OOR.DOC

(last update: August 18, 2004)

CONTENTS

A. INTRODUCTION

OOR determines orientations (THETA,PHI,OMEGA= θ,ϕ,ω) and origin (X,Y) values for particle images. This, of course, is critically important information for producing a 3D reconstructed density map from many, independent images. OOR uses cross-correlation procedures to compare unfiltered or filtered particle images against a database of reference projection views produced from a 3D density map. The map might be a 3D reconstruction or model, for example, computed from an atomic, X-ray crystallographic structure. The last major enhancements to the program included use of compressed data and the sliding window approach to refinement (released in May 1998) and most recently the use of contrast transfer function (CTF) corrections as (August 1999).

Use of CTF to assist comparison of image and projection data

The user has three different options for imposing the microscope CTF on the model projections or raw images with the goal of getting better refinement of orientation and origin parameters. The user, of course, has the option (set CTFMODE = 0) of NOT using the CTF correction software.

The four CTF options are:

CTFMODE = 0 Skip CTF correction.

CTFMODE = 1 Multiply the projections of the 3D model by the CTF.

CTFMODE = 2 Multiply the raw image data by the CTF.

CTFMODE = 3 Multiply the raw image data by the inverse CTF.

CTFMODE = 1 is designed for the situation in which you have imposed a CTF correction in the 3D map in the program EM3DR. In theory, OOR will work best if the model projections and raw images are compared 'apples to apples'. That is to say, by imposing a CTF onto the model projections, you are trying to recreate what noise-free data (the model) would look like if the projection pictures had been recorded in the TEM like the raw particle images. **IMPORTANT WARNING:** In OOR's present state use of this option is ONLY CORRECT for image date from ONE MICROGRAPH. If you are refining image data from multiple micrographs (with different CTFs), then the use of CTFMODE = 1 ***may*** lead to errors in refinement.

CTFMODE = 2 employs the strategy used by Steve Fuller's group. Here, raw images are multiplied by the microscope CTF and this has the effect of weighting the image data to emphasize the most significant (i.e. highest S/N) and de-emphasize the least significant (low S/N) portions of each image. This down weights data near the nodes of the CTF, precisely where the noise is highest in the image transforms. So, option 2 weights the data to emphasize the reliable portions of the image while down weighting the most unreliable portions. This, in turn, is designed to help the PFT routine perform a more reliable refinement of orientation and origin parameters.

CTFMODE = 3 causes each raw image to be multiplied by the INVERSE CTF as a means to create image data that more properly compares with the UNMODIFED model projections.

Only careful testing will prove which of the above methods works best for you or your particular project!

Use of COMPRESSED DATA = faster computations

The basic premise is that, because we normally over sample our image data when we digitize it so the pixel size is 3-4 times as fine as the expected resolution, then we can realize significant computational effort merely by performing operations on compressed data. Hence, the program switch, BIN_FACTOR, was added to signal OOR to perform its computations on uncompressed data (BIN_FACTOR=1) as before or much faster on compressed data (BIN_FACTOR=2). Recent improvements in memory management have even provided modest program speed-up when OOR is run with BIN_FACTOR=1. If BIN_FACTOR=2 (the only other option), then the 3D map is compressed ~8-fold and hence requires a corresponding decrease in memory requirements at run time. Because the "volume" of the compressed 3D map is ~8-fold smaller than that of the original map, 2D projections of the map are computed about 8 times faster. The 'raw' image data are also compressed as they are read in, which speeds up by ~4-fold all manipulations of images and PFT data. The overall improvement in program performance is about 6-8 fold when BIN_FACTOR=2.

SLIDING WINDOW REFINE method

The impetus for developing this approach was a) to reuse rather than recalculate redundant projection data where possible, and b) to develop a routine that "easily" parallelizes (time will tell). The beauty of this method is that the computations get more efficient as the number of images increases, and is therefore well-suited for studying thousands or tens of thousands of images.

B. GENERAL DESCRIPTION OF PROGRAM

OOR performs the REFINEMENT over limited search "windows" (MODE=3 only - program will override user input to insure MODE=3 is enforced.). For this mode, the orientation search for each image is restricted to a 7 x 7 grid of θ,ϕ space, with the central grid point in the search window closest to the current best estimate of θ and ϕ for the image. For MODE=3 (REFINE) operation, the orientation search makes use of the most recent particle X,Y origin information. In addition, each particle origin is refined after the θ,ϕ,ω angles are determined for each particle. Here, a real-space cross-correlation of each particle image is made against the corresponding projected view (at angle θ,ϕ,ω) of the current 3D model.

The success of OOR in part results because comparisons made between the 'raw' particle images and model projections utilize all the available data, and, in part because the projection data generally have a high signal-to-noise (S/N) ratio. In addition, the OOR procedure can be sensitized by selecting just a portion of each image for comparison with the model data. For example, only information within defined radii in real or reciprocal space can be utilized in the calculations.

OOR computes two types of data from the 3D model:

1. Projections (**PRJ**s) and

2. Polar Fourier transforms (**PFT**s) of the projections.

These are used as reference data to determine θ,ϕ,ω,X,Y values for each 'raw' particle image. The high S/N ratios of the model 3D density map and the projections computed from it is what makes the orientation and origin searches work well. However, although this method is quite robust in analyzing images of many unstained, frozen-hydrated specimens, success requires the use of model data that include *some* elements representative of the correct structure (see *e.g.* Baker and Cheng [1996] *J. Struct. Biol.* **116:120-130**).

