Parallel 3 Dimensional Reconstruction

Yongchang JI (yji@cs.ucf.edu)

(Last update October 29, 2004)

CONTENTS

A. INTRODUCTION

B. PROGRAM INPUT

C. EXAMPLE CONTROL FILE

D. PROGRAM EXECUTION

E. PROGRAM NOTES

F. REFERENCES

G. FLOW CHART FOR P3DR

A. INTRODUCTION

P3DR (Parallel 3D Reconstruction) is a package of parallel programs for three-dimensional (3D) reconstruction of electron density in Cryo-TEM. It produces a 3D electron density map from a set of 2D particle projections given the orientation of each projection. The inputs are:

(1) 2D projections of particles, and

(2) the relative orientation of each projection.

The input 2D particle images and 3D density map data are stored in the Purdue *.PIF format.

The determination of the electron density at the points of a regular 3D grid of size N^3 requires $O(N^3)$ arithmetic operations. A technique to embed the pixel frames into larger arrays, called "zero-fill", helps lower the numerical errors in the reconstruction process, but increases the amount of space and the number of arithmetic operations. Now it also support different pixel size and different box size mixed for input 2D images.

P3DR works in Cartesian coordinates, and the calculations are done in the Discrete Fourier Transform (DFT) domain. First, we compute the 2D DFTs of the particle projections, apply CTF correction. Then we compute the values of the 3D DFT of the electron density by interpolate the 2D DFTs into 3D, and finally we apply an inverse 3D DFT to compute the electron density map. The algorithm does not require the object to exhibit any symmetry, but

it takes advantage of the symmetry, if exists. P3DR is normally run after and in combination with POD (Parallel Orientation Determination) or POR (Parallel Orientation Refinement). P3DR has no restrictions as to number of particles or the resolution limit.

A good overview of our algorithms is presented in Marinescu and Ji 2003.

Contact us: yji@cs.ucf.edu, dcm@cs.ucf.edu

B. PROGRAM INPUT

Input parameters control operation of P3DR. They are as follows (Keywords, values, type), Keywords can be more than 4 chars, but we only match the first 4 char and case ignored; Values are the values that user input; Type are the type of each values for this keywords; Keywords that have default value(s) can be omitted if user use the default value(s):

SYMM	symmetry	(I)
RESMAX	softborder_res, out_res	(F, F)
ZERO_FILL	zerofill	(F)
CTFMODE	ctfmode, filter, temperfac, wiener1, wiener2	(I, F, I, F, F)
MAPSIZE	map_magfac, map_dim	(F, I)
OUTMODE	outmode	(I)
ODDEVEN	oddeven	(I)
OUTFILENAME	mapname	(C)
VPRD	viewpipeline	(I)
INPUTPARAMETERFILES		

---

1. SYMM **symmetry** **(I)**

symmetry = 1 - if no symmetry (DEFAULT value)

2~8 - if n-fold about z-axis

532 - if icosahedral

222,32,52,622,72,822 - if dihedral

2. RESMAX **softborder_res, out_res** **(F, F)**

softborder_res specifies the resolution to which the final 3D map will be computed. **softborder_res** can't be set to be a value any SMALLER than twice the size of the IMAGE pixels (the so-called Nyquist limit imposed because the Fourier transform of digitized data only extends to a spatial frequency of 1.0/(2*pixel size). If you believe your IMAGE data are

good enough to achieve higher resolution, you need to rescan the IMAGE at smaller pixel resolution. The typical rule of thumb is to digitize your IMAGE data at a pixel resolution that is 3 or 4 times the expected resolution, in which case only information in the Fourier transform out to a radius of FFT_DIM/3 or FFT_DIM/4 is used. If, for example, you expect your IMAGE RESOLUTION to be about 2.0 nm, then you would normally digitize at 0.7 nm intervals (or smaller).

out_res specifies the effective resolution of the out 3D map. **out_res** must be equal or larger than **softborder_res**. Normally **out_res** is set equal to **softborder_res**, but it is useful to make a soft border between **out_res** and **softborder_res** for the map to make the map looks good. The values beyond the **out_res** will be faded out gradually to 0 at **softborder_res** resolution.

