DOCKside – An Interactive system for manual docking of molecules into Cryo-Electron Microscopy Maps
Technical Paper CS400
Department of Computer Science
University of Cape Town

Guy Stern gstern@cs.uct.ac.za

Andrew Snowden asnowden@cs.uct.ac.za

Dr. Michelle Kuttel mkuttel@cs.uct.ac.za

Dr. James Gain jgain@cs.uct.ac.za

Abstract

The process of cryo-electron microscopy allows scientists to view the complex structures of proteins as they bind and interact with one another. This process however, outputs low resolution noisy density maps which in their initial form are of little use. Through a process called *docking*, high resolution models of each of the interacting components can be fitted into these low resolution maps so that further study can occur.

DOCKside allows users to interact with 3D representations of the proteins and of the electron density maps. Design was an initial concern. Related work was studied to better create a efficient solution to the problem. Different techniques for visualising the various components are implemented and discussed. Docking can also be performed manually using interactive graphics or automatically using a range of mathematically intensive algorithms. These too are detailed and discussed. Through user testing, a review is made as to how efficient the docking process is in producing meaningful and accurate data when compared to the automatically docked solutions.

1. Introduction

One of the most important components of our physical world is the arrangements of chemical structures. There are many molecules such as proteins which are known to science but whose molecular conformation is not known. Without knowledge of the exact three-dimensional structures, scientists cannot fully understand the way molecules bind and interact with one another. For example, knowledge of the 3D structure of a virus helps in finding out how it interacts with molecular receptors on cells in the body. On the basis of this knowledge, drugs can be designed to interact with these viruses or cells to possibly prevent infection.

The problem is as follows: For many large molecules and molecular assemblies it is only possible to capture limited 3D shape information using electron microscopy techniques. The data comes in the form of electron density maps, which provide some structural information as to the different densities at different parts of the molecule/protein. However, these *images* are typically of a low resolution and suffer from noise. They have no fine detail and therefore do not give enough information about the, location of the atoms and hence, real structure of the artifact in question.

One method of finding a more exact image of a protein is via crystallography. By focusing an X-ray through protein crystals, the refraction patterns can be used to determine a much higher resolution picture of the protein. The problem with this method, however, is that the crystal environment often changes the structure of the proteins. In addition, crystallography is often not applicable where more complex structures need to be analyzed. Large structures become unstable and often their form is different under crystallization .

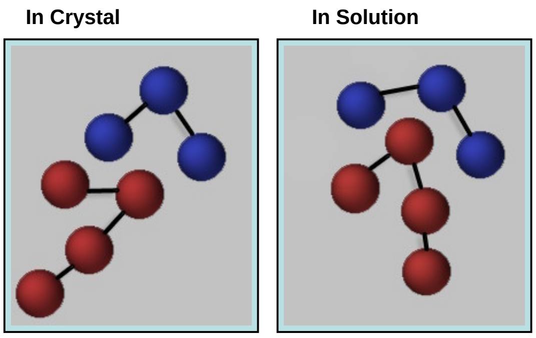
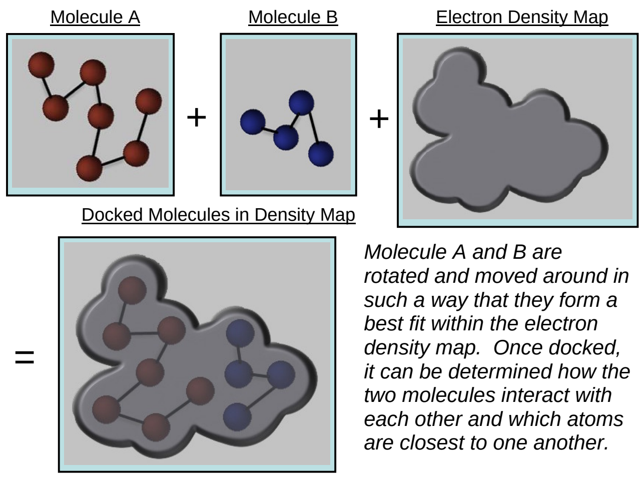


Figure 1.1 – Change in atomic structure

The solution is therefore to combine the two methods. Through electron microscopy (EM), a low resolution image of a molecular complex is found, giving a rough outline of the structure. And through crystallography, high resolution images of the different molecular components can be elucidated. Now the task that remains is fitting the smaller components into the bigger compound. This is the undertaking of this project.



Molecule A and B are rotated and moved around in such a way that they form a best fit within the electron density map. Once docked, it can be determined how the two molecules interact with each other and which atoms are closest to one another.

