Fast protein structure alignment using Gaussian overlap scoring of backbone peptide fragment similarity

David W. Ritchie1,*, Anisah W. Ghoorah1,2, Lazaros Mavridis3 and Vishwesh Venkatraman4
1Inria Nancy, 615 Rue du Jardin Botanique, 54600 Villers-lès-Nancy, 2Université de Lorraine, LORIA, 54506 Nancy, France, 3University of St. Andrews, St. Andrews KY16 9AJ, Scotland, UK and 4Norwegian University of Science and Technology, Høgskoleringen 5, Trondheim, Norway

Associate Editor: Anna Tramontano

ABSTRACT

Motivation: Aligning and comparing protein structures is important for understanding their evolutionary and functional relationships. With the rapid growth of protein structure databases in recent years, the need to align, superpose and compare protein structures rapidly and accurately has never been greater. Efficient pattern matching algorithms such as FASTA (Lipman and Pearson, 1985) and BLAST (Altschul et al., 1990) are now standard tools for searching nucleotide and amino acid sequence databases. Dynamic programming (DP) algorithms provide a rapid way to find the optimal global (Needleman and Wunsch, 1970) or local (Smith and Waterman, 1981) alignments of pairs sequences. However, there is still no generally accepted standard for how to align and compare two similar protein structures (Sippl and Wiederstein, 2008).

Results: We have developed a novel protein structure alignment algorithm called ‘Kpax’, which exploits the highly predictable covalent geometry of Cα atoms to define multiple local coordinate frames in which backbone peptide fragments may be oriented and compared using sensitive Gaussian overlap scoring functions. A global alignment and hence a structural superposition may then be found rapidly using dynamic programming with secondary structure-specific gap penalties. When superposing pairs of structures, Kpax tends to give tighter secondary structure overlays than several popular structure alignment algorithms. When searching the CATH database, Kpax is faster and more accurate than the very efficient Yakusa algorithm, and it gives almost the same high level of fold recognition as TM-Align while being more than 100 times faster.

Availability and implementation: http://kpax.loria.fr/

Contact: Dave.Ritchie@inria.fr.

Supplementary information: Supplementary data are available at Bioinformatics online.

Received on May 19, 2012; revised on October 4, 2012; accepted on October 14, 2012

1 INTRODUCTION

Aligning and comparing protein structures is important for understanding their evolutionary and functional relationships (Hasegawa and Holm, 2009; Sierk and Kleywegt, 2004). By quoting Lewis Carroll’s Red Queen character (it takes all the running you can do, to keep in the same place), Holm et al. (2008) recently alluded to the computational challenge of searching increasingly large protein structure databases. Today, 4 years later, with some 80000 protein structures in the Protein Databank and with the number of new structures being solved growing exponentially (Berman, 2008), the need to compare the 3D structures of protein molecules rapidly and reliably has never

*To whom correspondence should be addressed
From this starting point, we calculate a similarity score for putative pairs of backbone fragments using a sum of products of Gaussian functions centred on the $C_\alpha$ positions within each fragment and on a small number of further ‘virtual’ atom positions that encode the apparent centre of mass (COM) of each protein, as described below. We then use these Gaussian overlap scores to construct a DP scoring matrix with SSE-dependent gap penalties which allows an optimal set (according to our scoring function) of equivalent residues to be found and superposed efficiently. A particular feature of a scoring function based on sums of Gaussians is that it implicitly favours close contacts without necessarily needing to exclude long-range pairs. Although we implemented SSE-dependent gap penalties initially to improve database retrieval performance, we find that using such penalties together with our Gaussian scoring function tends to produce alignments with smaller numbers of aligned residues but with somewhat tighter 3D overlays of SSEs than the other alignment algorithms studied here.

It should be noted that Taylor and Orengo (1989) first demonstrated the utility of using local coordinate systems. They constructed a local coordinate frame for each residue using N-C and $C_\gamma$-H bond vectors, but crucially they used it only to compare patterns of intra-molecular vectors rather than inter-molecular distances. Although later approaches that use internal torsion angles such as Yakusa (Charpentier et al., 2005) and SABIC (Shen et al., 2010) share a similar insight to our approach, the inter-molecular scoring functions in these algorithms can only compare one angle or one distance for each pair of residues. In contrast, for each position along a backbone, our Gaussian scoring function can be used to score the similarity of two local backbone fragments by comparing the positions of up to three residues in each direction along the chain. Hence, we also use Gaussian overlap scores to define the secondary structure of each residue according to its similarity to a model $\alpha$-helix or $\beta$-sheet. On a contemporary workstation, our multi-threaded algorithm can calculate thousands of structural alignments per second. Hence we named it ‘Kpax’ (being short for ‘thousands of protein alignments by canonical $C_\alpha$ coordinate centres per second’).

