculating the Fourier coefficients of the indicator function of 3D objects having different number of triangles. We compare our algorithm with the best known algorithm calculating the Fourier coefficients (Frigo and Johnson, 2005) in combination with the voxelization proposed by Kaufman and Shimony (1987). From Table 1, it can be seen that our time is much better than that of Frigo and Johnson (2005), and that our time is proportional to the number of triangles of the model and the required bandwidth. In fact, our algorithm runs in $O(b^2 + \#T)$ time, where b is the bandwidth and $\#T$ is the triangle count.

In order to determine how well our method works in practice, we ran our approach on oriented triangulated solids and applied the obtained frequency-based representation on two applications filtering and texture transfer.

Progressive transmission

Progressive transmission of 3D models is an important application for remote visualization of 3D objects. Transmission of the whole model might take minutes or even hours for a large mesh using a low-bandwidth network. The progressive transmission using our approach is based on the following fact. The low frequency components reflect the overall shape of the objects and the higher components correspond to the finer details of the objects. Therefore, our method successively transmits the surface geometry starting with a set of the first few frequency components. Then the next order frequency components are then progressively transmitted to add more details to the object. Transmitting the whole object while getting an early impression of its appearance is easily and efficiently done by transmitting the frequency components calculated using our approach.

Figure 5 shows an example of calculated multi-resolution versions of a 3D object. Each multi- resolution level is computed by evaluating the set of frequency components that corresponds to the required level of details. A highly detailed version of the object is obtained only using 96 frequency components. This reduces the number of coefficients sent through the network to

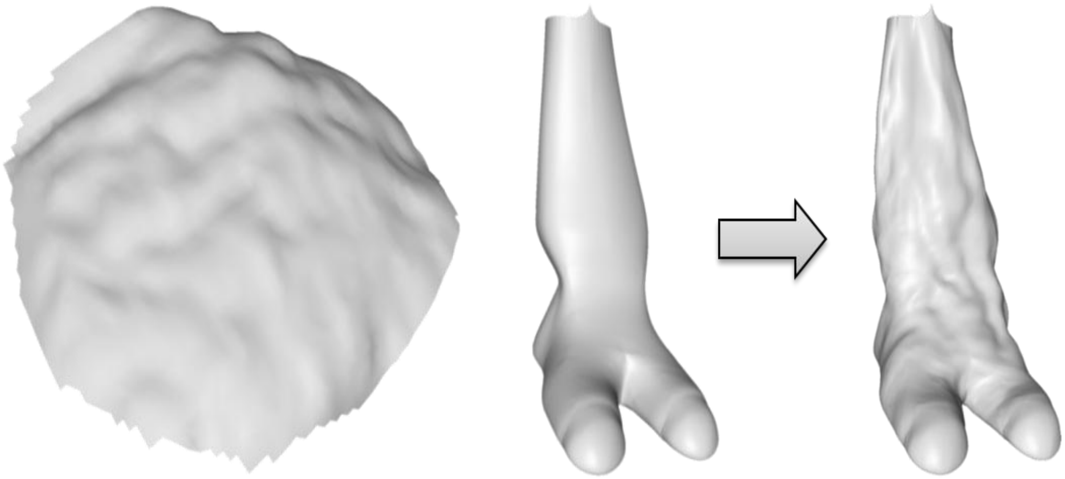

Figure 6. A geometric details are transferred from an initial surface (left) to another surface (middle). The rightmost object is the textured target.

visualize the objects.

In the view-dependent framework the user only requests a small part of the object stored at the server. Our approach handles this case by calculating only the frequency coefficients of the tetrahedrons corresponding to the mesh triangles that match the selection criteria given by the user.

Texture transfer

Another application of our approach is the transfer of geometric details from a surface to another (Bhat and Ingram, 2004; Ran and Meng, 2010). Geometric texturing uses localized 3D vertex displacements to represent surface texture in place of conventional 2D intensity based texture mapping. Here, we investigate “geometric texturing by example” - the transfer and synthesis of geometric texture from real surface meshes (source), captured via 3D laser scanner, onto synthetically created surface meshes (target). This application is based on the fact that the higher frequency components correspond to the details of the surface. Therefore, these high frequency components capture the required geometric details to be transferred.

The frequency components are computed for the two objects, the source and the target. The high frequency components of the source object are added to the corresponding frequency components of the target object. The transferred frequency components add some geometric details to the target objects similar to that of the source object.