Figure 1.2 – Docking process

1.1 The files used

Computer technology is widely used in every-day chemical sciences. Standard protocols are in place for storing the empirical data collected. This project will use to two files types, produced through laboratory investigation, common to the field. The first of the files represents the System for Processing Image Data in Electron Microscopy and Related fields (SPIDER). SPIDER files are used to store volumetric data recorded during the EM phase.

The program will also takes in Protein Data Bank (PDB) files which will represent the higher resolution components. The Protein Data Bank is an online repository for files containing atomic information about different chemical structures. Most important of these are proteins. As with SPIDER files, PDB's follow a strict code of formatting. They have many uses and can be opened by many different applications. Proteins are substances made of a chain of amino acids. The amino acids form a long chain often referred to as to polypeptide backbone. PDB files store the amino acids in the order in which they appear in this chain. Within these amino acids, the files store the names and 3D locations of each of the atoms making up the residue.

1.3 Project Structure

The program then has two main functions that allow the user to visualize the final protein. The first function is manual docking. Through manual docking, the user will move the PDB components around and attempt to fit them into the EM-image. This is similar to the way someone might attempt to complete a puzzle given the pieces. The result is not be exact and is, in many cases quite difficult for a user. Therefore a second method of molding the images will be an automated one. Here the program will use complex mathematical procedures to find the best fit of the components to the EM. The two components are usable by means of a shared graphical user interface.

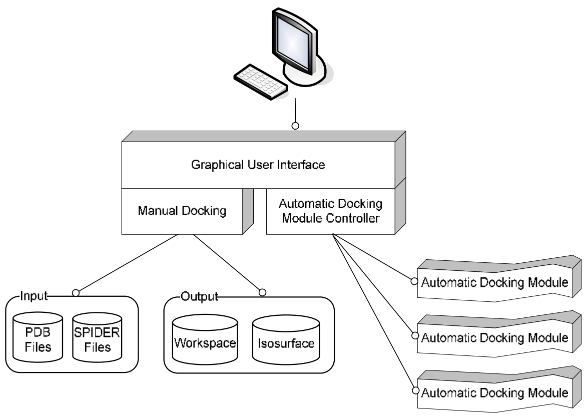


Figure 1.3.1 – High level project structure

1.4 Success factors

Whilst it may not have been possible to create a perfectly functional docking suite, there were some key success factors agreed upon at the outset of the project. The first of these was that there had to be a working graphical user interface in place. This required that both the PDBs and the SPIDER needed to be in some way visualized.

The second success factor with regards to the interface was that it be usable. Working with and manipulating complex shapes in a three-dimensional environment is a skill, which does not come naturally to all computer users. Training books and tutorials are often needed to familiarize one with the environment. In this project, however, the tasks that are required to be performed by the user are relatively simple.

The last of the key success factors, was that there needed to be a system in place for checking the validity of results. This meant that the results of the docking needed to be saved in such a way that they could be checked against an existing solution. In addition, this meant that results of the automatic docking process needed to be visualized in the same interface as the automatic docking system.

With respect to the automatic docking itself, the goals in mind were that, we achieved a correct implementation of two automatic docking techniques. Since there are many available docking techniques, the first design decision was which techniques to implement. While vector quantization offers near real time docking, its scope of application is severely limited. Contour based docking [3] produces highly accurate results and with modern computer hardware can be accomplished in a relatively short time. Similar to contour based docking is the principle of standard correlation fitting. Correlation fitting is one of the first techniques to be used and provides an interesting technique to compare against.

It was also thought important that the automatic docking module interface with the graphical interface in such a way that it was easily extendable to future work. In other words, developers should be able to code new automatic docking modules without needing to change the graphical system.

2 Related Work

2.1 Automatic Docking

The majority of automatic docking methods follow the same principle: low resolution density maps are generated from the atomic structure (high level data) [1], the algorithm then attempts to match a generated map with the map obtained from cryoEM techniques. This density map is usually calculated estimated using a Gaussian kernel to interpolate data from the atomic structure. Once we have these two comparable maps the problem is to find the correct rotation and translation in order to superimpose the two. Several complications arise from the differences in EM and crystallography techniques [1].

1) EM maps are often noisy and can be distorted by a number of factors
2) Some molecules may change structure when in crystalline form
3) The magnification obtained when using electron microscopes is not precise and as such the size of molecules in density maps may not be accurate [4]. If there is a large variation then this inappropriate scaling may fault the docking process.

