Orientation Determination in the 3D Reconstruction of Icosahedral Viruses using a Parallel Computer

C.A. Johnson[1], N.I. Weisenfeld[1], B.L. Trus[1,2], J.F. Conway[2], R.L. Martino[1], A.C. Steven[2]

1 Division of Computer Research and Technology
2 National Institute of Arthritis and Musculoskeletal and Skin Diseases
National Institutes of Health

submitted for publication in the proceedings of Supercomputing '94

Orientation Determination in the 3D Reconstruction of Icosahedral Viruses using a Parallel Computer

C.A. Johnson[1], N.I. Weisenfeld[1], B.L. Trus[1,2], J.F. Conway[2], R.L. Martino[1], A.C. Steven[2]

1 Division of Computer Research and Technology
2 National Institute of Arthritis and Musculoskeletal and Skin Diseases
National Institutes of Health

Abstract

Three dimensional image reconstruction from cryo-electron micrographs allows the capsid structures of icosahedral viruses to be studied. The most computationally demanding stages of the reconstruction process are those that find and refine the orientations of the virus particles. We have devised and implemented parallel solutions to the problem of determining these orientations. By enabling determination of the orientations of far more particles than previously had been possible, parallel processing methods have contributed to improvements in the quality of recent reconstructions.

1: Introduction

High resolution cryo-electron microscopy in combination with the three-dimensional computer image reconstruction of the obtained micrographs allows the structure of herpes simplex virus (HSV, 125 nm diameter, a reconstruction of which is presented in Figure 1) and other large virus particles to be studied. These reconstructions attain a resolution of approximately 3 nm and are proving adequate to map the relative locations of four major capsid proteins [13,15]. Our plan to extend the resolution of these studies depends heavily on high performance computer technology and parallel algorithms. This computational requirement is imposed by the need to handle more particles and to achieve finer sampling, correction of the phase contrast transfer function, and more exact determination of the orientational and positional parameters. To pursue this goal we are porting and where necessary adapting reconstruction software to run on a highly parallel computer, the 128-processor Intel iPSC/860 [11].

The reconstruction process takes advantage of the icosahedral symmetry of the virus particle and of the Fourier slice theorem [7,8,9]. The set of particles in a 2D cryo-electron micrograph can be thought of as identical in 3D but oriented differently in their 2D projections on the micrograph. In Fourier space, each 2D projection is equivalent to 59 other symmetry-related views that intersect at the Fourier origin. The reconstruction method used here requires that we interpolate, in Fourier space, all particles and their symmetry-related views into cylindrical coordinates and then take an inverse transform to real space. Figure 2 represents the reconstruction process as a set of discrete stages beginning with cryo-electron microscopy [5] and digitization using a Perkin-Elmer 1010MG microdensitometer. Pre-processing steps include selection of individual particles to be included in the reconstruction, removal of extraneous features such as the

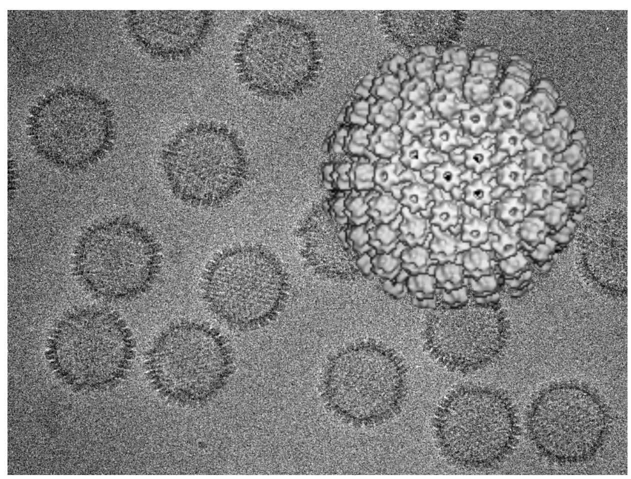

Figure 1: Background is a cryo-electron micrograph of herpes simplex virus (type 1) A-capsids, empty of DNA. Foreground shows a surface shaded representation of a 3D computer reconstruction obtained from 104 2D images. See acknowledgment.

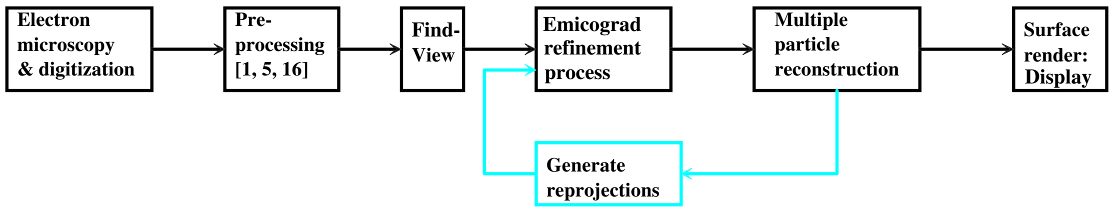

Figure 2: 3D reconstruction steps at NIH.

intruding neighboring particles or coprojected pieces of contaminating ice, background gradient subtraction, imposing a standard renormalization for constant mean and variance, and estimation of the center of each particle [5,16]. Before the reconstruction can occur, reasonable estimates of the orientation and origin of each particle must be made. Finally, the 3D reconstruction in Fourier space is performed as described above, and the resulting map is displayed using a variety of methods.