2 METHODS

The Kpax similarity score for a pair of residues $i$ and $j$ has the form:

$$K_{ij} = w_1 K^{\text{local}}_{ij} + w_2 K^{\text{partial}}_{ij} + w_3 K^{\text{Bloom}}_{ij},$$

where each $K$ is a normalized score with a value between zero and one, and each $w$ is a weight factor in the same range and normalized such that $w_1 + w_2 + w_3 = 1.0$. The first term gives a measure of the local similarity of a pair of residues when calculated in a common coordinate frame. The second term measures their spatial similarity with respect to the relative position of the COM of a protein (also calculated in the same frame). These two terms are described in more detail below. The final term is the Bloom62 amino acid similarity score, in which each pair-wise score has been scaled onto the above range. Here, we set $w_3 = 0$ in order to consider only structure-based alignments.

2.1 Constructing local $C_\alpha$ coordinate frames

Here, we set up a local 3D coordinate system for each amino acid residue by using the coordinates of its $C_\alpha$, C and N backbone atoms to construct...
positive

We then transform each of the 2 \( H \) and \( H_e \) atoms on the positive and negative \( y \) sides of the \( xz \) plane, respectively. For glycine, applying \( K \) will locate the \( H_a \) and \( H_p \) symmetrically on either side of the \( xz \) plane. However, none of the above backbone atoms play any subsequent role in our scoring function because their local-frame coordinates are almost invariant.

Figure 1 shows how fragments of a theoretical \( \alpha \)-helix and \( \beta \)-strand from the CCP4 fragment library (Cowtan, 2008) may be located in a canonical orientation at the origin using the \( K \)-transform. Using a different residue to calculate the \( K \)-transform would shift each fragment by the corresponding number of peptide units along its principal secondary structure axis. For an infinitely repeating polyalanine structure axis. For an infinitely repeating polyalanine fragment scoring function, we transform the COM into the local frame of each residue position. For a given pair of residues, \( i \) and \( j \), the ‘K-score’ is calculated as a product of Gaussian overlap integrals (Equation 6) using the local frame pair-wise \( C_a \) distances (right).

\[
K_{ij}^{\text{local}} = e^{-\sum_{k} R_{i,k}^{2} / 4\sigma_i^2},
\]

where \( R_{i,k} \) is the distance between the \( C_a \) atoms at positions \( i + k \) and \( j + k \) on chains A and B, respectively, and \( \sigma_i^2 \) is a scale factor which we currently set to unity. The summation excludes \( k = 0 \) because \( R_{i,k,0} = 0 \) by construction. In other words, the local similarity between residues \( i \) and \( j \) is calculated as a Gaussian sum of the squared distances between pairs of up-stream and down-stream \( C_a \) atoms.

To obtain suitable values for the parameters \( \sigma_i \), we treat each \( \sigma \) as the standard deviation (SD) of a normal Gaussian distribution, and by considering each residue in turn of each domain in the CATH database (Cuff et al., 2009), we calculated the mean and SDs of all residues at relative positions ±1 to ±3 with respect to the residue under consideration to obtain the values: \( \sigma_{+1} = 1.46 \), \( \sigma_{-1} = 1.03 \), \( \sigma_{+2} = 3.72 \), \( \sigma_{-2} = 3.54 \), \( \sigma_{+3} = 5.52 \) and \( \sigma_{-3} = 5.74 \). To apply the above scoring function to residues near the N and C termini of one or both chains, we use a simple wrapping scheme in order to maintain the total number of terms in Equation (6). For example, when \( i = 1 \), there are no residues with negative offsets, and so the contribution from \( \psi_1^{+1} \) to \( \psi_n^{+n} \) is doubled in Equation (6). When \( i = 2 \), the contributions from \( \psi_1^{+2} \) and \( \psi_1^{+1} \) are calculated using Equation (6), but the contribution from \( \psi_1^{+2} \) to \( \psi_1^{+1} \) is doubled, and so on. A similar scheme is used for the C-terminal residues.

