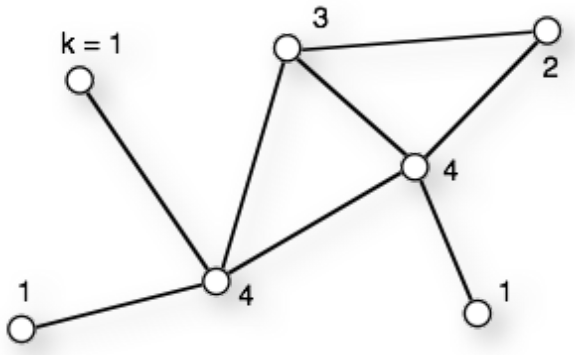
V2(a) Graph Basics – needed for assignments 1 & 2

A **graph** *G* is an ordered pair *(V, E)* of a set *V* of **vertices** and a set *E* of **edges**.

Degree distribution *P(k)*

$$P(k) = \frac{n_k}{N}$$



k	0	1	2	3	4
P(k)	0	3/7	1/7	1/7	2/7

Random network:

also called the "Erdös-Renyi model“:

- start with set of given nodes
- then add links randomly

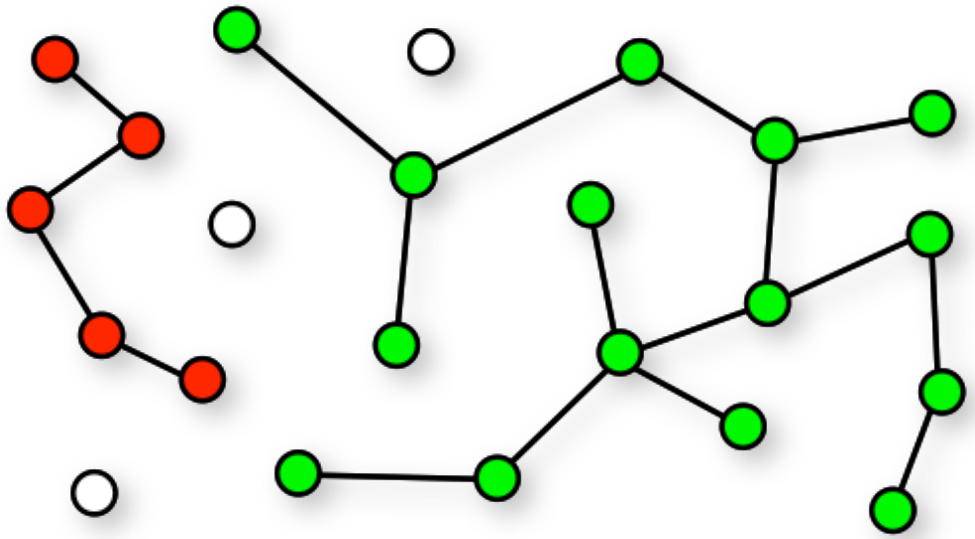
P(k) = "Poisson“ (will show this on the next slides)

$$P(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

Connected Components

Connected graph <=> there is a path between all pairs of nodes

In large (random) networks: complete $\{V\}$ is often not connected
→ identify connected subsets $\{V_i\}$ with $\{V\} = \cup \{V_i\}$
→ **connected components** (CC)



#CC = 5
N_{max} = 15
N_{min} = 1

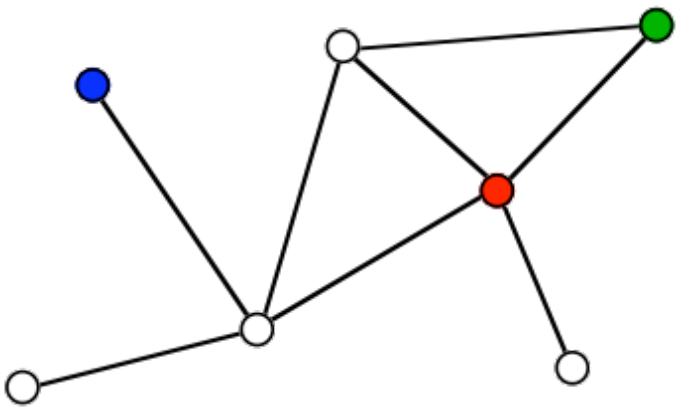
Connectivity of the Neighborhood

How many of the neighboring vertices are themselves neighbors?
=> this is measured by the **clustering coefficient** *C(k)*

Number of possible undirected edges between *k* nodes: $n_{max} = \frac{k\,(k-1)}{2}$

n_k is the actual number of edges between the neighbor nodes.

Fraction of actual edges ≅ **clustering coefficient** $C(k, n_k) = \frac{2n_k}{k\,(k-1)}$



green: $k = 2,\ n_k = 1 \rightarrow C = 1$

red: $k = 4,\ n_k = 2 \rightarrow C = 1/3$

blue: $k = 1,\ n_k = ? \rightarrow$ C is not defined

Note: clustering coeff. is sometimes also defined via fraction of possible triangles

Clustering Coefficient of a Graph

Data: C_i for each node i $\rightarrow$ N values

Statistics:

average at **fixed k**

$$\rightarrow \quad C(k) = \frac{1}{n_k} \sum_{k_i = k} C_i$$

average over **all nodes**

$$\rightarrow \quad \langle C \rangle = \frac{1}{N} \sum C_i$$

Note: it is also possible to average the *C(k)*
$\Rightarrow$ This yields a different value for <*C*> !!!
because no weighting is done for different occupancy of k's.

Basic Types: (1) Random Network

Generally: *N* vertices connected by *L* edges

More specific: **distribute** the edges **randomly** between the vertices

Maximal number of links between *N* vertices:

$$L_{max} = \frac{N(N-1)}{2}$$

=> **probability** *p* for an edge between two randomly selected nodes:

$$p = \frac{L}{L_{max}} = \frac{2L}{N(N-1)}$$

=> **average degree** λ

$$\lambda = \frac{2L}{N} = p\,(N-1)$$

path lengths in a random network grow with `ln`(*N*) => **"small world"**

Random Network: *P(k)*

Network with *N* vertices, *L* edges
=> probability for a random link:

$$p = \frac{2L}{N(N-1)}$$

Probability that random node has links to *k* other particular nodes:

$$W_k = p^k (1-p)^{N-k-1}$$

Probability that random node has links to any *k* other nodes:

$$P(k) = \binom{N-1}{k} W_k = \frac{(N-1)!}{(N-k-1)!\,k!} W_k$$

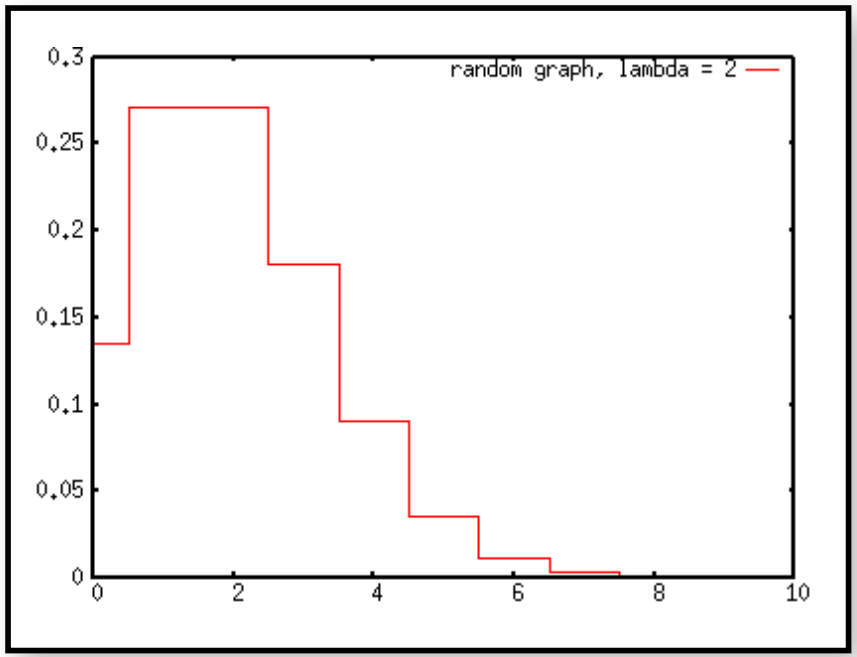
Limit of large graph: $N \to \infty$, $p = \lambda / N$

$$\begin{aligned}
\lim_{N\to\infty} P(k) &= \lim_{N\to\infty} \frac{N!}{(N-k)!\,k!} p^k (1-p)^{N-k} \\
&= \lim_{N\to\infty} \left(\frac{N(N-1)\dots(N-k+1)}{N^k}\right) \frac{\lambda^k}{k!} \left(1-\frac{\lambda}{N}\right)^N \left(1-\frac{\lambda}{N}\right)^{-k} \\
&= \qquad 1 \qquad\qquad \frac{\lambda^k}{k!} \qquad \mathrm{e}^{-\lambda} \qquad\quad 1 \\
&= \frac{\lambda^k}{k!} \mathrm{e}^{-\lambda}
\end{aligned}$$

