Details of Computation

Stephen Jasina, Loqman Salamatian, Joshua Mathews, Scott Anderson,
Paul Barford, Mark Crovella, and Walter Willinger

This document details the strategy for computing important quantities on a mesh, like curvature, as well as the loss functions based on these quantities. Furthermore, details of computing gradients of these quantities are presented.

Due to the necessity of much more notation than in the main paper, some choices of letters will differ.

0.1 Inputs and Outputs

As input to the optimization process, we take an weighted undirected graph $G = (V_G, E_G)$. Each vertex $s \in V_G$ represents a node in a network and is annotated with location information. We therefore can compute quantities like $\mathrm{GCL}(s, s')$, the Great Circle Latency between s and s'.

Additionally, an edge $\{s, s'\} \in E_G$ has an associated measured Round Trip Time $\mathrm{RTT}(s, s')$. In practice, latencies are collected for almost every pair of nodes, so G is nearly complete.

We also take several hyperparameters. First is the residual latency threshold ε. Next are the λ's, which are weighting parameters for each component of the loss function. Finally, there are a few hyperparameters describing the structure of the mesh.

We return a triangle mesh $M = (V_M, E_M)$, stored in a doubly connected edge list format. That is, each edge is actually stored as a pair of directed edges, except for on the boundary, where only a single directed edge is used. These directed edges trace out each face of the mesh counterclockwise.

As hinted at above, the actual vertex-edge connectivity of the mesh is to be selected before running the optimization. That said, each vertex has coordinates in $\mathbb{R}^3$ which are to be chosen by the optimization algorithm. For the purposes of the algorithm and mesh regularity, we parameterize each vertex position with a single number. In our current implementation, each vertex $v \in V_M$ can be broken into parts as (p_v, z_v), where p_v is a latitude-longitude pair, and z_v is an altitude. The optimization algorithm then determines the best z values.

Note that the above parameterization implies we can map vertices $s \in V_G$ to positions on the mesh $\pi(s)$. For the purposes of the following computations, assume the stronger statement $\pi(s) \in V_M$. While this assumption is not strictly necessary, it significantly simplifies the geodesic distance computation. Furthermore, provided our mesh is fine enough, the stronger assumption will lead to minimal numerical error.

0.2 Laplacian

Here and in the subsequent sections, computations can take serious advantage of vectors and matrices. Therefore, while notationally inelegant, we will assign indices to the vertices in V_M.

On that note, if i and j are two indices for which $(v_i, v_j) \in E_M$, let $\mathrm{nxt}(i, j)$ be the index such that $v_i \to v_j \to v_{\mathrm{nxt}(i,j)}$ traces a triangle counterclockwise. If $\mathrm{nxt}(i, j)$ does not exist, then the half-edge (v_i, v_j) lies on the boundary.

We also write ∂M to represent the boundary of our mesh. Abusing notation, we can write $v_i \in \partial M$ and $(v_i, v_j) \in \partial M$ to denote that a vertex or an edge is on the boundary, respectively.

In this and the following sections, we will lay out what each variable is before the details of computing it. We also give the (total) runtime in big-O notation to compute the set of variables and its gradients. For the Laplacian, we have:

$N_{i,j}$	Outward normal of triangle $v_i \to v_j \to v_{\mathrm{nxt}(i,j)}$	$O(\lvert V_M \rvert)$
$A_{i,j}$	Area of triangle $v_i \to v_j \to v_{\mathrm{nxt}(i,j)}$	$O(\lvert V_M \rvert)$
$D_{i,j}$	Vertex triangle areas; diagonal	$O(\lvert V_M \rvert)$
$\theta_{i,j}$	Measure of $\angle v_i v_{\mathrm{nxt}(i,j)} v_j$	$O(\lvert V_M \rvert)$
L_C^{N}	Cotangent operator with zero-Neumann boundary condition	$O(\lvert V_M \rvert)$
L_C^{D}	Cotangent operator with zero-Dirichlet boundary condition	$O(\lvert V_M \rvert)$

