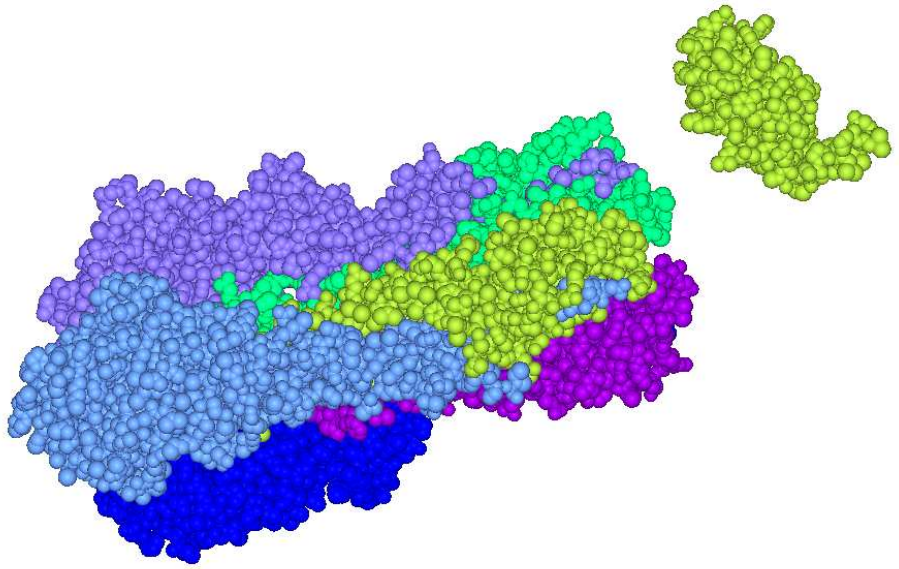
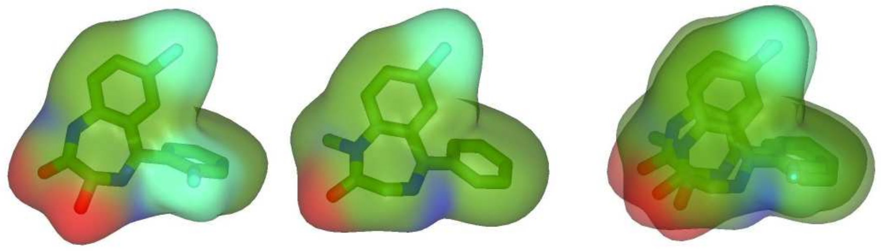
Protein Docking and 3D Ligand-Based Virtual Screening

Part 1





Dave Ritchie

Orpailleur Team

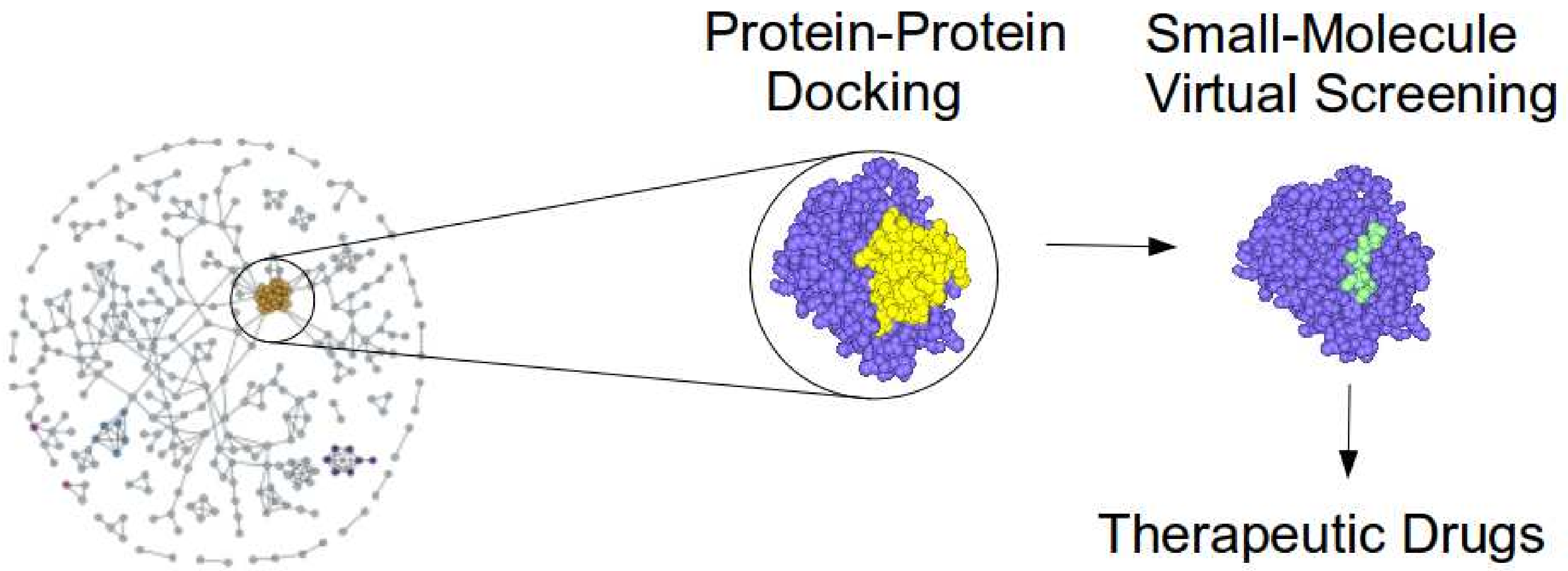
INRIA Nancy – Grand Est

Schedule

- **Lecture 1 – Rigid Body Protein Docking**
  - Introduction / Motivation
  - Protein Docking and the CAPRI Blind Docking Experiment
  - The “Hex” Spherical Polar Fourier Correlation Algorithm
  - Ultra-Fast Docking Using Graphics Processors (+ some GPU programming)
- **Lecture 2 – New Developments in Protein Docking and Virtual Screening**
  - Simulating Protein Flexibility During Docking
  - Data-Driven and Knowledge-Based Docking
  - Multi-Component Assembly and Cross-Docking
  - Shape-Based Virtual Screening – ROCS, ParaSurf, ParaFit
- **Lecture 3 – Spherical Harmonic Virtual Screening**
  - Case Study – HIV Entry Inhibitors for the CXCR4 and CCR5 Receptors
  - Recent Work – Detecting Polypharmacology Using Gaussian Ensemble Screening

Protein-Protein Interactions and Therapeutic Drug Molecules

- Protein-protein interactions (PPIs) define the machinery of life
- Humans have about 30,000 proteins, each having about 5 PPIs



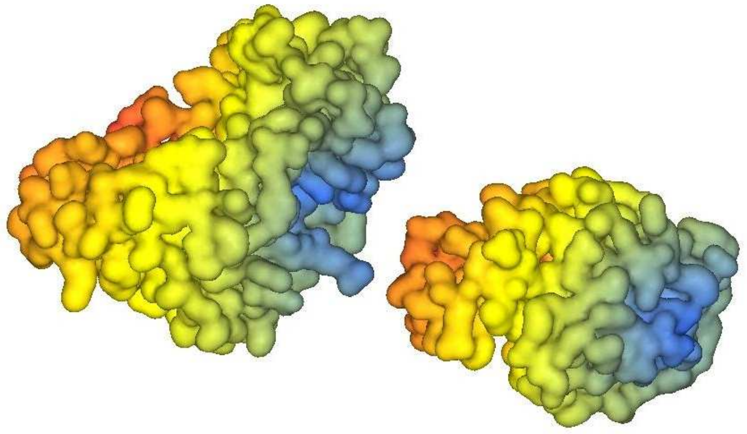
- Understanding PPIs could lead to immense scientific advances
- Small "drug" molecules often inhibit or interfere with PPIs

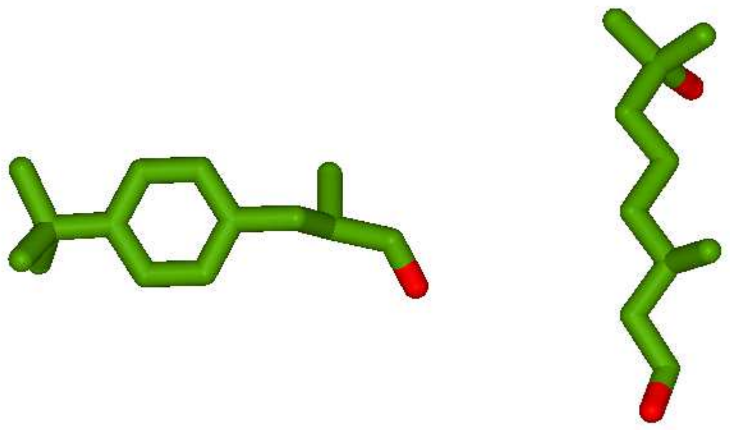
Grosdidier et al. (2009) Advances & Applications in Bioinformatics & Chemistry, 2, 101–123

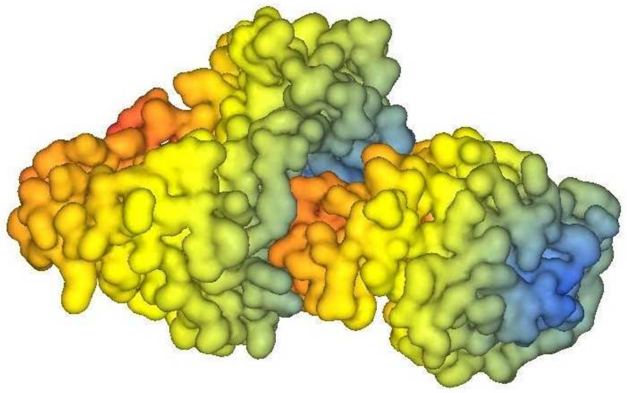
Pujol et al. (2009) Trends in Pharmaceutical Science, 31, 115–123

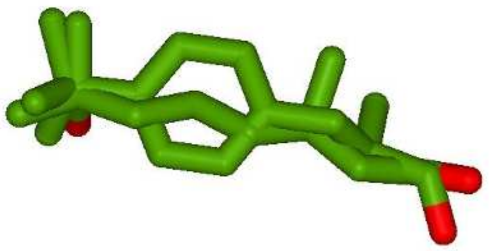
Docking and Shape Matching are Both Recognition Problems

- Ignoring flexibility, docking and shape matching are both 6D search problems





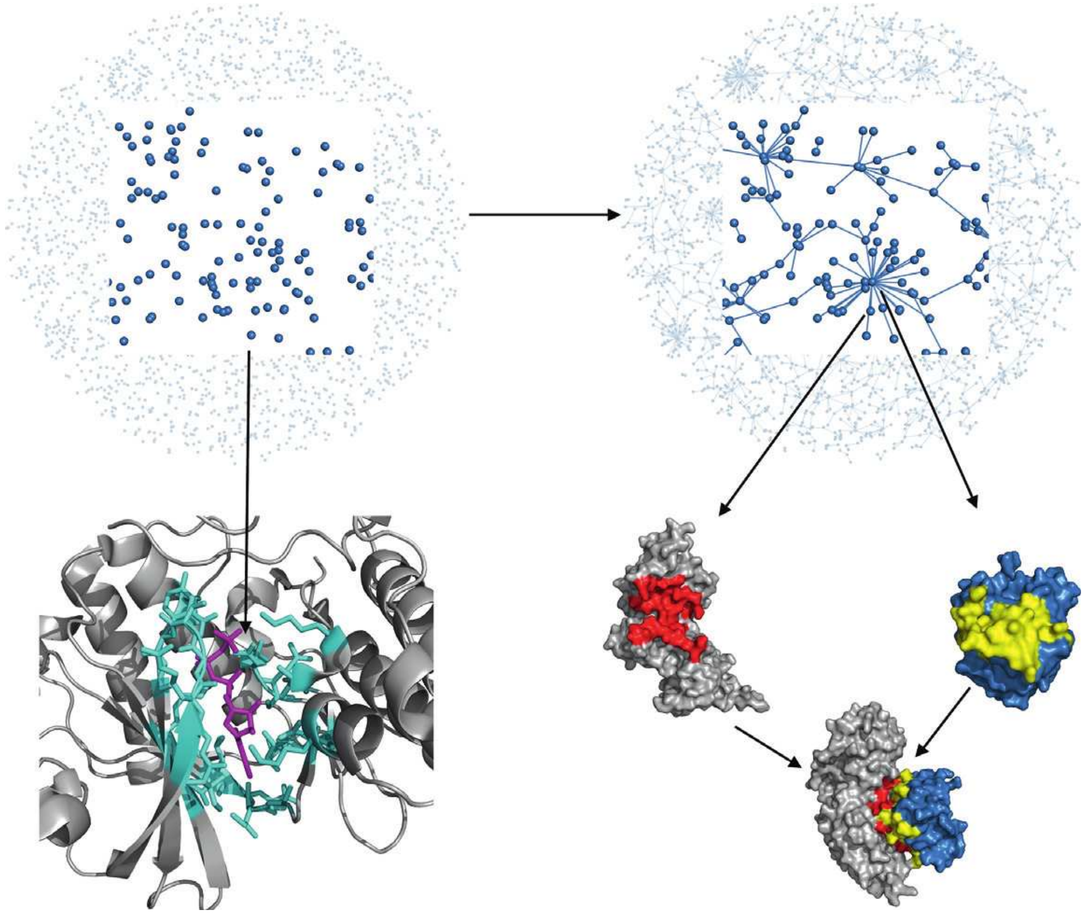




- The challenge – find computationally efficient representations for:
  - protein docking $\leftrightarrow$ translational + rotational search
  - ligand shape matching $\leftrightarrow$ mainly rotational search

Protein-Protein Interaction Challenges

- Can we predict the interactions within a proteome – i.e. predict the interactome ?

