Summer 2013

Protein Loop Length Estimation From Medium Resolution Cryoem Images

Andrew R. McKnight
Old Dominion University

PROTEIN LOOP LENGTH ESTIMATION FROM MEDIUM RESOLUTION CRYOEM IMAGES

by

A. R. McKnight
B.S. May 2012, Old Dominion University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
August 2013

Approved by:

_____
Nikos Chrisochoides (Director)

_____
Jing He (Member)

_____
Andrey Chernikov (Member)

ABSTRACT

PROTEIN LOOP LENGTH ESTIMATION FROM MEDIUM RESOLUTION CRYOEM IMAGES

A. R. McKnight
Old Dominion University, 2013
Director: Dr. Nikos Chrisochoides

In the post-genomic era, proteomics research presents a new frontier in life science. Proteins play roles in virtually every biological process, and understanding their atomic structures is the key to unraveling how they carry out their work. Compared to the over half million protein sequences in UniProt, only around 25,000 unique sequences have been atomically modeled and deposited to PDB (Protein Databank). Cryoelectron Microscopy (cryoEM) is an important biophysical technique that produces 3D subnanometer resolution images of molecules not amenable to past approaches like x-ray crystallography or nuclear magnetic resonance. De novo modeling is becoming a promising approach to derive the atomic structure of proteins from the cryoEM 3D images at "medium" resolutions–between 5 and 10 Å.

Distance measurement along 1D skeletons of 3D images is an important step in de novo modeling. Despite the need of such measurement, little has been investigated about its accuracy in searching for an effective method. We propose a method to refine the skeletal length via line simplification after selecting the appropriate segmentation from the density map using Hausdorff distances. Complementarily, we developed a motion planning approach to estimate the minimum length of a loop lying completely within a contour of the density map. To test the methods, loops between 1 and 10 residues in length were extracted from atomic structures in PDB and used to generate density maps at 8 Å resolution, along with experimentally derived density maps from EMDB (Electron Microscopy Databank).

ACKNOWLEDGMENTS

I would like to thank my advisors for their knowledge and guidance, both academic and worldly, the classmates who introduced me to the field of bioinformatics and later, those who helped me in my studies, my friends and family for always believing in me, and finally, my wife, who supported me throughout this endeavor with her inspiration, faith in my abilities, infinite patience, and most importantly, her unbounded love.

For Natalie Isabelle Smith and Tyler Martin Young

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Over the last ten years, cryoelectron microscopy (CryoEM) experiments have yielded increasing numbers of 3D electron density images of protein molecules. The Electron Microscopy Data Bank (EMDB) currently archives the 3D images, referred to as density maps, with a wide range of resolutions from 3 Å to over 80 Å [1]. When the density map is generated at high resolution (3-5 Å) such as in recent EMDB deposits 5499, 5520, 5495 and 5496 [2, 3], it is possible to derive the near atomic structure directly from the density map. However, when the density map is not in the high resolution range, it is still challenging to derive the structure of the imaged molecule [4, 5]. Fitting and comparative modeling approaches have been developed to utilize the existing atomic structures in the Protein Data Bank (PDB). These approaches apply when a component of the target density map has been resolved to near atomic resolution structure or when the target protein shares significant homology with existing atomic structures [6, 7, 8, 9].

Modeling protein molecules using de novo methods are advancing in their ability to derive the atomic structure from medium resolution (5-10 Å) density maps [10, 11, 12, 13]. Only the 3D image (top left in Figure 1) and amino acid sequence (top right) are used in de novo processes–a priori structural information from databases such as PDB is not included. Secondary structure elements (SSEs) such as α-helices (red sticks) and β-sheets are first identified using pattern recognition methods [14, 15, 16, 17, 18, 19]. Skeletonization methods detect the underlying 1D structure (green, left in Figure 1) of an iso-contour from the density map [13, 20]. Next, the amino acid sequence segments (red cylinders, right of Figure 1) of the SSEs can be predicted using existing tools [21, 22, 23, 24]. Using a dynamic programming algorithm on a weighted, directed graph of cells in a 2D matrix (whose rows and columns correspond to helices detected in the sequence and density map respectively), the correct SSE topology is derived and atomic modeling may commence [11, 25, 26, 27, 12, 16].

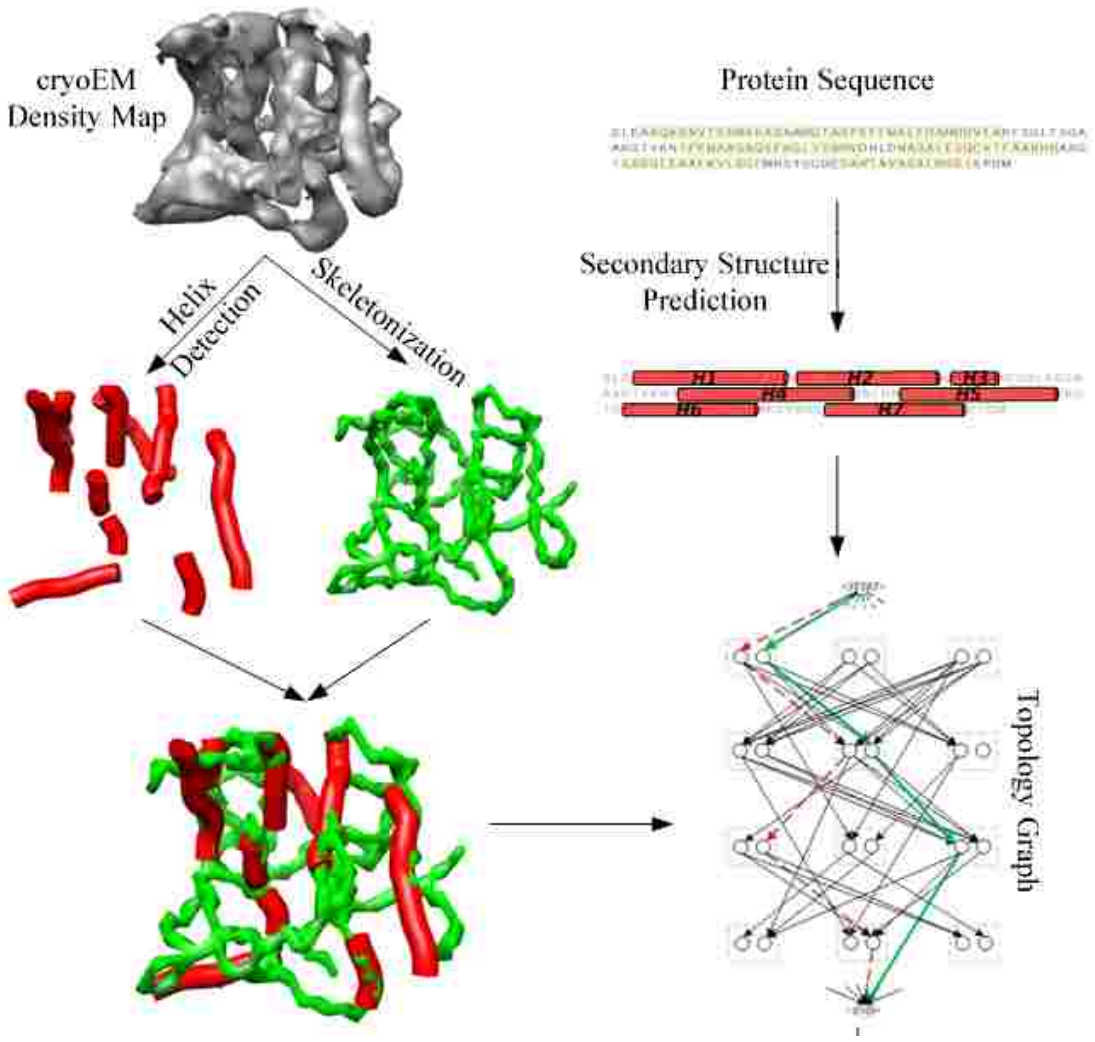

FIG. 1: The general steps in the *de novo* modeling process.

An amino acid sequence has a direction, starting with a nitrogen atom (N-terminal) and ending with a carbon atom (C-terminal). The SSE topology is the order in which this sequence traverses the protein's helices and sheets, including the direction of entry into and exit from each SSE. The native topology of a protein's SSEs is likely to produce the lowest energy state compared to incorrect topologies [28], hence determining the native state is a crucial step in de novo modeling. Previous work developed the aforementioned dynamic graph matching approach, which helps handle errors in density maps and preprocessing [11, 25, 26] .

The edges in the dynamic topology graph are weighted with some distance metric

between SSEs. Using the density map input, the Euclidean distance between SSE entry and exit points was first used [4], and more recently the length along the grayscale skeleton was measured [10, 26]. From the input amino acid sequence, the length of gaps between SSEs can be estimated assuming a 3.8 Å distance between residues on the backbone. The dynamic graph algorithm considers all possible combinations of gaps between SSEs in the 3D image and amino acid sequence and selects the best fit.

Despite the relative accuracy of skeletonization algorithms in finding a topological skeleton, overestimation may occur if length is measured directly along skeletons' piecewise linear curves, which contain many right angles. Error from the thinning process, helix detection and producing the 3D image itself may also contribute to errors in measurement. Following is a method to estimate the length of backbone segments between SSEs using CryoEM images for use in topology determination. By combining several graph-theoretic and computational geometric techniques, we refine the images' skeletons and obtain accurate length estimations. We tested the method using both simulated and experimentally derived density maps. The measured length appears to agree with the expected length when the SSEs are detected fairly accurately.

CHAPTER 2

ESTIMATION OF LOOP LENGTH VIA REFINED GRAYSCALE SKELETONS

2.1 ALGORITHM

The overall process to measure the loop length along the skeleton contains two tasks: preprocessing and length calculation (Figure 2). Preprocessing derives the skeleton and the endpoints of the two helices from the density map. Once such information is obtained, our algorithm uses graphs and computational geometric concepts to derive the simplified curve.

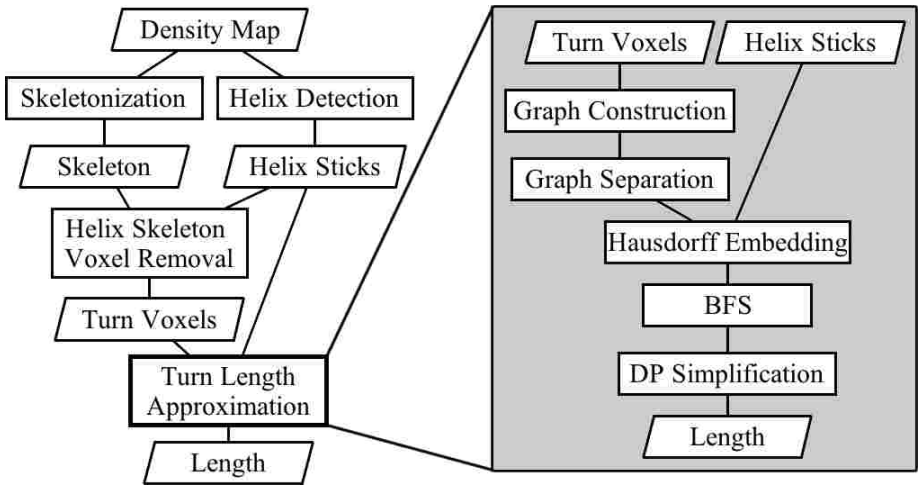

FIG. 2: Preprocessing and algorithm flow diagram for estimating loop length via grayscale skeletonization.

2.1.1 PREPROCESSING

We applied a skeletonization method that utilizes the local maximum points and clustering to derive the skeleton points from the density map. The density regions corresponding to helix-loop-helix (HLH) motifs for cases in in Table 2 were extracted from entire density images downloaded from EMDB. After helix detection using SSETracer, it is necessary to remove the skeleton voxels that belong to the helix region in order to obtain the skeleton belonging to the loop. We removed those skeleton voxels that are within 2.3 Å of the central axis of the helix. Note that an α-helix is between 2.3 and 2.5 Å in radius [14, 29]. After such processing, the skeleton voxels that presumably belong to the loop are segmented from the rest of the skeleton voxels and and ready for length calculation.

2.1.2 LOCAL CONNECTIVITY GRAPHS

A local connectivity graph (LCG) represents a cluster of skeleton voxels. While building a graph of all leftover skeleton voxels, we impose a constraint on the maximum allowable edge length l, possibly yielding multiple disconnected components. For our tests, we normalized the distances between the image's voxels to unity, and chose $l = 2$, producing individual connected subcomponents clustered into distant groups.

2.1.3 SELECTING CONNECTED COMPONENTS

Oftentimes, segmented or sparse density data yield multiple LCGs. Also, in general, it is not known which helix endpoints the loop actually lies between. We must then determine the best LCG for each possible pair of helix endpoints. For two helices, one with endpoints p and q and the other with r and s, there exists a set Z of four possible endpoint pairs: $Z := \{\{p, r\}, \{p, s\}, \{q, r\}, \{q, s\}\}$. For each endpoint pair $z \in Z$, let the directed Hausdorff distance to an LCG [30] be defined as

$$h(z, b) = \max_{z_i \in z} \min_{b_j \in b} d(z_i, b_j), \tag{1}$$

where z is the set of helix endpoints (comprised of voxels denoted z_i) and b is an LCG (comprised of voxels denoted b_j) from the set B of all LCGs; $d(z_i, b_j)$ is then the Euclidean distance between a helix endpoint voxel and LCG voxel. In the presence of multiple LCGs, we choose the best LCG $\hat{l}_z$ per endpoint pair $z \in Z$ by taking the

minimum directed Hausdorff distance over all LCGs:

$$\hat{l}_z = \min_{b \in B} h(z, b). \tag{2}$$

We can then use the voxels of $\hat{l}_z$ to build our model of the loop between the endpoints of z.