The original versions of the reconstruction software were given to us in scalar form, to run under the VAX/VMS environment, by Dr. T.S. Baker of Purdue University. We have adapted these programs to run in parallel on a 128-processor Intel iPSC/860, and have written a front-end interface for a Unix workstation. Two major computationally intensive steps are required for determining the particle orientation: an initial orientation estimation stage (FindView), and an orientation refinement stage (Emicograd). The latter actually describes an entire process, with two components (global and local refinement) that require separate parallel implementations.

2: Methods

2.1: FindView

The FindView procedure relies on the common-lines procedure of Fourier analysis [7,8,9], as implemented and modified by Baker [1,2,3] and Fuller [10]. According to Crowther, "An observed section of the transform intersects an identical symmetry-related section in a line, along which the transform must have the same value in both sections. The common line lies in the original section. But, regarded as lying in the symmetry-related section, it must have been generated by the symmetry operation from some other line in the original section. We have therefore a pair of lines in the original transform plane along which the transform must have identical values. A similar pair of lines will be generated by each possible choice of pairs of symmetry operations. The angular positions of these lines depend on the orientation of the particle." [7]. Figure 3 illustrates the spherical coordinate orientation parameters θ and ϕ, as well as the asymmetric search unit, which is defined as a triangle with its apex at θ=69.09^o and its base at θ=90^o. At the base of the triangle, ϕ is defined from -31.72^o to 31.72^o. Icosahedral symmetry conditions permit a search to be limited to this asymmetric search unit because orientations outside the search unit can be mapped to an equivalent orientation inside the triangle. A “brute force” search scans, in approximately 1^o (or some other fixed) increments, the entire search unit. For every sampled orientation, the program calculates the sum of the squared differences between the Fourier phases along the common-line pairs. Those orientations with the lowest *phase residuals* are chosen as candidates for the actual orientation of that particle.

FindView itself consists of several steps that need to be performed prior to the search for the best orientations. The block diagram of Figure 4 summarizes these steps. The program reads in an image file corresponding to a single projected view, and then performs a 2D fast Fourier transform (2D FFT). From information supplied regarding the estimated origin of a particle image, a complex exponential phase correction factor is applied to the image transform. To facilitate processing of the common lines, the Fourier image transform is interpolated from Cartesian coordinates onto a polar coordinate system. Due to Hermitian symmetry of the transform, only the upper half-plane is needed, and the transform is interpolated onto the radial lines Ω=1^o,1^o,2^o,...,180^o. The user selects an inner radius r_0 and an outer radius r_1, and the program deletes those points outside these half-circles, thus filtering out the low-frequency terms and high-frequency noise (refer for example to Figure 5). The next step zeros out the values of the transform where the magnitude is less than a linear regression model's expected magnitude for each ring, thereby suppressing other possible sources of noise.

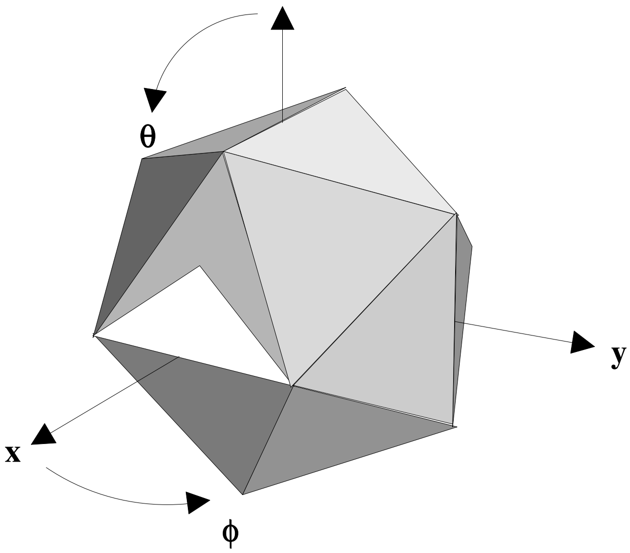

Figure 3: Icosahedron showing spherical coordinate parameters θ and ϕ. The asymmetric search unit is the white triangle (adapted from [3]).

For each new orientation *{θ,ϕ}* in the asymmetric search unit we generate a set of 37 common line pairs *{α1(ν),α2(ν)}*, *ν*=1,2,...,37. These common line pairs should be unique to that particular orientation, and are calculated using rotation matrices that reflect the icosahedral symmetry conditions. FindView determines *J(θ,ϕ),* the number of common line pairs which are not independent or unique for a particular orientation, resulting in 37-*J(θ,ϕ)* independent common line pairs per orientation. The rotation angle *Ω* indicates the position of the common lines relative to the Cartesian axes [2,10]. For each orientation we rotate the α pairs by $\Omega=0^o,1^o,2^o,...,180^o$ and determine the rotation angle which provides the smallest average phase residual over the 37-*J(θ,ϕ)* independent common lines. For a given *{θ,ϕ}*, the *Ω* that yields the best phase residual is stored in the array **Ω***[θ,ϕ]*, and the associated phase residual is stored in the residual array **Γ***[θ,ϕ].*