Despite these problems, it has been shown that, in most cases, electron microscopy and crystallography produce nearly identical images [4]. In addition, electron microscopy has the advantage over many techniques in the fact that it can image the entire structure and not just the surface of a molecule [1]. The successful combination of these techniques can result in models with an accuracy of 4-5Å from a 20Å density map.

2.1 Molecular Visualisation

In order to visualise the surfaces of the proteins and of the density map, an iso-surface needs to be generated. Here the input is volumetric data and the output should be in the form of a surface mesh.

In [5] an algorithm, called '*marching cubes,'* for visualising 3D volumes is detailed. The process works by splitting up the volume into cubes. Each cube can be inspected and each of the eight corners can be seen as being either inside or outside of the object. The result is a mesh of triangles which is easy to manipulate, scale and rotate. In [6] a method is detailed regarding reducing the number of triangles created from the marching cubes algorithm. It proposes using octrees to store the cells and cells which are part of a relatively flat area are merged so long as the error caused by merging is less than a user-specified ratio. Many cells can be merged with neighbours without affecting the shape of the resulting mesh in any way.

Programs such as [7] are designed to visualize the PDB files in 3D so that the atoms are seen as small spheres and the bonds between them as lines. This is commonly referred to as the *ball and stick* model. In [8] it is explained that volume rendering, iso-surface extraction and virtual reality each do not work properly in successfully visualizing microscopy data. But by combining the three approaches, users can get a good understanding of the data. To prove this hypothesis, they use a process called immunofluoresecence microscopy to establish locality of proteins during cell - cell interactions. They fluorescently tag antibodies of a particular protein and then this protein lights up under the cover slip of a cell interaction and the protein's exact location can be seen.

The images, obtained using a digital camera, are deconvolved before use. The reason that deconvolution is needed is that light creeps into the image from many sources and these all need to be physically modeled. Using a complex algorithm the exact light contributions are found so that they can be subtracted from the raw data.

3. Design

3.1 The Open Source nature of the project

One of the requirements of this project was that it be extendable to an Open Source framework. This meant that in addition to the application needing to be functional by the end of the project, it would also have to be easily extendable. Therefore, early on in the design phase, care needed to be taken to create objects in such a way that they would be modular, reusable and replaceable. Interfaces between the classes needed to be structured and limited to the function of the program itself. The code also needed to be written in a way that it would be easily understood so as to be modifiable by future developers. This meant that care needed to be taken to create meaningful variable names and comment code. Being Open Source meant that all libraries and classes used by the project needed to be non-commercial.

3.2 Molecular structure

One of the initial design decisions was to have the molecular classes, work in a similar way to the atomic structure of the proteins themselves. It was therefore sensible to start the design by creating the simple atom objects and working outwards building complexity.

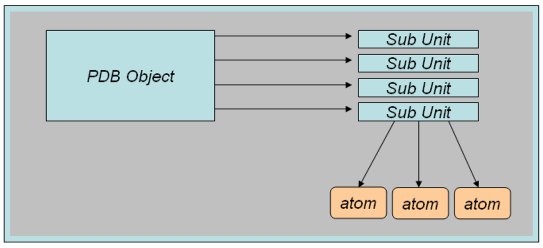

PDB Object
Sub Unit
Sub Unit
Sub Unit
Sub Unit
atom
atom
atom
One PDB object has many sub-units and each sub unit has a collection of atoms.

Figure 3.2.1 – Molecular class association structure

3.3 The Solution Set approach

Once design of the basic components, namely the molecules, was in place, we needed to have a system in place for storing results from docking. Through manual and automatic docking, the sub units would be rotated and moved into a configuration and this configuration needed to be continuously updated. The image below illustrates how sub units' values are transformed before output.

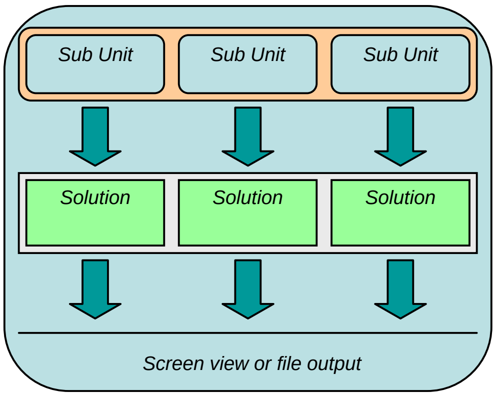


Figure 3.3.1 – Solution set class structure

4. Implementation

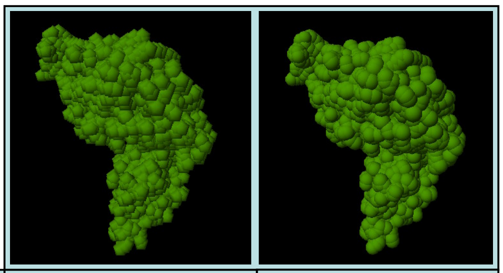
4.1 Visualisation

Effort was taken to find methods which would execute in real time and adapt to the input provided. For this reason, we decided that openGL would be the perfect platform on which to write the code. OpenGL is a highly portable graphics based language, developed by the Open Source community. It is widely used and is therefore readable and understandable to many computer scientists. In the following section we will outline the process taken to accomplish the visualisations found in the final version of the application.

