GPU-based Conformal Flow on Surfaces

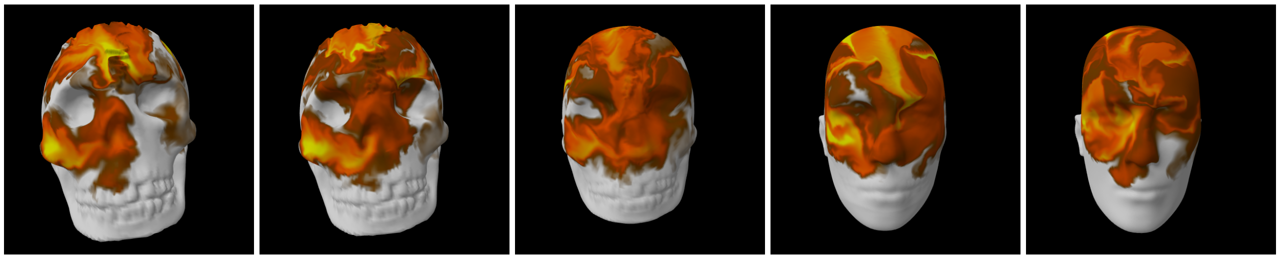

Figure 1: *A surface is morphed while fluid flows over it.*

Abstract

Accurately simulating fluid dynamics on arbitrary surfaces is of significance in graphics, digital entertainment, and engineering applications. This paper aims to improve the efficiency and enhance interactivity of the simulation without sacrificing its accuracy. We develop a GPU-based fluid solver that is applicable for curved geometry. We resort to the conformal (i.e., angle-preserving) structure to parameterize a surface in order to simplify differential operators used in Navier-Stokes and other partial differential equations. Our conformal flow method integrates fluid dynamics with Riemannian metric over curved geometry. Another significant benefit is that a conformal parameterization naturally facilitates the automatic conversion of mesh geometry into a collection of regular geometry images well suited for modern graphics hardware pipeline. Our algorithm for mapping general genus zero meshes to conformal cubic maps is rigorous, efficient, and completely automatic. performance. The proposed framework is very general and can be used to solve other types of PDEs on surfaces while taking advantage of GPU acceleration.

Keywords: GPU, fluid simulation, conformal structure

1 Introduction

Simulating fluid dynamics with realistic physical behavior is important for interactive 3D graphics and other fields. With the rapid advancement of modern hardware, accurate, physics-based simulation of fluid is gaining momemtum and popularity. However, despite the existing work in this field, performing such simulation for realistically complex systems remains technically challenging if real-time or interactive performance, flow over deformable surfaces, high-fidelity appearance, and intuitive user control are all required simultaneously. This paper proposes a new approach to interactive fluid dynamics over curved (and possibly deforming) geometry. Specifically, we design special coordinates on a surface in a mathematically rigorous way to simplify the governing PDEs and map any surface to a cube. With such cubic mapping, we can easily transform any curved surfaces to geometry images and exploit the power of modern graphics hardware to maximize the efficiency, accuracy, and interactivity.

Many efficient techniques exist for discretizing Navier-Stokes equations (and other PDEs) and simulating their dynamics when the flow occurs in a volumetric setting in R^3. When the fluid moves over a curved geometry, however, PDEs governing the flow must be modified in order to properly take into account Riemannian metric of the underlying surface. This requires a coordinate system for the surface. Yet, typically no convenient global coordinate system for the entire surface is immediately available and its parameterization must be explicitly constructed. One natural approach involves covering the surface with a collection of local, possibly overlapping charts, each with its own coordinate system. These coordinate systems are coupled through transformations called *transition maps*. The PDEs are then solved on each of the local coordinate charts and transition maps are used to propagate the results from one chart to another in a globally consistent manner.

Because of its intrinsic physical nature, the solution of a PDE is independent of the choice of surface parameterization. In practice, however, the choice of local coordinates will undoubtedly affect the stability and efficiency of the simulation process. According to the theory of Riemannian geometry, among the infinite number of local coordinate systems covering the same region on the surface, angle-preserving conformal (a.k.a. iso-thermal) coordinates induce the simplest form of differential operators used in most PDEs. For an arbitrary surface, we will therefore seek a collection of local coordinate charts (an atlas), such that all local coordinates are conformal and all the transition maps between them are conformal. Such an atlas is called Riemann surface structure of the surface.

General surfaces are typically represented as triangular meshes with irregular connectivity. To take advantage of computing power of modern GPUs, we convert such meshes into a more regular representation such as a geometry image [Gu et al. 2002]. Combining the benefits of conformal mapping and regular connectivity, we arrive at a conformal cube map as a natural solution. In our algorithm, we conformally map a genus zero closed surface to the unit sphere. We also map a canonical cube with regular connectivity to the sphere. Integrating these two spherical maps results in a conformal mapping between the surface and the cube. We then create a geometry image from each face of the cube. Different geometry images share the boundaries for fluid propagation.