It should be noted here that the directed Hausdorff is not commutative–in general, $h(M, N) \neq h(N, M)$–and we always chose M as a set (pair) of helix endpoints, and N as an LCG (see Appendix B for more information). Figure 3 shows the configuration for case 30 (PDB 1O6L) from Table 1, where we want to find $\hat{l}_z$ among the set of LCGs $B := \{1, 2, 3, 4, 5, 6\}$ to search for the loop that *may* lie between the helix endpoint pair a. After finding $\hat{l}_z$ using equation (2), we do the same thing for each other helix endpoint pair. We try connecting the helix endpoints to their respective closest voxels in $\hat{l}_z$ w.r.t. Euclidean distance. If either of the new edges connecting p or r is longer than 5 Å, we discard the combination as an infeasible path.

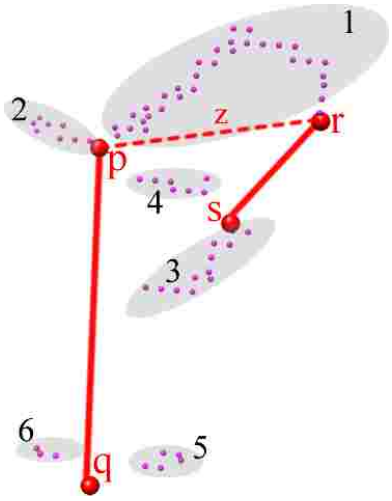

FIG. 3: Hausdorff distance comparison of the connected skeleton point groups. Two detected helices (solid red lines), with a pair z of helix endpoints (connected by the red dashed line) and several LCGs (gray ellipses) from PDB 1O6L. In this case, LCG *1* is closest to z in terms of directed Hausdorff distance.

2.1.4 PATHFINDING

After finding the best LCG for a given possible helix endpoint pair, the next step is constructing a path that traverses it in a way that will approximate the loop. We simply performed a breadth-first search starting from one of the helix endpoints we added, and reconstruct the path that ends at the other one in the graph [31], with a helix endpoint as the source. For a given HLH, we find four such paths, one for each possible helix endpoint pair.

2.1.5 PATH SIMPLIFICATION

Ideally, the distance between the endpoints of two helices should be measured along the skeleton connecting the two ends by using our initial path. If we simply add the length of its line segments, there is a danger of over estimation due to the potential zigzagging induced from drawing a path along the edges of the cubic lattice of the 3D image.

Douglas-Peucker line simplification is the systematic removal of points that lie beyond some distance ϵ from a line describing the general orientation of a piecewise linear curve (polyline) or one of its subsegments. Consider a two-dimensional example as in Figure 4. Part (i) shows an initial polyline $\overline{a...b}$. The algorithm is recursive, and takes as parameters the tolerance ϵ (ii) and a multi-point segment of a polyline. At each recursive iteration it finds an interior point of the current segment which is the most distant from the straight line connecting the end points of the segment, as in (ii) and (iii). If all of the current segment's vertices lie within the ϵ band, the segment is replace with a straight line segment containing only its endpoints. Otherwise, the segment is split at the most distant point and each subsegment is handled recursively.

In Figure 4 (iii), $\overline{ac}$ and $\overline{cb}$ are treated in different recursive calls; e is the farthest point from $\overline{cb}$, and no points lie outside the epsilon band for $\overline{ac}$. Overall, the initial polyline $\overline{a...b}$ is simplified into polyline $\overline{aceb}$, which approximates the length of the loop between helix endoints.

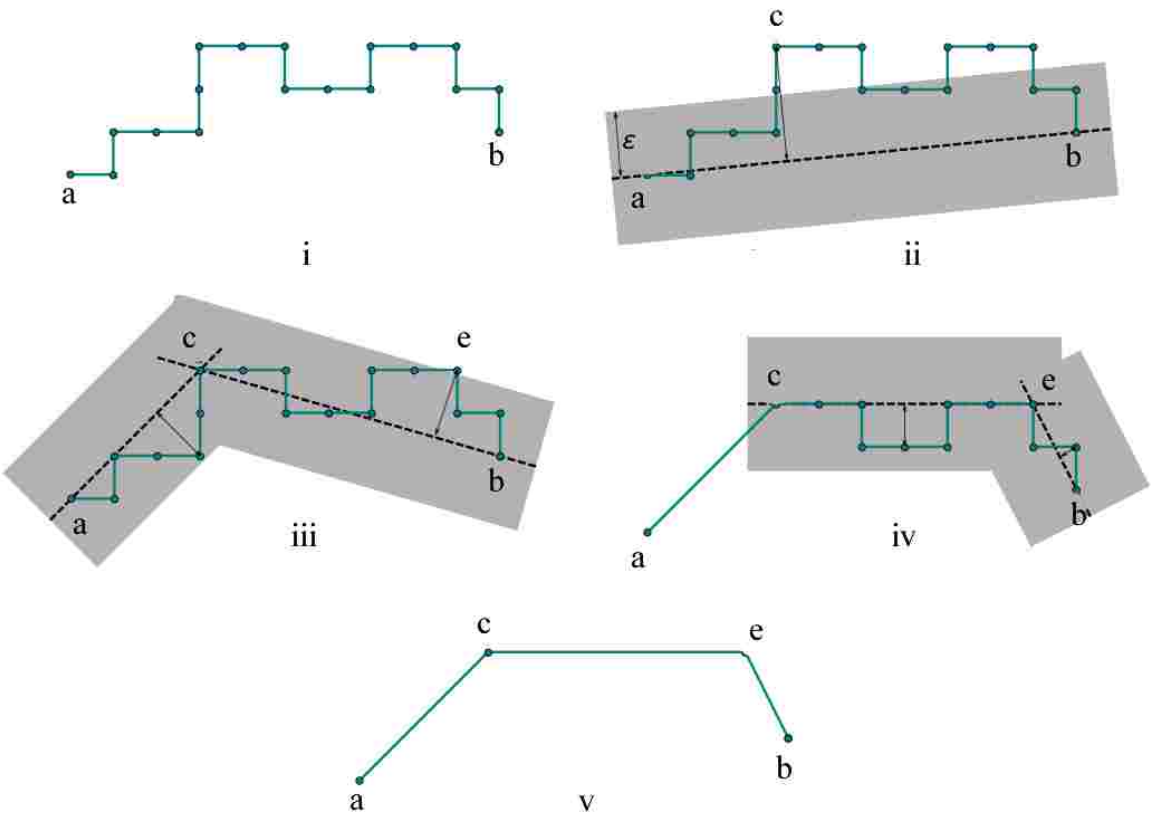

FIG. 4: Recursive iterations of the Douglas Peucker line simplification algorithm. Each gray region, as in (ii), illustrates the distance from the test line ($\overline{ab}$ in (ii)) defined by ϵ.

2.2 EXPERIMENTAL DESIGN

Two data sets were used in testing performance. The simulated data set consists of fifty randomly selected HLH motifs from atomic structures in PDB. The proteins extracted exhibit less than 10% sequence identity. Each extracted HLH of the protein structure was used to generate a 3D density map using EMAN's *pdb2mrc* function [32]. The density maps were simulated to 8 Å resolution.

The real data set consists of 18 cases whose density maps were downloaded from EMDB with resolution between 4.2 Å and 6.8 Å. Multiple HLH motifs were extracted from each of the EMDB entries: 5030 (6.4 Å), 1733 (6.8 Å), 5001 (4.2 Å), 1740 (6.8 Å) and 5168 (6.6 Å). Each of these density maps is aligned with their PDB structures at download.

The length of a loop was measured along the skeleton voxel points between (and

including) the end points of the two surrounding helices. The endpoint of a helix represents the end of the central axis of the helix [14, 15]. The helices were detected using SSETracer, a simplified version of SSELearner [19]. The skeleton was detected using a local maximum clustering method, more details of which are forthcoming in a separate paper. In order to test the accuracy of our algorithm, we visually inspected the detected helices and included only those cases in which the helices were roughly accurate. This was done to isolate the potential error in our loop length estimation from that of helix detection.

2.2.1 RESULTS

Measurement accuracy was evaluated using both the simulated set and real data from EMDB. Table 1 summarizes the results for the simulated data. The input to our method includes two pieces of information: the detected helix (red sticks in Figure 5 B and D) end points and the skeleton voxels (red dots). Each measured length along the skeleton was compared with the expected length of the loop. The expected length was calculated as 3.8 Å $\times$ $(n + 1)$, where n is the number of amino acids on the loop and 3.8 Å is the average distance between two amino acids. We add 1 to n because there is always one more interval between amino acids than there are amino acids in the loop. For example, a loop with one amino acid has a gap between each helix endpoint (two total), so with $n = 1$ we have 2 gaps and an expected length of 3.8 Å $\times$ 2 = 7.6 Å.

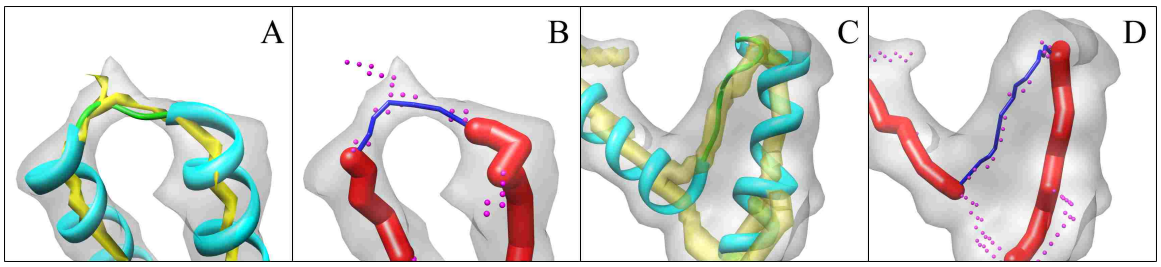

FIG. 5: Loop length estimation from a simplified curve. The density map (gray), detected helices (red sticks), and true structure (cyan) are shown for the HLH portion of the structure for 1DU0 (PDB) in (A, B) and 1MW8 in (C, D). The detected skeleton (yellow) is shown as a surface in (A) and (C) and as voxels (red dots) in (B) and (D), where the final simplified curve is shown in blue.

TABLE 1: Accuracy of loop length estimation via grayscale skeletonization in the simulated data set.

No	ID	AA	Expected	Measured	Diff	RelErr	DP ϵ
1	1ARO	1	7.6	7.4396	0.1604	2.1	1.00
2	1B0B	1	7.6	7.7384	0.1384	1.8	1.25
3	1BGP	1	7.6	7.6755	0.0755	1.0	1.30
4	1BQB	1	7.6	8.0995	0.4995	6.6	2.30
5	1GUX	1	7.6	7.8102	0.2102	2.8	6.00
6	1B43	2	11.4	11.4264	0.0264	0.2	0.45
7	1B89	2	11.4	11.8811	0.4811	4.2	2.55
8	1BD8	2	11.4	11.3578	0.0422	0.4	0.00
9	1BPY	2	11.4	11.4800	0.0800	0.7	2.25
10	1BR1	2	11.4	11.1461	0.2539	2.2	0.00
11	1FJL	3	15.2	15.4724	0.2724	1.8	1.35
12	1FK5	3	15.2	14.9523	0.2477	1.6	0.00
13	1FUR	3	15.2	15.2643	0.0643	0.4	6.00
14	1H0M	3	15.2	15.3601	0.1601	1.1	2.70
15	1DU0	3	15.2	14.9900	0.2100	1.4	0.60
16	1A87	4	19.0	18.8901	0.1099	0.6	0.95

TABLE 1 Continued

No	ID	AA	Expected	Measured	Diff	RelErr	DP ϵ
17	1AIH	4	19.0	19.2057	0.2057	1.1	6.00
18	1AJ8	4	19.0	4.1231	14.8769	78.3	0.00
19	1BMT	4	19.0	19.2313	0.2313	1.2	5.55
20	1BOU	4	19.0	18.9609	0.0391	0.2	0.70
21	1D8L	5	22.8	23.1403	0.3403	1.5	0.60
22	1DI1	5	22.8	22.9243	0.1243	0.5	4.25
23	1DLC	5	22.8	22.5618	0.2382	1.0	0.00
24	1DNP	5	22.8	23.1044	0.3044	1.3	1.70
25	1DP7	5	22.8	22.7786	0.0214	0.1	2.10
26	1CQX	6	26.6	26.2583	0.3417	1.3	0.00
27	1CSH	6	26.6	26.9157	0.3157	1.2	1.85
28	1HM6	6	26.6	7.1461	18.8539	26.3	0.00
29	1MW8	6	26.6	26.2419	0.3581	1.3	0.00
30	1O6L	6	26.6	26.6271	0.0271	0.1	6.00
31	1DJX	7	30.4	30.7842	0.3842	1.3	3.85
32	1E5Q	7	30.4	30.5342	0.1342	0.4	4.65
33	1FFV	7	30.4	30.0703	0.3297	1.1	2.50
34	1H99	7	30.4	30.1897	0.2103	0.7	0.00
35	1IRX	7	30.4	30.7213	0.3213	1.1	6.00
36	1O6L	8	34.2	34.6762	0.4762	1.4	6.00
37	1QVR	8	34.2	34.2838	0.0838	0.2	0.60
38	1S0V	8	34.2	34.2505	0.0505	0.1	0.95
39	1TAU	8	34.2	34.3267	0.1267	0.4	0.70
40	1U09	8	34.2	34.1468	0.0532	0.2	2.05
41	1D6M	9	38.0	38.1574	0.1574	0.4	1.00
42	1FUR	9	38.0	38.3249	0.3249	0.9	2.85
43	1H32	9	38.0	38.1491	0.1491	0.4	0.70
44	1QPC	9	38.0	37.9111	0.0889	0.2	0.00
45	1SU8	9	38.0	37.9337	0.0663	0.2	0.65
46	1QRT	10	41.8	41.7369	0.0631	0.2	0.75
47	1R1H	10	41.8	41.3131	0.4869	1.2	0.00

TABLE 1 Continued

No	ID	AA	Expected	Measured	Diff	RelErr	DP ϵ
48	1RJB	10	41.8	41.8528	0.0528	0.1	1.00
49	1XO0	10	41.8	41.8814	0.0814	0.2	1.05
50	2B63	10	41.8	41.4589	0.3411	0.8	4.60

The fifty tested cases were sorted by the length of the loop, ranging from 1 to 10 amino acids. Almost all the 50 test cases appear to have an error within 0.5 Å (column 6 of Table 1). As an example, the loop in 1DU0 (row 15 of Table 1) has three amino acids and the expected length of the loop is 15.2 Å. The measured length of the loop along the skeleton using our method is 14.99 Å. The relative error is 1.4% of the expected loop length. The simplified curve (blue in Figure 5 B and D) detected by the algorithm appears to be close to the skeleton points (red dots). Another example is from 1MW8 (Figure 5 C, D, row 29 of Table 1) with six amino acids on the loop. The error of the measurement is 0.358 Å in this case (column 6 of row 29, Table 1). Note that even when the skeleton points branch into multiple directions (Figure 5 D), the algorithm correctly measured the length between the two ending points of the helices by using Hausdorff measurements (see skeletonSegmentationSelection). In some cases, as in rows 18 and 28 in Table 1, the configuration of the detected helices and skeleton causes the greedy step in the Hausdorff computation to break down and we use the wrong skeleton segment to model the loop length.

