Single-particle reconstruction

In this chapter we'll consider some of the basic concepts behind cryo-EM single-particle structure determination.

Image-matching

Last time we saw how, if we know the projection directions, we can use the Fourier slice theorem to make a 3D reconstruction. The trick will be to determine the projection direction corresponding to each of many single-particle images. Another thing we will want to do is to group together similar images and average them, so that we can see what our particles look like. For both of these steps we wish to find the best match between a given particle image and a reference image of some sort--either a class-average image or a projection of a 3D model. There are two popular ways to compare images.

1. Cross-Correlation. The comparison between two images is made just by multiplying each pair of corresponding pixel values, and then summing up the products to yield the value c_i. Suppose $\rho(x,y)$ is the pixel at (x,y) of particle image and $r_i(x,y)$ is the corresponding pixel of the i^{th} member of a set of reference images. Then we compute the correlation c_i for each of the references

$$c_i = \sum_{x,y} \rho(x,y) r_i(x,y), \qquad (1)$$

If ρ and r_i match well, the value of c_i will be a large positive value; if the two images are uncorrelated, the value of c_i will be about zero.

A problem with the correlation as defined here is that a maximum value could result from a poorly-matching reference that however has very large pixel values. To avoid this problem one can instead use the *correlation coefficient*, $\hat{c}_i$ which is obtained by normalizing the value of c_i by the magnitudes of ρ and r_i,

$$\hat{c}_i = \frac{\sum_{x,y} \rho(x,y) r_i(x,y)}{\sqrt{\sum_{x,y} \rho^2(x,y)} \sqrt{\sum_{x,y} r_i^2(x,y)}}$$

2. Squared difference. We can just subtract the two images and compute the sum of the squared residuals,

$$s_i^2 = \sum_{x,y} [\rho(x,y) - r_i(x,y)]^2 \qquad (2)$$

Which we can write compactly as

$$s_i^2 = \|\rho - r_i\|^2$$

If we expand the square in eqn. (2) we see that

$$s_i^2 = \|x\|^2 + \|r_i\|^2 - 2\sum_{x,y} \rho(x,y) r_i(x,y) \qquad (3)$$

where for image ρ the squared magnitude $\|\rho\|^2$ is defined to be the sum of the squares of all the pixel values. If the squared magnitude of all the r_i are the same, then minimizing s_i^2 is just the same as maximizing the correlation c_i (eqn. 1).

Cross-correlation and particle picking

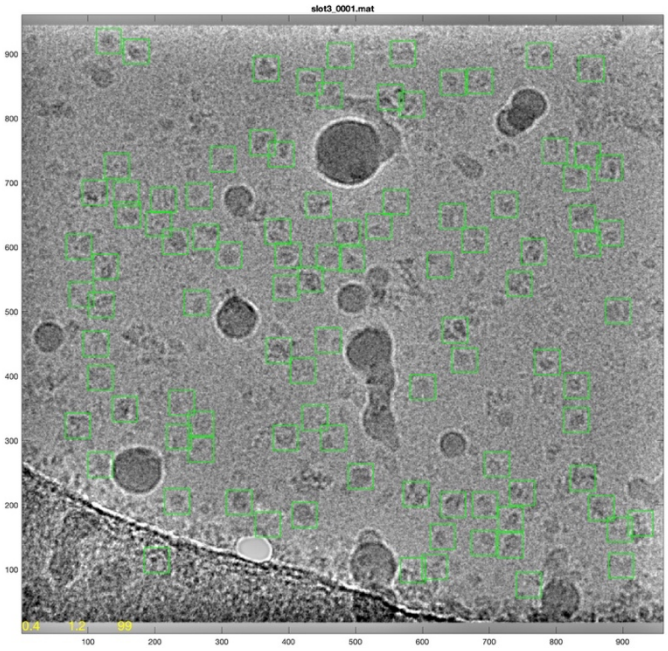


Figure 1. Particle picking in a micrograph. Putative particles are indicated by green boxes. Axes are labeled with pixel numbers for this image, which was "binned" by a factor of 4 from a K2 micrograph.

An important use for the ideas of image matching is the process of particle selection ("particle picking") on micrographs. Basically, we take one or more reference particle images and compute the correlation, correlation coefficient, or squared difference between the reference and each local position in the micrograph. We'll note the places where we get a good score, and say we've found a particle there.

To start with, we'll do the translational cross-correlation. We want to search, at every pixel position (x, y) in a micrograph, the possible match of that neighborhood with a reference particle $r_i(s, t)$. We'll evaluate the correlation as a sum over discrete s and t,

$$c_i(x,y) = \sum_{s,t} m(x+s, y+t) r_i(s,t) \qquad (4)$$

The correlation function c_i will have maxima where there is a good match, and if we choose the positions (x, y) of the maxima you will have found a place where m locally matches r_i. There is an efficient way to compute the correlation function using the Fourier transform (see Appendix 1).

