Stereo music visualization through manifold harmonics

THOMAS LEWINER[1], CLARISSA MARQUES[1], JOÃO PAIXÃO[1], SCARLETT DE BOTTON[1], ALLYSON CABRAL[1], RENATA NASCIMENTO[1], VINÍCIUS MELLO[2], ADELAILSON PEIXOTO[3], DIMAS MARTINEZ[3] AND THALES VIEIRA[3]

[1] Department of Matematics — Pontifícia Universidade Católica — Rio de Janeiro — Brazil
[2] Institute of Mathematics — Universidade Federal da Bahia — Salvador — Brazil
[3] Institute of Mathematics — Universidade Federal de Alagoas — Maceió — Brazil

Abstract. Music visualizations are nowadays included with virtually any media player. They usually rely on harmonic analysis of each sound channel, which automatically generate parameters for procedural image generation. However, only few music visualizations make use of 3d shapes. This paper proposes to use spectral mesh processing techniques, here manifold harmonics, to produce 3d stereo music visualization. The images are generated from 3d models by deforming an initial shape, mapping the sound frequencies to the mesh harmonics. A symmetry criterion is introduced to enhance the stereo effects on the deformed shape. A concise representation of the frequency mapping is proposed to allow for an animated gallery interface with genetic reproduction. Such galleries let the user quickly navigate between visual effects. Rendering such animated galleries in real-time is a challenging task, since it requires computing and rendering the deformed shapes at a very high rate. This paper introduces a direct GPU implementation of manifold harmonics filters, which allows the displaying of the animated galleries.

Keywords: ***Manifold Harmonics. Symmetry. Sound Visualization. Stereophony. Geometry Processing. GPU. Design Galleries.***

Figure 1: *Music visualization by deforming a 3d model according to the music amplitudes.*

1 Introduction

The illustration of music became a necessary part of the audio industry. On the one hand video clip creation represent a whole part of a song production, while on the other hand any computer program that renders sound content offers several visualizations. Most audio visualization techniques rely on Fourier transforms that extract the harmonic amplitudes of the sound samples for the left and right channels. These amplitudes serve as parameters to algorithms that generate beautiful or exciting images in real time, using procedural techniques from simple digital peak meters to psychedelic dynamical systems. We propose to generate images obtained by deforming an initial discrete 3d model (Figure 1).

Since sound analysis relies on sound harmonics, a natural approach is to define correspondences with geometric harmonics to deform the 3d model. A definition of such geometric harmonics, called *manifold harmonics* has been recently proposed by Vallet and Lévy [26]. Amplifying some harmonics of a given mesh leads to coherent deformations, in the sense that filtering low frequencies actually deforms the global shape of the mesh, while altering high frequencies changes its details. This paper is an extended version of a conference publication [15], as it uses the left and right channels of stereo sound to define the harmonic filters. More precisely, a fast approximate characterization of the symmetric nature of manifold harmonic is proposed as a fast approximation of Ovsjanikov *et al.* proposal [20]. This separation between symmetric and non-symmetric harmonics permits to map the sound frequencies common to the left and right left channels [4] to symmetric manifold harmonics, and the channels difference to the other harmonics.

However, using manifold harmonics for sound visualiza-

Preprint MAT. 15/10, communicated on December 17^{th}, 2010 to the Department of Mathematics, Pontifícia Universidade Católica — Rio de Janeiro, Brazil. The corresponding work was published in The Visual Computer.

tion poses a two-fold challenge: First, manipulating the amplitudes of each harmonic is a delicate task, since closer frequencies have very different and dramatic impacts on the shape. Second, the 3d deformation must be rendered in real-time and maintain synchronization with the music. In this paper, we propose to model the mapping of sound harmonics amplitudes to manifold harmonics amplitude using a design gallery with genetic reproduction, similarly to volume visualization [23].

The use of galleries makes the second challenge even more difficult, since an animated gallery requires computing and rendering several deformations of the initial mesh for each frame. We propose here a direct GPU implementation of the manifold harmonics filter that copes with such requirements. For models containing around 50,000 vertices, we can render a gallery of 12 animated deformations in real-time.

2 Related work

There are several techniques for automatic music visualization, as seen in many media players. However, only a few of them use 3d models [12]. To the best of our knowledge, the closest work automatically relating sound and 3d objects comes from granular mechanics simulation [3]. It dates back to the studies of vibration modes [10], when *Modal Analysis* [5] became a very important tool in the understanding of structure responses to mechanical forces. Modal Analysis was first introduced to Computer Graphics by Pentland and Williams [22], where it was used to simulate deformations in non-rigid objects from a sound signal. A reduced version of such simulations has recently been brought to real-time through a GPU implementation, but only using the first few vibration modes [30]. In this paper, we propose a music visualization scheme instead of a mechanical simulation, and achieve real-time performance in an entirely spectral processing system. Note that the inverse process, i.e. creating audio content from a 3d animation, has ben proposed by O'Brien *et al.* [19].

Stereo music effects have been used in music recording and reproduction since many decades [6]. It adapts to the human binaural audition system, which helps the brain to derive 3d perception of the sound. Although advanced techniques can be used to analyze and synthesize 3d sound effects, such as holophonics or sound 3d positioning [4, 7], this work uses very basic sound analysis as far as symmetry between the left and right channels are concerned.

Since the seminal work of Taubin [25], several approaches have been proposed to adapt signal processing techniques to discrete surfaces. Among those, spectral processing has gained a lot of attention [14]. Those methods rely on defining an equivalent for Fourier harmonics (basically sine and cosine) as eigenvectors of Laplace-like operators. Among those works, Vallet and Lévy proposed a manifold harmonics adapted to mesh edition [26]. This work motivated several applications in connected fields: spectral mesh deformation [24], mesh watermarking [17, 28] and shape analysis [29], and in particular symmetry detection [20], which we extend here for our symmetry criterion. In this paper, we use manifold harmonics filters, and propose a GPU implementation of spectral filtering to obtain real-time deformations.

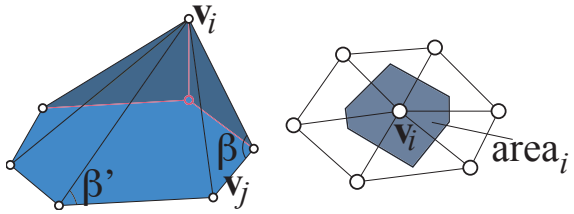


Figure 2: *Geometric elements for the coefficients of the discrete Laplace operator.*

3 Manifold Harmonics

In this section, we recall the basics of manifold harmonics following the original work of Vallet and Lévy [26].