4.1.1 Visualising the PDB

4.1.1.1 The ball model

Most PDB viewing applications have an option to represent the atoms as spheres. By doing so, one begins to see the molecule not as a collection of joining lines, but as a single object. We accomplished this using the *glSphere* primitive. For each sphere, the program had to translate to the relevant place in world coordinates to draw the object and then return to its previous place. This required the use of many *push* and *pop* matrix operations, which made the ball drawing procedure somewhat inefficient. Level of detail of the spheres was a performance consideration.



	Low quality		High quality	
Level of detail:	5 x 5	*Appropriate for graphical user interfaces such as manual docking*	9 x 9	*Appropriate for high quality images and publication*
Time to create:	0.5 Seconds		6 Seconds	
Refresh rate:	6 per Second		4 per Second	

Figure 4.1.1.1.1 – Comparison of level of detail for spheres in molecular model

4.1.2 Atom Colouring

Atoms were coloured according to a conventional atom colouring system. Each atom object had an integer colour value ranging from 1 to 7. This number was stored in the atom object. At the time of drawing, the atoms were coloured according to their colour value. A table of these colours is show below:

1) Carbon	●	Green
2) Oxygen	●	Red
3) Nitrogen	●	Blue
4) Sulphur	●	Yellow
5) Hydrogen	●	White
6) Phosphorus	●	Purple
7) All others	●	Grey

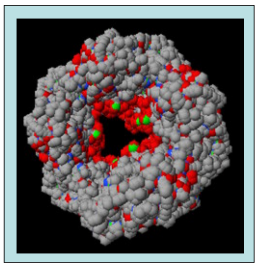


Figure 4.1.2.1 – Atom colour reference table and example coloured molecule

4.1.3 Iso-surface visualisation

Docking entails that the one object fits into the space of the other object. As such, it was decided that it was most useful that the PDB look as similar to the SPIDER as possible during docking. The final visualisation which was created for the molecule was therefore, an iso-surface.

In order to convert the molecule into an iso-surface, an interpolation method was used whereby a 3D array representing the atoms was created. Depending on an atom's proximity to each corner of the voxel, each of the 8 positions in the array surrounding the particular atom was incremented by the relevant amount.

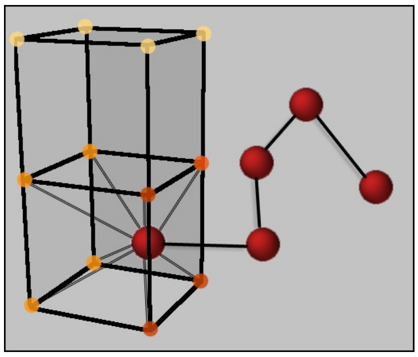


Figure 4.1.3.1 – Atoms are interpolated in a grid

This is a basic model of the density distribution of the atoms which does of course not take all factors into account. This array was then passed to the marching cubes method for iso-surface visualisation. Iso-surface representation allows the user to see the shape of the molecule whilst hiding the atomic complexity. Below is the resulting iso-surface visualisation for a protein. This can be compared to its coloured ball model equivalent. The first image shows the iso-surface itself and the second, the same protein in ball representation. The third image shows the ball model overlaid by the iso-surface.

