Using Pictorial Structures to Identify Proteins in X-ray Crystallographic Electron Density Maps

Frank DiMaio[1,2]
dimaio@cs.wisc.edu

Jude Shavlik[1,2]
shavlik@cs.wisc.edu

George N. Phillips, Jr.[3,1]
phillips@biochem.wisc.edu

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

Abstract

One of the most time-consuming steps in determining a protein's structure via x-ray crystallography is interpretation of the electron density map. This can be viewed as a computer-vision problem, since a density map is simply a three-dimensional image of a protein. However, due to the intractably large space of conformations the protein can adopt, building a protein model to match in the density map is extremely difficult. This paper describes the use of *pictorial structures* to build a flexible protein model from the protein's amino acid sequence. A pictorial structure is a way of representing an object as a collection of parts connected, pairwise, by deformable springs. Model parameters are learned from training data. Using an efficient algorithm to match the model to the density map, the most probable arrangement of the protein's atoms can be found in a reasonable running time. We test the algorithm is on two different tasks. The first is an amino-acid sidechain-refinement task, in which the location of the protein's backbone is approximately known. The algorithm places the remaining atoms into the region of density quite accurately, placing 72% of atoms within 1.0 Å of their actual location (as determined by a crystallographer). In the second task, a classification task, the algorithm is used to predict the type of amino acid contained in an unknown region of density. In this task, the algorithm is 61% accurate in discriminating between four different amino acids.

Introduction and Background

A fundamental problem in molecular biology involves the determination of a protein's shape. Known as the protein's *folding*, the protein sequence alone usually determines its shape. The issue is important because knowing a protein's structure provides great insight into the mechanisms in which that protein is involved. Knowledge of these mechanisms is needed for disease treatment or drug development. Additionally, knowing the structure of a protein allows biologists to get one step closer to the "holy grail" – a direct mapping from sequence to structure. In the current state of structural biology, no algorithm exists that accurately maps sequence to structure, and one is forced to use "wet" laboratory methods to elucidate the structure of proteins. The most common technique in use today for determining the structure of proteins is x-ray crystallography, a complex experimental technique that allows molecular-scale visualization. The process is quite time-consuming, and requires a number of steps before a protein's structure can be determined. With the large research effort recently put into high-throughput structural genomics [1], speeding up this tedious process is increasingly important. Our work looks to speed up the process of x-ray crystallography by automating interpretation of the electron density map, that is, taking the three-dimensional image of the protein and finding the corresponding atomic coordinates.

The process by which x-ray crystallography is used to determine a protein's structure is very complex. First, the protein needs to be produced in significant quantities, purified, and crystallized. The crystal must then be placed in an x-ray beam. The diffraction pattern of x-rays through the crystal is collected and processed, producing the protein's electron density map. This map is nothing more than a three-dimensional snapshot of the protein. This map details the actual structure of the protein. However, it is in such a large, unwieldy format, that it is all but useless. To make this data usable by biologists, some core information – the protein's atomic coordinates – needs to be extracted.

Thus, the final step in the crystallographic process is interpreting this map and building a molecular model of the protein. Once the crystals have been prepared, map interpretation is the most time-consuming step in the subsequent analysis. For most proteins this interpretation is performed manually, although a number of attempts have been made at automated density map interpretation [2,3,4,5,6]. The amino-acid sequence of the protein is known in advance. This gives the complete topology of the protein for which we are searching. The main difficulty in interpretation is the large number of possible conformations a protein can adopt, and the similarity of "features" in the map, where each atom appears as a Gaussian blob, with little distinction between atoms of different elements [7]. The fact that the protein's conformation may involve free rotation about any single bond leads to an extremely large conformation space. This makes building a model of the protein to match to the density map extremely difficult.

We describe a computational framework for building a flexible model of a protein, given the protein's sequence, and the electron density map. An overview of our algorithm is shown in Table 1. Pictorial structures [8] allow the representation of an object as a collection of parts, which are linked, pairwise, by deformable spring-like connections. Each connection defines the relationship between the two parts it connects. When building an atomic model, the connections correspond to covalent bonds. The relationships they define maintain the "bond invariants" (e.g., interatomic distance, bond angles), while allowing the "bond variable" (torsion angle) to freely vary. This model allows one to model arbitrary-sized protein fragments. A recent dynamic programming-based matching algorithm of Felzenszwalb and Huttenlocher (hereafter referred to as Felz-Hut) [9] allows pictorial structures to be quickly matched into a two-dimensional image. Their matching algorithm finds the globally optimal position and orientation of each part in the pictorial structure, by making some simplifying assumptions concerning independence of parts and connections. A simple "face" pictorial structure is shown in Figure 1.