The idea behind manifold harmonics is to transpose usual Fourier tools to 3d meshes. In Fourier analysis, a functional basis $\mathbf{h}_\omega(t) = \mathbf{e}^{-2\pi\cdot i\cdot\omega\cdot t}$ of so-called *harmonics* is used to decompose an input signal $f(t)$ into a combination of those harmonics:

$$f(t) = \int_{\mathbb{R}} \tilde{f}(-\omega)\cdot\mathbf{h}_\omega(t)\mathbf{d}\omega, \text{ with } \tilde{f}(\omega) = \int_{\mathbb{R}} f(t)\cdot\mathbf{h}_\omega(t)\mathbf{d}t.$$

(a) Laplace harmonics

The main observation is that harmonics $\mathbf{h}_\omega$ are the eigenvectors of the differential Laplace operator Δ_∂:

$$\Delta_\partial(\mathbf{h}_\omega) \equiv \frac{\partial^2\mathbf{h}_\omega}{\partial t^2} = \lambda_\omega\cdot\mathbf{h}_\omega \quad with \quad \lambda_\omega = -4\pi^2\,\omega^2\,.$$

To transpose such decomposition on a mesh, a natural option is to look for the eigenvectors of a discrete Laplace operator. Vallet and Lévy derive a Laplace-De Rham operator from Discrete Exterior Calculus [26]. On the vertices of a mesh, this operator turns out to be linear, and can thus be expressed as an $n\times n$ matrix Δ, where n is the number of vertices of the mesh. The coefficients Δ_{ij} are zero if vertices i and j are not adjacent, and otherwise:

$$\Delta_{ij} = -\frac{cot\,(\beta_{ij}) + cot\,(\beta'_{ij})}{\sqrt{area_i\cdot area_j}}\,, \quad \Delta_{ii} = -\sum_j \Delta_{ij}\,,$$

where $area_i$ and $area_j$ are the areas of the restricted Voronoi region of vertex i and j, respectively, and the angles β_{ij} and β'_{ij} are opposite to the edge between i and j (Figure 2).

(b) Manifold harmonics transform

With a rescaling of the areas $area_i$ [26], the matrix of this discrete Laplace operator Δ is symmetric, and can thus be diagonalized, producing an orthogonal basis eigenvectors of $\mathbf{H}^k \in \mathbb{R}^n$ associated to eigenvalues $\Lambda_k \in \mathbb{R}$. Since this is a basis in $\mathbb{R}^n$, any function $F : i \in \{0, \ldots, n-1\} \mapsto \mathbb{R}$ defined on the vertices of the mesh can be decomposed onto this basis:

$$F(i) = \sum_{k=0}^{n-1} \tilde{F}(k) \cdot \mathbf{H}_i^k \ , \text{ with } \tilde{F}(k) = \sum_{i=0}^{n-1} area_i \cdot F_i \cdot \mathbf{H}_i^k \ .$$

Using the analogy with Fourier analysis, the frequency associated with Λ_k is $\sqrt{\Lambda_k}$, and we consider that frequencies are ordered: $\Lambda_0 \leq \Lambda_1 \leq \ldots \leq \Lambda_{n-1}$.

(c) Filtering

The signal $F(i)$ is thus expressed as a combination of harmonics $\mathbf{H}^k$, with respective amplitudes $\tilde{F}(k)$. A linear filter can then be derived by amplifying each harmonic $\mathbf{H}^k$ by a factor $\varphi(k)$. The filtered signal $F_\varphi(i)$ is given by:

$$F_\varphi(i) = \sum_k \varphi(k) \cdot \tilde{F}(k) \cdot \mathbf{H}_i^k \ .$$

Since we are interested in deforming the mesh, we will consider the signal $F(i)$ to be the coordinates $x(i)$, $y(i)$, $z(i)$ of vertex i. We therefore get three harmonic amplitudes $\tilde{x}(k), \tilde{y}(k), \tilde{z}(k)$ for each frequency k. Since the mesh is not aligned *a priori*, we will filter all the three coordinates with the same gain φ. Finally, since high frequencies ($k \geq \#k$) correspond to very small perturbations, appearing as noise, we filter them by a single gain φ_d:

$$F_\varphi(i) = \sum_{k=0}^{\#k-1} \varphi(k)\tilde{F}(k)\mathbf{H}_i^k + \varphi_d \, d_i, \text{ with } d_i = \sum_{k=\#k}^{n-1} \tilde{F}(k)\mathbf{H}_i^k.$$

The coefficients d_i can be computed at preprocessing.

4 Tuning Manifold Harmonics

We want to apply manifold harmonics filters to illustrate signals $f(t)$ such as audio content. Since manifold harmonics filters are very sensitive to the gain function, eventually leading to large deformation for small variations of the filter, the mapping of signal harmonic amplitudes $\tilde{f}(k)$ to the manifold harmonic amplitudes $\varphi(k)$ would require a very careful edition if done manually. In this section, we introduce a simple design model for such mapping. This design allows a gallery interface [18]. Design galleries presents the user a broad, selection of different mappings as real-time animations generated for each mapping. Once the user chooses the results he prefers, a new gallery is proposed by genetic reproduction of the user choices [23], letting him quickly navigate and eventually converge to a desired effects (Figure 4).

(a) Mapping to manifold harmonics filters

We want to design a filter $\varphi(k)$ from the harmonic amplitudes $\tilde{f}(\omega)$ of an input signal, where the dependency $\varphi(k) = \Phi(\tilde{f})(k)$ of φ from $\tilde{f}$ is not necessarily linear. Moreover, the number of frequencies $\#\omega$ computed from the signal may differ from the number of harmonics $\#k$ of the mesh. We decompose this mapping into two steps: a frequency transfer function $t : \omega \mapsto k \in \{0, \ldots, \#k - 1\}$ combined with an amplification function $a : k \mapsto a(k) \in \mathbb{R}$ applied on the manifold harmonic amplitudes.

We want each harmonic of the mesh to receive contributions from different harmonics of the signal, so that a musical instrument, which covers different frequencies, could be mapped to a single manifold harmonic. Therefore, the transfer function maps sound frequencies ω to mesh frequencies k, and a mesh frequency k will receive contributions from all the sound frequencies in $t^{-1}(\{k\})$. We propose a harmonic mapping $\Phi_{t,a} : \tilde{f} \mapsto \varphi$ as (Figure 3):

$$\Phi_{t,a}(\tilde{f})(k) = a\,(k) \cdot \Big(\sum_{\omega \in t^{-1}(\{k\})} \tilde{f}\,(\omega) \Big) + 1 \ .$$

By adding one, we maintain the usual intuition of amplification: amplifying all the mesh frequencies to 0 (*i.e.* $a \equiv 0$) does not deform the mesh. Note that, since the harmonic amplitudes of the sound may be negative, the amplification $a(k)$ may also be negative.

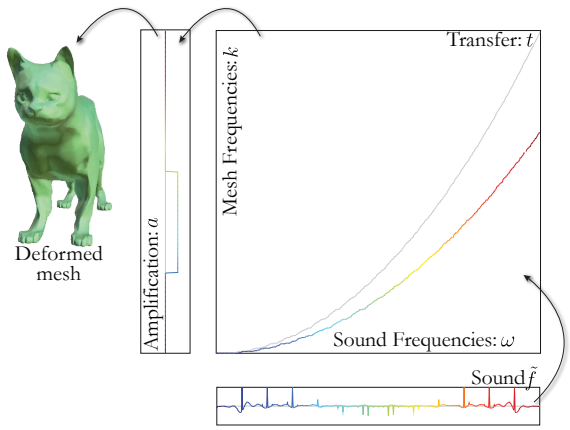


Figure 3: *Filter design as the composition of a frequency transfer function t and an amplification function a (drawn vertically). The grey curve for the transfer corresponds to the direct mapping from $\omega \cong \sqrt{\Lambda_{t(\omega)}}$.*

(b) Tuning through design galleries

The filter design above gives a concise representation of the harmonic mapping $\Phi_{t,a}$ from sound harmonic amplitudes to manifold harmonic amplitudes. Indeed $\Phi_{t,a}$ is represented as two vectors: $t \in \mathbb{N}^{\#\omega}$ is an integer vector of size $\#\omega$, and $a \in \mathbb{R}^{\#k}$ is a real vector of size $\#k$. This allows to easily mix two harmonic mappings by combinations of those vectors. Using terminology from genetic algorithms, the harmonic mapping $\Phi_{t,a}$ is represented by two chromosomes a and t, which can reproduce by combination.

