BIOINFORMATICS

Vol. 22 no. 14 2006, pages e81–e89
doi:10.1093/bioinformatics/btl252

A probabilistic approach to protein backbone tracing in electron density maps

Frank DiMaio[1,2,*], Jude Shavlik[1,2] and George N. Phillips[3,1]

[1]Computer Sciences Dept., [2]Biostatistics and Medical Informatics Dept. and [3]Biochemistry Dept., University of Wisconsin—Madison, USA

ABSTRACT

One particularly time-consuming step in protein crystallography is interpreting the electron density map; that is, fitting a complete molecular model of the protein into a 3D image of the protein produced by the crystallographic process. In poor-quality electron density maps, the interpretation may require a significant amount of a crystallographer's time. Our work investigates automating the time-consuming initial backbone trace in poor-quality density maps. We describe ACMI (Automatic Crystallographic Map Interpreter), which uses a probabilistic model known as a Markov field to represent the protein. Residues of the protein are modeled as nodes in a graph, while edges model pairwise structural interactions. Modeling the protein in this manner allows the model to be flexible, considering an almost infinite number of possible conformations, while rejecting any that are physically impossible. Using an efficient algorithm for approximate inference—belief propagation—allows the most probable trace of the protein's backbone through the density map to be determined. We test ACMI on a set of ten protein density maps (at 2.5 to 4.0 Å resolution), and compare our results to alternative approaches. At these resolutions, ACMI offers a more accurate backbone trace than current approaches.

Contact: dimaio@cs.wisc.edu

1 INTRODUCTION

Determining the folding of a protein—that is, the three-dimensional spatial configuration of the atoms in a protein—has long been an important problem in biochemistry. With some exceptions, a protein's structure is uniquely determined from its linear amino-acid sequence. Unfortunately, no known algorithm can determine this unique structure from sequence, and scientists are forced to rely upon laboratory methods in order to determine protein structures. Several experimental methods exist, the most popular of which—accounting for about 80% of protein structures determined to date—is x-ray crystallography.

There has been significant recent interest in high-throughput structure determination [1]. One particularly time-consuming step in crystallography is interpretation of the electron map, that is, finding the location of all the protein's atoms in a three-dimensional image of the protein. In this paper, we describe ACMI (Automatic Crystallographic Map Interpreter), an algorithm that automates the process of tracing the backbone in electron density maps.

ACMI consists of two main components: a *local matching* component that locates individual amino acids in the density map, and a *global constraint* component that uses prior knowledge of the protein's structure to eliminating false positives from the local matching. ACMI combines these two with an efficient inference algorithm that can infer the protein's backbone in an electron density map. ACMI's model is *probabilistic*: throughout the interpretation it represents each residue as a probability distribution over the electron density map. This property—not being contrained to force each residue to a single location—is advantageous as it naturally handles noise in the map, errors in the input sequence, and disordered regions in the protein.

2 CRYSTALLOGRAPHY BACKGROUND

Protein crystallography is a very labor-intensive undertaking. First, the protein must be produced in large quantities and purified. Protein crystals then have to be grown, which usually requires testing a significant number of crystallization conditions and solvents. Once the crystals are finally available, a beam of x-rays is fired through the crystal. The lattice of protein molecules that comprise the crystal diffracts this x-ray beam, and produces a pattern of spots on a plate. These spots represent the intensities of a Fourier-transformed picture of the protein. Further laboratory experiments are used to determine the phases corresponding to these intensities. Finally, a Fourier transform converts these intensities into an *electron density map*: a three-dimensional image of the protein.

The final step in x-ray crystallography is *interpreting* this electron density map, converting it into a representation that is usable by biologists. During interpretation, the crystallographer must locate—given the amino-acid sequence of the protein—the coordinates of the centers of all the protein's atoms. This interpretation can be extremely time-intensive; a crystallographer may spend weeks (even months!) interpreting a poor-quality electron density map.

The electron density map is defined on a 3D lattice of points covering the *unit cell*, or basic repeating unit in the protein crystal. The crystal's unit cell may contain multiple copies of the protein related by *crystallographic symmetry*, one of the 65 regular ways a protein can pack into the unit cell. Rotation/translation operators relate one region in the unit cell (the *asymmetric unit*) to all other symmetric copies. Furthermore, the protein may form a multimeric complex (e.g. a dimer, tetramer, etc.) within the *asymmetric unit*. In all these cases is up to the crystallographer to isolate and interpret a single copy of the protein.

An overview of the interpretation task is illustrated in Figures 1 and 2. In both figures, the electron density map—a 3D function over the unit cell—is illustrated as an isocontoured surface. Figure 1a

*To whom correspondence should be addressed.

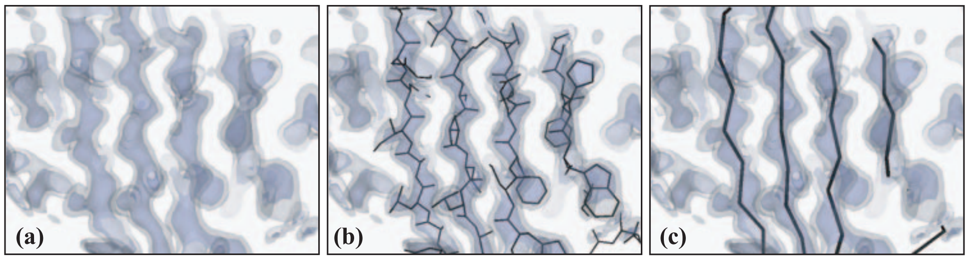

Fig. 1. An over view of electron density map interpretation. Given the amino acid sequence of the protein and a density map (a), the crystallographer's goal is to find the positions of all the proteins atoms (b). Alternatively, a backbone trace (c), reduces each residue to a single point. ACMI automates determination of the backbone trace.

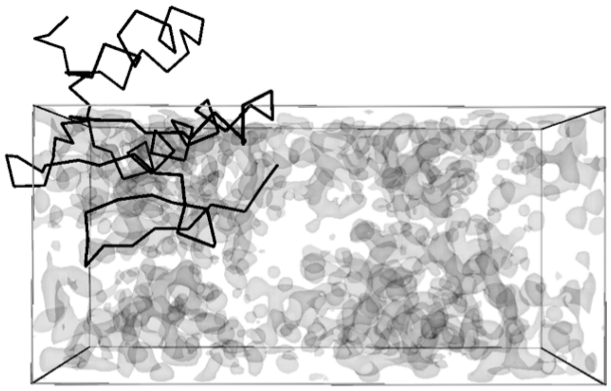

Fig. 2. The electron density map over an entire unit cell. One copy of the protein is indicated. This unit cell contains two symmetric copies, which wrap around the map boundary.

illustrates a small portion of the electron density map. The sticks in Figure 1b show the location of bonds between atoms. Figure 1c shows only the lines between adjacent C_α atoms of the protein. This C_α trace (or backbone trace) is the main concern of this paper. Finally, Figure 2 shows the scale of the problem, illustrating a complete unit cell's electron density. This map contains two crystallographically symmetric copies of a protein.