Random Network: *P(k)*

Many independently placed edges => **Poisson statistics**

$$P(k) = \frac{\lambda^k}{k!} e^{-\lambda}$$



k	*P(k / λ = 2)*
0	0.14
1	0.27
2	0.27
3	0.18
4	0.090
5	0.036
6	0.012
7	0.0034
8	0.00086
9	0.00019
10	3.82e-05

=> Small probability for $k >> \lambda$

Basic Types: (2) Scale-Free

Growing network a la Barabasi and Albert (1999):

- start from a small "nucleus“ of m_0 connected nodes
- add new node with n links
- connect new links to existing nodes with probability p_i proportional to degree k_i of each existing node (“preferential attachment”);

=> "the rich get richer" $$p_i = \left(\frac{k_i}{\sum k_i} \right)^{\beta}$$ in BA-model β = 1

Properties:

- this leads to a power-law degree distribution:

$$P(k) \propto k^{-\gamma}$$ with γ = 3 for the BA model

- self-similar structure with highly connected hubs (no intrinsic length scale)

=> average path length grows with `ln` (N) / `ln`(`ln`(*N*))

=> this grows much slower than for random graphs

=> **“very small world”**

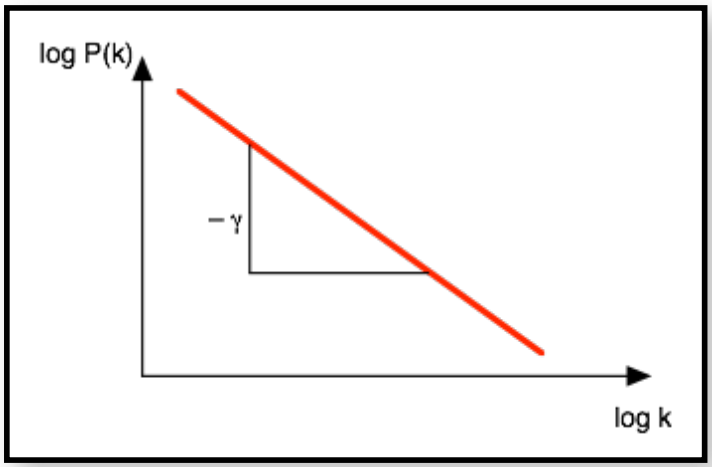
The Power-Law Signature

Power law $$P(k) \propto k^{-\gamma}$$

Take log on both sides:

$$\log(P(k)) = -\gamma \log(k)$$

Plot log(*P*) vs. log(*k*) => straight line



Note: for fitting γ against experimental data it is often better to use the integrated *P(k)*
=> integral smoothes the data

$$\int_{k_0}^{k} P(k)dk = \left[-\frac{k^{-(\gamma-1)}}{\gamma}\right]_{k_0}^{k}$$

Scale-Free: Examples

The World-Wide-Web:

=> growth via links to portal sites

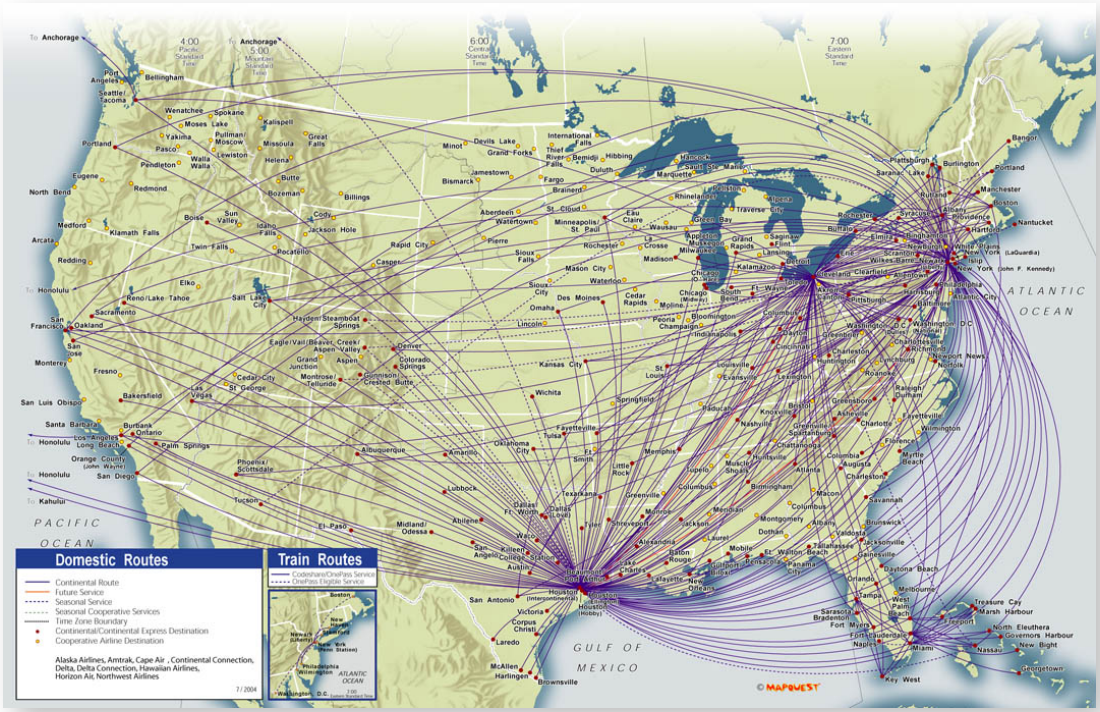
Flight connections between airports

=> large international hubs, small local airports

Protein interaction networks

=> some central,

ubiquitous proteins

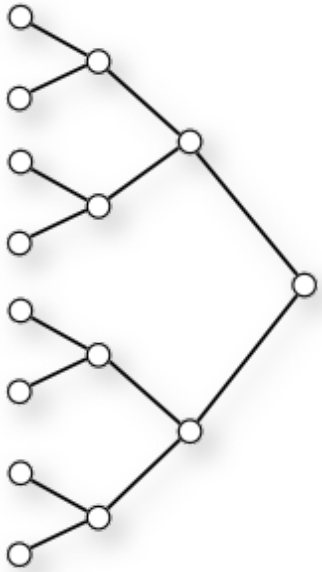


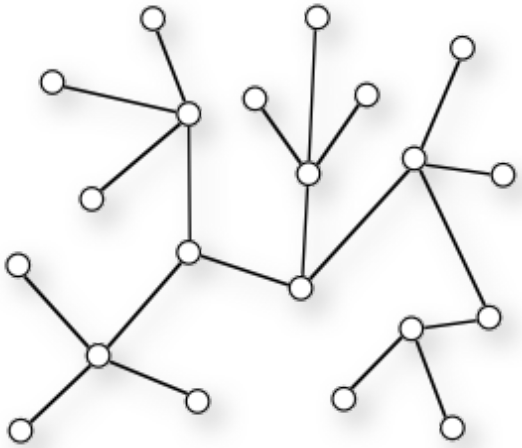
http://a.parsons.edu/~limam240/blogimages/16_full.jpg

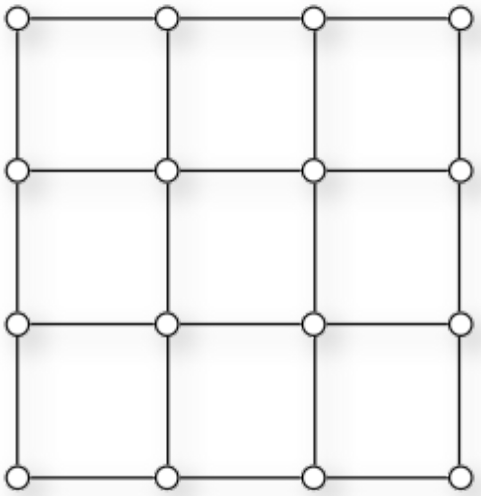
Hierarchical, Regular, Clustered…

Tree-like network with similar degrees
=> like an organigram
=> hierarchic network

All nodes have the same degree
and the same local neighborhood
=> regular network







P(k) for these example networks? (finite size!)

Note: most real-world networks are somewhere in between the basic types

C(k) for a Random Network

Clustering coefficient when *m* edges exist between *k* neighbors

$$C(k,m) = \frac{2m}{k(k-1)}$$

Probability to have exactly *m* edges between the *k* neighbors

$$W(m) = \binom{k}{m} p^m (1-p)^{\frac{k(k-1)}{2}-m}$$

In this way, we pick the *m* start nodes for the *m* edges from the *k* nodes.