Figure 2 shows an example of a cross-correlation calculation underlying the particle selection process. A portion of the micrograph (Fig. 2A) is correlated with the reference image in Fig. 2C. This reference was obtained by projecting and CTF-filtering the 3D map of a protein homologous to the one being analyzed. A peak in the correlation function (Fig.

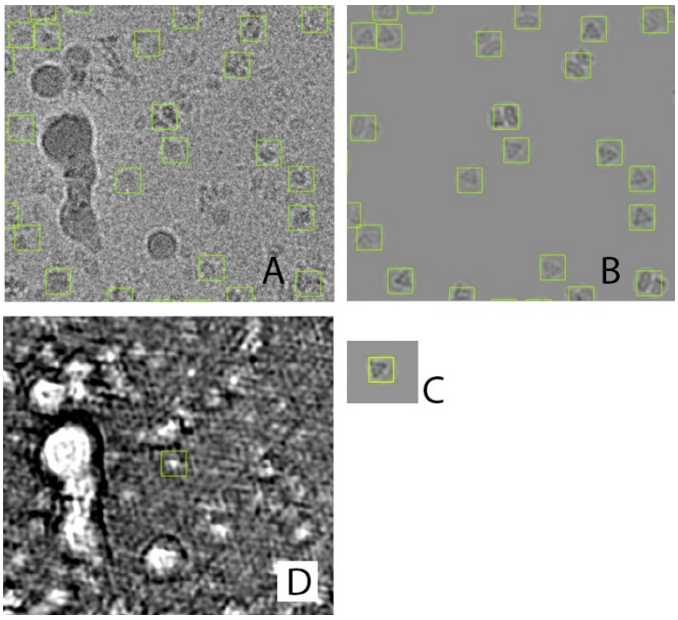


Figure 2. Cross-correlation with a reference. A, portion of a micrograph with the final particle locations marked. B, idealized picture showing the best-matching references. C, a single reference projection. D, the cross-correlation of the image with that reference. The white spot in the middle represents a correlation peak.

2D) indicates a good match with the reference. This reference will match only one view of a 3D particle. For completeness in this case the final result was obtained by performing a total of 109 correlation calculations. A set of 72 references were obtained as various projections of the model. Another set of 33 "decoy" references were also used, designed to catch, and eliminate from detection, ice balls and other artifacts. Fig. 2B shows models of the detected particles, based on the best-matching of the 72 references.

Random variables and noise

Before we go further, let's review some properties of random variables. We say that

$$b \sim \mathcal{N}(0, \sigma^2) \qquad (5)$$

means that the random variable b is drawn from a normal distribution with mean equal to zero and variance equal to σ^2. If we take many instances of b (for example, imagine repeated experiments in which the quantity b is measured) we will find that their average value tends to zero and the average value of b^2 will tend to σ^2. The variables will be statistically independent, which means that individual instances are uncorrelated. Suppose b_1 and b_2 are two instances of b. "Uncorrelated" means that the product $b_1 b_2$ will, over many trials, average out to zero.

The sum of two such variables, say $y = b_1 + b_2$ will have variance $2\sigma^2$, and in general, if you add two random variables, their variances will add. What if you make a weighted sum of k random variables? Let

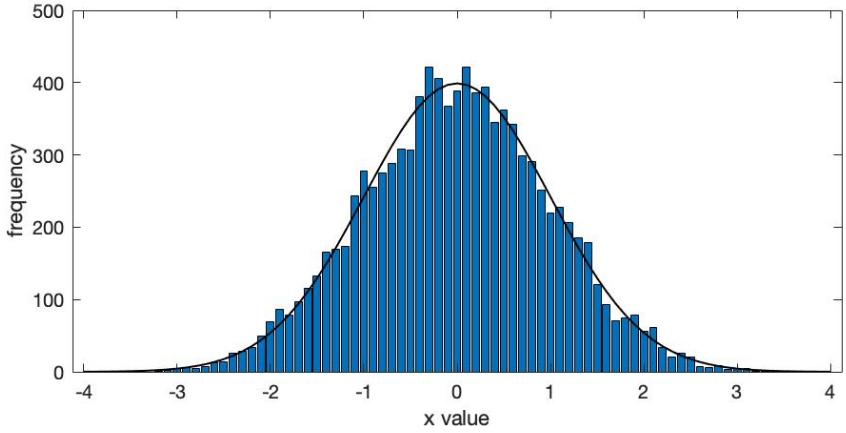
$$y = \sum_{i=1}^{k} a_i b_i \qquad (6)$$

Where the a_i are known, non-random numbers that represent weights, and b_i are random variables as in eqn. (5). Then the variance of y is given by the variances of the b_i times the square of the weights,

$$\text{var}(y) = \sum_{i=1}^{k} a_i^2 \sigma^2 \qquad (7)$$

You may be familiar with this, as it is a standard formula in computing the propagation of errors in an experimental measurement.

