Chapter 1

Machine Learning in Structural Biology: Interpreting 3D Protein Images

Frank DiMaio, Ameet Soni and Jude Shavlik
University of Wisconsin – Madison

1.1 Introduction

This chapter discusses an important problem that arises in structural biology: given an *electron density map* – a three-dimensional "image" of a protein produced from crystallography – identify the chain of protein atoms contained within the image. Traditionally, a human performs this *interpretation*, perhaps aided by a graphics terminal. However, over the past 15 years, a number of research groups have used machine learning to automate density map interpretation. Early methods had much success, saving thousands of crystallographer-hours, but require extremely high-quality density maps to work. Newer methods aim to automatically interpret poorer and poorer quality maps, using state-of-the-art machine learning and computer vision algorithms.

This chapter begins with a brief introduction to structural biology and x-ray crystallography. This introduction describes in detail the problem of density map interpretation, a background on the algorithms used in automatic interpretation, and a high-level overview of automated map interpretation. The chapter also describes four methods in detail, presenting them in chronological order of development. We apply each algorithm to an example density map, illustrating each algorithm's intermediate steps and the resultant interpretation. Each algorithm's section presents pseudocode and program flow diagrams. The chapter concludes with a discussion of the advantages and

0-8493-0052-5/00/$0.00+$.50

shortcomings of each method, as well as future research directions.

1.2 Background

Knowledge of a protein's *folding* – that is, the sequence-determined three-dimensional structure – is valuable to biologists. A protein's structure provides great insight into the mechanisms by which a protein acts, and knowing these mechanisms helps increase our basic understanding of the underlying biology. Structural knowledge is increasingly important in disease treatment, and has led to the creation of catalysts with industrial uses. No existing computer algorithm can accurately map sequence to 3D structure; however, several experimental "wet lab" techniques exist for determining macromolecular structure. The most commonly used method, employed for about 80% of structures currently known, is *x-ray crystallography*. This time-consuming and resource-intensive process uses the diffraction pattern of x-rays off a crystallized matrix of protein molecules to produce an electron density map. This electron density map is a three-dimensional "picture" of the electron clouds surrounding each protein atom. Producing a protein structure is then a matter of identifying the location of each of the protein's atoms in this 3D picture.

Density map interpretation – traditionally performed by a crystallographer – is time-consuming and, in noisy or poor-quality maps, often error-prone. Recently, a number of research groups have looked into automatically interpreting electron density maps, using ideas from machine learning and computer vision. These methods have played a significant role in high-throughput structure determination, allowing novel protein structures to quickly be elucidated.

1.2.1 Protein structure

Proteins (also called *polypeptides*) are constructed from a set of building blocks called amino acids. Each of the twenty naturally-occurring amino acids consists of an amino group and a carboxylic acid group on one end, and a variable chain on the other. When forming a protein, adjacent amino groups and carboxylic acid groups condense to form a repeating backbone (or mainchain), while the variable regions become dangling sidechains. The atom at the interface between the sidechain and the backbone is known as the alpha carbon, or C_α for short (see Figure 1.1). The linear list of amino acids composing a protein is often referred to as the protein's primary structure (see Figure 1.2a).

A protein's secondary structure (see Figure 1.2b) refers to commonly occurring three-dimensional structural motifs taken by continuous segments in

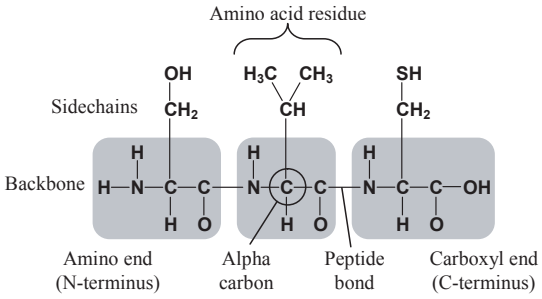

FIGURE 1.1: Proteins are constructed by joining chains of amino acids in peptide bonds. A chain of three amino acid residues is illustrated.

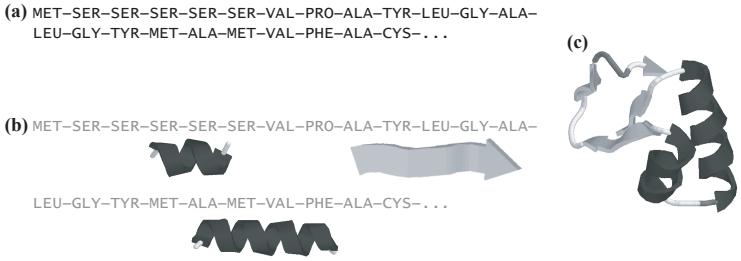

FIGURE 1.2: An illustration of: (a) a protein's *primary structure*, the linear amino-acid sequence of the protein, (b) a protein's *secondary structure*, which describes local structural motifs such as alpha helices and beta sheets, and (c) a protein's *tertiary structure*, the global three-dimensional conformation of the protein.

the protein. There are two such motifs: α-helices, in which the peptide chain folds in a corkscrew, and β-strands, where the chain stretches out linearly. In most proteins, several β-strands run parallel or antiparallel to one another. These regular structural motifs are connected by less-regular structures, called loops (or turns). A protein's secondary structure can be predicted somewhat accurately from its amino-acid sequence [1].

Finally, a protein's three-dimensional conformation – also called its tertiary structure (see Figure 1.2c) – is uniquely determined from its amino acid sequence (with some exceptions). No existing computer algorithm can accurately map sequence to tertiary structure; instead, we must rely on experimental techniques, primarily x-ray crystallography, to determine tertiary structure.

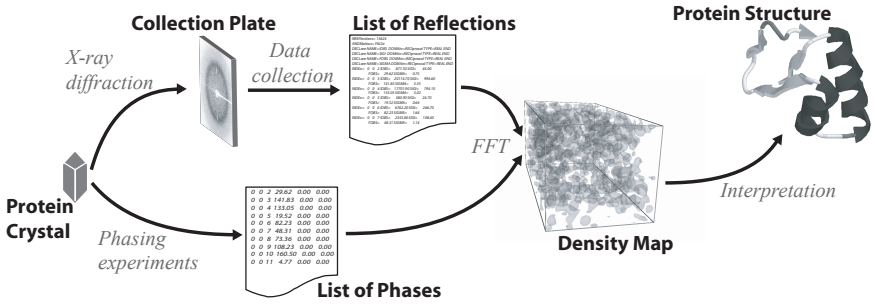

FIGURE 1.3: An overview of the crystallographic process.

1.2.2 X-ray crystallography

An overview of protein crystallography appears in Figure 1.3. Given a suitable target for structure determination, a crystallographer must produce and purify this protein in significant quantities. Then, for this particular protein, one must find a very specific setting of conditions (i.e., pH, solvent type, solvent concentration) under which protein crystals will form. Once a satisfactory crystal forms, it is placed in front of an x-ray source. Here, this crystal diffracts a beam of x-rays, producing a pattern of spots on a collector. These spots – also known as *reflections* or *structure factors* – represent the Fourier-transformed electron density map. Unfortunately, the experiment can only measure the intensities of these (complex-valued) structure factors; the phases are lost.

Determining these missing phases is known as the *phase problem* in crystallography, and can be solved to a reasonable approximation using computational or experimental techniques [2]. Only after estimating the phases can one compute the electron-density map (which we will refer to as a "density map" or simply "map" for short).

The electron density map is defined on a 3D lattice of points covering the *unit cell*, the 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.

Figure 1.4 shows a sample fragment from an electron density map, and the corresponding interpretation of that fragment. The amino-acid (primary) sequence of the protein is typically known by the crystallographer before interpreting the map. In a high-quality map, every single (non-hydrogen) atom in the protein can be placed in the map (Figure 1.4b). In a poor-quality map

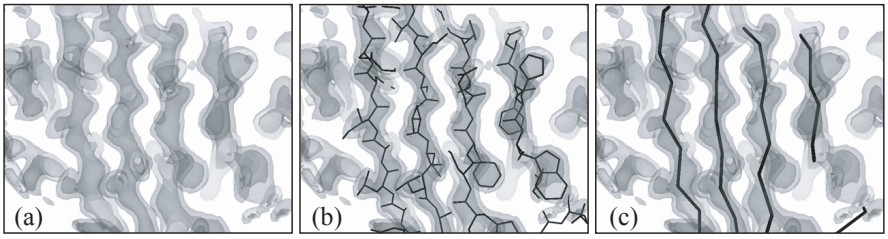

FIGURE 1.4: An overview of electron density map interpretation. Given the amino-acid sequence of the protein and a density map (a), the crystallographer's goal is to find the positions of all the protein's atoms (b). Alternatively, a *backbone trace* (c), represents each residue by its C_α location.

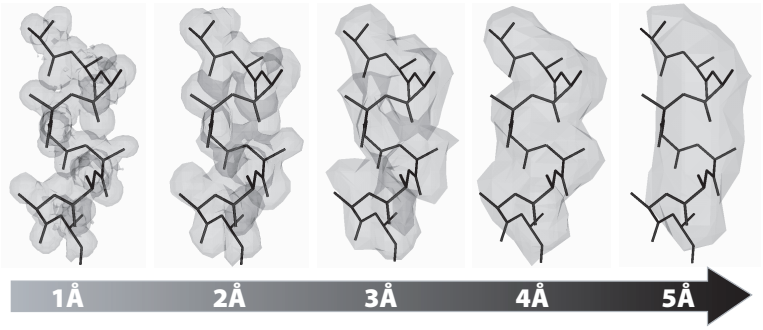

FIGURE 1.5: Electron density map quality at various resolutions. The "sticks" show the position of the atoms generating the map.

it may only be possible to determine the general locations of each residue. This is known as a C_α trace (Figure 1.4c). This information - though not as comprehensive as an all-atom model - is still valuable to biologists.

The quality of experimental maps as well as the sheer number of atoms in a protein makes interpretation difficult. Certain protein crystals produce a very narrow diffraction pattern, resulting in a poor-quality, "smoothed" density map. This is quantified as the *resolution* of a density map, a numeric value which refers to the minimum distance at which two peaks in the density map can be resolved. Figure 1.5 shows a simple density map at a variety of resolutions.

Experimentally determined phases are often very inaccurate, and make interpretation difficult as well. As the model is built, these phases are iteratively improved [3], producing a better quality map, which may require resolving large portions of the map. Figure 1.6 illustrates the effect poor phasing has on density-map quality. In addition, noise in diffraction-pattern data collection also introduces errors into the resulting density map.

Finally, the density map produced from x-ray crystallography is not an

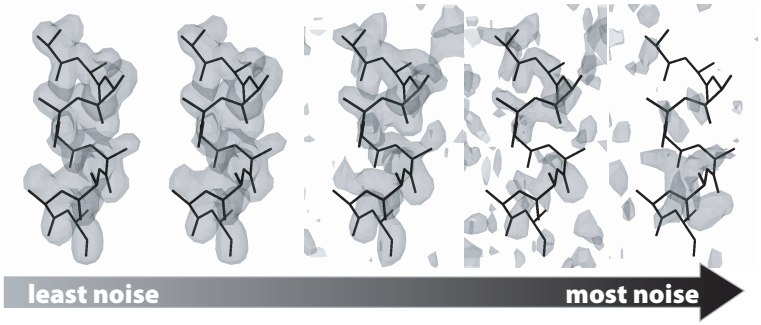

FIGURE 1.6: Electron density map quality as noise is added to the computed phases. The "sticks" show the position of the atoms generating the map.

"image" of a single molecule, but rather an average over an ensemble of all the molecules contained within a single crystal. Flexible regions in the protein are not visible at all, as they are averaged out.

