Tutorial: Tensor Approximation in Visualization and Graphics

Implementation Examples in Scientific Visualization

Renato Pajarola, **Susanne K. Suter**, and Roland Ruiters

University of Zurich[UZH]

VISUALIZATION AND MULTIMEDIA LAB

universität bonn

Institute of Computer Science II
Computer Graphics

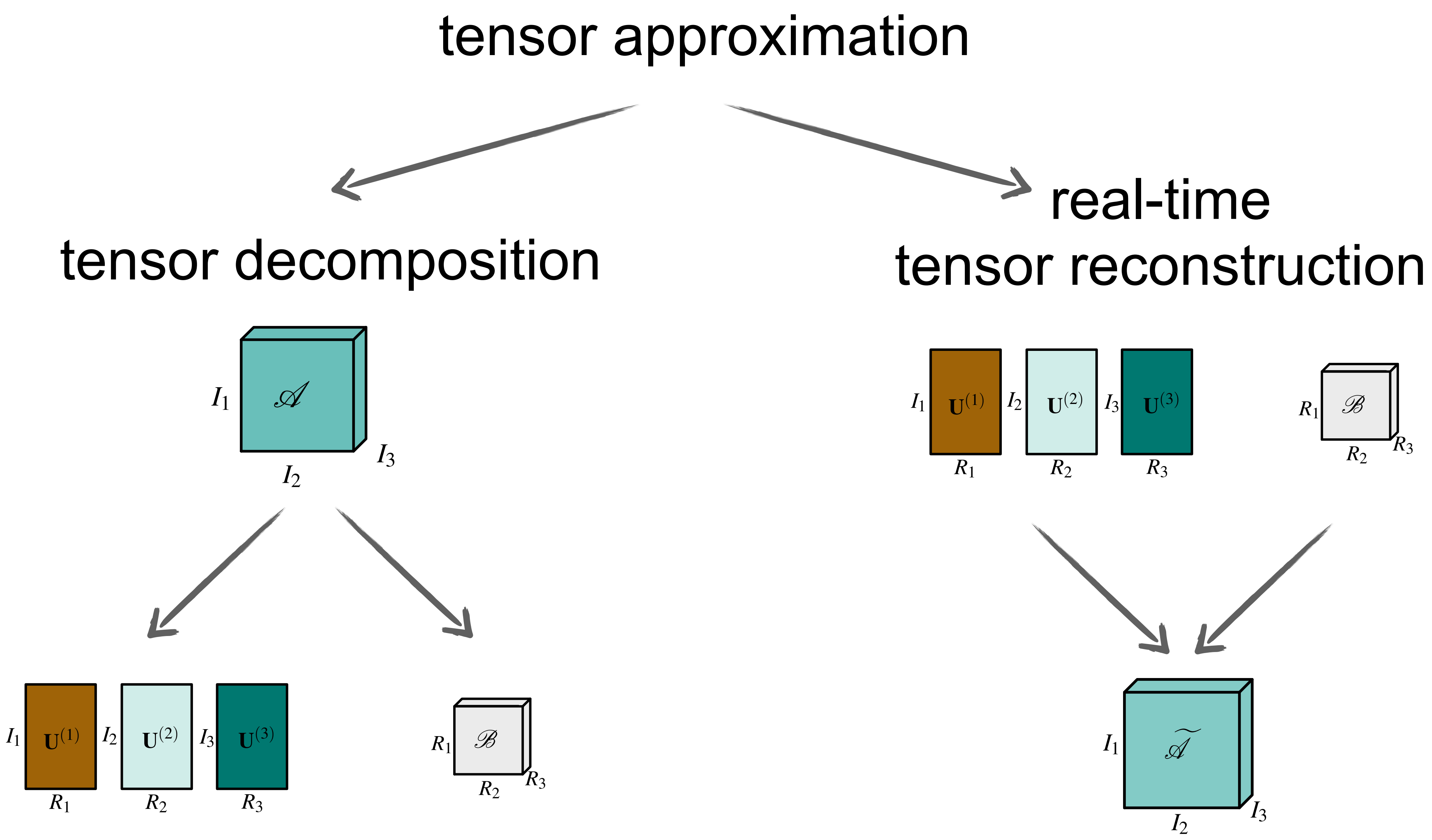
Tutorial Continued...

- **Tuesday May. 7 from 9:00 to 10:40**
- Location: Room B.1
  - Implementation Examples in Scientific Visualization (Suter, 25min)
  - Graphics Applications (Ruiters, 30min)
  - Clustering and Sparsity (Ruiters, 25min)
  - Summary/Outlook (Pajarola, 10min)

Outline

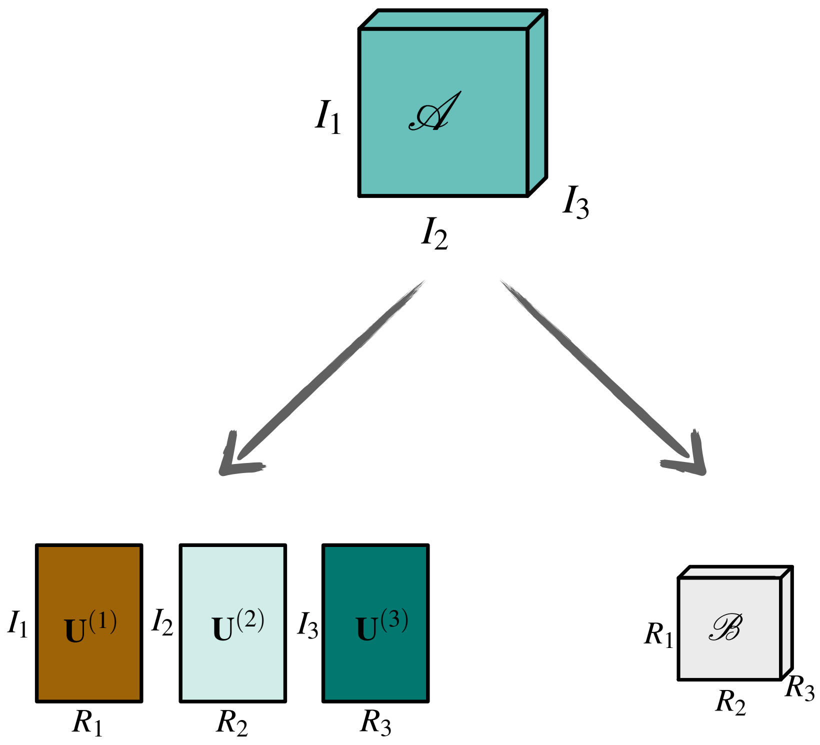
- Part 1: Typical decomposition algorithms/operations
- Part 2: GPU-based tensor reconstruction

Typical TA Operations



Tensor Decomposition

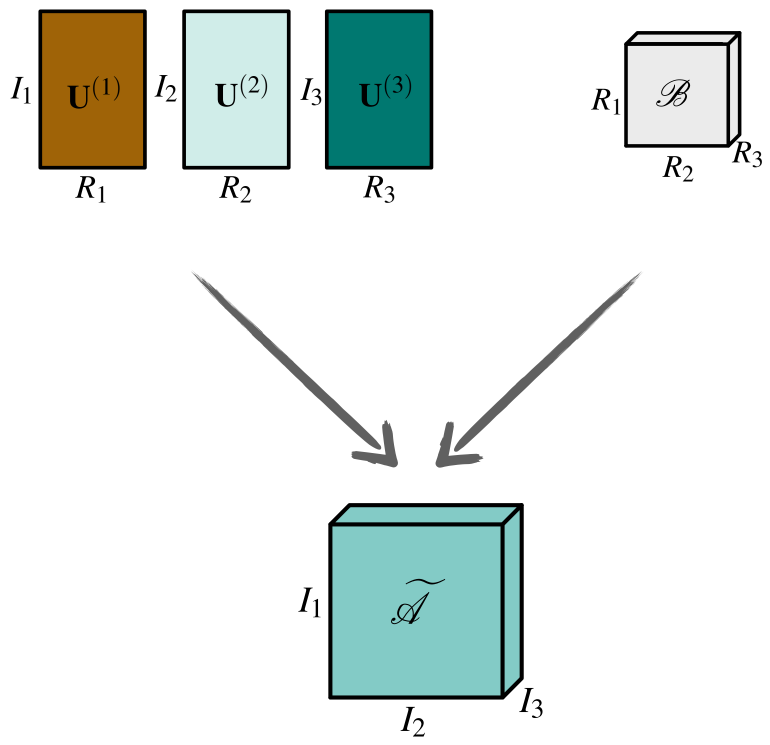
- Create factor matrices
  - higher-order SVD (HOSVD)
    - tensor unfolding
  - alternating least-squares (ALS) algorithms
    - higher-order orthogonal iteration (HOOI)
    - higher-order power method (HOPM)
- Generate core tensor
  - tensor times matrix (TTM) multiplications



