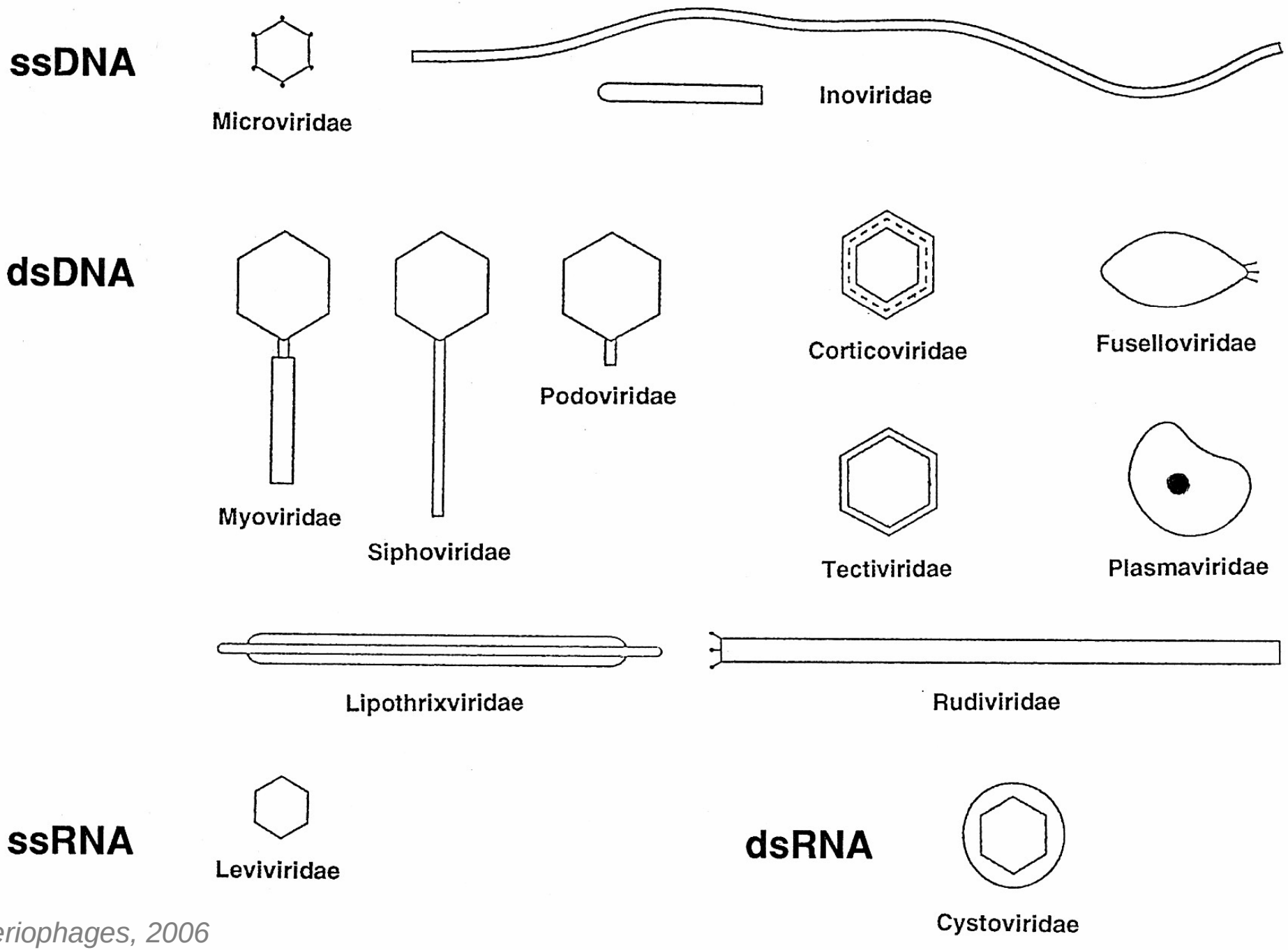
Icosahedral Reconstruction

Wen Jiang

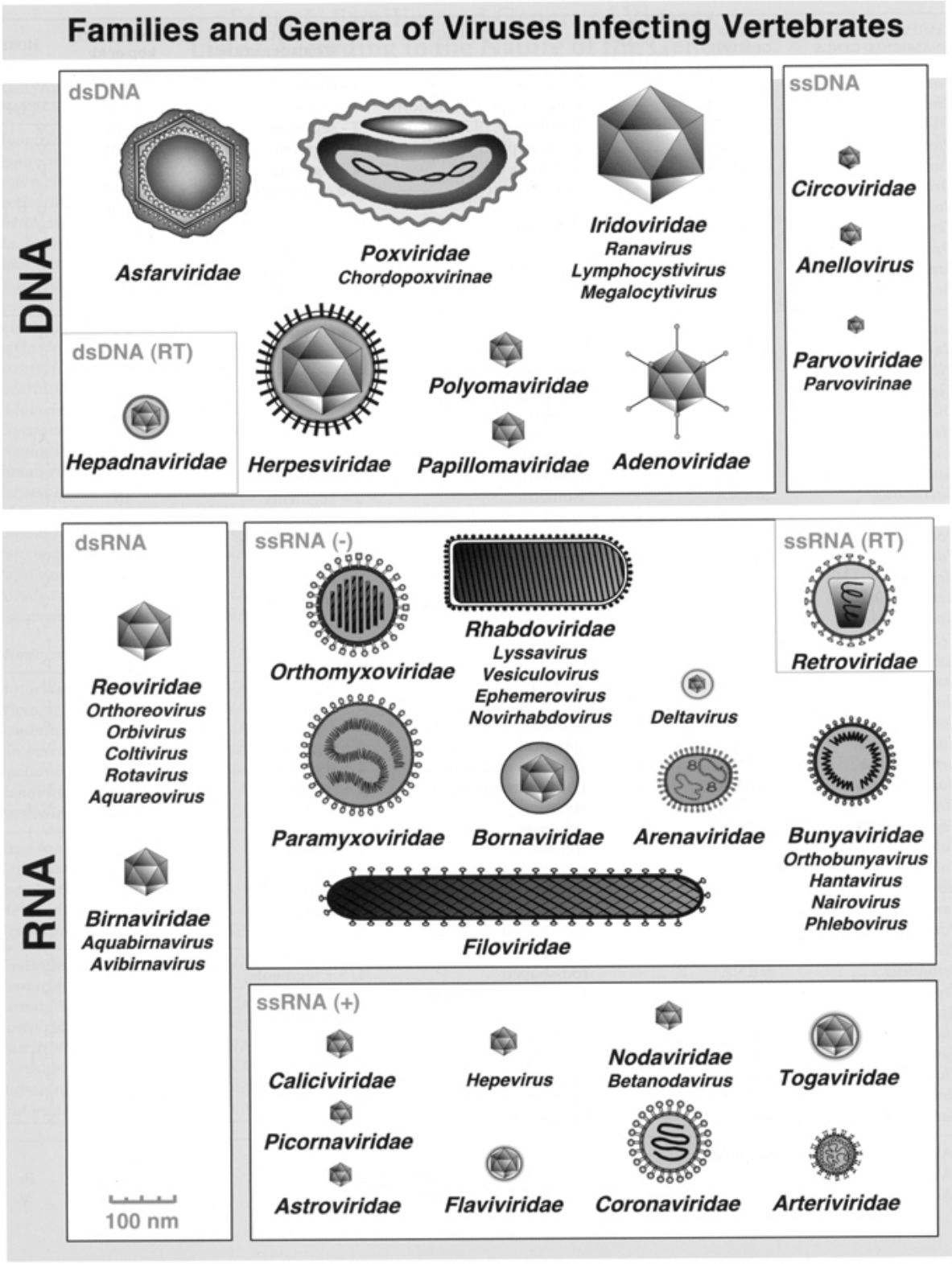
PURDUE
UNIVERSITY

Icosahedral Viruses: Bacteriophages



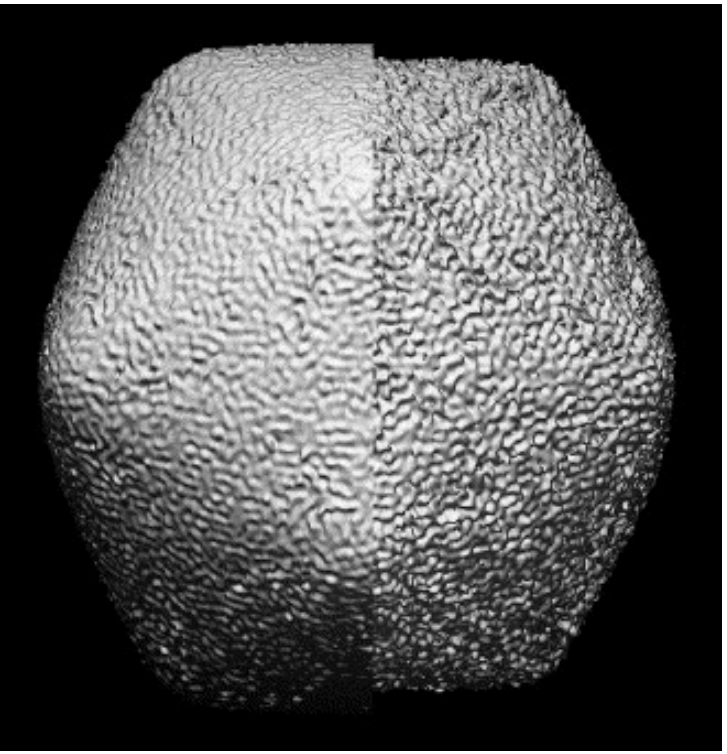
Bacteriophages, 2006

Icosahedral Vertebrate Viruses

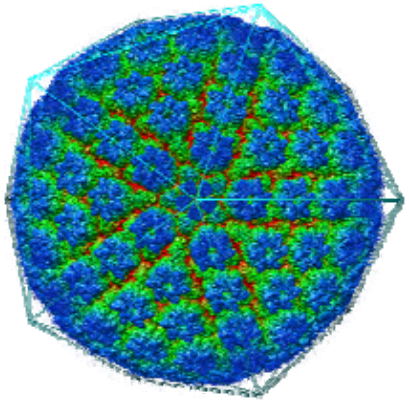


Fields Virology, 2006

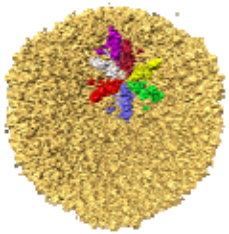
Icosahedral Particles



Mimivirus
5000Å



HSV
1250Å



ε15
700Å

HBV
300Å

PDC
200Å

Icosahedral Symmetry

- 5, 3, 2 fold axes
  - 6 five fold axes
  - 10 three fold axes
  - 15 two fold axes
- 60 fold symmetry in total
