gEMfitter: A highly parallel FFT-based 3D density fitting tool with GPU texture memory acceleration

Thai V. Hoang, Xavier Cavin, David Ritchie

gEMfitter: A Highly Parallel FFT-Based 3D Density Fitting Tool With GPU Texture Memory Acceleration

gEMfitter: A GPU-Accelerated 3D Density Fitting Tool

Thai V. Hoang[*], Xavier Cavin, and David W. Ritchie

Inria Nancy - Grand Est, 54600 Villers-lès-Nancy, France

[*] *Corresponding author*
E-mail address: vanthai.hoang@inria.fr, *Telephone*: +33 3 54 95 85 60, *Fax*: +33 3 83 27 83 19.

Abstract

Fitting high resolution protein structures into low resolution cryo-electron microscopy (cryo-EM) density maps is an important technique for modeling the atomic structures of very large macromolecular assemblies. This article presents “gEMfitter”, a highly parallel fast Fourier transform (FFT) EM density fitting program which can exploit the special hardware properties of modern graphics processor units (GPUs) to accelerate both the translational and rotational parts of the correlation search. In particular, by using the GPU’s special texture memory hardware to rotate 3D voxel grids, the cost of rotating large 3D density maps is almost completely eliminated. Compared to performing 3D correlations on one core of a contemporary central processor unit (CPU), running gEMfitter on a modern GPU gives up to 26-fold speed-up. Furthermore, using our parallel processing framework, this speed-up increases linearly with the number of CPUs or GPUs used. Thus, it is now possible to use routinely more robust but more expensive 3D correlation techniques. When tested on low resolution experimental cryo-EM data for the GroEL–GroES complex, we demonstrate the satisfactory fitting results that may be achieved by using a locally normalised cross-correlation with a Laplacian pre-filter, while still being up to three orders of magnitude faster than the well-known COLORES program.

Keywords: cryo-EM density fitting, normalised cross-correlation, Laplacian filter, fast Fourier transform, graphics processor unit, texture memory, parallel processing.

1 Introduction

Cryo-electron microscopy (cryo-EM) is an important technique for elucidating the structures of very large macromolecular assemblies. However, the resolution that may be achieved using cryo-EM is usually lower than that of X-ray crystallography. One way to obtain 3D molecular structures with atomic resolution is to fit high resolution X-ray structures into very large low resolution cryo-EM maps. In this case, the X-ray structure is often converted to the same level of resolution as the cryo-EM map by applying a Gaussian filter (Wriggers, 2010). Several algorithms and programs have been described for 3D cryo-EM fitting. Some examples of successful and widely used programs include CoAn (Volkmann and Hanein, 1999), EMfit (Rossmann, 2000), DOCKEM (Roseman, 2000), Foldhunter (Jiang et al., 2001), COLORES (Chacón and Wriggers, 2002), 3SOM (Ceulemans and Russell, 2004), MOD-EM (Topf et al., 2005), NORMA (Suhre et al., 2006), EMatch (Lasker et al., 2007), ADP_EM (Garzón et al., 2007), MolMatch (Förster et al., 2010), and PyTom (Hrabe et al., 2012).

However, thanks to the recent advances in imaging technology and 3D reconstruction techniques, it is becoming common to have maps of 256^3 voxels or greater. For example, a simple analysis of the density maps at the EMDataBank[1] shows that average map sizes are increasing steadily, and that a typical map deposited in 2012 is around 109 Mb which is nearly twice the size of a 256^3 voxel map in ordinary single precision floating point format. There is therefore a growing need for algorithms which can handle ever larger 3D density maps without sacrificing resolution. In order to try to meet this need, we have developed a new program called "gEMfitter" which can exploit the special hardware features of modern graphics processor units (GPUs) to accelerate both the translational *and* the rotational parts of a 6D correlation search. Indeed, gEMfitter is the sister program of our recently described "gEMpicker" software, a highly parallel cryo-EM particle picking tool for multi-CPU and multi-GPU systems. Because gEMfitter uses the same parallel framework as gEMpicker (Hoang et al., 2013), it can also run on multi-processor systems in the same way.

Nowadays, the scoring function and search algorithm used in most cryo-EM density fitting programs takes the form of a correlation, calculated using a series of 3D fast Fourier transforms (FFTs). This approach is also used in the related protein–protein docking problem (Katchalski-Katzir et al., 1992). In both cases, a 3D Cartesian FFT is used to accelerate the translational part of the search, which is then repeated for multiple rotational samples in order to cover the 6 rigid body degrees of freedom (DOFs). However, despite the benefit of the 3D FFT, the overall calculation is still expensive due to the cost of iterating over the remaining 3 rotational DOFs. Although it has been shown that the FFT may be used to calculate 1D, 3D, or even 5D rotational correlations (Ritchie and Kemp, 2000; Kovacs and Wriggers, 2002; Garzón et al., 2007; Ritchie et al., 2008), matching a given problem to different multi-processor configurations can lead to some unexpected implementation choices. For example, we showed that on a modern GPU it is much faster to perform protein–protein docking calculations as a series of 1D rotational FFTs than using conventional 3D FFT correlations (Ritchie and Venkatraman, 2010).

Here, we present highly efficient CPU and GPU implementations of cryo-EM density fitting. We show that the rotational part of the calculation may be accelerated considerably by using the special characteristics GPU texture memory. In order to demonstrate the utility of our approach, we compare the calculation times obtained using gEMfitter with the well-known COLORES program. To assess the quality of the solutions obtained, we show the results obtained for fitting a RecA monomer into a simulated map of RecA hexamer (Yu and Egelman, 1997), and for fitting three monomers of the GroEL–GroES complex into an experimentally determined density map (Loriot et al., 2001). These examples are chosen for ease of comparison with existing cryo-EM fitting programs.

[1] http://www.ebi.ac.uk/pdbe/emdb/statistics_map_size.html/

2 Methods

2.1 Correlation-based multi-resolution fitting

Given a 3D target density map, T, a search map, S, and a scoring function f, the problem of fitting S into T involves determining the highest values of f as a function of the position and orientation of S in T. We chose to implement only correlation-based scoring functions in gEMfitter because they can be computed rapidly using FFT, and because a recent study by Vasishtan and Topf (2011) has confirmed the utility of correlation-based techniques in the rigid-body fitting problem.

Probably the most commonly used correlation-based scoring function is the basic cross-correlation (CC). The CC between S and T is computed as

$$f(\mathbf{v})_{\text{CC}} = \sum_{\mathbf{x}} S(\mathbf{x})\, T^{\mathbf{v}}(\mathbf{x}), \tag{1}$$

where $\sum_{\mathbf{x}}$ denotes summation over all voxel coordinates $\mathbf{x} \equiv (x, y, z)$, and $T^{\mathbf{v}}$ represents a region of T having the same size as S and at a distance $\mathbf{v}$ from the origin. In order to reduce the effects of noise, Roseman (2000) proposed a locally normalised cross-correlation (NCC) scoring function,

$$f(\mathbf{v})_{\text{NCC}} = \frac{1}{P} \sum_{\mathbf{x}} \frac{(S(\mathbf{x}) - \mu_S)}{\sigma_S} \frac{(T^{\mathbf{v}}(\mathbf{x}) - \mu_{T^{\mathbf{v}}})}{\sigma_{T^{\mathbf{v}}}} \tag{2}$$

where $\mu_{T^{\mathbf{v}}}, \mu_S, \sigma_{T^{\mathbf{v}}}$, and σ_S represent the mean and standard deviation of the P voxel intensities under the "footprint" of the search object and the corresponding region of the target map, and where the summation is performed over all voxel coordinates under that footprint. Following Roseman (2003), it is shown in the Supplementary material that all of the quantities in Equation 2 may be calculated in a total of seven FFTs. Chacón and Wriggers (2002) showed that the matching of similar surface features could be enhanced if a Laplacian operator, ∇^2, is applied to the search and target maps before applying the correlation. We have therefore implemented an analytic Laplacian pre-filter in gEMfitter which may be used in conjunction with both the above CC and NCC calculations (see Supplementary material for details).