One measure of overall quality of an electron density map is the *resolution* of the map. When placed in an x-ray beam, some protein crystals diffract the beam better than others. In general, the more the crystal diffracts the beam, the better quality the map. This is illustrated in Figure 3, which shows a short protein's electron density at a variety of resolutions (lower values of resolution mean a higher-quality electron density map). At high resolutions (2Å or better resolution) individual atoms are visible, and automated interpretation is straightforward [2]. However, above about 2.5Å, details of individual atoms are smeared, and atom-based methods tend to fail. Several approaches have attempted automatically interpreting these maps [3,4] and have met with some success. However, interpretations produced by these methods are often messy and require significant crystallographer effort to ''fill in the gaps.''

3 OVERVIEW OF THE ALGORITHM

A high-level overview of ACMI's two main components is illustrated in Table 1. ACMI includes a *local matching* component, where individual residues are probabilistically located in the map, independent of all other residues, and a *global constraint*

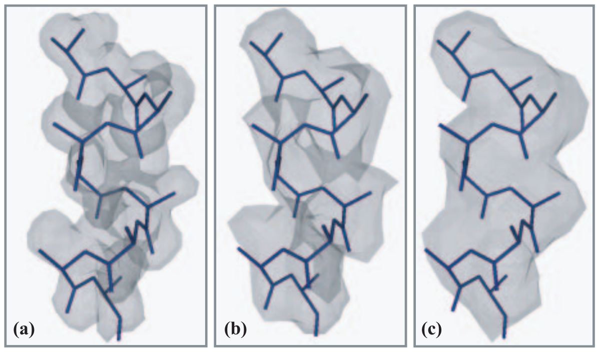

Fig. 3. The electron density map for the same protein fragment at (a) 2Å, (b) 3Å, and (c) 4Å map resolution.

component, where the backbone chain is built up, also probabilistically, from the local matches, taking into account the chemical laws governing the physical structure of proteins.

The local-matching component of our algorithm makes use of a library of existing sequence-specific 5-mer templates. That is, when searching for an individual residue, we actually look for all common conformations of the 5-mer centered at that residue. The local search has high sensitivity, usually matching well to the residue's correct location. However, it suffers from low specificity, producing a significant number of false positives.

ACMI's global-constraint component probabilistically refines these local search results to takes into account prior knowledge of protein structure. Using this prior knowledge, it adjusts the local-match probabilities based on the local match probabilities of other residues. It produces a *physically feasible* interpretation that maximizes the probabilities from the local matching.

ACMI models this physical feasibility with a pairwise Markov field, which represents the probability of a conformation as the product of probabilities between pairs of residues. This pairwise potential is analogous to the pairwise potential energy calculations used in molecular dynamics [5] (although our model does not optimize physical energy but rather *statistical* ''energy'').

4 LOCAL MATCHING

Local matching in ACMI is used to locate individual protein residues in an electron density map. In the poor-quality maps for which ACMI is designed, simple atom-based-refinement methods [2] perform poorly. Empirically, methods using rotamer searching [6], skeletonization [7], or critical points [8] also perform poorly in

Table 1. A pseudocode overview of ACMI's algorithm

```
Procedure ACMI
Given: sequence 'seq' and electron density map 'M'
Find:   putative backbone trace $\mathbf{W}=\{ w_i \}$
  foreach residue $i$ do
      $P(\mathbf{M}|w_i) \leftarrow$ doLocalMatch($seq_i$, $\mathbf{M}$)
  $P(\mathbf{W}) \leftarrow$ enforceGlobalConstraints($P(\mathbf{M}|w_i)$)
  optimal_trace $\leftarrow \{w_i^* | \forall i\ w_i^* = \text{argmax}\ (P(w_i))\}$

Procedure doLocalMatch(seq, M)
Given: sequence 'seq' and electron density map 'M'
Find:   prob. dist. $P(\mathbf{M}|w_i)$ of each residue over map
  • Consider 5-mer centered at each residue
  • Extract instances of 5-mer from PDB, cluster to characterize 5-mer's
    conformational space
  • Perform a 6D search for 5-mer over density map
  • Use a tuning set to convert squared density differences to probabilities
    $P(\mathbf{M}|w_i)$ for each residue $i$

Procedure enforceGlobalConstraints($P(\mathbf{M}|w_i)$)
Given: individual residue probability distributions
Find:   marginal probabilities given structure constraints
  • Model protein backbone structure as a graph
      ○ Nodes model $\alpha$-carbon positions
      ○ Edges enforce structural constraints
  • Probability of an interpretation $\mathbf{W} = \{w_i\}$ given as the product of node
    potentials and edge potentials
        $P(\mathbf{W}\,|\,\mathbf{M}) \propto \prod_{\text{residues } i,j} P(w_i, w_j) \prod_{\text{residues } i} P(\mathbf{M}\,|\,w_i)$
  • Infer marginal probs. given structural constraints
```

these low-resolution maps. The methods that have had the most success in low-resolution maps are those based upon finding large fragments of protein electron density [9]. Thus, we use *sequence-specific* 5-*mer search* to locate individual residues in the electron density map.

Our method is divided into two basic parts, illustrated in Figures 4 and 5. First, we use previously solved structures from the Protein Data Bank to construct a basis set of sequence-specific 5-mer templates. We then perform a 6D (rotation + translation) search in the map for each of the 5-mers in our basis set. The output of this local search is—for each residue—an estimated probability distribution of that residue's presence over the unit cell.

Constructing a Sequence-Specific 5-*mer Basis Set*. ACMI begins this step—illustrated in Figure 4—by walking along the one-dimensional protein sequence, considering a 5-mer centered at each residue. Given this 5-mer, we search a *non-redundant subset* of the PDB [10] (restricted to have less than 25% sequence similarity) for three-dimensional instances of that 5-mer. If there are less than 50 such instances then we search for near neighbors to the 5-mer using increasing PAM distance [11] until we have 50 structures.

It is infeasible to search for all 50+ conformations in the electron density map, so we instead cluster the structures and represent each cluster as a centroid fragment and a weight. When clustering the fragments, we use rotationally-aligned all-atom RMS deviation between fragments as a distance metric (quickly computed as an optimization problem [12]). We use complete-linkage hierarchical

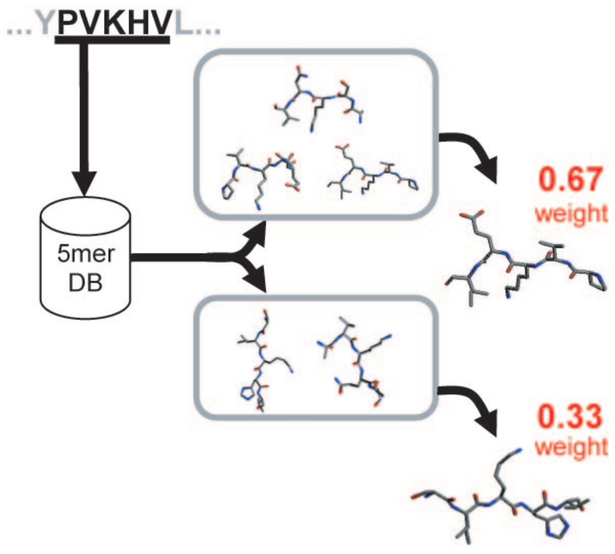

Fig. 4. The 5-mer clustering process. Walking along the amino-acid sequence, we consider a 5-mer centered at each position. We search the database for instances of that 5-mer, and cluster them. Finally, we extract a representative member from each cluster. This characterizes the conformational space of the 5-mer sequence.

