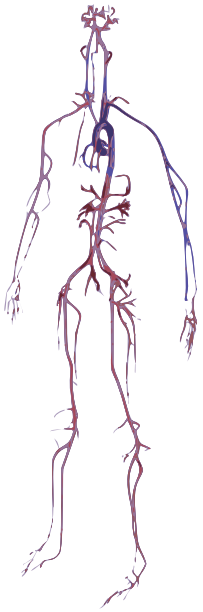
Tubular Partitions of Vascular Geometry

Shane Scott
Host Site: Lawrence Livermore National Laboratory
Mentor: Peer-Timo Bremer

Summer 2017

Abstract

Researchers at Lawrence Livermore National Lab have created HARVEY, a predictive vascular simulation modeling blood flow in patient-specific geometries. HARVEY is a massive fluid dynamics computation run in parallel on many computational cores. A current bottleneck of the computation speed is the load balancing, which does not account for the tubular geometry of vascular systems. We propose a novel geometric algorithm for partitioning the vascular system into connected regions of equal computational complexity. While we have no yet implemented the partitioning as a load balancing scheme in HARVEY, we have implemented the partitioning algorithm prototyping the algorithm in C++. We hope to next implement and measure the performance of the scheme in HARVEY load balancing.



A vascular system with HARVEY computed blood flow

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

1 Internship Project

This summer at LLNL, I worked with Peer-Timo Bremer to design and code the surface partitioning algorithm presented below!

1.1 Motivation: HARVEY

Imagine the medical power of a doctor with complete knowledge of her patient's blood flow. Blood flow information could be used to assess potential clots or strokes, or even enable local control of drug delivery. Researchers at Lawrence Livermore National Laboratory have built a powerful blood flow simulation called HARVEY [1] [8]. HARVEY can compute the bloodflow in an individual's vascular geometry. According to HARVEY's authors, Erik Draeger and Amanda Randles:

> "Building a detailed, realistic model of human blood flow is a formidable mathematical and computational challenge. Livermore researchers are addressing this challenge through the enhancement of HARVEY, an open-source parallel fluid dynamics application designed to model blood flow in patient-specific geometries
>
> LLNL researchers will use HARVEY to achieve a better understanding of vascular diseases as well as cancer cell movement through the bloodstream. Computational results will be validated through rigorous comparison with in vivo and in vitro measurements." [2]

HARVEY requires immense computing power to compute a meter scale sized simulation with resolutions as high as 10μm However, one bottleneck in the speed of HARVEY's simulation comes from its unbalanced distribution of the computational work load to its 1,572,864 parallel computing cores.

1.2 Problem: Geometric Computational Load Balancing

To enable computation, the HARVEY simulation discretizes space as a cubical grid. The geometry of a patient's vascular structure is represented by marking grid elements as either inside the vascular system or

[1] Named for pioneering 16th century physiologist William Harvey.
[2] https://computation.llnl.gov/projects/predictive-vascular-modeling

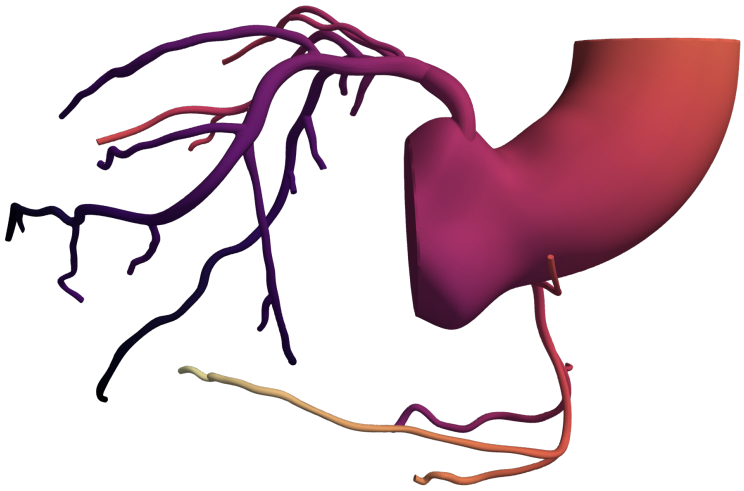


Figure 1: The geometry of a piece of a coronary artery. Observe the wide variation in the density of occupied space between the larger aorta and its smaller, branching arteries. The tree like tubes do not divide well into any rectilinear grid.

outside. The simulation is distributed over approximately 1.6 million computational cores. Each core is assigned some region of the grids. Grid elements outside pose essentially no computational burden since they require no updates.

