Point Movement in a DSL for Higher Order FEM Visualization

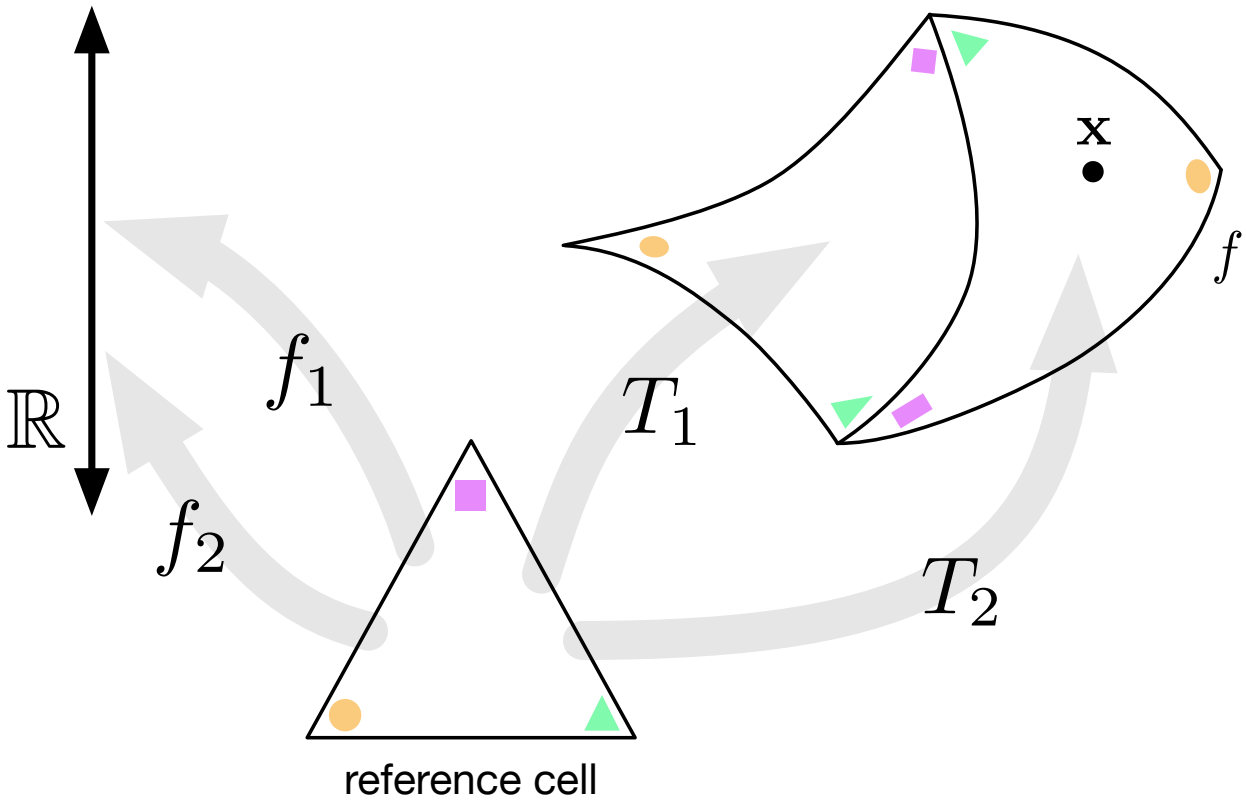
Teodoro Collin, Charisee Chiw, L. Ridgway Scott, John Reppy, Gordon Kindlmann

University of Chicago and Galois, Inc

November 5, 2019

Definition: Higher Order Finite Element Data

Type of data we seek to visualize:



$$f(\mathbf{x}) = f_2(T_2^{-1}\mathbf{x})$$
$$= (f_2 \circ T_2^{-1})\mathbf{x}$$

$$f = \sum_i f_i \circ T_i^{-1}$$

Key Cites

Previous higher order FEM visualization work:

G. Coppola, S.J. Sherwin, and J. Peiro. "Nonlinear Particle Tracking for High-Order Elements". In: *Journal of Computational Physics* 172.1 (Sept. 2001), pp. 356–386. ISSN: 0021-9991. DOI: 10.1006/jcph.2001.6829

Miriah Meyer et al. "Particle Systems for Efficient and Accurate High-Order Finite Element Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 13.5 (Sept. 2007), pp. 1015–1026

