Estimating Lower Bounds on the Length of Protein Polymer Chain Segments using Robot Motion Planning

Andrew McKnight, Jing He, Nikos Chrisochoides and Andrey Chernikov
Department of Computer Science
Old Dominion University
Norfolk, VA, USA
{*amcknigh, jhe, nikos, achernik*}@*cs.odu.edu*

***Abstract*—Finding the 3D structure a protein will assume given only its amino acid sequence remains largely an open question in bioinformatics. New developments have incorporated 3D images–"density maps"–of molecules from electron microscopy, but this presents its own problems. We seek to measure the lengths of segments of protein polymer chains associated with specific regions of the density map. We briefly discuss past efforts, and introduce an analog to robot motion planning as a novel approach. We will show this new approach's superiority through complexity analysis, and present some experimental results in the 2D case, leaving the 3D case to future work.**

***Keywords*-simplification; bounded; skeleton; shortest-path; graph**

I. Introduction

In the field of proteomics, an important question is how to efficiently determine the structure of a protein given its amino acid sequence: determining the phenotype of the protein from just the genotype, usually using machine learning methods. This approach has a vast solution space and exponential complexity. One promising advance in this area includes data from electron microscopy images of the protein molecules, the latest development being cryogenic electron microscopy (cryoEM). This approach can aid in discovering the structure and topology being sought and hopefully provide insight into the mechanisms that transform a given amino acid sequence into a functioning structure.

The microscopes produce three dimensional grayscale images whose voxels represent measured electron densities at the corresponding locations in space. These images are used in two ways to derive the protein's topology:

1) α-helix detection: Helixhunter [10] was previously used to determine the location of any α-helices, one of two secondary structure elements (SSEs) commonly found in proteins. More recently, we have developed and used our own gradient-based tool to detect the helices [19].
2) skeletonization: the Gorgon software package [16] produces a set of line segments and planes making up the basic "shape" described by the 3D density map in a process called morphological thinning [14,15]. Gorgon requires an input parameter; we have developed a tool that requires no input parameter, but that produces slightly more connected skeletal structures [18].

Kamal et al.'s topology ranking algorithm [3] attempts to find the correct pathway through the skeleton and α-helices that the amino acid sequence takes. They showed how to shrink the original solution space of $O(n!2^n)$ possible topologies, where n is the number of α-helices, to $O(n^2 2^n)$ using dynamic programming methods. The algorithm is able to narrow the possible topologies by comparing the distances between detected α-helices and the number of amino acids that are known to lie between them (it is generally accepted that the distance between two amino acids is between 3.5 and 3.8 Å).

Topology matching relies on knowing the distances between α-helix endpoints along the protein backbone, seen in Figure 1. We can use the skeleton to approximate the location of the polymer chain, and therefore measure its length, but it often contains many right angles, inflating the length estimates we derive from them. As a countermeasure, we employed the Douglas-Peucker line simplification algorithm [4]. While this is a widely accepted algorithm for these purposes, it requires an input threshold which affects its output, hence another degree of freedom in our own solution space. Other line simplification algorithms exist, claiming optimality under different metrics and independence from this input parameter [5,6,7,8].

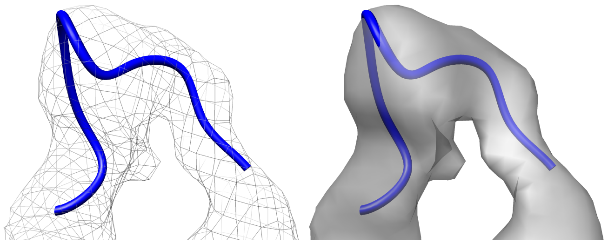


Figure 1. The tunnel-like partition of space induced from all matrix values over a threshold value in the 3D electron density map, and the associated segment of protein backbone whose length we want to know.

There are several limitations with our previous approach, both theoretical and practical. First, by constraining our pathfinding to the skeleton of an image, we dramatically reduce the size of our search space. A protein turn may twist around in the general areas of high electron density as encoded in the image–the skeleton represents one of many possible such paths. By creating a roadmap containing all locations within an iso-surface in the image, we are able to try many different paths. We also avoid the need for input parameters for skeletonization and curve simplification, by working directly with the density map values, in exchange for one parameter to construct the iso-surfaces.

