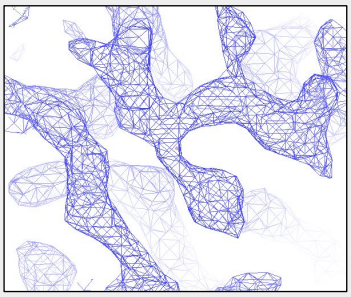
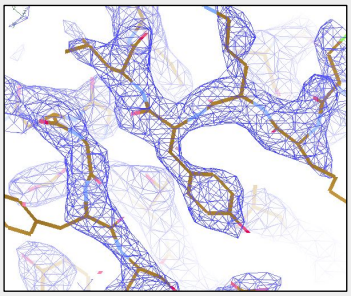
Model Building

Paul Bond
University of York

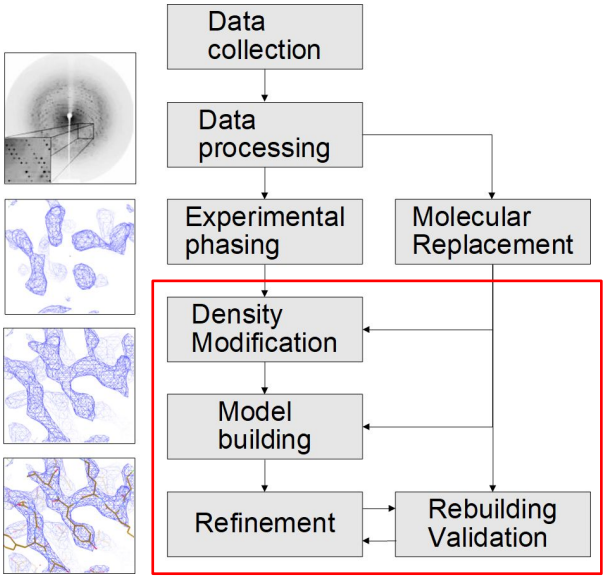
paul.bond@york.ac.uk





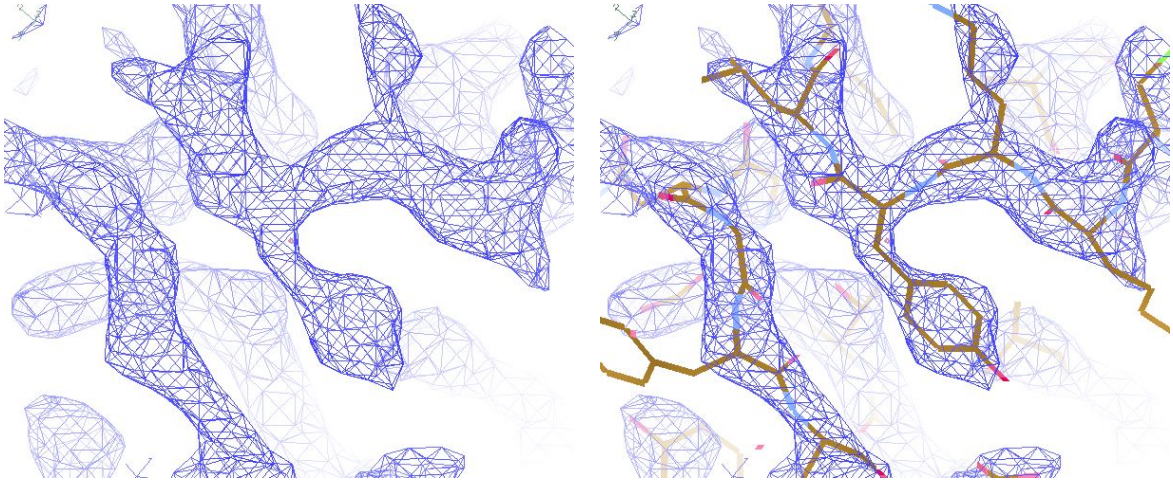
This presentation is on model building in X-ray crystallography. It mainly focuses on ModelCraft, which is a new automated model-building pipeline developed in the group of Kevin Cowtan at the University of York. Kevin is the author of Buccaneer (a program for automated protein building) and Nautilus (a program for automated nucleic acid building), and ModelCraft is a new pipeline that includes both of those programs along with others for density modification, refinement and validation.

X-ray Crystallography



This is an overview of the structure solution process for X-ray crystallography. Once you've done the hard work of growing a crystal, you need to collect images and process them to get structure factor amplitudes. In order to get an electron density map you also need to get some phase estimates either through experimental phasing or molecular replacement. If you've used experimental phasing you will need to do density modification to improve the phases and hence the quality of the map. You then use automated model building to build an initial model into the improved map, followed by multiple rounds of refinement, rebuilding and validation. However, it's more likely that you'll get phases from molecular replacement. If your molecular replacement model is poor then you might need to do density modification first before you can improve it with automated building. If the MR model is very good then you might even want to skip automated building and go straight to interactive rebuilding and validation in Coot. The goal of ModelCraft is to automate the steps in the red box as much as possible.

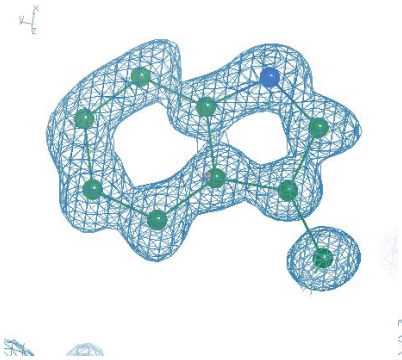
Buccaneer - Task



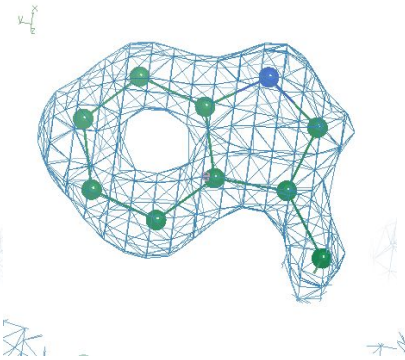
Buccaneer is the program that ModelCraft uses to build protein models. It takes a map, target protein sequences and optionally an existing model, and attempts to build a protein model to fit the map.

Buccaneer - Task

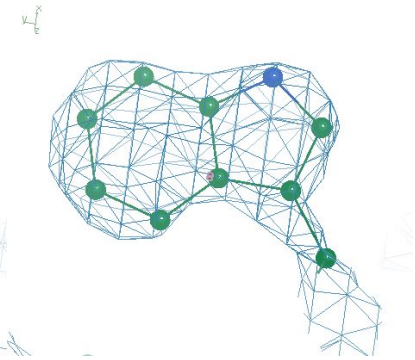
Resolution: 1Å



Resolution: 2Å



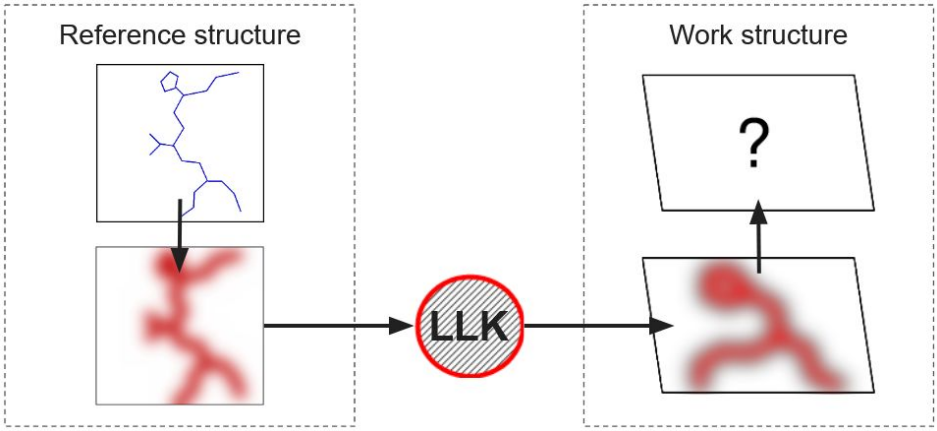
Resolution: 3Å



The *Buccaneer* method was designed to address the problem that electron density maps look very different at different resolutions. At high resolution you can distinguish peaks for individual atoms, but if you write a program that looks for separate atomic peaks then it wont work when you have a low resolution map and you can't see them.