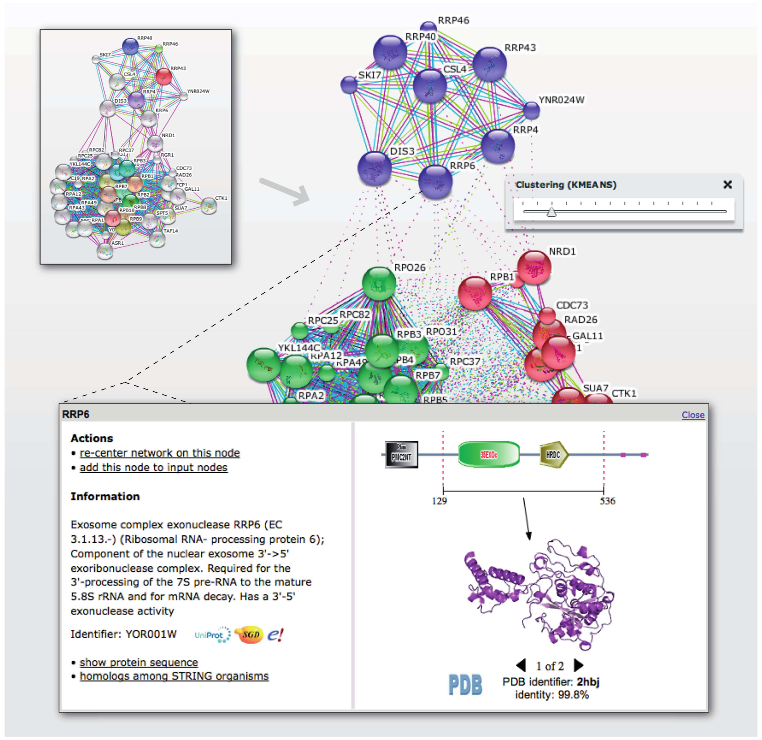


- For each interaction, can we predict the interface surfaces and the 3D complex ?
- For each protein can we predict its ligand binding sites ?

Wass, David, Sternberg (2011) Current Opinion in Structural Biology, 21, 382–390

Protein-Protein Interaction Resources

- STRING – Search Tool for Retrieval of Interacting Genes – http://string.embl.de
  - 12 million known PPIs; 44 million predicted
- 3DID – 3D Interacting Domains – http://3did.irbbarcelona.org
  - 160,000 3D domain-domain interactions (DDIs)



RRP6 Close

Actions
- re-center network on this node
- add this node to input nodes

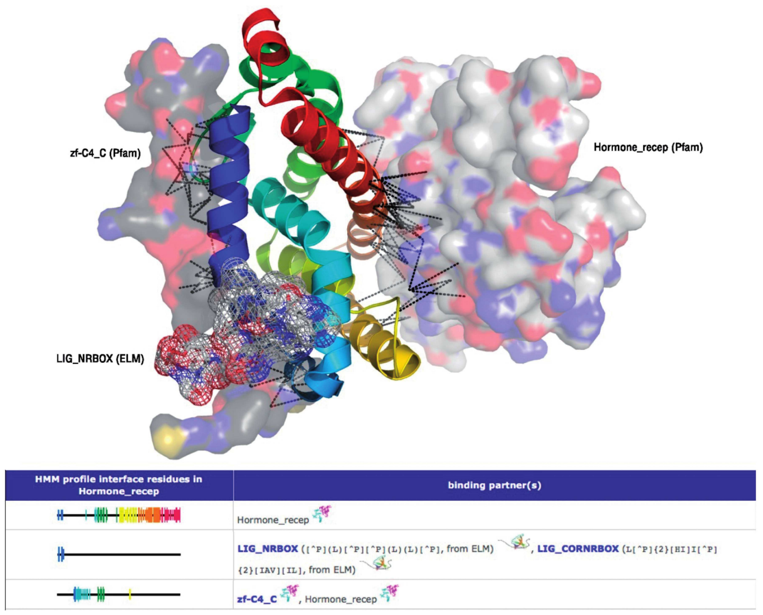
Information

Exosome complex exonuclease RRP6 (EC 3.1.13.-) (Ribosomal RNA- processing protein 6); Component of the nuclear exosome 3'->5' exoribonuclease complex. Required for the 3'-processing of the 7S pre-RNA to the mature 5.8S rRNA and for mRNA decay. Has a 3'-5' exonuclease activity

Identifier: YOR001W

- show protein sequence
- homologs among STRING organisms

1 of 2
PDB identifier: **2hbj**
identity: 99.8%



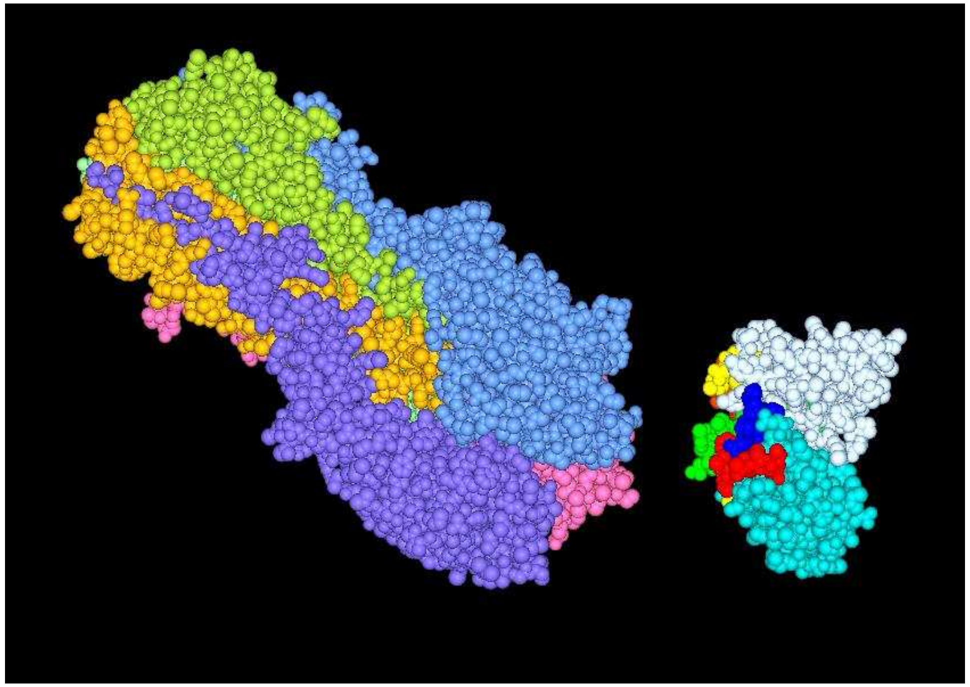
HMM profile interface residues in Hormone_recep	binding partner(s)
	Hormone_recep
	LIG_NRBOX ([^P](L)[^P][^P](L)(L)[^P], from ELM), **LIG_CORNRBOX** (L[^P]{2}[HI]I[^P]{2}[IAV][IL], from ELM)
	zf-C4_C, Hormone_recep

Stein et al. (2010) Nucleic Acids Research, 33, D413–D417 (3DID)

Szklarzyk et al. (2011) Nucleic Acids Research, 39, D561–D568 (STRING)

What is Protein Docking and Why is Docking Difficult ?

- Protein docking = predicting protein interactions at the molecular level



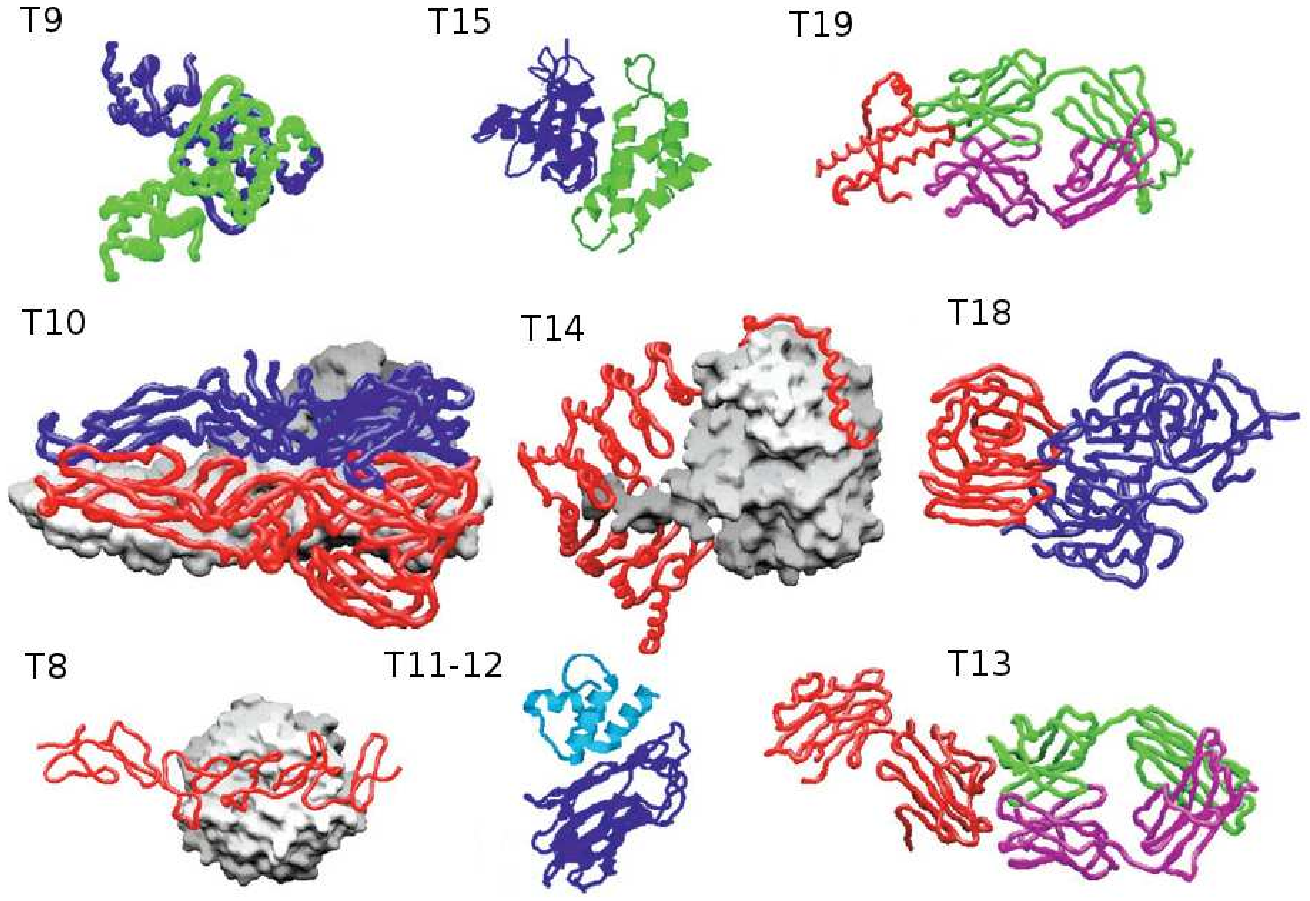
- If proteins are rigid => six-dimensional search space
- But proteins are flexible => multi-dimensional space!
- Modeling protein-protein interactions accurately is difficult!

Halperin et al. (2002), Proteins, 47, 409–443

Ritchie (2008), Current Protein & Peptide Science, 9, 1–15

The CAPRI Blind Docking Experiment

- Critical Assessment of PRedicted Interactions – http://www.ebi.ac.uk/msd-srv/capri/
- Given the unbound structure, particiants have to predict the unpublished 3D complex