Finally, we need to understand the probability density function. This is the function that you would need to know if you wish to fit a histogram of values of a random variable. Suppose you make n=10000 measurements of the b_i and plot a histogram like this:



How do you compute the smooth curve to plot on top of it? The values will depend on n and on the width of the bins (0.1 in this case), and you will multiply both of these times a function that tells the probability of an instance of b_i falling in the neighborhood of a particular value χ. That function is called the *probability density function*, and for a normal distribution it is

$$f(\chi) = \frac{1}{\sqrt{2\pi}} e^{-\chi^2/2\sigma^2}. \qquad (8)$$

Roughly speaking, $wf(\chi)$ is the probability of finding an b_i value in a band of width w in the vicinity of χ. Formally to find the probability P that the value of an b_i falls in the interval between values χ_1 and χ_2 you integrate values of f,

$$P(b_i \in [\chi_1, \chi_2]) = \int_{\chi_1}^{\chi_2} f(\chi) d\chi \qquad (9)$$

Likelihood

We can compute the probability that image ρ actually is a noisy copy of the underlying reference image r_j. To do this we make a model of the process by which image ρ is formed. We'll model the noise as a Gaussian random value added to each pixel, with the random numbers having zero mean and variance σ^2. We'll represent this added noise as a "noise image" n:

$$\rho(x, y) = r_j(x, y) + n(x, y) \qquad (10)$$

The probability of observing a particular value at a particular pixel, say $\rho(x_1, y_1)$ can be calculated from the probability density function of a Gaussian centered on the expectation value $r(x_1, y_1)$,

$$P\{\rho(x_1, y_1) \in [\chi - w, \chi + w]\} = \frac{2w}{\sqrt{2\pi}} e^{-[\rho - r(x_1, y_1)]^2/2\sigma^2} \qquad (11)$$

Specifically, we are considering the probability of ρ falling in a band of values of width $2w$ centered on a given value χ.

What is the probability that all of the pixels in an image match a reference? Since noise in the pixels is statistically independent, we can calculate this as the product of the probabilities of every pixel. Suppose there is a total of K pixels. The probability of obtaining the pixel values of a given experimental image ρ, assuming that the underlying true image is r_i, is

$$P(\rho|r_i) = \frac{1}{(2\pi\sigma^2)^{K/2}} e^{-s_i^2/2\sigma^2}, \qquad (12)$$

where s_i^2 is the sum of squared differences given by eqn. (2), which we wrote compactly as $s_i^2 = \|\rho - r_i\|^2$. Note that I've left out, and will ignore, the factor w^K that multiplies the probability density to give an actual probability.[1]

Now consider the problem of determining, based on a noisy image ρ, what is the underlying true image r_j. We can test a variety of different r_i and we expect that the highest probability $P(\rho|r_i)$ will occur when we've found the best one, where $r_i = r_j$.

Perhaps a better way to formulate this problem is to try to maximize the probability of a given r_i given a particular random observation ρ. We'll formalize this by calling r_i the model and ρ the data, and say that we want to maximize $P(\text{model}|\text{data})$.[2] By Bayes' rule one can write

$$P(\text{model}|\text{data}) = P(\text{data}|\text{model}) \times \frac{P(\text{model})}{P(\text{data})} \qquad (13)$$

where $P(\text{data}|\text{model})$ is something we know how to compute: it is just what we calculated in eqn. (12), and it has a special name, the *likelihood*. One writes

---

[1] This is a common trick: because we will never care about the absolute magnitude of the probability, but will be only taking ratios of probabilities, we just ignore the astronomically small w^K factor!

[2] Statisticians argue whether such a thing can even be called a probability, as a model is not strictly a random thing; however, others argue that we can imagine a model being drawn from an infinite distribution of possible models.

$$\text{Lik(model|data)} = P\text{(data|model)}$$

Or in words, the likelihood of the model, given the data, is just equal to the probability of obtaining the data given the model. (We call it "likelihood" because it really isn't a probability in itself.) Maximum-likelihood estimation is just the process of maximizing the likelihood by picking the model that gives the largest value for P(data|model). This would be called the maximum-likelihood estimate (MLE). For example, the MLE of that maximizes the quantity in eqn. (12) is just the same as the model r_i that minimizes the squared differences as in eqn. (2).