Spatial Coherence: Pseudo Code

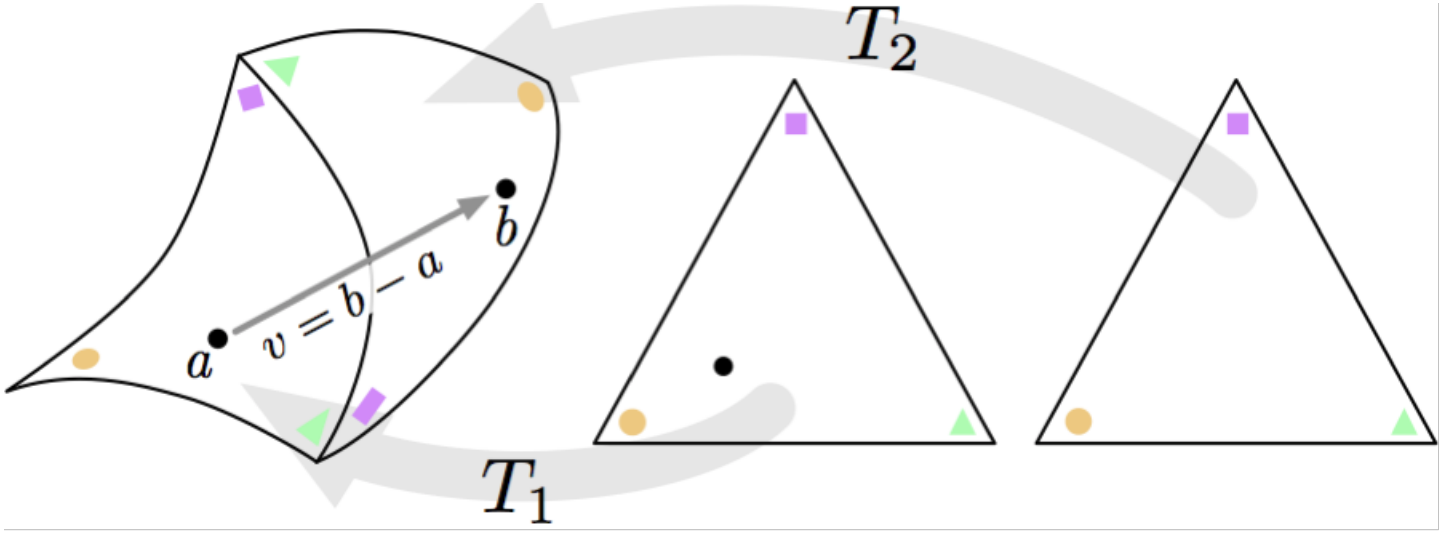
```
1    up = computeUpdate(startPos, f, ∇f, ...)
2    nextPos = startPos + up
3    ...
```

```
1    up1 = h * f(x)
2    up2 = h * f(startPos + 0.5 * up1)
3    nextPos = startPos + up2
4    ...
```

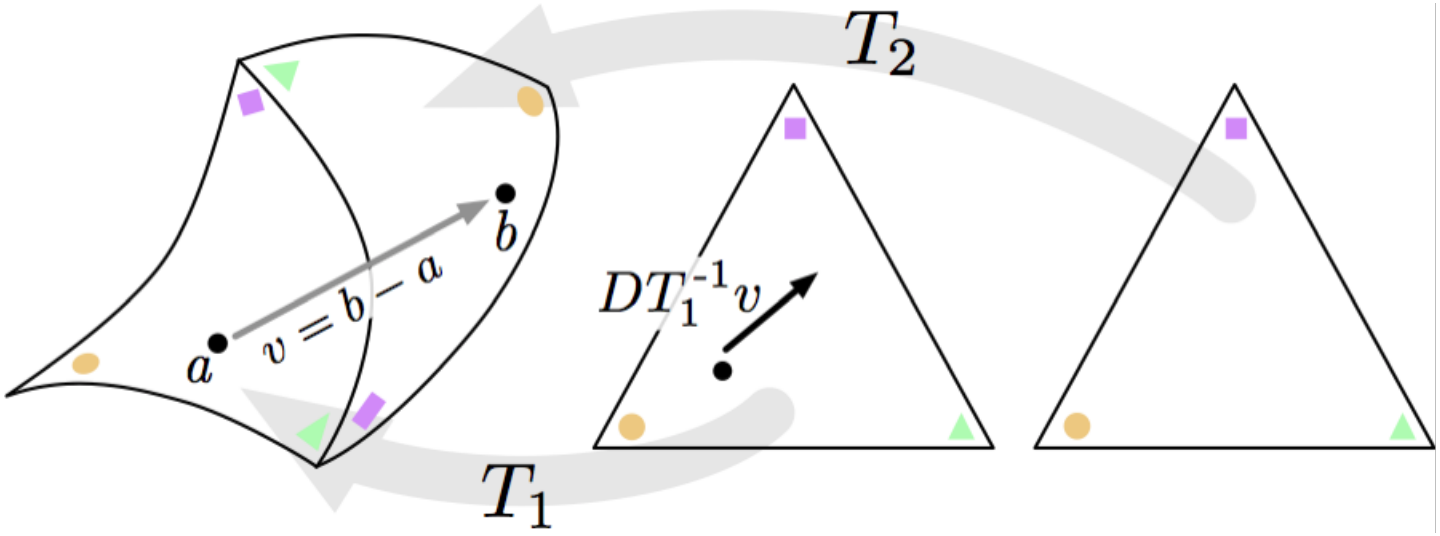
Add Approximation: Pseudo Code

```
1       up = computeUpdate(cell, refPos, f, ∇f, ...)
2       (refPos, cell) = approxUpdate(cell, refPos, up)
3       ...
```