TABLE 2: Accuracy of loop length estimation via grayscale skeletonization in the real data set.

No	ID	AA	Expected	Measured	Diff	RelErr	DP ϵ
1	5030	1	7.6	9.5128	1.9128	25.2	6.00
2	5138	1	7.6	8.2690	0.6690	8.8	6.00
3	5138	2	11.4	11.5490	0.1490	1.3	2.35
4	1733	3	15.2	14.3661	0.8339	5.5	4.05
5	1733	3	15.2	15.0790	0.1210	0.8	3.80
6	5001	3	15.2	11.1189	4.0811	26.8	0.00
7	5001	3	15.2	12.5132	2.6868	17.7	0.00
8	5001	3	15.2	15.6095	0.4095	2.7	2.35
9	5030	3	15.2	15.3747	0.1747	1.1	6.00
10	5030	3	15.2	14.6116	0.5884	3.9	1.75
11	5030	3	15.2	15.1321	0.0679	0.4	3.50
12	5138	3	15.2	14.2916	0.9084	6.0	5.30
13	1733	4	19.0	18.2477	0.7523	4.0	0.00
14	5001	4	19.0	19.1872	0.1872	1.0	6.00
15	5168	4	19.0	21.8790	2.8790	15.2	6.00
16	1740	5	22.8	26.4127	3.6127	15.8	6.00
17	1740	6	26.6	29.3993	2.7993	10.5	6.00
18	5168	6	26.6	22.4231	4.1769	15.7	0.00

The test using the experimentally derived density data involves 18 HLH motifs from density maps with 4-7 Å resolution from EMDB. 12 of the 18 cases have measured error within 2 Å, and 6 have error between 2 Å and 5 Å. The real density maps from the experiments are often more challenging with missing density and additional densities that do not align with the true structure. The helices and skeletons detected from the real maps are therefore often less accurate than those from the simulated density maps. Figure 6 shows an example of experimentally derived data in EMDB 5168 (row 15 in Table 2). Its relative error is 15.2%, higher than a comparable case with a synthetic density map used instead. In general, we saw an increase in error using the real density images, due to greater errors in helix detection and skeletonization induced by the noise present.

The algorithm uses a simplification parameter ϵ that is user defined. ϵ is the width

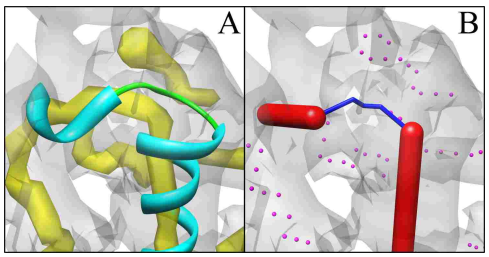

FIG. 6: Detected simplified curve for a loop in an experimentally derived CryoEM image (EMDB 5168). The color scheme is the same as that in Figure 5.

of the vertex removal band (refer to §2.1.5 more details). In general, the smaller the ϵ value, the less change in the simplified curve compared to the initial path. In some cases $\epsilon = 0$ and no simplification was needed; in other cases a much larger value was needed. In order to see the degree of simplification that produced the most accurate results, we sampled ϵ's range inside the interval [0,6] in increments of 0.05. The measured lengths w.r.t. ϵ values appear to form a step function, and the value closest to the expected value (Figure 7 left) was marked. As seen from this case, the measured length reduces as ϵ increases stepwise, and since the expected length falls between two steps, the lesser of the two distances a and b is chosen for the best estimate.

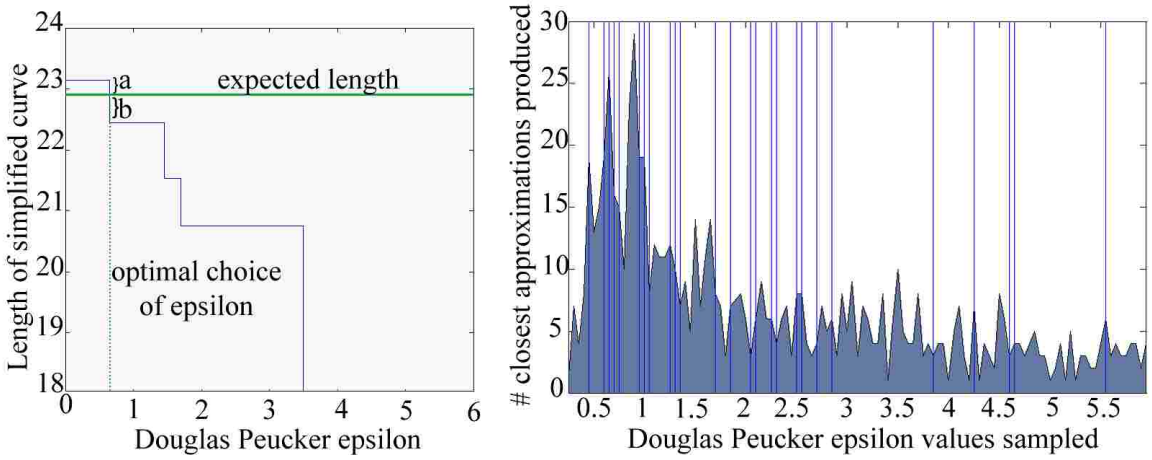

FIG. 7: The Douglas-Peucker ϵ step function. (Left) The ϵ step function for case 21 in Table 1 (PDB 1D8L), with the value of ϵ used for the best estimate. (Right) Distribution of the best ϵ in the simulated data set of 800 loops. The vertical lines show the values that are listed in Table 1.

Figure 7 (right) shows the distribution of the values of ϵ for about 800 simulated cases that had less than 0.5 Å difference. The function bounding the shaded region represents the amount of cases using a particular value of ϵ to obtain an estimate, and the vertical lines represent values of ϵ for cases in Table 1. It appears that most of the ϵ values are between 0.0 and 1.5 to minimize the error in the measurement (Figure 7, right). However, we observed larger ϵ values for the experimentally derived data than for the simulated density maps. This difference is likely associated with the quality of the skeleton and helix detection. For the simulated cases, ϵ between 0.0 and 1.5 is more likely to produce a good estimate given good preprocessing of the density maps, as observed from the skew of the function and vertical lines in Figure 7 (right).

2.2.2 CONCLUSIONS

We have developed a new approach to estimate loop length along the skeleton from a CryoEM density map. Our tests, using both simulated and experimentally derived images at medium resolution, show that it is possible for our proposed method to estimate fairly accurately the loop length along the skeleton if the SSEs and skeleton are detected fairly accurately.

CHAPTER 3

ESTIMATION OF LOOP LENGTH BOUNDS VIA ISO-CONTOUR ROADMAPS

There are several limitations with our previous approach, both theoretical and practical. First, by constraining our pathfinding to the skeleton of an density map, we dramatically reduce the size of our search space. A protein loop 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 motion planning roadmap containing all locations within an iso-contour of a density map, we are able to try many different paths. We also avoid the need for input parameters to control skeletonization and curve simplification, working directly with the density map values in exchange for a single parameter to construct the iso-contours.

3.1 ALGORITHM

A grayscale image $\mathcal{I}_G$ is provided as input, and an iso-contour $\mathcal{C}_\tau$ is extracted using a particular threshold τ. The other input, a pair of detected helices $\mathcal{H} := \{h_1, h_2\}$ in the form of piecewise linear curves, provides endpoints for the path to draw through $\mathcal{C}_\tau$. The Hausdorff distance is again used to resolve segmentation in $\mathcal{C}_\tau$ and choose the correct endpoints from h_1 and h_2. A fully-connected graph $\mathcal{G}$ of the voxels interior to the contour is constructed, and edges in this graph that intersect $\mathcal{C}_\tau$ are removed to derive a new graph $\hat{\mathcal{G}}$ representing all possible positions for a path within $\mathcal{C}_\tau$. The process is summarized in Figure 8.

3.1.1 EXTRACTING ISO-CONTOURS FROM GRAYSCALE IMAGES

With a given grayscale image $\mathcal{I}_G$ (Figure 9 A shows a 2D example) and iso-contour threshold τ, a binary image $\mathcal{I}_{B\tau}$ is created by visiting each voxel $\mathcal{I}_{G(x,y,z)}$ in $\mathcal{I}_G$ and comparing its value $v(\mathcal{I}_{G(x,y,z)})$ to τ to set the value of the corresponding voxel in $\mathcal{I}_{B\tau}$ to either 0 or 1, summarized in Equation 3. Figure 9 B shows the binary image

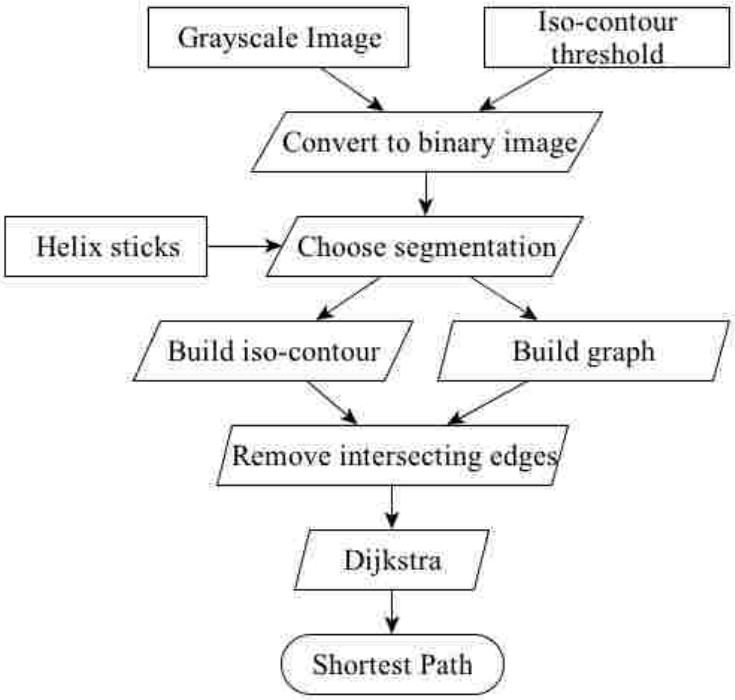

FIG. 8: The overall process of iso-contour extraction, roadmap construction and pathfinding.

generated for $\tau = 30$. Once $\mathcal{I}_{B\tau}$ is generated, the iso-contour polyhedron (polygon in 2D) is found by inserting simplices (lines in 2D, triangles in 3D) between each pair of adjacent voxels in $\mathcal{I}_{B\tau}$ with opposite values. Figure 9 C shows the resulting contour $\mathcal{C}_\tau$ extracted from $\mathcal{I}_G$ with $\tau = 30$.

$$v(\mathcal{I}_{B(x,y,z)}, \tau) = \begin{cases} 1 & : v(\mathcal{I}_{G(x,y,z)}) \geq \tau \\ 0 & : v(\mathcal{I}_{G(x,y,z)}) < \tau \end{cases} \tag{3}$$

The surface where a pair of adjacent voxels touch is a square, but we want to define $\mathcal{C}_\tau$ with a set of triangles to satisfy the requirement of CGAL's intersection routine. We simply subdivide each such square into two triangles as in Figure 10 A and B. The union of all such triangles roughly approximates the partition of the space in $\mathcal{I}_G$ with values $< \tau$; in future work a marching cubes algorithm would be better suited to extract $\mathcal{C}_\tau$. Figure 11 shows nine contours from a synthetic density map generated from a loop portion of the protein structure 1FUR in PDB, with the percentage of voxels discarded during conversion to binary images (refer to §3.2).

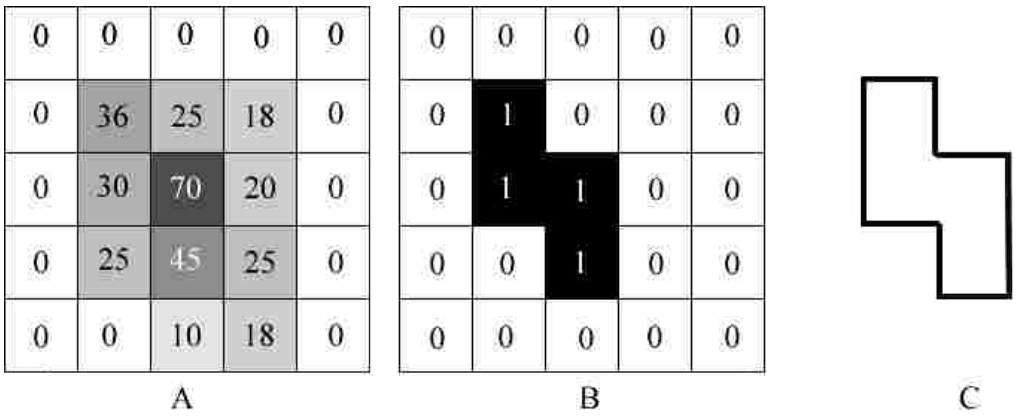

FIG. 9: A grayscale image with assigned values (A); the binary image extracted using $\tau = 30$ (B); the edges that comprise the iso-contour, surrounding the binary voxels with 1 values (C).

