Informative 3D Visualization of Multiple Protein Structures

Paulo Lai
School of Computing Science and Engineering
University of New South Wales
NSW 2051, Australia
pyview@paulolai.net

Warren Kaplan
Garvan Institute of Medical Research
St Vincents Hospital
384 Victoria St, Darlinghurst 2010, NSW, Australia
w.kaplan@garvan.unsw.edu.au

W. Bret Church
School of Molecular & Microbial Biosciences
University of Sydney
NSW 2006, Australia
b.church@biotech.usyd.edu.au

Raymond K. Wong
School of Computing Science and Engineering
University of New South Wales
NSW 2051, Australia
wong@cse.unsw.edu.au

Abstract

With the continued growth of three dimensional structural information databases comes a corresponding increase in interest in this data for the study of new sequences and an ever-increasing incentive to improve automatic structure annotation methods. We examined various methods of presenting structural information in 3D, focussing on means of identifying regions of interest and the display of related structures. Using such information we investigated how this can be used to view variations between structures.

As the first choice in bioinformatics searches is often BLAST, we created PyView to examine how best to display its results in 3D, choosing to use the open source Biopython library and the PyMol molecular modelling system as the foundation. Depending on the level of detail involved and number of structures to be shown, different visualisation modes are selected for each structure to be overlaid. Alternatively the domains can be located and highlighted resulting in detailed 3D displays of a protein's structure.

Keywords: 3D Visualization, BLAST, PyMol, Biopython

1 Introduction

New adopters of bioinformatics are faced with a bewildering array of databases and tools, many of which seem to solve the same problems, while experienced users often are challenged to integrate available resources in novel and compelling ways.

As a single protocol is rare, end-users are faced with different interfaces and data formats for each program and database. In addition, as most programs don't provide a programmer's interface, parsers are required to be written for their output.

To facilitate research, federated models such as Kleisli (Chung and Wong 1999, Buneman 1998) were used to provide a unified interface to dispersed resources. Next came the data warehouses of Ensembl (Hubbard et. al. 2002), UCSC (Kent et. al. 2002), and others, and finally a number of ad hoc Web services, such as those provided by GenBank (Benson et. al. 2003) / EMBL (http://www.ebi.ac.uk/Tools/webservices/). It has been proposed by Stein (2002) that the logical conclusion of this is a formal Web services model, where all data providers will register with a formalized service registry, which will remove the need for researchers to be concerned about the differences in the interfaces of each database.

However a common interface providing access to all bioinformatics services is far from eventuating; it could also prove undesirable as it may constrain future developments. As such, we examined the challenges in adding bioinformatic features to an existing end-user

application, preferring to concentrate our efforts on driving the analysis rather than rewriting existing code to produce high quality 3D images.

To illustrate this process we looked into developing software to produce information-rich views of proteins. This could, for example, be used to assist in addressing the growing gap between the number of known sequences and solved protein structures.

The number of solved structures in the Protein Data Bank (PDB) (Berman, Westbrook et al. 2000) continues to grow exponentially and at present constitutes approximately 22000 entries. At the same time the number of protein sequences in the NCBI's non-redundant database exceeds 1.5 million and the gap between the number of known structures and sequences continues to grow (http://www.genome.ad.jp/dbget/db_growth.html).

While there are initiatives to automate the solving of experimental structures (Adams, Grosse-Kunstleve et al. 2002), another method of closing the gap is the construction of theoretical models (Sali 1998). The resolution of models built by *ab initio*, or first principles techniques, is regarded as insufficient for the derivation of atomic-level structural information (Marti-Renom, Yerkovich and Sali 2002). Presently only theoretical models built by comparative homology modelling are regarded as being able to provide useful atomic-level structural information (Marti-Renom, Madhusudhan et al. 2002). Homology modelling relies on the principle that despite homologous proteins having little conservation of amino acid sequences, their overall structures are conserved.

The first step in building a homology model is the identification of a structural template, and this is done by matching the sequence of interest against a databank containing the sequences of the proteins in the PDB. One of the most popular sequence databank searching tools is BLAST (Altschul, Gish et al. 1990). Every databank sequence found by BLAST is given a Bit score and an 'Expect' value, commonly referred to as the E-value. Matches in a BLAST result are ranked from best, i.e. those with a high Bit score and low E-value, to worst, those matches with a low Bit scores and high E-value.

