Supplementary Material: The Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems

AHMAD NASIKUN, Delft University of Technology, The Netherlands and Universitas Gadjah Mada, Indonesia
KLAUS HILDEBRANDT, Delft University of Technology, The Netherlands

This document contains additional experiments concerned with the evaluation of the Hierarchical Subspace Iteration Method, which is introduced in [Nasikun and Hildebrandt 2021].

1 JUSTIFICATION OF DESIGN CHOICES

In this section, we present experiments that address the justification of our design and evaluation choices for the Hierarchical Subspace Iteration Method (HSIM).

1.1 Distance computation

The construction of the basis functions, see Equation (11) in [Nasikun and Hildebrandt 2021], requires the computation of geodesic distances. For the evaluation of HSIM, we used Dijkstra's algorithm on the weighted edge graph of the mesh using the edge lengths as weights. Since the basis functions have local support, we stop the single source Dijkstra computation when all vertices in the support of the basis function have been processed. Alternatives to Dijkstra's algorithm are the Short Term Vector Dijkstra (STVD) algorithm [Campen et al. 2013] and the Heat Method [Crane et al. 2013]. Table 1 compares timings and iteration counts obtained by using Dijkstra's algorithm, the STVD algorithm and the heat method for basis construction. One can see that the required numbers of iterations are similar for all three methods with some slight variations. Therefore, the timings for the case that Dijkstra's algorithm is used are comparable to those when the STVD algorithm is used. There is only a small overhead resulting from the additional computational effort of the STVD algorithm compared to Dijkstra's algorithm. The heat method is much slower since for each point the distance to all other points is computed instead of only in a local neighborhood. We would like to note that there are also possibilities to localize the distance computation with the heat method [Herholz et al. 2017]. This, however, would be beyond the scope of this experiment. These results illustrate our impression that the STVD algorithm or a localized version of the heat method can be used as alternatives for the basis construction. Since we did not observe any substantial advantages of STVD or the Heat method over Dijkstra's algorithm in our experiments, and to keep the method simple, we used Dijkstra's algorithm for our evaluation of HSIM.

Model (#Verts.)	Type	Acc.	Hier.	Solve		Total
				Iter	Time	
Gargoyle (85k)	Dijkstra	1e-2	2.2	F|1|1	8.4	10.5
		1e-4		F|3|3	15.8	18.0
	STVD	1e-2	4.0	F|1|1	8.9	13.0
		1e-4		F|4|4	20.8	24.8
	Heat M.	1e-2	203.6	F|1|1	10.4	214.0
		1e-4		F|4|3	22.6	226.2
Fertility (270k)	Dijkstra	1e-2	9.6	F|1|1	26.2	35.8
		1e-4		F|3|4	59.3	68.9
	STVD	1e-2	30.4	F|1|1	27.3	57.7
		1e-4		F|3|2	44.8	75.2
	Heat M.	1e-2	1218.8	F|1|1	30.8	1249.6
		1e-4		F|3|3	53.7	1272.5
Oil-pump (570k)	Dijkstra	1e-2	30.3	F|1|1	63.6	93.9
		1e-4		F|4|3	112.6	142.8
	STVD	1e-2	100.1	F|1|1	65.6	166.6
		1e-4		F|4|3	113.1	214.1
	Heat M.	1e-2	4339.4	F|2|2	88.9	4628.3
		1e-4		F|4|4	136.6	4676.0

Table 1. The timings and iteration counts for computing 100 eigenpairs on different meshes with three different schemes for approximating the geodesic distances are shown. The timings for the construction of the hierarchy are additionally listed.