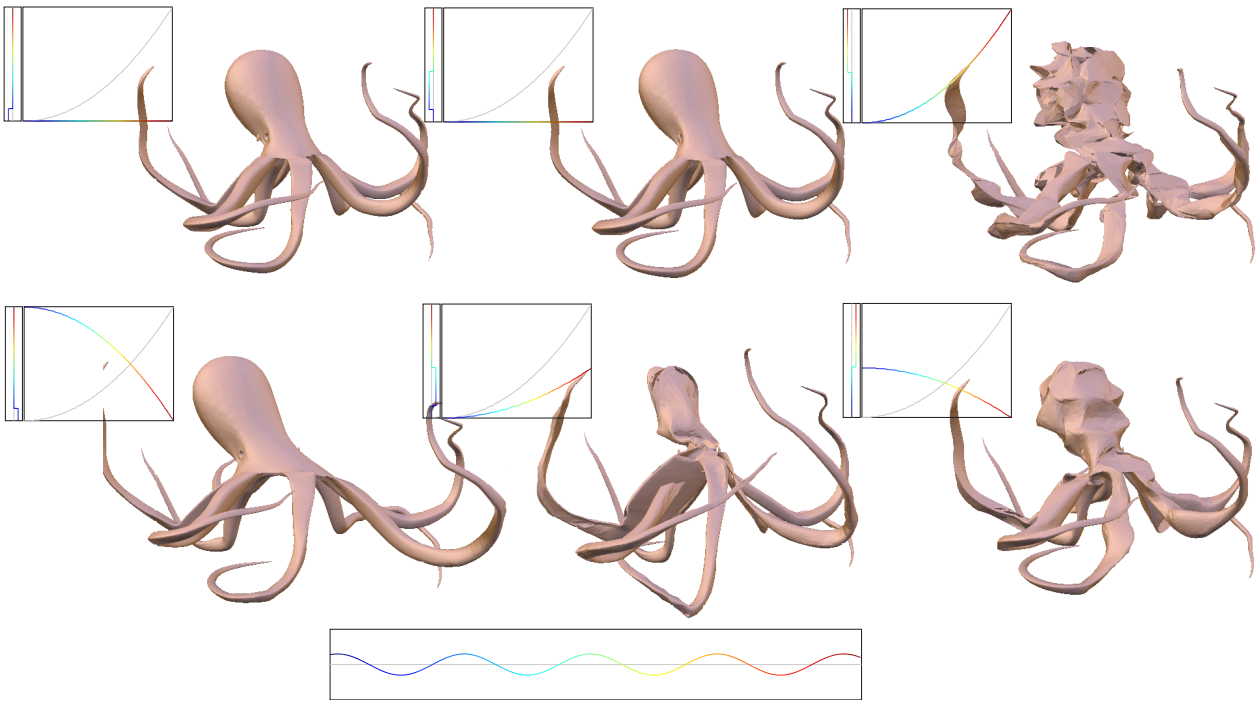


Figure 4: *Initial gallery applied to an octopus model, with the corresponding transfer and amplification functions (rainbow curves respectively at the right and left of each element, see Figure 3). The frequency input $\tilde{f}$ is drawn at the bottom.*

This leads to a direct design gallery interface, where different harmonic mappings are proposed to the user, who can select the ones he likes the most. From this selection, a new gallery is generated using genetic reproduction, until the user chooses only one harmonic mapping, as explained in the next section. The following section will detail the initial gallery creation. The harmonic mapping can then be directly edited from the two curves of t and a.

(c) Reproduction

The reproduction generates a new gallery of S harmonic mappings from a selection of old mappings. To do so, S pairs of distinct selected old mappings are randomly chosen. Each pair is then combined into a new mapping as follows.

Since the frequency transfer and amplification functions t and a have complementary effects, we reproduce them independently. This also reduces the initial gallery size, as explained in the next subsection. In practice, this means that we first decide if we combine the frequency transfer functions of the pair using a $\frac{1}{2}$-Bernoulli trial ("heads or tails"). We also decide in a similar manner if the amplification functions will be combined.

The combination of the frequency transfer functions t' and t'' of the pair is done as follow. First, we randomly choose an integer value, n_k^0, as a geometric random variable in $\{1,\dots,\#k\}$, and a random real value w^0 uniformly in $[0,1]$. We then set the first n_k^0 coefficients of vector t as the first n_k^0 coefficients of $w^0 \cdot t' + (1-w^0) \cdot t''$. We choose again random values $n_k^1 \in \{1,\dots,\#k\}$ and $w_1 \in [0,1]$, and clamp n_k^1 to ensure $n_k^0 + n_k^1 \leq \#k$ (the geometric random process intends to reduce the effect of this clamping). We then set the following n_k^1 values of t as above, and repeat until completing all the frequencies. We perform the same operations for the amplifications (Figs. 4 and 5).

This combination method avoids producing combination that varies too quickly, as compared to randomly choosing real values w at each frequency.

(d) Gallery initialization

We propose a systematic gallery initialization (Figure 4) that could theoretically, with at least two elements, generate any harmonic mapping by the above reproduction. Since the reproduction of the frequency transfer and amplification are independent, although combined to generate each animation of the gallery, we can use a given number S of elements of the initial gallery to span the frequency transfer functions and the same S elements to span the amplification functions. This reduces the size of the initial gallery, although it requires one more reproduction to get non-trivial mappings.
The first transfer function is the direct mapping:

$$t_{ini}(\omega) = \min \left\{ k \text{ such that } \frac{\omega}{\omega_{\#\omega}} \leq \sqrt{\frac{\Lambda_k}{\Lambda_{\#k}}} \right\} .$$

This expression ensures that, if there exist a unique k such that $\frac{\omega}{\omega_{\#\omega}} = \sqrt{\frac{\Lambda_k}{\Lambda_{\#k}}}$, then $t_{ini}(\omega) = k$. This function maps the sound low (respectively high) frequencies to the mesh low (respectively high) frequencies. The function $t_{rev} = \#k - t_{ini}$ maps high sound frequencies to low ones.