- Asymmetric unit: 1/3 of the triangular face

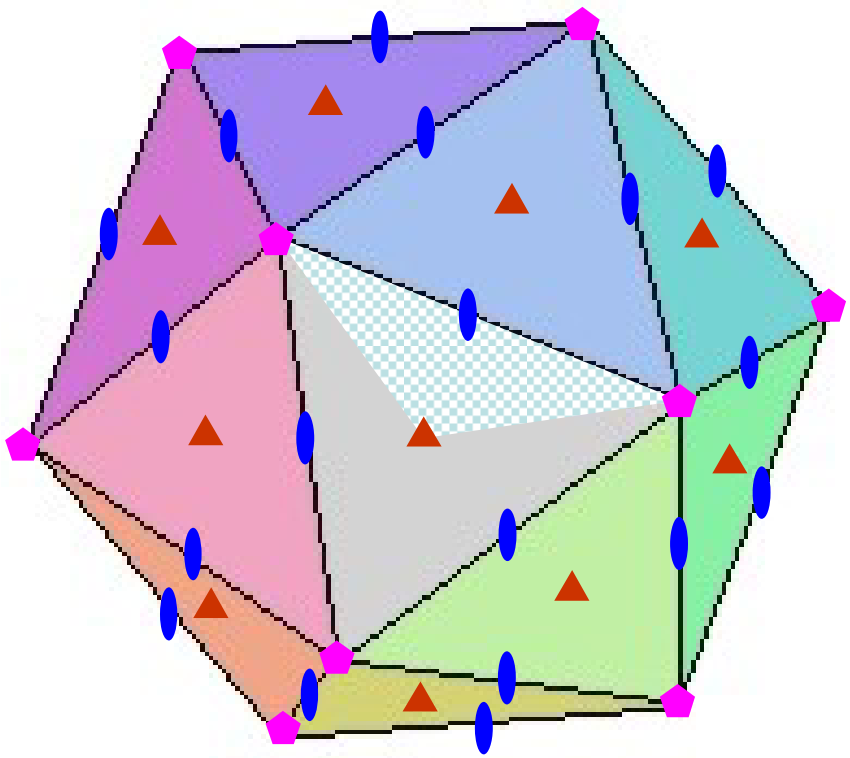


Image Processing

- In EMAN, icosahedral reconstruction is essentially treated the same as other symmetries

- However, one should pay special attention to the **“large”** problems

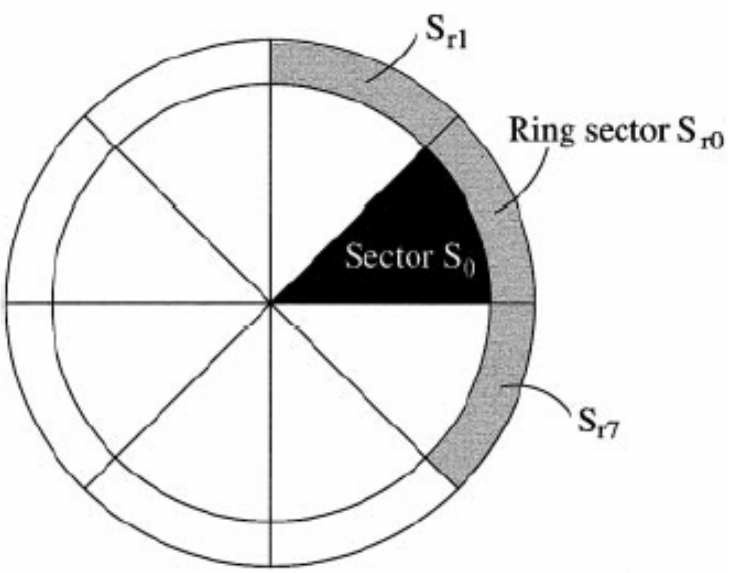
- Specialized methods exist for more efficient processing

Image Processing

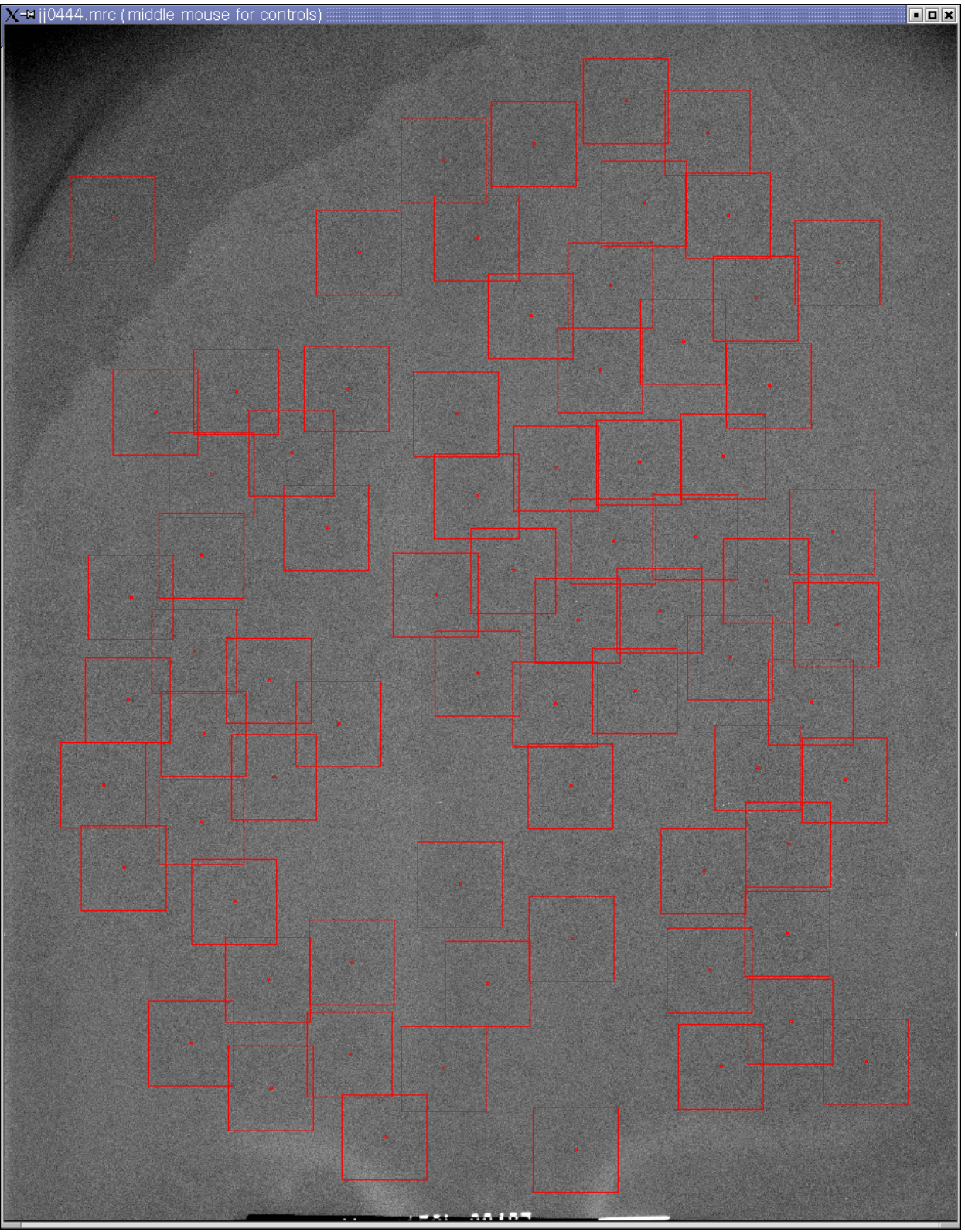
- Preprocessing
  - Particle selection
  - CTF fitting
- Build initial model
- Image refinement
  - Orientation determination
  - 3-D reconstruction