3. ZERO_FILL zerofill (F)

zerofill specifiy the ratio of FFT size(FFT_DIM) to the images size. It must greater or equal to 1.0. For example, if your images size is 221x221, and you want to use FFT size 512x512, then you can set the **zerofill** to be 521/221 = 2.31. The P3DR programs will automatically fit the fft size (FFT_DIM) to a proper value near the value imagesize***zerofill**. The FFT_DIM is not confined to be 2**n in our programs.

Since we support different box size and pixel size and map magnifactor, the **zerofill** will adjust according to them in program automatically based on the **zerofill** you input.

Default value: **zerofill** = 1.0.

4. CTFMODE ctfmode, filter, temperfac, wiener1, wiener2 (I, F, I, F, F)

These parameters are used to affect CTF corrections of the images. The following is a simple description of the nature of the use of these variables. More information will be supplied as we become more adept at using CTF corrections with real data.

ctfmode is a switch that signifies the type of CTF correction to make.

ctfmode = 0 - No CTF correction made to the FFT data.

1 - Full CTF correction (flips phases and modifies amplitudes)

2 - Only flip phases in the FFTs

3 - Only modify amplitudes in the FFTs

4 - Only apply the temperfac in the FFTs

filter = 1 - FFTnew = FFTold / (ABS(CTF) + wiener1*(1-ABS(CTF)) before first peak

= FFTold*ABS(CTF)/(CTF**2+wiener2*(1-ABS(CTF)) after first peak

2 - FFTnew = FFTold * ABS(CTF) /(CTF**2 + wiener1*(1-ABS(CTF))

3 - FFTnew = FFTold * ABS(CTF) /(CTF**2 + wiener1)

4 - FFTnew = FFTold /(ABS(CTF) + wiener1*(1-ABS(CTF))

5 - FFTnew = FFTold /(ABS(CTF) + wiener1)

temperfac is a variable that specifies an estimate of the temperature factor for the IMAGE data. Values in the range of 200 to 500 Angstroms square are typical. However, it is recommended that you NOT enter a value for **temperfac** until you are trying to refine your 3D density map. Hence, **temperfac** should normally be set = 0.0.

wiener1 and **wiener2** are the factor to make sure noise is not magnified too much near the nodes of the ctf. They are associated with the **filter. wiener** are typically set to 0.1. Use values less than 0.1 at your OWN RISK. Values greater than 1.0 are also not recommended!

5. MAPSIZE map_magfac, map_dim (F, I)

map_magfac is used to adjust the size of pixels in the final 3D density map and also (if **map_dim** = 0) will automatically resize the dimensions of the 3D map according to the image size and **map_magfac**. Default value (**map_magfac** = 1.0, **map_dim** = 0) will make a 3D density map with pixels identical to the minimum pixel size of the input IMAGE data and the map will have NCOL**3, dimensions, where NCOL is the maximum dimension size of the input IMAGE data. Set **map_magfac** > 1.0 if you wish to produce a larger than normal reconstruction. That will get smaller pixels size MAP to achieve much nicer 3D rendering effects for making high quality figures for slides or publication. For example, if **map_magfac** = 2.0, the final 3D map will have dimensions = 2*NCOL+1 and pixel size which is half of original pixel size. Under these conditions P3DR will take considerably longer than normal to finish and may 'consume' very large amounts of disk space to store the finely sampled output map.

