Metric-Aware Processing of Spherical Imagery

Michael Kazhdan
Johns Hopkins University

Hugues Hoppe
Microsoft Research

Abstract

Processing spherical images is challenging. Because no spherical parameterization is globally uniform, an accurate solver must account for the spatially varying metric. We present the first efficient metric-aware solver for Laplacian processing of spherical data. Our approach builds on the commonly used equirectangular parameterization, which provides differentiability, axial symmetry, and grid sampling. Crucially, axial symmetry lets us discretize the Laplacian operator just once per grid row. One difficulty is that anisotropy near the poles leads to a poorly conditioned system. Our solution is to construct an adapted hierarchy of finite elements, adjusted at the poles to maintain derivative continuity, and selectively coarsened to bound element anisotropy. The resulting elements are nested both within and across resolution levels. A streaming multigrid solver over this hierarchy achieves excellent convergence rate and scales to huge images. We demonstrate applications in reaction-diffusion texture synthesis and panorama stitching and sharpening.

Keywords: Equirectangular, panoramas, Laplace-Beltrami.

1 Introduction

Spherical images are often used to represent photographed or rendered environments. Many common formats are based on the equirectangular map, which parameterizes ray directions by zenith and azimuth angles (Figure 1). This simple representation offers a number of advantages. Because the map is cylindrical, lines of constant inclination in the spherical scene are mapped to horizontal lines in the parametric domain, i.e. the map has axial symmetry. In addition, the map has a simple analytic description that is globally defined and infinitely differentiable (except at the poles). And unlike the conformal Mercator projection, the equirectangular map has a compact domain, which can be discretized using a regular grid of sample points or finite elements.

The equirectangular representation does have drawbacks. One limitation is that a uniform grid in the parametric domain oversamples the sphere near the poles. Fortunately, the situation is not so dire because no region is undersampled; the nonuniformity only contributes an over-abundance of samples near the poles. Thus, as long as the spherical signal is properly bandlimited during analysis and discretization, the reconstruction can have uniform appearance and be free of aliasing artifacts.

The more significant obstacle is that the distortion near the poles complicates signal processing. Indeed, taking the metric into account when constructing the Laplace-Beltrami operator (the extension of the Laplacian to non-Euclidean domains) results in a poorly conditioned system. For applications that require the solution of a Poisson-like system (e.g. diffusion, wave-propagation, gradient-domain image processing, etc.), this leads to unacceptably slow convergence, even with conventional multigrid techniques.

Figure 1: *Equirectangular representation of a spherical panorama.*

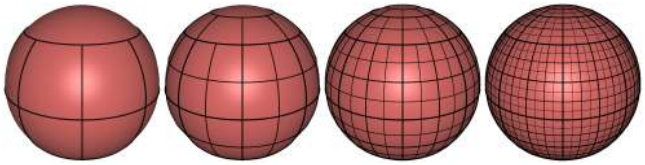


Figure 2: *Our adapted grid hierarchy is regularly spaced along meridians but adaptively spaced across circles of latitude. The multiresolution refinement (left-to-right) supports a set of nested finite elements, which enable an efficient, metric-aware multigrid solver.*

One can try to address this convergence problem by using a spherical parameterization with bounded distortion, but this presents additional difficulties. There are two strategies. (1) The first is to assume that the distortion is small enough, and define a linear system that is metric-unaware by assigning equal weight to each element of the parameterization; unfortunately the inaccuracies lead to undesirable artifacts for common signal-processing applications. (2) The other strategy is to define a metric-aware system, relying on the bounded distortion to provide good conditioning. However, a critical aspect is that the geometry of element neighborhoods is spatially varying, so it is necessary to compute and store the coefficient masks of the discretized Laplace-Beltrami operator at each grid node. Performing this over high-resolution grids is unwieldy.

In this work we introduce a new representation that builds on the simple analytic description of the equirectangular map, thereby inheriting the advantages of differentiability, axial symmetry, and regularity. To address the poor conditioning due to the parameteric distortion at the poles, we adaptively refine the domain grid, generating spherical elements with bounded anisotropy (Figure 2).

By retaining the underlying grid structure, we obtain finite-element basis functions that nest within and across the multiresolution levels. Thus, the representation is ideally suited for streaming multigrid processing. Additionally, thanks to the axial symmetry of the adapted equirectangular grid, we need only compute and store the mask coefficients once per row of the domain grid — a big savings in both space and time. Combining these benefits, we present the first spherical solver that is both efficient and metric-aware.

We demonstrate the new representation and multigrid solver using two quite different applications: reaction-diffusion simulation (for texture synthesis) and spherical gradient-domain image processing (for panorama stitching and enhancement). We envision that the efficient spherical solver may also provide a contribution to other disciplines, where the accurate solution to a spherical Poisson equation allows for the modeling of phenomena such as weather patterns, electromagnetic potentials, and plate tectonics.

2 Related Work

Spherical Parameterization The challenge of effectively representing functions over a spherical domain has been well studied in computer graphics. Many approaches define a partition of the sphere by recursively subdividing the faces of platonic solids, e.g., cubes/octahedra [Miller and Hoffman 1984; Górski et al. 2005; Wan et al. 2007; Fu et al. 2009] and icosahedra [Fekete 1990; Schröder and Sweldens 1995a]. The use of a platonic solid as a base mesh lets the final spherical partition inherit the associated symmetries, and the subdivision of the base faces ensures the topological regularity of the final facets.

Spherical Processing Metric-aware spherical processing is generally difficult. Górski et al. [2005] use the isolatitude property of HEALPix samples (also present in the isocube map [Wan et al. 2007]) to achieve efficient integration over the sphere, thus enabling spherical harmonic transforms. Schröder and Sweldens [1995b] demonstrate smoothing and sharpening of environment maps using spherical wavelets; they compute local integrals using numerical quadrature. Laplacian processing requires the evaluation of derivatives over neighborhoods of elements, and the geometry of these neighborhoods generally varies. Kazhdan et al. [2010] adapt a so-called TOAST parameterization to perform Laplacian processing of spherical images, but resort to a metric-unaware solver.

Metric-Aware Signal Processing Turk [1991] computes reaction-diffusion over an irregular sampling of a triangle mesh, adjusting diffusion based on induced Voronoi regions. Witkin and Kass [1991] simulate reaction-diffusion over a surface patch by accounting for the parameterization Jacobian in the diffusion algorithm. Similarly, Stam [2003] simulates fluid flow over an arbitrary subdivision surface by computing the spatially varying metric on the subdivided mesh. Texture synthesis schemes such as [Ying et al. 2001; Lefebvre and Hoppe 2006] use the varying metric to create predistorted texture that looks correct when mapped on the surface.

Multigrid Solvers It is well known that anisotropy leads to poor multigrid convergence [Briggs et al. 2000]. If the anisotropy is axis-aligned and spatially homogeneous, an effective solution is semicoarsening, whereby the grid is only coarsened in one direction [Schaffer 1998]. Our adapted equirectangular grid is related to semicoarsening in that we coarsen the rows independently. However, our approach is more general because we apply different coarsening to different rows, resulting in a domain that is not restricted to have a tensor-product structure.

Algebraic multigrid [Stüben 2001] is a more general framework for handling anisotropy. Its adaptive clustering leads to irregular grids and thus requires storing matrices explicitly. In comparison, our multigrid algorithm maintains the regularity of the system and does not require constructing the system matrices.