After the FFT calculation, $f(\mathbf{v})$ will contain the correlation scores for all possible voxel translations, $\mathbf{v}$, of S with respect to T. In order to accumulate the scores for multiple rotations it is convenient to let S_k and $f_k(\mathbf{v})$ denote the search map and correlation result for the k^{th} rotation of S, respectively, and to define two 3D arrays, $f(\mathbf{v})$ and $g(\mathbf{v})$ which will store the final correlation scores and angular orientations. The individual correlation arrays, $f_k(\mathbf{v})$, may then be combined using

$$f(\mathbf{v}) = \max_k f_k(\mathbf{v}) \tag{3}$$

and

$$g(\mathbf{v}) = \operatorname*{argmax}_k f_k(\mathbf{v}). \tag{4}$$

In other words, each element of $f(\mathbf{v})$ stores the maximum correlation value observed over all rotational samples calculated for $T^{\mathbf{v}}(\mathbf{x})$, and each element of $g(\mathbf{v})$ stores the index of the angular sample that gives rise to the corresponding maximum. The $f(\mathbf{v})$ and $g(\mathbf{v})$ arrays are then scanned using a watershed algorithm (Roerdink and Meijster, 2000) to find non-redundant local maxima of the correlation and to recover the corresponding translational and rotational sample values that locate the search structure in the density map (see Supplementary material for details). These coordinates are then optimised by Powell optimisation (Press et al., 2007), if desired.

2.2 Rotating 3D voxel grids

Because the above calculations are well understood, the rest of this section will focus on developing an efficient technique for rotating and re-sampling the search template in multiple orientations. Rotating the search map S to give S_k involves rotating the position of each voxel (here represented by the location of its centroid) into a new location and estimating the density to assign to the rotated position from the original unrotated data. Assuming that a volume-based linear interpolation is used, computing the density value involves fetching from memory the eight density values of the eight neighbouring unrotated voxels $(V_{000}, V_{100}, \ldots, V_{111})$, where V_{100} represents the density at the next voxel in the x direction, etc., and determining the fractional coordinates of the new voxel centre (x, y, z) relative to V_{000} (see Supplementary Figure 1). This allows the density value V at (x, y, z) to be calculated as

$$\begin{aligned} V &= (1-x)\,(1-y)\,(1-z)\,V_{000} + xyz\,V_{111} + \\ &\quad x\,(1-y)\,(1-z)\;V_{100} + (1-x)\,yzV_{011} + \\ &\quad (1-x)\,y\,(1-z)\,V_{010} + x\,(1-y)\,z\,V_{101} + \\ &\quad xy\,(1-z)\;V_{110} + (1-x)\,(1-y)\,zV_{001}. \end{aligned} \tag{5}$$

However, using this approach to rotate volumetric data on a conventional CPU is expensive for several reasons. For example, even though the CPU can load several consecutive memory values into its memory cache in a single operation, the usual strided memory layout of 3D volumetric data normally does not allow the required eight density values to be loaded together. Thus, processing a series of rotated positions can cause many “cache misses” which means that many CPU clock cycles are wasted. Furthermore, even once the density values become available, they need to be combined using a relatively large number of addition and multiplication operations per voxel (Equation 5). On the other hand, it is straight-forward to distribute the above interpolation calculation over multiple fine-grained threads on a GPU because the calculation for each rotated position can be performed independently of all the other rotated positions. However, such an approach is still not optimal because the GPU threads also need to access the voxel data in a pseudo-random fashion, and this can cause the GPU thread scheduler to serialise the memory accesses which also wastes many clock cycles (Nvidia Corporation, 2012a).

2.3 Rotating 3D voxel grids using GPU texture memory

In order to minimise the above problems, we have developed a simple but novel rotational sampling technique which exploits the special properties of GPU texture memory. In computer graphics, texture memory was originally designed to help accelerate the calculation of certain pixel blending operations by the graphics hardware. Although the specific details vary amongst different GPU manufactures, most modern texture memory implementations use the notion of space-filling curves to project 2D or 3D data onto linear memory addresses in such a way that similar spatial locations map to nearby memory addresses. Furthermore, the Nvidia hardware allows texture data to be accessed using floating-point coordinates and it provides automatic volumetric interpolation and array boundary handling. This allows random access to the 3D data in a way which exploits the hardware texture cache. Tagare et al. (2010) previously exploited GPU texture memory to accelerate interpolation calculations on 2D cryo-EM micrographs. In order to exploit the above hardware features, gEMfitter binds the main GPU memory arrays that contain 3D volumetric data to textures (Nvidia Corporation, 2012b). Thus, using textures, we can induce the GPU to perform the entire rotational interpolation calculation in GPU hardware. However, the current Nvidia texture hardware only supports nearest-neighbour and linear interpolations and it stores 3D coordinates in a 9-bit fixed point format with only 8 bits for the fractional value. Consequently, some precision is lost compared to performing the calculation on a conventional CPU. However, when working with low resolution cryo-EM density maps, we find that the introduction of a small amount of additional “noise” in this way appears to have a negligible effect on the final result (see Section 3).

2.4 Implementation and availability

gEMfitter has been implemented in the C++ programming language, and using the “Boost” C++ library (*http://www.boost.org/*) for parallel processing. The GPU version requires a system with a “CUDA” GPU device from nVidia Corp. (*http://www.nvidia.com*), and requires the CUDA run-time library to have been installed. A pre-compiled Linux workstation version of the program is available at: *http://gem.loria.fr/*. Interested readers should contact the authors to request a cluster version.

3 Results and discussion

3.1 FFT-based NCC performance comparison

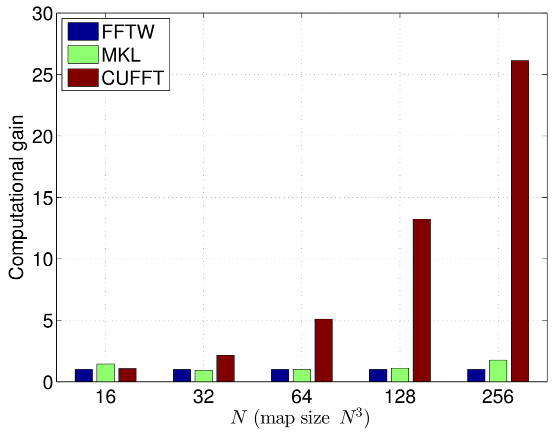
Figure 1 compares the relative computational speed of 3D NCC calculations using the open source FFTW and proprietary (Intel Corp.) MKL libraries on a CPU (dual quad-core Intel i7-965 at 3.2 GHz with 8 Gb RAM) and the CUFFT library on a GPU (Nvidia C2075 with 448 cores at 575 MHz) for a range of target volume sizes. It can be seen that while MKL is always somewhat faster than FFTW, the speed-up obtained by performing the calculation on a GPU is quite dramatic, especially for large target volumes. For example, for a target map of size 256^3 and a search map of size 64^3, using CUFFT gives about a 26x speed-up. Supplementary Figure 2 shows a similar trend for double precision calculations, although here the speed-up is only around 15x. In our experience, we find that single precision arithmetic is quite sufficient for numerical stability in the FFT (Ritchie and Venkatraman, 2010). Hence all subsequent calculations here will use single precision, unless indicated otherwise.