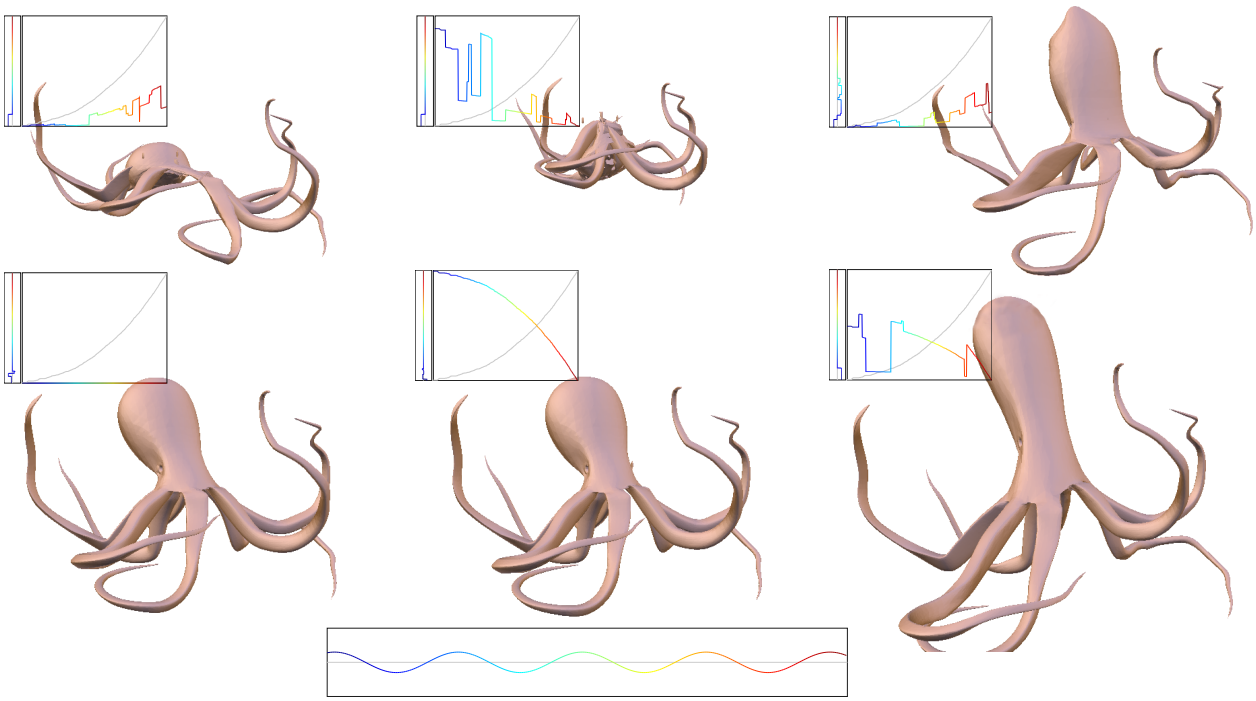


Figure 5: *Gallery after one reproduction from the 1^{st}, 4^{th} and 5^{th} items of Figure 4 in reading order, with the genetically reproduced transfer and amplification functions.*

Usually, altering the low frequencies of the mesh produces more visible effects. We thus define the frequency transfer functions of the initial galleries as functions whose image lie in the low frequency domain: $t(\omega) = t_{ir}(\alpha \cdot \omega)$, where $\alpha \in \{0, \frac{2}{S-1}, 2\frac{2}{S-1}, \ldots, 1\}$ and t_{ir} is either t_{ini} or t_{rev}. The first value $\alpha = 0$ is a constant mapping, and is included so that any transfer function can be obtained by combination.

The amplification functions of the initial gallery are simple band-pass filters, with positive or negative factors. The manifold harmonic frequencies $\{0, \ldots, \#k - 1\}$ is divided in intervals I_α, for $\alpha \in \{\frac{2}{S}, 2\frac{2}{S}, \ldots, \frac{4}{S^2}\}$. After we define the amplification function for the first half of the gallery $a_\alpha(k) = M$ if $k \in I_\alpha$, and $a_\alpha(k) = 0$ otherwise, where M is the maximal amplification factor. The other half is defined similarly using $-M$. If the sound amplitudes are normalized to $[-1, 1]$ and the if the mesh is reasonably smooth, the order of magnitude of M is 5,000. Since we try to emphasize the low frequencies, we define the intervals $I_\alpha = [\alpha^2, (\alpha + \frac{2}{S-1})^2]$.

5 Stereo and symmetric effects

Stereo sound offers richer information on the sound signal f. In particular, the left f_L and the right f_R channels are naturally combined into a center (symmetric) signal $f_s = \frac{1}{2}(f_L + f_R)$ and a side (antisymmetric) signal $f_u = \frac{1}{2}(f_L - f_R)$, such as for stereo widening. We propose here to classify the manifold harmonics basis $\mathbf{H}^k$ in symmetric and antisymmetric harmonics. Then we map the center signal frequencies $\tilde{f}_s$ into symmetric harmonics and the side signal into the other harmonics.

(a) Symmetry classification of manifold harmonics

There exist several works on intrinsic symmetry detection using spectral representation of the geometry, in particular using manifold harmonics [20]. These works look for discrete transformations $T : \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$, mapping vertices to vertices that preserve the intrinsic geometry of the shape, typically geodesic or diffusion distances between the vertices. Here, we do not look for transformations, but for mesh frequencies k that would be preserved by such transformations. A simple option would be to compute first a set of transformations T and check if they generally preserve the k^{th} harmonic: $\mathbf{H}^k_{T(i)} \approx \mathbf{H}^k_i$. We propose here a fast approximation, which avoids generating explicitly the set of transformations.

Following the definitions of Ovsjanikov *et al.* [20], T is an intrinsic symmetry if for any two vertices i, j:

$$\sum_k \frac{1}{\Lambda_k} \left(\mathbf{H}^k_i - \mathbf{H}^k_j\right)^2 = \sum_k \frac{1}{\Lambda_k} \left(\mathbf{H}^k_{T(i)} - \mathbf{H}^k_{T(j)}\right)^2 .$$

Moreover, if we consider only frequencies k associated to non-repeating eigenvalues Λ_k, T either preserves or changes the sign of the harmonics $\mathbf{H}^k$: $\exists \epsilon^k_T \in \{-1, +1\}$ such that $\forall i, \mathbf{H}^k_{T(i)} = \epsilon^k_T \mathbf{H}^k_i$. A harmonic $\mathbf{H}^k$ is said to be positive for T if they are preserved by T ($\epsilon^k_T = +1$) and negative otherwise [20]. We say a harmonic is symmetric if it is positive for a majority of intrinsic symmetries and antisymmetric otherwise.