3 Motivation and Approach

Spherical Poisson Problem Many interesting image processing operations can be expressed in the gradient domain, including image stitching, dynamic range reduction, removal of shadows or reflections, and gradient-based sharpening [Agrawal and Raskar 2007]. The basic strategy is to extract gradient fields from one or more images, modify the gradient values, and find the new image that best approximates the desired gradients.

Using the notation of [Kazhdan et al. 2010], in the spherical setting the user provides a vector field $\tilde{V} : S^2 \to TS^2$ specifying the desired spherical gradients of an image, a scalar field $\tilde{W} : S^2 \to \mathbb{R}$ specifying the desired values, and interpolation weights α, β, and the system finds the best-fitting function $\tilde{U} : S^2 \to \mathbb{R}$, minimizing:

$$\min_{\tilde{U}} \int_{S^2} \alpha \|\nabla \tilde{U} - \tilde{V}\|^2 + \beta (\tilde{U} - \tilde{W})^2. \tag{1}$$

The function $\tilde{U}$ is obtained by solving a screened Poisson equation [Bhat et al. 2008]:

$$(\alpha \Delta - \beta)\tilde{U} \;=\; \alpha \nabla \cdot \tilde{V} \;-\; \beta \tilde{W}\,. \tag{2}$$

Here Δ is the Laplace-Beltrami operator which accounts for the geometry of the sphere. This continuous formulation is discretized by introducing a set of finite element basis functions and applying the Galerkin method (Section 4). The result is a linear system

$$(\alpha L - \beta D)\, u = f, \tag{3}$$

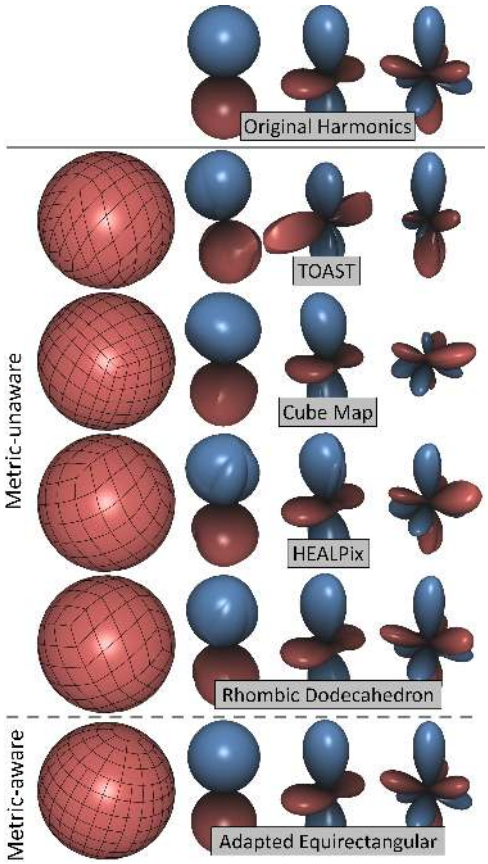
where matrix L is the discretized Laplace-Beltrami operator, D is the mass matrix, u is the vector of coefficients associated with the finite elements, and f is the vector giving the inner products of the constraint function with the finite elements.

Motivation for a Metric-Aware Solution One could use any of the spherical parameterizations introduced in the previous section to perform Laplacian processing on the sphere. Because these parameterizations bound the distortion, a metric-unaware formulation might seem adequate. (For example, in performing gradient-domain processing over an adapted TOAST parameterization, Kazhdan et al. [2010] show that results obtained using a metric-unaware solution are very similar to results obtained when incorporating the metric.) However, even for parameterizations like HEALPix [Górski et al. 2005] and Rhombic Dodecahedron [Fu et al. 2009], which are specifically designed to have low distortion, the spatial nonuniformity of the finite differences/elements introduces significant errors.

Figure 3 provides a visualization of this error in simple cases where the analytic solution is known. It considers solutions $\tilde{U} : S^2 \to \mathbb{R}$ to the Poisson equation

$$\Delta \tilde{U} = -l \cdot (l+1) \tilde{Y}_l^m, \tag{4}$$

where the constraint is a scaled version of a spherical harmonic $\tilde{Y}_l^m$ of degree l. Because these spherical harmonics are eigenvectors of the Laplace-Beltrami operator, with eigenvalues $-l \cdot (l+1)$, we expect the solutions of (4) to be the original spherical harmonics $\tilde{Y}_l^m$. However, as Figure 3 indicates, relying on the mesh connectivity alone to define a metric-unaware finite element system results in significant error. It is only when derivatives and integrals are computed with respect to the sphere metric that the solution accurately reproduces the original spherical harmonics.



	Resolution					
	$N=32$	$N=64$	$N=128$	$N=256$	$N=512$	$N=1024$
TOAST	$6.8\cdot10^{-1}$	$6.8\cdot10^{-1}$	$6.8\cdot10^{-1}$	$6.8\cdot10^{-1}$	$6.8\cdot10^{-1}$	$6.8\cdot10^{-1}$
Cube	$3.4\cdot10^{-1}$	$3.4\cdot10^{-1}$	$3.4\cdot10^{-1}$	$3.4\cdot10^{-1}$	$3.4\cdot10^{-1}$	$3.4\cdot10^{-1}$
HEALPix	$2.7\cdot10^{-1}$	$2.7\cdot10^{-1}$	$2.7\cdot10^{-1}$	$2.7\cdot10^{-1}$	$2.7\cdot10^{-1}$	$2.7\cdot10^{-1}$
Rhombic	$1.2\cdot10^{-1}$	$1.2\cdot10^{-1}$	$1.2\cdot10^{-1}$	$1.2\cdot10^{-1}$	$1.2\cdot10^{-1}$	$1.2\cdot10^{-1}$
Adaptive	$1.4\cdot10^{-3}$	$4.0\cdot10^{-4}$	$1.2\cdot10^{-4}$	$3.5\cdot10^{-5}$	$9.7\cdot10^{-6}$	$2.6\cdot10^{-6}$

Figure 3: *Solutions to the Poisson equation* (4) *obtained using a metric-unaware solver with various spherical parameterizations, compared to the expected harmonic solutions (degree 1,2,3) in the top row, and our metric-aware solver in the bottom row. (Functions are visualized by scaling points on the unit sphere in proportion to the absolute function value and coloring the points with positive value in red and negative value in blue.) The table shows the RMS errors for the degree-3 harmonic at solution convergence.*

Our Approach In the discretization of the metric-aware finite element system, the coefficients of matrices L and D in (3) are expensive to compute because they require inner products of functions defined on the sphere. To avoid a situation in which the construction of the linear system is computationally more expensive than its solution, we seek to exploit symmetries. In particular, the axial symmetry of the equirectangular parameterization lets us amortize the cost of computing the coefficients across all finite elements at a fixed latitude. Thus, if we think of a sphere discretization as having complexity $O(N^2)$, computing the system coefficients using our partition requires only $O(N)$ work. In contrast, using a partition based on subdivision of platonic solids, we could get up to 48-fold (cube/octahedron) or 120-fold (dodecahedron/icosahedron) symmetry, still requiring $O(N^2)$ computation to set up the system. (For further analysis, please see the discussion in Appendix A.)

It is important to note that although our discretization of the sphere is not isotropic, the anisotropy is accounted for by defining all system coefficients in terms of integrals over the sphere. Because the solver is metric-aware, it converges to the correct solution as the resolution N is increased, independent of the axis of the equirectangular parameterization.

