EMatch: Discovery of High Resolution Structural Homologues of Protein Domains In Intermediate Resolution Cryo-EM Maps

Keren Lasker, Oranit Dror, Maxim Shatsky, Ruth Nussinov, and Haim Wolfson

APPENDIX I

EFFICIENT NCCC CALCULATION IN REAL SPACE

Given a target grid f of size N and a template grid t of size $n = n_x \cdot n_y \cdot n_z$, we describe an efficient calculation of NCCC values (Equation 1) for all voxels in f in $\Theta(nN)$ time. The algorithm accelerates the practical running time of the naive calculation and does not affect the accuracy of the results when the template grid's reference frame is parallel to the principal axes of the shape it holds. We recommend to use this algorithm in cases where $n << N$, since otherwise calculating the convolution in frequency space is much more efficient.

As demonstrated in [**?**], the numerator in the NCCC formula (Equation **??**) is a convolution between f and t', where $t' = t - \bar{t}$. This convolution can be calculated in real space in $\Theta(nN)$

· K. Lasker, O. Dror, M. Shatsky and H.J. Wolfson are with the School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv 69978, Israel. Email: {kerenl,oranit,maxshats,wolfson}@post.tau.ac.il

· R. Nussinov is with the Department of Human Genetics and Molecular Medicine, Sackler Faculty of Medicine, Tel Aviv University, Tel Aviv 69978, Israel and with the Basic Research Program, SAIC-Frederick, Center for Cancer Research Nanobiology Program, NCI-Frederick, Bldg 469, Rm 151, Frederick, MD 21702, USA. Email: ruthnu@post.tau.ac.il

time. The calculation of the term under the square root in the denominator can be divided into the calculation of two independent factors, σ_t and σ_f:

$$\sigma_f(x) = \sum_{u_j \in U} (f(u_j + x) - \bar{f}_U(x))^2 \qquad (1)$$
$$\sigma_t = \sum_{u_j \in U} (t(u_j) - \bar{t})^2$$

Note that σ_t is constant. We show that $\sigma_f(x)$ can be efficiently calculated in $\Theta(N)$ time for all voxels. For each voxel $x = (x_x, x_y, x_z)$ in f the following holds:

$$\begin{aligned}\sigma_f(x) &= (\sum_{u_j \in U} f^2(u_j + x)) - \frac{1}{n}(\sum_{u_j \in U} f(u_j + x))^2 \\ &= s_2(x) - \frac{1}{n} s_1(x)^2 \qquad (2)\end{aligned}$$

The values of s_1 for all voxels of the target grid are recursively calculated in linear time as follows. Given $s_1(x)$, $s_1(x+l_z)$ (where l_z is a step of one voxel in the Z-axis direction) is equal to: $s_1(x) - b_{xy}(x) + b_{xy}(x + (n_z - 1) \cdot l_z)$, where $b_{xy}(x)$ is the sum of all voxels in the EM grid such that their centers $\{(v_x, v_y, v_z)\}$ satisfy $\{(v_x, v_y, v_z) | (v_x, v_y, v_z) \in U + x, v_z = x_z\}$. The $\{b_{xy}\}$ values are calculated in $\Theta(n)$ [?]. Similar manipulations are applied on s_2.

APPENDIX II

SATISFACTORY CYLINDER SEGMENTATION

We are given an undirected graph $G = (V, E)$ and a cylinder predicate D, as defined in **??**. The output is a satisfactory cylinder segmentation of G.

Algorithm Description. We begin with an initial segmentation $S = \{\{v_i\}\}_{i=0}^{|V|-1}$ such that each v_i is associated with a single non-background voxel from the input EM grid. Then, we join pairs of regions until no pair of regions can be joined.

Specifically, the satisfactory cylinder segmentation is constructed from a number of seed vertices using a variant of Breadth First Search (BFS) as follows. First, we sort all vertices according to the score of their associated voxels (as defined in the method section) and add them to a *seed-queue* in descending order. Then, the vertex, R, at the top of the seed-queue is given as a seed vertex for the BFS traversal. In each iteration of the traversal we join to R a newly discovered vertex R_i that satisfies $D(R \cup R_i)$. The neighboring vertices of R_i in G are explored only if $D(R \cup R_i)$ is satisfied. If R_i has already been discovered by another seed vertex, we add it to a *visited-queue* of R. When no vertex can be joined to R we mark all the vertices that have been joined to R as being discovered and remove them from the seed-queue.