0.2.1 Forward Computation

We have the following (standard) definition of the Laplace-Beltrami operator on a mesh:

$$N_{i,j} = \left(v_i - v_{\mathrm{nxt}(i,j)}\right) \times \left(v_j - v_{\mathrm{nxt}(i,j)}\right),$$
$$A_{i,j} = \frac{1}{2}\left\|N_{i,j}\right\|_2,$$

$$D_{i,j} = \begin{cases} \frac{1}{3} \sum\limits_{\substack{k \\ (v_i,v_k)\in E_M}} A_{i,k} & \text{if } i=j, \\ 0 & \text{otherwise,} \end{cases}$$

$$\cot\Big(\theta_{i,j}\Big) = \frac{\Big(v_i - v_{\mathrm{nxt}(i,j)}\Big)\cdot\Big(v_j - v_{\mathrm{nxt}(i,j)}\Big)}{2A_{i,j}},$$

$$\Big(L_C^{\mathrm{N}}\Big)_{i,j} = \begin{cases} \frac{1}{2}\cot\Big(\theta_{i,j}\Big) & \text{if } \Big(v_i,v_j\Big)\in\partial M, \\ \frac{1}{2}\cot\Big(\theta_{j,i}\Big) & \text{if } \Big(v_j,v_i\Big)\in\partial M, \\ \frac{1}{2}\Big(\cot\Big(\theta_{i,j}\Big)+\cot\Big(\theta_{j,i}\Big)\Big) & \text{if } \Big(v_i,v_j\Big),\Big(v_j,v_i\Big)\in E_M, \\ -\frac{1}{2}\left(\sum\limits_{\substack{k \\ (v_i,v_k)\in E_M}} \cot\Big(\theta_{i,k}\Big) + \sum\limits_{\substack{k \\ (v_k,v_i)\in E_M}} \cot\Big(\theta_{k,i}\Big)\right) & \text{if } i=j, \\ 0 & \text{otherwise,} \end{cases}$$

$$\Big(L_C^{\mathrm{D}}\Big)_{i,j} = \begin{cases} \frac{1}{2}\Big(\cot\Big(\theta_{i,j}\Big)+\cot\Big(\theta_{j,i}\Big)\Big) & \text{if } \Big(v_i,v_j\Big)\in E_M,\ v_i\notin\partial M, \text{ and } v_j\notin\partial M, \\ -\frac{1}{2}\sum\limits_{\substack{k\notin\partial M \\ (v_i,v_k)\in E_M \\ (v_k,v_i)\in E_M}} \Big(\cot\Big(\theta_{i,k}\Big)+\cot\Big(\theta_{k,i}\Big)\Big) & \text{if } i=j \text{ and } v_i\notin\partial M, \\ 0 & \text{otherwise.} \end{cases}$$

0.2.2 Reverse Computation

We compute

$$\frac{\partial v_i}{\partial z_\ell} = \begin{cases} e_3 & \text{if } \ell = i, \\ 0 & \text{otherwise,} \end{cases}$$

$$\frac{\partial N_{i,j}}{\partial z_\ell} = \begin{cases} \Big(v_{\mathrm{nxt}(i,j)} - v_j\Big)\times\frac{\partial v_\ell}{\partial z_\ell} & \text{if } \ell = i, \\ \Big(v_i - v_{\mathrm{nxt}(i,j)}\Big)\times\frac{\partial v_\ell}{\partial z_\ell} & \text{if } \ell = j, \\ \Big(v_j - v_i\Big)\times\frac{\partial v_\ell}{\partial z_\ell} & \text{if } \ell = \mathrm{nxt}\big(i,j\big), \\ 0 & \text{otherwise,} \end{cases}$$

$$\frac{\partial A_{i,j}}{\partial z_\ell} = \frac{1}{4A_{i,j}} N_{i,j}\cdot\frac{\partial N_{i,j}}{\partial z_\ell},$$

$$\left(\frac{\partial D}{\partial z_\ell}\right)_{i,j} = \begin{cases} \frac{1}{3}\sum\limits_{\substack{k \\ (v_i,v_k)\in E_M}} \dfrac{\partial A_{i,k}}{\partial z_\ell} & \text{if } i=j, \\ 0 & \text{otherwise,} \end{cases}$$