Average *C(k)* for degree *k*:

$$C(k) = \frac{\sum_{m=0}^{\frac{k(k-1)}{2}} W(m)\, C(k,m)}{\sum_{m=0}^{\frac{k(k-1)}{2}} W(m)} = \ldots = p$$

→ *C(k)* is independent of *k*
<=> same local connectivity throughout the network

Clusters in scale free graphs

Scale-free network <=> no intrinsic scale

→ same properties at any *k*-level

→ same local connectivity

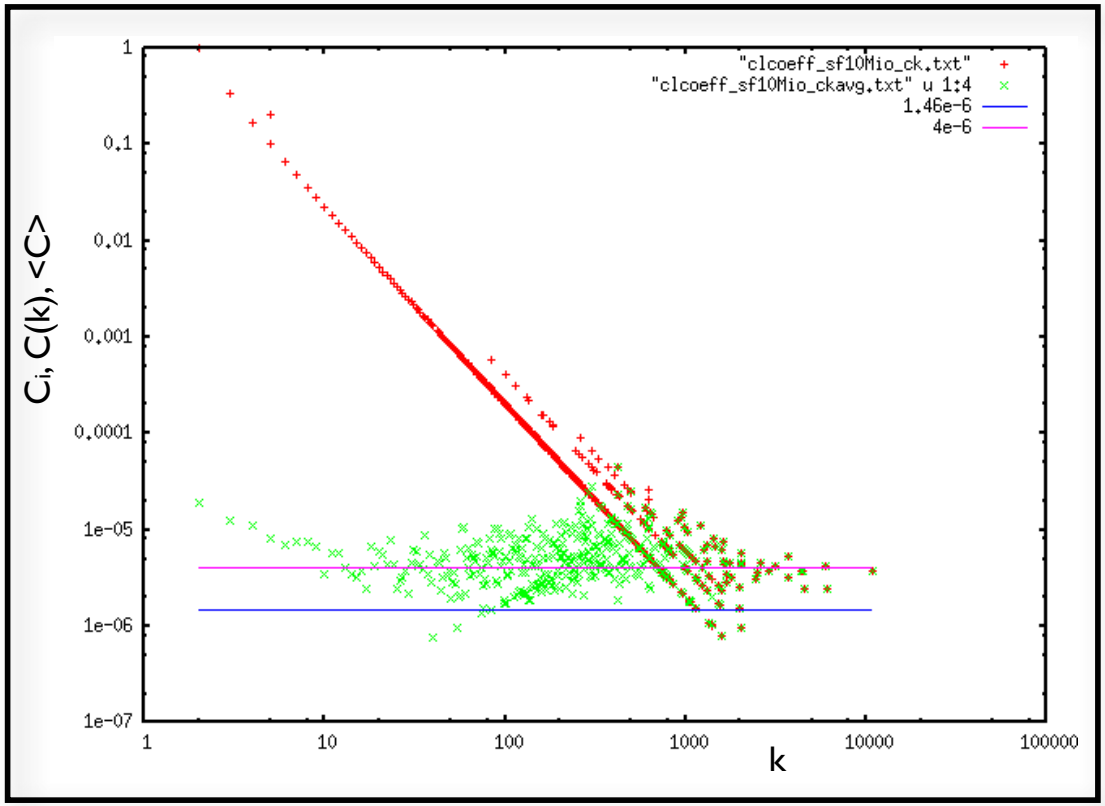
→ *C(k)* = const.

"Real" biological data

→ missing links

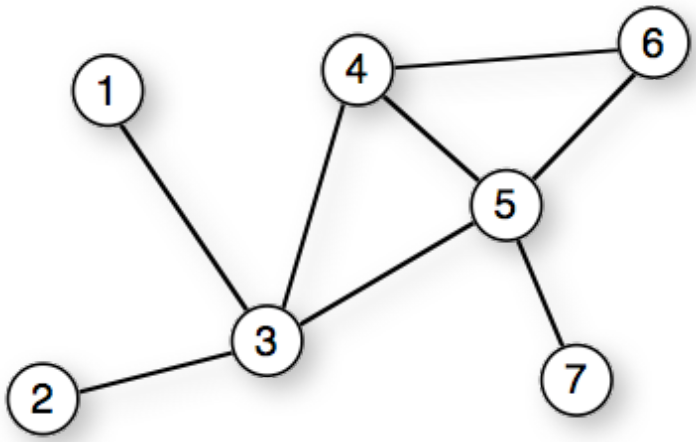
→ multiple clusters

Is the metabolic network of a cell fully connected?



Algorithms on Graphs

How to **represent** a graph in the **computer**?



1. **Adjacency list**

=> list of neighbors for each node

1: (3)
2: (3)
3: (1, 2, 4, 5)
4: (3, 5, 6)
5: (3, 4, 6, 7)
6: (4, 5)
7: (5)

+ minimal memory requirement
+ vertices can easily be added or removed

– requires $O(\lambda)$ time to determine whether a certain edge exists

Note: for weighted graphs store pairs of (neighbor label, edge weight)

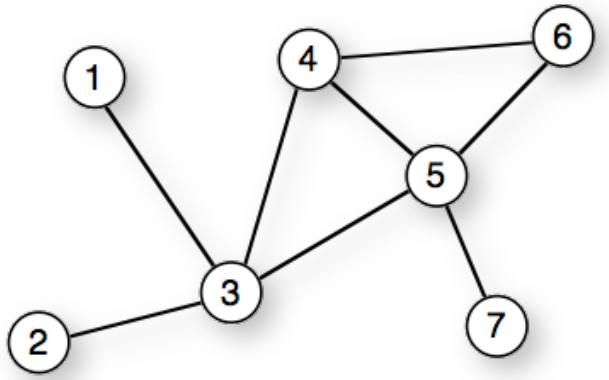
Graph Representation II

2. **Adjacency matrix** (see V1)

→ N x N matrix with entries M_{uv}

M_{uv} = weight when edge between u and v exists,
0 otherwise

→ symmetric for undirected graphs



	1	2	3	4	5	6	7
1	–	0	1	0	0	0	0
2	0	–	1	0	0	0	0
3	1	1	–	1	1	0	0
4	0	0	1	–	1	1	0
5	0	0	1	1	–	1	1
6	0	0	0	1	1	–	0
7	0	0	0	0	1	0	–

+ fast *O(1)* lookup of edges
– large memory requirements
– adding or removing nodes is expensive

Note: very convenient in programming languages that support sparse multi-dimensional arrays
=> Perl

Graph Representation III

3. **Incidence matrix**

→ *N* x *M* matrix with entries M_{nm}

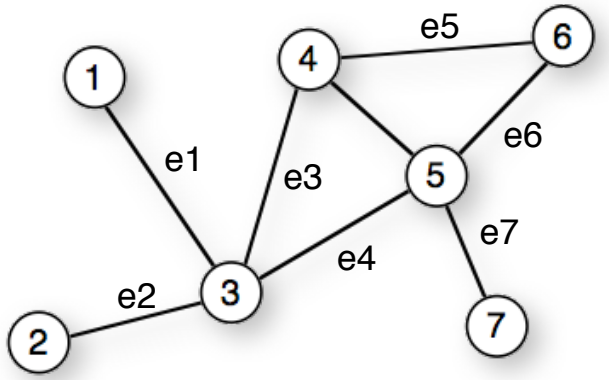
M_{nm} = weight when edge *m* ends at node *n*

0 otherwise

→ for a plain graph there are two entries per column

→ directed graph:
indicate direction via sign (in/out)

The incidence matrix is a special form of the stoichiometric matrix of reaction networks.

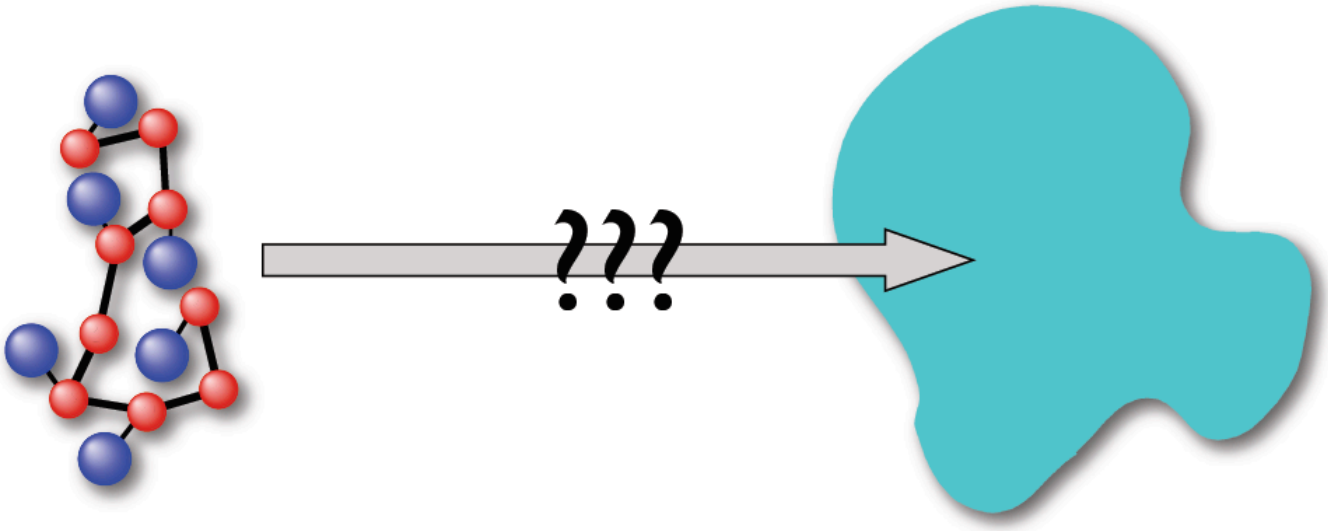


	e1	e2	e3	e4	e5	e6	e7
1	1						
2		1					
3	1	1	1	1			
4			1		1		
5				1		1	1
6					1	1	
7							1