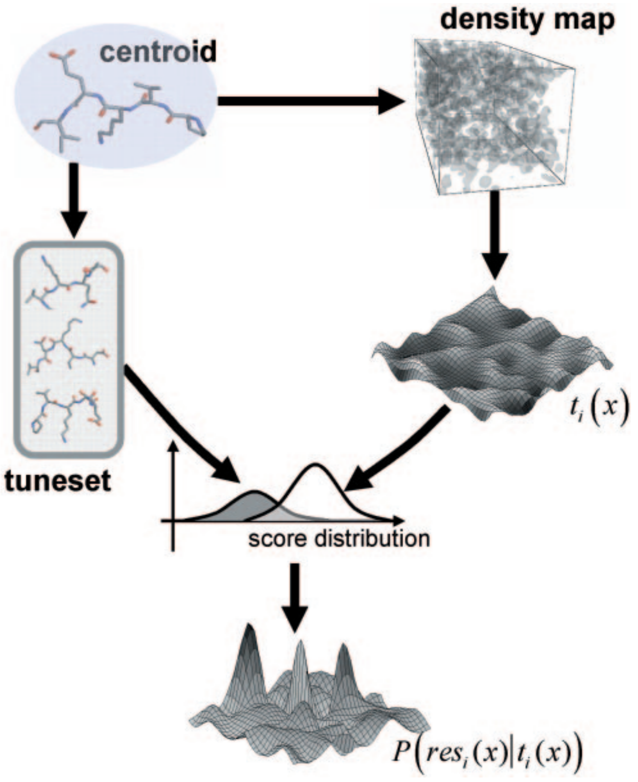

Fig. 5. An overview of the 5-mer template matching process. After we have extracted a representative et of 5-mers for each residue i, we perform a 6D (rotation + translation) search for the fragment in the density map. By also matching the fragment to a tuning set of known structures, we can use Bayes' rule (see Equation 3) to determine the probability distribution of the residue over the density map.

clustering, limiting clusters to have a maximum diameter of 3Å. Any cluster with under 10% representation is thrown out (to limit CPU time in the next step); in all remaining clusters we find a centroid (i.e. representative) fragment. We also record the cluster weight with each centroid fragment, that is, the percent of structures that fell into that cluster. Depending on the ''sequence structural entropy'' of the 5-mer [13], anywhere from 1 to 7 clusters (and resultant centroid fragments) are produced.

The cluster centroids and the weights determined by ACMI represent the conformational space of each specific 5-mer fragment. Using fragments of length five is our way of balancing the trade-off between template size and template specificity. Larger fragments are preferred for recognition in poor-quality maps, but larger fragments have lower representation in the set of already-solved structures. Our non-redundant PDB subset contains about 20% of the 3.2×10^6 possible 5-mers.

Searching for 5-mer centroid fragments. Once the clustering is complete and the cluster centroids have been extracted, we search for instances of the centroids in the electron density map. This process is illustrated in Figure 5. Given a fragment and a target resolution, we can build a map corresponding to what we would expect to see, given the fragment. Then, at each map location, we can compute the mean squared electron density difference $t(\vec{x})$ between the map and the fragment. We compute this difference over all points $\vec{x} = <x_i, y_i, z_i>$ in the electron density map within some distance of the fragment,

$$t(\vec{x}) = \sum_y \varepsilon_f(\vec{y}) \left(\rho'_f(\vec{y}) - \frac{1}{\sigma_\rho(\vec{x})} [\rho(\vec{y}-\vec{x}) - \bar{\rho}(\vec{x})] \right)^2 \qquad (1)$$

where $\rho(\vec{x})$ is the map in which we are searching, $\rho'_f(\vec{x})$ is the standardized fragment electron density, $\varepsilon(\vec{x})$ is a masking function that is nonzero only for points near the fragment, and $\sigma_\rho(\vec{x})$ scales the standard deviations of the fragment and map densities,

$$\sigma^2_\rho(\vec{x}) = \sum_y \varepsilon_f(\vec{y}) [\rho(\vec{y}-\vec{x}) - \bar{\rho}(\vec{y})]^2 \Big/ \sum_y \varepsilon_f(\vec{y}) \qquad (2)$$

We need to perform the fragment search as a 6D search over all rotations plus all translations; fortunately, we can compute $t(\vec{x})$ quickly at a single rotation using FFTs [14]. Additionally, at each position we store the best-matching 5-mer fragment, and the corresponding rotation, for later use.

The electron density difference function $t(\vec{x})$ is a good measure of similarity between regions of density, but we need a way to convert these scores into probability distributions, that is, the probability $P(\vec{x}_i \mid score_i)$ that there is an instance of a specific 5-mer cluster i at location $\vec{x}_i$ given match score $score_i$. ACMI computes this using a tuning set and the application of Bayes' rule. Bayes' rule states that this probability is given as

$$P(\vec{x}_i \mid score_i) = P(score_i \mid \vec{x}_i) \cdot \frac{P(\vec{x}_i)}{P(score_i)} \qquad (3)$$

The terms on the right-hand side are computed or estimated as follows. The probability distribution of match scores over the map, $P(score_i)$, is derived from the actual distribution of match scores over the (unsolved) map. The prior probability on a residue's location over the map, $P(\vec{x}_i)$, is simply a normalization term: we already know (by knowing the protein's sequence) the number of

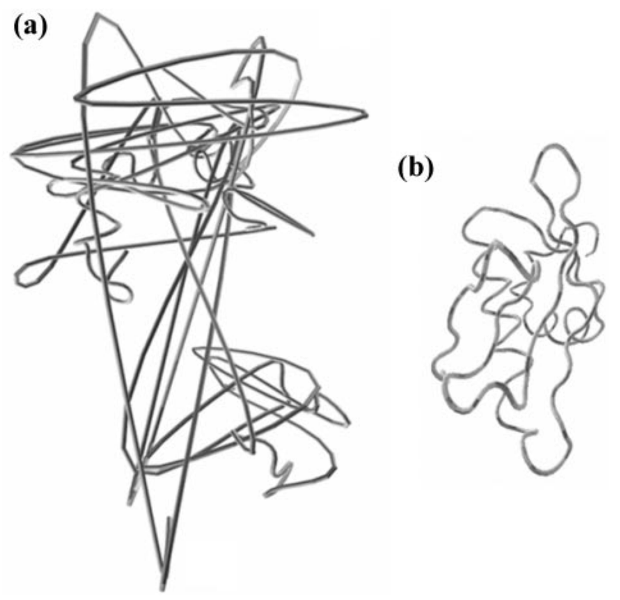

Fig. 6. Two possible backbone traces. The trace (a) maximizes the product of 5-mer match probabilities; however, the resultant protein is physically impossible. We would prefer trace (b) with a lower 5-mer match probability, but which corresponds to a physically-possible structure.

copies of the 5-mer in the electron density map, and we normalize probabilities over the map to sum to this value. However, the first term—the distribution of scores when the 5-mer *matches* the map—is trickier to compute. ACMI *estimates* this term using a tuning set derived from different protein structures from the PDB. This tuning set contains *other instances* from the 5-mer cluster for which we are searching. We match each cluster centroid's density map with each tuneset density map in that centroid's cluster to estimate the distribution of scores given a 5-mer match.