Grid elements on the boundary or the interior of the vascular system experience different fluid dynamics, and so represent different computational burdens. Grid elements on the boundary regions assigned to different cores also require communication between cores, decreasing parallelization of the simulation. Thus poorly assigned regions might require some cores to perform more computations than others.

Thus a naive scheme of randomly assigning grid elements to regions may assign some cores a much heavier computational load. This may result in much longer computational times, as cores cannot proceed with their next regional computation until neighboring regions have completed their own.

The computational load balancing problem can be phrased as follows. Equip the integer lattice $\mathbb{Z}^3$ with a graph structure by be setting $x, y \in \mathbb{Z}^3$ adjacent if their distance is 1. Let $G \subset \mathbb{Z}^3$ be a finite,

connected grid. Assume a fixed number of cores N and independent, positive complexity costs per grid element:

· c_0 per grid element interior to vascular system

· c_1 per grid element on the boundary of the vascular system

· c_2 per grid element on the boundary two regions

so that given a partition of the grid $G = \bigcup_{j \in N} R_j$ into connected regions, we assume a linear cost per region

$$c(R_j) = \sum_{i \in \mathbb{Z}/3, g \in R_j} c_i s_i(g)$$

where $s(g) \in \{0,1\}^3$ is a binary vector labeling the type of the grid element $g \in G$. The problem is to compute a partition $\{R_j\}_{j \in \mathbb{Z}/N}$ of G minimizing the variance of $\{c(R_j)\}_{j \in \mathbb{Z}/N}$. While we do not hope to solve this optimization exactly, we will instead attempt to compute a low variance partition by making some assumptions on the geometry of a patient's vascular system and considering the smooth vascular structure.

1.3 The Smooth Strategy

We do not consider capillaries moving into the organs, only veins and arteries. This restriction gives the vascular system a simply connected, tree-like structure of tubes. We thus assume a smooth topological three-ball embedded in three-space $B \subset \mathbb{R}^3$ representing the vascular system with boundary surface $S = \partial B$. In this smooth representation the load balancing problem asks to compute a partition of the ball $B = \bigcup_j R_j$ into connected regions $\{R_j\}_{j \in N}$ minimizing the variance in the costs

$$c_j = c_0 V_j + c_1 A_j + c_2 A'_j$$

where V_j, A_j, A'_j are the volume of R_j, the area of $R_j \cap S$, and the area of $\partial R_j - S$, respectively. While the core-communicating-cost c_2 is of practical concern, we first disregard the inter-core-communication by setting $c_2 = 0$.

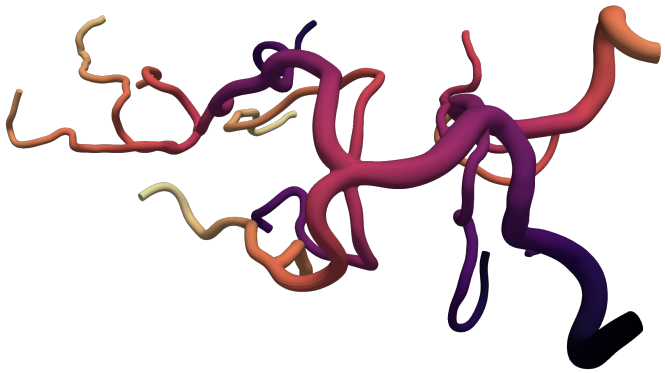


Figure 2: This cerebral arterial network is a series of tubes. Color indicates the length along the tube, which can be computed to give tubular coordinates for the surface

We suggest a strategy of dimensionality reduction of the problem. This reduction is performed by considering the surface S. The algorithm can be summarized as:

1. Compute tubular coordinates q on the surface S
2. Obtain a measured tree T to reduce the problem dimensionality
3. Partition the tree by integration
4. Correspondingly partition the surface S