4 Geometry-Aware Finite Element Solution

In this section we describe the basic approach using a regular tessellation of the equirectangular domain. Although this tessellation results in a poorly conditioned solver, the design of the system incorporates many of the basic building blocks that we leverage in our well-conditioned adapted discretization, described in Section 5.

The main tasks in defining a metric-aware spherical solver are to choose the set of spherical finite elements that span the (approximating) subspace of functions, and to compute the coefficients of the linear system using the sphere's metric. We first show how an equirectangular parameterization can be used to transform B-splines defined on a planar domain into B-spline-like functions on the sphere. Using these functions, we derive the coefficients of the linear system. Finally, we describe how the nesting of B-splines can be leveraged to define a multigrid solver for the spherical system and briefly describe the implications for solving large systems.

4.1 Choosing the Finite Elements

The equirectangular parameterization $\Phi : [0,\pi]\times[0,2\pi] \to S^2$ maps a rectangle domain to the sphere as

$$\Phi(\theta,\varphi) = (\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta)\,.$$

To establish the subspace of spherical functions, we first create a set of finite element basis functions over the domain and then use the parameterization inverse to pull-back the functions to the sphere, as shown in Figure 4.

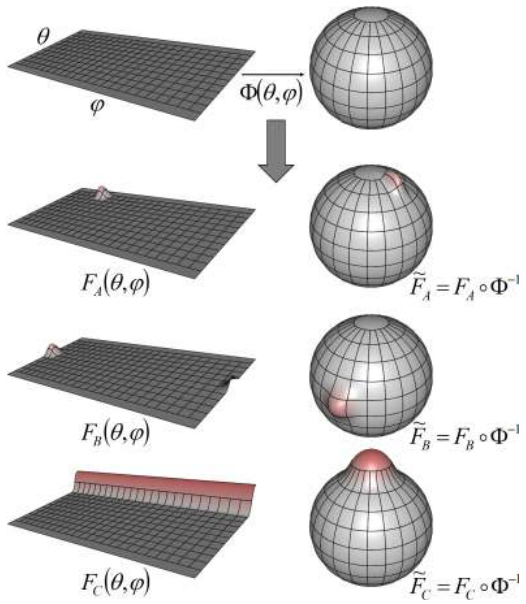


Figure 4: *Using the equirectangular parameterization (top), we define our spherical finite elements (right) as the pull-back of planar B-splines (left) through the parameterization inverse.*

We partition the planar (θ, φ) domain into a grid of cells, such that cells $[0, N-1] \times [0, 2N-1]$ cover the desired range $[0, \pi] \times [0, 2\pi]$. Let $B^N_{i,j}(\theta, \varphi)$ be the biquadratic B-spline centered on cell (i, j). Because B-splines form a partition of unity, their sum (over the slightly extended grid $[-1, N] \times [-1, 2N]$) is the constant function with value one:

$$\sum_{i \in [-1,N]} \sum_{j \in [-1,2N]} B^N_{i,j}(\theta, \varphi) = 1 \quad \text{for} \quad \theta \in [0, \pi], \varphi \in [0, 2\pi].$$

Second-order B-splines have many benefits, including a tensor-product structure that simplifies computation, continuous first derivatives, a small support that results in sparse linear systems, and a nesting structure compatible with a multigrid solver.

In the domain interior, we set the test function to equal the B-spline, $F_{i,j} = B_{i,j}$, and define the spherical function as its pull-back $\tilde{F}_{i,j} = F_{i,j} \circ \Phi^{-1}$, illustrated by functions F_A and $\tilde{F}_A$ in Figure 4.

Next we create test functions near the domain boundaries in accordance with the spherical topology. For the spherical function to be 2π-periodic in the φ direction, we define boundary functions as $F^N_{i,0} = B^N_{i,0} + B^N_{i,2N}$ and $F^N_{i,2N-1} = B^N_{i,2N-1} + B^N_{i,-1}$, illustrated by F_B and $\tilde{F}_B$ in Figure 4. And, for the spherical function to have constant value and a well-defined (zero) derivative at each of the two poles, we create special polar functions as follows. At the north pole ($\theta = 0$), we sum the B-spline functions along the two grid rows $i = -1$ and $i = 0$, thus forming

$$F^N_0(\theta) = B^N_{-1}(\theta) + B^N_0(\theta) \quad \text{with} \quad B^N_i(\theta) = \sum_{j \in [-1,2N]} B^N_{i,j}(\theta, \varphi),$$

illustrated by F_C and $\tilde{F}_C$ in Figure 4. The function at the south pole is defined similarly as $F^N_{N-1}(\theta) = B^N_{N-1}(\theta) + B^N_N(\theta)$. Thus, the dimension of our discretization at resolution N is $(N-2) \times 2N + 2$.

In defining these functions, our approach is inspired by previous work in polar subdivision (e.g. [Karčiauskas and Peters 2006]) which supports the processing of quad-dominant meshes by using polar caps to handle extraordinary vertices.

4.2 Defining the Linear System

To form the discretized linear system (3), for each pair of spherical elements $\tilde{F}_I(p) = F_I \circ \Phi^{-1}(p)$ and $\tilde{F}_J(p) = F_J \circ \Phi^{-1}(p)$ we must compute the coefficients:

$$\begin{aligned} D_{I,J} &= \left\langle \tilde{F}_I, \tilde{F}_J \right\rangle = \int_{S^2} \tilde{F}_I(p) \cdot \tilde{F}_J(p)\, dp \\ L_{I,J} &= \left\langle \Delta \tilde{F}_I, \tilde{F}_J \right\rangle = \int_{S^2} \Delta \tilde{F}_I(p) \cdot \tilde{F}_J(p)\, dp. \end{aligned}$$

These integrals can be expressed in terms of the functions F_I and F_J defined over the parameterization domain:

$$\begin{aligned} D_{I,J} &= \int_0^\pi \int_0^{2\pi} F_I(\theta, \varphi) F_J(\theta, \varphi) \sin\theta \, d\varphi \, d\theta \\ L_{I,J} &= -\int_0^\pi \int_0^{2\pi} \left(\sin\theta \frac{\partial F_J}{\partial \theta} \frac{\partial F_I}{\partial \theta} + \frac{1}{\sin\theta} \frac{\partial F_J}{\partial \varphi} \frac{\partial F_I}{\partial \varphi} \right) d\varphi \, d\theta. \quad (5) \end{aligned}$$

Because the functions F_I are tensor products of B-splines, computing the system coefficients reduces to integrating products of polynomials with trigonometric functions. For integrals of the form $\int x^n \sin(x) dx$ we can obtain a closed-form value using the identities:

$$\begin{aligned} \int x^n \sin x \, dx &= -x^n \cos x + n \int x^{n-1} \cos x \, dx \\ \int x^n \cos x \, dx &= \quad x^n \sin x - n \int x^{n-1} \sin x \, dx. \end{aligned}$$

For the cosecant term arising in the Laplace-Beltrami coefficients, $\int x^n / \sin(x) dx$, we can compute the integral by evaluating the polylogarithm function (e.g. using the Cephes Library [Cephes 1995]).

4.3 Nesting and Multigrid

The B-spline basis has a nesting property that enables efficient multigrid solutions. In this section, we derive the prolongation matrix to transition between levels of a multigrid hierarchy.