map_dim can be used to specifically fix the dimensions of the 3D map and thereby override the effects of any change in map dimensions brought about by using a value for **map_magfac** other than 1.0. Though not likely to be used too often, **map_dim** adds flexibility to how you create output 3D map data. **map_dim** may also be used to keep the dimensions of the 3D output map constant, while at the same time allowing the pixel size of the 3D map data to be magnified or demagnified according to the value of **map_magfac**. If **map_dim** = 0, the program will adjust the size of the 3D output map according to the value of **map_magfac** and the input IMAGE dimensions. For example, you may choose to calculate a 3D map with, say **map_magfac** = 0.75, but with **map_dim** set equal to the size the original 3D map. This would give a 3D map in which the reconstructed particle would appear shrunk by 25% but still inside a 3D box of dimensions equal to the dimensions of the original input IMAGE data. Beware of possible "side effects" of **map_dim**, for example if you set it too

small depending on what **map_magfac** is set to, the map may be cut off beyond a radius that lies within the particle!

Default value: **map_magfac** = 1.0, **map_dim** = 0

6. OUTMODE outmode (I)

outmode = 20 - 2-byte methods (default value)

= 40 - 4-byte methods

outmode is the output mode of 3D density map. If **outmode** = 20, it will store the 3D map using 2-byte methods. If **outmode** = 40, it will store the 3D map using 4-byte methods, and it will also double the size of output 3D map compared with the 2-byte method.

7. ODDEVEN oddeven (I)

oddeven = 0 - use all set of images to compute MAP (DEFAULT value)

= 1 - use only odd set of images to compute MAP

= 2 - use only even set of images to compute MAP

When **oddeven** = 1 or = 2 then only EVERY OTHER image in the total input set of images is used for the calculation of the 3D map. Note that "odd" or "even" does NOT mean that images are selected according to particle ID, but instead just means that the particle images are selected according to the SEQUENCE in which they are read from the PARAM file(s). For example, if your FIRST PARAM file only lists particles with *even-numbered* IDs (e.g. 22, 34, 36, 58, 100, 268, 982, 1046, etc.) and **oddeven** is set = 1, then only the images with ID #s 22, 36, 100, 982, etc. will be used. Also, if in this last example, the PARAM file was *instead* the 2nd PARAM file listed in the P3DR control file *AND* the first PARAM file specified an *odd number* of images, then only images with ID #s 34, 58, 268, 1046, etc. would be selected from this PARAM file. This strategy assures that, no matter how many PARAM files there are and no matter what the particle IDs are, the number of images selected will always be equal to or within one of being half of the total number of images listed in the PARAM files.

8. OUTFILENAME mapname (C)

Specify the name of output 3D density map which store in PIF format.

9. VPRD viewpipeline (I)

It is the pipeline for read images. It means how many images are read for each node per time. For example: if **viewpipeline** = 3, and nodes number is 8, then each time node 0 will read 3*8=24 images, and send each node with 3 different images. After each node finish the 3

images, then node 0 will read another 24 images again, and send each node with 3 new images for processing. This will save some reading images time, and in some degree to make all the nodes balance their load. You can specify **viewpipeline** as large as you can, and the programs will automatic adjust according to the image number in files and processor number. Just remember, this will consume more memory.

Default value: **viewpipeline** = 3.

10. INPUTPARAMETERFILES

Enter the names of up to 999 different PARAM files, each one on a new input line. A description of the contents of PARAM files is given: The 1st line of each PARAM file should list the name of the IMAGE file which contains raw image data stored in *.PIF format. The 2nd line in each PARAM file contains the values of: PIXSIZ, UNITS, VOLTAGE, AMPFAC, DELF_MAJ, DELF_MIN, ANG_MAJ, and CS_COEFF. The remaining lines in each PARAM file contain the current set of **ID, THETA, PHI, OMEGA, X, Y, MAG_FAC**, et al. values for specific particles in the raw image file. **MAG_FAC** defaults to be 1.0 if no value or 0.0 is supplied.

ID is the image number--the storage sequences in the IMAGE data.

THETA, PHI, OMEGA are the three orientation angles (in degrees) for each image.