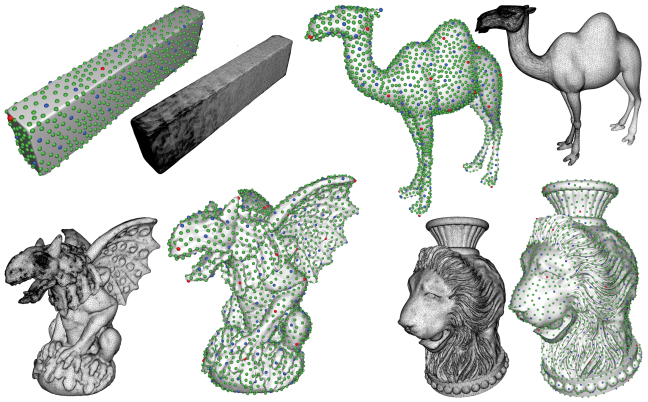


Fig. 1. Results of vertex hierarchy construction using farthest point sampling on meshes with spatially varying sampling densities are shown.

1.2 Sampling method

The nested function spaces we use for HSIM are constructed from a vertex hierarchy, which assigns a level to every vertex of the mesh. We use farthest point sampling for computing the distribution of

Authors' addresses: Ahmad Nasikun, Delft University of Technology, Department of Intelligent Systems, Delft, The Netherlands and Universitas Gadjah Mada, Department of Electrical and Information Engineering, Yogyakarta, Indonesia, A.Nasikun@tudelft.nl; Klaus Hildebrandt, Delft University of Technology, Department of Intelligent Systems, Delft, The Netherlands, K.A.Hildebrandt@tudelft.nl.

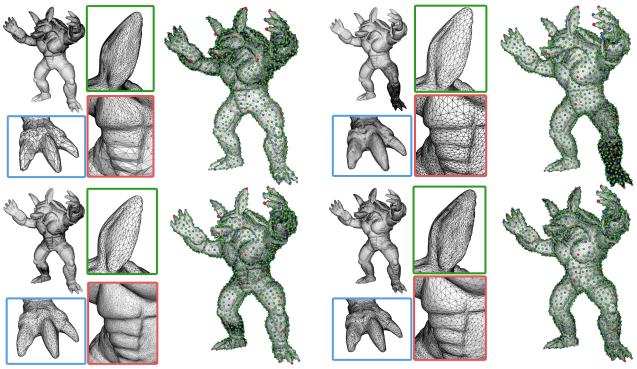
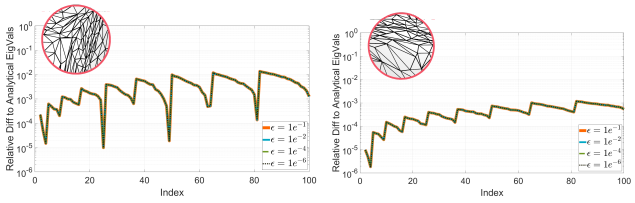


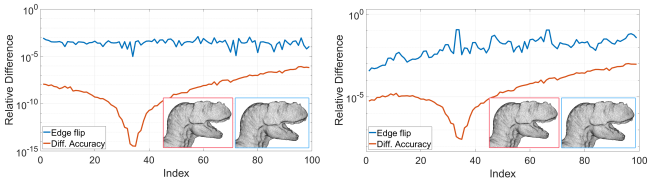
Fig. 2. Results of vertex hierarchy construction using farthest point sampling are shown for four meshes that approximate the same surface but have different spatially varying sampling densities.



(a) HSIM, 10k vertices (b) HSIM, 100k vertices

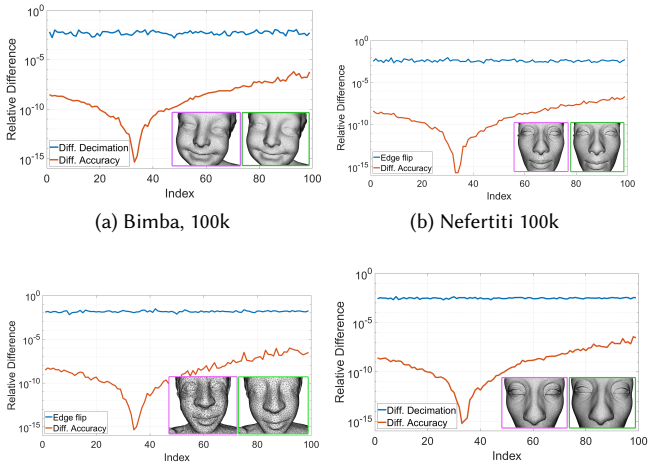
(c) HSIM, 1m vertices (d) IPM, 100k vertices

Fig. 3. Relative difference of numerical approximations of the eigenvalues of the unit sphere to the analytic solutions.