C. OOR PROCEDURE

OOR is a hybrid PFT and cross correlation projection matching algorithm that compares the Fourier transform of a raw image with Fourier transforms of 7x7 model projections and then converts each raw image from Cartesian (x,y) to polar (r,ψ) format (analogous to PFTSEARCH). The comparison is done for each omega value separately and the best of (2*SO+1) (SO is parameter for omega search interval) matches of Fourier transforms is selected.

ORIENTATION REFINEMENT PROCEDURE

This uses a SLIDING WINDOW approach to refine the θ,ϕ,ω values. The particle data set, which may consist of images from multiple micrographs (specified in one or more PARAM files), is first sorted so images are processed in order of increasing θ value. OOR then grabs as much memory as it can (whole asymmetric unit if possible) and computes projections of the current 3D map on a regular grid ("lattice") whose spacing is set by the program input parameter DELTA_THETA (see later). Projections of the 3D map are computed for a 7 by 7 grid of θ,ϕ values for each image in the sorted data set. The grid point at the center of each window is arranged so the θ,ϕ value falls closest to the current θ,ϕ value for the image. OOR then finds the 'best' θ,|ϕ| value in the local search window in the same manner as PFTSEARCH (recall, in PFTSEARCH the search is performed for each image over the **entire** asymmetric unit). In addition a search is made for the best omega value in the search interval (2*SO+1).

The number of projections computed in the search window of a single image will often be less or much less than 49 (=7x7) for one of two reasons: 1) If a previous search window overlaps the current window, then the redundant data is reused, and this cuts down on needless recomputing. 2) If the current particle ϕ value is close to zero and the search window 'encroaches' -ϕ space, only grid points with positive ϕ value are searched (projections corresponding to θ,ϕ and θ,-ϕ orientations only differ in hand for objects with at least an n-fold axis of symmetry - [*NOTE: need to verify this statement*]). By far the greatest savings in compute time for OOR comes from #1. Indeed, program efficiency markedly *improves* as the number of particles *increases* because the reuse of projection data increases accordingly. Note, however, for a fixed number of images, the number of window overlaps decreases as DELTA_THETA is reduced because the extent of each search window shrinks.

A disadvantage of computing projections only at specific θ,ϕ grid values is the *potential* loss of precision for each orientation determination. However, Table 1 ought to dispel fear of this being a problem with typical cryoEM data.

Table 1: Positional accuracy of orientation search as function of particle size

			Δ°		
D(Å)	2.0°	1.0°	0.5°	0.25°	0.10°
250	8.7	4.4	2.2	1.1	0.43
500	17.5	8.7	4.4	2.2	0.87
1000	34.9	17.5	8.7	4.4	1.75
2000	69.8	34.9	17.5	8.7	3.49

The values in the Table (in Å) represent the distance that points on opposite sides of a spherical object (diameter, D), appear to move in the projection plane when the object is rotated Δ°. The following formula is used to compute this distance:

$$\text{Point separation} = \frac{2\pi D\Delta}{360}\ \text{Å}$$

D. PROGRAM OUTPUT

The primary output from (also input to) OOR is stored in one or more PARAM files. These contain basic information about each scanned IMAGE file (pixel size, voltage, etc.) as well as information about each particle image (ID, THETA, PHI, OMEGA, FFT_ORIGX, FFT_ORIGY, MAG_FAC, and three different correlation coefficients, FFTCC, PRJCC, CMPCC). In addition, various statistics on the correlation coefficients, as a function of radius (PFT.RADS) and as a function of resolution (PFT.RES1 and PFT.RES2) can be generated and stored in output files.

Three correlation coefficients (FFTCC, PRJCC and CMPCC) are listed for each particle in the PARAM file. The first, FFTCC, is a cross correlation coefficient computed between the FFTs of each 'raw' image and its corresponding model projection FFTs. FFTCC incorporates the effects of any resolution limits in reciprocal space. PRJCC is a global, real-space correlation coefficient computed directly between each 'raw' image (in polar coordinate format) and the corresponding model projection (also in polar format). For PRJCC, the coefficient is only computed for data between radii PFTRAD_LO and PFTRAD_HI. This coefficient is computed from polar real space data. The third correlation coefficient, CMPCC, is similar to PRJCC in that the computations are performed on real space data, but CMPCC uses normal Cartesian (x,y) format image and model projection data. **Note:** the values of PRJCC and CMPCC differ, in part (*need to check this*) because low radius data are overweighted in the polar representations of images and model projections. Though opinions are mixed, the CMPCC correlation coefficient *seems* to give the most reliable assessment of particle quality because FFTCC and PRJCC are more designed to help OOR screen for good θ and ϕ values (FFTCC) and for good X,Y, and ω values (PRJCC).

E. BATCH JOB PROGRAM INPUT

Oor now require a keyword organized input. The input is designed as follows:

```
keyword = value
```

where the "keyword" field denotes the data attribute you would like to set and the "value" field the value to which you would like to set it. For example to set the BIN_FACTOR you would like to use in oor to "2" you would enter the following line:

```
BIN_FACTOR = 2
```

Normally one would place all the keyword input in an input file and then pipe that into the program all at one time. For example, if your input file is named "input.dat" you might run as follows:

```
     % oor < input.dat > oor.log &
```

with oor.log containing all of the program output during the run.

The END_OF_KEYS keyword signals the program to stop gathering keyword input and continue by first gathering parameter filenames.