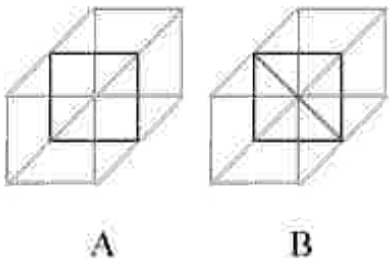

FIG. 10: Splitting each intersection square (dark square in A) into triangles (B) to define the iso-contour surface.

3.1.2 SEGMENTATION RESOLUTION

As seen in Figure 11, the binary image can become segmented at high threshold values, producing multiple disconnected components. To decide which component most likely represents the turn between a pair of helices, the directed Hausdorff distance is used as in §2.1.3, using helix endpoints and $\mathcal{C}_\tau$'s interior voxels. Again, we have two pairs of helices for a total of four possible pairs of path endpoints, and we must simultaneously choose the correct segmented image component along with the correct helix endpoint pair. The selected segment of the contour is henceforth referred to as $\mathcal{C}_\tau$ for simplicity.

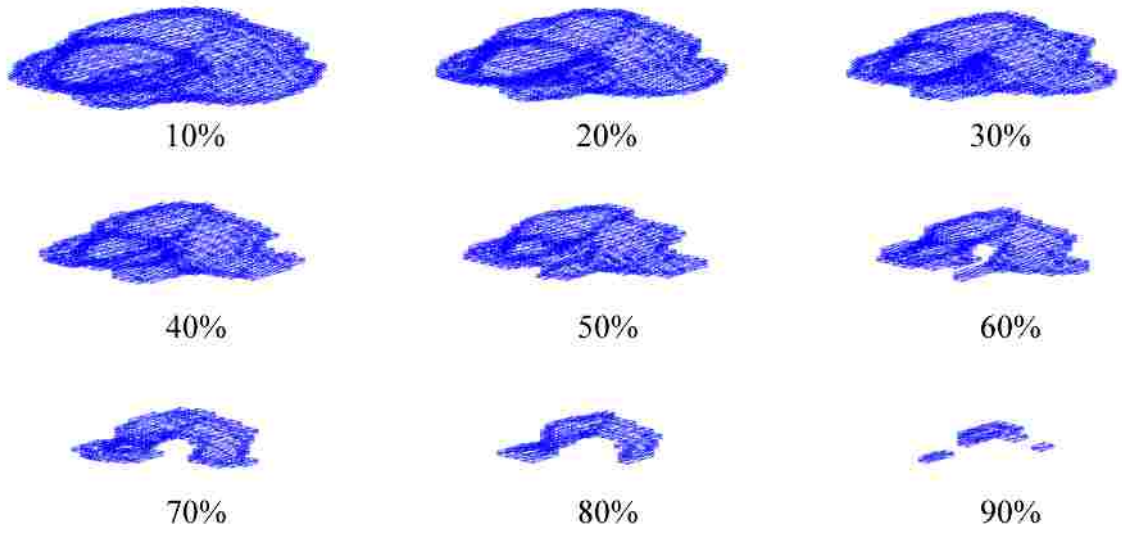

FIG. 11: Contours from the density map synthetically generated from a portion of the structure 1FUR in PDB corresponding to a nine residue loop between two consecutive helices in the sequence. The percentage below each contour represents the range of grayscale values discarded during binary conversion–10% means the lowest 10% of grayscale values were discarded and 90% means only the top 10% of grayscale-valued voxels were converted into binary voxels of value 1.

3.1.3 CONSTRUCTING ROADMAPS OF INTERIOR VOXELS

We construct a graph of the voxels interior to the contour $\mathcal{C}_\tau$ representing all possible locations a path could take using an interior voxel. First, a fully connected graph $\mathcal{G}$ of all voxels in $\mathcal{I}_{B\tau}$ is constructed, and each edge in $\mathcal{G}$ that intersects a facet in $\mathcal{C}_\tau$ is removed. The resulting graph, $\hat{\mathcal{G}}$, represents the set of segments that can be used to construct a path lying completely within $\mathcal{C}_\tau$.

Euclidean-Weighted Roadmaps

Taking the edges from $\hat{\mathcal{G}}$, we construct a roadmap $\mathcal{G}^E$ with each edge's weight as the Euclidean distance between its endpoints e_1 and e_2, with the well-known formula

$$w_E(\overline{e_1 e_2}) = \sqrt{(x_{e1} - x_{e2})^2 + (y_{e1} - y_{e2})^2 + (z_{e1} - z_{e2})^2}. \tag{4}$$

By finding the shortest path through this Euclidean-weighted graph, we estimate the minimum feasible length of a protein turn through a particular density iso-contour.

Intensity-Weighted Roadmaps

In addition to searching through a solution space with a purely Euclidean metric, we can also weight each edge using the grayscale values of the voxels in $\mathcal{I}_G$. Similar to the approach in [33] where intensity values are used to produce a modified minimal spanning tree, we generate an intensity-weighted roadmap $\mathcal{G}^I$, using the weight formula

$$w_I(\overline{e_1 e_2}) = \frac{w_E(\overline{e_1 e_2})}{2I(e_1)I(e_2)}, \tag{5}$$

where

$$I(\mathcal{I}_{B(x,y,z)}) = v(\mathcal{I}_{G(x,y,z)}), \tag{6}$$

the grayscale value at voxel (x, y, z) in $\mathcal{I}_G$. Many density images have real-valued intensities (in $\mathbb{R}$), weighting some edges in $\mathcal{G}^I$ negatively with Equation 8. Because Dijkstra's algorithm is used to search through $\mathcal{G}^I$, all the values must be translated into the positive reals ($\mathbb{R}^+$) by searching for the minimum intensity d_{min} in $\mathcal{I}_G$ and applying Equation 7 to each voxel to derive a positive-valued image $\mathcal{I}_G^+$:

$$v(\mathcal{I}^+_{G(x,y,z)}) = v(\mathcal{I}_{G(x,y,z)}) + |d_{min}| \tag{7}$$

3.1.4 SHORTEST-PATH SEARCHES

The goal of this approach is to estimate the minimum feasible length a protein turn could take between two helix endpoints using the grayscale information in its density map. Therefore, we use Dijkstra's algorithm to search for the shortest path in an iso-contour roadmap to represent this minimum estimate, with paths computed from $\mathcal{G}^E$ and $\mathcal{G}^I$ denoted $\mathcal{P}^E$ and $\mathcal{P}^I$, respectively. Figure 12 shows the difference between $\mathcal{P}^E$ and $\mathcal{P}^I$ as computed from a contour in PDB 1FUR's density map (see Figure 11, 70%).

Each path is computed starting at one helix endpoint chosen in §3.1.2 and ending at the other. However, these endpoints rarely overlap with any voxels inside $\mathcal{C}_\tau$, as its voxels are only adjacent to intersections of the cubic lattice described in the density image (see Appendix A). Therefore, to obtain the start and end vertices in $\hat{\mathcal{G}}$ for shortest path computation, we find the closest vertex to each helix endpoint w.r.t. Euclidean distance. In the case of intensity-weighting, the closest voxel to a helix endpoint e_n is found using a formula similar to Equation 8:

$$w_I(\overline{e_1 e_2}) = \frac{w_E(\overline{e_1 e_2})}{2I(e_1)}, \tag{8}$$

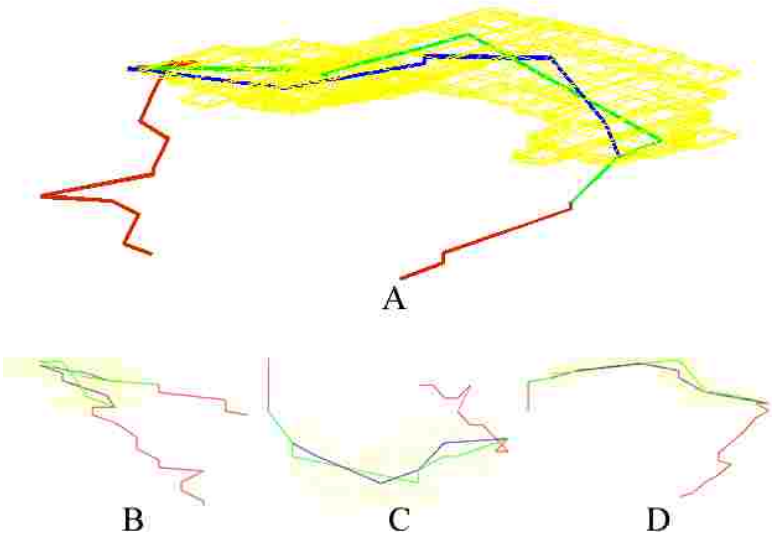

FIG. 12: Euclidean-weighted (blue) and intensity-weighted (green) shortest paths through an iso-contour's (yellow) non-intersecting interior voxel graph. Part A shows an off axis view; B, C and D respectively show views from the x-y, x-z and y-z perspectives.

where e_1 is a voxel in $\mathcal{G}^I$ being tested for the closeness to the helix endpoint e_2, which is implicitly assigned an intensity value of 1.

3.1.5 COMPLEXITY

The inputs are $\mathcal{I}_G$, which has n voxels, and the two detected helices, with exactly 4 endpoints. Converting $\mathcal{I}_G$ to $\mathcal{I}_{B\tau}$ requires visiting each voxel once and performing a constant number of operations, specifically those outlined in Equation 3. Building $\mathcal{C}_\tau$ also requires a single traversal of each voxel with a constant number of comparisons: one for each neighboring voxel cube–6 in total–constructing a surface with $O(n)$ facets. To resolve segmentation in $\mathcal{C}_\tau$, the directed Hausdorff computation requires quadratic time in the number of voxels in each set. However, we know that one set holds exactly two voxels for the helix endpoint pair and the other contains those present within a contour segment. Therefore computing $d_h(A, B)$ requires only linear time w.r.t. the amount of contour segment voxels, which is $O(n)$. Building $\mathcal{G}$ requires a quadratic amount of operations to fully connect the set of voxels. Deriving $\hat{\mathcal{G}}$ requires constructing an *axis-aligned bounding box* (AABB) tree [34] of $\mathcal{C}_\tau$'s $O(n)$ facets–doable in $O(n \log n)$ time–and testing each edge in $\mathcal{G}$ for intersection, merely

TABLE 3: Roadmap based algorithm with time complexities, receiving as input a grayscale image $\mathcal{I}_G$ and pair of helices H. All steps consider the number n of voxels within $\mathcal{C}_\tau$.

minimalPath($\mathcal{I}_G$, H)	$f(n)$	c	n_0
1) Convert $\mathcal{I}_G$ to $\mathcal{I}_{B\tau}$	$\Theta(n)$	–	–
2) Extract $\mathcal{C}_\tau$	$\Theta(n)$	–	–
3) Choose segment and endpoints	$O(n)$	0.6×10^{-3}	150
4) Connect $\mathcal{G}$'s vertices	$\Theta(n^2)$	1.0×10^{-4}	80
6) Compute $\mathcal{C}_\tau \cap \mathcal{G}$	$\Theta(n^2)$	0.2×10^{-3}	50
7) Dijkstra's algorithm	$O(n^4)$	0.2×10^{-8}	100

linear time w.r.t. $||\mathcal{C}_\tau||$. However, because we must search over all $\Theta(n^2)$ edges in $\mathcal{G}$, the entire intersection detection step grows quadratically with respect to the number of interior voxels. Searching $\mathcal{G}i$ for the minimal path with Dijkstra's algorithm requires quadratic time w.r.t. the number of edges in $\mathcal{G}i$ [35], which, because we have $O(n^2)$ edges in $\hat{\mathcal{G}}$, is quartic with respect to n. Although technically $||\mathcal{G}i|| = O(n^2)$, for turns longer than two or three residues the number of nonintersecting edges is far from n^2. Table 3 summarizes the complexities of the major steps along with their complexity constants.

The experiment was performed using an Intel i5 1.7 GHz dual-core processor with 4 GB of 1333 MHz DDR3 RAM. Figure 13 shows the runtimes from the experiment for the cases in Table 5, and Figure 14 shows the runtimes proportional to the expected complexity functions to determine the constants for each operation. Table 3 includes the approximate values for the complexity constant c and voxel amount n_0 where the runtime growth becomes bounded by the theoretical asymptote.

