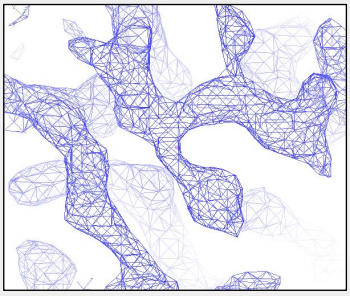
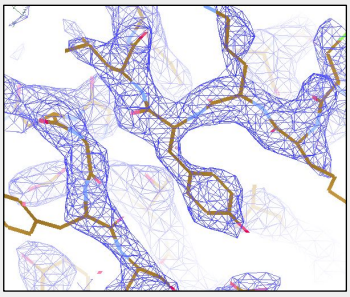
Automatic Model Building

CCP4 / CCP-EM

Paul Bond
University of York

paul.bond@york.ac.uk

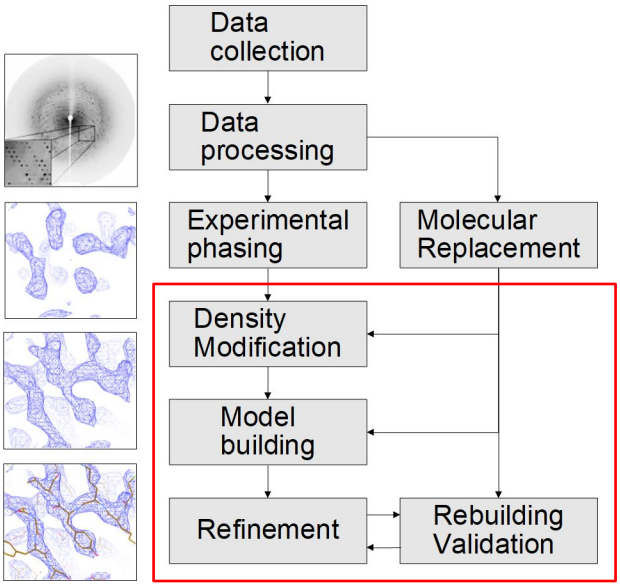




This presentation is on automatic model building in CCP4 and CCP-EM. It should explain what programs are available and some details on how they work, but information about how to run them is left out as that is covered in separate practical sessions.

X-ray Crystallography

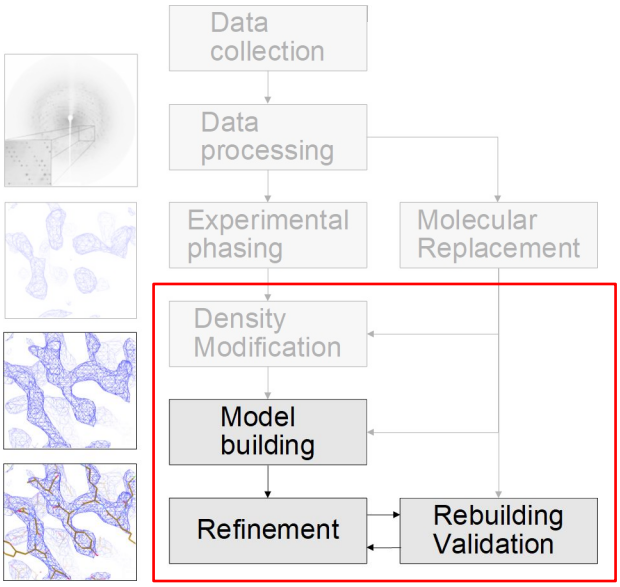
K Cowtan



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 (MR). If your MR 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. Automated model-building programs try to automate the steps in the red box as much as possible.

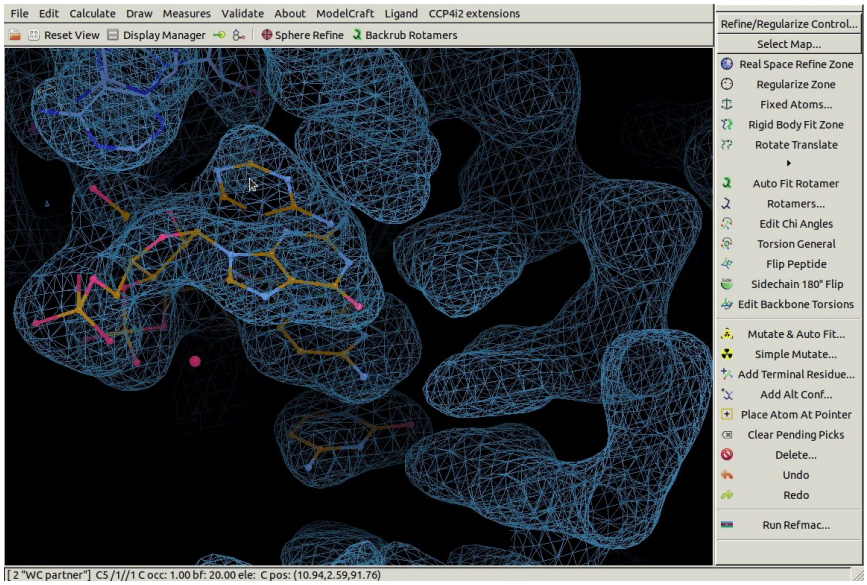
Cryo-EM

K Cowtan



Cryo-EM has different data processing stages, but the result is still a map that needs a model building into it. There is still some density modification done in the form of sharpening and blurring, but the map is not changed during automated model building.

Interactive Model Building



This talk will only cover fully-automated model-building programs where the user has no interaction with the model. There are also interactive model-building programs such as Coot and Isolde that contain many automated tools.

ARP/wARP

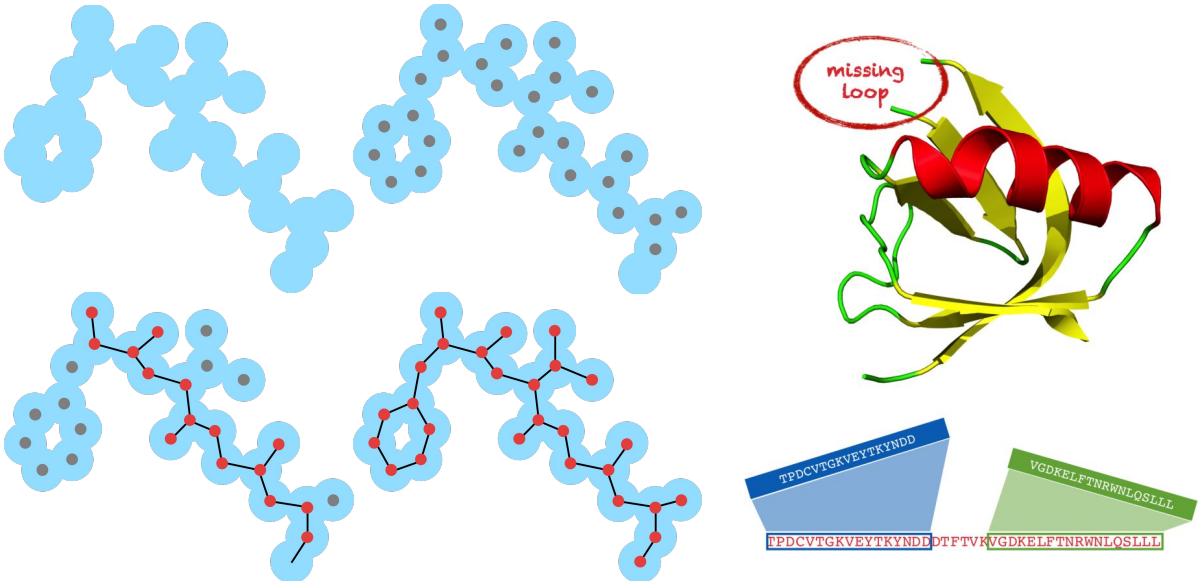
Victor Lamzin
Grzegorz Chojnowski
EMBL Hamburg

Anastassis (Tassos) Perrakis
Netherlands Cancer Institute

The first program we will look at is ARP/wARP, which was the first program to provide fully-automated model building from an empty map to a complete model. Thank you to Grzegorz Chojnowski who provided these slides in the presentation.

ARP/wARP - Method

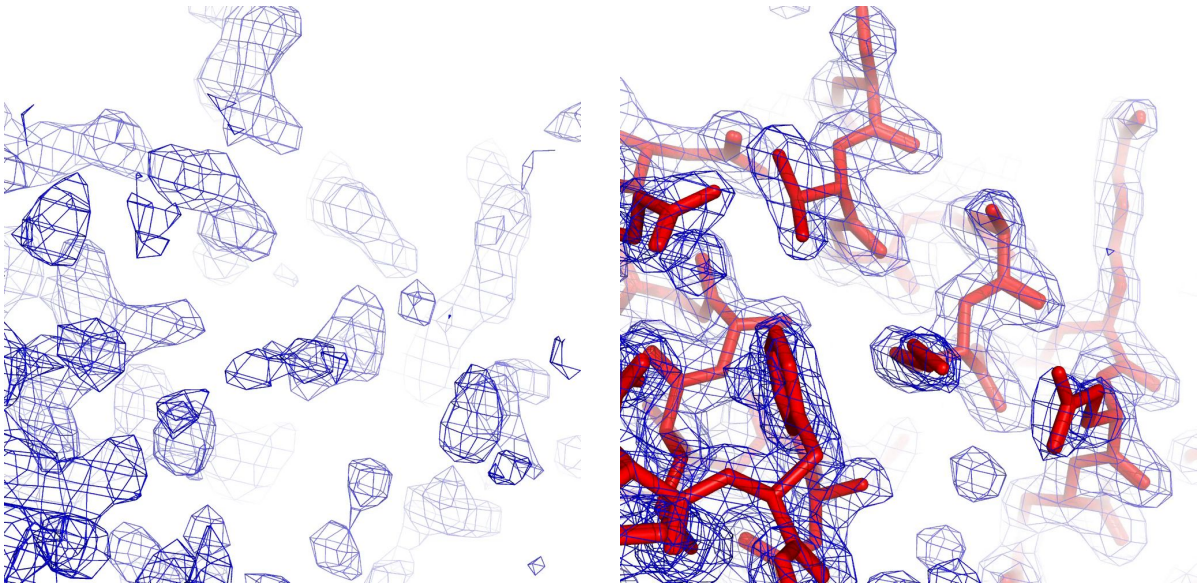
Grzegorz Chojnowski



The first step in ARP/wARP is to build and refine a free-atom model into the map (where the atoms do not have any chemical identity). The main-chain is then traced through the free atoms, then the chains are sequenced and side chains are built. The process is repeated in cycles so you will have a mixed model that contains both built residues and free atoms. Once the chains have been assigned to the sequence, missing regions are identified and built.

ARP/wARP - Model expansion

Grzegorz Chojnowski

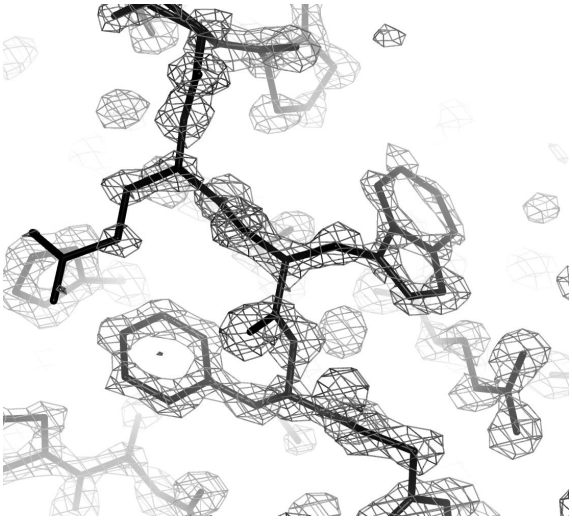


Here is a real example at 2.0 Å resolution showing the input and output of ARP/wARP. It has gradually added free atoms to the map, improved their positions and interpreted them as an atomic model of a protein. You can see by comparing the maps on the left and right that the quality of the map itself also gradually improves throughout this process as the new model provides new phases (in X-ray crystallography).

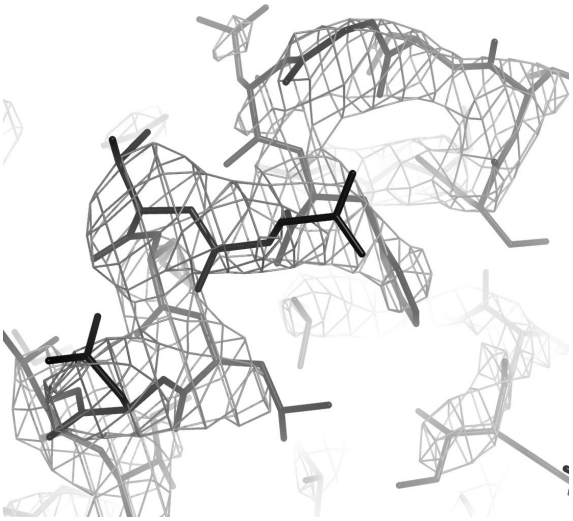
ARP/wARP - Resolution

Grzegorz Chojnowski

1.5 Å : 16 reflections per atom



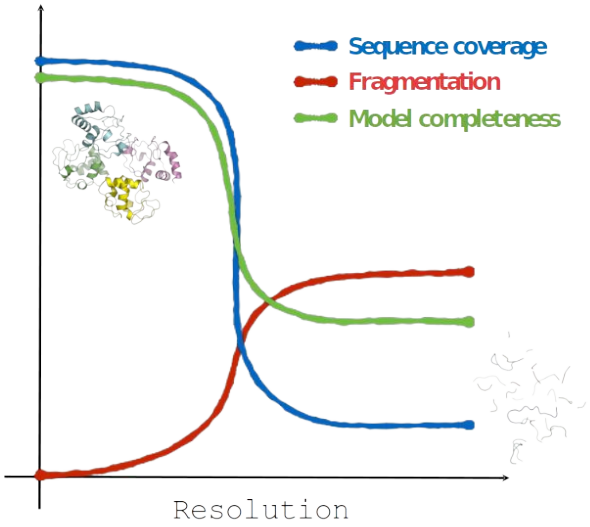
3.0 Å : 2 reflections per atom

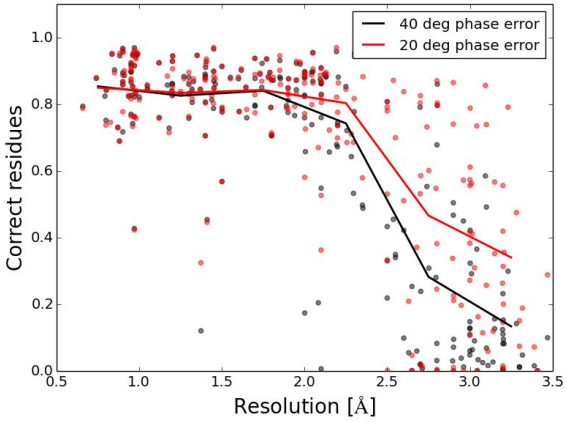


The problem with this approach is that at low resolution there is not a much information about individual atoms. At 1.5 Å resolution there might be ~16 reflections per atom but at 3.0 Å resolution there are probably only ~2 reflections per atom. This means that each atom will not have it's own peak in the map and so getting accurate free-atom positions is less likely.

ARP/wARP - Resolution

Grzegorz Chojnowski

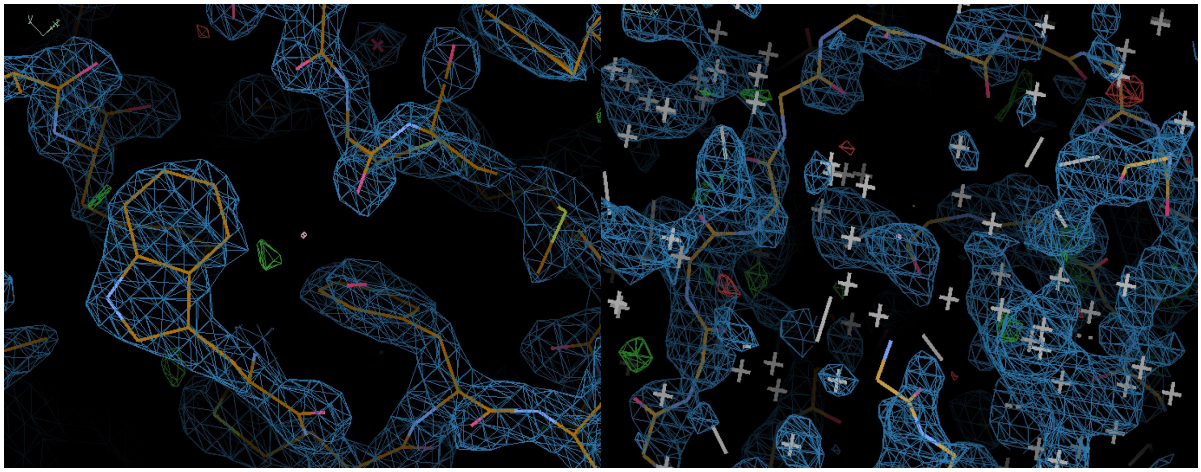




At low resolution the model becomes less complete, more fragmented and it is more difficult to assign the correct sequence to the parts that have been built. The chart on the right shows some real examples at different resolutions. With resolutions better than 2.0 to 2.5 Å, classic ARP/wARP is usually able to build a very complete model even if the phases are initially quite poor. It is still possible to build some of the model, but there is more chance of this with more accurate phases.