At the end of the local matching procedure, ACMI has computed—for each residue—a probability distribution over the unit cell. That is, for each point in 3D space, we have a probability that each specific 5-mer is positioned at that location. The remainder of the paper describes how our algorithm uses prior knowledge about the structure of the protein to estimate the most probable backbone trace given these probability distributions. Run times for the local matching are significant: for each fragment we have to search ~1900 rotations (20-degree discretization) over the entire electron density map. The total compute time is on the order of CPU-weeks; however, 5-mer matching is trivially parallelized [15].

5 GLOBAL CONSTRAINTS

In Section 4, we computed—for each residue i—the probability distribution over every position x in the unit cell. We can think of this probability as the probability that this map was *generated* by residue i at location and rotation $\vec{w}_i$, that is, $P(\mathbf{M} \mid \vec{w}_i)$. One could presumably select, for each residue, the $\vec{w}_i$ that maximized this probability. However, the resultant trace would likely look like that in Figure 6a. ACMI somehow needs to account for the *structural probability* on the model. That is, it needs to ensure that the proposed structure is a *physically feasible* protein molecule. What we ultimately want to find—given map $\mathbf{M}$—is the configuration of all residues $\mathbf{W} = \{\vec{w}_1, \ldots, \vec{w}_N\}$, such that

$$\underset{\mathbf{W}=\{\vec{w}_1,\ldots,\vec{w}_N\}}{\arg\max}\ P(\mathbf{W}\,|\,\mathbf{M}) \propto P(\mathbf{W}) \cdot \prod_{i=1\ldots N} P(\mathbf{M}\,|\,\vec{w}_i) \qquad (4)$$

The first term accounts for this physical feasibility, in which a proposed structure like that of Figure 6b would have a much higher probability of configuration than Figure 6a.

5.1 Markov field model

To model this "global constraint" probability ACMI uses a *pairwise Markov field model* [16]. A pairwise Markov field model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $i \in \mathcal{V}$ connected by edges $(i,j) \in \mathcal{E}$. Each node in the graph is associated with a (hidden) random variable $\vec{w}_i \in \mathbf{W}$. The graph is conditioned on observation variables $\mathbf{M}$. Each vertex has a corresponding *observation potential* $\psi_i(\vec{w}_i, \mathbf{M})$, and each edge is associated with a *conformational potential* $\psi_{ij}(\vec{w}_i, \vec{w}_j)$. We can represent the full joint probability as

$$p(\mathbf{W}\,|\,\mathbf{M}) = \prod_{(i,j)\in\mathcal{E}} \psi_{ij}(\vec{w}_i, \vec{w}_j) \cdot \prod_{i\in\mathcal{V}} \psi_i(\vec{w}_i, \mathbf{M}) \qquad (5)$$

We are concerned with finding the $\vec{w}_i \in \mathbf{W}$ maximizing this probability, given some $\mathbf{M}$.

Figure 7 shows how we encode a protein in a Markov field model. Each node i represents an amino-acid residue in the protein. The label $\vec{w}_i = \langle \vec{x}_i, \vec{q}_i \rangle$ for each amino-acid residue consists of seven terms: the 3D Cartesian coordinates $\vec{x}_i$ of the residue's *alpha Carbon* (C_α), and four internal parameters $\vec{q}_i$ (an alternate parameterization of three rotational parameters plus the ''bend'' angle formed by three consecutive residues). The *observation potential* $\psi_i(\vec{w}_i, \mathbf{y})$ associated with each residue is the 5-mer probability $P(\mathbf{M}\,|\,\vec{w}_i)$ computed in Section 4.

The *conformation potentials* $\psi_{ij}(\vec{w}_i, \vec{w}_j)$, which model the probability of a particular conformation of the residues in the protein, are further divided into two basic types. Following Suddereth *et al.*'s hand-tracking model [17], ACMI defines *adjacency potentials* associated with each edge connecting neighboring residues (Figure 7b). These potentials ensure that adjacent residues maintain the proper 3.8Å spacing and the proper C_α—C_α—C_α angle. ACMI also defines *occupancy potentials* between non-adjacent residues (Figure 7c), which prevent two residues from occupying the same region in three-dimensional space. Thus, our joint probability is now defined

$$p(\mathbf{W}, \mathbf{M}) = \prod_{\substack{\vec{w}_i,\vec{w}_j\in\mathbf{W} \\ i,j \text{ adjacent}}} \psi_{adj}(\vec{w}_i, \vec{w}_j) \times \prod_{\substack{\vec{w}_i,\vec{w}_j\in\mathbf{W} \\ i,j \text{ nonadjacent}}} \psi_{occ}(\vec{w}_i, \vec{w}_j) \times \prod_{\vec{w}_i\in\mathbf{W}} P(\mathbf{M}\,|\,\vec{w}_i) \qquad (6)$$

Because residues distant on the protein chain are not necessarily distant in space, the graph must be fully connected; that is, every pair of residues is joined by an edge in the Markov field model.

5.1.1 Adjacency potentials The adjacency potentials, which connect every adjacent pair of residues, are further broken down into the product of two constraining functions, a distance constraint function and a rotational constraint function

$$\psi_{adj}(\vec{w}_i, \vec{w}_j) = p_x(\|\vec{x}_i - \vec{x}_j\|) \cdot p_\theta(\vec{w}_i, \vec{w}_j) \qquad (7)$$

The distance constraint is based on the physical fact that, in proteins, the C_α—C_α distance is a nearly invariant 3.8Å. Thus, the potential p_x takes the form of a tight Gaussian around this ideal value.

The internal parameters $\vec{q}_i$ model the 3D rotation of each residue and the angle formed by the residue triple centered at residue i. To simplify the definition of p_θ , we choose to parameterize these four degrees of freedom as two pairs of θ-φ spherical coordinates: the most likely direction of the forward (i + 1) residue and the backward (i − 1) residue. Our local 5-mer matching of Section 4—in addition to computing the probability at a specific location—also remembers the most likely 5-mer centroid and rotation of that centroid. *At each location in the map*, we store four values—θ_f, φ_f, θ_b, and φ_b—indicating the direction of both adjacent residues, based on the direction of these residues in this rotated, best-matching 5-mer.

The angular constraint function p_θ, illustrated in Figure 8, is then—at each position x_i in the map—just a fixed-width Gaussian on a sphere, centered on this preferred orientation. That is, given residue i at the center of the sphere, the highest potential p_θ is when residue i+1 is located on the lightest points on the sphere, at $\langle \theta_f, \varphi_f \rangle$.