The arterial structure is reminiscent of a combinatorial tree, with many local cylinders joining. We wish to construct corresponding coordinates. The surface is almost a product $T \times S^1$ of a metric tree T, and a circle S^1. This product fails to represent the surface S at the vertices of T, where the circle should be replaced by a quotient of circles. So we consider coordinates given by a quotient map

$$q : \bigsqcup_{e \in E} e \times S^1 \to S$$

where E is the set of metric edges of the tree T. The quotient is such that q is injective away from vertices, and at a vertex v of T so $q(v)$ is a wedge of $\deg(v) - 1$ circles. We discuss construction of q in the

following section. Compute from q measured metric tree which can then be partitioned by integration from a desired point as follows. Let $x_e : e \to [0, \ell_e]$ be a coordinate on length ℓ_e edge $e \in E$. If $x \in T$ is not a vertex then $q(\{x\} \times S^1)$ is an embedded circle in $\mathbb{R}^3$. Then the length ℓ_x of $q(\{x\} \times S^1)$ gives the rate of change $\frac{dA}{dx}$ in surface area A of the surface S. The minimal surface area A_x of a disk bounded by $q(\{x\} \times S^1)$ gives the rate of change of the volume $\frac{dV}{dx}$. Obtain a measure m on the tree by

$$\begin{aligned}\frac{dm}{dx_e} &= c_0 \frac{dV}{dx_e} + c_1 \frac{dA}{dx_e} \\ &= c_0 A_{x_e} + c_1 \ell_{x_e}\end{aligned}$$

representing an estimate of the computing cost per edge length. Integrating from a leaf, we can compute recursively a set of partition points $X \subset T$ such that $T - X$ consists of N components of equal measure $\frac{m(T)}{N}$. Generically no partition point is a vertex.

Then we have $q^{-1}(x)$ for $x \in X$ is a curve on S. Choose a minimal area disk D_x bounded by $q^{-1}(x)$. The components of $B - \bigcup_x D_x$ give a partition of B approximating a solution to the discrete load balancing problem.

1.4 Computing Tubular Coordinates

The smooth strategy outlined above requires tubular coordinates for the surface S. In this section we outline a method for computing such coordinates from the local properties of the surface. We begin by computing a function measuring length along a tube.

We first compute the local *curvature* of the surface. At every point $p \in S$ we have a unit normal vector n_p perpendicular to the tangent plane T_pS at the point p. Choose any tangent direction $v \in T_pS$ and consider the derivative of the normal vector field in that direction $\partial_v n_p$. Since n is unit length, its derivative in any direction at p can have no component in the n_p direction. So we have that $\partial_v n_p \in T_pS$. This allows us to define the shape operator at every point

$$\begin{aligned}Z_p : T_pS &\to T_pS \\ v &\mapsto -\partial_v n_p\end{aligned}$$

Figure 3: The minor principle curvature gives a tangent line, but no direction preference. Locally computing the shape operator results in essentially randomly pointing forward or backward arrows, as shown by the arrows to the right. The choice of unit eigenvector of the shape operator must be locally aligned to obtain a smooth, integrable vector field.

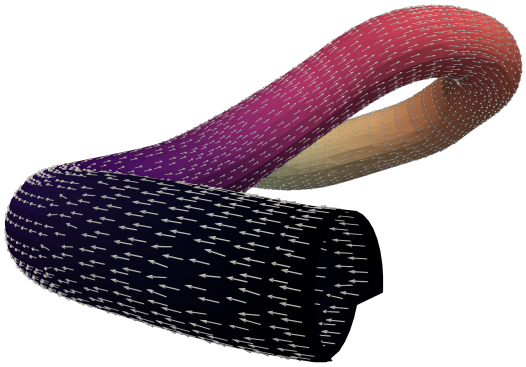


Figure 4: We can maximizing the local alignment of the unit minor principle curvatures by flipping some subset of the discrete vector field. Now the arrows point consistently in one direction along the least-curvature line. The potential ϕ shown by color gives distance along the tube and is obtained by integrating the vector field.