universitätbonn

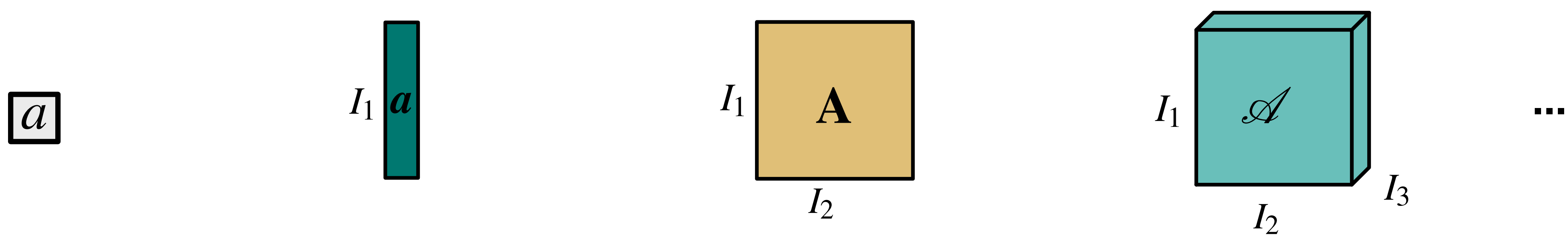
Tensor Reconstruction

- Realtime (!) reconstruction
  - tensor times matrix (TTM) multiplications



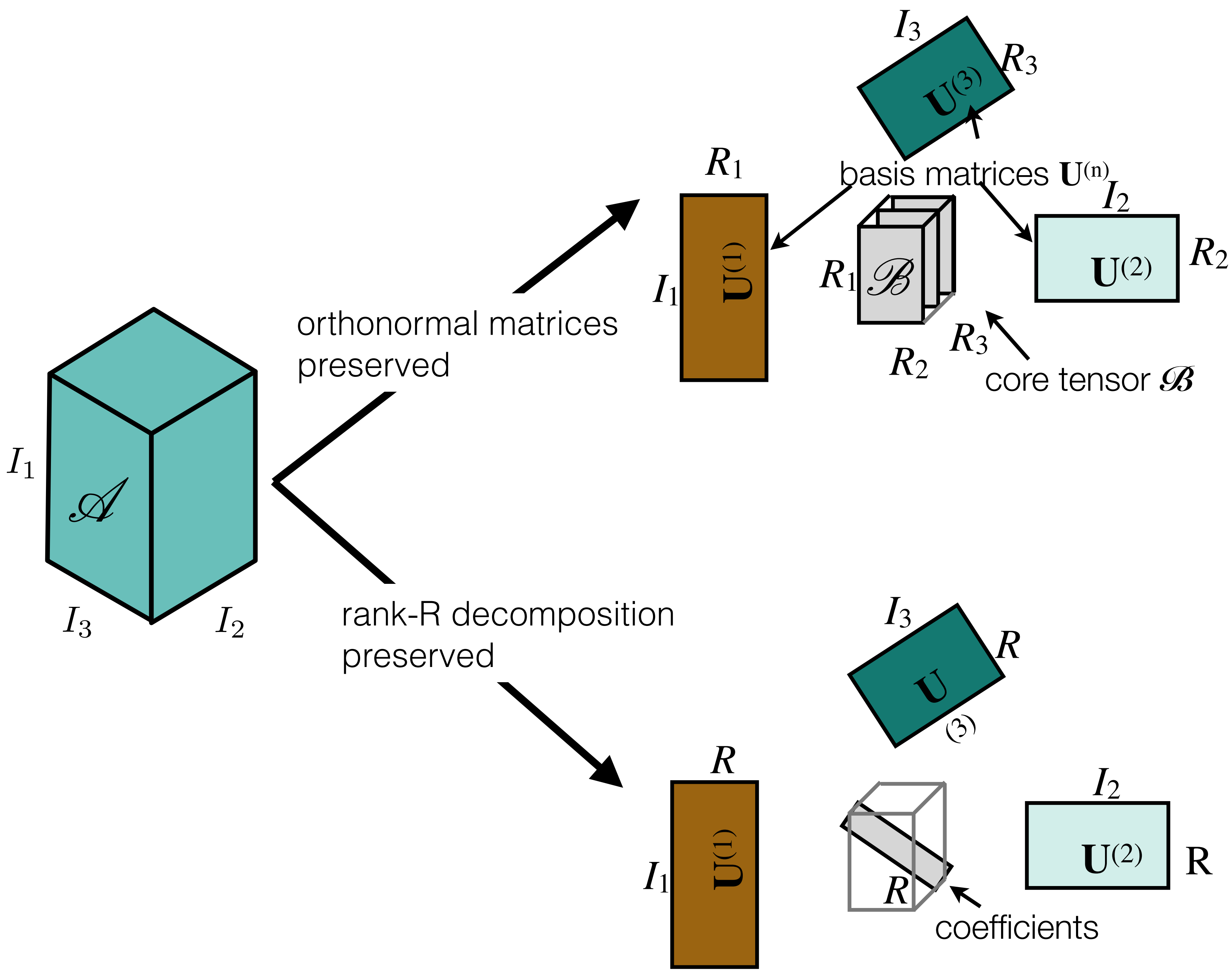
Tensor: A Multidimensional Array

0^{th}-order tensor | 1^{st}-order tensor | 2^{nd}-order tensor | 3^{rd}-order tensor



$i_1 = 1, \ldots, I_1$ $i_2 = 1, \ldots, I_2$ $i_3 = 1, \ldots, I_3$

SVD Extension to Higher Orders



Tucker

- Three-mode factor analysis (**3MFA/Tucker3**) [Tucker, 1964+1966]
- Higher-order SVD (**HOSVD**) [De Lathauwer et al., 2000a]

CP

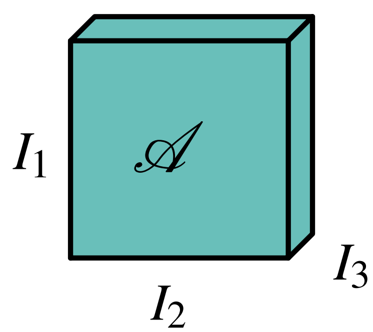
- **PARAFAC** (parallel factors) [Harshman, 1970]
- **CANDECOMP** (CAND) (canonical decomposition) [Caroll & Chang, 1970]

Part 1: Typical Decomposition Algorithms and Operations

Downloads

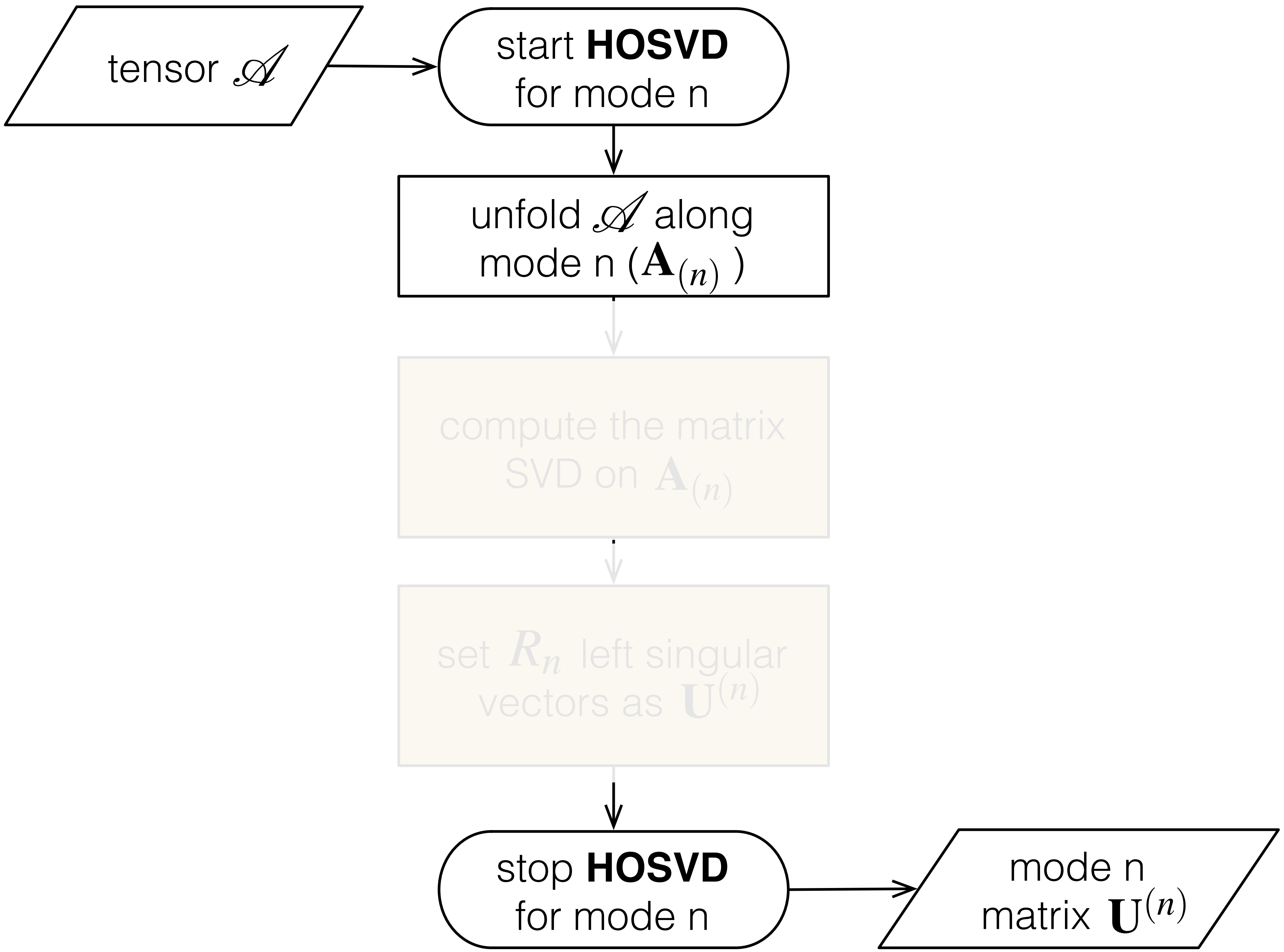
- MATLAB tensor toolbox
  - http://www.sandia.gov/~tgkolda/TensorToolbox
- vmmlib: C++ library for vectors, matrices, and tensor approximation
  - http://vmml.github.io/vmmlib/
- Tensor tutorial notes
  - http://vmml.ifi.uzh.ch/links/TutorTensorAprox.html