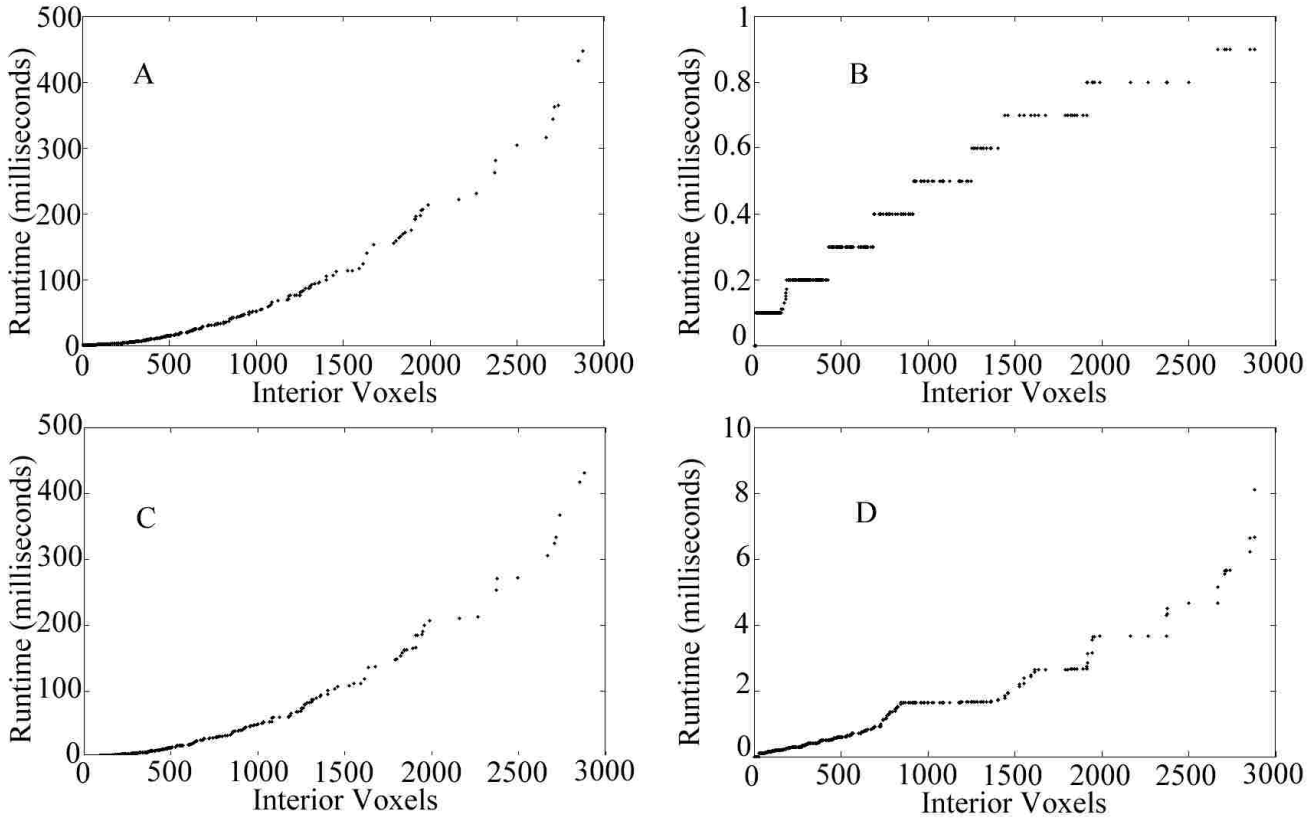

FIG. 13: Runtimes for the graph construction (A), segmentation resolution (B), intersection detection (C) and pathfinding (D) phases of the algorithm.

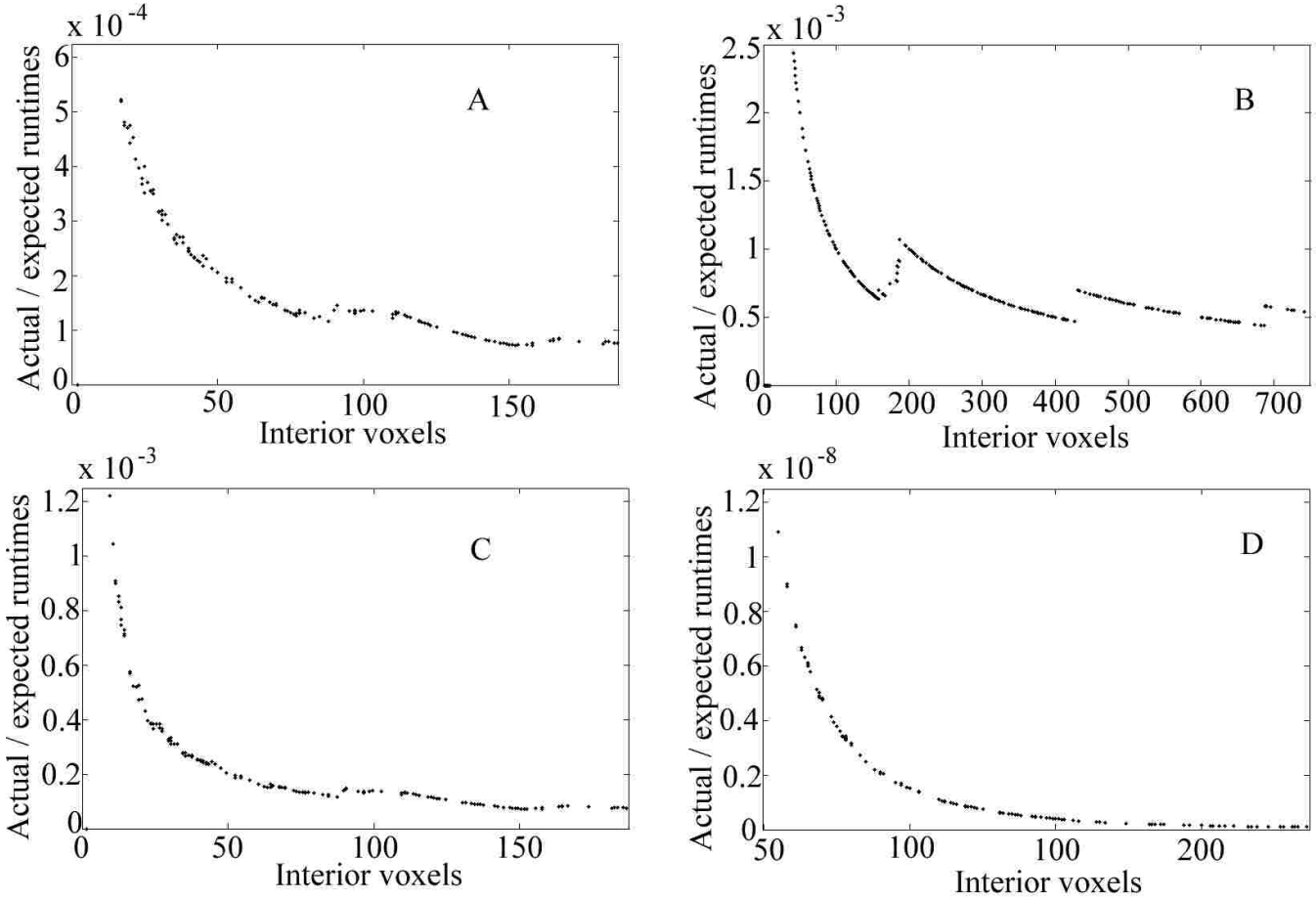

FIG. 14: Experimental runtime proportional to the expected complexity function of each algorithm phase as outlined in Table 3. Labels are identical to those in Figure 13.

3.2 EXPERIMENTAL DESIGN

The test set was adapted from that described in §2.2, by extracting the data from the PDB structure files pertaining only to loop residues in each HLH motif. Density maps were similarly generated for the extracted structure elements at 8 Å, and the same detected helices were used for the basic evaluation of the algorithm. The maximum (d_{max}) and minimum (d_{min}) density values for each map were gathered and nine equidistant values for τ in $[d_{min}, d_{max}]$ were used to extract a series of iso-contours. The shortest paths $\mathcal{P}^E$ and $\mathcal{P}^I$ were then calculated for each contour.

3.2.1 RESULTS

Our goal is to search for an estimate on the minimum feasible length of a given protein loop's density image. With the exception of two test cases, every estimate is shorter than the expected length of the test case. As expected, the length of $\mathcal{P}^E$ decreases as τ increases (extracting narrower contours from $\mathcal{I}_G$). $||\mathcal{P}^I||$ seems to be more sensitive to changes in τ, but approaches $||\mathcal{P}^E||$ as $\tau \to 1$. Figure 15 shows $\mathcal{P}_E$ and $\mathcal{P}^I$ for each $\mathcal{C}_\tau$ for $\tau \in [0.1, 0.9]$. and Table 4 displays the corresponding empirical data.

Tables 5 and 6 show the relative errors of $\mathcal{P}^E$ and $\mathcal{P}^I$ (respectively) for each contour extracted from each test case, and Figure allContourRelativeErrorGraph shows the results in graphical format. In general, the error tends to decrease as τ increases to a certain point, and in many cases, especially where segmentation is present, may slightly increase as $\tau \to 1$. An important observation is that we are searching for a minimum bound, and nearly all reported errors are negative values, meaning the measurements are shorter than the true length. The only exceptional cases are for very short turns with only a single residue, where even small errors in helix detection greatly reduce the accuracy of the measurements in terms of relative error. See §3.2.2 for a more in-depth treatment of helix detection error and its effect on measurements. For subsequent tests, we use $\tau = 0.7$ to extract contours. A more sophisticated approach to automatically determining the optimal value for τ, perhaps using segmentation information, is desirable and a possible goal in future work.

TABLE 4: Accuracies of $\mathcal{P}^E$ and $\mathcal{P}^I$ (corresponding to the superscripts in the column headers) for a 9 residue turn in PDB 1FUR. τ is the normalized threshold used to obtain $\mathcal{C}_\tau$. Diff and RelErr are calculated as in §2.2.1, assuming an expected loop length $l = 3.8\text{Å} * (1+\text{AA})$. For this table, $l = 3.8\text{Å} * (1+9) = 38.0\text{Å}$.

τ	Measured^E	Diff^E	RelErr^E	Measured^I	Diff^I	RelErr^I
0.1	24.152203	-13.847797	-36.441571	24.942386	-13.057614	-34.362142
0.2	24.257274	-13.742726	-36.165068	25.371037	-12.628963	-33.234113
0.3	25.797116	-12.202884	-32.112852	27.445488	-10.554512	-27.775031
0.4	26.365347	-11.634653	-30.617507	28.691454	-9.308546	-24.496175
0.5	26.86568	-11.13432	-29.300843	29.7043	-8.2957	-21.83079
0.6	28.321766	-9.678234	-25.469038	30.402267	-7.597733	-19.994034
0.7	28.650608	-9.349392	-24.603662	32.282134	-5.717866	-15.047015
0.8	29.056169	-8.943831	-23.536397	31.817974	-6.182026	-16.268489
0.9	25.319426	-12.680574	-33.369932	25.319426	-12.680574	-33.369932

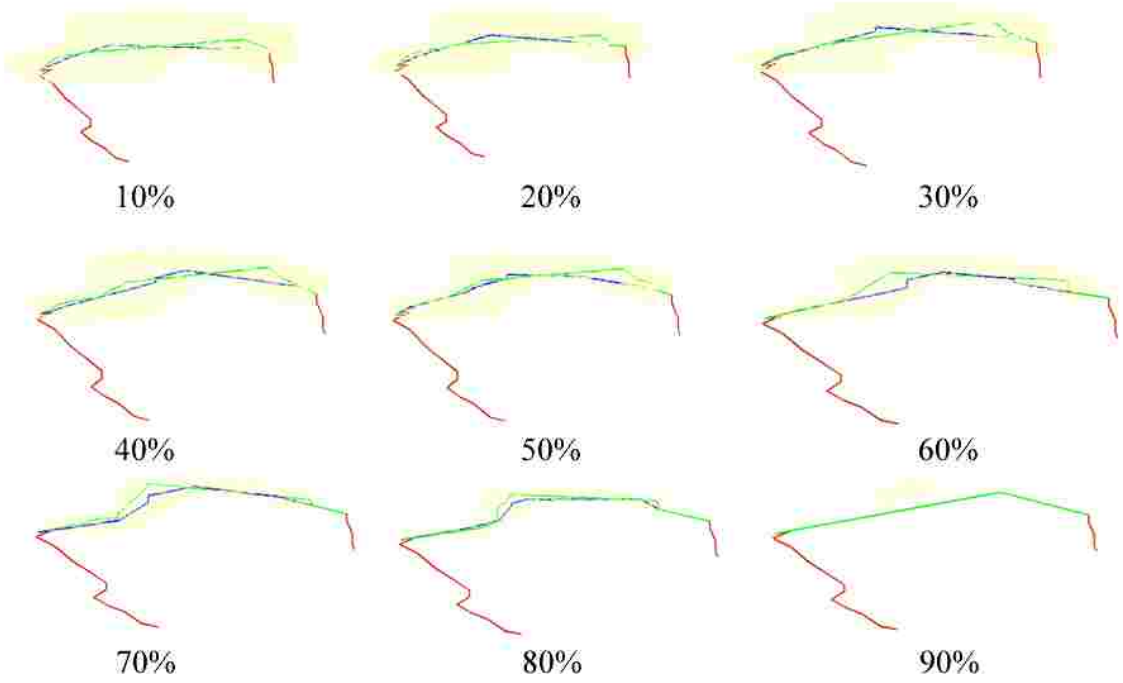

FIG. 15: Euclidean-weighted (blue) and intensity-weighted (green) paths connecting detected helices (red) through the iso-contours (yellow) from Figure 11.

