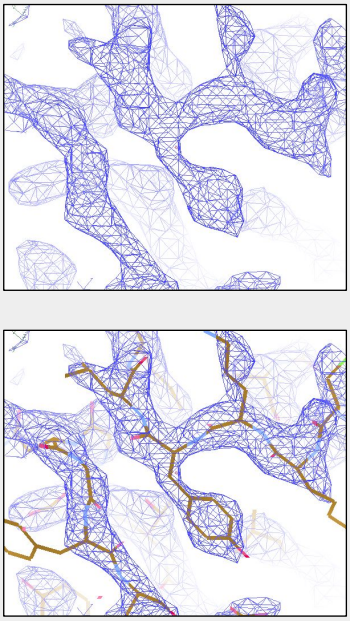
Buccaneer & ModelCraft for cryo-EM

Paul Bond, Soon Wen Hoh, Kathryn Cowtan

University of York

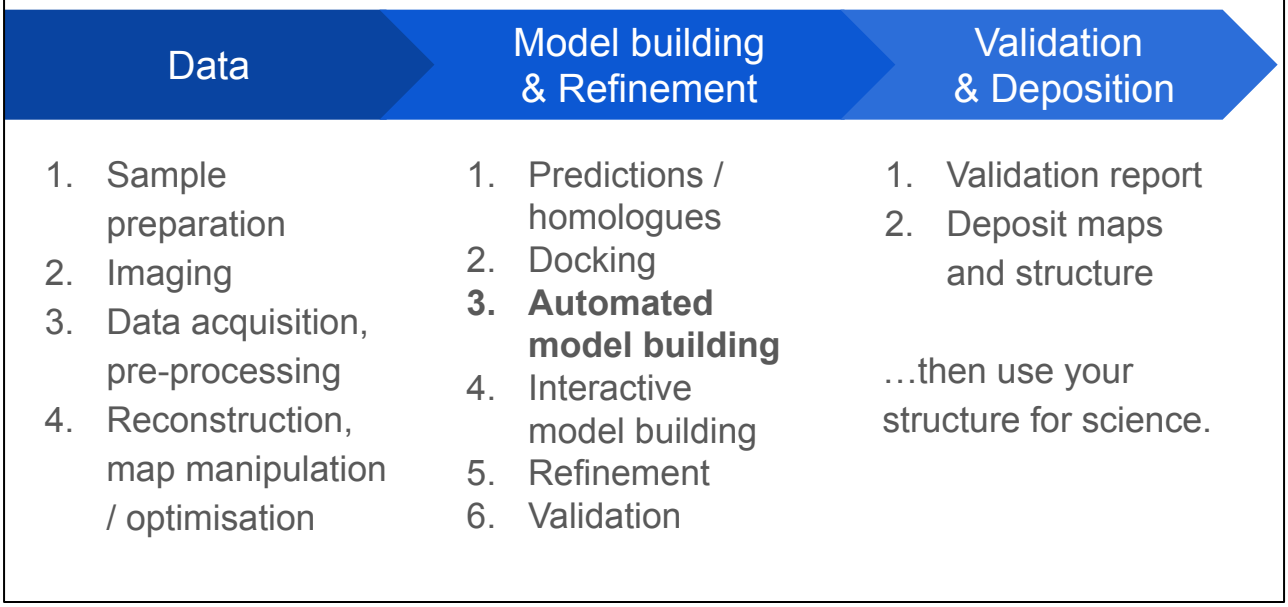
paul.bond@york.ac.uk



This presentation is on automated model building using Buccaneer and ModelCraft in CCP-EM. It focuses on the programs developed in the group of Kathryn Cowtan at the University of York: *Buccaneer* (automated protein building), *Nautilus* (automated nucleotide building) and *ModelCraft* (a pipeline that includes all of these programs as well as *Servalcat* for refinement). These programs were written for X-ray diffraction but they can also work with cryo-EM maps.

Electron cryo-microscopy

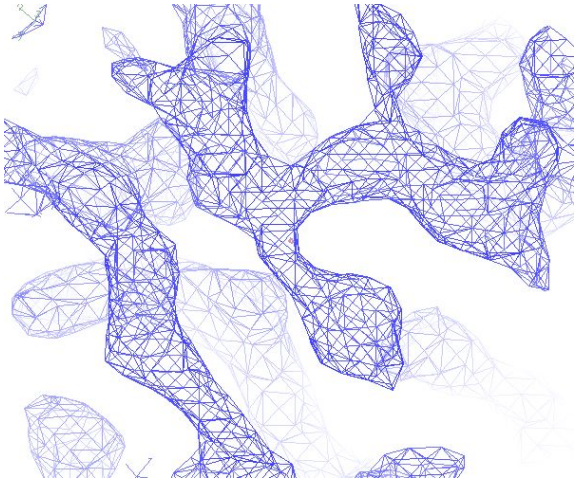
Soon Wen Hoh

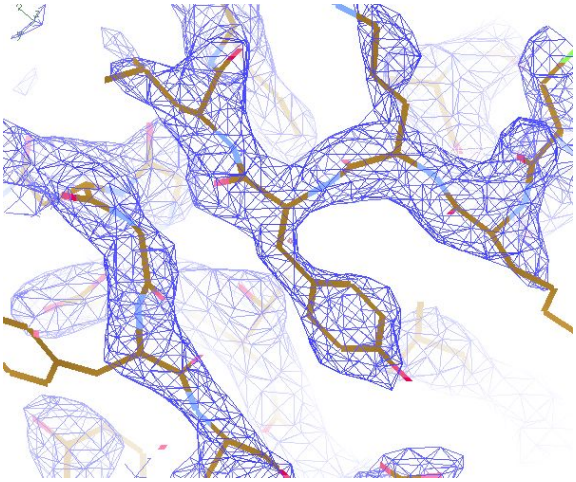


This is a simple overview of the structure solution process in EM. With the map obtained at the end of your postprocessing, you can use de-novo automated model building to build an initial model, followed by multiple rounds of refinement, rebuilding and validation. If you have a model already (from a prediction or a previously-solved structure), you can dock that into the map and use it as a starting point. If the model is very good then you may even want to skip automated building and go straight to interactive building with *Coot* or *Isolde*.