T8 = nidogen/laminin

T9 = LiCT dimer

T10 = TEV trimer

T11-12 = cohesin/dockerin

T13 = Fab/SAG1

T14 = PP1δ/MYPT1

T15 = colicin/ImmD

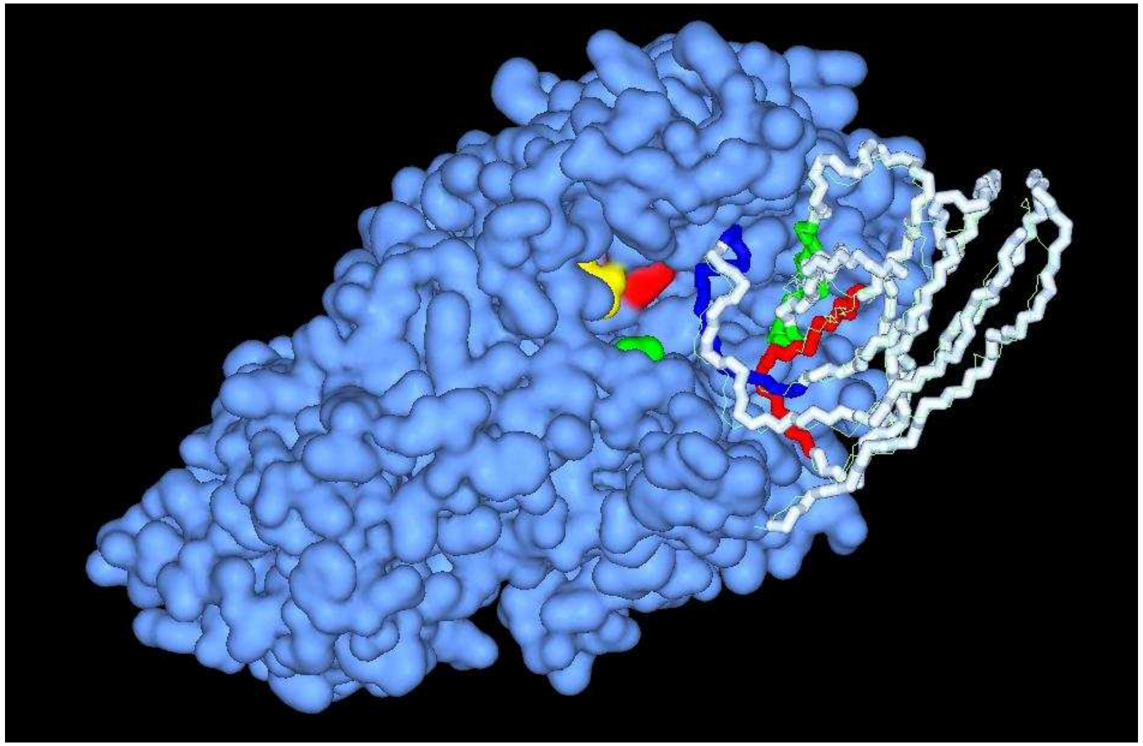
T18 = Xylanase/TAXI

T19 = Fab/bovine prion

Janin (2005) Proteins, 60, 170–175

CAPRI Target T6 Was A Relatively Easy Target

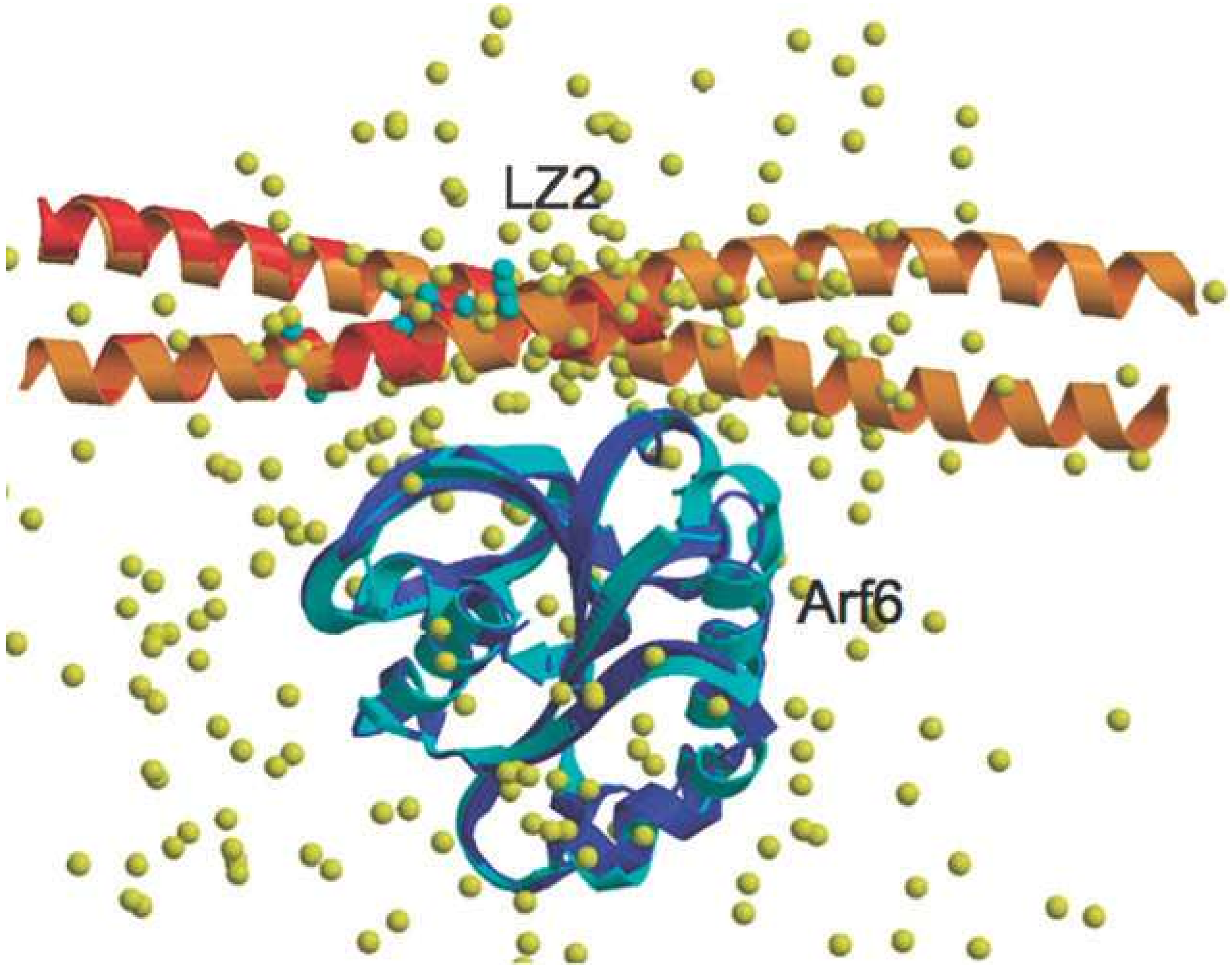
- **Amylase / AMD9 showed little difference between unbound & bound conformations**
- **It also had a classic binding mode, with antibody loops blocking the enzyme active site**



- **Several CAPRI predictors made "high accuracy" models (Ligand RMSD $\leq$ 1Å)**

CAPRI Target T27 Was A Surprisingly Difficult Target

- Arf6 GTPase / LZ2 Leucine zipper was difficult for most CAPRI predictors



- Best = superposition
- Circles show LZ2 centres:

  blue = high quality

  green = medium quality

  cyan = acceptable qlauity

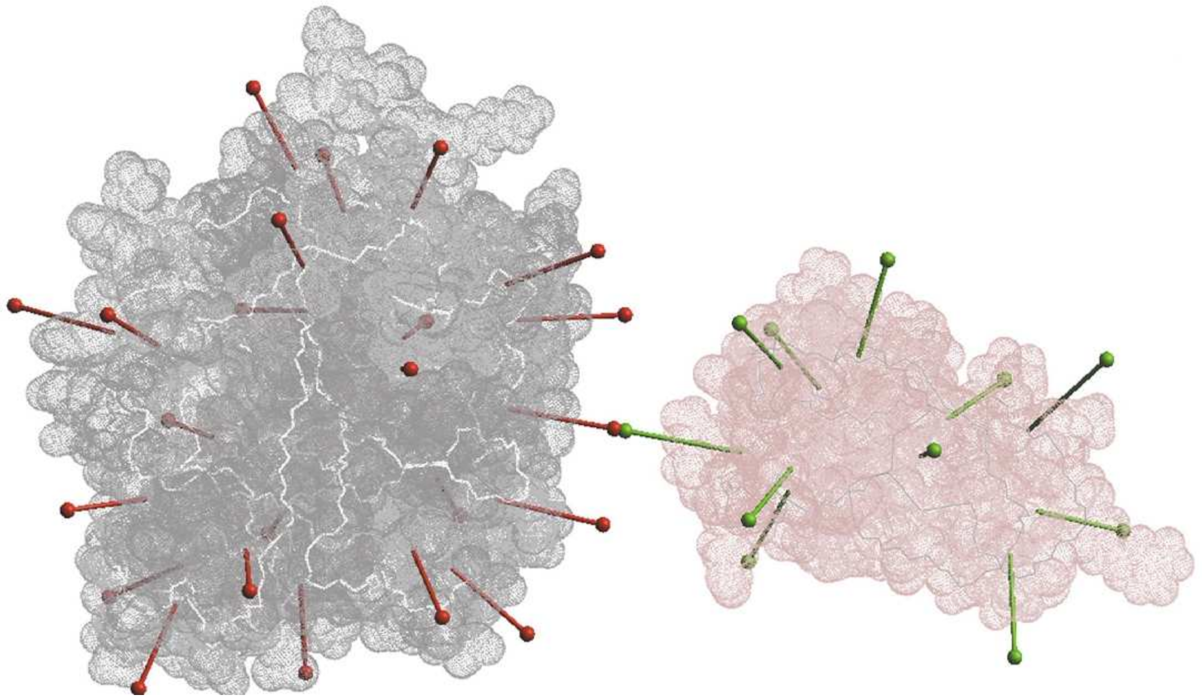
  yellow = wrong

Janin (2010) Molecular BioSystems, 6, 2362–2351

ICM – Multi-Start Pseudo-Brownian Monte-Carlo Energy Minimisation

- Start by sticking "pins" in protein surfaces at 15Å intervals
- Find minimum energy for each pair of starting pins (6 rotations each):

$$E = E_{HVW} + E_{CVW} + 2.16E_{el} + 2.53E_{hb} + 4.35E_{hp} + 0.20E_{solv}$$

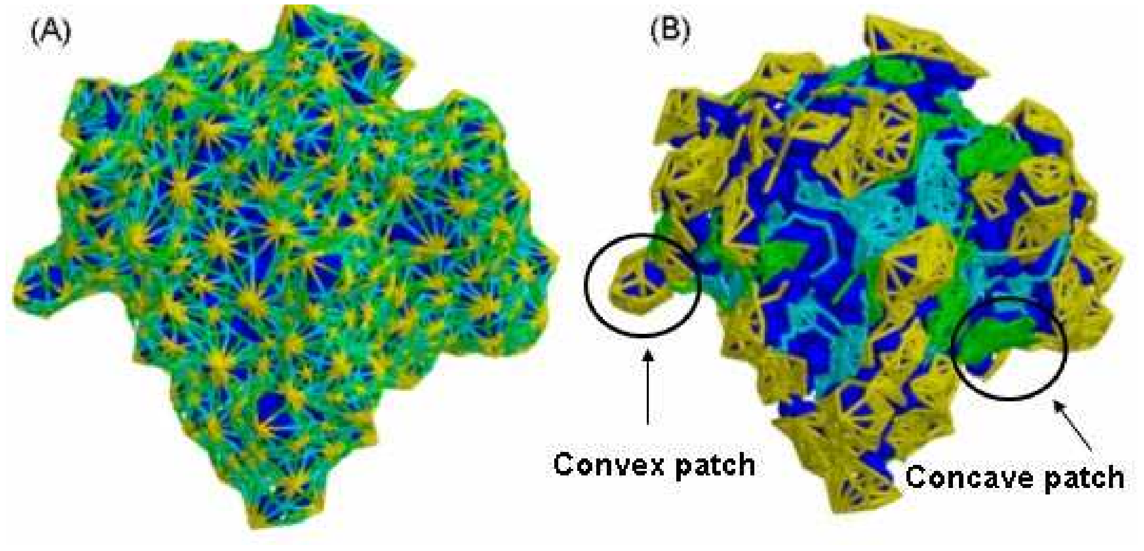


- ICM achieved the best overall results in the first few rounds of CAPRI ...

Fernández-Recio, Abagyan (2004), J Mol Biol, 335, 843–865

PatchDock – Docking by Geometric Hashing

- Use "MS" program to calculate mesh surfaces for each protein
- Divide the mesh into convex "caps", concave "pits", and flat "belts"



- For docking, match pairs of concave $\leftrightarrow$ convex, and flat $\leftrightarrow$ any ...

  ... then test for interpenetrations (steric clashes) between rest of surfaces
- The method is fast (minutes/seconds), and gave good results in CAPRI

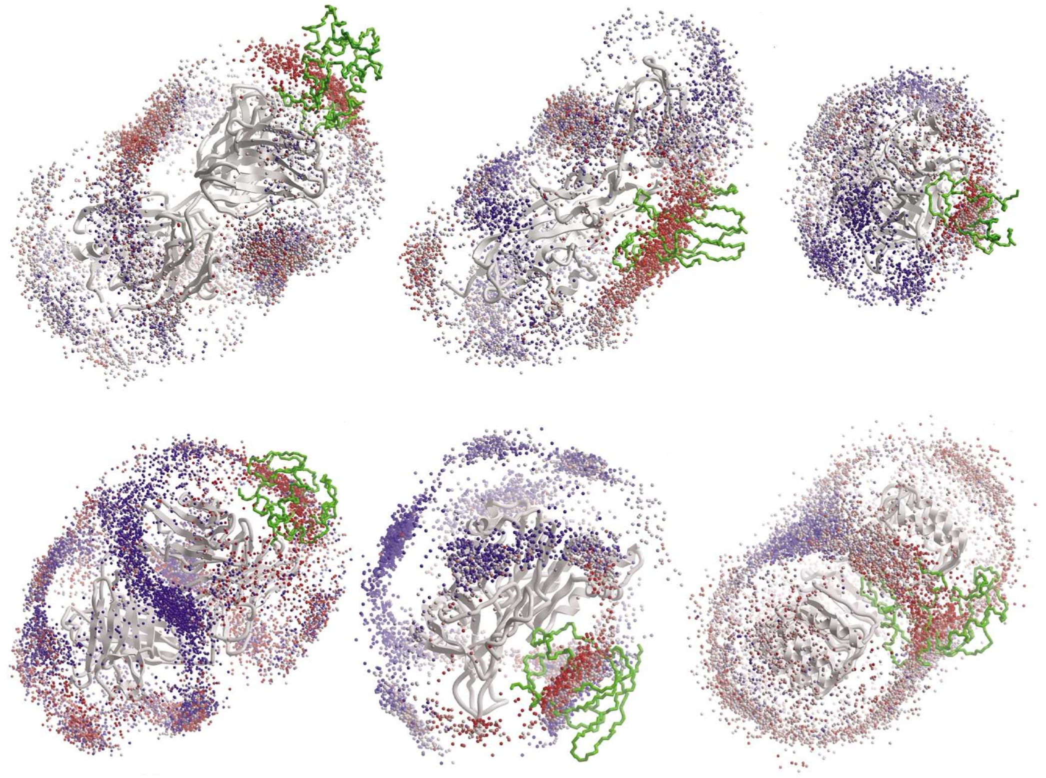
Duhovny et al. (2002), LNCS 2452, 185–200

Schneidman-Duhovny et al. (2005), Nucleaic Acids Research, 33, W363–W367

Connolly (1983), J Applied Crystallography, 16, 548–558

Predicting Protein-Protein Binding Sites

- Many algorithms / servers are available for predicting protein binding sites
- For recent review, see: Fernández-Recio (2011), WIREs Comp Mol Sci 1, 680–698
- Many docking algorithms often show clusters of preferred orientations – docking "funnels"



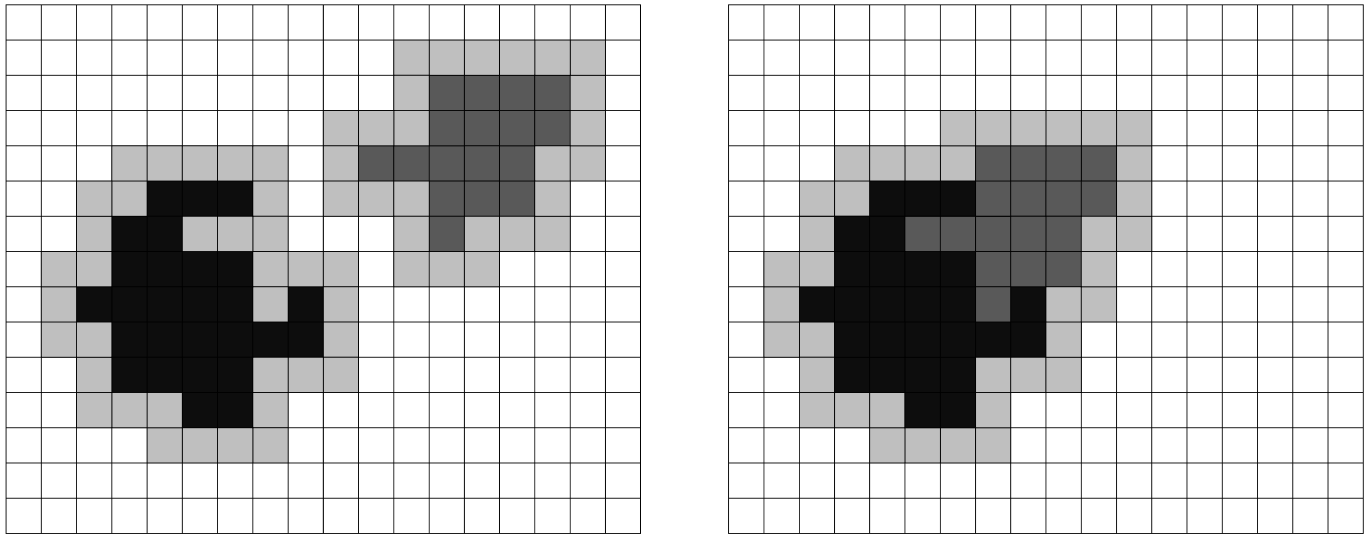
- Lensink & Wodak proposed that docking methods are the best predictors of binding sites

Fernández-Recio, Abagyan (2004), J Molecular Biology, 335, 843–865

Lensink, Wodak (2010), Proteins, 78, 3085–3095

Protein Docking Using Fast Fourier Transforms

- Conventional approaches digitise proteins into 3D Cartesian grids...