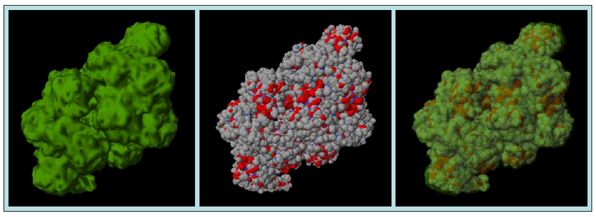


Figure 4.2.1.6.1 – Different visualizations of a complex molecule

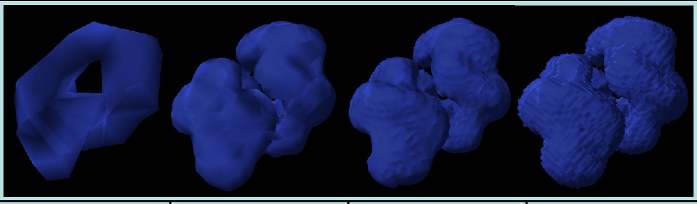
4.1.4 Visualising the SPIDER

4.1.4.1 Iso-surface visualisation

The solution to the problem of visualising the density map as a volume was similar to the approach taken in visualising the PDB. The density map in its entirety was sent into the marching cubes class as an argument. A further argument was the cut-off value for the density. A cut-off value (or iso-value) is particularly important in marching cubes, as it dictates which vertices are inside and which vertices are outside of the view volume.

4.1.5 Marching Cubes

The bulk of the work of the marching cubes class is done when the *runMarchingCubes* method is called. This method samples the array at a user defined amount to produce the isosurface.



10 x 10 x 10 Inadequate for docking.	*30 x 30 x 30 Shape of density field can be seen.*	*80 x 80 x 80 Number of samples is exactly equal to array size. Best representation*	*175 x 175 x 175 The effects of over-sampling can be seen here.*

Figure 4.1.5.1 – Iso-surface of the same SPIDER at different dataset sizes

At user-defined target value is also used. All values lower than this value are outside and all values greater than or equal to this value are in the view volume. A cube is creates and each of its 8 vertices are inspected defined as being in or out of the view volume. The configuration of the vertices is indexed in an edge table. This edge table contains the 256 intersection cases is an array common to almost all marching cubes implementations. In each case where an edge is found, the position along that edge where a vertex will lie is calculated with the *fGetOffset* method which interpolates between the end points of the edge. Finally, the edges are looked up in a predefined triangle table and the mesh is constructed with normals calculated. The mesh is drawn as a collection of openGL triangles.

4.1.6 Transparency

To dock the PDB components within the SPIDER volume, it was initially decide that the volume should be drawn as semi transparent. By rendering it in this way, the user could see where in the volume, the sub unit appeared. .

4.1.6.1 Using back and forward face culling

A problem exists in OpenGL that when blending multiple levels of transparency, the layers are not always properly ordered. Whilst OpenGL does not compute an order of polygons, it does calculate back and forward facing polygons. There exists an elegant solution for the alpha blending problem when drawing perfectly convex objects. It works by first rendering all the back faces and then all the front faces. The following pseudo code illustrates the method:

```
GlCullFace (front faces); // tells openGL to ignore all front faces when drawing
Draw the object // draws only back faces
GlCullFace (GL_BACK); // tells openGL to ignore all front faces when drawing
Draw the object again // draws only back faces
```

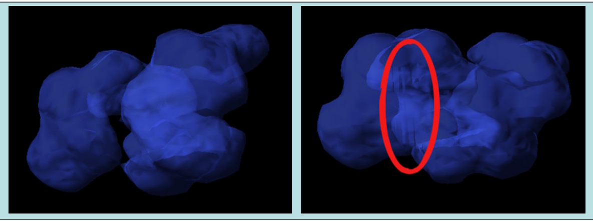


Figure 4.16.1.1 - This method produced best results however, some artefacts remain

4.2. Fourier Accelerated Docking

The two docking techniques that have been implemented rely highly on the notion of correlation. Correlation is a method to determine how closely two data patterns are related. In our case, it is used to determine how similar two density fields are.

If we have two density fields then the simplest way to calculate their similarity is to loop through each point and compare the voxel values from each density field. The common method of comparing these points is to simply multiply them. The larger and closer the values are then the higher the correlation will be. The correlation coefficient can thus be expressed as the following formula:

$$c = \sum_x \sum_y \sum_z \left(A_{x,y,z} * B_{x,y,z} \right)$$

Equation 4.2.1. Correlation Coefficient

In order to find our best fit, we need to calculate this correlation coefficient at every possible translation. Here we notice that this process can be achieved through convolution. If we convolve the one dataset with the other we are left with a dataset of the same size containing the correlation coefficients for all possible translations.

Now unfortunately this calculation is extremely expensive when dealing with 3D spaces and a large number of Euler angles. We can decrease the number of calculations we must perform by using the fact that that a convolution in Real space is the equivalent of a multiplication in Fourier space[2].