Buccaneer - LLK Target Function

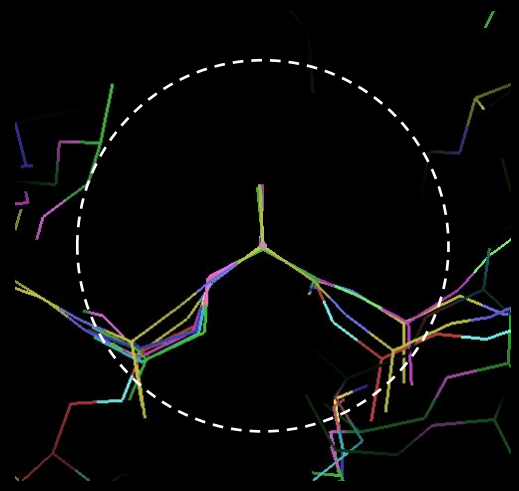
Compare simulated map and reference model to obtain likelihood target then search for this target in the unknown map

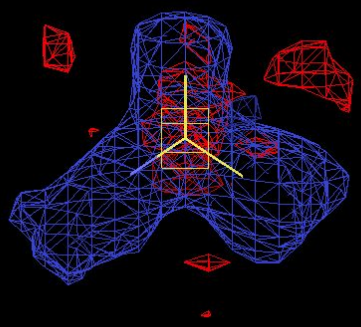


To help with this problem, *Buccaneer* uses a known reference structure. You give B*uccaneer* the map you're trying to build a model into (the work map) and it generates a map for a reference structure with the same resolution and same level of noise as this map. Then it uses the reference structure to find out what features in the reference map look like, and looks for the same features in the work map.

Buccaneer - LLK Target Function

Measure mean and variance of reference map in a 4Å sphere around Cα





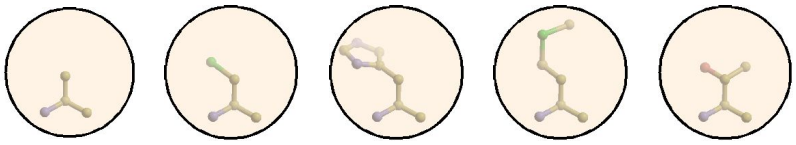
The target to search for individual residues is obtained by superimposing all the residues in the reference structure on top of each other. The mean and variance of the density is then used to construct a likelihood target within a 4A sphere. The blue density shows regions of conserved high density and the red density shows regions of conserved low density. Both are important when finding residues in the work map. If there is low density in the blue regions or high density in the red regions then it is likely not the correct position for a residue.

Buccaneer - LLK Target Function

Likelihood functions based on conserved density features

Finding, Growing Cα environment (4.0Å sphere)

Sequencing Cβ environment (5.5Å sphere)



ALA CYS HIS MET THR … x20

The 4A C-alpha target is used in the first step of the program to find initial residue positions and the second step to grow those residues into chains. The program also uses 5.5A targets centred around C-beta atoms. A different target is made for each residue type and those are used for sequencing. Rotamers are not separated before making the targets, so the mean and variance of the density is calculated for side chains pointing in lots of different directions, but this still provides enough information to find the correct sequence.

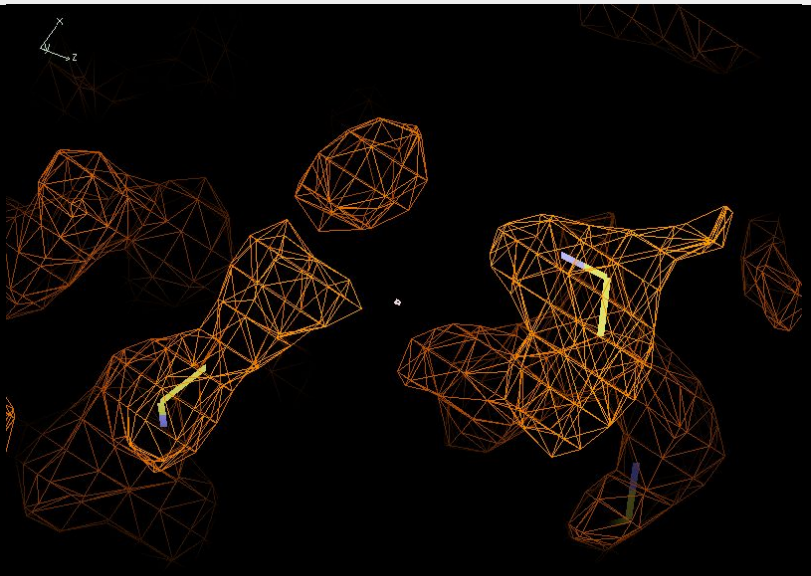
Buccaneer - Steps

1. **Find** oriented residue positions
2. **Grow** each residue into a chain fragment
3. **Join** overlapping fragments and resolve branches
4. **Link** nearby termini
5. **Sequence** the chains
6. **Correct** insertions and deletions
7. **Filter** to remove short chains
8. **NCS** superposition to extend chains
9. **Prune** residues to resolve clashes
10. **Rebuild** side chains

Buccaneer runs as a cyclic calculation, where each cycle has these 10 steps.

Buccaneer - Example

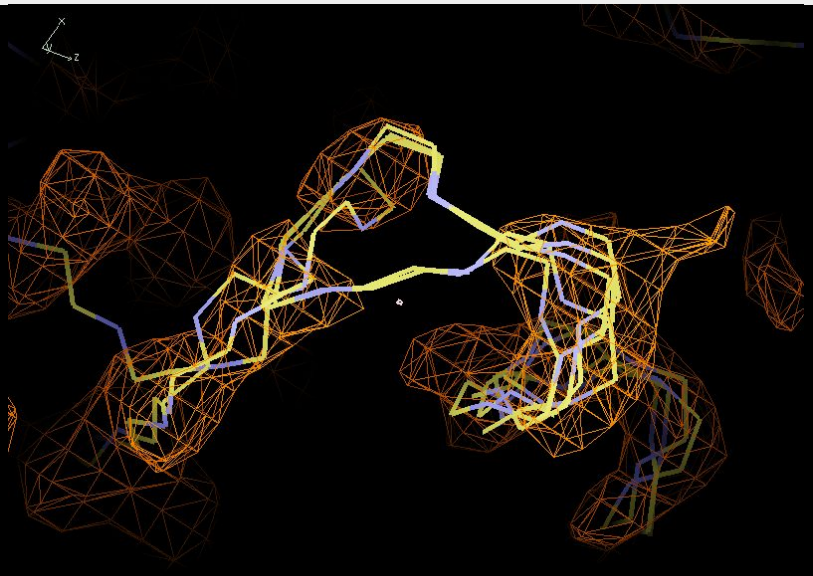
1. ***Find***
2. Grow
3. Join
4. Link
5. Sequence
6. Correct
7. Filter
8. NCS
9. Prune
10. Rebuild



This is an example of *Buccaneer* building a difficult loop in a 2.9Å map. The first step is to find possible positions for new residues. It has placed residues at three positions in this figure (two in the foreground and one in the background) where the density has the right shape according to the C-alpha target function. There are also other residues placed out of this view.