MAP ("Bayesian") estimation

We can be a bit more rigorous by returning to eqn. (13) and considering the other terms. The quantity P(model) codifies what might be known *a priori* about the model; for example we might know that our model images are "smooth" in some sense, or always have positive pixel values, and this could be enforced by favoring models with these properties. It is called the *prior* or *prior probability* of a model, in the sense that it is known "prior" to looking at any experimental data.

The quantity P(data) is more of an imponderable; what would make one experimental image (or one dataset) more probable than another? So, not knowing what to do with this, everyone assumes it is a constant.

With these considerations we can do one better than maximum-likelihood estimation by taking into account the prior probability P(model). Formally we define the posterior probability (the probability of the model after we've considered both the prior information and information from the data),

$$\text{Posterior probability} = P\text{(data|model)} \times P\text{(model)}$$

and finding a model that maximizes the posterior probability is called the "maximum *a posteriori*" estimate (MAP estimate) of the model. The relevance of all this is that the software programs Relion and cryoSPARC find MAP estimates of models (e.g. 3D density maps) from data (i.e. stacks of single-particle images).

3D reconstruction in FREALIGN

Before considering methods for 2D classification, let's consider two modern programs for the 3D single-particle reconstruction (SPR). The first program that was developed for CTF-corrected 3D reconstruction was FREALIGN (Grigorieff 1998; 2007). It uses the Wiener filter framework to compensate for the variable amounts of information available from images due to zeros in the image CTFs.

Frealign works with Fourier transforms of images. In general I'll denote FTs of images with capital letters, and we'll denote the FT of an experimental image ρ as X. The first step in a Frealign iteration is to find, for each (Fourier transformed) experimental image X_i the five

parameters specifying the location and rotation of the original particle that gives rise to the image. We denote these parameters by ϕ_i, and we'll define the *projection operator* P_ϕ the operation that produces a 2D projection from the 3D density, with the projection angle specified by ϕ. The estimated value of ϕ_i is obtained through finding the value of ϕ that maximizes the correlation coefficient between X_i and $P_\phi V_1$. In this step we also take note of the maximum correlation coefficient, and from that value, which gives an estimate of the quality of that image, compute a weighting value w_i that we'll use in the reconstruction step.

Just as one can compute magnitudes, differences and correlations for images, it is also straightforward to compute these things for the Fourier transforms of images. A very important theorem says that the sum of squares, for real-space images is equal to the sum of squared differences for Fourier images:

$$\|\rho - r_i\|^2 = \|X - R_i\|^2 \qquad (14)$$

This is called Parseval's theorem and will allow us to calculate Gaussian probabilities just as before. Again, we'll call X_i the Fourier transform of the experimental image, V_1 is the 3D Fourier representation of the density map, and P_ϕ is the operator for obtaining the FT of an image from the FT of a volume. Indeed, the technical advances in Frealign resulted from working in the Fourier domain, and Niko Grigorieff dubbed his program Frealign for "Fourier Reconstruction and Alignment".

The reason for switching to the Fourier domain was twofold. First, by the slice theorem we know that the P_ϕ operation is very simple, it is just taking the 2D slice from the 3D Fourier volume corresponding to the angles ϕ.[3] Second, we can easily apply the contrast transfer function in the Fourier domain by multiplying the Fourier transformed image pixel values by it. We'll denote the CTF corresponding to particle image *i* by C_i.

Before we look at the second step in the Frealign algorithm, recall from earlier the idea of a Wiener filter. The problem was how to deconvolve an image from the contrast-transfer function to obtain a good estimate of the underlying object. We can't just divide the image's Fourier transform by the CTF, because we will be dividing by zero at the zeros of the CTF. The Wiener filter says we can get *R*, an optimum (least-squares) restoration of the original object by combining the image's Fourier transform *X* and the CTF *C* in this way:

$$R = \frac{CX}{\tau + C^2}$$

That is, we multiply *X* pixel-wise by the CTF, and then divide, again pixel-wise, by the CTF squared, with a small "Wiener constant" τ added to keep the denominator from going to zero.

[3] And, translations are accounted for by the multiplication by phase factors, as we know from the shift property of FTs.

The second step in a Frealign iteration is related to this. The goal is to obtain a refined estimate of the 3D density map. Based on assignments of the ϕ_i based on the initial volume $V^{(1)}$, a refined volume is obtained from the set of FTs of experimental images X_i according to

$$V^{(2)} = \frac{\sum_i w_i P_{\phi_i}^T C_i X_i}{\tau + \sum_i w_i P_{\phi_i}^T C_i^2} \tag{15}$$

Here P^T is the transpose of the P operator. Instead of *extracting* a 2D slice from a 3D Fourier volume, P^T *inserts* a slice. It takes a 2D image and creates a 3D volume, whose voxels are all zero except along the slice where the image values have been placed. Understanding this, we can describe the numerator of eqn. (15) as the following: we take each FT image X_i and multiply it, pixelwise, by the contrast-transfer function. Then we insert it into the 3D volume as a slice at the orientation ϕ, multiply by the weighting value, and add up all the slices. Because we have many images which, we hope, produce slices at many different angles, the result is a Fourier volume that has nonzero values at (nearly) every voxel.