Test Dataset: Hazelnut



- A microCT scan of dried hazelnuts
- $I_1 = I_2 = I_3 = 512$
- Values: unsigned char (8bit)
- http://vmml.ifi.uzh.ch/research/datasets.html

Higher-order SVD (HOSVD)

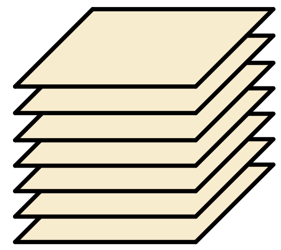


De Lathauwer, de Moor, Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

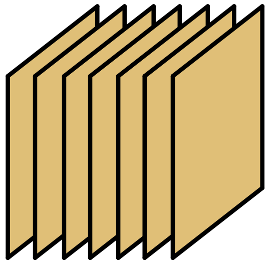
Slices of a Tensor3

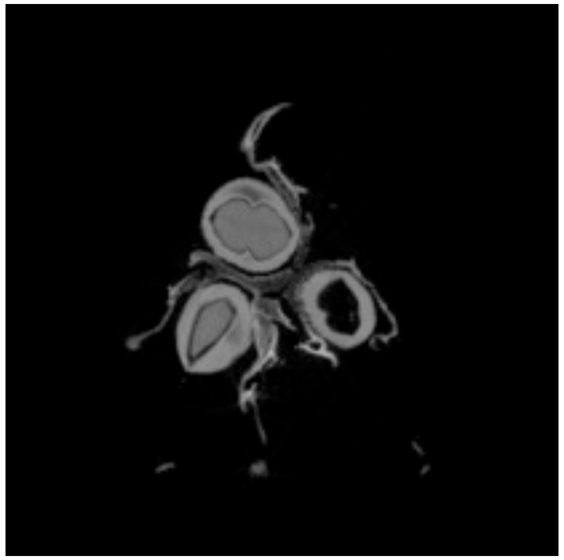
frontal slices

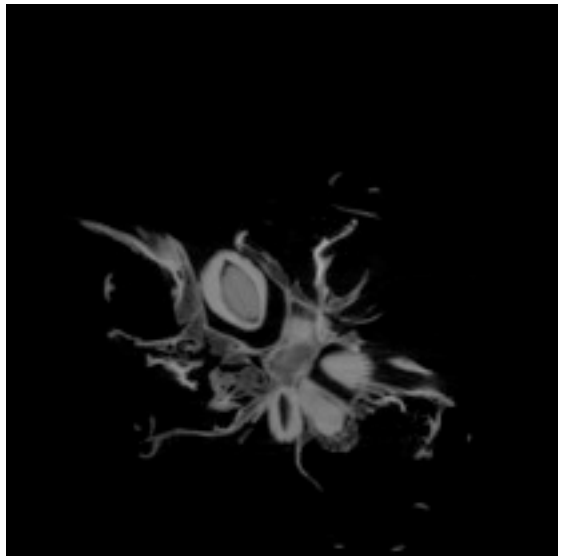
horizontal slices

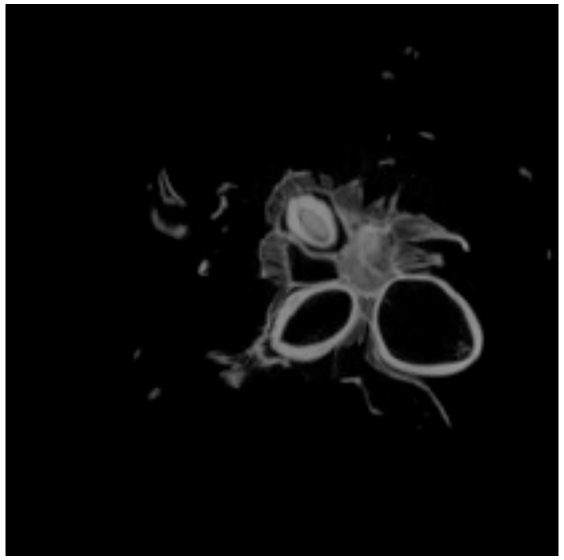


lateral slices









[vmmlib]

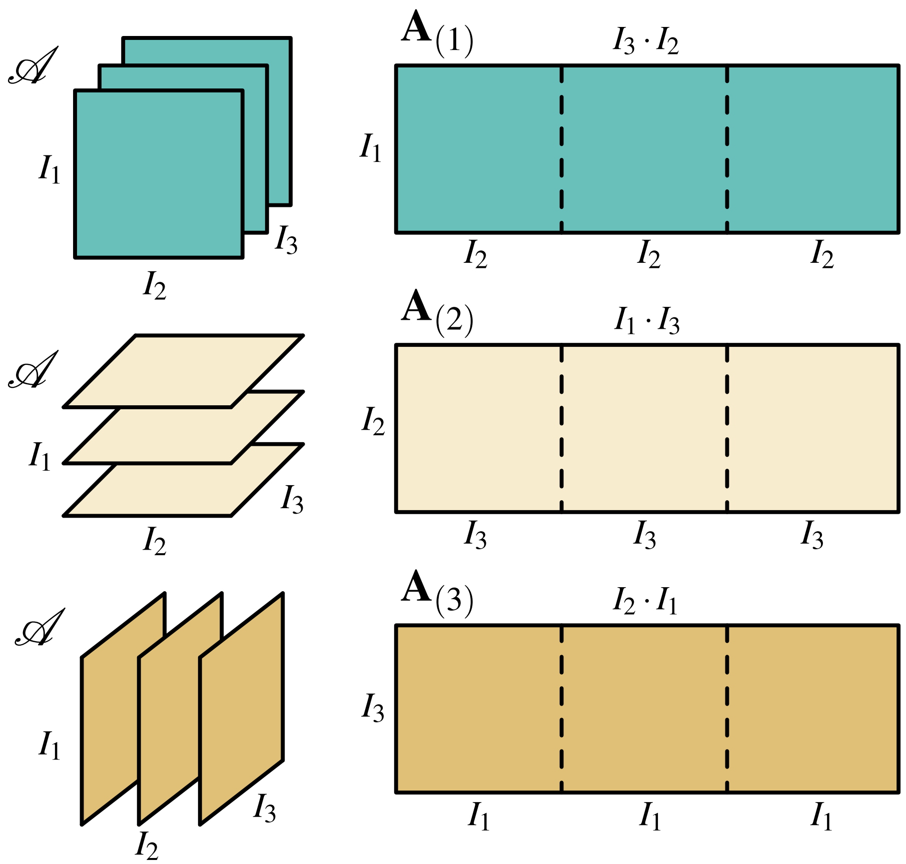
```
matrix< 512, 512, values_t > slice;
t3.get_frontal_slice_fwd( 256, slice );
t3.get_horizontal_slice_fwd( 256, slice );
t3.get_lateral_slice_fwd( 256, slice );
```

VISUALIZATION AND MULTIMEDIA LAB

universität bonn

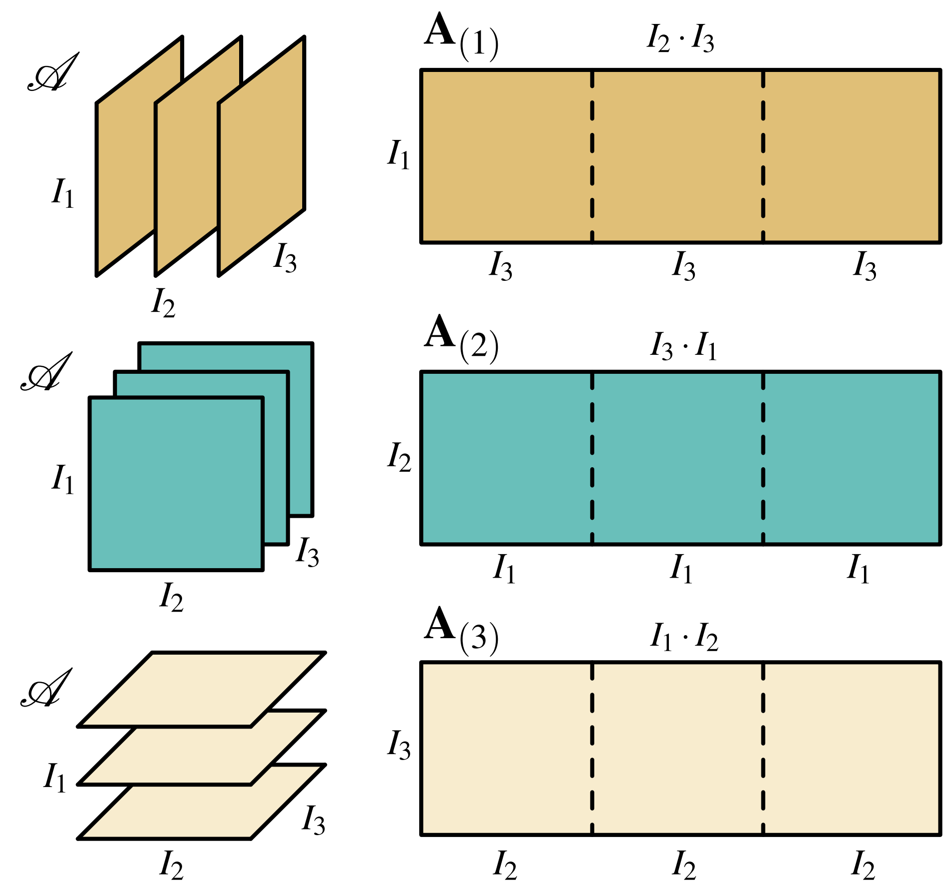
Tensor Unfolding (Matricization)

forward cyclic unfolding



Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics*, 14(3):105–122, 2000.