Buccaneer - Example

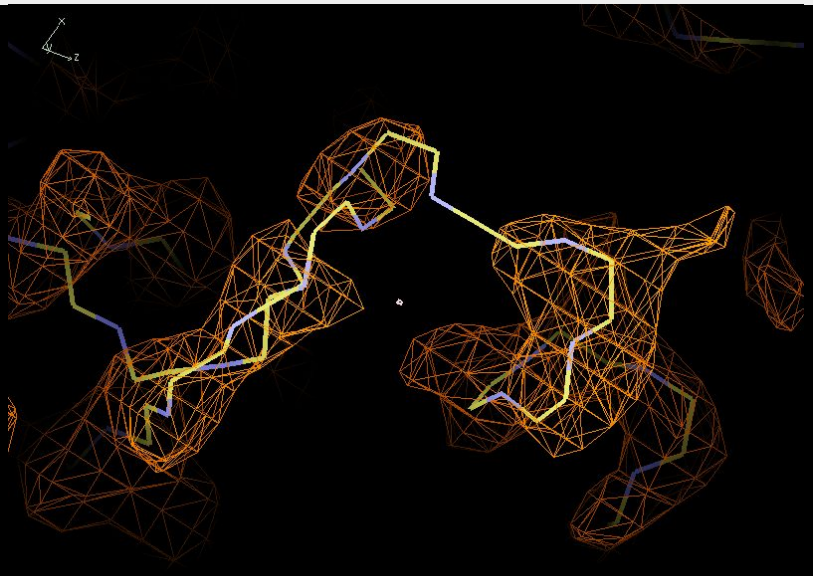
1. Find
2. ***Grow***
3. Join
4. Link
5. Sequence
6. Correct
7. Filter
8. NCS
9. Prune
10. Rebuild



The next step is to grow each residue into a long chain fragment. New residues get added at both ends using an exhaustive search over allowed Ramachandran angles and the same C-alpha target. Once the target score drops below a threshold value the growing stops. This step gives lots of fragments, many of which overlap. In this example it has built a helix on the right where the chains agree fairly well, two different paths to bridge the gap in the middle and a contradictory chain on the left that has been built in the opposite direction.

Buccaneer - Example

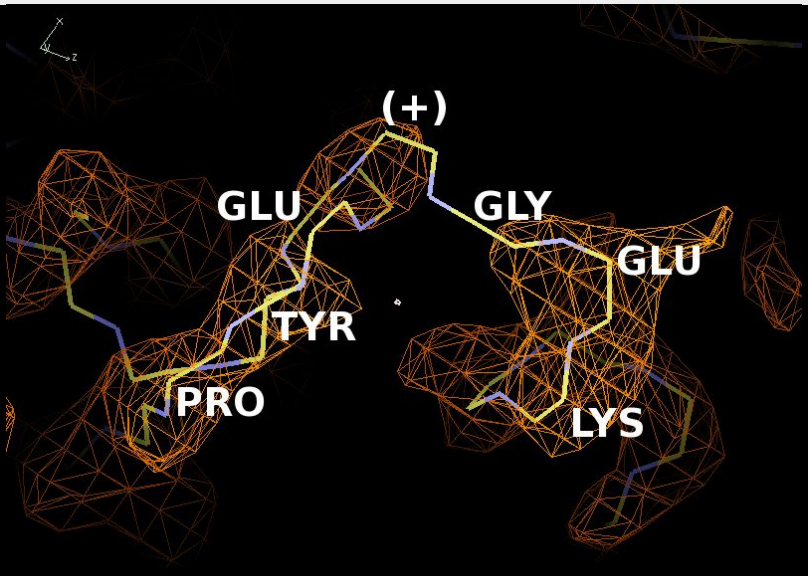
1. Find
2. Grow
3. ***Join***
4. Link
5. Sequence
6. Correct
7. Filter
8. NCS
9. Prune
10. Rebuild



The joining step merges all the chain fragments that agree with each other into single chains. This step also has to make decisions about which routes to follow. In addition to looking at how well the residues fit the density, it also tries to build the longest chains (e.g. it chose the longest route over the middle loop). This bias towards long chains helps to build helices correctly. The chain built in the reversed direction is still there because it can't be merged and at this stage it hasn't been decided which is correct.

Buccaneer - Example

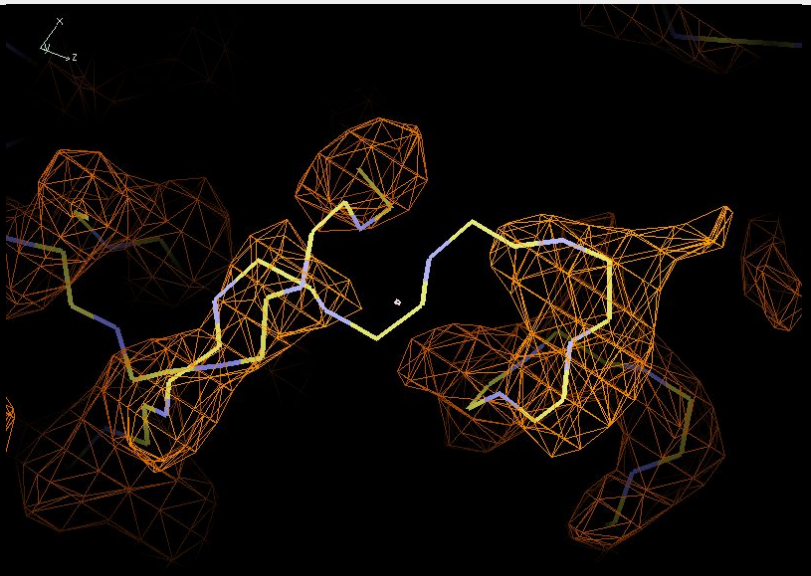
1. Find
2. Grow
3. Join
4. Link
5. ***Sequence***
6. Correct
7. Filter
8. NCS
9. Prune
10. Rebuild



We have skipped the linking step because there are no termini that can be linked in this view. The sequencing step works by assigning each residue a probability that it is each of the 20 residue types using the C-beta targets. These probabilities are then compared to the known sequence(s) to look for continuous runs that stand out with a high probability. In this example, the sequence only fits the chain well if an extra symbol is added in the middle. This is called an insertion and (unless the sequence is wrong) it means the chain has been built incorrectly. The reversed chain on the left wasn't sequenced as there was no part of the sequence that fits well.

Buccaneer - Example

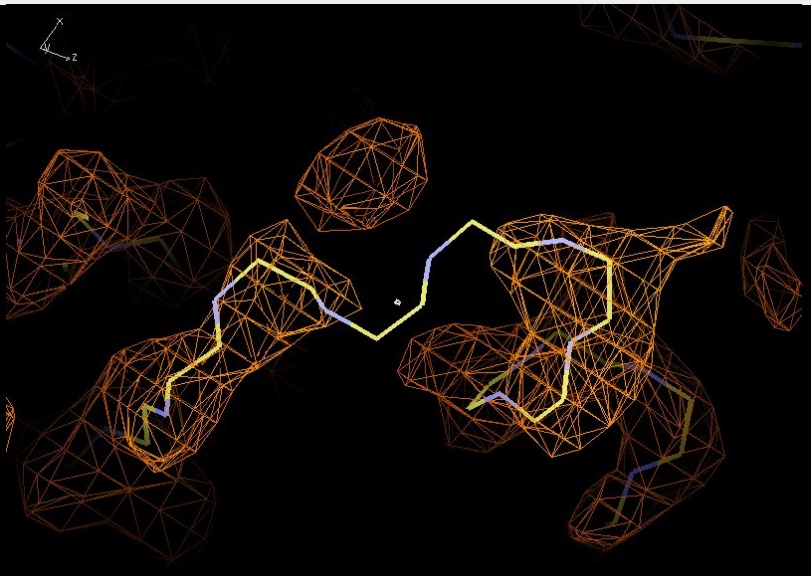
1. Find
2. Grow
3. Join
4. Link
5. Sequence
6. ***Correct***
7. Filter
8. NCS
9. Prune
10. Rebuild



The next step is to correct any insertions (where it has built one extra residue) and deletions (where it has built one fewer residue) found during the sequencing step. The insertion that was over this loop as been corrected by removing three residues and rebuilding two.

