Nonisometric Flows on Planar Curves via Reduced Coordinates

ALEXANDER HAVRILLA, Carnegie Mellon University

This project generalizes part of the work of Crane, Pinkall, Schroder in [2] on computing flows via reduced coordinates. We find nonisometric flows less stable than the isometric version, particularly on large inputs. A scaling term on our length update rule is introduced to mitigate instability and compare edge length metrics, robustness, and aesthetic.

ACM Reference Format:
Alexander Havrilla. 2020. Nonisometric Flows on Planar Curves via Reduced Coordinates. 1, 1 (December 2020), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 BACKGROUND

We begin by introducing the ideas behind the field of differential geometry and how it relates to this project.

1.1 Discrete Differential Geometry

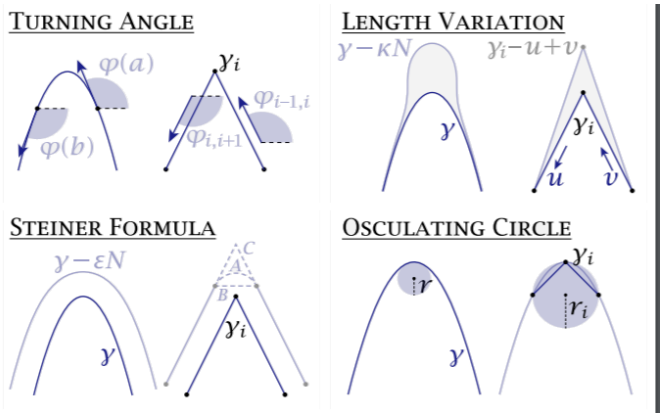
Ironically the study of discrete differential geometry is best motivated by the continuous case. Differential Geometry seeks to study properties of different structures on which calculus can be developed. Usually the setting is a *smooth* domain called a *manifold* which locally, ie. in a small ball around a point, looks like euclidean space. This allows us to port results from calculus in a natural fashion onto manifolds. Often times framing a problem in terms of differential geometry is useful because it removes the use of cartesian coordinates, which often obfuscate the problem, and gives access to many deep and powerful theorems, giving insight into efficient algorithms.

Discrete differential geometry then seeks to translate the results from differential geometry into a framework we can do computation with. Often times however there are many equivalent definitions for a quantity in the smooth setting that, when discretized, don't give the same properties. A great example of this is curvature of a curve, ie. how much it bends.

In the smooth setting curvature is defined as the second second derivative our parameterization(assuming arclength): Let $\gamma : [0, 1] \to \mathbb{R}^2$ be a parameterization of a curve. Then curvature κ is

$$|\gamma''(s)| = |T'(s)| = \kappa(s)$$

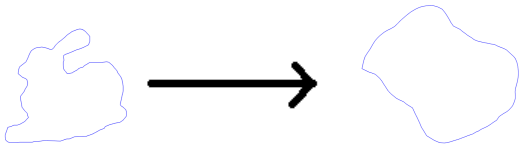
Yet when we turn to the discrete picture, there are many ways with think about discretizing curvature:



None of these notions are equivalent in the discrete setting, though they are continuously. Thus our choice of definition affects properties of our implementation.

1.2 Curvature Flows

This paper studies curvature flows which serve as a great example for the kinds of tradeoffs encountered when choosing a definition. A curvature flow on a planar curve $\gamma : [0, 1] \to \mathbb{R}^2$ can intuitively be thought of as a process which gradually transforms the flow to minimize curvature, ie. make it "smoother". Generally this is done via a minimization of an *energy* E which takes in a curve γ and returns a real number, a function on curves.



The definition of curvature we take here it the turning angle definition. In particular this paper studies generalizations of the isometric wilmore flow to the nonisometric case which smooths a curve by minimizing the energy functional

$$E_w[\gamma] = \int_0^1 \kappa^2 dl$$

which when discretized becomes

$$E_w[\gamma] = \sum_{i \in \gamma} \frac{2\kappa_i^2}{l_{i-1,i} + l_{i,i+1}}$$

where l_{ij} is the length of an edge from vertex i to vertex j. A flow is isometric if it preserves preserves edge lengths on each iteration. Nonisometric flows allow for these lengths to change.