FFT_ORIGX, FFT_ORIGY are the pixel coordinates of the particle center (the point 0.0, 0.0 corresponding to the lower left corner of the boxed particle image).

MAG_FAC specifies the size of each particle image relative to some standard, such as a 3D reconstruction model. Hence, a MAG_FAC = 1.02 means that the particle image is 2% larger than the model. P3DR uses this value to assure that all particle transforms are sampled so the data are all combined at a uniform magnification. With images of frozen-hydrated particles boxed from a single micrograph, MAG_FAC = 1.0 (DEFAULT) is normally assumed for all particles.

Since we can handle different box size and pixel size, so each image file can have different pixel size and box size of image. The final 3D map will use the smallest pixel size and biggest box size if no **mapsize** is specified

C. EXAMPLE P3DR INPUT CONTROL PARAMETERS FILE

```
# Initial with # is a comment line. The format is Keyword and follows by value(s).
# Keywords can be more than 4 chars, but we only match the first 4 char and case ignored.
# 1. SYMM is used for define symmetry of the object (Default value is 1), (I)
SYMM    532
```

```
# 2. RESM is used for define resolution range,  (2F)
#    in P3DR, the output map resolution should be up to the 2nd value,
#    and from the 2nd value down to the 1st value is a soft border region
Resmax    9.8, 10.
# 3. ZERO specifies FFT size enlarge ratio, FFT_DIM = box_size * Zero_fill, (F)
#    Default value is 1
Zero_fill    1.15
# 4. CTFM specifies CTF correction parameters,
#     CTFMODE, Filter, TEMPFAC, WIENER, WIENER2 (2I,3F)
Ctfmode    1, 4, 0.0, 0.05, 0.1
# 5. MAPS specifies the mapmagfactor and the dimension of output map in pixel, (F,I)
#     Default value is 1.0 and the pixel size of input image.
Mapsize    1.0, 257
# 6. OUTM specifies the output mode, (I)
#    20: 2-byte density map; 40: 4-byte density map. Default value is 20
Outmode    20
# 7. OUTF specifies the output map file name, (C)
Outfilename    indbis-e2-n318q.map
# 8. ODDE specifies to generate density map with odd, even, all images, (I)
#     odd = 1, even =2, all = 0. Default value is 0.
Oddeven    0
# 9. VPRD specifies the pipeline size, (I)     Default value is 3.
Vprd    10
# 10. INPU is the keyword for input orient files. (C)
#     only orientation file names are allowed, no space line, no comment line or others.
Inputparameterfiles
8824.sel_003
8825.sel_003
8826.sel_003
8827.sel_003
```

For example: a simple control file:

```
SYMM  532
Resmax  7.37, 7.37
Zero_fill  2.0
Ctfmode  1,1, 0.0, 0.1, 0.1
Outfilename  rec.pif
```

Inputparameterfiles

8824.sel_003

8825.sel_003

D. PROGRAM EXECUTION

Normal operation of P3DR leads to a 3D density map that contains the entire reconstructed particle. All parameters with default values can be omitted in the input control parameter file, such as SYMMETRY, ZERO_FILL, MAPSIZE, OUTMODE, ODDEVEN, VPRD. P3DR programs will automatically assign them with default values if these keywords are omitted.

For example:

Execute P3DR with 40 nodes using MPI, the command will be:

% mpirun –nolocal -machinefile mach -np 40 P3DR < Ctrl.p3d > Output

or

% mpirun -np 40 P3DR < Ctrl.p3d > Output

Ctrl.p3d is the script of the input control parameters file, and the **Output** file will be all the output debug information generated by P3DR standard output. We use I/O redirect methods. The **mach** file is used for specifying the hosts of the cluster that you want to use. Please modify it to indicate the host names of your cluster. If use second command, it will use the default machine configuration under MPICH directory.

For details of running MPI and specifying hosts in the host file, see user guide of MPICH at http://www-unix.mcs.anl.gov/mpi/mpich/.