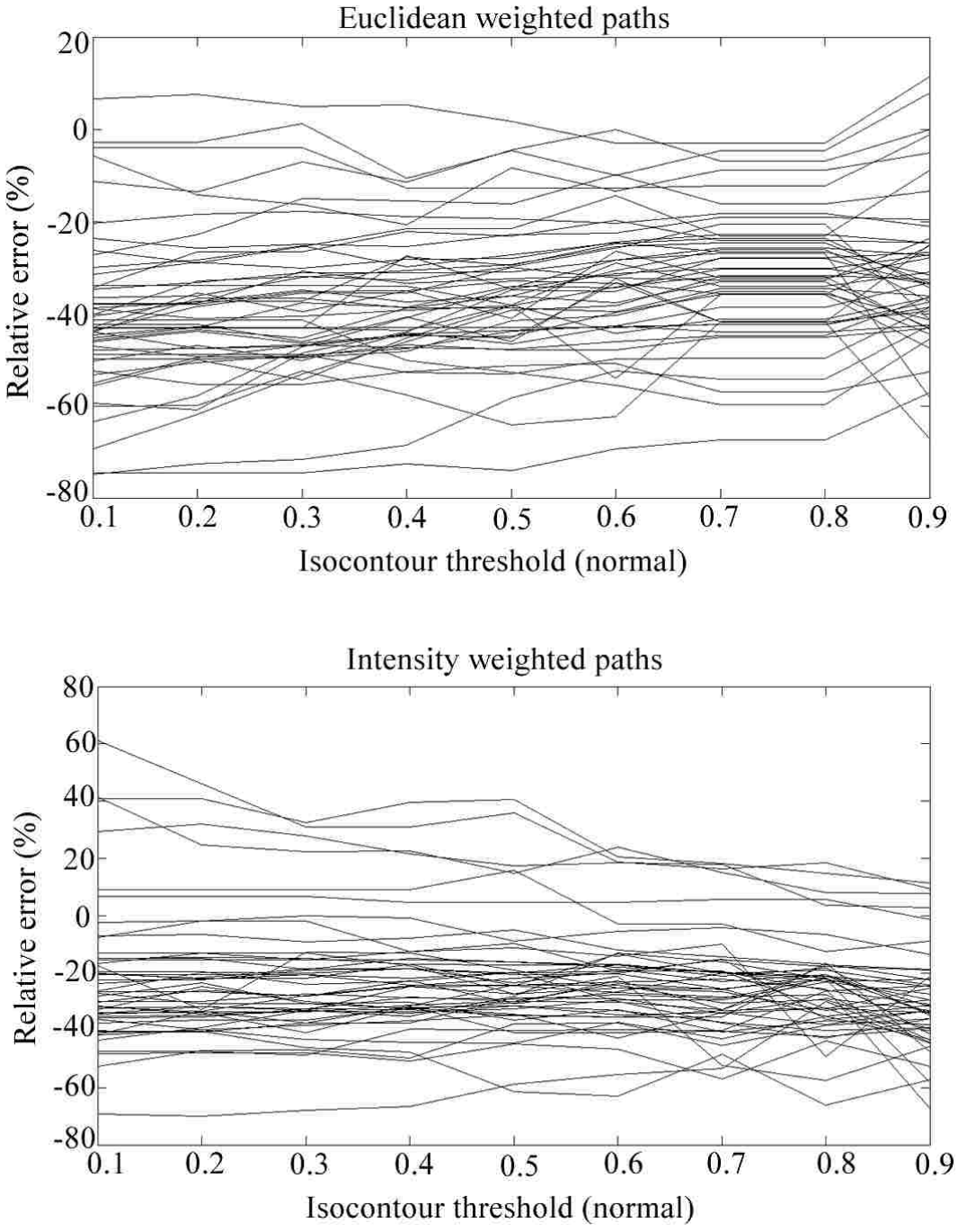

FIG. 16: Relative error plotted against the iso-contour thresholds for each test case in Tables 5 (top) and 6 (bottom). In both cases, $\tau = 0.7$ seems to produce the best results in terms of relative error.

TABLE 5: Relative errors of $\mathcal{P}^E$ for contours with $\tau \in [0.1, 0.9]$ ID is the PDB identifier the test case comes from, AA is the number of residues in the turn, used to calculate the expected length as described in Table 4. Columns headings 0.1, 0.2, et cetera are values of τ used to extract $\mathcal{C}_\tau$ and the column data are the resulting relative errors as percentages.

ID	AA	Expected	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1ARO	1	7.6	-11.30	-13.54	-7.11	-11.45	-4.48	-0.06	-6.84	-6.84	-0.11
1B0B	1	7.6	-2.89	-2.89	1.20	-10.73	-4.63	-10.05	-4.65	-4.65	7.81
1BGP	1	7.6	6.57	7.52	4.96	5.24	1.77	-2.93	-2.93	-2.93	11.45
1BQB	1	7.6	-4.01	-4.01	-4.01	-12.73	-12.73	-12.73	-12.28	-12.28	-1.19
1GUX	1	7.6	-41.39	-41.39	-41.39	-50.23	-52.74	-55.54	-59.75	-59.75	-45.52
1B43	2	11.4	-52.39	-55.29	-55.29	-52.63	-51.22	-50.87	-57.06	-57.06	-52.66
1B89	2	11.4	-26.23	-29.00	-25.37	-21.50	-21.50	-14.42	-23.17	-23.17	-8.88
1BD8	2	11.4	-42.34	-42.34	-46.14	-40.75	-46.14	-32.48	-41.81	-41.81	-39.47
1BPY	2	11.4	-43.13	-43.13	-43.24	-43.60	-38.19	-44.39	-41.19	-41.19	-36.47
1BR1	2	11.4	-47.04	-49.34	-49.34	-46.71	-47.89	-47.89	-45.10	-45.10	-43.13
1DU0	3	15.2	-5.75	-14.26	-16.32	-20.68	-8.37	-13.48	-8.93	-8.93	-5.11
1FJL	3	15.2	-33.85	-33.85	-33.85	-34.00	-38.11	-29.00	-34.14	-34.14	-31.00
1FK5	3	15.2	-43.00	-43.00	-43.00	-43.00	-43.00	-43.00	-42.38	-42.38	-38.61
1FUR	3	15.2	-44.05	-36.62	-36.62	-38.78	-38.78	-39.55	-34.62	-34.62	-36.22

TABLE 5 Continued

ID	AA	Expected	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1H0M	3	15.2	-44.04	-47.54	-46.77	-47.41	-47.80	-46.13	-44.80	-44.80	-36.35
1A87	4	19	-44.56	-42.82	-43.13	-27.47	-34.77	-31.23	-30.18	-30.18	-33.56
1AIH	4	19	-48.93	-48.93	-48.93	-52.76	-53.01	-49.88	-49.62	-49.62	-38.62
1AJ8	4	19	-74.57	-74.57	-74.57	-72.56	-74.14	-69.32	-67.33	-67.33	-57.12
1BMT	4	19	-54.98	-49.57	-46.85	-44.46	-43.60	-42.74	-44.01	-44.01	-42.46
1BOU	4	19	-55.77	-50.00	-54.35	-45.35	-38.35	-54.03	-35.88	-35.88	-25.32
1D8L	5	22.8	-59.45	-60.86	-48.52	-44.01	-34.95	-35.13	-27.96	-27.96	-26.69
1DI1	5	22.8	-43.73	-38.38	-35.48	-35.52	-30.08	-32.35	-32.80	-32.80	-23.63
1DLC	5	22.8	-46.06	-43.90	-46.42	-44.69	-45.26	-33.26	-41.84	-41.84	-25.01
1DNP	5	22.8	-50.30	-46.91	-50.17	-44.18	-46.56	-42.86	-42.41	-42.41	-37.47
1DP7	5	22.8	-23.72	-25.82	-25.04	-25.48	-22.80	-22.57	-19.18	-19.18	-19.65
1CQX	6	26.6	-37.78	-37.78	-30.74	-33.49	-32.36	-31.84	-28.05	-28.05	-33.53
1CSH	6	26.6	-40.64	-37.29	-37.38	-27.70	-29.47	-24.74	-23.72	-23.72	-34.32
1HM6	6	26.6	-74.82	-72.68	-71.55	-68.46	-58.26	-52.51	-54.22	-54.22	-42.51
1MW8	6	26.6	-27.24	-22.81	-15.12	-15.54	-16.16	-9.87	-16.19	-16.19	-13.46
1O6L	6	26.6	-40.32	-32.99	-31.89	-32.75	-29.90	-25.76	-20.54	-20.54	-41.36
1DJX	7	30.4	-69.33	-61.90	-53.03	-46.07	-39.33	-38.37	-32.02	-32.02	-43.56
1E5Q	7	30.4	-49.78	-49.85	-49.56	-48.21	-41.62	-40.44	-35.13	-35.13	-44.45
1FFV	7	30.4	-45.74	-43.56	-45.23	-38.77	-36.14	-37.63	-33.04	-33.04	-43.11

.

TABLE 5 Continued

ID	AA	Expected	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1H99	7	30.4	-37.98	-37.93	-35.28	-35.03	-32.90	-33.46	-35.68	-35.68	-41.38
1IRX	7	30.4	-43.11	-35.55	-39.58	-34.97	-41.06	-26.39	-32.21	-32.21	-27.81
1O6L	8	34.2	-31.40	-28.92	-30.13	-28.25	-27.99	-24.42	-22.85	-22.85	-24.59
1QVR	8	34.2	-34.35	-26.53	-26.57	-22.17	-23.25	-19.78	-24.09	-24.09	-24.71
1S0V	8	34.2	-63.53	-57.86	-47.26	-40.88	-34.39	-28.37	-26.87	-26.87	-58.23
1TAU	8	34.2	-45.30	-43.00	-37.66	-38.62	-34.01	-34.40	-30.30	-30.30	-27.31
1U09	8	34.2	-20.40	-18.45	-17.85	-18.94	-19.46	-20.48	-18.32	-18.32	-21.12
1D6M	9	38	-47.96	-47.59	-47.00	-44.79	-45.04	-39.80	-35.93	-35.93	-47.54
1FUR	9	38	-36.44	-36.17	-32.11	-30.62	-29.30	-25.47	-24.60	-24.60	-33.37
1H32	9	38	-53.32	-50.70	-49.07	-47.60	-43.89	-42.62	-38.62	-38.62	-33.29
1QPC	9	38	-38.93	-41.10	-40.56	-39.23	-38.31	-34.19	-31.80	-31.80	-32.65
1QRT	10	41.8	-60.04	-59.91	-52.38	-57.57	-64.13	-62.30	-41.71	-41.71	-67.31
1RJB	10	41.8	-34.69	-33.21	-31.07	-31.20	-30.92	-28.40	-25.78	-25.78	-27.42
1XO0	10	41.8	-30.07	-28.38	-25.22	-29.85	-27.31	-24.70	-26.57	-26.57	-31.70
2B63	10	41.8	-38.84	-37.44	-34.88	-37.53	-36.15	-30.59	-26.08	-26.08	-37.47

TABLE 6: Relative errors of $\mathcal{P}^I$ for contours with $\tau \in [0.1, 0.9]$. All columns are the same as in Table 5.

ID	AA	Expected	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1ARO	1	7.6	-11.30	-13.54	-7.11	-11.45	-4.48	-0.06	-6.84	-6.84	-0.11
1B0B	1	7.6	-2.89	-2.89	1.20	-10.73	-4.63	-10.05	-4.65	-4.65	7.81
1BGP	1	7.6	6.57	7.52	4.96	5.24	1.77	-2.93	-2.93	-2.93	11.45
1BQB	1	7.6	-4.01	-4.01	-4.01	-12.73	-12.73	-12.73	-12.28	-12.28	-1.19
1GUX	1	7.6	-41.39	-41.39	-41.39	-50.23	-52.74	-55.54	-59.75	-59.75	-45.52
1B43	2	11.4	-52.39	-55.29	-55.29	-52.63	-51.22	-50.87	-57.06	-57.06	-52.66
1B89	2	11.4	-26.23	-29.00	-25.37	-21.50	-21.50	-14.42	-23.17	-23.17	-8.88
1BD8	2	11.4	-42.34	-42.34	-46.14	-40.75	-46.14	-32.48	-41.81	-41.81	-39.47
1BPY	2	11.4	-43.13	-43.13	-43.24	-43.60	-38.19	-44.39	-41.19	-41.19	-36.47
1BR1	2	11.4	-47.04	-49.34	-49.34	-46.71	-47.89	-47.89	-45.10	-45.10	-43.13
1DU0	3	15.2	-5.75	-14.26	-16.32	-20.68	-8.37	-13.48	-8.93	-8.93	-5.11
1FJL	3	15.2	-33.85	-33.85	-33.85	-34.00	-38.11	-29.00	-34.14	-34.14	-31.00
1FK5	3	15.2	-43.00	-43.00	-43.00	-43.00	-43.00	-43.00	-42.38	-42.38	-38.61
1FUR	3	15.2	-44.05	-36.62	-36.62	-38.78	-38.78	-39.55	-34.62	-34.62	-36.22
1H0M	3	15.2	-44.04	-47.54	-46.77	-47.41	-47.80	-46.13	-44.80	-44.80	-36.35
1A87	4	19	-44.56	-42.82	-43.13	-27.47	-34.77	-31.23	-30.18	-30.18	-33.56
1AIH	4	19	-48.93	-48.93	-48.93	-52.76	-53.01	-49.88	-49.62	-49.62	-38.62

TABLE 6 Continued

ID	AA	Expected	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1AJ8	4	19	-74.57	-74.57	-74.57	-72.56	-74.14	-69.32	-67.33	-67.33	-57.12
1BMT	4	19	-54.98	-49.57	-46.85	-44.46	-43.60	-42.74	-44.01	-44.01	-42.46
1BOU	4	19	-55.77	-50.00	-54.35	-45.35	-38.35	-54.03	-35.88	-35.88	-25.32
1D8L	5	22.8	-59.45	-60.86	-48.52	-44.01	-34.95	-35.13	-27.96	-27.96	-26.69
1DI1	5	22.8	-43.73	-38.38	-35.48	-35.52	-30.08	-32.35	-32.80	-32.80	-23.63
1DLC	5	22.8	-46.06	-43.90	-46.42	-44.69	-45.26	-33.26	-41.84	-41.84	-25.01
1DNP	5	22.8	-50.30	-46.91	-50.17	-44.18	-46.56	-42.86	-42.41	-42.41	-37.47
1DP7	5	22.8	-23.72	-25.82	-25.04	-25.48	-22.80	-22.57	-19.18	-19.18	-19.65
1CQX	6	26.6	-37.78	-37.78	-30.74	-33.49	-32.36	-31.84	-28.05	-28.05	-33.53
1CSH	6	26.6	-40.64	-37.29	-37.38	-27.70	-29.47	-24.74	-23.72	-23.72	-34.32
1HM6	6	26.6	-74.82	-72.68	-71.55	-68.46	-58.26	-52.51	-54.22	-54.22	-42.51
1MW8	6	26.6	-27.24	-22.81	-15.12	-15.54	-16.16	-9.87	-16.19	-16.19	-13.46
1O6L	6	26.6	-40.32	-32.99	-31.89	-32.75	-29.90	-25.76	-20.54	-20.54	-41.36
1DJX	7	30.4	-69.33	-61.90	-53.03	-46.07	-39.33	-38.37	-32.02	-32.02	-43.56
1E5Q	7	30.4	-49.78	-49.85	-49.56	-48.21	-41.62	-40.44	-35.13	-35.13	-44.45
1FFV	7	30.4	-45.74	-43.56	-45.23	-38.77	-36.14	-37.63	-33.04	-33.04	-43.11
1H99	7	30.4	-37.98	-37.93	-35.28	-35.03	-32.90	-33.46	-35.68	-35.68	-41.38
1IRX	7	30.4	-43.11	-35.55	-39.58	-34.97	-41.06	-26.39	-32.21	-32.21	-27.81
1O6L	8	34.2	-31.40	-28.92	-30.13	-28.25	-27.99	-24.42	-22.85	-22.85	-24.59

.