\[
K_{ij}^{\text{spatial}} = e^{-\sum_{k=-\infty}^{\infty} R_{i,k}^{2} / 4\sigma_i^2},
\]
where \( R_{i+k,j+k} \) is the distance between the VAs at positions \( i+k \) and \( j+k \), and the summation excludes \( k=0 \) as before. Each \( r_k \) is considered as the SD of a Gaussian distribution of the VA coordinates. Hence, VAs were placed on all of the residues of the CATH database and the SDs were calculated from the resulting distributions to give \( r_{-1} = 2.43 \), \( r_{-2} = 4.13 \), \( r_{-3} = 5.74 \) and \( r_{-4} = 5.58 \).

2.4 Assigning secondary structure elements

By placing the centre residue of a theoretical model of an \( \alpha \)-helix and a \( \beta \)-strand at the local coordinate origin, Equation (6) provides a straightforward way to detect the SSE type of a given residue by calculating its local-frame similarity to each model fragment. Here, we use the five-residue template files \( \text{theor-helix-5.pdb} \) and \( \text{theor-strand-5.pdb} \) from the CCP45 fragment library (Cowtan, 2008), and for each residue in a given structure we assign type \( \alpha \) to residue \( i \) if \( K_{i,3} > 0.1 \), \( K_{i,5} \), \( K_{i,10} > 0.1 \), and we assign \( \gamma \) (i.e. loop or coil) otherwise. Any singleton \( \alpha \) or \( \beta \) residues within a \( \gamma \) region are re-assigned to \( \gamma \).

2.5 Dynamic programming structural alignments

The optimal ‘pose-invariant’ alignment for two chains, \( A \) and \( B \), of length \( N_A \) and \( N_B \) is calculated by first initializing a DP matrix of dimension \((N_A + 1) \times (N_B + 1)\) using \( D_{0,0} = D_{0,0} = 0.0 \), and by filling the remaining elements using

\[
D_{ij} = \max(D_{i-1,j-1} + K_{ij}, D_{i-1,j} + P_{i,1}, D_{i,j-1} + P_{1,1} - P_{i,j}),
\]

where \( K_{ij} \) is the similarity score for residues \( i \) and \( j \) (Equation 1) and \( P_{i,j} \) is the penalty for introducing a gap between residues \( i \) and \( j \). Since a typical \( C_u-C_u \) distance is about 3.8 Å, we set \( \sigma = (\sigma_{C_u} + \sigma_{-1})/2 = 1.245 \) Å and we calculate the gap penalty unit, \( \rho \), using \( \rho = e^{-3.8/4 \times 1.245} \approx 0.1 \). In other words, the gap penalty is derived directly from the physical length scale of one peptide unit.

To penalize alignment gaps within regular secondary structures and to encourage gaps in loop regions, we set \( P_{i,1} = 2 \rho \) if positions \( i \) and \( j \) were both called as \( \alpha \), \( P_{i,j} = \rho \) if both \( \beta \), \( P_{i,j} = \rho/2 \) for both \( \gamma \), and \( P_{i,j} = 0 \) otherwise. No gap extension penalty is used here, nor is any penalty applied for overhangs at the start or end of a chain. However, if a chain contains any physical breaks (we assume a chain break exists if the distance between consecutive \( C_u \) atoms exceeds 1.5 x 3.8 Å), we set the gap opening penalty to zero for the two residues that border each break. Because the local Gaussian scoring function automatically generates low similarity scores for any residues near a physical break, this is sufficient to ensure that chains with missing segments are handled gracefully.

By tracing through the DP matrix in the usual way, we obtain a global alignment in which each diagonal step corresponds to a matched pair of residues. An overall ‘K-score’ is then calculated as

\[
K_{AB} = \sum_{i=1}^{N_A} \sum_{j=1}^{N_B} \mu_{ij} K_{ij},
\]

where \( \mu_{ij} = 1 \) if residues \( i \) and \( j \) are matched, and \( \mu_{ij} = 0 \) otherwise. Despite being a global structural similarity score, it is worth noting that this penalty-free score is ‘pose-invariant’ in that it does not depend on the orientations of the given proteins, and that for two perfectly matching backbones it will be numerically equal to the number of aligned residues. It is also worth noting that this score does not depend on the order in which the chains are given, and it does not involve any least-squares fitting calculations. Indeed, our technique of placing backbone peptide fragments in a canonical orientation at the coordinate origin costs only \( O(N) \) operations per protein. Populating the DP scoring matrix still requires \( O(N^2) \) operations, although the form of Equations (6) and (7) allows this cost to be reduced to essentially just two exponential function calls per matrix element.