3.2 Accelerating 3D rotations using GPU texture memory

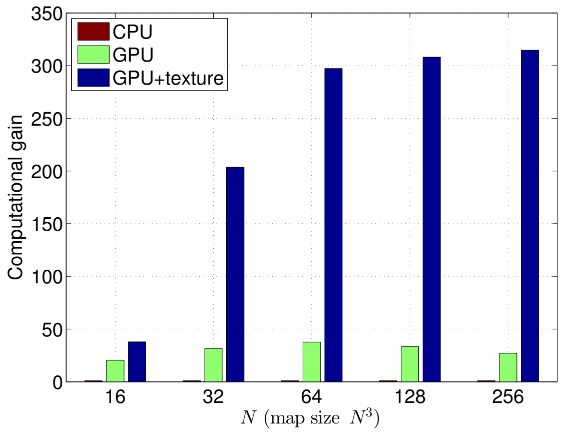
The relative computational speed of rotating 3D volumetric data for a range of map sizes is shown in Figure 1. It can be seen that directly porting the 3D rotation function from the CPU to a GPU gives $\sim$37x speed-up at map size 64^3. However, using the GPU texture memory rotation technique with the same map size gives a further 8-fold speed-up, which is effectively around 300 times faster than performing the same calculation on the CPU. With larger maps, the benefit of using texture memory increases slightly. For example, for a 256^3 map, using GPU texture memory gives $\sim$313x speed-up compared to the original CPU calculation, and a $\sim$12x speed-up compared to the initial GPU calculation. Consequently, the cost of calculating rotations on the GPU is now almost negligible, and the speed of the overall calculation is now almost entirely governed by the efficiency of the 3D translational FFT.

3.3 gEMfitter scales up linearly on multi-node clusters

As mentioned above, gEMfitter can also run on multi-node computer clusters. Supplementary Figure 3 shows the speed-up obtained when calculating the 3D single precision correlation maps between a target map of size 256^3 and 92,160 rotations of a search map of size 64^3 on our “Adonis” cluster (8 nodes x two quad-core CPUs E5520 at 2.3 GHz, 24 Gb RAM). Each node of this cluster is equipped with two C1060 GPUs (240 cores at 602 MHz) and 40 GB/s Infiniband interconnect. Supplementary Figure 3 shows that the speed-up achieved by gEMfitter increases almost linearly with the total number of GPUs available.



(a) 3D FFT speed-up



(b) 3D rotation speed-up

Figure 1: Left: comparison of the relative speed of 3D single precision FFT normalised cross-correlation (NCC) calculations at different target map sizes (N^3 voxels) using the MKL and FFTW libraries on one workstation CPU core (i7-965, 3.2GHz) and the CUFFT library on one C2075 GPU (448 cores, 575MHz). The size of the search maps are $(N/4)^3$. All timings are normalised to the FFTW (one unit). Right: comparison of the relative speed of single-precision 3D rotations on the same hardware, with and without GPU texture memory. All timing are normalised to the CPU (one unit).

3.4 Dense sampling performance using simulated 3D search maps

As a first test of gEMfitter using experimental data, we sought to model the 3D structure of the RecA hexamer (PDB code 2REC) by using one RecA monomer as the search structure and a simulated low-resolution map of the entire hexamer as the target map. Chacón and Wriggers (2002) used this example previously to demonstrate their COLORES program. The density maps were generated by using the PDB2VOL program (Wriggers, 2010) to interpolate the mass-weighted atomic coordinates to a 3D grid of 1.0 Å voxels, and by convolving each grid point with a Gaussian function whose variance corresponds to the desired resolution. Fitting was performed using a 6D exhaustive search using multiple FFTs for the 3D translational scans. Each translational scan was repeated for 92,160 icosahedral angular samples of approximately 5° generated from an icosahedral tessellation of the sphere (Ritchie and Kemp, 2000). Although these sampling densities are still small for most of the maps in the EMDB, they are not unrealistic for modern EM maps which are beginning to approach atomic resolution.

To illustrate the different scoring functions used here, Figure 2 shows 2D cross-sections of correlation-based scoring functions with and without Laplacian pre-filtering at 15 Å resolution. It can be seen that the CC maxima (Figure 2a) form a continuous ring of similar correlation values in which the expected six correlation peaks cannot be distinguished. Indeed, in this case, the C_α root-mean-square deviation (RMSD) between the CC-based reassembly and the original structure is 30.04 Å. Thus, the reassembly of the RecA hexamer using the first six highest-scoring peaks is clearly incorrect at this resolution. However, using CC with Laplacian pre-filtering (Figure 2b) improves the discrimination of the scoring function and gives six prominent peaks at the correct positions of the monomer within the target map. Using NCC by itself (Figure 2c) also gives six clear peaks at this resolution, and hence also allows the RecA hexamer to be modeled correctly. This suggests that NCC is better than CC for the multi-resolution fitting problem at 15 Å resolution. Finally, by combining NCC with Laplacian pre-filtering (Figure 2d), the scoring function again has six prominent peaks. Supplementary Figure 4 shows the result when using NCC to fit a RecA monomer into a target map simulated from the original hexamer at 15 Å. In this case, the fit is almost perfect, being just 0.35 Å RMSD from the original structure. Similar results were obtained using Laplacian pre-filtering of the CC and NCC calculations, which both gave C_α RMSD values of 0.32 Å.

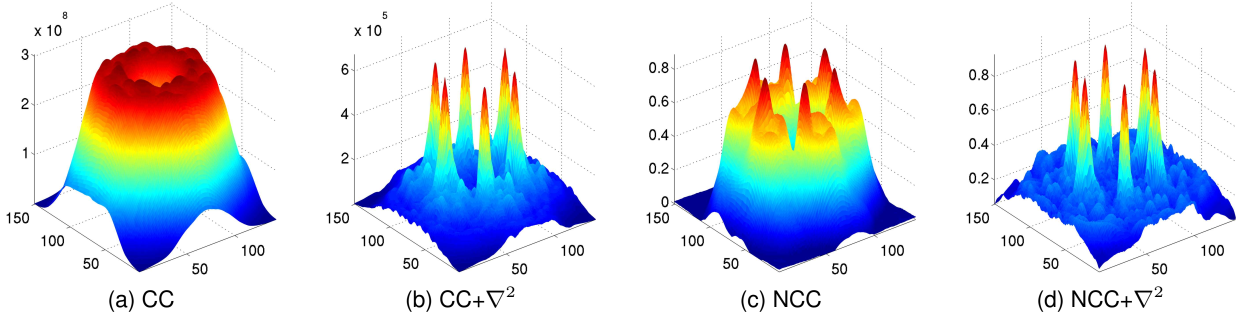


Figure 2: 2D cross-sections of the four scoring functions investigated here when fitting a monomer of the RecA hexamer into the simulated low-resolution map of the entire hexamer at 15 Å resolution. CC: cross-correlation; NCC: normalized cross-correlation; ∇^2: Laplacian pre-filtering.