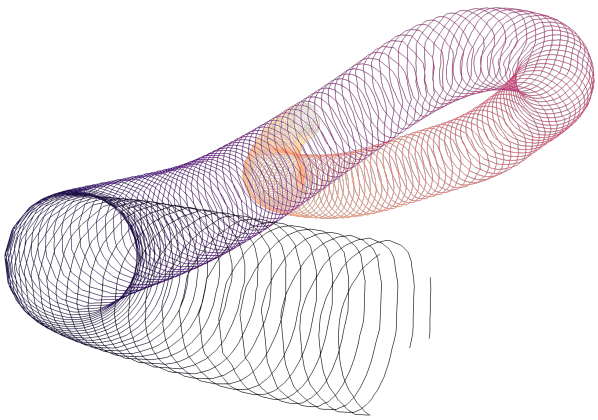


Figure 5: Level sets of the tubular parametrization ϕ. Tangent lines to these curves are the major principle curvature directions. Each curve gives a point in the Reeb graph of ϕ, so that the tube gives a linear edge.

The shape operator is symmetric, where for any vector $v, w \in T_pS$

$$\langle v | Z_p w \rangle = \langle Z_p v | w \rangle.$$

This guarantees that Z_p is orthogonally diagonalizable with real eigenvalues $k_p^{(1)} \geq k_p^{(2)}$, called the major and minor *principle curvatures*, respectively. For example, a perfect cylinder with radius r we would have at any point p principal curvatures $k_p^{(1)} = \frac{1}{r}$ in the circular direction of the cylinder, and $k_p^{(2)} = 0$ in the direction of the cylinder's height. Hence the curvature directions suggest local gradient directions to our desired coordinates. We hope to construct a Morse potential function $\phi : S \to \mathbb{R}$ such that the gradient $\nabla\phi(p) \in T_pS$ of ϕ at a point $p \in S$ lies in the minor principle curvature eigendirection at p. An approximate discrete computation is discussed in the next section. The *Reeb graph* of ϕ is a quotient space of S where the equivalence classes are the connected components of level sets of ϕ. Since S is a topological sphere, its Reeb graph is a metric tree T, discussed above. This gives the desired coordinates on the surface. Observe however that this method is heavily dependent on the local curvature directions. Small bumps or rapid turns of the tubes will distort the local curvature frame, invalidating the strategy even in the smooth case. In the computable discrete case, relatively small amounts of noise can also distort the local curvature and invalidate any proposed results.

1.5 Discretization and Local Computation

In practice the surface is represented as a simplicial complex given by triangle mesh. This data consists of a finite set of vertices V_S and triplets of vertices defining the tringles T_S approximating S. The vertices of a triangle $t \in T_S$ are ordered $t = (p_{(t)0}, p_{(t)1}, p_{(t)2})$ so that the surface is consistently oriented. We represent the edges of t by the vectors $e_{(t)i} = p_{(t)i+2} - p_{(t)i+1}$ for $i \in \mathbb{Z}/3$. We can consider the surface S_D a piecewise-linear manifold, and we define a discrete analog of the shape operator per triangle $t \in T_S$. If we consider the perpendicular edge vector ${e_{(t)i}}^{\perp} = e_{(t)i} \times n_t$ and angle θ_i between the normals across edge i, the discrete shape operator can be written as

$$Z_t = \sum_i \frac{\theta_i}{\|e_{(t)i}\|} \left(e_{(t)i}^{\perp}\right)^{\dagger} e_{(t)i}^{\perp}.$$

This local formulation depends only on the six points that make up the triangle t and its adjacent triangles. From Z_t we can compute the minor principle curvature direction z_t as the eigendirection for the smaller eigenvalue.

Unfortunately there are two opposite unit vectors in this eigenspace. Chose either unit vector. This may result in vectors pointing in opposite directions at adjacent triangles, ruining the local continuity of z_t. See figure 3.

In [4] this is corrected by lifting the line bundle associated to the eigendirection to a branched double cover of the surface, but we also hope to correct small patches where the local geometry fails to agree with the tubular structure. We discuss realignment in the following section.

From the discrete vector field z, we hope to find the potential ϕ whose gradient is given by z. Even if vectors are assigned locally consistently, there does not exist any smooth unit vector field z on S, by the hairy ball theorem. There are many points where the shape operator is degenerate with the principle curvatures equal and the minor principle curvature direction undefined. So the discrete unit vector field z_t thus defined cannot be the gradient of any potential. But we may hope for a least squares solution.