II. Approach

Our approach is analogous to the robot motion planning problem [2]. This can be done in any number of dimensions–and the protein problem is in three dimensions–but for purposes of simplicity we will present our approach in only two. Our robot is a point with no size, and its work space and configuration space (which are identical for point robots) are the area inside the iso-surface defined in the image, as illustrated in Figure 1 in the 3D case. Such a tunnel can be obtained by removing all voxels from the 3D image under a certain threshold value.

The problem, as stated formally, is the following: given a simple polygon $\mathcal{P}$ subdividing the plane and a start point s and end point f, find the shortest path from s to f that lies completely inside the polygon. $\mathcal{P}$ and the robot's configuration space $\mathcal{C}$ are both embedded in $\mathbb{Z}^2$, hence our solution will also contain only points in $\mathbb{Z}^2$. In future work, we may choose to subdivide the intervals between points to obtain a finer discretization of the tunnel interior for more precise shortest path estimates. Figure 2 illustrates a simple case.

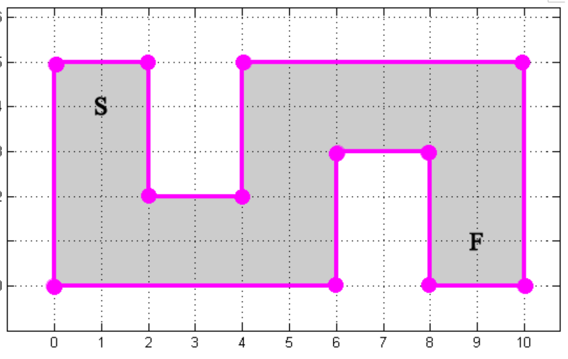


Figure 2. A 2D tunnel embedded in $\mathbb{Z}^2$ with its vertices and start and end configurations. The shaded region represents the configuration space we will work in.

Beginning with the grayscale density image $\mathcal{I}_G$, we generate a binary image $\mathcal{I}_B$ by setting all voxels with values lesser than some threshold $\mathcal{T}$ value to 0, and those voxels with values higher than the threshold to 1. The union of edges between pixels of opposite binary value forms an *iso-surface* polygon $\mathcal{P}$ in 2D, and the threshold value that produced $\mathcal{I}_B$ is called the *iso-surface threshold*. The pixels inside the iso-surface can be readily discerned from the binary image, by simply taking all entries with a value of 1.

Next, we construct our roadmap graph $\mathcal{G}$ by examining all pairs of interior pixels and testing whether a straight line segment between them intersects any edge in $\mathcal{P}$. If there is no intersection, then this is a possible path to take through $\mathcal{P}$, and two segments are added to $\mathcal{G}$ with the Euclidean distance between endpoints as the edge weight: one in each direction. When complete, $\mathcal{G}$ represents our entire search space for all possible locations within the iso-surface. Figures 3 and 4 summarize this process.

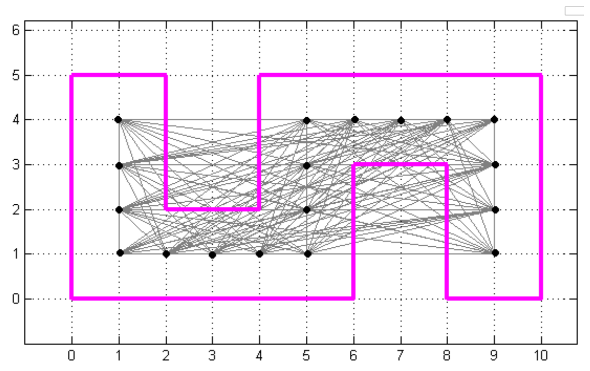


Figure 3. The initial tunnel voxel graph.

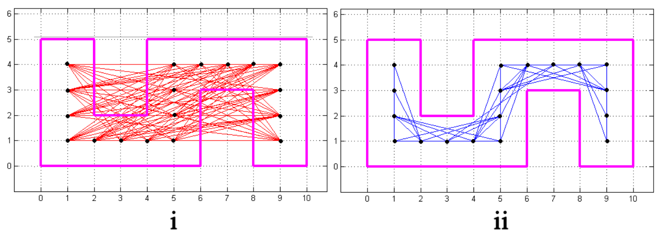


Figure 4. i) The removed edges that intersect the tunnel boundary. ii) The legal moves our robot can make to find the shortest path between s and f.