The isometric case has been well studied in [2]. Our project considers the more general nonisometric wilmore flow and two other flows minimizing the following energies respectively:

Author's address: Alexander Havrilla, alumhavr@andrew.cmu.edu, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburg, Pennsylvania, 15213.

XXXX-XXXX/2020/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

$$E_{sl}[\gamma] = E_w[\gamma] + \sum_{i\in\gamma} l_i^2$$
$$E_{sc}[\gamma] = \sum_{i\in\gamma} \kappa_i^2$$

We study this by starting with the isometric algorithm provided in [2] and modifying it to compute gradients and constraints involving changing edge lengths. Further we compute gradients and edge lengths for the two additionally proposed energies and compare the resulting flows in stability/robustness, convergence, edge length variation, and aesthetic.

2 IMPLEMENTATION

Each iteration of the flow can be broken down as follows:

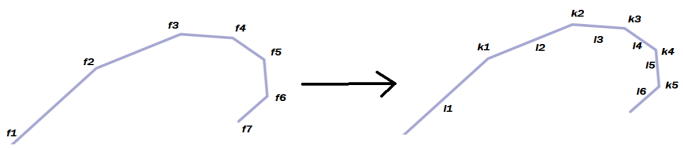
1. Evaluate lengths/curvatures
2. Calculate gradient ∇ of $E[\gamma]$
3. Project ∇ for closedness and 2π curvature
4. Update lengths/curvatures via ∇ with step h
5. Solve Poisson equation to minimize discrete error

We implemented this flow using a modified existing framework in Javascript from the CMU's discrete differential geometry class.

2.1 Evaluating Reduced Coordinates

Initially the curve is represented as a set of vertices in an obj file. Once read in a geometry is constructed by making a halfedge mesh, which can be thought of as a doubly linked list on a curve. To convert the pointwise representation of each curve to reduced coordinates, ie. in terms of lengths and angles, it suffices to iterate around the curve, and at each point compute curvature via a dot product calculation.

Note that the curve is defined up a global rotation and translation by these reduced coordinates.



2.2 Computing the Gradient

The particular gradient being computed is dependent on the flow being performed, ie. energy being minimized. We consider the energies

$$E_w[\gamma] = \sum_{i\in V} \frac{2\kappa_i^2}{l_{i-1,i}, l_{i,i+1}}$$
$$E_{sl}[\gamma] = \sum_{i\in V} \frac{2\kappa_i^2}{l_{i-1,i}, l_{i,i+1}} + \sum_{i\in V} l_i^2$$
$$E_{sc}[\gamma] = \sum_{i\in V} \kappa_i^2$$

with corresponding gradients $\nabla E_w, \nabla E_{sl}, \nabla E_{sc}$ which we compute by hand and evaluate pointwise.

2.3 Computing and Applying Constraints

Because we are working with closed curves, we must apply constraints to ensure the curve remains closed after an update is applied. In particular we want

$$\sum_{i\in V} \kappa_i = 2\pi$$

which ensures total curvature is always 2π. This prevents intersections of two edges.

$$\int_0^1 \kappa' f dl = 0 \implies \sum_{i\in V} l_i T_i = 0$$

ensures endpoints meet as we sum all tangent vectors as we traverse the curve, the result should be 0(we should end up where we started).

To apply these constraints we calculate the gradient of each and and form an orthonormal basis which we project the gradient of our curve onto. Note that if we have n vertices, this provides $2n$ length and curvature variables. The above conditions provide 3 scalar equations. Hence we are projecting our gradient of a surface in $\mathbb{R}^{2n}$ onto a three dimensional subspace. Once the projection is done we subtract it from the curve gradient which removes any changes in a direction that violates closedness or self-intersection.