Particle Selection

- Manual selection
  - *boxer*
- Automated selection
  - *batchboxer*
  - *ethan (for spherical particles)*



- By Kivioja and Bamford *et al.*
- "Ring Filter"
- Only ONE parameter (*Particle Size*)
- Fast, 10-15 seconds / micrograph
- Tend to over-select



Ethan Example

Herpes
2.1 Å/pxiel
6662x8457 pixels
15 seconds in total

```
$ time ethan.py jj0444-8bit.mrc 298 600 jj0444.box
jj0444.img
Opening file jj0444-8bit.mrc
Width:  6662
Height: 8457
Averaging 121 pixels
Filtering jj0444-8bit.mrc: **********
Total number of squares is 567
Number of peaks after sector test is 163
Number of peaks after first distance check is 81
Number of peaks after discarding too small ones is
80
Refining centers
74 particles found from jj0444-8bit.mrc

10.31s user 2.08s system 84% cpu 14.578 total
```

Particle Selection

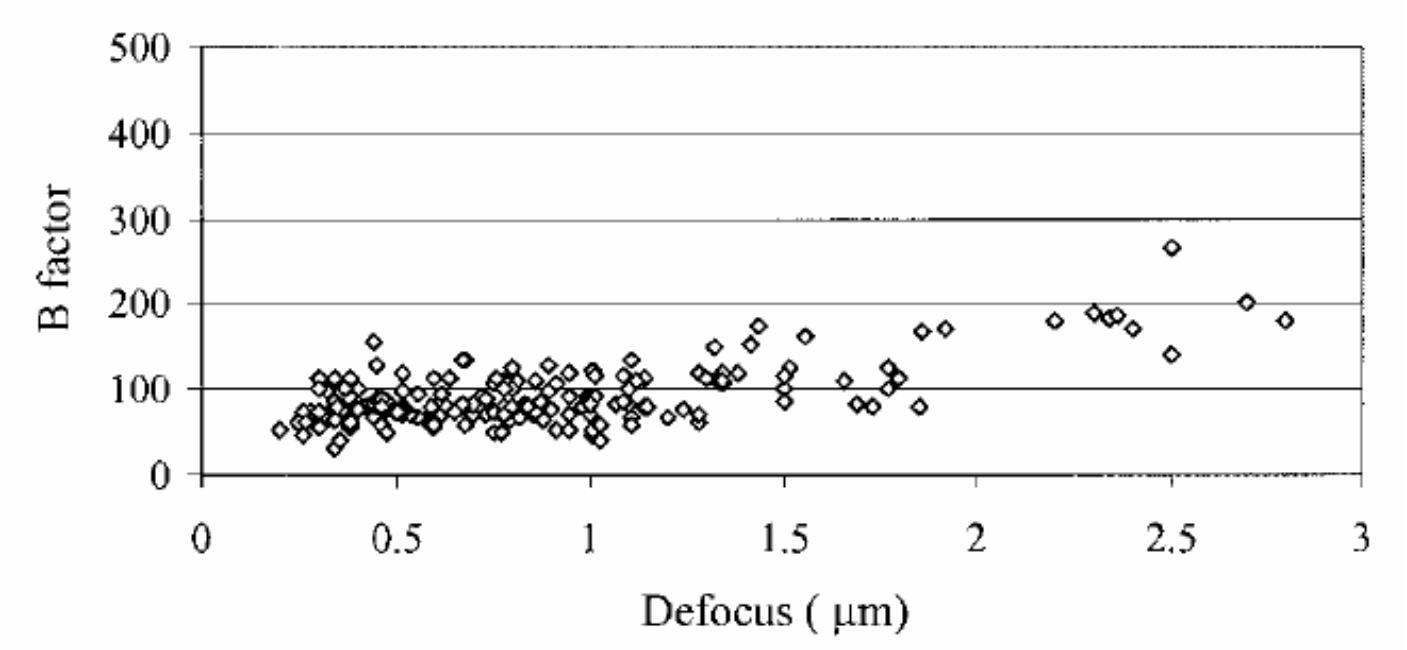
Suggested practice:

1. *Automated selection using ethan*

2. *Manual screening using boxer*

CTF Fitting

To reach high resolution, small defocuses (0.5-1.5 µm) are often used for imaging large icosahedral particles



Saad et al, JSB 2001

CTF Fitting

- Manual fitting
  - *ctfit*
- Automated fitting
  - *fitctf*
  - *fitctf.py*

$$I_{image}(s) = I_s(s)CTF^2(s)e^{-2Bs^2} + N(s)$$

$$ctf(s) = A(\sqrt{1-Q^2}\sin(\gamma(s)) + Q\cos(\gamma(s)))$$

$$\gamma(s) = -2\pi(C_S\lambda^3 s^4 + Z\lambda s^2/2)$$

$$N(s) = n_3 e^{-n_1\sqrt{s}-n_2 s-(\pi n_4 s)^2/4}$$

8 unknown parameters: *Z, B, A, Q, n1, n2, n3, n4*

Algorithm:
constrained nonlinear optimization using a sequential quadratic programming (SQP) method)

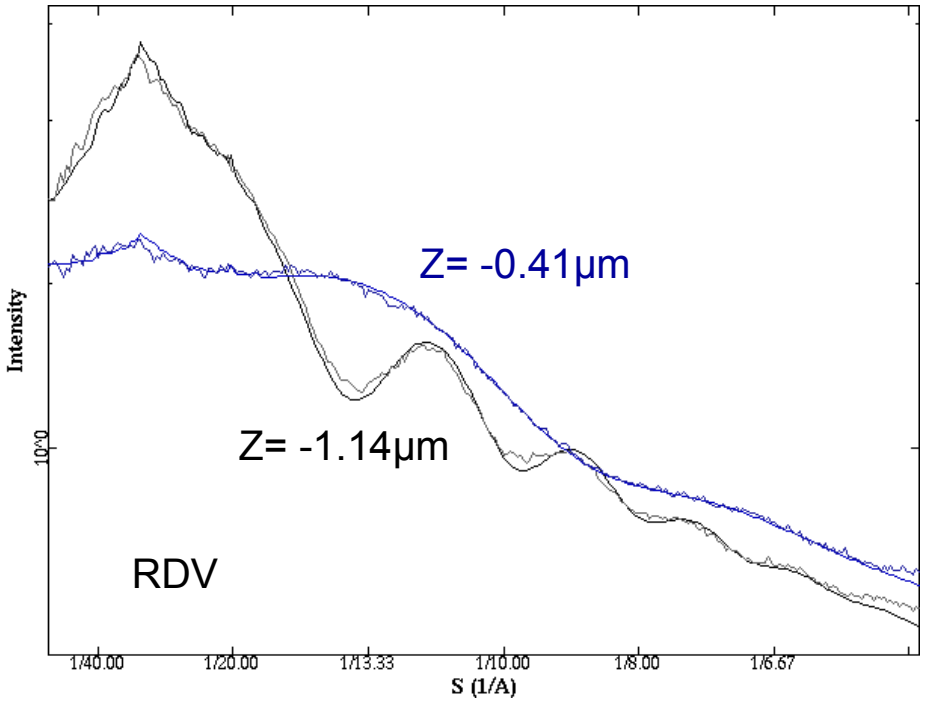
$$\min\left|I_{image}(s) - I_s(s)CTF^2(s)e^{-2Bs^2} - N(s)\right|^2$$