In practice it is frequently desirable to make a conformal cube map consistent across different surfaces. For example, we may want to map major feature points on several surfaces to corresponding locations on the cube. In this case, we use a Mobius transformation to ensure the feature correspondence of up to three separate feature points. A more theoretical issue is that the edges (and corners) of the constructed cube map are not covered by any local charts. We describe how to add special charts to cover these singularities, such that all local parameters are still conformal and the transition functions to other existing charts are conformal as well. In practice, we found it was not necessary to actually use these extra charts but we include the corresponding discussion in the interest of completeness and theoretical rigorousness.

Our approach combines the benefits of several recent developments. Most important novel contributions to computer graphics are:

- A new algorithm for computing conformal cube map of curved geometry. This structure enables us to simplify Navier-Stokes equation and other types of PDEs on surfaces without losing accuracy;
- By using cubic conformal mapping combined with regular geometry image representation, we demonstrate how to adopt off-the-shelf GPU-based fluid solvers with minimal modifications;
- By tightly coupling deformable geometry with fluid dynamics, we develop to the best of our knowledge the first GPU-based interactive fluid solver for arbitrary surfaces.

We tested our system on several surfaces with complex geometric details. We demonstrate interactive user control and also conduct fluid experiments over shape morphing sequences. In all our examples we achieve interactive performance, while preserving stability and accuracy.

1.1 Previous Work

Floater and Hormann [2004] provide an extensive survey of the state of the art in parameterization. We refer the reader to this paper for a more detailed discussion of this subject and briefly mention only the most related work below.

Geometry images were introduced By Gu et al. [2002], who represent geometric surfaces with a regular image format to enable GPU-based geometric processing. Spherical parameterizations have been discussed in [Praun and Hoppe 2003]. Conformal parameterizations for topological disks have been presented in [Lévy et al. 2002], [Desbrun et al. 2002], and [Sheffer and de Sturler 2001].

Stam [2003] introduced flows on surfaces to the graphics community. His technique uses the same underlying 2D Stable Fluid solver [Stam 1999] as our method, but the parameterization he chose leads to hundreds of charts for meshes of reasonable complexity. Our conformal cubic parameterization can handle similar meshes using only six charts. A fixed number of charts that can handle many different models is a key factor in simplifying a GPU implementation. A somewhat different approach to the flows on surfaces problem was taken by later researchers [Shi and Yu 2004; Fan et al. 2005]. Instead of using a parameterization to solve the equations, they solve the system using an irregular discretization based on the triangular mesh. Conformal parameterizations are also popular in scientific computing (see [Lui et al. 2005] for an application to flow on surfaces problem) but the goals in this case are usually very different from those of computer graphics.

2 Background and Overview

We first present a brief summary of the theory of conformal parameterizations. More detailed information can be found in the literature on differential geometry [Lang 1999].

Suppose S is a surface embedded in the Euclidean space $\mathbb{R}^3$ parameterized through variables (u,v). Then S can be represented as a vector valued function $\mathbf{r}(u,v)$. The first fundamental form of S is a differential form which measures the squared distance on the surface:

$$ds^2(u,v) = E(u,v)du^2 + 2F(u,v)dudv + G(u,v)dv^2, \qquad (1)$$

with $E =< \frac{\partial \mathbf{r}}{\partial u}, \frac{\partial \mathbf{r}}{\partial u} >, G =< \frac{\partial \mathbf{r}}{\partial v}, \frac{\partial \mathbf{r}}{\partial v} >, F =< \frac{\partial \mathbf{r}}{\partial u}, \frac{\partial \mathbf{r}}{\partial v} >$. Here $<,>$ denotes the inner (dot) product in $\mathbb{R}^3$. The first fundamental form is also called *Riemannian metric*. A parameterization with (u,v) is called *conformal* if $E = G$ and $F = 0$. Geometrically, this means that the angle between any two intersecting lines on the surface is the same as the angle between their images in (u,v)-plane, i.e. only the area (but not the angles) is changed locally by the parameterization. Under conformal parameterization, we introduce notation

$$E = G = \lambda^2,$$

and refer to λ as *conformal* or *stretching factor*. λ^2 is equal to the ratio of differential area on the surface to that in parametric domain.

It is practically important that for any local region on the surface a conformal parameterization exists. The commonly used differential operators, such as gradient or divergence can be generalized onto surfaces by incorporating the Riemannian metric. It turns out that most differential operators have their simplest forms when conformal parameters are used, which is valuable for solving complicated PDEs. This new operator form often uses conformal factor λ which therefore plays a central role in our computation.