Model Building

Kathryn Cowtan

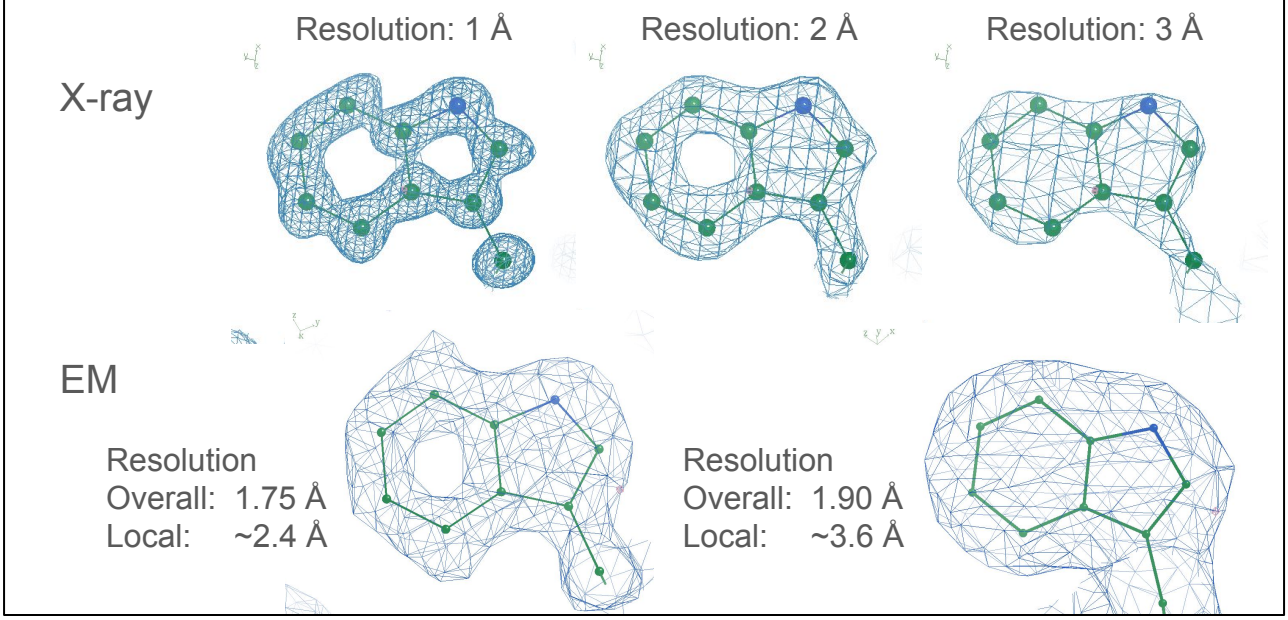




Now we will look at automated model building. *Buccaneer* and *Nautilus* are programs for automated protein and nucleotide building, respectively. They are provided with a density map, target sequences and optionally an existing model, and they attempt to automatically build a model to fit the map.

Resolution

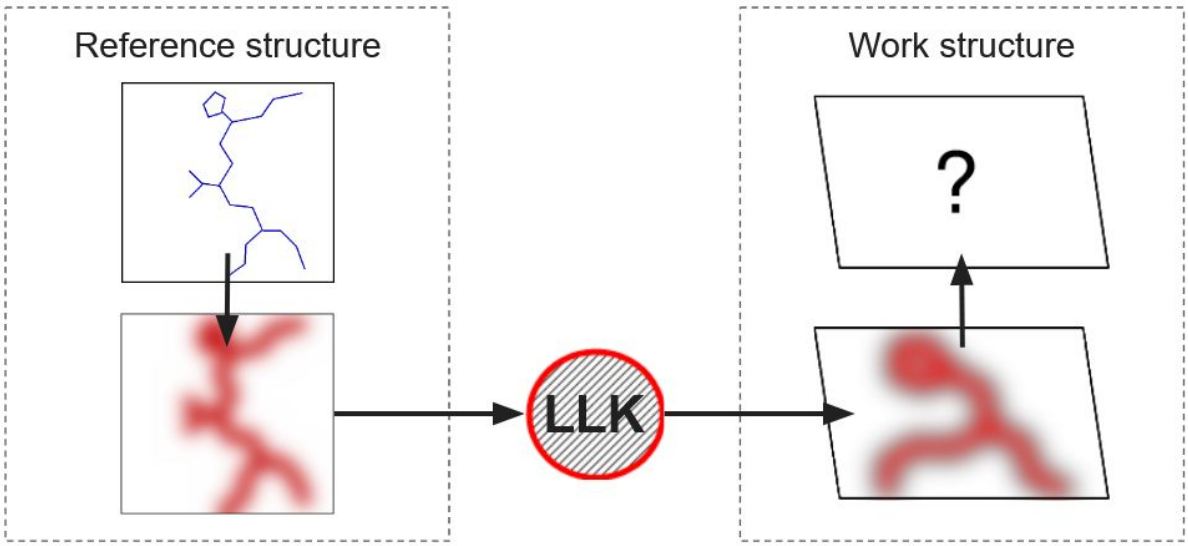
Kathryn Cowtan



A big problem with automated model building is 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. The local resolution gives an indication of the map quality in that region. Tryptophan has a large, rigid side chain that makes it easier to spot in the density. At lower resolutions, most side chains are not identifiable without previous knowledge of the sequence.

Buccaneer - LLK Target Function

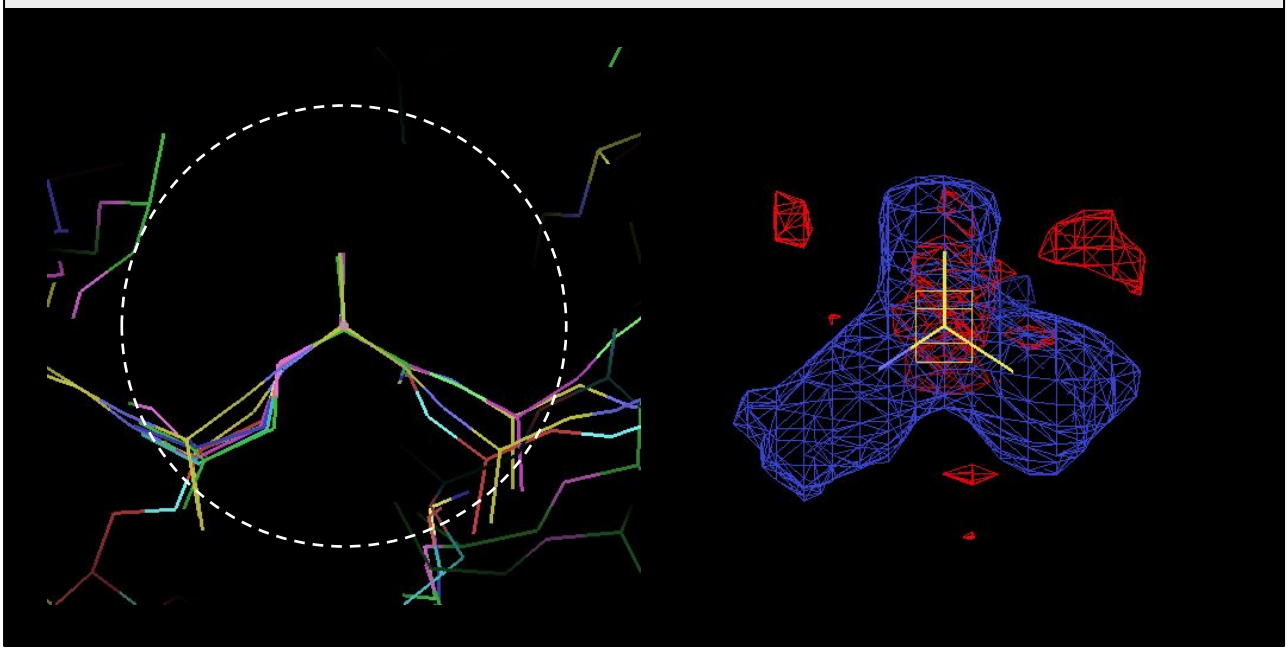
Kathryn Cowtan



Buccaneer uses a known reference structure in order to help it build at both high and low resolution. 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 by creating a log-likelihood (LLK) target. The LLK target is then used to search for the same features in the work map. One of the changes made to improve performance with EM data is the change of the default reference map model/map used to an EM map/model.