backward cyclic unfolding



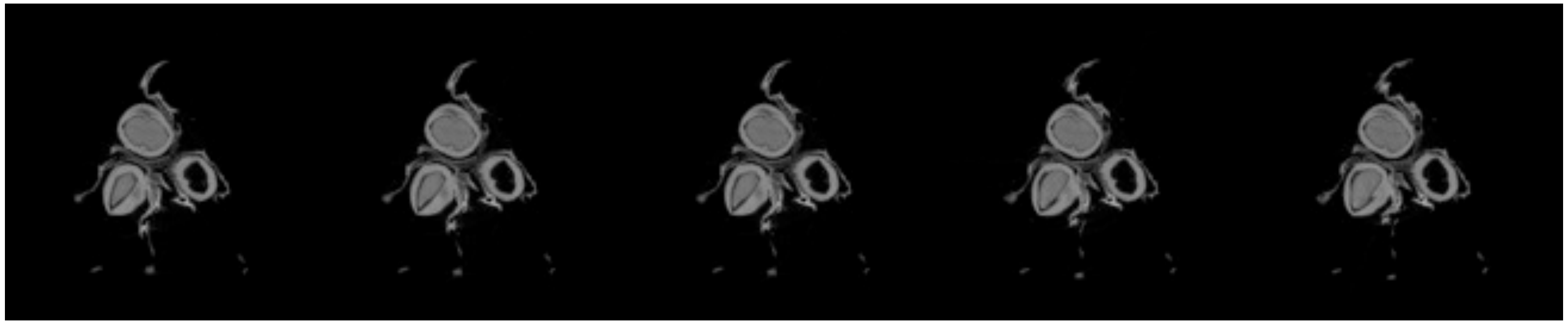
De Lathauwer, de Moor, Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

Tensor Unfolding Example

262’144

512

mode-1 unfolding …

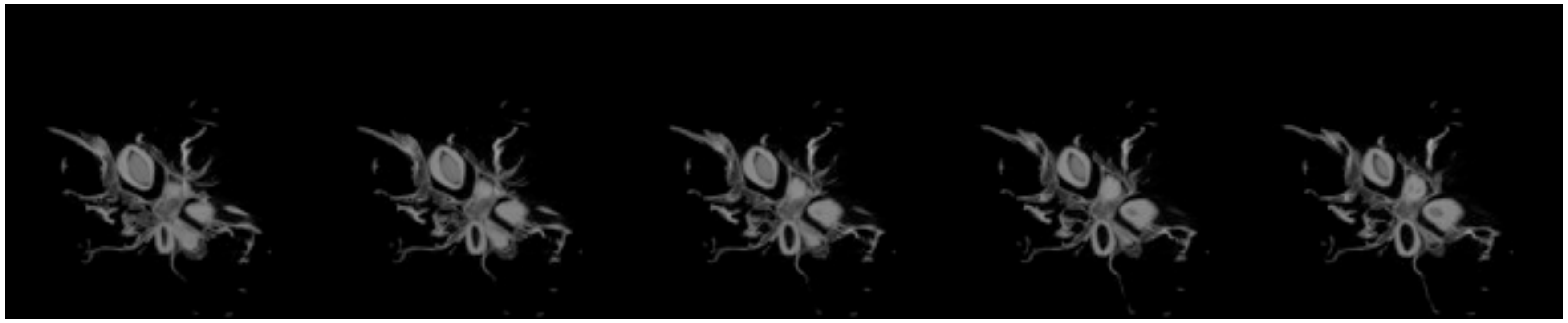


…

262’144

512

mode-2 unfolding …

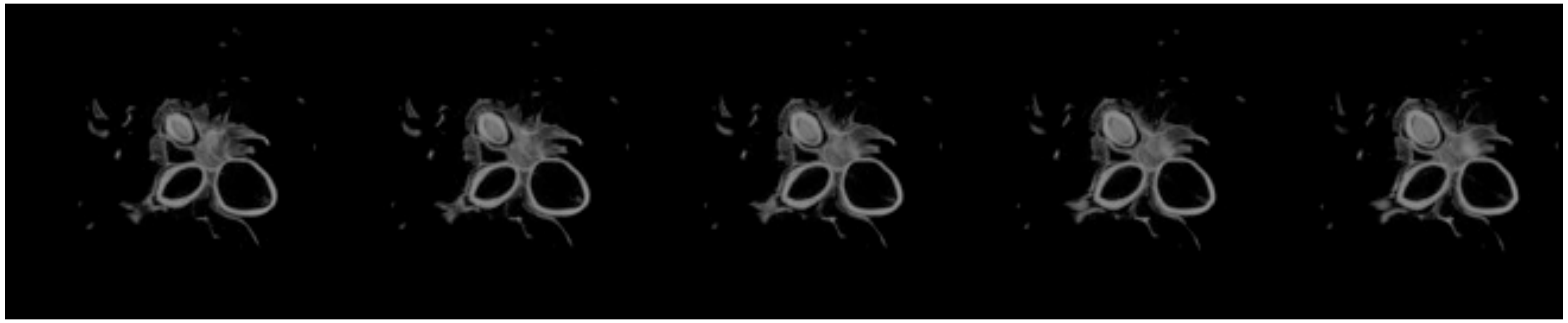


…

262’144

512

mode-3 unfolding …

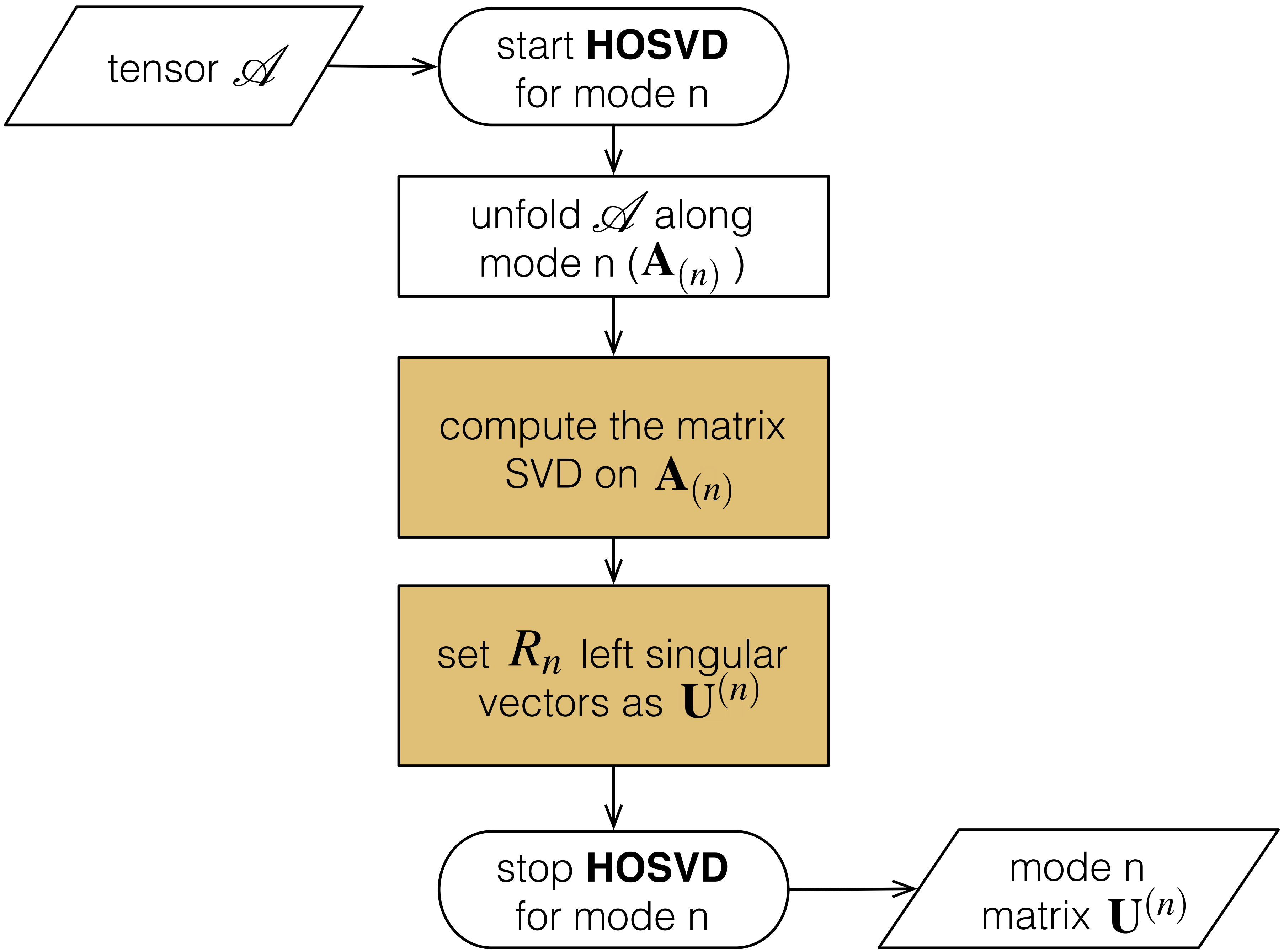


…

VISUALIZATIONAND MULTIMEDIALAB

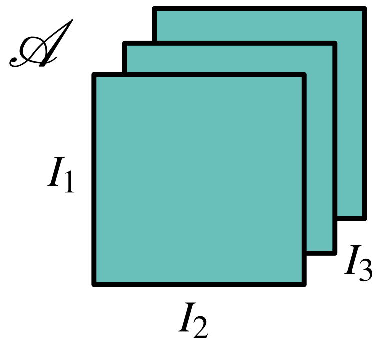
universitätbonn

Higher-order SVD (HOSVD)