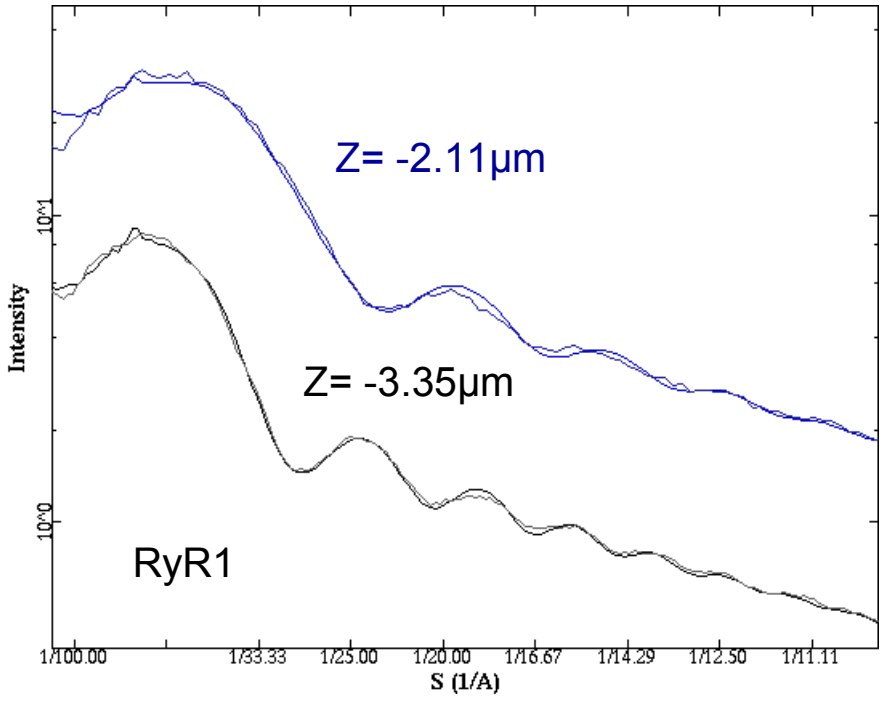
subject to:

$$I_{image}(s) > N(s)$$
$$A > 0$$
$$B \geq 0$$
$$1 \geq Q \geq 0$$
$$\Delta Z < 0 \text{ (under focus)}$$

Yang C. et al:
http://ncmi.bcm.edu/software/fitctf

fitctf.py Examples





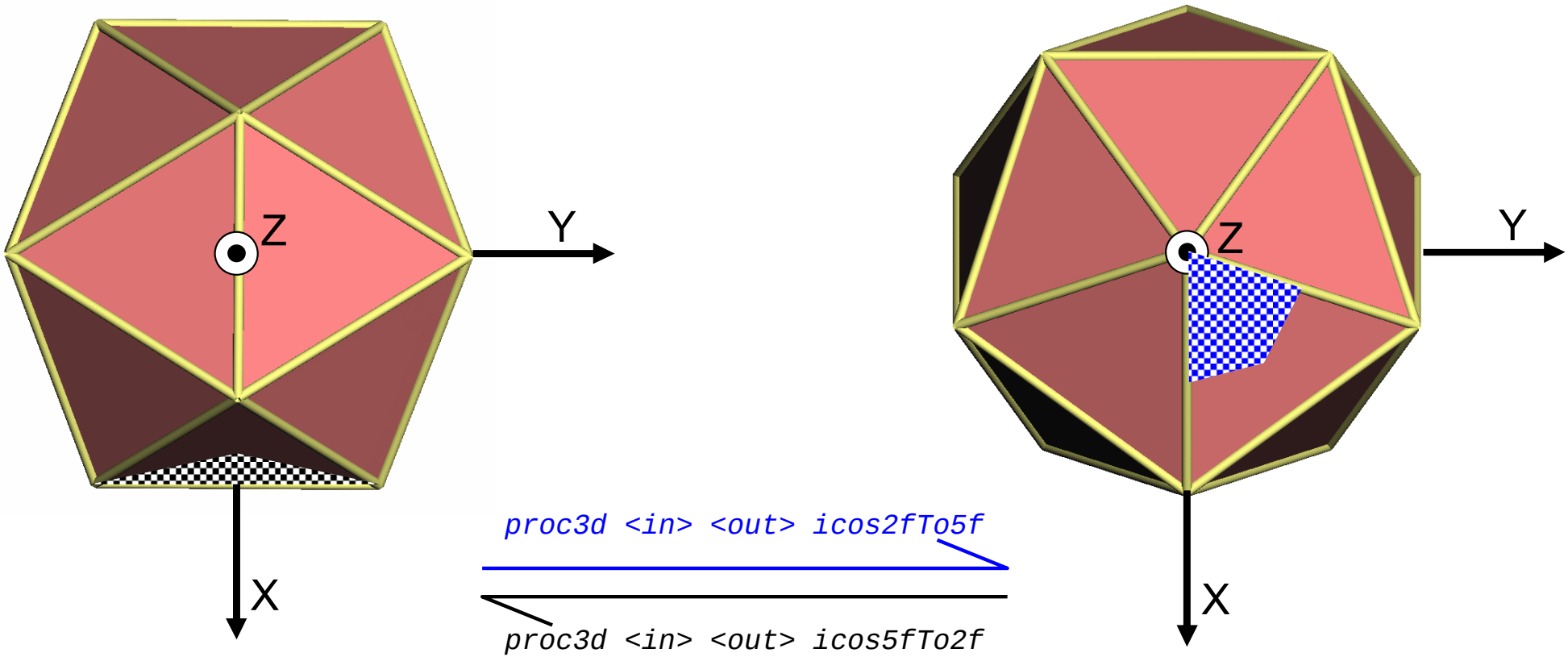
CTF Fitting

Suggested practice:

1. *Automated fitting using fitctf.py*

2. *Manual screening using ctfit*

Conventions: Orientation Origin



MRC

- X, Y, Z axes along 2 fold sym axes for orientation *(0,0,0)*
- Euler: Z → Y' → Z''

EMAN

- Z along 5 fold sym axis
- Y along 2 fold sym axis
- X near 3 fold sym axes for orientation *(0,0,0)*
- Euler: Z → X' → Z''

Build Initial Model

- Classic method: self common-line
  - Small number of particle with large defocus
  - Self common-lines cluster for views near symmetry axes

- Standard EMAN method: *starticos*
  - Find best 5-, 3-, 2-fold view particles
  - Spherical particles cause problems