E. PROGRAM NOTES

The directories of P3DR source code are:

```
|-- Commpk        !common routine directory
|-- P3DRsrc1      !P3DR algorithm 1(whole fft map, less communication)
|-- P3DRsrc3dm    !P3DR algorithm 1 with double precision memory
|-- P3DRsrc3      !P3DR algorithm 3(less memory, more communication)
|-- P3DRsrc3dm    !P3DR algorithm 3 with double precision memory
|-- Vfftpk        !FFT library codes directory
|-- include       !include files directory
|-- Makefile
```

We have 4 P3DR programs. P3DRsrc1 and P3DRsrc1dm using algorithm 1, which stores half of the 3D DFT space on all the nodes, and have less communication (only exchange data at the end of interpolation of all images). P3DRsrc3 and P3DRsrc3dm using algorithm 3, which divides the whole 3D DFT space by number of nodes into slab, and each node only store a slab, so it will consume less memory space, but it will have more communication (exchange data at the end of interpolation of each image). Also, P3DRsrc1 and P3DRsrc3 use single precision during interpolation. P3DRsrc1dm and P3DRsrc3dm use double precision during interpolation, and this will increase the precision, but will use more memory during interpolation (about 8 times).

1. CTF Factor Formula:

CTF = -[sqrt(1 - amp_fac**2)*sin(chi) + amp_fac*cos(chi)]

CTF = CTF * Es * Et

TMP = 10 / fft_dim / pixel_size

D_STAR = Sqrt(x**2 + y**2) * TMP

T = EXP ((TEMPFAC/400.0) * (D_STAR**2))

enlargefac is the scale factor of all the images to make them comparable.

CTF_factor = filter 4 if before first peak or filter 2 if after first peak Filter1

or = enlargefac*T*ABS(CTF) /(CTF**2 + wiener1*(1-ABS(CTF)) Filter2

or = enlargefac*T*ABS(CTF) /(CTF**2 + wiener1) Filter3

or = enlargefac*T /(ABS(CTF) + wiener1*(1-ABS(CTF)) Filter4

or = enlargefac*T /(ABS(CTF) + wiener1) Filter5

If in the soft border region: (detail see ctf_para.f)

ds_max = (fft_dim*pixel_size) / out_res

ds_min = (fft_dim*pixelsize) / softborder_res

t_max = exp((TEMPFAC/400.0) * (ds_min*tmp)**2)

delsq = -(ds_max-ds_min)**2 / log(0.000001/t_max)

CTF_factor = CTF_factor * t_max / T * exp(-(D_STAR - ds_min)**2/delsq)

Else if beyond the soft border resolution:

CTF_factor = 0

2. In Fourier Domain: annulus = FFT_DIM * pixel_size / resolution

annulus <= FFT_DIM / 2

so: resolution >= 2 * pixel_size

3. Rotmat matrix: ROT_MAT = ROT_1 * ROT_2 * ROT_3

$$rot\_1 = \begin{vmatrix} \cos(theta) & 0 & \sin(theta) \\ 0 & 1 & 0 \\ -\sin(theta) & 0 & \cos(theta) \end{vmatrix}$$

$$rot\_2 = \begin{vmatrix} \cos(phi) & -\sin(phi) & 0 \\ \sin(phi) & \cos(phi) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

$$rot\_3 = \begin{vmatrix} \cos(omega) & -\sin(omega) & 0 \\ \sin(omega) & \cos(omega) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

4. FFT transforms used (N is even):

Analysis: $F(h, k)=\frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x, y)\exp(-2\pi i(xh+yk)/N)$

Synthesis: $f(x, y)=\frac{1}{N}\sum_{h=0}^{N-1}\sum_{k=0}^{N-1} F(h, k)\exp(+2\pi i(xh+yk)/N)$