However some voxels will have contributions from many slices, while others have contributions from only a few. Further, the image contributions to voxels will vary, depending on where the CTF zeros fall. To compensate for these variations, in the denominator we accumulate a similar sum of slices, only with the values along the slices being simply the CTF squared. A small constant τ is added to the denominator to avoid dividing by zero. Note that this formula is analogous to the Wiener filter for correcting for CTF effects, where we multiplied by the CTF in the numerator and by the CTF squared in the denominator, with a net result of approximately dividing by the CTF. In this case we're using a Wiener filter for the combination of data from many images. The resulting Fourier volume is a refined version of the starting volume $V^{(2)}$ and with it the two steps described above are repeated 10-20 times until a stable result is obtained. Frealign was the first SPR program to handle the CTF correctly, and was the first to yield near-atomic-resolution structures from single particles.

3D reconstruction in RELION

In the past few years, the majority of high-resolution structures have been obtained with RELION (Regularized Likelihood Optimization; Scheres 2012). This program provides a "maximum likelihood" solution, actually a MAP solution, to the SPR problem. Imagine the set of all possible 3D density maps (having all possible combinations of voxel values). The MAP estimate is the 3D density map that has the highest probability of arising from the given dataset of particle images. A solution to this seemingly impossible problem is provided by a famous mathematical result called the *Expectation-Maximization* (E-M) algorithm, which guarantees that in each of its iterations the result comes closer to a local maximum of the likelihood. If we begin with a good starting volume that is "close" to the

best solution, then the E-M algorithm will provide a refined result that is indeed the best solution.

Relion can be thought of as a refined version of Frealign. Whereas Frealign optimizes a correlation coefficient to find an optimal orientation ϕ_i for each image, Relion starts with a function $\Gamma_i(\phi)$, called the *latent probability*, which is the probability that a given value of ϕ is the correct assignment for image X_i. Using this probability function, rather than assigning a single "best" ϕ_i to each image, has the advantage that Relion can handle particle images having such low signal-to-noise ratios that reliable orientation determination is not possible. It also provides a smoother and more reliable convergence to a final structure, and allows the benefits from a rigorous proof that the E-M algorithm provides an optimum solution.

The advantage of Relion can be understood from what is being optimized in comparison to Frealign. Frealign provides basically an optimization of Lik(model|data) which in this case is

$$\mathrm{Lik}_{\mathrm{Frealign}} = P(\boldsymbol{X}\,|V, \{\phi_i\})$$

where the model is the unknown volume V and the set of unknown orientation angles $\{\phi_i\}$. $\boldsymbol{X}$ represents the entire set of experimental images. Relion is agnostic about the orientation angles, and instead optimizes the simple likelihood

$$\mathrm{Lik}_{\mathrm{Relion}} = P(V|\boldsymbol{X}).$$

This has the important advantage that as the number of experimental images grows, the number of parameters to be estimated does not; the goal is only to assign the voxel values in V. The way this is done is by taking an integral over all the orientation angles,

$$\mathrm{Lik}_{\mathrm{Relion}} = \int P(V|\boldsymbol{X}, \{\phi_i\}) \times P(\{\phi_i\}|\boldsymbol{X}) d\phi_i,$$

where this is actually a huge multiple integral over each of the orientation angles for each particle.

To understand the Relion algorithm we start with an explicit model for the experimental images. (Here again we are working in the Fourier domain.) We say that

$$X_i = C_i P_{\phi_i} V + N_i \qquad (16)$$

That is, a given particle image is a CTF-modified slice of the true Fourier volume V, with the instance of random noise N_i added. We specify that the noise has variance σ^2.

We need to compute the probability of the orientations $P(\{\phi_i\}|\boldsymbol{X})$, which for an individual image X_I we'll call $\Gamma_i(\phi)$. We'll use reference images of the form $C_i P_\phi V^{(1)}$, each one of which is a slice of our initial Fourier volume $V^{(1)}$ at orientation ϕ that is

filtered by the CTF. The probability that a particle image X_i matches that reference is given by

$$\acute{\Gamma}_i(\phi) = \frac{1}{(2\pi\sigma^2)^{P/2}} e^{-\left\|X_i - C_i P_\phi V^{(1)}\right\|^2 / 2\sigma^2}$$