A typical BLAST result has three sections: the header contains information about the query sequence and the database searched, a one-line description of each sequence that was found follows, and finally, the actual alignments of the query sequence to every matched sequence.

While the alignment of the BLAST result is in itself valuable, linear alignments alone are not a sufficient basis for selecting appropriate template structures for a high resolution theoretical model. They also fail to take into account additional knowledge the end-user may wish to factor into the model, such as expected domains which would greatly increase the value of potential templates structures in which they occur.

Alternatively we may wish to produce a highly annotated 3D view of a structure simply to assist our understanding of a protein's function.

2 Related Work

ExPASy (Gasteiger, et. al. 2003) provides a web interface to scan PROSITE (Sigrist, et. al. 2002) patterns against a user supplied sequence. The results page displays the locations of matching patterns within the sequence both as a list and graphically in a Java viewer; however it does not show the domains on the 3D structure, even if that information is available. ExPASy's 'NiceSite' view of a PROSITE entry provides a list of PDB (Berman, et. al.2000) accessions which contain that particular pattern. Each accession links to a page displaying the locations of matches on structures of the protein. These linked pages present a static image with options to load the bare PDB file, or the annotated Chime or RasMol (http://www.umass.edu/microbio/rasmol) script. However this is limited to a single domain at a time, and only structures found within the PDB can be displayed.

ENDscript (Gouet, Robert, and Courcelle 2003) is an online service which produces a static 3D structural image from a PDB file, which shows both sequence and structural similarity with related sequences. The radius of the tube representing the backbone is proportional to the calculated RMS deviation between this structure and related structures found via BLAST. The tube's colour, white to red, is based on similarity scores (low to high) calculated from a multiple alignment. As the site's main purpose is generating high quality figures for known structures with well-characterised properties, it is unsuited to our problem.

With the limitations of the services provided online, the only viable alternative is processing using local software. The most common programs used to view protein models are DeepView, also known as Swiss-PdbViewer (Guex and Peitsch 1997) and Protein Explorer, an adaptation of Rasmol (Martz 2003).

Of the two, DeepView provides a more comprehensive environment and is designed with a focus on protein modelling rather than as an educational tool. It supports BLASTing of a sequence against either a derivative version of the PDB called ExPDB, or the SwissProt (Boeckmann et. al. 2003) databases, with resulting structures selectively downloaded to the viewer and aligned in turn. A PROSITE search will then locate any patterns within the sequences of each structure returning a list of matching patterns that can be selectively highlighted on the structure one at a time.

For our purposes, there are three main limitations with this approach: no provisions for the inclusion of additional online or local databases; necessary maintenance of a local PROSITE pattern database, and an inability to simultaneously highlight multiple patterns on a structure.

These existing software limitations suggest the necessity of creating a novel solution for the information-rich view of protein structures we have described.

3 Implementation

3.1 Selecting Appropriate Software

In the interests of efficiency, we focused on extending existing viewers. DeepView, while possessing a scripting language, does not appear to be flexible enough to add new functionality, and more importantly does not allow the alteration of basic program operations. While it is provided without a fee, it cannot be extended without changing its code, code that we lack access to. As such we either have to write our own viewer or modify existing open-source software.

Although versions of RasMOL do exist in the public domain, its ~60,000 lines of C source code are provided without a clear API (Application Program Interface) and lacking an intuitive structure for performing minor modifications. There are two open source Java based viewers, JMVS (Crossley 2001) and Java Mage (Richardson and Richardson 2002), while for Python there is PyMol (DeLano 2002). PyMOL, in addition to having by far the best facilities for generating images, is offered under a BSD-like license, permitting use of its code for any purpose, on the understanding that appropriate acknowledgement will be present within derivate software. Unlike RasMOL, PyMOL also provides a full Python interface for programmers.

While PyMol provides some facilities to build and alter models, it provides almost no tools for performing analyses, hence the use of the Biopython (Chapman and Chang 2000) library. Biopython and related Bio* libraries for other languages arose from a difficulty in writing and maintaining code to interface with the various bioinformatics resources, such as BLAST, GenBank, and the like, with an active community forming to share the work such that the community can benefit. Once we had PyMol and Biopython installed we set about adding the desired functionality to this environment.