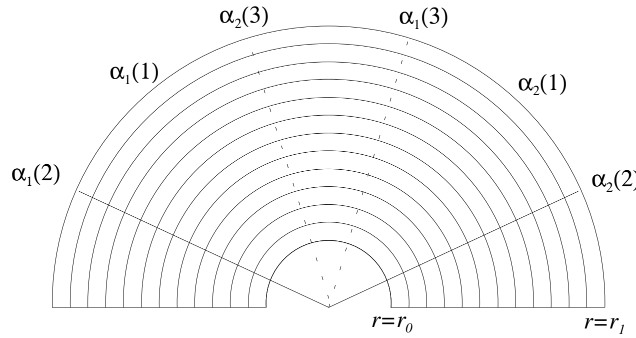

Figure 5: Primary particle plane in Fourier space, showing 3 of the 37 pairs of common lines and the cutoff resolutions r_0 and r_1. This is the configuration for a particle at Ω=0 in the search.

Recognizing that the residual array favors those orientations with the fewest common line pairs, Fuller introduced a χ^2 weighting scheme which uses 37-*J(θ,ϕ)* degrees of freedom per orientation [10]. The residual array is weighted according to the probability that a χ^2 variable will exceed the average residual **Γ***[θ,ϕ]* at the chosen **Ω***[θ,ϕ]*, normalized with respect to the average residual over all *Ω*. The weighting factor takes into account the average aforementioned probability over all orientations. Note that the residual arrays **Γ***[θ,ϕ]* and **Ω***[θ,ϕ]* are occupied only within the triangle defined by the asymmetric search unit. A third array, $\mathbf{P}_\gamma[\theta,\phi]$, which represents the χ^2 probability, is also stored during the search so that the weighting factors may be applied following the search loop while allowing the average probability to be taken into consideration. Following the search, a procedure constructs the *best list* of possible orientations for the particle in question, accounting for local minima trends in the residual array and mirror images across ϕ=0. Finally, the original estimates of the center of the particle are updated using common line information for every "best listed" orientation. Upon its conclusion, FindView produces the list of *n* best estimated orientations with corrected origins for every particle in the reconstruction study.

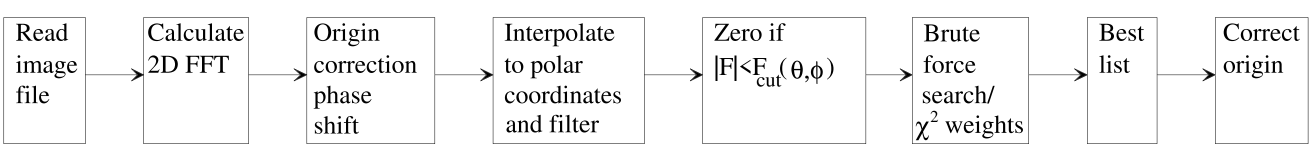

Figure 4: FindView block diagram

2.2: Emicograd

The orientation refinement code, known as "Emicograd", refines the initial orientation supplied by FindView and produces a final set of particle orientations to be used in the reconstruction. In order to compare the accuracy of the estimated orientation of a given particle against the others in the set, Emicograd determines the mean-squared difference (residual) between particles along their corresponding "cross-common lines", radial lines in the FFT of the projections representing where the symmetry-related views of the two particles would intersect. Emicograd seeks to find the minimal cross residual $R_{i,j}$ between F_i and F_j (the *ith* and *jth* particle Fourier images) for all particle combinations. The refinement process requires many separate runs of Emicograd. Early runs build and refine the orientations to obtain a basis set of stable particles, adding particles to this set incrementally. Once completely built, the basis set is then used to refine all other particles' orientations by running a "local refinement" (all particles constrained except the last) on each additional particle. The complete set of refined particles can then be entered to a "global refinement" (all particles unconstrained) of all particles in the set.

Starting with the initial orientations from FindView, Emicograd computes the cross-residuals $R_{i,j}$, $i,j=1...N$ between every particle F_i and every other particle F_j in the study set of N particles. Then Emicograd enters a refinement cycle where the orientations θ_i (polar angle), ϕ_i (azimuthal angle), and Ω_i (in-plane rotation) are altered by some fixed amount in a *simplex move* [12], initially plus and minus 1.5 degrees. The cross residuals of these moved primary images are calculated against the stationary secondary images $j=1...N$ for all 6 possible simplex moves. If any of the modified residuals are less than the existing residual for the stationary image F_i, then particle F_i takes a simplex move in the favored direction. If the existing residual is lowest, Emicograd finds the direction gradient of the minimum residual, tries moving the primary particle in that direction and compares the gradient-moved residual with the existing stationary residual. If the gradient move fails to improve the average cross residual between particle F_i and the other particles by 1%, then particle F_i is grounded and cannot move for the rest of the refinement cycle. Otherwise, the gradient move is performed and particle F_i remains on the list of active particles. Because the refined orientations are sometimes ambiguous in the polarity of Ω (i.e., Ω versus $\Omega+180^o$), Emicograd also iteratively determines the most self-consistent "handedness" polarity for the particles in the set, and automatically flips the particle images (i.e., replaces Ω_i with Ω_i+180^o) when necessary. A number of automated flipping strategies were developed to allow the refinement program to extend the possibilities offered by the original software distribution.