In order to compare the robustness of the above scoring functions at different map resolutions, Figure 3 shows plots of the RMSD between the six highest-scoring fitted monomers and the original hexamer as a function of simulated resolution from 4 to 50 Å with an increment of 1 Å for each of the four scoring functions investigated here. Figure 3 shows that all scoring functions perform well with high resolution data, but as the map resolution becomes lower each scoring function breaks down at a certain point and the RMSD of the expected solution jumps to a large value (around 30 Å in this example). The breaking resolutions for CC, CC plus Laplacian, NCC, and NCC plus Laplacian are around 15, 36, 29, and 46 Å, respectively. While these values will probably be lower with real noise, the relative order shows that, at least for this example, NCC provides a much better scoring function than CC, and that using Laplacian pre-filtering helps to increase the breaking resolutions of both NCC and CC. Supplementary Figure 4 shows plots of the RMSD between the six highest-scoring fitted monomers and the original hexamer under different levels of added Gaussian white noise. This Figure confirms that NCC performs better than CC, and that and NCC with Laplacian pre-filtering performs better than CC with Laplacian pre-filtering at all levels of noise tested.

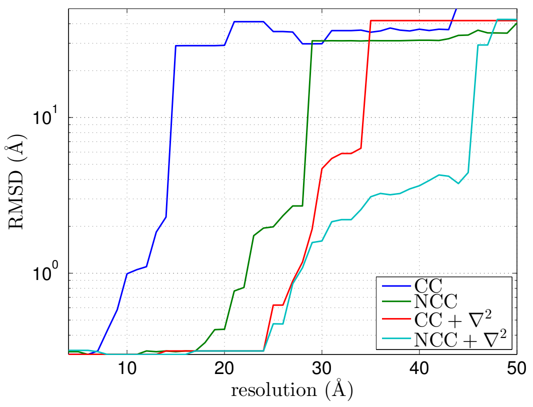


Figure 3: Plots of the RMSD of the fitted RecA monomer as a function of map resolution using the CC and NCC scoring functions with and without Laplacian filtering. The voxel spacing of the maps is 1 Å and the angular sampling is 5°.

3.5 Fitting performance with experimentally determined data

We also evaluated the performance of gEMfitter with an experimentally determined 128^3 voxel map (EMD code 1046) of the GroEL–GroES complex. This example was used to test several fitting methods in the recent Cryo-EM Modelling Challenge (Pintilie and Chiu, 2012). Fitting atomic structures into this density map is difficult due to its relatively large voxel spacing of 2.8 Å and low resolution of 23.5 Å. For this reason, only Segger (Pintilie et al., 2010) and Multifit (Lasker et al., 2009) achieved satisfactory fitting results for this complex. However, both of these methods required some help to find a solution. For Segger, it was necessary to segment manually the complex into small regions before fitting each subunit into each region. For Multifit, it was necessary to fit the monomers simultaneously to avoid clashes between the subunits.

Figure 4 shows the solution obtained by gEMfitter when fitting chains A, H, and the whole heptamer in the lid (GroES) of the PDB 1GRU into the complex using NCC with Laplacian pre-filtering and 5° angular sampling. The heptameric double-ring of the whole chaperonin (GroEL) is correctly modeled by fitting chains A and H. However, gEMfitter fails when chain O is used to model the lid's heptamer. We believe this is due to the small size of this subunit because the lid can be modeled correctly if the whole heptameric lid is used as the search structure. In other words, gEMfitter also needs some help to model this example correctly. The C_α RMSD between the search models of the cis, trans, and lid rings and the corresponding modeled structures are 4.54, 2.5, and 6.15 Å with dense sampling, and 3.97, 2.78, and 5.26 Å when using Powell off-grid optimization, respectively. Compared to the corresponding values reported for Segger (5.07, 3.06, and 6.03 Å), (Pintilie et al., 2010), these values seem quite satisfactory. It is worth noting here that running gEMfitter with 2° angular samples gave almost identical results to those shown above, but at a significantly greater cost (details not shown). Unfortunately, RMSD results for Multifit are not available.

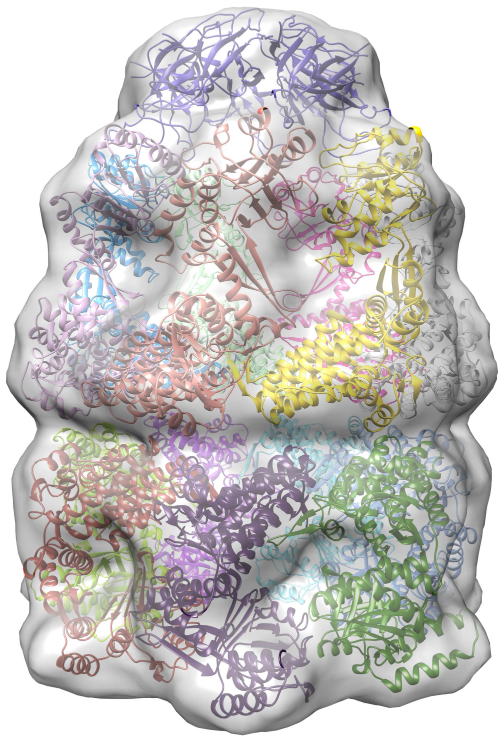


Figure 4: The gEMfitter reassembly for the experimentally determined EM density of the GroEL–GroES complex (EMD code 1046) at 23.5 Å resolution obtained using the NCC plus Laplacian scoring function with PDB code 1GRU as the search structure. The ADP and ATP GroEL subunits were fitted separately to model the cis and trans heptameric rings. For the GroES lid, the whole heptamer was used. This figure was made using Chimera (Pettersen et al., 2004).

In this test, only the NCC with Laplacian pre-filtering correlation was able to model correctly the cis and trans rings of the complex. This supports the superior sensitivity of the NCC plus Laplacian scoring. Furthermore, it is interesting to note that in our hands, COLORES was not able to model the heptameric double-ring of the whole chaperonin by fitting chains A and H separately, both when using its default off-grid optimisation mode and also when we forced it to use the same dense sampling as gEMfitter. We presume this is because the COLORES program does not use NCCs. However, it may be noted that COLORES can be used to calculate a local normalisation when invoked from the Situs/Sculptor environment (Rusu et al., 2012).

3.6 Timing comparison with COLORES

Tables 1 and 2 show the raw timings measured when using gEMfitter and COLORES for the above GroEL calculations with the above dense sampling rates and some more typical, or "coarse", sampling step sizes. For coarse sampling, we requested each program to use 24° angular steps and to apply off-grid optimisation to the top 7 local peaks. In order to achieve a reasonably fair comparison, we forced both COLORES and gEMfitter to use the same 2.8Å voxel spacing as the original map. Overall, the timings in Table 1 show that for coarse sampling with off-grid optimisation, the CPU version of gEMfitter is from around 25x (CC) to 84x (CC with Laplacian) faster than COLORES. Furthermore, for CC with Laplacian, using gEMfitter on one GPU is around 1,170x faster than using COLORES on one CPU. However, it should be taken into account that gEMfitter used 900 Euler angle samples generated from an icosahedral tessellation, whereas COLORES used 1,264 samples calculated by its default "proportional" sampling method. Thus, the real speed-up of the underlying correlation calculations should be scaled accordingly, giving speed-up factors of approximately 19x, 63x, and 875x, respectively. It is worth noting that optimising more peaks would increase the running times of both programs approximately equally.

Table 1: Timing comparison (in minutes) between COLORES and gEMfitter for fitting chain A of the GroEL–GroES complex using coarse sampling + off-grid optimisation. See main text for details. ∇^2 denotes using a Laplacian pre-filter.