Translation: $g(x, y) = f(x+dx, y+dy) \iff G(h, k) = F(h, k)\exp(+2\pi i(h*dx+k*dy)/N)$

Periodicity: $F(h, k) = F(h+N, k) = F(h, k+N) = F(h+N, k+N)$

Conjugate: $F^*(-h, -k) = F(h, k)$

Scaling: $f(ax, by) \iff F(h/a, k/b) / |ab|$

5. Interpolation for each 2D DFT image P:

a. map_magfac: (new 3D DFT **F′** after map_magfac)

$F'(h, k, l) = F(h*map\_magfac, k*map\_magfac, l*map\_magfac) * (map\_magfac**3)$

$= F(h', k', l') * (map\_magfac**3)$

b. piecewise constant interpolation:

$F(h', k', l') = P(u', v', \Delta w) = P(u', v', 0) = P(u', v')$

c. img_magfac:

$P(u'', v'') = P(u'/img\_magfac, u'/img\_magfac) / (img\_magfac**2)$

d. 2D interpolation:

$$P(u'', v'') = P(u+\Delta u, v+\Delta v) = \Delta u*\Delta v*P(u, v) + (1-\Delta u)*\Delta v*P(u+1, v) + \Delta u*(1-\Delta v)*P(u, v+1) + (1-\Delta u)*(1-\Delta v)*P(u+1, v+1)$$

e. weight method:

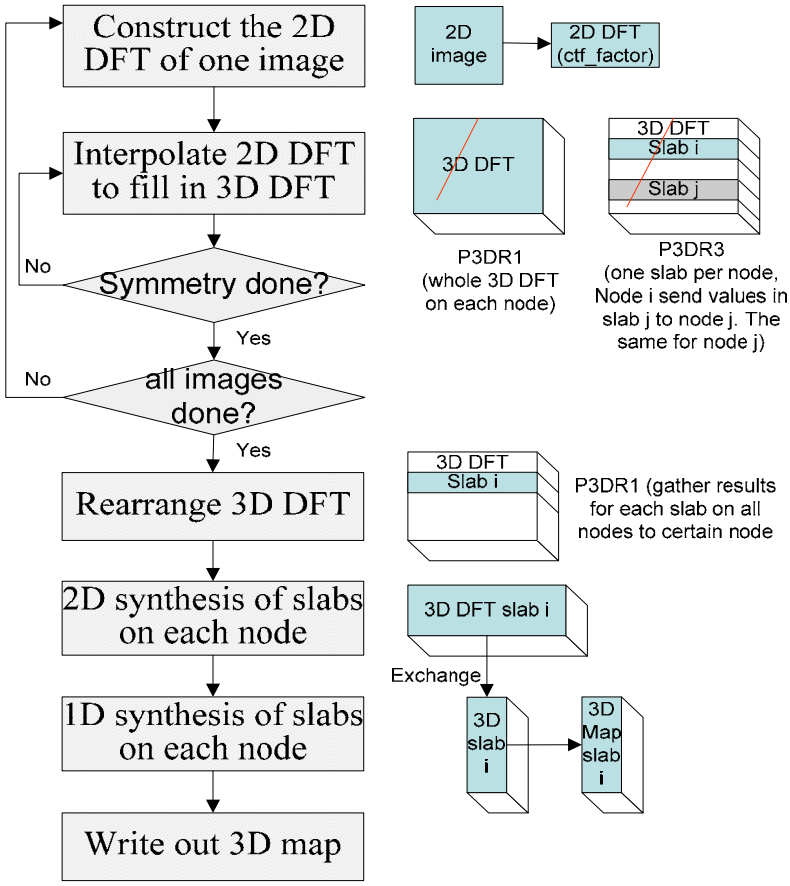
$$W(h, k, l) = (1-\Delta w) * |CTF(u'', v'')| = (1-\Delta w) * (\Delta u*\Delta v*|CTF(u, v)| + (1-\Delta u)*\Delta v*|CTF(u+1, v)| + \Delta u*(1-\Delta v)*|CTF(u, v+1)| + (1-\Delta u)*(1-\Delta v)*|CTF(u+1, u+1)|)$$

f: $F_{final}(h, k, l) = \Sigma (F_i(h, k, l)*W_i(h, k, l)) / \Sigma W_i(h, k, l)$