The refinement process, as represented in Figure 6, actually requires many cycles of building study sets, submitting refinement jobs, studying the results, and properly disposing of the results. The process of building these study sets can be quite laborious, and with the increase in computational power available to run Emicograd jobs, it seemed appropriate or even necessary to build more sophisticated software for driving these runs. Thus we developed the "gradhost" user interface for control of the entire refinement process.

3: Parallel Implementation

3.1: FindView

The FindView program has been successfully ported to a 128-processor MIMD parallel computer. Running sequentially, the brute force search comprises the vast majority of CPU time within FindView. For that reason, we have concentrated on modification of the search to run in parallel. The chosen strategy processes each particle sequentially while running in parallel within the search, i.e. the residual calculation for each view is simply assigned to a different available processor element (PE). The other components of the FindView code (with the exception of the origin correction, to be discussed below) run on only one PE, but those components contribute little to the overall computational burden.

Fortunately, the residual calculations performed during the search at a given orientation are independent of the residual calculations at other orientations. This suggests that individual PEs can perform the calculations over certain specified orientations, allowing the other PEs to complete the remaining orientations. Figure 7 illustrates how this strategy allows a set of orientations in the asymmetric search unit to be distributed over the number of PEs allocated for the job. At the start of the search loop, the residual arrays $\mathbf{\Gamma}$, $\mathbf{\Omega}$, and $\mathbf{P}_\gamma$ are all initialized to zero. During the loop, any given PE processes only its assigned orientations, leaving the remaining array elements zero. At loop termination, the PEs exchange information about the contents of the (moderately small) arrays in a global sum operation. Every PE sums its own values in the array $\mathbf{\Gamma}$ (for example) with the values stored in the other PEs for $\mathbf{\Gamma}$. This results in every PE storing the full array because the unused elements were zero prior to the summation. Following the global operation, the residual array $\mathbf{\Gamma}$ is weighted by $\mathbf{P}_\gamma$, in the manner described in Section 2, and the "best list" is generated on a single PE. Because the PEs do not need to communicate until the end of the search, this process represents a perfectly parallel application. The load balance varies only slightly due to small differences in the number of independent common lines per orientation.

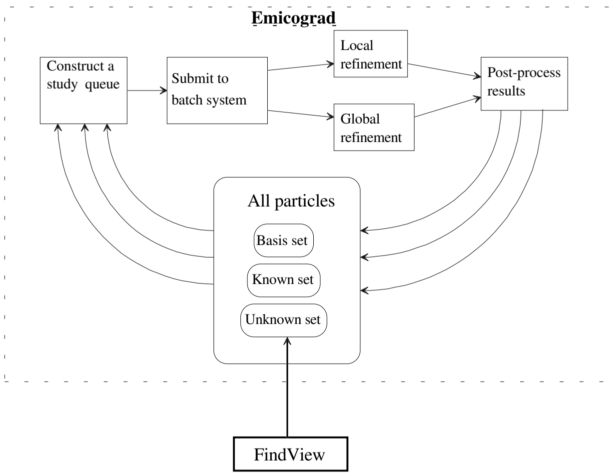

Figure 6 - The refinement process, as implemented at NIH on the Intel iPSC/860 parallel computer, with Unix front-end host software. The *n* best initial orientation estimates per particle from FindView are stored in the unknown set. The "gradhost" front-end software allows users to construct study sets from the particles in the basis set of stable particles, the known set, and/or the unknown set. These study sets are submitted to either the local refinement or global refinement engine via the queuing system. The gradhost software also provides an interface for post-processing the refinement results and re-ordering the particles with respect to set membership.

The other component of FindView which has been parallelized is the origin correction code. After the 'best list" of orientation estimates has been generated, FindView proceeds to correct the origin (center of the particle $\{x_o,y_o\}$ in the image plane) estimate which was made during preprocessing. An origin correction is performed for every orientation in the "best list". An origin shift in real space corresponds to a phase shift in Fourier space. Therefore, to find the origin shift with the lowest average phase residual we recompute the average phase residual, taking into account the phase shift due to the correction being tested. The origin correction iterates, searching for the origin $\{x_o,y_o\}$, starting with a search interval of $\Delta x = \Delta y = 0.5$ pixels over a range of ±2.5 pixels, and gradually reducing the search interval. Parallel implementation consists of evenly distributing to the processors the calculation of the average residual over the radii of the common lines involved. This approach requires very little communication and is highly scalable.

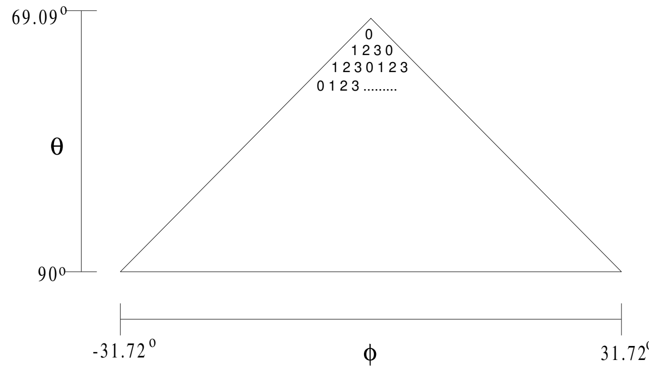

Figure 7: Asymmetric search unit, illustrating pattern of parallelization. Numbers inside triangle represent processors assigned to that orientation in the search (4 PE case).