2.4 Fitting atomistic structures into EM maps

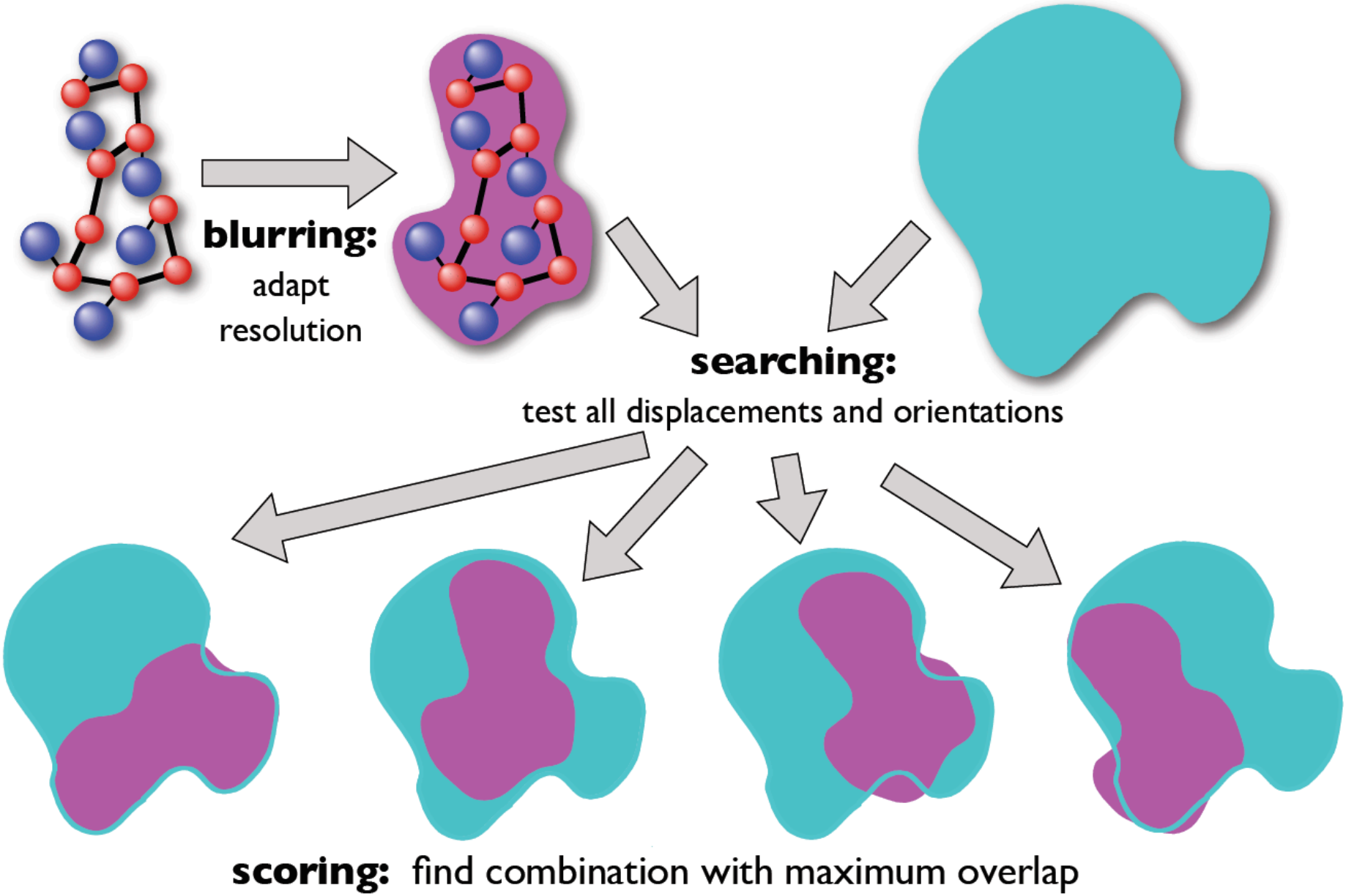
Atomistic structure of
a part of the complex

Coarse EM structure of
the whole complex



- same resolution for both structures
- exhaustive search with scoring
- choose best pose(s)

The procedure



Step 1: blurring the picture

Mathematically:
convolution of (exact) atomistic structure f(x)
with experimental resolution g(x)

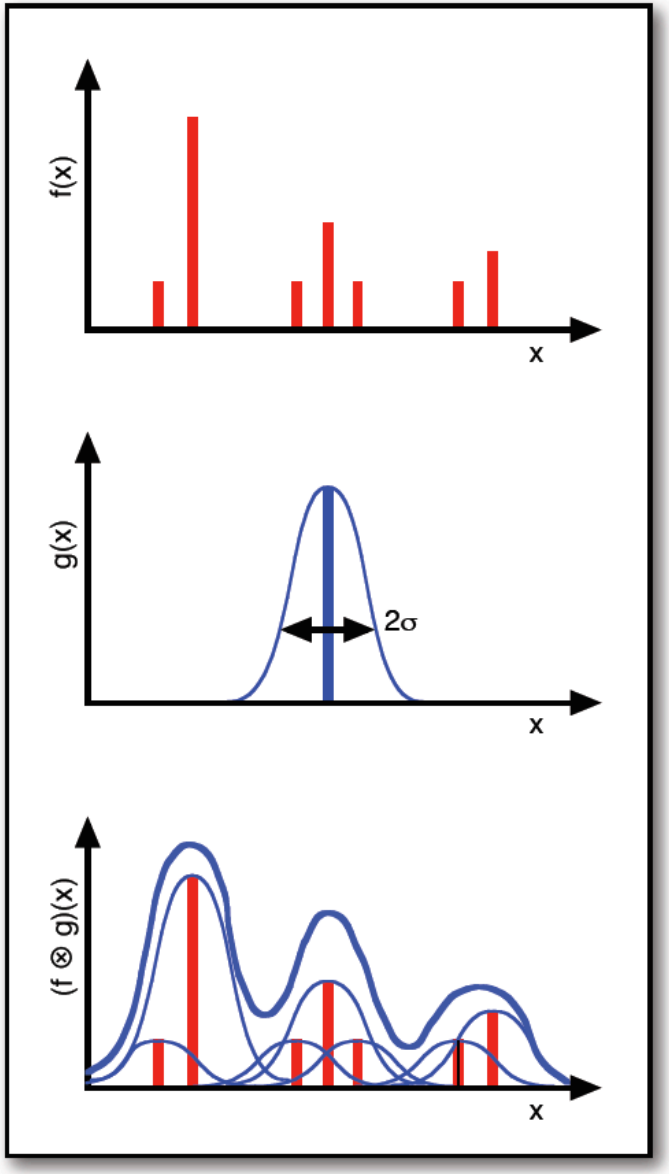
$$\tilde{f}(x) = (f \otimes g)(x) = \int dz\, f(z)\, g(x-z)$$

original data seen through the imaging apparatus

original signal

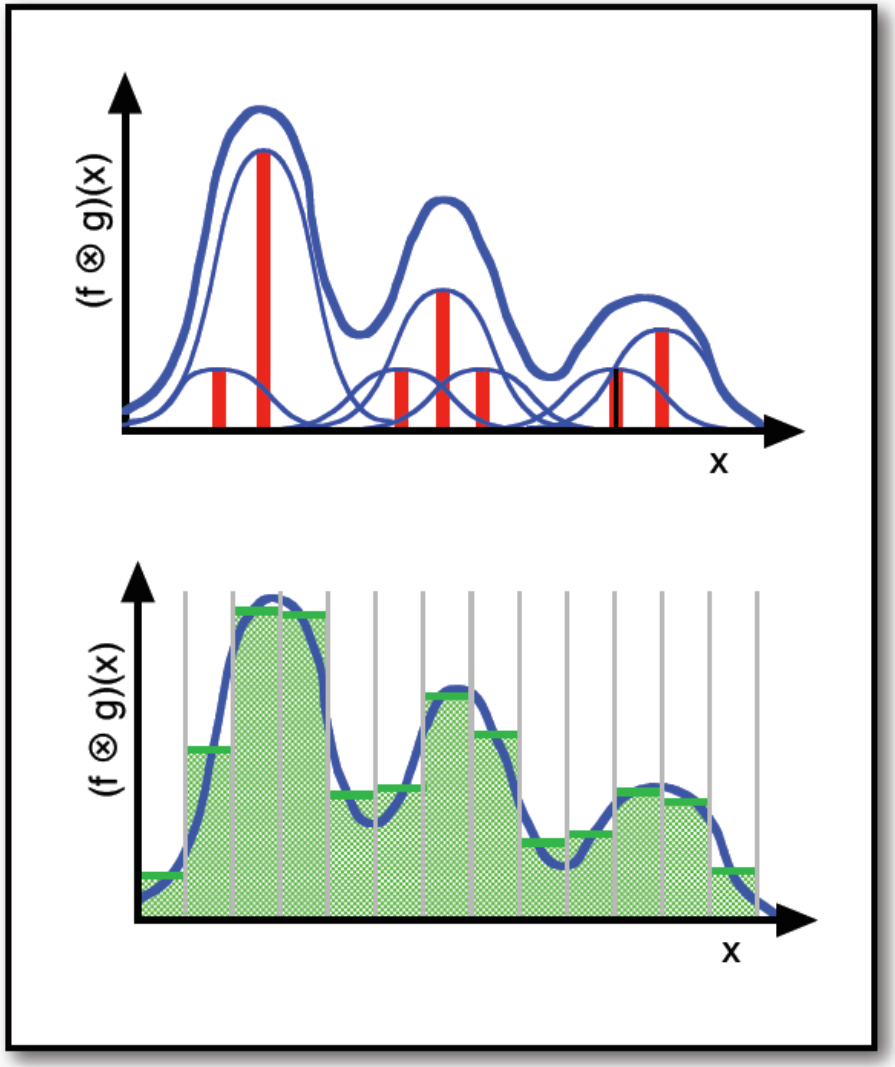
"kernel" => what is the image of a single point? (=delta signal)