3.2 BLAST

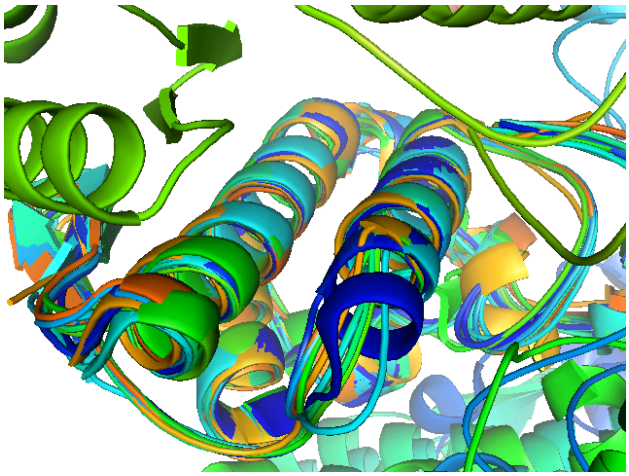
The initial problem simplified to this – given a PDB accession, display the structure, perform a BLAST search using the accession and overlay the top hits, downloading the structures as needed. For preliminary testing, all new code was executed from PyMOL's command line interface via the 'run' command.

Given a PDB accession, we wish to download and display the structure. The simplest way to do so is via the PDB website and as Python can work transparently with gzip compressed files the following PDB link was chosen: http://www.rcsb.org/pdb/cgi/export.cgi/1POD.pdb.gz?format=PDB&pdbId=1POD&compression=gz. From the URL it is apparent that the *pdbId* field value is simply the name of the entry, and downloading the files is a simple matter using the *urlretrieve* function from the *urllib* module. By default, all downloaded files are cached, speeding-up future analysis. Python's *gzip* module is then used to access the file's contents and the command importing.read_pdbstr(compressedPdb.read(),structureId) used to load the data into PyMOL.

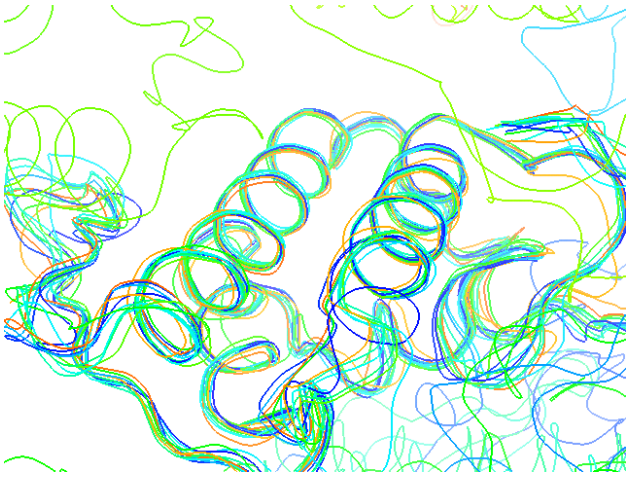
To perform a BLAST search, the function *blast* from *Bio.Blast.NCBIWWW* was used, again caching the results. For now PyView only BLASTs a PDB accession rather than a sequence, sequence functionality was added at a later stage. After using the *blast* module we discovered that Biopython's BLAST parser worked only in certain situations. Fortunately this fault was corrected in Biopython version 1.21 and updating from version 1.1 fixed this. Upon completion of the BLAST search, the user is presented with a list of the top ten hits and their E-values. The user selects from these by specifying in a string either the positions of the entries in the list, the accessions, or simply saying 'all'.

The selected structures are then retrieved, loaded into PyMOL and aligned against the query structure. The alignment function uses cmd.align(id, self.base) from the PyMOL API, where *id* is the current model being aligned and *self.base* is where the name of the query structure is stored. Query names from previous steps are also stored, along with any other information that might be reused subsequently. This saves the user from re-entering information unnecessarily, but the option is still provided for each command to be fully specified in case the cached values are not needed. Unfortunately PyMOL's 'align' command does not appear to be the most robust; possibly requiring the use of STAMP by Russell and Barton (1992), or another means of aligning structures in a later revision. However this is unlikely to be a major concern in this case as the structures being aligned are generally quite similar.