ARP/wARP - Resolution

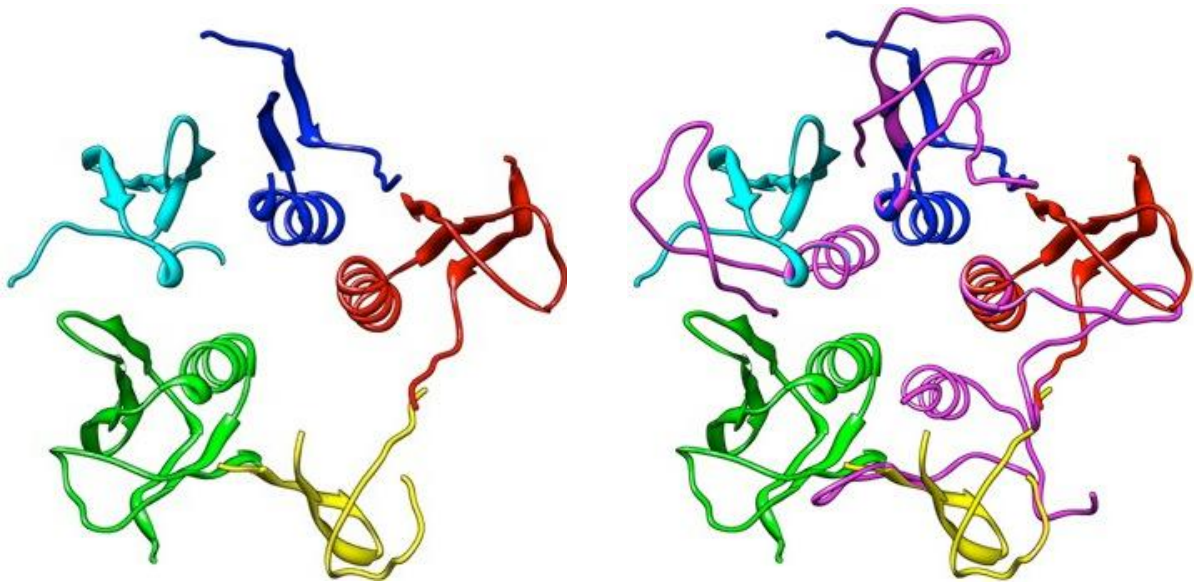
Grzegorz Chojnowski



The model on the left shows a good model from ARP/wARP. If the model reaches a certain quality the free atoms are removed from the output structure. On the right is an example of a poor model from ARP/wARP. The model is very fragmented and there are also a lot of unidentified free atoms, although some of the structure may still be correct.

ARP/wARP - NCS detection and extension

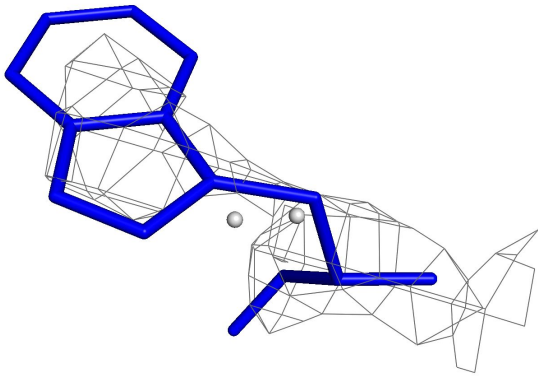
Grzegorz Chojnowski

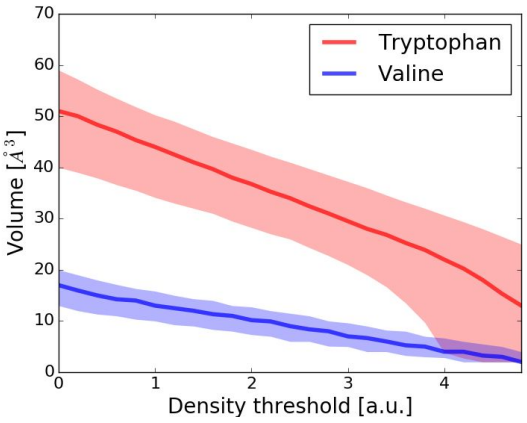


There have been a number of improvements to the classic ARP/wARP method. One is able to detect that partially-built chains are non-crystallographic (NCS) copies of each other and use the better copies to help complete the worse ones.

ARP/wARP - ML sequence assignment

Grzegorz Chojnowski

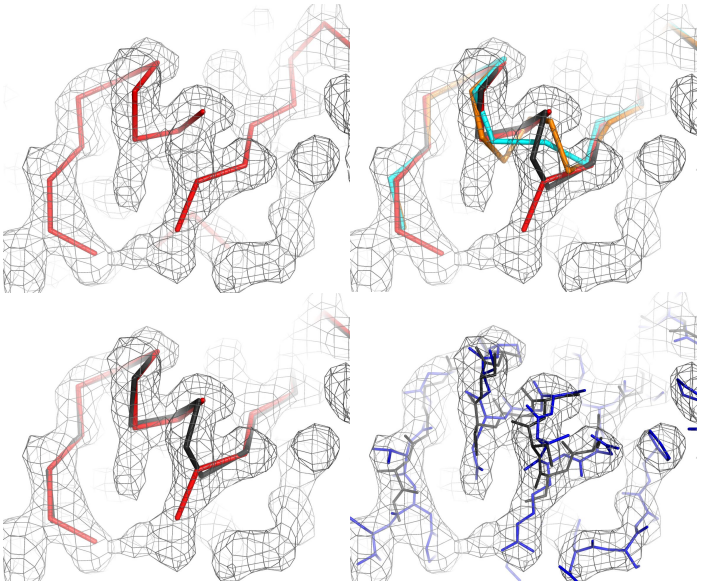




ARP/wARP used to decide the side chain type by looking at the patterns of free atoms in the density. However, this doesn't work very well at low resolution. The newer algorithm looks at the change in the density volume at different density thresholds (Chojnowski et al. 2019). In the chart we can see that the volume profiles of tryptophan and valine are quite different. The algorithm looks at the volume profile for the side unknown side chain and converts this into a probability for each side chain type.

ARP/wARP - Sequence-free loop building

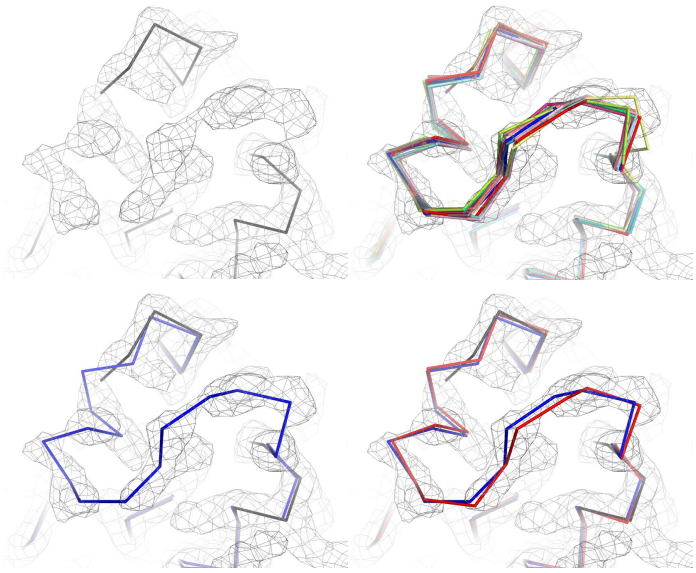
Grzegorz Chojnowski



ARP/wARP has a sequence-free loop building algorithm. Short fragments from a purpose-built library are superposed over the intermediate models. This algorithm is able to correct fragments that were built in the reverse direction (Chojnowski et al. 2019).

ARP/wARP - Homology-based extension

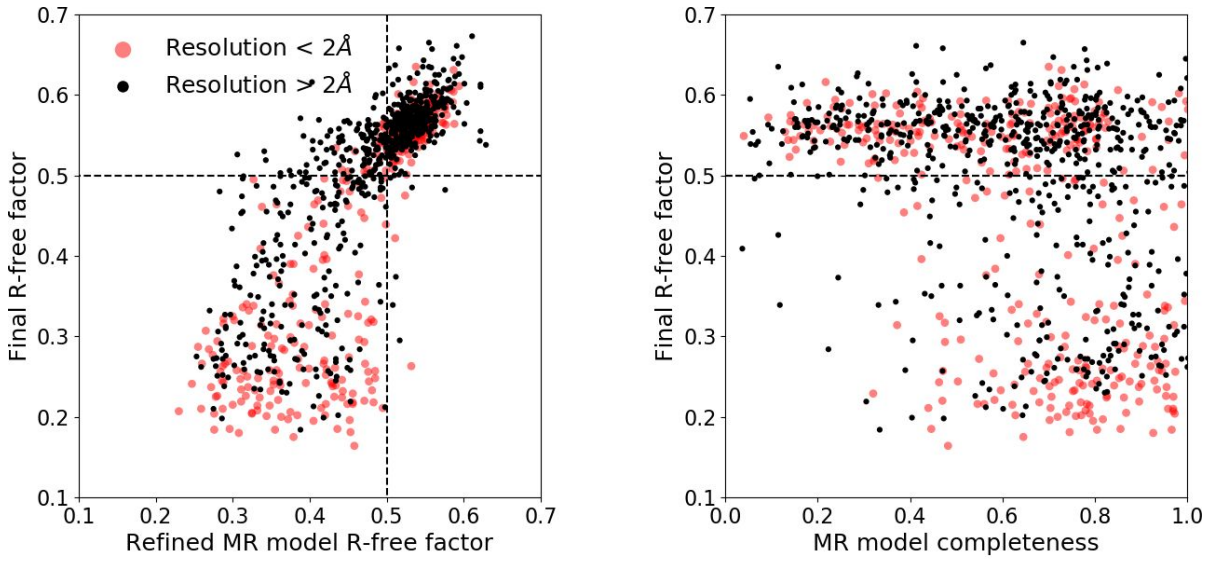
Grzegorz Chojnowski



There is also a newer algorithm that uses the sequence information of the model. It searches for structures that are similar in sequence and uses GESAMT to align the structures locally to help complete the model (Chojnowski et al. 2020).

ARP/wARP - Results (X-ray)

Grzegorz Chojnowski



This slide shows some results from ARP/wARP. The chart on the left compares the free-R factor of molecular replacement (MR) models before and after ARP/wARP. If the MR model has an R-free below 0.5 and the resolution is better than 2.0 Å then there is a good chance that ARP/wARP will be able to improve the model. If the resolution is worse 2.0 Å then a better initial MR model may be needed. The chart on the right shows the completeness of the initial MR model against the free-R factor after ARP/wARP. If the MR model has a very low completeness then the phases may be too poor for ARP/wARP to improve the model.

ARP/wARP - Tips and tricks

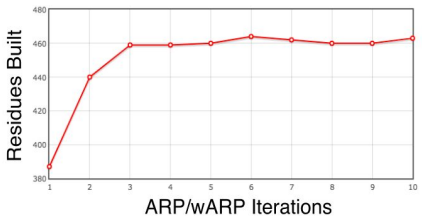
Grzegorz Chojnowski

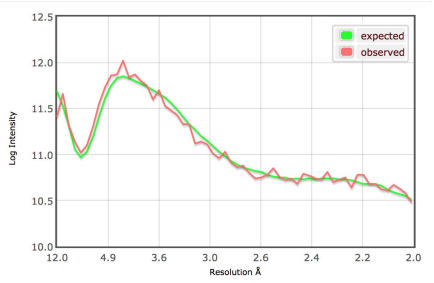
Resolution
best results above 2.6 Å
but phases can be poor

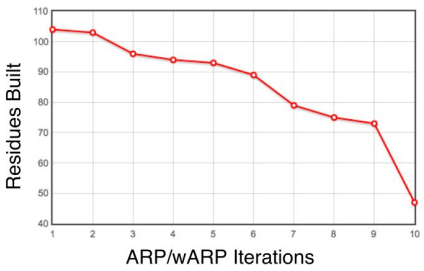
Space group
double check if
model building fails

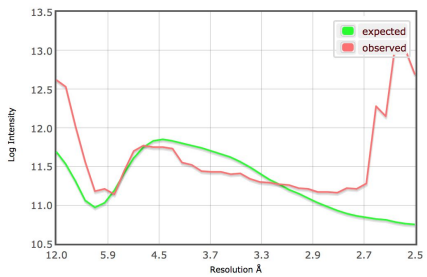
Data completeness
as high as possible

Data quality
(outliers, ice rings)
always check Wilson plot

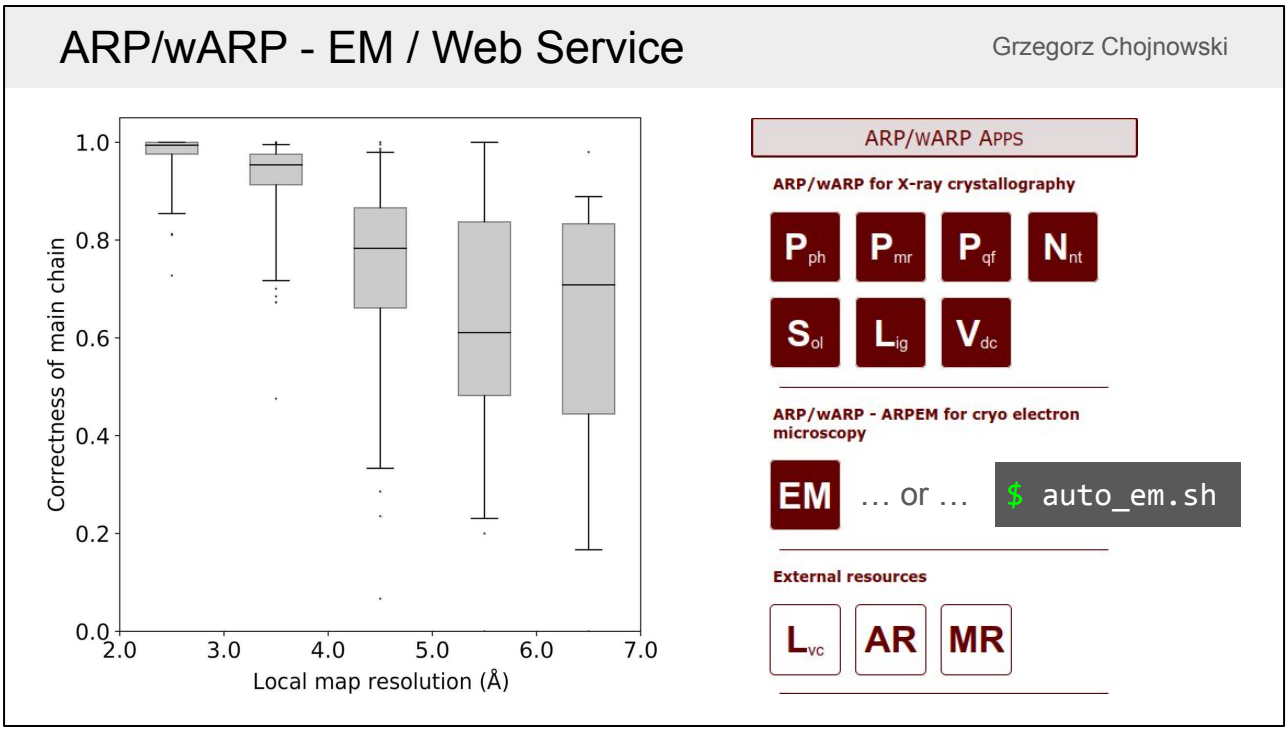




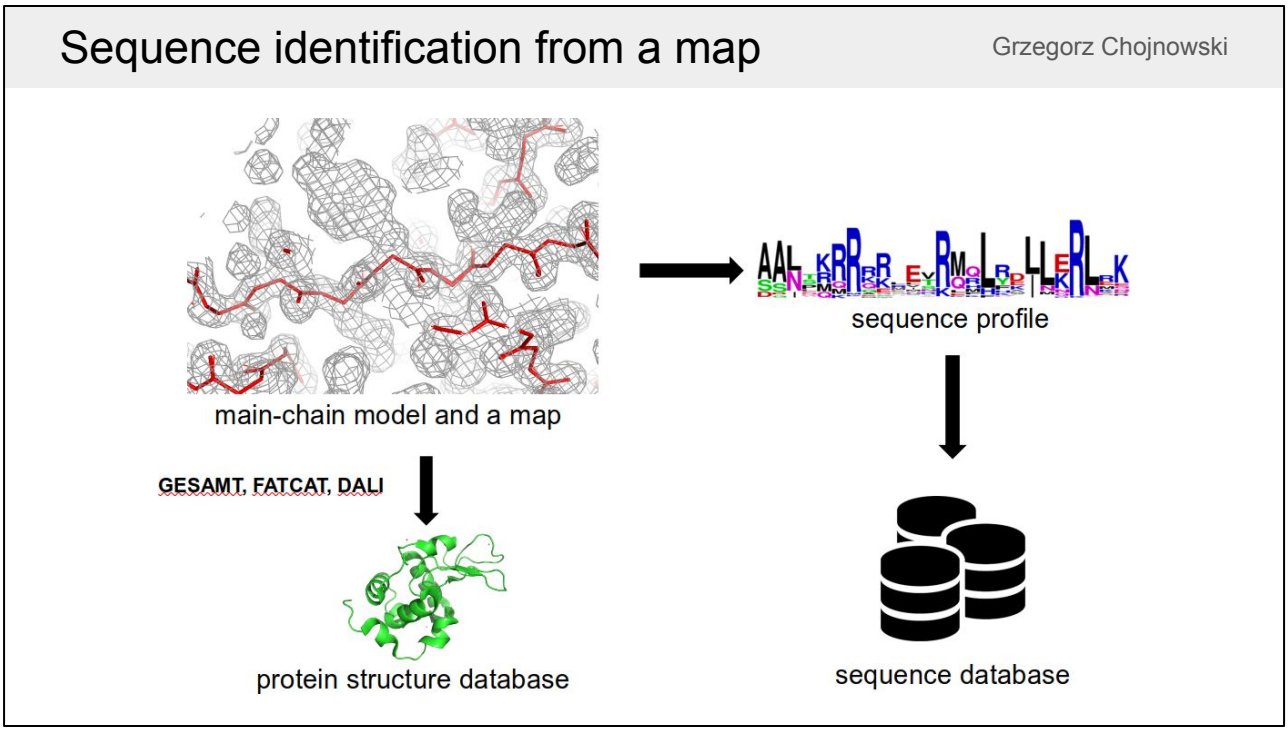




Classic ARP/wARP works best when the resolution is better than 2.6 Å, but when the resolution is high it is able to build even if the initial phases are quite bad. The top left chart shows a good ARP/wARP run where the number of residues built goes up with the number of iterations. The bottom left chart shows what can happen if it does not work well. If you get a poor result but have high resolution data it a common error is that the space group is incorrect. It may be worth switching to a lower symmetry space group or trying different enantiomorphs if possible. ARP/wARP also can be sensitive to both the completeness and the quality of the data. The Wilson plot can indicate whether there are problems at specific resolutions. The top and bottom right charts show examples of good and bad Wilson plots.