	Xeon i7-965 (1 CPU core)			C2075 GPU	
	CC	CC $+\nabla^2$	NCC$+\nabla^2$	CC$+\nabla^2$	NCC$+\nabla^2$
COLORES	16.30 + 11.75	116.40 + 35.70	–	–	–
gEMfitter	0.65 + 0.46	0.65 + 1.16	1.65 + 4.94	0.05 + 0.08	0.14 + 0.69
Speed-up	25x	84x		1170x	

Table 2 shows the corresponding timings measured for the above fitting problem when using dense sampling in both programs (requested 5° angular samples and forced 2.8 Å translational steps), and with off-grid optimisation turned off. The speed-up factors obtained by using gEMfitter are even higher in this case, with the CPU version of gEMfitter being from around 23x (CC) to 164x (CC with Laplacian) faster than COLORES. For CC with Laplacian, using gEMfitter on one GPU is around 3,084x faster than using COLORES on one CPU. However, since gEMfitter again produces fewer angular samples than COLORES (92,160 compared to 119,664), the actual speed-up given by the GPU is effectively a factor of 2,300x, for example.

While gEMfitter's overall speed-up over COLORES is impressive, a significant part of this difference arises from the high cost of calculating the Laplacian filter in COLORES. In gEMfitter, the Laplacian filter is applied only once to the search and target maps, and the volumetric rotations are then applied to the filtered search map. Thus, there is almost no additional overhead in using the Laplacian in gEMfitter, whereas this is clearly not the case in COLORES. On the other hand, comparing just the CPU and GPU results for gEMfitter's CC plus Laplacian calculation shows that the GPU gives a speed-up of (0.65+1.16)/(0.05+0.08) $\simeq$ 14x for coarse sampling with optimisation and 69.7/3.7 $\simeq$ 19x for dense sampling. Hence we find that using a GPU gives a fairly consistent benefit in our calculations.

Table 2: Timing comparison (in minutes) between COLORES and gEMfitter the same complex as Table 1 when using dense sampling (5° angular samples, 2.8 Å translational steps). See Table 1 caption for further details.

	Xeon i7-965 (1 CPU core)			C2075 GPU	
	CC	CC $+\nabla^2$	NCC$+\nabla^2$	CC$+\nabla^2$	NCC$+\nabla^2$
COLORES	1581.6	11410.9	–	–	–
gEMfitter	69.7	69.7	171.2	3.7	11.1
Speed-up	23x	164x		3084x	

4 Conclusions

We have presented gEMfitter, a highly efficient 3D density fitting tool. When running on a modern GPU, gEMfitter exploits the special graphics texture memory architecture to rotate 3D voxel grids directly in the graphics hardware. This essentially eliminates the cost of calculating 3D rotations, and the speed of the overall calculation is now almost entirely governed by the efficiency of the 3D translational FFT and off-grid optimisation steps. Additionally, by calculating the Laplacian filter just once on the initial 3D voxel grid, gEMfitter is up to three orders of magnitude faster than the widely used COLORES program. These algorithmic improvements allow more sensitive correlation calculations to be performed without sacrificing speed. For example, by using the NCC with Laplacian pre-filtering, our results for the GroEL–GroES complex demonstrate that the resolution limit can be extended to around 40 Å. Clearly, very fast correlations will be beneficial when multiple correlations must be calculated, such as when fitting to maps which might be segmented in different ways or when fitting different conformations of a flexible protein. On the other hand, it would also be possible to sacrifice some speed in order to achieve even denser sampling, and thus potentially better quality fits, without requiring prior knowledge of the data. Thus, we believe the very fast gEMfitter provides a useful and flexible tool for processing increasingly large cryo-EM density maps.

Acknowledgments

This work was funded in part by CNRS and by Agence Nationale de la Recerche, grant reference ANR-MNU-006-02. We thank Willy Wriggers for helpful advice on using the COLORES program, and we thank the French Grid5000 network (https://www.grid5000.fr) for access to the Adonis cluster.

References

Ceulemans, H., Russell, R. B., 2004. Fast fitting of atomic structures to low-resolution electron density maps by surface overlap maximization. Journal of Molecular Biology 338 (4), 783–793.

Chacón, P., Wriggers, W., 2002. Multi-resolution contour-based fitting of macromolecular structures. Journal of Molecular Biology 317 (3), 375–384.

Förster, F., Han, B.-G., Beck, M., 2010. Visual proteomics. In: Jensen, G. J. (Ed.), Methods in Enzymology. Vol. 483 of Methods in Enzymology. Elservier, pp. 215–243.

Garzón, J. I., Kovacs, J., Abagyan, R., Chacón, P., 2007. ADP_EM: fast exhaustive multi-resolution docking for high-throughput coverage. Bioinformatics 23 (4), 427–433.

Hoang, T. V., Cavin, X., Schultz, P., Ritchie, D. W., 2013. gEMpicker: A highly parallel GPU-accelerated particle picking tool for cryo-electron microscopy. BMC Structural Biology, (accepted).

Hrabe, T., Chen, Y., Pfeffer, S., Cuellar, L. K., Mangold, A.-V., Förster, F., 2012. PyTom: a python-based toolbox for localization of macromolecules in cryo-electron tomograms and subtomogram analysis. Journal of Structural Biology 178 (2), 177–188.

Jiang, W., Baker, M. L., Ludtke, S. J., Chiu, W., 2001. Bridging the information gap: computational tools for intermediate resolution structure interpretation. Journal of Molecular Biology 308 (5), 1033–1044.

Katchalski-Katzir, E., Shariv, I., Eisenstein, M., Friesem, A. A., Aflalo, C., Vakser, I. A., 1992. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. Proceedings of the National Academy of Sciences 89 (6), 2195–2199.

Kovacs, J. A., Wriggers, W., 2002. Fast rotational matching. Acta Crystallographica Section D 58 (8), 1282–1286.

Lasker, K., Dror, O., Shatsky, M., Nussinov, R., Wolfson, H. J., 2007. EMatch: discovery of high resolution structural homologues of protein domains in intermediate resolution cryo-EM maps. IEEE/ACM Transactions on Computational Biology and Bioinformatics 4 (1), 28–39.

Lasker, K., Topf, M., Sali, A., Wolfson, H. J., 2009. Inferential optimization for simultaneous fitting of multiple components into a CryoEM map of their assembly. Journal of Molecular Biology 388 (1), 180–194.

Loriot, S., Cazals, F., Bernauer, J., 2001. ATP-bound states of GroEL captured by cryo-electron microscopy. Cell 107 (7), 869 – 879.

Nvidia Corporation, 2012a. CUDA C best practices guide. http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/index.html/.

Nvidia Corporation, 2012b. The Nvidia CUDA toolkit. http://developer.nvidia.com/cuda/cuda-toolkit/.

Pettersen, E. F., Goddard, T. D., Huang, C. C., Couch, G. S., Greenblatt, D. M., Meng, E. C., Ferrin, T. E., 2004. UCSF Chimera–a visualization system for exploratory research and analysis. Journal of Computational Chemistry 25 (13), 1605–1612.

Pintilie, G., Chiu, W., 2012. Comparison of Segger and other methods for segmentation and rigid-body docking of molecular components in Cryo-EM density maps. Biopolymers 97 (9), 742–760.