Using our search space, we would like to find the shortest possible path, representing the lower bound on the length of the protein's turn. Our roadmap is a positively weighted, directed graph, and so we use Dijsktra's algorithm to find all shortest paths from s, and reconstruct the path that ends at f. A simple case is shown in Figure 5. There are several situations where the choice of endpoints differ:

1) An endpoint lies inside $\mathcal{P}$. We simply use the endpoint.
2) An endpoint lies outside $\mathcal{P}$. We search through $\mathcal{P}$'s interior voxels for the one closest to the endpoint in terms of Euclidean distance.

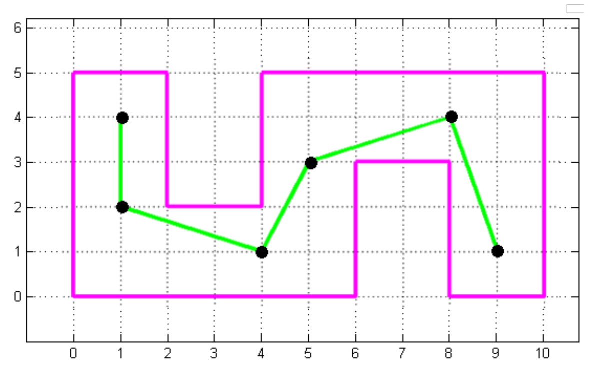
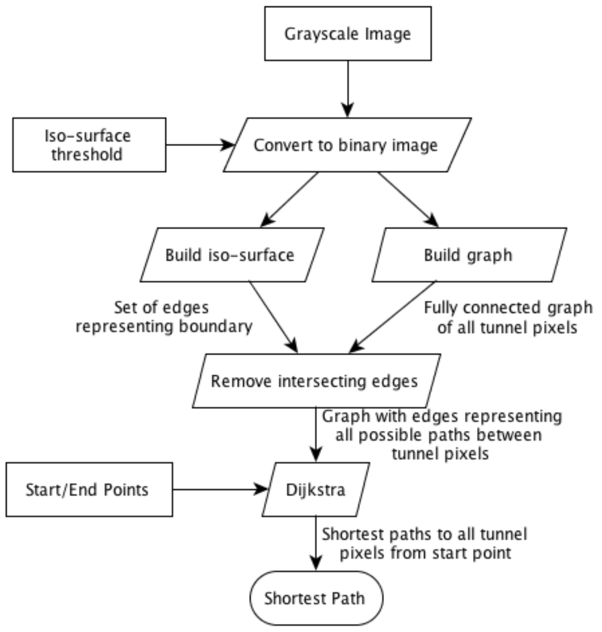


Figure 5. The shortest path between s and f that lies inside the tunnel.

The distances between vertices can be easily computed with the result and summed to obtain our lower bound estimation on the length of the protein polymer chain. Figure 6 summarizes the process:



Grayscale Image
Iso-surface threshold
Convert to binary image
Build iso-surface
Build graph
Set of edges representing boundary
Fully connected graph of all tunnel pixels
Remove intersecting edges
Graph with edges representing all possible paths between tunnel pixels
Start/End Points
Dijkstra
Shortest paths to all tunnel pixels from start point
Shortest Path

Figure 6. The complete algorithm.

III. Analysis

Our 2D images are square matrices of size $s \times s$, so there are exactly $n = s^2$ pixels. The binary images we produce are sparse binary matrices, with $\hat{n} = O(n)$ vertices, usually much less than n. Generating the $\mathcal{I}_B$ requires visiting each pixel once and so is $\Theta(n)$. Collecting $\mathcal{P}$'s interior pixel also takes linear time w.r.t. n. To collect $\mathcal{P}$'s edges from $\mathcal{I}_B$, each pixel is again visited once, and its four adjacent neighbors are inspected, again a linearly asymptotic operation with a constant number of operations at each pixel.

The most intensive operations are composing the roadmap by testing for intersections between graph edges and $\mathcal{P}$'s edges, and searching the roadmap for the optimal shortest path from s to f. By testing all possible pairs of interior pixels, we must make $O(n^2)$ comparisons. At each comparison, we are testing the intersection between the line segment l and any edge in $\mathcal{P}$, which can be checked in constant time. Dijkstra's algorithm is quadratic in the number of vertices in the graph, and so is also $O(n^2)$. Reconstructing the path afterwards is trivial in comparison.