> NOTE: The minimum required input to identify the keyword is indicated in CAPITAL LETTERS. For example, the keyword "VERbose" will be recognized if only "VER" appears in the input stream. All other characters are then superfluous.

Line-by-line descriptions of program input:

1. INPut_mode, BIN_factor, SYMmetry, DELTA_Theta, CTF_Mode, VERbose, FILTER_FACTOR_1

INPUT_MODE = 3 Refine mode This is a dummy argument for the input. If the user inputs anything in this field it will be reset by the program to ensure INPUT_MODE=3.

CAUTIONARY NOTE: If you are reading in PFT/PRJ data stored on disk (generated by EMPFTPRJ), then it is **imperative** that you specify the same values for BIN_FACTOR, SYMMETRY, DELTA_THETA, RAD_LO, RAD_HI, and RAD_STEP that you used in EMPFTPRJ. Otherwise, the program is likely to crash or perform other unseemly (and unwanted) calculations.

BIN_FACTOR = 1 Binfactor of 1 (DEFAULT; *i.e.* no data compression)
= 2 Binfactor of 2 (compresses 3D data eightfold and 2D data fourfold)

The current limit on BIN_FACTOR is 2.

SYMMETRY = 1 for no symmetry (like a ribosome)
= 2-31 for n-fold cyclic symmetry (about z-axis)
= 532 for 532 icosahedral symmetry (DEFAULT)
= 222 for 222 dihedral symmetry
= 32 for 32 dihedral symmetry
= 422 for 422 dihedral symmetry
= 52 for 52 dihedral symmetry
= 622 for 622 dihedral symmetry
= 72 for 72 dihedral symmetry
= 822 for 822 dihedral symmetry
= 92 for 92 dihedral symmetry

SYMMETRY specifies the point group symmetry of the particle you are studying. The program searches one-half of the asymmetric unit appropriate for the chosen symmetry. The available symmetries are listed above.

In all search modes the half asymmetric unit (ASU) defines the limits of the search area. Orientations that stray outside the half ASU are folded back to the equivalent view within the ASU. The limits of the half ASU for various symmetries are as follows:

Symmetry	θ min	θ max	ϕ min	ϕ max
1	0.0	90.0	-180.0	180.0
5	0.0	90.0	-36.0	36.0
422	0.0	90.0	0.0	45.0
52	0.0	90.0	0.0	36.0
532	69.09	90.0	0.0	31.71

DELTA_THETA is used to specify the step size in the θ and ϕ directions (in degrees: DEFAULT = 1.0). The step size in the θ direction remains constant (= DELTA_THETA) but it varies in the ϕ direction from a smallest value (= DELTA_THETA) when $\theta=90°$ (at the 'equator') and *increases* thereafter for progressively smaller values of θ. Varying the ϕ step size assures uniform sampling of the ASU in regions where θ is $<90°$. The step size in the ϕ direction is given by the formula:

$$\text{DELTA\_THETA}/\sin\theta$$

If the ϕ step size was not varied as given in the above formula, the grid sampling would be much too fine near the 'poles'. When θ approaches 0° or 180° (N and S poles), the program adjusts the ϕ step size to maintain even sampling. For example, at $\theta = 30°$, the ϕ step size will be twice DELTA_THETA (*i.e.* 2.0° if DELTA_THETA is 1.0°, since {1.0/sin(30°)}=2.0).

The value you choose for DELTA_THETA has a **tremendous** impact on OOR run time. As Table 1 showed (**Section C.**), it makes no sense to set this parameter very small, especially at the beginning of data analysis when GLOBAL orientation searches are performed. Table 2 demonstrates how the number of program calculations significantly increases as DELTA_THETA is decreased. Also, Table 2 clearly demonstrates how the "problem" becomes even more severe in the case of particles with lower symmetry.

Table 2: Number of views per ASU as a function of particle symmetry and orientation search angle increment (Δ°)

	Point Group Symmetry			
Δ°	532	52	5	1
3.00	48	266	519	2,353
2.00	91	573	1,127	5,253

1.00	370	2,173	4,309	20,809
0.50	1,430	8,471	16,869	82,869
0.25	5,606	33,439	66,733	330,751
0.10	34,327	207,358	414,353	2,064,448

CTFMODE determines if and how the CTF is used. **Note:** It only makes sense to use this option **IF** the 3D model is a CTF-corrected map. If the map is an uncorrected one, then the use of CTFMODE = 1 or 2 will force the program to compare CTF-modified and CTF-unmodified data, which is undesirable and may lead to undetermined errors in refinement. The values of CTFMODE may be set as follows:

CTFMODE = 0 No CTF modifications to the data are made.

= 1 Projections of the 3D model are multiplied by the CTF, and these are compared to the unmodified, raw images.

= 2 The raw images are multiplied by the CTF, and these are compared to unmodified projections of the 3D model.

= 3 The raw images are multiplied by the INVERSE CTF, and these are compared to unmodified projections of the 3D model.

ILIST signals OOR to generate various forms of output:

ILIST = 0 Minimal output (refined orientations and origins)

= 1 Correlation coefficients are computed as functions of radius and resolution and results are stored in output files EMPFT.RADS, EMPFT.RES1, EMPFT.RES2.