Buccaneer - LLK Target Function

Kathryn Cowtan



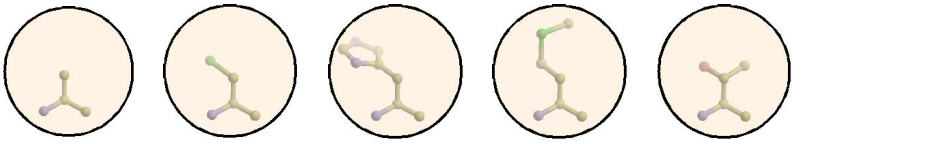
The target to search for individual residues is obtained by superimposing all the residues in the reference structure on top of each other. The mean and variance of the density is then used to construct a likelihood target within a 4 Å sphere. The blue density shows regions of conserved high density 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/orientation for a residue.

Buccaneer - LLK Target Function

Kathryn Cowtan

Finding, Growing Cα environment (4.0Å sphere)

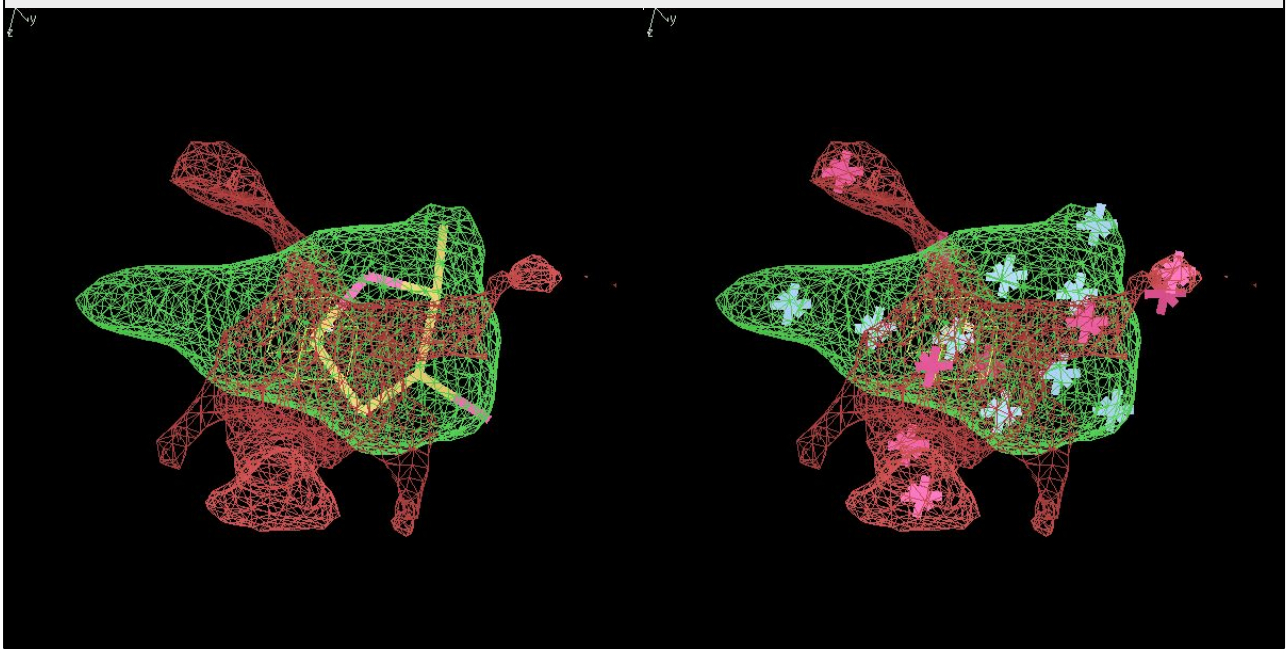
Sequencing Cβ environment (5.5Å sphere)



The 4 Å 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 5.5 Å 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.

Nautilus - Fingerprints

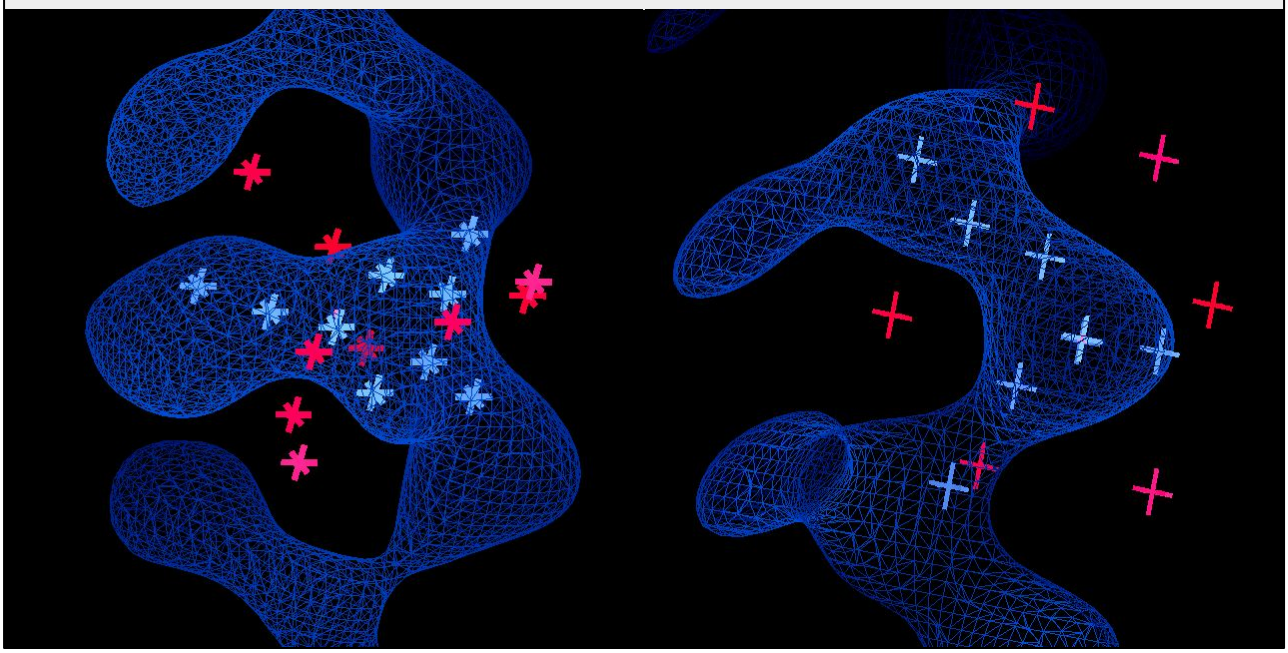
Kathryn Cowtan



Like *Buccaneer*, *Nautilus* also identifies residue positions by looking for patterns of conserved high and low density, but it does so using a fingerprint detection method instead of simulating a map for a reference structure. The green map shows conserved high density for overlapped sugars. The the first part of the base is visible but phosphates aren't because they have multiple possible conformations. The red map shows conserved regions of low density. Probe points are placed at positions where high and low density are expected. This is a faster method than having a full LLK target function because it only looks at a few key points.