Pintilie, G. D., Zhang, J., Goddard, T. D., Chiu, W., Gossard, D. C., 2010. Quantitative analysis of cryo-EM density map segmentation by watershed and scale-space filtering, and fitting of structures by alignment to regions. Journal of Structural Biology 170 (3), 427 – 438.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P., 2007. Numerical Recipes: The Art of Scientific Computing, 3rd Edition. Cambridge University Press.

Ritchie, D. W., Kemp, G. J. L., 2000. Protein docking using spherical polar Fourier correlations. Proteins: Structure, Function, Genetics 39 (2), 178–194.

Ritchie, D. W., Kozakov, D., Vajda, S., 2008. Accelerating protein-protein docking correlations using a six-dimensional analytic FFT generating function. Bioinformatics 24 (4), 810–823.

Ritchie, D. W., Venkatraman, V., 2010. Ultra-fast FFT protein docking on graphics processors. Bioinformatics 26, 2398–2405.

Roerdink, J. B. T. M., Meijster, A., 2000. The watershed transform: definitions, algorithms and parallelization strategies. Fundamenta Informaticae 41, 187–228.

Roseman, A. M., 2000. Docking structures of domains into maps from cryo-electron microscopy using local correlation. Acta Crystallographica Section D 56 (10), 1332–1340.

Roseman, A. M., 2003. Particle finding in electron micrographs using a fast local correlation algorithm. Ultramicroscopy 94 (3–4), 225–236.

Rossmann, M. G., 2000. Fitting atomic models into electron-microscopy maps. Acta Crystallographica Section D 56 (10), 1341–1349.

Rusu, M., Starosolski, Z., Wahle, M., Rigort, A., Wriggers, W., 2012. Automated tracing of filaments in 3D electron tomography reconstructions using *Sculptor* and *Situs*. Journal of Structural Biology 178 (2), 121–128.

Suhre, K., Navaza, J., Sanejouand, Y.-H., 2006. *NORMA*: a tool for flexible fitting of high-resolution protein structures into low-resolution electron-microscopy-derived density maps. Acta Crystallographica Section D 62 (9), 1098–1100.

Tagare, H. D., Barthel, A., Sigworth, F. J., 2010. An adaptive expectation-maximization algorithm with GPU implementation for electron cryomicroscopy. Journal of Structural Biology 171 (3), 256–265.

Topf, M., Baker, M. L., John, B., Chiu, W., Sali, A., 2005. Structural characterization of components of protein assemblies by comparative modeling and electron cryo-microscopy. Journal of Structural Biology 149 (2), 191–203.

Vasishtan, D., Topf, M., 2011. Scoring functions for cryoEM density fitting. Journal of Structural Biology 174 (2), 333–343.

Volkmann, N., Hanein, D., 1999. Quantitative fitting of atomic models into observed densities derived by electron microscopy. Journal of Structural Biology 125 (2–3), 176–184.

Wriggers, W., 2010. Using Situs for the integration of multi-resolution structures. Biophysical Reviews 2, 21–27.

Yu, X., Egelman, E. H., 1997. The RecA hexamer is a structural homologue of ring helicases. Nature Structural Biology 4 (2), 101–104.

gEMfitter: A Highly Parallel FFT-Based 3D Density Fitting Tool With GPU Texture Memory Acceleration

Supplementary Materials

Thai V. Hoang*, Xavier Cavin, and David W. Ritchie

Inria Nancy - Grand Est, 54600 Villers-lès-Nancy, France

* *Corresponding author*
E-mail address: vanthai.hoang@inria.fr, *Telephone*: +33 3 54 95 85 60, *Fax*: +33 3 83 27 83 19.

1 FFT-based computation of local normalised cross-correlation

In its basic form, the normalised cross-correlation (NCC) between a reference (target map), T, and a template (search map), S, may be calculated as (Goshtasby et al., 1984):

$$\text{NCC}(\mathbf{v}) = \frac{1}{N}\sum_{\mathbf{x}} \frac{(S(\mathbf{x})-\mu_S)}{\sigma_S}\frac{(T^{\mathbf{v}}(\mathbf{x})-\mu_{T^{\mathbf{v}}})}{\sigma_{T^{\mathbf{v}}}}, \tag{1}$$

where $\sum_{\mathbf{x}}$ denotes summation over all voxel coordinates $\mathbf{x} \equiv (x, y, z)$, $T^{\mathbf{v}}$ represents a local region of T at a distance $\mathbf{v}$ from the origin having the same size as S, and the quantities $\mu_{T^{\mathbf{v}}}, \mu_S, \sigma_{T^{\mathbf{v}}}$, and σ_S represent the mean and standard deviation of $T^{\mathbf{v}}$ and S voxel intensities, respectively. To control the extent of the non-trivial range of the summation, it is convenient to define a mask map, $M(\mathbf{x})$, of binary values (zero or one):

$$\text{NCC}(\mathbf{v}) = \frac{1}{P}\sum_{\mathbf{x}} \frac{(\tilde{S}(\mathbf{x})-\mu_{\tilde{S}})}{\sigma_{\tilde{S}}}\frac{(\tilde{T}^{\mathbf{v}}(\mathbf{x})-\mu_{\tilde{T}^{\mathbf{v}}})}{\sigma_{\tilde{T}^{\mathbf{v}}}}, \tag{2}$$

where P denotes the number of non-zero mask elements, $\tilde{S}(\mathbf{x}) = M(\mathbf{x})S(\mathbf{x})$ and $\tilde{T}^{\mathbf{v}}(\mathbf{x}) = M(\mathbf{x})T^{\mathbf{v}}(\mathbf{x})$ denote the masked versions of $S(\mathbf{x})$ and $T^{\mathbf{v}}(\mathbf{x})$, respectively. By letting $\tilde{S}(\mathbf{x}) = (\tilde{S}(\mathbf{x}) - \mu_{\tilde{S}})/\sigma_{\tilde{S}}$ denote a normalised masked template, and by noting that $\sum_{\mathbf{x}} \tilde{S}(\mathbf{x}) = 0$ by definition, the above expression reduces to

$$\text{NCC}(\mathbf{v}) = \frac{1}{P}\sum_{\mathbf{x}} (\tilde{S}(\mathbf{x})\tilde{T}^{\mathbf{v}}(\mathbf{x}))/\sigma_{\tilde{T}^{\mathbf{v}}}. \tag{3}$$

Clearly, when all of the mask elements are unity, Equation 2 reverts to Equation 1. However, $\sigma_{\tilde{T}^{\mathbf{v}}}$ in Equation 2 must be recalculated for each value of the shift $\mathbf{v}$. Nonetheless, the general expression can still be evaluated using a total of seven FFTs by padding the template and mask with zeros up to the same size of the reference and re-writing NCC as follows (Roseman, 2003). Firstly, by expanding the denominator we obtain

$$\text{NCC}(\mathbf{v}) = \big(\sum_{\mathbf{x}} \tilde{S}(\mathbf{x})\tilde{T}^{\mathbf{v}}(\mathbf{x})\big)/\big(P\sum_{\mathbf{x}}(\tilde{T}^{\mathbf{v}}(\mathbf{x}) - \mu_{\tilde{T}^{\mathbf{v}}})^2\big)^{1/2}. \tag{4}$$

Then, using the fact that $\sum_{\mathbf{x}} \tilde{T}^{\mathbf{v}}(\mathbf{x}) = P\mu_{\tilde{T}^{\mathbf{v}}}$, the denominator may be expressed entirely as sums:

$$\text{NCC}(\mathbf{v}) = \big(\sum_{\mathbf{x}} \tilde{S}(\mathbf{x})\tilde{T}^{\mathbf{v}}(\mathbf{x})\big)/\big(P\sum_{\mathbf{x}}\tilde{T}^{\mathbf{v}}(\mathbf{x})^2 - (\sum_{\mathbf{x}}\tilde{T}^{\mathbf{v}}(\mathbf{x}))^2\big)^{1/2}. \tag{5}$$