F. REFERENCES

- Dan C. Marinescu and Yongchang Ji**, A Computational Framework for the 3D Structure Determination of Viruses with Unknown Symmetry**, *Journal of Parallel and Distribute Computing,* Vol. 63, pp. 738--758, 2003
- Robert E. Lynch, Yongchang Ji, Dan C. Marinescu and Hong Lin**, A Parallel 3D Piecewise Constant Reconstruction Algorithm for Asymmetric Virus Structures**, *The International Conference on Computational Science 2003*, Saint Petersburg, Russia, *Lecture Notes in Computer Science*, Springer-Verlag Heidelberg, vol. 2657, pp. 437-446, 2003
- Dan C. Marinescu, Yongchang Ji, Robert E. Lynch, **Space-Time Tradeoffs for Parallel 3D Reconstruction Algorithms for Virus Structure Determination**, *CP&E*, Vol. 13, pp. 1083--1106, 2001.
- Dan C. Marinescu, Yongchang Ji, Robert E. Lynch, **A Comparison of Three Algorithms for Parallel 3D Reconstruction**. *High-Performance Computing and Networking: 9th International Conference, HPCN Europe 2001, Proceedings* (B. Hertzberger, A. Hoekstra, R. Williams, Eds.), *Lecture Notes in Computer Science*, Springer-Verlag Heidelberg, vol. 2110, pp. 173-182, 2001
- Robert E. Lynch, Dan C. Marinescu, Hong Lin, **Parallel algorithms for 3D reconstruction of asymmetric objects from electron micrographs**, *Proc. IPPS/SPDP'99* San Juan, Puerto Rico, April 12-16, 1999.
- T. S. Baker, N. H. Olson, and S. D. Fuller, **Adding the third dimension to virus life cycles: Three-dimensional reconstruction of icosahedral viruses from cryo-electron micrographs**. *Microbiol. Mol. Biol. Reviews* **63:862-922**, 1999
- R. A. Crowther, L. A. Amos, J. T. Finch, D. J. DeRosier and A. Klug, **Three dimensional reconstructions of spherical viruses by Fourier synthesis from electron micrographs**. *Nature*, **226:421-425,** 1970