In the pictorial structure model, the parts and (pairwise) connections form a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of parts, and an edge $e_{ij} \in \mathbf{E}$ connects neighboring parts v_i and

Table 1: A high level outline of our algorithm.

```
Given: Amino-acid sequence of protein seq, electron density map densityMap
Predict: Atomic coordinates of the protein in the density map

Algorithm:
    PS ← build pictorial structure from sequence seq
    bestMatch ← run Felz-Hut algo to find match for PS in densityMap

    while illegal_structure(bestMatch)
        bestMatch ← run soft-max algo to heuristically find non-optimal match

    return (bestMatch)
```
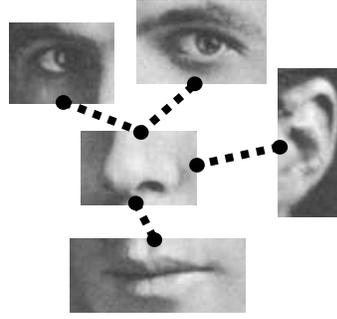
Figure 1: A sample pictorial structure for locating a face in an image. The collection of parts includes the pictures shown here. The dotted lines indicate statistical constraints between pairs of parts

v_j if there is an explicit dependency between the configuration of parts v_i and v_j. The configuration l_i of a part v_i consists of both the part's *position* and *orientation*. The Felz.-Hut. matching algorithm treats the graph as a Markov Random Field (MRF), where the probability of a part's configuration is conditionally independent of every other part in the model, given the configuration of all the part's neighbors in the graph. Each edge has associated with it a *deformation cost* $\mathbf{d_{ij}}(l_i, l_j)$, which, probabilistically, is the negative log likelihood of seeing a certain configuration of a pair of parts in a model. Similarly, each edge has an associated *matching cost* $\mathbf{match}_i(l_i, \mathbf{I})$, which is the negative log likelihood of the probability of seeing a part at configuration l_i in image $\mathbf{I}$.

The matching algorithm finds the globally most probable configuration of each part in the image. By assuming independence of parts for the matching function (i.e., ignoring occlusion) and the Markov Random Field assumption of independence of edges, the algorithm simply aims to find the configuration L of the parts in model Θ in image I, to minimize

$$P(L|I,\Theta) \propto P(I|L,\Theta)P(L|\Theta)$$

$$= \frac{1}{Z} e^{-\sum_i \mathbf{match}_i(l_i,\mathbf{I})} e^{-\sum_{(v_i,v_j)\in E} d_{ij}(l_i,l_j)}$$

Notice that, by monotonicity, this is equivalent to minimizing the following

$$\sum_i \mathbf{match}_i(l_i,\mathbf{I}) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i,l_j).$$

The Felz-Hut matching algorithm places several additional limitations on the topology of the Markov Random Field, and on the form that the deformation cost function $\mathbf{d_{ij}}$ must take. First, the MRF defined by the pictorial structure must be *tree structured*; cyclic constraints are not allowed. Secondly, the deformation cost function must take the following form, since the Felz-Hut quickly computes the minimization step in the DP algorithm as a distance transform in $\mathbf{T_{ji}^{-1}}$ space:

$$\mathbf{d_{ij}}(l_i,l_j) = \left\| \mathbf{T_{ij}}(l_i) - \mathbf{T_{ji}}(l_j) \right\|.$$

In the preceding, $\mathbf{T_{ij}}$ and $\mathbf{T_{ji}}$ are some arbitrary functions and $\|\cdot\|$ is some norm (i.e., distance metric). Under these conditions, the Felz-Hut matching algorithm can find the globally optimal configuration of parts in linear time (with respect to the number of configurations considered).

A number of different objects have been located into two-dimensional images using pictorial structures. They have been used in recognizing faces [8,10], general scenes, such as a waterfalls or mountains [11], cars, and bodies [9]. Felzenszwalb and Huttenlocher's paper also discusses two broad classes of connections: flexible revolute joints and prismatic joints. They provide

definitions for both classes of connection functions $\mathbf{d_{ij}}$ (and $\mathbf{T_{ij}}$). In building a general-purpose molecule recognition framework, we construct another class of connection, the *screw-joint*. Unlike previous work, this type of connection relates two objects in three-dimensional space. The screw-joint, based upon the rotation about a covalent bond, allows free rotation only around a single axis.