$$\frac{\partial}{\partial z_\ell}\cot\Big(\theta_{i,j}\Big) = \begin{cases} \dfrac{\Big(v_j - v_{\mathrm{nxt}(i,j)}\Big)\cdot\frac{\partial v_\ell}{\partial z_\ell} - 2\cot\Big(\theta_{i,j}\Big)\frac{\partial A_{i,j}}{\partial z_\ell}}{2A_{i,j}} & \text{if } \ell = i, \\ \dfrac{\Big(v_i - v_{\mathrm{nxt}(i,j)}\Big)\cdot\frac{\partial v_\ell}{\partial z_\ell} - 2\cot\Big(\theta_{i,j}\Big)\frac{\partial A_{i,j}}{\partial z_\ell}}{2A_{i,j}} & \text{if } \ell = j, \\ \dfrac{\Big(2v_{\mathrm{nxt}(i,j)} - v_i - v_j\Big)\cdot\frac{\partial v_\ell}{\partial z_\ell} - 2\cot\Big(\theta_{i,j}\Big)\frac{\partial A_{i,j}}{\partial z_\ell}}{2A_{i,j}} & \text{if } \ell = \mathrm{nxt}\big(i,j\big), \\ 0 & \text{otherwise,} \end{cases}$$

$$\left(\frac{\partial L_C^{\mathrm{N}}}{\partial z_\ell}\right)_{i,j} = \begin{cases} \frac{1}{2}\frac{\partial}{\partial z_\ell}\cot\left(\theta_{i,j}\right) & \text{if } \left(v_i, v_j\right) \in \partial M, \\ \frac{1}{2}\frac{\partial}{\partial z_\ell}\cot\left(\theta_{j,i}\right) & \text{if } \left(v_j, v_i\right) \in \partial M, \\ \frac{1}{2}\left(\frac{\partial}{\partial z_\ell}\cot\left(\theta_{i,j}\right) + \frac{\partial}{\partial z_\ell}\cot\left(\theta_{j,i}\right)\right) & \text{if } \left(v_i, v_j\right), \left(v_j, v_i\right) \in E_M, \\ -\frac{1}{2}\left(\sum\limits_{\substack{k \\ \left(v_i, v_k\right) \in E_M}} \frac{\partial}{\partial z_\ell}\cot\left(\theta_{i,k}\right) + \sum\limits_{\substack{k \\ \left(v_k, v_i\right) \in E_M}} \frac{\partial}{\partial z_\ell}\cot\left(\theta_{k,i}\right)\right) & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases}$$

$$\left(\frac{\partial L_C^{\mathrm{D}}}{\partial z_\ell}\right)_{i,j} = \begin{cases} \frac{1}{2}\left(\frac{\partial}{\partial z_\ell}\cot\left(\theta_{i,j}\right) + \frac{\partial}{\partial z_\ell}\cot\left(\theta_{j,i}\right)\right) & \text{if } \left(v_i, v_j\right) \in E_M,\ v_i \notin \partial M, \text{ and } v_j \notin \partial M, \\ -\frac{1}{2}\sum\limits_{\substack{k \notin \partial M \\ \left(v_i, v_k\right) \in E_M \\ \left(v_k, v_i\right) \in E_M}} \left(\frac{\partial}{\partial z_\ell}\cot\left(\theta_{i,k}\right) + \frac{\partial}{\partial z_\ell}\cot\left(\theta_{k,i}\right)\right) & \text{if } i = j \text{ and } v_i \notin \partial M, \\ 0 & \text{otherwise.} \end{cases}$$

Note that a runtime of $O\left(|V_M|\right)$ is achievable for these computations by using an *accumulation* strategy and iterating over vertices, half-edges, or faces where appropriate.