Suppose ϕ is a differentiable bijective mapping from surface S_1 to S_2 and $\tau : S_2 \to (u,v)$ is an arbitrary conformal parameterization of S_2. ϕ is called *conformal*, if mapping $\phi \circ \tau : S_1 \to (u,v)$, with $\circ$ denoting composition operation, is a conformal parameterization of S_1. It is easy to validate that the inverse of a conformal map is conformal and the composition of two conformal maps is conformal.

According to Riemann mapping theorem [Lang 1999], any simply connected surface with one boundary can be conformally mapped to the unit disk $\mathbb{D}^2$. Such mapping is not unique. In fact, there are 3 degrees of freedom which can be used to achieve other desirable properties. Similarly, any genus zero closed surface can be conformally mapped to the unit sphere $\mathbb{S}^2$. This mapping has 6 degrees of freedom. Furthermore, suppose $\phi : S \to \mathbb{R}^3$ is an arbitrary mapping defined on S with components $\phi = (\phi_0, \phi_1, \phi_2)$. Then the *harmonic energy* of ϕ is defined as

$$E(\phi) = \sum_{i=0}^{2} \int_S < \nabla\phi_i, \nabla\phi_i > d\sigma, \qquad (2)$$

where σ is the surface area. Intuitively, harmonic energy is proportional to the elastic stretching energy of the surface due to its distortion which is created by its mapping into another surface. It has been proven that if $\phi : S \to \mathbb{S}^2$ is conformal, then it has the minimal harmonic energy in all maps from S to the unit sphere. We will compute conformal maps by minimizing this energy.

Harmonic energy minimizers (conformal maps) are not unique. They form a 6-dimensional Möbius transformation group. In order to obtain a unique solution, we need to apply additional constraints. For example, we can set the "center of mass" of the resulting surface to the origin by ensuring that

$$\int_S \phi_i(p) d\sigma = 0, i = 1,2,3 \qquad (3)$$

This constraint will remove three degrees of freedom. The remaining three are then equivalent to rotations. Alternatively, we can choose positions of three arbitrary feature points on the surface. This is used by Möbius transformation described below.

If the surface is a topological disk, we can use the conventional double covering technique [Lang 1999] to turn it into a symmetric closed surface by gluing two copies of the surface along their boundaries. Therefore, in the following discussion, we only focus on closed genus zero surfaces.

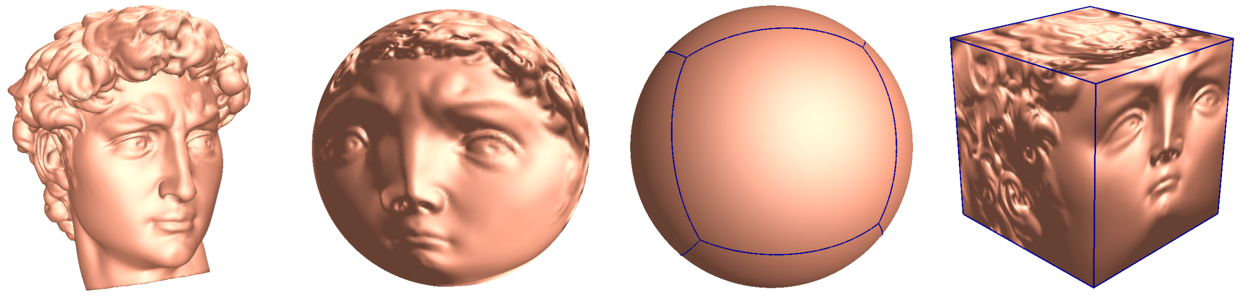
(a) David Surface
(b) Conformal Spherical map of David
(c) Conformal Spherical of a cube
(d) Conformal Cube map of David

Figure 2: ***Conformal Cube Map of David Head Surface***. *The Michelangelo's David head surface (a) is conformally mapped to the unit sphere (b). The unit cube is also mapped to the sphere (c). These two maps induce a conformal cube map of the David head surface (d).*

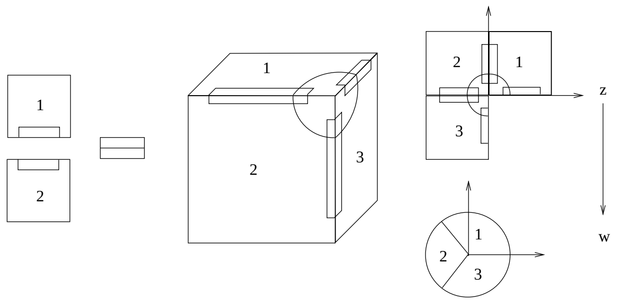

Figure 3: ***Full conformal atlas of a cube map.*** *Three face charts, three edge charts and one corner chart are illustrated. The chart transition map from face to corner is $w = z^{\frac{4}{3}}$, which is conformal.*

Suppose S_1 is an arbitrary genus zero surface and we want to construct its conformal mapping to another surface S_2. In all our examples S_2 is a simple cube, which makes hardware implementation particularly simple (see Section 5), but it can be a polycube to better represent the shape of S_1. We first compute conformal maps $\phi_1 : S_1 \rightarrow \mathbb{S}^2$ and $\phi_2 : S_2 \rightarrow \mathbb{S}^2$. Then the composition $\phi_2^{-1} \circ \phi_1 : S_1 \rightarrow S_2$ is the sought conformal map from S_1 to S_2.