We obtain our desired probability of ϕ given the image X_i by normalizing it,

$$\Gamma_i(\phi) = \frac{\acute{\Gamma}_i(\phi)}{\int \acute{\Gamma}_i(\phi) d\phi} \tag{17}$$

Computing this function is called the "expectation step" of the algorithm. Evaluating $\Gamma_i(\phi)$for each particle image is very costly, as it has to be evaluated over all values of the five − dimensional ϕ variable (three Euler angles of orientation and two translational coordinates); however in Relion the computation has been highly optimized. Part of the optimization is determining, for each particle image, the (usually quite limited) domain of ϕ for which $\Gamma_i(\phi)$ is significant; outside this domain Γ_i is simply set to zero. Another optimization has been the use of graphics processors (GPUs) to perform operations on many pixels or voxels in parallel; this is particularly useful in evaluating quantities such as the slice extraction $P_\phi V$.

The "maximization" step of the E-M algorithm is complicated to derive, so I'll just present the result. A refined Fourier volume, obtained from an iteration starting with an initial estimate of the volume $V^{(1)}$, is found as

$$V^{(2)} = \frac{\sum_i \int \Gamma_i(\phi) P_\phi^T C_i X_i d\phi}{\frac{\sigma^2}{T\tau^2} + \sum_i \int \Gamma_i(\phi) P_\phi^T C_i^2 d\phi} \tag{18}$$

with Γ_i computed on the basis of $V^{(1)}$. This is called the "maximization step" of Relion's refinement. The reconstruction is quite similar to the Frealign refinement step, with two differences. First, an integral over all possible orientations and translations ϕ has been added. If for image X_i there is clearly a best value of ϕ, then Γ_i will be like a delta function, and eqn. (18) will be basically the same as the Frealign step, eqn. (15). The second difference is that the Wiener constant is determined explicitly from the noise variance and an estimate τ^2 of the amount of signal contained in the reconstructed volume[4]. The value τ^2 is actually multiplied by an *ad hoc* constant *T*, typically set by the user to values between 2 and 4, to accelerate convergence.[5]

---

[4] The inclusion of the Wiener constant is actually dictated by a prior probability function in the MAP framework; Scheres chose a Gaussian prior which constrains *V* to be well-behaved.

[5] Actually in Relion the variables σ^2 and τ^2 are taken to be functions of spatial frequency. (They are 1D functions of the magnitude of the spatial frequency.) This allows the program to take into account noise that is not statistically independent from pixel to pixel, and also to take into account the fact that the signal in a reconstructed volume decays at high resolutions. Therefore σ^2 is actually the power spectrum of the noise, and an estimate of it is updated in each iteration of the

To work towards the final 3D map, one takes
$V^{(2)}$ from the maximization step and goes back to the expectation step to compute a new set of the Γ_i functions. Then another maximization step yields $V^{(3)}$, and this process is repeated perhaps 15-25 times, until a stable result is obtained.

3D classification in RELION

A big advantage of the maximum-likelihood framework is that it can be readily (and rigorously) extended to include variability in the particle images beyond just the orientations. For example, suppose that we are imaging a heterogeneous mixture of *K* distinct kinds of macromolecular particles. We model this by saying that a given particle image might arise from the particular 3D Fourier volume V_k,

$$X_i = C_i P_{\phi_i} V_{k_i} + N_i \qquad (19)$$

And we can extend the latent probability function also to include the probability that an image came from the k^{th} volume at the orientation
ϕ. We'll call this $\Gamma_i(\phi, k)$, and it is calculated on the basis of initial volumes $V_1^{(1)}, V_2^{(1)} \ldots V_K^{(1)}$. Then we can refine each of the volumes in the maximization step by

$$V_k^{(2)} = \frac{\sum_i \int \Gamma_i(\phi, k) P_\phi^T C_i X_i d\phi}{\frac{\sigma^2}{T\tau_k^2} + \sum_i \sum_k \int \Gamma_i(\phi, k) P_\phi^T C_i^2 d\phi} \qquad (20)$$

This process is called 3D classification, because in the end we can look at Γ_i and answer the question, to which of the *K* volumes did image *i* contribute the most? We can then assign that image to that "class". Notice that this sort of assignment is actually not rigorously correct, as depending on the latent probability function a given image contributes to *all* of the volumes. But at the end of the iterations there often is a clearly higher latent probability for one volume than the others.

2D classification in RELION

Relion can also do 2D classification, by almost exactly the same process as 3D classification. We can describe how it's done with the same equations too, we just have to re-define some variables.