(a) Eigenvalues (b) Eigenvectors

Fig. 4. Comparisons of the relative difference of the eigenvalues and the eigenvectors between two meshes that approximate the same surface (blue graph) and solutions for different tolerance on one of the meshes (red graph).



(a) Bimba, 100k (b) Nefertiti 100k

(c) Ramses 100k (d) Ramses, 500k.

Fig. 5. For four pairs of meshes, each pair approximating the same surface, comparisons of the relative differences between the eigenvalues of the two meshes (blue graphs) and solutions for different convergence tolerance on one of the meshes (red graphs) are shown.

the vertices. In this section, we show examples of vertex hierarchies on different meshes to illustrate why we think farthest point sampling is suitable for this process. In the examples, we use meshes with spatially varying resolution, in some areas of the surfaces the triangles are much smaller than in others. Figure 1 shows vertex hierarchies on four surfaces. The hierarchies shown have four levels, color-coded in red, blue, and green, with the finest levels encompassing all vertices and not shown. It can be seen that the vertices in the different levels are distributed fairly uniformly over the surface despite the irregular mesh. In Figure 2 four more examples are shown. In this case, we show vertex hierarchies with the same numbers of vertices in each level on different meshes that approximate the same surface. As illustrated in the shown results, we consider the farthest point sampling as a suitable method to build up the vertex hierarchies for HSIM.

Alternative sampling schemes such as Poisson disk sampling [Corsini et al. 2012], which is used in [Nasikun et al. 2018] for the construction of function spaces, could be used instead of farthest point sampling. Figure 6 shows a comparison of samplings produced with farthest point sampling and Poisson disk sampling. A potential benefit of Poisson disk sampling is that the sampling step could be accelerated. Table 2 compares timings required to compute Poisson disk and farthest point samplings of different size on different meshes. On the other hand, Poisson disk sampling also has drawbacks compared to farthest point sampling. While farthest point sampling measures intrinsic distances in the surface, the Poisson disk sampling we compare to uses distances in ambient space, which may deviate when coarser samplings are computed. Another point is that farthest point samplings can be easily split into a hierarchy of levels. For Poisson disk sampling, a division would have to be computed. Lastly, the control parameter for Poisson disk sampling is the disk size rather than the number of samples and our hierarchy

Model (#Vertices)	#Eigs	#Samples	FPS	PDS
Kitten (137k)	50	1k	1.08	0.07
	250	11.7k	2.01	0.18
	1000	16.6k	2.37	0.24
Vase-Lion (200k)	50	1k	1.37	0.11
	250	14.1k	2.93	0.24
	1000	200k	3.42	0.32
Knot-Stars (450k)	50	1k	4.25	0.27
	250	21.9k	9.96	0.54
	1000	30.9k	11.98	0.68
Oilpump (570k)	50	1k	5.05	0.38
	250	23.9k	12.12	0.64
	1000	33.8k	14.59	0.77
Red-Circular (700k)	50	1k	5.09	0.46
	250	26.5k	14.69	0.71
	1000	37.5k	17.64	0.91

Table 2. Comparison of computation time of farthest point sampling (FPS) and Poisson-disk sampling (PDS).

construction prescribes the number of samples in each level. We show a visual comparison of the point samplings produced by both sampling methods in Figure 6.

1.3 Convergence tolerance

This paragraph includes further experiments related to the discussion of the convergence tolerance that we used for the evaluation of HSIM, see also Section 5 of [Nasikun and Hildebrandt 2021]. Figure 3 shows a variant of Figure 7 from [Nasikun and Hildebrandt 2021], where we consider non-regular meshes inscribed to the sphere. Figure 4 shows a variant of Figure 8 from [Nasikun and Hildebrandt 2021], using a different mesh.