We construct a conformal atlas in a manner similar to [Grimm 2002] and [Ying and Zorin 2004]. Each face, edge and corner vertex are associated with its own local chart. Each face chart covers only interior points of corresponding face and leaves off edges of the face. Each edge chart covers interior points of the edge but leaves off corner vertices. The corner vertices are covered by corner charts. Figure 3 demonstrates face, edge and corner charts of a unit cube. The transition map from a face chart to an edge chart is a planar rigid motion. The transition map from a face chart to a corner chart is $w = z^{\frac{4}{3}}$ composed with a planar rigid motion, where z is the complex coordinates on the face chart, and w is the complex coordinates on the corner chart. Therefore, all transition maps are conformal.

A constructed conformal atlas can be used to solve any partial differential equations if theoretical correctness of the solution is required. In practice we found it sufficient for graphics applications to compute and store only face charts with shared boundaries. This does not lead to visually noticeable artefacts and simplifies data structures and algorithms considerably.

Figure 2 illustrates the complete process of conformal cube map creation. After the parameterization is computed, we use it in the final step to resample the surface into a set of geometry images using regular grid on each face of the cube. The collection of resulting images forms a conformal cubic map of the entire surface which can be used to solve differential equations on the surface as described in Section 4.

3 Cube Parameterization

In this section, we explain in more detail our algorithm for constructing conformal cube maps for genus zero surfaces. All surfaces are represented as piecewise planar polygonal meshes. We use notation v_i for vertices of the mesh and $[v_i, v_j]$ to denote its edges.

3.1 Spherical Conformal Map

We need to construct conformal maps $\phi_k : S_k \rightarrow \mathbb{S}^2, k = 1,2$. First we compute normals $\mathbf{n}_i$ at each vertex v_i of S_1. We then define the Gauss map parameterization

$$\phi : S_1 \rightarrow \mathbb{S}^2, v_i \rightarrow \mathbf{n}_i.$$

which will serve as our initial map from S_1 to $\mathbb{S}^2$. Note that it will be more convenient for us to interpret $\phi(p)$ as a vector from the origin to a point on the sphere (which is, of course, normal to the sphere) rather than as the point itself. We adapt Polthier's method [Polthier 2002] to approximate the harmonic energy defined by Equation 2:

$$E(\phi) = \sum_{[v_i,v_j] \in S_1} \frac{1}{2} w_{ij} < \phi(v_i) - \phi(v_j), \phi(v_i) - \phi(v_j) >, \tag{4}$$

here weight w_{ij} is defined as

$$w_{ij} = \frac{< v_i - v_k, v_j - v_k >}{|(v_i - v_k) \times (v_j - v_k)|} + \frac{< v_i - v_l, v_j - v_l >}{|(v_i - v_l) \times (v_j - v_l)|},$$

where v_k and v_l are the two vertices sharing a face with edge $[v_i, v_j]$. We will use a heat flow method [Schoen and Yau 1997] to minimize harmonic energy. This technique updates $\phi(p)$ according to a variant of the heat equation:

$$\frac{d\phi(p)}{dt} = -\Delta^T \phi(p), \forall \phi \in S, \tag{5}$$

where Δ is the Laplacian-Beltrami operator and $\Delta^T \phi(p)$ represents its tangential component. The purpose of updating the map along the tangential direction of its Laplacian (rather than along the direction of full Laplacian) is to keep the image of point p on the target surface (sphere) while minimizing the harmonic energy.

The discrete Laplacian of ϕ at vertex v_i is the derivative of $E(\phi)$ with respect to $\phi(v_i)$:

$$\Delta\phi(v_i) = \sum_{[v_i,v_k]\in S_1} w_{ij}(\phi(v_i)-\phi(v_j)). \tag{6}$$

At each vertex, $\Delta\phi(v_i)$ can be split into the normal component $\Delta^{\perp}\phi(v_i) =< \Delta\phi(v_i), \phi(v_i) > \phi(v_i)$ and the tangential component which we need for Equation 5:

$$\Delta^T\phi(v_i) = \Delta\phi(v_i) - \Delta^{\perp}\phi(v_i).$$

The heat flow dynamics given by Equation 5 is approximated by updating $\phi(v_i)$ along the negative direction of this vector. The center of mass constraint 3 is simplified to

$$\sum_{v_i\in S_1} \phi(v_i) = 0.$$

which is enforced by subtracting $\sum_i \phi(v_i)$ from all $\phi(v_i)$'s and renormalizing the result to unit length after each iteration.