- ...and use FFTs to calculated TRANSLATIONAL correlations:

$$C[\Delta x, \Delta y, \Delta z] = \sum_{x,y,z} A[x, y, z] \times B[x + \Delta x, y + \Delta y, z + \Delta z]$$

- BUT for docking, have to REPEAT for many rotations – EXPENSIVE!
- Conventional grid-based FFT docking = SEVERAL CPU-HOURS

Katchalski-Katzir et al. (1992) PNAS, 89 2195–2199

Knowledge-Based Protein-Protein Docking Potentials

- Several groups have developed “statistical” potentials based on “inverse Boltzmann” models
- Example – PIPER + DARS – “Decoys As Reference State” – http://structure.bu.edu/
- Define 18 atom types (based on ACP potential): N, CA, C, O, GC, CB, KN, KC, DO, ...
- Define interaction energy: $E_{IJ} = -RT\ln(P_{IJ}^{nat}/P_{IJ}^{ref})$
  - P_{IJ}^{nat} = probability of contact between atom I and J in a native complex

    (use 20 CAPRI complexes as examples containing native complexes)
  - P_{IJ}^{ref} = probability of contact between atom I and J in a reference state

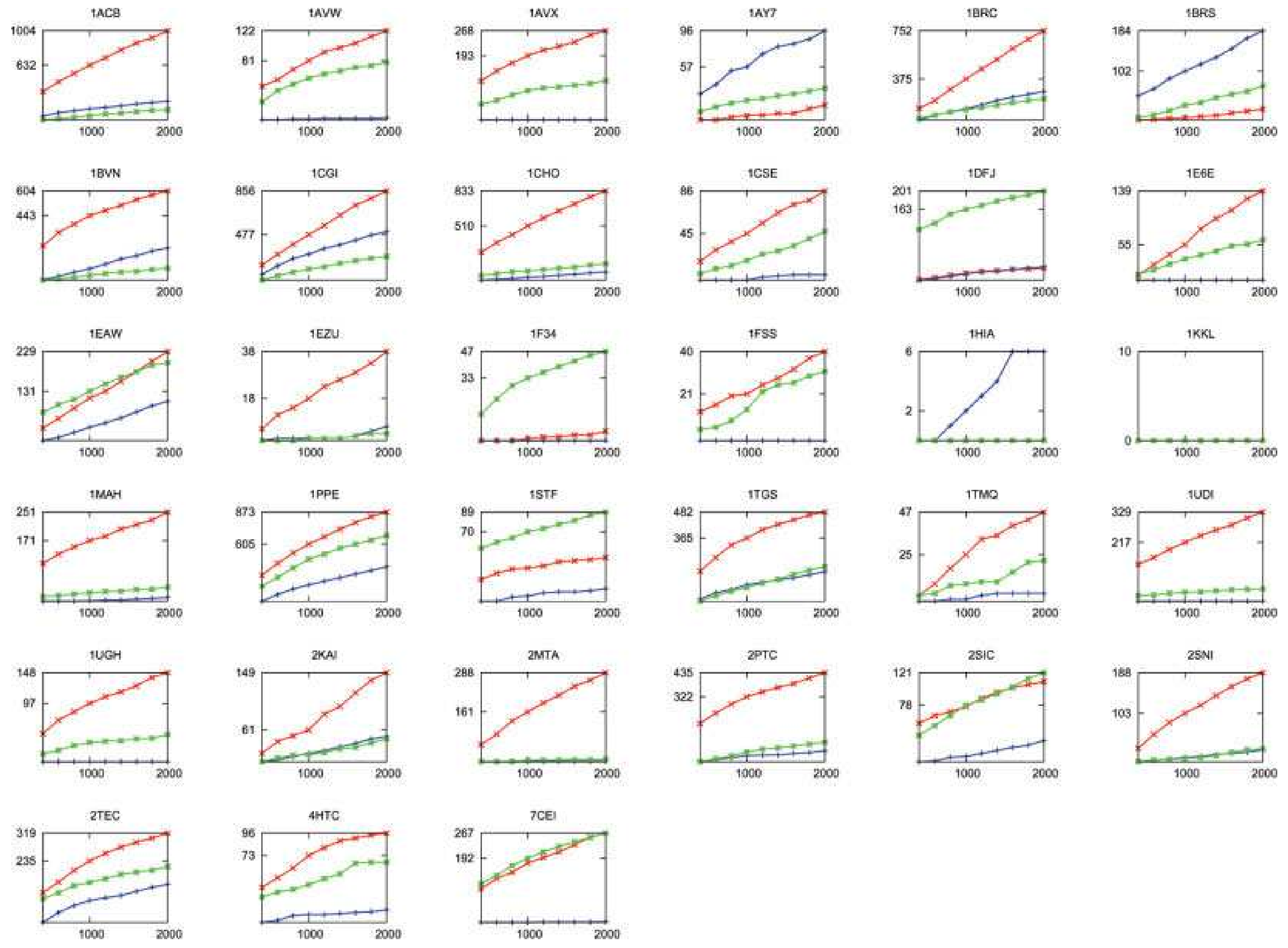
    (use PIPER Cartesian FFT to generate 20,000 “decoy complexes” for each native)
  - Count each type of contact (6Å threshold) to make the probabilities
- This gives a matrix of 18 x 18 atomic interaction energies
- Clever trick: diagonalise the matrix to get the first 4 or 6 leading terms...

  (allows PIPER to use 4 or 6 FFTs instead of 18)
- PIPER + DARS is one of the best approaches in CAPRI...

Kozakov et al. (2006) Proteins, 65, 392–406

DARS Finds More Hits Than ZDOCK and Shape-Only Docking

- Comparing the no. of "hits" for 33 enzyme-inhibitor complexes...

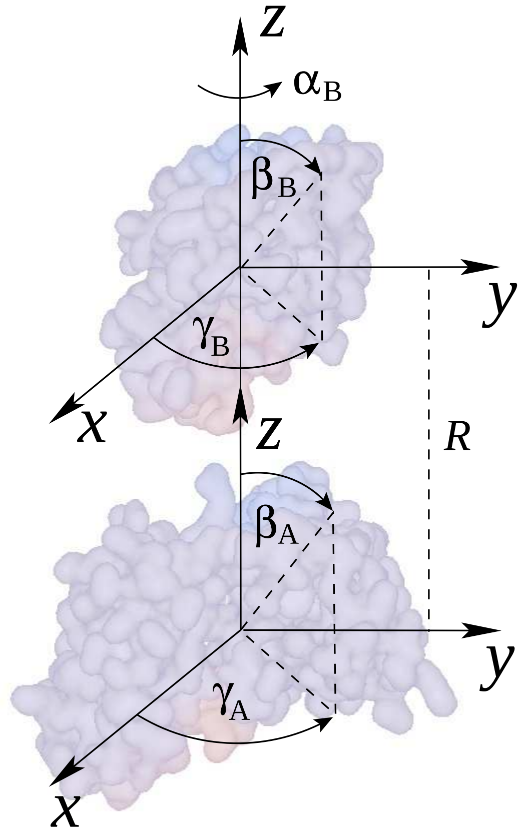


- DARS potential = red; ZDOCK (ACP) = green; shape-only = blue

Kozakov et al. (2006) Proteins, 65, 392–406

Protein Docking Using Polar Fourier Correlations

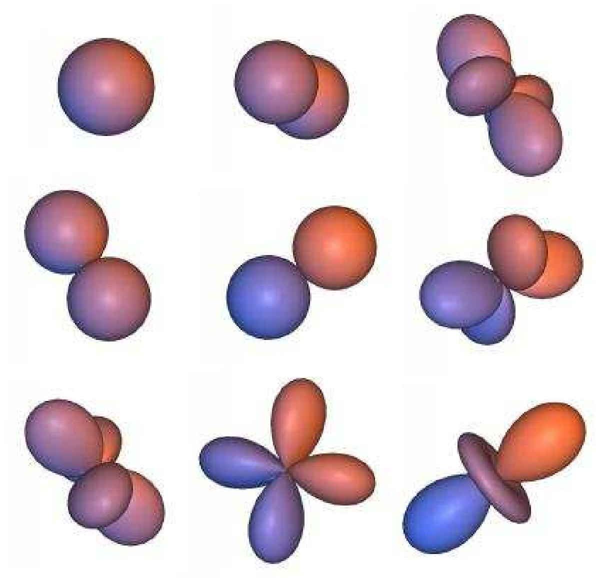
- Rigid body docking can be considered as a largely ROTATIONAL problem
- This means we should use ANGULAR coordinate systems



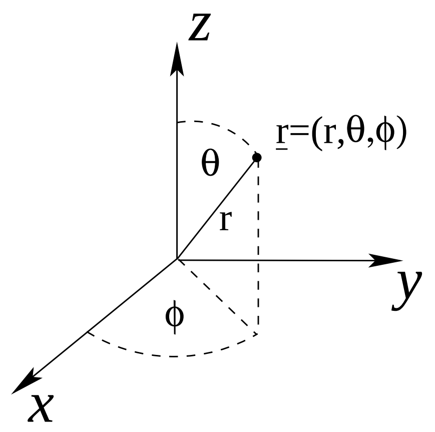
- With FIVE rotations, we should get a good speed-up?

Some Theory – The Spherical Harmonics

- **The spherical harmonics (SHs) are examples of classical "special functions"**



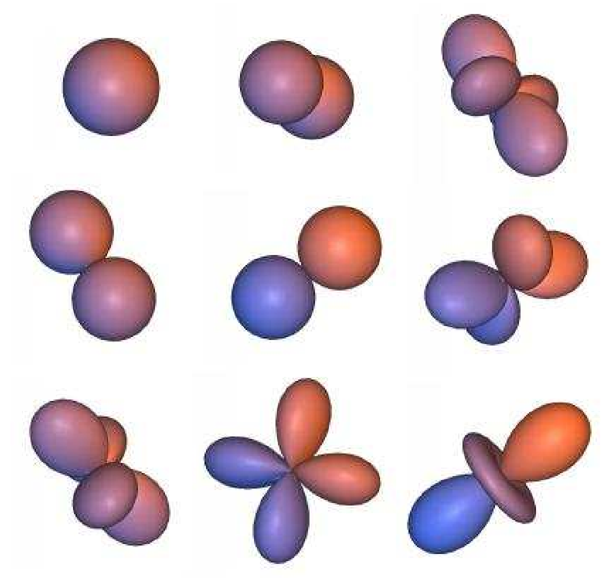
- **Spherical polar coordinates:** $\underline{r} = (r, \theta, \phi)$



- **The spherical harmonics are products of <u>Legendre polynomials</u> and <u>circular functions</u>:**
  - **Real SHs:** $y_{lm}(\theta, \phi) = P_{lm}(\theta)\cos m\phi + P_{lm}(\theta)\sin m\phi$
  - **Complex SHs:** $Y_{lm}(\theta, \phi) = P_{lm}(\theta)e^{im\phi}$
  - **<u>Orthogonal:</u>** $\int y_{lm}y_{kj}\mathrm{d}\Omega = \int Y_{lm}Y_{kj}\mathrm{d}\Omega = \delta_{lk}\delta_{mj}$
  - **<u>Rotation:</u>** $y_{lm}(\theta', \phi') = \sum_j R^{(l)}_{jm}(\alpha, \beta, \gamma) y_{lj}(\theta, \phi)$

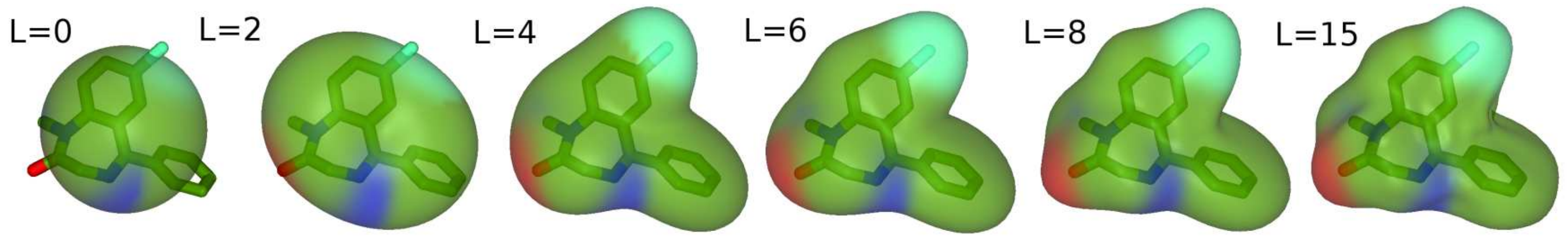
Spherical Harmonic Molecular Surfaces

- Use SHs as orthogonal shape "building blocks":



- Encode distance from origin as SH series to order L:
- $r(\theta, \phi) = \sum_{l=0}^{L} \sum_{m=-l}^{l} a_{lm} y_{lm}(\theta, \phi)$
- Reals SHs: $y_{lm}(\theta, \phi)$
- Coefficients: a_{lm}
- Solve the coefficients by numerical integration

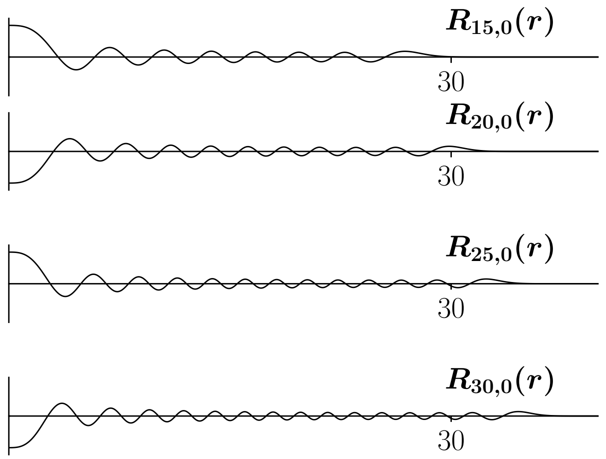
- Normally, L=6 is sufficient for good overlays