Nautilus - Fingerprints

Kathryn Cowtan



The fingerprints are translated and rotated in the map to find positions that have high and low density in the right places. The same process was followed to create a fingerprint for phosphate. It has high probe points on the phosphate atoms and in the neighbouring sugars and low probe points in the surrounding regions. This shows the sugar and phosphate fingerprints in place over some example density.

Buccaneer - Steps

Kathryn Cowtan

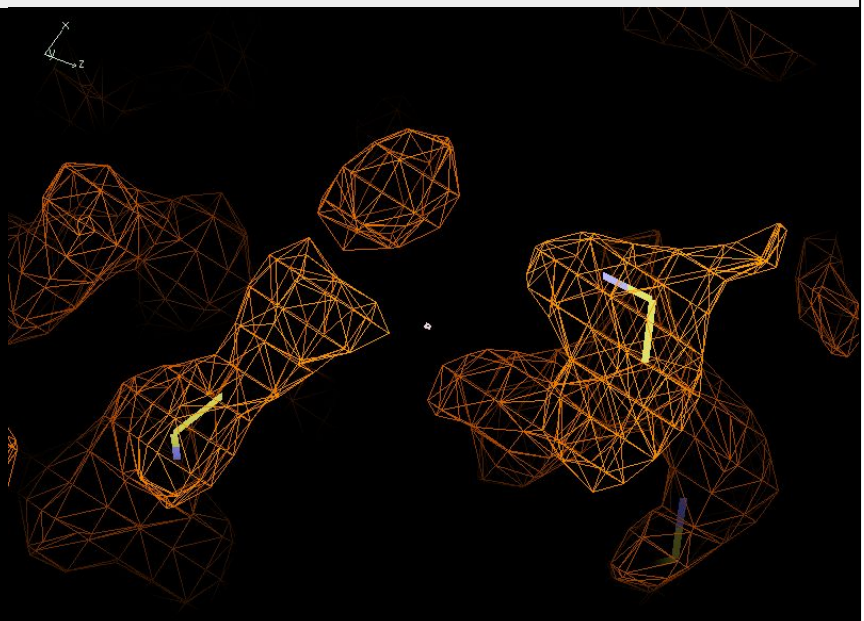
1. **Find** oriented residue positions
2. **Grow** each residue into a chain fragment
3. **Join** overlapping fragments and resolve branches
4. **Link** nearby termini
5. **Sequence** the chains
6. **Correct** insertions and deletions
7. **Filter** to remove short chains
8. **NCS** superposition to extend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

Buccaneer and *Nautilus* both run as a cyclic calculation with multiple internal steps. These are the 10 steps in *Buccaneer*. The steps for *Nautilus* are similar.

Buccaneer - Example

Kathryn Cowtan

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

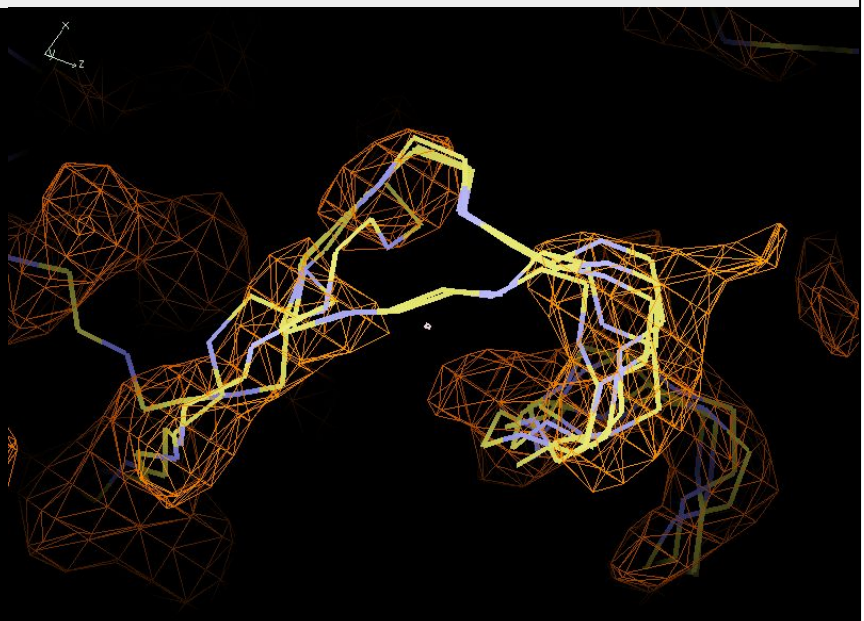


This is an example of *Buccaneer* building a difficult loop in a 2.9Å X-ray map. The first step is to find possible positions for new residues. It has placed residues at three positions in this figure (two in the foreground and one in the background) where the density has the right shape according to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

Buccaneer - Example

Kathryn Cowtan

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

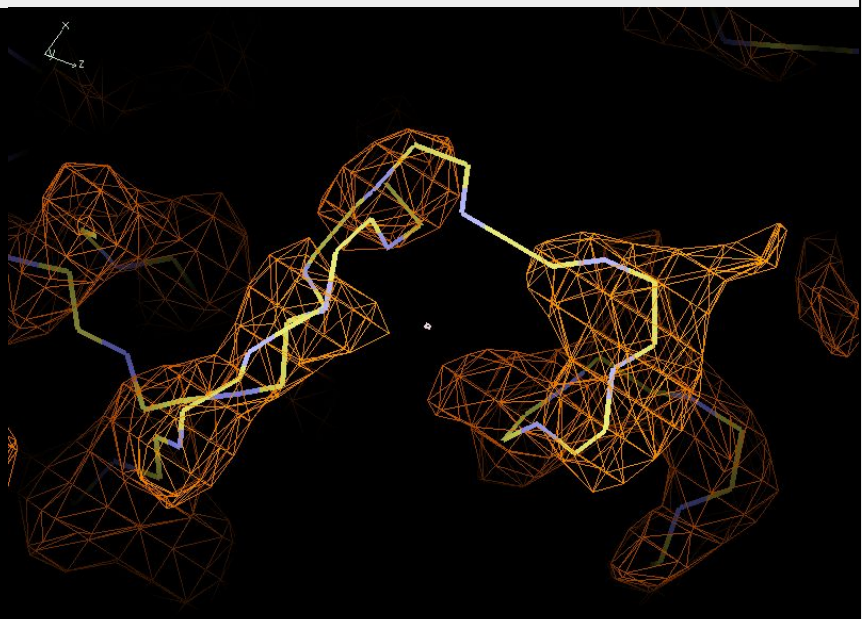


The next step is to grow each residue into a long chain fragment. New residues get added at both ends using an exhaustive search over allowed Ramachandran angles and the same C-alpha target. Once the target score drops below a threshold value the growing stops. This step gives lots of fragments, many 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