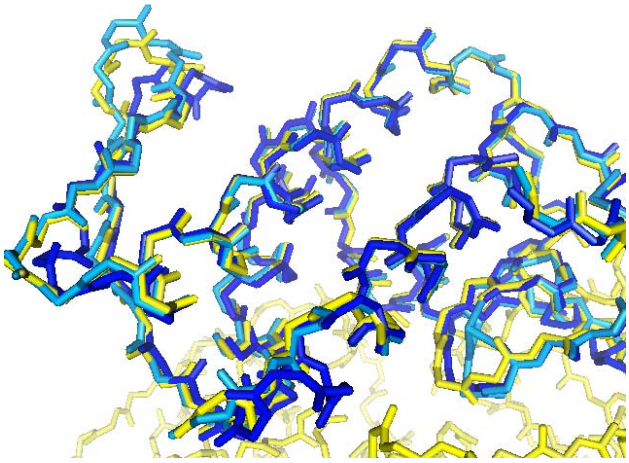
With the selected structures downloaded and aligned the challenge moves to finding a meaningful representation of the structures. A few of the initial attempts at this are shown in **Figure 1**, however the most useful representation will most likely depend on the purpose of visualising the structures and this will become apparent with time.



(a) cartoon representation



(b) ribbon representation



(c) sticks representation

Figure 1: Overlay of related structures on the query 1POD, shown in dark blue. A BLAST search was performed against the PDB and a number of the top hits were overlaid against 1POD, the query. (a) and (b) show the ten best matches in addition to 1POD, while (c) shows the top two, 1C1J (yellow) and 1JIA (light blue).

3.3 PROSITE

At this stage only general structural information is displayed, the next step involves locating and highlighting motifs on the structure.

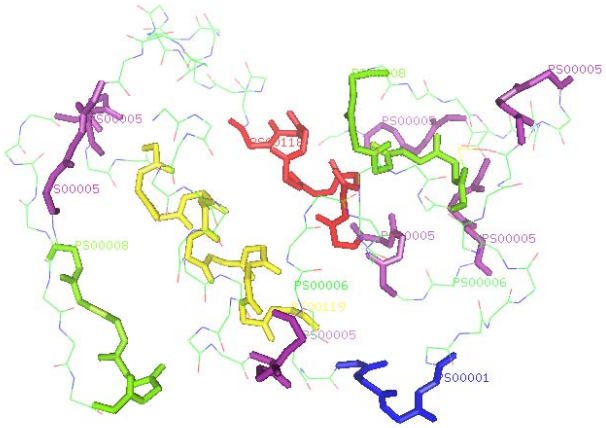
Before a search for PROSITE patterns in the structures can be performed, the sequence information contained within the PDB files is required. This does not appear to be accessible from within PyMOL, and so a PDB parser from Schuerer's and Letondal's online course notes (2002) was used. This PDB parser does not convert the three-letter animo acid codes to the more commonly used single-letter codes, requiring the addition of a new function to perform this translation. The translator, which uses a dictionary, Python's map structure, ignores anything not recognized as a valid three-letter amino acid code, such as HOH which represents a water molecule in the PDB file rather than a part of the structure.

After obtaining the sequence the next requirement was access to PROSITE. While Biopython provides code for this purpose, it is incompatible with the current release of PROSITE, 18.5. After comparing the current web form with the code in Biopython it became apparent that although the URL was listed correctly, the field name for the sequence is case sensitive and the current PROSITE release has changed "SEQ" to the lower case "seq". This was fixed in the file C:\Python22\Lib\site-packages\Bio\WWW\ExPASy.py allowing PROSITE search results to be downloaded, although the results were not yet able to be parsed.

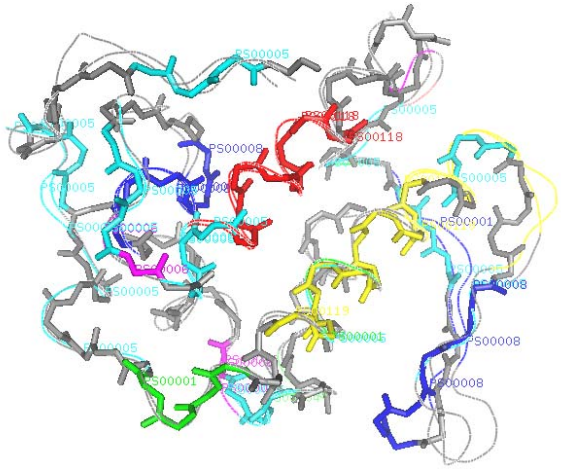
For reasons of efficiency we elected to write a new parser instead of attempting to fix the existing one. The pre-existing parser was written to parse HTML output and disregarded the availability of plain text output that proved trivial to parse. Debugging the existing code could have taken a significant amount of time as it attempted to retrieve the information via a much more difficult route and consequently could also prove more difficult to maintain in future. The functions of the existing parser will be re-examined later, and if our own code proves a suitable replacement it will be submitted for inclusion in Biopython as our contribution to the collaborative effort.