3.2: Emicograd

The computational characteristics of local and global refinements differ enough that they have been implemented as separate programs on the parallel computer. In a global refinement, all particles are unconstrained and their orientations can be moved. Since global refinements are often performed on large numbers of particles, individual jobs can be quite large. In a local refinement, all but the last particle are constrained, so the refinement is only performed on the last particle in the study set. Local refinements are usually performed on many, if not all, of the "best list" orientations in order to determine which of the FindView orientations is best. Local refinements run on small to moderately sized sets, typically 10 to 30 particles. If the subset of constrained particles is the same between different local refinements, then the results of those refinements are independent of each other, and many local refinements can be run concurrently. The choice of parallelization strategies clearly depends on the problem size and the scalability of the problem decomposition. Global refinements have been parallelized *within-process* (i.e., the kernel of parallelization is a procedure within the process) with the images distributed across the PEs. Local refinement processes are each distributed to a single PE, with many processes running asynchronously in parallel.

3.2.1: Global Refinement

In a global refinement, memory constraints dictate that the Fourier images be distributed across the PEs.

Most of the work in Emicograd is done by the subroutine that computes $R_{i,j}$, the residual between primary image F_i and secondary image F_j. That subroutine is called in two different places: first, when calculating the residual between image F_i and all the secondary images, usually as part of the process to calculate the residual matrix; and second, to calculate the six average moved residuals when the primary image is moved in the six simplex directions. In both cases the scenario is quite similar: the parent PE of image F_i broadcasts F_i to all other PEs. The PEs then compute $R_{i,j}$ only for their assigned secondary images F_j. Consider the 6 image, 4 PE example depicted in Figure 8, where i=1 and PE 0 broadcasts image F_1 to all the other PEs. PEs 1, 2, and 3 receive image F_1 as a "guest image" so that they may use it as the primary image on their PE to compute the cross residuals against the secondary images F_j that they own.

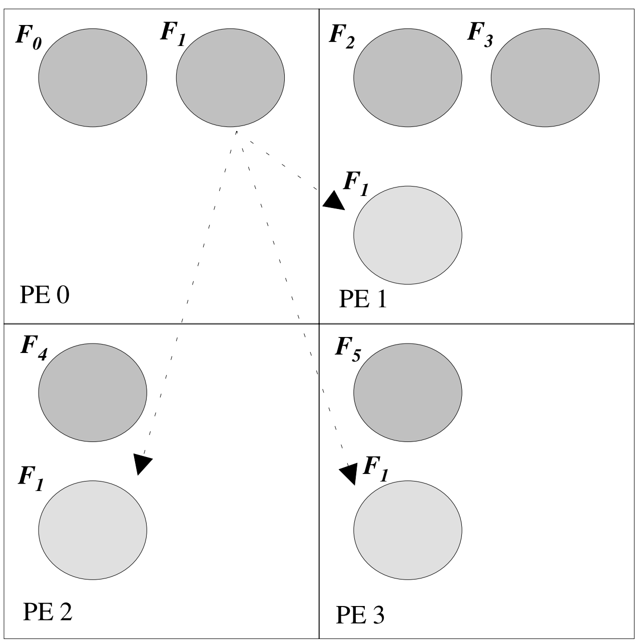

Figure 8: Global refinement computation of $R_{1,j}$ for all j. Shown is a 4 processor, 6 particle image example. PE 0 broadcasts image F_1 to all other PEs.

The global refinement program taxes memory resources considerably. Each PE stores certain images in full, plus some information about all images (e.g. orientation, origin, average cross-residual), all of which contribute to image allocation overhead. The current release of the global refinement distributes the Fourier images, although the residual matrix $R_{i,j}$, which stores the phase residual of every image F_i against every other image F_j, is replicated on all PEs. The storage of the residual matrix greatly contributes to the image storage overhead for large problems. A certain amount of memory must be allocated as overhead to each PE (to store the global information and residual matrix), and this overhead reduces the number of images that any PE may host. By decreasing the image allocation overhead M_i, the maximum number of images that can be processed on a given sized cube increases, subject to the constraint of the maximum number of images imposed by M_i (see Figure 9). Distribution of the residual matrix would drastically reduce image overhead and thereby increase the maximum allowable problem size, but has not yet been implemented because it requires additional communication time between the processors and the number of particles involved in reconstructions has not been large enough to require it.

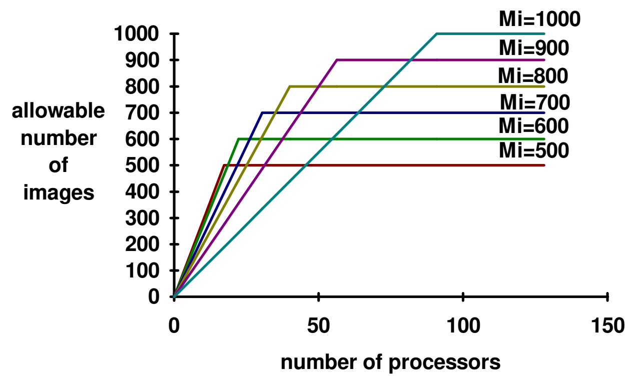

Figure 9: Global Emicograd problem size in allowable number of images as a function of number of PEs and image allocation overhead M_i, with residual matrix replicated.