2.4 Updating Reduced Coordinate Representation

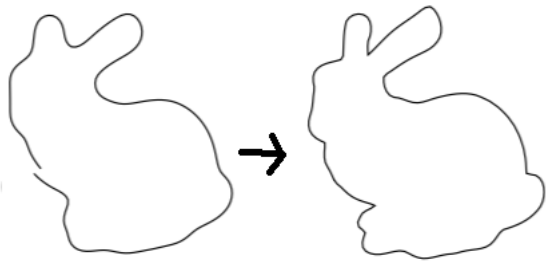
Once we have our constrained gradient, we update the curvatures and lengths using a forward euler scheme, ie. gradient descent:

$$(l^{k+1}, \kappa^{k+1}) = (l^k, \kappa^k) - h\nabla E$$

on the kth iteration where h is our timestep. In fact we introduce a scaling term $g > 0$ on the updates to l in the nonisometric case to increase stability. This will be discussed in our analysis.

2.5 Reconstruction with Error Minimization

Despite applying constraints ensuring closedness, due to the discrete nature of computation we may still end up with curve that fails to remain closed. To close the curve, we find a closest approximation with presecribed tangent vectors which is closed:



We do this by solving a linear poisson equation using LU decomposition $Lf = b$ where the the right hand side is given by $b_i = T_{i-1,i}/l_{i-1,i} - T_{i,i+1}/l_{i,i+1}$ where T_j is a prescribed tangent along the curve. The entries of L are given by $L_{ii} = 1/l_{i-1,i} + 1/l_{i,i+1}$ on the diagonal and $L_{ij} = -1/l_{ij}$ otherwise.

Note that each iteration is $O(n)$ where n is the number of vertices. In particular the last step invovling a linear system solve can be done in $O(n)$ time, where n is the number of vertices in the curve.

3 RESULTS

We focus mostly on the nonisometric case, constrasting to the isometric case. In evaluating the nonisometric flows we examine stability, robustness to different inputs, speed of convergence, edge length change, and aesthetic. It should be noted I was not able to resolve the cusp issues in this report. Once these are resolved this will likely impact stability, edge length change, and aesthetic.

In general we found that the nonisometric flow performs as expected on small inputs with a less than 50 vertices. As input sizes get larger stability starts to break down forcing us to take smaller step sizes. To compensate for this we introduced a penalty term $g > 0$ on our updates to lengths:

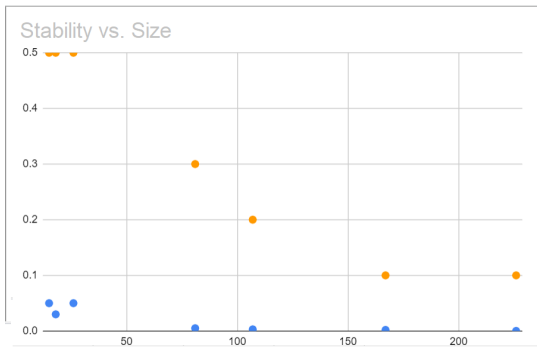
$$(l^{k+1}) = (l^k) + h * g * \nabla E_l$$

This prioritizes curvature update over length update to promote stability. Note larger changes in edge length lead to lower stability because large changes in length make it harder to retain the closure condition.

Recall that we consider the wilmore energy E_w, the squared lengths energy E_{sl}, and the squared curvatures energy E_{sc}. The squared lengths curvatures turns out to be very similar to the isometric case as the energy only involves the curvatures κ_i which leads to only updating curvature. In our analysis we will only compare the the wilmore energy flow E_w to this squared curvatures energy E_{sc}. However this can also be thought of as an implicit comparison of the nonisometric flow E_w with the isometric version. Throughout the comparison orange data points represent the squared curvature/isometric case and blue data points represent the nonisometric wilmore flow case.