Up until this point the PDB parser and PROSITE interface code had been developed outside of PyMOL, and the first attempt to integrate them with PyMOL proved problematic. The difficulty arose from PyMOL and Biopython wishing to use different versions of the same library, versions 20.0 and 23.0 of *Numeric* respectably. This was resolved by replacing instances of *Numeric* in the PDB parser with *Numarray*, *Numeric*'s replacement.

After successfully integrating the PDB parser with PyMOL, selections were automatically defined in PyMOL to represent each of the PROSITE patterns found in the sequence. Each selection was coloured, with motif representations in the primary structure changed to sticks (also known as cylindrical bonds) for emphasis. Finally the code was modified to work with multiple structures, generating results such as those seen in **Figure 2** (b).



(a) 1POD



(b) 1POD, 1BBC and 1JIA

Figure 2: 1POD is shown using the sticks representation. In (b) 1BBC and 1JIA are shown using the ribbon representation, and the structures are coloured grey where no domain matched. In both images the instances of PROSITE patterns are coloured and labelled with the pattern's PROSITE accession code.

4 Ongoing Work

There are three main areas which stand to benefit most from additional work. A greater variety of input formats, input sources and processing options can be added, for example SCOP (Murzin et. al. 1995) and ProDOM (Servant et. al. 2002) as sources of data and Dali (Holm and Sander 1993) as an alternative to BLAST; secondly the interface of the program and presentation of the results can be improved, we can extend the existing GUI, and optimise the presentation of the structures; finally PyView itself can be integrated into an existing bioinformatics system, such as our semi structured Protein database project, described by Shui et. al. (2003).

The addition of a structurally-based search module such as Dali, would complement the sequence-based search strategy utilised by BLAST, and may yield more accurate results. One significant consideration in adding Dali is the additional computational requirements of such a technique, as a search in Dali can take anywhere from 10 minutes to 10 hours.

A command line interface, while relatively simple to program and preferred by some users, is rarely appealing to the novice, favouring an extension of PyMOL's existing GUI to incorporate our additions. Ideally we will develop a 'plug-in' interface which will allow programmers to develop additional PyMOL modules without worrying about an explicit link from PyMOL's GUI. It would also allow users to code their own functions using generic hooks from the GUI such that if PyMOL's current Tcl/Tk interface is replaced, there will be little or no need to update the plug-ins.

When we look at a structure for the first time our first action is usually to rotate the structure to the view that we find most informative. This simple action turns into a challenge when one structure becomes ten structures overlaid on top of one another, the complexity increases once annotation of the structures is performed. To address this challenge we seek to develop a better understanding of the processes involved in simultaneous visualization of multiple structures. If we can generalize what makes an informative view we can then attempt to automatically orientate structures in Pymol.

One component of this approach involves determining what information can be discarded; are people interested in seeing the complete structures if we need to use a ribbon representation? Would they be more interested in examining only the differences between aligned structures? What about viewing only two aligned structures at a time and cycling through all the structures, changing once every second? Most likely it will be a representation between these extremes; viewer default settings under different circumstances are also something we wish to investigate.

Question then arises of how the view should be orientated once an appropriate representation of the structures has been chosen. Sverud and MacCallum (2003) have also examined this problem, however it is unclear if their use of dimension reduction is appropriate and it is likely computer vision techniques will need to be used.

Finally we are working on integrating this project as a visualisation interface for our semi-structured Protein database project. The database can be used as a gateway that can hook into different bioinformatic data sources, allowing us to use a single interface to access any type of similarity search, and another to utilise any type of protein motif database.

The integration of heterogeneous biological databases is not a simple task due to the complicated schema transformations required for each database. A further complication is that many mappings are required between components. Integrating heterogeneous databases also requires the identification of information that is implicitly or explicitly shared. A global schema should be general enough to handle all manner of heterogeneous data models. In these types of situations, the strict schema constraint of relational database model becomes problematic.

Shui et. al. (2003) presented a framework based upon eXtensible Mark-up Language (XML) databases that would allow integrated storage, processing and retrieval of biological data. The framework also supplies an automated cascading mechanism for the updating of data between mapped databases. It proposed a new approach for using an XML database system called SODA as the physical storage unit and a map between heterogeneous biological data and a more open XML meta-data format for exchange and integration. It also uses active rules to produce and maintain extended information from the meta-data.