Buccaneer - Example

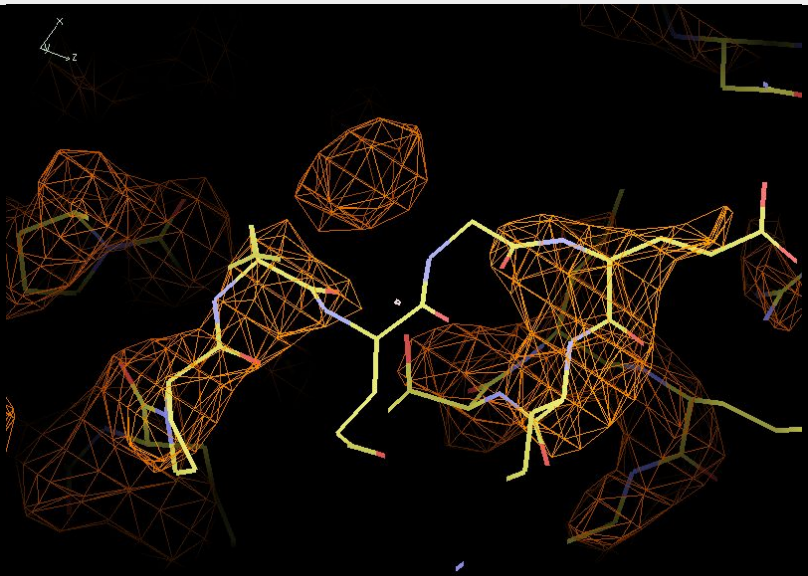
1. Find
2. Grow
3. Join
4. Link
5. Sequence
6. Correct
7. Filter
8. NCS
9. ***Prune***
10. Rebuild



The pruning step removes residues from clashing chains to produce a single consistent model. If the pruned chains have less than 6 residues remaining they are removed too. Pruning is done at this late stage to have the most information about which chain is correct. Residues that are unsequenced or are from shorter chains with worse fit to the density will be removed preferentially. In this case the reversed chain on the left was removed.

Buccaneer - Example

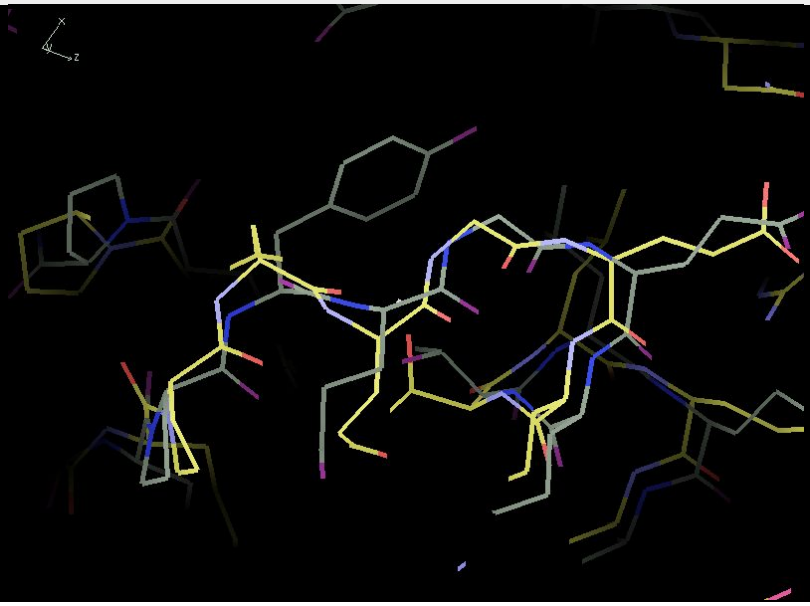
1. Find
2. Grow
3. Join
4. Link
5. Sequence
6. Correct
7. Filter
8. NCS
9. Prune
10. ***Rebuild***



Finally, the last step is to build side chains for each residue using a rotamer library. The rotamer with the highest average density at the atomic positions is chosen. If two side chains end up clashing, *Buccaneer* will try and find a pair of rotamers that do not clash. If this fails both residues will be truncated to C-beta.

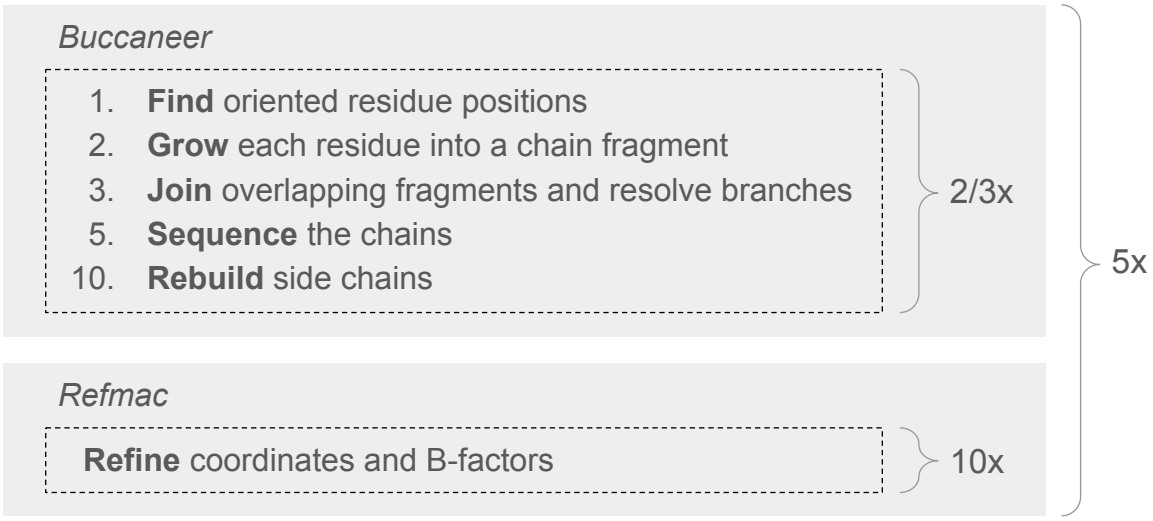
Buccaneer - Example

1. Find
2. Grow
3. Join
4. Link
5. Sequence
6. Correct
7. Filter
8. NCS
9. Prune
10. Rebuild



This shows a comparison with the deposited model. *Buccaneer* built most of the model quite well but got some things wrong, for example the the tyrosine side chain. The model will improve after refinement with *Refmac* and further cycles of *Buccaneer*.

Buccaneer - Pipeline



Buccaneer does not perform global refinement of the model, so when it is used through one of the interfaces (i.e. *CCP4i*, *CCP4i2* or *CCP4 Cloud*) it actually runs a pipeline that combines *Buccaneer* with *Refmac*, which refines the model coordinates and B-factors and produces an updated map for further building. The original pipeline (in *CCP4i* and *CCP4 Cloud*) runs 2 or 3 cycles of *Buccaneer* followed by 10 cycles of *Refmac*, repeats that 5 times, and outputs the final model. There have been some additional developments to the pipeline in *CCP4i2*, which now runs up to 25 cycles, adds waters in *Coot* before refinement and outputs the model from the cycle with the lowest R-free.

