Parallel Orientation Refinement

Yongchang JI (yji@cs.ucf.edu)

(Last update October 29, 2004)

CONTENTS

A. INTRODUCTION

Several methods including the method of "common lines" can be used for orientation refinement. But our parallel orientation refinement algorithm is radically different. It does not make any assumption about the symmetry of the object; it constructs cuts from Fourier Space of the electron density map, and compares the projections with these cuts. This algorithm is described in detail in Y. Ji, D. C. Marinescu, W. Zhang, and T. S. Baker paper.

Parallel Orientation Refinement (POR) program is based upon a multi-resolution search and a sliding window mechanism. The program POR based on this algorithm use as input:

(1) 2D projections of particle image files,

(2) the relative orientation of each projection, and

(3) a model generated by P3DR using these projections and orientations.

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

POR is used in conjunction with our parallel 3D reconstruction algorithm (P3DR) in Cartesian coordinates for objects without symmetry. A technique embedding the pixel frames into larger arrays, "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.

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

B. PROGRAM INPUT

Input parameters control operation of POR. They are as follows (Keywords, values, format), 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):

RESMAX	res_hi, res_low	(F, F)
ZERO_FILL	zerofill	(F)
CTFMODE	ctfmode, filter, temperfac, wiener1, wiener2	(I, F, I, F, F)
IN3DFILENAME	modelname	(C)
VPRD	viewpipeline	(I)
Dangle	angle_res, angle_range, method, weight	(F, I, I, F)
Dcenter	center_res, center_range	(F, I)
INPUTPARAMETERFILES		

Input parameters control the operation of POR as follows:

1. RESMAX res_hi, res_low (F, F)

res_hi and **res_low** specify the resolution range to which the cuts of 3D model compared with the projections, and the regions beyond this range will not be compared. **res_hi** is the upper radius resolution of this range, and it can't be set to 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). **res_low** is the lower radius resolution of this range, and it must be equal or larger than **res_hi**. Normally **res_low** can be set to maximum value which means from the canter where radius nearly 0.

2. 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 POR 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.

3. CTFMODE ctfmode, filter, temperfac, wiener1, wiener2 (I, I, F, 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 refining your 3D density map. Usually the model are generated with **temperfac** = 0.0 in P3DR, and if you specify in POR **temperfac** > 0.0, then the POR will apply **temperfac** to the model.

WARNING: you MUST generate model for POR with **temperfac** = 0 in P3DR.

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!

4. IN3DFILENAME modelname (C)

Specify the name of input 3D density map which store in PIF format to be used as a model.

5. 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

6. Dangle **angle_res, angle_range, method, weight** **(F, I, I, F)**

angle_res is the refine resolution for angle in degree. **angle_range** specifies the step number in searching. For example: if **angle_res** = 0.1 and **angle_range** = 4, which means that we will compare a projection of orientation (θ, φ, ω) with the cuts of orientation ($\theta \pm i*0.1$, $\varphi \pm i*0.1$, $\omega \pm i*0.1$), $0 \le i \le 4$, so totally, we will compare (4+1)**3 = 125 cuts.

method is a switch that specifies the type of comparison method of the projections and cuts. We suggest to use distance method (**method** = 1).

method = 0 - Use CC method with real part & imaginary part together as two real

= 1 - Use distance method

= 2 - Use amplitude method

= 3 - Use phase method

weight is used to specifies how to emphasis the contribution at different resolution

weight <= 10.0 : $F_{i,j} = F_{i,j} * (radius_{i,j}$ **weight), which means enlarge the values at certain radius by radius**weight

weight > 10.0 : $F_{i,j} = F_{i,j} * \exp(weight/400 * (D\_STAR_{i,j}**2))$, which means exponent enlarge the value. This method is similar to **Temperfac**.

7. Dcenter **center_res, center_range** **(F, I)**

center_res is the refine resolution for center in pixel. **center_range** specifies the step number in searching. For example: if **center_res** = 0.1 and **center_range** = 4, which means that we will compare the cut (got by the angle refinement) with the projection (center (x, y)) at center ($x \pm i*0.1$, $y \pm i*0.1$), $0 \le i \le 4$, so totally, there will be (4+1)**2 = 25 center translations of projection.

Default value: **center_res = angle_res, , center_range = angle_range.**

8. 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 byte-packed 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 1.0 if no value or a 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. POR 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 3D model should have the smallest pixel size and biggest box size. You can use P3DR with the **mapsize** keyword to get 3D model map with different pixel size and box size as you want.