In 1D, a B-spline at resolution N can be expressed as the linear combination of B-splines at resolution $2N$. For second-order B-splines, the nesting relationship is:

$$B^N_i(x) = \eta_0 B^{2N}_{2i-1}(x) + \eta_1 B^{2N}_{2i}(x) + \eta_2 B^{2N}_{2i+1}(x) + \eta_3 B^{2N}_{2i+2}(x),$$

with interpolation weights $\eta_0 = \eta_3 = 0.25$ and $\eta_1 = \eta_2 = 0.75$.

For 2D tensor-product B-splines, the 2D interpolation weights are simply the products of the 1D interpolation weights:

$$B^N_{i,j}(x, y) = \sum_{k,l=0}^{3} \eta_k \eta_l B^{2N}_{2i-1+k, 2j-1+l}(x, y).$$

In the context of our spherical elements, the linearity of the pull-back operator implies that we can continue to use the nesting structure of B-splines to express coarser spherical elements $\tilde{F}^N$ as the linear combination of finer spherical elements $\tilde{F}^{2N}$. One important aspect is to show that this nesting property is satisfied for the special polar elements. The finite element $F^N_0(\theta)$ associated with the north pole can be expressed as

$$\begin{aligned} F^N_0(\theta) = B^N_{-1}(\theta) + B_0(\theta) = {} & \eta_0 B^{2N}_{-3}(\theta) \; + \\ & \eta_1 B^{2N}_{-2}(\theta) \; + \\ & \eta_1 B^{2N}_{-1}(\theta) \; + \; \eta_0 B^{2N}_{-1}(\theta) + \\ & \eta_0 B^{2N}_{0}(\theta) \; + \; \eta_1 B^{2N}_{0}(\theta) + \\ & \qquad \eta_1 B^{2N}_{1}(\theta) + \\ & \qquad \eta_0 B^{2N}_{2}(\theta) \quad . \end{aligned}$$

Using the fact that $\theta \in [0, \pi]$ and that the second-order B-spline $B_i(x)$ is supported in the range $[i-1, i+2]$, the expression for the north-pole element becomes:

$$\begin{aligned} F^N_0(\theta) &= (\eta_0 + \eta_1) \left(B^{2N}_{-1}(\theta) + B^{2N}_0(\theta) \right) + \eta_1 B^{2N}_1(\theta) + \eta_0 B^{2N}_2(\theta) \\ &= F^{2N}_0(\theta) + \eta_1 B^{2N}_1(\theta) + \eta_0 B^{2N}_2(\theta). \end{aligned}$$

Finally, since the B-splines form a partition of unity, we get:

$$\begin{aligned} F^N_0(\theta) &= F^{2N}_0(\theta) + \left(\eta_1 B^{2N}_1(\theta) + \eta_0 B^{2N}_2(\theta) \right) \sum_{j=-1}^{4N} B^{4N}_j(\varphi) \\ &= F^{2N}_0(\theta) + \sum_{j=0}^{4N-1} \left(\eta_1 F^{2N}_{1,j}(\theta, \varphi) + \eta_0 F^{2N}_{2,j}(\theta, \varphi) \right). \end{aligned}$$

Thus, the element associated with the north pole at resolution N can be expressed as a linear combination of elements at resolution $2N$. An analogous argument holds at the south pole.

Gathering the interpolation weights into the matrix P^{2N}_N we obtain the prolongation operator that can be used to transition between the different resolutions of a multigrid solver.

Note that by construction, if L^N (resp. D^N) denotes the Laplace-Beltrami matrix (resp. mass matrix) at resolution N, we have:

$$L^N = \left(P_N^{2N}\right)^T L^{2N} P_N^{2N} \qquad \text{and} \qquad D^N = \left(P_N^{2N}\right)^T D^{2N} P_N^{2N}.$$

Thus, the system matrices defined in Equation (5) automatically satisfy the Galerkin condition and we can compute the matrices for the coarser resolutions directly, without having to perform the sparse matrix multiplications required by an algebraic multigrid solver.

4.4 Efficient System Solution

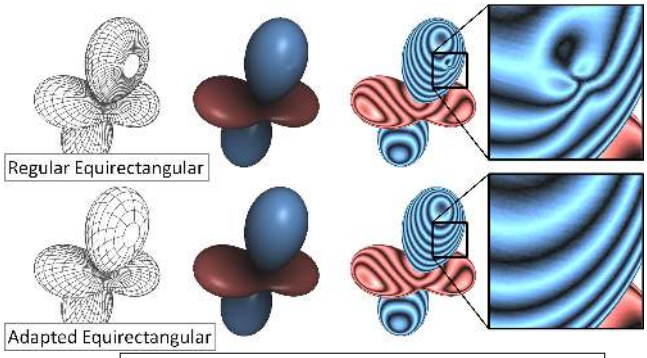
Creating the linear system using an equirectangular parameterization provides two efficiency advantages.

First, due to rotational symmetry, for any pair of spherical elements $\tilde{F}_{i,j}$ and $\tilde{F}_{i',j'}$, and any longitudinal offset δ, we have:

$$\langle \tilde{F}_{i,j}, \tilde{F}_{i',j'} \rangle = \langle \tilde{F}_{i,j+\delta}, \tilde{F}_{i',j'+\delta} \rangle$$
$$\langle \Delta\tilde{F}_{i,j}, \tilde{F}_{i',j'} \rangle = \langle \Delta\tilde{F}_{i,j+\delta}, \tilde{F}_{i',j'+\delta} \rangle.$$

Thus, we can re-use the computation performed in defining the row of the system matrix corresponding to element $F_{i,0}$ to set the values for all other elements $F_{i,j}$. This is crucial for large datasets, where the system matrix becomes too large to store and computation of the system coefficients at high precision would overwhelm the solver time if it had to be performed for every element independently.

The second advantage is that the resulting matrix L is banded, because elements in one image row only overlap elements in nearby rows. Specifically, for any $|i-i'|>2$, elements $F_{i,j}$ and $F_{i',j'}$ are always disjoint. Thus, a θ-major data ordering lets us implement a streaming solver akin to that described in [Kazhdan and Hoppe 2008]. Such a solver requires only a few rows of the grid to be resident in memory at any given time — providing a way to solve a system of complexity $O(N^2)$ with only $O(N)$ working memory.



	V-Cycles					
	0	1	2	3	4	5
Regular	$1.0\cdot10^{0}$	$8.8\cdot10^{-2}$	$7.1\cdot10^{-2}$	$6.2\cdot10^{-2}$	$5.7\cdot10^{-2}$	$5.4\cdot10^{-2}$
Adapted	$1.0\cdot10^{0}$	$4.1\cdot10^{-3}$	$3.4\cdot10^{-5}$	$1.4\cdot10^{-7}$	$2.8\cdot10^{-10}$	$7.2\cdot10^{-13}$

Residual (Degree 2)

Figure 5: *Visualizations (top) and residual norms (bottom) for the multigrid solution to the Poisson equation with degree-2 spherical harmonic constraints, using a regular equirectangular parameterization (first row) and our adapted equirectangular elements (second row). The visualizations show the parameterization alignment (left), the solution (center), and isophotes highlighting the solver imprecision near the north pole (right).*

5 Adapted Elements for Better Conditioning

While the finite element system developed in the previous section offers a multiresolution hierarchy, it is difficult to directly leverage this hierarchy in a classical multigrid solver. The challenge is that the severe anisotropy in elements near the poles hinders the convergence of traditional relaxation techniques (e.g. Gauss-Seidel).