TABLE 6 Continued

ID	AA	Expected	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
1QVR	8	34.2	-34.35	-26.53	-26.57	-22.17	-23.25	-19.78	-24.09	-24.09	-24.71
1S0V	8	34.2	-63.53	-57.86	-47.26	-40.88	-34.39	-28.37	-26.87	-26.87	-58.23
1TAU	8	34.2	-45.30	-43.00	-37.66	-38.62	-34.01	-34.40	-30.30	-30.30	-27.31
1U09	8	34.2	-20.40	-18.45	-17.85	-18.94	-19.46	-20.48	-18.32	-18.32	-21.12
1D6M	9	38	-47.96	-47.59	-47.00	-44.79	-45.04	-39.80	-35.93	-35.93	-47.54
1FUR	9	38	-36.44	-36.17	-32.11	-30.62	-29.30	-25.47	-24.60	-24.60	-33.37
1H32	9	38	-53.32	-50.70	-49.07	-47.60	-43.89	-42.62	-38.62	-38.62	-33.29
1QPC	9	38	-38.93	-41.10	-40.56	-39.23	-38.31	-34.19	-31.80	-31.80	-32.65
1QRT	10	41.8	-60.04	-59.91	-52.38	-57.57	-64.13	-62.30	-41.71	-41.71	-67.31
1RJB	10	41.8	-34.69	-33.21	-31.07	-31.20	-30.92	-28.40	-25.78	-25.78	-27.42
1XO0	10	41.8	-30.07	-28.38	-25.22	-29.85	-27.31	-24.70	-26.57	-26.57	-31.70
2B63	10	41.8	-38.84	-37.44	-34.88	-37.53	-36.15	-30.59	-26.08	-26.08	-37.47

3.2.2 BENCHMARKS

Several sources of error exist throughout the approach: noise may be present in the density image and helices may not be perfectly detected. In order to understand the error induced only by the roadmap approach, we isolate the two former sources of error by generating a series of benchmarks for each. EMAN [32] was used to generate specified amounts of noise in the existing density maps–we generated images with 5, 10, 15 and 20% flat-band noise levels as seen in Figure 17, denoted $\mathcal{I}_G^{(0.05)}$, $\mathcal{I}_G^{(0.1)}$, $\mathcal{I}_G^{(0.15)}$ and $\mathcal{I}_G^{(0.2)}$ (which generate iso-contours denoted $\mathcal{C}_\tau^{(0.05)}$ et cetera). The contour with $\tau = 0.7 \times (d_{max} - d_{min})$ (throwing out the lowest 70% of grayscale valued voxels) was extracted from each noisy map for testing. For the ranges of noise tested (up

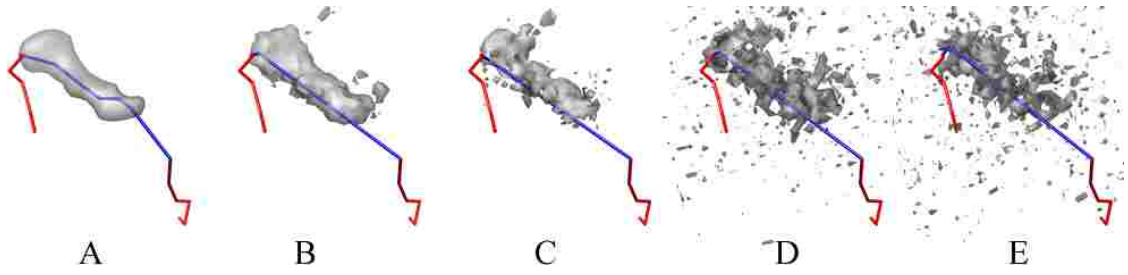

FIG. 17: Increasing levels of artificial flat-band noise applied to the density image generated from an 8 residue turn in PDB 1O6L. The helices are in red and $\mathcal{P}^E$ in blue, with $\mathcal{C}_{\tau=0.7}$ in gray. Part A is the synthetically generated map from the PDB structure; B, C, D and E are $\mathcal{I}_G^{(0.05)}$, $\mathcal{I}_G^{(0.1)}$, $\mathcal{I}_G^{(0.15)}$ and $\mathcal{I}_G^{(0.2)}$ respectively.

to 20%), no drastic change was observed in the length estimates. Figure 18 shows a typical result, with overall relative error changing at mostly the same rate as the noise level increases, in some cases improving. The actual paths drawn can be seen in Figure 17. The deletion of voxels within regions usually without holes in $\mathcal{C}_\tau$ creates a highly segmented contour. In some cases with higher noise, $\mathcal{C}_\tau^{(0.2)}$ is so segmented that $\hat{\mathcal{G}}$ contains no non-intersecting edges. The solution reduces to finding the single segmented voxel with a minimum Hausdorff distance as found in §3.1.2, and drawing a curve containing only the voxel and the two helix endpoints. In future work, a more elegant way to use multiple segmentations would be more appropriate to maximize use of image data.

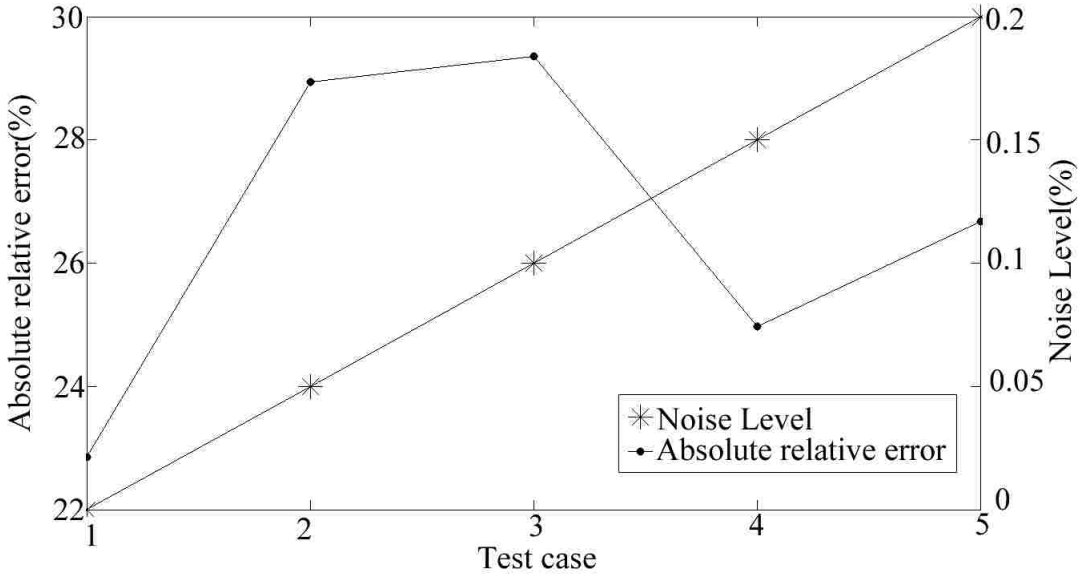

FIG. 18: Absolute relative error (left y-axis) and noise level (right y-axis) of density image for test case PDB 1O6L from Figure 17.

The detected helices were eroded by successively removing endpoints from the piecewise linear curve representations until only a point remained (see Figure 19), recording the detection specificity measure [26] for each erosion. By comparing the marginal error in our algorithm to that of each benchmark set we can isolate the error introduced solely by our method. Helix benchmarking, like noise benchmarking, uses contours with $\tau = 0.7 \times (d_{max} - d_{min})$ Figure 20 shows the helix specificity for each helix erosion plotted with the absolute value of the relative error of the resulting estimate. As in the noise benchmarks, the marginal increase in relative error as helix error increases is insignificant.

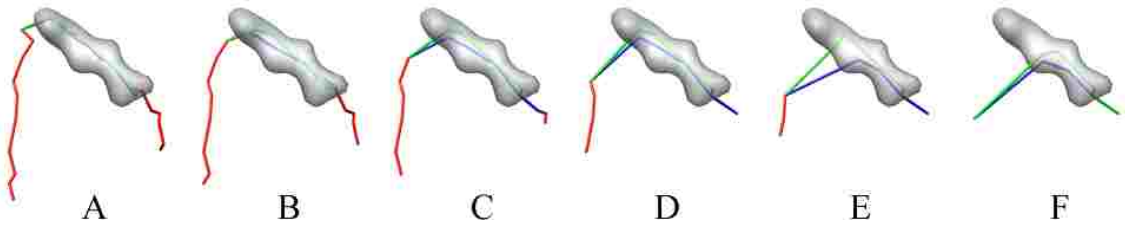

FIG. 19: Eroding detected helices (red lines) surrounding a 7 residue turn's density (gray region) in PDB 1H99, with specificities ranging from 82.8% (leftmost) to 10.3% (rightmost) The blue and green lines represent the euclidean- and intensity-weighted paths respectively.

3.2.3 CONCLUSIONS

Iso-contour roadmaps are a robust method of determining the minimum possible length of a protein loop through a region of density in a cryoEM image. Most of the results have a lower-than-expected estimate on length, as desired, and degrade well in the presence of density noise and preprocessing error. Currently, by taking estimates from a range of contours, the results could be interpreted as a range of possible actual lengths for the protein loop. This presents another goal for future work–to combine the information from several contours.

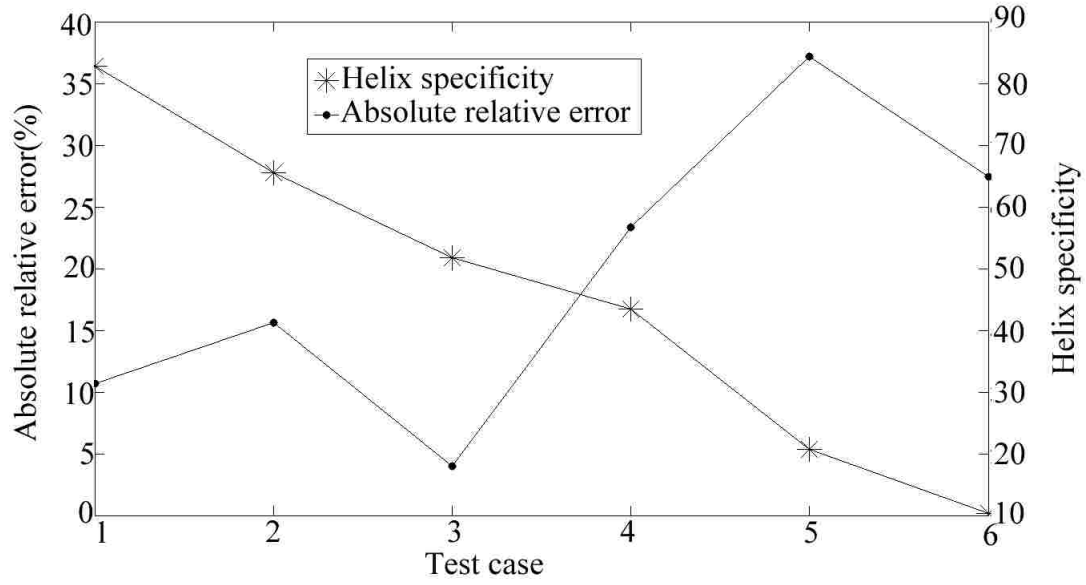

FIG. 20: Absolute relative error (left y-axis) and helix specificity (right y-axis) for the six helix erosions for the test case from PDB 1H99.

CHAPTER 4

CONCLUSIONS

Chapter 2 introduces a robust method to estimate the actual length of a protein loop and Chapter 3 proposes methods for placing bounds on a feasible estimate of that length for a given density map. Both methods perform quite well under the controlled experiments described. More analysis on the parameters for each algorithm should help to push the processes towards full automation. Further work on iso-contour roadmaps may provide new ways to formulate the desired estimates, by extending the intensity-weighted approach and placing additional biological constraints on the paths, such as bond angles and lengths.

BIBLIOGRAPHY

[1] Lawson C, et al.: **EMDatabank.org: unified data resource for CryoEM**. *Nucleic Acids Res* 2011, **39**(Database issue):D456–64.

[2] Zhang X, Sun S, Xiang Y, Wong J, Klose T, Raoult D, Rossmann M: **Structure of Sputnik, a virophage, at 3.5-A resolution**. *Proc Nat Acad Sci USA* 2012, **109**:18431–18436.

[3] Zhang X, Ge P, Yu X, Brannanand J, Bi G, Zhang Q, Schein S, Zhou Z: **Cryo-EM structure of the mature dengue virus at 3.5-A resolution**. *Nat Struct Mol Biol* 2012, **20**:105–110.

[4] Lu Y, Strauss C, He J: **Incorporation of Constraints from Low Resolution Density Map in Ab Initio Structure Prediction Using Rosetta**. In *Proceeding of 2007 IEEE international Conference on Bioinformatics and Biomedicine Workshops* 2007:67–73.

[5] Baker M, Jiang W, Wedemeyer W, Rixon F, Baker D, Chiu W: **Ab initio modeling of the herpes virus VP26 core domain assessed by CryoEM density**. *PLoS Comput Biol* 2006, **2**(10):e146.

[6] Topf M, Lasker K, Webb B, Wolfson H, Chiu W, Sali A: **Protein structure fitting and refinement guided by cryo-EM density**. *Structure* 2008, **16**(2):295–307.

[7] Topf M, Baker M, John B, Chiu W, Sali A: **Structural characterization of components of protein assemblies by comparative modeling and electron cryo-microscopy**. *J Struct Biol* 2005, **149**(2):191–203.

[8] Lu Y, He J, Strauss C: **Deriving topology and sequence alignment for the helix skeleton in low-resolution protein density maps**. *J Bioinform Comput Biol* 2008, **6**:183–201.