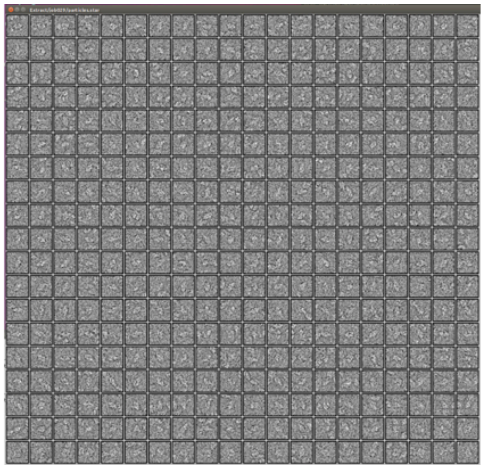
---

refinement. In Relion's auto-refine routine, τ^2 is determined from the Fourier shell correlation (FSC) which gives a rigorous estimate of the actual signal in the reconstructed volume.

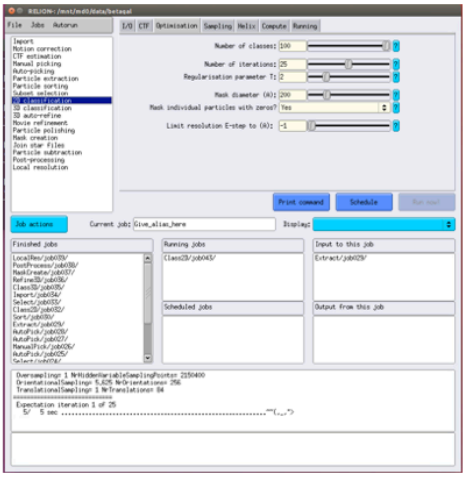
Quantity	Meaning in 3D classification	Meaning in 2D classification
V_k	Class volume	Class average image
ϕ	3 Euler angles of orientation + 2 translations	1 angle of rotation + 2 trans-lations
P_ϕ	Projection operator 3D → 2D	Image rotation and shift
P_ϕ^T	Back-projection operator 2D→3D	Reverse shift and rotation

In this case eqn. (19) represents the formation of an image based on one of the K underlying object images V_k, and eqn. (20) shows how we can improve the estimate of that class average. So essentially the same algorithm can be applied to 2D classification of images, with a big advantage that CTF correction is also applied.

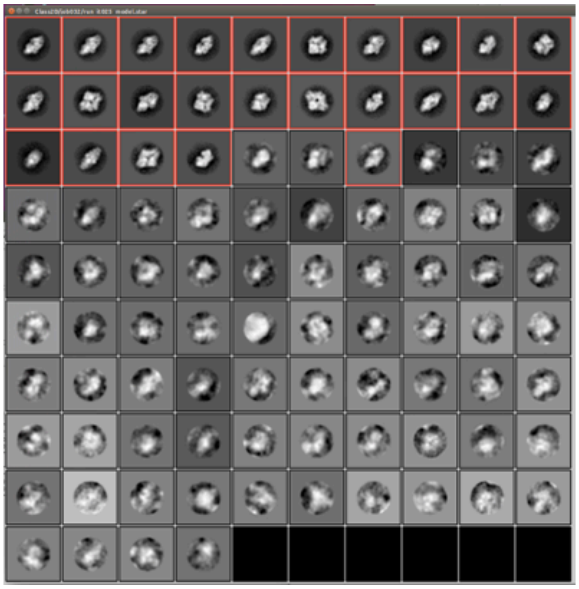
A few of the 9000 particle images extracted from the micrographs.



This is the setup for 2D classification of the 9000 images



100 class averages. The ones at the top have the strongest signal. Their particles were used for 3D classification.

