Finding knotted and linked vorticity lines in 3D vector fields

Ryan G. Coleman

Tufts University

Co-Advisors: Bruce M. Boghosian and Diane L. Souvaine
Outline

• Problem Setup and Motivation
• Algorithms
• Results
• Conclusions and Future Work
• Acknowledgements
• Questions
A sample knot in a box
Sample with vectors
To create a vector

• Let influence of knot falls off like a Gaussian distribution as distance from each piece of the knot increases (graph from Weisstein's Mathworld)

• Let knot have an orientation.

• To find any single vector, integrate over the whole knot, summing contributions from each part based on the Gaussian.
  - Modification for periodic domain
Create a whole vector field

- Want to sample uniform positions over the entire box.
- Derive a vector with a tail at each position.
- Store in a matrix.
- Typical sizes $100^3$ to $1000^3$. 
Knot with derived vector field
Vector field with knot removed
Motivation

• Problem motivated by work in Fluid Dynamics which wants to explore connections between knots and fluids (Ricca and Berger 1996, Ricca et al. 1999, Sutcliffe and Winfree 2003) among others.
  
  – Particularly explore simulation of knots embedded in periodic box as described earlier

  – Assisted greatly by other researchers at Tufts, Bruce Boghosian and Lucas Finn. Used Lucas's code for embedded various knots and simulating them.

• Lack of algorithms for finding knots in vector fields
  
  – Want to find explicit ordered cycle of vector positions that represents or approximates the underlying knot.
Outline

- Problem Setup and Motivation
- Algorithms
- Results
- Conclusions and Future Work
- Acknowledgements
- Questions
Goal of first Algorithm

- Compute set of edge distances between close pairs of vectors.
- Orient the edges according to the vector direction.
- Want distances to be shorter if the vectors have high magnitude and are pointed toward each other, longer if the vectors have low magnitude or the vectors don't point in the same general direction.
Motivation for edge distances

- Want to create a distance measurement based on a vector to the 26 (in 3D) surrounding nearby positions

- Idea: Create a metric tensor from the vector
  - The metric tensor skews the unit ball used to measure distances
Properties of the metric tensors

- The limit as the magnitude of the vector goes to zero is a normal unit ball.
- The amount of skew used for each metric tensor as the magnitude increases is a parameter for the algorithm.
- Use $L_\infty$ instead of $L_2$ so that the following are equal:
Properties of the edges

- Orientation of the edges is defined by the halfplane with the vector as the normal

- The distance and direction of each edge between two positions is calculated by:
  - Averaging the distance according to each point
  - The direction must agree or the edge is thrown out
Edge Calculator Algorithm

• For each position in the domain
  – Calculate the metric tensor from the vector
  – Store the distance to each 26 neighboring positions

• For each pair of neighboring positions
  – Average the distance according to each
  – If their defined orientations agree, save the edge, otherwise throw it out
Edge Calculator Algorithm Complexity

• Let $n =$ total number of positions (like $100^3$)
• Takes $O(n)$ time, though the constant ($\sim 30$) is high, where $n$ is the number of positions.
• Creates a Directed Graph structure with:
  – Exactly $n$ nodes
  – $O(n)$ edges, actually $13n$ edges unless some are thrown out
  – Distances for each edge
Algorithm Results

- Shortest 20000 edges shown, as desired these are near the underlying knot
Goals of second algorithm

• Take input from first algorithm's Graph
• Output an explicit ordered cycle of positions that represent (hopefully) the underlying knot.
  – Optionally output more than one knot.
First idea and why it doesn't work

- **First algorithm:**
  - Start from shortest edge, make best local choice at each node until a knot is found.

- Agrees with Banks and Singer (1995) where they showed local choices don't work well in a completely different algorithm.
Motivation for real algorithm

- Using concept from alpha-shapes (Edelsbrunner and Mücke 1994), specifically a *filtration*

- Idea is to add simplices (in our case edges) to the structure according to size (in our case weight) and keep track of global properties through hopefully local updates.
  - In our case we will add edges until we get a complete directed cycle, or knot. This knot is the knot where the maximum edge weight is minimized.
Knot Finding Algorithm
Preliminaries

- Initialize a UnionFind data structure for each node in the input Graph
- Take the input Graph, add all the edges into a MinHeap.
- Initialize a new Graph with all the nodes and no edges (for Searching).
Knot Finding Algorithm

• Remove the shortest edge from the MinHeap
  – If Find(Source(edge)) != Find(Sink(edge))
    • Union(Source(edge), Sink(edge))
    • Add the edge to the Graph
  – Else
    • Start a Depth-First Search from Sink(edge) searching for Source(edge)
      – If it returns successfully, report the knot walked over
    • Add the edge to the Graph
Knot Finding Algorithm
Knot Finding Algorithm
Knot Finding Algorithm
Knot Finding Algorithm
Knot Finding Algorithm
Knot Finding Algorithm
Complexity Analysis

• Many O(n) steps
• O(n) runs that take O(log n) to fix the MinHeap, but O(n) to depth first search
• Total time O(n^2) and space O(n)
• In practice, finding the first knot takes less than 30 seconds of computer time (to run both algorithms)
Two modifications of the Algorithm

• Instead of stopping after first knot is found
  – Report K first knots found
  – Report K distinct knots found, where distinct means no node previously used in a knot is re-used.

• All three algorithms could prove useful in different cases
Outline

• Problem Setup and Motivation
• Algorithms
• Results
• Conclusions and Future Work
• Acknowledgements
• Questions
Two quick demonstrations

• Watching Knot Evolution
  – A (2,3) torus knot evolved with Lattice Boltzmann simulation, viscosity was set to 0.002
  – 5000 to 10000 iterations visualized

• Finding one topological change
  – A (2,3) torus knot evolved with Lattice Boltzmann simulation, viscosity was set to 0.01
  – 1400 to 1700 iterations visualized
Watching Knot Evolution
Finding one topological change
Finding one topological change
Outline

- Problem Setup and Motivation
- Algorithms
- Results
- Conclusions and Future Work
- Acknowledgements
- Questions
Conclusions and Future Work

• Algorithms shown to work for some test cases
  – Advantage over other work is that it returns an explicit ordered oriented knot.
  – Hopefully useful to researchers
  – Tighter integration with simulation code?

• Future Improvements
  – Change O(n) DFS to sublinear time step
    • Many options and literature reviewed without a good answer
Outline

• Problem Setup and Motivation
• Algorithms
• Results
• Conclusions and Future Work
• Acknowledgements
• Questions
Acknowledgements

• Committee members Bruce Boghosian, Diane Souvaine, Peter Love
• Lucas Finn
• Shawn Doughty – A&S Unix
• Andrew Fant – TCCS
Outline

- Problem Setup and Motivation
- Algorithms
- Results
- Conclusions and Future Work
- Acknowledgements
- Questions