The advantage of this approach is that Fourier transforms have been heavily optimized and can execute dramatically faster than convolutions. In our specific case, a Fast Fourier Transform implementation is of the order $N^3 \log N^3$ compared to the N^6 required for the convolution case. To implement this we calculate each correlation using equation *4.2.2.*

$$A \star B = IFFT\left(\overline{FFT(A)} \times FFT(B)\right)$$

Equation 4.2.2. Cross-Correlation

4.3 Laplacian and Gaussian Filtering

In order to transform our PDB into a comparable data type, we use trilinear interpolation to project it onto a regular grid. When we initially interpolate our PDB we are left with a somewhat grainy representation of the density field (*Figure 4.3.1a*).

This representation can be improved by approximating the decaying density around an atoms center by applying a simple Gaussian filter (*Figure 4.3.1b*).
A Gaussian filter averages pixel values according to the values around it. As such, any pixel next to a shaded one will receive some of its shading – hence the effect of extending the density between atoms. This method is only a simple approximation of density, but is sufficient for our comparisons and is computationally efficient.

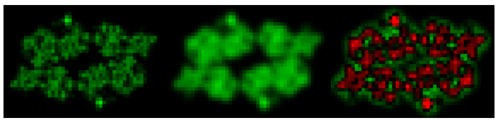

(a) (b) (c)

Figure 4.3.1. Interpolated, Gaussian smoothed and Laplacian Filtered representations of a slice of a molecule

In our system, a standard 3x3x3 Gaussian kernel is order to smooth the interpolated map. Laplacian filtering is achieved through the following equation (Where *np* is the new pixel value and *p* is the original value):

$$np_{x,y,z} = -6p_{x,y,z} + p_{x-1,y,z} + p_{x+1,y,z} + p_{x,y-1,z} + p_{x,y-1,z} + p_{x,y+1,z} + p_{x,y,z-1} + p_{x,y,z+1}$$

Equation 4.3.2. Laplacian Filtering

The equation stated above simply uses the values of the orthogonally adjacent pixel values and the original value to generate the new value. The result of applying this filter can be seen in *Figure 4.3.1c.* All contour (or edge) values result in a positive value while interior points result in positive values (or values close to 0). The rationalization behind this is the following:

- A contour point (outside the object) will have its own pixel (representing density) value set as 0 (or close to it). As such the negative factor will be small while the adjacent points (which are inside the object) will contribute positive factors. The result is a positive value for any point which is partially or fully surrounded by points with greater densities.
- An interior point will have a high density and as such the effect of the -6 factor will be large. The points surrounding it will generally be of the same magnitude (or less) and as such the 6 additions will not balance off the negative influence thus resulting in a negative value.

The effect on correlation is primarily due to the introduction of contour information. The edges of our molecule are now positive and the interior points are mostly negative. This means that our correlation will score higher in places where the edges of the two density maps coincide. This counters the negative result of high correlation in high density areas (regardless of actual corresponding data). In addition, the Laplacian filter converts the representation from displaying density information to showing the relationship between each density and its neighboring densities. After Laplacian filtering, a high density area will not remain as such. The Laplacian image shows fluctuation in densities rather than the actual density itself.

4.4. 6D FFT Accelerated Search

Now that we have established the processes required to implement our Contour Docking [3], we can examine how each of these processes interacts to form the complete system. *Figure 4.4.1.* shows an overview of the interaction and flow between each of these processes. Each process and their implementation will be briefly explained to further understanding. The following discussion covers the process involved in contour-based docking. The implementation of standard correlation is identical except for the removal of all Laplacian filtering.

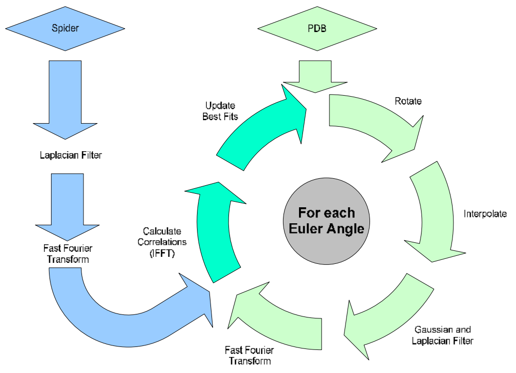


Figure 4.4.1. 6D FFT Accelerated Search Process

When the docking module is called it is initially passed wrapper objects for both the PDB and Spider files. The density field is then normalized so that the values range from 0 to 1 (inclusive). This normalization step is also performed on the PDB data in order to standardize the densities from each data type.

The density field is then Laplacian filtered and finally Fourier transformed .Fourier transforms were implemented by the use of an external FFT library named FFTW (Fastest Fourier Transform in the West). FFTW accomplishes optimized Fourier transforms through the generation of an optimal transform plan and then the execution of this plan on multiple datasets.