Given a scalar function $f : V_S \to \mathbb{R}$ defined by on vertices of S, we define the discrete gradient per triangle $t = (p_0, p_1, p_2) \in T_S$ using the difference operator

$$\nabla f(t) = (f(p_2) - f(p_1)) \frac{e_{(t)0}}{\|e_{(t)0}\|} + (f(p_0) - f(p_2)) \frac{e_{(t)1}}{\|e_{(t)1}\|}$$

We can construct the potential ϕ by finding the least square solution to to $\nabla\phi = zof$ that is $\phi = (\nabla^\dagger\nabla)^{-1}\nabla^\dagger z$, though in practice this is a linear system with on the order of 10^8 variables, so is solved numerically by gradient descents.

1.6 Curvature Field Alignment

The shape operator Z defined per triangle gives two options for a unit vector, as shown in figure 3 But additionally small patches of local geometry where the tube turns sharply may have major curvature direction parameterizing the cylinder's height direction. See for example 6.

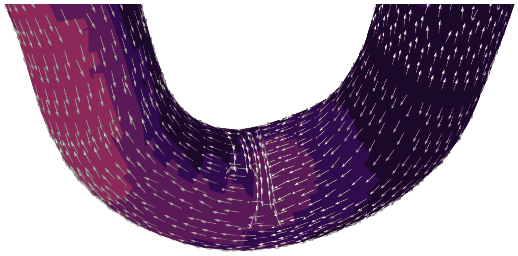


Figure 6: Here the tube turns quickly, with a turn radius smaller than its cross-sectional radius. This results in a small patch where the minor principle curvature direction points around the tube's circumference, rather than down its length. To parametrize length we remove such patches by rotating the vectors by 90°.

So we assign random unit vectors for the eigenvectors of Z to form a vector field u. The vectors u_t at many triangles t will need to be flipped 180° in the tangent space of t. But we assume a minority of triangles will also have vectors that should be flipped 90° or 270°. Thus to maximize local continuity we would like to compute a *rotation scheme* $r : T_S \to \mathbb{Z}/4$ achieving the maximum

$$\max_{r \in \mathbb{Z}_4^{T_S}} \sum_{t,t' \in T_S} u_t^\dagger R_{\frac{\pi}{2}(r_{t'} - r_t)}(n_{t'})\ u_{t'} \tag{1}$$

where $R_\theta(w)$ is the rotation matrix rotating about the axis $w \in \mathbb{S}^2$ by angle θ.

We propose computing a rotation scheme by two successive solves of MAXCUT on problem on the dual graph to the triangulation of the surface. The MAXCUT problem is typically stated as follows: Given a graph with edge weights, compute a bipartition of the vertices which maximizes the total weight of edges between the two sides of the partition. Though MAXCUT is in general NP-hard, there exists polynomial time algorithms for planar graphs. [9]

We propose using the planar MAXCUT algorithm to compute first which vectors should be rotated 90° or 270°, i.e. compute $x \equiv r \pmod 2$. Then we change edge weights and compute MAXCUT again to decide which vectors to flip 180°.

Let G be the dual graph to the surface triangulation, so the vertices of G are the triangles T_S, and two vertices of G are adjacent to each

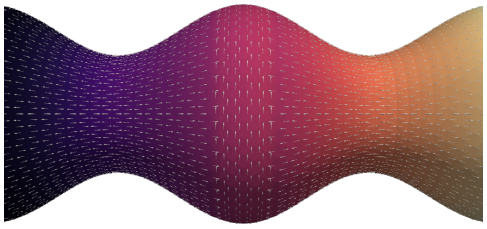


Figure 7: Changes in tube diameter can make the minor principle curvature eigenline tangent to the diameter.

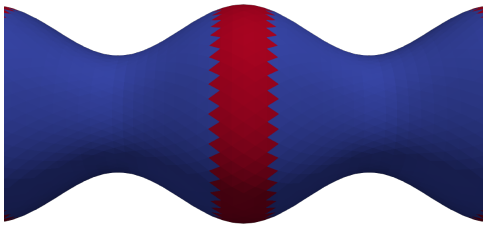


Figure 8: The first MAXCUT partitions the triangles into two sets, shown in red and blue, with internal agreement on an eigenline of the shape operator.