Next, we iterate over the vertices $\{R_j\}$ in the visited-queue and examine whether they can be joined to R. If $D(R_j \cup R)$ is satisfied, we join R_j to R. Finally, if R was joined with any node from the visited-queue, we assign R to be the new R_k, where R_k is a region with lowest region index that was joined with R and we update the edges of G accordingly. We repeat the BFS procedure until the seed-queue is empty.

Theorem 1: The algorithm results in a satisfactory cylinder segmentation.

Proof: S **is not too Coarse.** We show that S is not too coarse under the assumption that D is true for pairs of connected regions. Let us assume, on the contrary, that S is too coarse. This means that there is a refinement $S' \neq S$ that is not too fine. By definition, each $R'_i \in S'$ is contained or equal to some $R_j \in S$ and there is at least one region $R_k \in S$ that is broken into $\{R'_{k_i} \in S'\}_{i=1}^{l}$ such that $R'_{k_i} \subset R_k$ and $\bigcup\limits_{i=0}^{l} R'_{k_i} = R_k$. Since R_k satisfies D, R_k is a connected component. Thus, for each $R'_{k_i} \subset R_k$, there it at least one $R'_{k_j} \subset R_k$ such that R'_{k_i} and R'_{k_j} are neighboring regions. Since D is true for connected regions, $R'_{k_i} \cup R'_{k_j}$ satisfies D. Let us assume that R'_{k_i} was constructed after R'_{k_j}. According to the algorithm the two regions should have been linked after R'_{k_j} was inserted to the visited-queue of R'_{k_i}, which is a contradiction.

S **is not too Fine.** We show that S is not too fine under the assumption that D is true for pairs of connected regions. Let us assume, on the contrary, that S is too fine. This means that there is at least one subset of regions $\{R_{k_i} \in S\}_{i=1}^{l}$ such that each R_{k_i} satisfied the cylinder-like predicate and $S' = S \cup (\bigcup_{i=1}^{l} R_{k_i}) \setminus \{R_{k_1}, ..., R_{k_l}\}$ is a valid segmentation. Thus, $\bigcup_{i=1}^{l} R_{k_i}$ is a connected region since S' and as shown in the first section of the proof should have been linked into a single region according to the algorithm, which is a contradiction. ■

Complexity Analysis. We show that the complexity of the satisfactory cylinder segmentation algorithm is $O(|V| \log |V|)$. First we sort the vertices in V according to the scores of their associated voxels in $\Theta(|V| \log |V|)$ time. Iterating over all vertices in the seed-queue is done in $O(|V|)$ time. Since we visit each edge a constant amount of times and the cylinder predicate verification is done in constant time the total running time of the graph traversals is $O(|V|)$ (since $|E|$ is linear in $|V|$).

We show that the verification of whether $R_i \cup R_j$ satisfies D can be done in constant time. Each region R_i maintains its principal components and direction-array. The direction-array stores all of the most likely helix orientations of the region's voxels. For a given sampling angle ρ, the size of the direction-array is bounded by $\frac{\pi}{4\rho}$, which for the default sampling ($\rho = \pi/12$) equals to eighteen. Given a pair of regions R_i and R_j with pre-calculated principal components and direction-arrays, verification of $D(R_i \cup R_j)$ requires constant time. Verification of the first and second conditions of the predicate requires the calculation of the principal components of $R_i \cup R_j$. The principal components of a set S of points in R^3 are the eigen vectors of its 3×3 covariance matrix. Assuming that we have already calculated the average point and covariance matrix of S_i and S_j, we can calculate the average point and the covariance matrix of $S_i \cup S_j$ in

$O(1)$. Hence, the calculation of the principal components of $R_i \cup R_j$ is done in constant time. Verification of the second condition of D is linear in the number of elements in the direction-arrays of both R_i and R_j. Since this number is bounded by $\frac{\pi}{2\rho}$ and does not depend on the size of the input EM grid, the time complexity of this test is practically constant. The third condition is implicitly satisfied by the cylinder segmentation method, since in each iteration we link only neighboring regions.