In Figure 5, we show results of an additional experiment. As in Figure 4, we computed eigenpairs on two meshes with tolerances $\varepsilon = 10^{-2}$ and as reference with $\varepsilon = 10^{-8}$. We generated the meshes by simplifying one mesh with two different mesh coarsening algorithms. We used the Bimba mesh with 500k vertices to get two simplified meshes with 100k vertices each, the Nefertiti mesh with 1m vertices to obtain two simplified meshes with 100k vertices and the Ramses mesh with 750k vertices to get two meshes with 500k vertices and two meshes with 100k vertices. In Figure 5, we plot for all four pairs of meshes the differences between the reference results that are computed with a tolerance of $\varepsilon = 10^{-8}$ on both meshes and for one mesh the difference between the results for $\varepsilon = 10^{-2}$ and $\varepsilon = 10^{-8}$. For all four pairs of meshes, the difference between the reference results on the two meshes is much larger than the difference between the results for $\varepsilon = 10^{-2}$ and $\varepsilon = 10^{-8}$.

2 COMPARISONS

In this section, we show additional comparisons to alternative approaches for solving eigenproblems.

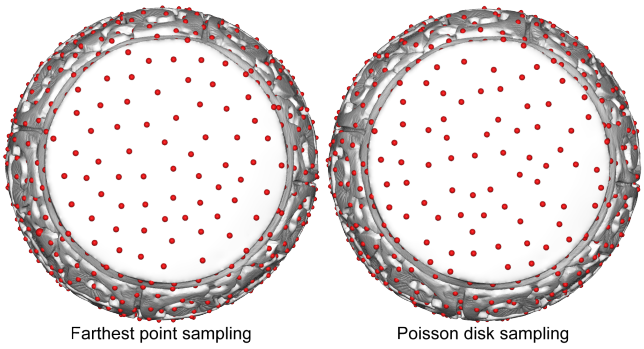


Fig. 6. Samplings we computed with farthest point sampling (left) and Poisson disk sampling (right).

2.1 Lanczos and preconditioned eigensolver

In Section 6 of [Nasikun and Hildebrandt 2021], the timings of HSIM are compared with the timings of Lanczos solvers and LOBPCG. Table 3 shows additional results that complement Table 2 in [Nasikun and Hildebrandt 2021].

Model	#Verts.	#Eigs.	HSIM	MATLAB	LOBPCG
Gargoyle	85k	50	4.2	5.1	27.1
		250	19.9	49.3	154.7
		1000	88.6	442.3	995.0
Chinese Dragon	135k	50	7.5	7.6	57.7
		250	30.5	56.3	298.0
		1000	127.5	523.3	1678.7
Dragon	150k	50	7.2	9.6	83.0
		250	36.5	65.7	325.1
		1000	143.8	795.9	2102.5
Blade	200k	50	10.5	14.1	97.9
		250	49.2	93.9	453.6
		1000	177.8	1091.9	2591.0
Fertility	240k	50	14.6	16.6	133.3
		250	90.8	121.6	678.6
		1000	236.1	1369.9	4003.0
Rocker-Arm	270k	50	17.5	22.2	135.9
		250	73.9	175.8	744.4
		1000	252.1	1537.2	4837.9
Pulley	300k	50	19.5	21.1	228.3
		250	85.3	222.1	837.3
		1000	339.5	1795.0	5416.9
Eros	400k	50	30.7	43.6	194.5
		250	127.2	267.7	1305.5
		1000	322.6	2748.4	Mem. Bound
Bimba	500k	50	29.4	31.4	236.1
		250	132.8	254.9	1255.3
		1000	569.3	3208.0	Mem. Bound
Oilpump	570k	50	41.1	46.1	310.3
		250	154.2	315.6	1864.4
		1000	690.9	3354.7	Mem. Bound
Rolling stage	680k	50	54.6	57.8	326.3
		250	197.6	386.5	2301.2
		1000	891.6	4064.1	Mem. Bound
Ramses	825k	50	49.1	62.6	458.9
		250	221.4	413.4	2339.9
		1000	1149.1	4979.2	Mem. Bound
Nefertiti	1m	50	64.0	62.8	396.2
		250	305.4	682.7	2482.2
		500	277.9	1654.5	Mem. Bound

Table 3. Comparisons of timings of HSIM, MATLAB's Lanczos solver and LOBPCG for Laplace–Beltrami eigenproblems on different meshes. Renderings of the meshes are show in Figure 7.