For typical protein domains, we find that \( K_{AB}^{\text{local}} \) and \( K_{AB}^{\text{initial}} \) are almost equally effective scoring functions. Therefore, by default, we use a combination of both scores with weights \( w_1 = 0.5 \), \( w_2 = 0.5 \). When superposing similar domains, the best path through the DP matrix is often near the main diagonal. Hence, in principle, many pair-wise similarity scores never need to be calculated. However, because the overall algorithm is so fast, we find that the main rate limiting step comes from reading the coordinate data into computer memory. Therefore, we currently do not apply any banding or lazy evaluation techniques to accelerate the DP calculation.

2.6 Calculating and optimizing 3D superpositions

Given an alignment from the DP matrix, an initial 3D structural superposition is calculated by least-squares fitting (Kabsch, 1976) in which the pair-wise K-scores are used as fitting weights. This is then refined by one further cycle of fitting with uniform weights in which the weight for any pair of residues is set to zero if the distance between their \( C_u \) atoms exceeds 8 Å. This often produces a visually acceptable superposition. We then optimize this initial superposition by applying further rounds of DP and fitting using a pose-dependent Gaussian score based on pairs of \( C_u \) distances

\[
G_{ij}^{\text{sce}} = e^{-r_{ij}^2/\sigma_{ij}^2},
\]

where a Cartesian grid is used to find residue pairs within the above distance threshold, and where no gap penalty is used in the DP matrix. This procedure allows some additional residue pairs (e.g. in loop regions) to be found and added to the alignment. In analogy to Equation (9), we then define a pose-dependent Gaussian superposition score as

\[
G_{AB} = \sum_{i=1}^{N_A} \sum_{j=1}^{N_B} \mu_{ij} G_{ij}^{\text{sce}}.
\]

When using \( \sigma_{\text{sce}} = 1.4 \) Å, a gap penalty of \( \rho = 0.1 \), and uniform fitting weights, we find that applying just two rounds of Gaussian optimization is normally sufficient to give a satisfactory superposition with a low \( C_u \) RMSD. Sippl and Wiederstein (2012) recently described a similar Gaussian sum expression to rank alignments generated by their TopMatch algorithm.

2.7 Searching structural domain databases

To allow efficient queries against structural databases such as CATH or SCOP (Murzin et al., 1995), we first pre-calculate and store the up-stream and down-stream fragment coordinates for every database residue (i.e. 6 \( C_u \) and 6 VA coordinates per residue). This allows a database to be searched rapidly using a single round of DP to calculate the K-score similarity between the query and each member of the database.
However, in order to rank alignments and superpositions of chains of different lengths, Kpax uses normalized scores defined by

\[
\bar{K}_{AB} = K_{AB}/(N_A N_B)^{1/2},
\]

\[
\bar{G}_{AB} = G_{AB}/(N_A N_B)^{1/2}.
\]

In particular, \(\bar{K}_{AB} = 1\) and \(\bar{G}_{AB} = 1\) represent the perfect alignment and superposition of two identical backbones, respectively. Except for queries involving highly populated domain families such as the immunoglobulins, often only a relatively small number of database structures will superpose well onto the query. Hence, Kpax calculates 3D superpositions and G-scores only for the top 300 structures with the best K-scores.

2.8 Implementation details and availability

Kpax has been implemented in the C programming language using thread-safe programming techniques. Hence all database searches are multi-threaded by default. The program and some command scripts for building CATH and SCOP databases on Linux systems are available at [http://kpax.loria.fr/](http://kpax.loria.fr/). Databases of user-defined collections of PDB files may be built in a similar way. However, Kpax currently assumes that each PDB file contains just one chain, and it only considers the first lent residues for each pair of structures without requiring any iteration or least-squares fitting.