The process converges to a discrete conformal map from a surface to the unit sphere. The convergence speed is determined mainly by the geometric properties of the surface. For all models used in this paper (tens of thousands of vertices) the heat flow computation takes about 30 seconds. Figure 2 a and b illustrates the result of the conformal spherical map of the David head surface. For more details on heat flow method, including a formal proof of convergence of the constrained heat flow, we refer readers to [Schoen and Yau 1997]. The convergence of discrete Laplacian-Beltrami operator has been extensively discussed in [Xu 2004].

The final piece of information we need for our PDE solver is the conformal factor λ. For each vertex v_i we compute the sum of areas of all faces adjacent to v_i (its one-ring) in $\mathbb{R}^3$ and the sum of areas of one-ring neighbor faces in (u,v) plane. The ratio of these quantities gives an approximation for conformal factor $\lambda^2(v_i)$.

3.2 Feature Alignment with Möbius Transformation

For some applications, such as morphing, we need to create conformal spherical parameterizations of several surfaces which align their major features. For example, we might want to align the eyes and the nose center of both David head model and the skull model in their conformal cube maps.

First we separately conformally map both surfaces to the sphere, with maps ϕ_1, ϕ_2. Then we conformally map the sphere to the plane using stereographic projection,

$$\tau : (x,y,z) \rightarrow (\frac{2x}{1-z}, \frac{2y}{1-z}), (x,y,z) \in \mathbb{S}^2.$$

We then use a special conformal map from the plane to itself, a Möbius transformation, to move three arbitrary feature points into any new desired positions. Suppose for the first surface, the three feature points are z_0, z_1, z_2. We first construct the Möbius transformation which takes them into $0, 1, \infty$:

$$\psi_1 = \frac{(z-z_0)(z_1-z_2)}{(z-z_2)(z_1-z_0)}.$$

We can construct ψ_2 in a similar way. Then $\psi_1^{-1} \circ \psi_2$ maps the feature points on the second surface into those on the first one. The two conformal spherical parameterizations ϕ_1 and

$$\tau^{-1} \circ \psi_1^{-1} \circ \psi_2 \circ \tau \circ \phi_2$$

are therefore two consistent conformal spherical parameterizations, which align three feature points accurately on the sphere. Figure 4 demonstrates the alignment using Möbius transformation. The three feature points used are centers of the eyes and the tip of the nose for both skull and David head surface.

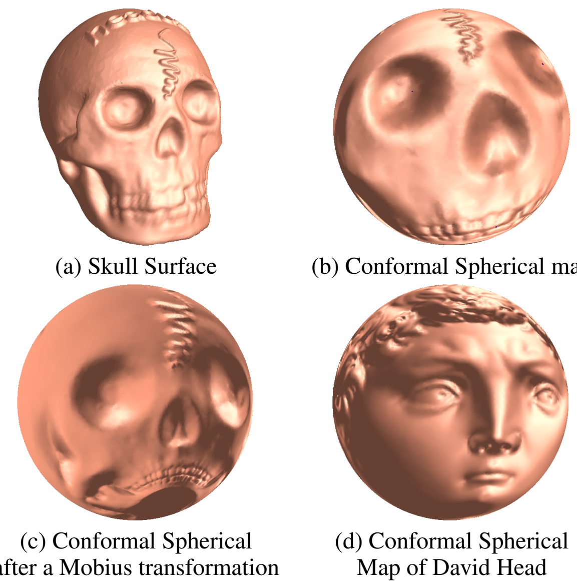
(a) Skull Surface
(b) Conformal Spherical map
(c) Conformal Spherical after a Mobius transformation
(d) Conformal Spherical Map of David Head

Figure 4: ***Consistent Conformal Spherical Maps**. The skull surface (a) is conformally mapped to the unit sphere. A Möbius transformation is acted on the sphere to align eyes and nose tip with those features on the conformal spherical image of the David head(d).*

4 Conformal Flow

Our goal is to use the described parameterization as a tool for solving partial differential equations on surfaces. We are particularly interested in solving Navier-Stokes equations for inviscid incompressible fluid flow:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \mathbf{f}; \qquad \nabla\cdot\mathbf{u} = 0 \tag{7}$$

where $\mathbf{u}$ and ρ are velocity and density of the fluid, p is pressure and $\mathbf{f}$ is external body forces such as gravity.