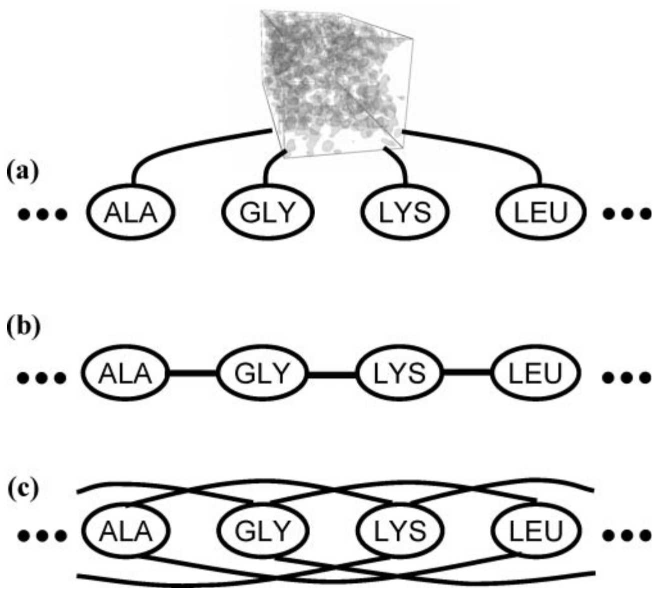

Fig. 7. The structure of our graphical model. The joint probability of a conformation of residues is the product of (a) an observation potential at each node, (b) an adjacency potential between adjacent residues, and (c) an occupancy potential between all pairs of non-adjacent residues.

5.1.2 Occupancy potentials Occupancy potentials are in place to ensure that two residues do not occupy the same location in space. They are defined independently of orientation, and are merely a step function that constrains two (nonadjacent) C_α's be at least 3.0Å apart (the closest distance two nonadjacent residues may get),

$$\psi_{occ}(w_i, w_j) = \begin{cases} 1 & \|x_i - x_j\| \geq 3.0 \\ 0 & \text{otherwise} \end{cases} \qquad (8)$$

It is in this structural potential function that ACMI deals with crystallographic symmetry. We can slightly modify our potential function so that—given symmetric operators $K = \{K_n\}$—two residues may not occupy the same space, nor may *any of their*

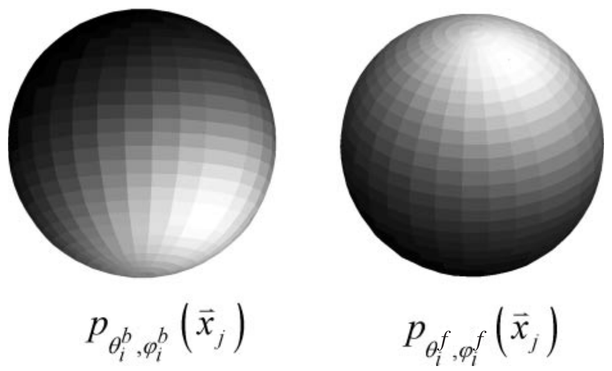
$p_{\theta_i^b,\varphi_i^b}(\vec{x}_j)$
$p_{\theta_i^f,\varphi_i^f}(\vec{x}_j)$

Fig. 8. The angular component $p_\theta(w_i, w_j)$ of ACMI's adjacency potential. When performing our 5-mer matching, ACMI remembers the positions of the adjacent residues in the most-likely match. The potential p_θ is a Gaussian on the sphere's surface centered on this most likely location of each adjacent residue. *This figure shows p_θ at a single location x_i in the unit cell.*

symmetric copies:

$$\psi_{occ}(w_i, w_j) = \begin{cases} 1 & \left(\min\limits_{\substack{\text{symmetric} \\ \text{transforms } K}} \|x_i - K_n(x_j)\| \right) \geq 3.0 \\ 0 & \text{otherwise} \end{cases} \qquad (9)$$

Multiple chains in the asymmetric unit are also handled by ACMI: separate chains are fully connected by edges enforcing occupancy constraints.

5.2 ACMI's inference algorithm: finding the most probable backbone trace

The ultimate goal of ACMI is producing a backbone trace: finding the labels $\mathbf{W} = \{w_i\}$ that maximize the probability of the local *observational potentials* and the global *conformational potentials*,

$$\arg\max_{\mathbf{W}=\{w_i\}} \prod_{\text{residues } i,j} \psi_{ij}(w_i, w_j) \cdot \prod_{\text{residue } i} \psi_i(w_i, \mathbf{M}) \qquad (10)$$

However, solving this exactly for arbitrary graphs is infeasible (dynamic programming can solve this in quadratic time for tree-structured graphs). As an alternative, ACMI uses belief propagation (BP) to compute an approximation to the marginal probability $P(w_i\,|\,\mathbf{M})$ for each residue i, then chooses the maximum marginal label for each residue as the final trace.

Belief propagation is an inference algorithm—based on Pearl's polytree algorithm [18]—that computes marginal probabilities using a series of local messages. At each iteration, a node (i.e., residue) computes an estimate of its marginal distribution (i.e., an estimate of the residue's location in the unit cell) as the product of all incoming messages. The residue then passes a convolution of this product with the corresponding edge potential along each outgoing edge.

$$m^n_{i\to j}(w_j) \propto \int_{\text{unitcell}} \psi_{ij}(w_i, w_j) \cdot \frac{\hat{p}^n_i(w_i)}{m^{n-1}_{j\to i}(w_i)} dw_i \qquad (11)$$

Above, $\hat{p}^n_i$ denotes the estimation of i's marginal at iteration n, that is,

$$\hat{p}^n_i(w_i) \propto \psi_i(w_i, \mathbf{M}) \cdot \prod_{k\in\Gamma(i)} m^n_{k\to i}(w_i) \qquad (12)$$

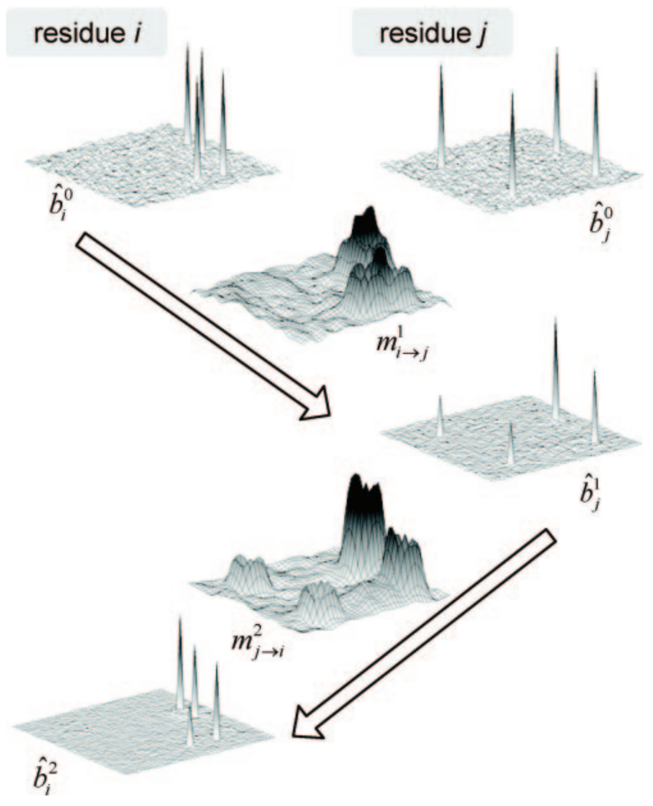

Fig. 9. A simple example of message passing using belief propagation. Given prior probabilities $\hat{b}_i^0$ and $\hat{b}_j^0$, at each iteration, a node i passes a message to a node j indicating i's belief of j's position. For example, a residue knows that an adjacent residue must be 3.8Å away; residue i's message to j consists of these 3.8 Å "bubbles" around its peaks. As BP iterates, the matches that are *structurally supported* by other residues begin to emerge.