The combination of these steps leaves our Spider file in the format necessary to perform our correlation. The preparation of our PDB file is slightly more complicated with the addition of rotation, interpolation and Gaussian filtering.

Before the start of our 6D search, we generate a set of Euler angles [9] by which the PDB must be rotated. For each iteration, the original PDB must be rotated by the next Euler angle set (Phi, Theta and Psi).

In order to achieve the correct rotations we center our object according to the mass weighted mean co-ordinates. This assures us that the molecule will essentially rotate around its center of gravity and not an arbitrary point. In order to actually rotate the object, we generate a rotation matrix from our Euler set and then simply apply this matrix to each point in our point set.

Once the molecule it can then be interpolated onto a regular grid. We then apply a Gaussian followed by a Laplacian filter. The combination of these filters is commonly known as a Mexican Hat filter. At this stage, we then Fourier transform this data in the same way as the Spider data.

We then calculate the inverse Fourier Transform given in *Equation 4.4.2.* This inverse Fourier Transform gives us the correlation coefficients for the given rotation. At this stage we check our correlation coefficients against the best fits found previously. If a correlation coefficient exceeds the maximum coefficient found for any rotation, then it is saved along with the rotation that resulted in the correlation.

$$A \star B = IFFT\left(\overline{FFT(A)} \times FFT(B)\right)$$

Equation 4.4.2. Fourier Correlation

5. Results

5.1 The User Test

In order to evaluate the usability of the different docking methods, a test was designed which would have the users run the program in various ways. In the translation test, users would move the PDBs into place and in the rotation test, they would rotate the objects into the density field.

5.1.2 Extraction of result data

The results of the user test were the docked PDB files. A program was written to extract the results from raw experimental data. It would compare each test file against the model solution . When run with the model solution as an input file, the answer outputted was zero. During testing each file was given a meaningful name based on the subject number and the test type.

```
file: S1boxtrans.pdb
ans: 0.116653
 file: S2boxtrans.pdb
ans: 0.118034
file: S3boxtrans.pdb
ans: 0.218211
file: S4boxtrans.pdb
```

Figure 5.1.2.1 – Sample data from the result file for input box translation

5.1.3 Statistical results

Through analysis of the resulting test data, it was found that translation was significantly faster than rotation. This is attributed to the fact that translation is perhaps more intuitive than rotation.

Within the rotation and translation tests, neither method was significantly faster or more accurate.

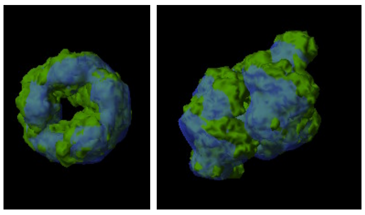
5.1.4 Qualitative Test Results

During the test, suggestions and comments were raised by the subjects. One issue found, was that users often pressed buttons and moved sliders that they were not supposed to use. Sometimes this was out of curiosity and other times this was by mistake. A solution to this problem was the change the entire look of the interface. Tabs were added in such a way that all viewing options for the SPIDER and PDB appear separately from the docking buttons, themselves. As such, users only had access to the required buttons shown in the tutorial. This had the added effect of clearing the interface of complexity improving on the general look and feel of the application.

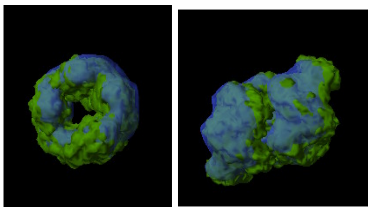
5.2 Automatic Docking Testing

Both automatic docking techniques were run on test data provided by Trevor Sewell. Unfortunately the original placement of the molecule will have an effect on the quality of the docked solution. The reason for this is that the Euler angles are sampled with regular spacing. The sampling may introduce an error up to half the sampling step. Decreasing the sampling step reduces this error but is extremely time consuming due to the exponential nature of the angle generation and long running time of the searching algorithm. Decreasing the sampling step from 30° to 15° increases the number of angles from 1038 to over 7000.

Figure 5.2.1. shows the best fit solution returned by the contour based docking algorithm. A full search spanning the entire rotational spectrum was performed with a 15° (0.261 radians) sampling step. A detailed examination of the best fit showed a near perfect dock with only a slight displacement of the PDB file. This can be attributed to the fact that translations are constrained by the resolution of the Spider file.


(a) (b)
Figure 5.2.1 Contour based docking result from two angles

In order to compare our two docking techniques, the molecule was then docked using standard correlation (with the same parameters). The best fit solution returned by this procedure was not identical to that obtained through contour based fitting. The molecule was shifted slightly backwards in the Y-axis. This gave a slightly less perfect fit as can be seen in *Figure 5.2.2b*. The solution obtained through contour docking was presented as one of the possible fits but did not obtain the highest correlation score.