For most proteins, this interpretation is done by an experienced crystallographer, who can, with high-quality data, fit about 100 residues per day. However, for poor-quality structures or structures with poor phasing, interpretation can be an order of magnitude slower. Consequently, map interpretation is the second-most time-consuming step in the crystallographic process (after crystal preparation).

A key question for computational methods for interpreting density maps is the following: how are candidate 3D models scored? Crystallographers typically use a model's *R-factor* (for *residual index*) to evaluate the quality of an interpretation. Formally, the R-factor is defined, given experimentally determined structure factors F_{obs} and model-determined structure factors F_{obs}, as:

$$R = \frac{\sum ||F_{obs}| - |F_{calc}||}{\sum |F_{obs}|} \tag{1.1}$$

The R-factor measures how well the proposed 3D structure *explains* the observed electron-density data. Crystallographers usually strive to get R-factors under 0.2 (or lower, depending on map resolution), while also building a physically-feasible (i.e. valid bond distances, torsion angles, etc.) protein model without adding too many water molecules. One can always reduce the R-factor by placing extra water molecules in the density map; these reductions are a result *overfitting* the model to the data, and don't correspond to a physically feasible interpretation.

Another commonly used measure is *free R-factor*, or R_{free}. Here, 5-10% of reflections are randomly held out as a test set before refinement. R_{free} is the R-factor for these held-aside reflections. Using R_{free} tends to avoid overfitting the protein's atoms to the reflection data.

1.2.3 Algorithmic background

Algorithms for automatically interpreting electron density maps draw heavily from the machine learning and statistics communities. These communities have developed powerful frameworks for modeling uncertainty, reasoning from prior examples, and statistically modeling data, all of which have been used by researchers in crystallography. This section briefly describes the statistal and machine learning methods used by the interpretation methods presented in this paper. This section is intended as a basic introduction to these topics. Interested readers should consult Russell and Norvig's text [4] or Mitchell's text [5] for a thorough overview.

1.2.3.1 Probabilistic models

A *model* here refers to a system that simulates a real-world event or process. Probabilistic models simulate uncertainty by producing different outcomes with different probabilities. In such models, the probabilities associated with certain events is generally not known, and instead has to be estimated from a *training set*, a set of previously solved problem instances. Using *maximum likelihood estimation*, the probability of a particular outcome is estimated as the frequency at which that outcome occurs in the training set.

The *unconditional* or *prior probability* of some outcome A is denoted $P(A)$. Assigning some value to this probability, say $P(A) = 0.3$, means that in the absence of any other information, the best assignment of probability of outcome A is 0.3. As an example, when flipping a (fair) coin, $P(\text{“}heads\text{”}) = 0.5$. In this section, we use "outcome" to mean the setting of some random variable; $P(X = x_i)$ is the probability that variable X takes value x_i. We will use the shorthand $P(x_i)$ to refer to this same value.

The *conditional* or *posterior probability* is used when other, previously unknown, information becomes available. If other information, B, relevant to event A is known, then the best assignment of probability to event A is given by the conditional $P(A|B)$. This reads "the probability of A, *given* B." If more evidence, C, is uncovered, then the best probability assignment is $P(A|B, C)$ (where "," denotes "and").

The *joint probability* of two or more events is the probability of both events occurring, and – for two events A and B – is denoted $P(A, B)$ and is read "the probability of A *and* B". Conditional and joint probabilities are related using the expression:

$$P(A, B) = P(A|B)P(B) = P(B|A)P(A) \tag{1.2}$$

This relation holds for any events A and B. Two events are *independent* if their joint probability is the same as the product of their unconditional probabilities, $P(A, B) = P(A)P(B)$. If A and B are independent we also have $P(A|B) = P(A)$, that is, knowing B tells us nothing about A.

One computes the *marginal probability* by taking the joint probability and summing out one or more variables. That is, given the joint distribution

$P(A, B, C)$, one computes the marginal distribution of A as:

$$P(A) = \sum_B \sum_C P(A, B, C) \tag{1.3}$$

Above, the sums are over all possible outcomes of events B and C. The marginal distribution is important because it provides information about the distribution of some variables (A above) in the full joint distribution, without requiring one to explicitly compute the (possibly intractable) full joint distribution.

Finally, *Bayes' rule* allows one to reverse the direction of a conditional:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1.4}$$

Bayes' rule is useful for computing a conditional $P(A|B)$ when direct estimation (using frequencies from a training set) is difficult, but when $P(B|A)$ can be estimated accurately. Often, one drops the denominator, and instead computes the *relative likelihood* of two outcomes, for example, $P(A = a_1|B)$ versus $P(A = a_2|B)$. If a_1 and a_2 are the only possible outcomes for A, then exact probabilities can be determined by normalization; there is no need to compute the prior $P(B)$.

1.2.3.2 Case-based reasoning

Broadly defined, *case-based reasoning* (CBR) attempts to solve a new problem by using solutions to similar past problems. Algorithms for case-based reasoning require a database of previously solved problem instances, and some distance function to calculate how "different" two problem instances are. There are two key aspects of CBR systems. First, learning in such systems is *lazy*: the models only generalize to unseen instances when presented with such a new instance. Second, they only use instances "close" to the unseen instance when categorizing it.

The most common CBR algorithm is *k-nearest neighbor* (kNN). In kNN, problem instances are feature vectors, that is, points in some n-dimensional space. The learning algorithm, when queried with a new problem instance $X = \langle x_i, \ldots, x_n \rangle$ for classification or regression, finds the k previously solved problem instances closest to the query in Euclidean space. That is, one chooses the examples minimizing the distance:

$$d(X||Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1.5}$$

Then, the k neighbors "vote" on the query instance's label: usually the majority class label (for classification) or average label (for regression) of the k neighbors is used. One variant of kNN weights each neighbor's vote by its similarity to the query. Another variant learns weights for each dimension, to be used when computing the distance between two instances.

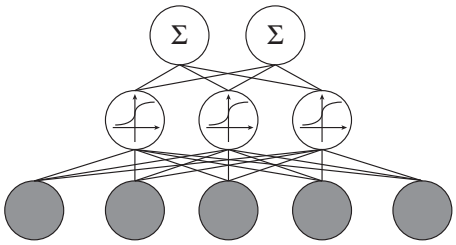

FIGURE 1.7: A multilayer, feed-forward neural network. The network consists of an input layer fully connected to a hidden layer of sigmoid units, fully connected to an output layer.

1.2.3.3 Neural networks

An *artificial neural network* (ANN) is a nonlinear function estimator used to approximate real or discrete target functions. Inspired by neurons in the brain, ANNs consist of a number of *units* connected in a network. Each unit is connected with multiple *inputs* and a single *output*. A given unit's output is some function of the weighted sum of the inputs. This function is know as the unit's *activation function*. For a *perceptron* – a simple, one-layer network – this function is usually a step function.

More commonly, the network structure has multiple, feed-forward layers, as shown in Figure 1.7. This network consists of a *hidden layer* which fully connects the input and outputs. Learning the weights in the network requires a differentiable activation function; often one uses a sigmoidal function:

$$\sigma(y) = \frac{1}{1 + e^{-y}} \tag{1.6}$$

Above, y is the weighted sum of inputs, $y = \sum_{i=0}^{N} w_i x_i$.

The *backpropagation* algorithm learns the weights for a multilayer network, given a network structure. Backpropagation uses gradient descent over the network weight space to minimize the squared error between *computed* output values and *desired* output values over some training set. The goal of backpropagation is to find some point in weight space – that is, some setting of all the weights in the network – that (locally) minimizes this squared error.

1.2.4 Approaches to automatic density map interpretation

A number of research groups have investigated automating the interpretation of electron-density maps. This section presents a high-level overview of some of these methods, while the remainder of this chapter takes an in-depth look at four of these methods, describing algorithmically how they have approached this problem.

By far the most commonly used method is ARP/WARP [6, 7, 8]. This atom-based method heuristically places atoms in the map, connects them, and

refines their positions. To handle poor phasing, ARP/WARP uses an iterative algorithm, consisting of alternating phases in which (a) a model is built from a density map, and (b) the density map's phases are improved using the constructed model. This algorithm is widely used, but has one drawback: fairly high resolution data, about 2.3Å or better, is needed. Given this high-resolution data, the method is extremely accurate; however, many protein crystals fail to diffract to this extent.

Several approaches represent the density map in an alternate form, in the process making the map more easily interpretable for both manual and automated approaches. One of the earliest such methods, *skeletonization*, was proposed for use in protein crystallography by Greer's BONES algorithm [9]. Skeletonization, similar to the medial-axis transformation in computer vision, gradually thins the density map until it is a narrow ribbon approximately tracing the protein's backbone and sidechains. More recent work by Leherte *et al.* [10], represents the density map as an acyclic graph: a minimum spanning tree connecting all the critical points (points where the gradient of the density is 0) in the electron density map. This representation accurately approximates the backbone with 3Å or better data, and separates protein from solvent up to 5Å resolution.

Cowtan's FFFEAR efficiently locates rigid templates in the density map [11]. It uses *fast Fourier transforms* (FFTs) to quickly match a learned template over all locations in a density map. Cowtan provides evidence showing it locates alpha helices in poorly-phased 8Å maps. Unfortunately, the technique is limited in that in can only locate large rigid templates (e.g. those corresponding to secondary-structure elements). One must trace loops and map the fit to the sequence manually. However, a number of methods use FFFEAR as a template-matching subroutine in a larger interpretation algorithm.

X-AUTOFIT, part of the QUANTA [12] package, uses a gradient refinement algorithm to place and refine the protein's backbone. Their refinement takes into account the density map as well as bond geometry constraints. They report successful application of the method at resolutions ranging from 0.8 to 3.0Å.

Terwilliger's RESOLVE contains an automated model-building routine [13, 14] uses a hierarchical procedure in which helices and strands are located and fitted, then are extended in an iterative fashion, using a library of tripeptides. Finally, RESOLVE applies a greedy fragment-merging routine to overlapping extended fragments. The approach was able to place approximately 50% of the protein chain in a 3.5Å resolution density map.

Levitt's MAID [15] approaches map interpretation "as a human would," by first finding the major secondary structures, alpha helices and beta sheets, connecting the secondary-structure elements, and mapping this fit to the provided sequence. MAID reports success on density maps at around 2.8Å resolution.

Ioerger's TEXTAL [16, 17, 18] attempts to interpret poor-resolution (2.2 to 3.0Å) density maps using ideas from pattern recognition. Ioerger constructs

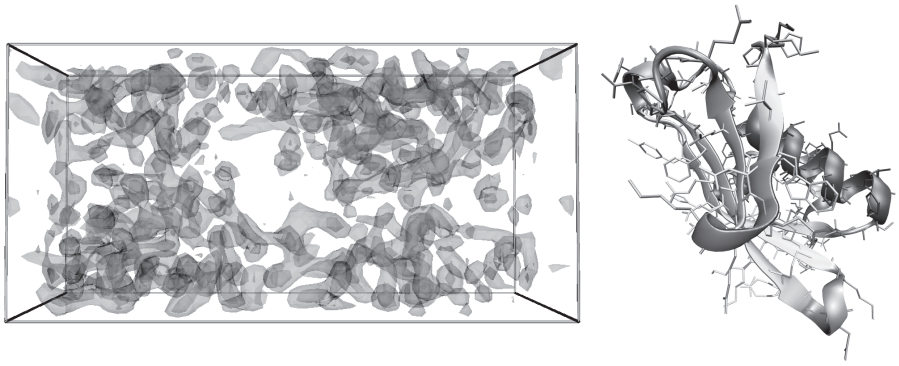

FIGURE 1.8: A 3.5Å resolution electron density map – containing two copies of a 95-amino-acid protein – and its crystallographer-determined solution. This map, at various resolutions, will be used as a running example throughout the section.

a set of 15 rotation-invariant density features. Using these features at several radii, a subroutine, CAPRA, trains a neural network to identify C_α atoms. TEXTAL then identifies sidechains 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.