Figure 5 demonstrates this phenomenon, providing both a visualization and residual magnitudes for the multigrid solutions to the Poisson equation, obtained when the input constraint is the degree-2 spherical harmonic from Figure 3. As these results indicate, the solver defined by the regular equirectangular parameterization fails to converge to the correct solution, and barely improves with increased number of V-cycles.

To overcome this, we use the equirectangular map, but *adaptively discretize* its domain to bound the anisotropy of the resulting elements (Figure 2). Specifically, in discretizing the sphere at resolution N we still use N circles of latitude, but partition the k-th circle of latitude into $N(k)$ intervals (rather than always $2N$), choosing $N(k)$ so that it is a power of two and the lengths of the longitudinal and latitudinal arcs are within a factor of $\sqrt{2}$. Using the fact that the circle at latitude θ has length $2\pi\sin\theta$ and that the longitudinal arcs have length π/N, we set $N(k)$ to be the power of two that satisfies:

$$\frac{\pi}{N}\frac{1}{\sqrt{2}} \leq \frac{2\pi\sin(\theta_k)}{N(k)} < \frac{\pi}{N}\sqrt{2} \qquad \text{with} \qquad \theta_k = \frac{k+0.5}{N}\pi.$$

The motivation is similar to that in the isocube map [Wan et al. 2007]. What distinguishes our method is that we restrict $N(k)$ to be a power of two. As a result, the geometric structure of samples across adjacent rows remains regular, allowing us to leverage the axial symmetry in efficiently defining a geometry-aware system.

Note that, as in [Grinspun et al. 2002], our implementation has the B-splines centered on the facets of the parameterization and defined independently of the size and shape of adjacent faces. Thus, the function-space does not exhibit any T-junction discontinuities.

Defining the Linear System Thanks to the nesting of B-splines, our adapted discretization defines a space of functions that is a *subspace* of the original space of functions. Thus, the matrix and constraint coefficients defined by the adapted system can also be expressed in terms of the coefficients of the non-adapted system.

The key to establishing the transition between the adapted and non-adapted system is the definition of a refinement operator R that expresses elements in the adapted system as linear combinations of elements in the non-adapted system from Section 4. Using R, the non-adapted system $(\alpha L - \beta D)\,u = f$ can be transformed into the adapted system $(\alpha\bar{L} - \beta\bar{D})\,\bar{u} = \bar{f}$, with:

$$\bar{L} = R^T L R, \qquad \bar{D} = R^T D R, \qquad \bar{f} = R^T f \qquad \text{and} \qquad u = R\,\bar{u}.$$

We define the operator R per row of the domain grid. To do this, we express an element defined by sampling a circle at resolution $N(k) = 2^c$ in terms of the elements defined by sampling the circle at resolution $2N = 2^C$. Assuming that $C \geq c$, this can be accomplished in an iterative manner:

- If $c = C$ then the refinement operator is the identity and we are done with the current row.
- Otherwise, we refine along the circle, using the weights $\{\eta_0, \eta_1, \eta_2, \eta_3\} = \{0.25, 0.75, 0.75, 0.25\}$ to express a univariate B-spline sampled at 2^c locations in terms of univariate B-splines sampled at 2^{c+1} locations and repeat the refinement of the current row with the exponent c replaced by $c+1$.

It is important that the adapted discretization still maintains the nesting property between different multigrid levels. It can be shown that this is satisfied if at all levels, the refinement of adjacent circles of latitude never differ by more than a factor of two. The only time this could be violated is near the poles and at low resolutions, and it can be shown this does not happen.

6 Results

The design of our adapted equirectangular finite-element system is motivated by several key requirements. From the theoretical perspective, our goal is to design a system that correctly incorporates information about the geometry of the sphere. From the numerical perspective, our goal is to design a multigrid solver that exhibits the convergence properties of standard multigrid solvers defined over regular (Euclidean) grids. And finally, from a practical perspective, our goal is to design a solver that is both fast and scalable, supporting image processing over large spherical panoramas.

6.1 Geometry Awareness

As discussed in Section 3, even when the parameterization of the sphere is chosen to be either area-preserving (HEALPix) or approximately area- and angle-preserving (Rhombic Dodecahedron), the obtained solution can deviate significantly from the true (geometry-aware) solution. This is demonstrated in the table in Figure 3 which gives the RMS difference between the correct (spherical harmonic) solution and the one returned by the different solvers.

Running the solvers to convergence (the residual never exceeds 5×10^{-12} for any of the solvers), we see that geometry-unaware solvers converge to the wrong solution and that this error is not reduced by increasing the sampling resolution N. In contrast, our geometry-aware solver returns a solution that approximates the correct solution even at low resolutions and the quality of the approximation improves steadily with refinement.

6.2 Multigrid Convergence

To address the anisotropy of the regular equirectangular parameterization at the poles, we have proposed adapted equirectangular elements that define an approximately uniform partition of the sphere. As the table in Figure 5 shows, our adapted elements resolve the difficulty with anisotropy, providing a multigrid solver that quickly converges to the correct solution. Furthermore, this table demonstrates that as the number of V-cycles is increased, the solver exhibits the exponential decay in residual norm typical of multigrid solvers defined over regular domains.

6.3 Reaction-diffusion processing on the sphere

To demonstrate the practical implications of geometry awareness and multigrid convergence, we use the different parameterizations to perform the semi-implicit time-stepping in a reaction-diffusion process, shown in Figure 6. (See Appendix B for details of the implementation.)

As observed by Turk [1991] and Witkin et al. [1991], failure to accommodate the geometry in a reaction-diffusion process can result in undesirable artifacts, and we see these in the synthesis of Turing's "spots" texture [1952]. For TOAST and Cube-Map, which do not preserve area, the spots have non-uniform sizes. For the (nearly) area-preserving HEALPix and Rhombic-Dodecahedron parameterizations, the spots take on elliptical shapes. By comparison, our efficient metric-aware solver obtains nicely uniform spots.

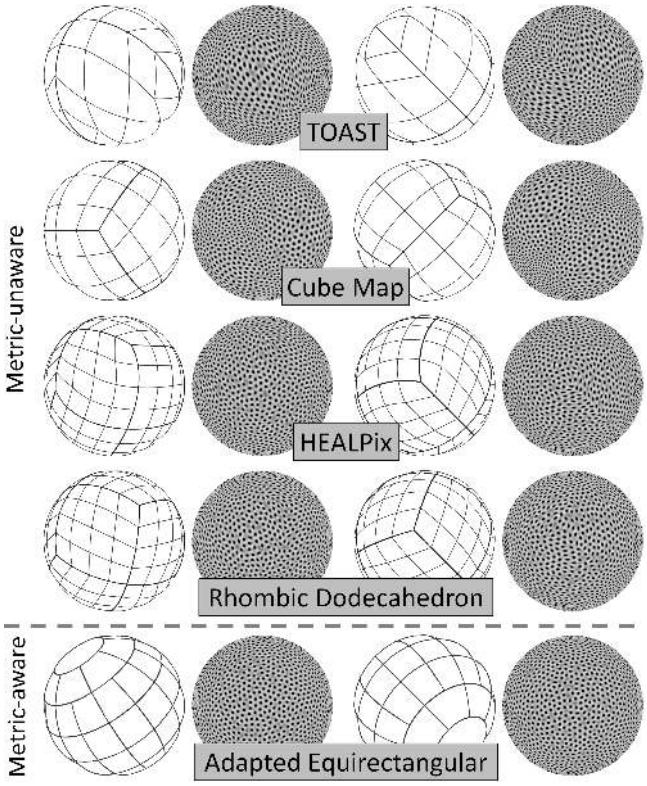


Figure 6: *Visualizations of the "spots" patterns obtained by running reaction-diffusion processes using finite elements systems derived from different spherical parameterizations. The use of a metric-unaware linear system results in patterns where the size and orientation of the spots reflects the underlying parameterization. In contrast, using a metric-aware linear system results in a uniformly sized and isotropic pattern.*

6.4 Spherical Image Processing

We apply our adapted equirectangular elements to two problems in spherical image processing: stitching and sharpening.