= 2 Same as ILIST=1, but also gives correlation coefficients for each particle view

2. PFTRAD_LO, PFTRAD_HI, PFTRAD_STEP, ANNUNLUS_Low, ANNUNLUS_High, RADius, TEMperature_factor

These variables (real space pixel units) define the annular portion of the projected data (PFTRAD_LO to PFTRAD_HI) to be used in the PFT calculations. PFTRAD_STEP sets the radial step interval and hence determines the number of annuli in each PFT (NANNULI = [[PFTRAD_HI-PFTRAD_LO]/PFTRAD_STEP] + 1). PFTRAD_LO is normally left = 1.0 and PFTRAD_HI is usually set to a value just larger than the particle boundary (but usually much smaller than NCOL/2 if you boxed the original particle images conservatively). Use PFTRAD_LO > 1.0 if you suspect that the projected data at higher radii (*i.e.* projection only of outer capsid features) will give a more sensitive measure of the orientation parameters.

PFTRAD_LO can't be set lower than 1.0 because the center of the projected view doesn't change with orientation and hence gives no useful information for the PFT calculations. The default for

PFTRAD_HI is (NCOL+1)/2, but note that this will generally be too large especially if the particle boxing was performed conservatively.

PFTRAD_STEP is normally left = 1.0. Using a larger value will reduce the number of computations (smaller NANNULI) but use cautiously (especially if BIN_FACTOR=2) so you don't sample the data too coarsely. If PFTRAD_STEP is < 1.0, you may make needless calculations (NANNULI too large), especially if BIN_FACTOR=1 and the images were digitized at a pixel resolution at least twice as fine as the expected resolution limit. NOTE: This variable is slated for removal in the future.

ANNUNLUS_LOW,ANNUNLUS_HIGH define the range of annuli from the real space polar coordinate image and projection data that are included in the calculations. The best way to determine optimum values for these parameters is to run OOR with the initial DEFAULT values (0,NANNULI-1), and check the output file EMPFT.RADS to see the correlation coefficients a function of radius. The **average** correlation coefficient typically undergoes a large drop near the outer edge of the particle (ANNUNLUS_HIGH). Also, the correlations are generally lower at low radii (ANNUNLUS_LOW) corresponding to the 'core' part of the structure. Thus, ANNUNLUS_LOW and ANNUNLUS_HIGH may require some fine tuning to optimize the orientation and origin search procedure.

The chosen bin_factor value (Line #1) does not change the entered value of PFTRAD_LO, PFTRAD_HI, ANNUNLUS_LOW and ANNUNLUS_HIGH. Conversions are made automatically by the program.

RADIUS is not used by OOR, it is a placeholder for compatibility with other programs.

TEMPERATURE_FACTOR is an inverse temperature factor applied to the Fourier spectrum of both projections and raw images.

3. RESOLUTION_LOW, RESOLUTION_HIGH, JCUT, SIG, DELTA_XY, NUM_DELTA_XY, DELTA_OMEGA, NUM_DELTA_OMEGA (2F,I,5F)

RESOLUTION_LOW and RESOLUTION_HIGH define the lower and upper resolution limits of the data to be included in the calculations. These program input variables should be specified in the same units as PIXSIZE, where PIXSIZE is the size of each pixel in the digitized images and is specified for each image in the corresponding PARAM file. PIXSIZE may be defined in any units you choose (Å, nanometers, pixels, yards, etc.) but you **MUST** be consistent and use the **same** units to define PIXSIZE, RESOLUTION_LOW and RESOLUTION_HIGH. A value of RESOLUTION_HIGH **smaller** than 2*PIXSIZE (the DEFAULT) is disallowed (this would exceed the Nyquist limit which is two-pixel resolution). You should treat this value for RESOLUTION_HIGH as an **absolute lower limit**, which, if used, is likely to be an unrealistic value for real (*i.e.* noisy) data. Hence, **use careful judgment** in setting the value of RESOLUTION_HIGH.

The DEFAULT for RESOLUTION_LOW [IDIM1*PIXSIZE] is probably unrealistically large. Again, PFT works best if some of the low resolution data are ignored. This is because, for example, for particles with icosahedral symmetry the very low resolution data only carry information about spherical symmetry and little if any information about icosahedral symmetry. **General rule of thumb:** set RESOLUTION_LOW to a value **no smaller** than one-fifth the size of the particle diameter. For example, if the particle diameter is 500Å, then a value for RESOLUTION_LOW of 100Å *might* be an appropriate starting point for OOR. If you prefer to specify PIXSIZE in pixel units (instead of Å or another unit), and the pixel size of the digitized image corresponds, for example, to 3.86Å units, then RESOLUTION_LOW would be 25.9 (=500/(5*3.86).

REMINDER: Use good judgment in setting the above program variables!!! The success or failure of OOR may very well depend on your ability or lack thereof to set these values.

JCUT specifies the minimum rotational Bessel order (Jn) to include in the calculations. The DEFAULT (=1) omits the Jo term, which is recommended in analyzing icosahedral particles because this removes the spherically symmetric image components and boosts the sensitivity of determining icosahedral orientations. To include Jo, set JCUT to any negative value. To cut out higher orders, JCUT is set to a value greater than 1 (NOTE: to my knowledge, no one has ever carefully tested the effects of doing this).

SIG allows the program to filter the PFT data on the basis of the variance of the PFT data. In theory, this option should greatly sensitize OOR. However **BE FOREWARNED:** this option is still **UNTESTED** so it may not and probably does not work!!! SIG specifies the threshold for the variance mask. With SIG=0.0 (DEFAULT), the masking option is disabled.