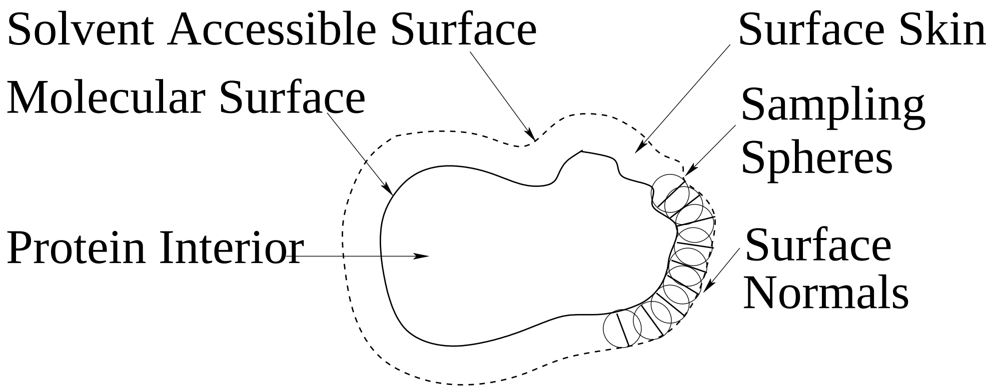


Ritchie and Kemp (1999) J Computational Chemistry, 20, 383–395

Docking Needs a 3D "Spherical Polar Fourier" Representation

- **Need to introduce special orthonormal Laguerre-Gaussian radial functions, $R_{nl}(r)$**
- $R_{nl}(r) = N_{nl}^{(q)} e^{-\rho/2} \rho^{l/2} L_{n-l-1}^{(l+1/2)}(\rho); \qquad \rho = r^2/q, \quad q = 20.$





- **Surface Skin:** $\sigma(\underline{r}) = \begin{cases} 1; & \underline{r} \in \text{surface skin} \\ 0; & \text{otherwise} \end{cases}$ **Interior:** $\tau(\underline{r}) = \begin{cases} 1; & \underline{r} \in \text{protein atom} \\ 0; & \text{otherwise} \end{cases}$

- **Parametrise as:** $\sigma(\underline{r}) = \sum_{n=1}^{N} \sum_{l=0}^{n-1} \sum_{m=-l}^{l} a_{nlm}^{\sigma} R_{nl}(r)\, y_{lm}(\theta, \phi)$

- **TRANSLATIONS:** $a_{nlm}^{\sigma\prime\prime} = \sum_{n'l'}^{N} T_{nl,n'l'}^{(|m|)}(R) a_{n'l'm}^{\sigma}$

Ritchie (2005) J Applied Crystallography, 38, 808–818 (for translation formulae)

SPF Protein Shape-Density Reconstruction

Interior density: $$\tau(\underline{r}) = \sum_{nlm}^{N} a^{\tau}_{nlm} R_{nl}(r) y_{lm}(\theta, \phi)$$

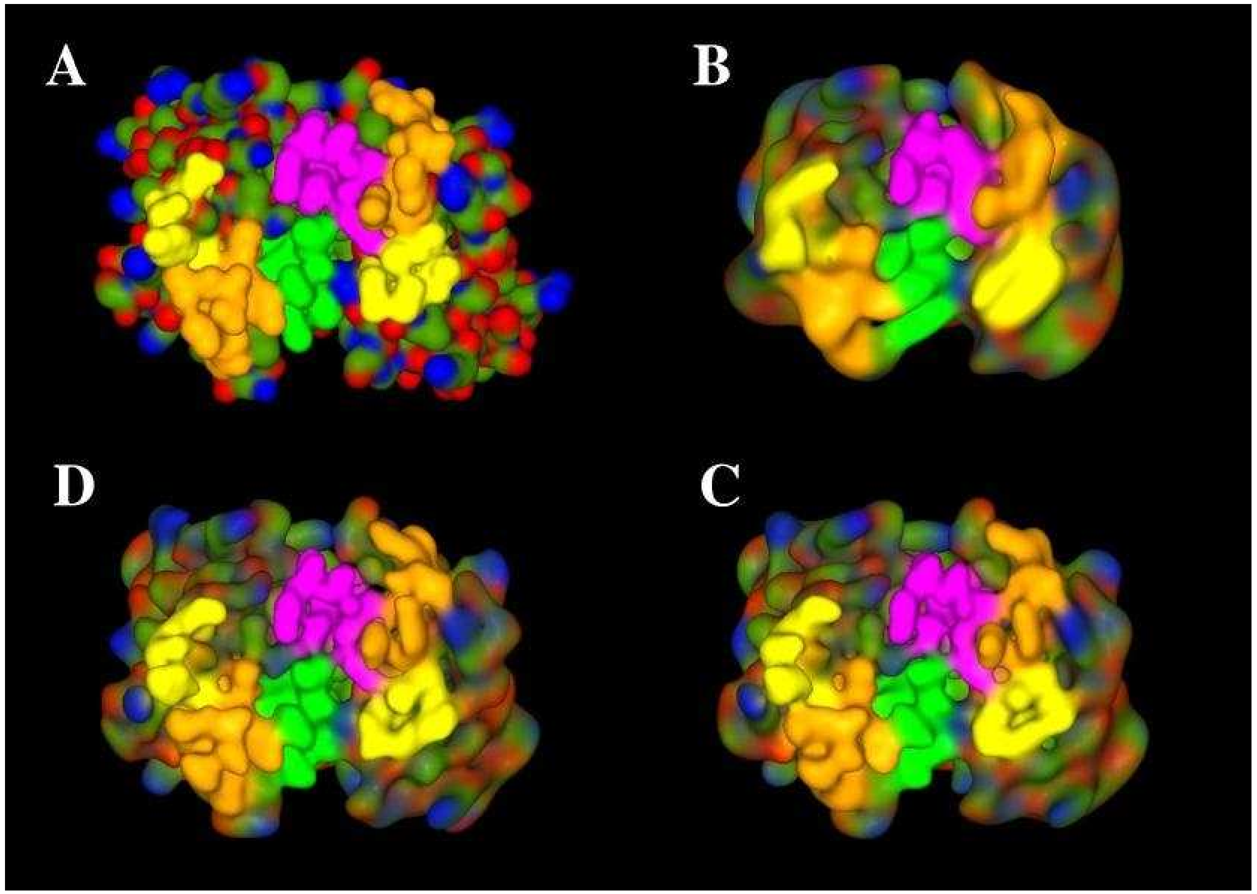
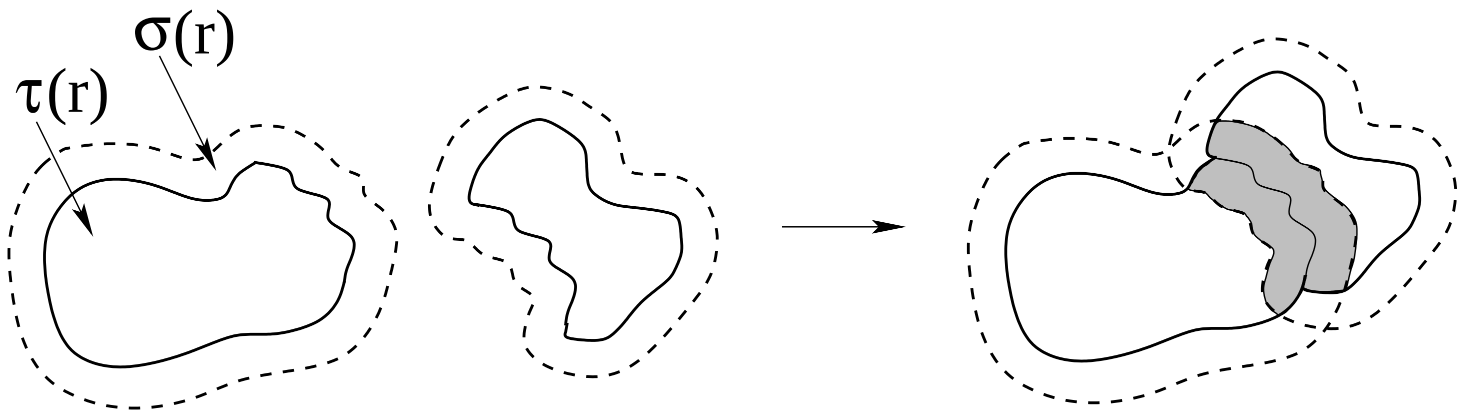


Image	Order	Coefficients
A	Gaussians	-
B	N = 16	1,496
C	N = 25	5,525
D	N = 30	9,455

Ritchie (2003), Proteins, 52, 98–106

Protein Docking Using SPF Density Functions



Favourable: $$\int (\sigma_A(\underline{r}_A)\tau_B(\underline{r}_B) + \tau_A(\underline{r}_A)\sigma_B(\underline{r}_B))\mathrm{d}V$$

Unfavourable: $$\int \tau_A(\underline{r}_A)\tau_B(\underline{r}_B)\mathrm{d}V$$

Score: $$S_{AB} = \int (\sigma_A\tau_B + \tau_A\sigma_B - Q\tau_A\tau_B)\mathrm{d}V$$ **Penalty Factor:** $Q = 11$

Orthogonality: $$S_{AB} = \sum_{nlm} \left(a^{\sigma}_{nlm}b^{\tau}_{nlm} + a^{\tau}_{nlm}\left(b^{\sigma}_{nlm} - Qb^{\tau}_{nlm}\right)\right)$$ **(in units of volume)**

Search: **6D space = 1 distance + 5 Euler rotations:** $(R, \beta_A, \gamma_A, \alpha_B, \beta_B, \gamma_B)$

Ritchie, Kemp (2000), Proteins, 39, 178–194

Hex Polar Fourier Correlation Example – 3D Rotational FFTs

- Set up 3D rotational FFT as a series of matrix multiplications...

Rotate: $$a'_{nlm} = \sum_{t=-l}^{l} R^{(l)}_{mt}(0, \beta_A, \gamma_A) a_{lt}$$

Translate: $$a''_{nlm} = \sum_{kj}^{N} T^{(|m|)}_{nl,kj}(R) a'_{kjm}$$

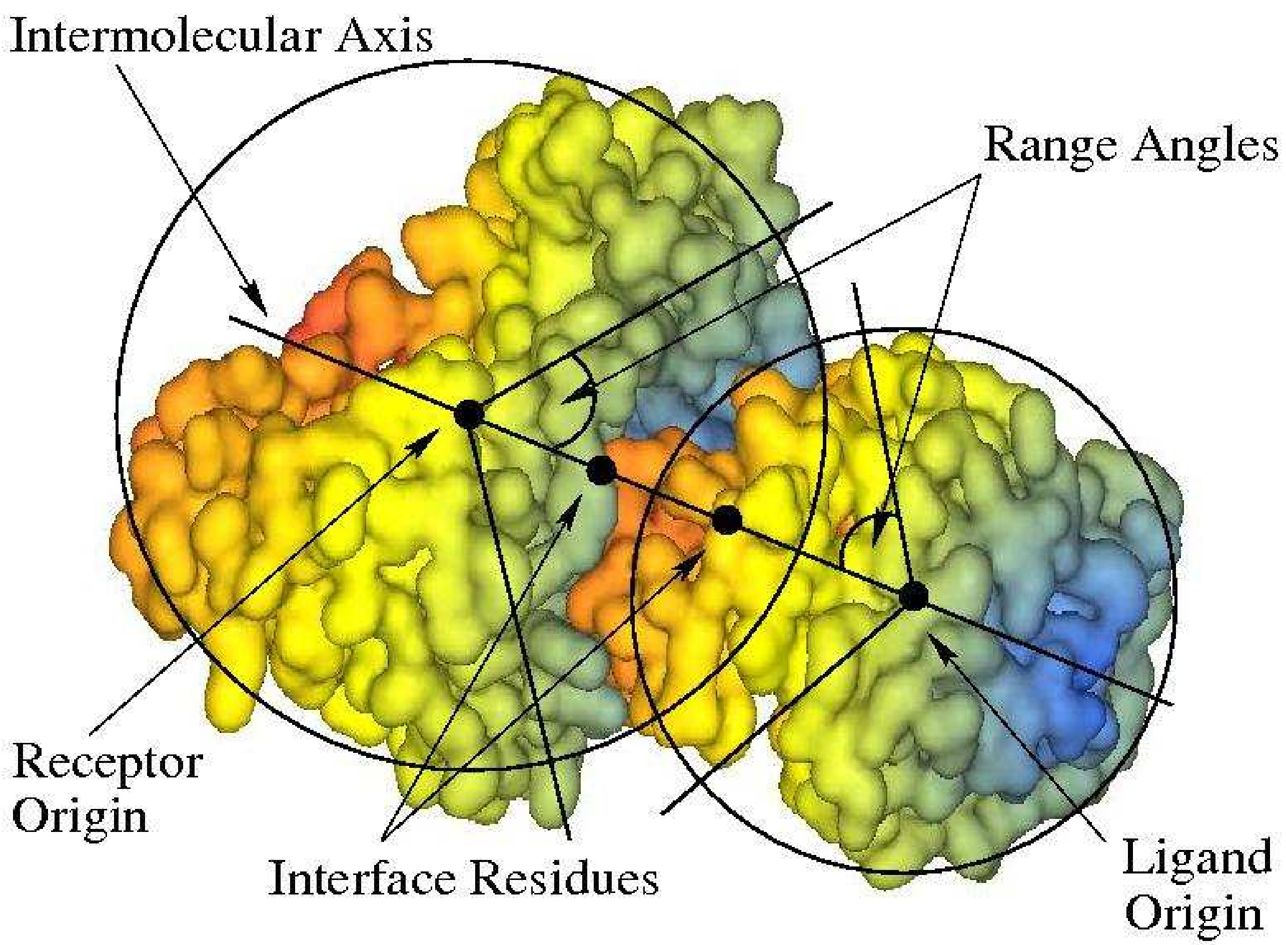
Real to complex: $$A_{nlm} = \sum_t a''_{nlt} U^{(l)}_{tm}, \qquad B_{nlm} = \sum_t b_{nlt} U^{(l)}_{tm}$$

Multiply: $$C_{muv} = \sum_{nl} A^*_{nlm} B_{nlv} \Lambda^{um}_{lv}$$

3D FFT: $$S(\alpha_B, \beta_B, \gamma_B) = \sum_{muv} C_{muv} e^{-i(m\alpha_B + 2u\beta_B + v\gamma_B)}$$

- On one CPU, docking takes from 15 to 30 minutes

Exploiting Proir Knowledge in SPF Docking



- Knowledge of even only one key residue can reduce search space enormously...
- This accelerates the calculation and helps to reduce false-positive predictions

CAPRI Results: Targets 1–7 (2000 – 2003)