To understand better the differences between the selected alignment methods, Supplementary Table S1 lists for each pair of methods the individual and average \(C_r\) RMSD differences between the computed positions of each superposed structure when calculated with respect to a common reference structure. This table shows that, on average, the 3D superpositions produced by Kpax resemble most closely those of TM-Align (with an average \(C_r\) RMSD of 1.31 \(\AA\)), and indeed that the Kpax and TM-Align superpositions are more similar than the superpositions calculated by all other pairs of algorithms (CE/Sheba: 2.77; CE/TM-Align: 2.05; CE/Kpax: 2.21; Sheba/TM-Align: 2.04; Sheba/Kpax 2.70 \(\AA\) RMSD). The largest individual difference between Kpax and TM-Align is seen in the first pair (PDB codes: 1fxi/1ubq), in which the superposed positions of the ubiquitin domain (1ubq) differ by 3.83 \(\AA\) RMSD. An example of the SSE assignments obtained using the Kpax template-matching procedure (Section 2.4) in comparison with the SSEs calculated by DSSP (Kabsch and Sander, 1983) and Stride (Frishman and Argos, 1995). Supplementary Figure S1 shows a colour version of this figure along with a further nine examples. These figures show that our algorithm often assigns quite similar \(\alpha\) and \(\beta\) SSEs to Stride and DSSP, although there are often some small differences around the start and end positions of each SSE. Also, as expected, Kpax does not distinguish specific types of turn from random coil regions. Nonetheless, because the main aim here is to provide SSE-dependent alignment gap penalties, the above procedure allows the secondary structure environment of each residue to be estimated rather well and very rapidly without requiring external software.

3 RESULTS AND DISCUSSION

3.1 Comparing SSE assignments with Stride and DSSP

Figure 4 shows an example of the SSE assignments obtained using the Kpax template-matching procedure (Section 2.4) in comparison with the SSEs calculated by DSSP (Kabsch and Sander, 1983) and Stride (Frishman and Argos, 1995). Supplementary Figure S1 shows a colour version of this figure along with a further nine examples. These figures show that our algorithm often assigns quite similar \(\alpha\) and \(\beta\) SSEs to Stride and DSSP, although there are often some small differences around the start and end positions of each SSE. Also, as expected, Kpax does not distinguish specific types of turn from random coil regions. Nonetheless, because the main aim here is to provide SSE-dependent alignment gap penalties, the above procedure allows the secondary structure environment of each residue to be estimated rather well and very rapidly without requiring external software.

3.2 Comparison with CE, Sheba and TM-Align

As a first test of the Kpax alignment algorithm, we compared its performance with CE, Sheba and TM-Align using the 10 ‘difficult’ pairs of structures from Fischer et al. (1996) that were first used as a reference benchmark by Shindyalov and Bourne (1998) and later by several other groups (Lackner et al., 2000; Novotny et al., 2004; Shen et al., 2010; Shibberu and Holder, 2011). Table 1 indicates that all of these algorithms can calculate good alignments for these structures, although the variation in some of the numbers of aligned residues and RMSDs suggests that these examples still appear to be difficult to align consistently. On the other hand, Supplementary Figure S2 confirms graphically that each algorithm gives rather similar 3D superpositions for each pair of folds. This demonstrates the difficulty of trying to compare different structure alignment algorithms directly using such raw numerical measures. Nonetheless, it is worth noting that the Kpax superpositions have RMSDs that are lower than CE and TM-Align in all cases, and which are generally comparable with, but slightly worse than, those of Sheba. Furthermore, the final column of Table 1 (column ‘Kpax-K’) shows that using just one round of DP with the pose-invariant K-scores provides a fast way to calculate a good initial superposition. This shows that the K-score is identifying many equivalent residues for each pair of structures without requiring any iteration or least-squares fitting.

To understand better the differences between the selected alignment methods, Supplementary Table S1 lists for each pair of methods the individual and average \(C_r\) RMSD differences between the computed positions of each superposed structure when calculated with respect to a common reference structure. This table shows that, on average, the 3D superpositions produced by Kpax resemble most closely those of TM-Align (with an average \(C_r\) RMSD of 1.31 \(\AA\)), and indeed that the Kpax and TM-Align superpositions are more similar than the superpositions calculated by all other pairs of algorithms (CE/Sheba: 2.77; CE/TM-Align: 2.05; CE/Kpax: 2.21; Sheba/TM-Align: 2.04; Sheba/Kpax 2.70 \(\AA\) RMSD). The largest individual difference between Kpax and TM-Align is seen in the first pair (PDB codes: 1fxi/1ubq), in which the superposed positions of the ubiquitin domain (1ubq) differ by 3.83 \(\AA\) RMSD. Figure 5 shows the TM-Align and Kpax overlays for this case. A large colour version of this image is shown in Supplementary Figure S3. Visual inspection of these figures shows that Kpax produces a tighter overlay of the main \(\alpha\)-helix and the three large \(\beta\)-strands in this pair than TM-Align.