De Lathauwer, de Moor, Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

Large Data Tensors (in vmmlib)



[vmmlib]

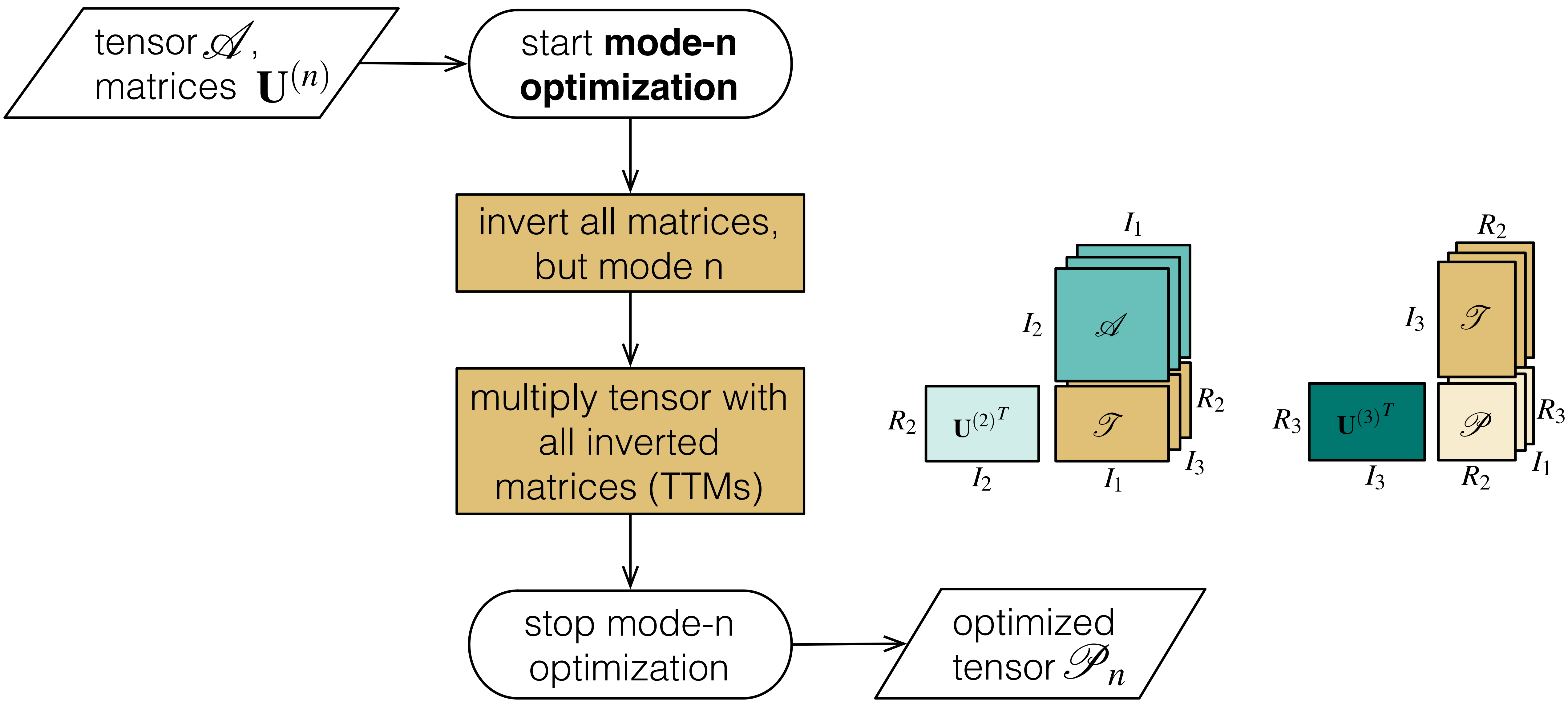
```
const size_t d = 512;
typedef tensor3< d,d,d, unsigned char > t3_512u_t;
typedef t3_converter< d,d,d, unsigned char > t3_conv_t;
typedef tensor_mmapper< t3_512u_t, t3_conv_t > t3map_t;


std::string in_dir = "./dataset";
std::string file_name = "hnut512_uint.raw";
t3_512u_t t3_hazelnut;
t3_conv_t t3_conv;

t3map_t t3_mmap( in_dir, file_name, true, t3_conv ); //true -> read-only
t3_mmap.get_tensor( t3_hazelnut );
```

Optimize Factor Matrices

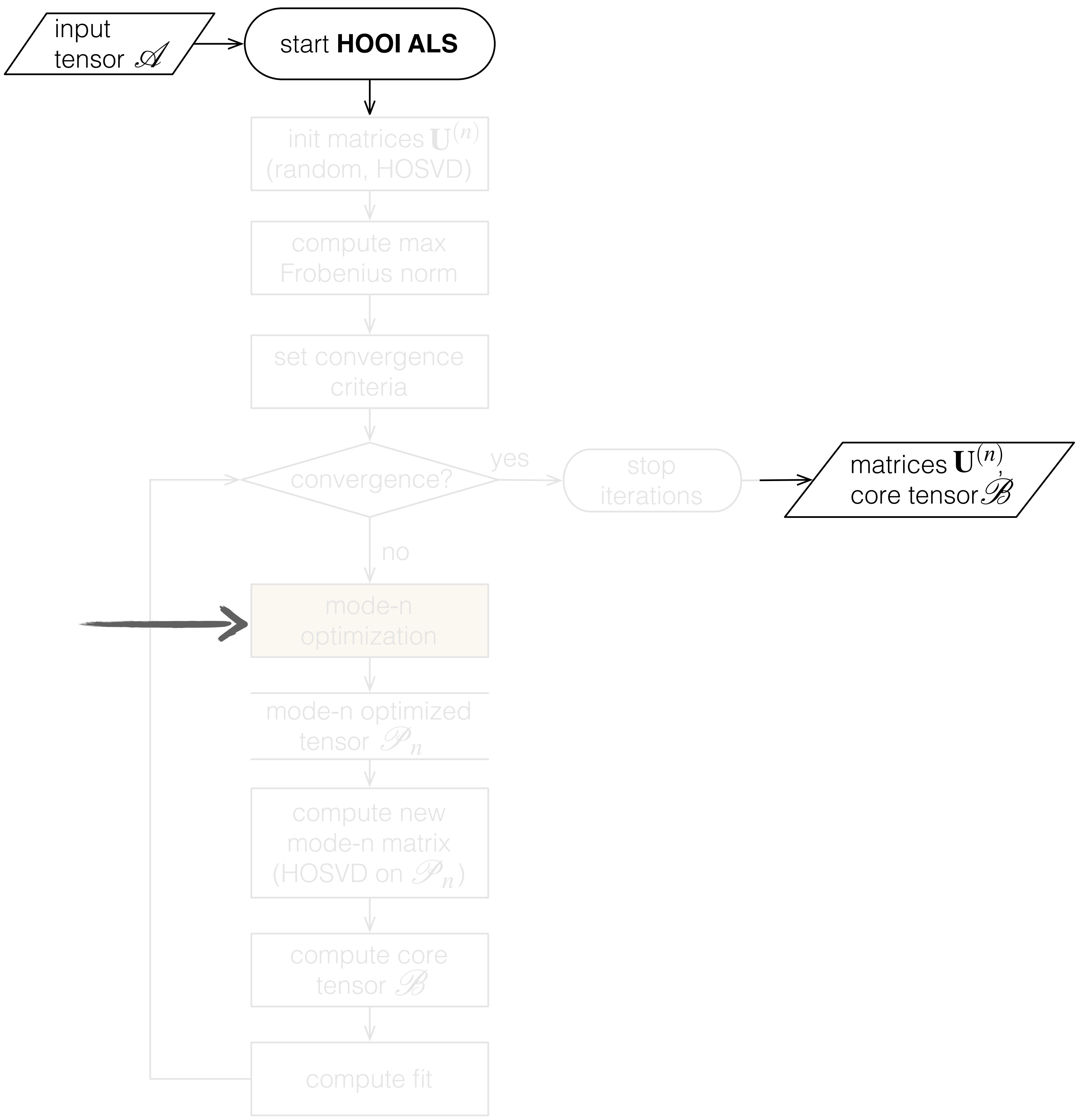
- Higher-order orthogonal iteration
  - optimize factor matrix of mode n
  - keep factor matrices of all other modes fixed
  - generate optimized data tensor
    - project original data tensor on the inverted factor matrices of all other modes
  - receive optimized mode-n factor matrix
    - apply HOSVD to the optimized tensor

De Lathauwer, de Moor, Vandewalle. On the best rank-1 and rank-(R1,R2,...,RN) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