However, several challenges are posed in order to integrate the visualisation interface for SODA. The most important one is regarding the efficiency and optimization issues. The SODA database is primarily designed as a database engine for text-based XML output and rendering the query results from SODA for visualization will be too demanding for the database engine unless appropriate optimization mechanisms are in place.

Alternate approaches include the projects BioMoby (Wilkinson and Links 2002) and MyGrid (Stevens, Robinson and Goble 2003). The BioMoby project follows the web-services paradigm in integrating bioinformatics resources, while the MyGrid project focuses on capturing and reusing workflows and the application of Grid technologies to bioinformatics. While these projects have similar goals, a unified view of bioinformatics resources, each differs markedly in their approach, meaning it may be worthwhile to examine separately how PyView can fit into these other projects.

5 Example

To demonstrate how overlaid structures can be used to assist biological understanding, an examination of the Phospholipase A_2 family follows.

The Phospholipase A_2 family includes Ca^{2+}-dependent enzymes which fall into 2 major categories: low molecular weight, disulfide linked, highly α-helical structures and high molecular weight cytosolic PLA_2's ($cPLA_2$) (>60 kDa). PLA_2s are activated on membrane surfaces, hydrolysing the sn-2 fatty acyl ester bond of phosphoglycerides to produce fatty acids, lysophospholipids and subsequent metabolites that are important physiological regulators; each differently composed phospholipid has its own specific alterations. The $sPLA_2$s have a number of interesting features associated with their structure and function, providing useful insights into the relationships of the many accrued experimental structures and their functional/structural regions. (For more extensive reviews of the phospholipase A_2 enzymes see Six and Dennis, 2000 and Murakami and Kudo, 2002).

The $sPLA_2$s have relatively conserved active domains and Ca binding sites, but aspects of the mechanism of action of the enzyme such as its activation is of great interest. Substrate is cleaved in a hydrophobic channel, the end of which contains the active site Histidine and Aspartic acid, with the Calcium site adjacent. Many cells respond to Calcium ions as an extracellular stimulus. Binding of Ca by Calmodulin (calcium modulating protein) effects conformation changes that enhance Calmodulin's affinity for a multitude of regulatory target proteins. This enables the release of calcium to trigger events such as muscle contraction. Other enzymes may be Ca concentration-dependent, such as $sPLA_2$, and incorporate Ca binding sites that are intimately linked with catalytic activity. 10 sub-types of $sPLA_2$ have been classified on the basis of a number of criteria first suggested by Dennis (Dennis 1997) that include structural features. These structural considerations are observations such as the number of disulfide bonds, loops and the extensions comprising, for instance, the so-called group I loop and group II extensions.

To demonstrate how PyView can be used to highlight such structural variations, we performed a BLAST search of the human $sPLA_2$ 1POD against the PDB, and overlaid the top ten hits (with varying sequences) onto its structure. The representation of these structures was changed to ribbons and only the primary chains of each structure were displayed; this is shown as **Figure 1** (b). We then examined the differences between the various structures and noted the following:

The largest variation within the group of 10 structures appears in the small β-sheet region, 2 strands sometimes referred to as the β-wing. This region can be involved in a disulfide bond back to the rest of the (α-helical) core. The results suggest that this is the most flexible part of the molecule.

The next largest variation is seen in the "C-terminal extension". This is a series of approximately 8 residues that have been used to distinguish the PLA_2s (initially used to describe an extension seen with group II), but that also appear flexible.

A number of differences are immediately obvious in Piratoxin III from *Bothrops Pirajai* (1GMZ). The loop at the end of the long helix containing the active site Asp (Helix D) is in a markedly different conformation. Also marked and perhaps functionally more significant is the obvious difference at the Ca binding loop, which in 1GMZ is not actually seen to be binding Calcium, and may therefore be responsible for this anomaly.

Another interesting deviation seen in the two human $sPLA_2$s, 1BBC and 1POD is a variation in the loop at the end of the active-site histidine-containing helix (helix C). An Arginine is in a markedly different conformation in 1BBC, which in 1POD, while still different, is closer to the other example PLA_2s.

6 Conclusion

In summary, the contributions of this project can be summarised as follows:

- Provided an example of how a bioinformatics library, Biopython, can be used to extend a narrowly-focused end user application, PyMol. Outlined the strengths of this approach as well as some issues that may arise.