0.3 Curvature

We will define the following:

E_G^ε	The set of network edges at threshold ε	
κ_e^{R}	The Ollivier-Ricci curvature of the edge E_G^ε	
$\widetilde{\kappa^{\mathrm{G}}}_i$	The discrete Gaussian curvature at v_i, scaled by vertex area	$O\left(\lvert V_M\rvert\right)$
κ_i^{G}	The discrete Gaussian curvature at v_i	$O\left(\lvert V_M\rvert\right)$
N_i	An outward pointing vector at v_i	$O\left(\lvert V_M\rvert\right)$
$\widetilde{\kappa^{\mathrm{H}}}_i$	The mean curvature normal at v_i	$O\left(\lvert V_M\rvert\right)$
κ_i^{H}	The mean curvature at v_i	$O\left(\lvert V_M\rvert\right)$
κ_i^+	The first principal curvature at v_i	$O\left(\lvert V_M\rvert\right)$
κ_i^-	The second principal curvature at v_i	$O\left(\lvert V_M\rvert\right)$

0.3.1 Ollivier-Ricci Curvature

We use the GraphRicciCurvature library [2] to compute the Ollivier-Ricci curvatures of the edges of the graph $\left(V_G, E_G^\varepsilon\right)$. Here, $E_G^\varepsilon \subseteq E_G$ is the set of edges whose RTTs are at most ε milliseconds higher than their GCLs.

Note that κ_e^{R} is then only defined for edges in E_G^ε, as opposed to being defined for all edges in E_G. We elide the ε to reduce notational density.

0.4 Forward Computation

For these computations (particularly the mean curvature one), consider v as a matrix of vertex positions, where each row corresponds to a vertex (so v is of shape $|V_M| \times 3$). We will also use e_i to denote the ith standard basis vector. We have

$$\begin{aligned} \theta_{i,j} &= \arctan\left(\frac{1}{\cot\left(\theta_{i,j}\right)}\right) \bmod \pi, \\ \widetilde{\kappa^{\mathrm{G}}}_i &= 2\pi - \sum_{\substack{k \\ \left(v_i, v_k\right) \in E_M}} \theta_{k,c(i,k)}, \\ \kappa^{\mathrm{G}} &= D^{-1}\widetilde{\kappa^{\mathrm{G}}}, \\ \widetilde{N}_i &= \sum_{\substack{k \\ \left(v_i, v_k\right) \in E_M}} N_{i,k}, \\ \widetilde{\kappa^{\mathrm{H}}}_i &= -\frac{1}{2} e_i^{\mathsf{T}} D^{-1} L_C^{\mathrm{N}} v, \\ \kappa_i^{\mathrm{H}} &= \operatorname{sgn}\left(\widetilde{N}_i^{\mathsf{T}}\widetilde{\kappa^{\mathrm{H}}}_i\right)\left\|\widetilde{\kappa^{\mathrm{H}}}_i\right\|_2, \\ \kappa_i^+ &= \kappa_i^{\mathrm{H}} + \sqrt{\left(\kappa_i^{\mathrm{H}}\right)^2 - \kappa_i^{\mathrm{G}}}, \end{aligned}$$

$$\kappa_i^- = \kappa_i^{\mathrm{H}} - \sqrt{\left(\kappa_i^{\mathrm{H}}\right)^2 - \kappa_i^{\mathrm{G}}}.$$