DELTA_XY specifies the step size in pixels and NUM_DELTA_XY specifies the number of steps in searching. For example, if DELTA_XY = 0.3 and NUM_DELTA_XY = 4 then there will be + and - 4 steps in each direction each at 0.3 pixels in X and Y for a total of 9x9 steps or 81 overall. DELTA_OMEGA and NUM_DELTA_OMEGA are similar to DELTA_XY and NUM_DELTA_XY but in the omega direction.

4. MAG_CEN, MAG_STEP, MAG_NUM, MAG_NORM (2F,2I)

These program input parameters are used to direct the magnification factor refinement and CMP correlation coefficient (CMP_CC) calculations.

MAG_CEN specifies the midpoint of the magnification scale factor search. The program tests for magnification factors that are (MAG_NUM-1)/2 steps above and below MAG_CEN. Set MAG_CEN = 0.0 or a negative number to force the program to use the MAG factor for each image as stored in the PARAM input data file. Hence, when *any* positive value of MAG_CEN (e.g. 1.0) is chosen, the MAG search "window" for *all* images will be over the *same range*. *** NEED TO CHECK THIS OUT ***

MAG_STEP defines the grid size of the magnification search. A value of MAG_STEP = 0.005, for example, corresponds to 0.5% increments.

MAG_NUM establishes the extent of the magnification search "window". This should be an odd integer > 0. For example, with MAG_CEN = 1.0, MAG_STEP = 0.005, and MAG_NUM = 11, the search window will encompass magnification factors ranging between 0.975 and 1.025. Entering '1' will force MAG to be either MAG_CEN or the value read from the PARAM file (which occurs whenever MAG_CEN is set ≤ 0.0).

MAG_NORM is a switch used to normalize the MAG scale factors so that the *average* MAG is 1.0. This occurs only when MAG_NORM is set equal to 1, otherwise, the MAG values determined by the program are output without being normalized.

5. MAP_filename - 3D model input filename (A FORMAT)

Enter the name of the file that contains the 3D model from which new PRJs and PFTs are to be calculated. **Note:** OOR currently only works if the dimension (NCOL_MAP = NROW_MAP = NSEC_MAP) of the 3D model **exactly** matches the image dimension.

6. PRJ_filename - PRJ input filename (A FORMAT)

Enter the name of the file that contains the model projection data (DEFAULT = PFT.PRJS). This data is only read in if MODE = -1 or MODE = -2, otherwise the program calculates what it needs at run time.

7. PFT_Filename - PFT input filename (A FORMAT)

Enter the name of the file that contains the model PFT data (DEFAULT = PFT.PFTS). This data is only read in if MODE = -1 or MODE = -2, otherwise the program calculates what it needs at run time.

8. PFTRADS_filename - empft.rads filename (A format)

Enter the name of the file that will contain the radial distribution of correlation coefficients.

9. PFTRES1_filename - empft.res1 filename (A format)

Enter the name of the file that will contain the distribution of cross correlation coefficients PMAP versus POLAR_PRJ.

10. PFTRES2_filename - empft.res2 filename (A format)

Enter the name of the file that will contain the distribution of cross correlation coefficients filtered PMAP versus POLAR_PRJ.

11. PARAM input filename(s) (A FORMAT)

Enter the names of up to 999 PARAM files, with each filename on a new input line. The format of each PARAM file is:

LINE	INPUT
1	IMAGE filename (A) This is the name of the IMAGE file which contains byte-packed raw image data stored in *.PIF format.
2	PIXSIZE, UNITS, VOLTS, AMP_FAC, DELF_MAJ, DELF_MIN, ANG_MAJ, Cs (F,I,6F)

PIXSIZE	= pixel size for data in IMAGE file. May be specified in any units (*e.g.* Å, nanometers, pixels, etc.) as long as you make sure to specify the correct value for UNITS.
UNITS	= specifies the units assigned to PIXSIZE. UNITS is defined as follows:
	= 0 assumes PIXSIZE is given in dimensionless pixels
	= 1 assumes PIXSIZE is given in Ångstroms
	= 2 assumes PIXSIZE is given in nanometers
VOLTS	= microscope voltage (in volts) for the image data in the IMAGE file.
AMP_FAC	= amplitude factor. For cryoEM data, a DEFAULT value of 0.07 is reasonable and anything above 0.15 might be suspicious.
FOCUS_MAJ	= defocus value (μm) along the major axis of astigmatism. **Note**: Positive values are used to designate underfocus.
FOCUS_MIN	= defocus value (μm) along the minor axis of astigmatism.
ANG_MAJ	= angle between major axis and X-direction in FFT, measured positive in a counter-clockwise direction (X-axis = 0.0)
Cs	= spherical aberration coefficient (mm) for the microscope used to record the data in the IMAGE file.

LINE	INPUT
3	ID, THETA, PHI, OMEGA, FFT_ORIGX, FFT_ORIGY, MAG_FAC, FFTCC, PRJCC, CMPCC (I,9F)

ID	= specifies the particle # in the *.PIF format IMAGE file.
THETA,PHI,OMEGA	= orientation of particle #ID
FFT_ORIGX,FFT_ORIGY	= center of particle #ID (FFT coordinates)
MAG_FAC	= scale of particle #ID relative to a standard (model)
FFTCC,PRJCC,CMPCC	= three correlation coefficients

N+2	Same as #3 for as many particles (N) as needed.