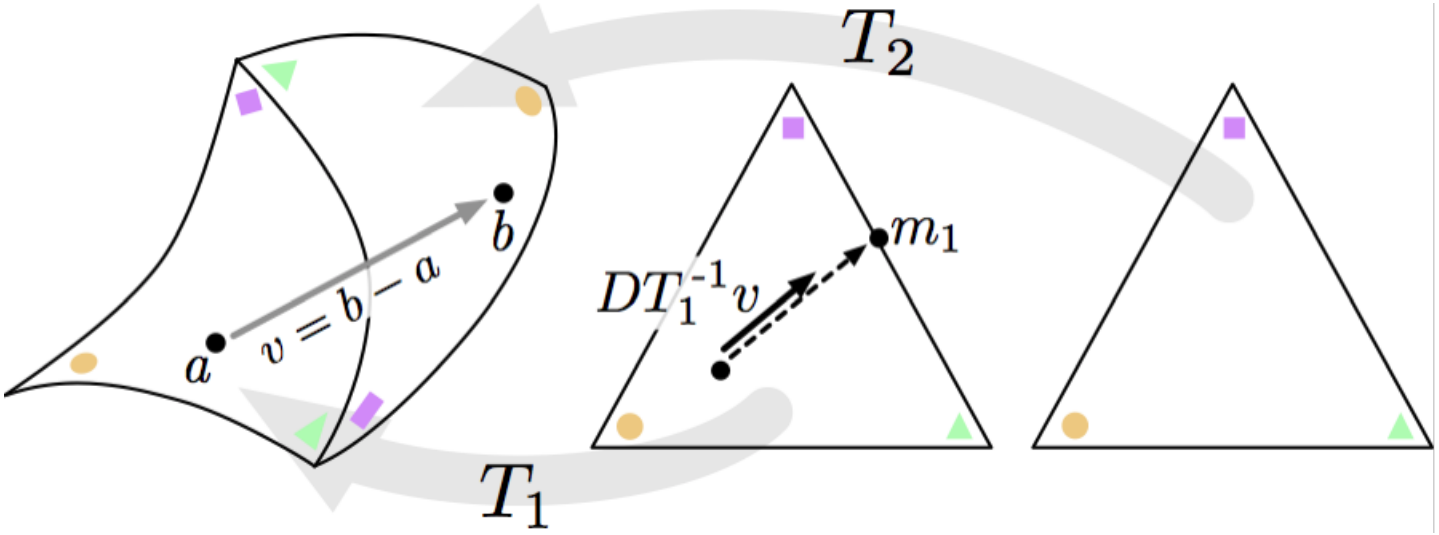
How to Approximate World Space Updates in Reference Space: Objective