0.4.1 Reverse Computation

Differentiating,

$$\begin{aligned}
\frac{\partial \theta_{i,j}}{\partial z_\ell} &= -\frac{\partial \cot\left(\theta_{i,j}\right)}{\partial z_\ell} \cdot \frac{1}{1+\cot^2\left(\theta_{i,j}\right)}, \\
\frac{\partial \widetilde{\kappa^{\mathrm{G}}}_i}{\partial z_\ell} &= -\sum_{\substack{k \\ \left(v_i,v_k\right)\in E_M}} \frac{\partial \theta_{k,c(i,k)}}{\partial z_\ell}, \\
\frac{\partial \kappa^{\mathrm{G}}}{\partial z_\ell} &= D^{-1}\left(\frac{\mathrm{d}\widetilde{\kappa^{\mathrm{G}}}}{\partial z_\ell} - \frac{\mathrm{d}D}{\partial z_\ell}\kappa^{\mathrm{G}}\right), \\
\frac{\partial \widetilde{N}_i}{\partial z_\ell} &= \sum_{\substack{k \\ \left(v_i,v_k\right)\in E_M}} \frac{\partial N_{i,k}}{\partial z_\ell}, \\
\frac{\partial \widetilde{\kappa^{\mathrm{H}}}_i}{\partial z_\ell} &= -\frac{1}{2} e_i^{\mathsf{T}} D^{-1}\left(\left(\frac{\partial L_C^{\mathrm{N}}}{\partial z_\ell} - \frac{\partial D}{\partial z_\ell} D^{-1} L_C^{\mathrm{N}}\right) v + L_C^{\mathrm{N}} \frac{\partial v}{\partial z_\ell}\right), \\
\frac{\partial \kappa_i^{\mathrm{H}}}{\partial z_\ell} &= \frac{\operatorname{sgn}\left(\widetilde{N}_i^{\mathsf{T}} \widetilde{\kappa^{\mathrm{H}}}_i\right)}{\left\|\widetilde{\kappa^{\mathrm{H}}}_i\right\|_2} \widetilde{\kappa^{\mathrm{H}}}_i^{\mathsf{T}} \frac{\partial \widetilde{\kappa^{\mathrm{H}}}_i}{\partial z_\ell}, \\
\frac{\partial \kappa_i^+}{\partial z_\ell} &= \frac{2\kappa_i^+ \frac{\partial \kappa_i^{\mathrm{H}}}{\partial \rho_i} - \frac{\partial \kappa_i^{\mathrm{G}}}{\partial \rho_i}}{\kappa_i^+ - \kappa_i^-}, \\
\frac{\partial \kappa_i^-}{\partial z_\ell} &= \frac{\frac{\partial \kappa_i^{\mathrm{G}}}{\partial \rho_i} - 2\kappa_i^- \frac{\partial \kappa_i^{\mathrm{H}}}{\partial \rho_i}}{\kappa_i^+ - \kappa_i^-}.
\end{aligned}$$

0.5 Curvature Loss

We will define the following:

$B_r(e)$	The ball of radius r around e	$O\left(\left\|E_G\right\| \cdot \left\|V_M\right\|\right)$
$\mathscr{L}_{\text{curvature}}(M)$	Sum of squares of the differences between vertices actual and desired curvatures	$O\left(\left\|E_G\right\| \cdot \left\|V_M\right\|\right)$

Note that, in practice, the runtimes here will be significantly lower than their upper bounds since balls around edges will not typically contain a large proportion of the mesh's vertices. For example, a square mesh (as described in the main paper) with a relatively small ε will have the runtimes here be closer to $O\left(\left|E_G\right|\sqrt{\left|V_M\right|}\right)$.

0.5.1 Determining the Ball Around an Edge

Before tackling the loss functional, we must determine what it means for a point to be close to an edge. Suppose s and s' are vertices in V_G, and let $e = \left\{s, s'\right\}$. Recall that p_s (similarly $p_{s'}$) is the point in $\mathbb{R}^2$ corresponding to s. Define $\pi(e)$ to be the edge connecting p_s and $p_{s'}$. We say that $v \in B_r(e)$ when p_v is within distance r of $\pi(e)$. Figure 2 shows this setup.

To determine whether a vertex is in the ball, we break the problem into three parts:

- Determine whether $p_v \in B_r\left(p_s\right)$. This is the case when $\left\|p_v - p_s\right\|_2 < r$.
- Determine whether the distance from p_v to $\pi(e)$ is less than r. To do this, we first project p_v onto the line containing $\pi(e)$ to get $p_{v,e}$:

$$p_{v,e} = p_s + \frac{\left(p_{s'} - p_s\right)^{\mathsf{T}}\left(p_v - p_s\right)}{\left\|p_{s'} - p_s\right\|_2^2}\left(p_{s'} - p_s\right).$$

  To determine whether this projection lies on the actual segment and that the projection is not too far away from the original point, we simultaneously check the three conditions

$$\begin{aligned}
\left(p_v - p_s\right)^{\mathsf{T}}\left(p_{s'} - p_s\right) &\geq 0, \\
\left(p_v - p_{s'}\right)^{\mathsf{T}}\left(p_s - p_{s'}\right) &\geq 0, \\
\left\|p_v - p_{v,e}\right\|_2 &< r.
\end{aligned}$$

- Determine whether $p_v \in B_r\left(p_{s'}\right)$. This is the case when $\left\|p_v - p_{s'}\right\|_2 < r$.