The overall complexity of the algorithm is dominated by the $O(n^2)$ terms, so is quadratic w.r.t. the number of voxels in the cube. Figure 7 summarizes the complexities of the algorithm's components.

```
shortestPath($\mathcal{I}_G$, $s$, $f$)
  1) convert $\mathcal{I}_G$ to $\mathcal{I}_B$ $\mathcal{I}_B$        | $\Theta(n)$
  2) construct $\mathcal{P}$'s edges                  | $\Theta(n)$
  3) gather $\mathcal{G}$'s vertices                  | $\Theta(n)$
  4) construct $\mathcal{G}$                          | $O(n^2)$
  5) find intersections between $\mathcal{P}$ and $\mathcal{G}$ | $O(n^2)$
  6) Dijkstra's algorithm                             | $O(n^2)$
```

Figure 7. The algorithm with time complexities.

IV. Results

Testing was performed with a 1.7 GHz quad-core Intel i5 and 4 GB 1333 MHz DDR3 RAM, with cases ranging in size from $9 \leq a \leq 248$. A quadratic trend is evident in the observed test cases, as can be seen in Table 1 and Figure 8. Table 1 shows the amount of tunnel voxels per case, runtimes, and multiplication factors between subsequent cases' a and runtime values.

Table I
Runtimes with respect to a.

Case (i)	a	Runtime[1]	$\frac{a_i}{a_{i-1}}$	$\frac{\text{Runtime}_i}{\text{Runtime}_{i-1}}$
0	9	0.48	-	-
1	10	0.86	1.11	1.79
2	10	0.6	1.0	0.7
3	11	0.69	1.1	1.15
4	12	0.84	1.09	1.22
5	15	0.186	1.25	0.22
6	16	0.226	1.07	1.21
7	18	0.179	1.13	0.79
8	19	0.208	1.06	1.16
9	22	1.64875	1.16	7.93
10	25	1.64755	1.14	1.0
11	26	0.623	1.04	0.38
12	27	0.399	1.04	0.64
13	48	6.65008	1.78	16.67
14	94	24.6529	1.96	3.71
15	248	262.386	2.64	10.64

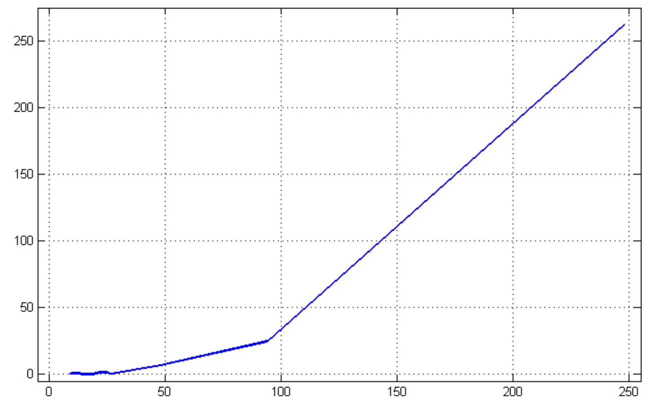


Figure 8. Runtimes with respect to a.

We implemented the algorithm in C++, utilizing the CGAL [15] library for applicable geometric computations and the Boost library [14] for graph operations. Some basic geometries were supplied as test cases, as well as some larger ones representative of the molecular structures that the algorithm is meant to evaluate, all of which can be seen in Figures 9-16. Cases 0-4 show the effect of eroding the corner of an l-shaped polygon; also tested were polygons with xy-, x-, y- and no monotonicity. Square-like and start-like polygons are both present. All the test cases are in two dimensions. However, our actual interest is in 3D molecules, and therefore 3D tunnels and corresponding paths. We are nearing completion of expanding our algorithm to 3D for work on actual density maps.

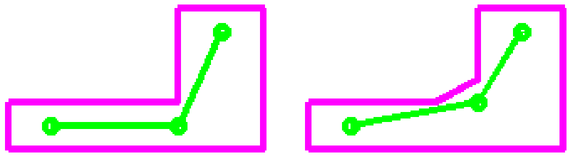


Figure 9. Left: case i = 0. Right: case i = 1.