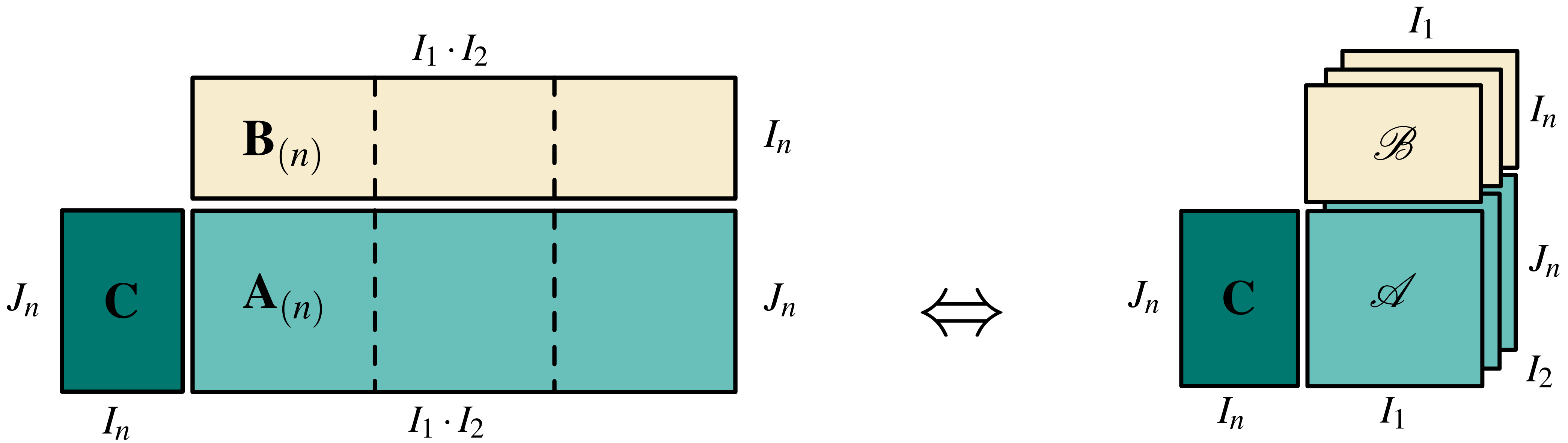
Optimize Mode-n Factor Matrix



Higher-order Orthogonal Iteration (HOOI)



Tensor Times Matrix Multiplication



$\mathbf{A}_{(n)} = \mathbf{C}\mathbf{B}_{(n)}$

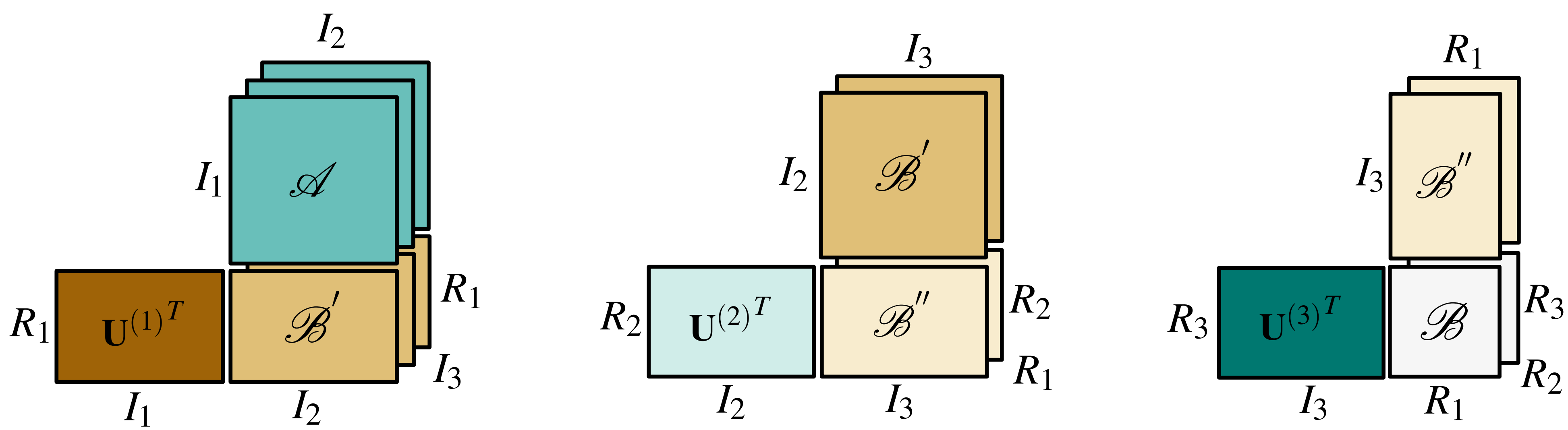
$\mathscr{A} = \mathscr{B} \times_n \mathbf{C}$

n-mode product

[De Lathauwer et al., 2000a]

$$(\mathscr{B} \times_n \mathbf{C})_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} b_{i_1 i_2 \ldots i_N} \cdot c_{j_n i_n}$$

Example TTMs: Core Computation



$$\mathscr{B} = \mathscr{A} \times_1 \mathbf{U}^{(1)^{(-1)}} \times_2 \mathbf{U}^{(2)^{(-1)}} \times_3 \cdots \times_N \mathbf{U}^{(N)^{(-1)}}$$

orthogonal → **factor matrices**

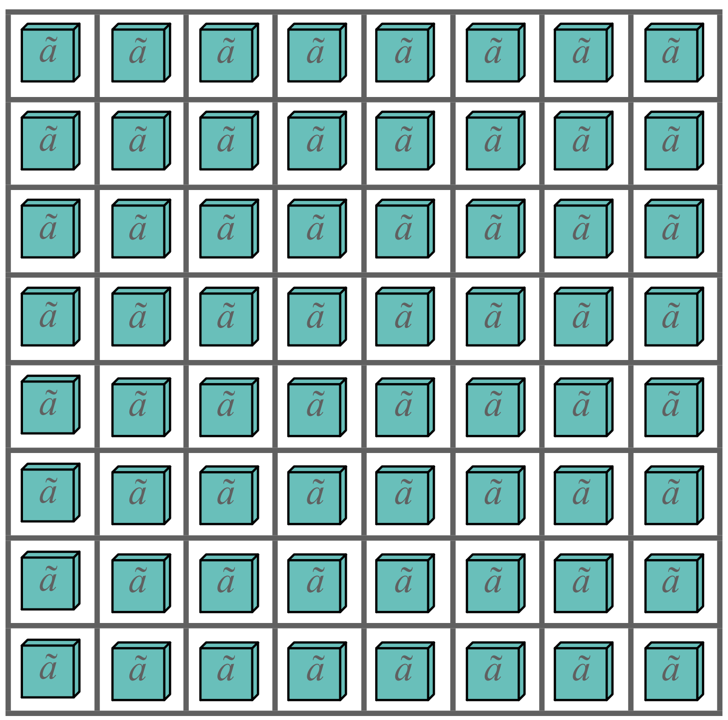
$$\mathscr{B} = \mathscr{A} \times_1 \mathbf{U}^{(1)^T} \times_2 \mathbf{U}^{(2)^T} \times_3 \cdots \times_N \mathbf{U}^{(N)^T}$$

- Three consecutive TTM multiplications
- For orthogonal matrices, use the transposes of the three factor matrices (otherwise the (pseudo)-inverses)

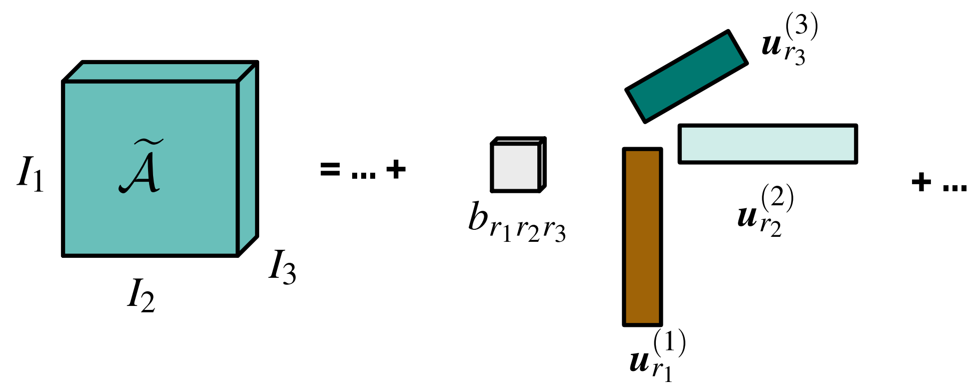
Part 2:
GPU Reconstruction

Suter et al.. Interactive multiscale tensor reconstruction for multiresolution volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2135–2143, December 2011.