If any of the three above parts yields a positive response, then $v \in B_r(e)$.

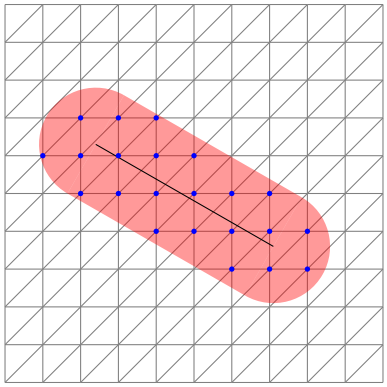

Fig. 2: An overhead visualization of $B_r(e)$. The mesh is in gray, $\pi(e)$ is in black, and $B_r(e)$ is in blue.

0.5.2 Forward Computation

We have

$$\mathscr{L}_{\text{curvature}}(M) = \frac{1}{\left|E_G^{\varepsilon}\right|} \sum_{e \in E_G^{\varepsilon}} \sum_{\substack{k \\ v_k \in B_r(e)}} \left(\kappa_e^{\text{R}} - \kappa_k^{\text{G}}\right)^2.$$

0.5.3 Reverse Computation

Differentiating,

$$\frac{\partial\left(\mathscr{L}_{\text{curvature}}(M)\right)}{\partial \rho_\ell} = \frac{1}{\left|E_G\right|} \sum_{e \in E_G} \sum_{\substack{k \\ v_k \in B_\varepsilon(e)}} -2\left(\kappa_e^{\text{R}} - \kappa_k^{G}\right) \frac{\partial \kappa_k^{G}}{\partial z_\ell}.$$

0.6 Smoothness Loss

We will define the following:

$$\mathscr{L}_{\text{smooth}}(M) \;\Big|\; \text{The surface area independent MVS}_{\text{cross}} \text{ energy [1]} \;\Big|\; O\left(|V_M|\right)$$

0.6.1 Forward Computation

We use

$$\mathscr{L}_{\text{smooth}}(M) = \left(-\left(\kappa^{+}\right)^{\intercal} L_C^{\text{D}} \kappa^{+} - \left(\kappa^{-}\right)^{\intercal} L_C^{\text{D}} \kappa^{-}\right) \sum_{\left(v_i, v_j\right) \in E_M} A_{i,j}.$$

0.6.2 Reverse Computation

Differentiating, and using matrix symmetry,

$$\begin{aligned}\frac{\partial\left(\mathscr{L}_{\text{smooth}}(M)\right)}{\partial z_\ell} = &\left(-\left(\kappa^{+}\right)^{\intercal}\left(\frac{\partial L_C^{\text{D}}}{\partial z_\ell}\kappa^{+} + 2L_C^{\text{D}}\frac{\partial \kappa^{+}}{\partial z_\ell}\right) - \left(\kappa^{-}\right)^{\intercal}\left(\frac{\partial L_C^{\text{D}}}{\partial z_\ell}\kappa^{-} + 2L_C^{\text{D}}\frac{\partial \kappa^{-}}{\partial z_\ell}\right)\right) \sum_{\left(v_i, v_j\right) \in E_M} A_{i,j} \\ &+ \left(-\left(\kappa^{+}\right)^{\intercal} L_C^{\text{D}} \kappa^{+} - \left(\kappa^{-}\right)^{\intercal} L_C^{\text{D}} \kappa^{-}\right) \sum_{\left(v_i, v_j\right) \in E_M} \frac{\partial A_{i,j}}{\partial z_\ell}.\end{aligned}$$

References

[1] P. Joshi and C. H. Séquin. Energy minimizers for curvature-based surface functionals. *Computer-aided Design and Applications*, 4:607–617, 2007. 5

[2] C.-C. Ni, Y.-Y. Lin, F. Luo, and J. Gao. Community detection on networks with ricci flow. *Scientific reports*, 9(1):1–12, 2019. 3