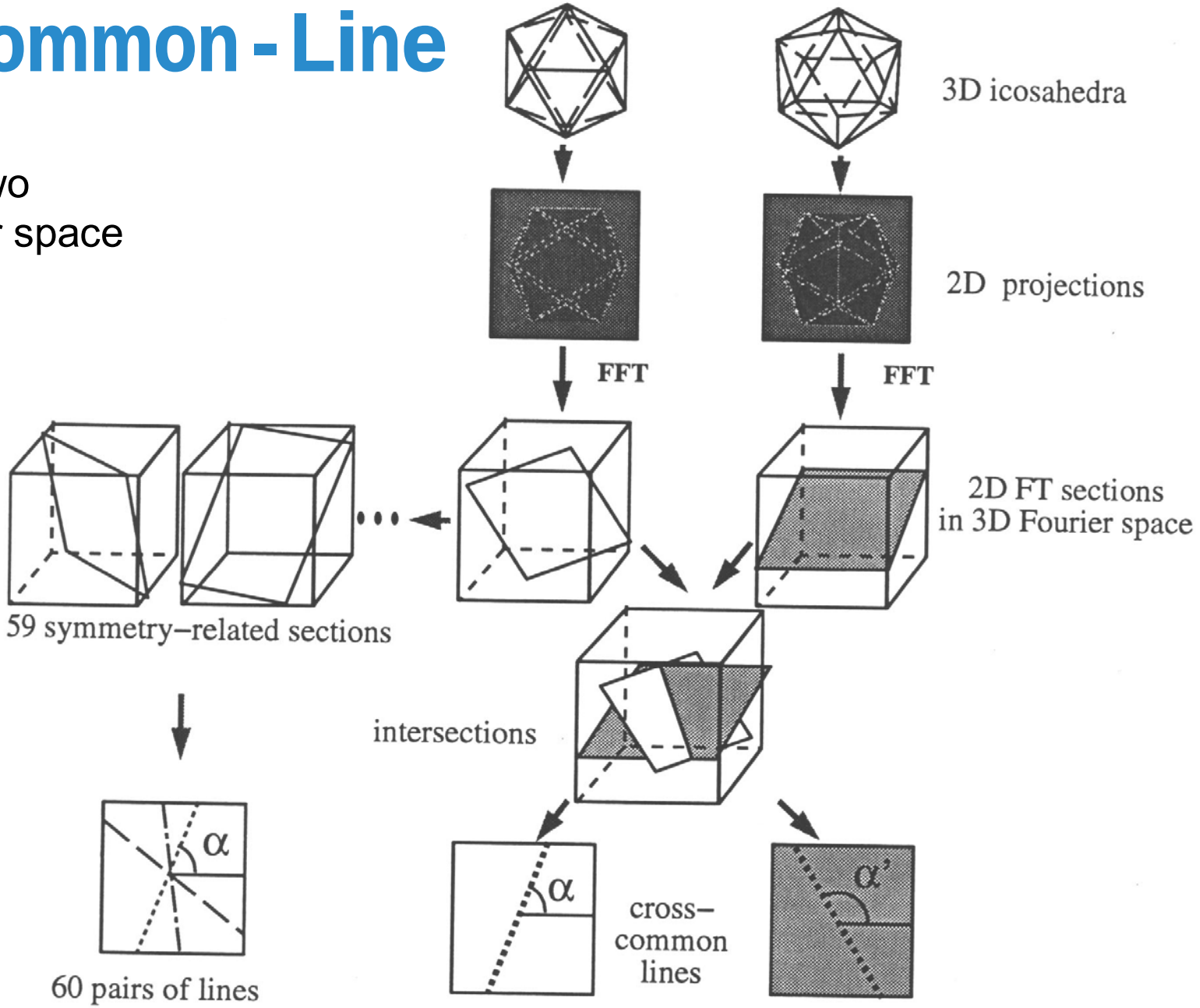
- New method: random model
  - Build 3-D model from particles with randomly assigned orientations
  - Relying on the wide convergence range to reach a correct structure

2-D Alignment Methods

- Common-line method:
  - Classic implementation: self common-line (global search to get coarse initial orientation) and cross common-line (local refinement) in tandem
  - EMAN implementation: Cross common-line method for both global search and local refinement (EMAN *crossCommonLineSearch.py* command)
- Projection matching method (EMAN *refine* command)

Cross Common-Line

Intersection of two
planes in Fourier space



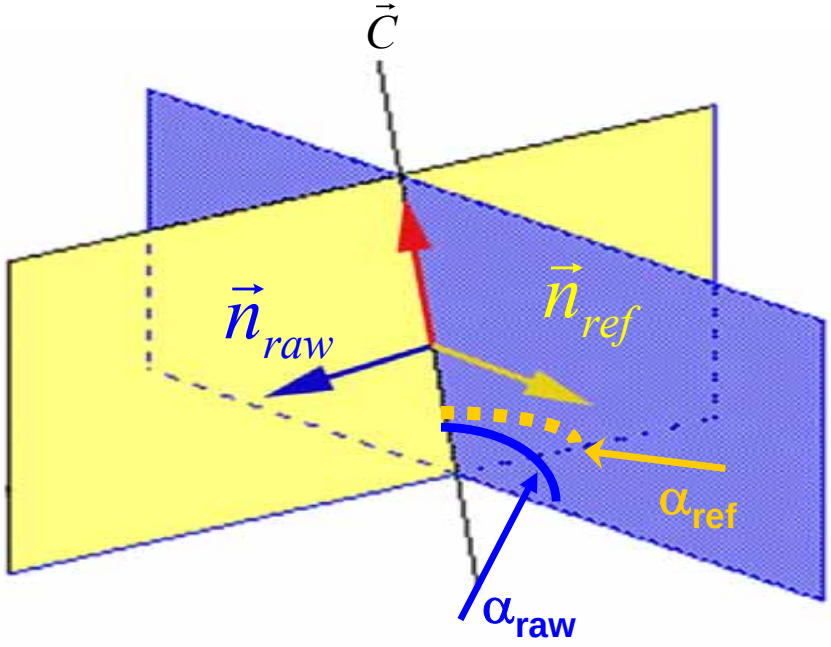
Z. Hong Zhou's thesis, p36

Orientation ➔ Common Line Locations

Fourier plane normal in 3D:

$$\vec{n}_{ref} = \vec{R}_{ref} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \vec{n}_{raw} = \vec{R}_{raw} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

common line in 3D: $\vec{C} = \vec{n}_{raw} \times \vec{n}_{ref}$



common line in 2D:

$$\vec{C}_{ref} = \vec{R}_{ref}^{-1} \cdot \vec{C} \qquad \vec{C}_{raw} = \vec{R}_{raw}^{-1} \cdot \vec{C}$$

$$\alpha_{ref} = \cos^{-1}(\vec{C}_{ref} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}) \qquad \alpha_{raw} = \cos^{-1}(\vec{C}_{raw} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix})$$

Common Line Search: Scoring Functions

search for the **orientation** and **center** parameters that minimize the mean phase differences among all the pairs of common lines

$$R_i(\phi,\theta,\omega,x,y)=\frac{\sum_{j=1}^{N}\sum_{k=1}^{k_{\max}(j)}\sum_{s=s_{\min}}^{s_{\max}} f\left(\psi_i(s,x_i,y_i,\alpha_{i,j,k}),\ \psi_j(s,x_j,y_j,\alpha_{j,i,k})\right)\times w(s,\alpha_{i,j,k},\alpha_{j,i,k})}{(s_{\max}-s_{\min})\sum_{j=1}^{N}k_{\max}(j)}$$

i: particle image

s: spatial frequency

k: common - lines

j: reference projections

f(): scoring function

w(): weighting function

Search Strategy

- Classic implementation

  1. **center** the particle by cross-correlation
  2. **self common-line** to determine initial orientation by exhaustive searching **orientation** at 1° step
  3. **cross common-line** to locally refine the **orientation and center** by **Gradient** or **Simplex**.

  Issues:

  1. Self common-line is not very robust
  2. Error propagation

Search Strategy

- Exhaustive cross common line search
  - enumerate all centers and orientations in an asymmetric unit (5-level loops !)
  - simple but really slow
  - for 1 degree, 1 pixel step: $3X10^7$ trials/particle

Search Strategy

- Multi-path Simulated Annealing
  - Multiple randomly seeded SA processes
  - Cross-communication among SA processes
  - Global search for both orientation and center
  - Accurate and fast
  - ~4×10^3 trials/particle.
  - >10^4 times faster than exhaustive search

Is The Best Solution Correct?