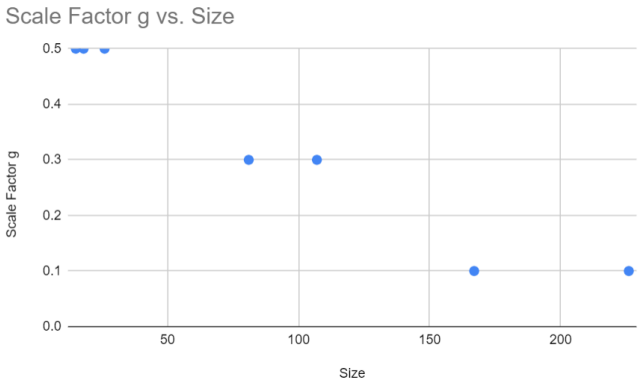
3.1 Stability and Robustness

We found that the stability of the flow is highly dependent on the size of the input mesh. This is particularly true in the nonisometric case, whereas we see the isometric case is more robust. Note that we measure stability as the largest time step h we can take without breaking constraints.



As mentioned above we expect this is due to the compounding nature of updating edge lengths. As input size increases this necessarily leads to a decrease in maximal step size h to maintain stability. In general time steps above .1 are quite large and lead to fast rates of convergence as will be discussed below.

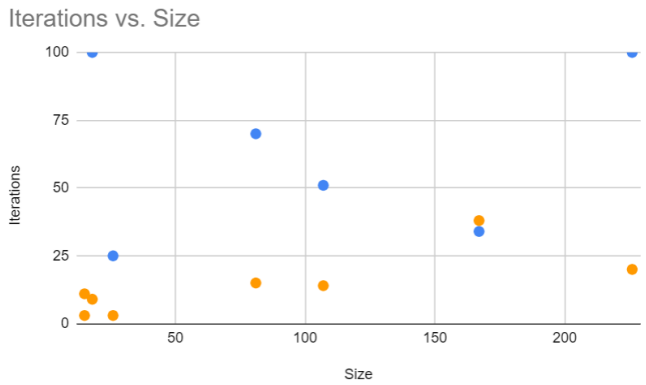
Note further that the maxmial scale factor g drops as input size increases too. We only include scale factor for nonisometric flow points:



Empirically this suggests we halve the scale factor when input increases by an order of magnitude.

3.2 Convergence

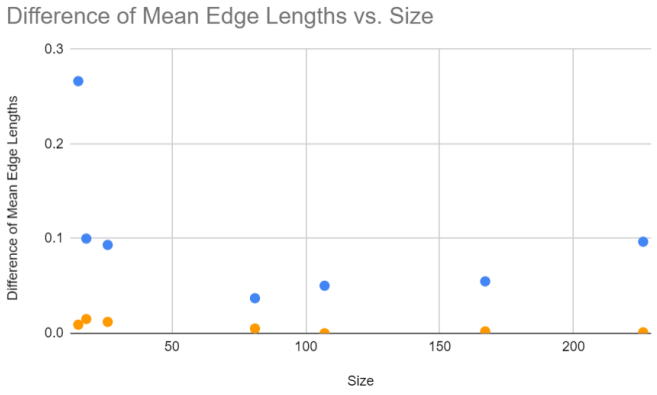
Here we define convergence to be the number of iterations at which we converge to a local optimum, or we fail to decrease the energy by more than $10e-2$. The wilmore flow is degenerate in the nonisometric case in the sense that we can increase edge lengths without bound and decrease the energy asymptotically to 0. In this case we terminate after we fail to decrease by more than $10e-2$ or use another heuristic to be discussed below. Because of this degeneracy and the necessarily smaller time step and scale factor, this leads to a substantially larger number of iterations to converge relative to the isometric case:



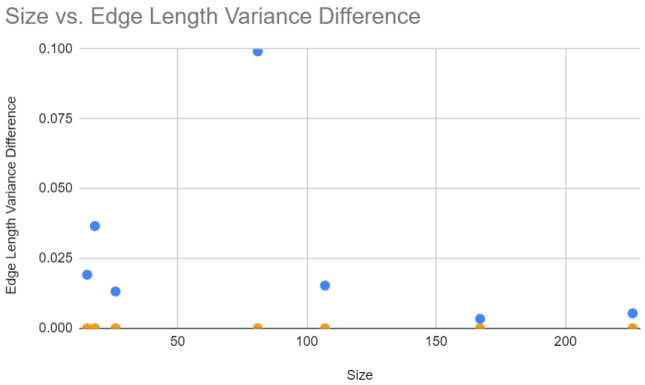
This seems to only weakly suggest that increasing input size increasing input size also increases the number of iterations to convergence. However in several cases, particularly for small inputs, the wilmore flow attains a reasonably good smoothness for the input curve, which will be shown aesthetically, but continues to run due to the inherent degeneracy(anything running for more than 100 iterations can be regarded as effectively converged). What is clear is the robustness the squared curvatures flow has to input size relative to the number of iterations needed to converge. It also does not seem to be as susceptible to local optima as the nonisometric flow.