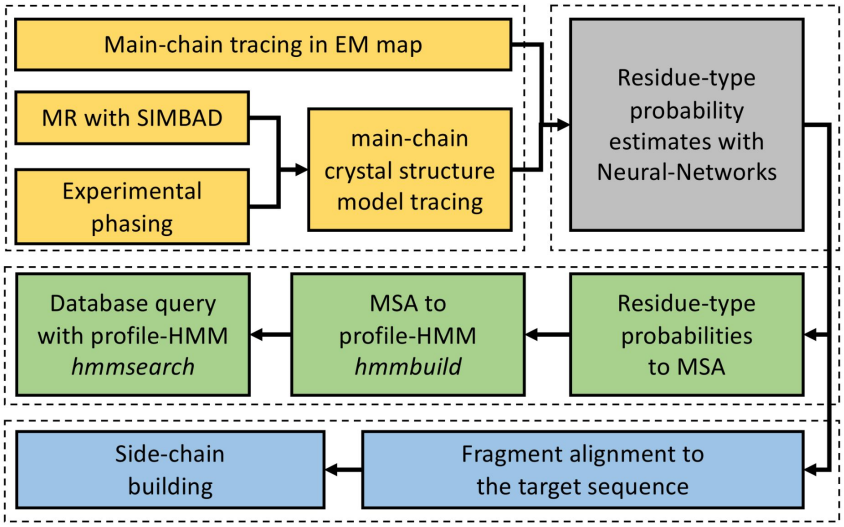
ARP/wARP 8.0 has improvements to build at lower resolutions and is able to build into cryo-EM maps. It builds nearly all of the main-chain correctly when the local resolution is 4.0 Å or better (Chojnowski et al. 2021). ARP/wARP for EM is not available in the CCP-EM graphical interface but it is distributed with CCP4 and can be used on a terminal with `auto_em.sh` or can be used through the ARP/wARP Web Service.

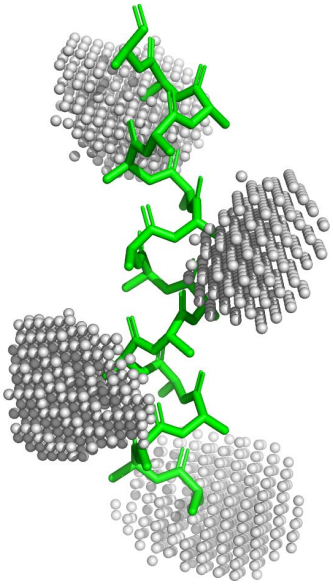


One useful thing to note is that ARP/wARP is able to build the backbone of a model without requiring a known sequence. There are two options to identify the protein from a backbone trace: you can use a program such as GESAMT, FATCAT or DALI to look for structural homologues using a 3D structure search or you can create a sequence profile from the density and search a sequence database. Grzegorz Chojnowski has written a program called findMySequence that does the latter.

Sequence identification with findMySequence

Grzegorz Chojnowski





If you have an EM map you can build the main chain without knowing the sequence of the structure. In crystallography, it is harder to solve the phase problem without knowing the structure because we can't just use a homologue or a predicted model in molecular replacement. We can use SIMBAD for molecular replacement without a sequence or experimental phasing before building the main chain. Then findMySequence looks at a cloud of points in the density map where the side chain is expected to be and uses a neural network to turn these into probability estimates for each residue type. It then creates a hidden Markov model (HMM) to search sequence databases. Then the sequence can be aligned and the side chains built.

Buccaneer
Nautilus

Kevin (Kathryn) Cowtan
University of York

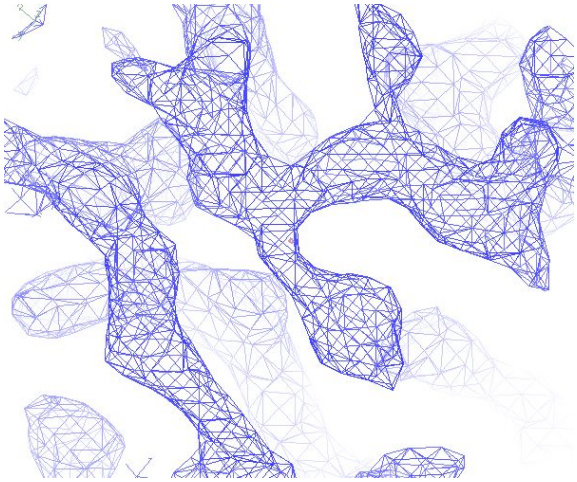
ModelCraft

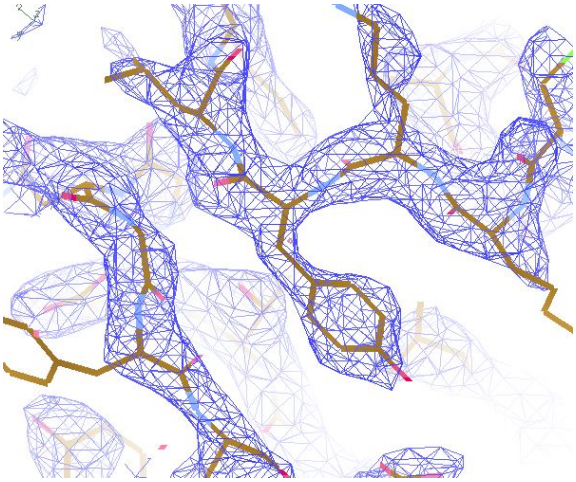
Paul Bond
University of York

The next programs we will look at are Buccaneer, Nautilus and ModelCraft. Kevin (Kathryn) 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.

Buccaneer - Task

K Cowtan



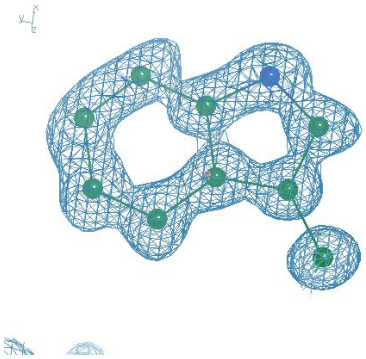


Buccaneer and *Nautilus* are provided with a density map, target sequences and optionally an existing model, and they attempt to automatically build a model to fit the map.

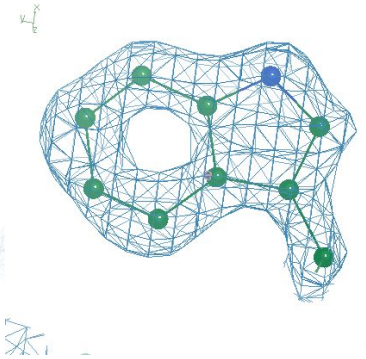
Buccaneer - Resolution

K Cowtan

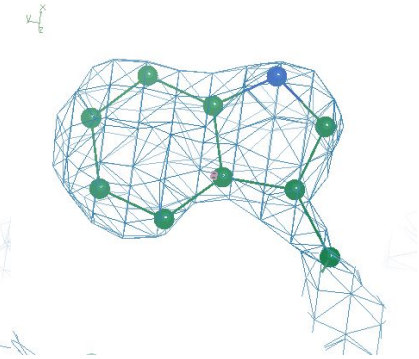
Resolution: 1Å



Resolution: 2Å



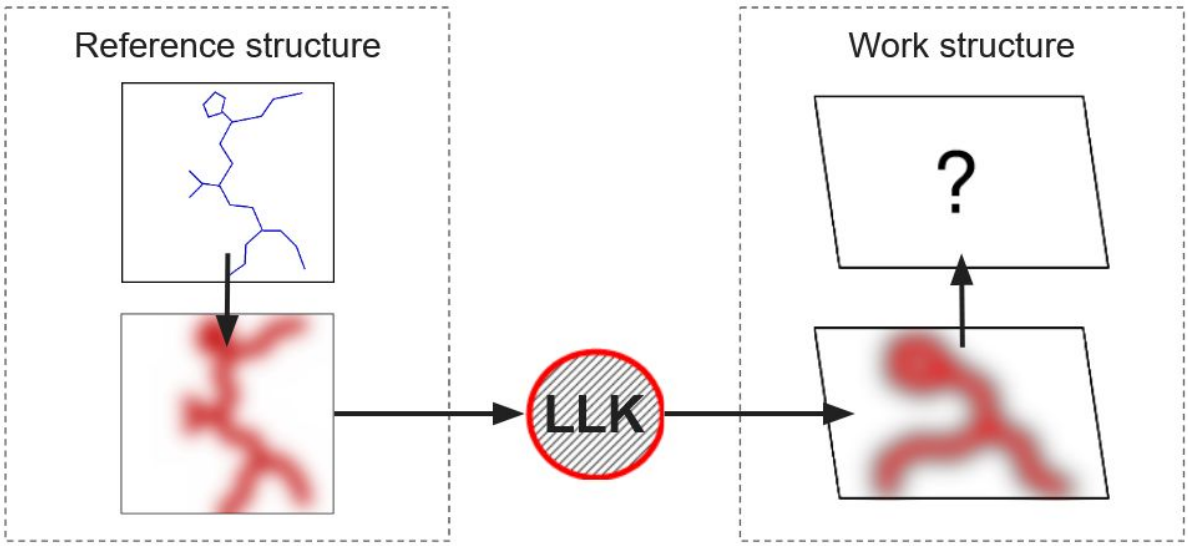
Resolution: 3Å



As we have already seen, electron density maps look very different at different resolutions. At high resolution we have individual atomic peaks and at low resolution we have blobs.

Buccaneer - LLK Target Function

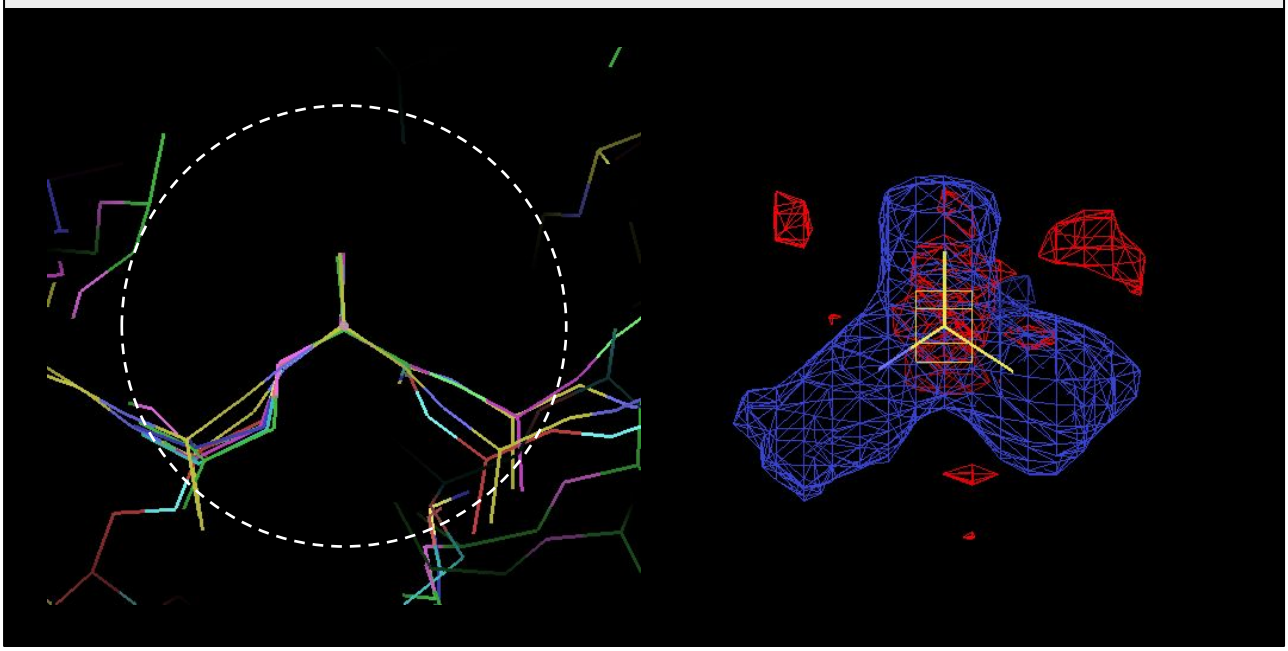
K Cowtan



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.

Buccaneer - LLK Target Function

K Cowtan



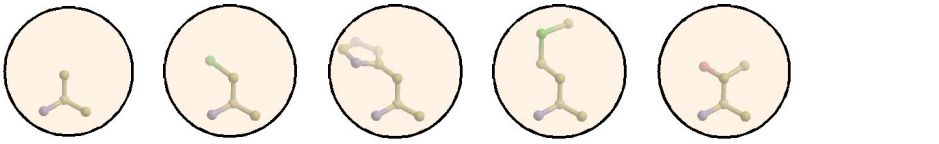
The target to search for individual residues is obtained by superimposing all the residues in the reference structure on top of each other. The mean and variance of the density is then used to construct a likelihood target within a 4 Å sphere. The blue density shows regions of conserved high density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

K Cowtan

Finding, Growing Cα environment (4.0Å sphere)

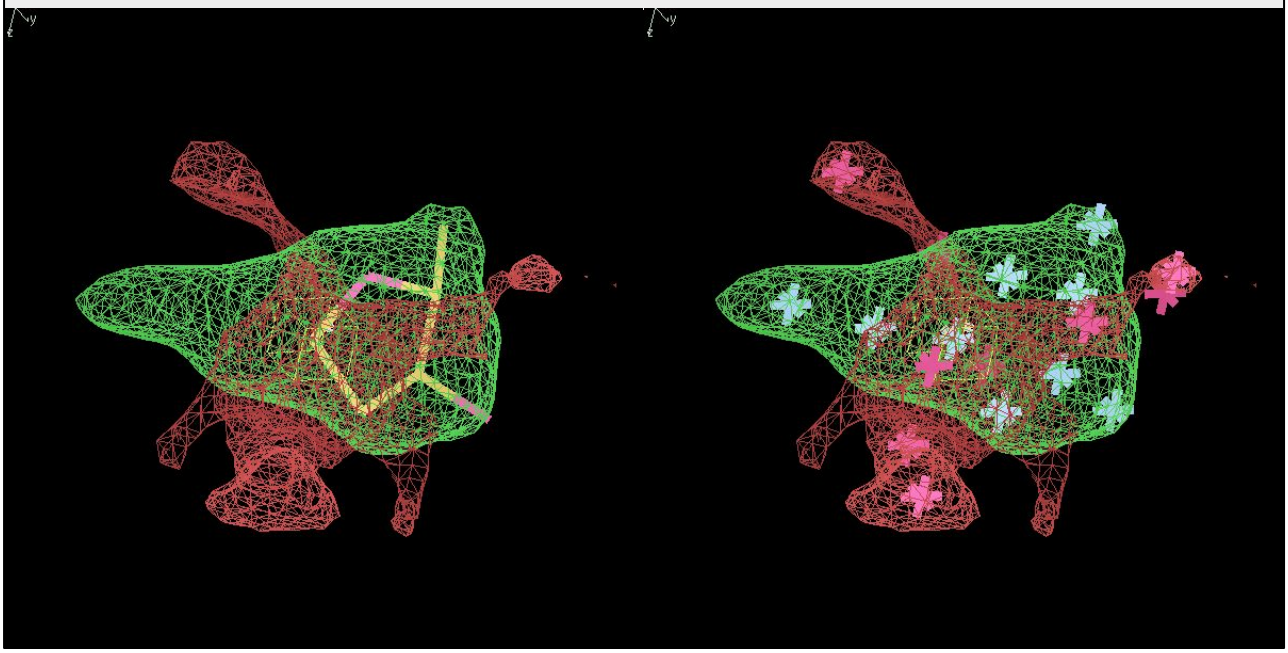
Sequencing Cβ environment (5.5Å sphere)