OOR outputs a new PARAM file for each one used as input. The new PARAM data files are named with a specific convention: a "_00#" is tagged after each input PARAM filename. For example, suppose you had but one input PARAM file named "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 OOR, in which case the subsequent PARAM file would automatically be named "MYDATA.DAT_002", and so forth. The importance of understanding the way this works is **crucial**! If, by 'accident', you forget to update the correct PARAM filename in your BATCH COMMAND PROCEDURE file prior to the next cycle of OOR (*e.g.* leaving the name "MYDATA.DAT" in the command file), the output will end up in a file named "MYDATA.DAT_001" **INSTEAD** of "MYDATA.DAT_002".

FINAL NOTE: if you use several PARAM files as input to OOR, then use some rational naming system so the output files will be clearly distinct from the input files. Here's one example in which hypothetical PARAM files for four different micrographs are used:

OOR Cycle#	Input PARAM filenames	Output PARAM filenames
1	HSV_1856.DAT_001	HSV_1856.DAT_002
	HSV_1858.DAT_001	HSV_1858.DAT_002
	HSV_1900.DAT_001	HSV_1900.DAT_002
	HSV_1989.DAT_001	HSV_1989.DAT_002
2	HSV_1856.DAT_002	HSV_1856.DAT_003
	HSV_1858.DAT_002	HSV_1858.DAT_003
	HSV_1900.DAT_002	HSV_1900.DAT_003
	HSV_1989.DAT_002	HSV_1989.DAT_003

```
!*************************************************
!* EXAMPLE OOR BATCH JOB RUN                     *
!*************************************************
3, 2, 532, 1.0, 0, 0
1.0, 46.0, 1.0, 0, 46, 0, 200.0
200., 40., 1, 0.0, 0.7, 4.0, 0.9, 3.0
0.0, 0.001, 11, 0
HSV.PIFMAP
HSV.PRJS
HSV.PFTS
Empft.rads
Empft.res1
Empft.res2
HSV_1856.DAT_002
HSV_1858.DAT_002
HSV_1989.DAT_002
```

Point to appropriate directory:

```
% cd ~tsb/v/hsv/test
```

Run OOR with the file created above ("oor.bch") and create log file "oor.log":

```
% OOR < oor.bch > oor.log &
```

F. STRATEGY FOR USE OF OOR AND EM3DR

See Section F.) in the PFTSEARCH document for a complete description of this strategy.

G. DISCLAIMER

To date (5/98) this program has worked quite well in the analysis of a wide variety of specimen images. The routines seem to work as long as the starting model as well as subsequent reconstruction maps are *on track*. If your starting model is seriously flawed, don't expect miracles! Also, for particle image data that exhibits very weak enantiomorphic features, the model **must include** some correct enantiomorphic character or the refinement will lead to a 3D map that exhibits mirror line symmetry about the equatorial line. In our experience this has **not** been a problem for icosahedral particles with handed surface lattices (e.g. T=7 papovaviruses) because the arrangement of morphological units is clearly enantiomorphic even at very low resolution (>50Å). In many instances, enantiomorphic features do not become apparent until much higher resolutions (<30-40Å), and therefore proper refinement of data cannot proceed until the model incorporates information at the higher resolutions. In tricky situations, it is still advisable to use programs like EMICOGRAD with small data sets (5 particles or less) to try and make sure that the particles are oriented with respect to a consistent hand. A crude 3D reconstruction computed from such a limited data set, although noisy, may give a much better model for further refinement with OOR.

H. CREDITS

The original code for OOR was developed by T. Baker in ~1988 and tested in a class project by J. Tesmer in ~1990. R. H. Cheng performed more extensive and rigorous tests which led to a full scale working version of the program in 1993. The cross correlation algorithm was implemented by Xing Zhang in 2002. A description of the program as used during the period from about 1993 up through 1997 appears in:

Baker, T. S. and R. H. Cheng (1996) A model-based approach for determining orientations of biological macromolecules imaged by cryoelectron microscopy. *J. Struct. Biol.* **116:120-130.**

Some preliminary discussion of the routine also appears in:

Cheng, R. H., V. S. Reddy, N. H. Olson, A. J. Fisher, T. S. Baker, and J. E. Johnson (1994) Functional implications of quasi-equivalence in a T=3 icosahedral animal virus established by cryo-electron microscopy and X-ray crystallography. *Structure* **2:271-282.**

A number of people have and continue to make valuable additions/corrections/suggestions to OOR. These include (among others and in alphabetical order), Robert Ashmore, Misha Sherman, Steve Walker, Wei Zhang at Purdue, David Belnap and James Conway (NIH), and Stephen Fuller (EMBL).

We acknowledge with thanks permission to use an interpolation subroutine supplied by Michael Unser (NIH). This routine is described in:

M. Unser, A. Aldroubi, and M. Eden (1991) Fast B-Spline transforms for continuous image representation and interpolation. *IEEE Trans. Pattern Anal. Machine Intell.* **13(2):277-285.**

I. FINAL NOTES

1. OOR currently ONLY works with cubic 3D MAP data (NCOL=NROW= NSEC). Also, the program expects ALL particle images to have the same dimensions (NCOL x NROW) as in the 3D model.
2. DEXTRO3:[TSB.FOR]PFT.BCH is an example BATCH command file used to run OOR.

J. PROGRAM MODIFICATIONS

The following gives a history of significant changes that have been made to the program. More recent changes can be found in the header of the program itself.