Image Stitching In this application, we stitch together a set of registered images comprising a spherical panorama. We do this by first constructing a target vector-field representing the approximate gradients of the seamless image and then solving the Poisson equation to fit an image to this vector-field [Pérez et al. 2003; Levin et al. 2004; Agarwala et al. 2004]. The target vector-field is constructed by merging the gradients from the individual images and setting seam-crossing gradients to zero. (Details on converting the finite-difference representation of the gradients to a finite-elements representation are provided in Appendix C.)

Figure 7 show the results of image stitching on a 32-megapixel spherical panorama comprised of 15 images (top row) and a 2-gigapixel spherical panorama comprised of 153 images (middle row). Starting with the composite image (center) and a label file indicating the source of the individual pixels in the composite (left), we construct the target vector-field. Then, solving the associated Poisson equation, we obtain the seamless images (right).

Gradient-Based Sharpening In this application we sharpen an image by solving a linear system that simultaneously aims to preserve the color values in the original image and amplify the gradients [Bhat et al. 2008]. For a source image $\tilde{W}$, we do this by solving the system in Equation 2 with $\alpha, \beta \neq 0$ and $\tilde{V} = \gamma \nabla \tilde{W}$ for $\gamma > 1$.

Figure 7: *Use of our metric-aware solver for image stitching (top and middle) and gradient-based sharpening (bottom). The figures on the left show the input images, and the ones on the right show the results of the image processing.*

The bottom row of Figure 7 shows the result of sharpening a 128-megapixel spherical panorama of an indoor scene. It can be seen that in regions of low-frequency, where the gradients are small, the value interpolation constraints dominate and the sharpened image preserves the original colors. In regions of high-frequency the (amplified) gradient interpolation constraints dominate and the sharpened image accentuates the color differences.

As discussed in [Kazhdan et al. 2010], metric-unaware solvers can still provide convincing solutions for image-processing applications such as stitching and sharpening. However, even in this context, an adapted equirectangular parameterization provides an advantage by allowing us to work directly with data sampled on an equirectangular grid, without requiring lossy sampling or expensive conversion.

Implementation As discussed in Section 4.4, we design a solver similar to that described by Kazhdan and Hoppe [2008]. Using multi-level streaming, our solver requires just two passes over the data and maintains a small window of image rows in working memory at any time.

Though our implementation requires that a separate stencil be computed for each row of the image, the cost of this computation is amortized across the relaxation of all the pixels within a row and results in a negligible overall increase in computation time.

Table 1 summarizes the performance numbers for the stitching and sharpening experiments described above, measured on a laptop with a 2.54 GHz Intel Core 2 Extreme Q9300 processor and a 6 MB L2 cache. Comparing the results in Table 1 to the results described in [Kazhdan and Hoppe 2008], we see that our spherical solver exhibits similar performance characteristics, requiring 1.2-1.4 seconds of computation time per megapixel (compared to the 1.4-1.6 seconds/megapixel in [Kazhdan and Hoppe 2008]) and keeping only a small (sub-linear) subset of the data in working memory at any time.

	Size	Time	Peak Memory	Size on Disk
Stitching 1	$8,192 \times 4,096$	45 sec.	83 MB	510 MB
Sharpening	$16,384 \times 8,192$	156 sec.	99 MB	2 GB
Stitching 2	$65,536 \times 32,768$	3276 sec.	207 MB	32 GB

Table 1: *Solution size, running time, and memory usage for the image stitching and sharpening applications shown in Figure 7, obtained with a solver running a single V-cycle, with five Gauss-Seidel iterations at each level, and using single-precision floating point values.*

We have confirmed the accuracy of our solver by comparing the above results with "ground-truth" results obtained using a solver running five V-cycles, with ten Gauss-Seidel iterations per level, and using double-precision floating point values. (Using this solver, the residual norms were smaller than 10^{-12}, suggesting convergence.) For both applications, a *single* V-cycle sufficed to obtain an image whose color values never differed by more than 1/255 from the values of "ground-truth" solution — the precision of a color channel in a 24-bit image.

7 Conclusion

In this work, we have introduced an adapted finite element system that supports the geometry-aware processing of spherical imagery. Using this system, we have shown how to extend traditional (planar) image processing techniques to the sphere, without sacrificing either speed or accuracy. In doing so, we have maintained essential regularity of the parameterization, enabling the design of a streaming multigrid solver that supports the processing of huge images.

In addition, because this representation is a simple extension of a commonly used spherical parameterization, the equirectangular map, our technique can be readily applied to existing datasets without any lossy resampling or expensive conversion.

Acknowledgments

We are very grateful to Alexandre Jenny for providing spherical image data (http://autopano.kolor.com/). We also thank Jeremy Goldhaber-Fiebert for assistance with image capture and Greg Turk for publishing his implementation of reaction diffusion. This work was funded in part by an NSF Career Grant (#6801727).

References

AGARWALA, A., DONTCHEVA, M., AGRAWALA, M., DRUCKER, S., COLBURN, A., CURLESS, B., SALESIN, D., AND COHEN, M. 2004. Interactive digital photomontage. *ACM Trans. on Graphics 23*, 3, 294–302.

AGRAWAL, A., AND RASKAR, R. 2007. Gradient domain manipulation techniques in vision and graphics. In *ICCV 2007 Course*.

BHAT, P., CURLESS, B., COHEN, M., AND ZITNICK, L. 2008. Fourier analysis of the 2D screened Poisson equation for gradient domain problems. In *European Conference on Computer Vision*, 114–128.

BRIGGS, W., HENSON, V., AND MCCORMICK, S. 2000. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics.

CEPHES, 1995. `http://www.netlib.org/cephes/`.

FEKETE, G. 1990. Rendering and managing spherical data with sphere quadtrees. In *Proceedings of Visualization '90*, 176–186.

FU, W., WAN, L., WONG, T., AND LEUNG, C. 2009. The rhombic dodecahedron map: An efficient scheme for encoding panoramic video. *IEEE Trans. on Multimedia 11*, 4, 634–644.

GÓRSKI, K., HIVON, E., BANDAY, A., WANDELT, B., HANSEN, F., REINECKE, M., AND BARTELMANN, M. 2005. HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal 622*, 759–771.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: A simple framework for adaptive simulation. *Trans. on Graphics 21*, 3, 281–290.

KARČIAUSKAS, K., AND PETERS, J. 2006. A C^2 polar jet subdivision. In *Symposium on Geometry Processing*, 173–180.