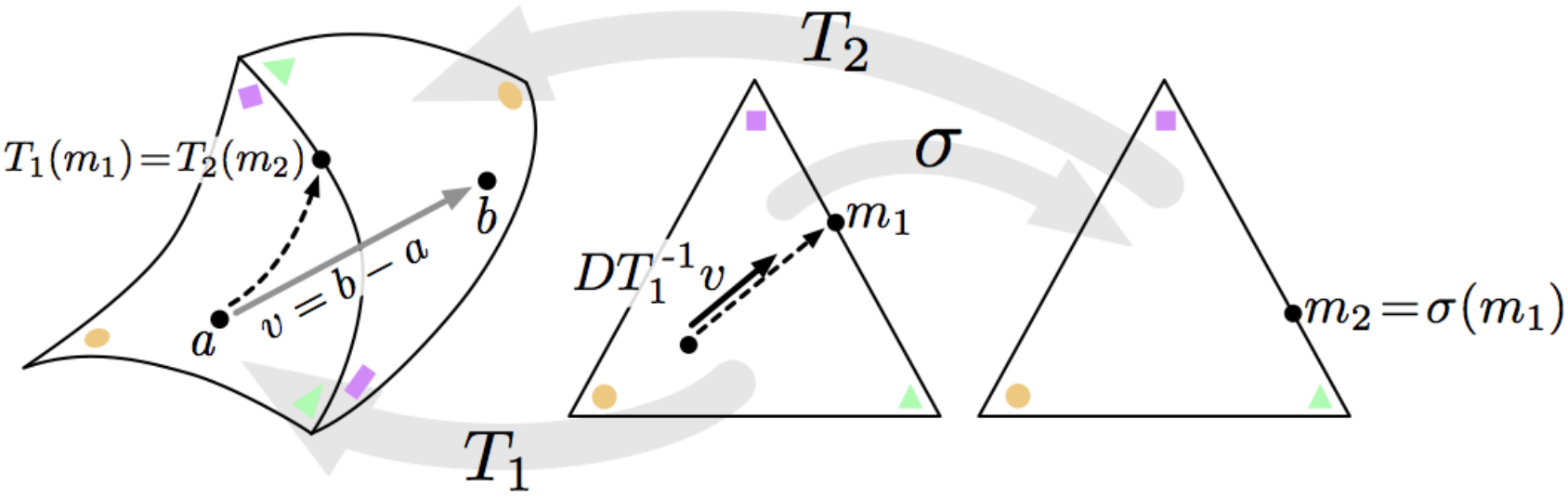
How to Approximate World Space Updates in Reference Space: Approximate Update



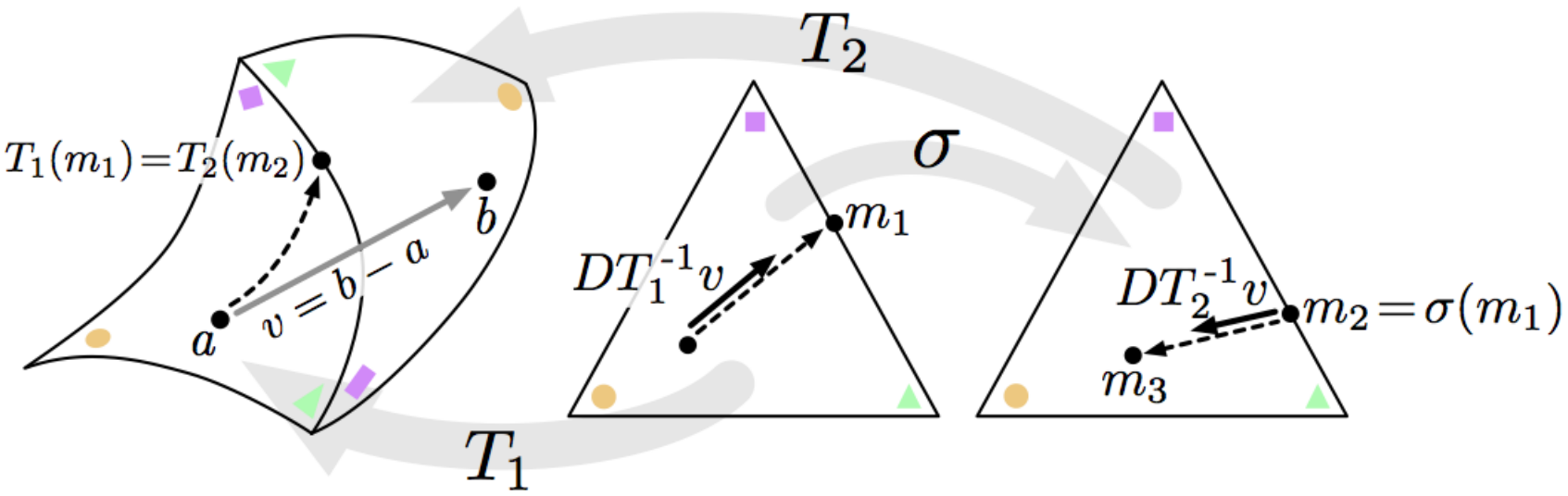
How to Approximate World Space Updates in Reference Space: Intersect Boundary