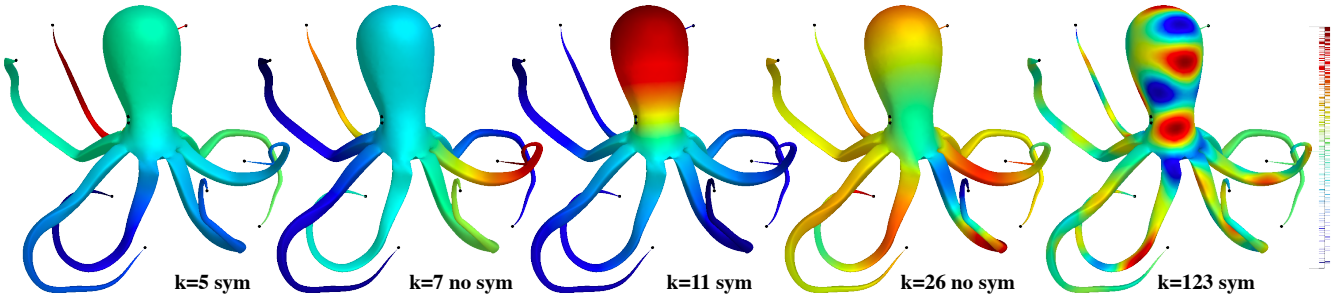


Figure 6: *Some eigenvectors $\mathbf{H}^k$ of low frequency: vertices are colored in blue if $\mathbf{H}^k_v$ is small, and red if big. With the pairs of Figure 8, symmetric eigenvalues ends up having similar variations on at least 7 of the 8 tentacles.*

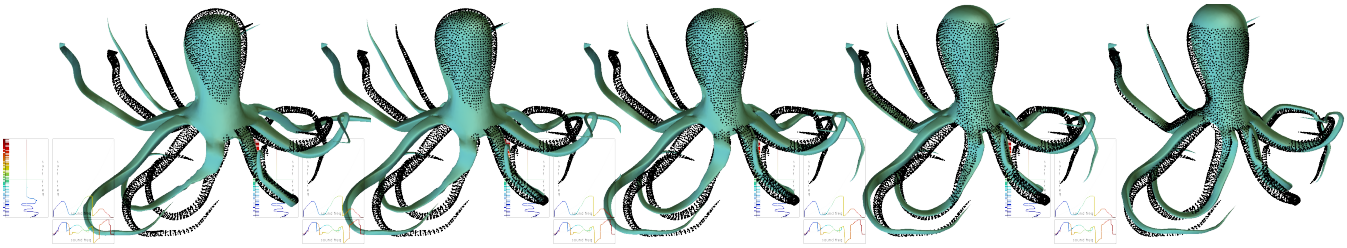


Figure 7: *Effects of using only antisymmetric harmonics, mainly deforming the tentacles (left) to only symmetric ones, focusing more on the head (right). The black dots represents the original model.*

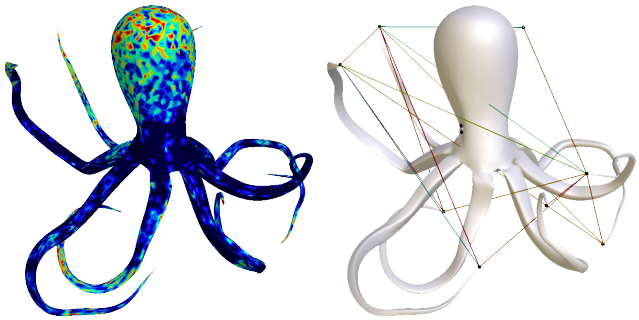


Figure 8: *Number of times a vertex is an extremum for the first 15 frequencies (left), coded from blue (0) to dark red (12), and the selected vertices (black dots) and pairs (lines) of extrema likely to be exchanged by a symmetry (right).*

(b) Fast symmetry class approximation

Computing all the intrinsic symmetries T is an expensive task, but we can estimate if a harmonic would be positive for a majority of them. Since an intrinsic symmetry T preserves (resp. inverses) each harmonic, it maps the local maxima of $\mathbf{H}^k$ to local maxima (resp. minima), and vice-versa. Therefore, a pair (i,j) of extrema of $\mathbf{H}^k$ can be exchanged by T only if $\mathbf{H}^k_i = \epsilon^k_T \mathbf{H}^k_j$. Observe that this is not a sufficient condition, since a symmetry T exchanging vertices i and j may not exist.

We restrict ourselves to pairs (i,j) of extrema that are likely to be exchanged by an intrinsic symmetry. More precisely, we select the pairs of vertices having the closest absolute GPS signature $s(i) = \left(\frac{|\mathbf{H}^k_i|}{\Lambda_k}, k < n_k\right)$, with n_k typically around 15 [20]. To avoid looking for all pairs of vertices, we further restrict ourselves to pairs of vertices being critical for at least c of the first n_k harmonics, with c around 10.

Putting all together, we first compute for the first n_k frequencies the critical points of $\mathbf{H}_k$, and select the vertices being extrema for at least c frequencies (Figure 8). Then we compute the set of pairs of selected vertices having the closest absolute GPS signature. Finally, we classify a frequency k as symmetric if the set of pairs (i,j) for which $\mathbf{H}^k_i = \mathbf{H}^k_j$ has more elements than the pairs for which $\mathbf{H}^k_i = -\mathbf{H}^k_j$. Although this is an approximation, it gives a coherent classification for animation purposes (Figure 6) and it is several orders faster than enumerating discrete transformations [20].

(c) Stereo sound mapping

The center sound signal f_s is the symmetric part of the sound and is used mainly for the symmetric harmonics, while the side sound f_u is used for the antisymmetric ones. This distribution is controlled by a parameter μ such that this occurs for $\mu = 0$, and for $\mu = 1$, the left channel is used for symmetric harmonics and the right one for antisymmetric:

$$\Phi_{t,a}(\tilde{f}_s, \tilde{f}_u)(k) = \begin{cases} \Phi_{t,a}(\tilde{f}_s - \mu\, \tilde{f}_u)(k) & \text{if } k \text{ is symmetric} \\ \Phi_{t,a}(\tilde{f}_u + \mu\, \tilde{f}_s)(k) & \text{otherwise} \end{cases}.$$

The main difficulty of this approach is that the sound recorded on the right microphone is usually audible by the left one. Therefore, the sound is never completely antisymmetric, which means that the side signal f_u is usually much weaker than the center signal. We compensate for this by amplifying f_u independently, by a factor that we experimentally estimate to 10 (Figure 7). This extra parameter μ is included in the gallery with a random initialization, forcibly including 0, 1.

The corresponding work was published in The Visual Computer.

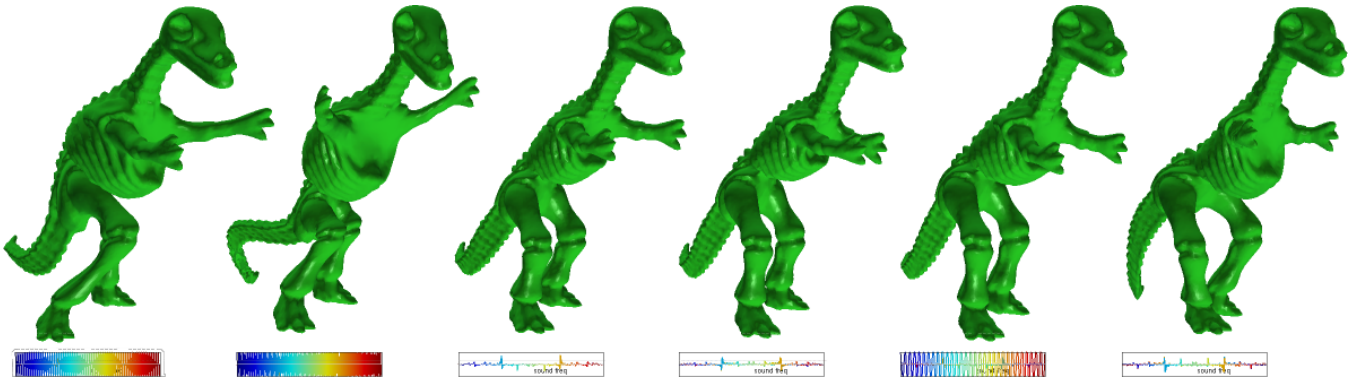


Figure 9: *Frames of forró music visualization using the dinosaur model.*

6 Running It Real-Time

The main challenge for the above interface to work with sound signals is to compute and render the deformation of each gallery element synchronously with the music (Figure 9). If we have S elements in the gallery, each of which is a mesh of n vertices with $\#k$ manifold frequencies and $\#\omega$ sound frequencies, a single frame represents $O(S \cdot \#\omega \cdot \#k \cdot 3n)$ operations! (The $\#\omega$ factor comes from the evaluation of $\Phi_{t,a}$). We thus propose a GPU implementation of the harmonics filter, while the decomposition is pre-computed on the CPU.