Finally, ACMI takes a probabilistic approach to density map interpretation [19]. Residues of the protein are modeled as nodes in a graph, while edges model pairwise structural interactions arising from chemistry. An efficient inference algorithm determines the most probable backbone trace conditioned on these interactions. ACMI finds accurate backbone traces in well-phased 3.0 to 4.0Å density maps.

The rest of this chapter further describes four of these methods – ARP/WARP, RESOLVE, TEXTAL, and ACMI – in detail. Each section presents a method, describing strengths and weaknesses. High-level pseudocode clarifies important subroutines. Throughout the chapter, a small running example is employed to illustrate intermediate steps of the various algorithms. The example uses the density map of protein 1XMT, a 95-amino-acid protein with two symmetric copies in the unit cell.

The running example is not meant as a test of the algorithms (although a comparison appears in [19]), but rather as a real-world illustrative example. The example map is natively at 1.15Å resolution. Full native resolution is used for ARP/WARP; the map is downsampled to 2.5Å resolution for RESOLVE, and 3.5Å resolution for TEXTAL and ACMI. Maps are downsampled by smoothly diminishing the intensities of higher-resolution reflections. The resolution values chosen are – for this particular map – the worst-quality maps for which the respective algorithms produce an accurate trace. The 3.5Å density map and its crystallographer-determined solution appears in Figure 1.8.

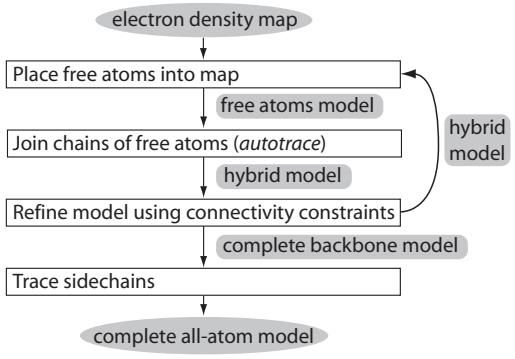

FIGURE 1.9: A flowchart of WARPNTRACE.

1.3 ARP/WARP

The ARP/WARP (automated refinement procedure) software suite is a crystallographic tool for the interpretation and refinement of electron density maps. ARP/WARP's WARPNTRACE procedure was the first automatic interpretation tool successfully used for protein models. Today, it remains one of the most used tools in the crystallographic community for 3D protein-image interpretation. ARP/WARP concentrates on the best placement of individual atoms in the map: no attempt is made to identify higher-order constructs like residues, helices, or strands. ARP/WARP's "atom-level" method requires high-quality data, however. In general, ARP/WARP requires maps at a resolution of 2.3A or higher to produce an accurate trace.

Figure 1.9 illustrates an overview of WARPNTRACE. WARPNTRACE begins by creating a *free atom model* – a model containing only unconnected atoms – to fill in the density map of the protein. It then connects some of these atoms using a heuristic, creating a *hybrid model*. This hybrid model consists of a partially-connected backbone, together with a set of unconstrained atoms. This hybrid model is refined, producing a map with improved phase estimates. The process iterates using this improved map. At each iteration, WARPNTRACE removes every connection, restarting with a free-atom model.

1.3.1 Free-atom placement

ARP/WARP contains a atom placement method based on ARP, an interpretation method for general molecular models. ARP randomly places unconnected atoms into the density map, producing a *free atom model*, illustrated in Figure 1.10a. To initialize the model, ARP begins with a small set of atoms in the density map. It slowly expands this model by looking for areas above a density threshold, at a bonding distance away from existing atoms. The

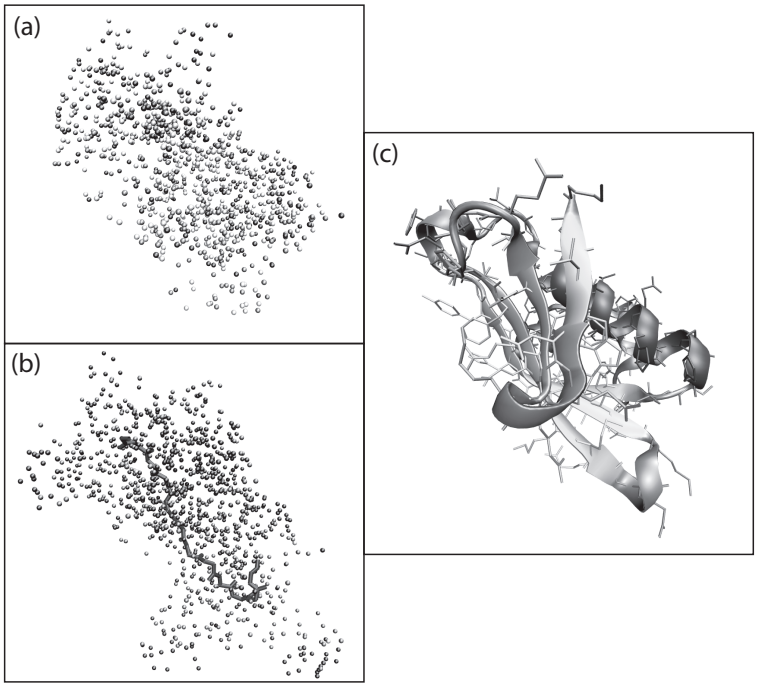

FIGURE 1.10: Intermediate steps in ARP/WARP's structure determination: (a) the free atom model, where waters are placed in the map, (b) the hybrid model, after some connections are determined, and (c) the final determined structure.

density threshold is slowly lowered until ARP places the desired number of free atoms.

For small molecules, ARP's next step is *refining* the free-atom model; that is, iteratively moving atoms to better explain the density map. Free-atom refinement ignores stereochemical information, and moves each atom independently to produce a complete structure. ARP's free-atom refinement, in addition to moving atoms, considers adding or removing atoms from the model. Multiple randomized initial structures are used to improve robustness. Further details are available from Perrakis *et al.* [7].

However, with molecules as large as proteins, free-atom refinement alone is insufficient to produce an accurate model. Performing free-atom refinement with tens of thousands of atoms leads to overfitting, producing a model that is not physically feasible. For determining protein structures, ARP/WARP makes use of connectivity information in its refinement, using free-atom placement as a starting point. The procedure WARPNTRACE adds connectivity information to the free atom model.

ALGORITHM 1.1: WARPNTRACE's model-building algorithm

Given: electron density map **M**, free-atom model **F**, sequence **seq**
Find: all-atom model

for $i = 1$ to $nIterations$ **do**
  $\mathbf{H} \leftarrow \mathbf{F}$ // *initialize hybrid model*
  CA_pairs $\leftarrow$ highest-scoring atom pairs 3.8 ± 0.5Å apart
  foreach $c_i \in$ **CA_pairs do**
    if (c_i does not match backbone template) **then**
      delete c_i from **CA_pairs**
    end
  end
  while a C_α chain of length ≥ 5 remains in **CA_pairs do**
    bestChain $\leftarrow$ longest fragment in DB overlapping **CA_pairs**
    remove *bestChain*'s atoms from **CA_pairs**, **H**
    add *bestChain* to **H**
  end
  $\mathbf{H}' \leftarrow refine(\mathbf{H})$
  $\mathbf{F} \leftarrow$ remove connections from hybrid model $\mathbf{H}'$
end
model $\leftarrow$ sidechainTrace($\mathbf{H}'$)

1.3.2 Main-chain tracing

Given a free-atom model of a protein, one can form a crude backbone trace by looking for pairs of free atoms the proper distance apart. WARPNTRACE formalizes this procedure, called *autotracing*, using a heuristic method. The method is outlined in Algorithm 1.1. WARPNTRACE assigns a score – based on density values – to each free atom. The highest scoring atom pairs 3.8 ± 0.5Å apart become candidate C_α's. The algorithm verifies candidate pairs by overlaying them with a peptide template. If the template matches the map, WARPNTRACE saves the candidate pair.

After computing a list of C_α pairs, WARPNTRACE constructs the backbone using a database of known backbone conformations (taken from solved protein structures). Given a chain of candidate C_α pairs, WARPNTRACE considers all backbone conformations in the database with matching C_α positions, ordered by length. The longest candidate backbone is then added to the model. The algorithm connects the corresponding free atoms, and removes these atoms from the free-atom pool. The process repeats as long as there remain candidate C_α chains at least 5 residues in length.

Autotracing produces a *hybrid model*, shown in Figure 1.10b. A hybrid model contains a set of connected chains together with a set of free atoms. Autotracing identifies some atom types and connectivity, which enables the use of some stereochemical information in refinement. Added restraints increase the number of observations available, and increase the probability of

producing a good model. The tracing is initially very conservative, with many free atoms remaining in the model. Adding too many restraints too early leads to overfitting the model.

Finally, a modified version of ARP refines this hybrid model. ARP uses the refined structure to improve the experimentally determined phases, making the map clearer to interpret. At each iteration of this "autotrace–refine–recompute phases" cycle, WARPNTRACE returns to a free-atom model, by removing previous traces. Since the map is better-phased, autotracing produces a more complete model. This, in turn, provides a better refinement, improving the phases further.

This cycle continues for a fixed number of iterations, or until a complete trace is available. Finally, WARPNTRACE adds on side-chains by identifying patterns of free atoms around C_α's. It aligns these free-atom patterns to the sequence to produce a complete model. Figure 1.10c illustrates the complete ARP/WARP-determined trace on the running example.

1.3.3 Discussion

ARP/WARP is the preferred method for automatically interpreting electron density maps, assuming sufficiently high-resolution data is available. Its placement of individual atoms, followed by atom-level refinement, produces an extremely accurate trace with no user action required in 2.3Å or better density maps. It is widely used by crystallographers to rapidly construct a protein model. Unfortunately, many protein crystals fail to produce maps of sufficient quality, and one must consider alternate methods.

1.4 RESOLVE

While ARP/WARP is extremely accurate with high-resolution data, many protein crystals fail to diffract to a sufficient level for accurate interpretation. In general, ARP/WARP requires individual atoms to be visible in the density map, which happens at about 2.3Å resolution or better. The next three methods – RESOLVE, TEXTAL, and ACMI – all aim to solve maps with $>$ 2.3Å resolution. All three methods take different approaches to the problem; however, all three – in contrast to ARP/WARP – consider higher-level constructs than atoms when building a protein model. This allows accurate interpretation even when individual atoms are not visible.

RESOLVE is a method developed by Terwilliger for automated model-building in poor-quality (around 3Å) electron density maps [13, 14]. Figure 1.11 outlines the complete hierarchical approach. RESOLVE's method hinges upon the construction of two model secondary structure fragments – a short α-helix and

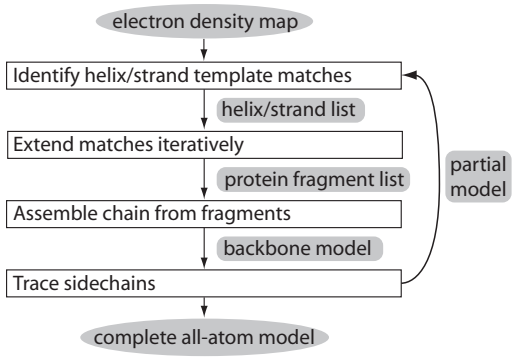

FIGURE 1.11: A flowchart of RESOLVE.

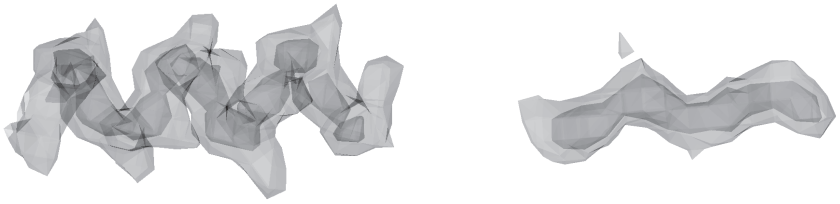

FIGURE 1.12: The averaged helix (left) and strand (right) fragment used in RESOLVE's initial matching step.