G. FLOW CHART FOR P3DR PROGRAMS



P3DR1 !! main program

|--- call usage !! print out usage

|--- call mpi_init !! mpi initialization

|--- call info !! read input control files

|--- call bcast_parameters !! broadcast parameters

|--- call intlz_params !! initialize parameters

|--- call intlz_arrays !! initialize arrays

|--- For each input orientation files do

|--- call pif_open !! open image file

|--- call read_1_pif !! read first image

|--- call mpi_bcast !! broadcast parameter

|--- call ctf_para !! initialize CTF factor for this image file

```
        |--- call cmpt_intrps          !! interpolate 3D grid from 2D DFT
        |--- call pif_close            !! close image file
    |--- call exch_intp                !! exchange 3D DFT
    |--- call fftsynth_1_m_slab        !! 2D Fourier synthesis
    |--- call exchange_2_slab          !! exchange data for 1D fft synthesis
    |--- call rearrange_2_slab         !! rearrange data for 1D fft synthesis
    |--- call fftsynth_2_m_slab        !! 1D Fourier synthesis
    |--- call density_clear            !! fill in density map and purify the corner
    |--- call output_density           !! write out the density map
    |--- call mpi_finalize             !! finalize mpi

INFO                                 !! read the input control parameter file
   |--- call symmcode                 !! calculate the symmetry
   |--- call pfile_info               !! read head of each image files

INTLZ_PARAMS                         !! initialize internal parameters
    |--- call trans_length_essl       !! calculate suitable FFT_DIM

INTLZ_ARRAYS                         !! initialize internal arrays
    |--- call set_indices              !! calculate frst_lst_hk
    |--- call set_indices              !! calculate frst_lst_y
    |--- call set_indices              !! calculate frst_lst_u
    |--- call set_indices              !! calculate frst_lst_z

READ_1_PIF                           !! read the first image to get the std deviation.
    |--- call pif_read_dh              !! read the data header of the image file.
    |--- call rpifimag_f               !! read the first image in the image file.

RPIFIMAG_F                           !! read image in the image file
    |--- call pif_read_byte_image      !! read 1_byte image in the image file.
    |--- call pif_read_short_image     !! read 2_byte short image in the image file.
    |--- call pif_read_boxi4           !! read 4_byte image in the image file.
```

```
CTF_PARA                          !! compute the ctf factor for this image file
    |--- call ctf_astig           !! compute defocus value for astigmatic data
    |--- call ctf                 !! compute CTF at particular spatial frequency
            |--- ctf_es           !! exponential decay from spatial coherence
            |--- ctf_et           !! exponential decay from temporal coherence
    |--- call ctf_firstpeak       !! computer the first peak radius

CMPT_INTRPS                       !! interpolate 3D grid from 2D DFT
    |--- call readorient          !! read orientation file of the image file
    |--- call mpi_bcast           !! broadcast the orientation file to all nodes
    |--- for each viewpipeline do
        |--- call set_indices     !! calculate frst_lst_view
        |--- call read_files      !! read and scatter the images for this pipeline
        |--- for each images on the node do
            |--- call getorient   !! get the orientation of this image
            |--- call ico_em4imr  !! convert the orientation into unit vector, 532
            |--- call ico_em4imr_vector  !! convert the orientation into unit vector, 532
            |--- call setrotmat   !! form matrix, map 2D pixel plane with 3D
            |--- call eight_symmetry     !! fill in 8 equivalent views for symm 8
            |--- call dihedral_symmetry  !! fill in equivalent views for dihedral symm
            |--- call fftanaly    !! 2D DFT of the pixel image
            |---                  !! apply ctf factor to the 2D DFT
            |--- for each symmetry do
                |--- call setrotmat      !! form matrix, map 2D pixel plane with 3D
                |--- call rottrans       !! set matrix, determine intersection of 2D and 3D
                |--- call rottrans       !! set matrix, determine intersection of 2D and 3D
                |--- call matrixentries_slab !! Interpolation by piecewise constant method
                                         !! fill in 3D grid from 2D DFT

READ_FILES                        !! read the image from image file.
    |--- call pif_read_dh         !! read the data header of the image file.
    |--- call rpifimag_f          !! read one certain image in the image file.
```

```
FFTSYNTH_1_M_SLAB                    !! 2D fft synthesis
    |--- call fftsynth_1_slab        !! 2D fft synthesis
            |--- call vrffti         !! initialize ffts parameter
            |--- for each section do
                    |--- call fftsynth_1_slab    !! 2D fft synthesis a section
                            |--- call vrfftb     !! backward fft transform

REARRANGE_2_SLAB                    !! rearrange data for 1D fft synthesis
  |--- call move_data_2_slab         !! rearrange data for 1D fft synthesis

OUTPUT_DENSITY                      !! write out the density map
    |--- call pif_init_head          !! initialize the pif file head
    |--- call pif_open               !! open the pif file for write
    |--- call pif_write_gh           !! write the global header of the pif file
    |--- call pif_write_dh           !! write the data header of the pif file
    |--- for each section do
          |--- call pif_write_seci4  !! write out 1 section to the pif file in 4-byte
          |--- call pif_write_seci2  !! write out 1 section to the pif file in 2-byte
    |--- call pif_close              !! close image file
```