For flows on surfaces, fluid velocity is restricted to the tangent plane. Given a mapping of the surface to (u,v)-plane, we can attempt to solve differential equations in this plane and then map the result back onto the surface. This involves computation only in two dimensions and therefore we can take advantage of fast off-the-shelf 2D solvers, including those using graphics hardware. However, we first need to express all differential operators in the new domain taking into account any distortion caused by parameterization. Unfortunately, for most surface-to-plane mappings the resulting equations in (u,v)-plane are significantly different from being a simple 2D version of equations 7 above (including appearance of new terms) and extensive non-trivial modification of existing solvers is required. The key advantage of using a conformal mapping is that all differential operators can be obtained by simple incorporation of conformal factor $\lambda(u,v)$. We use subscript g to emphasize that these operators are computed differently. Symbols without subscripts refer to “standard” operators in corresponding domain.

Gradient of a scalar function ϕ is expressed as:

$$\nabla_g \phi = \frac{1}{\lambda^2}\left[\frac{\partial \phi}{\partial u} \frac{\partial \phi}{\partial v}\right] = \frac{\nabla \phi}{\lambda^2} \tag{8}$$

and its Laplacian:

$$\Delta_g \phi = \frac{1}{\lambda^2}\left(\frac{\partial^2 \phi}{\partial u^2} + \frac{\partial^2 \phi}{\partial v^2}\right) = \frac{\Delta \phi}{\lambda^2} \tag{9}$$

Given fluid velocity in (u,v)-plane $\mathbf{u} = (u_1(u,v), u_2(u,v))$, the divergence operator becomes:

$$\nabla_g \cdot \mathbf{u} = \frac{1}{\lambda^2}\left(\frac{\partial\left(\lambda^2 u_1\right)}{\partial u} + \frac{\partial\left(\lambda^2 u_2\right)}{\partial v}\right) \tag{10}$$

Since we will be using solvers based on Stam's stabe fluid algorithm [Stam 1999], it is useful to explicitly present here expressions for major steps of this technique (see original paper for details). In particular, advection equation becomes:

$$\frac{\partial \mathbf{u}}{\partial t} = -\frac{1}{\lambda^2}(\nabla \cdot \mathbf{u})\mathbf{u} \tag{11}$$

For projection operation, $1/\lambda^2$ terms cancel leading to :

$$\frac{\partial^2 p}{\partial u^2} + \frac{\partial^2 p}{\partial v^2} = \frac{\partial\left(\lambda^2 u_1\right)}{\partial u} + \frac{\partial\left(\lambda^2 u_2\right)}{\partial v} \tag{12}$$

We also make use of a vorticity confinement force [Fedkiw et al. 2001] to reduce numerical dissipation.

5 GPU Implementation

A nice property of the conformal parameterization is that it requires only minor changes to an existing solver. To demonstrate this, we modified the GPU implementation of the Stable Fluid method [Stam 1999] described in [Harris 2004]. We will limit our discussion to the changes we have made to support conformal flow. For a detailed description of the GPU-based solver itself, please refer to [Harris 2004].

The mesh and parameterization data are input into the system as 6 geometry images [Gu et al. 2002] using the standard cube layout shown in Figure 5. In addition to position in R^3, each pixel also contains the squared conformal factor λ^2 at this point. This information is precomputed using the procedure described in section 3. The fluid solver also stores velocity in texture memory using the same cube map layout. This arrangement allows to query λ^2 in the same manner as velocity. This makes it easy to modify fragment programs used by original solver according to expressions in the previous section. For example, relevant fragment of the original (written in Cg language) advection procedure might be

```
float2 pos = coords - dt * rdx * f2texRECT(u, coords);
xNew = f4texRECTbilerp(x, pos);
```

To modify it for the conformal solver according to Equation 11, we simply add a texture lookup for λ^2 and a few other operations:

```
half rlambda_sqr = 1.0 / h1texRECT(stretch, coords);
float2 pos = coords -
   dt * rdx * f2texRECT(u, coords) * rlambda_sqr;
xNew = f4texRECTbilerp(x, pos);
```

Modifications to other parts of the solver are similarly straightforward.