3.3 Comparing the local and spatial scoring functions

To investigate the strengths and weaknesses of the Kpax scoring functions, we compared the performance of Kpax’s local, spatial and local-plus-spatial scores with TM-Align using six low sequence identity pairs of domains identified previously by Sippl and Wiederstein (2008) and six further pairs from Gerstein and Levitt (1998). Table IV of Mavridis et al. (2012) gives some results from CE, SSM, Dali and 3D-Blast for these examples.

Although we consider these pairs to be more challenging than those of Fischer et al., Table 2 shows that both TM-Align and the combined Kpax scoring function find full-length superpositions in all twelve cases (see Supplementary Figs S4 and S5 for images of the 3D superpositions). As in Table 1, Kpax produces alignments with lower RMSDs but also with lower numbers of aligned residues than TM-Align for the majority of cases. In terms of trading between RMSD and number of aligned residues, these results reaffirm the tendency for Kpax to produce
superpositions towards the low RMSD, or ‘tight’, end of the number/RMSD performance metric.

On the other hand, Table 2 also shows that the individual local and spatial functions (tabulated ‘Kpax-L’ and ‘Kpax-S’, respectively) both fail to find the full alignment for the pair d1mtby/d1rtya1, and the spatial function for one further pair (d1l3a/d1efya2). The first of these cases involves correctly matching the 4-helix bundle of rubrerythrin (d1rtya1) within the larger 9-helix bundle of the methyl monoxygenase hydroxylase (d1mtby). This is clearly difficult for both scoring functions because locally all helices will look the same, and because spatially the COMs of these different-sized domains will not closely coincide in a good overlay. Similarly, for the second pair, the different positions of their COMs with respect to each other will fail to find the full alignment for the pair d1l3a/d1efya2. These examples show that using the combined Kpax score appears to overcome the weaknesses in the individual scoring functions.

The last column of Table 2 shows that, as before, Kpax generally gives quite similar overlays to TM-Align. However, the final example involving a pair of β-propellers (SCOP codes: d4aaha/d1gofa3) has a large RMSD difference of 11.6 Å between the Kpax and TM-Align superpositions. Figure 6 shows these superpositions graphically, and Supplementary Figure S6 provides a large colour version of the same figure. From close visual inspection of these figures, we believe the tighter Kpax alignment to be superior despite the smaller calculated number of aligned residues.

### 3.4 CATH database search comparison

As a final and more demanding test, we compared the ability of Kpax, TM-Align and Yakusa to retrieve structural homologues from CATH using a diverse set of structural queries. We chose TM-Align because in our opinion (based on a preliminary comparison of several structural aligners) it is one of the best structural aligners available, and we chose Yakusa because it was specifically designed for rapid database searching. For this test,

---

**Table 1.** Superposition performance comparison for the 10 ‘difficult’ pairs of structures from Fischer et al. (1996)*

<table>
<thead>
<tr>
<th>PDB codes</th>
<th>CEa</th>
<th>Shebaa</th>
<th>TM-Alignd</th>
<th>Kpaxb</th>
<th>Kpax-Sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>1fiaA/1ubqA</td>
<td>64</td>
<td>49</td>
<td>63</td>
<td>64</td>
<td>63</td>
</tr>
<tr>
<td>1tenA/3hrB</td>
<td>87</td>
<td>82</td>
<td>87</td>
<td>84</td>
<td>84</td>
</tr>
<tr>
<td>3hlaA/2rheA</td>
<td>85</td>
<td>62</td>
<td>80</td>
<td>66</td>
<td>41</td>
</tr>
<tr>
<td>2azaA/1paxA</td>
<td>85</td>
<td>74</td>
<td>86</td>
<td>69</td>
<td>64</td>
</tr>
<tr>
<td>1cewl/1molA</td>
<td>81</td>
<td>74</td>
<td>82</td>
<td>69</td>
<td>63</td>
</tr>
<tr>
<td>1clid/2rheA</td>
<td>98</td>
<td>83</td>
<td>100</td>
<td>76</td>
<td>56</td>
</tr>
<tr>
<td>1crA/1edeA</td>
<td>220</td>
<td>139</td>
<td>235</td>
<td>156</td>
<td>93</td>
</tr>
<tr>
<td>2simA/1nsbB</td>
<td>297</td>
<td>235</td>
<td>312</td>
<td>255</td>
<td>201</td>
</tr>
<tr>
<td>1bgeB/2gmpA</td>
<td>109</td>
<td>81</td>
<td>111</td>
<td>77</td>
<td>68</td>
</tr>
<tr>
<td>1tieA/4gfA</td>
<td>117</td>
<td>93</td>
<td>117</td>
<td>100</td>
<td>84</td>
</tr>
<tr>
<td>Average time (s)</td>
<td>1.88</td>
<td>0.48</td>
<td>0.13</td>
<td>0.07</td>
<td>&lt;0.001</td>
</tr>
</tbody>
</table>