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

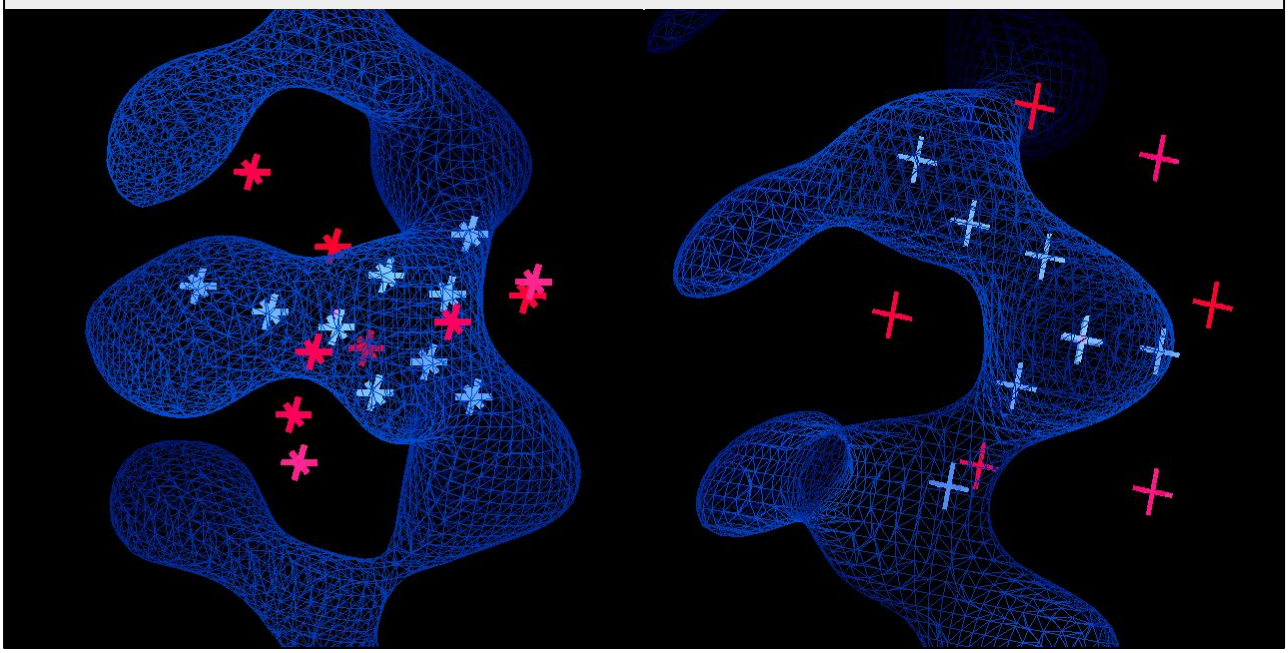
K Cowtan



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

K 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

K Cowtan

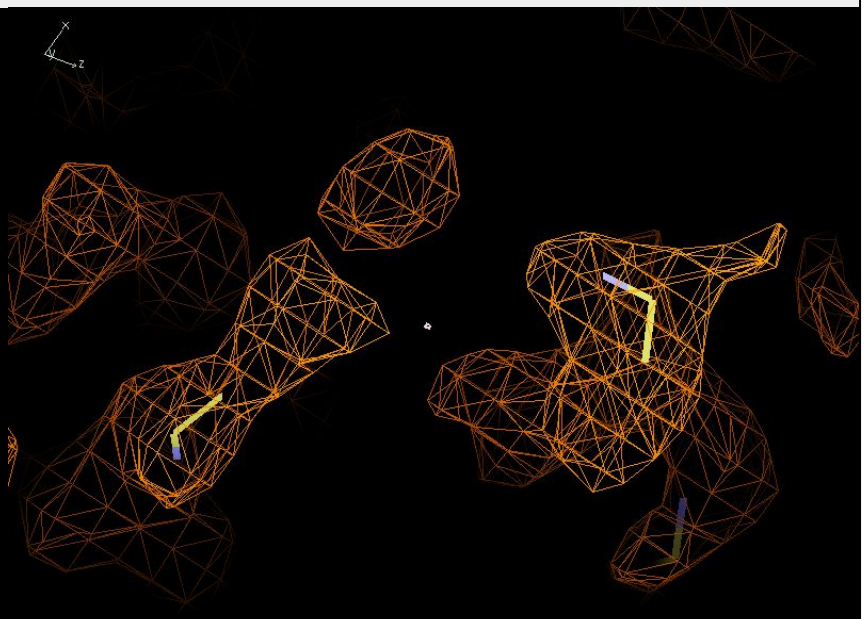
1. **Find** oriented residue positions
2. **Grow** each residue into a chain fragment
3. **Join** overlapping fragments and resolve branches
4. **Link** nearby termini
5. **Sequence** the chains
6. **Correct** insertions and deletions
7. **Filter** to remove short chains
8. **NCS** superposition to extend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

K Cowtan

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

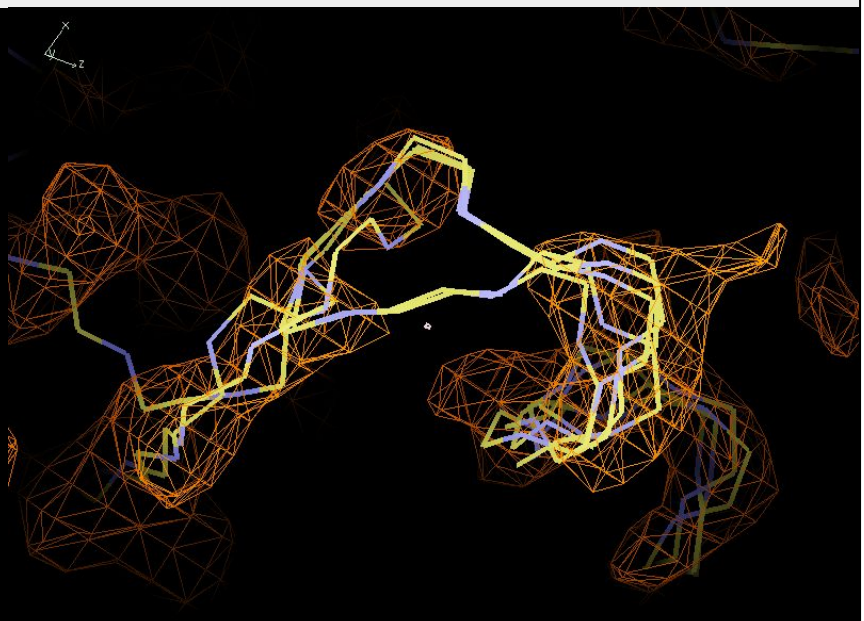


This is an example of *Buccaneer* building a difficult loop in a 2.9Å map. The first step is to find possible positions for new residues. It has placed residues at three positions in this figure (two in the foreground and one in the background) where the density has the right shape according to the C-alpha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

Buccaneer - Example

K Cowtan

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

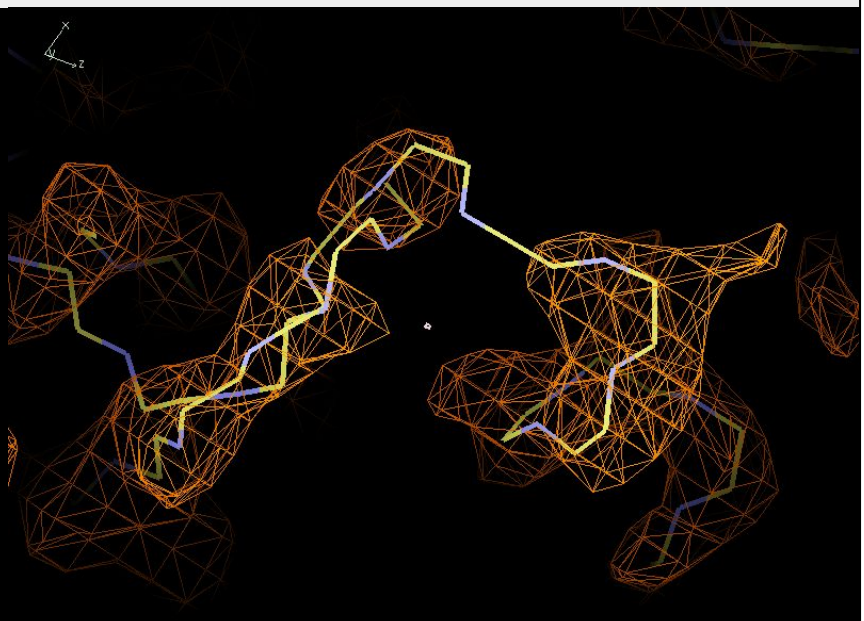


The next step is to grow each residue into a long chain fragment. New residues get added at both ends using an exhaustive search over allowed Ramachandran angles and the same C-alpha target. Once the target score drops below a threshold value the growing stops. This step gives lots of fragments, many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

K Cowtan

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

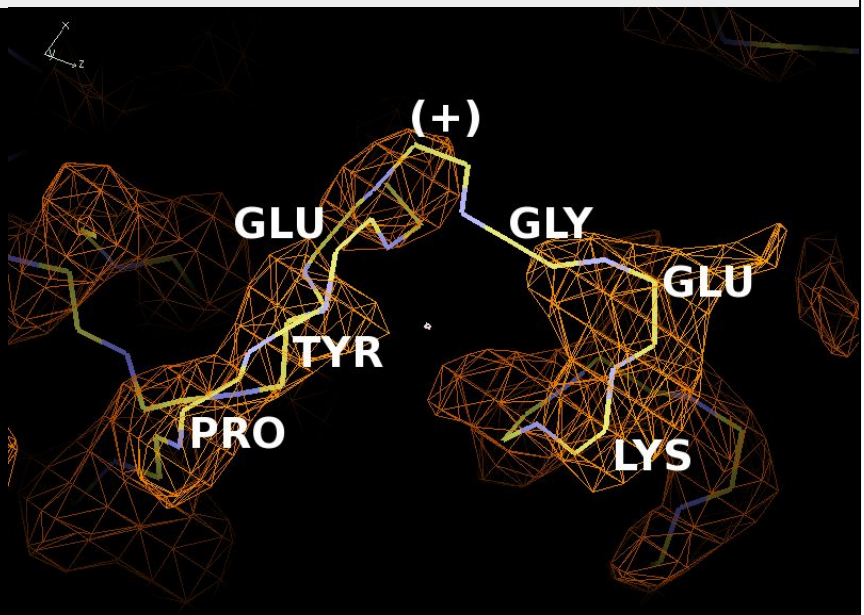


The joining step merges all the chain fragments that agree with each other into single chains. This step also has to make decisions about which routes to follow. It tries to build the longest chains (e.g. it chose the longest route over the middle loop), which helps to build helices correctly. The chain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

K Cowtan

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

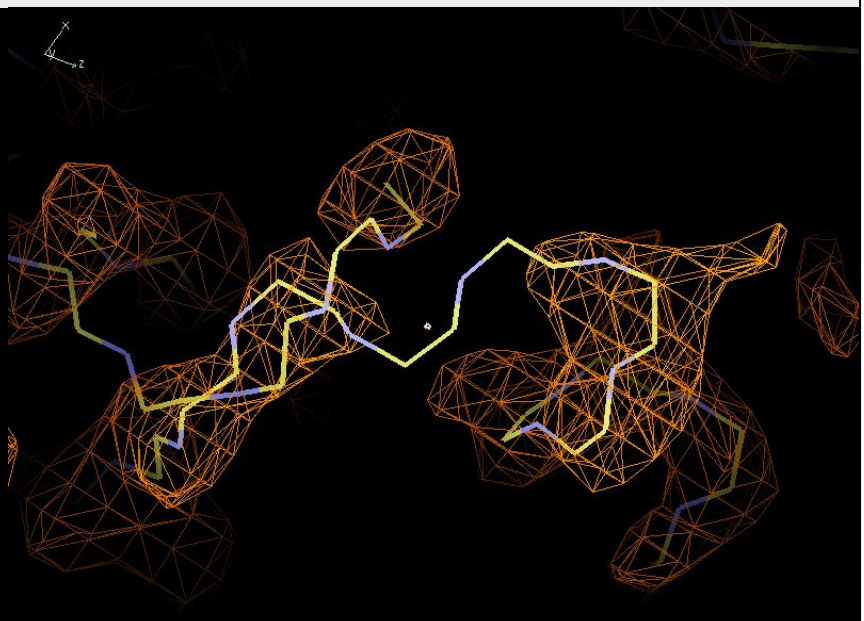


We have skipped the linking step because there are no termini that can be linked in this view. The sequencing step works by assigning each residue a probability that it is each of the 20 residue types using the C-beta targets. These probabilities are then compared to the known sequence(s) to look for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

K Cowtan

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

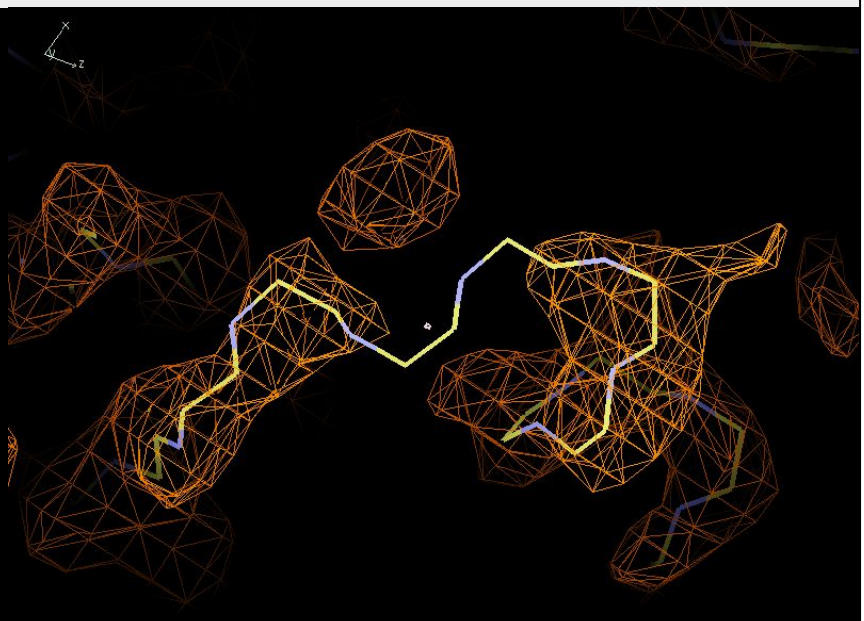


The next step is to correct any insertions (where it has built one extra residue) and deletions (where it has built one fewer residue) found during the sequencing step. The insertion that was over this loop as been corrected by removing three residues and rebuilding two.