Model	#Eigs	Laplacian				Hamiltonian (t=0.1)				Hamiltonian (t=1.0)			
		Hier.	Solve	#Iter	Total	Hier.	Solve	#Iter	Total	Hier.	Solve	#Iter	Total
Cube (25k)	50	0.3	2.1	F|1	2.4	0.3	2.1	F|1	2.4	0.3	2.1	F|1	2.4
	250	0.6	6.7	F|1|1	7.3	0.7	6.6	F|1|1	7.3	0.7	9.2	F|3|1	9.9
	1000	0.7	67.1	F|5|2	66.8	0.8	65.9	F|5|2	66.8	0.6	67.3	F|5|2	67.9
Blade (200k)	50	2.8	7.4	F|1	10.2	2.9	7.4	F|1	10.3	2.8	10.5	F|2	13.3
	250	7.3	42.8	F|2|1	50.1	7.1	42.7	F|2|1	49.8	7.2	45.8	F|3|1	53.0
	1000	8.8	158.7	F|2|1	167.5	8.8	167.5	F|2|1	176.3	8.9	204.0	F|4|1	212.9
Bimba (500k)	50	7.9	22.6	F|1	30.5	8.0	23.8	F|1	31.7	7.7	30.9	F|2	38.7
	250	26.4	105.1	F|2|1	131.6	26.4	106.8	F|2|1	133.2	26.3	121.3	F|3|1	147.6
	1000	34.1	519.5	F|3|1	553.6	33.5	510.0	F|3|1	543.5	34.9	643.3	F|6|1	678.2

Table 4. Timings and iteration counts for Laplace–Beltrami and Hamiltonian eigenproblems are shown.

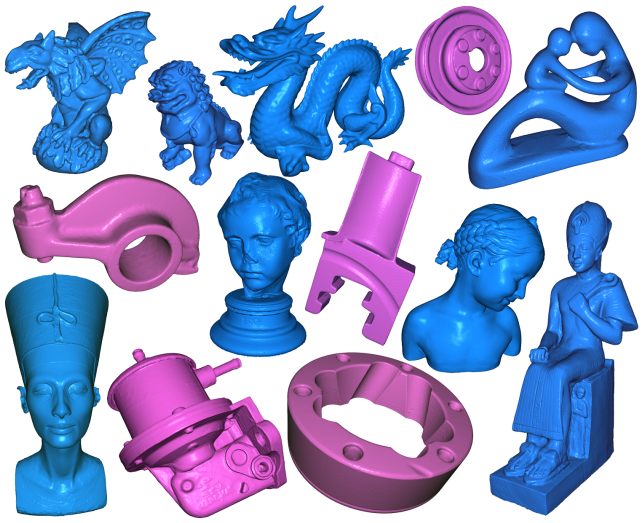


Fig. 7. Renderings of the meshes used for the comparisons listed in Table 3.

2.2 Fast Approximation

We compare HSIM with the fast approximation method from [Nasikun et al. 2018]. The approximation method has the advantage that the computation times are much shorter and storing the approximate eigenfunctions requires less memory. On the other hand, the approximation errors of [Nasikun et al. 2018] are much larger than the errors resulting from HSIM. The top row of Figure 8 shows plots of residuals of eigenpairs computed with the approximation scheme from [Nasikun et al. 2018] and compares them with the residuals from HSIM with tolerances $\epsilon = 10^{-2}$ and $\epsilon = 10^{-4}$. The residuals obtained for the approximation scheme from [Nasikun et al. 2018] are 10^0. In contrast, HSIM allows for controlling the residuals. The bottom row of the figure additionally shows the computed eigenvalues. While visually there is no difference between the two HSIM results, the eigenvalues computed with [Nasikun et al. 2018] differ significantly from the results of HSIM. We would like to note that in [Nasikun et al. 2018] it is advised to use only the first half of the computed eigenvalues. However, significant deviations can be observed in the first half as well.