Building a Flexible Atomic Model

Our task is basically a problem in computer vision; given a three-dimensional image of a large molecule and the topology (i.e. amino-acid sequence) of that molecule, find a molecular model of the protein. That is, for each atom within the protein, determine the (Cartesian) coordinates of the center of that atom. The method we describe attempts to find these atomic coordinates by building a pictorial structure corresponding to the protein, then uses the Felz-Hut matching algorithm to find the most probable location of the atoms of the protein.

We first focus our efforts on the deformation cost of the molecule, that is, the prior probability of seeing a certain configuration of that molecule. Ideally, our deformation cost function would be the inverse of the atomic potential function, since molecule configurations with lower potential energy are more probable. However, the atomic potential function is a very complicated expression that takes into account both bonded and non-bonded interactions. Although this expression can be roughly approximated as a sum of pairwise potentials, it is impossible to do so in a manner that maintains the tree-structured MRF that the fast matching algorithm requires.

Alternatively, what we have done is to create a simplified atomic model by *ignoring non-bonded interactions*. We can then build a pictorial structure model such that each part in the model corresponds to an atom in the protein. Each connection, then, corresponds to a covalent bond. Knowing the protein's sequence in advance makes construction of the model trivial. When matching this model, the configuration we assign to each part consists of six parameters: three *translational*, and three *rotational*. The rotational parameters are Euler angles, which we refer to as (α, β, γ), where α is a rotation about the *z*-axis, β is a rotation about the *x*-axis, and γ is (another) rotation about the *z*-axis.

In this simplified model, we want to allow low-cost rotation about (most) bonds, while steeply penalizing any other rotation or translation. In order to do this, we introduce another broad class of connection, the *screw-joint*. Much like a rotating screw, the screw joint only allows rotation about a single axis. In order to simplify the cost function specification, we always consider this axis of rotation to be the *z*-axis. Since our matching algorithm considers all possible orientations of each part, this does not limit the model; instead, it requires all parts in the model to first be rotated into a *canonic orientation* in which the corresponding axis of rotation is the *z*-axis.

In the atomic model, defining the relationship between parts involves first considering a directed version of the MRF. Since the MRF is constrained by the fast matching algorithm to take a tree structure, an arbitrary root can be chosen. Then a directed graph corresponding to the MRF can be constructed such that every edge points up the tree, toward the root node. Each edge now concerns the relationship of a *parent* and a *child*. The deformation cost of each edge can be defined in terms of three parameters stored at each edge, $\bar{\mathbf{x}}_{\mathbf{ij}} = (x_{ij}, y_{ij}, z_{ij})^{\mathsf{T}}$. These three parameters define the optimal translation between parent and child, in *the coordinate system of*

4

the parent. The model learns these parameters using a technique that will be discussed later. Since the *z*-axis is axis of free rotation, we rotate the child such that its +*z* axis is coincident with $\bar{\mathbf{x}}_{ij}$ in the parent's coordinate frame. Figure 2 presents this graphically.

At this point, we have all we need to construct each edge's deformation cost function. Extending the work of Felzenszwalb and Huttenlocher to joints in three-dimensional space, we define an optimal relationship between configuration l_i and configuration l_j. The cost function is used for each edge is the distance between $(l_i - l_j)$, a vector between l_i and l_j in conformational space, and the optimal configuration l_{ij}. The distance measure we use is the 1-norm, weighted in each dimension, that is,

$$
\begin{aligned}
\mathbf{d}_{ij}(l_i, l_j) = {}& w_{ij}^{rotate} \; \left\| \left(\alpha_i - \alpha_j \right) \right\| \\
& + w_{ij}^{orient} \; \left(\left\| \left(\beta_i - \beta_j \right) + \operatorname{atan2}\left(\sqrt{x'^2 + y'^2}, -z' \right) \right\| \;\; + \;\; \left\| \left(\gamma_j - \gamma_i \right) - \pi/2 + \operatorname{atan2}(y', x') \right\| \right) \\
& + w_{ij}^{translate} \; \left(\left\| \left(x_i - x_j \right) - x' \right\| \;\; + \;\; \left\| \left(y_i - y_j \right) - y' \right\| \;\; + \;\; \left\| \left(z_i - z_j \right) - z' \right\| \right).
\end{aligned}
$$

In the preceding, (x', y', z') is the rotation of the bond parameters (x_{ij}, y_{ij}, z_{ij}) to world coordinates, i.e.,

$$
(x', y', z')^{\mathsf{T}} = \mathbf{R}_{\alpha_i, \beta_i, \gamma_i} (x_{ij}, y_{ij}, z_{ij})^{\mathsf{T}},
$$

where $\mathbf{R}_{\alpha_i, \beta_i, \gamma_i}$ is the rotation matrix corresponding to Euler angles $(\alpha_i, \beta_i, \gamma_i)$.

