New Features in Coot

Paul Emsley, Laboratory of Molecular Biophysics, University of Oxford, UK

SHELX Interface

A few enhancements have been made to the SHELX/Coot interface.

First, errors have been fixed in the way that Coot read SHELX .res files. The off-diagonal elements of the U matrix for anisotropic atoms are now read in the correct order. Also a pseudo-isotropic B-factor has been introduced (the average of the trace) which corresponds to the B factor in a SHELX output pdb file. Previously Coot used 10.0 as a placeholder here.

As a consequence, the Validation B factor variance plot has become useful - particularly so with SHELX's nicely refined high resolution atomic displacement parameters.

Coot now finds (large) gaps in the residue numbering and splits the chain on finding a such a gap. Thus waters are split into their own chain and hence water-specific analysis now works using .res files.

Secondly, Coot reads the SHELX listing (.lst) file and produces a GUI for it. Thus we are provided with a dialog box of buttons which describe and navigate to such things as potentially split atoms, bad DANGs or bumps.

Thirdly, Coot reads the output .fcf file from SHELXL - it does so by first converting the CIF file to an mmCIF file. The resulting structure factors are passed through the (relatively new) σ_a code in Clipper 2.0 to provide a σ_a map and σ_a difference map.

And lastly on the new SHELX functions, there is a convenience GUI interface which will read all the above files (.res, .lst, .fcf) given a single file-name.

NCS

Several enhancements have been made to the NCS handling in Coot. The copy residue range function in particular has been made more robust.

NCS skip

The '**O**'[1] key has been bound to the `skip-to-next-ncs-chain` function. When working on (or looking at) a particular residue in a chain that has NCS, one might think "what does the other chain look like here" - the '**O**' key provides a quick means to navigate to the corresponding residue other NCS related chains.

[1] think "Other Chain"

Non-default NCS operators

There are two scenarios here, depending on if the related structure has secondary structure or not.

For example, consider the case of a single chain composed of 2 domains, with a variable elbow angle between them and this single chain has 2 NCS copies in the asymmetric unit. It is quite possible that the different copies of the chain have different elbow angles and so the default operator which maps the whole copy on to the master molecule would not be optimal. It would better to have NCS copies for each domain separately. This is particularly important for NCS map overlays.

In this case, the recommended procedure is to use the Copy Fragment function[1] to create a new molecule that contains an atom selection that is just those parts of the asymmetric unit that are related by close NCS (e.g. the first 200 residues of a chain). Coot can then find the NCS in that partial structure and that can be used to generate an NCS operator to overlay the maps.

The other scenario is that the reference structure and NCS related copies are not protein and therefore do not have protein secondary structure[2].

In this case, Coot will not be able automatically determine the NCS operators. The solution is to provide a residue range and a pair of chain-ids to the function `manual-ncs-ghosts`. Coot will then use the residue range information to find the operator from a least-squares fit of one fragment onto the other (the automatic method is to use SSM to do this) and Coot is force-fed this operator. Coot will act as if it had generated the operator itself and proceed as normal.

Repo moved to Google Code

Coot now uses Google Code as the code repository:
http://code.google.com/p/coot/

Google Code has proved to be fast and reliable.

The current spec and schedule are available in the TODO file.

The mailing list has moved to JISCMail, this has relieved the system administrator in York of a burden, but the web representation of the threads leave something to be desired when compared to MHonArc or hypermail.

Structure as s-expressions

Using `(add-molecule molecule-expression molecule-name)` Coot can construct an internal representation of a molecule from a s-expression - a physical representation (i.e. a pdb file on a disk) is no longer required. The structure of the s-expression follows the mmdb coordinate hierarchy. Using an s-expression in this way, it is possible to send Coot structure data via a network connect without having to deal with files on a file-system. The

[1] `new-molecule-by-atom-selection`
[2] for example, RNA, DNA or ligands

application for this is from "driver programs", automaticed structure solution pipe-lines, refinement packages and the like that would like to show particular features to the user - perhaps even ask questions about them.

There is also a mechanism to update a given molecule using `(clear-and-update-molecule molecule-number molecule-expression)`. In this way it is possible to replace a given molecule with (for example) atoms in different places. This allows us to represent on-the-fly molecular dynamics, rotamers options, various loops or other partial models.

set-atom-attribute

The individual "atomic" attributes of atoms[1] are accessible to modification using the set-atom-attribute function.

The Active Residue

After discussions with Frank von Delft[2] the concept of "Active Residue" was added[3]. What that means is that Coot finds the residue[4] that is close to (or at) the centre of the screen and provides a residue specification for it that can be used in scripting functions.

Key Bindings

If you look at the console when you type a key letter, those without a binding result in something like

```
key: 110
```

when you press it. Those that do have a internal binding don't say anything[5].

Redefining `graphics-general-key-press-hook` is how you attach your bindings:

```
(define graphics-general-key-press-hook
  (lambda (key)
    (cond
     ; H key
     ((= key 120) ; X key
      (refine-active-residue))
     ((= key 104) ; H key
      (refine-active-residue-triple))
     ((= key 107) ; K key
      (auto-fit-rotamer-active-residue))
     ;; other binding (place atom at pointer when V key hit)
     ((= key 118) ; V key
      (set-pointer-atom-is-dummy 1)
      (place-atom-at-pointer)))))
```

[1] such as the x, y, z coordinates, the occupancy or alt-conf

[2] Frank's philosophy (if I may paraphrase) is "I want to get my Coot session over with as quickly as possible and I want to do it using the keyboard - making me use menu items is bad and slows me down" - I have some sympathy with this view.

[3] I'd been resisting the idea because I thought that it would be impossible or very difficult to port to Python, I needn't have worried as it turns out - Bernhard Lohkamp made it work

[4] actually, it's the atom

[5] Coot already has several key bindings, for example R,T,Y,A,S,I,D,L,C,E,Q,W,O,P,B,N,M,Esc,Ret

I hope that this is clear that this binds `(refine-active-residue)` to the 'X' key, `(refine-active-residue-triple)` to the "H' key, `(auto-fit-active-rotamer)` to the 'K' key and `(place-atom-at-pointer)` to the 'V' key. Quite useful functions[1] - especially 'X' and "K'.

Coot has many functions which take specifications for a residue or an atom - using key bindings and the results of `(active-residue)` bespoke short-cuts can be created that work on the residue at the centre of the screen[2].

Concluding remarks

I should note however, that fixing crashes, bugs and mis-feature oversights has been the highest priority. For the last 2 years, this has pretty much been the mainstay. I'd like this to change - which requires rather more discipline in the initial writing of the code on my part than I had currently been used to. An embedded testing architecture has been introduced - tests can be (and have been) written in Coot's scripting language, which makes testing portable and automated[3]. After writing a new feature I now think "How can I write tests for this?", rather than using the GUI one or twice to see that it vaguely works in the example case. I'm hoping that substantial use of this architecture will result in less reworking of functions and ultimately expedite Coot development. We'll see.

[1] Perhaps they should be the default.

[2] Note that the P key is bound to the "Update from Current Position" function, which takes you to the nearest displayed atom, (with a preference for CAs, if it can find one) this will also update the attributes in the Go To Atom window, so that "Next Residue" will take you to the next residue relative to the one at the current position.

[3] Peter Zwart and Ralf Grosse-Kuntsleve double-teamed me to add straws to my resistant humpy back