C. EXAMPLE POR 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. RESM is used for define comparison range of resolution,  (2F)
Resmax    9.8, 200.
# 2. ZERO specifies FFT size enlarge ratio, FFT_DIM = particlesize * 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. IN3D specifies the input 3D model  name, (C)
```

In3dfilename indbis-e2-n318q.map
6. VPRD specifies the pipeline size, (I) Default value is 3.
Vprd 10
7. DANG specifies the angle resolution, search range, compare method, weight, (F, I, I, F)
Ex: angle resoltion 0.1 degree; search range {-4, 4}; compare method 1, weight 500.
Dangle 0.1, 4, 1, 500
8. DCEN specifies the center resolution, search range, (F, I)
default value is the same as DANG
Ex: the center resoltion 0.1 pixel; search range {-4, 4}.
Dcenter 0.1, 4
9. 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:

Resmax 8.3, 30

Zero_fill 2.0

Ctfmode 1, 1, 190.0, 0.1, 0.1

In3dfilename rec.pif

Vprd 30

Dangle 0.1, 5, 1, 1.3

Dcenter 0.1, 5

Inputparameterfiles

8824.sel_003
8825.sel_003

D. PROGRAM EXECUTION

Normal operation of POR leads to a set of new orientation files corresponding to a set of input PARAM files. The new PARAM data files are named by tagging a "_00?" after the input filename. For example, if the name of your input PARAM file was "MYDATA.DAT", then the name of the output PARAM file would be "MYDATA.DAT_001". Normally you would use this file as the input file for the next run of POR, in which case the subsequent

PARAM file would automatically be named "MYDATA.DAT_002", and in 3rd run of POR using "MYDATA.DAT_002" as the input file, the output file would automatically be named "MYDATA.DAT_003", and so forth. So, if you are using several PARAM files to POR, then use some rational naming system so the output new orientation files will be clearly distinct from the input files.

All parameters with default values can be omitted in the input control parameter file, such as ZERO_FILL, VPRD, and DCENTER. POR programs will automatically assign them with default values if these keywords are omitted.

Normally POR is run several times with different angle resolution and center resolution.

For example:

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

% mpirun -nolocal -machinefile mach -np 40 POR < Ctrl.p3d1 > Output1

% mpirun -nolocal -machinefile mach -np 40 POR < Ctrl.p3d2 > Output2

% mpirun -nolocal -machinefile mach -np 40 POR < Ctrl.p3d3 > Output3

or

% mpirun -np 40 POR < Ctrl.p3d1 > Output1

% mpirun -np 40 POR < Ctrl.p3d2 > Output2

% mpirun -np 40 POR < Ctrl.p3d3 > Output3

Ctrl.p3d? are the script of the input control parameters file, and the **Output?** files will be all the output debug information generated by POR standard output in each run. 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 POR source code are:

```
|-- Commpk        !common routine directory
|-- PORsrc1dm     !POR source codes directory
|-- Vfftpk        !FFT library codes directory
|-- include       !include files directory
|-- Makefile
```

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 the images and cuts 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

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

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

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

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: $f(ax, by) <=> F(h/a, k/b) / |ab|$

5. Interpolation 3D DFT into 2D DFT cut P:

a. img_magfac:

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

b. 3D interpolation:

$$P(u', v') = F(h', k', l') = F(h+\Delta h, k+\Delta k, l+\Delta l)$$
$$= \Delta h*\Delta k*\Delta l*F(h, k, l) + (1-\Delta h)*\Delta k*\Delta l*F(h+1, k, l) +$$
$$\Delta h*(1-\Delta k)*\Delta l*F(h, k+1, l) + \Delta h*\Delta k*(1-\Delta l)*F(h, k, l+1) +$$
$$(1-\Delta h)*(1-\Delta k)*\Delta l*F(h+1, k+1, l) + \Delta h*(1-\Delta k)*(1-\Delta l)*F(h, k+1, l+1) +$$
$$(1-\Delta h)*\Delta k*(1-\Delta l)*F(h+1, k, l+1) + (1-\Delta h)*(1-\Delta k)*(1-\Delta l)*F(h+1, k+1, l+1)$$

6. radius (resolution) weighting method

Fp: DFT of projection; Fc: DFT of cut;

a. compare methods

(1) - CC method (here we consider one complex value to be two real values)

$$CC = \Sigma((Fp - \overline{Fp})(Fc - \overline{Fc})) / sqrt(\Sigma((Fp - \overline{Fp})**2) * \Sigma((Fc - \overline{Fc})**2))$$

(2) - distance method

$$Distance = \Sigma\ sqrt((real(Fp) - real(Fc))**2 + (aimag(Fp) - aimag(Fc))**2) / n$$

(3) - amplitude method

$$Amp = \Sigma\ |sqrt(real(Fp)**2 + aimag(Fp)**2) - sqrt(real(Fc)**2 + aimag(Fc)**2))| / n$$

(4) - phase method

$$Phase = \Sigma\ |atan(aimag(Fp) / |real(Fp)|) - atan(aimag(Fc) / |real(Fc)|)\ | / n$$

b. weight method (only apply to compare method 1,2, and 3; do not apply to phase method)

$F_{i,j} = F_{i,j} * radius_{i,j}**weight$ if weight <= 10.0

$F_{i,j} = F_{i,j} * \exp(weight/400 * (D\_STAR_{i,j}**2))$ if weight > 10.0

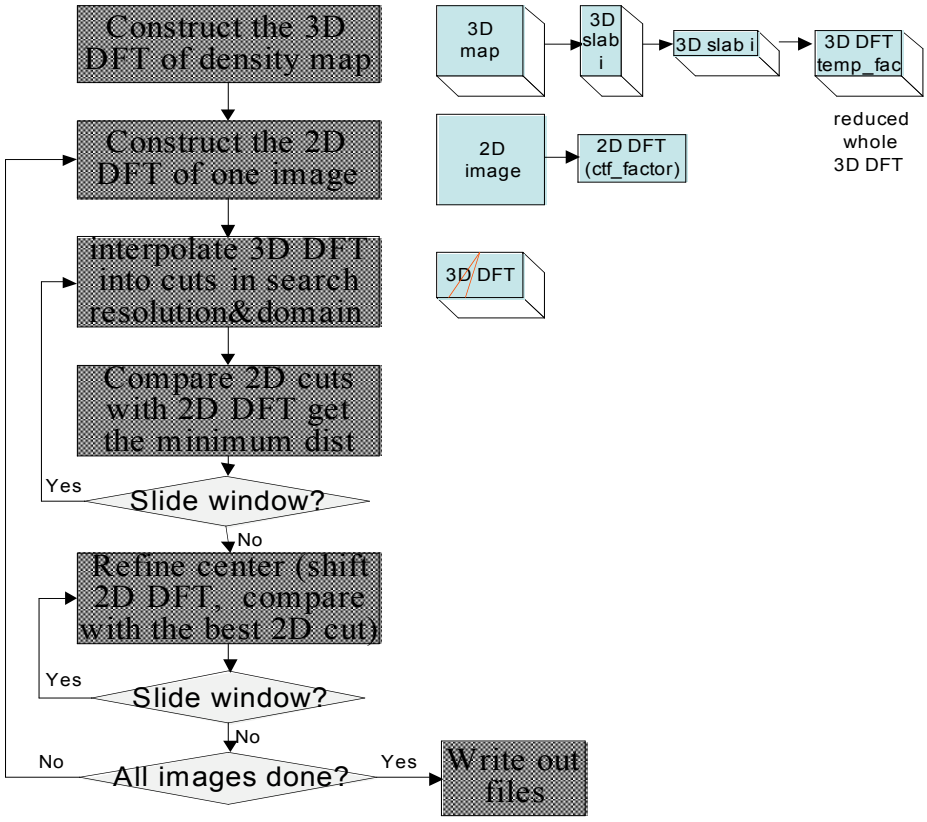
F. REFERENCES

- Yongchang Ji, Dan C. Marinescu, Wei Zhang, and Timothy S. Baker, **Orientation refinement of virus structures with unknown symmetry**, *CD ROM Proc. 17th Ann.*

Int'l Parallel & Distrib. Process. Symp. Nice, France, IEEE Press. Los Alamitos, Ca, ISBN 0-7695-1926-1, 2003

- 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
- 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 POR PROGRAM



POR !! 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
    |--- call read_map                       !! read 3D model map and scatter slab to each node
    |--- call vrffti                         !! initialize fft parameter
    |--- call fft_2dfft                      !! apply 2D DFT onto the 2-slab of 3D model
    |--- call arrange_3d_1                   !! rearrange the slab for exchange
    |--- call exch_3d_1                      !! exchange slabs among nodes for 1D DFT
    |--- call cfft_1d                        !! 1D DFT along z axis with complex number
    |--- call maptempfac                     !! apply temper factor to the 3D Fourier Space
    |--- call gather3d                       !! gather slabs on each node to form a whole 3D
    |--- 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 disef_para                !! initialize radius weight factor for the image file
         |--- call ctf_para                  !! initialize CTF factor for this image file
         |--- call cmpt_ort                  !! refine orientation for this image file
         |--- call pif_close                 !! close this image file
    |--- 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_MAP                               !! read and scatter slab of 3D model map
    |--- call pif_open                   !! open the pif file for read
    |--- call pif_read_gh                !! read the global header of the pif file
    |--- call pif_read_dh                !! read the data header of the pif file
    |--- for each slab do
        |--- call pif_read_mapi4         !! read 1 slab of pif file in 4-byte
        |--- call pif_read_mapi2         !! read 1 slab of pif file in 2-byte
    |--- call pif_close                  !! close image file

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_ORT                               !! compute orientation of a image file
    |--- call readorient                 !! read orientation file of the image file
    |--- call increment_fname            !! write out new orientation file with head
    |--- 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 fftanaly              !! 2D DFT of the pixel image
          |---                            !! apply ctf factor to the 2D DFT
          |--- call setrotmat             !! form matrix, map 2D pixel plane with 3D
          |--- call interpl_3d            !! interpolate 3D to 2D at certain orientation
          |--- call imgcomport    !! compute distance of 2D cut and 2D DFT for orientation
          |--- call slide_angles          !! slide the angle window
          |--- call setrotmat             !! form matrix, map 2D pixel plane with 3D
          |--- call interpl_3d            !! interpolate 3D to 2D at certain orientation
          |--- call recenter              !! transform the 2D DFT with certain center
          |--- call imgcompcet       !! compute distance of 2D cut and 2D DFT for center
          |--- call slide_angles          !! slide the center window
     |--- call set_indices                !! calculate frst_lst_view
     |--- call mpi_gatherv                !! gather all orientations of this image file

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.
```