There are several important things to note about the previous expression. First, the complicated expressions in the β and γ terms are trigonometric relationships defining the optimal orientation of each child: pointing up the parent's bond. Second, in the above equation, w_{ij}^{rotate} is the cost of rotating about a bond, w_{ij}^{orient} is the cost of rotating around any other axis, and $w_{ij}^{translate}$ is the cost of translating in *x*, *y* or *z*. For the screw-joint model, we set the weights such that w_{ij}^{rotate} is low, while w_{ij}^{orient} and $w_{ij}^{translate}$ are high. This gives a high cost to any configuration with movements other than bond rotations. The definitions of $\mathbf{T}_{ij}(l_i)$ and $\mathbf{T}_{ji}(l_j)$ follow straightforwardly from this definition.

Now that the configuration cost of the model is specified, the match function needs to be constructed. This match function provides the probability of seeing a certain part at a specified location in the image. Empirical evidence suggests setting the cost of placing a part in the image to the Euclidean distance between a local neighborhood in the image and a 3x3x3 template. We learn the template during a model-training process (described later).

Figure 2: Showing the screw-joint connection between two parts in the model. In the directed version of the MRF, v_i is the parent of v_j. By definition, v_j is oriented such that its local *z*-axis is coincident with it's ideal bond orientation $\bar{\mathbf{x}}_{ij} = (x_{ij}, y_{ij}, z_{ij})^{\mathsf{T}}$ in v_i. The bond parameters $\bar{\mathbf{x}}_{ij}$ are learned by our algorithm.
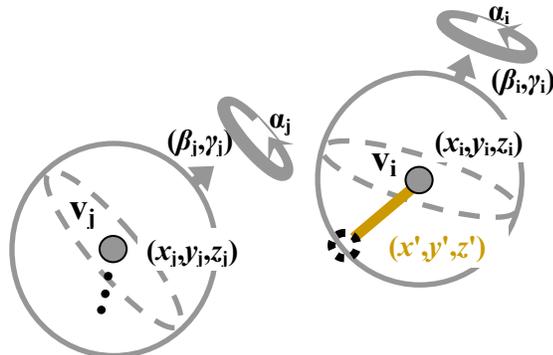
Once we construct the model, the parameters for the model – including the optimal orientation $\bar{x}_{ij}$ corresponding to each edge, and the template for each part – are learned by training the model on a crystallographer-determined structure. Learning the orientation parameters is fairly simple: we rotate each atom into a *canonic orientation*. In this orientation, the parent bond is in the +*z* direction. For template averaging, to avoid averaging out useful features, we also require that the first child bond be pointed in the +*x* direction in the *x-y* plane. If there is no child of this atom, we don't worry about the second rotation. For the root, we specify a child that is rotated into the +*z* direction instead; the other child (if one exists) becomes the +*x* direction. Then, for each *child* of this atom, we record the distance *r* and orientation (*θ,φ*) *in the canonic orientation*. We average over all atoms *of a given type* in our training set – e.g., we average over all lysine C_β's – to determine average parameters r_{avg}, θ_{avg}, and φ_{avg}. Converting these averages from spherical to Cartesian coordinates gives precisely the ideal orientation parameters $\bar{x}_{ij}$. A similar rotation into canonic orientation is employed when learning the model templates; in this case, the algorithm simply samples a 3x3x3 neighborhood around the atom and averages each of the 27 points over all atoms of a given type in out training set. It is important to note that the supervised learning algorithm should train on data with a similar resolution to the test data! Figure 3 shows an overview of the learning process.

For each part in the model, the matching algorithm searches through a six-dimensional conformation space $(x, y, z, \alpha, \beta, \gamma)$. We consider breaking each dimension into a number of discrete bins. The translational parameters *x*, *y*, and *z* are allowed to range over a region in the unit cell, the rotational parameters range such that $\alpha \in [0, 2\pi)$, $\beta \in [0, \pi]$, and $\gamma \in [0, 2\pi)$.

One issue that arose in our implementation involved matching amino acids with rings. Several amino-acid topologies include cycles. This presents a problem for our fast matching algorithm, which requires an acyclic graph. However, this is not really a problem as all the rings can be treated as rigid objects. While not entirely true (proline's ring has a somewhat variable "pucker" to it), this approximation is close enough for this application. By disallowing rotation about ring bonds, we can ignore one bond in the ring without letting the split ring freely flop around.