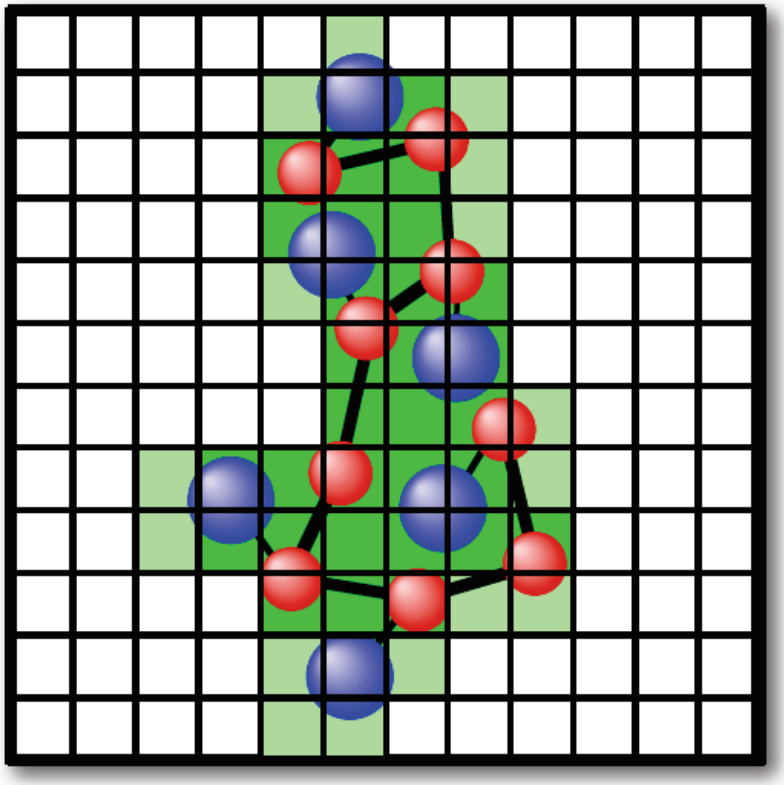
Often: blurring with gaussian



Put it on a grid

Discretize:





2.5 Fourier Transformation

Forward

$$F(k) = \int_{-\infty}^{\infty} dx\, e^{-ikx}\, f(x)$$

and inverse Fourier transform

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} dk\, e^{ikx}\, F(k)$$

with $$e^{ikx} = \cos(kx) + i \sin(kx)$$

=> convert between real and frequency (Fourier) space

short distances <=> high frequencies

long distances <=> low frequencies

Shift of the Argument

$$
\begin{aligned}
FT[f(x+\Delta x)] &= \int_{-\infty}^{\infty} dx\, e^{-ikx}\, f(x+\Delta x) \\
&= \int dy\, e^{-ik(y-\Delta x)}\, f(y) \\
&= e^{ik\,\Delta x} \int dy\, e^{-iky}\, f(y) \\
&= e^{ik\,\Delta x}\, FT[f(x)]
\end{aligned}
$$

Variable transformation:
$y = x + \Delta x$

change name of integration variable back from y to x

Convolution

$$\tilde{f}(x) = (f \otimes g)(x) = \int dz\, f(z)\, g(x-z)$$

Apply FT on both sides:

$$FT[\tilde{f}(x)] = FT[(f \otimes g)(x)] =$$

$$=$$

$$=$$

$$= FT[f(x)]\ FT[g(x)]$$

Integration in real space is replaced by simple multiplication in Fourier space.

But FTs need to be computed.

What is more efficient?

If the same width g(x) is used for multiple displaced datasets
=> do FT[g(x)] only once

Fourier on a Grid

On a finite grid: => maximum wavelength = length of grid
=> minimum wavelength = grid spacing
=> sum instead of integral

$$F_k = \sum_{j=0}^{N-1} e^{-2i\pi\, j\, k/N}\, f_j \qquad f_k = \frac{1}{N} \sum_{j=0}^{N-1} e^{+2i\pi\, j\, k/N}\, F_j$$

2.5.5 FFT by Danielson and Lanczos (1942)

Danielson and Lanczos showed that a discrete Fourier transform of length *N* can be rewritten as the sum of two discrete Fourier transforms, each of length *N/2.*

One of the two is formed from the even-numbered points of the original *N*, the other from the odd-numbered points.

F_k^e : *k*-th component of the Fourier transform of length *N/2* formed from the even components of the original f_j 's

F_k^o : *k*-th component of the Fourier transform of length *N/2* formed from the odd components of the original f_j 's

$$
\begin{aligned}
F_k &= \sum_{j=0}^{N-1} e^{-2\pi i k \frac{j}{N}} f_j \\
&= \sum_{j=0}^{\frac{N}{2}-1} e^{-2\pi i k \frac{2j}{N}} f_{2j} + \sum_{j=0}^{\frac{N}{2}-1} e^{-2\pi i k \frac{2j+1}{N}} f_{2j+1} \\
&= \sum_{j=0}^{\frac{N}{2}-1} e^{-2\pi i k \frac{j}{\frac{N}{2}}} f_{2j} + e^{-2\pi i k \frac{1}{N}} \sum_{j=0}^{\frac{N}{2}-1} e^{-2\pi i k \frac{j}{\frac{N}{2}}} f_{2j+1} \\
&= F_k^e + e^{-2\pi i k \frac{1}{N}} F_k^o
\end{aligned}
$$

FFT by Danielson and Lanczos (1942)

The wonderful property of the Danielson-Lanczos-Lemma is that it can be used recursively.

Having reduced the problem of computing F_k to that of computing F_k^e and F_k^o , we can do the same reduction of F_k^e to the problem of computing the transform of **its** *N/4* even-numbered input data and *N/4* odd-numbered data.

We can continue applying the DL-Lemma until we have subdivided the data all the way down to transforms of length 1.

What is the Fourier transform of length one? It is just the identity operation that copies its one input number into its one output slot.

For every pattern of $\log_2 N$ e‘s and o‘s, there is a one-point transform that is just one of the input numbers f_n

$$F_k^{eoeeoeo\ldots oee} = f_n \qquad \text{for some } n$$

FFT by Danielson and Lanczos (1942)

The next trick is to figure out which value of *n* corresponds to which pattern of e's and o's in $F_k^{eoeeoeo\ldots oee} = f_n$

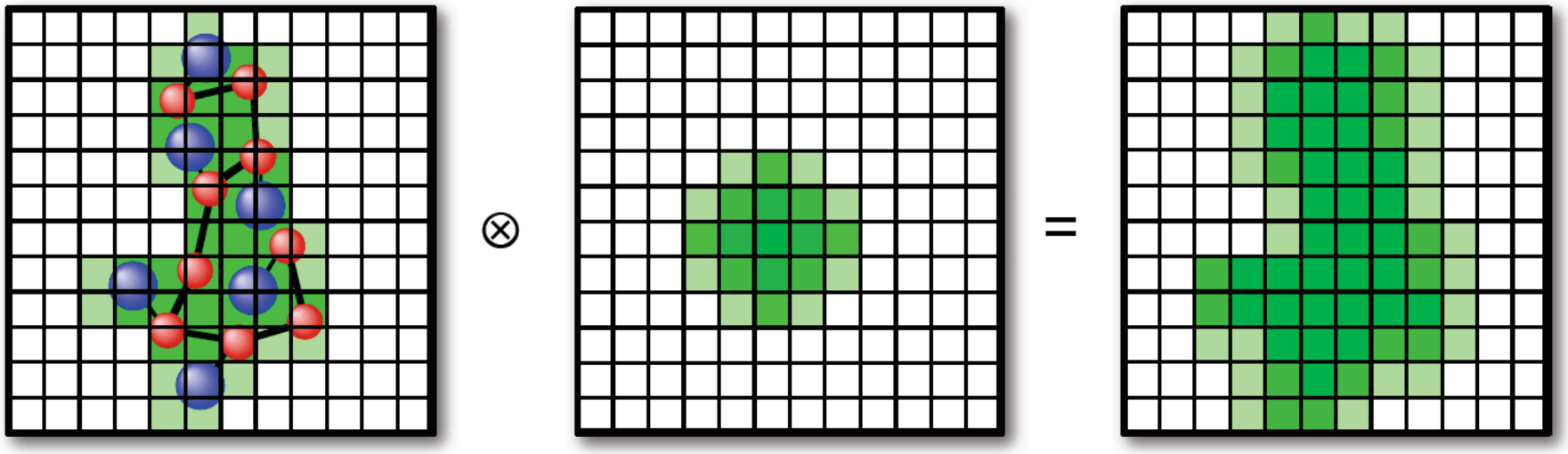
Answer: reverse the pattern of e's and o's, then let e = 0 and o = 1,
and you will have, in binary the value of n.