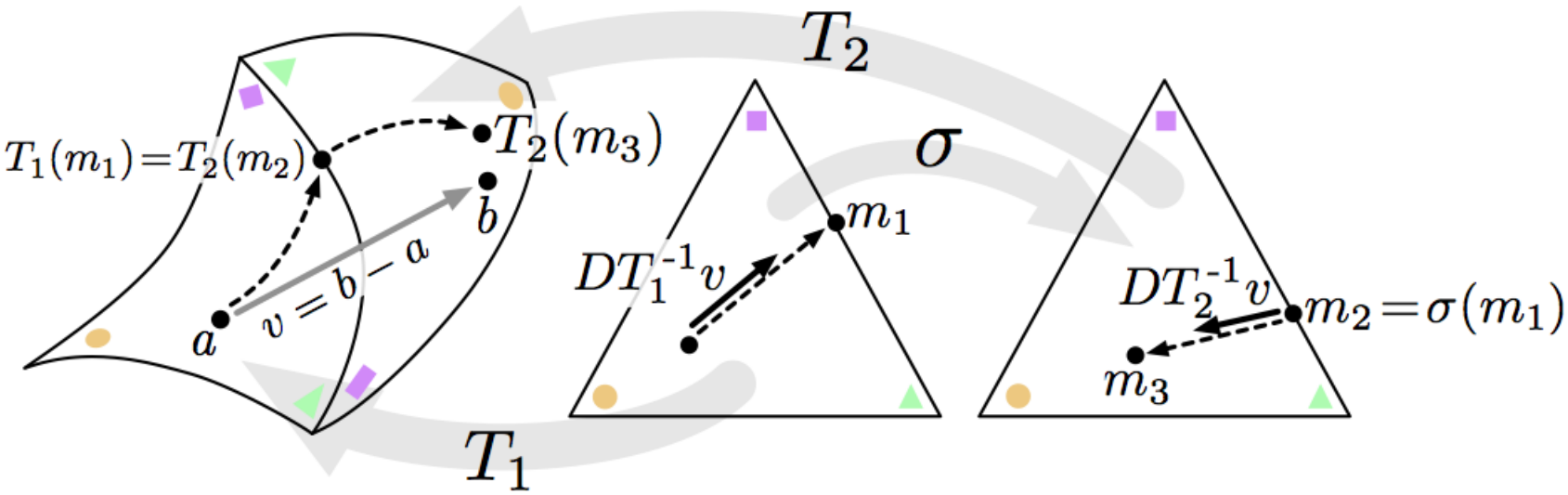
How to Approximate World Space Updates in Reference Space: Warp Reference Space



How to Approximate World Space Updates in Reference Space: Approximate Update



How to Approximate World Space Updates in Reference Space: Terminate



Observations

- Coppola et al and Meyer et al consider this idea in the context of the algorithms that they advance.
- However, the algorithm only approximates $p + v$ where p is a position and v is a vector; you could use it move a position within any algorithm.
- Moreover, when we write code, we should be able to just copy and paste this algorithm into traditional visualization algorithms to get a higher order FEM visualization algorithm.

Diderot: Support Idiomatic Expression of Scientific Visualization Algorithms

Diderot has roughly two elements:

1. A field language:

```
field#2(3)[3] F = bspln3⊛image("file.nrrd")
field#1(3)[] G = |∇det(∇⊗F)|^2
```

2. A bulk synchronous parallel system to organize computations.
3. See our previous papers for more information:
   Gordon Kindlmann et al. "Diderot: a Domain-Specific Language for Portable Parallel Scientific Visualization and Image Analysis". In: *TVCG* 22.1 (2016), pp. 867–876

Diderot + = FEM

1. We add types for FEM inputs:

```
type mesh mesh_t = file("type.json");
const int dim = mesh_t.dim;
type functionSpace{mesh_t}[] fns_t = ...;
type femFunction{fns_t} func_t = ...;
input mesh_t mesh;
```

2. We add ways to access FEM data (as fields):

```
field(3)[3] Ti = mesh.cells[i].transform
field(3)[3] invTi = mesh.cells[i].invTransform
field(3)[] oneish = det(Ti ∘ invTi)
```