A much more difficult issue that arose involved collisions. Occasionally, the matching algorithm would return a structure with overlapping atoms, or atoms so close together, the structure could not have possibly occurred physically. Since such a structure corresponds to the global optimal configuration of parts in this simplified model, it is not clear how to handle this case. What we have opted to do is to explore the space of non-optimal solutions via "soft maximums". In the Felz-Hut dynamic program, when considering all possible child locations, instead of taking the maximal scoring (minimal cost) location, we take a location that scores *x*, with *probability*

$$P(x) = \frac{e^{-x^2/\sigma^2}}{\sum_i e^{-x_i^2/\sigma^2}}$$ (where the denominator sums over all possible choices).

The parameter σ controls the softness of the maximum, i.e., the likelihood of a non-optimal solution. Our algorithm repeats the soft-maximum process with softer and softer maximum until a legal structure is found. The legal structure may not be the correct structure, though, it only means that it is physically possible. Additionally, using soft maxima leads to quadratic running time rather than the original DP algorithm's linear runtime. However, by "pruning" candidate locations that are clearly impossible, the actual performance degradation is acceptable.
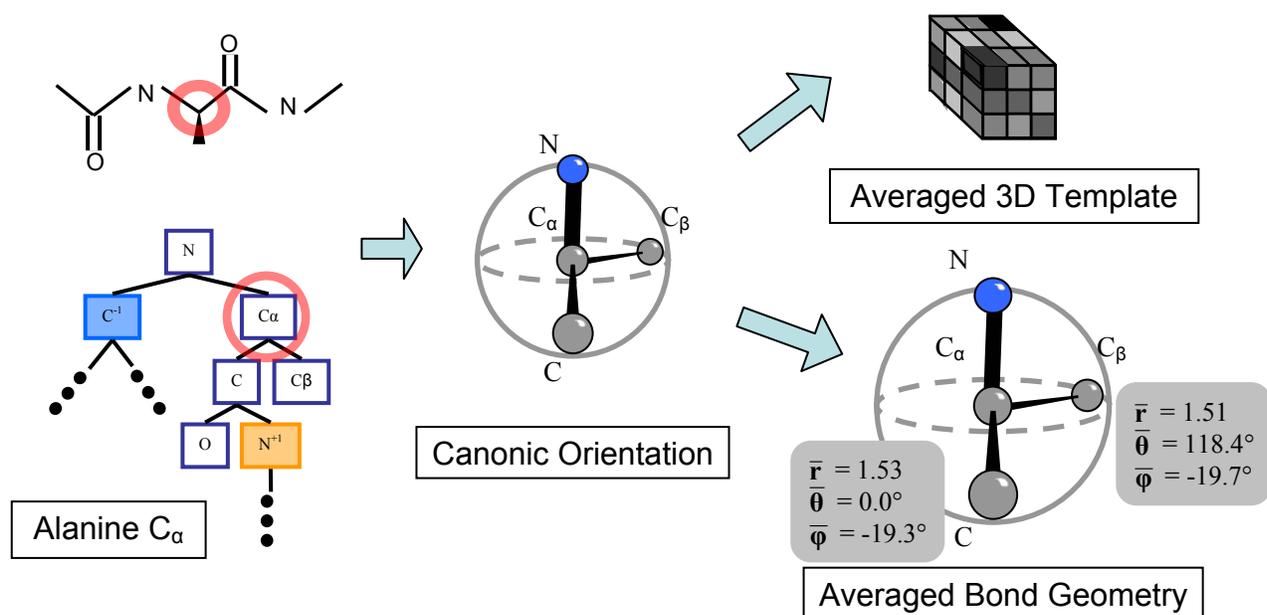
Figure 2: An overview of the parameter-learning process. For each atom of a given type – in this case alanine C_α – we rotate the atom into a canonic orientation, where the parent bond is located in the +z direction, and the first child is in the +x direction in the x-y plane. We then average over every atom of that type to get an average template and average bond geometry.

Experimental Studies

We built pictorial-structure models for 4 of the 20 amino acids (alanine, valine, tyrosine, and lysine), and learned the parameters using a single 1.7 Å resolution protein as our training set. A single 1.9 Å resolution protein served as our test set. Both proteins structures had already been elucidated by crystallographers, allowing us to compare our results to the "correct" mapping. As a simple experiment, we tried locating a single amino acid within a 10 Å diameter sphere of density. We chose these spheres such that they completely contained the entire amino acid for which we were searching. To avoid (possibly correctly) matching the wrong region in the density map, we assumed that we are given a 2.0 Å sphere in which each alpha carbon (one of the backbone atoms) is known to exist.