3.3 Edge Lengths

In practice if this is the case we can choose a target "smoothness" and run our flow until this attained. This is target is set by examining the variance of the edge lengths vs. the mean edge length size and running until variance is small with a target mean edge length, then rescaling. First we examine change in mean edge length via the nonisometric flow vs squared curvatures. This also addresses the natural question of how much edge lengths change over the course of a flow and justifies the assertion the squared curvatures flow behaves similarly to the isometric flow:



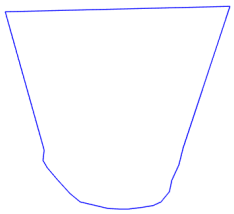
We observe that the nonisometric flow changes the edge lengths orders of magnitudes more than squared curvatures. Furthermore edge lengths only strictly increase, as expected. Note that this is edge length change recorded after convergence. We also examine the edge length variance difference before and after flow:



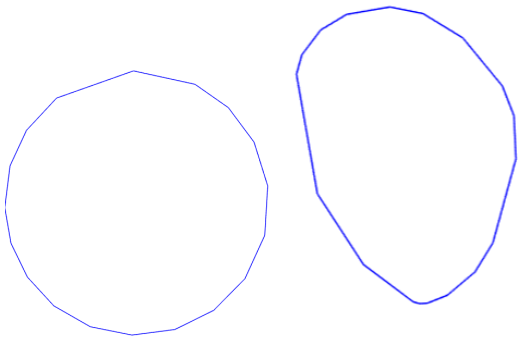
With one exception we see variation increases little relative to the increase in mean edge length. Furthermore this seems to be invariant across input size. This suggests the nonisometric wilmore flow acts effectively "smooths out" lengths in addtion to curvatures by decreasing the relative difference in disparate edge lengths each iteration. This results in a smoother overall result which can be rescaled as necessary.

3.4 Aesthetic

Finally we examine the results of the wilmore flow and squared curvatures flow visually. First consider the following small input:



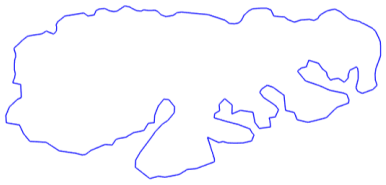
Respectively the nonisometric wilmore flow and squared curvatures result in respectively:



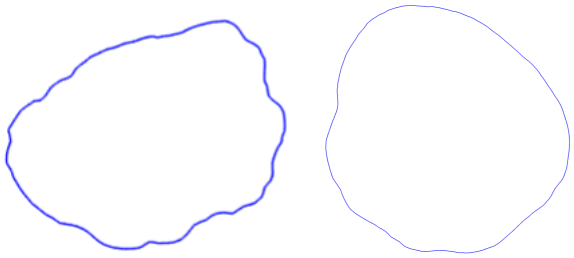
This example highlights the strengths of the nonisometric flow. The input contains three very large edges relative to the rest. The squared curvature flow, which is effectively isometric, is unable to completely smooth these out, resulting in an ovular minima. On the other hand the nonisometric wilmore flow is able to increase the

other edges relative to the large three, creating a very nice smooth approximation of the circle. Note the figures are not to scale.