3. We add a position and a position overload:

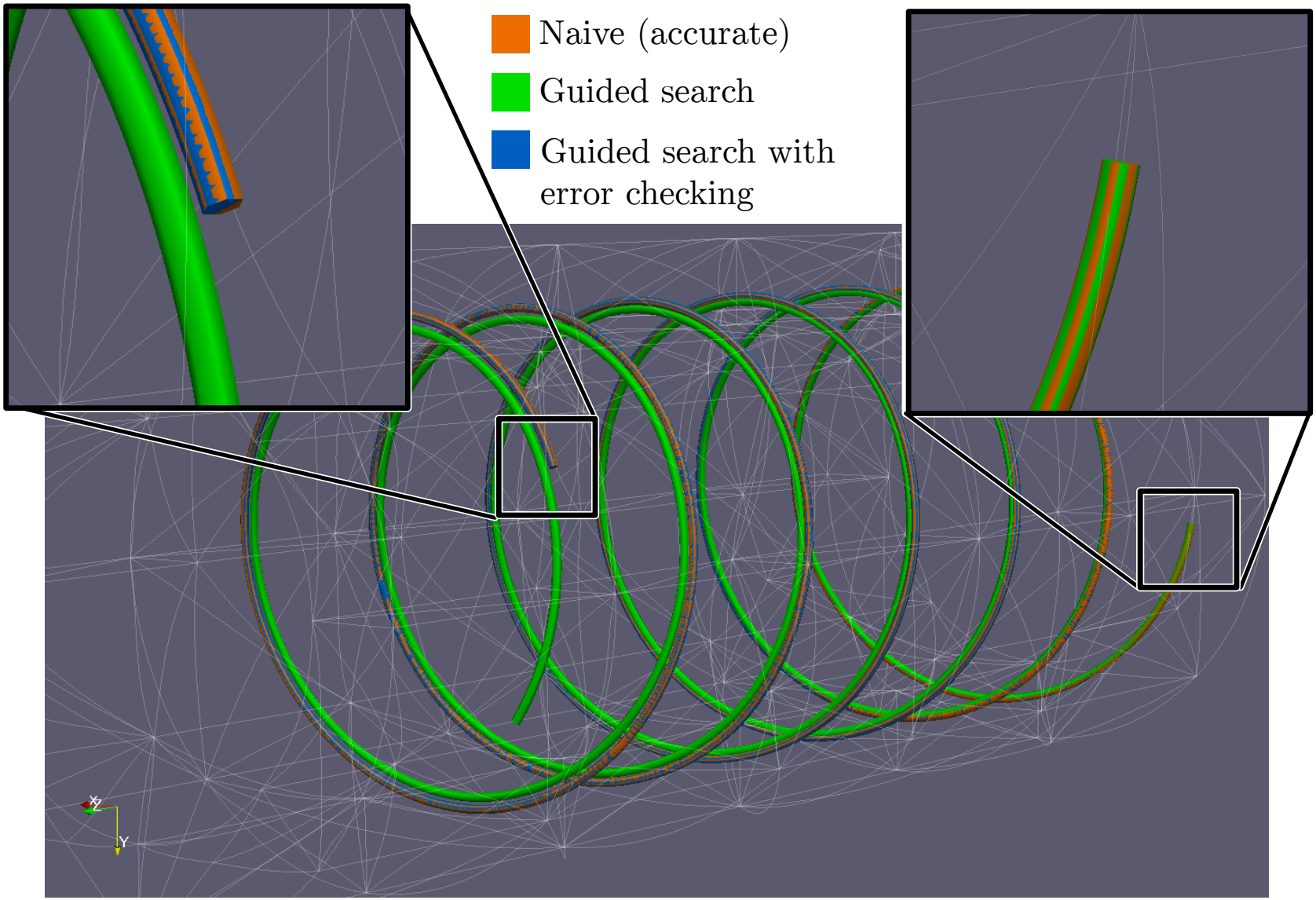
```
position{mesh_t} pos = mesh.findPos(worldVal);
position{mesh_t} pos' = mesh.cells[i].meshPos(refPos)
overload position{mesh_t} + (position{mesh_t} x,...)
{...}
```

Results: Lines of code

Program	Original Line Count	Lines Changed Or Added
Streamlines	∼30	∼15
Isosurface	∼80	∼ 30
Ridge surface	∼300	∼ 40