Within this 10 Å sphere, we considered a 0.5 Å grid on which each atom must be centered. While fairly crude, refinement techniques could later be used to fine tune the coarsely solved structure. We considered 12 bins in α, 7 bins in β, and 12 bins in γ. Running time varies based on the size of the amino acid for which we are looking, but on a 2.25 GHz Pentium IV, it took about 30 seconds to find the globally optimal location. Using soft maximums took about 90 seconds for each additional non-optimal solution.

Our first task looked at, given a region of density and the type of amino acid contained within, how closely could we place the atoms to the actual atom locations. We ran the pictorial-structure matching algorithm on every instance of each of the four amino acids for which we had models. We then compared, among the 599 atoms placed, the distance between the predicted and actual distance. Figure 4 shows the accuracy with which atoms were placed. The curve in this figure

Figure 3: Amino acid placement accuracy. This plot shows the fraction (*y*-axis) of atoms that were placed with the specified accuracy (*x*-axis) or better.
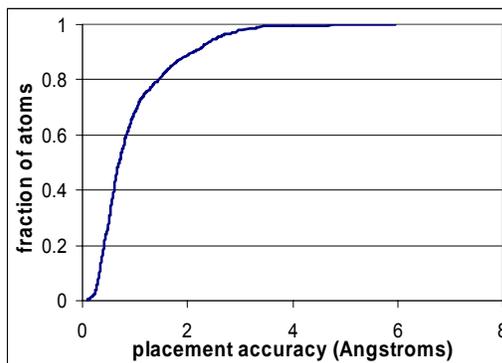
shows the fraction of atoms that were placed with *at least* the corresponding accuracy. As the plot shows, 30% of atoms were placed within 0.5 Å, 72% within 1.0 Å, and 93% within 2.0 Å.

While this result seems quite good, it does not allow for a very meaningful comparison to other methods. Hence, we also used the pictorial-structure model for an artificial discrimination task. Given the same spheres of density as before, we did not provide the matching algorithm the amino acid type contained within. Instead, the algorithm matched each of the four pictorial structures into the image, normalized the score, and returned the type of the highest-scoring model. Table 2 shows a confusion matrix for this prediction task.

As Table 1 shows, the algorithm does not do a particularly good job on the discrimination task, scoring 61% accuracy. These four amino acids vary quite wildly, however, and with quite good 1.9 Å data, the predictive accuracy should be somewhat better. This task is an artificial one, though; the algorithm is designed to place a known molecule into a region rather than predict the contents of a region.

Finally, Figures 5 and 6 show our algorithm's output for several good (Figure 5) and bad (Figure 6) matches. In cases where the match the pictorial structure matching algorithm found was poor, usually the quality of the density map was poor as well. In some cases the correctness of the crystallographer's solution is uncertain.

Related Work

A number of attempts have been made in the past several decades to automate the interpretation of electron density maps. By far the most successful is ARP/wARP [4,5]. The rather proprietary technique involves placing and moving atoms in the density map randomly, until they do a sufficiently good job of explaining the density observed. By using this Monte Carlo approach multiple times, and averaging the output, the results are quite good. This method has been used

Table 2: Confusion matrix showing accuracy for the predictive benchmark.