Kathryn Cowtan

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

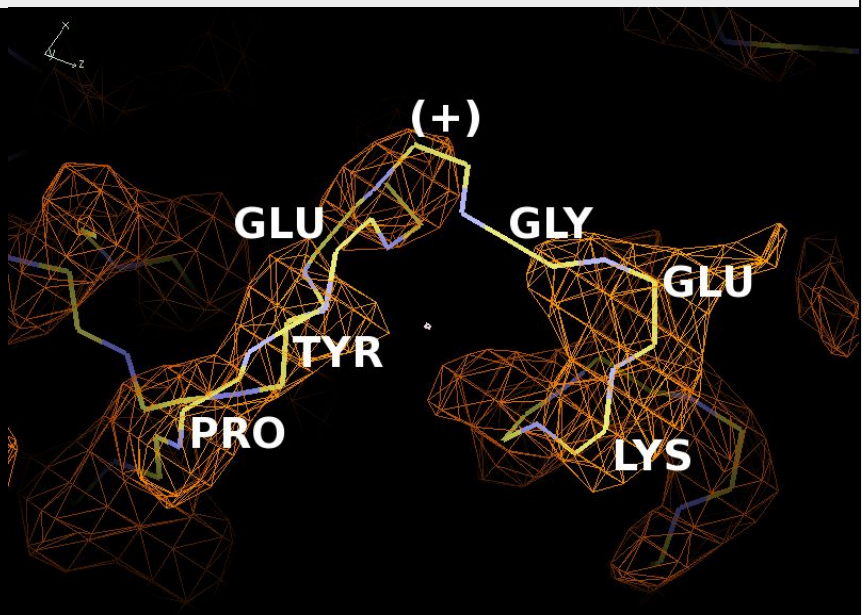


The joining step merges all the chain fragments that agree with each other into single chains. This step also has to make decisions about which routes to follow. It tries to build the longest chains (e.g. it chose the longest route over the middle loop), which helps to build helices correctly. The chain 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

Kathryn Cowtan

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

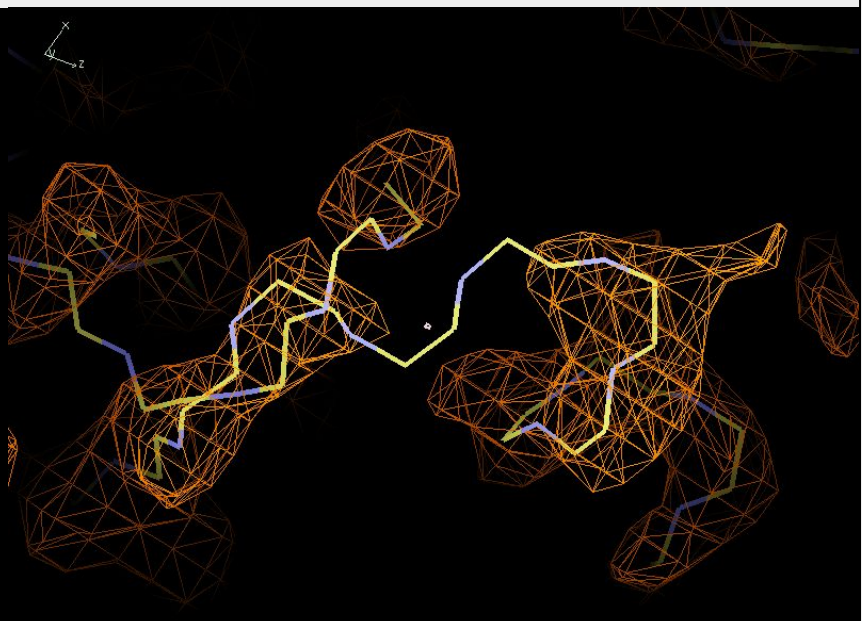


We have skipped the linking step because there are no termini that can be linked in this view. The sequencing step works by assigning each residue a probability that it is each of the 20 residue types using the C-beta targets. These probabilities are then compared to the known sequence(s) to look for 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

Kathryn Cowtan

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

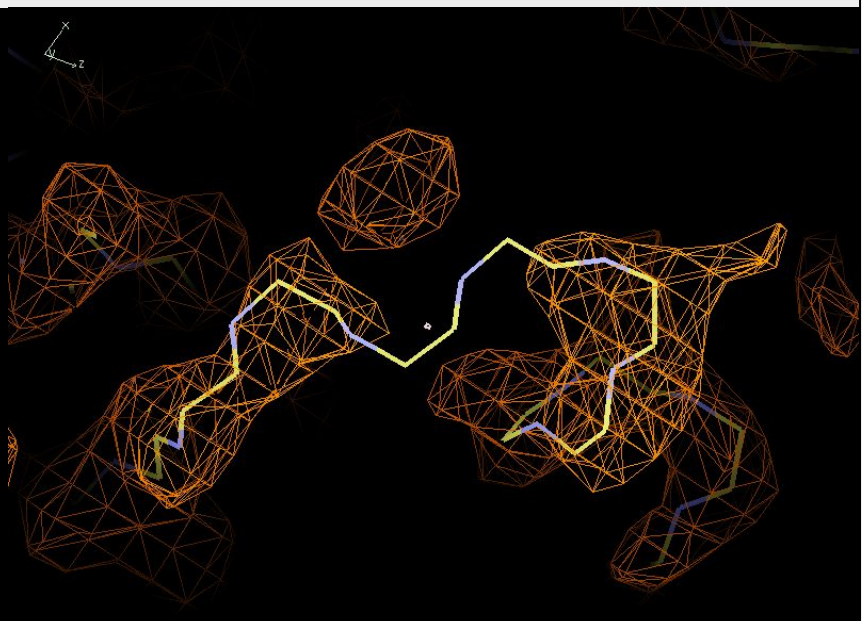


The next step is to correct any insertions (where it has built one extra residue) and deletions (where it has built one fewer residue) found during the sequencing step. The insertion that was over this loop as been corrected by removing three residues and rebuilding two.