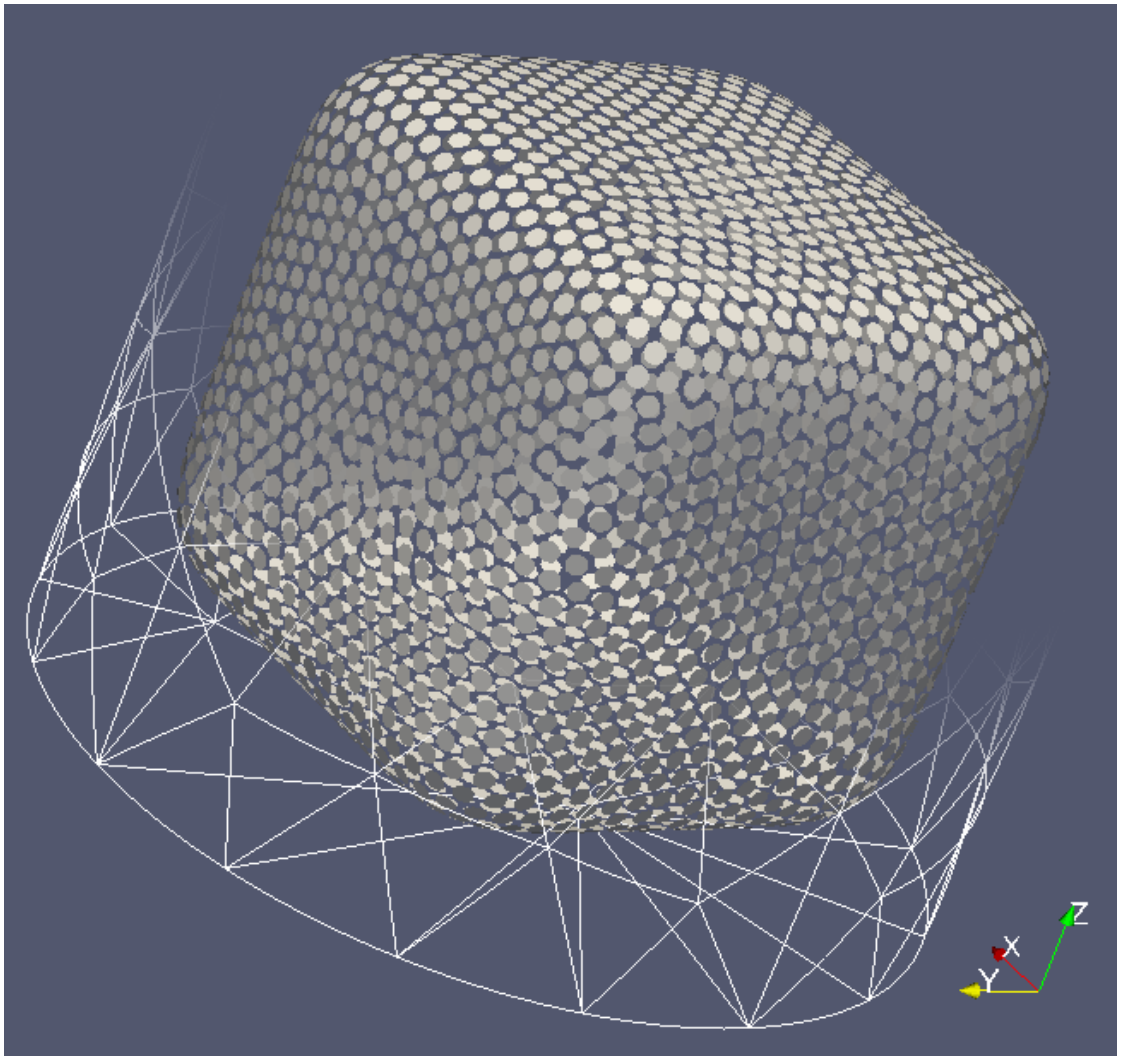
Line counts are not the best measure, but we feel that these are at worst over estimates of the mental effort required in conversions. In the supplemental materials, you can see a discussion of the changes for each program along with the code.

Results: Streamlines, Replication of Coppola et al.