β-strand – for the initial matching. RESOLVE first searches over all of rotation and translation space for these fragments; after placing a small set of overlapping model fragments into the map, the algorithm considers a much larger template set as potential extensions. RESOLVE joins overlapping fragments and, finally, identifies sidechains corresponding to each C_α – conditioned on the input sequence – and places individual atoms into the model.

1.4.1 Secondary structure search

Given an electron density map, RESOLVE begins its interpretation by searching all translations and rotations in the map for a model 6-residue α-helix and a model 4-residue β-strand. RESOLVE constructs these fragments by aligning a collection of helices (or strands) from solved structures; it computes the electron density for each at 3Å resolution, and averages the density across all examples. The "average" models used by RESOLVE are illustrated in Figure 1.12.

Given these model fragments, RESOLVE considers placing them at each position in the map. At each position it considers all possible rotations (at a 30° or 40° discretization) of the fragment, and computes a standardized squared-density difference between the fragment's electron density and the

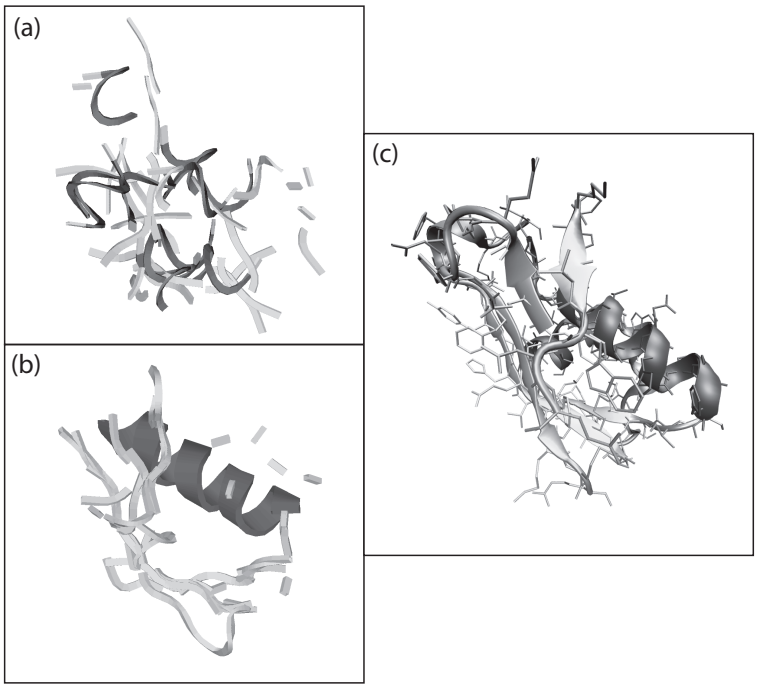

FIGURE 1.13: Intermediate steps in RESOLVE's structure determination: (a) locations in the map that match short helical/strand fragments (b) the same fragments after refinement and extension, and (c) the final determined structure.

map:

$$t(\vec{x}) = \sum_{\vec{y}} \epsilon_f(\vec{y}) \Big(\rho'_f(\vec{y}) - \frac{1}{\sigma_f(\vec{x})} \big[\rho(\vec{y} - \vec{x}) - \bar{\rho}(\vec{x}) \big] \Big) \tag{1.7}$$

Above, $\rho(\vec{x})$ is the map in which we are searching, $\rho'_f(\vec{x})$ is the *standardized* fragment electron density, $\epsilon_f(\vec{x})$ is a masking function that is nonzero only for points near the fragment, and $\bar{\rho}(\vec{x})$ and $\sigma_f(\vec{x})$ standardize the map in the masked region ϵ_f centered at $\vec{x}$:

$$\begin{aligned} \bar{\rho}(\vec{x}) =& \frac{\sum_{\vec{y}} \epsilon_f(\vec{y}) \rho(\vec{y} - \vec{x})}{\sum_{\vec{y}} \epsilon_f(\vec{y})} \\ \sigma_f^2(\vec{x}) =& \frac{\sum_{\vec{y}} \epsilon_f(\vec{y}) \big[\rho(\vec{y} - \vec{x}) - \bar{\rho}(\vec{x}) \big]^2}{\sum_{\vec{y}} \epsilon_f(\vec{y})} \end{aligned} \tag{1.8}$$

RESOLVE computes the matching function $t(\vec{x})$ quickly over the entire unit cell using FFFEAR's FFT-based convolution [11].

After matching the two model fragments using a coarse rotational step-size, the method generates a list of best-matching translations and orientations of

each fragment (shown in Figure 1.13a). Processing these matches in order, RESOLVE refines each fragment's position and rotation to maximize the real-space correlation coefficient (RSCC) between template and map:

$$RSCC(\rho_f, \rho) = \frac{\langle \rho_f \cdot \rho \rangle - \langle \rho_f \rangle \langle \rho \rangle}{\sqrt{\langle \rho_f^2 \rangle - \langle \rho_f \rangle^2} \sqrt{\langle \rho^2 \rangle - \langle \rho \rangle^2}} \tag{1.9}$$

Here, $\langle \rho \rangle$ indicates the map mean over a fragment mask. RESOLVE only considers refined matches with an RSCC above some threshold.

1.4.2 Iterative fragment extension

At this point, RESOLVE has a set of putative helix and strand locations in the density map. The next phase of the algorithm extends these using a much larger library of fragments, producing a model like that in Figure 1.13b. Specifically, RESOLVE makes use of four such libraries for fragment extension:

(a) 17 α-helices between 6 and 24 amino acids in length

(b) 17 β-strands between 4 and 9 amino acids in length

(c) 9,232 tripeptides containing backbone atoms only *for N-terminus extension*

(d) 4,869 tripeptides containing a partial backbone (the chain $C_\alpha - C - O$ with no terminal N) plus two full residues *for C-terminus extension*

RESOLVE learns these fragment libraries from a set of previously solved protein structures. It constructs the two tripeptide libraries by clustering a larger dataset of tripeptides.

1.4.2.1 α-helix/β-strand extension

For each potential model's placement in the map, RESOLVE consider extending it using each fragment in either set (a), if the model fragment is a helix, or set (b), if the model fragment is a strand. For each fragment, RESOLVE chooses the longest segment of the fragment such that every atom in the fragment has a density value above some threshold.

To facilitate comparison between these 17 segments of varying length (one for each fragment in the library), each matching segment is given a score $Q = \langle \rho \rangle \sqrt{N}$, with $\langle \rho \rangle$ the mean atom density, and N the number of atoms. The algorithm computes a Z-score:

$$Z = \frac{Q - \langle Q \rangle}{\sigma(Q)} \tag{1.10}$$

RESOLVE only considers segments with $Z > 0.5$. Notice there may be a large number of overlapping segments in the model at this point.

1.4.2.2 Loop extension using tripeptide libraries

For each segment in the model, RESOLVE attempts to extend the segment in both the N-terminal and C-terminal direction using the tripeptide library. RESOLVE tests each tripeptide in the library by superimposing the first residue of the tripeptide on the last residue of the current model segment. It then tests the top scoring "first-level" fragments for overlap (steric clashes) with the current model segment. For those with no overlap, a lookahead step considers this first-level extension as a starting point for a second extension. The score for each first-level extension is:

$$score_{\text{first-level}} = \langle \rho_{\text{first-level}} \rangle + \max_{\text{second-level}} \langle \rho_{\text{second-level}} \rangle \tag{1.11}$$

Above, $\langle \rho_{\text{first-level}} \rangle$ denotes the average map density at the atoms of the first-level extension.

RESOLVE accepts the best first-level extension – taking the lookahead term into account – only if the average density is above some threshold density value. If the density is too low, and the algorithm rejects the best fragment, several "backup procedures" consider additional first-level fragments, or stepping back one step in the model segment. If these backup procedures fail, RESOLVE rejects further extensions.

1.4.3 Chain assembly

Given this set of candidate model segments, RESOLVE's next step is assembling a continuous chain. To do so, it uses an iterative method, illustrated in Algorithm 1.2. The outermost loop repeats until no more candidate segments remain. At each iteration, the algorithm chooses the top-scoring candidate segment not overlapping any others. It considers all other segments in the model as extensions: if at least two C_α's between the candidate and extension overlap, then RESOLVE accepts the extension. Finally, the extension becomes the current candidate chain.

1.4.4 Sidechain trace

RESOLVE's final step is, given a set of C_α positions in some density map, to identify the corresponding residue type, and to trace all the sidechain atoms [14]. This sidechain tracing is the first time that RESOLVE makes use of the analyzed protein's sequence. RESOLVE's sidechain tracing uses a probabilistic method, finding the most likely layout conditioned on the input sequence. RESOLVE's sidechain tracing procedure is outlined in Algorithm 1.3.

RESOLVE's sidechain tracing relies on a rotamer library. This library consists of a set of low-energy conformations – or *rotamers* – that characterizes each amino-acid type. RESOLVE builds a rotamer library from the sidechains in 574 protein structures. Clustering produces 503 different low-energy side-

ALGORITHM 1.2: RESOLVE's chain-assembly algorithm

Given: electron density map **M**, set of high scoring fragments **F**
Find: putative backbone trace $\mathbf{X} = \{\vec{x}_i\}$ including C_β positions

repeat
  $frag_{best} \leftarrow$ top scoring unused segment
  for each $frag_i \in \{\mathbf{F} \backslash frag_{best}\}$ **do**
    if $frag_i$ and $frag_{best}$ overlap at ≥ 2 C_α positions
    and extension does not cause steric clashes **then**
      extend $frag_{best}$ by $frag_i$
    end
  end
until no candidates remain

chain conformations. For each cluster member, the algorithm computes a density map; each cluster's representative map is the average of its members.

For each C_α, RESOLVE computes a probability distribution of the corresponding residue type. Probability computation begins by first finding the correlation coefficient (see Equation 1.9) between the map and each rotamer. For each rotamer j, the correlation coefficient at the kth C_α is given by cc_{jk}. A Z-score is computed, based on rotamer j's correlation at every other C_α:

$$Z_{jk}^{rot} = \frac{cc_{jk} - \langle cc_j \rangle}{\sigma_j} \tag{1.12}$$

The algorithm only keeps a *single best-matching rotamer* of each residue type. That is, for residue type i:

$$Z_{ik}^{res} = \max_{\text{fragment } j \text{ is of type } i} Z_{jk}^{rot} \tag{1.13}$$

RESOLVE uses a Bayesian approach to compute probability from the Z-score. Amino-acid distributions in the input sequence provide the *a priori* probability P_{0j} of residue type j. Given a set of correlation coefficients at some position, RESOLVE computes the probability that the residue type is i by taking the product of probabilities that *all other correlation coefficients were generated by chance*. It estimates this probability using the Z-score:

$$P(cc_{ik}) \propto exp(-(Z^r es_{ik})^2/2) \tag{1.14}$$

Substituting and simplifying, the probability of residue type i at position k is:

$$P_{ik} \leftarrow P_{i0} \cdot \frac{exp\big((Z_{ik}^{res})^2/2\big)}{\sum_l P_{l0} \cdot exp\big((Z_{lk}^{res})^2/2\big)} \tag{1.15}$$

Finally, given these probabilities, RESOLVE finds the alignment of sequence to structure that maximizes the product of probabilities. The final step is,

ALGORITHM 1.3: RESOLVE's sidechain-placement algorithm

Given: map **M**, backbone trace $\mathbf{X} = \{\vec{x}_i\}$ (including C_β's), sidechain library **F**, sequence *seq*
Find: all-atom protein model

for each sidechain $f_j \in \mathbf{F}$ **do**
  for each C_α $\vec{x}_k \in \mathbf{X}$ **do**
    $cc_{jk} \leftarrow RSCC(\mathbf{M}(\vec{x}_k), f_j))$ *// see Equation 1.9*
    $Z_{jk} \leftarrow (cc_{jk} - \langle cc_j \rangle)/\sigma_j$
  end
end

// Estimate probabilities P_{ik} that residue type i is at position k
for each residue type i **do**
  $P_{i0} \leftarrow$ *a priori* distribution of residue type i
  $Z_{ik} \leftarrow \max_{\text{fragment } j \text{ of type } i} Z_{jk}$
  for each alpha carbon $\vec{x}_k \in \mathbf{X}$ **do**
    $P_{ik} \leftarrow P_{i0} \cdot \frac{exp(Z_{ik}^2/2)}{\sum_l P_{l0} \cdot exp(Z_{lk}^2/2)}$
  end
end

// Align trace to sequence, place individual atoms
align *seq* to chains maximizing product of P_{ik}'s
if (good alignment exists) **then**
  place best-fit sidechain of alignment-determined type at each position
end

given an alignment-determined residue type at each position, placing the rotamer of the correct type with the highest correlation coefficient Z-score. RESOLVE's final computed structure on the running example, both backbone and sidechain, is illustrated in Figure 1.13c.