3.2.2: Local Refinement

The parallel global refinement program can actually be used to perform local refinements, and is used for that purpose when over 30 particles are involved. The problem with doing so is that since local refinements are usually performed on small run sets (10 to 15 particles), the global parallelization method would not be scalable for the problem. As was mentioned previously, a local refinement refines a particular unknown particle against a set of constrained particles which presumably require no further refinement, at least for the present. The correct orientation of the unknown particle from the FindView "best list" may not be clear, so several (or all) of the best list orientations of the unknown particle may need to be attempted. The orientation which provides the best (lowest) average residual of the unconstrained particle usually wins. The set of constrained particles used to refine one unknown particle with multiple orientations can also be used to refine other particles. Many local refinements can in fact be executed concurrently because the results of these refinements are independent.

Defining a single run of a local refinement to be a single execution of Emicograd on a single orientation of a single unknown particle, local Emicograd is parallelized by distributing these runs. The local refinement distributes the individual runs of a *refinement queue* across the PEs. A worker-manager approach has been employed where logical PE 0 assigns runs to processors as they become available. When a processor completes a run, PE 0 transfers the output files from that run back to the remote host workstation to await post-processing. The first *N*-1 images (out of *N* in the study set) are always common to all runs in the queue, and are read in and broadcast by PE 1 on the first run only and retained for subsequent runs. The *Nth* image, which may or may not be common to the runs, is read by each run, thereby causing some file contention. A local refinement queue can run thousands of small refinement runs. Each of these refinements must read input files from, and write output files to, the remote workstation's file system. Although much of the data can be discarded, this can only be done after all of the data has been processed and then only through user interaction.

4: Results

4.1: FindView

Table 1 lists the parallel benchmarks measured for a one-particle job. Because the multiple particle analyses of which a typical FindView job is composed are run sequentially, the single-particle benchmark is a meaningful measure of parallel performance. Table 1 provides run-time measurements and performance improvement (speedup) values for the overall FindView process as well as the parallel brute force search and the origin correction components of FindView. Figure 10 compares the efficiency of these same three FindView components. The search is remarkably efficient, with near-linear performance speedup. The origin correction includes non-parallelized operations such as finding the common line angles. Consequently, the origin correction performance is not nearly as impressive as that of the search. Together, the orientation search and origin correction code consume the vast majority of CPU time in FindView runs (99.67% of CPU in the single-PE case), although other operations (FFT, file I/O) do contribute slightly.

4.2: Emicograd

Table 2 provides the performance measures for the global refinement for a 55 image set. The single-PE result is a projection because at least two PEs are required to refine 55 particles due to memory restrictions. A 128 processor test would achieve no speedup on a 55 particle set because 73 processors would sit idle. The best load balance occurs when the number of images per PE divides evenly. Load balance suffers when the range of particles/PE is high, as is the case when the number of images approaches the number of PEs. Despite the heavy communications load of transferring images, the global refinement parallelizes with reasonable performance improvement.

Tables 3 and 4 list the speedup obtained by a 60-run local refinement queue excluding and including file transfer to the remote host, respectively. Load balance is adversely effected at 32 and especially 64 PEs because the number of runs assigned sequentially to each processor is limited. In practice, most refinement queues are much larger than 60 runs, and the load balance issue is not critical. Efficiency also suffers when small numbers of PEs are used because PE 0 is underutilized in that case. Comparing Table 3 to Table 4, the transfer of files to the remote host clearly reduces efficiency at 64 processors and generally increases run-time, but by assigning the file

Table 1 - FindView parallel benchmarks on a single particle.

number of processors	search time (sec.)	origin correct (sec.)	other components (sec.)	total time (sec.)	search speedup	origin correct speedup	overall speedup
1	423.67	64.92	1.66	490.25	1.00	1.00	1.00
2	211.94	33.35	2.10	247.39	2.00	1.95	1.95
4	106.17	18.00	2.74	126.91	3.99	3.61	3.99
8	53.32	10.18	3.46	66.93	7.95	6.38	7.95
16	27.06	6.09	4.04	37.19	15.66	10.66	15.66
32	13.75	3.78	5.10	22.63	30.82	17.17	30.82
64	7.23	2.69	5.31	15.23	58.59	24.13	32.18
128	3.71	2.10	5.99	11.80	114.25	30.91	41.52

transfer task to the manager processor, the effect of this performance loss is reduced.

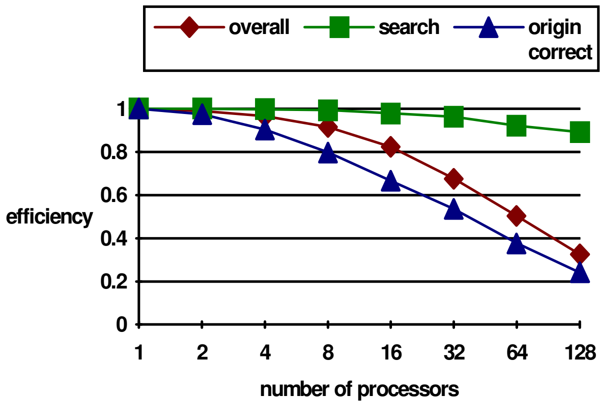

Figure 10 - FindView parallel benchmark efficiency, from data in Table 1.