ModelCraft - Motivation

1. Improve the *Buccaneer* pipeline for molecular replacement structures

2. Build protein, RNA & DNA in a single pipeline
   (and other components in the future)

3. Share the new developments with all interfaces
   - Command line
   - CCP4i2
   - CCP4Cloud
   - CCP-EM (planned)

The original *Buccaneer* pipeline was fast and showed good performance when tested on structures solved by experimental phasing, but when we tested it on a large set of MR structures (with some poor models and incomplete solutions) we found many where the pipeline did not produce a very complete model. We therefore wanted to improve the pipeline to do a better job with MR structures. We also wanted to create a pipeline that could build RNA and DNA at the same time as protein (and hopefully sugars, modifications and ligands in the future). Finally, we wanted to have the new developments shared between all interfaces without having to re-write them in multiple frameworks.

ModelCraft - Pipeline

Shift-field refinement (*Sheetbend*) - *Refmac*

1. Prune residues (*Coot*) - *Refmac*
2. Density modification (*Parrot*)
3. Add dummy atoms (*Coot*) - *Refmac*
4. Build protein (*Buccaneer*) - *Refmac*
5. Prune chains (*Coot*) - *Refmac*
6. Build RNA & DNA (*Nautilus*) - *Refmac*
7. Add waters (*Coot*) - *Refmac*

≤ 25x

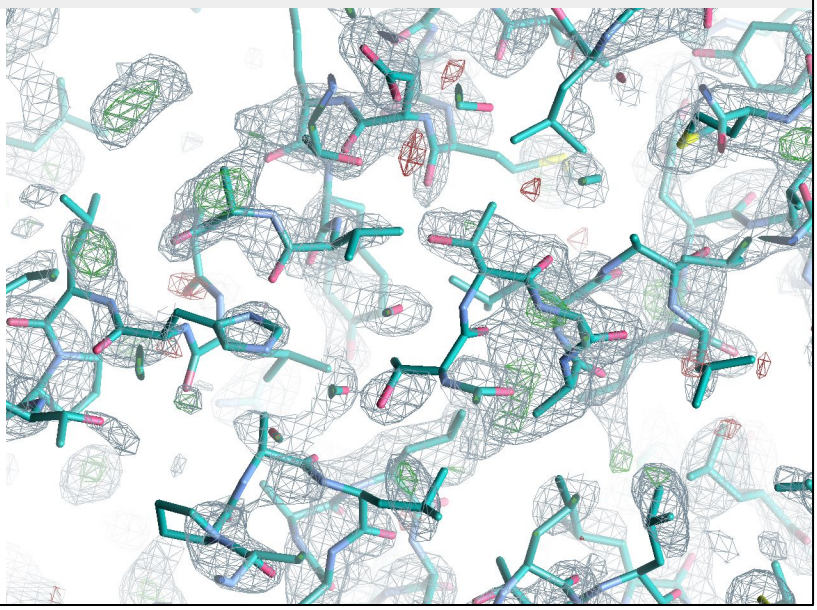
Rebuild side chains (*Coot*) - *Refmac*

This is the overall *ModelCraft* pipeline as it stands. If you provide a starting model it is first refined using *Sheetbend* then *Refmac*. Then a single cycle of the pipeline has 7 steps, most of which are followed by refinement with *Refmac*. With high resolution data, it starts by pruning individual residues using *Coot*. Then it uses *Parrot* for density modification, which is especially powerful if there is non-crystallographic symmetry that can be determined from the current model. After *Parrot*, it adds dummy atoms using *Coot* to improve the phases, but the dummy atom structure is only accepted if R-free improves. The dummy atoms are removed from the model then *Buccaneer* builds the protein and *Coot* is used to remove incorrect chain fragments. The pipeline also builds nucleic acids with *Nautilus* and finally adds waters with *Coot*, but again only accepting the result if R-free improves. Once the model stops improving (or it reaches 25 cycles) the cycle with the best model is used as the output. If the resolution is high then *Coot* is used to rebuild missing or incorrect side chains.

Example - Cycle 0

1. Prune residues
2. Density modification
3. Add dummy atoms
4. Build protein
5. Prune chains
6. Build RNA & DNA
7. Add waters

R-free: 54%

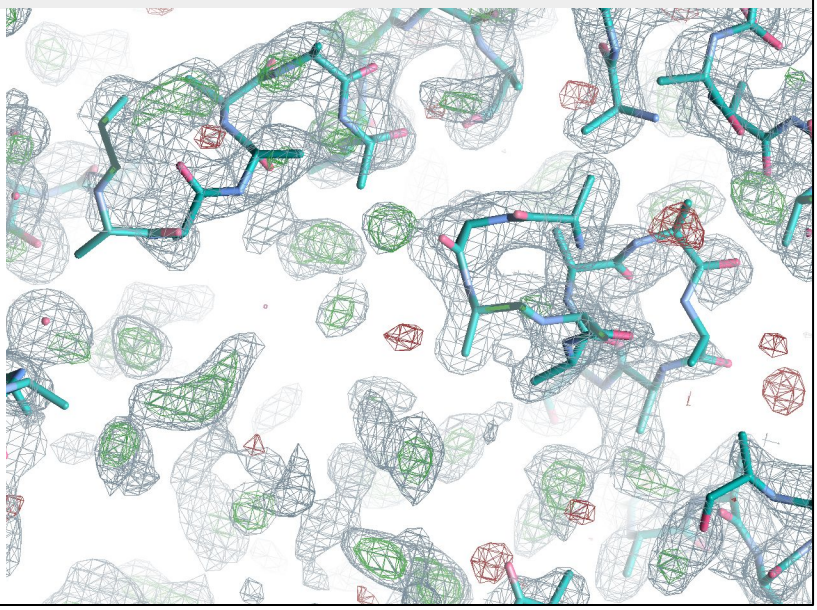


This is a difficult example where *ModelCraft* did well. The target structure is 3GVP with 4 copies of a 435 residue protein and data to 2.25 Å resolution. The *AlphaFold* model was trimmed to remove residues with high pLDDT but it was not split into separate domains for molecular replacement. *MOLREP* placed four copies but only three of them were placed correctly. The F-map correlation after MR was 0.3, which is on the threshold of what would be considered a solution. The phases are made worse by the relative domain positions being slightly different in the true structure and the *AlphaFold* model. This is a view over the incorrectly placed copy after refinement with *Sheetbend* and *Refmac*, with the R-free value stuck around 54%.

Example - Cycle 3

1. Prune residues
2. Density modification
3. Add dummy atoms
4. Build protein
5. Prune chains
6. Build RNA & DNA
7. Add waters

R-free: 50%

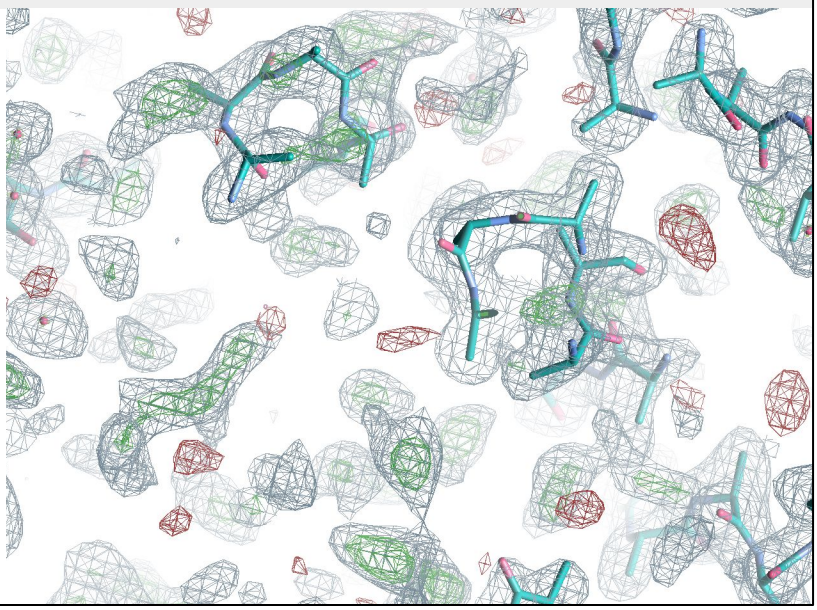


This is the same view but after 3 cycles of *ModelCraft*. The original structure in this region has mostly been removed and now there are just a few small fragments built by *Buccaneer*. R-free has reduced from 54% to 50%.