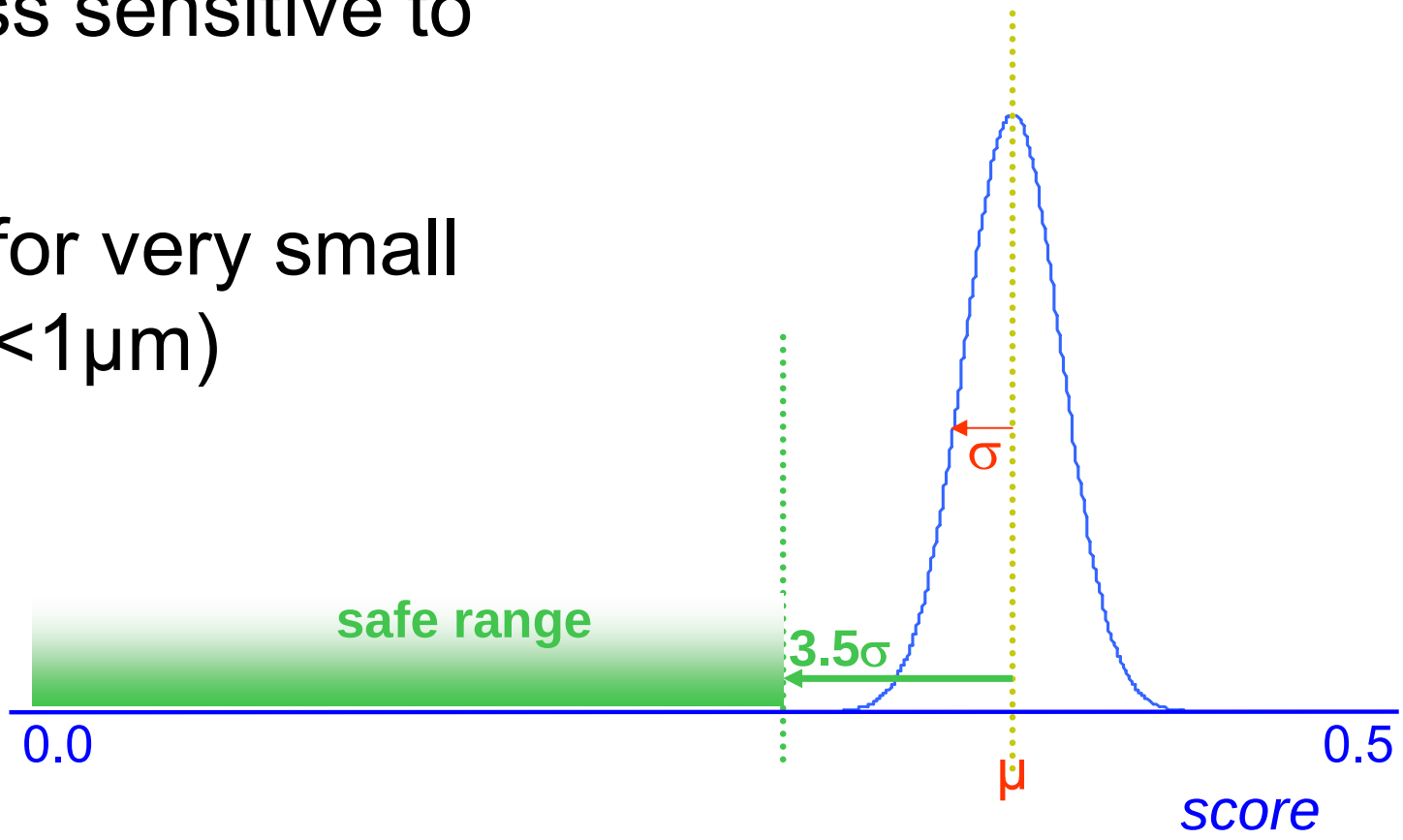
- Absolute score (residual) cutoff
  - popular criterion
  - needs different cutoff values for images at different defocuses
  - how to pick a single number that suits all imaging conditions?

Is The Best Solution Correct?

- Z score cutoff

$$Z = \frac{x - \mu}{\sigma}$$

  - better, less sensitive to defocus
  - unstable for very small defocus (<1μm)



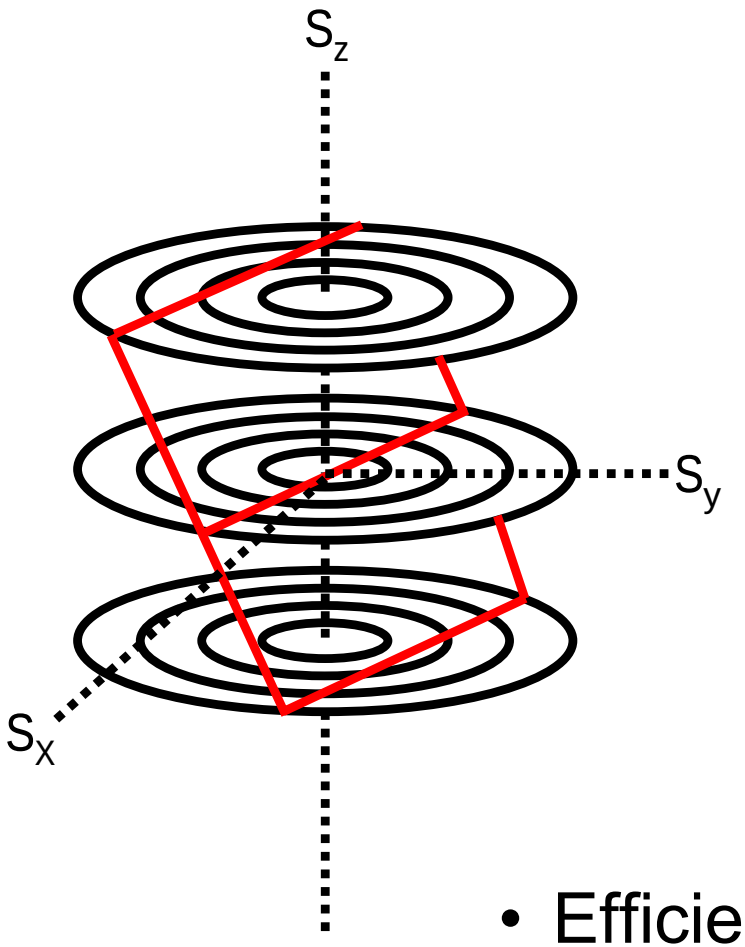
Is The Best Solution Correct?

- Consistency criterion
  - agreement among different methods (projection matching vs. common line)
  - agreement among multiple runs of the global common line search
  - more reliable than absolute value and Z-score cutoff
  - bias toward “safer side”

3-D Reconstruction Methods

- Fourier-Bessel Synthesis:
  - Classic method for icosahedral particles
  - Very efficient in computation
- Direct Fourier Inversion
  - EMAN *make3d* command
  - Needs lot of memory

Fourier Bessel 3-D Reconstruction



$$F(R,\Phi,Z) = \sum_{n=-\infty}^{\infty} G_n(R,Z) i^n e^{in\Phi}$$

FFT of 2D images

$$g_n(r,Z) = \int_0^{\infty} G_n(R,Z) J_n(2\pi Rr) 2\pi R dR$$

$$\rho(r,\varphi,z) = \sum_{n=-\infty}^{\infty} \int_{-\infty}^{\infty} g_n(r,Z) e^{in\varphi} e^{2\pi izZ} dZ$$

3D map

- Efficient but incomplete use of data
- Anisotropic artifacts often seen

Icosahedral Reconstruction Using EMAN

Two approaches

- Standard way using projection matching:

  refine sym=icos <other options>

- Cross common-line method:

  crossCommonLineSearch.py <options>

Special Considerations

- Load projections to memory in batches instead all at once: add “projbatches=<n>” with n>1 to *refine*
- Use half maps instead of full map to start *refine*: use “proc3d <in> <out> tophalf” to generate top half map
- Use fewer CPUs for projection and class averaging step to avoid IO timeout errors. Use “proc=16,64” instead of “proc=64” option for refine