- Created a tool to inspect the quality of a 3D model built via homology modelling, for example you would expect that a structural model to be rejected if the required carbohydrate binding site was found to be buried. Whilst unsuited for use in a high throughput environment, it can be used to assist in the development of an automated tool to judge 3D models.
- Provided a simple means by which high quality structural images can be generated with the location of domains highlighted, suitable for use in teaching and publications.
- Given users of PyView the ability to generate a 3D structural alignment for a similar amount of effort as a standard sequence alignment, e.g. providing an alternate step in protein classification.

This paper also serves several secondary purposes:

- Illustrated the difficulties in displaying large amounts of structural information as well as providing some thoughts on how these issues can be addressed
- A survey of existing visualisation software and the current state of open-source software with regards to bioinformatics
- Demonstrated an interactive environment in which BLAST result can be shown in 3D, with the options of highlighting multiple domains on a structures

In the creation of PyView we have suggested one approach to visualisation in bioinformatics software. While PyMol by itself is limited in terms of what it can offer the bioinformatics community, we have shown that its use, in conjunction with other freely available software, enables the creation of powerful, graphical bioinformatics tools. It has made high quality modular visualisation accessible to bioinformaticians without previous visualisation experience, supporting its use within software ranging from single scripts to large discovery platforms.

7 Availability

After all ongoing works are completed PyView will be made available for public use from http://pyview.org. Meanwhile the current intermediate version can be obtained by contacting us directly.

8 Acknowledgements

We thank Johann Lenffer for his contributions, his assistance in the writing and revision of this manuscript has been invaluable.

9 References

Adams, P.D., Grosse-Kunstleve, R.W., Hung, L.W., Ioerger, T.R., McCoy, A.J., Moriarty, N.W., Read, R.J., Sacchettini, J.C., Sauter, N.K. and Terwilliger, T.C. (2002): "PHENIX: building new software for automated crystallographic structure determination." *Acta Crystallogr D Biol Crystallogr* **58**(11): 1948-54.

Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990): Basic local alignment search tool. *Journal of Molecular Biology* **215**:403-410.

Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J. and Wheeler, D.L. (2003): GenBank. *Nucleic Acids Research* **31**(1):23-7.

Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N. and Bourne, P.E. (2000): The Protein Data Bank. *Nucleic Acids Research* **28**:235-242.

Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.-C., Estreicher, A., Gasteiger, E., Martin, M.J., Michoud, K., O'Donovan, C., Phan, I., Pilbout, S. and Schneider, M., (2003): The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research* **31**:365-370.

Buneman, P., Crabtree, J., Davidson, S. B., Overton, C., Tannen, V. and Wong, L. (1998): BioKleisli. In *Bioinformatics*, Letovsky S., (ed). New York, Kluwer Academic Publishers.

Chapman, B. and Chang, J. (2000): Biopython: Python tools for computation biology. *Sig-Bio Newsletter of the Association for Computing Machinery* **20**(2):15-19.

Chung, S.Y. and Wong, L. (1999): Kleisli: a new tool for data integration in biology. *Trends in Biotechnology* **17**(9):351-5.

Crossley, A.D. (2001): Java3D Molecular Visualisation System: http://www.dcs.shef.ac.uk/aajc/jmvs/info.htm. Accessed 3 Sep 2003.

DeLano, W.L. (2002): The PyMOL Molecular Graphics System. DeLano Scientific, San Carlos, CA, USA. http://www.pymol.org. Accessed 4 Sep 2003.

Dennis, E.A., The growing phospholipase A2 superfamily of signal transduction enzymes. *Trends Biochem Sci.* **22**(1):1-2.

EBI Tools - WSDbfetch. http://www.ebi.ac.uk/Tools/webservices/. Accessed 2 Sep 2003.

Hubbard, T., Barker, D., Birney, E., Cameron, G., Chen, Y., Clark, L., Cox, T., Cuff, J., Curwen, V., Down, T., Durbin, R., Eyras, E., Gilbert, J., Hammond, M., Huminiecki, L., Kasprzyk, A., Lehvaslaiho, H., Lijnzaad, P., Melsopp, C., Mongin, E., Pettett, R., Pocock, M., Potter, S., Rust, A., Schmidt, E., Searle, S., Slater, G., Smith, J., Spooner, W., Stabenau, A., Stalker, J., Stupka, E., Ureta-Vidal, A., Vastrik, I. and Clamp, M. (2002): The Ensembl genome database project. *Nucleic Acids Research* **30**(1):38-41, Oxford University Press.