[9] DiMaio F, Tyka M, Baker M, Chiu W, Baker D: **Refinement of Protein Structures into Low-Resolution Density Maps Using Rosetta**. *Journal of Molecular Biology* 2009, **392**:181–190.

[10] 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.

[11] Nasr KA, Ranjan D, Zubair M, He J: **Ranking valid topologies of the secondary structure elements using a constraint graph**. *J Bioinform Comput Biol.* 2011, **9**(3):415–30.

[12] Lindert S, Staritzbichler R, Wotzel N, KarakaS M, Stewart P, Meiler J: **EM-fold: De novo folding of alpha-helical proteins guided by intermediate-resolution electron microscopy density maps.** *Structure* 2009, **17**(7):990–1003.

[13] Baker M, Abeysinghe S, Schuh S, Coleman R, Abrams A, Marsh M, Hryc C, Ruths T, Chiu W, Ju T: **Modeling protein structure at near atomic resolutions with Gorgon**. *Journal of Structural Biology* 2011, **174**(2):360–373.

[14] Jiang W, Baker M, Ludtke S, Chiu W: **Bridging the information gap: computational tools for intermediate resolution structure interpretation**. *J Mol Biol* 2001, **308**(5):1033–44.

[15] Palu A, He J, Pontelli E, Lu Y: **Identification of Alpha-Helices from Low Resolution Protein Density Maps**. In *Proceeding of Computational Systems Bioinformatics Conference (CSB)* 2006:89–98.

[16] Baker M, Ju T, Chiu W: **Identification of secondary structure elements in intermediate-resolution density maps**. *Structure* 2007, **15**:7–19.

[17] Kong Y, Zhang X, Baker T, Ma J: **A Structural-informatics approach for tracing beta- sheets: building pseudo-C(alpha) traces for beta-strands in intermediate- resolution density maps**. *J Mol Biol* 2004, **339**:117–30.

[18] Zeyun Y, Bajaj C: **Computational Approaches for Automatic Structural Analysis of Large Biomolecular Complexes**. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on* 2008, **5**(4):568–582.

[19] Si D, Ji S, Nasr K, He J: **A machine learning approach for the identification of protein secondary structure elements from electron cryo-microscopy density maps**. *Biopolymers* 2012, **97**(9):698–708.

[20] Ju T, Baker ML, Chiu W: **Computing a family of skeletons of volumetric models for shape description**. *Computer Aided Design* 2007, **39**(5):352–60.

[21] McGuffin L, Bryson K, Jones D: **The PSIPRED protein structure prediction server**. *Bioinformatics* 2000, **16**(4):404–5.

[22] Ward J, McGuffin L, Buxton B, Jones D: **Secondary structure prediction with support vector machines**. *Bioinformatics* 2003, **19**(13):1650–5.

[23] Pollastri G, McLysaght A: **Porter: a new, accurate server for protein secondary structure prediction**. *Bioinformatics* 2005, **21**(8):1719–20.

[24] Pollastri G, Przybylski D, Rost B, Baldi P: **Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles**. *Proteins* 2002, **47**(2):228–35.

[25] Abeysinghe S, Ju T: **Shape modeling and matching in identifying protein structure from low resolution images**. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling* 2007.

[26] Biswas A, Si D, Nasr KA, Ranjan D, Zubair M, He J: **Improved efficiency in cryo-EM secondary structure topology determination from inaccurate data**. *J Bioinform Comput Biol.* 2012, **10**(3):1242006–1–1242006–16.

[27] Wu Y, Chen M, Lu M, Wang Q, Ma J: **Determining Protein Topology from Skeletons of Secondary Structures**. *J Mol Biol* 2005, **350**:571–586.

[28] Sun W, He J: **Native Secondary Structure Topology has Near Minimum Contact Energy among All Possible Geometrically Constrained Topologies**. *Proteins: Structure, Function and Bioinformatics* 2009, **77**:159–73.

[29] Whitford D: *Proteins: Structure and Function*. Wiley 2005.

[30] Veltkamp RC: **Shape Matching: Similarity Measures and Algorithms**. In *Proceedings of the International Conference on Shape Modeling & Applications*, SMI '01, Washington, DC, USA: IEEE Computer Society 2001:188–, [[http://dl.acm.org/citation.cfm?id=882486.884078]].

[31] Cormen T, Leierson C, Rivest R, Stein C: *Introduction to Algorithms*. MIT Press, 3 edition 2009.

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

[33] Yuan X, Trachtenberg J, Potter S, Roysam B: **MDL constrained 3-D grayscale skeletonization algorithm for automated extraction of dendrites and spines from fluorescence confocal images**. *Neuroinformatics* 2009, **7**(4):213–232.

[34] **Cgal, Computational Geometry Algorithms Library**. [Http://www.cgal.org].

[35] Siek J, Lee L, Lumsdaine A: *Boost Graph Library, The: User Guide and Reference Manual*. C++ In-Depth Series, Addison-Wesley Professional, 1st edition 2001.

[36] Short J: **Image Processing Software: MRC Laboratory of Molecular Biology** 2013, [[http://www2.mrc-lmb.cam.ac.uk/research/locally-developed-software/image-processing-software/image]].

[37] Munkres J: *Topology*. Prentice Hall, 2 edition 1999.

APPENDIX A

MRC IMAGE FORMAT

One of the inputs to the algorithms is a 3D grayscale image. EMDB distributes these images in the MRC format [36], and the synthetic test images were generated in the same format using EMAN. Following is a brief explanation of the data structure.

The voxels are grouped hierarchically by slices (i), then rows (j), and finally columns (k). The origin can be found by looking down on the first slice from above and locating the bottom left voxel, marked as (O_x, O_y, O_z) with a black dot in Figure 21. The parameters *xlen*, *ylen* and *zlen* are provided in the MRC header and can be used to find the dimensions of the image.

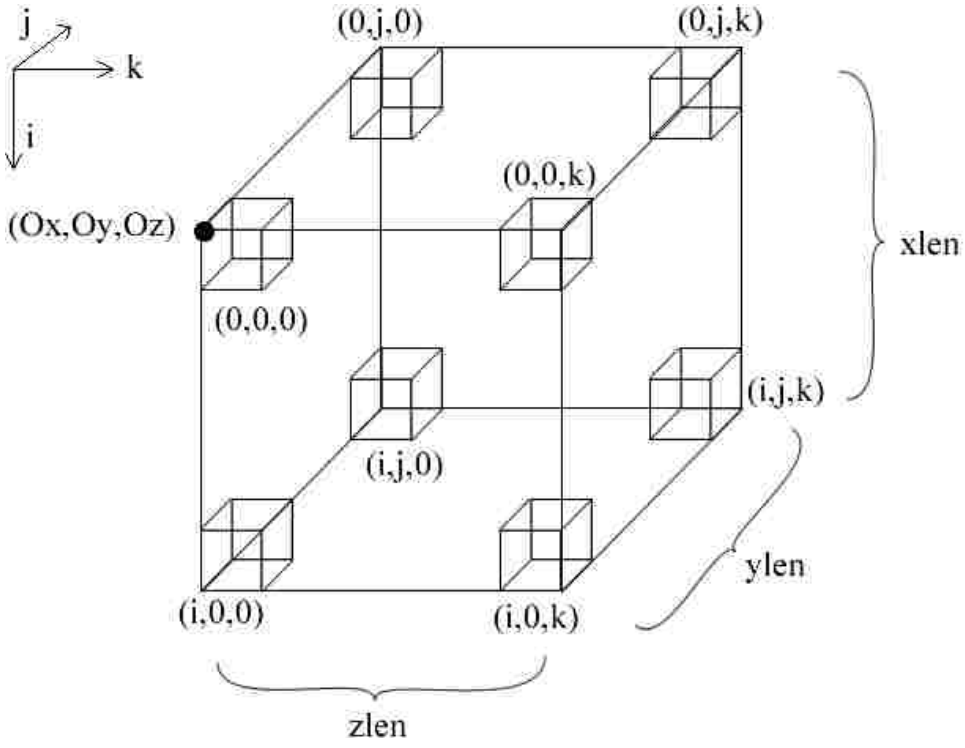

FIG. 21: The layout and addressing of voxels in an MRC image, including coordinate orientation. The eight extremal voxels are shown with their addresses.

Figure 22 shows how to derive the distance between voxels using MRC header data detailing the dimensions of each voxel (m_x, m_y and m_z). For our purposes, $m_x = m_y = m_z$ and the distances between voxel centers ($\frac{xlen}{m_x}$, $\frac{ylen}{m_y}$ and $\frac{zlen}{m_z}$) are always 1 Å.

MRC images are integer addressed, so the true location of each voxel must be calculated to align the image with helices and draw the correct path. For a given voxel $(i, j, k) \in \mathbb{Z}^3$, its actual location in $\mathbb{R}^3$ is:

$$(x, y, z) = (O_x + i * m_x, O_y + j * m_y, O_z + k * m_z), \tag{9}$$

which, because $m_x = m_y = m_z = 1$, becomes

$$(x, y, z) = (O_x + i, O_y + j, O_z + k). \tag{10}$$

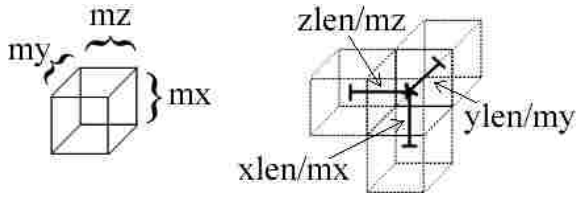

FIG. 22: Relationship between size of MRC voxels and distances between them.

APPENDIX B

HAUSDORFF DISTANCE

The Hausdorff distance is a geometrical relation between two sets of points that utilizes the Euclidean norm to form a metric space [37]. It comes in an assymmetric form, called the *directed* Hausdorff distance, defined for two sets of points A and B as:

$$h(A, B) = \max_{a \in A} \min_{b \in B} d(A_a, B_b) \tag{11}$$

and

$$h(B, A) = \max_{b \in B} \min_{a \in A} d(A_a, B_b). \tag{12}$$

In general, $h(A, B) \neq h(B, A)$; however, there is also a symmetric version, called the *undirected* Hausdorff distance, defined as:

$$H(A, B) = H(B, A) = \max\{h(A, B), h(B, A)\}. \tag{13}$$

Figure 23 provides a simple example. On the left is depicted the computation of $h(A, B)$ and the right shows $h(B, A)$. To find $h(A, B)$, one visits each element $b \in B$, recording the shortest distance to an element $a \in A$ for each, and taking the maximum of all recorded distances. Likewise is done for $h(B, A)$ on the right, and $H(A, B) = H(B, A)$ is simply the larger of the two results. In this case, $h(A, B) = H(A, B) = H(B, A)$.

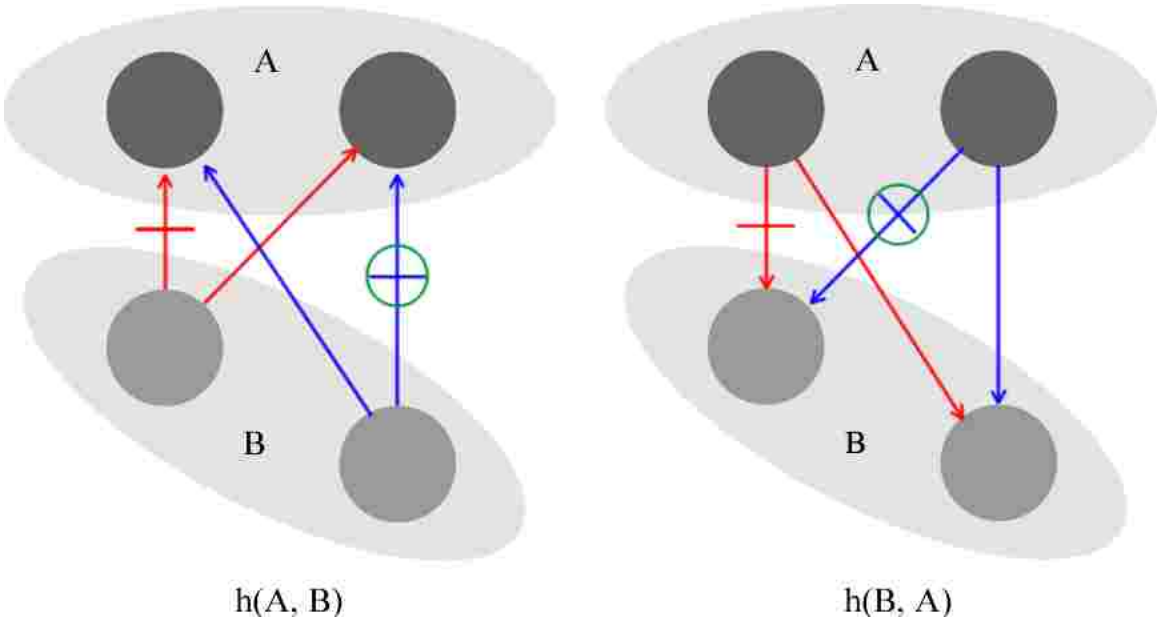

FIG. 23: A simple example of Hausdorff measurement between two sets of points A and B. The short bars intersecting the distance arrows between set elements mark the shortest distances for the inner minimizing function, and the circled bar marks the maximum of the minima, defined as the directed Hausdorff distance.

VITA

A. R. McKnight
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

Andrew McKnight first began programming on a Texas Instruments TI-85 calculator in a middle school geometry class. His post-secondary academic career began at community college in northern Virginia, and after an extended break pursuing a culinary career and traveling, resumed at community college in Charlottesville, VA, where he received his Associate's degree in computer science. He then transferred to Old Dominion University, where he enrolled in an accelerated Master's program soon after his exposure to the field of bioinformatics. After receiving his Bachelor's degree, he would publish three research papers (with a fourth in the works at the time of this writing) and present findings at conferences held by IEEE in Philadelphia and by VMASC (Virginia Modeling and Simulation Center) in Suffolk, VA, where he received the Gene Newman Award for Outstanding Research for best conference paper. He currently resides in Boston, MA with his wife.

Typeset using LaTeX.