Furthermore, because multiplication by a binary mask does not change the template and because the actions of squaring and translating voxel intensity values obviously commute, this expression may be calculated as

$$\text{NCC}(\mathbf{v}) = A(\mathbf{v})/\big(P \times B(\mathbf{v}) - C(\mathbf{v})^2\big)^{1/2}, \tag{6}$$

where

$$A(\mathbf{v}) = \sum_{\mathbf{x}} \tilde{S}(\mathbf{x}) \times \mathcal{S}(\mathbf{v})T(\mathbf{x}), \tag{7}$$

$$B(\mathbf{v}) = \sum_{\mathbf{x}} M(\mathbf{x}) \times \mathcal{S}(\mathbf{v})T(\mathbf{x})^2, \tag{8}$$

$$C(\mathbf{v}) = \sum_{\mathbf{x}} M(\mathbf{x}) \times \mathcal{S}(\mathbf{v})T(\mathbf{x}), \tag{9}$$

and where $\mathbf{v}$ represents a voxel location and $\mathcal{S}(\mathbf{v})$ represents the operation of translation by $\mathbf{v}$. In other words, the general local NCC may be calculated for all translational shifts by calculating four forward FFTs (i.e. of $\tilde{S}(\mathbf{x})$, $M(\mathbf{x})$, $T(\mathbf{x})$, and $T(\mathbf{x})^2$), and the three inverse FFTs implied by the products in the above three expressions.

In the EM multiresolution fitting problem, it is common that one target map is correlated with multiple orientations of a search map. In such cases, the forward FFTs of $T(\mathbf{x})$ and $T(\mathbf{x})^2$ need only be calculated once for all orientations. Hence, the overall computational cost can be reduced to essentially that of two forward FFTs and three inverse FFTs per search map. Furthermore, if a uniform mask is used, the computational cost essentially reverts to only one forward FFT and one inverse FFT.

2 The Laplacian kernel

In gEMfitter, the Laplacian kernel is defined as the discrete version of the Laplacian operator. We therefore choose to use the following formula to compute the filtered value at the voxel index (l, m, n):

$$\begin{aligned}
\nabla^2 f_{l,m,n} = & \qquad\qquad 0.1875 f_{l-1,m-1,n} \\
+ & \; 0.1875 f_{l-1,m,n-1} + 0.6250 f_{l-1,m,n} + 0.1875 f_{l-1,m,n+1} \\
+ & \qquad\qquad 0.1875 f_{l-1,m+1,n} \\
+ & \; 0.1875 f_{l,m-1,n-1} + 0.6250 f_{l,m-1,n} + 0.1875 f_{l,m-1,n+1} \\
+ & \; 0.6250 f_{l,m,n-1} - 6.0000 f_{l,m,n} + 0.6250 f_{l,m,n+1} \\
+ & \; 0.1875 f_{l,m+1,n-1} + 0.6250 f_{l,m+1,n} + 0.1875 f_{l,m+1,n+1} \\
+ & \qquad\qquad 0.1875 f_{l+1,m-1,n} \\
+ & \; 0.1875 f_{l+1,m,n-1} + 0.6250 f_{l+1,m,n} + 0.1875 f_{l+1,m,n+1} \\
+ & \qquad\qquad 0.1875 f_{l+1,m+1,n}
\end{aligned}$$

This formula is slightly different from the one used in the COLORES program (Chacón and Wriggers, 2002). In terms of implementation, we rely on the convolution theorem and use FFTs to perform the filtering. Thus, the actual definition of the kernel has no effect on the filtering time.

3 The watershed transform

The watershed transform (Roerdink and Meijster, 2000) has been widely used in image processing for region-based segmentation – the task of deciding which pixels belong to each object. For two-dimensional data, the principal idea underlying this method comes from geology: when the landscape is flooded by water, watersheds are the lines that divide the landscape into catchment basins, and each basin corresponds to one segmented region. In other words, each catchment basin can be assigned a unique label by mean of the watershed transform.

In the multiresolution fitting problem, it is expected that each of the catchment basins of the inverted correlation map, $-f(\mathbf{v})$, corresponds to one fitting solution. The deepest point in each basin is the local minimal and represents the translational sample value of the fit. The corresponding rotational sample value can be determined by means of the value of index map, $g(\mathbf{v})$, at that local minimal. Thus, using the watershed transform to segment the inverted correlation map and assigning a single fitting solution to each catchment basin can avoid exploring redundant fitting solutions.

In order to increase the robustness of the watershed-based peak detection step, our implementation of the watershed transform imposes a constraint on the distance between the deepest points of nearby basins. It has been observed that using this constraint effectively overcomes the problem of small granularity/noise near the global optimum. The distance value is chosen to be proportional to the dimension of the fitting structure.

4 Supplementary figures

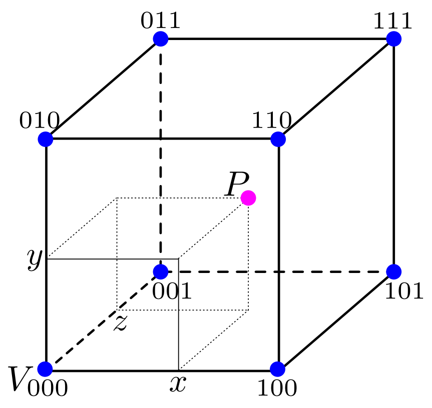


Figure S1: Illustration of volume-based 3D linear interpolation. The density value at the point P is determined using the density values of the eight neighbouring voxels $(V_{000}, V_{100}, \ldots, V_{111})$ and the fractional coordinates of P relative to V_{000}, (x, y, z), using Equation 5 in the main text.

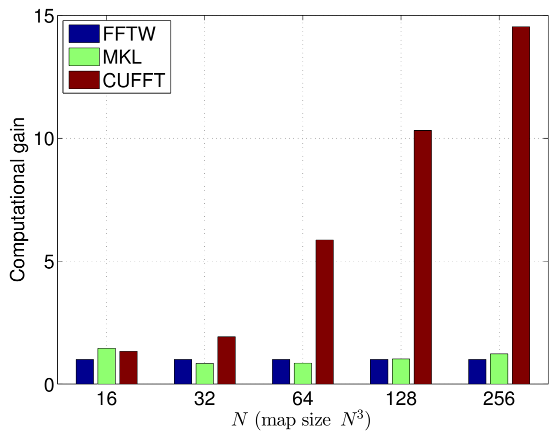


Figure S2: Comparison of the relative speed of 3D double precision FFT NCC calculations at different target map of sizes N^3 voxels using the MKL and FFTW libraries on one workstation CPU core (i7-965, 3.2GHz) and the CUFFT library on one C2075 GPU (448 cores, 575MHz). The size of the search maps are $(N/4)^3$. All timings are normalised to the FFTW (one unit).