Example - Cycle 4

1. **Prune residues**
2. Density modification
3. Add dummy atoms
4. Build protein
5. Prune chains
6. Build RNA & DNA
7. Add waters

R-free: 50%

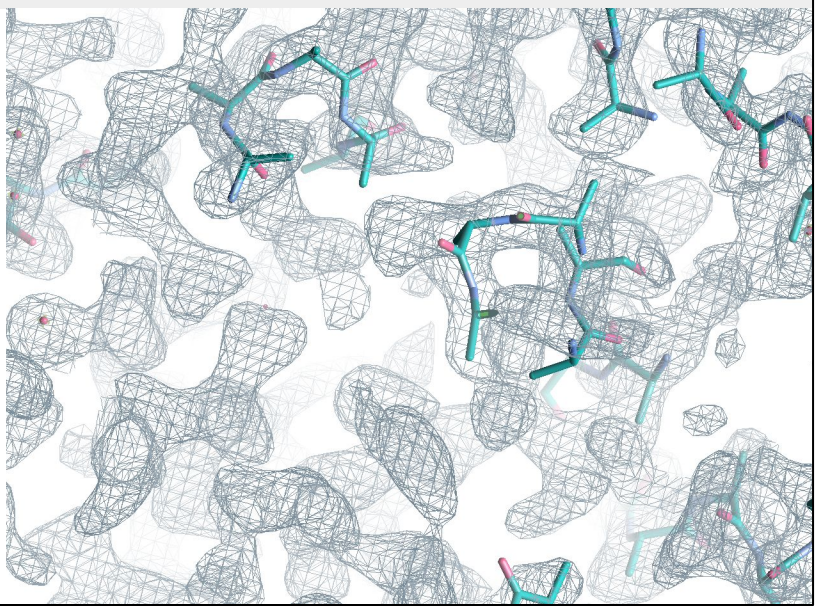


A large improvement is seen in the 4th cycle. The first step is to remove the worst residues using *Coot* and refine again. This step is only done if the resolution is better than 2.3 Å. The density for the removed residues in the top left disappeared after further refinement, suggesting that the density was biased.

Example - Cycle 4

1. Prune residues
2. **Density modification**
3. Add dummy atoms
4. Build protein
5. Prune chains
6. Build RNA & DNA
7. Add waters

R-free: 50%

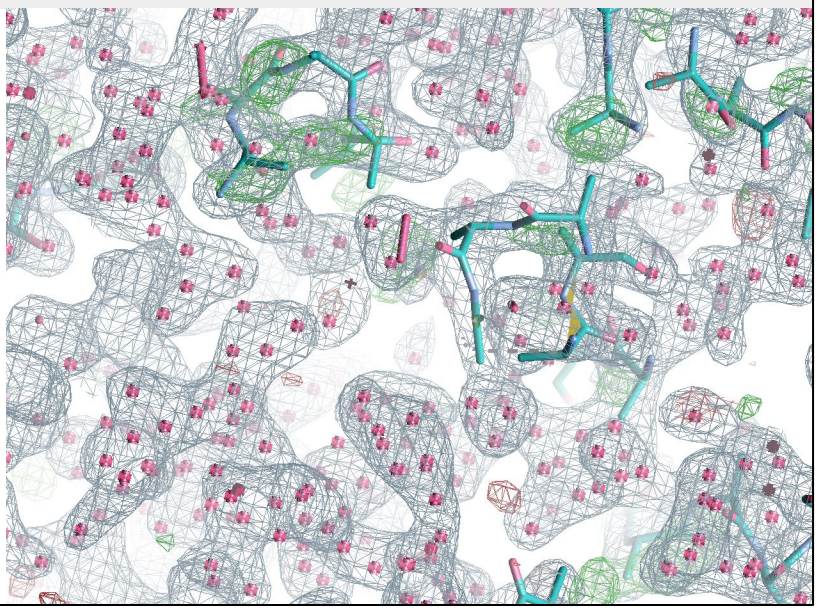


The second step is density modification using *Parrot*, which is especially powerful when there are multiple copies of a chain and the non-crystallographic symmetry (NCS) operators can be determined from the model. NCS-related regions of the map will be averaged, weighted by how similar the density is. The area that gets averaged tends to grow as the phases improve. In this case, *Parrot* only determined symmetry operators between the three correctly-placed copies, and not the copy in this region.

Example - Cycle 4

1. Prune residues
2. Density modification
3. **Add dummy atoms**
4. Build protein
5. Prune chains
6. Build RNA & DNA
7. Add waters

R-free: 32%

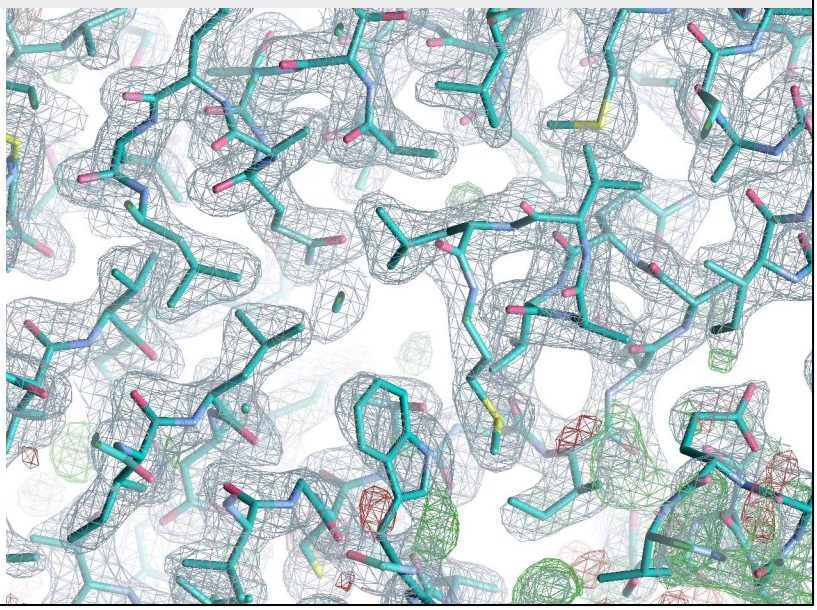


The third step adds dummy atoms into the *Parrot* map using *Coot*. The existing structure and the dummy atoms are then refined with *Refmac*. No attempt is made to interpret the dummy atoms as it is a difficult task and at lower resolutions it is unlikely that they correspond to atom positions. Instead, they are used as a form of density modification. The resulting map is only accepted if R-free improves from the previous refinement of the protein model, which it did in this case from 50% to 32%.