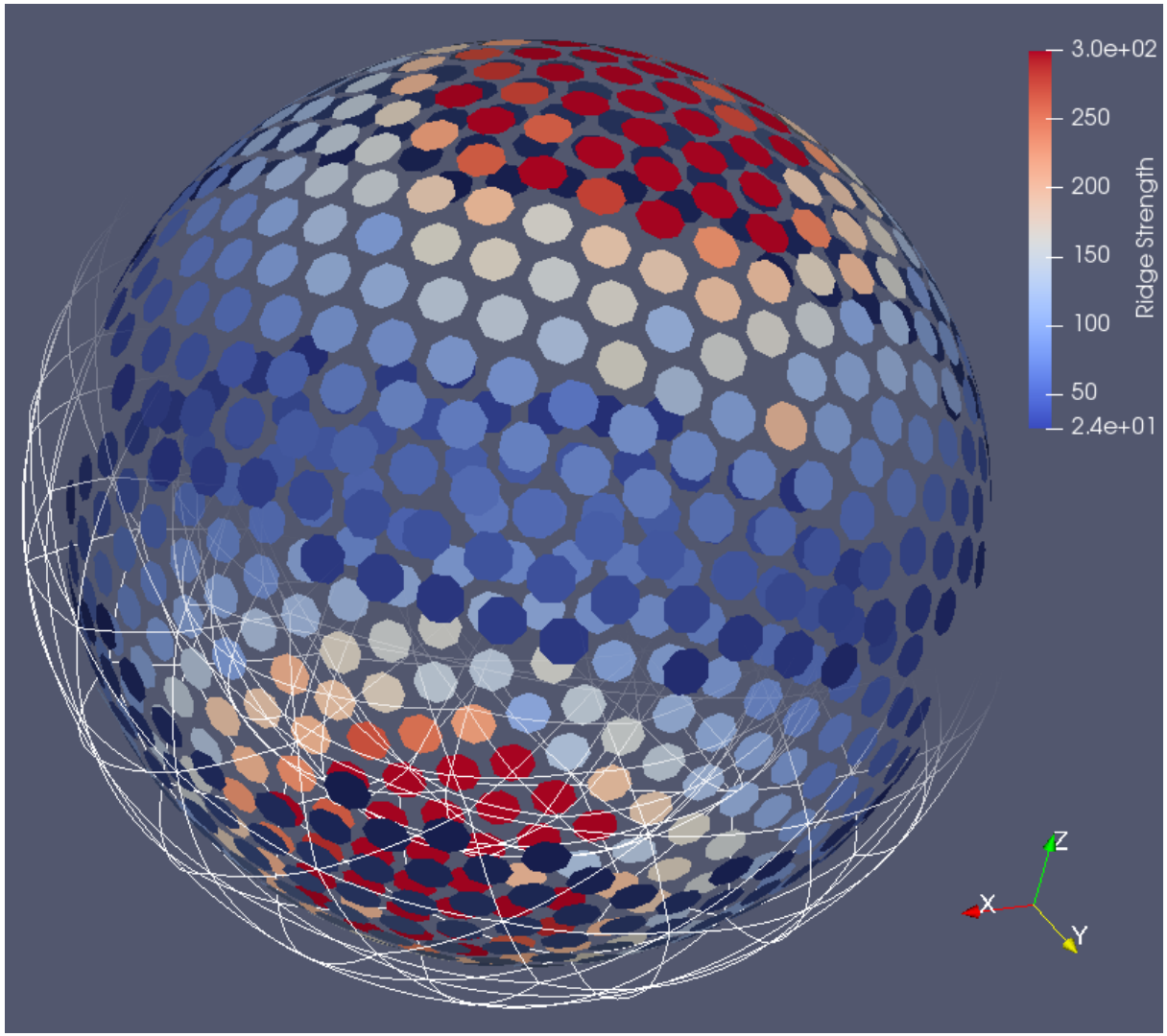
Results: Isosurfacing similar to Meyer et al.

$$f(x, y, z) = x^6 + y^6 + z^6 - 1$$



Results: Novel Ridge Surface

$$f(x, y, z) = z^2 \sin(x^2 + y^2 + z^2)$$



Future Work

- Experiment with other algorithms that use point movement.
- Try other approximations to the reference space update.
- Scale it up.
- Use Diderot to analyze other aspects of FEM Viz in a similar way: isolate algorithmic building blocks as language features and generalize the usage of these blocks.
- Reimplement the ideas of Diderot in Julia:
  - Julia's metaprogramming, optimization, and dispatch system allow us to emulate Diderot's components (possibly with comparable efficiency).
  - Julia's rich ecosystem is appealing.
  - We can study Diderot's components independently.
  - It might be easier for others to use or contribute.

Thanks And Questions

1. Thanks for your attention!
2. Thank you to the NSF: NSF grant CCF-1564298.
3. This code is unstable, but on github:
   `https://github.com/wraith1995/diderot/`
4. Contact: teocollin@uchicago.edu
5. The vanilla Diderot github is fairly stable:
   `https://github.com/Diderot-Language`
6. Questions?