*a Listed are the number of residues aligned by each method, along with the corresponding RMSD in parentheses. Calculation times were measured on a 2.8-GHz quad-core Intel Xeon workstation.

*b Calculated using jCE (Prlic et al., 2010).

*c Sheba version 4.0.1 (Jung and Lee, 2000).

*d TM-Align version 20120126 (Zhang and Skolnick, 2005).

**Table 2.** Structural alignment results for 12 low sequence identity pairs*

<table>
<thead>
<tr>
<th>SCOP domains</th>
<th>Kpax-L</th>
<th>Kpax-S</th>
<th>Kpax</th>
<th>TM-Align Δ-RMSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1euda/d1ccwa</td>
<td>79 (2.9)</td>
<td>78 (2.1)</td>
<td>79 (2.2)</td>
<td>103 (3.3)</td>
</tr>
<tr>
<td>d1euda/d1ccwa</td>
<td>79 (2.9)</td>
<td>93 (2.4)</td>
<td>94 (2.3)</td>
<td>101 (3.1)</td>
</tr>
<tr>
<td>d1euda/d1euda</td>
<td>70 (2.9)</td>
<td>72 (3.1)</td>
<td>72 (3.1)</td>
<td>99 (3.4)</td>
</tr>
<tr>
<td>d1gtba/d1m NSStringFromClass</td>
<td>132 (1.8)</td>
<td>130 (1.8)</td>
<td>141 (2.2)</td>
<td>165 (3.0)</td>
</tr>
<tr>
<td>1te2b/1zolA</td>
<td>60 (2.2)</td>
<td>61 (2.2)</td>
<td>62 (2.1)</td>
<td>67 (2.9)</td>
</tr>
<tr>
<td>d1l3a/d1efya2</td>
<td>69 (2.6)</td>
<td>34 (3.4)</td>
<td>63 (2.7)</td>
<td>103 (3.7)</td>
</tr>
<tr>
<td>d1amfu/d1mbpu</td>
<td>151 (2.2)</td>
<td>159 (2.9)</td>
<td>161 (2.7)</td>
<td>217 (3.5)</td>
</tr>
<tr>
<td>d1vdca/d2tmda</td>
<td>99 (1.5)</td>
<td>103 (1.9)</td>
<td>103 (1.9)</td>
<td>115 (2.3)</td>
</tr>
<tr>
<td>d1sqc1/d1leema</td>
<td>200 (2.5)</td>
<td>210 (2.5)</td>
<td>210 (2.5)</td>
<td>288 (3.6)</td>
</tr>
<tr>
<td>d1nall/d1dbaa</td>
<td>119 (3.7)</td>
<td>125 (3.6)</td>
<td>132 (3.7)</td>
<td>229 (4.9)</td>
</tr>
<tr>
<td>d1mtby/d1rtya1</td>
<td>26 (2.3)</td>
<td>38 (2.0)</td>
<td>100 (2.6)</td>
<td>142 (2.4)</td>
</tr>
<tr>
<td>d4aaha/d1gofa3</td>
<td>207 (3.3)</td>
<td>197 (3.6)</td>
<td>178 (3.9)</td>
<td>336 (5.4)</td>
</tr>
</tbody>
</table>

*a Listed are the number of residues aligned by each method with the corresponding Cα RMSD in parentheses.

*b Local scoring (weights: w1 = 1.0, w2 = 0.0) plus grid search.

*c Spatial scoring (weights: w1 = 0.0, w2 = 1.0) plus grid search.

*d Local-plus-spatial scoring (w1 = 0.5, w2 = 0.5) plus grid search.

*The Cα difference in the coordinates of the moving domain between the Kpax and TM-Align superpositions (calculated using ProFit: http://www.bioinf.org.uk/software/profit/).