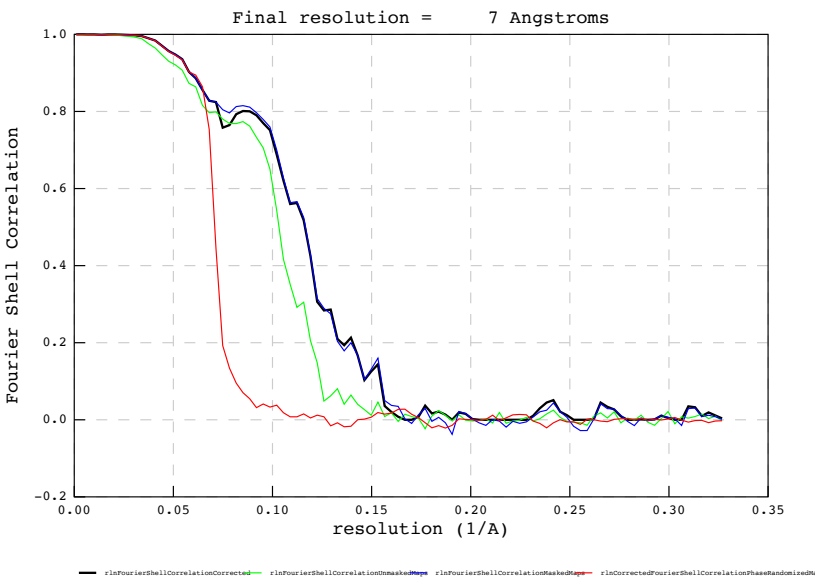


Estimating the resolution: the Fourier Shell Correlation

A problem with single-particle reconstruction is that there is no objective way to judge the quality of raw data. The visibility of particles in a micrograph gives an idea of image quality as far as the signal-to-noise ratio (SNR) is concerned. Generally the SNR increases as ice thickness decreases, but at extremely low ice thickness the particles look great but might be deformed by mechanical forces or give particularly large signal as fixed charges on "freeze-dried" particles give rise to large electrostatic potentials.

So if we make a 3D reconstruction from very noisy single-particle images, how will we know its resolution? The generally accepted approach is to compute the Fourier shell correlation (FSC) of two Fourier volumes,

$$\mathrm{FSC}(f_0) = \frac{\sum_{\mathbf{f}\in f_0} V_1(\mathbf{f})V_2^*(\mathbf{f})}{\sum_{\mathbf{f}\in f_0} \|V_1(\mathbf{f})\|\|V_2(\mathbf{f})\|}$$



which is simply a correlation coefficient between the Fourier voxel values on a shell a distance f_0 from the origins of the Fourier volumes. Below is an example of an FSC curve for a reconstruction of a membrane protein. It is the FSC computed between two independent reconstructions, each made from 12000 particles from a 24000 particle dataset. It is clear that the correlation decreases from unity at low spatial frequencies, e.g. .05 Å^{-1} which corresponds to 20 Å resolution, down to around zero at about 6 Å resolution. But what is the resolution of the map? In an important paper, Rosenthal and Henderson (2003) showed that a correlation coefficient of 0.143 between two reconstructions from half-datasets should be the same as a correlation of 0.5 between the full dataset and a noiseless "perfect" map, and this corresponds to the figure of merit m=0.5 or a weighted phase residual of 60° in X-ray crystallography. By this criterion the quality of a cryo-EM map at the resolution that gives FSC=0.143 is roughly the same as the quality of an X-ray map at that resolution.

Another question that arises is, how independent must be the two reconstructions from half-datasets? Until recently it was thought that it is sufficient to separate the two

reconstructions only during the last iteration of refinement. It became clear, however, that this gave excessive correlations due to the two reconstructions having the same starting model. In the past few years the standard in the field has become the "gold standard FSC", where the two reconstructions are started from an initial model having no signal beyond say 40 Å resolution, and are kept independent through the entire refinement process. Except for possible artifacts from masks, the resulting volumes should have nothing in common at resolutions better than 40 Å, and all correlations are due to data, not artifacts.

Another way to check for artefactual correlations is by high-frequency phase randomization. The idea is to take the entire set of single-particle images, and randomize the phases of every Fourier component higher than a particular frequency. If there is a correlation between two reconstructions from data processed in this way, it must be an artifact. The red curve in the FSC plot above is with phases randomized beyond 15 Å.

References

Grigorieff, N. Three-dimensional structure of bovine NADH:Ubiquinone oxidoreductase (Complex I) at 22 Å in ice. J. Mol. Biol. 277:1033-1046, 1998.

Grigorieff, N. FREALIGN: High-resolution refinement of single particle structures. J. Struct. Biol. 157: 117-125, 2007.

Scheres, S.H.W. RELION: Implementation of a Bayesian approach to cryo-EM structure determination. J. Struct. Biol. 180:519-530, 2012.

Appendix 1. Correlation and convolution

If we were dealing with a continuous image $m(x, y)$ and reference $r_i(x, y)$ the cross-correlation would be written as

$$c_i(x, y) = \int_{-\infty}^{\infty} m(x + s, y + t) r_i(s, t)\, ds\, dt \qquad \text{(A1)}$$

Or, compactly, people write

$$c_i = m \otimes r$$

Doesn't the correlation integral (4) look a whole lot like a two-dimensional convolution? For example this is a convolution:

$$g(x, y) = \iint_{-\infty}^{\infty} M(x - s, y - t) R_i(s, t) ds\, dt$$

Like convolution, the calculation of the correlation can be vastly accelerated using the Fourier transform. If M and R_i are the FTs of m and r_i, then the FT of the correlation c_i is given by just the product

$$C_i = MR^* \tag{A2}$$

Where the star indicates the complex conjugate. The conjugate (just flipping the sign of the imaginary part) of a function in the Fourier domain corresponds to taking a reflection in the spatial domain, i.e. reversing the sign of s and t in this case.

In practice we have micrographs and references that have finite numbers of pixels, and we have to compute sums rather than integrals, but the idea is the same.