This works because the successive subdividisions of the data into even and odd are tests of successive low-order (least significant) bits of *n*.

Thus, computing a FFT can be done efficiently in O(N log(N)) time.

Discretization and Convolution

For practical applications:
=> first put atomistic data onto the grid, then blur with FFT



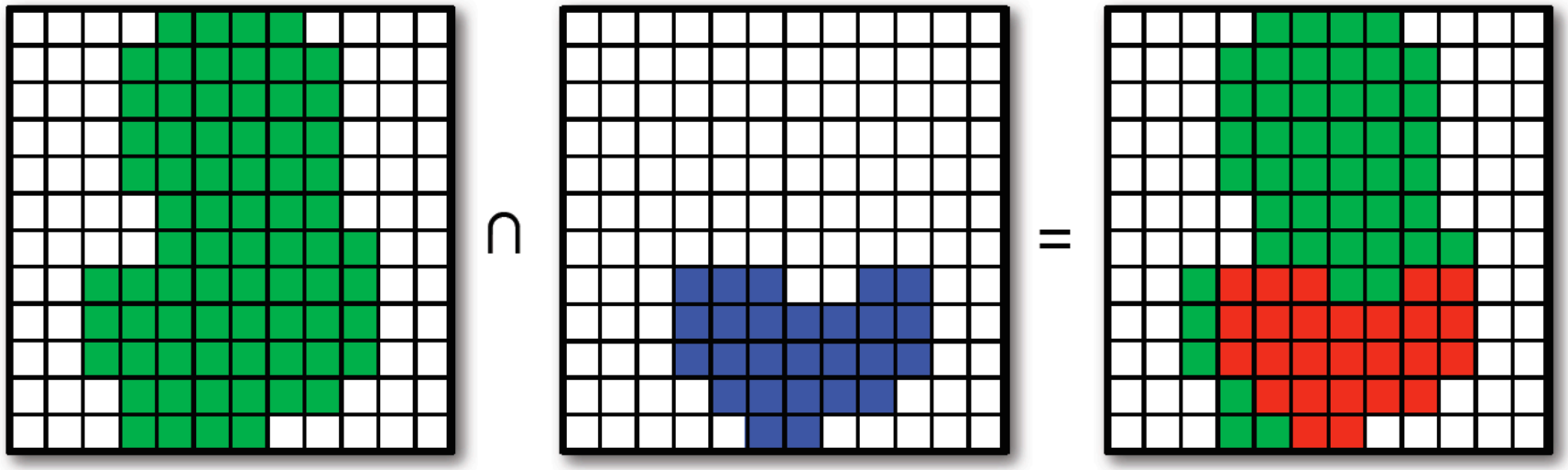
discretized hi-res data ⊗ blurring kernel = low-res image

Step 3: Scoring the Overlap

Most simple case:

- apply density threshold and count overlapping voxels
- displace images relative to each other, recount

=> find displacement with maximum overlap



In matrix form with displacements x, y:

$$c(x,y) = \sum_{l=1}^{N} \sum_{m=1}^{N} a_{l,m}\, b_{l+x,m+y}$$

Cross Correlation

Generalization: maximize cross correlation of grided densities with respect to displacement (and orientation)

$$C_{x,y,z} = \sum_{l=1}^{N} \sum_{m=1}^{N} \sum_{n=1}^{N} a_{l,m,n} \times b_{l+x,m+y,n+z}$$

Note: maximize the cross-correlation <=> minimize the squared difference

On a grid with N^3 gridpoints => N^3 possible displacements
=> runtime $O(N^6)$

Further complication: the convolution

$$C_{x,y,z} = \sum_{l=1}^{N} \sum_{m=1}^{N} \sum_{n=1}^{N} a_{l,m,n} \times (g \otimes b_{l+x,m+y,n+z})$$

Correlation and Fourier

Apply Fourier transformation to both sides of

$$C_{x,y,z} = \sum_{l=1}^{N} \sum_{m=1}^{N} \sum_{n=1}^{N} a_{l,m,n} \times b_{l+x,m+y,n+z}$$

=> matrix multiplication

$$FT[C] = FT[A]^* \times FT[B]$$

Runtime of 3D FFT = $O(N^3 \log^3(N)) << N^6$
=> all possible displacements tested simultaneously

Note: FT[A] only calculated once initially
=> two FFTs per orientation
=> scan orientation via Euler angles

<== Step 2: exhaustive search

Include convolution

Maximize

$$C_{x,y,z} = \sum_{l=1}^{N} \sum_{m=1}^{N} \sum_{n=1}^{N} a_{l,m,n} \times (g \otimes b_{l+x,m+y,n+z})$$

In Fourier space:

Insert convolution $FT[f \otimes g] = FT[f] \times FT[g]$

Into correlation:

$$\begin{aligned} FT[C] &= FT[A]^* \times FT[G \otimes B] \\ &= FT[A]^* \times (FT[G] \times FT[B]) \\ &= (FT[A]^* \times FT[G]) \times FT[B] \end{aligned}$$

can be precomputed

2 FFTs + 1 matrix multiplication

2.7 Katchalski-Kazir algorithm

Proc. Natl. Acad. Sci. USA
Vol. 89, pp. 2195–2199, March 1992
Biophysics

Molecular surface recognition: Determination of geometric fit between proteins and their ligands by correlation techniques

(protein–protein interaction/surface complementarity/macromolecular complex prediction/molecular docking)

EPHRAIM KATCHALSKI-KATZIR[†‡], ISAAC SHARIV[§], MIRIAM EISENSTEIN[¶], ASHER A. FRIESEM[§], CLAUDE AFLALO[‖], AND ILYA A. VAKSER[†]

Departments of [†]Membrane Research and Biophysics, [§]Electronics, [¶]Structural Biology, and [‖]Biochemistry, Weizmann Institute of Science, Rehovot 76100, Israel

Contributed by Ephraim Katchalski-Katzir, October 24, 1991

Developed for protein-ligand docking
<=> same techniques applicable for docking "on the inside"

Discretization for docking

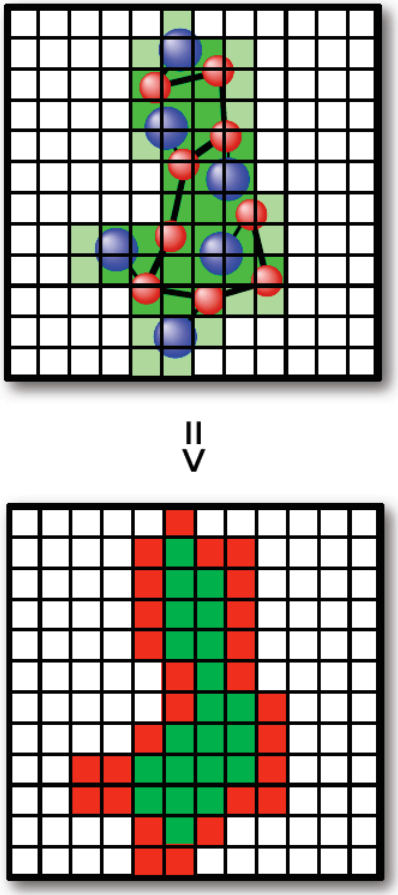
Next, to distinguish between the surface and the interior of each molecule, we retain the value of 1 for the grid points along a thin surface layer only and assign other values to the internal grid points. The resulting functions thus become

$$\bar{a}_{l,m,n} = \begin{cases} 1 & \text{on the surface of the molecule} \\ \rho & \text{inside the molecule} \\ 0 & \text{outside the molecule,} \end{cases} \qquad [2a]$$

and

$$\bar{b}_{l,m,n} = \begin{cases} 1 & \text{on the surface of the molecule} \\ \delta & \text{inside the molecule} \\ 0 & \text{outside the molecule,} \end{cases} \qquad [2b]$$

where the surface is defined here as a boundary layer of finite width between the inside and the outside of the molecule. The parameters ρ and δ describe the value of the points inside the molecules, and all points outside are set to zero. Two-