Buccaneer - Example

K Cowtan

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

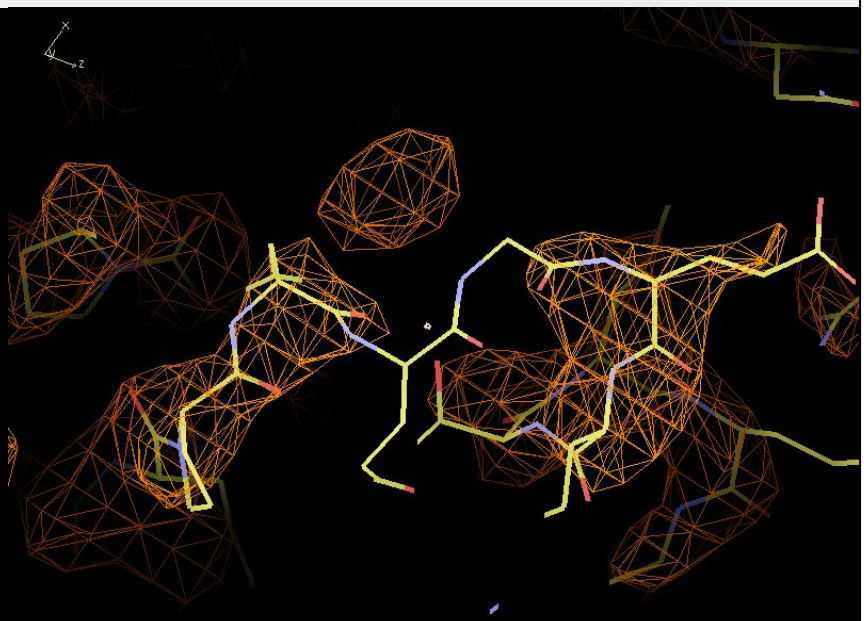


The pruning step removes residues from clashing chains to produce a single consistent model. If the pruned chains have less than 6 residues remaining they are removed too. Pruning is done at this late stage to have the most information about which chain is correct. Residues that are unsequenced or are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

K Cowtan

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

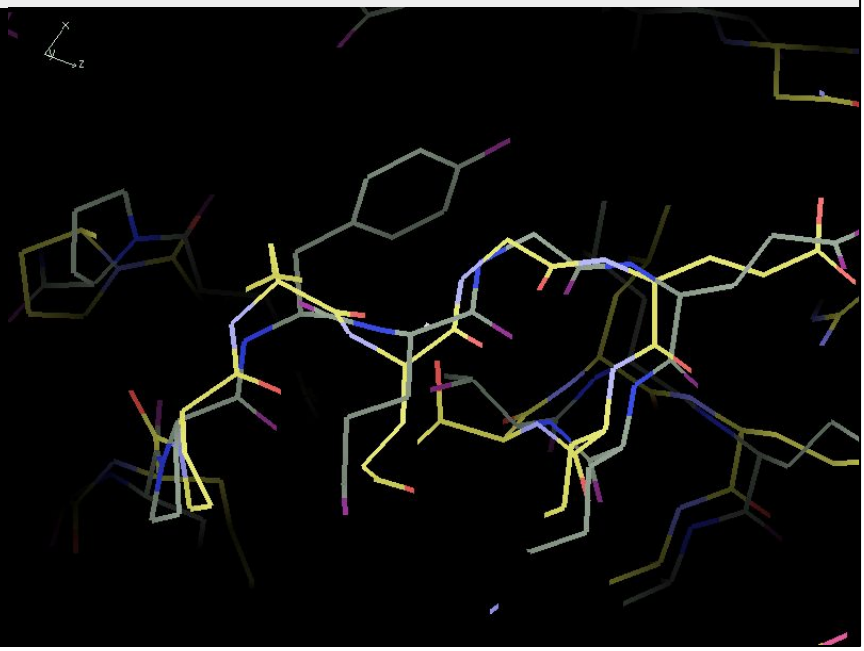


Finally, the last step is to build side chains for each residue using a rotamer library. The rotamer with the highest average density at the atomic positions is chosen. If two side chains end up clashing, *Buccaneer* will try and find a pair of rotamers that do not clash. If this fails both residues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

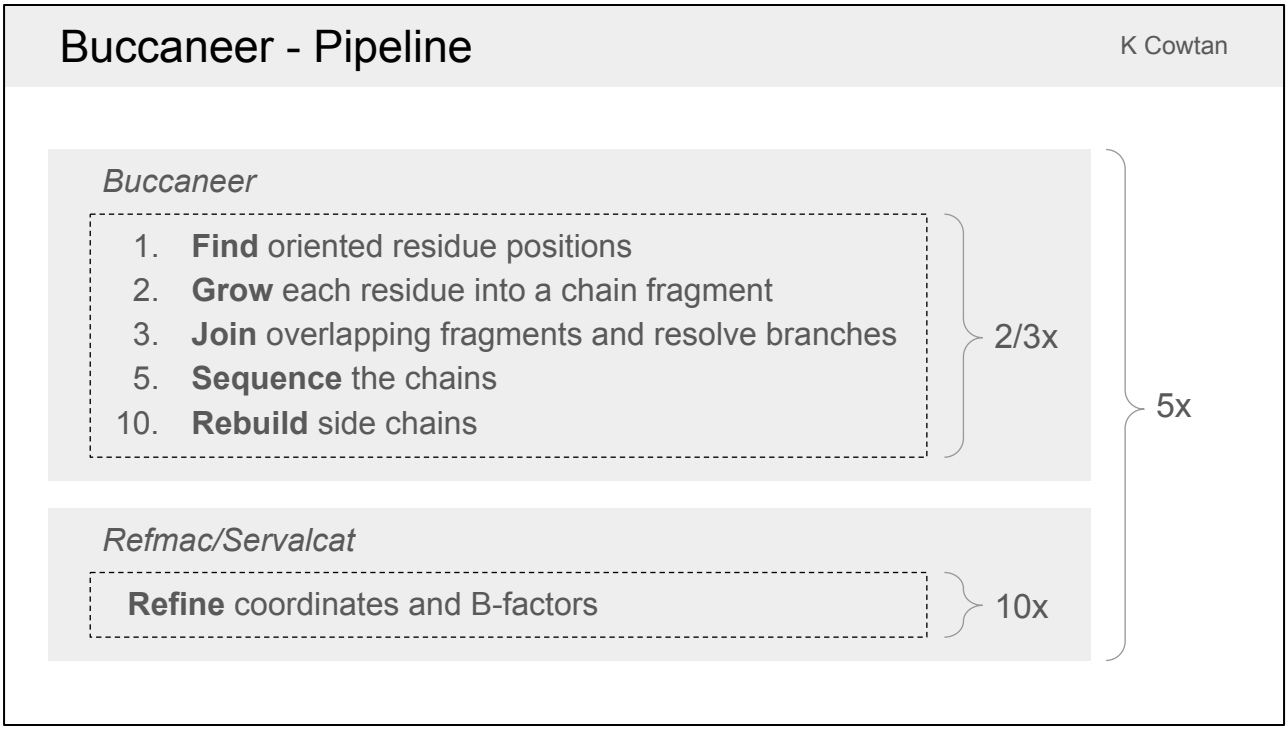
Buccaneer - Example

K 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 when they are used through one of the interfaces (i.e. *CCP4i*, *CCP4i2* or *CCP4 Cloud*) it actually runs a pipeline that combines them with *Refmac*, which refines the model coordinates and B-factors and produces an updated map for further building. The original pipelines (in *CCP4i* and *CCP4 Cloud*) run 2 or 3 cycles of *Buccaneer* or *Nautilus* followed by 10 cycles of *Refmac*, repeat that 5 times, and output the final model. There have been some additional developments to the *Buccaneer* 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 - Pipeline (X-ray)

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*

ModelCraft is a new pipeline that was created to help build more structures when the initial phases are poor, such as after molecular replacement with a very incomplete or dissimilar model. It was also designed to build both protein and nucleic acids in the same pipeline. 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.

ModelCraft - Pipeline (EM)

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

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

≤ 25x

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

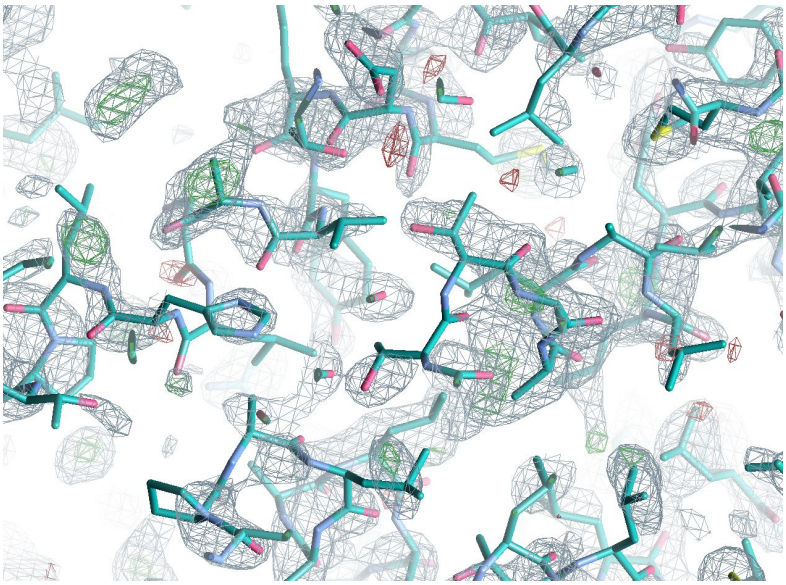
ModelCraft can also be used to build into EM maps. However, the pipeline is much simpler because the density modification steps aren't needed for EM maps and the pruning steps were only designed for crystallographic maps. There will be developments to the EM pipeline in the future, but at the moment it simply runs Buccaneer and Nautilus, both followed by Servalcat for refinement. The pipeline will stop if average FSC does not improve in 4 cycles and will output the model with the best average FSC.

ModelCraft - 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%



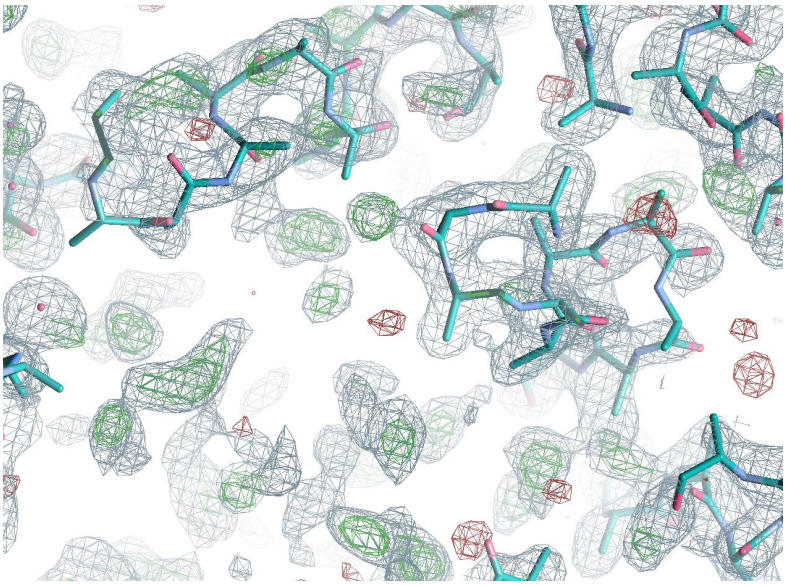
This is a difficult X-ray example where *ModelCraft* worked well. The target structure is 3GVP with 4 copies of a 435 residue protein and data to 2.25 Å resolution. The *AlphaFold* model was trimmed to remove residues with high pLDDT but it was not split into separate domains for molecular replacement. *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%.

ModelCraft - 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%



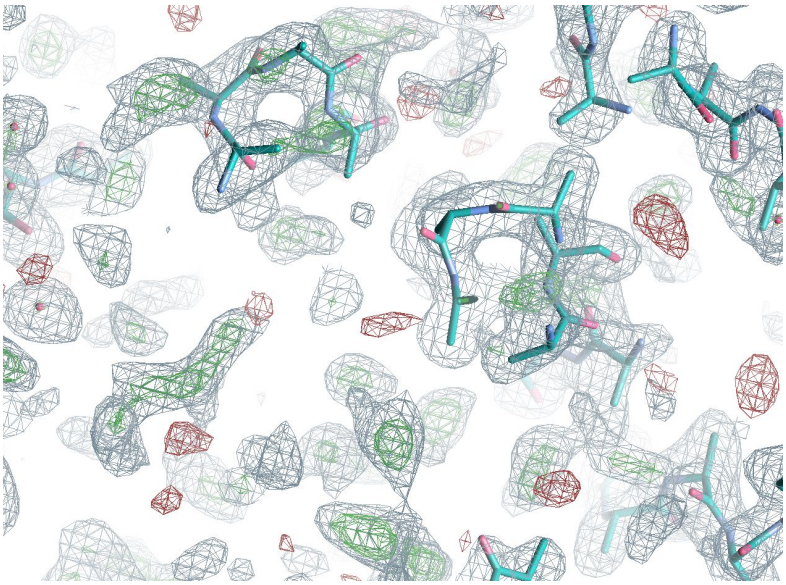
This is the same view but after 3 cycles of *ModelCraft*. As you can see, the model and map look completely different. The original structure in this region has been removed and now there are just a few small fragments built by *Buccaneer*. R-free has reduced slightly from 54% to 50%.

ModelCraft - 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%



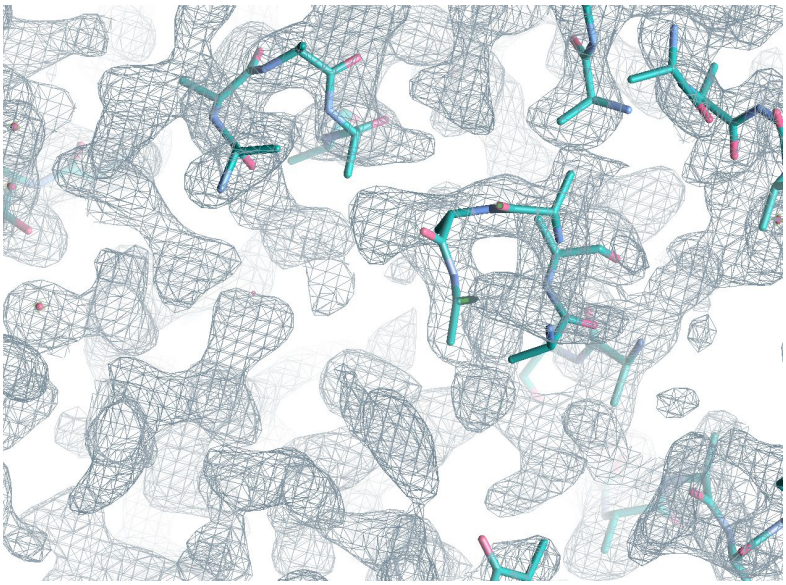
A large improvement is seen in the 4th cycle. The first step is to remove the worst residues using *Coot* and refine again. This step is only done if the resolution is better than 2.3 Å. The density for the removed residues in the top left disappeared after further refinement, suggesting that the density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

ModelCraft - 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%



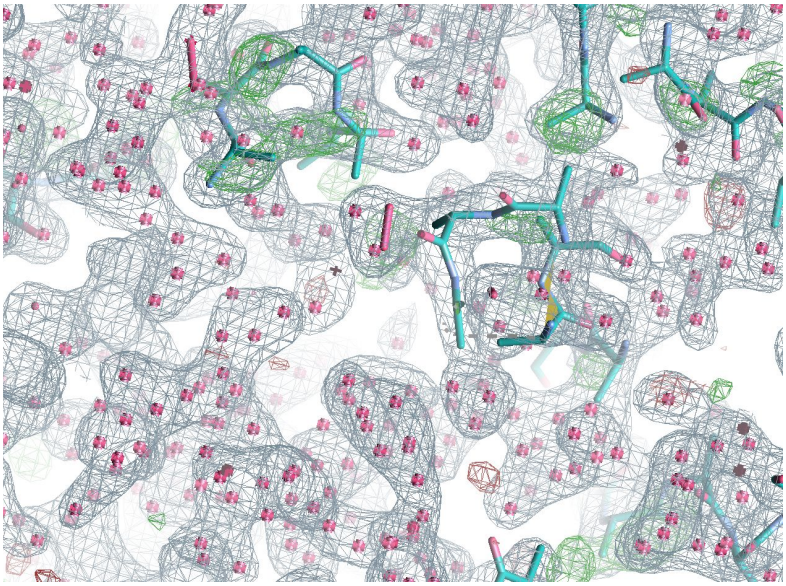
The second step is density modification using *Parrot*, which is especially powerful when there are multiple copies of a chain and the non-crystallographic symmetry (NCS) operators can be determined from the current model. NCS-related regions of the map will be averaged, weighted by how similar the density 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, but there is still a very big improvement to the density.

ModelCraft - 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%



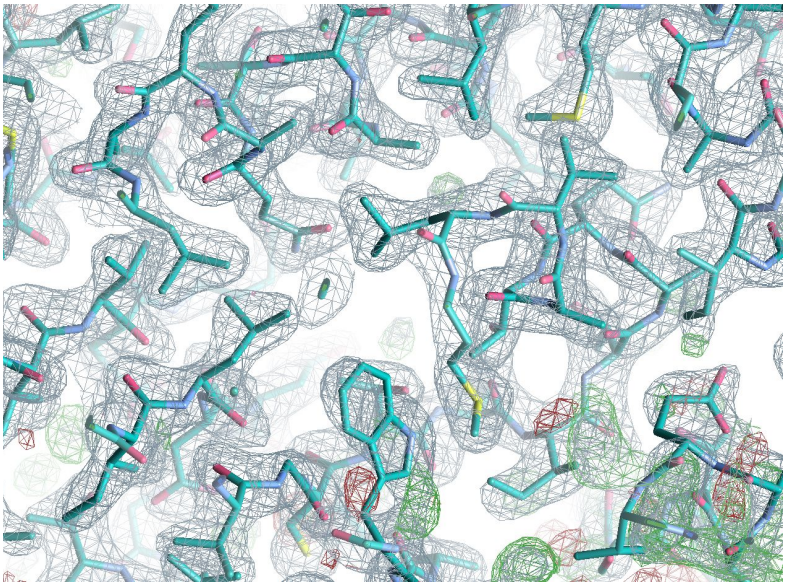
The third step adds dummy atoms into the *Parrot* map using *Coot*. The existing structure and the dummy atoms are then refined with *Refmac*. No attempt is made to interpret the dummy atoms as it is a difficult task and at lower resolutions it is unlikely that they correspond to atom positions. Instead, they only 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%.