*These are CATH domains.

*These are cases in which the expected full length alignment was not found.
we selected 213 CATH families for which each family has at least 10 members, and we used CATH's representative structure for each family as the query. These families have an average of 30.7 members, and the query structures have an average of 160 residues. Here, we treat any structure having the same four-digit CATH code as the query as a ‘positive’ instance, and all other structures as ‘negative’ instances. We then measured the ability of each algorithm to retrieve structures having the same CATH code as the query by plotting the rate of true positives against the rate of false positives as the ranked list of matching structures is traversed. The area under the curve (AUC) of such receiver-operator-characteristic (ROC) plots may be used as a single objective measure of performance (Fawcett, 2006).

Figure 7 shows the aggregate ROC curves obtained for each algorithm by vertically averaging each set of 213 ROC curves. This figure shows that TM-Align gives the best overall performance, with an average AUC of 0.976, closely followed by Kpax with an AUC of 0.966. Yakusa gives an average AUC of 0.915. Closer examination of the individual curves (not shown here) indicated that the slightly better performance of TM-Align in this test comes from its ability to superpose more distant related structures. For example, TM-Align achieves AUC >0.99 for 140 of the queries, compared with 102 for Kpax, and just 56 for Yakusa. On the other hand, Kpax is extremely efficient compared with existing structural alignment algorithms. For example, on a 2.8-GHz quad-core workstation, the above calculations took 46 h using TM-Align compared with 2.2 h for Yakusa, and just 22.5 min for Kpax. Furthermore, using only the pose-invariant K-scores without calculating superpositions takes just 13.5 min and gives an almost indistinguishable ROC curve to the superposition search. This corresponds to an average rate of 2980 structural alignments per second. In contrast, from the timing results of a previous study (see Table IV of Mavridis et al., 2012), we estimate that performing the above database searches using CE, SSM, Dali and our own 3D-Blast algorithm would require approximately 286, 454, 1322 and 2012 CPU-hours, respectively. Thus, Kpax offers a useful way to keep ahead of the Red Queen.

Although Kpax allows large domain structure databases such as CATH and SCOP to be searched rapidly, only a single global alignment and superposition is reported for each pair of compared structures. In the future, we would like to use Kpax to search the entire PDB directly. However, if the aim is to detect sub-structure matches e.g. when comparing a single small domain with large multi-domain structures, we expect that the spatial scoring function will be less useful than the local similarity score. Additionally, it will require further work to be able to handle arbitrary PDB files, which might contain multiple structures and conformations, and which would therefore require additional processing to collect and report multiple possible structural alignments.

4 CONCLUSION

We have presented Kpax, a novel protein structure alignment and superposition algorithm that uses multiple $C_{\beta}$ coordinate systems and a Gaussian peptide fragment scoring scheme to provide a sensitive structural similarity score. For the pairs of structures studied here, Kpax gives similar alignment statistics to Sheba, and it generally calls fewer aligned residues with lower RMSDs than CE and TM-Align. However, this does not imply low alignment quality. We have shown that the superpositions produced by Kpax resemble more closely those produced by TM-Align than those of CE and Sheba, and we have demonstrated that Kpax produces tighter superpositions of SSEs than TM-Align in several cases.

We have also shown that Kpax may be used to perform fast and sensitive structural database searches. In our comparison with Yakusa and TM-Align using the CATH database, we showed that Kpax is faster and more accurate than the very efficient Yakusa algorithm, and it gives almost the same high level of fold recognition as TM-Align while being more than 100 times faster. Our timing estimates for CE, SSM and Dali predict even greater speed-ups with respect to these algorithms. These results demonstrate that Kpax is both fast and accurately in comparison with the current state of the art. However, it still has some caveats. For example, it produces only a single rigid global alignment for each pair of compared structures, it cannot handle permutations or multi-structure PDB files, and its spatial scoring function is not well suited for comparing protein domains that differ significantly in size. Nonetheless, with the number of solved protein structures growing ever more rapidly, we believe the publicly available Kpax program will provide a useful tool for high throughput comparisons of 3D protein structures.

ACKNOWLEDGEMENTS

We thank the Associate Editor and three anonymous referees for several suggestions for improvements to the presentation of this article.

Funding: Part of this work was funded by the Agence Nationale de la Recherche, grant number ANR-11-MONU-006-02.

Conflict of Interest: none declared.

REFERENCES