(a) GPU implementation

For the sake of portability, we use `GLSL` [11] as GPU language. The manifold harmonics filters actually require a single fragment shader, which computes the filtering F_φ for each coordinate x, y, z, together with a render-to-vertex-buffer mechanism [1]. We also tested the same process in the vertex shader to avoid the two-pass rendering, however the texture fetch in the vertex shader is much slower and gives less than half frame per seconds.

Data textures. The manifold harmonics is sent to the GPU as textures: a texture $\tilde{x}\tilde{y}\tilde{z}$ containing the harmonic amplitudes $\tilde{x}(k), \tilde{y}(k), \tilde{z}(k)$ of the original mesh, a texture $\mathsf{d}_{\mathsf{xyz}}$ containing the sum of high frequencies contributions for each coordinate (section 3(c)) and a texture H_k containing the manifold harmonics eigenvectors. The filter φ must be sent to the GPU at each gallery element of each frame. Since the computed φ has a smaller size than the sound frequencies $\tilde{f}$ and the t and a vectors, we compute φ on the CPU and send it as texture ϕ.

Texture storage. All the textures are stored using 32 bits floating point numbers to keep the precision of the vertices coordinates. Since the number of vertices of the mesh is usually larger than the maximal texture size for 1D textures, we use two texture coordinates in $\{0 \ldots \lceil\sqrt{n}-1\rceil\}$ as vertex indexes. The high frequency contributions $\mathsf{d}_{\mathsf{xyz}}$ are stored as a 2D RGB texture of size $\lceil\sqrt{n}\,\rceil \times \lceil\sqrt{n}\,\rceil$ containing the coordinates.

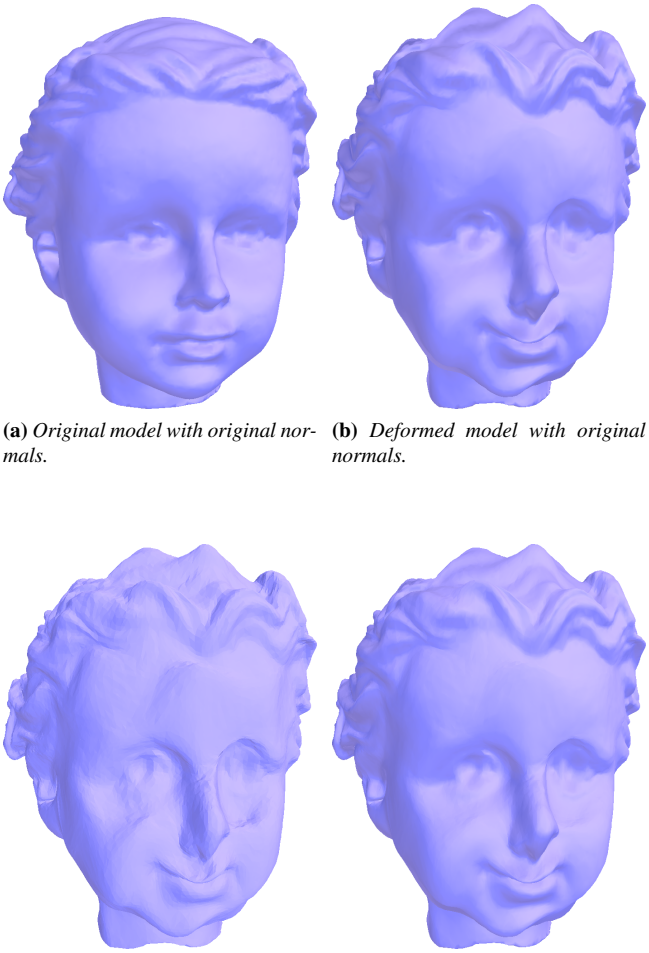
```
uniform sampler1D x̃ỹz̃;
uniform sampler2D d_xyz;
uniform sampler3D H_k;
uniform sampler1D φ;
uniform float δk;

void main() {
    vec3 texcoord = gl_TexCoord[0].stp ;
    vec3 pos = texture2D(d_xyz,texcoord.st).xyz ;
    for( float k=0.0; k ≤ 1.0; ) {
      texcoord.p = k ;
      vec4 H = texture3D(H_k, texcoord);
      vec4 f = texture1D(φ, k);
      vec3 x̃ỹz̃_0 = texture1D(x̃ỹz̃, k).xyz ; k += δk ;
      vec3 x̃ỹz̃_1 = texture1D(x̃ỹz̃, k).xyz ; k += δk ;
      vec3 x̃ỹz̃_2 = texture1D(x̃ỹz̃, k).xyz ; k += δk ;
      vec3 x̃ỹz̃_3 = texture1D(x̃ỹz̃, k).xyz ; k += δk ;
      pos += f[0] * H[0] * x̃ỹz̃_0 + f[1] * H[1] * x̃ỹz̃_1 +
             f[2] * H[2] * x̃ỹz̃_2 + f[3] * H[3] * x̃ỹz̃_3 ;
    }
    gl_FragColor.rgb = pos.xyz ;
}
```

Figure 10: *`GLSL` fagment shader for the harmonics filter.*

Since the number of manifold frequencies $\#k$ stored fits in a texture row, the original harmonic texture amplitudes $\tilde{x}\tilde{y}\tilde{z}$ are stored as a 1D RGB texture of size $\#k$, containing the $\tilde{x}, \tilde{y}, \tilde{z}$ components.

Finally, the scalar data H_k and ϕ of the manifold harmonics eigenvector n coordinates and the filter can be stored in the RGBA components to optimize space: ϕ is then a $\lceil\frac{\#k}{4}\rceil$ 1D and H_k an $\lceil\sqrt{n}\,\rceil \times \lceil\sqrt{n}\,\rceil \times \lceil\frac{\#k}{4}\rceil$ 3D RGBA textures.

Fragment shader for the filter. When all the above textures are bound, the rendering of a single square of size $\lceil\sqrt{n}\,\rceil \times \lceil\sqrt{n}\,\rceil$ will call the fragment shader for each of the vertex index and compute the new vertex positions as frame color (Figure 10). The fragment shader renders to a frame buffer containing the filtered vertex coordinates, which is then copied to the vertex buffer inside the GPU [1]. The shader receives a uniform variable which is the normalized increment $\delta k = \frac{1}{4(\#k-1)}$ for iteration inside normalized texture coordinates.