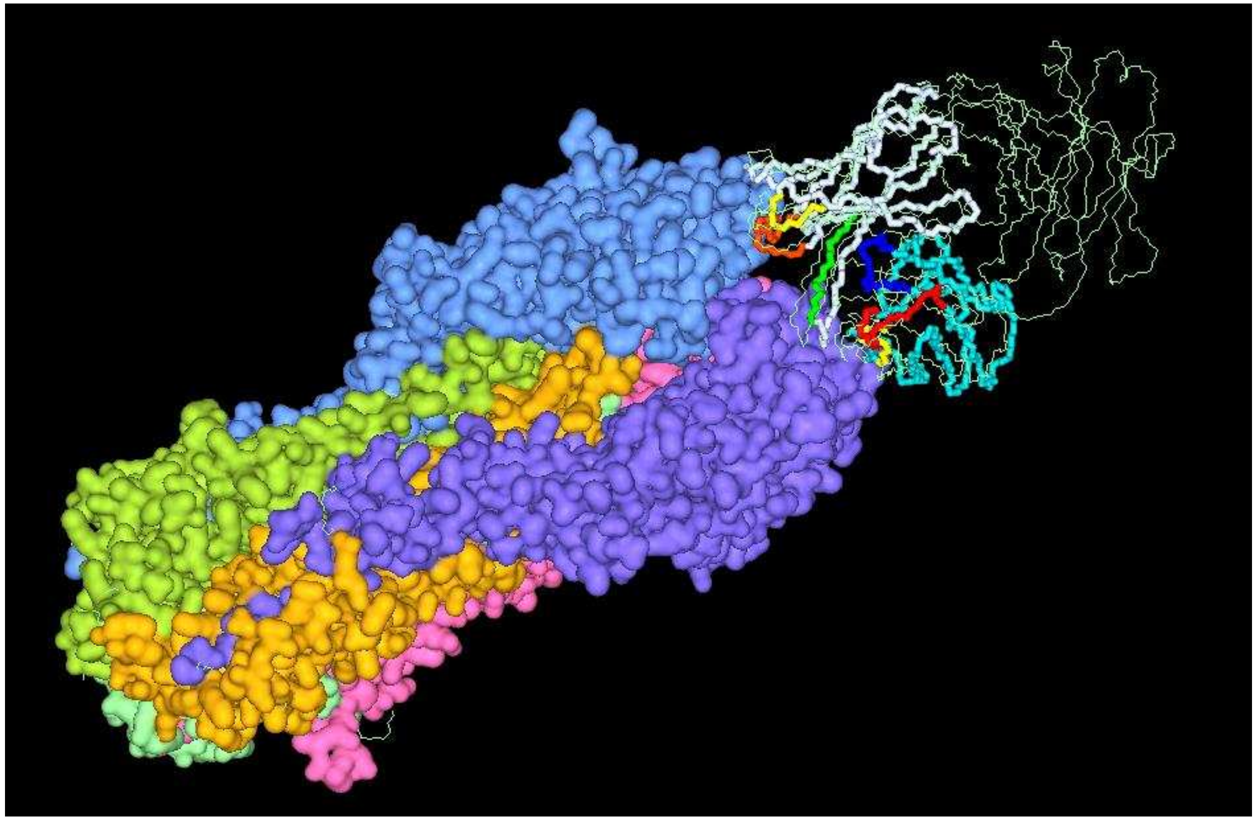
Predictor	Software	Algorithm	T1	T2	T3	T4	T5	T6	T7
Abagyan	ICM	FF			**			***	**
Camacho	CHARMM	FF	*					***	***
Eisenstein	MolFit	FFT	*	*					***
Sternberg	FTDOCK	FFT		*				**	*
Ten Eyck	DOT	FFT	*	*				**	
Gray		MC						**	***
Ritchie	Hex	SPF			**			***	
Weng	ZDOCK	FFT		**					**
Wolfson	BUDDA/PPD	GH	*						***
Bates	Guided Docking	FF	-	-	-				***
Palma	BIGGER	GF	-		-			**	*
Gardiner	GAPDOCK	GA	*	*	-	-	-	-	-
Olson	Surfdock	SH	*			-	-	-	-
Valencia		ANN	*	-	-	-	-	-	-
Vakser	GRAMM	FFT		*		-	-	-	-

$*$ low, $**$ medium, $***$ high accuracy prediction; — no prediction

Mendez et al. (2003) Proteins, 52, 51–67

Hex Protein Docking Example – CAPRI Target 3

- **Example: best prediction for CAPRI Target 3 – Hemagglutinin/HC63**



Ritchie, Kemp (2000), Proteins 39, 178–194

Ritchie (2003), Proteins, 52, 98–106

CAPRI Results: Targets 8–19 (2003 – 2005)

Predictor	Software	T8	T9	T10	T11	T12	T13	T14	T15–T17	T18	T19
Abagyan	ICM	**		*	**	***	*	***		**	**
Wolfson	PatchDock	**	*	*	*	*	-	**		**	*
Weng	ZDOCK/RDOCK	**			*	***	***	***		**	**
Bates	FTDOCK	*		*	**	*		**		**	*
Baker	RosettaDock	-			**	***	**	***			***
Camacho	SmoothDock	**				***	***	**		**	*
Gray	RosettaDock	***	-	-	**	***					**
Bonvin	Haddock	-	-	**	**		***	***			
Comeau	ClusPro	**				***	*				*
Sternberg	3D-DOCK	**			*	*		**			*
Eisenstein	MolFit	***			*	***		**			
Ritchie	Hex				**	***	*	*			
Zhou		-	-		-	***	**	*		*	
Ten Eyck	DOT					***	***	**			
Zacharias	ATTRACT	**		-	-	-	-	***			**
Valencia		*			*	*	-				-
Vakser	GRAMM	-	-		-	-	-	**		**	
Homology	modelling				#			#			#
Cancelled									#		

Mendez et al. (2005), Proteins, 60, 150–169

High Order FFTs, Multi-Threading, and Graphics Processors

- Spherical polar coordinates give an analytic formula for 6D correlations:

In particular:
$$S_{AB} = \sum_{jsmlvrt} \Lambda_{js}^{rm} T_{js,lv}^{(|m|)}(R) \Lambda_{lv}^{tm} e^{-i(r\beta_A - s\gamma_A + m\alpha_B + t\beta_B + v\gamma_B)}$$

- This allows high order FFTs to be used – 1D, 3D, and 5D
- ... multiple FFTs can easily be executed in parallel
- ... also, it is relatively easy to implement on modern GPUs

- Up to 512 arithmetic “cores”
- Up to 6 Gb memory
- Easy API with C++ syntax
- Grid of threads model (“SIMT”)

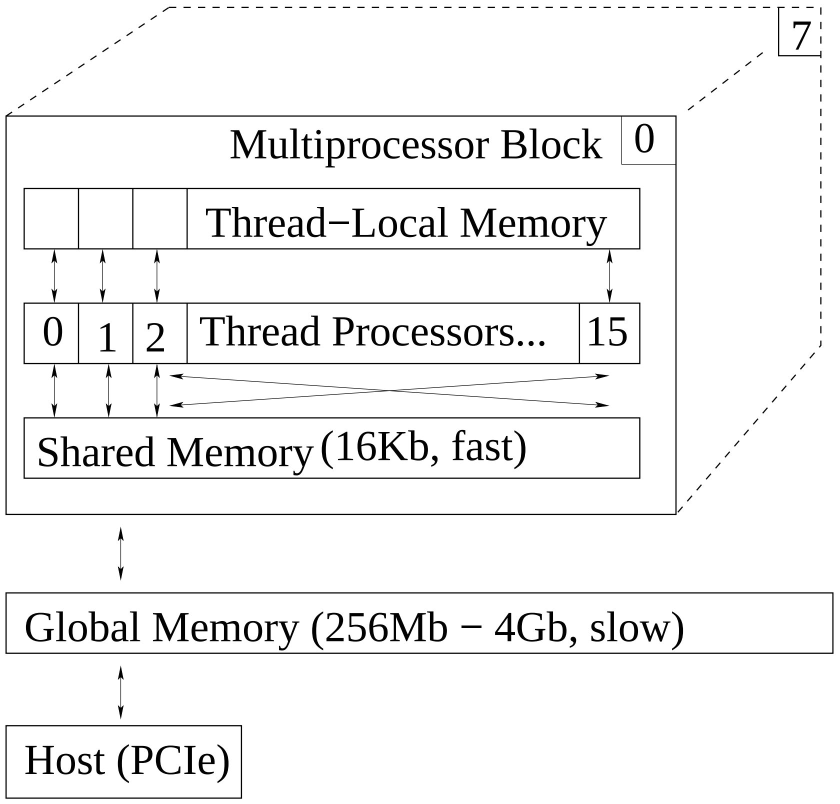
- Due to memory latency effects, 1D FFTs are MUCH FASTER than 3D FFTs ...

Ritchie, Kozakov, Vajda (2008), Bioinformatics, 24, 1865–1873

Ritchie, Venkatraman (2010), Bioinformatics, 26, 2398–2405

The CUDA Device Architecture

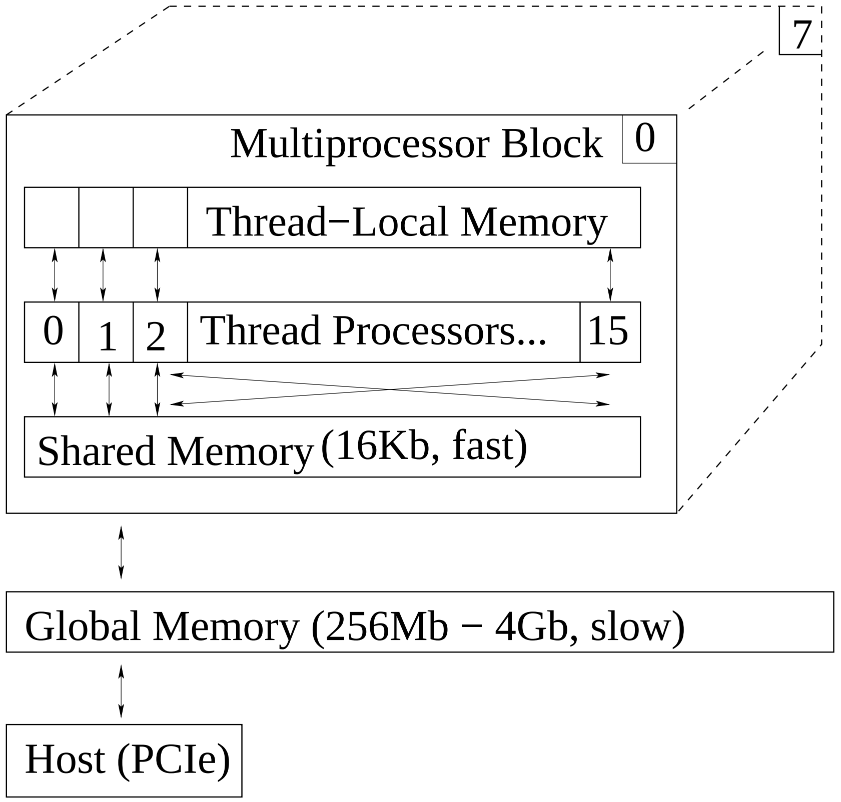
- Typically 8–16 multi-processor blocks, each with 16 thread units



- NB. only a very small amount of fast shared memory is available
- NB. global memory is ABOUT 80x SLOWER than shared memory

An Alternative View of the CUDA Device Architecture

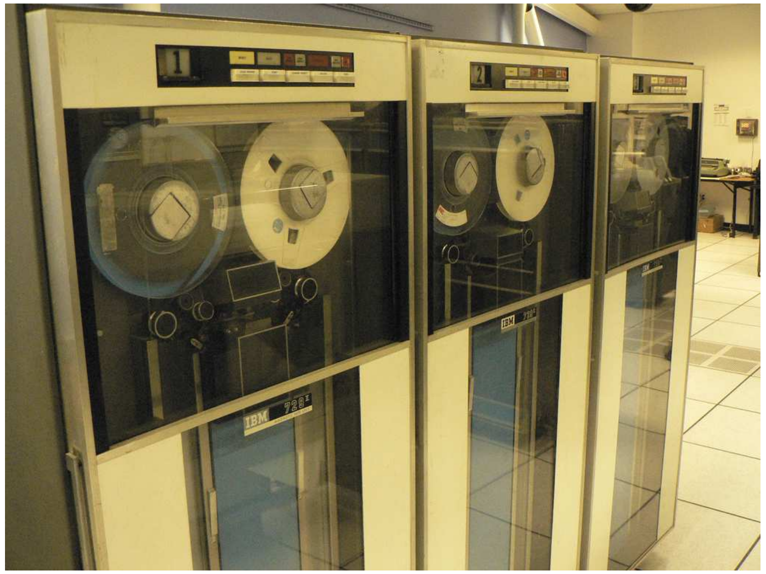
- Reading and writing global memory is like doing slow I/O

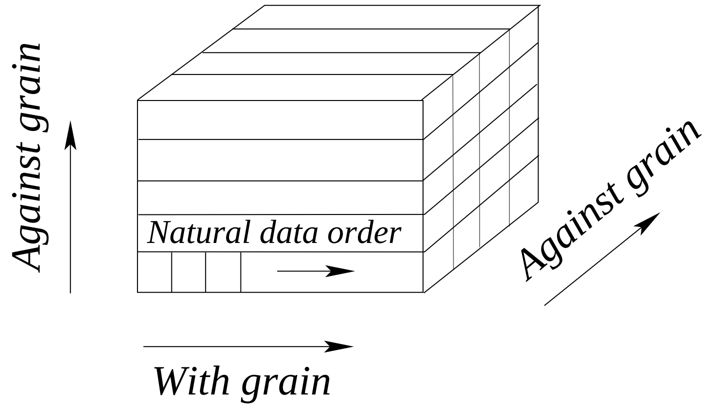


- Strategy: aim for “high arthmetic intensity” in fast shared memory

Slow Devices are Not Well Suited for Random Access

- On the GPU, think of global memory as a SLOW device ...
- ... and that accessing array data "against the grain" is like random access