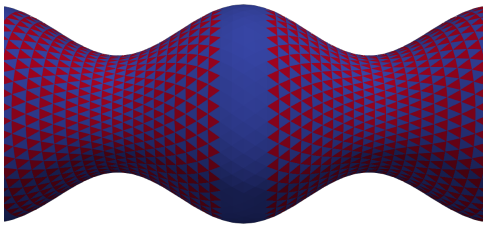


Figure 9: The second MAXCUT partitions the triangles into two sets, shown in read and blue, with internal agreement on a forward direction.

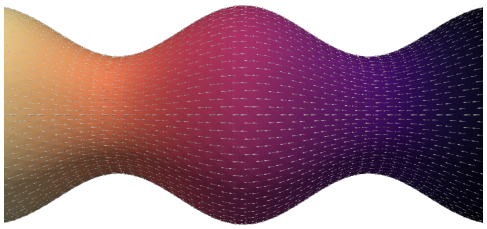


Figure 10: A realigned vector field post MAXCUT gives length along the tube.

other if and only if they share an edge as triangles. Since the surface is a sphere, the dual graph G lies on the surface of a sphere, and so is planar. Since both MAXCUT solves are on a planar graph, this can be computed in polynomial time.

The first MAXCUT aligns the vectors' linear spans. Assign edge weights

$$c_{tt'} = \frac{1}{2}\left(u_t^\dagger R_{\frac{\pi}{2}}(n_{t'})\ u_{t'}\right)^2 + \frac{1}{2}\left(u_t^\dagger R_{\frac{\pi}{2}}^\dagger(n_t)\ u_{t'}\right)^2 - \left(u_t^\dagger u_{t'}\right)^2$$

to adjacent pairs of triangles. This weight is positive if the v_i and v_j are nearly perpendicular and negative if they are nearly aligned. So the MAXCUT divides the triangles into two sets, where have largely perpendicular vectors at their boundaries. We solve for a maximum weight cut

$$T_S = S_0 \cup S_1$$

with S_1 the smaller of the two sides. Then $x_t = 0$ for all $t \in S_0$ and $x_t = 1$ for all $t \in S_1$. Update the vectors $v_t \leftarrow R_{\frac{\pi}{2}}(n_t)v_t$ for all $t \in S_1$.

Now decide if vectors are aligned or anti-aligned, i.e. we compute $y_t = \text{floor}(r_t/2)$. Now assign new edge weights

$$c'_{tt'} = -v_t^\dagger v_{t'}$$

which are positive if the vectors are nearly anti-aligned, and negative if the vectors are nearly aligned. Again solve for a maximum weight cut $T_S = S'_0 \cup S'_1$ with S'_1 the smaller set. Then $y_t = 0$ for all $t \in S_0$ and $y_t = 1$ for all $t \in S_1$. Update the vectors $v_t \leftarrow -v_t$ for all $t \in S'_0$. So the rotation scheme is $r = 2y + x$.

After the two MAXCUT computations are completed, we obtain a locally aligned vector field, as in figure 10. I do not know if in general it is true maximum of the local alignment measure given in Expression 1. It is conceivable that maximally aligning the linear spans in the first MAXCUT forces suboptimal anti-alignments in the second, which might otherwise be perpendicular with a lower total cost. However, since the line bundle associated to the minor principle curvature direction is actually aligned away from singularities, such solutions seem geometrically implausible.

1.7 Future Work

We have here constructed a novel partitioning algorithm for tree-like tubular surfaces and implemented C++ code available at https://github.com/scottsha/minorcurvature. This allows geometry-specific segmentation of a patient's vascular system to provide better load balancing for the HARVEY blood flow simulation. The most immediate future work is to implement the partitioning into HARVEY's current load balancing and determine the change in the load balancing.

Once the load is balanced, one may also need to consider noise and the computational cost of communication between cores, as discussed in the problem description.

The current algorithm should be extremely sensitive to local noise. The curvature directions are computed locally per triangle from the angles around the three edges. Lumpy surfaces or jitter in the positions of points would invalidate computation of the tubular coordinates. A more general method will need smoothing or alternative heuristics for deciding when the local curvature does not agree correctly describe the tubes.