ModelCraft - 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%



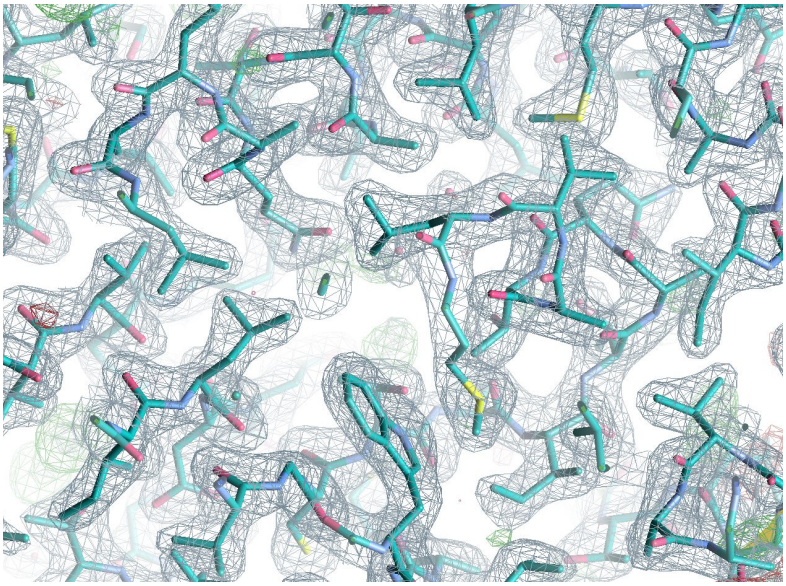
The dummy atoms are then discarded and *Buccaneer* is used to build a protein model. Because of the better phases, it has built most of this view correctly apart from some parts in the bottom right. R-free has increased slightly (from 32% to 33%) compared to the dummy atom model, but there will be much less overfitting as it is now a protein model with restraints applied.

ModelCraft - Example

Cycle 5

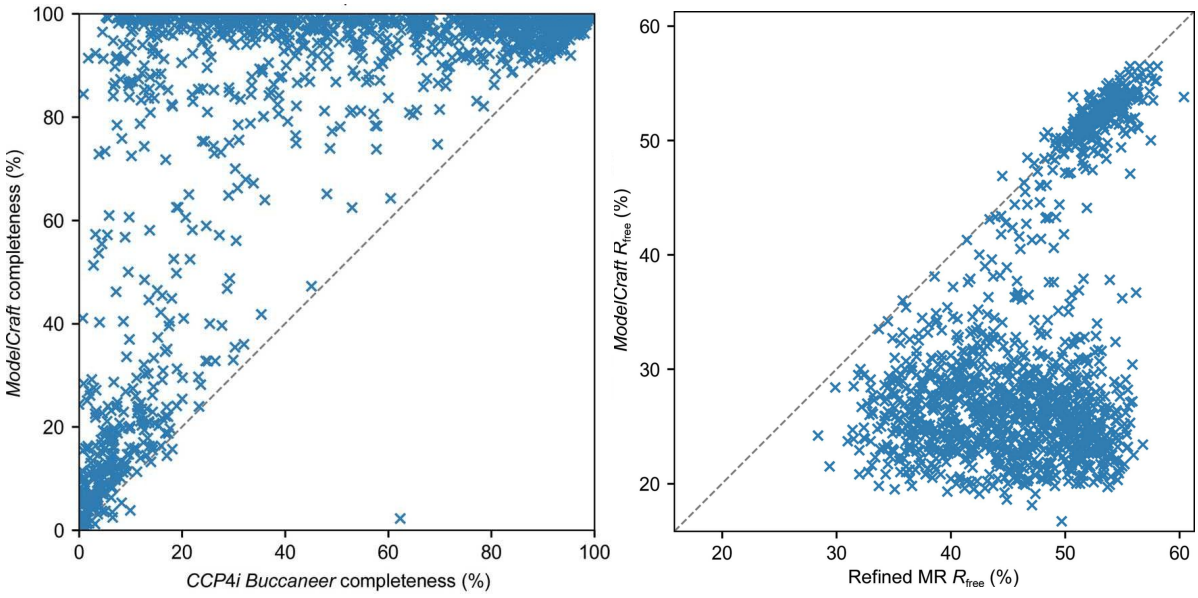
1. Prune residues
2. Density modification
3. Add dummy atoms
4. Build protein
5. Prune chains
6. Build RNA & DNA
7. Add waters

R-free: 26%



At the end of cycle 5 most of those issues have been fixed and R-free has improved to 26%. In this case, this is the best structure that ModelCraft produces. It 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 (X-ray)



These charts show the performance of ModelCraft on a large molecular replacement test set with resolutions spread evenly between 1 and 3.5 Å and examples with both good and bad phases. On the left is a comparison of the completeness of protein models after the old *Buccaneer* pipeline from *CCP4i* and *ModelCraft*. There are a lot of cases at the top of the chart where ModelCraft produces a much more complete model. There is one case in the bottom of the chart where the old pipeline produces a much more complete model, but this issue was fixed in a later version of ModelCraft. On the right is a comparison of the free-R factor before and after ModelCraft. ModelCraft is sometimes able to build a good model even if R-free after MR is as high as 55%. If the MR model has an R-free below 50% then there is a good chance that ModelCraft will be able to improve it.

ModelCraft - Tips

Running *ModelCraft*:

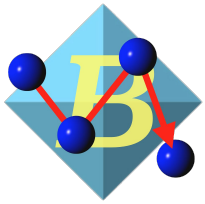
- Get the right number of copies in your asymmetric unit contents
- Remove incorrect parts of the model before starting
- Include non-incorporated heavy atoms in the input model

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.

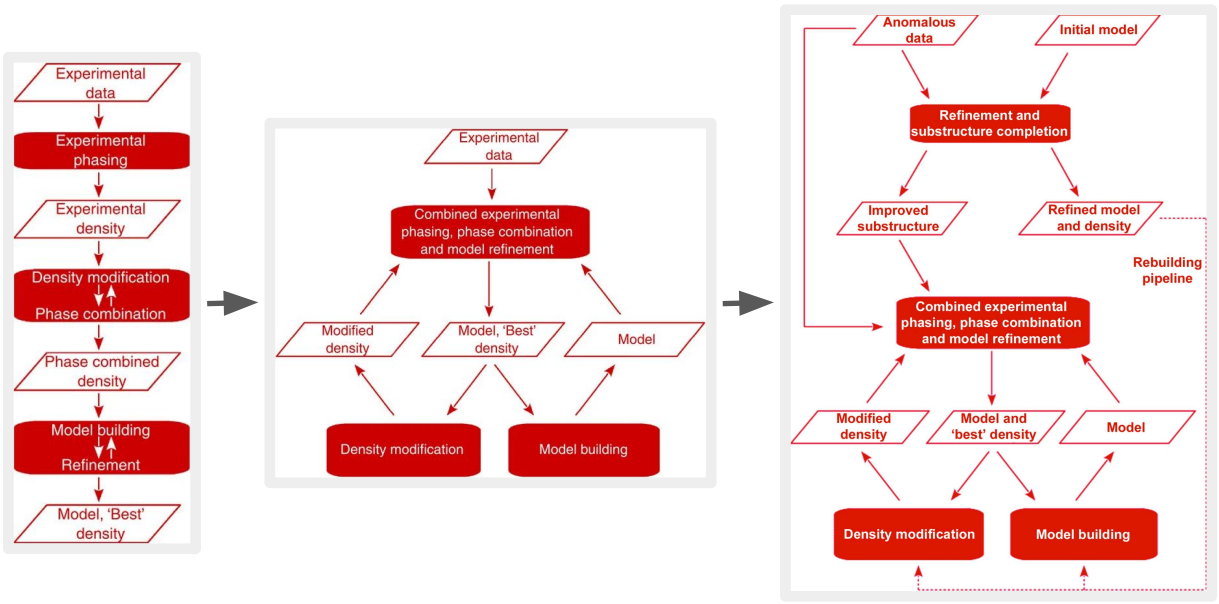
CCP4Build



- Try Buccaneer with and without Parrot density modification
- Use EDSTATS to trim the model
- Reconstruct side-chains in Coot
- Fit protein and perform real-space refinement in Coot
- Add waters
- All stages have rollback based on R-free

CCP4Build is a model building pipeline available in *CCP4 Cloud* that is similar to ModelCraft as it also uses Buccaneer and includes density modification steps and pruning. It starts by running Buccaneer both with and without Parrot density modification beforehand and chooses the best result. It then uses EDSTATS to calculate per-residue real-space Z-difference (RSZD) scores and uses those to trim the model. Then there are multiple steps in Coot to reconstruct the side chains, fit the protein, perform real-space refinement and add waters if the model and resolution are good enough. All of the steps can be turned on or off, but by default they will be attempted, followed by refinement with Refmac, and only accepted if R-free improves.

CRANK2



CRANK2 is another pipeline that uses Buccaneer for model building. It is not just a model-building pipeline, but a larger structure-solution pipeline for experimental phasing with SAD, MAD, SIRAS and MR-SAD. Previously, experimental phasing was only performed once and the experimental phase information was not reused other than through phase-combination during density modification. CRANK2 uses refinement with a multivariate probability function that simultaneously combines experimental phase information with information from density modification and model building (Skubák & Pannu, 2013). More recently, this approach was extended to include rebuilding from an initial partial model with MR-SAD (Skubák et al., 2018), which can aid completion with low-resolution data or a weak anomalous signal.

SHELXE

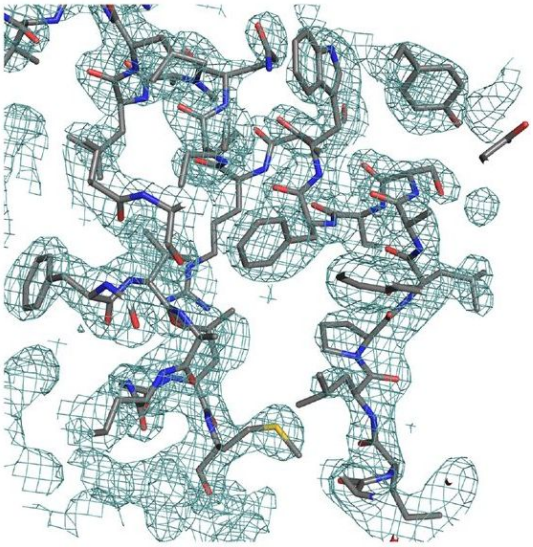
George Sheldrick
University of Göttingen

Isabel Usón
IBMB-CSIC

The next program we will look at is SHELXE.

SHELXE

- Density modification (2002)
- Main-chain tracing (2010)
- Side-chain tracing (2022)



Usón & Sheldrick (2022)

SHELXE was originally a density modification program that uses a sphere-of-influence algorithm that works better at high resolution, similar to density modification with ACORN. In 2010 a main-chain tracing algorithm was added (as building an atomic model is a good way to improve phases) and recently (in 2022) side-chain tracing algorithms were also added, so that now SHELXE is capable of building complete protein models.

SHELXE - Main-chain tracing

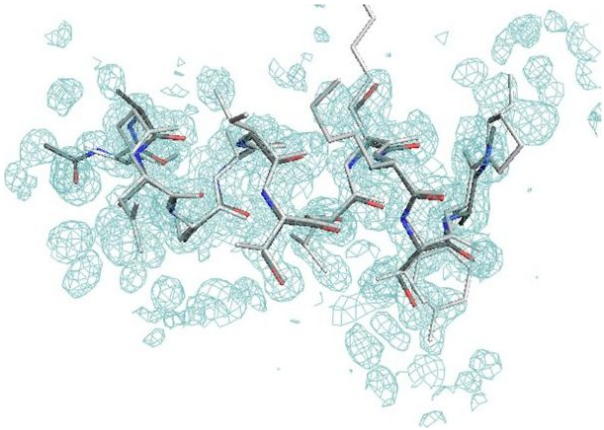
- Create a “no-go” map from symmetry elements and the current model
- Search for α-helices and common tripeptide fragments
- Extend by looking two residues ahead with common φ/ψ pairs
- Decide whether to accept each chain
- Combine overlapping chains through cutting and splicing
- Bridge gaps in the trace
- Refine per-residue B-factors

Sheldrick (2010)

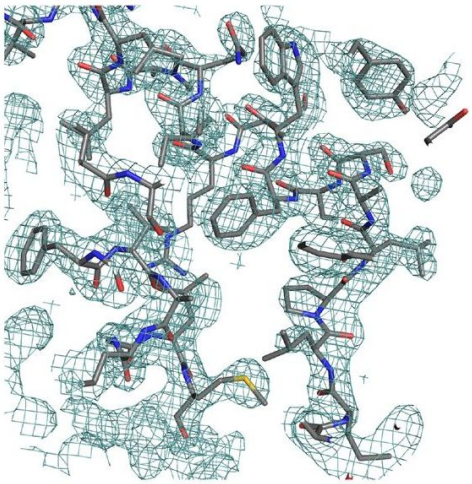
Main-chain tracing in SHELXE works by first creating a “no-go” map where the algorithm is not allowed to build into. Initially, this only includes symmetry elements and any starting model provided, but as new chains are grown and accepted this area expands. Alpha helices and common tripeptide fragments are placed into unmodelled regions of density and minimised using a weighted density score. Fragments are then extended using a two-residue look-ahead aproach that minimises common φ/ψ pairs using the same weighted density score. Extended chains are accepted based on their density score, length, Ramachandran outliers, secondary structure, and possible hydrogen bonding from N atoms. Overlapping chains are combined by cutting both chains at the point that they are closest and joining the best fragments from either side. There is also a step to bridge small gaps between close fragments. After building, per-residue B-factors are refined and a new map is produced for further building cycles.

SHELXE - Side-chain tracing

Gamma extension



Full sequencing

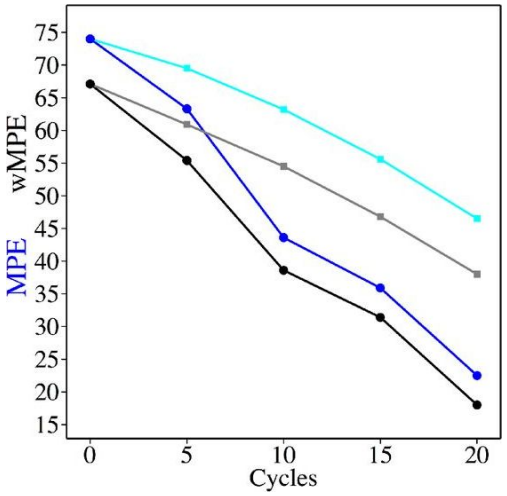


Usón & Sheldrick (2022)

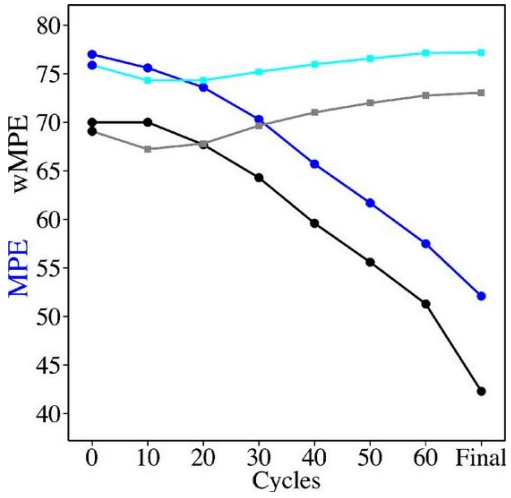
Two side-chain building algorithms were added to SHELXE: one that does not require a sequence, which works by looking at the density for gamma atoms if the χ1 angle was +60º, -60º and 180º; and also a sequencing algorithm that aligns the built main chain to the sequence and builds the full side chains.

SHELXE - Results

1ZZK DNA-binding protein at 0.95 Å



4J5M Human Myosin 5b at 2.07 Å

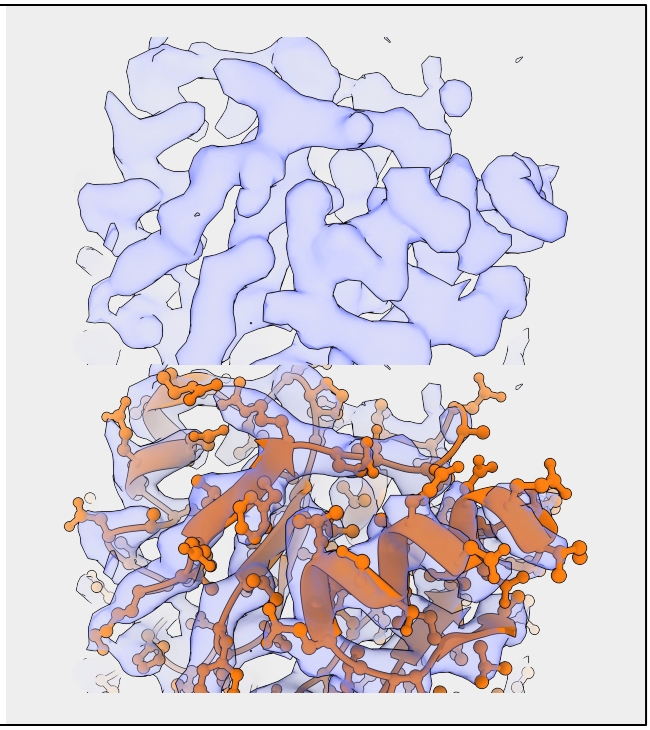


Usón & Sheldrick (2022)

These graphs show the mean phase error for two examples with (dark lines with circles) and without (light lines with squares) autotracing. The example on the left is 1ZZK, a DNA-binding protein with 0.95 Å resolution data. It was phased using only four residues. In this case, low phase errors can be achieved with and without autotracing (after more cycles). The example on the right is 4J5M, human myosin at 2.07 Å resolution. Here SHELXE can only improve the phases if autobuilding is used. Changes to the main-chain building algorithms allow it to work at lower resolutions: typically better than 2.5 Å for molecular replacement and better than 3 Å for experimental phasing (Usón and Sheldrick, 2018). At higher resolutions it is very useful for improving poor phases.