3 GENERALIZATION

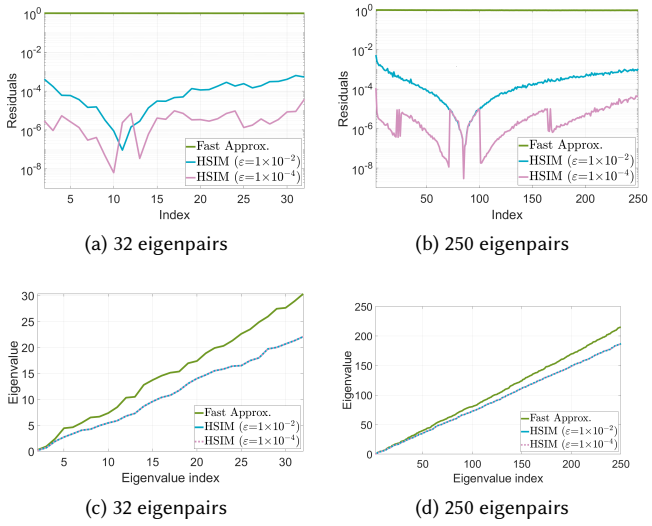
3.1 Hamiltonian operators

Our evaluation of HSIM is focused on Laplace–Beltrami eigenproblems. In this section, we consider a related operator, the Hamiltonian operator, and present some results for solving Hamiltonian eigenproblems using HSIM. For a background on Hamiltonian operators and their use in spectral analysis, we refer to [Choukroun et al. 2020]. The Hamilton operators on surfaces we consider are of the form

$$H : u \rightarrow \Delta u + Vu,$$

where Δ is the Laplace–Beltrami operator and V a scalar potential function. For our experiments, we used the scalar potential

$$V = t(\kappa_1^2 + \kappa_2^2),$$



(a) 32 eigenpairs (b) 250 eigenpairs

(c) 32 eigenpairs (d) 250 eigenpairs

Fig. 8. Top row: Plot of the residuals for the computation of the lowest 32 and 250 Laplace–Beltrami eigenvalues of the Dragon model with 150k vertices. Results for the fast approximation scheme from [Nasikun et al. 2018] and HSIM with tolerance $\epsilon = 10^{-2}$ and $\epsilon = 10^{-4}$ are shown. Bottom row: The computed eigenvalues are plotted.

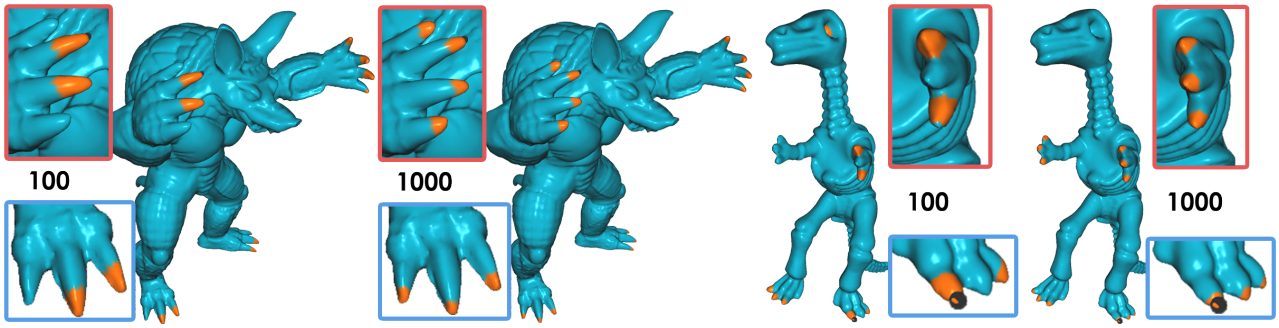


Fig. 9. Points similar to a fingertip of the armadillo mesh and to a toe tip for the dinosaur mesh are indicated by binary color-coding. The similarity is computed using the heat kernel distance. Results for heat kernel distance estimation using 100 and 1000 eigenpairs are shown.