Buccaneer - Example

Kathryn Cowtan

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

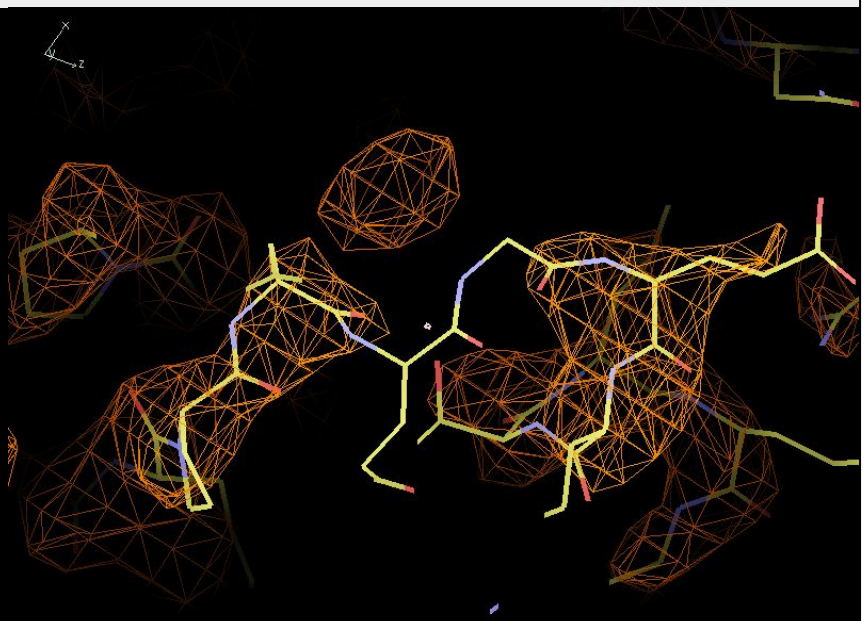


The pruning step removes residues from clashing chains to produce a single consistent model. If the pruned chains have less than 6 residues remaining they are removed too. Pruning is done at this late stage to have the most information about which chain is correct. Residues that are unsequenced or are 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

Kathryn Cowtan

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

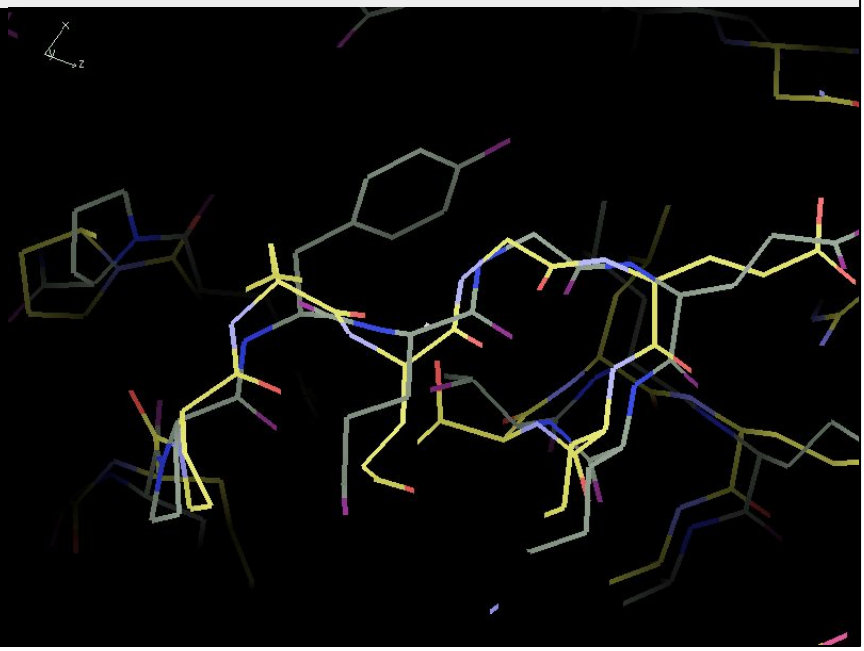


Finally, the last step is to build side chains for each residue using a rotamer library. The rotamer with the highest average density at the atomic positions is chosen. If two side chains end up clashing, *Buccaneer* will try and find a pair of rotamers that do not clash. If this fails both residues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

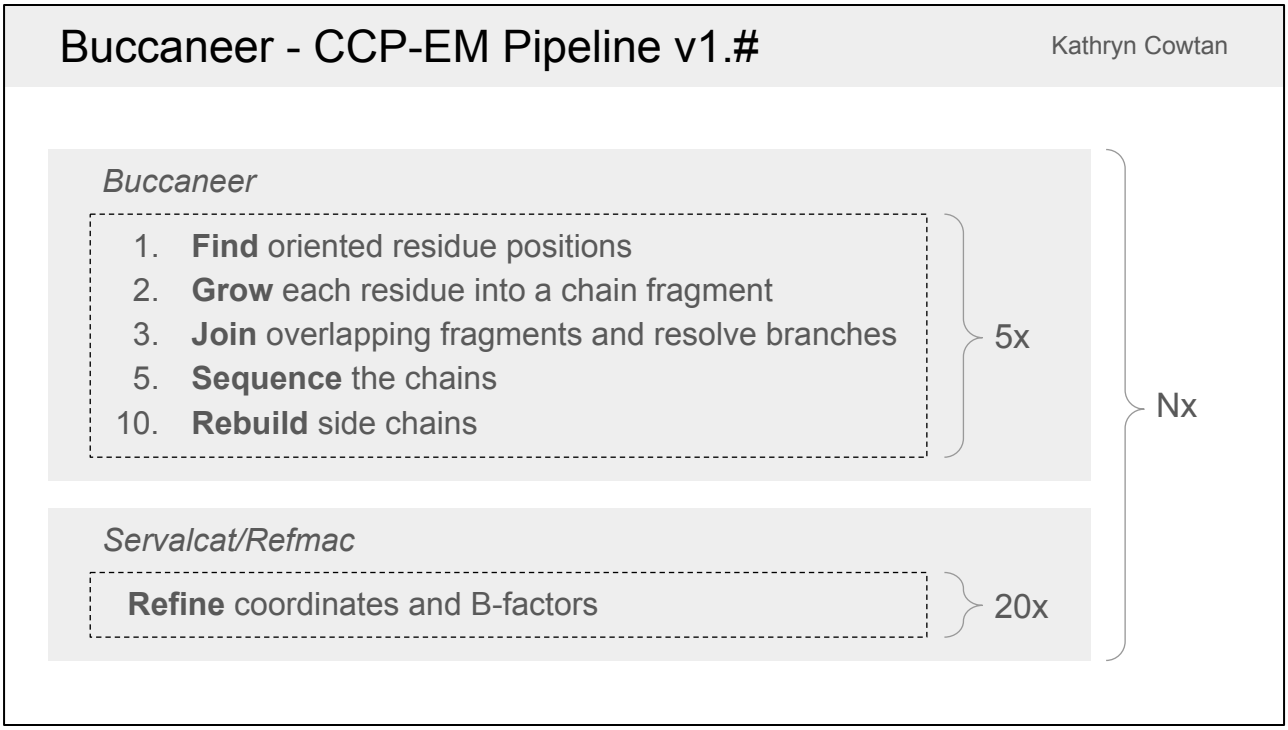
Buccaneer - Example

Kathryn Cowtan

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 and *Nautilus* do not perform global refinement of the model, so the graphical interface actually runs a pipeline that combines them with *Refmac or Servalcat*, which refine the model coordinates and B-factors and produces an updated map for further building. The first version of the pipeline in CCP-EM runs 5 cycles of *Buccaneer* or *Nautilus* followed by 20 cycles of *Refmac* and repeats that N times. Users then have to choose which intermediate model to continue with.