DATE MODIFIED	BY WHOM	COMMENT(S)
20-MAY-1995	DMB	Added GLOBAL search option
XX-JUL-1996	DMB	Added SCALE, CC_CMP, & PFT_CALC_AVGS
xx-JAN-1997	DMB	Changed/moved data input & THE,PHI calc.
XX-MAY-1997	RWA/JC	Added PIF format
19-DEC-1997	TSB/RWA	PIF MAP compatible
23-DEC-1997	TSB/RWA	PRJ/PFTS in core
7-JAN-1998	TSB	Implemented BIN mode
8-JAN-1998	TSB	Using MAG instead of SCALE
9-JAN-1998	TSB	PFT_GLOBAL subroutine
15-JAN-1998	TSB/RWA	*.PIF input of PFTs/PRJs
19-JAN-1998	TSB	Multiple param file input capable
4-FEB-1998	TSB	Read/store parameter data in arrays
6-FEB-1998	TSB	Add MAG normalization
11-FEB-1998	TSB	Add DMB's FLIP changes
6-MAR-1998	TSB	Make UNIX compatible
27-JUL-1999	TSB	Incorporate use of CTF as per SBW
25-AUG-1999	TSB	Incorporate use of TSB CTF routines

14-NOV-2003	XZ	Implementation of Cross correlation algorthm
14-NOV-2003	RWA/MBS	Code cleanup and excise pftsearch code.

♦♦♦
♦
The FORTRAN code for OOR is in the source code archive (/bio/baker9d3/rwa).
♦♦♦
♦