5: Discussion

The highly parallel computing methods presented herein have enhanced our ability to perform structural studies of icosahedral virus particles. We can now easily process many more particles in our reconstructions than were previously possible, and can do so in much less time. Certain aspects of the reconstruction process have changed dramatically. Whereas the earlier scalar versions of the software required extensive use of an editor, manual submission of a great many sequentially run batch jobs, and time-consuming analysis of the results between runs, the parallel front-end software is quite user friendly, automatically submits batch jobs, and provides for easy analysis of the results. The new handedness flipping methods in the parallel global refinement program now provide a convenient means to perform the once laborious task of generating a self-consistent set of refined particles. In the past, the amount of CPU time and user interaction required limited the number of local refinements that could be run in a reconstruction study. The ability to submit very large queues of local refinements using a convenient user interface has transformed the reconstruction process. Recent reconstructions of the Hong Kong 97 Bacteriophage (materials by Dr. R.L. Duda and Dr. R.W. Hendrix of the University of Pittsburgh, electron microscopy by Dr. N. Cheng) have involved more than 300 particles using an all-inclusive approach where local refinements are performed on 40 candidate orientations of every unrefined particle. These reconstructions have shown that the local refinement has become the step which requires the most overall CPU time (Table 5).

Table 2: Global refinement speedup, 55 particles.
*** - Single-PE result is a projection.**

processors	time, minutes	speedup	efficiency
1	165.02 *	1.0	1.0
2	85.06	1.94	0.97
4	43.35	3.80	0.94
8	22.32	7.08	0.88
16	13.17	12.53	0.78
32	7.26	22.75	0.71
64	4.21	37.23	0.58

Table 3: Local refinement speedup for a 60 run queue with file transfer disabled.

processors	time, minutes	speedup	efficiency
1	30.55	1.0	1.0
2	34.55	0.88	0.44
4	11.65	2.62	0.66
8	5.20	5.87	0.73
16	2.91	10.52	0.66
32	1.88	16.26	0.51
64	1.31	23.32	0.36

Table 4: Local refinement speedup for a 60 run queue with file transfer enabled.

processors	time, minutes	speedup	efficiency
1	31.60	1.0	1.0
2	34.56	0.92	0.46
4	11.65	2.71	0.68
8	5.24	6.03	0.75
16	2.92	10.83	0.68
32	1.96	16.17	0.51
64	1.80	17.61	0.28

The parallel implementation of FindView and Emicograd has allowed simultaneous refinement of over 100 large capsids (HSV, 125 nm diameter) and of over 300 medium-size capsids (HK97, 50 nm diameter). The nominal resolution achieved [6] is of the order of 3.0 nm in both cases. This is only slightly better than was achieved with smaller data sets (3.5 nm), but the signal-to-noise ratios in the resulting maps are significantly improved. Booy et. al. [4] report having identified the individual copies of the small VP26 protein on the hexomer tips, in part due to the signal-to-noise improvements. Figure 11 demonstrates the difference between earlier reconstructions with 36 and 40 particles (top) and more recent reconstructions with 135 and 167 particles (bottom). The left hemisphere is a difference image between chemically depleted HSV capsid shells from which all VP26 is removed and capsid shells onto which the VP26 is rebound. On the upper hemisphere, the VP26 at the hexomer tips appears in the form of non-descript rings, whereas the lower reconstruction reveals the six individual proteins in each ring.

Table 5: CPU time comparison per computation stage, HSV and HK97 viruses. HSV used 120 particles for FindView and local Emicograd, repeating global refinements with successively fewer views, until the final reconstruction was performed on 100 particles. HK97 FindView and Emicograd stages involved 335 particles. Platform A is an Intel iPSC/860 running 32 of 128 PEs; platform B is a VAX 4000/500.

step	plat-form	HSV time, hours (%)	HK97 time, hours (%)
FindView	A	0.9 (5.1)	5.5 (8.3)
local Emicograd	A	8.7 (47.7)	33.5 (50.6)
global Emicograd	A	1.0 (5.5)	0.8 (1.1)
origin refinement	B	2.3 (12.6)	7.2 (10.6)
crude single particle reconstructions	B	0.0 (0.0)	18.2 (27.5)
final reconstruction stages	B	5.3 (29.1)	1.0 (1.5)
TOTAL TIME		18.2	66.2

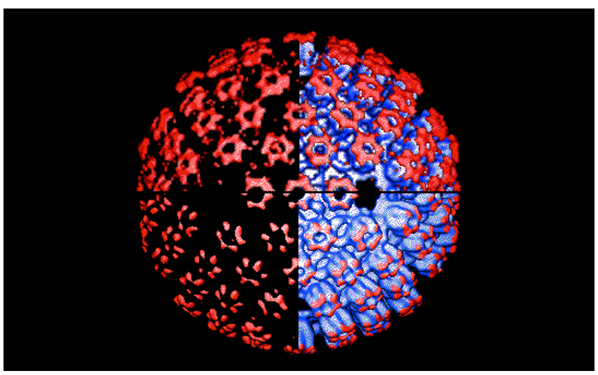

Figure 11: HSV reconstruction with VP26 difference map in red - lower hemisphere at higher resolution.