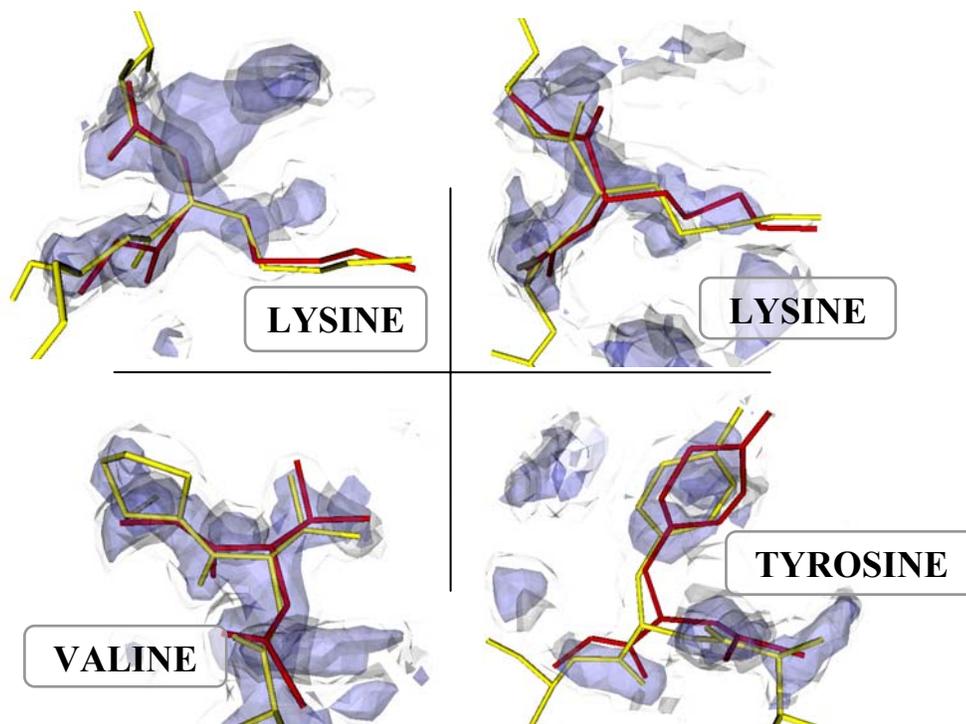|		actual			
		ala	lys	tyr	val
predicted	**ala**	7	0	0	2
	lys	1	6	1	3
	tyr	2	7	8	2
	val	0	1	0	9

Figure 4: Some examples of **good** matches. The crystallographer-determined structures are shown in a *lighter* shade, while the algorithm-determined structures are shown in a *darker* shade. The light and dark clouds show two different contours of density. Note that the actual structure shows additional atoms not in the pictorial structure.
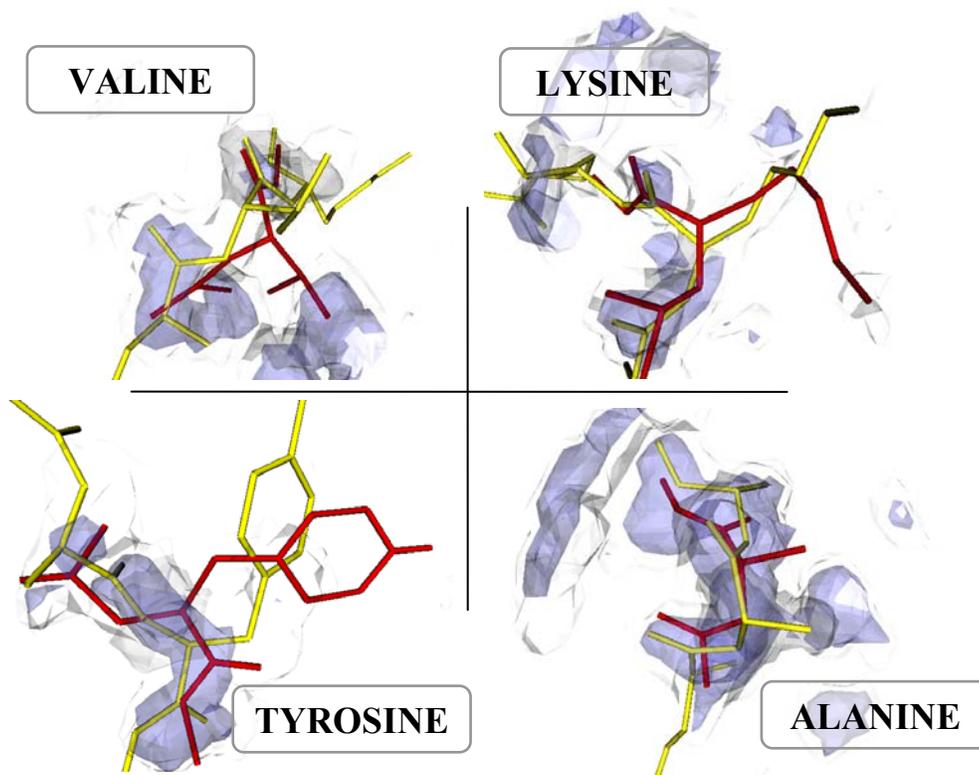

Figure 5: Some samples of **poor** matches. In all of these cases, the density map itself is of poor quality, bringing into question the crytallographer's interpretation.

successfully a number of times in the past, but it has one fairly significant drawback: the density map must be of a fairly high resolution, about 2.5 Å or better.