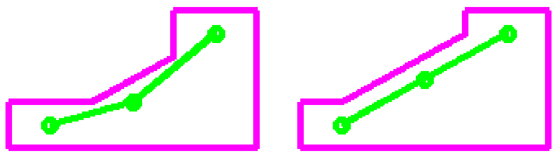


Figure 10. Left: case i = 2. Right: case i = 3.

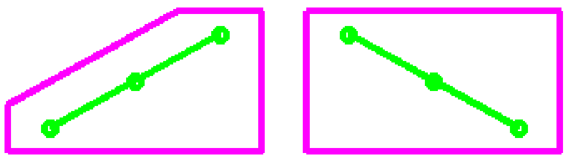


Figure 11. Left: case i = 4. Right: case i = 5.

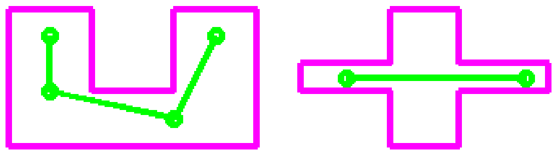


Figure 12. Left: case i = 6. Right: case i = 7.

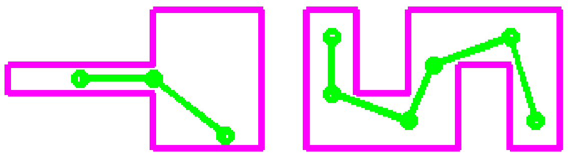


Figure 13. Left: case i = 8. Right: case i = 9.

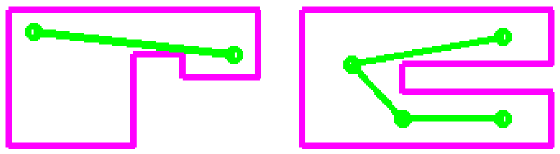


Figure 14. Left: case i = 10. Right: case i = 11.

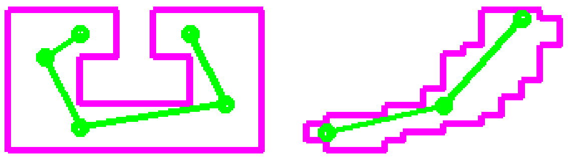


Figure 15. Left: case i = 12. Right: case i = 13.

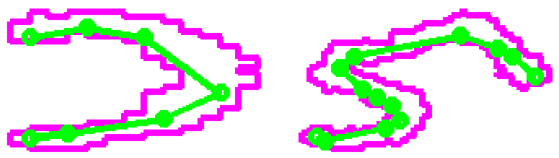


Figure 16. Left: case i = 14. Right: case i = 15.

We also included, as a real-world example, a two-dimensional slice from a density map. The map was generated using EMAN [20], by extracting the turn residues from a PDB atomic structure file and supplying them as input, to produce a synthetic density map representing only the turn. The endpoints of the path are the corresponding endpoints from the detected helix curves, as detected from a density map generated from the surrounding helix-turn-helix motif the turn in which the actual turn is located. We are currently using these synthetic maps to reduce the noise present, which is usually seen in real data, for initial development of this new technique. Figure 17 shows the results from this case.

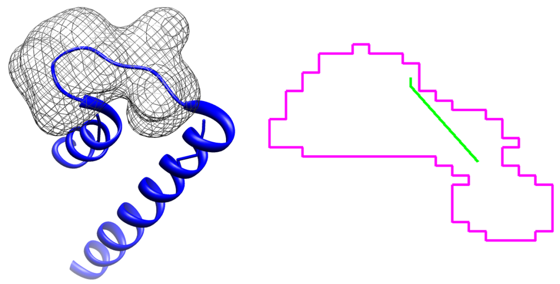


Figure 17. Left: a helix-turn-helix motif from PDB 1R1H in blue, with the density map generated using the turn residues as the grey mesh. Right: The iso-surface boundary for the median density value in magenta, with the shortest path between the two helix endpoints in green.

V. Conclusion

Robot-motion planning principles show promise in estimating the minimum length of a protein chain between two helices given only the 3D density map for the molecule. It replaces several steps: skeletonization, all-pairs shortest path computation and line simplification, thereby reducing the asymptotic complexity of the general problem's solution. It is readily translatable into the 3D case, which we have left for further study. Overall, it is a more elegant solution to the problem of estimating the lower bounds of polymer lengths from 3D images.