Special Considerations

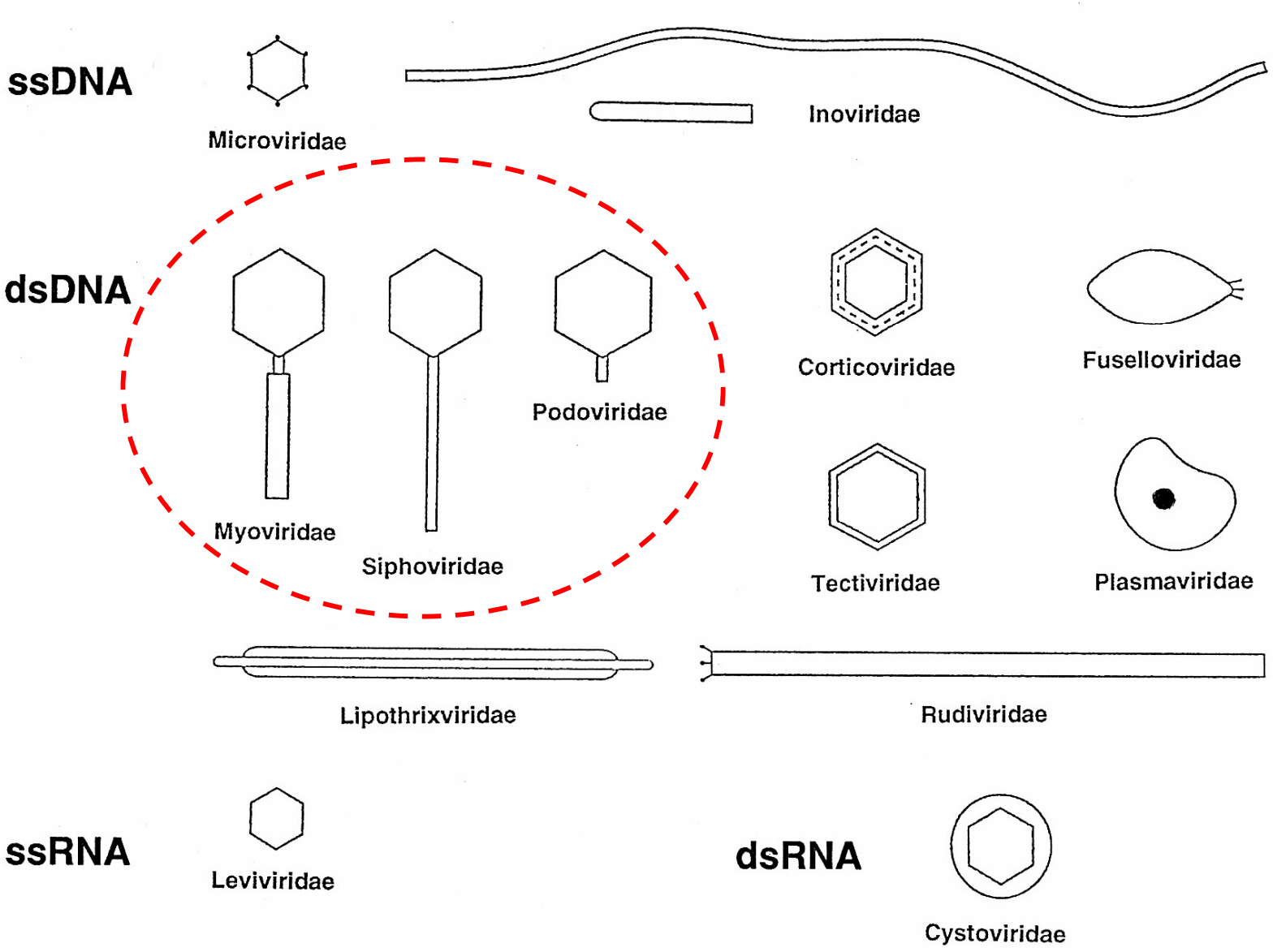
- Use parallel reconstruction instead of single CPU reconstruction. make3d is now parallelized using mpi.

  `mpiexec -n <# cpu> make3d …`

- Map size limits: 512^3 for 4GB, 768^3 for 8GB, 900^3 for 12GB

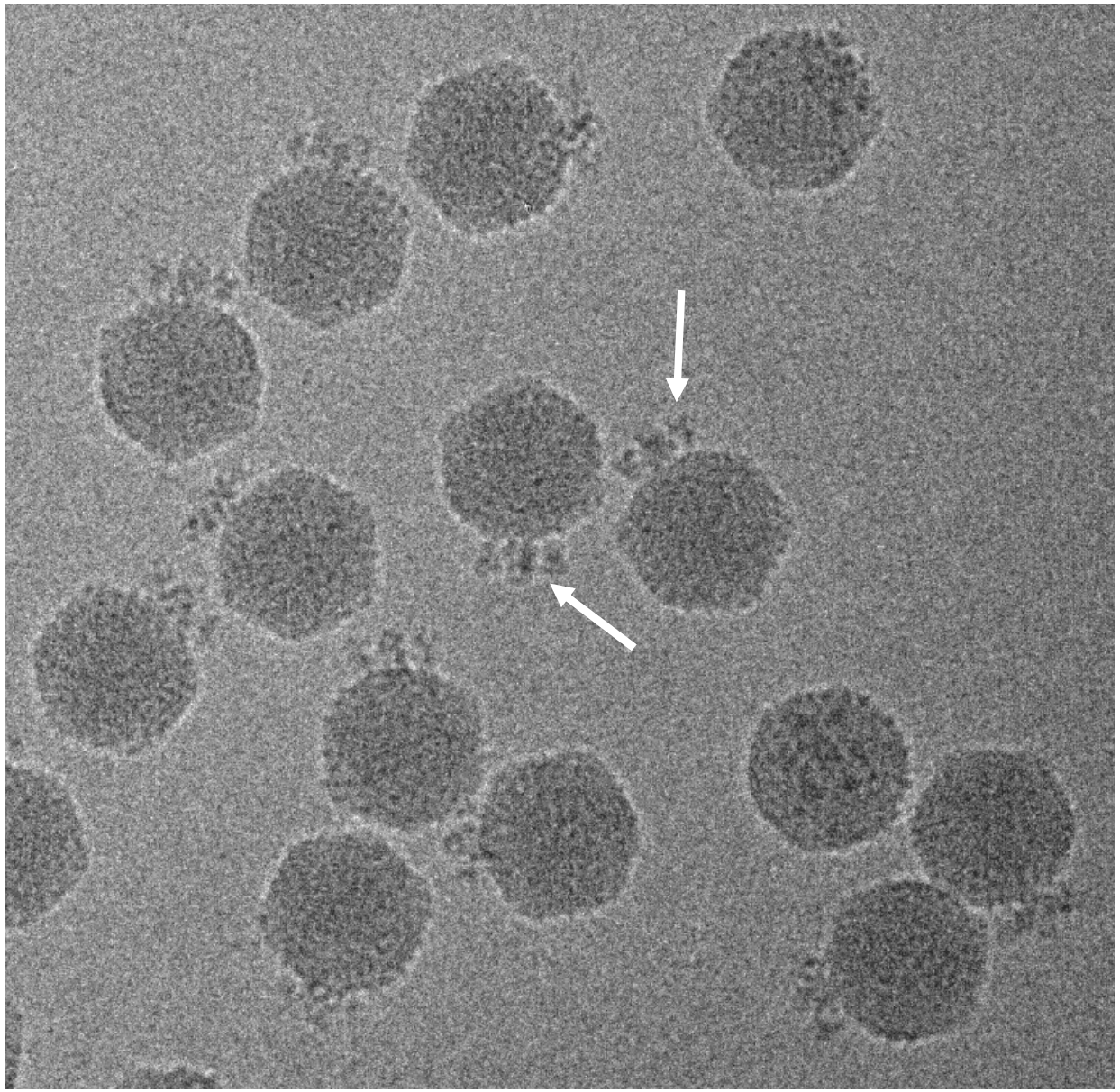
- Support mixed platform processing: one 64bit node with 12GB memory + many 32bit nodes with 4GB memory