1.4.5 Discussion

Unlike ARP/WARP, RESOLVE uses higher-order constructs than individual atoms in tracing a protein's chain. Searching for helices and strands in the map lets RESOLVE produce accurate traces in poor-quality maps, in which individual atoms are not visible. This method is also widely used by crystallographers. RESOLVE has been successfully used to provide a full or partial interpretation at maps with as poor as 3.5Å resolution. Because the method is based on heuristics, when map quality gets worse, the heuristics fail and the interpretation is poor. Typically, the tripeptide extension is the first heuristic to fail, resulting in RESOLVE traces consisting of disconnected secondary structure elements. In poor maps, as well, RESOLVE is often unable to iden-

tify sidechain types. However, RESOLVE is able to successfully use background knowledge from structural biology in order to improve interpretation in poor-quality maps.

1.5 TEXTAL

TEXTAL – another method for density map interpretation – was developed by Ioerger *et al.* Much like RESOLVE, TEXTAL seeks to expand the limit of interpretable density maps to those with medium to low resolution (2 to 3Å). The heart of TEXTAL lies in its use of computer vision and machine learning techniques to match patterns of density in an unsolved map against a database of known residue patterns. Matching two regions of density, however, is a costly six-dimensional problem. To deal with this, TEXTAL uses of a set of rotationally invariant numerical features to describe regions of density. The electron density around each point in the map is converted to a vector of 19 features sampled at several radii that remain constant under rotations of the sphere. The vector consists of descriptors of density, moments of inertia, statistical variation, and geometry. Using rotationally invariant features allows for efficiency in determination – reducing the problem from 6D to 3D – and better generalization of patterns of residues.

TEXTAL's algorithm – outlined in Figure 1.14 – attempts to replicate the process by which crystallographers identify protein structures. The first step is to identify the backbone – the location of each C_α – of the protein. Tracing the backbone is done by CAPRA (C-Alpha Pattern Recognition Algorithm), which uses a neural network to estimate each subregion's distance to the closest C_α given the rotationally invariant features discussed above. CAPRA's putative C_α locations are then sent into the second part of the algorithm, LOOKUP, which identifies the sidechains corresponding to each C_α's. LOOKUP uses these same rotationally invariant features, but instead uses a *nearest neighbor* approach to find a small subset of the database that best matches the region of unknown density. LOOKUP rotationally aligns the best match to the unknown residue, and places the corresponding atoms into the map. Finally, TEXTAL cleans up its trace using a set of post-processing routines.

1.5.1 Feature extraction

The most important component of TEXTAL is its extraction of a set of numerical features from a region of density. These numerical features allow rapid identification of similar regions from different (solved) maps. A key aspect of TEXTAL's feature set is invariance to arbitrary rotations of the region's density. This eliminates the need for an expensive rotational search for each

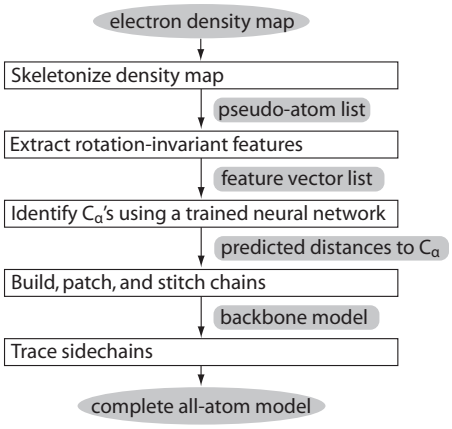

FIGURE 1.14: A flowchart of TEXTAL.

fragment; additionally, a discrete rotational search is likely to underestimate some match scores if the true rotation falls between rotational samples.

TEXTAL uses 76 such numerical features to describe a region of density in a map. These features include 19 rotationally invariant features, sampled at four different radii: 3,4,5 and 6Å. The use of multiple radii is critical for differentiation between side-chains: large residues often look similar at smaller radii but greatly differ at 6Å, while small amino acids may have no density in the outer radii and thus are only differentiated at small radii.

The 19 rotation-invariant features fall into four basic classes, shown in Table 1.1. The first class describes statistical properties of these neighborhoods of density, treating density values in the neighborhood as a probability distribution. These features include mean, standard deviation, skewness, and kurtosis, the last two of which provide descriptions of the lopsidedness and peakedness of the distribution of density values. The second class of features is really just a single feature: the distance from the center of mass to the center of the neighborhood.

A third class of descriptors includes moments of inertia (MOI), which provides six features describing how elliptical is the density distribution. Moments of inertia are calculated as the Eigenvectors of the inertia matrix I:

$$I = \sum_i \left(\rho_i \begin{vmatrix} y_i^2 + z_i^2 & -x_i y_i & -x_i z_i \\ -x_i y_i & x_i^2 + z_i^2 & -y_i z_i \\ -x_i z_i & -y_i z_i & x_i^2 + y_i^2 \end{vmatrix} \right)$$

Above, ρ_i is the density at point $\langle x_i, y_i, z_i \rangle$. As a rotation-invariant description, TEXTAL only considers the moments and the ratios between moments, not the axes themselves (the Eigenvectors of the inertia matrix).

The final class of features represent higher-level geometrical descriptors of the region. Three "spokes of maximal density" are extended from the center of the region, spaced $> 75°$ apart. These aim to approximate the direction

TABLE 1.1: The rotation invariant features used by TEXTAL

Class	Description	Quantity
Statistical Features of Density		4
	average, standard deviation, skewness, kurtosis	
Center of Mass		1
	distance from center of sphere to center of mass	
Moments of Inertia		6
	magnitude of primary, secondary, tertiary moments; ratios between these moments	
Spokes/Geometry of Density		8
	angles between three "spokes of maximal density" sum of angles, radial densities of each spoke, area of triangle formed by spokes	

ALGORITHM 1.4: TEXTAL's CAPRA subroutine for calculating the initial backbone trace

Given: electron density map $\mathbf{M}$
Find: putative backbone trace $\mathbf{X} = \{\vec{x}_i\}$

$\mathbf{M}' \leftarrow normalize(\mathbf{M})$
pseudoAtoms $\leftarrow skeletonize(\mathbf{M}')$
for $p_i \in$ **pseudoAtoms do**
  $\mathbf{F} \leftarrow$ rotation invariant-features in a neighborhood around p_i
  distance-to-$C_\alpha \leftarrow neuralNetwork(\mathbf{F})$
end
$\mathbf{X} \leftarrow$ construct chain using predicted distances-to-C_α

of the backbone N-terminus, the backbone C-terminus, and the sidechain. Rotation-invariant features derived from these spokes include the min, mid, max, and sum of the angles, the density sum along each spoke, and the area of the triangle formed by connecting the end points of the spokes.

1.5.2 Backbone tracing

CAPRA, a subroutine of TEXTAL, produces the initial C_α trace. CAPRA constructs a backbone chain using a feed-forward neural network. An overview of the process is illustrated in Algorithm 1.4.

In order to accurately compare maps, CAPRA begins by first normalizing density values in the map, ensures feature values from different maps are comparable. Next, CAPRA *skeletonizes* the map, creating a trace of *pseudo-atoms* that identifies the medial axis of some density map contour. Figure 1.15a illustrates this skeletonization. This trace is a very crude approximation of the backbone, and may traverse the side-chains or form multiple distinct chains.

A feed-forward neural network – a nonlinear function approximator used for both classification and regression – is trained to learn which pseudo-atoms

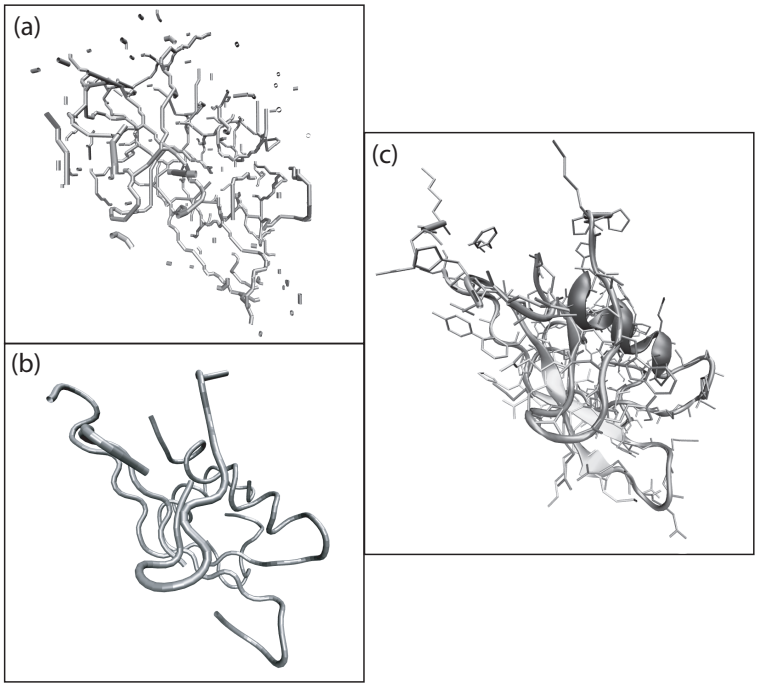

FIGURE 1.15: TEXTAL's intermediate steps: (a) the skeletonized density map, which crudely approximates the protein backbone, (b) a backbone trace, which TEXTAL builds by determining C_α's from the skeleton points, and (c) the final determined structure.

correspond to actual C_α's. Specifically, the network is trained on a set of previously solved maps to predict the *distance* of each pseudo-atom to the nearest C_α. The rotation invariant features are inputs to the network; a single output node estimates the distance to the closest C_α. A hidden layer of 20 sigmoidal units fully connects input and output layers.

Given a predicted distance-to-C_α for each pseudo-atom, CAPRA uses a greedy trace to find a linear chain linking C_α's together. Further post-modifications have been added to improve performance of CAPRA, such as refining chains and patching missing links. The output for CAPRA, on the sample map at 3.5Å resolution, is illustrated in Figure 1.15b.

1.5.3 Sidechain placement

After CAPRA returns its predicted backbone trace, TEXTAL must next identify the residue type associated with each C_α. This identification is performed by a subroutine LOOKUP. Algorithm 1.5 shows a pseudocode overview of LOOKUP. Essentially, the subroutine compares the density around each C_α to a database of solved maps to identify the residue type. LOOKUP uses

ALGORITHM 1.5: TEXTAL's LOOKUP subroutine for placing sidechains.