(a) (b)
Figure 5.2.2. Standard correlation docking result

The most noticeable difference between the results produced by the two different docking procedures was the number of possible fits. The contour based docking presented four possible results all scoring a normalized coefficient of above 0.9. Alternatively, the standard correlation returned over 30 results with very similar correlation scores. The reason for this discrepancy is that contour based fitting penalizes fits when the edges of the shapes do not overlap. This causes a sharp drop in correlation coefficients when the molecule moves outside the density map. This effect is not present in standard correlation and as such the possible results are not as clearly defined.

6. Future Work

In this application, it would be useful, if the user could see the protein as an iso-surface and also see the atom colouring in one visualisation. This could be accomplished by using some variation of the marching cubes algorithm which was extended to interpolate colours in a similar way that it currently interpolates density values.

Due to the nature of the proteins used, it would be ideal, if the SPIDER could be viewed in a way that the inner sub units were a different surface to the outer ones. This would be possible simply by creating multiple meshes, but would only be useful if polygon depth test ordering was implemented. Without this testing, transparency issues mentioned above would cause further problems.

The field of automatic docking is extensive and as such there are many possible extensions to the project. The component architecture was designed so that additional docking techniques could be easily added. As such, the implementation of any of these techniques (Vector Quantization etc.) could provide future work. A genetic algorithm solution seems plausible for both Vector Quantization or as an optional mechanism for contour-based docking.

Vector quantization generates a number of code-vectors for both high and low resolution data. The pairing of these code-vectors is an O(N!) calculation and thus severely limits the number of code-vectors that can be used. A genetic algorithm could be used to pair these code-vectors thus allowing for a much larger number of code-vectors. While it is not apparent if this genetic algorithm would provide more accurate solutions, its implementation could possibly yield more accurate results while still maintaining near real-time computation.

A full 6D search (as required for many docking techniques) is an extremely computationally expensive operation. The distribution of the best results usually lie at local maxima on a reasonably smooth gradient. This indicates that a genetic algorithm could possibly be used to focus searches on areas where the best fit is more likely to occur. This reduction in the search space would allow for much finer angle increments and should yield highly accurate results.

8. Conclusion

By researching the field of bio-informatics and molecular docking, we were able to get a grasp on the exact requirement of the project. These requirements were then translated into a design which was used to implement a working docking application. Different methods of visualization and automatic docking were attempted and documented up to the point that adequate solutions were found to the various challenges. The application is successful in being able to dock molecules both through mathematical automatic docking and by means of manual docking. It is also able to save all results in a way that makes them useful in many applications. The project has been designed in a way that makes it extensible and future work has been suggested so that this project can be taken beyond its current state.

References:

[1] TS Baker, JE Johnson. *Low resolution meets high: towards a resolution continuum from cells to atoms.* Curr. Opin. Struct. Biol, 1996

[2] Ted Schlicke. *Breaking Waves and the Dispersion of Surface Films.* http://www.ph.ed.ac.uk/~ted/thesis/node42.html

[3] P Chacon, W Wriggers. *Multi-resolution contour-based fitting of macromolecular structures*. J. Mol. Biol, 2002 - biomachina.org

[4] DM Belnap, A Kumar, JT Folk, TJ Smith, TS Baker. *Low-Resolution density maps from atomic models: how stepping ''back''can be a step ''forward''*. J. Struct. Biol, 1999

[5] W.Lorensen, H.Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics, 21 (4): 163-169, July 1987

[6] Raj Shekharly, Elias Fayyad, Roni Yage, J. Fredrick Cornhill, Octree-Based Decimation of Marching Cubes Surfaces, IEEE Visualization, 1996

[7] Tolga Can, Yujun Wang, Yuan-Fang Wang, Jianwen Su. Bioinformatics: FPV: fast protein visualization using Java 3D™. ACM SAC: 88-95, 2003

[8] Colin R. F. Monks, Patricia J. Crossno, Constantine Pavlakos, George Davidson, Abraham Kupfer, Cliudio Silva, Brian Wylie: Three Dimensional Visualization Of Proteins In Cellular Interactions. Proceedings of the 7th conference on Visualization: 363 - ff, October 1996

[9] E.E. Lattman. *Optimal Sampling of the Rotation Function*, pp 179-185. The Molecular Replacement Method (Ed. M.G. Rossmann), Gordon and Breach, Science Publishers Inc., New York, 1972.