The accuracy with which the orientation angles can be determined is a well-defined and potentially resolution-limiting factor. The signal-to-noise ratio of the original images most likely constrains the accuracy with which these angles can be determined - by any method, common-line or otherwise - and this limit is likely to be particle-specific (better for particles with strongly defined topographic features; worse for smoothly surfaced, rounder icosahedral) and view-specific. On the other hand, efforts to reduce the noise level (at least, the shot-noise component thereof by increasing electron doses) will degrade the finer structural features. Quite how far the resolution can be pushed in reconstructions based on sets of free-standing particle images is a fascinating challenge of acute current interest. It may be necessary to develop new high performance methods, other than common-lines, to determine the particle orientations within the accuracy required in order to obtain the desired resolution, and certainly comprehensive correction for the contrast transfer function is required. The present study illustrates the essential role of high performance computing and, in particular, some of the advantages of highly parallel distributed memory architectures.

6: Acknowledgment

We gratefully acknowledge Dr. T.S. Baker of Purdue University, who generously provided us with the scalar (VMS) version of the reconstruction codes. Figures 1 and 11 contain contributions from Dr. F.P. Booy, NIH (electron microscopy and computer analysis) and from Mr. W.W. Newcomb and Dr. J.C. Brown, University of Virginia (virology).

7: References

[1] Baker, T.S., Newcomb, W.W., Booy, F.P., Brown, J.C., and Steven, A.C., "Three-dimensional structures of maturable and abortive capsids of equine herpesvirus 1 from cryo-electron microscopy," *J. Virology* **64**, 563-573 (1990).

[2] Baker, T.S., Drak, J., and Bina, M., "Reconstruction of the three-dimensional structure of simian virus 40 and visualization of the chromatin core," *Proc. Natl. Acad. Sci. USA* **85**, 422-426 (1988).

[3] Baker, T.S., Drak, J., and Bina, M., "The capsid of small papova viruses contains 72 pentameric capsomers: direct evidence from cryo-electron microscopy of simian virus 40," *Biophys. J.* **55**, 243-253 (1989).

[4] Booy, F.P., Trus, B.L., Newcomb, W.W., Brown, J.C., Conway, J.F., and Steven, A.C., "Finding a needle in a haystack: detection of a small protein (the 12 FDa VP26) in a large complex (the 200 MDa capsid of herpes simplex virus), *Proc. Natl. Acad. Sci. USA* **91:** 5652-5656 (1994).

[5] Booy, F.P., Newcomb, W.W., Trus, B.L., Brown, J.C., Baker, T.S., and Steven, A.C., "Liquid-crystalline, phage-like packaging of encapsidated DNA in herpes simplex virus," *Cell* **64**, 1007-1015 (1991).

[6] Conway, J.F., Trus, B.L., Booy, F.P., Newcomb, W.W., Brown, J.C., and Steven, A.C., "The effects of radiation damage on the structure of frozen hydrated HSV1 capsids," *J. Struct. Biol.* **111:** 222-233 (1993).

[7] Crowther, R.A., "Procedures for three-dimensional reconstruction of spherical viruses by Fourier synthesis from electron micrographs," *Phil. Trans. Roy. Soc. Lond. B*. **261**, 221-230 (1971).

[8] Crowther, R.A., "The reconstruction of a three-dimensional structure from projections and its application to electron microscopy," *Proc. Roy. Soc. Lond. A*. **317**, 319-340 (1970).

[9] Crowther, R.A., Amos, L.A., Finch, J.T., DeRosier, D.J., and Klug, A., "Three dimensional reconstructions of spherical viruses by Fourier synthesis from electron micrographs," *Nature* **226**, 421-425 (1970).

[10] Fuller, S.D., "The T=4 envelope of Sinbis virus is organized by interactions with a complementary T=3 capsid," *Cell* **48**, 923-934 (1987).

[11] Martino, R.L., Johnson, C.A., Yap, T.K., Suh, E.B., Pfeifer, J.C., Hodoscek, M., and Brooks, B.R., "Highly parallel computing at the National Institutes of Health," *Supercomputing USA/Pacific 91 Conference,* 110-111 (1991).

[12] Nelder, J.A., and Mead, R., "A simplex method for function minimization," *Computer Journal* **7**, 308-313 (1965).

[13] Newcomb, W.W., Trus, B.L., Booy, F.P., Steven, A.C., Wall, J.S., and Brown, J.C., "Structure of herpes simplex virus capsid: molecular composition of the pentons and the triplexes," *J. Mol. Biol.* **232**, 499-511 (1993).

[14] Saxton, W.O., and Baumeister, W., "The correlation averaging of a regularly arranged bacterial cell envelope protein," *J. Microscopy* **127**, part 2, 127-138 (1982).

[15] Trus, B.L., Newcomb, W.W., Booy, F.P., Brown, J.C., and Steven, A.C., "Distinct monoclonal antibodies separately label the hexons or the pentons of herpes simplex virus capsid," *Proc. Natl. Acad. Sci. USA* **89**, 11508-11512 (1992).

[16] Trus, B.L. and Steven, A.C., "Digital image processing of electron micrographs - the PIC System," *Ultramicroscopy* **6**, 383-386 (1981).