Two other attempts that have been made are MAID and Textal. In MAID [6], the computer approaches interpretation "as a human would" – by first finding the major secondary structures, alpha helices and beta sheets, and then filling in the remaining regions. Textal [7], on the other hand, takes a pattern-recognition approach to solving the problem. It uses a set of fifteen rotation-invariant features at each of four resolutions to try to determine the type of amino acid contained in a certain region. The features used are mostly statistical, looking at the deviation and skew, for example, of the contained densities. Once the amino acid type is known, a database lookup attempts to place the atoms within the density map. The technique was shown to have good results on moderate resolution (3.0 Å) data, provided it was given the locations of every alpha carbon, in advance. Both of these techniques, unlike our approach, make use of a hierarchy of routines in order to construct a protein model.

Finally, a fourth attempt, fffear [8] uses Fast Fourier Transforms to find secondary structures in poor-quality density maps. For example, it can find alpha helices in maps with as poor as 5.0 Å resolution; beta sheets can be located in maps with resolutions of 4.0 Å or better. Unlike our algorithm, fffear is constrained to the use of rigid templates. None of the four approaches constructs a flexible atomic model in order to interpret the electron density map.

Conclusions and Future Work

Pictorial structures seem to be a powerful tool for building a flexible molecular model, and the fast matching algorithm seems to be useful at placing these models into a region of unknown density. This paper extends the work of Felzenszwalb and Huttenlocher by extending the pictorial-structure framework to three dimensions. It uses this framework to build a flexible atomic model. However, the amount of information we assume is available – a precise 10 Å sphere containing the entire amino acid and knowledge of the approximate alpha-carbon locations – makes this method currently impractical for automated map interpretation. Furthermore, while the matching algorithm is quite efficient, it will not scale to an entire protein, which would require a pictorial structure with thousands of parts.

We next plan to use our current approach as the refinement phase that complements a coarser method. Rather than model the configuration of each individual atom, as in our current method, a coarser model would treat each amino acid as a single (rigid) feature, and only concern itself with rotations along the backbone. This model could place large pieces of the protein at once into the density map on a much coarser grid. Then, with approximate amino-acid locations and alpha-carbon locations, our current finer-grained algorithm could place each individual atom.

There is also room in our current method for improvement within the **match$_i$** function. Algorithms like fffear [8] have had much success finding rigid templates in poor-quality maps. The modular nature of the **match$_i$** function in the pictorial-structure matching algorithm makes taking advantage of another, more powerful matching algorithm quite easy. These improvements would prove an important step in development of an accurate, automated density map interpretation tool.

Acknowledgements

References

[1]   Z. Zolnai *et al*. (2003). Project Management System for Structural and Functional Proteomics: Sesame. *Journal of Structural and Functional Genomics*.

[2]   V. Lamzin & K. Wilson (1993). Automated refinement of protein models. *Acta Crystallographica*. D49.

[3]   A. Perrakis, T. Sixma, K. Wilson, & V. Lamzin (1997). wARP: improvement and extension of crystallographic phases by weighted averaging of multiple refined dummy atomic models. *Acta Crystallographica*. D53.

[4]   D. Levitt (2001). A new software routine that automates the fitting of protein X-ray crystallographic electron density maps. *Acta Crystallographica*. D57: pp. 1013-1019.

[5]   T. Ioerger, T. Holton, J. Christopher, & J. Sacchettini (1999). TEXTAL: a pattern recognition system for interpreting electron density maps. *Intl. Conf. on Intelligent Systems for Molecular Biology*. pp. 130-137.

[6]   K. Cowtan (1998). Modified phased translation functions and their application to molecular-fragment location. *Acta Crystallographica*. 54(5): pp. 750-756.

[7]   D. Cromer & J. Mann (1967). Compton scattering factors for spherically symmetric free atoms. *J. Chemical Physics*. 47: pp. 1892-1983.

[8]   M. Fischler & R. Elschlager (1973). The representation and matching of pictorial structures. *IEEE Transactions on Computers*. 22(1): pp. 67-92.

[9]   P. Felzenszwalb & D. Huttenlocher (2000). Efficient matching of pictorial structures. *IEEE Conference on Computer Vision and Pattern Recognition*. pp. 66-73.

[10] M. Burl, T. Leung & P. Perona (1998). A probabilistic approach to object recognition using local photometry and global geometry. *European Conference on Computer Vision*. 2: pp. 628-641.

[11] P. Lipson, E. Grimson & P. Sinha (1997). Configuration based scene classification and image indexing. *IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1001-1013.