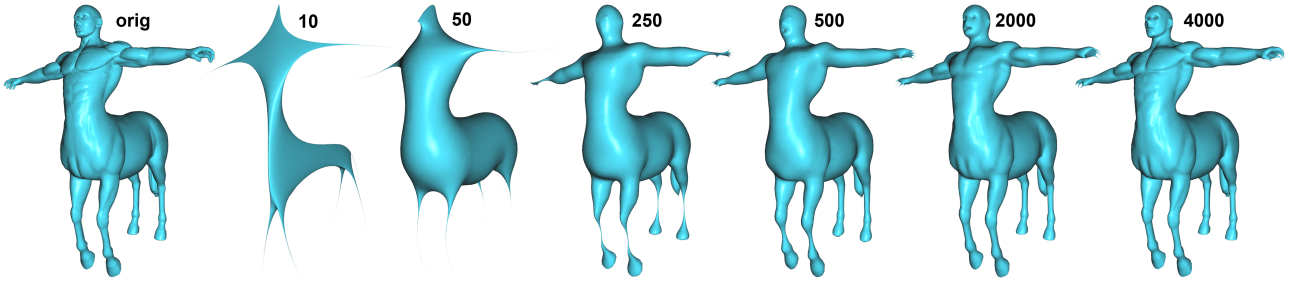


Fig. 10. Geometric reconstruction of the Centaur model (left-most) using an increasing number of Laplace–Beltrami eigenfunctions. A sufficient number of eigenfunctions is required to obtain reconstruction that preserves details of the shape.

where $t \in \mathbb{R}^{\geq 0}$ and κ_1 and κ_2 are the principal curvatures. The eigenmodes of this operator have been studied in the context of shape analysis in [Hildebrandt et al. 2010, 2012]. In contrast to the Laplace–Beltrami eigenfunctions, the eigenfunctions of this operator depend not only on the intrinsic properties of the surface but also on its extrinsic curvatures. Even for $t = 0.1$, the eigenfunctions of this operator are fundamentally different from those of the Laplace–Beltrami operator as illustrated in Figure 11. Table 4 lists iteration counts and timings for solving Hamiltonian eigenproblems for $\alpha = 0.1$ and $\alpha = 1$. As a reference, we also list the timings for the corresponding Laplace–Beltrami eigenproblem. For $\alpha = 0.1$, we obtain almost the same timings as for the Laplace–Beltrami eigenproblems and for $\alpha = 1$, we noticed in some cases an increase of the required computation time of up to 30%.

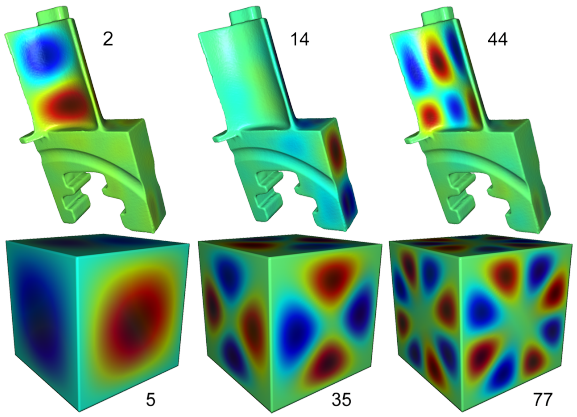
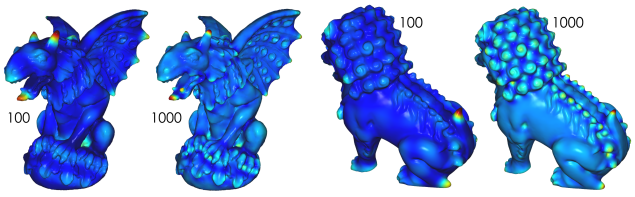


Fig. 11. Eigenfunctions of the Hamilton operator are shown.