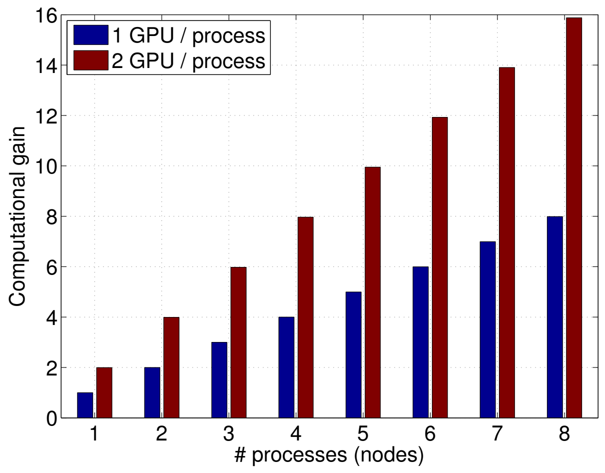


Figure S3: The speed-up obtained when using a 8-node GPU cluster having a total of 16 C1060 GPUs (602MHz, 240 cores each) to calculate 92,160 3D single-precision NCCs of size 256^3. Here, the number of threads in each process is equal to the number of GPUs used for NCC calculation (i.e. one or two depending on experiment).

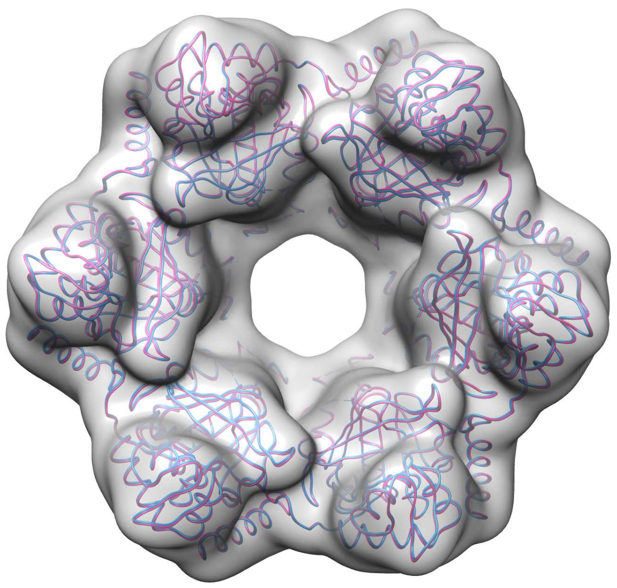
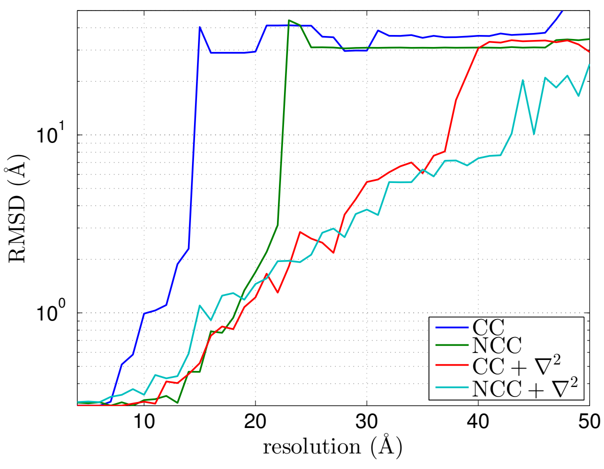
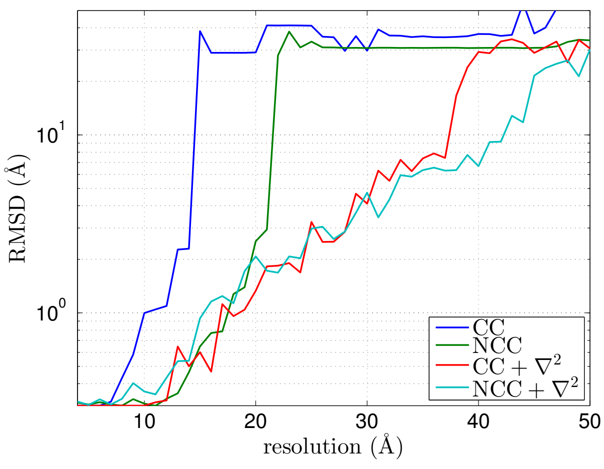


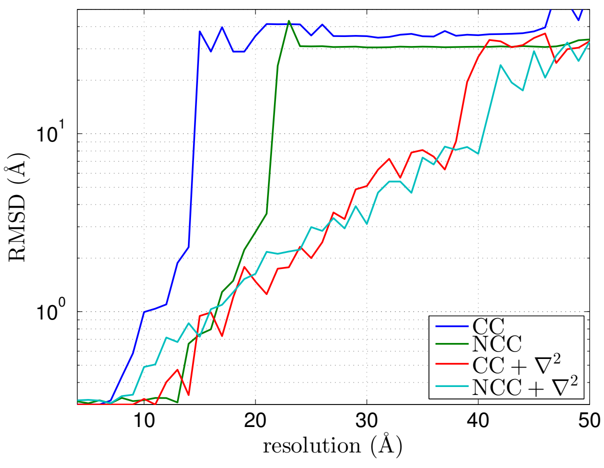
Figure S4: Fitting of the RecA (PDB code 2REC) monomer into the simulated target map of the hexamer at 15 Å resolution using NCC. An exhaustive search with translational and rotational steps of 1 Å and 5°, respectively, was carried out. The six highest-scoring fits (red ribbon) are superimposed onto the original structure (blue ribbon) that was used to generate the simulated target map.



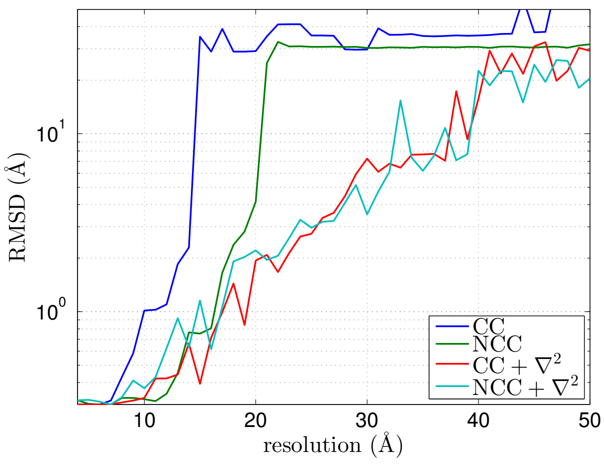
(a) SNR = 1.0



(b) SNR = 0.8



(c) SNR = 0.6



(d) SNR = 0.4

Figure S5: Plots of the RMSD of the fitted RecA monomer as a function of map resolution at different levels of added Gaussian noise using the CC and NCC scoring functions, both with and without Laplacian filtering. The voxel spacing of the maps is 1 Å and the angular sampling is 5°. Here, SNR is defined as $var(s)/var(n)$, where $var(s)$ and $var(n)$ are the variance of the original map (signal) and the variance of the added white Gaussian noise, respectively.

References

Chacón, P., Wriggers, W., 2002. Multi-resolution contour-based fitting of macromolecular structures. Journal of Molecular Biology 317 (3), 375–384.

Goshtasby, A. A., Gage, S. H., Bartholic, J. F., 1984. A two-stage cross-correlation approach to template matching. IEEE Trans. Pattern Anal. Mach. Intell. 6 (3), 374–378.

Roerdink, J. B. T. M., Meijster, A., 2000. The watershed transform: definitions, algorithms and parallelization strategies. Fundamenta Informaticae 41, 187–228.

Roseman, A. M., 2003. Particle finding in electron micrographs using a fast local correlation algorithm. Ultramicroscopy 94 (3–4), 225–236.