ModelCraft - Pipeline

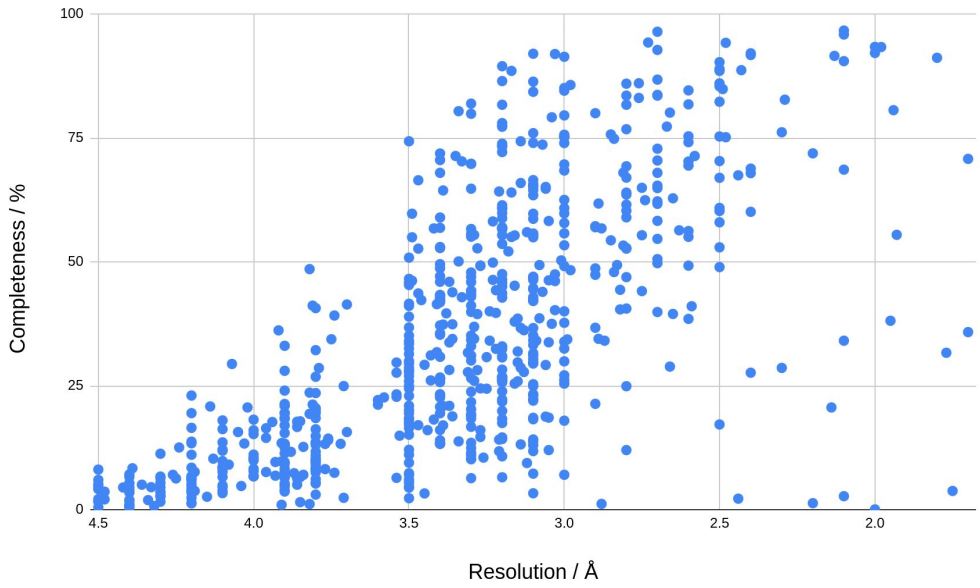
Shift-field refinement (*Sheetbend*) - *Refmac*	X-ray
1. Prune residues (*Coot*) - *Refmac*	X-ray
2. Density modification (*Parrot*)	X-ray
3. Add dummy atoms (*Coot*) - *Refmac*	X-ray
4. Build protein (*Buccaneer*) - *Refmac/***Servalcat**	**X-ray/EM**
5. Build RNA & DNA (*Nautilus*) - *Refmac/***Servalcat**	**X-ray/EM**
6. Combine protein & NA - *Refmac*	X-ray
7. Prune chains (*Coot*) - *Refmac*	X-ray
8. Add waters (*Coot*) - *Refmac*	X-ray
Rebuild side chains (*Coot*) - *Refmac*	X-ray

≤ 25x

ModelCraft is a new pipeline that was created to help build more structures in X-ray crystallography when the initial phases are poor. There are lots of extra steps for phase improvement that can't be used for EM data, but it has the advantage over the previous pipeline of running both *Buccaneer* and *Nautilus* to build protein, RNA and DNA at the same time. By default, it runs for up to 25 cycles but will stop if FSC average has not improved in 4 cycles.

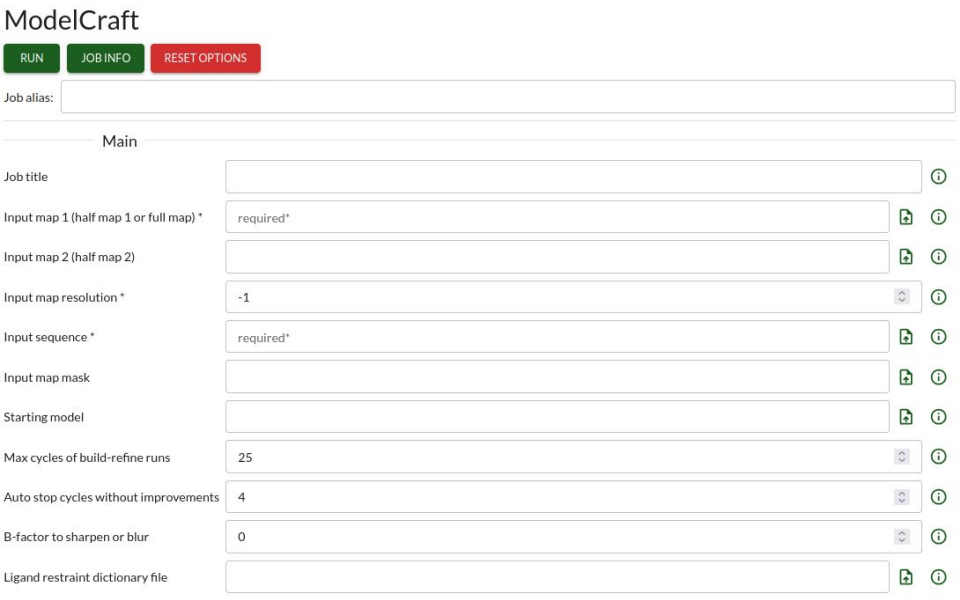
ModelCraft - Results

Soon Wen Hoh



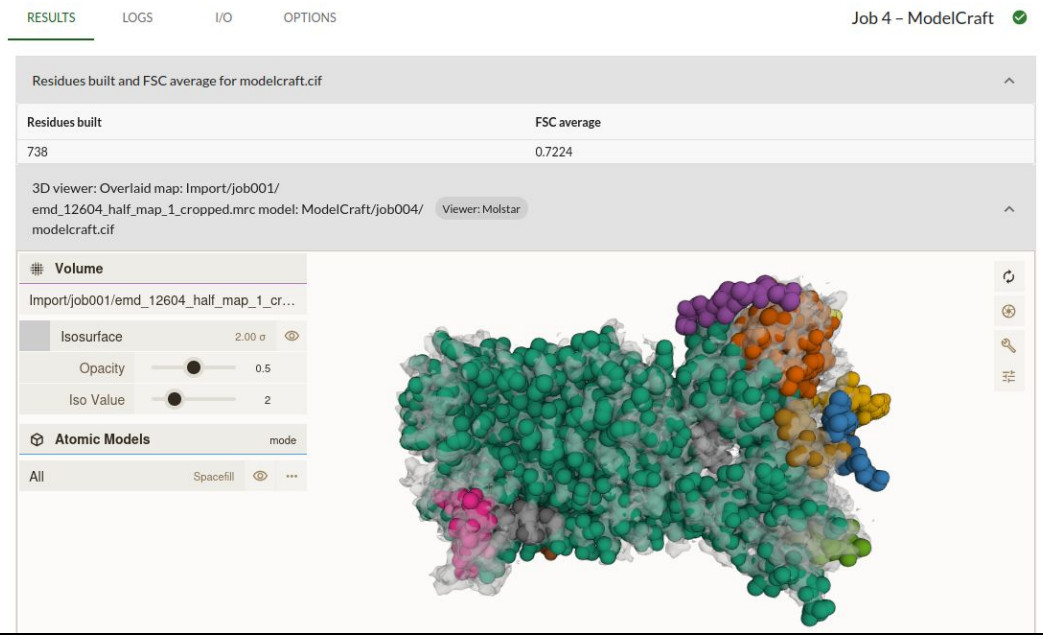
This chart shows the completeness of protein models built by ModelCraft as a function of the overall resolution of the map. Completeness was calculated as a percentage of residues in the deposited model that have N, CA and C all within a 1 Å cutoff of a residue in the ModelCraft model. Results vary by map quality and local resolution, and providing a starting model is likely to improve results. These results may not be fully representative of the latest version of ModelCraft with default settings. ModelCraft and other model-building programs can be quite sensitive to the map sharpening, if they fail to build it can help to optimise the map first through post-processing methods.