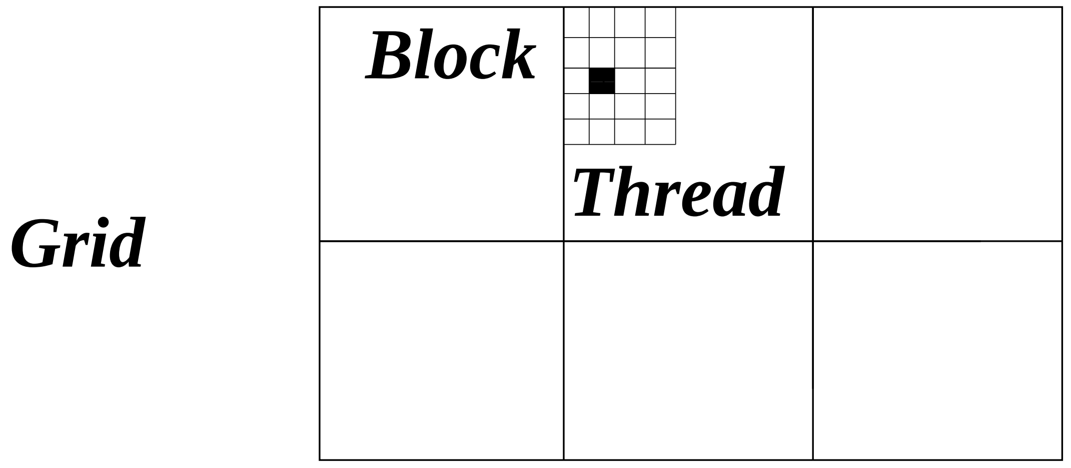




- This explains why 3D FFTs are SLOW on current GPUs...
- Good strategies:
  - avoid unnecessary "I/O" on global memory
  - make threads cooperate by reading consecutive blocks of global memory linearly
  - do "random access" (e.g. to transpose a matrix) only in shared memory

The CUDA Grid-Block Programming Model

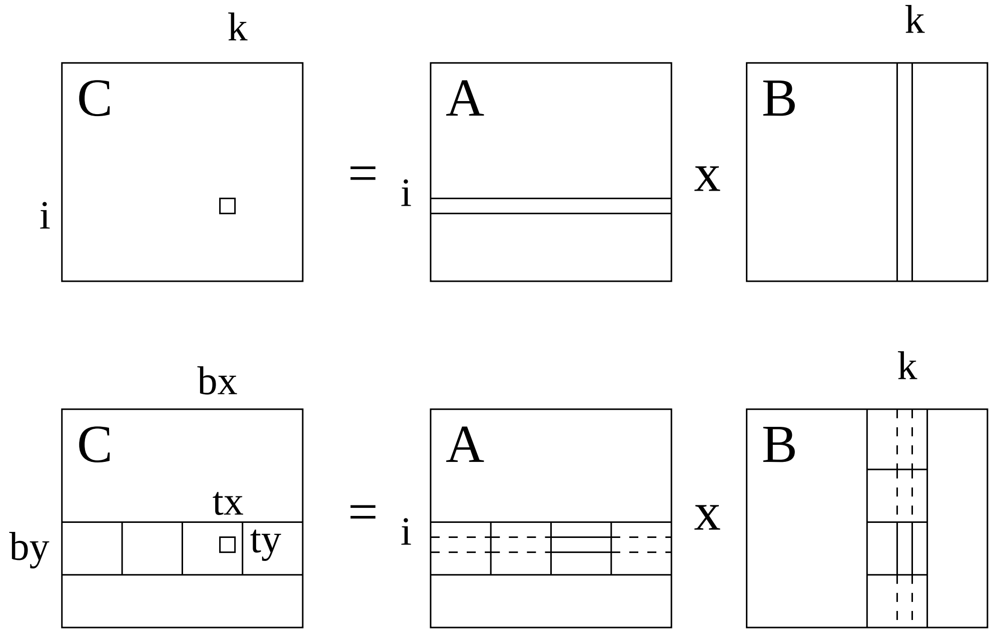
- CUDA implements SIMT using a GRID of BLOCKS of THREADS
- Each THREAD executes a simple “kernel” function
- A BLOCK of related threads all execute the same kernel
- The scheduler launches multiple blocks in parallel, making a GRID of blocks



- For example, in matrix arithmetic:
  - the matrix is divided into a grid of blocks
  - one thread calculates one element of the result

CUDA Programming Example - Matrix Multiplication

- Matrix multiplication C = A * B
- Each thread is responsible for calculating one element: C[i,k]



- Conventional algorithm: rows and columns
- C[i,k] = A[i] * B[k]
- Thread-block algorithm working on TILES

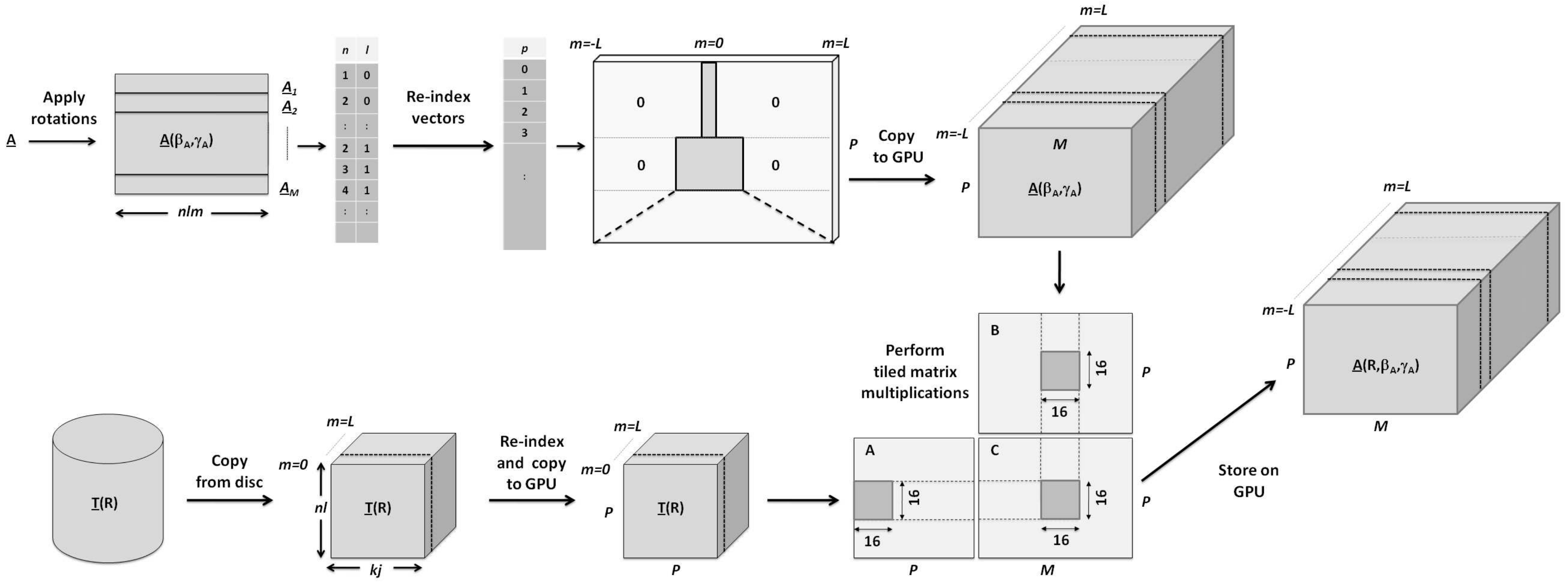
- A tile size of 16x16 is just right!
- Threads co-operate by reading & sharing tiles of A & B
- Multi-processor launches multiple blocks to compute all of C
- Executing thread-blocks concurrently hides global memory latency

CUDA Programming Example – Matrix Multiplication Kernel

```
__global__ void matmul(int wA, int wB, float *A, float *B, float *C)
{
   float Cik = 0.0;                                // thread-local result variable
   int bx = blockIdx.x, tx = threadIdx.x;          // thread subscripts
   int by = blockIdx.y, ty = threadIdx.y;          // ("this" thread is one of a 2-D grid)

   __shared__ float a_sub[16][16], b_sub[16][16]; // declare shared memory

   for (int j=0; j<wA; j+=16) {                    // thread-local loop over tiles of A and B

      int ij = (16*by+ty)*wA + (j+tx);             // thread-local array subscripts
      int jk = (j+ty)*wB + (16*bx+tx);

      a_sub[ty][tx] = A[ij];                       // copy global data to shared memory ("I/O")
      b_sub[ty][tx] = B[jk];

      __syncthreads();                             // wait until all memory I/O has finished

      for (int jj=0; jj<16; jj++) {

         Cik += a_sub[ty][jj] * b_sub[jj][tx];    // multiply row*column in current tiles
      }

      __syncthreads();                             // synchronise threads before starting more I/O
   }

   C[(16*by+ty)*wB + (16*bx+tx)] = Cik;           // copy local result -> global memory
}
```

Hex GPU Docking – Rotate and Translate Protein A

1. On CPU, calculate multiple (β_A, γ_A) rotations of protein A
2. On CPU, re-index translation matrices and rotated coefficients into regular sparse arrays
3. On GPU, translate multiple protein A coeffcients using tiled matrix multiplication



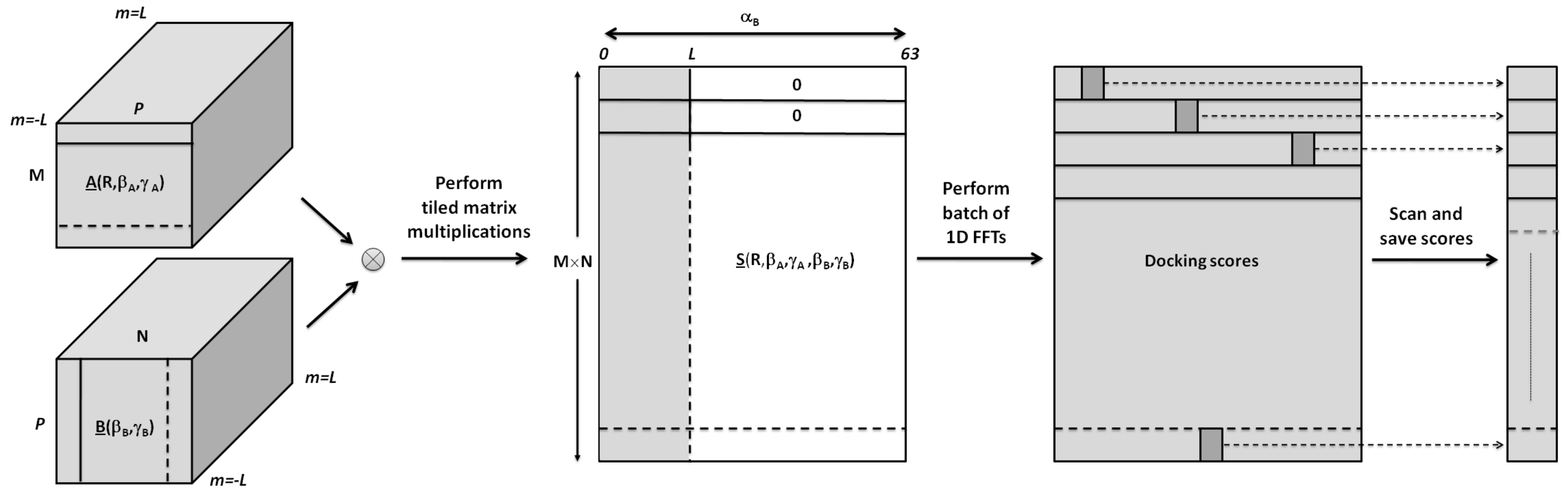
Hex GPU Docking – Perform Multiple 1D FFTs

- Next, calculate multiple 1D FFTs of the form:

$$S_{AB}(\alpha_B) = \sum_m e^{-im\alpha_B} \sum_{nl} A^{\sigma}_{nlm}(R, \beta_A, \gamma_A) \times B^{\tau}_{nlm}(\beta_B, \gamma_B)$$

4. On GPU, cross-multiply transformed A with rotated B coefficients (as above)

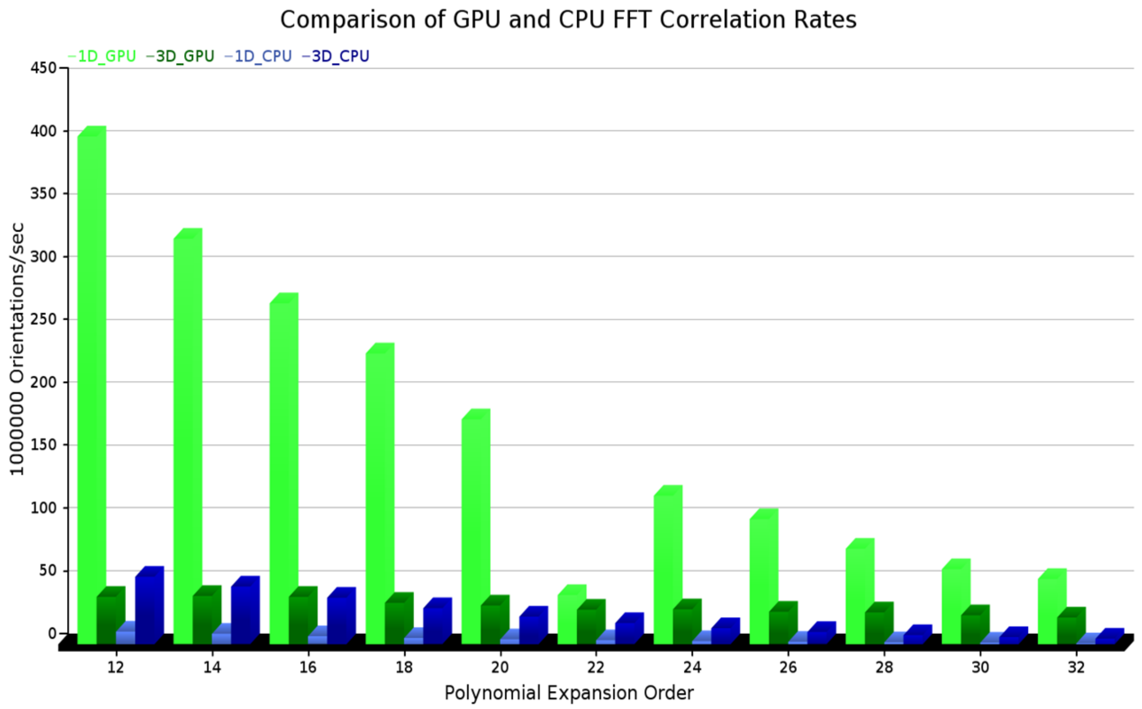
5. On GPU, perform batch of 1D FFTs using cuFFT and save best orientations



- 3D FFTs in $(\alpha_B, \beta_B, \gamma_B)$ can be calculated similarly, ...