Figure 9 illustrates the message-passing with a simple two-dimensional example. In this example, two residues' prior probabilities have their probability mass split among several peaks. Our structural knowledge tells us that residue i must be next to residue j. In the first iteration, residue i passes a message to residue j, that indicates where residue i expects to find residue j (essentially, in a ring around residue i's peaks). Messages in BP are probability distributions marginalized to the message recipient's random variables; that is, this message from residue i to residue j is a function over residue j's position in the unit cell. Residue j passes a message back to residue i indicating where j expects to find i. This example shows that in just two iterations, BP is able to reduce the number of peaks through the use of structural priors.

In graphs without cycles, BP is exact. In graphs with arbitrary topologies, such as ACMI's protein model, there are no guarantees of convergence or correctness; however, empirical results show that *loopy BP* often produces a good approximation to the true marginal [19,20].

5.3 Technical challenges

Even with the computational savings afforded by BP, the size and complexity of both the graph and the space of labels presented ACMI with a number of implementation challenges. Though beyond the scope of this paper, the modifications necessary to BP in order to scale to this problem are discussed in another paper by the authors [21]. This section will briefly discuss some of these scaling issues.

5.3.1 Representation of potentials The label associated with each residue is a continuously-valued, 7-dimensional variable. Nonparametric belief propagation (NBP) [17] is a variant of BP that can handle continuous-valued labels; previous work represented the belief as the sum-of-Gaussians. Our work introduces Fourier-Series NBP, a variant of NBP which represents messages and belief as a set of 3D Fourier coefficients in Cartesian space, which offer a number of benefits for this problem domain. These benefits include natural treatment of periodic boundary conditions and symmetry, no explicit initialization required (as is required with the sum-of-Gaussians), and an efficient message-passing implementation.

5.3.2 Efficient message passing Each message passed requires integrating over the entire unit cell, which naïvely takes running time of the order O(K^2), where K is the number of Fourier coefficients. Unfortunately, for a typical protein, K may be 10^6 to 10^7! For adjacency messages, it is not too much of a problem, as we only need to integrate over a thin spherical shell where ψ_{adj} is nonzero. However, for occupancy messages, this message computation time is significant. Fortunately, because the occupancy potential is only a function of the *distance* between the two connected residues, we can pass the message in O($K \log K$) as a multiplication in Fourier-space.

5.3.3 Structural message aggregation Because our graph is fully connected, in each iteration O(N^2) messages need to be computed and stored, where N is the number of amino-acid residues in the protein. As each message is a probability distribution over the entire unit cell, this is demanding computationally and storage-wise. However, the outgoing structural messages (see Equation 11) at a given node are all quite similar: they only differ in the denominator, which serves to avoid double-counting, making the method exact in tree-structured graphs [19]. However, in loopy graphs, this double-counting is unavoidable. Furthermore, the structural potentials are very diffuse, high-entropy potentials. Other authors have suggested [22] that approximation errors in graphs with this type of potential tend to stabilize.

We can save a significant amount of work if we aggregate all the non-bonded residues, sending them a single structural message (that is, dropping the denominator). ACMI does this, only sending O(N) messages per iteration. Combined, these BP optimizations allow ACMI to handle large proteins with large unit cells. Typical run times (for the BP inference) vary from several hours to a day.

6 EXPERIMENTS

We obtained a set of ten model-phased electron density maps from the Center for Eukaryotic Genomics at the University of Wisconsin-Madison. The maps are all of fairly good resolution—natively 1.5 to 2.5Å—and all have crystallographer-determined solutions. To test ACMI's performance on poor-quality (2.5+ Å) data, we down-sampled these maps by *smoothly diminishing* the intensities of higher-resolution reflections. To avoid truncation effects, and give a more realistic model of low-resolution data, we scaled structure factors by $\exp(-K/R^2)$, where R is the resolution of the structure factor and K is a scaling constant chosen based on the desired resolution (higher values of K smooth the map more). We down-sampled each of our maps to 2.5, 3.0, 3.5, and 4.0 Å resolutions, giving us a total of 32 maps on which to test. We chose $K = R_0^2$, so the signal strength was weakened by $1/e$ at the point of truncation.

We compared the performance of ACMI on these maps to two other automated techniques specialized to low-resolution maps: Ioerger's TEXTAL [4], and Terwilliger's Resolve [3,6]. These two approaches have had the most success handling interpretation in poor-quality maps.

TEXTAL is based on ideas from pattern recognition. Ioerger constructs a set of 15 rotation-invariant density features. Using these features at several radii, TEXTAL trains a neural network to identify C_α atoms. Sidechains are identified by looking at the electron density around each putative alpha carbon, efficiently finding the most similar region in a database, and laying down the corresponding sidechain.

Terwilliger takes a different approach with Resolve. Resolve first looks for large secondary-structure elements, places them into the map, and extends them. A rotamer search places sidechains, aligning sequence to backbone. Both methods have some success in 2.5 to 3.5 Å maps.

After running all three algorithms on the test set, we measured the results using three different metrics:

(1) C_α RMS error between predicted and true structure
(2) percent of the chain solved
(3) percent correct residue identity

Ideally, a method would find a trace with low RMS error, high percent of the chain solved, and high residue identity.

The results at each resolution are summarized in Figure 10. TEXTAL was unable to run on one protein's density maps (at any resolution)—rather than including a terrible score for this map, we gave the benefit of the doubt to TEXTAL and only report results on the nine maps on which it ran. In terms of RMS error (Figure 10a), our algorithm consistently out performs TEXTAL at all resolutions tested. Using a two-tailed pair t test, ACMI outperforms TEXTAL with p values of 0.091, 0.057, 0.012 and 0.11 at 2.5, 3, 3.5, and 4Å, respectively. Resolve performs roughly equivalent to ACMI at 2.5Å resolution; however, at 3, 3.5 and 4 Å, ACMI's performance is much better: a two-tailed t test yields p values of 0.0068, 0.00002 and 0.00004, respectively (both of these t tests only take into account RMS error and not chain coverage).