Acknowledgment

The authors would like to thank Dong Si, Lin Chen and Kamal al-Nasr of ODU for their work in related areas of the protein topology matching problem, and for the software they have provided. They would also like to acknowledge the authors of the CGAL and Boost libraries.

References

[1] Thomas H. Cormen et al, *Introduction to Algorithms*, 3rd ed. Cambridge, Massachusetts: The MIT Press, 2009.

[2] Mark de Berg et al, *Computational Geometry: Algorithms and Applications*, 3rd ed. Berlin: Springer-Verlag, 2008.

[3] Kamal al Nasr et al, *Ranking Valid Topologies of the Secondary Structure Elements Using a Constraint Graph*, Journal of Bioinformatics and Computational Biology 9(3), 2011, pp. 415-430.

[4] John Hershberher and Jack Snoeyink, *Speeding Up the Douglas-Peucker Line-Simplification Algorithm*, Proc. 5th Intl. Symp. on Spatial Data Handling, 1992, pp. 134-143.

[5] Pankaj K. Agarwal et al, *Near-Linear Time Approximation Algorithms for Curve Simplification*, Algorithmica 43, 2005, pp. 203-219.

[6] Prosenjit Bose et al, *Area-Preserving Approximations of Polygonal Paths*, Journal of Discrete Algorithms 4, 2006, pp. 554-566.

[7] Wang Xiao-li and Zhang De, *Selecting Optimal Threshold Value of Douglas-Peucker Algorithm Based on Curve Fit*, First Intl. Conf. on Networking and Dist. Computing, 2010.

[8] Veregin, Howard, *Line Simplification, Geometric Distortion, and Positional Error*, Cartographica 36(1), 1999, pp. 25-39.

[9] Bernard Chazelle et al, *Algorithms for Bichromatic Line-Segment Problems and Polyhedral Terrains*, Algorithmica 11, 1994, pp. 116-132.

[10] A. Dal Palu et al, *Identification of α-Helices from Low Resolution Density Maps*, Comp. Syst. Bioinformatics Conf., 2006, pp. 89-98.

[11] Khalid Saeed et al, *K3M: A Univeral Algorithm for Image Skeletonization and a Review of Thinning Techniques*, Int. J. Appl. Math. Comp. Sci. 20(2), 2010, pp. 317-335.

[12] Sasakthi S. Abeysinge et al, *Segmentation-free Skeletonization of Grayscale Volumes for Shape Understanding*, SMI 2008, pp. 63-71.

[13] Ju, Tao, Matthew L. Baker and Wah Chiu, *Computing a Family of Skeletons of Volumetric Models for Shape Description*, Computer Aided Design 39(5), 2007, pp. 352-360.

[14] Siek, Jeremy G., Lie-Quan Lee and Andrew Lumsdaine, *The Boost Graph Library: User Guide and Reference Manual*, Addison-Wesley Professional, 2001.

[15] NA. *CGAL: User and Reference Manual*, Release 4.1, October 2012. Retrieved from http://www.cgal.org/Manual/latest/doc_pdf/cgal_manual.pdf

[16] NA. *The Gorgon Project*, Retrieved from http://gorgon.wustl.edu/index.php

[17] Andrew McKnight et al, *CryoEM Skeleton Length Estimation using a Decimated Curve*, IEEE BIBM CSBW, 2012, pp. 109-113.

[18] Nasr KA, Chen L, Si D, Ranjan D, Zubair M, He J, *Building the Initial Chain of the Proteins through De Novo Modeling of the Cryo-Electron Microscopy Volume Data at the Medium Resolutions,* ACM Conference on Bioinformatics, Computational Biology and Biomedicine, Orlando, FL 2012.

[19] Si D, Ji S, Nasr K, He J, *A machine learning approach for the identifcation of protein secondary structure elements from electron cryo-microscopy density maps.*hskip 1em plus 0.5em minus 0.4emBiopolymers 2012, 97(9):698-708.

[20] Ludtke S, Baldwin P, Chiu W, *EMAN: semiautomated software for high-resolution single-particle reconstructions*, J Struct Biol 1999, 128:82-97.