Results – GPU v's CPU Docking Performance

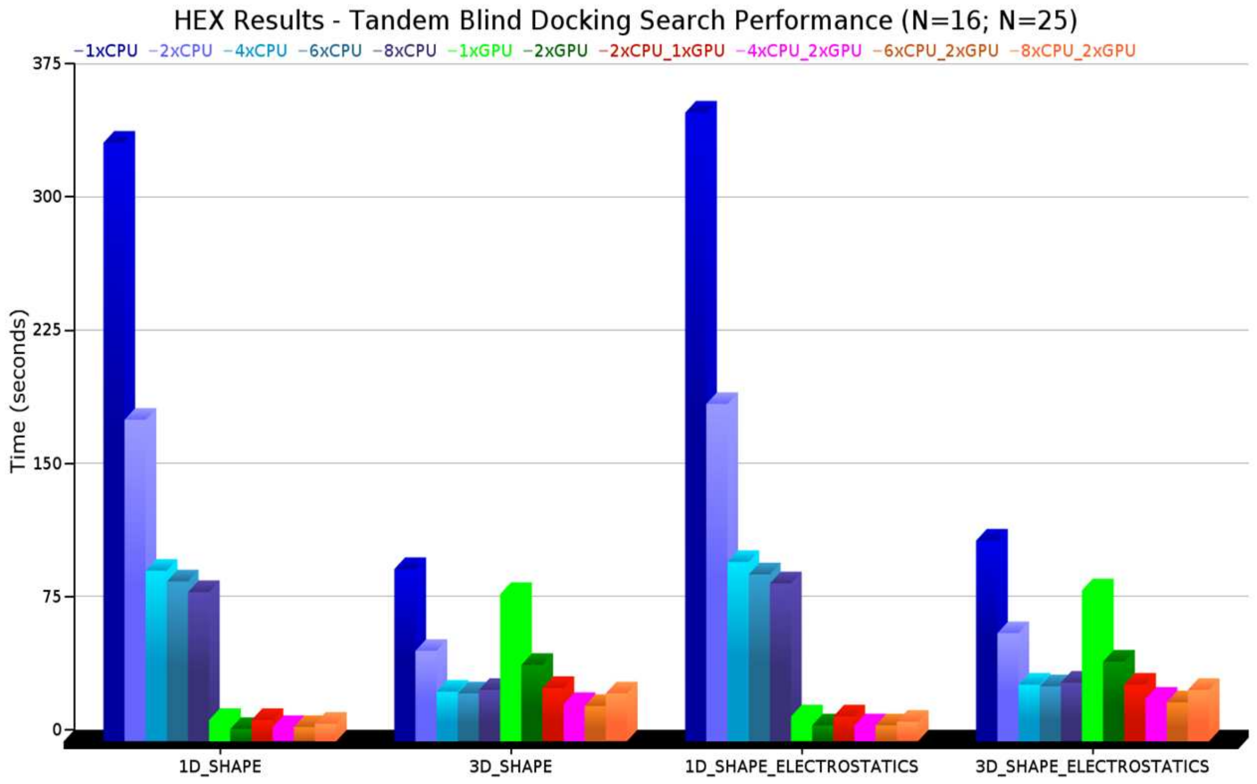
- Key Hex functions implemented using only 5 or 6 CUDA kernels
- 1D and 3D FFTs are calculated using Nvidia's cuFFT library
- Here, GPU = Nvidia FX-5800, CPU = Intel i7-965



- Hex 1D correlations are up to 100x faster on FX-5800 than on iCore7
- Overall, including set-up, Hex 1D FFT is about 45x faster on FX-5800 than on iCore7

Protein Docking Speed-Up using Multiple GPUs and CPUs

- With multi-threading, we can use as many GPUs and CPUs as are available



- For best performance: use 2 GPUs alone, or 6 CPUs plus 2 GPUs
- With 2 GPUs, docking takes about 10 seconds – very important for large-scale!

Speed Comparison with ZDOCK and PIPER

- Hex: 52000 x 812 rotations, 50 translations (0.8Å steps)
- ZDOCK: 54000 x 6 deg rotations, 92Å 3D grid (1.2Å cells)
- PIPER: 54000 x 6 deg rotations, 128Å 3D grid (1.0Å cells)
- Hardware: GTX 285 (240 cores, 1.48 GHz)

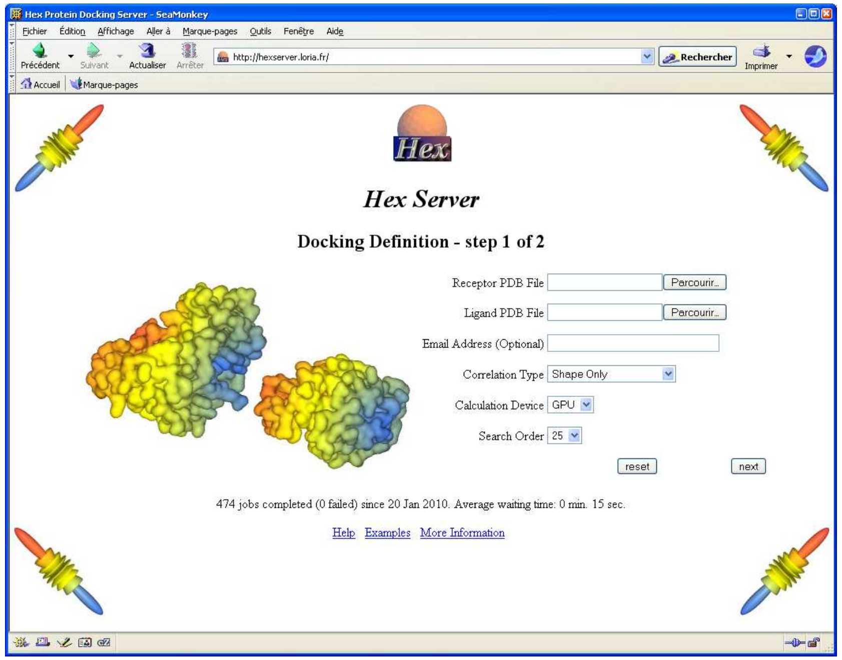
	Kallikrein A / BPTI (233 / 58 residues)#					
	ZDOCK	PIPER[†]	PIPER[†]	Hex	Hex	Hex[‡]
FFT	1xCPU	1xCPU	1xGPU	1xCPU	4xCPU	1xGPU
3D	7,172	468,625	26,372	224	60	84
(3D)[⋆]	(1,195)	(42,602)	(2,398)	224	60	84
1D	–	–	–	676	243	15

# execution times in seconds
* (times scaled to two-term potential, as in Hex)

- Several other bioinformatics applications also run well on GPUs – See:
- https://biomanycores.org/
- http://www.nvidia.com/object/bio_info_life_sciences.html

“Hex” and “HexServer”

- Multi-threaded Hex: first (only) docking program to get full benefit of GPUs


Hex Protein Docking Server - SeaMonkey
Fichier Édition Affichage Aller à Marque-pages Outils Fenêtre Aide
Précédent Suivant Actualiser Arrêter
http://hexserver.loria.fr/
Rechercher
Imprimer
Accueil Marque-pages
Hex

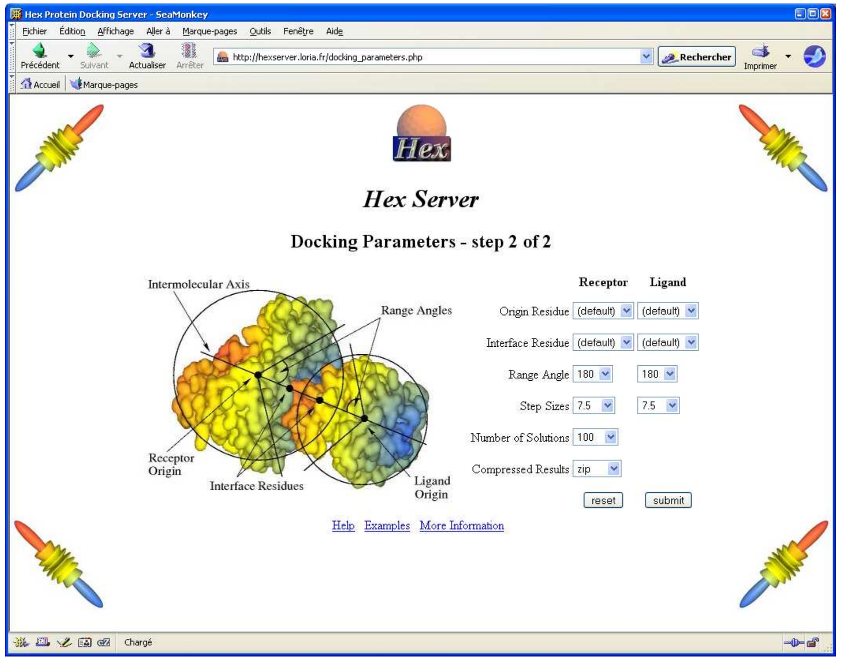
Hex Server

Docking Definition - step 1 of 2

Receptor PDB File	Parcourir...
Ligand PDB File	Parcourir...
Email Address (Optional)	
Correlation Type	Shape Only
Calculation Device	GPU
Search Order	25

474 jobs completed (0 failed) since 20 Jan 2010. Average waiting time: 0 min. 15 sec.

Help Examples More Information


Hex Protein Docking Server - SeaMonkey
Fichier Édition Affichage Aller à Marque-pages Outils Fenêtre Aide
Précédent Suivant Actualiser Arrêter
http://hexserver.loria.fr/docking_parameters.php
Rechercher
Imprimer
Accueil Marque-pages
Hex
Intermolecular Axis
Range Angles
Receptor Origin
Interface Residues
Ligand Origin

Hex Server

Docking Parameters - step 2 of 2

	Receptor	Ligand
Origin Residue	(default)	(default)
Interface Residue	(default)	(default)
Range Angle	180	180
Step Sizes	7.5	7.5
Number of Solutions	100	
Compressed Results	zip	

reset
submit
Help Examples More Information
Chargé

- Hex: Over 25,000 down-loads, over 280 citations in bio literature...

- HexServer: About 1,000 docking jobs per month...

Ritchie, Kemp (2000) Proteins, 39, 178–194

...

Ritchie, Venkatraman (2010) Bioinformatics, 26, 2398–2405

Macindoe et al. (2010), Nucleic Acids Research, 38, W445–W449

Conclusions

- There is an increasing number of on-line resources for studying PPIs
- Docking is becoming increasingly important for modeling PPIs
- CAPRI experiment has stimulated the development of docking algorithms
- The spherical polar Fourier representation is useful for protein docking
  - Rigid-body protein docking on a GPU now takes only a few seconds
  - This was implemented using only 5 or 6 GPU kernels
  - But a lot of low-level CPU code had to be re-written
  - Worth the effort – rigid body docking is no longer a rate-limiting step
- Fast docking could open the door for other shape matching problems ?
  - Cryo-EM density fitting ?
  - 3D Virtual screening ?

Extra Slides

CUDA Matrix Multiplication Kernel – Launching a GPU Kernel

- CUDA adds some programming “extensions” to support the grid-block model
- compile with “nvcc” compiler ...
- (here, we assume matrix dimensions are multiples of 16)

```
__host__ void matmul(                                     // CPU launch function
                     int wA,                              // width of array A (no. columns)
                     int hA,                              // height of array A (no. rows)
                     int wB,                              // width of array B (no. columns)
                     float *A,                            // input array A (in global mamory)
                     float *B,                            // input array B (in global mamory)
                     float *C)                            // result array C (in global memory)
{
   dim3 dimBlock(16, 16, 1);                              // set block size (16x16=256 threads)

   dim3 dimGrid(wB/16, hA/16, 1);                         // set grid size

   matmul<<<dimGrid, dimBlock>>>(wA, wB, A, B, C); // launch instances of kernel function

   (void) cudaThreadSynchronize();                        // wait for kernel to finish
}
```

5D FFT Correlations from Complex Overlap Expressions

(Ritchie, Kozakov, Vajda, (2008) Bioinformatics, 24, 1865–1873)

Complex SHs, Y_{lm}: $$y_{lm}(\theta,\phi) = \sum_t U^{(l)}_{mt} Y_{lt}(\theta,\phi)$$

Complex coefficients: $$A_{nlm} = \sum_t a_{nlt} U^{(l)}_{tm}$$

Complex overlap: $$S = \sum_{kjsmnlv} D^{(j)*}_{ms}(0,\beta_A,\gamma_A) A^*_{kjs} T^{(|m|)}_{kj,nl}(R) D^{(l)}_{mv}(\alpha_B,\beta_B,\gamma_B) B_{nlv}$$

Collect coefficients: $$S^{(|m|)}_{js,lv}(R) = \sum_{kn} A^*_{kjs} T^{(|m|)}_{kj,nl}(R) B_{nlv}, \qquad k > j; n > l$$

To give: $$S = \sum_{jsmlv} D^{(j)*}_{ms}(0,\beta_A,\gamma_A) S^{(|m|)}_{js,lv}(R) D^{(l)}_{mv}(\alpha_B,\beta_B,\gamma_B)$$

Expand as exponentials: $$D^{(l)}_{mv}(\alpha,\beta,\gamma) = \sum_t \Gamma^{tm}_{lv} e^{-im\alpha} e^{-it\beta} e^{-iv\gamma}$$

Hence: $$S = \sum_{jsmlvrt} \Gamma^{rm}_{js} S^{(|m|)}_{js,lv}(R) \Gamma^{tm}_{lv} e^{-i(r\beta_A - s\gamma_A + m\alpha_B + t\beta_B + v\gamma_B)}$$

Translation Matrices From Fourier-Bessel Transform Theory

Using spherical Bessel transforms:

$$\tilde{R}_{nl}(\beta) = \sqrt{\frac{2}{\pi}} \int_0^\infty R_{nl}(r) j_l(\beta r) r^2 \mathrm{d}r; \qquad R_{nl}(r) = \sqrt{\frac{2}{\pi}} \int_0^\infty \tilde{R}_{nl}(\beta) j_l(\beta r) \beta^2 \mathrm{d}\beta$$

it can be shown that

$$T^{(|m|)}_{n'l',nl}(R) = \sum_{k=|l-l'|}^{l+l'} A_k^{(ll'|m|)} \int_0^\infty \tilde{R}_{nl}(\beta) \tilde{R}_{n'l'}(\beta) j_k(\beta R) \beta^2 \mathrm{d}\beta$$

where

$$A_k^{(ll'|m|)} = (-1)^{\frac{k+l'-l}{2}+m} (2k+1) \left[(2l+1)(2l'+1)\right]^{1/2} \begin{pmatrix} l & l' & k \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} l & l' & k \\ m & \overline{m} & 0 \end{pmatrix}$$

- **Can derive analytic formulae for both GTO and ETO radial functions**
- **Requires high precision math library (GMP)...**
- **Calculate once for $R = 1, 2, 3, ...50\mathring{A}$ and store on disk ($\sim$ 200Mb)**