Parallel Tensor Reconstruction



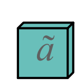
parallel computing grid per brick



$$\tilde{a}_{i_1 i_2 i_3} = \sum_{r_1}\sum_{r_2}\sum_{r_3} b_{r_1 r_2 r_3} \cdot u^{(1)}_{i_1 r_1} \cdot u^{(2)}_{i_2 r_2} \cdot u^{(3)}_{i_3 r_3}$$

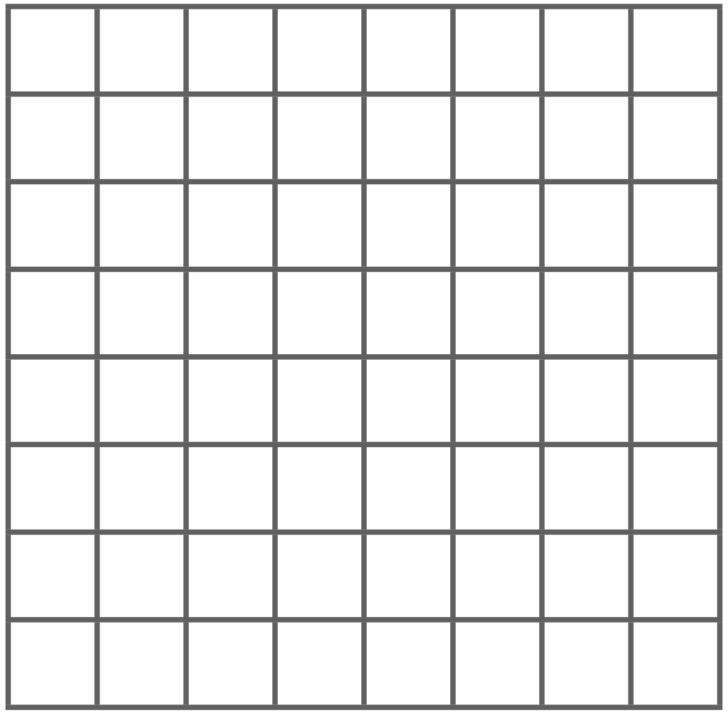
↑

triple-for-loop



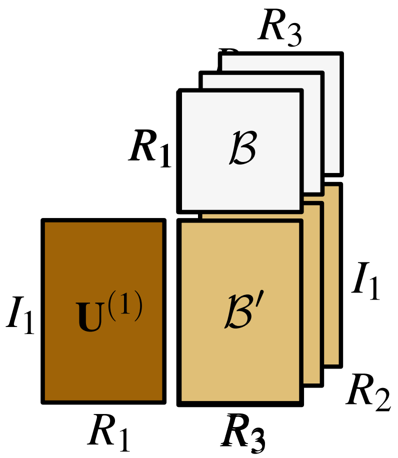
computational cost per voxel is <u>cubic</u>: $O(R^3)$

Faster Parallel Tensor Reconstruction

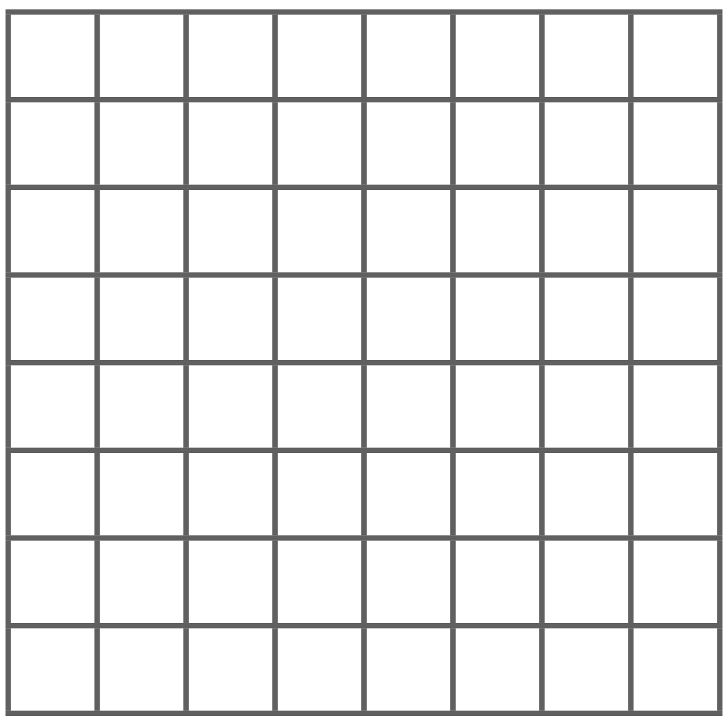


parallel computing grid per brick

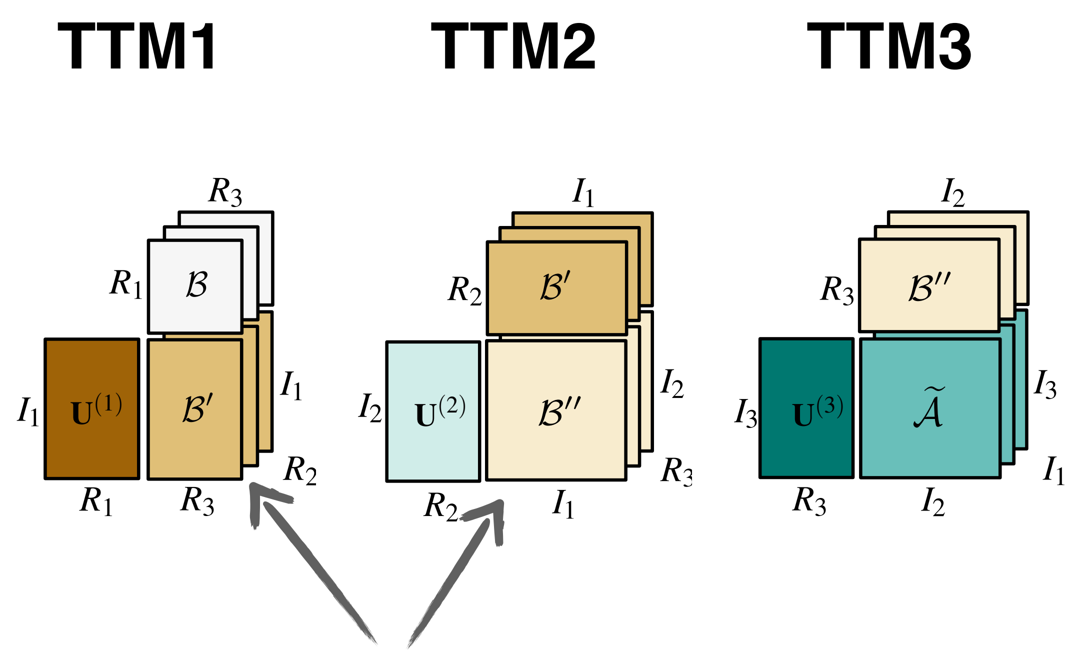
tensor times matrix (TTM) multiplication or n-mode product



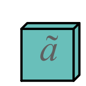
Faster Parallel Tensor Reconstruction



parallel computing grid per brick

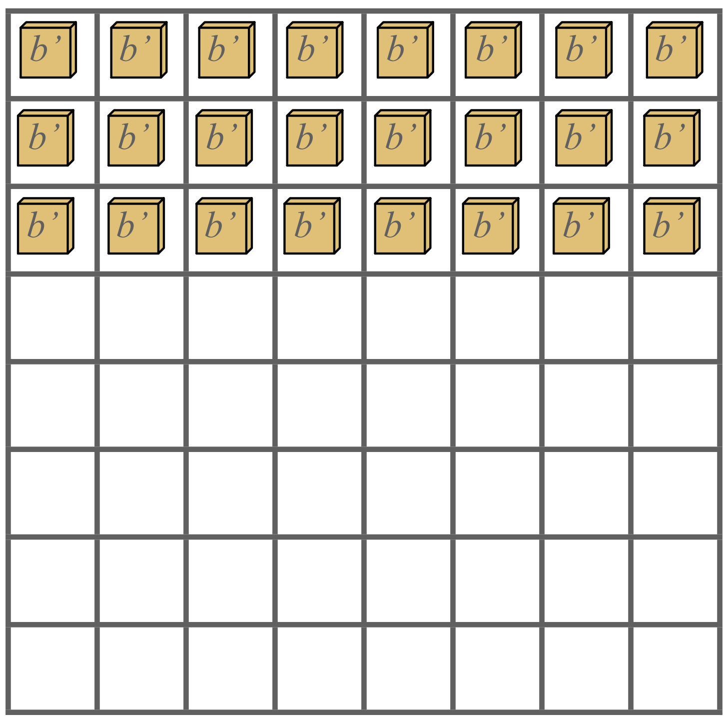


store intermediate results (B' and B'')



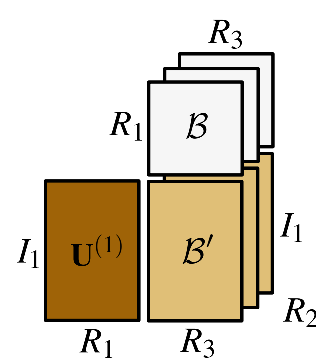
computational cost per voxel is linear: O(R)

Compute Intermediate Tensor B’