Figure 6 shows an example of transferring a geometric texture from the source object (left) to the target object (middle). The level of details to be transferred depends on the number of high frequency components evaluated from the source object.

Conclusion

In this paper, we have presented a novel method for calculating the Fourier coefficients of 2-manifolds from oriented triangular meshes. An explicit method to compute the Fourier transform from the mesh representation directly is given. Our method differs from past approaches in that it leverages Stokes' theorem to provide a method for distributing the calculations over the tetrahedrons composing the volume of the mesh. Consequently, we provide a computational method that is both simple and efficient. The distribution of computations over the tetrahedron would enable us to run our method on each of the local surface patches independently, allowing us to overcome the memory bottleneck that restricts the calculations of higher bandwidths. We have shown that the method is robust and can be used in many applications such as progressive transmission and geometric texture transfer.

The proposed algorithm still has some room for improvement. For example, the Fourier transform may lose its computational efficiency if many coefficients are required simultaneously. More research is in progress to speed this up.

REFERENCES

Arnold D, Falk R (2006). Finite element exterior calculus, homological techniques, and applications. Acta Numerica (15):1-155.

Bhat P, Ingram S (2004). Geometric texture synthesis by example. Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. Nice, Franc, ACM, pp. 41-44.

Brandolini L, Colzani L (1997). Average decay of Fourier transforms and integer points in polyhedra. Arkiv för Matematik 35(2):253-275.

Carey R (1998). The Virtual Reality Modeling Language Explained. IEEE MultiMedia 5(3): 84-93.

CGAL (2011). Computational Geometry Algorithms Library, http://www.cgal.org.

Frigo M, Johnson SG (2005). The Design and Implementation of

FFTW3. Proceed. IEEE. 93(2):216-231.

Guskov I, Sweldens W (1999). Multiresolution signal processing for meshes. Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing: pp. 325-334.

Hildebrandt K, Polthier K (2006). On the convergence of metric and geometric properties of polyhedral surfaces. GEOMETRIAE DEDICATA. pp. 89-112.

Karni Z, Gotsman C (2000). Spectral compression of mesh geometry. Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co.: pp. 279-286.

Kaufman A, Shimony E (1987). 3D scan-conversion algorithms for voxel-based graphics. Proceedings of the 1986 workshop on Interactive 3D graphics. Chapel Hill, North Carolina, United States, ACM: pp. 45-75.

Knoll A (2006). A Short Survey of Octree Volume Rendering Techniques. Proceedings of 1st IRTG Workshop.

Lee AWF, Sweldens W (1998). MAPS: multiresolution adaptive parameterization of surfaces. Proceedings of the 25th annual conference on Computer graphics and interactive techniques, ACM, pp. 95-104.

Li H, Xu Y (2009). Discrete Fourier analysis on a dodecahedron and a tetrahedron. Math. Comput. 78:999-1029.

Meyer M, Desbrun M (2003). Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. Vis. Math 3:35-57.

Pauly M, Gross M (2001). Spectral processing of point-sampled geometry. Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM. 379-386.

Pinkall U, Polthier K (1993). Computing discrete minimal surfaces and their conjugates. Exp. Math. Comput. 2(1):15-36.

Ran LQ, Meng XX (2010). Geometry texture synthesis based on Laplacian texture image. J. Comput. Sci. Technol. 25(3):606-613.

Taubin G (1995). A signal processing approach to fair surface design. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, ACM. pp. 351-358.

Tolimieri R, An M, Chao Lu (1997). Algorithms for Discrete Fourier Transform and Convolution, Springer.

Vacavant A, Coeurjolly D (2009). A novel algorithm for distance transformation on irregular isothetic grids. Proceedings of the 15th IAPR international conference on Discrete geometry for computer imagery, Springer-Verlag, pp. 469-480.

Wardetzky M, Bergou M (2007). Discrete quadratic curvature energies. Comput. Aided Geom. Des. 24(8-9):499-518.

Wardetzky M, Mathur S (2008). Discrete Laplace operators: no free lunch. ACM SIGGRAPH ASIA 2008 courses. Singapore, ACM: 1-5.

Zhang H (2004). Discrete combinatorial Laplacian operators for digital geometry processing. Proceedings of SIAM Conference on Geometric Design, pp. 575-592.

Zhang H, Kaick Ov (2007). Spectral Methods for Mesh Processing and Analysis. Proceeding of Eurographics State-of-the-art Report, pp. 1-22.