ModelAngelo

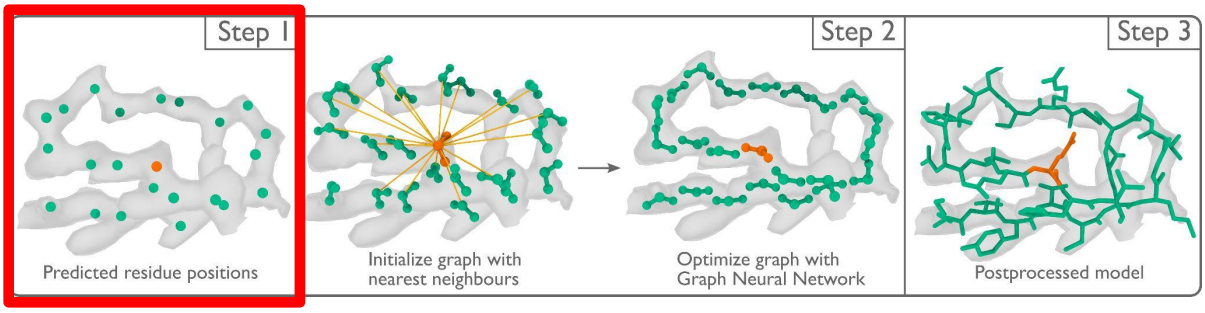
Kiarash Jamali
MRC LMB



The final model-building program we will look at is ModelAngelo. It uses neural networks to build protein and nucleotide models into cryo-EM maps. Thank you to Kiarash Jamali who provided these slides in the presentation.

ModelAngelo - Step 1

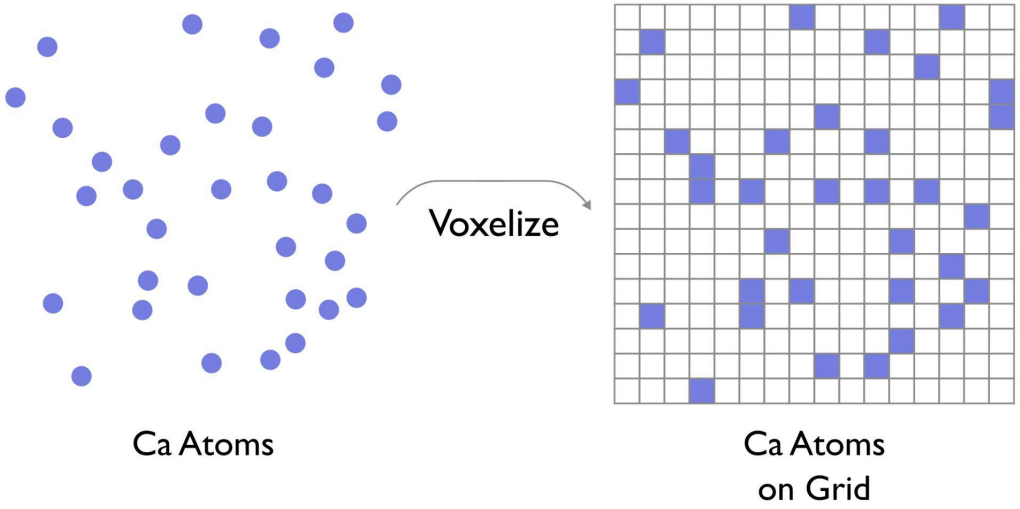
Kiarash Jamali



This is a rough overview of the process. The first step is to predict the positions of residues in the map using a convolutional neural network.

ModelAngelo - Residue segmentation

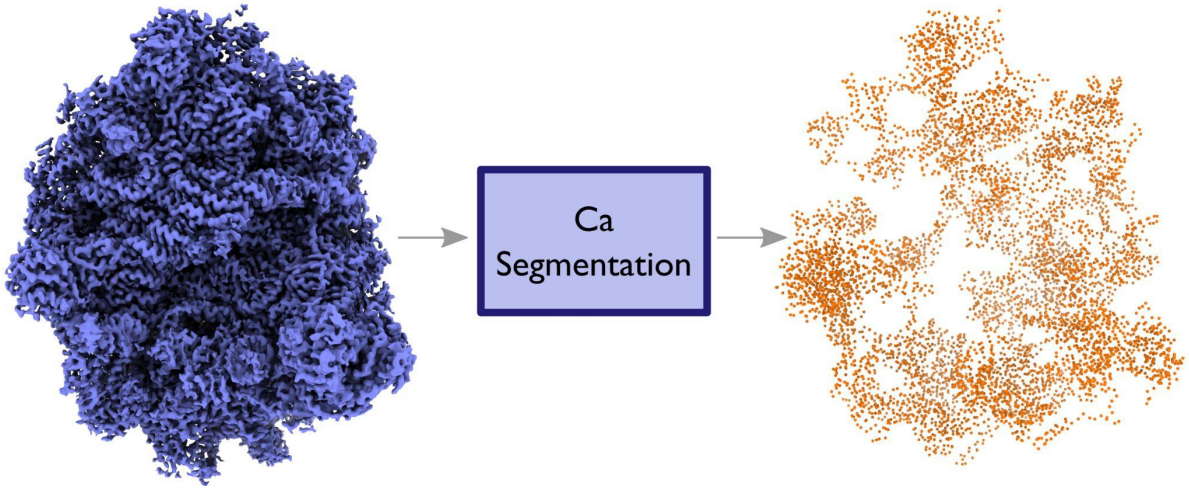
Kiarash Jamali



First we will look at the network that predicts residue positions. It is trained by taking deposited cryo-EM maps and sampling them with a voxel size of 1.5 Å. Each voxel is marked as having or not having a C-alpha atom inside it (for proteins). This can then be treated as a common classification/segmentation problem by the neural network, which receives the density and predicts whether each voxel has a C-alpha. One difficulty with this is that it is a very imbalanced problem. Most voxels do not have a C-alpha and the loss function needs to take this into account.

ModelAngelo - Residue segmentation

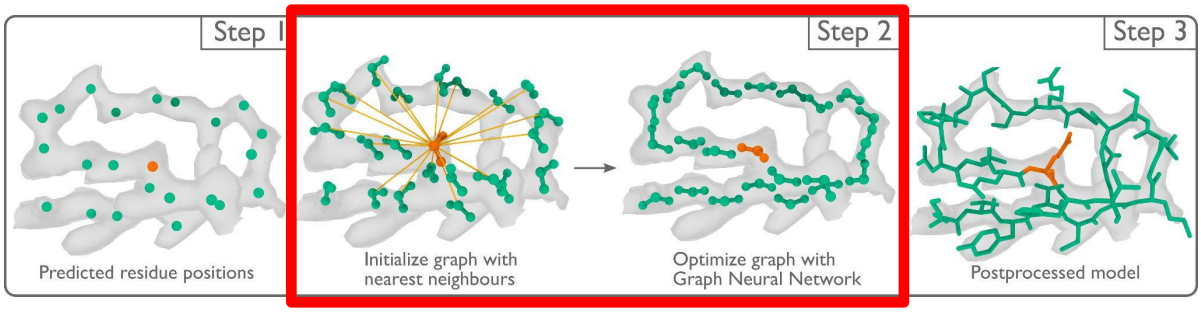
Kiarash Jamali



This is a real example showing a map and the predicted C-alpha positions for that map. There is also a network to predict phosphorus atoms in nucleotides.

ModelAngelo - Step 2

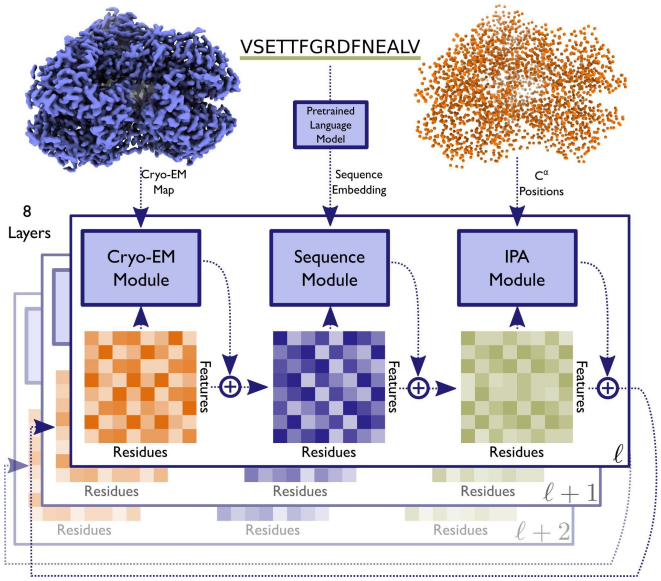
Kiarash Jamali



The second step places a residue at each of the predicted positions in a random orientation and initialises a graph connecting each residue to its 20 nearest neighbours. This graph is optimised with the main graph neural network that combines information from multiple sources.

ModelAngelo - Graph neural network

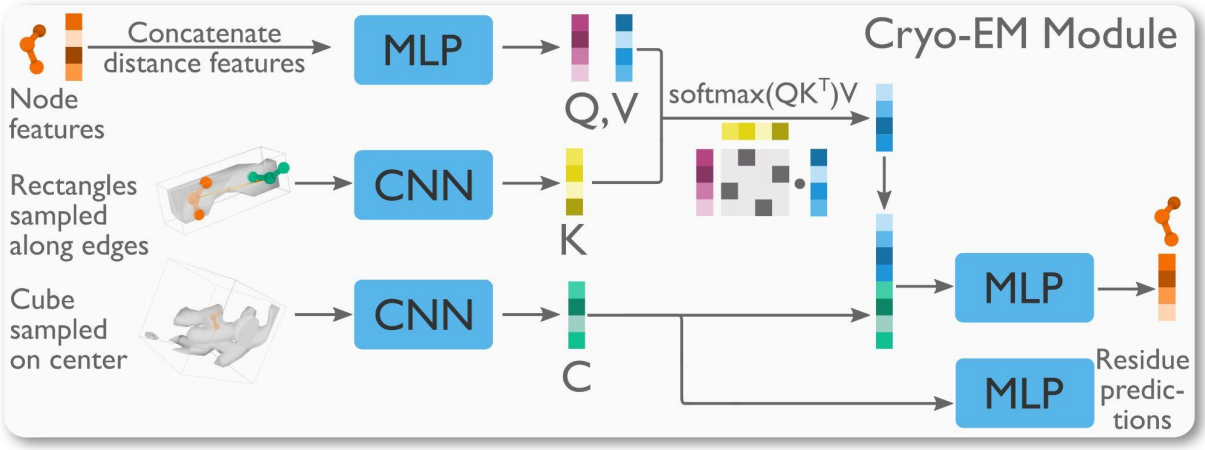
Kiarash Jamali



This is an overview of the graph neural network (GNN). ModelAngelo stores a feature vector for each residue that is an abstract representation of what is known about that residue. The feature vectors get updated in a sequential manner: first with the cryo-EM module, which gets information from the map, then the sequence module, which gets information from the expected sequence, and finally from the IPA module which looks at the neighbouring residues. This process is repeated 8 times.

ModelAngelo - Cryo-EM module

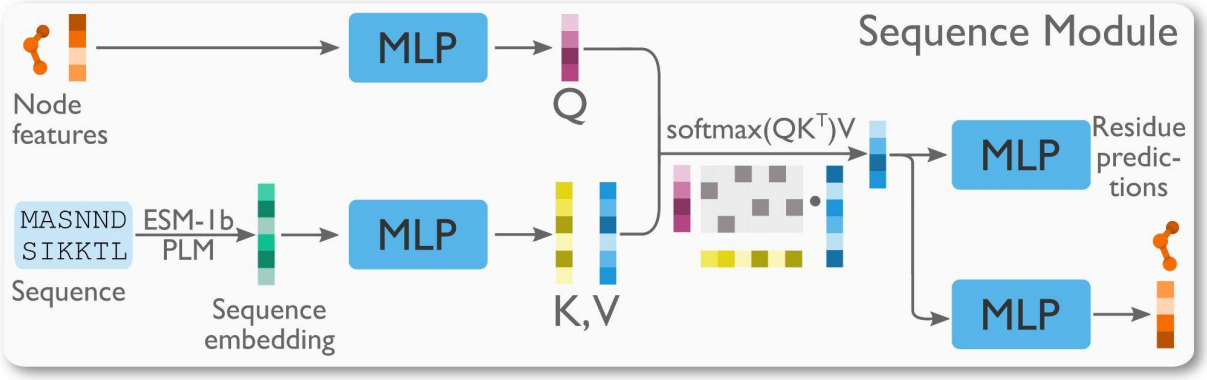
Kiarash Jamali



This is the cryo-EM module. The current residue information (node features) are used as input and updated in the output. There are two different types of input from the map. One is a rectangle of density between a residue and its neighbours, which is useful to determine whether the residues are connected or not, and the other is a cube of electron density at the residue centre (with the rotation dependent on the orientation of the residue), which is useful to to determine what the side-chain type is and what rotations and translations are required.

ModelAngelo - Sequence module

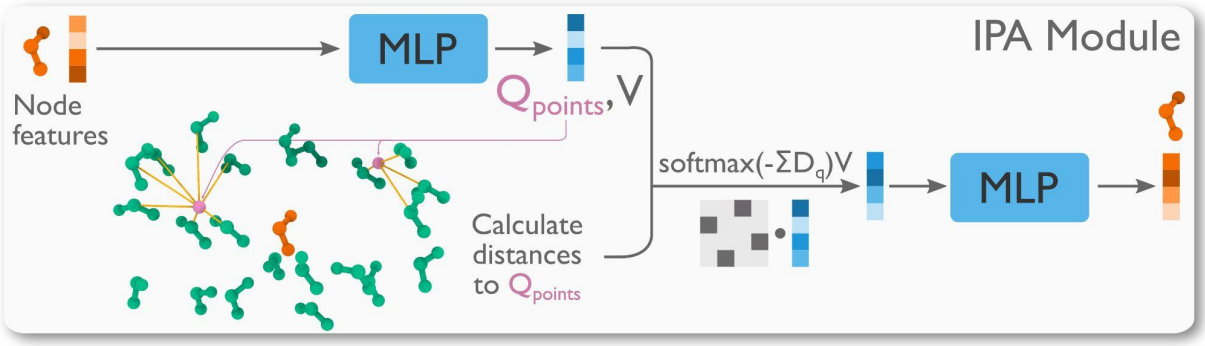
Kiarash Jamali



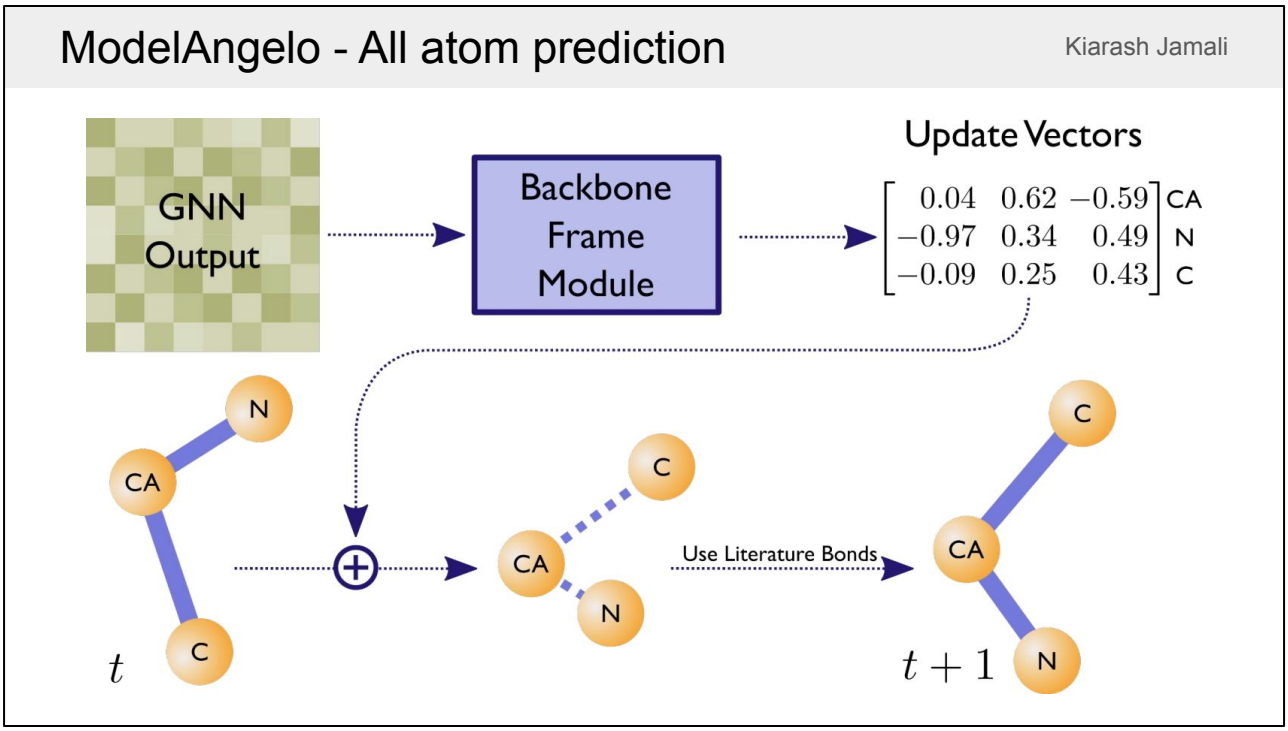
This is the sequence module, which uses the protein language model (PLM) from Meta's Evolutionary Scale Modeling (ESM). ESMFold is an alternative to AlphaFold that, instead of creating a multiple sequence alignment using sequence databases, uses its PLM, which is a large-language model (LLM) trained to extract information from protein sequences.