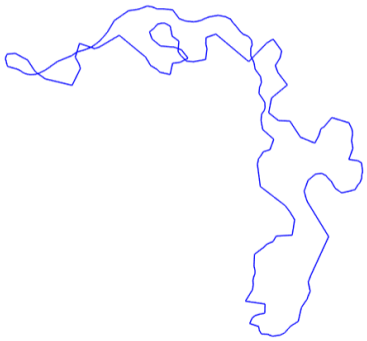
However as we've seen, nonisometric behavior begins to breakdown for larger inputs. Consider the following large input:



Respectively the nonisometric wilmore and squared curvatures result in:



Neither flow performs extremely well. But the wilmore flow looks noticeably worse, due to an inability to escape local minima. Furthermore the flow can be seen to start developing sharp cusps should we let the iterations continue. Contrast this to the squared curvatures flow which produced a smoother approximation in much less time. What is worse, if we make the step size for this input much larger we quickly get the following result:



which is highly degenerate and illustrates the importance of taking small enough step sizes with a small enough scaling factor.

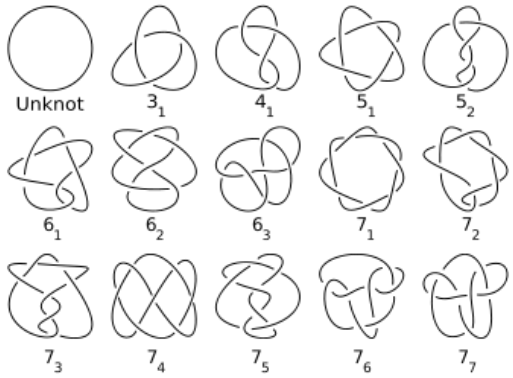
4 CONCLUSION

Overall it seems the nonisometric flows work as hoped on small inputs, but more work is needed for similar results on a larger scale. In particular the flows start to develop cusps which should not happen and severely impact aesthetic and stability/robustness. We see stability suffer with a need to decrease the maximal step size h and introduce a scaling parameter g. This in turn results in longer convergence times and undesirable aesthetic. The silver lining is that we seem to retain the edge smoothing property even on large inputs. It was interesting to see the squared curvatures flow exhibit behavior similar to the isometric flow despite being derived in a nonisometric setting.

5 FURTHER WORK

I did not expect to run into as much trouble as I did implementing the non isometric case. The allowance for variable edge lengths really seems to severely impact stability even on small scales. I aim to fix the issue with cusp development as soon as possible to eliminate sources of error and generalize to non-closed curves. Additionally I think it would pay to spend some time thinking about better energies to minimize which have nicer mathematical properties than those presented. For example, the squared curvatures energy is scale invariant, ie. a global scaling of the edge lengths does not change it. In general scale invariance is a desirable property since it leads to robustness over arbitrarily sized inputs, as we saw.

The next natural generalization after this is to explore nonisometric flows on space curves, which are embedded in three dimesnions instead of two. It seems possible that nonisometric flows in this case could be useful in computational knot theory, which seeks to understand how we can computationally determine whether a given space curve can be "continuously deformed" into a circle:



This semester I feel I did not take as much advantage of the members of the Geometry Collective Research Group as I should have. In the future I think it would be very good practice to reach out more for help with technical problems or confusion on theory points. I found myself spending a lot of time on non-essential points not especially important to the core of the project(time sinks). Overall I thoroughly enjoyed the research experience, despite this semester's many complications and am excited to continue my work in this direction.

REFERENCES

[1] Alexander I. Bobenko and Peter Schröder. 2005. Discrete Willmore flow. In Proceedings of the third Eurographics symposium on Geometry processing (SGP '05). Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, , Article 101

[2] Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust fairing via conformal curvature flow. ACM Trans. Graph. 32, 4, Article 61 (July 2013), 10 pages. DOI: https://doi.org/10.1145/2461912.2461986

[3] Keenan Crane. 15-458 Introduction to Discrete Differential Geometry. http://brickisland.net/cs177fa12/?p=320

[4] Keenan Crane. Lecture Notes for Introduction to Discrete Differential Geometry. http://www.cs.cmu.edu/ kmcrane/Projects/DDG/paper.pdf

[5] Or Hershkovitz. Topics in Differential Geometry. https://web.math.princeton.edu/ chaoli/notes/Hershkovits