4 APPLICATIONS

In this section, we consider methods that use the Laplace–Beltrami eigenfunctions for shape analysis and processing. We demonstrate that the methods can benefit from using a larger number of eigenfunctions. HSIM facilitates the computation of larger numbers of eigenfunctions.



(a) Gargoyle (b) Chinese Dragon

Fig. 12. Heat Kernel Signatures (HKS) computed with 100 and 1000 eigenpairs are shown.

4.1 Shape Signatures

We first consider the Heat Kernel Signature [Sun et al. 2009] as an example of a shape signature. Figure 12 shows the Heat Kernel Signature color-coded on two meshes. For both meshes, results using 100 and 1000 eigenfunctions are shown. One can see that the surface details such as the curls of the Chinese lion model are better resolved when 1000 eigenfunctions are used. As a consequence, the Heat Kernel Distance delivers better results for finding similar points on a surface when more eigenfunctions are used. Figure 9 shows results where similar points to a given point are searched. The results are shown by binary color-coding, where similar points are orange. On the Armadillo mesh, a point at a fingertip is given and on the dinosaur mesh, a point on a toe is given. It can be seen that if 1000 eigenfunctions are used, on both meshes all fingertips and toe tips are found. For 100 eigenfunctions this is not the case. Only about half of the fingertips and toes are found.

4.2 Projection

Methods such as mesh filtering [Vallet and Lévy 2008] and mesh compression [Karni and Gotsman 2000] need to project the embedding of a surface to the space spanned by the lowest n eigenfunctions. Figure 10 shows the results of this projection for the centaur mesh with different values of n ranging from 10 to 4000. One can see that the higher the number of eigenfunctions is, the more surface details are preserved. Even when 2000 eigenfunctions are used, the resulting projection is visually smoother than the original mesh.

REFERENCES

Marcel Campen, Martin Heistermann, and Leif Kobbelt. 2013. Practical Anisotropic Geodesy. *Comp. Graph. Forum* 32, 5 (2013), 63–71.

Yoni Choukroun, Alon Shtern, Alexander M. Bronstein, and Ron Kimmel. 2020. Hamiltonian Operator for Spectral Shape Analysis. *IEEE Trans. Vis. Comput. Graph.* 26, 2 (2020), 1320–1331.

Massimiliano Corsini, Paolo Cignoni, and Roberto Scopigno. 2012. Efficient and flexible sampling with blue noise properties of triangular meshes. *IEEE Transactions on Visualization and Computer Graphics* 18, 6 (2012), 914–924.

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)* 32, 5 (2013), 1–11.

Philipp Herholz, Timothy A. Davis, and Marc Alexa. 2017. Localized solutions of sparse linear systems for geometry processing. *ACM Trans. Graph.* 36, 6 (2017), 183:1–183:8.

Klaus Hildebrandt, Christian Schulz, Christoph von Tycowicz, and Konrad Polthier. 2010. Eigenmodes of surface energies for shape analysis. In *Advances in Geometric Modeling and Processing (Lecture Notes in Computer Science, Vol. 6130)*. Springer, 296–314.

Klaus Hildebrandt, Christian Schulz, Christoph von Tycowicz, and Konrad Polthier. 2012. Modal shape analysis beyond Laplacian. *Computer Aided Geometric Design* 29, 5 (2012), 204–218.

Zachi Karni and Craig Gotsman. 2000. Spectral Compression of Mesh Geometry. In *ACM SIGGRAPH*. 279–286.

Ahmad Nasikun, Christopher Brandt, and Klaus Hildebrandt. 2018. Fast Approximation of Laplace–Beltrami Eigenproblems. *Comp. Graph. Forum* 37, 5 (2018).

Ahmad Nasikun and Klaus Hildebrandt. 2021. The Hierarchical Subspace Iteration Method for Laplace–Beltrami Eigenproblems. (2021).

Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. 2009. A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *Comp. Graph. Forum* 28, 5 (2009), 1383–1392.

Bruno Vallet and Bruno Lévy. 2008. Spectral Geometry Processing with Manifold Harmonics. *Comp. Graph. Forum* 27, 2 (2008), 251–260.