KAZHDAN, M., AND HOPPE, H. 2008. Streaming multigrid for gradient-domain operations on large images. *ACM Trans. on Graphics 27*, 3.

KAZHDAN, M., SURENDRAN, D., AND HOPPE, H. 2010. Distributed gradient-domain processing of planar and spherical images. *ACM Trans. on Graphics 29*, 2.

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. *ACM Trans. on Graphics 25*, 3.

LEVIN, A., ZOMET, A., PELEG, S., AND WEISS, Y. 2004. Seamless image stitching in the gradient domain. In *European Conference on Computer Vision*, 377–389.

MILLER, G., AND HOFFMAN, R. 1984. Illumination and reflection maps: Simulated objects in simulated and real environments. In *SIGGRAPH 84: Advanced Computer Graphics Animation Seminar Notes*.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Trans. on Graphics 22*, 3, 313–318.

SCHAFFER, S. 1998. A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients. *SIAM J. Sci. Comput. 20*, 1, 228–242.

SCHRÖDER, P., AND SWELDENS, W. 1995. Spherical wavelets: Efficiently representing functions on the sphere. In *ACM SIGGRAPH Conference Proceedings*, 161–172.

SCHRÖDER, P., AND SWELDENS, W. 1995. Spherical wavelets: Texture processing. In *Proceedings of Rendering Techniques*.

STAM, J. 2003. Flows on surfaces of arbitrary topology. *ACM Trans. on Graphics 22*, 3, 724–731.

STÜBEN, K. 2001. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics 128*, 1–2.

TURING, A. 1952. The chemical basis of morphogenesis. *Philosophical Trans. of the Royal Society of London 237*, 37–72.

TURK, G. 1991. Generating textures on arbitrary surfaces using reaction-diffusion. In *ACM SIGGRAPH Conference Proceedings*, 289–298.

WAN, L., WONG, T., AND LEUNG, C. 2007. Isocube: Exploiting the cubemap hardware. *IEEE Trans. on Visualization and Computer Graphics 13*, 4, 720–731.

WITKIN, A., AND KASS, M. 1991. Reaction-diffusion textures. In *ACM SIGGRAPH Conference Proceedings*, 299–308.

YING, L., HERTZMANN, A., BIERMANN, H., AND ZORIN, D. 2001. Texture and shape synthesis on surfaces. In *Eurographics Workshop on Rendering Techniques*, 301–312.

A Metric-Aware Systems

While Section 3 compares the performance of our metric-aware adapted finite element system with the metric-unaware systems defined by the other parameterizations, we also consider metric-aware implementations of these solvers by incorporating the metric tensor in the definition of the system. (For an example of such an implementation, we refer the reader to Stam's work on flows on surfaces [2003].)

Table 2 shows the results of using metric-aware finite element systems derived from the different parameterizations. The incorporation of the metric tensor results in solutions that are more accurate and improve with mesh refinement. However, due to the discretization of the integrals, the solutions derived from the TOAST, CubeMap, HEALPix, and Rhombic Dodecahedron parameterizations are not as accurate and do not improve as quickly under refinement as the solution derived using our adapted equirectangular system, which computes the integrals analytically.

Independent of accuracy, the use of metric-aware parameterizations without axial symmetry also leads to undesirable computational

costs. In our implementation, constructing the geometry-aware system matrices for the TOAST, Cube-Map, HEALPix, and Rhombic Dodecahedron parameterizations had an overhead of 12 seconds per megapixel (compared to 1.2-1.4 seconds/megapixel of solver time for our adaptive equirectangular solver).

Although this computational bottleneck can be partially resolved by pre-computing the system matrices and storing them on disk, such a solution would still be costly due to the associated I/O. (For second-order B-splines, each row of the matrix has 25 nonzero values, so storage costs would be on the order of 100 MB per megapixel.) This situation would be further exacerbated in the context of streaming large images, since the restricted working-set size would imply that matrix entries are likely to be evicted from memory before all the symmetric instances of a pixel are visited, making it harder to leverage symmetry for efficient processing.

Finally, even if the system can be fit into working memory, the lack of redundant matrix entries would necessarily slow down processing as it would more quickly overwhelm lower levels of the memory hierarchy, resulting in costly cache misses.

B Evolving the Reaction-Diffusion

We describe the way in which our finite-elements system can be used to perform geometry-aware reaction-diffusion on the sphere. For simplicity, we begin by describing a single-chemical system and then discuss generalizations to multi-chemical systems.

B.1 Single-Chemical Systems

Following the presentation in [Turk 1991], we consider the distribution of a chemical over the sphere, represented by the function $A(p)$. Given initial concentrations of the chemical, we would like to evolve the concentration, subject to the differential equation:

$$\frac{\partial A}{\partial t} = R(A) + \lambda \Delta A.$$

Here, the reaction function $R : \mathbb{R} \to \mathbb{R}$ defines the (nonlinear) manner in which the chemical either grows or decays in response to its current concentration, and the value λ gives the rate of diffusion.[1]

To advance the process, we update the values of A using a semi-implicit approach. Specifically, given function A^t corresponding to the chemical concentrations at time t, and setting $R^t = R(A^t)$, we obtain the concentration of A at time $t+\delta$ by solving:

$$\frac{A^{t+\delta} - A^t}{\delta} = R^t + \lambda \Delta A^{t+\delta} \quad \Longrightarrow \quad (\mathrm{id} - \delta\lambda\Delta)A^{t+\delta} = \delta R^t + A^t.$$

Galerkin Discretization Assuming that the functions A^t and R^t are expressed in terms of their coefficients with respect to the function basis $\{F_I\}$, with $A^t(p) = \sum a^t_I F_I(p)$ and $R^t(p) = \sum r^t_I F_I(p)$, we discretize the linear system by integrating both-sides of the equation against the basis functions $\{F_J\}$. This gives the system of equations:

$$(D - \delta\lambda L)a^{t+\delta} = \delta D r^t + D a^t. \tag{6}$$

Function Evaluation Although we can assume that we know the values of the coefficients a^t from the solution at the previous time-step, we cannot assume that the coefficients of $R(A^t)$ are known *a priori*. In particular, for most common reaction-diffusion processes, the function R is defined by the way it acts on function values, not function coefficients. Thus, to perform a semi-implicit time step, we need to:

1. Evaluate the coefficients of the chemical concentration,
2. Apply the reaction function to get the reaction rates, and
3. Fit coefficients to the reaction rate values.

To do this, we define the set $\{p_J\} \subset S^2$ to be the centers of the elements and construct the (sparse) evaluation matrix:

$$E_{I,J} = F_I(p_J).$$

Given this matrix, we compute the reaction coefficients r^t as:

$$r^t = E^{-1} \cdot R(E a^t). \tag{7}$$

(Overloading notation, we define R to be a vector-valued function by applying R to each entry of the vector.)

Thus, we evolve the chemical concentration by solving two linear systems (first Equation 7 and then Equation 6). Using the nesting structure of the basis functions $\{F_I\}$, both are solved in linear time using a multigrid framework (as described in Section 5).