ModelAngelo - IPA module

Kiarash Jamali



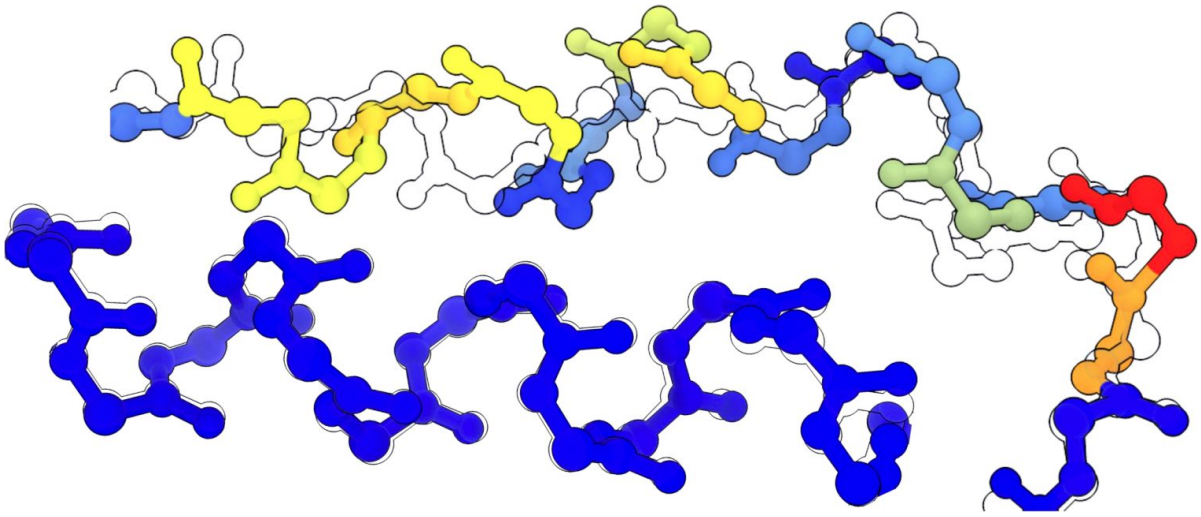
This is the spatial invariant point attention (IPA) module. It chooses points in space and looks at the distances between those points and the nearest residues. This helps it to learn about common geometric features, such as realising that a residue is part of an alpha helix.



The abstract per-residue information obtained from the graph neural network (GNN) is put into different modules that each predict something specific about the model. For example, the backbone frame module predicts where each atom in a residue should be moved. Those movements are made and then the geometry is restored using literature values. This is how the model (where residues start with random orientations and inaccurate positions) slowly gets refined to the correct place.

ModelAngelo - Confidence measure

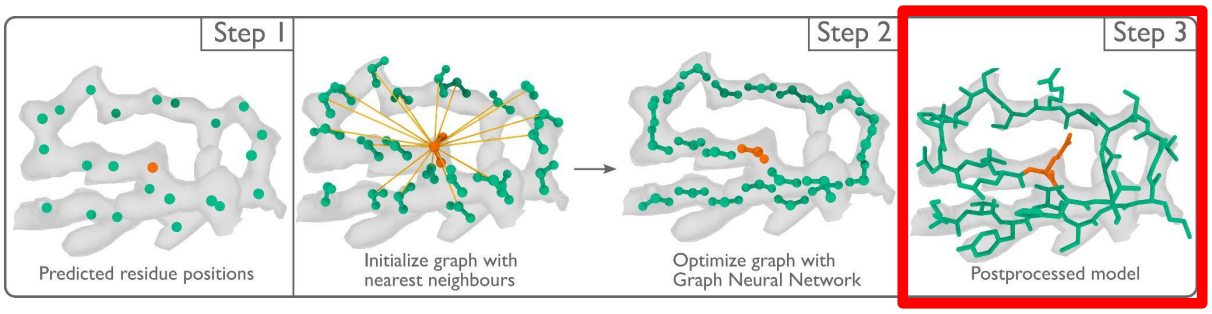
Kiarash Jamali



Another prediction ModelAngelo makes is a per-residue confidence measure, i.e. how much it thinks the residue will differ from the deposited model.

ModelAngelo - Step 3

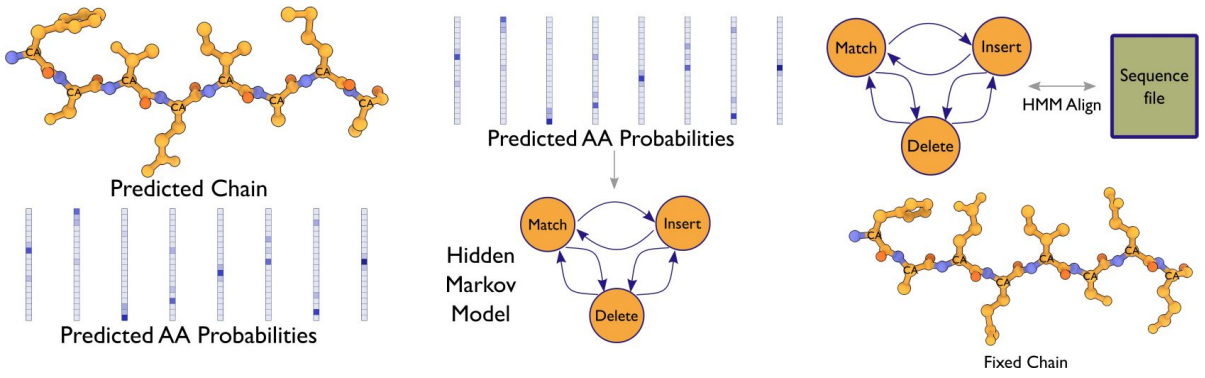
Kiarash Jamali



The final step is to model that is produced by the neural network undergoes a postprocessing step to assign the sequence to the model and build side chains.

ModelAngelo - HMM post-processing

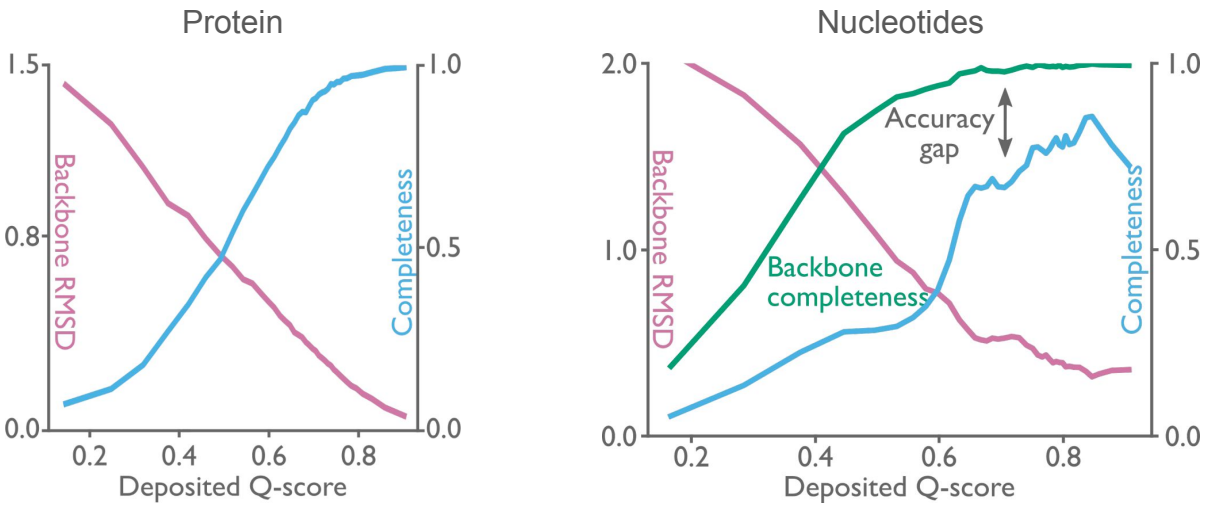
Kiarash Jamali



ModelAngelo produces amino-acid type probabilities for each residue. It is possible to just choose the most probable type for each residue, but some predictions may be incorrect. Instead, ModelAngelo converts the probabilities to a hidden Markov model (HMM) that takes into account the per-residue confidence measure. The HMM is then used to align the built model with the given sequence. If you do not know your sequence then ModelAngelo can use the same HMM to search a sequence database to identify the protein.

ModelAngelo - Results

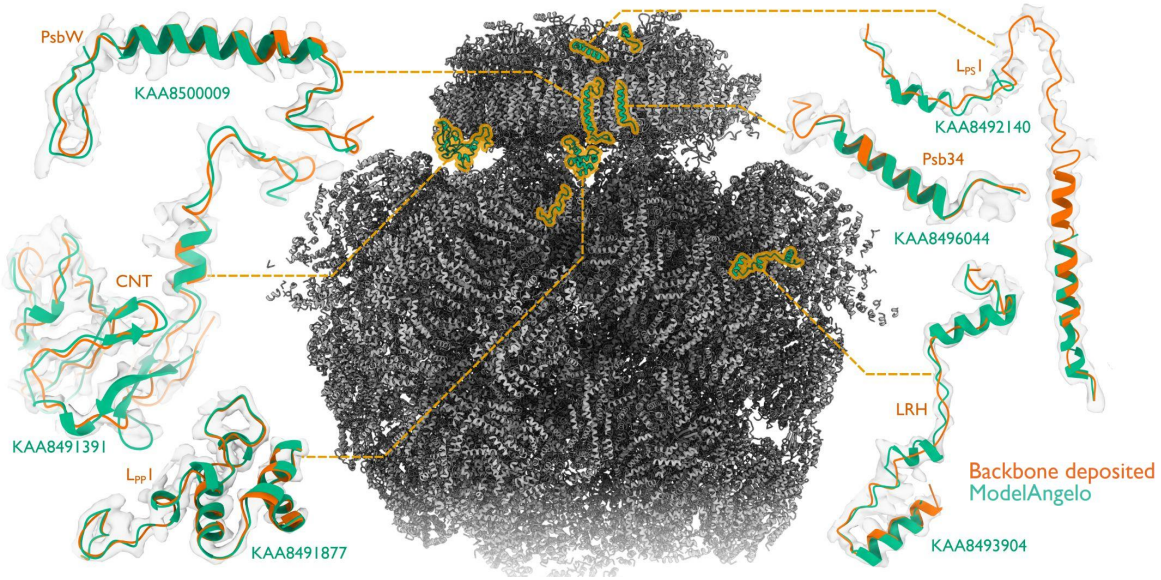
Kiarash Jamali



These charts show the completeness and backbone RMSD of models from ModelAngelo against the deposited Q-score (a measure of how well atoms can be resolved in the map). For proteins, ModelAngelo was trained on structures deposited up to April 2022 and tested on ~170 structures deposited after this with map resolutions better than 4 Å which were not similar (more than 10% sequence identity) to models it was trained with. ModelAngelo is usually able to build complete protein models provided the map is of sufficient quality. For nucleic acids, there were fewer test structures available. ModelAngelo is able to build the sugar-phosphate backbone very well but is less able to sequence the chains.

ModelAngelo - Phycobilisome

Kiarash Jamali



As of early 2023, this structure (7Y5E) is the biggest single-particle cryo-EM structure in the PDB at larger than 16 MDa. It contains 80 unique protein chains, but 6 of them were not identified by the authors and were traced by hand. ModelAngelo was able to identify all 6 proteins and build them with the correct sequences.

ModelAngelo - Conclusions

Kiarash Jamali

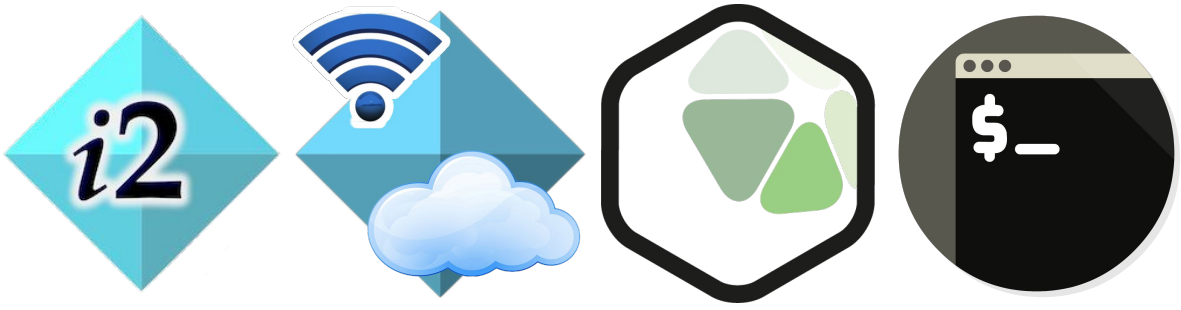
- Human parity in building proteins when sequence is known and resolution is high (up to 3.5 Å)
- Good nucleotide backbone quality, sequence remains a challenge
- Better than humans at identification of novel proteins from map alone when resolution is high (up to 3.5 Å)

Try it out!

https://github.com/3dem/model-angelo

- Open source software, MIT licence
- Version 1.0 with nucleotides and protein identification now released

Interfaces



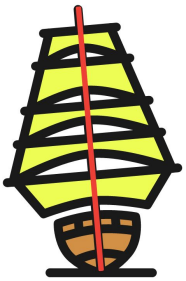
CCP4i2 CCP4 Cloud CCP-EM Terminal

For X-ray data you can run programs from either CCP4i2 or CCP4 Cloud. Cryo-EM data programs can be run in CCP-EM. Most programs can also be run from a terminal.

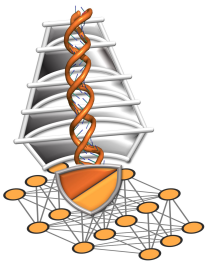
Outside CCP4 and CCP-EM

AutoBuild

DeepTracer



MAINMAST



CryoRead

These are some of the other model-building programs that are available outside of CCP4 and CCP-EM. *PHENIX AutoBuild* is a very good model building pipeline available in the PHENIX suite. It can take a long time to run but is good at model completion. It works with both X-ray and cryo-EM data. The other programs are only for cryo-EM. DeepTracer and MAINMAST build proteins and CryoRead is a new program for building nucleic acids.

Acknowledgements

Grzegorz Chojnowski

ARP/wARP

Victor Lamzin

Anastassis (Tassos) Perrakis

MRC Laboratory, Cambridge :

Garib Murshudov's group

CCP4 core group: Eugene Krissinel,

Andrey Lebedev, Ronan Keegan

EMBL HH, HD, EMBL EBI:

Thomas Schneider, Gerard Kleywegt, Sameer Velankar

Philipp Heuser, Daria Beshnova, Ciarán Carolan, Serge Cohen, Zbyszek Dauter, Helene Doerksen, Guillaume Evrard, Francisco Fernandez, Johan Hattne, Saul Hazledine, Marouane Jelloul, Krista Joosten, Matheos Kakaris, Olga Kirillova, Gerrit Langer, Wijnand Mooij, Richard Morris, Venkat Parthasarathy, Joana Pereira, Tilo Strutz, Ioan Vancea, Tim Wiegel, Keith Wilson, Peter Zwart, Umut Oezugurel, Sravya Kantamneni, Egor Sobolev, **Grzegorz Chojnowski**

findMySequence

CCP4 Core Team
Ronan Keegan
Charles Ballard
Eugene Krissinel
Kyle Stevenson

University of Liverpool
Daniel Rigden
Adam Simpkin

EMBL
Wolfram Seifert-Davila
Matthias Wilmanns
Isabel Bento

São Carlos Institute of Physics
Diego A. Leonardo

Laboratorio de Biolgia Molecular, Peru
Dan E. Vivas-Ruiz

checkMySequence

CCP4 Core Team
Ronan Keegan

University of Liverpool
Daniel Rigden
Filomeno Sanchez Rodriguez

https://gitlab.com/gchojnowski/

GitLab

YOU, CCPEM and CCP4 workshop students!

EMBL

Acknowledgements

Kiarash Jamali

Sjors Scheres
Dari Kimanius
Johannes Schwab
Joe Greener
Sofia Lövestam
Chao Qi
Yang Yang

Collaborators
Lukas Käll
Alan Brown
Rui Zhang

Scientific Computing
Jake Grimmett
Toby Darling
Ivan Clayson

Second and University Supervisors
Garib Murshodov
Carola Bibiane Schönlieb

First users
Rafael Fernandez-Leiro
Zephyr Ford
Katerina Naydenova

MRC Laboratory of Molecular Biology

Acknowledgements

Paul Bond

Supervisors
Kevin (Kathryn) Cowtan
Keith Wilson

Cowtan Group
Jordan Dialpuri
Soon Wen Hoh
Stuart McNicholas
Mateusz Olek
Filomeno Sanchez Rodriguez

Collaborators
Charles Ballard
Eleanor Dodson
Maria Fando
Ronan Keegan
Oleg Kovalevskiy
Eugene Krissinel
Andrey Lebedev
Rob Nicholls
Martin Noble
Marcin Wojdyr
Keitaro Yamashita

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