Given: electron density map $\mathbf{M}$, backbone trace $\mathbf{X} = \{x_i\}$
Find: all-atom protein model

for $\vec{x}_i \in \mathbf{X}$ **do**
  $\mathbf{F} \leftarrow$ rotation-invariant features in a neighborhood around x_i
  $\mathbf{N} \leftarrow k$ examples in DB minimizing weighted Euclidean distance
  for $\vec{n}_i \in \mathbf{N}$ **do**
    $\vec{n}'_i \leftarrow$ optimal superposition of $\vec{n}_i$ into map at $\vec{x}_i$
    $score_i \leftarrow RSCC(n_i, \mathbf{M}(\vec{x}_i))$ *// see Equation 1.9*
  end
  Choose n'_i maximizing $score_i$
  Add individual atoms of n'_i to model
end

TEXTAL's rotation invariant features, and builds a database of feature vectors corresponding to C_α's in solved maps. To determine the residue type of an unknown region of density, LOOKUP finds the *nearest neighbors* in the database, using weighted Euclidian distance:

$$D(\rho_1||\rho_2) = \Big[\sum_i \lambda_i \cdot \big(F_i(\rho_1) - F_i(\rho_2)\big)^2\Big]^{1/2} \tag{1.16}$$

Above, F_i refers to the ith feature in the vector, while ρ_1 and ρ_2 are two regions of density. Feature weights λ_i are optimized to maximize similarity between matching regions and minimize between non-matching regions, where ground truth is the optimally-aligned $RSCC$ (Equation 1.9). TEXTAL sets weights using the SLIDER algorithm [20] which considers features pairwise to determine a good set of weights.

Since information is lost when representing a region as a rotation invariant feature vector, the nearest neighbor in the database does not always correspond to the best-matching region. Therefore, LOOKUP keeps the top k regions – those with the lowest Euclidean distance – and considers these for a more time-consuming $RSCC$ computation (see Equation 1.9). Ideally, LOOKUP wants to find the rotation and translation of each of the k regions to maximize this correlation coefficient. It quickly approximates this optimal rotation and translation by aligning the moments of inertia between the template density region ρ_1 and target density region ρ_2. LOOKUP computes the real-space correlation at each alignment, and selects the highest-correlated candidate. Finally, LOOKUP retrieves the translated and rotated coordinated atoms of the top-scoring candidate and places them in the model.

1.5.4 Post-processing routines

Since each residue's atoms are copied from previous examples and glued together in the density map, the model produced by LOOKUP may contain some physically infeasible bond lengths, bond angles, or $\phi - \psi$ torsion angles. TEXTAL's final step is improving the model using a few simple post-processing heuristics. First, LOOKUP often reverses the backbone direction of a residue; TEXTAL's post-processing makes sure that all chains are oriented in a consistent direction. Refinement, like that of ARP/WARP, corrects improper bond lengths and bond angles, iteratively moving individual atoms to fit the density map better. Finally, TEXTAL takes into account the target protein's sequence to fix mismatched residues.

TEXTAL makes use of a provided sequence by aligning the map-determined model sequence to the provided input sequence, using a Smith-Waterman dynamic-programming alignment. If there is agreement between the sequences above some threshold, then a second LOOKUP pass corrects residues where the alignment disagrees. In this second pass, LOOKUP is restricted to only consider residues of the type indicated by the sequence alignment. Like RESOLVE, TEXTAL's end result is a complete all-atom protein model, illustrated for our example map in Figure 1.15c.

1.5.5 Discussion

TEXTAL – like RESOLVE – uses higher-order constructs than atoms in order to successfully solve low-quality maps. In practice, TEXTAL works well on maps at around 3Å resolution. TEXTAL's key contribution is the use of rotation-invariant features to recognize patterns in the map. This feature representation allows accurate C_α identification using a neural network; it also does well at classification of amino-acid type. TEXTAL tends to do better than RESOLVE at sidechain identification due to this feature set. One key shortcoming, however, is limiting the initial backbone trace to skeleton points in the density map. In very poor maps, skeletonization is inaccurate; this is in part responsible for TEXTAL's failure in maps worse than 3Å. However, TEXTAL has successfully employed previously solved structures and domain knowledge from structural biology to produce an accurate map interpretation.

1.6 ACMI

ACMI (automatic crystallographic map interpreter) is a recent method developed by DiMaio *et al.* for tracing protein backbones in poor-quality (3 to 4Å resolution) density maps [19]. ACMI takes a probabilistic approach to electron density map interpretation, finding the most likely layout of the backbone

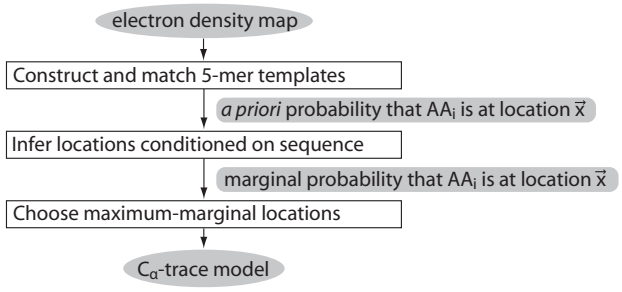

FIGURE 1.16: A flowchart of ACMI.

under some likelihood function. This likelihood function takes into account local matches to the density map as well as the global pairwise configuration of residues. The key difference between ACMI and the preceding methods is that ACMI represents each residue not as a single location, but rather as a complete *probability distribution* over the full electron density map. This property – not forcing 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.

Figure 1.16 shows a high-level overview of ACMI. The algorithm is comprised of two main components: a *local matching* component that probabilistically matches individual amino acids to the density map, and a *global constraint* component where the backbone chain is probabilistically refined from the local matches. Global refinement is based on physical laws governing the structure of proteins. ACMI's key is an efficient inference algorithm that determines the most probable *physically feasible* backbone trace in an electron density map.

1.6.1 Local matching

The *local matching* component of ACMI is provided the density map and the protein's amino acid sequence. It computes, for each residue in the protein, a probability distribution $P_i(\vec{w}_i)$ over all locations $\vec{w}_i$ in the unit cell. This probability distribution reflects the probability that residue i is at position and orientation $\vec{w}_i$ in the unit cell.

ACMI's local match – shown in Algorithm 1.6 – is based upon a 5-mer (that is, a polypeptide sequence five amino-acids in length) search. Using a set of previously solved structures, ACMI first constructs a basis set of structures describing the conformational space of each 5-mer in the protein. ACMI searches for each of these fragments in the map over all translations and rotations. This local search produces – for each residue i – an estimated probability distribution of that residue i's location and orientation in the unit cell, $P_i(\vec{w}_i)$.

ALGORITHM 1.6: ACMI's local-matching algorithm

Given: sequence **seq**, electron density map **M**
Find: probability distribution $P_i(\vec{w}_i)$ of each residue over map

for each residue $seq_i \in$ **seq do**
  N $\leftarrow$ instances of 5-mer $\langle seq_{i-2} \ldots seq_{i+2} \rangle$ from PDB
  C $\leftarrow$ cluster **N**, extract centroids
  $\lambda_i \leftarrow$ cluster weights
  for each cluster centroid $c_j \in$ **C do**
    $t(\vec{w}_i) \leftarrow$ Perform 6D search for c_j over density map
    Use a tuning set to convert $t(\vec{w}_i)$ to probabilities $P_{ij}(\vec{w}_i)$
  end
  $P_i(\vec{w}_i) \leftarrow \sum_{\text{clusters } j} \lambda_i P_{ij}(\vec{w}_i)$
end

1.6.1.1 Constructing a sequence-specific 5-mer basis set

Given some 5-amino-acid sequence, ACMI uses the Protein Data Bank (PDB), a repository of solved protein structures, to find all the instances of this sequence. If there are fewer than fifty such instances, ACMI considers "near-neighbors" (using the PAM-120 score, a measure of amino-acid similarity) until at least fifty distinct conformations are available. It uses these structures to represent the *conformational space* of the given 5-mer.

Searching for all these conformations in the electron density map is wasteful, because many are redundant, so ACMI clusters the structures into a smaller number of distinct conformations, representing each cluster with a single "central" instance (or *centroid*) of that cluster and a numeric weight. ACMI stores non-centroid instances from each cluster as well for tuning. Clustering uses the all-atom root mean squared deviation (RMSd) as a distance metric. ACMI can perform this clustering process in advance for all 3.2×10^6 possible 5-mers.

1.6.1.2 Searching for 5-mer centroid fragments

Given a protein sequence, ACMI considers the 5-mer sequence centered at each amino-acid. It extracts the fragments which represent the conformational space of that sequence, which it learns from the clustering process, and searches the density map for these fragments. Figure 1.17 illustrates this process graphically. Given these fragments and a resolution limit, ACMI builds an expected density map for each fragment. Then, at each map location, ACMI computes the standardized mean squared electron density difference $t(\vec{x})$ between the map and the fragment. Notice that this $t(\vec{x})$ is the same density map similarity function employed by RESOLVE, shown in Equation 1.7.

ACMI's fragment search is a 6D search: it considers every rotation of a fragment at every location in the map. Using FFFEAR's FFT-based convolution,

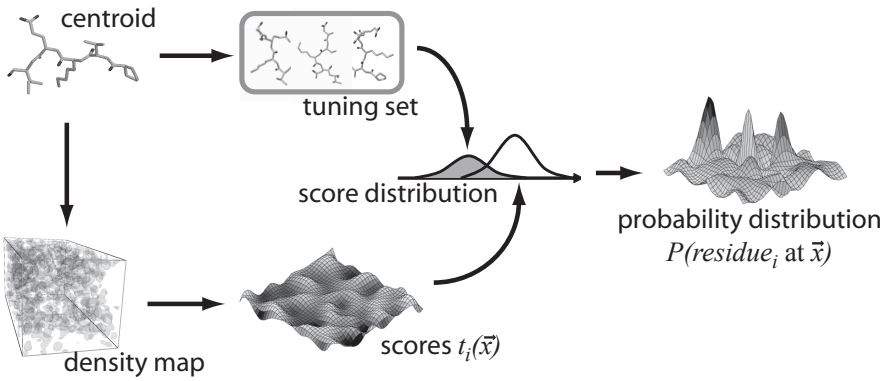

FIGURE 1.17: An overview of the 5-mer template matching process. Given a cluster of 5-mers for residue i, ACMI performs a 6D search for the fragment in the density map. ACMI also matches the fragment to a tuning set of known structures, using Bayes' rule to determine a probability distribution from the match scores $t(\vec{w})$.

one can compute this function very efficiently [11]. ACMI then stores – at each position – the best-matching 5-mer fragment and corresponding rotation.

The electron density difference function $t(\vec{x})$ is a good measure of similarity between regions of density; however, it doesn't allow comparison of scores from different templates. ACMI uses each cluster's tuning set in order to convert squared density differences into a probability distribution over the unit cell. To compute probabilities, Bayes' rule is used:

$$P(\text{res. } i \text{ at } \vec{x}_i | t(\vec{x}_i)) = \frac{P(t(\vec{x}_i) | \text{res. } i \text{ is at } \vec{x}_i) \cdot P(\text{res. } i \text{ at } \vec{x}_i)}{P(t(\vec{x}_i))} \qquad (1.17)$$

ACMI computes terms on the right-hand side: the denominator, $P_i(t(\vec{x}_i))$ is the distribution of match scores over the (unsolved) map. The prior probability, $P_i(\text{res. } i \text{ at } \vec{x}_i)$, is simply a normalization term. ACMI drops this term and simply ensures that probabilities over the map are normalized to sum to the number of copies of the 5-mer in the map. However, the first term in the numerator - the distribution of scores when the 5-mer matches the map - is trickier to compute. ACMI estimates this term using a *tuning set* saved from an earlier step. This tuning set contains instances of a particular 5-mer conformation other than the centroid. Matching the centroid 5-mer structure's density to each tuneset structure's density gives an accurate estimate to this term. As shorthand, we will refer to this probability distribution simply as $P_i(\vec{x}_i)$ for the remainder of this section. Figure 1.18a plots this probability distribution for two residues in the example protein.