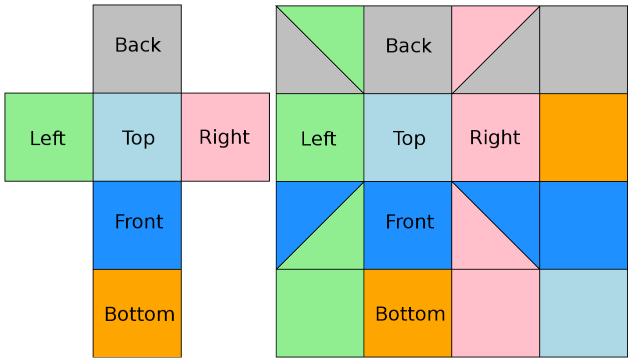

Figure 5: ***Left:*** *Cubic map.* ***Right:*** *full texture layout with boundary regions added. Colors correspond to those on the left and show which faces of the cube map provide the data for corresponding part of the texture. Texture is setup to wrap back over the edges.*

5.1 Boundary Conditions

As in [Stam 2003], we maintain a layer of boundary cells around the perimeter of the domain. Values are copied from the outer row of the adjoining face into this layer. This allows the differential operators to be evaluated without special handling of the boundary. Note that for vector fields, we have to transform the vector appropriately.

The most challenging aspect of the implementation is dealing with the boundary conditions for semi-Lagrangian advection procedure which involves tracing current position back in time using a velocity field. If a trace crosses the boundary of the cube, we need to return a valid value. The trace, however, can easily go beyond the size one boundary layer, requiring some special handling. Stam [2003] detects boundary crossings as part of the trace routine. We chose to seek an alternative approach for two reasons. First, this procedure involves a recursive while loop which, although it certainly is possible to implement in a fragment program, would lead to significant additional complexity of the system and extensive modification to any off-the-shelf GPU solver. This goes against our fundamental goals. Second, loops and branches in fragment programs are expensive operations. This additional cost would be incurred for all trace operations, not just those that cross a boundary.

Our alternative approach is to extend the boundary layer as far as possible. Figure 5, right, shows the full layout of our texture map with embedded original cube faces along with the boundary layer. In regions with two adjoining faces, we split the domain into two triangles and copy values from the appropriate region. Areas outside original cube faces are updated from corresponding cube face prior to each time step. Near corners where several faces meet, there is still potential for errors to be introduced as traces may sample from the wrong region but we found such cases not to cause serious problems for reasonably sized time steps. For graphics applications, we believe that this tradeoff of accuracy for simplicity is acceptable.

6 Examples

This section and accompanying video present the results of our approach for several meshed of genus zero. We used an Intel Pentium 1.8GHz PC with a 128MB NVIDIA GeForce 6800 GT graphics card. Each face of the cube has resolution 128x128 (512x512 total texture size) in all the examples. At this resolution, the system runs at approximately 15fps. Off-line preprocessing step to create geometry images and compute conformal mapping takes tens of seconds

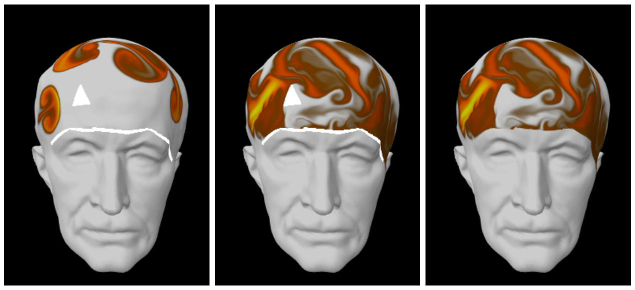

Figure 6: *The user can place internal boundaries (shown in white) on the mesh. The first two images demonstrate the effect of the added boundary on the flow and the third presents final appearance.*

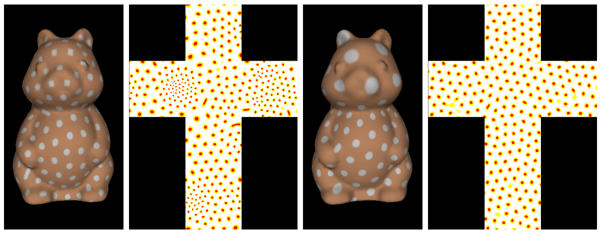

Figure 7: ***Left:*** *A simple reaction-diffusion texture synthesized with our approach. Contrast uniform feature size on the surface to that in the texture space.* ***Right:*** *The same texture computed without taking into account surface metric (using unmodified R-D equation).*

for our $10^4 - 10^5$ vertex meshes. Our first two video examples show fluid flowing over two surfaces of different geometric complexity - a sphere and a skull. In the second case, there are multiple sources introducing fluid.

One of the benefits of significant performance gain owed to implementing the system in hardware is that the user can directly interact with the flow. Our system allows users to introduce force on the surface using the mouse, as well as draw new internal boundaries. This allows to interactively manipulate the flow to create a desired effect. Figure 6 shows several frames from a session where the user has added internal boundaries (in white) preventing fluid from entering face area of the model. Internal boundaries can also serve to create regions on the surface forbidden for fluid flow. With appropriate (absorbing) boundary conditions this can be used to implement simple "holes" in the surface without resorting to solving equations on a non-genus zero surface.