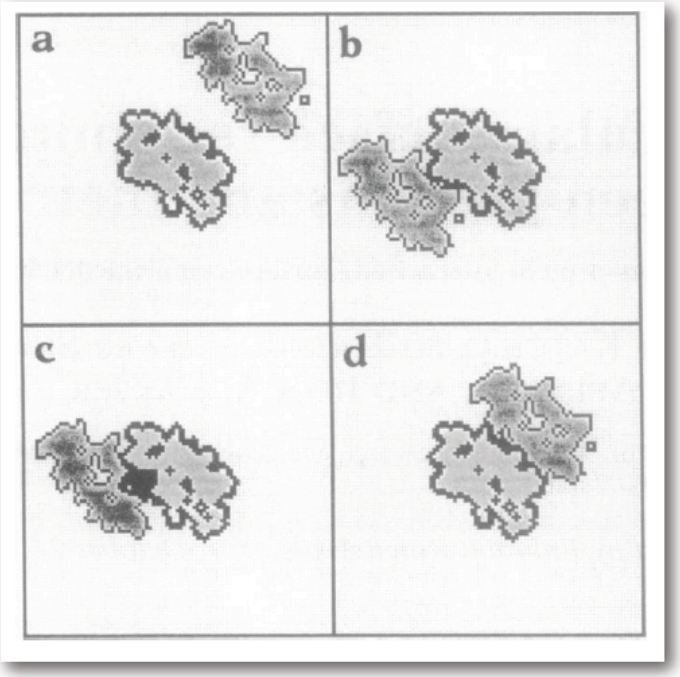


Typical values: $\rho = -15$, $\delta = 1$

=> penalty for overlap of volumes

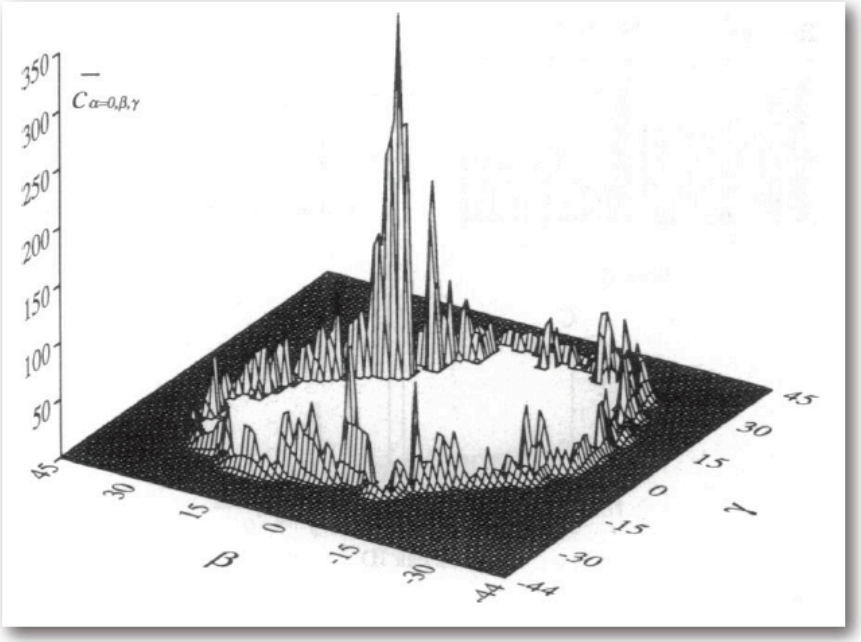
Docking the hemoglobin dimer

2D cross sections at I = 46 (N = 90)



a) no contact
b) limited contact
c) overlap (black area)
d) good geometric match

Correlation at α = 0



highest peak corresponds to native dimer arrangement

The algorithm

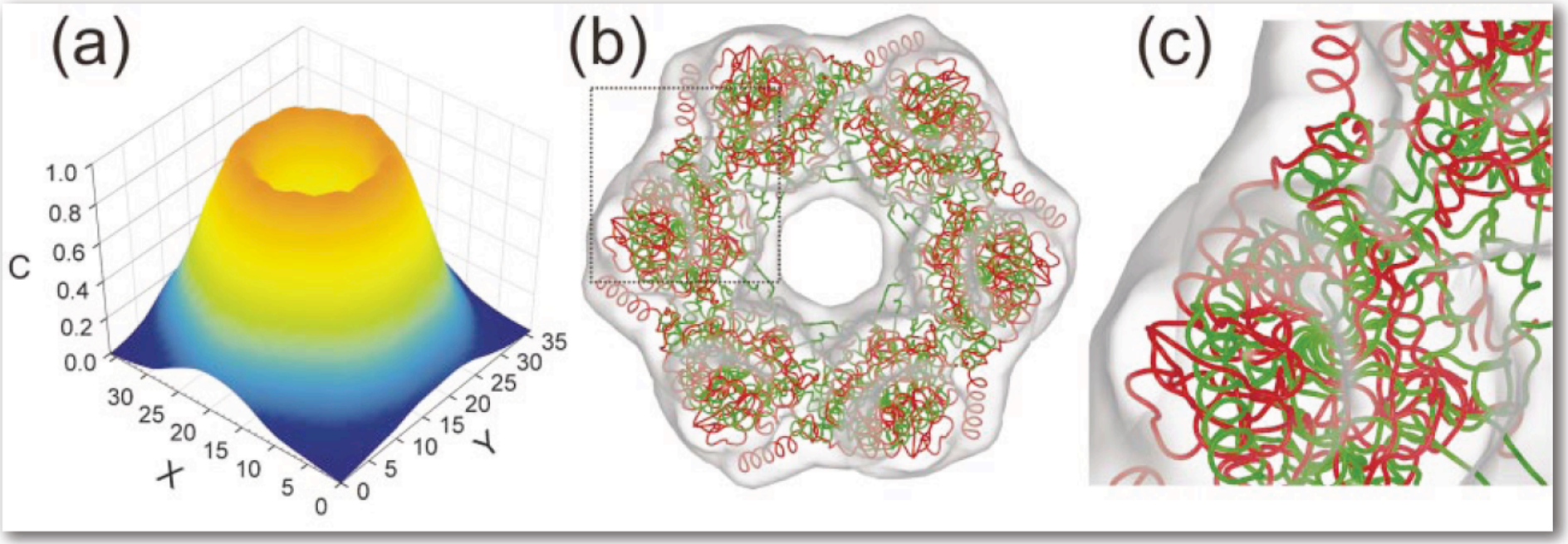
The entire procedure described above can be summarized by the following steps:
(*i*) derive $\bar{a}$ from atomic coordinates of molecule **a** (Eq. **2**),
(*ii*) $A^* = [\mathrm{DFT}(\bar{a})]^*$ (Eq. **4**),
(*iii*) derive $\bar{b}$ from atomic coordinates of molecule **b** (Eq. **2**),
(*iv*) $B = \mathrm{DFT}(\bar{b})$ (Eq. **4**),
(*v*) $C = A^* \cdot B$ (Eq. **5**),
(*vi*) $\bar{c} = \mathrm{IFT}(C)$ (Eq. **6**),
(*vii*) look for a sharp positive peak of $\bar{c}$,
(*viii*) rotate molecule **b** to a new orientation,
(*ix*) repeat steps *iii–viii* and end when the orientations scan is completed, and
(*x*) sort all of the peaks by their height.

Each high and sharp peak found by this procedure indicates geometric match and thus represents a potential complex. The relative position and orientation of the molecules within each such complex can readily be derived from the

Katchalski-Kazir et al. 1992

Algorithm has become a workhorse for docking and density fitting.

Problem I: limited contrast



Docking of the RecA helicase monomer into simulated EM density of the hexamer at 15 Å resolution
(exhaustive 6D search with 5Å / 9° steps plus off-lattice optimization)
=> multiple fits with similar correlations

Chacón, Wriggers, *J. Mol. Biol.* **317** (2002) 375

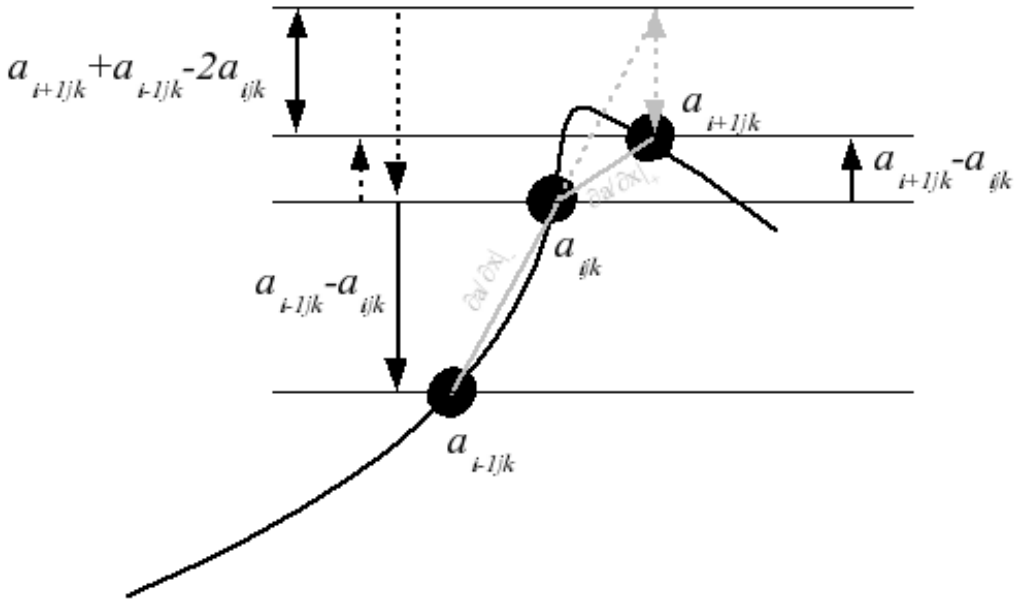
2.6 Laplace filter

Evaluate $$\nabla^2 = \frac{d^2}{dx^2} + \frac{d^2}{dy^2} + \frac{d^2}{dz^2}$$

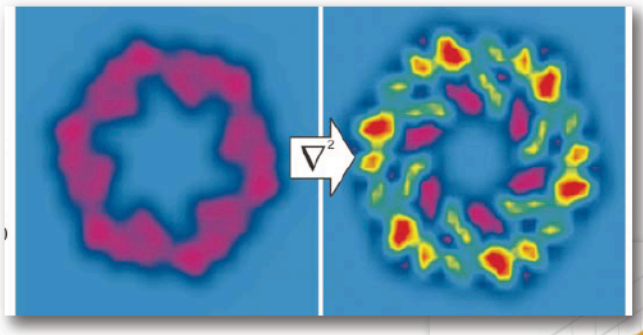
on a grid: $$\nabla^2 a_{l,m,n} = -6a_{l,m,n} + a_{l+1,m,n} + a_{l-1,m,n} + a_{l,m+1,n} + a_{l,m-1,n} + a_{l,m,n+1} + a_{l,m,n-1}$$