B.2 Multi-Chemical Systems

More generally, the reaction-diffusion process may be driven by the interaction of multiple chemicals, $\{A_1, \ldots, A_n\}$, subject to the differential equations:

$$\frac{\partial A_i}{\partial t} = R_i(A_1, \ldots, A_n) + \lambda_i \Delta A_i.$$

Following Turk's implementation [1991], we evolve the chemicals simultaneously, using a Jacobi-like update, by solving:

$$(\mathrm{id} - \delta\lambda\Delta)A_i^{t+\delta} = \delta R_i(A_1^t, \ldots, A_n^t) + A_i^t.$$

(Although it would also be possible to update the chemical concentrations sequentially, using the most recently computed concentration values to define the new reaction rates, we have chosen the former approach because it allows the different chemical concentrations to be updated in parallel.)

Discretizing as in the single-chemical system, we evolve each chemical by solving the system:

$$a_i^{t+\delta} = (D - \delta\lambda_i L)^{-1} \cdot D\left(\delta E^{-1} \cdot R_i(E a_1^t, \ldots, E a_n^t) + a_i^t\right),$$

which also requires solving two linear systems per chemical.

[1] To generate the random noise often used in these systems, we construct a spherical function in the frequency domain, assigning random values to the spherical harmonic coefficients. Sampling at the centers of the elements gives a pattern of spherical noise that is consistent across the parameterizations and whose frequency content is parameterization-independent.

	Resolution					
	$N=32$	$N=64$	$N=128$	$N=256$	$N=512$	$N=1024$
TOAST	$1.6\cdot10^{-2}$	$7.5\cdot10^{-3}$	$3.7\cdot10^{-3}$	$1.8\cdot10^{-3}$	$9.2\cdot10^{-4}$	$4.6\cdot10^{-4}$
Cube	$9.0\cdot10^{-3}$	$3.1\cdot10^{-3}$	$1.3\cdot10^{-3}$	$5.6\cdot10^{-4}$	$2.7\cdot10^{-4}$	$1.3\cdot10^{-4}$
HEALPix	$2.3\cdot10^{-2}$	$9.5\cdot10^{-3}$	$4.7\cdot10^{-3}$	$2.4\cdot10^{-3}$	$1.2\cdot10^{-3}$	$6.3\cdot10^{-4}$
Rhombic	$3.3\cdot10^{-2}$	$1.2\cdot10^{-2}$	$4.9\cdot10^{-3}$	$2.2\cdot10^{-3}$	$1.0\cdot10^{-3}$	$5.0\cdot10^{-4}$
Adaptive	$1.4\cdot10^{-3}$	$4.0\cdot10^{-4}$	$1.2\cdot10^{-4}$	$3.5\cdot10^{-5}$	$9.7\cdot10^{-6}$	$2.6\cdot10^{-6}$

Table 2: *Discretization error for solutions to the Poisson equation when a degree-3 spherical harmonic is used as a constraint. In this experiment, all the finite element systems are metric-aware.*

C Setting the Stitching Constraints

In this appendix, we describe how to compute the constraints presented to the Poisson system used in solving the image-stitching problem. Because our aim is to support the processing of images represented in the equirectangular parameterization, we assume that the input is represented as a set of desired finite differences between adjacent pixels in a regular $N \times 2N$ grid. These finite differences are most often obtained by copying the finite differences from within individual image-patches and setting the finite differences across the seams between different image patches to zero. (See, for example, [Pérez et al. 2003; Agarwala et al. 2004; Levin et al. 2004].)

The only condition that we make on these finite differences is that all horizontal finite differences in the first and last rows are zero and that the vertical finite differences across the top and bottom edges of the parameterization are also zero. This corresponds to the fact that we define the polar cap functions to be axially symmetric functions (so their latitudinal derivatives are zero) and we force them to have zero derivative at the poles (so the derivatives across the latitudes 90° and -90° vanish).

We define the constraints in three steps. First, we transform the prescribed finite differences into a continuous vector field on the parameterization domain. Then, we compute the divergence of the vector field to get constraints for the regular equirectangular finite-elements system defined in Section 4. Finally, we transform the constraints from the regular equirectangular finite elements to the adapted elements.

From Finite Differences to Gradient Fields To transition from finite differences to (regular) finite-elements constraints, we follow the approach described in [Kazhdan and Hoppe 2008] which we review here. This approach uses the fact that the derivative of an n-th order B-spline can be expressed as the differences of two (shifted) B-splines of order $n-1$:

$$\frac{d}{dx}B^n(x) = B^{n-1}(x+0.5) - B^{n-1}(x-0.5).$$

As a result, it follows that if $F(x)$ is a function expressed as the sum of regularly shifted B-splines of degree n,

$$F(x) = \sum_i \alpha_i B^n(x-i),$$

then the derivative of $F(x)$ can be expressed as the sum of regularly shifted B-splines of degree $n-1$, *with coefficients given by the finite difference of the coefficients of* $F(x)$:

$$\begin{aligned}\frac{dF}{dx} &= \sum_i \alpha_i \left(B^{n-1}(x-i+0.5) - B^{n-1}(x-i-0.5) \right) \\ &= \sum_i (\alpha_{i+1} - \alpha_i) B^{n-1}(x-i-0.5).\end{aligned}$$

So, treating the finite differences of an array of values as the coefficients of a function (with respect to B-splines of order $n-1$) is equivalent to treating the entries of the array as coefficients of a function (with respect to B-splines of order n) and taking the derivative of the function.

A similar arguments shows that 2D finite differences can be interpreted as the coefficients of a gradient field, expressed in terms of B-splines of (mixed) order. Thus, given the input finite differences, one can interpret these as the coefficients of a vector field in the equirectangular parameterization of the sphere. Here, the restriction that the finite differences vanish near the poles ensures that the behavior of the gradient field is well-defined (i.e. zero) at the poles.

From Gradient Fields to Non-Adapted Divergence Constraints To extend the approach described in [Kazhdan and Hoppe 2008] to spherical imagery, we modify the integration defining the system coefficients to take into account the geometry of the sphere. Given the vector field (F_1, F_2) (defined by the finite differences) and given the gradient of the I-th basis function $(\partial B_I/\partial\theta, \partial B_I/\partial\varphi)$, we obtain the I-th coefficient of the constraint vector:

$$\begin{aligned}f_I &= \int_0^{2\pi}\int_0^{\pi} (F_1,F_2) g^{-1} \left(\frac{\partial B_I}{\partial\theta}, \frac{\partial B_I}{\partial\varphi} \right)^T \sqrt{\det(g)}\, d\theta\, d\varphi \\ &= \int_0^{2\pi}\int_0^{\pi} \left(\sin\theta F_1 \frac{\partial B_I}{\partial\theta} + \frac{1}{\sin\theta} F_2 \frac{\partial B_I}{\partial\varphi} \right) d\theta\, d\varphi,\end{aligned}$$

where $g = d\Phi^T d\Phi$ is the metric tensor defined by the Jacobian of the parameterization $\Phi : [0,\pi] \times [0,2\pi) \longrightarrow S^2$.

Since F_1, F_2, $\partial B_I/\partial\theta$, and $\partial B_i/\partial\varphi$ are all piecewise polynomial functions, the computation of the I-th coefficient of the constraint vector, f_I, reduces to the integration of the products of sine and cosecant functions with polynomials, as in Equation (5). Moreover, as with the coefficients of the system matrices, the rotational symmetry of the finite elements means that the integrals only need to be computed once per image row.

From Non-Adapted Constraints to Adapted Constraints Finally, we apply the transpose of the refinement operator described in Section 5 to get the constraints for the adapted system. Similarly, after solving the adapted linear system, we obtain a regular solution by applying the refinement operator.