Gasteiger, E., Gattiker, A., Hoogland, C., Ivanyi, I., Appel, R.D. and Bairoch, A. (2003): ExPASy: the proteomics server for in-depth protein knowledge and analysis. *Nucleic Acids Research* **31**:3784-3788.

Growth of the Sequence and 3D Structure Databases, GenomeNet. http://www.genome.ad.jp/dbget/db_growth.html. Accessed 3 Sep 2003.

Gouet, P., Robert, X. and Courcelle, E. (2003): ESPript/ENDscript: Extracting and rendering sequence and 3D information from atomic structures of proteins. *Nucleic Acids Research* **31**(13):3320-3.

Guex, N. and Peitsch, M.C. (1997): SWISS-MODEL and the Swiss-PdbViewer: An environment for comparative protein modeling. *Electrophoresis* **18:** 2714-2723.

Holm, L. and Sander, C. (1993): Protein Structure Comparison by Alignment of Distance Matrices. *Journal of Molecular Biology* **233**:123-138.

Kent, W. J., Sugnet, C. W., Furey, T.S., Roskin, K.M. Pringle, T. H., Zahler, A.M., Haussler, D. (2002): The Human Genome Browser at UCSC. *Genome Research* **12**(5):996-1006.

Marti-Renom, M.A., Yerkovich, B. and Sali, A. (2002): Comparative protein structure prediction. Ed: John Wiley & Sons, Inc.. *Current Protocols in Protein Science* 1:2.9.1-2.9.22.

Marti-Renom, M.A., Madhusudhan, M.S., Fiser, A., Rost, B. and Sali, A. (2002): Reliability of assessment of protein structure prediction methods. *Structure* **10**(3): 435-40, Cambridge.

Martz, E. (2003): Protein Explorer FrontDoor. http://molvis.sdsc.edu/protexpl/. Accessed 2 Sep 2003.

Murakami, M. and Kudo, I. (2002). Phospholipase A_2 *Journal of Biochemistry* **131**:285-292, Tokyo.

Murzin, A.G., Brenner, S.E., Hubbard, T., Chothia, C. (1995): SCOP: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology* **247**:536-540

RasMol Home Page. http://www.umass.edu/microbio/rasmol/. Accessed 1 Sep 2003.

Richardson, D.C., Richardson, J.S. (2003): Java Mage: Kinemages in your web browser. http://kinemage.biochem.duke.edu/javamage/java.html. Accessed 3 Sep 2003.

Russell, R.B. and Barton, G.J. (1992): Multiple protein sequence alignment from tertiary structure comparison: assignment of global and residue confidence levels. *Proteins* **14**(2):309-23

Schuerer, K. and Letondal, C. (2002): Python course in Bioinformatics. *Pasteur Institute.* http://www.pasteur.fr/recherche/unites/sis/formation/python/. Accessed 30 Aug 2003.

Servant, F., Bru, C., Carrère, S., Courcelle, E., Gouzy, J., Peyruc, D. and Kahn, D. (2002): ProDom: Automated clustering of homologous domains. *Briefings in Bioinformatics* **3**(3):246-251

Sigrist, C.J., Cerutti, L., Hulo, N., Gattiker, A., Falquet, L., Pagni, M., Bairoch, A. and Bucher, P. (2002): PROSITE: a documented database using patterns and profiles as motif descriptors. *Briefings in Bioinformatics* **3**:265-274, London, H. Stewart Publications.

Sali, A. (1998): 100,000 protein structures for the biologist. *Nature Structural Biology* **5**(12): 1029-32.

Shui, W. M. et al (2003): A New Approach to Protein Structure and Function Analysis Using Semi-Structured Databases. *Proceedings of the First Asia-Pacific Bioinformatics Conference (APBC)* 61-70.

Stein, L. (2002): Creating a bioinformatics nation. *Nature* **417**:119 – 120

Stevens, R.D., Robinson A.J. and Goble, C.A. (2003): myGrid: Personalised Bioinformatics on the Information Grid. *Bioinformatics* **19(**Suppl. 1):I302-I304.

Sverud, O. and MacCallum, R.M. (2003): Towards optimal views of proteins. *Bioinformatics* **19**(7):882-888.

Wilkinson, M.D. and Links, M. (2002): BioMOBY: an open-source biological web services proposal. *Briefings in Bioinformatics* **3(**4):331-341.