Example - Cycle 4

1. Prune residues
2. Density modification
3. Add dummy atoms
4. **Build protein**
5. Prune chains
6. Build RNA & DNA
7. Add waters

R-free: 33%

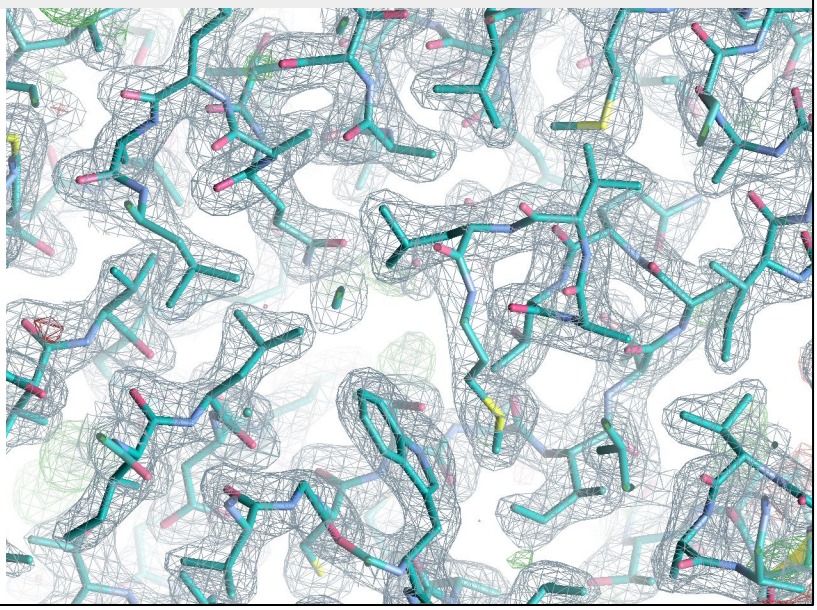


The dummy atoms are then discarded and *Buccaneer* is used to build a protein model. Because of the better phases, it has built most of this view correctly apart from some parts in the bottom right.

Example - Cycle 5

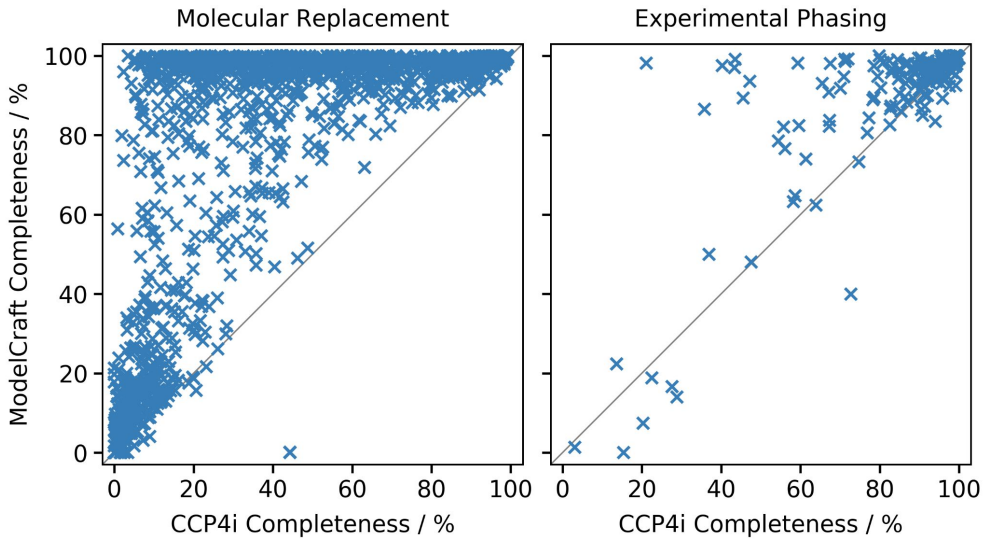
1. Prune residues
2. Density modification
3. Add dummy atoms
4. Build protein
5. Prune chains
6. Build RNA & DNA
7. Add waters

R-free: 26%



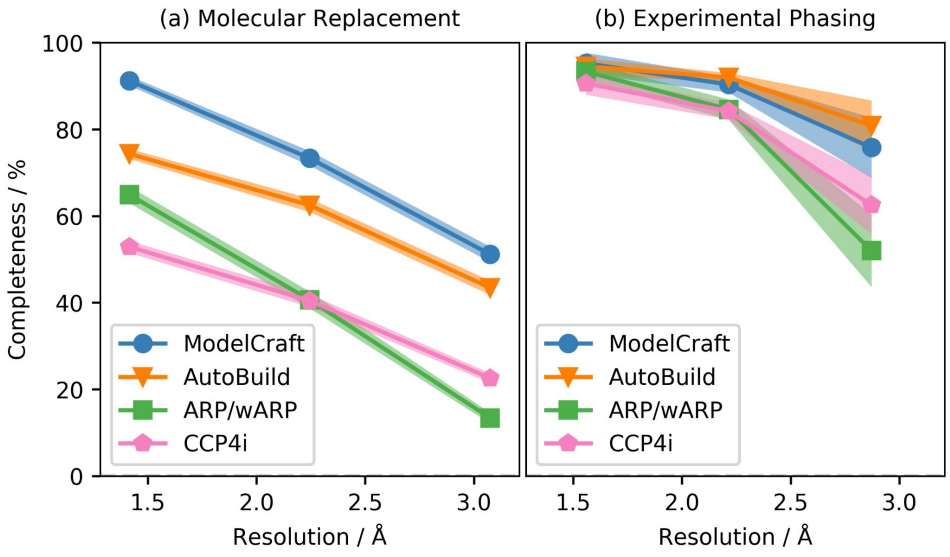
At the end of cycle 5 most of those issues have been fixed and R-free has improved to 26%. The model still requires some interactive rebuilding and validation using in *Coot*. For example, there is a twisted peptide bond in the bottom right as indicated by the yellow triangle.

ModelCraft - Results



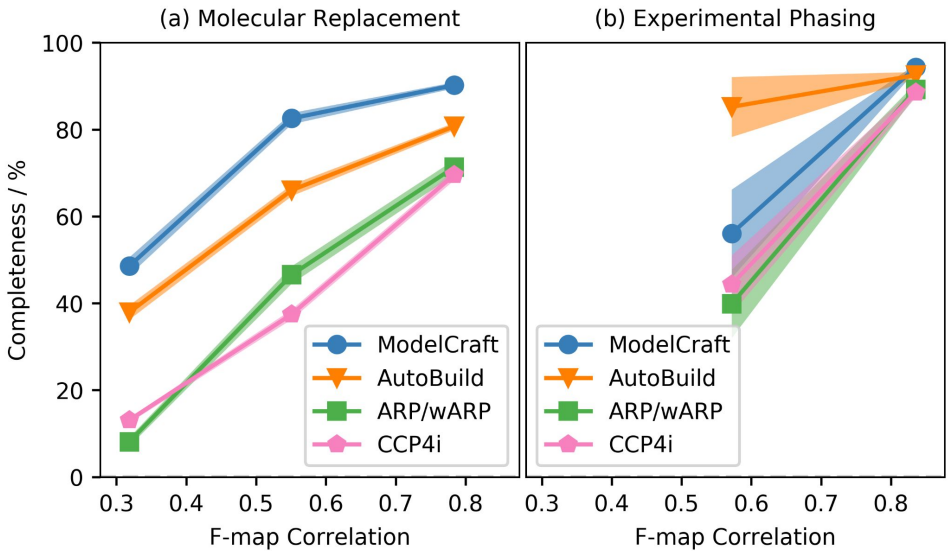
This is a comparison of the completeness of the protein model after the old *Buccaneer* pipeline from *CCP4i* and *ModelCraft*. Both were tested on 1348 molecular replacement datasets and 193 experimental phasing datasets. *ModelCraft* completes a lot more of the molecular replacement cases and nearly always does at least as well as *CCP4i*. Most of the experimental phasing cases are built well by both pipelines but *ModelCraft* still usually does better.