If a surface is modified, it might be important to see how editing operations affect the fluid flow over the surface. Since surface geometry is changing, this would, strictly speaking, require to recompute the conformal parameterization which is prohibitively slow for interactive applications. However, for small changes it might be possible to use the original parameterization and stretching factors modifying only the geometry. One special case involves key frame based morphing of surfaces. For this case, we can compute necessary information for the end positions (and, if needed, for several other key frames) and simply interpolate stretching factors at runtime. The resulting mapping is no longer conformal for in-between frames. For large changes in geometry, this can lead to visible artefacts as demonstrated by an example included in the video. Figure 1 shows an example of successful application of this technique.

Other PDEs can also be solved using our approach. Figure 7 shows a model with a simple texture that was generated using reaction-diffusion technique [Turk 1991]. Note the non-uniform size of spots in the texture domain which is corrected by the inverse mapping. Simply running the unmodified solver in texture domain creates uniform size spots leading to distorted pattern on the surface.

7 Conclusion

We have articulated a new approach to solving Navier-Stokes equations and other PDEs on surfaces for computer graphics, based on unique properties of conformal cube maps. The algorithm constructs such map for an arbitrary surface of genus zero and then converts surface into a collection of regular geometry images. This allows one to use existing solvers, including GPU-based ones, with minimal code modification. The system achieves visually compelling and physically accurate results at interactive frame rates. Several practical applications are demonstrated, including surface-with-flow morphing and direct interaction with the flow through adding additional forces and internal boundaries.

Possible directions for future work include developing more efficient algorithms for conformal map construction, extending present system to handle surfaces of arbitrary genus, and designing better ways of controlling flows on surfaces.

References

Desbrun, M., Meyer, M., and Alliez, P. 2002. Intrinsic parameterizations of surface meshes. *Comput. Graph. Forum 21*, 3.

Fan, Z., Zhao, Y., Kaufman, A., and He, Y. 2005. Adapted unstructured LBM for flow simulation on curved surfaces. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 245–254.

Fedkiw, R., Stam, J., and Jensen, H. W. 2001. Visual simulation of smoke. In *Proceedings of SIGGRAPH 2001*, E. Fiume, Ed., 15–22.

Floater, M. S., and Hormann, K. 2004. Surface parameterization: a tutorial and survey. In *Advances on multiresolution in geometric modelling*, Springer-Verlag, Heidelberg, M.S.F.N.Dodgson and M.Sabin, Eds.

Gotsman, C., Gu, X., and Sheffer, A. 2003. Fundamentals of spherical parameterization for 3d meshes. *ACM Trans. Graph. 22*, 3, 358–363.

Grimm, C. 2002. Simple manifolds for surface modeling and parameterization. In *Shape Modeling International*, 237–246.

Gu, X., Gortler, S. J., and Hoppe, H. 2002. Geometry images. *ACM Trans. Graph. 21*, 3, 355–361.

Harris, M. 2004. Fast fluid dynamics simulation on the GPU. In *GPU Gems*, R. Fernando, Ed. Addison Wesley.

Lang, S. 1999. *Fundamentals of Differential Geometry*. Springer.

Lévy, B., Petitjean, S., Ray, N., and Maillot, J. 2002. Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH*, 362–371.

Lui, L. M., Wang, Y., and Chan, T. F. 2005. Solving pdes on manifolds with global conformal parameterization. In *VLSM*, 307–319.

Polthier, K. 2002. Computational aspects of discrete minimal surfaces. In *Processing of the Clay Summer School on Global Theory of Minimal Surface*, J. Hass, D. Hoffman, A. Jaffe, H. Rosenberg, R. Schoen, and M. Wolf, Eds.

Praun, E., and Hoppe, H. 2003. Spherical parametrization and remeshing. *ACM Trans. Graph. 22*, 3, 340–349.

Schoen, R., and Yau, S.-T. 1997. *Lectures on Harmonic Maps*. International Press.

Sheffer, A., and de Sturler, E. 2001. Parameterization of faceted surfaces for meshing using angle-based flattening. *Eng. Comput. (Lond.) 17*, 3, 326–337.

Shi, L., and Yu, Y. 2004. Inviscid and incompressible fluid simulation on triangle meshes: Research articles. *Comput. Animat. Virtual Worlds 15*, 3-4, 173–181.

Stam, J. 1999. Stable fluids. In *Proceedings of SIGGRAPH '99*, 121–128.

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

Turk, G. 1991. Generating textures on arbitrary surfaces using reaction-diffusion. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 289–298.

Xu, G. 2004. Discrete laplace-beltrami operators and their convergence. In *Computer Aided Geometric Design*, 767–784.

Ying, L., and Zorin, D. 2004. A simple manifold-based construction of surfaces of arbitrary smoothness. *ACM Trans. Graph. 23*, 3, 271–275.