(a) *Original model with original normals.* **(b)** *Deformed model with original normals.*

(c) *Deformed model with deformed, flat normals.* **(d)** *Average normals with* 40% *deformed normals.*

Figure 11: *Normal enhancement for the deformed model.*

(b) Complementary effects

Normal enhancement. The previous method updates the vertex positions, but not the normal. Since a second render-to-vertex-buffer would be too costly, we use a geometry shader that computes, for each triangle, a constant normal. This normal is used in a per-pixel lighting via fragment shader. However, per triangle normal leads to flat shading. To obtain smoother result, we average, in the geometry shader, the triangle normal with the original vertex normal (Figure 11).

Beat detection. Until now, the whole deformation of the mesh is seen from a single point of view with a constant lighting. We propose to use those degrees of freedom to transpose global sound feature, such as beat. We implemented a simple beat detection [21], and at each detected beat we randomly choose to rotate the model or the light positions.

Harmonic mapping re-use. The gallery interface allows to quickly navigate between all the possible harmonic mappings within our proposed design. If a harmonic mapping gives a very exciting effect, it would be nice to be able to re-use it on other models. The main obstacle is that the number of manifold frequencies $\#k$ may differ from model to model. We can work around this problem by normalizing the image values of t to a constant interval $[0,1]$: $\bar{t}(\omega) = \frac{t(\omega)}{\#k}$, and adapt the definition of $\Phi_{t,a}$ to $\Phi_{\bar{t},a}(\tilde{f})(k) = a(k) \cdot \left(\sum_{\omega \in \bar{t}^{-1}(\{\frac{k}{\#k}\})} \tilde{f}(\omega)\right) + 1$.

(c) Implementation details

We used the Scalable Library for Eigenvalue Problem Computations (`SLEPc`) [8] software package to compute the first $\#k$ manifold harmonics eigenvalues and eigenvectors. We use the Compact Half Edge [13] data structure to represent the model mesh. The proposed shaders require an `OpenGL` 2.x compatible graphic card [11]. Finally, we use `FFmpeg` [2] for sound decompression and `OpenAL` [9] for stereo sound rendering in a separate thread. A nice tutorial for such sound configuration can be found at `kcat.strangesoft.net/openal.html`.

7 Results

We experimented the proposed filter design with gallery interface to check the feasibility of such approach. The actual validation of the interface is beyond the scope of this paper. However, our proposal is able to provide an animated gallery interface synchronized with sound in real time.

Performance. We first compare the CPU implementation of manifold filters [26] with our GPU implementation. Since the problem fits well for streaming process, we expect the GPU implementation to outperform the CPU counterpart (Table 1, comparing a single mesh deformation running on CPU with 6 and 12 deformations running on GPU). Furthermore, we validated that the GPU implementation supports real-time rendering to keep synchronization with the sound. Those experiments allow estimating the appropriate gallery size for a given graphics hardware (Table 1). We conclude that for models with around 50,000 vertices, a gallery size allowing interaction would be between 6 and 12 on an iMac with GeForce 130 with 48 cores running at 500MHz.

Music visualization. We use our music visualization for deforming different models in real-time (Figure 13 and the accompanying video). Since the music is decoded and analyzed on the CPU, the combination of sound does not alter the performance of the gallery. We introduce a callback that update the filter every 50 milliseconds, and the rendering is done following the rendering cycles, so that even with large galleries that would impact the real-time rendering, the sound playing does not stop. Finally, we add a parameter $m \in [0,1]$ to control how smoothly the frequencies are passed to the mesh (frequency decay): the sound frequency amplitudes $\tilde{f}(\omega)$ passed to the filter are continuously averaged by $\tilde{f}_{new}(\omega) = w \cdot \tilde{f}_{old}(\omega) + \tilde{f}(\omega)$. For very rhythmic music, this avoids flickering effects on the mesh (Figs. 1, 9 and 12, and the accompanying video).

The corresponding work was published in The Visual Computer.

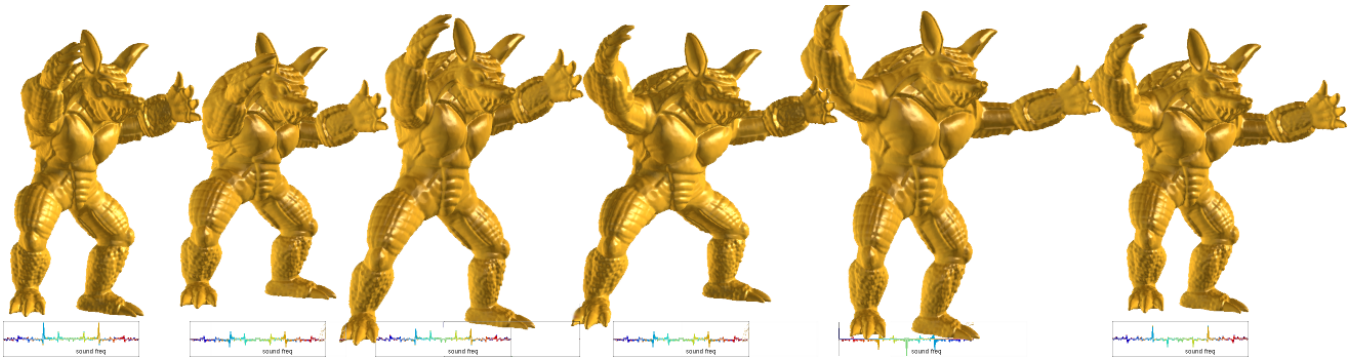


Figure 12: *Frames of rock music visualization using the armadillo model.*

Parameters list. All together, the user sets the filter parameter t and a by clicking on the gallery inerface, and control independently the frequency decay m, the normal blending, and the symmetry control μ. The frequency cut and number of gallery element is set to optimize the GPU performance according to Table 1, and the symmetry detection parameters are fixed to the values indicated in the text: $n_k = 15$, $c = 10$.

Limitations. The GPU implementation allows real-time animated galleries, but it prevents complex processing or further control on the deformed mesh. In particular, it does not permit to directly use quality measure or more advanced interface such as intelligent galleries [27]. The proposed method generates exciting animation on top of a given music. However, we used a very raw sound frequency, beat and symmetry analysis, which can be enhanced to get more correlated effects. Several complementary effects, in particular on the normal [16] and on the mesh textures, can improve our visualization.