Figure 10b shows that the percent of the chain covered was roughly equivalent for the three approaches. However, Figure 10c shows that our approach is much better than the others at identifying the proper residue type at a particular location. However, it is important to point out that these related methods are not optimizing residue-identification accuracy. Resolve, for example, will often return a long chain of alanine residues if it cannot identify sidechains, but still gives the correct backbone structure overall. This illustrates a significant difference between ACMI and these alternate approaches: TEXTAL and Resolve build a backbone model, then attempt to align the protein sequence to it. ACMI, alternatively, *uses the sequence of the protein to construct the model*. The result is better identification of amino acids in the map.

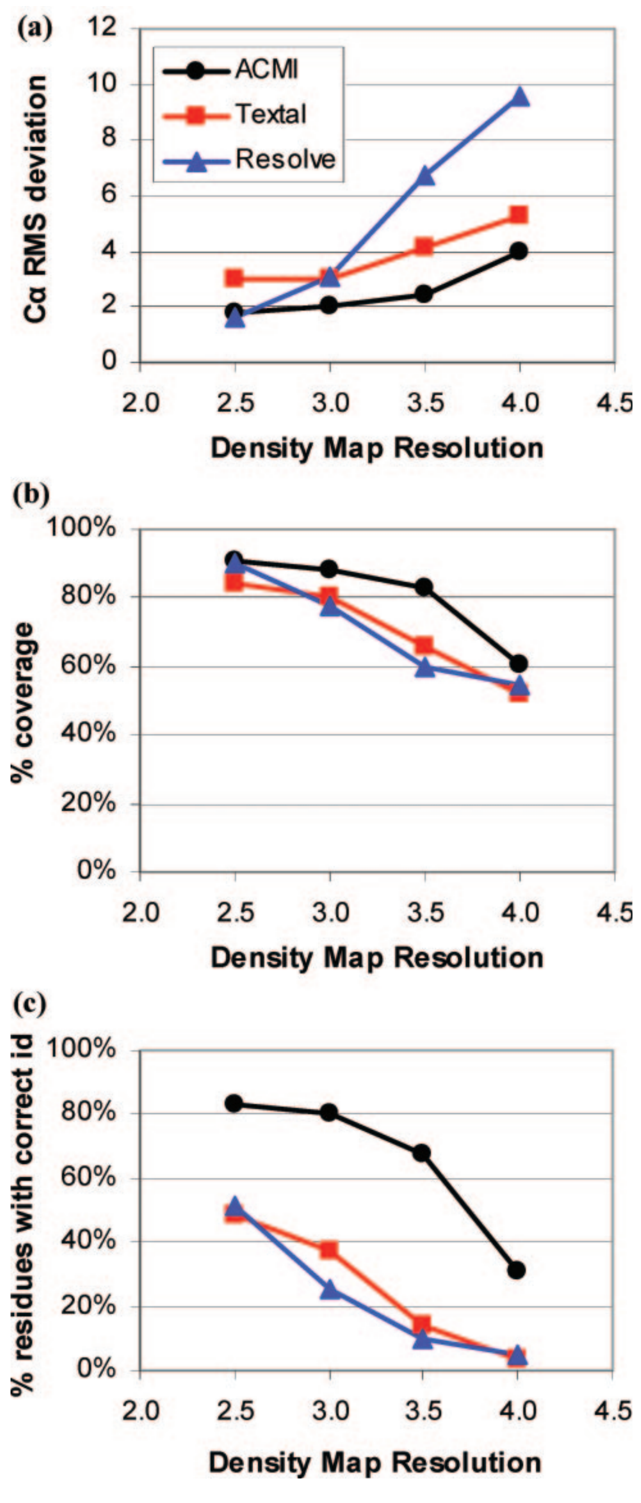

Fig. 10. Graphs showing a comparison of the three algorithms' average interpretation in terms of (a) RMS Error, (b) percent of the chain located, and (c) percent of residues correctly identified.

Additionally, Figure 11 shows scatterplots in which each individually solved electron density map is a point. The *x*-axis indicates ACMI's error; the *y*-axis indicates TEXTAL's (or Resolve's). All points above the diagonal line correspond to maps where ACMI

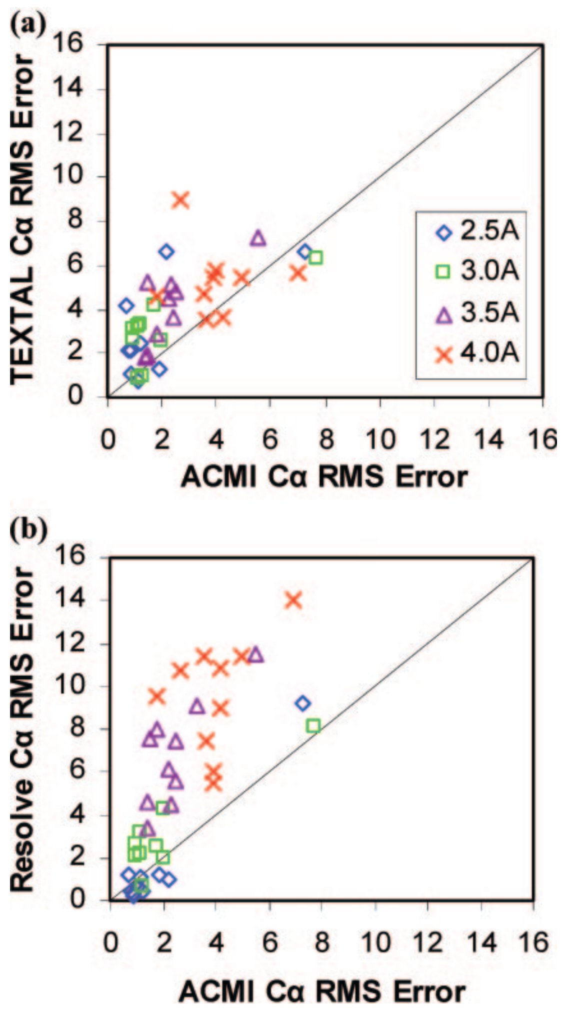

Fig. 11. A scatterplot showing the performance on a protein-by-protein basis, of ACMI versus (a) TEXTAL and (b) Resolve. Each mark is an interpreted map; points above the diagonal are cases where ACMI provided a more-accurate backbone trace.

outperformed TEXTAL (or Resolve). On the majority of structures, our interpretation has a lower RMS error then both of the other algorithms. ACMI is outperformed by Resolve on some high-resolution maps, however, ACMI currently does not perform any post-processing on predicted backbones (e.g. real-space refinement, energy minimization); also, residues are restricted on a grid, limiting accuracy to the grid spacing.

One advantage of ACMI's probabilistic framework is that, in addition to returning a putative trace, ACMI also returns a confidence (i.e. probability) level of each predicted residue. This confidence informs the crystallographer what areas in the map need improvement; alternatively, a high confidence partial trace could be used to improve phasing. Figure 12 illustrates this in an example trace at 3.5 Å resolution, on an structure consisting of two chains of 124 residues each. This is our sixth-best (of ten) traces at this resolution: ACMI finds nine segments with a C_α RMS deviation of 2.3 Å, covering 94% of the backbone. The trace's color indicates the likelihood of its prediction for each residue's location.