CCP-EM Doppio



This shows the input page in the CCP-EM Doppio interface. It requires either two half maps (recommended) or one full map, the resolution of the map and a sequence file that can contain a number of protein, RNA and DNA sequences. You can optionally provide a mask to trim the maps for building and refinement. The mask needs to have the same grid dimensions as the input map(s). When no mask is provided, *EMDA mapmask* will be used to automatically create a mask from the map, but this may have errors. A starting model is optional but it is likely to help with building if you have one. Protein, RNA, DNA and water residues will be rebuilt but other residues will be kept fixed. The default of maximum number of cycles to run is 25. However, the job will stop automatically when there is no improvements in FSC average for 4 cycles. Setting auto-stop cycles to 0 will cause the job to run the maximum number of cycles. If your starting model has a ligand without restraints in the monomer library you should provide a custom dictionary for refinement.

CCP-EM Doppio



The results tab for a ModelCraft job displays the number of residues built and the FSC average of the output model. It also has tabs with a viewer of the output model and graphs of FSC average and residues built per cycle.

ModelCraft - Command Line

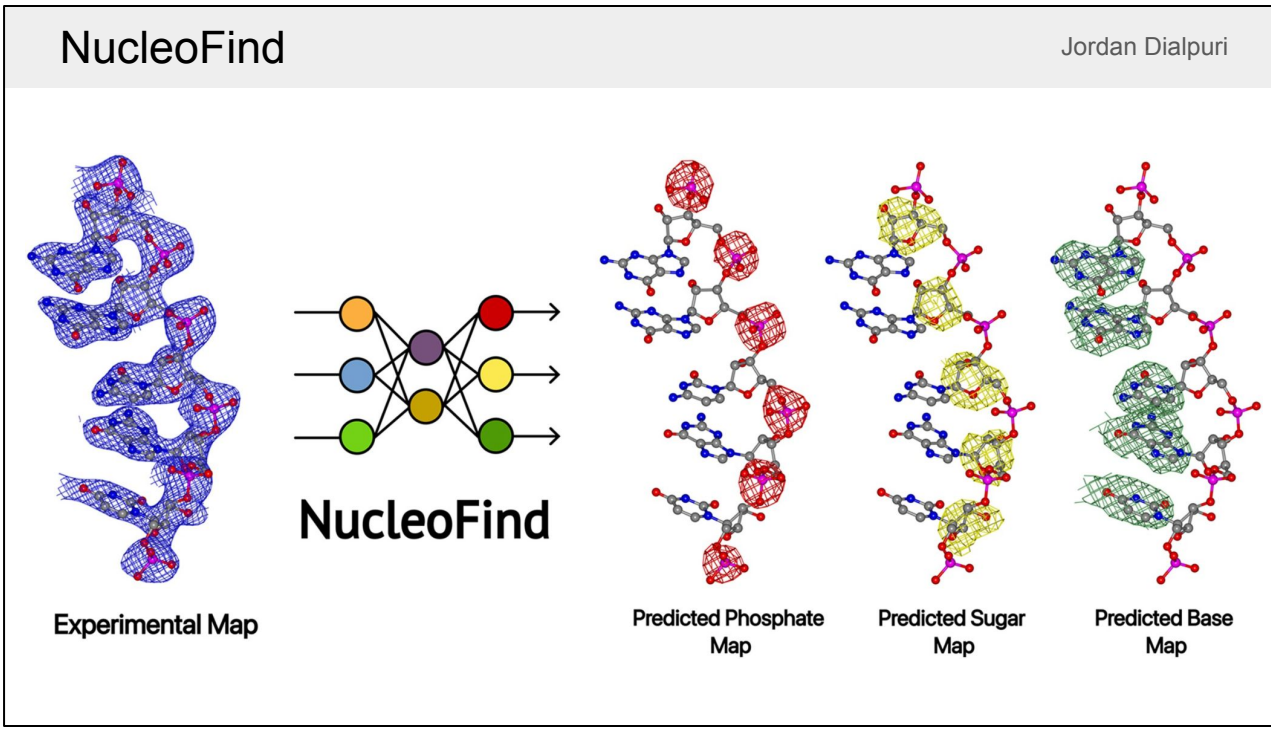
```
$ modelcraft --version
4.0.2

$ modelcraft em \
> --contents sequences.fasta \
> --map map.mrc \
> --resolution 3.0 \
> --model starting_model.cif \
> --mask mask.mrc
```

```
$ modelcraft em \
> --contents sequences.fasta \
> --map half1.mrc half2.mrc \
> --resolution 3.0 \
> --model starting_model.cif \
> --mask mask.mrc

$ modelcraft em --help
```

ModelCraft can also be used on the command line.



One future development is coming to ModelCraft is improved building of nucleic acids using *Nucleofind*, which is a deep-learning algorithm for locating phosphate, sugars and bases in maps.

Other Programs

map_to_model
dock_and_rebuild
PredictAndBuild

Kiharalab
MainMast
DeepMainMast
CryoRead
DiffModeler
ComplexModeler

It is recommended to try many automated model-building programs as some may do better with your map than others. It may also be useful to combine the results from multiple programs. This slide shows some of the other programs available. *ARP/wARP* was originally developed for high-resolution X-ray crystallography, but new methods for building into cryo-EM maps were added in version 8.0. *Phenix* has a number of tools for building into cryo-EM maps: *map_to_model* uses more traditional model-building algorithms adapted for cryo-EM, *dock_and_rebuild* fits a predicted model into the map then morphs/rebuilds it, and *PredictAndBuild* iterates this docking and rebuilding process with feedback to the actual model prediction step using templates. *DeepTracer* uses neural networks to identify protein features in maps. *ModelAngelo* does the same thing to find protein C-alpha or nucleotide phosphate positions but then uses those in a graph neural network to refine the structure. The Kihara lab has multiple programs for building. *MainMast* was the first one using statistical methods, then *DeepMainMast* adapted this to use deep-learning methods. *CryoRead* uses deep learning for nucleotide building. *DiffModeler* is the newest program using a diffusion model for protein backbone tracing and *ComplexModeler* combines *DiffModeler* and *CryoRead*.

Acknowledgements

Paul Bond

Supervisors
Kathryn Cowtan
Keith Wilson

Cowtan Group
Jordan Dialpuri
Soon Wen Hoh
Stuart McNicholas

Collaborators
Charles Ballard
Tom Burnley
Eleanor Dodson
Maria Fando
Matt Idanza
Agnel Joseph
Ronan Keegan
Oleg Kovalevskiy
Eugene Krissinel
Andrey Lebedev
Rob Nicholls
Martin Noble
Colin Palmer
Marcin Wojdyr
Keitaro Yamashita

Program authors
Buccaneer
CCP4 Cloud
CCP4i2
CCP-EM
Coot
CTRUNCATE
EMDA
Gemmi
Nautilus
Parrot
Refmac
Servalcat
Sheetbend