The current algorithm also does not consider the communication costs, which should be proportional to the area of the separating surfaces between regions. It is possible that HARVEY computational speed can be improved with a slightly less even load balance that reduces the area of adjacent regions, thus reducing inter-core communication. A more detailed analysis of HARVEY's computation is required to assess the relative time costs of core communication, and, if significant, an additional step in the load balancing algorithm will need to consider inter-core communication in the partitioning phase.

2 Impact of Internship on My Career

My career goals for this internship were to understand better the industrial and national lab demand for mathematicians. LLNL was the perfect host for understanding the goals and types of positions available at national laboratories.

LLNL's summer student scholar program organizes a large number of helpful seminars and programs. I attended interesting briefings on very diverse topics and at a variety of levels: Bayesian statistics, ma-

chine learning, neural network detection of extreme climate events from sparse data, github version control, word2vec word geometry, CPR, efficient computation of geometric intersections, resume writing, job interviewing, graduate fellowships at LLNL, and the mission of LLNL from the director.

One of the most exciting aspects of LLNL was the wide variety of research conducted by visiting students. I participated in a poster session held in early August and was able to learn a great deal about state of the art techniques from all over scientific and engineering disciplines from speaking to fellow interns.

I am most greatful that LLNL introduced me to computational topological analysis. My research interests have become much more computation focused over the summer. Most practically, this summer LLNL taught me C++ and a lot of coding and developement skills, and some of the concerns of numerical computation. I learned a great deal about existing software. I know much more about practical mathematical problems that occur outside of academic mathematics, and I am much more excited (and, frankly, vastly less anxious) about career opportunities. I cannot thank NSF, ORISE, ORAU, and LLNL enough for a fascinating, invigorating, and productive summer.

3 Acknowledgments

I would like to thank the Lawrence Livermore National Lab and its very welcoming summer student internship program. Peer-Timo Bremer is an amazing mentor with wide ranging guidance, helpful insight, and fascinating problem suggestions. I would also like to thank fellow LLNL students Benafsh Husain and Pavol Klacansky for their inexhaustible patience and good humor in helping me navigate the world of C++ compilation and software compatibility.

References

[1] Peer-Timo Bremer, Gunther Weber, Julien Tierny, Valerio Pascucci, Marc Day, and John Bell. Interactive exploration and analysis of large-scale simulations using topology-based data segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 17(9):1307–1324, 2011.

[2] Shen Dong, Scott Kircher, and Michael Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer aided geometric design*, 22(5):392–423, 2005.

[3] Felix Kälberer, Matthias Nieser, and Konrad Polthier. Quadcover-surface parameterization using branched coverings. In *Computer graphics forum*, volume 26, pages 375–384. Wiley Online Library, 2007.

[4] Felix Kälberer, Matthias Nieser, and Konrad Polthier. Stripe parameterization of tubular surfaces. In *Topological Methods in Data Analysis and Visualization*, pages 13–26. Springer, 2011.

[5] Vladimir Kolmogorov. Blossom v: a new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1):43–67, 2009.

[6] Frauke Liers and Gregor Pardella. A simple max-cut algorithm for planar graphs. In *CTW*, pages 351–354, 2009.

[7] Romulo Pinho, Jan Sijbers, and Toon Huysmans. Segmentation of the human trachea using deformable statistical models of tubular shapes. In *Proceedings of the 9th International Conference on Advanced Concepts for Intelligent Vision Systems*, ACIVS'07, pages 531–542, Berlin, Heidelberg, 2007. Springer-Verlag.

[8] Amanda Randles, Erik W. Draeger, and Peter E. Bailey. Massively parallel simulations of hemodynamics in the primary large arteries of the human vasculature. *Journal of Computational Science*, 9:70 – 75, 2015. Computational Science at the Gates of Nature.

[9] Nicol N Schraudolph and Dmitry Kamenetsky. Efficient exact inference in planar ising models. In *Advances in Neural Information Processing Systems*, pages 1417–1424, 2009.

[10] Sameer Sharma, S Vinuchakravarthy, and S J Subramanian. Estimation of surface curvature from full-field shape data using principal component analysis. *Measurement Science and Technology*, 28(1):015003, 2017.