ModelCraft - Results



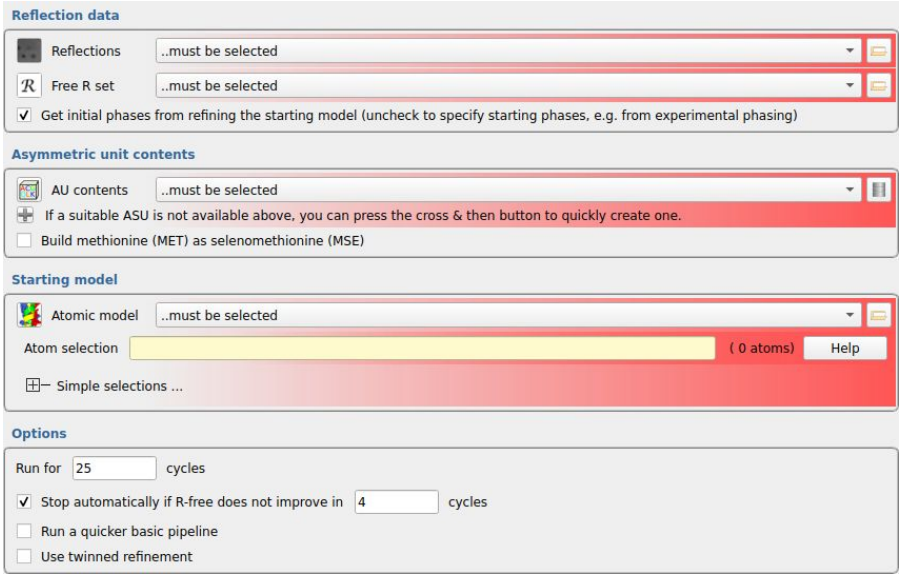
This is the mean completeness when the datasets are split into three bins based on their resolution, along with the standard error in the mean as a shaded area. Results for PHENIX AutoBuild and ARP/wARP were provided by Dr Emad Alharbi. The *CCP4i Buccaneer* pipeline and *ARP/wARP* perform similarly, but *ARP/wARP* is better at high resolution and *Buccaneer* is better at low resolution. *PHENIX AutoBuild* builds more complete models at all resolutions, but *ModelCraft* improves on this for molecular replacement cases.

ModelCraft - Results



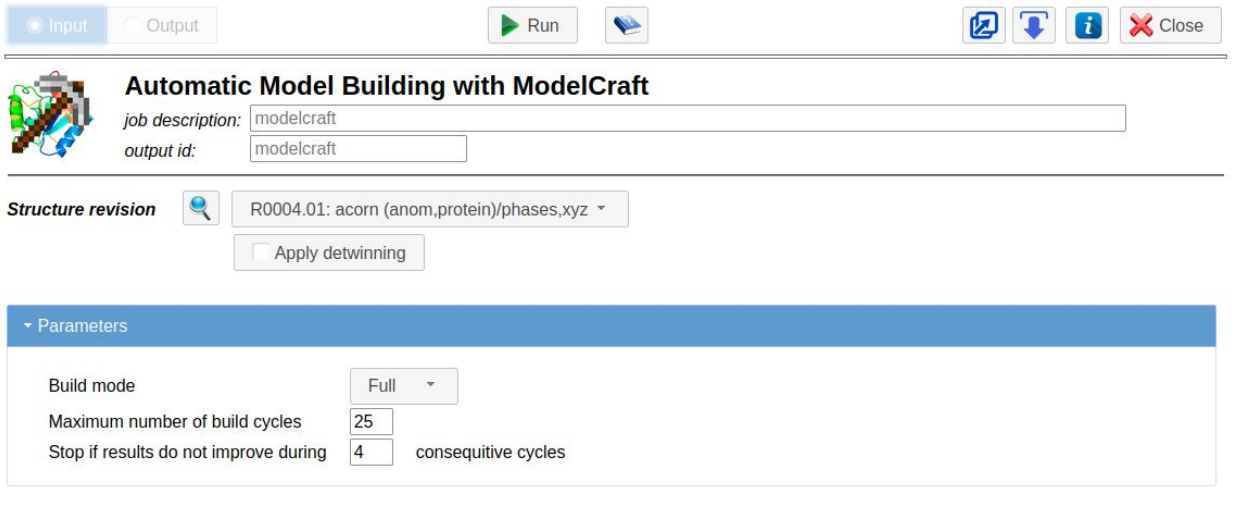
F-map correlation measures the quality of the starting phases by comparing them with the phases from the refined deposited structure. *ModelCraft* builds the most complete models after molecular replacement but *PHENIX AutoBuild* is better in difficult experimental phasing cases.

ModelCraft - CCP4i2



This is the input page for ModelCraft in CCP4i2. You need to provide reflections, a free-R set and a description of the asymmetric unit contents. Initial phases can come from refining the starting model (e.g. after molecular replacement) or can be provided explicitly (e.g. after experimental phasing). In the latter case the starting model is optional. The interface provides a few options, such as how many cycles you want to run, whether you want to run a quicker basic pipeline that is more similar to the old *Buccaneer* pipeline and whether your crystal is twinned.

ModelCraft - CCP4 Cloud



This is the interface to ModelCraft in CCP4 Cloud. There are fewer inputs to fill out here because CCP4 Cloud holds information together in a structure revision that comes from the task you are following on from. You can still choose the number of cycles, whether to run the full or the basic pipeline and whether to use twinned refinement.

ModelCraft - Command Line

```
$ modelcraft --version
2.4.1

$ modelcraft-contents 1abc contents.json

$ modelcraft-copies contents.json data.mtz

$ modelcraft xray \
> --contents contents.json \
> --data data.mtz \
> --model model.cif

$ modelcraft xray --help
```

ModelCraft can also be used on the command line. The JSON file holds a description of the asymmetric unit contents. There is a script to get the contents of a published PDB entry and a script to check the likely number of copies in your own data. If you want to run on the command line and not through *CCP4i2* or *CCP4 Cloud* there is more information at https://paulsbond.co.uk/modelcraft.

ModelCraft - Tips

Running *ModelCraft*:

- Get the right number of copies in your asymmetric unit contents
- Remove incorrect parts of the model before starting
- Include non-incorporated heavy atoms in the input model

If *ModelCraft* didn't produce a good model:

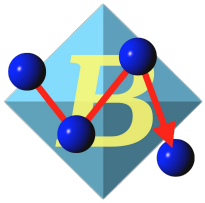
- Improve initial model or phases
- Interactively prune/build the output model in *Coot* and re-submit
- Other model building programs

When running *ModelCraft*, it is important to have the correct number of copies in order to calculate the solvent content for *Parrot*, otherwise density modification may flatten weak protein regions or amplify noise in the bulk solvent. If you know that part of your model is wrong, then it may be better to remove the incorrect residues first. It can also be useful to include non-incorporated heavy atoms (e.g. not Se from MSE) to improve the phases and to stop the model being built into that density. If *ModelCraft* hasn't done a very good job at building there are a number of things to try. The starting phases might be improved through density modification or by trying a different molecular replacement model. You might also be able to do some interactive model building to improve the initial model. The partially-built output model can also be tidied up and used as input to another run. Other model-building programs could be used with the same input data or with the partially-built model.

Other Programs

CRANK2

CCP4Build



CAB

ARP/wARP

SHELXE

PHENIX AutoBuild

These are some of the other model-building programs that are available. The top three are all pipelines that use of *Buccaneer*. *CRANK2* is full structure solution pipeline for experimental phasing. *CCP4Build* is a model building pipeline available in *CCP4 Cloud*. *CAB* is a pipeline in the *Sir* suite that can be used with a number of model building programs including *Buccaneer* for proteins and *Nautilus* for nucleic acids. *ARP/wARP* and *SHELXE* are both good programs to try if you have high resolution data but poor phases. *SHELXE* has recently added an option to include side-chain building as well as main-chain tracing. Finally, *PHENIX AutoBuild* is a very good model building pipeline available in the PHENIX suite.

Acknowledgements

Kevin Cowtan

Keith Wilson

Emad Alharbi

Eleanor Dodson

Marcin Wojdyr