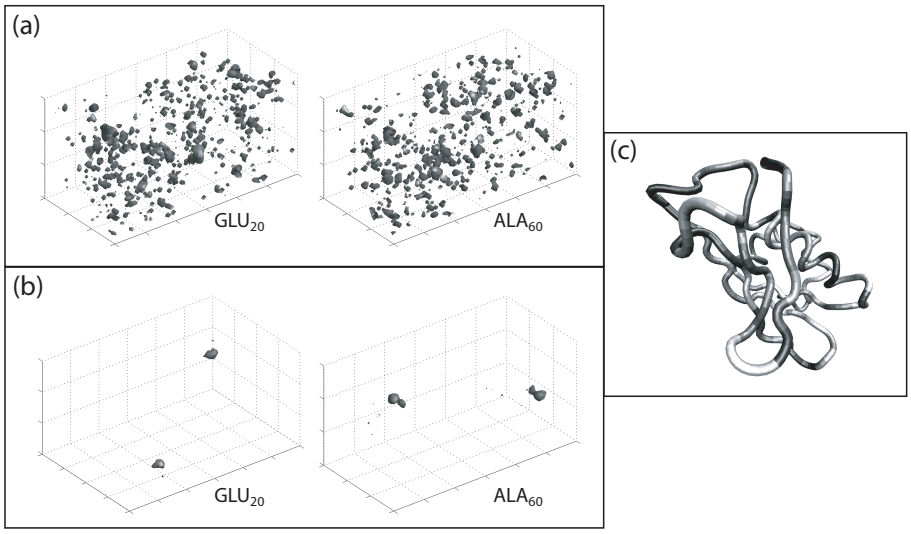

FIGURE 1.18: Intermediate steps in ACMI's structure determination: (a) the matching probability distributions $P_i(\vec{x}_i)$ on two residues, GLU_{20} and ALA_{60}, contoured at $p = 10^{-4}$, (b) the marginal probability distributions over the same two residues, and (c) the final (backbone-only) model, each residue shaded by likelihood (darker is more likely).

1.6.2 Global constraints

Given each residue's independent probability distribution over the unit cell, $P_i(\vec{x}_i)$, ACMI accounts for global constraints (enforced by *physical feasibility*) of a conformation by modeling the protein using a *pairwise Markov field*. A pairwise Markov field defines the joint probability distribution over a set of variables as a product of *potential functions* defined over vertices and edges in an undirected graph (see Figure 1.19).

1.6.2.1 Markov field model

Formally, the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ 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 $\vec{w}_i \in \mathbf{W}$, and the graph is conditioned on a set of observation variables $\mathbf{M}$. Each vertex has a corresponding *observation potential* $\psi_i(\vec{w}_i, \mathbf{M})$, and each edge is associated with an *edge potential* $\psi_{ij}(\vec{w}_i, \vec{w}_j)$. Then, the probability of a complete trace $\mathbf{W}$ is:

$$P(\mathbf{W}|\mathbf{M}) \propto \prod_{(i,j)\in\mathcal{E}} \psi_{ij}(\vec{w}_i, \vec{w}_j) \times \prod_{i\in\mathcal{V}} \psi_i(\vec{w}_i, \mathbf{M}) \tag{1.18}$$

Probabilistic inference finds the labels $\mathbf{W} = \{\vec{w}_i\}$ maximizing this probability, given some $\mathbf{M}$.

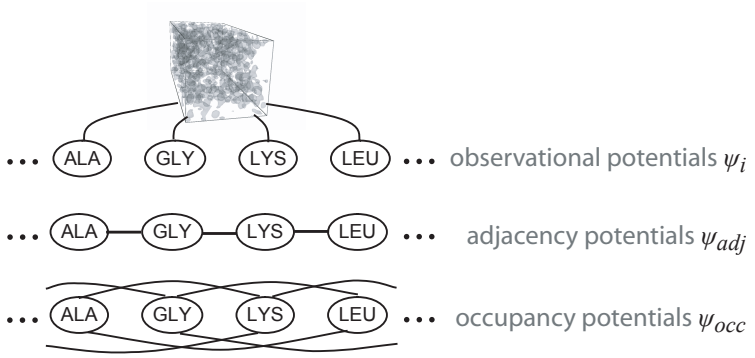

FIGURE 1.19: ACMI's protein backbone model. The joint probability of a conformation of residues is the product of an observation potential psi_i at each node, (b) an adjacency potential between adjacent residues, and (c) an occupancy potential between all pairs of non-adjacent residues..

To encode a protein in a Markov field model, ACMI constructs an undirected graph where each node i corresponds to an amino-acid residue in the protein. The label $\vec{w}_i = \langle \vec{x}_i, \vec{q}_i \rangle$ for each amino-acid consists of seven terms: the 3D Cartesian coordinates $\vec{x}_i$ of the residue's alpha Carbon (C_α), and four "orientation" parameters $\vec{q}_i$ (three rotational parameters plus a "bend" angle). The observation potential $\psi_i(w_i, \mathbf{M})$ associated with each residue is the previously computed probability distribution $P_i(\vec{x}_i)$, with – at each position – all the probability mass stored in the orientation of the best match.

Edges in the graph enforce global constraints in a pairwise manner. The graph is fully-connected (that is, every pair of residues is connected by an edge); DiMaio breaks the potential functions ψ_{ij} associated with each edge into two types. *Adjacency potentials* ψ_{adj} associated with edges between adjacent residues ensure that these neighboring C_α's maintain the proper 3.8 spacing, and C_α triples maintain the proper bend angle. *Occupancy potentials* ψ_{occ} on all other edges prevent two residues from occupying the same region in three-dimensional space. Thus, the full joint probability of a trace, given a map is:

$$\begin{aligned} P(\mathbf{W}|\mathbf{M}) \propto & \prod_{(\vec{w}_i,\vec{w}_j)\in\mathbf{W},|i-j|=1} \psi_{adj}(\vec{w}_i,\vec{w}_j) \\ & \times \prod_{(\vec{w}_i,\vec{w}_j)\in\mathbf{W},|i-j|>1} \psi_{occ}(\vec{w}_i,\vec{w}_j) \\ & \times \prod_{\vec{w}_i\in\mathbf{W}} \psi_i(\vec{w}_i,\mathbf{M}) \end{aligned} \tag{1.19}$$

1.6.2.2 Adjacency potentials

Adjacency potentials, which connect every neighboring pair of residues, are the product of two constraining functions, a distance constraint function and a rotational constraint function:

$$\psi_{adj}(\vec{w}_i, \vec{w}_j) = p_x(||\vec{x}_i - \vec{x}_j||) \cdot p_\theta(\vec{w}_i, \vec{w}_j) \tag{1.20}$$

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 ($\sigma = 0.03$Å) around this ideal value, softened a bit for grid effects. ACMI defines the potential p_θ using an alternate parameterization of the angular parameters $\vec{q}_i$.

Specifically, ACMI represents these four degrees of freedom as two pairs of $\theta - \phi$ spherical coordinates: the most likely direction of the forward $(i+1)$ residue and the backward $(i-1)$ residue. When matching the 5-mer templates into the density map, at each location x_i, ACMI stores four values – θ_b, ϕ_b, θ_f, and ϕ_f - indicating the direction of both adjacent C_α in the rotated, best-matching 5-mer.

The angular constraint function p_θ is then – in each direction – a fixed-width Gaussian on a sphere, centered on this preferred orientation, (θ_b, ϕ_b) or (θ_f, ϕ_f).

1.6.2.3 Occupancy potentials

Occupancy potentials 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 3Å apart. It is in this structural potential function that ACMI deals with crystallographic symmetry, by ensuring that all crystallographic copies are at least 3Å apart:

$$\psi_{occ}(\vec{w}_i, \vec{w}_j) = \begin{cases} 1 \left(\min_{\substack{symmetric \\ transforms\ K}} ||x_i - K(x_j)|| \right) \geq 3.0\text{Å} \\ 0 \text{ otherwise} \end{cases} \tag{1.21}$$

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

1.6.2.4 Tracing the backbone

ACMI's backbone trace – shown in Algorithm 1.7 – is equivalent to finding the labels $\mathbf{W} = \{\vec{w}_i\}$ that maximize the probability of Equation 1.18. Since the graph over which the joint probability is defined contains loops, finding an exact solution is infeasible (dynamic programming can solve this in quadratic time for graphs with no loops). ACMI uses belief propagation (BP) to compute an approximation to the marginal probability for each residue i (that is, the full joint probability with all but one variable summed out). ACMI chooses the label for each residue that maximizes this marginal as the final trace.

ALGORITHM 1.7: ACMI's global-constraint algorithm

Given: individual residue probability distributions $P_i(\vec{w}_i)$
Find: *approximate* marginal probabilities $\hat{b}_i(\vec{w}_i)$

$\forall i$ initialize belief $\hat{b}_i^0(\vec{w}_i)$ to $P_i(\vec{w}_i)$
repeat
  for each residue i **do**
    $\hat{b}_i(\vec{w}_i) \leftarrow P_i(\vec{w}_i)$
    for each residue $j \neq i$ **do**
      // *compute incoming message*
      if $|i-j| = 1$ **then**
        $m_{j\rightarrow i}^n(\vec{w}_i) \leftarrow \int_{\vec{w}_j} \psi_{adj}(\vec{w}_i, \vec{w}_j) \times \frac{\hat{b}_j}{m_{i\rightarrow j}^{n-1}}(\vec{w}_j) d\vec{w}_j$
      else
        $m_{j\rightarrow i}^n(\vec{w}_i) \leftarrow \int_{\vec{w}_j} \psi_{occ}(\vec{w}_i, \vec{w}_j) \times \hat{b}_j(\vec{w}_j) d\vec{w}_j$
      end
      // *aggregate messages*
      $\hat{b}_i(\vec{w}_i) \leftarrow \hat{b}_i(\vec{w}_i) \times m_{j\rightarrow i}^n(\vec{w}_i)$
    end
  end
until ($\hat{b}_i$'s converge)

Belief propagation is an inference algorithm - based on Pearl's polytree algorithm for [21] Bayesian networks - that computes marginal probabilities using a series of local messages. At each iteration, an amino acid computes an estimate of its marginal distribution (i.e., an estimate of the residue's location in the unit cell) as the product of that node's local evidence ψ_i and all incoming messages:

$$\hat{b}_i^n(\vec{w}_i) \propto \psi_i(\vec{w}_i, \mathbf{M}) \times \prod_{j \in \Gamma(i)} m_{j\rightarrow i}^n(\vec{w}_i) \tag{1.22}$$

Above, the belief $\hat{b}_i^n$ at iteration n is the best estimate of residue i's marginal distribution,

$$\hat{b}_i^n(\vec{w}_i) \approx \sum_{\vec{w}_0} \cdots \sum_{\vec{w}_{i-1}} \sum_{\vec{w}_{i+1}} \cdots \sum_{\vec{w}_N} P(\mathbf{W}, \mathbf{M}) \tag{1.23}$$

Message update rules determine each message:

$$m_{j\rightarrow i}^n(\vec{w}_i) \propto \int_{\vec{w}_j} \psi_{ij}(\vec{w}_i, \vec{w}_j) \times \psi_j(\vec{w}_j, \mathbf{M}) \times \prod_{k \in \Gamma(j)\setminus i} m_{k\rightarrow j}^{n-1}(x_j)\, d\vec{w}_j \tag{1.24}$$

Computing the message from j to i uses all the messages going into node j *except* the message from node i. When using BP in graphs with loops, 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 [22]. On the example protein, ACMI computes the marginal distributions shown in Figure 1.18b.