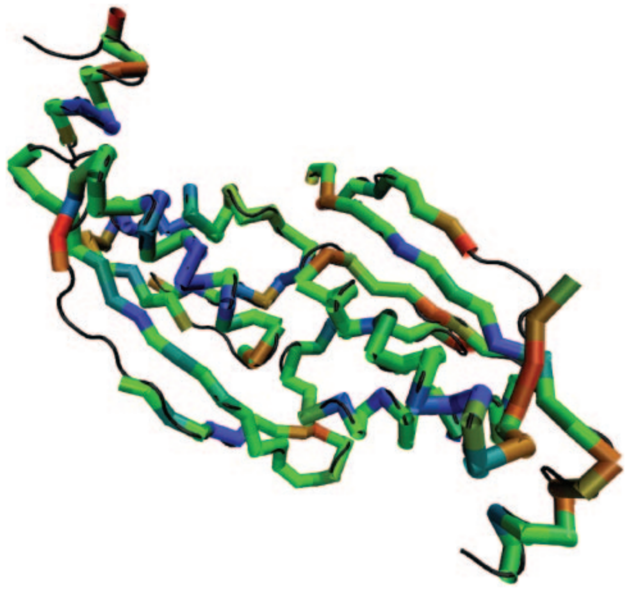

Fig. 12. A comparison of predicted versus actual structure on our sixth-best (of ten) interpretation at 3.5Å resolution. The thin continuous coil is the actual structure, while the thicker segmented chain is ACMI's prediction. The predicted structure is colored by log-likelihood, where least-likely residues are shown in red, and most-likely in blue.

7 CONCLUSIONS AND FUTURE WORK

We describe ACMI, a tool for automatically tracing protein backbones especially designed for poor-quality electron density maps. ACMI combines a local matching procedure and a global constraint procedure in a probabilistic framework that can efficiently infer the locations of backbone atoms in an electron density map. The algorithm provides accurate traces even in poor resolution electron density maps, outperforming both TEXTAL and Resolve above 3 Å map resolution.

One major shortcoming of ACMI is the significant compute time required by its local (5-mer) matching procedure. We need to search for approximately three 5-mer fragments per residue; for each fragment we consider $\sim$1900 rotations. Even for medium-sized unit cells, this takes on the order of CPU-weeks; larger proteins take months. ACMI exploits parallelism, running overnight, using the spare cycles from desktop computers [15]. However, we would like to investigate the use of machine learning algorithms, such as support vector machines or neural networks, to *quickly* match a 5-mer into the density map. We also would like to explore alternative feature representations.

Additionally, as a post-processing step, we would like to augment ACMI with a refinement and sidechain tracing algorithm. In our previous work, we used pictorial structures to place sidechain atoms, given a C_α trace [23]: combining this tool with ACMI would produce a complete molecular model.

Finally, we would like to explore the use of our probabilistic model for phase improvement. In some maps, initial phasing is quite poor. In these maps, a partial structure can be used to significantly improve the initial phasing, revealing previously blurred-out regions in the electron density. Using a high-confidence trace to iteratively improve phasing is a future research direction of ACMI.

By providing accurate interpretations from lower-resolution maps, ACMI reduces the burden on crystallographers when only poor-quality density map data is available. Even when obtaining higher-resolution electron density map data is possible, ACMI allows significant cost savings by making do with poorer-quality maps, speeding up the process of high-throughput protein structure determination.

ACKNOWLEDGEMENTS

We acknowledge support from the National Library of Medicine training grant T15-LM007359 (FD), NLM R01-LM008796 (JS), and National Institutes of Health Protein Structure Initiative Grant GM074901 (GP).

REFERENCES

[1] Berman,H. and Westbrook,J. The impact of structural genomics on the protein data bank. *Am. J. Pharmacogenomics*. 4(4).

[2] Perrakis,A., Sixma,T., Wilson,K. and Lamzin,V. (1997) wARP: Improvement and extension of crystallographic phases. *Acta Crystallographica*, D53.

[3] Terwilliger,T. (2002) Automated main-chain model-building by template-matching and iterative fragment extension. *Acta Crystallographica*, D59.

[4] Ioerger,T. and Sacchettini,J. (2003) The TEXTAL system: Artificial intelligence techniques for automated protein model building. *Methods in Enzymology*, 374.

[5] MacKerell,A.,Jr., Wiórkiewicz-Kuczera,J. and Karplus,M. (1995) An all-atom empirical energy function for the simulation of nucleic acids. *J. Am. Chem. Soc.*, 117.

[6] Terwilliger,T. (2002) Automated side-chain model-building and sequence assignment by template-matching. *Acta Crystallographica*, D59.

[7] Greer,J. (1974) Three-dimensional pattern recognition. *J. Molecular Biology*, 82.

[8] Leherte,L., Glasgow,J., Baxter,K., Steeg,E. and Fortier,S. (1997) Analysis of three-dimensional protein images. *Journal of AI Research*, 7.

[9] Cowtan,K. (2001) Fast Fourier feature recognition. *Acta Crystallographica*, D57.

[10] Wang,G. and Dunbrack,R. (2003) PISCES: A protein sequence culling server. *Bioinformatics*, 19.

[11] Jones,D., Taylor,W. and Thornton,J. (1992) The rapid generation of mutation data matrices from protein sequences. *CABIOS*, 8.

[12] Kabsch,W. (1976) A solution for the best rotation to relate two sets of vectors. *Acta. Crystallographica*, A32.

[13] Huang,S. and Hwang,J. (2004) Computation of conformational entropy from protein sequences. *Proteins: Structure, Function, and Bioinformatics*, 59(4).

[14] Cowtan,K. (1998) Modified phased translation functions and their application to molecular-fragment location. *Acta Crystallographica*, D54.

[15] Thain,D., Tannenbaum,T. and Livny,M. (2005) Distributed Computing in Practice: The Condor Experience. *Concurrency and Computation: Practice and Experience*, 17(2–4).

[16] Geman,S. and Geman,D. (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *PAMI*, 6.

[17] Sudderth,E., Mandel,M., Freeman,W. and Willsky,A. (2004) Visual hand tracking using nonparametric belief propagation. *MIT LIDS Technical Report 2603*.

[18] Pearl,J. (1988) *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufman, San Mateo.

[19] Weiss,Y. and Freeman,W.T. (2001) Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Comp.*, 13.

[20] Murphy,K., Weiss,Y. and Jordan,M. (1999) Loopy belief propagation for approximate inference. *Proc. UAI*.

[21] DiMaio,F. and Shavlik,J. (2006) Improving the Effciency of Belief Propagation in Large, Highly-Connected Graphs. *Working Paper 06-1, UW ML Research Group*.

[22] Ihler,A., Fisher,J. and Willsky,A. (2004) Message Errors in Belief Propagation. *Proc. NIPS*.

[23] DiMaio,F., Shavlik,J. and Phillips,G. (2004) Pictorial structures for molecular modeling: Interpreting density maps. *Proc. NIPS*.