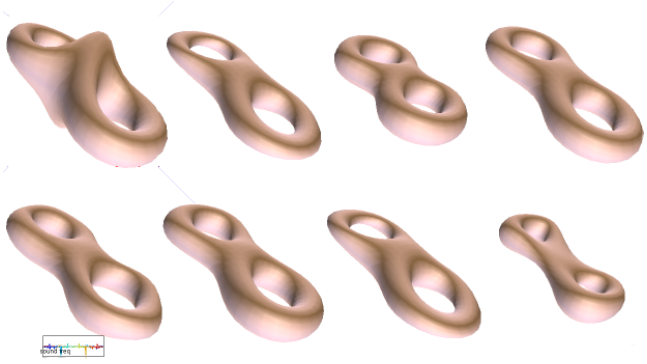


Figure 13: *Gallery of the eight model on world music.*

model	verts n	freqs $\#k$	pre-process *secs*	CPU S=1 *fps*	GPU S=6 *fps*	GPU S=12 *fps*
heart	622	256	4	99.7	322.1	203.7
eight	766	256	3	97.7	342.1	213.7
duck	2 108	256	7	52.7	208.3	123.9
spring	4 695	256	116	12.7	71.4	41.4
dinosaur	14 054	533	76	4.7	28.3	13.9
octa	15 136	529	27	4.1	26.8	13.2
octopus	20 351	546	132	3.0	21.4	7.1
alien	24 988	540	235	3.4	22.4	12.1
david	24 988	804	119	2.6	14.8	7.3
cat	30 059	271	381	9.6	45.7	23.0
gargoyle	30 059	1052	99	2.1	13.7	6.9
bunny	34 834	1070	479	1.7	12.1	5.8
buste	37 874	1075	255	1.5	13.4	3.4
blooby	42 432	1065	558	1.5	12.5	7.5
egea	63 739	275	724	0.9	7.2	3.2
head	65 002	1607	751	0.5	8.7	4.3
armadilo	86 488	2376	1 052	0.2	10.3	3.4
cow	97 803	256	1 082	0.1	9.1	3.1

Table 1: Performance tests: all models are normalized into a $[-1, 1]^3$ bounding box, and the gallery of S items is rendered in a 1024×768 window. All experiments are performed on a $3.06GHz$ processor with a GeForce GT 130 with $512MB$ of RAM. The deformation speed is measured in frame per second (*fps*), while the harmonic basis and symmetry classification pre-computation time is expressed in seconds.

8 Conclusion

In this paper we proposed a GPU implementation of manifold harmonics filters, which allows computing and rendering spectral mesh deformations at a very high rate. We applied this technique for music visualization, using animated design galleries for navigation between different visual effects. Each effect is represented as a mapping from music frequencies to manifold harmonics. We represent such mapping in a concise way to be able to couple genetic reproduction in the gallery.

Acknowledegments

This work has been partially financed by CNPq, FAPERJ and FAPEAL.

References

[1] Apple. PBORenderToVertexArray: render-to-vertex-array using FBO, PBO and VBO, 2006. developer.apple.com/mac/library/samplecode.

[2] F. Bellard. FFmpeg, 2004. www.ffmpeg.org.

[3] A. Bordignon, L. Sigaud, G. Tavares, H. Lopes, T. Lewiner and W. Morgado. Arch generated shear bands in granular systems. *Physica A: Statistical Mechanics and its Applications*, 388(11):2099 – 2108, 2009.

[4] J. Breebaart and C. Faller. *Spatial Audio Processing*. Wiley, 2007.

[5] R. W. Clough and J. Penzien. *Dynamics of Structures*. Mcgraw-Hill, 1975.

[6] H. Comstock. Radio adds third dimension. *Popular Science*, pages 104–106, 1953.

[7] W. Gardner. *3D audio using loudspeakers*. Kluwer, 1998.

[8] V. Hernandez, J. Roman and V. Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *Transactions on Mathematical Software*, 31(3):362, 2005.

[9] G. Hiebert. OpenAL programmer's guide, 2005. connect.creativelabs.com/openal.

[10] H. Jenny. *Cymatics: A Study of Wave Phenomena & Vibration*. Macromedia, 3rd edition edition, 2001.

[11] J. Kessenich. The OpenGL Shading Language v 4.0, 2010. www.opengl.org/documentation/glsl.

[12] O. Kubelka. Interactive music visualization. In *Central European Seminar on Computer Graphics*, 2000.

[13] M. Lage, T. Lewiner, H. Lopes and L. Velho. CHF: a scalable topological data structure for tetrahedral meshes. In *Sibgrapi*, pages 349–356. IEEE, 2005.

[14] B. Lévy and H. R. Zhang. Spectral mesh processing. In *Siggraph Asia Course Note*, pages 1–47. ACM, 2009.

[15] T. Lewiner, T. Vieira, A. Bordignon, A. Cabral, C. Marques, J. Paixão, L. Custódio, M. Lage, M. Andrade, R. Nascimento, S. de Botton, S. Pesco, H. Lopes, V. Mello, A. Peixoto and D. Martinez. Tuning manifold harmonics filters. In *Sibgrapi*, pages 110–117. IEEE, 2010.

[16] T. Lewiner, T. Vieira, D. Martínez, A. Peixoto, V. Mello and L. Velho. Interactive 3D caricature from harmonic exaggeration. *Computers & Graphics*, 35(3):586–595, 2011.

[17] Y. Liu, B. Prabhakaran and X. Guo. A robust spectral approach for blind watermarking of manifold surfaces. In *Multimedia and Security*, pages 43–52. ACM, 2008.

[18] J. Marks, B. Andalman, P. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Siggraph*, page 400. ACM, 1997.

[19] J. F. O'Brien, C. Shen and C. M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *Symposium on Computer animation*, pages 175–181. ACM, 2002.

[20] M. Ovsjanikov, J. Sun and L. Guibas. Global intrinsic symmetries of shapes. In *SGP*, pages 1341–1348. Eurographics, 2008.

[21] F. Patin. Beat detection algorithms, 2003. www.gamedev.net/reference/programming/features/beatdetection.

[22] A. Pentland and J. Williams. Good vibrations: modal dynamics for graphics and animation. *Siggraph*, 23(3):207–214, 1989.

[23] F. de Moura Pinto and C. M. D. S. Freitas. Two-level interaction transfer function design combining boundary emphasis, manual specification and evolutive generation. In *Sibgrapi*, pages 281–288. IEEE, 2006.

[24] G. Rong, Y. Cao and X. Guo. Spectral mesh deformation. *The Visual Computer*, 24(7):787–796, 2008.

[25] G. Taubin. A signal processing approach to fair surface design. In *Siggraph*, pages 351–358, 1995.

[26] B. Vallet and B. Lévy. Spectral geometry processing with manifold harmonics. In *Computer Graphics Forum*, volume 27, pages 251–260, 2008.

[27] T. Vieira, A. Bordignon, A. Peixoto, G. Tavares, H. Lopes, L. Velho and T. Lewiner. Learning good views through intelligent galleries. *Computer Graphics Forum (Eurographics Proceedings)*, 28(2):717–726, 2009.

[28] K. Wang, M. Luo, A. Bors and F. Denis. Blind and robust mesh watermarking using manifold harmonics. In *ICIP*, pages 3657–3660. IEEE, 2009.

[29] H.-Y. Wu, T. Luo, L. Wang, X.-L. Wang and H. Zha. 3D shape retrieval by using manifold harmonics analysis with an augmentedly local feature representation. In *VRCAI*, pages 311–313. ACM, 2009.

[30] C. Yinghui, W. Jing and L. Xiaohui. Real-time deformation using modal analysis on graphics hardware. In *Graphite*, pages 173–176. ACM, 2006.

The corresponding work was published in The Visual Computer.