Icosahedral Reconstruction ➔ Non-Icosahedral Reconstruction



Bacteriophages, 2006

Bacteriophage Epsilon15

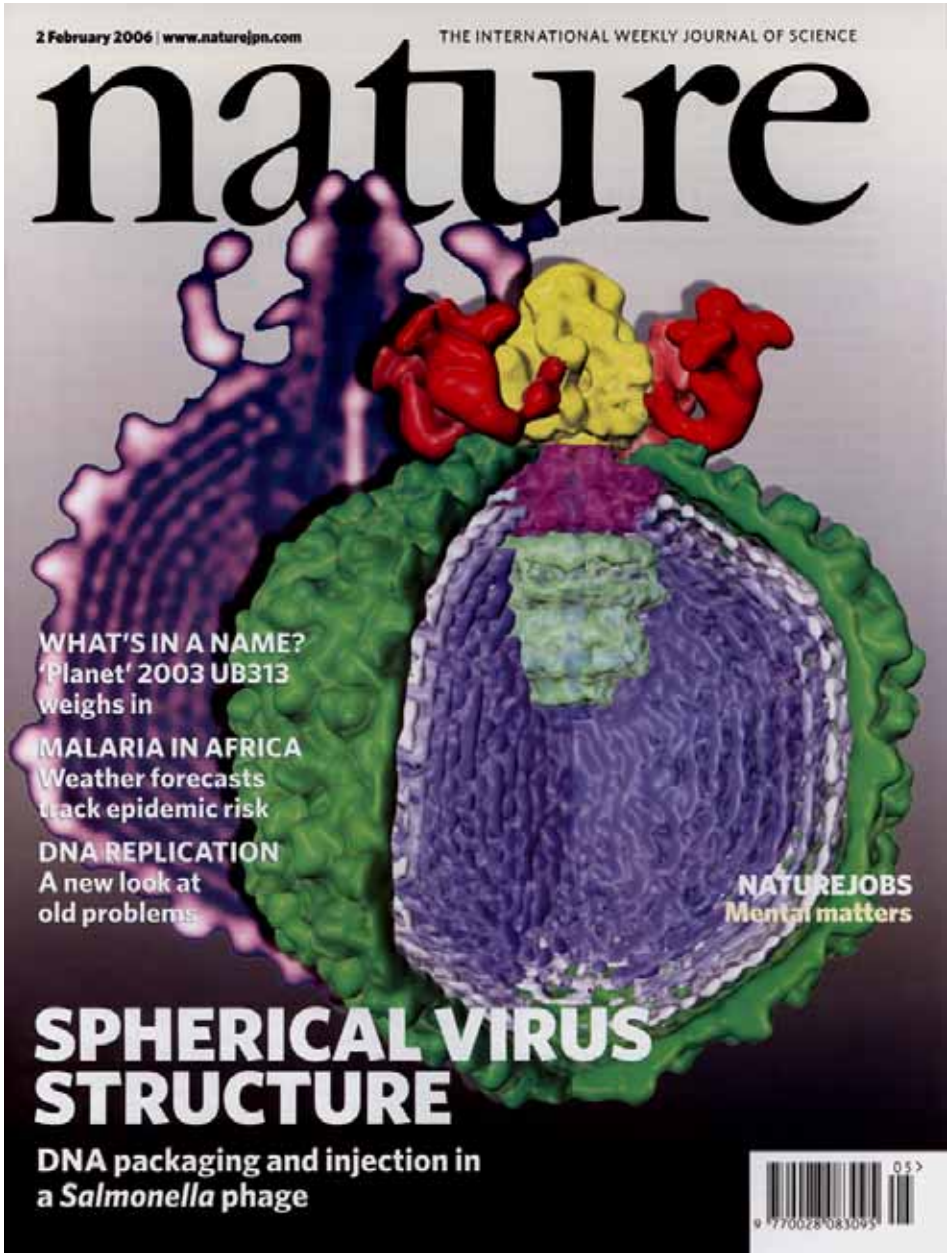


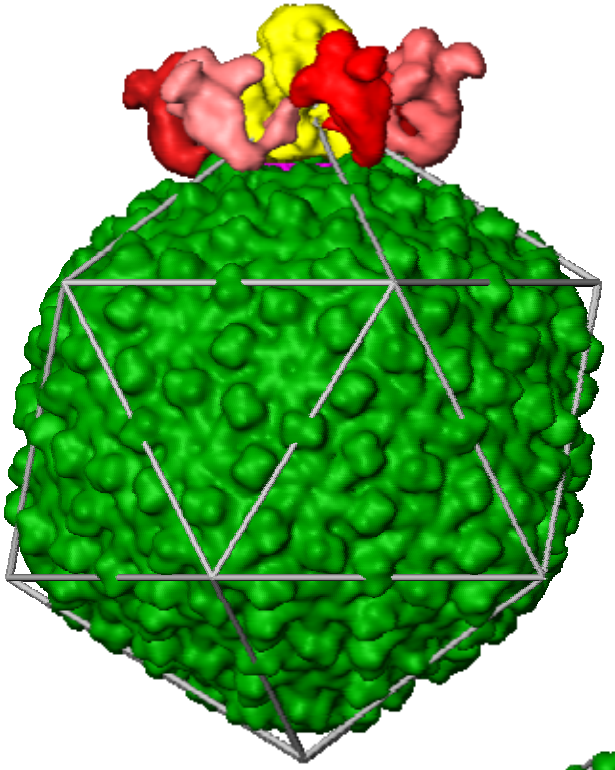
icosahedral reconstruction
(shell only)

asymmetric reconstruction

(shell, portal, tail, genome)

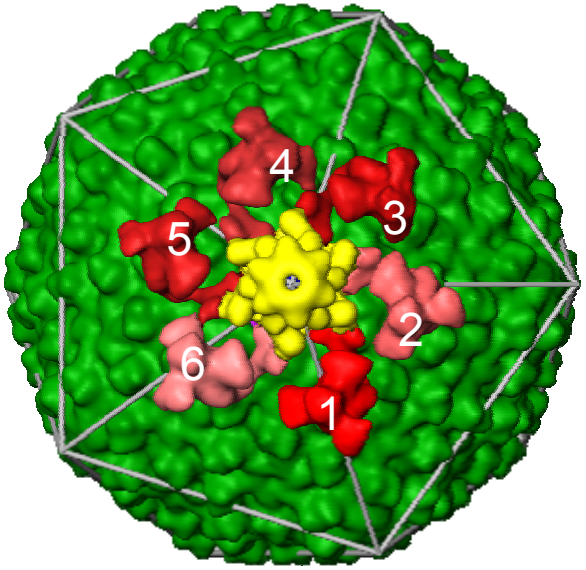
Asymmetric Reconstruction



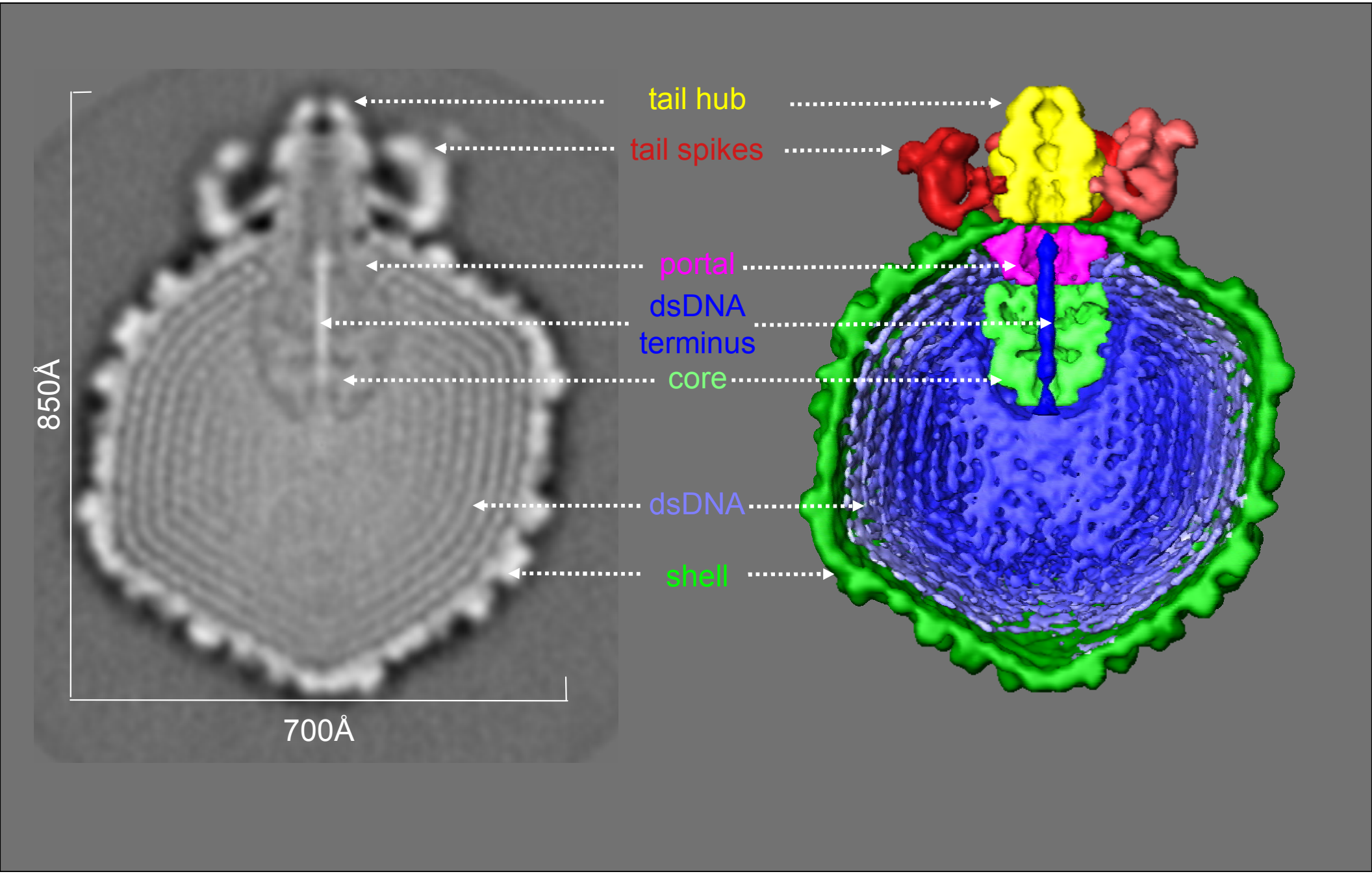


side view

top view



“Virus Anatomy”



Icosahedral Reconstruction ➔ Non-Icosahedral Reconstruction

refine

asymerfine.py