Correlation:

$$C_{x,y,z} = \sum_{l=1}^{N} \sum_{m=1}^{N} \sum_{n=1}^{N} (\nabla^2 \otimes a_{l,m,n}) \times (\nabla^2 \otimes g \otimes b_{l+x,m+y,n+z})$$

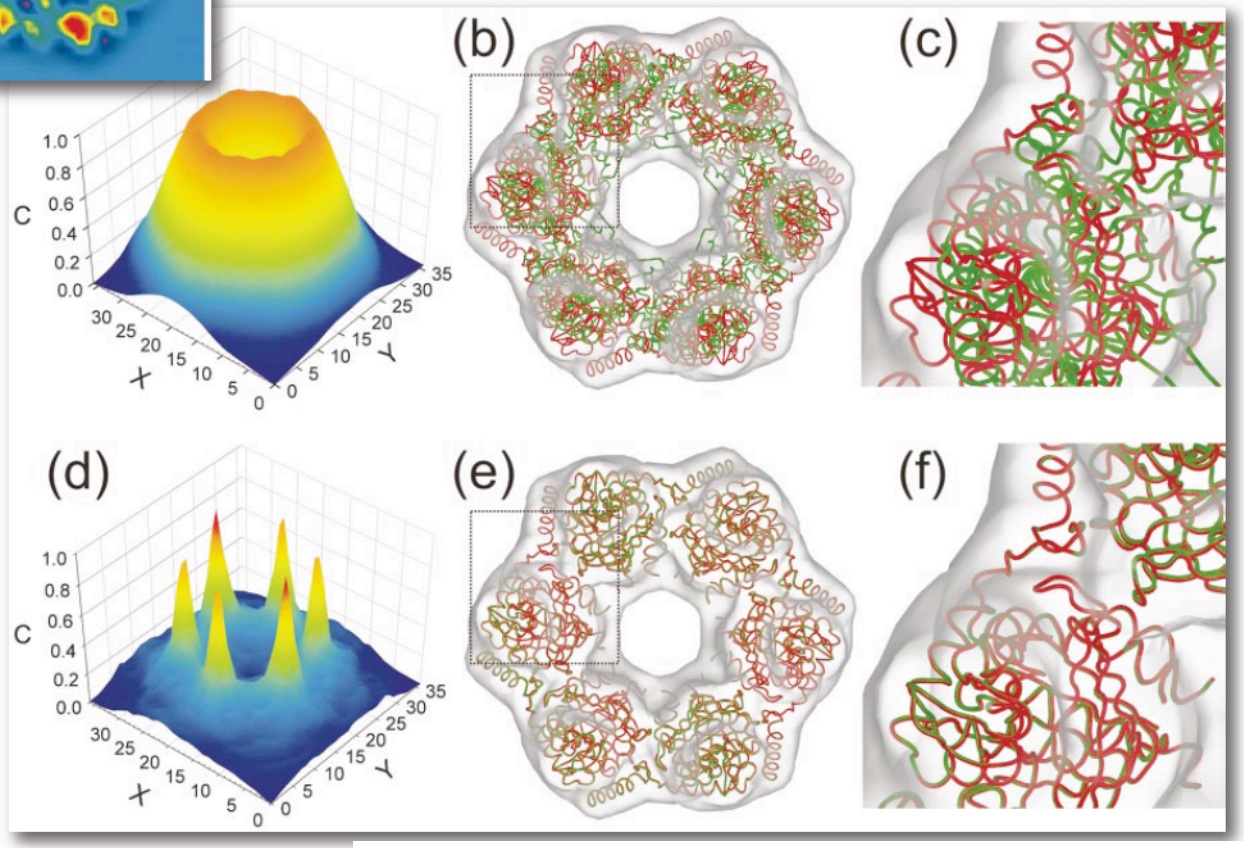


Enhanced contrast ➔ better fit



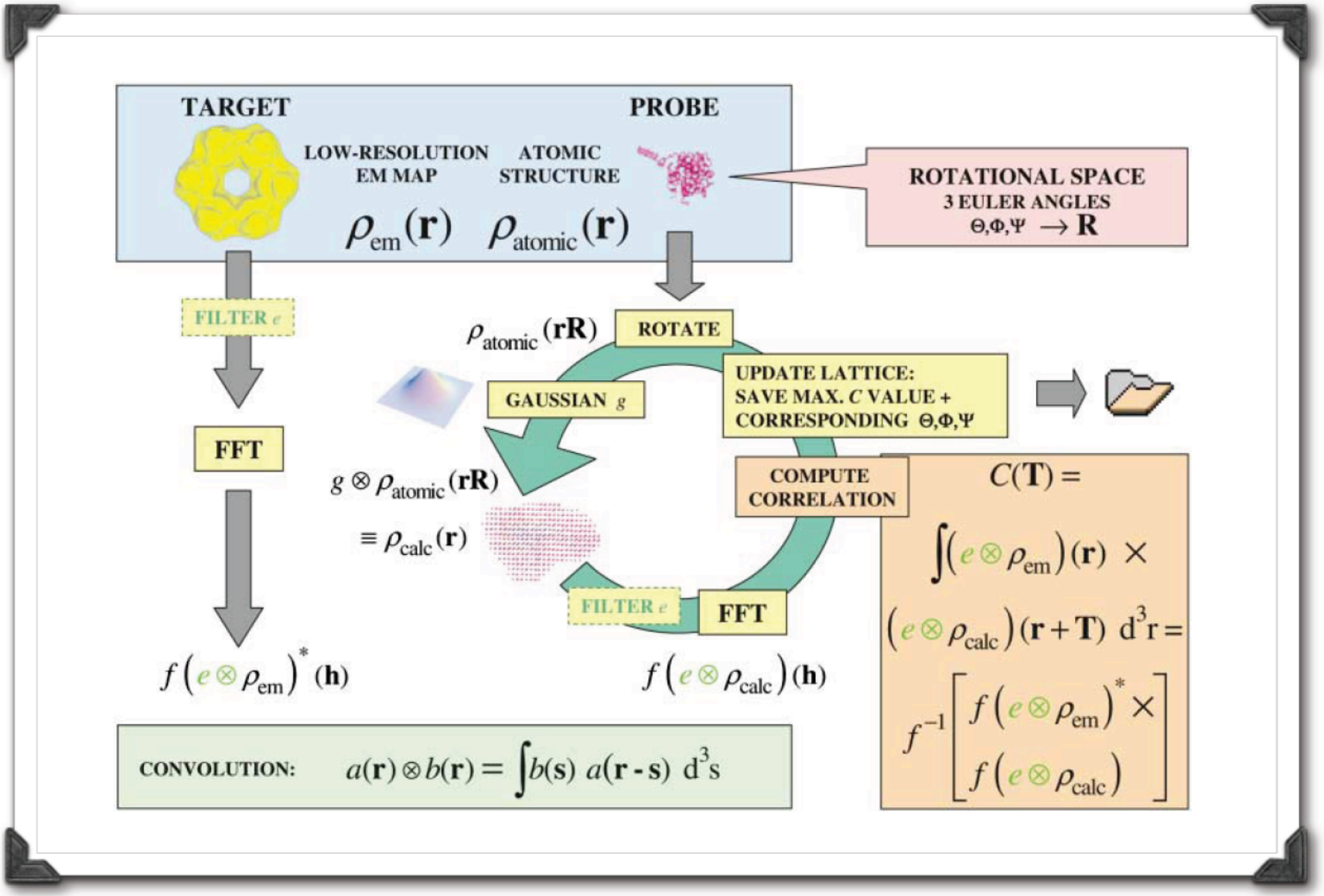
With the density alone:

With the Laplacian filter:



Chacón, Wriggers, *J. Mol. Biol.* **317** (2002) 375

The big picture



Wriggers, Chacón, *Structure* **9** (2001) 779