K. FLOW CHART FOR OOR PROGRAM

** means “see below”; # means end of program; ! signifies no subroutine calls

```
****************
*    MAIN      *
*   (OOR.FOR) *
****************
     *
     *         |- GET_NVIEW_MOD - GET_T1T2P1 !
     *         |- PIF_OPEN !
     *         |- PIF_READ_GH - differentEndian !
     *- INFO -|- PIF_READ_DH ----------------------|- differentEndian !
     *         |                                    |- convertBackFloat !
     *         |               |- PIF_CLOSE !
     *         |- PFILE_INFO -|- PIF_OPEN !
     *         |               |- PIF_READ_GH - differentEndian !
     *         |- PIRADDEG !
     *         |- FFT_SETDIM_DEF_SAME !
     *
     *- GETMEM0 - MALLOC !
     *
     *- FILL_PARAMS !
     *
     *- PARAM_SRT (Lixun’s routine)
     *
     *           |- STOP_WATCH_START !
     *           |- GETMEM_REFINE - MALLOC !
     *           |- PIF_READ_WMAPI2 - PIF_READ_MAPI2**
     *           |- PIF_CLOSE !
     *           |- CALC_TP_MASK - PIF_WRITE_DEBUGI4 - PIF_WRITE_BYTE_IMAGE !
     *           |- GETMEM_REFINE2 - MALLOC !
     *- REFINE -|- PIF_OPEN !
     *           |- PIF_READ_GH - differentEndian !
     *           |- GET_PRJSPFTS_R**
     *           |- PIF_CLOSE !
     *           |- PIF_READ_DH -|- differentEndian !
     *           |                |- convertBackFloat !
     *           |- PIF_READ_IMGI1 - PIF_READ_BYTESHORT_IMAGE !
     *           |- PIF_READ_IMGI2 - PIF_READ_SHORT_IMAGE !
     *           |- PIF_READ_IMGI4 - PIF_READ_FLOAT_IMAGE !
     *           |- STOP_WATCH_STOP !
     *           |- LOADSTATS_R - PIRADDEG !
     *           |- REFINE_CC**
     *           |- GET_ITHE2 !
     *           |- FREEMEM_REFINE - FREE !
     *
     *- WRITE_PARAMS - INCREMENT_FNAME !
     *- PIF_CLOSE !
     *- STOP_WATCH_STOP !
     *- CALC_AVGS !
     *- FREEMEM0 - FREE !
     #

================================================================================
=
|                               |- PIRADDEG !
|                              |- MAP_CLEAR !
|                  |- MAP_PRJ -|- MAP_PRJ_XZ --- MAP_CLEAR !
|                  |           |- MAP_PRJ_AXIS - MAP_CLEAR !
|                  |           |- MAP_PRJ_ALL -- MAP_CLEAR !
|                  |
|                  |- SAVE_PRJS — MAP_SYM**
|- GET_PRJSPFTS_G -|- MAP_POLAR -|- MAP_CLEAR !
|                  |             |- MAP$POLAR - MAP_POLAR_GRID !
|                  |
|                  |- PMAP_FFT - FOURT - L6TO9 !
================================================================================
=
|             |- PIRADDEG !
|             |- GETMEM4 - MALLOC !
|             |
|             |              |- MAP_CLEAR !
|             |              |- PIF_READ_MAPI4**
|             |- PRJAVG_FFT -|- MAP_FFT_FILL - FFT_CLEAR !
|             |              |- FFT_2D - FOURT - L6TOL9 !
|             |
|             |- PIF_OPEN !
|             |- PIF_READ_GH - differentEndian !
|             |- PIF_CLOSE !
|             |- PIF_READ_DH -|- differentEndian !
|             |               |- convertBackFloat !
|             |
|             |- PIF_READ_IMGI1 - PIF_READ_BYTESHORT_IMAGE !
|             |- IMG_MAP !
|             |- MAP_FFT_FILL - FFT_CLEAR !
|             |- FFT_2D - FOURT - L6TOL9 !
|             |- CTF_MULTIPLY — CTF_ASTIG !
|             |- FFT_2D_BACK - FOURT - L6TOL9 !
|             |- FFT_MAP_FILL !
|             |- MAP_FFT_FILL - FFT_CLEAR !
|             |- FFT_2D - FOURT - L6TOL9 !
|             |- GET_XY**
|             |- INTERP2D (NIH C routine)
|             |- MAP_FFT_FILL - FFT_CLEAR !
|             |- FFT_2D - FOURT - L6TOL9 !
|             |- INTERP2D (NIH C routine)
|             |- MAP_FFT_FILL - FFT_CLEAR !
|             |- FFT_2D - FOURT - L6TOL9 !
|             |
|             |- MAP_XFLIP !
|             |- COPY_R4 !
|             |- INTERP2D (NIH C routine)
|             |- MAP_FFT_FILL - FFT_CLEAR !
|             |- FFT_2D - FOURT - L6TOL9 !
|             |- GET_XY**
|             |               |- IMG_FFT_FILL - FFT_CLEAR !
|             |               |- MAP_FFT_FILL - FFT_CLEAR !
|             |               |- FFT_2D - FOURT - L6TOL9 !
|             |               |- FFT_LOP !
|             |- GET_BESTMAG -|- FFT_HIP !
|             |               |- FFT_2D_BACK - FOURT - L6TOL9 !
|             |               |- FFT_IMG_FILL !
|             |               |- FFT_MAP_FILL !
|             |               |- INTERP2D (NIH C routine)
|             |- LIST_CCS**
|             |                |- TO_NFOLD !
|             |- TO_ASYM_UNIT -|- RETURN_TO_UNIT_TRIANGLE**
|             |                |- TO_N22 !
|             |
|             |- FREEMEM4 - FREE !
================================================================================
=
|             |- PIRADDEG !
|             |- IMG_MAP !
|             |- MAP_FFT_FILL - FFT_CLEAR !
|             |- FFT_2D - FOURT - L6TOL9 !
|             |- CTF_MULTIPLY — CTF_ASTIG !
|             |- FFT_2D_BACK - FOURT - L6TOL9 !
|             |- FFT_MAP_FILL !
|             |- INTERP2D (NIH C routine)
|             |- MAP_FFT_FILL - FFT_CLEAR !
|             |- FFT_2D - FOURT - L6TOL9 !
|             |
|             |             |- MAP_POLAR -|- MAP_CLEAR !
|             |             |             |- MAP$POLAR - MAP_POLAR_GRID !
|- REFINE_CC -|- GET_TPO_R -|- PMAP_FFT - FOURT - L6TO9 !
|             |             |- AVG_PFTIMG !
|             |             |- GET_THEPHI_G !
|             |             |- GET_BESTPRJ - PIF_READ_MAPI4**
|             |             |- GET_PHIOMEGA !
|             |- MAP_XFLIP !
|             |- COPY_R4 !
|             |- INTERP2D (NIH C routine)
|             |- MAP_FFT_FILL - FFT_CLEAR !
|             |- FFT_2D - FOURT - L6TOL9 !
|             |- GET_XY**
|             |               |- IMG_FFT_FILL - FFT_CLEAR !
|             |               |- MAP_FFT_FILL - FFT_CLEAR !
|             |               |- FFT_2D - FOURT - L6TOL9 !
|             |               |- FFT_LOP !
|             |- GET_BESTMAG -|- FFT_HIP !
|             |               |- FFT_2D_BACK - FOURT - L6TOL9 !
|             |               |- FFT_IMG_FILL !
|             |               |- FFT_MAP_FILL !
|             |               |- INTERP2D (NIH C routine)
|             |- LIST_CCS**
|             |
|             |                |- TO_NFOLD !
|             |- TO_ASYM_UNIT -|- RETURN_TO_UNIT_TRIANGLE**
|             |                |- TO_N22 !
================================================================================
=
|            |- COPY_I2 !
|            |- IMG_MASK !
|            |- IMG_FFT_FILL - FFT_CLEAR !
|            |- FFT_2D - FOURT - L6TOL9 !
|            |- COPY_R4 !
|- LIST_CCS -|- MAP_2DMASK !
|            |- MAP_FFT_FILL - FFT_CLEAR !
|            |- PFTCC_RES !
|            |- FFT_FLT -|- FFT_HIP !
|            |           |- FFT_LOP !
|             |
|             |- FFT_2DBT -|- COPY_R4 !
|             |            |- FOURT - L6TOL9 !
|             |
|             |- IMG_MAP!
|             |
|             |- MAP_POLAR -|- MAP_CLEAR !
|             |             |- MAP$POLAR - MAP_POLAR_GRID !
|             |
|             |- PFTCC_RAD !
================================================================================
=
================================================================================
=
|            |- MAP_SYM_CAVG - MAP_STATS - MAP$STATS !
|            |- MAP_SYM_RAVG -|- MAP_SYM_GRID !
|- MAP_SYM -|- COPY_R4        |- MAP_STATS - MAP$STATS !
|            |- MAP_STATS - MAP$STATS !
================================================================================
=
|                            |- POLAR_TO_COSINES - NORM !
|- RETURN_TO_UNIT_TRIANGLE -|- EQUIVALENT_VIEW -|- CROWTHER_TO_MATRIX !
|                                                |- MATMUL !
|                                                |- MATRIX_TO_CROWTHER !
================================================================================
=
|
|- PIF_READ_MAPI4 - PIF_READ_MAP_INT_IMAGE - differentEndian !
|
================================================================================
=
|
|- PIF_READ_MAPI2 - PIF_READ_MAP_SHORT_IMAGE - differentEndian !
|
================================================================================
=
|
|- PFTCC_R (*** NOT WRITTEN YET ***)
|
================================================================================
=
```