To represent belief and messages, ACMI uses a Fourier-series probability density estimate. That is, in the Cartesian coordinates $\vec{x}_i$, marginal distributions are represented as a set of 3-dimensional Fourier coefficients f_k, where, given an upper-frequency limit, (H, K, L):

$$b_i^n(\vec{x}_i) = \sum_{h,k,l=0}^{H,K,L} f_{hkl} \times e^{-2\pi i(\vec{x}_i \cdot \langle h,k,l \rangle)} \tag{1.25}$$

In *rotational parameters*, ACMI divides the unit cell into sections, and in each section stores a single orientation $\vec{q}_i$. These orientations correspond to the best-matching 5-mer orientation. More importantly, these stored orientations and are not updated by belief propagation: messages are independent of the rotational parameters $\vec{q}_i$.

To make this method tractable, ACMI approximates all the outgoing occupancy messages at a single node:

$$m_{j\to i}^n(\vec{w}_i) \propto \int_{\vec{w}_j} \psi_{occ}(\vec{w}_i, \vec{w}_j) \times \frac{b_j^n(\vec{w}_j) d\vec{w}_j}{m_{i\to j}^{n-1}(\vec{w}_j)} \tag{1.26}$$

The approximation drops the denominator above:

$$m_{j\to *}^n(\vec{w}_i) \propto \int_{\vec{w}_j} \psi_{occ}(\vec{w}_i, \vec{w}_j) \times b_j^n(\vec{w}_j) d\vec{w}_j \tag{1.27}$$

That is, ACMI computes occupancy messages using all incoming messages to node j *including* the message from node i. All outgoing occupancy messages from node j, then, use the same estimate. ACMI caches the product of all occupancy messages $\prod_i m_{i\to *}$. This reduces the running time of each iteration in a protein with n amino acids from $O(n^2)$ to $O(n)$.

Additionally, ACMI quickly computes all *occupancy* messages using FFTs:

$$\mathcal{F}\big[m_{j\to i}^n(\vec{w}_i)\big] = \mathcal{F}\big[\psi_{occ}(\vec{w}_i, \vec{w}_j)\big] \times \mathcal{F}\Big[\big(\prod m_{j\to i}^{n-1}(\vec{w}_i)\big)\Big] \tag{1.28}$$

This is possible only because the occupancy potential is a function of the *difference* of the labels on the connected nodes, $\psi_{occ}(\vec{w}_i, \vec{w}_j) = f(||\vec{x}_i - \vec{x}_j||)$.

Finally, although ACMI computes the complete marginal distribution at each residue, a crystallographer is only interested in a single trace. The backbone trace consists of the locations for each residue that maximize the marginal

probability:

$$\begin{aligned}\vec{w}_i^* &= \arg\max_{\vec{w}_i} \hat{b}_i(\vec{w}_i) \\ &= \arg\max_{\vec{w}_i} \psi_i(\vec{w}_i, \mathbf{M}) \times \prod_{j \in \Gamma(i)} m_{j \to i}^n(\vec{w}_i) \end{aligned} \tag{1.29}$$

ACMI's backbone trace on the example map is illustrated in Figure 1.18c. The backbone trace is shaded by confidence: since ACMI computes a complete probability distribution, it can return not only a putative trace, but also a likelihood associated with each amino acid. This likelihood provides the crystallographer with information about what areas in the trace are likely flawed; ACMI can also produce a high-likelihood partial trace suitable for phase improvement.

1.6.3 Discussion

ACMI's unique approach to density map interpretation allows for an accurate trace in extremely poor maps, including those in the 3 to 4Å resolution range. Unlike the other residue-based methods, ACMI is model-based. That is, it constructs a model *based on the protein's sequence*, then searches the map for that particular sequence. ACMI then returns the most likely interpretation of the map, given the model. This is in contrast to TEXTAL and RESOLVE which search for "some backbone" in the map, then align the sequence to this trace after the fact. This makes ACMI especially good at sidechain identification; even in extremely bad maps, ACMI correctly identifies a significant portion of residues. Unfortunately, ACMI also has several shortcomings. Requiring complete probability distributions for the location of each residue is especially time consuming; this has limited the applicability of the method so far. Additionally, in poor maps, belief propagation fails to converge to a solution, although in these cases a partial trace is usually obtained. By incorporating probabilistic reasoning with structural biology domain knowledge, ACMI has pushed the limit of interpretable maps even further.

1.7 Conclusion

A key step in determining protein structures is interpreting electron density maps. In this area, bioinformatics has played a key role. This chapter describes how four different algorithms have approached the problem of electron density map interpretation:

- The WARPNTRACE procedure in ARP/WARP was the first method developed for automatic density map interpretation. Today, it is still the most

widely used method by crystallographers. ARP/WARP uses an "atom-level" technique in which free atoms are first placed into the density map, free atoms are next linked into chains introducing constraints, and finally, the combined model is refined to better explain the map. The method iterates through these three phases, at each iteration using the partial model to improve the map. Because it works at the level of individual atoms, it requires 2.3Å or better map resolution.

- RESOLVE is a method that searches for higher-level constructs – amino acids and secondary structure elements – than ARP/WARP's atom-level method. Consequently, it produces accurate traces in poor-quality (around 3Å) electron density maps, unsolvable by ARP/WARP. It, too, is widely used by crystallographers. RESOLVE begins by matching short secondary-structure fragments to the maps, then uses a large fragment library to extend these matches. Finally, overlapping matches are merged in a greedy fashion, and sidechains are traced. Incorporating structural domain knowledge is key to this method's success.

- TEXTAL also accurately interprets medium to low resolution (2 to 3Å), using residue-level matching. It represents regions of density as a vector of rotation-invariant features. This alternate representation serves several purposes. It is used to learn a classifier to identify C_α's, and it is also used to recognize the residue type of each putative C_α through a database comparison. The classifier is also very well suited to identifying the residue type in a given region, making it more accurate than RESOLVE in this respect. Additionally, TEXTAL's rotation-invariant representation enables fast matching of regions of density. This allows TEXTAL to easily make use of previously solved structures in its map interpretation, providing accurate traces even in poor-quality maps.

- ACMI uses a probabilistic model to trace protein backbones in poor-quality (3 to 4Å resolution) density maps. It finds the most likely layout of the backbone under a likelihood function which takes into account local matches to the density map as well as the global pairwise configuration of residues. Unlike other methods, ACMI represents each residue not as a single location but as a probability distribution over the entire density map. Also unlike other approaches, it constructs a model based on the protein's sequence, and finds the most likely trace of that particular sequence in the density map. ACMI's probabilistic, model-based approach results in accurate tracing and sidechain identification at poor resolutions.

As each of these methods extended the limit of what resolution maps are automatically interpreted, they brought with them two things: first, a higher-level "basic construct" at which the algorithm searches, and second, better incorporation of structural domain knowledge. These two key aspects are

what enables interpretation in maps where individual atoms are not distinguishable, and in the future, what will extend the threshold of interpretation further.

As fewer and fewer fine details are visible in poor-resolution maps, algorithms must use a higher and higher level basic construct – that is, the template for which they search – in order to be robust against blurred details. ARP/WARP has had success when most atoms are visible in the map. If individual atoms are blurred out, the atom-level method will fail. However, a residue-based method like TEXTAL – which will locate amino acids assuming entire residues are not blurred – is robust enough to handle interpretation when atom details are unclear. Similarly, using secondary-structure elements allows even better interpretation. Probabilistic methods like ACMI take this still further: its interpretation considers a single flexible protein-sized element, and is robust enough to handle maps where entire residues are indistinguishable.

Another important feature of these methods is the increasing use of structural domain knowledge. In a way, this makes sense: the crystallographers task is inherently no different than that of the *ab initio* protein folding algorithm. A crystallographer simply has the assistance of a "picture" of the protein (the density map). All four methods use structural domain knowledge, by employing previous solved structures in model building. Primarily, these structures are used in the construction of a set of "representative fragments;" however, ACMI and TEXTAL also use previously solved structures to learn model parameters. Future methods will increasingly rely on such domain knowledge.

In the future, with the rising popularity of high-throughput structure determination, automated map interpretation methods will play a significant role. Improvements in laboratory techniques are producing density maps at a faster rate. In addition, experimental techniques such as cryo-electron microscopy are producing density maps that approach 5-6Å resolution, and are continually improving. Automatically determining structure from maps of this quality will be increasingly important. To further extend the limit of what maps are interpretable, automated interpretation techniques will need to use more domain knowledge, and consider searching not for a single residue or a single tripeptide, but rather entire proteins, probabilistically. Providing automated systems with increasing amounts of a crystallographer's "expertise" is key to future improvements in these methods.

Acknowledgements

This work supported by NLM grant 1R01 LM008796-01 and NLM Grant 1T15 LM007359-01. The authors would also like to thank George Phillips and Tom Ioerger for assistance in writing this chapter.

References

[1] B. Rost and C. Sander (1993). Prediction of protein secondary structure at better than 70% accuracy. *J Mol Biol*.

[2] G. Rhodes (2000). *Crystallography Made Crystal Clear*. Academic Press.

[3] J. Abrahams, R. De Graaff (1998). New developments in phase refinement. *Curr Opin Struct Biol*.

[4] S. Russell and P. Norvig (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

[5] T. Mitchell (1997). *Machine Learning*. McGraw-Hill.

[6] V. Lamzin and K. Wilson (1993). Automated refinement of protein models. *Acta Cryst*.

[7] A. Perrakis, T. Sixma, K. Wilson and V. Lamzin (1997). wARP: Improvement and extension of crystallographic phases by weighted averaging of multiple refined dummy atomic models. *Acta Cryst*.

[8] R. Morris, A. Perrakis and V. Lamzin (2002). ARP/wARP's model-building algorithms: the main chain. *Acta Cryst*.

[9] J. Greer (1974). Three-dimensional pattern recognition. *J Mol Biol*.

[10] L. Leherte, J. Glasgow, K. Baxter, E. Steeg and S. Fortier (1997). Analysis of three-dimensional protein images. *JAIR*.

[11] K. Cowtan (2001). Fast Fourier feature recognition. *Acta Cryst*.

[12] T. Oldfield (2001). A number of real-space torsion-angle refinement techniques for proteins, nucleic acids, ligands and solvent. *Acta Cryst*.

[13] T. Terwilliger (2002). Automated main-chain model-building by template-matching and iterative fragment extension. *Acta Cryst*.

[14] T. Terwilliger (2002). Automated side-chain model-building and sequence assignment by template-matching. *Acta Cryst*.

[15] D. Levitt (2001). A new software routine that automates the fitting of protein X-ray crystallographic electron density maps. *Acta Cryst*.

[16] T. Ioerger, T. Holton, J. Christopher and J. Sacchettini (1999). TEXTAL: A pattern recognition system for interpreting electron density maps. *Proc ISMB*.

0-8493-0052-5/00/$0.00+$.50

[17] T. Ioerger and J. Sacchettini (2002). Automatic modeling of protein backbones in electron density maps via prediction of C-alpha coordinates. *Acta Cryst.*

[18] K. Gopal, T. Romo, E. Mckee, K. Childs, L. Kanbi, R. Pai, J. Smith, J. Sacchettini and T. Ioerger (2005). TEXTAL: Automated crystallographic protein structure determination. *Proc. IAAI.*

[19] F. DiMaio, J. Shavlik and G. Phillips (2006). A probabilistic approach to protein backbone tracing in electron density maps. *Proc ISMB.*

[20] K. Gopal, T. Romo, J. Sacchettini and T. Ioerger (2004). Weighting features to recognize 3D patterns of electron density in X-ray protein crystallography. *Proc CSB.*

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

[22] K. Murphy, Y. Weiss, and M. Jordan (1999). Loopy belief propagation for approximate inference: An empirical study. *Proc. UAI.*