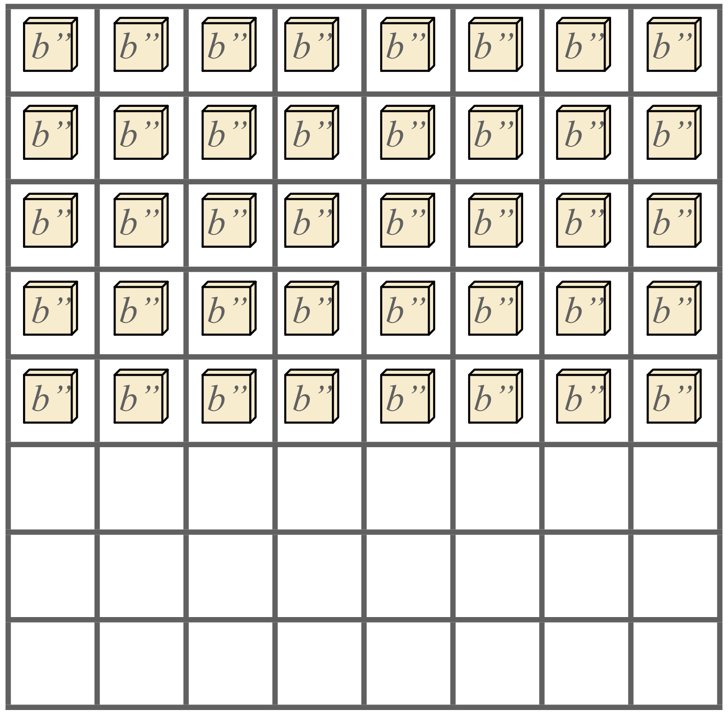


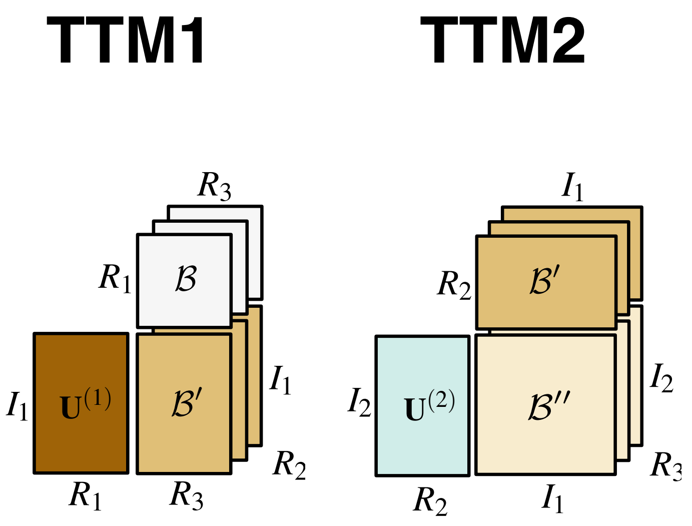
parallel computing grid per brick

TTM1



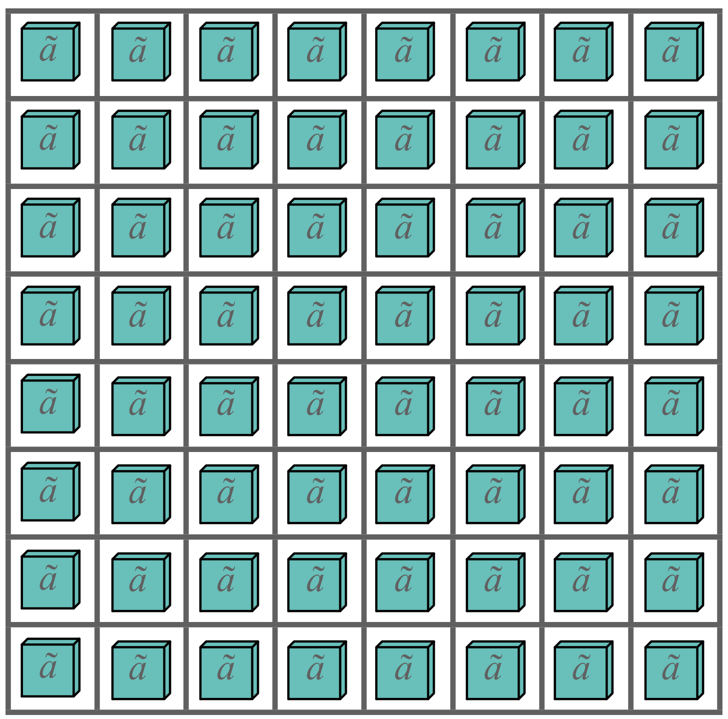
Compute Intermediate Tensor B”

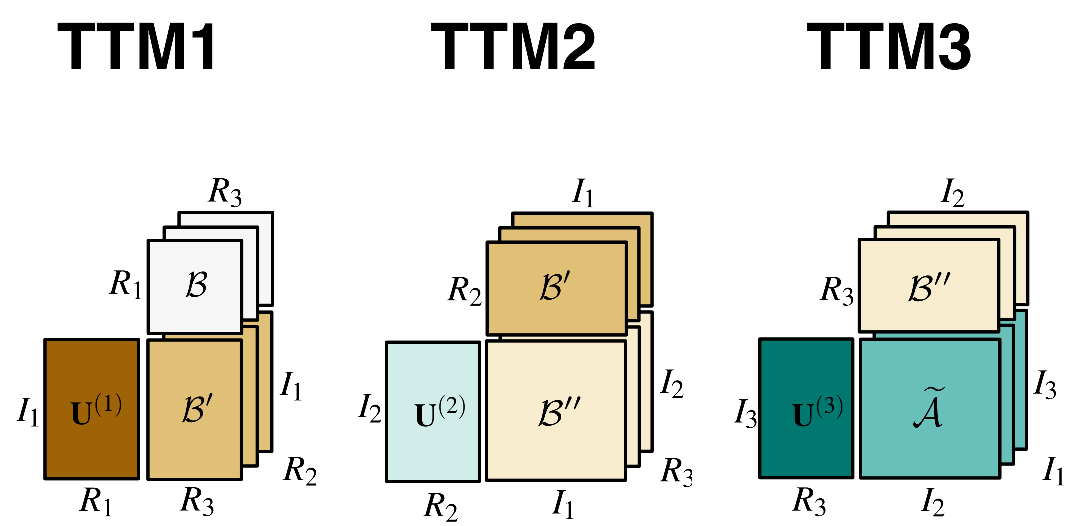




parallel computing grid per brick

Compute Approximated Tensor Ã

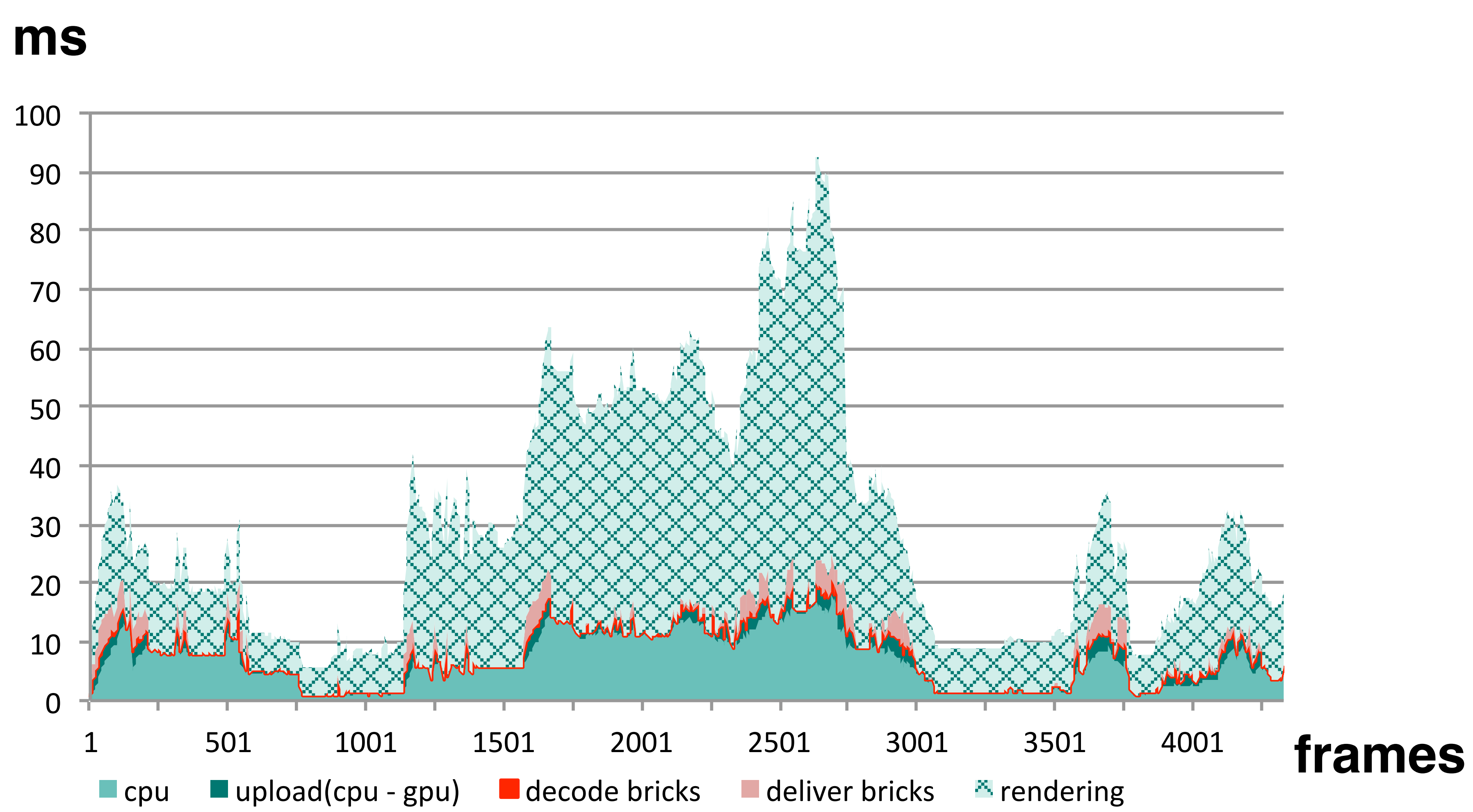


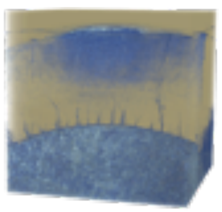


parallel computing grid per brick

Reconstruction Performance

2048^3





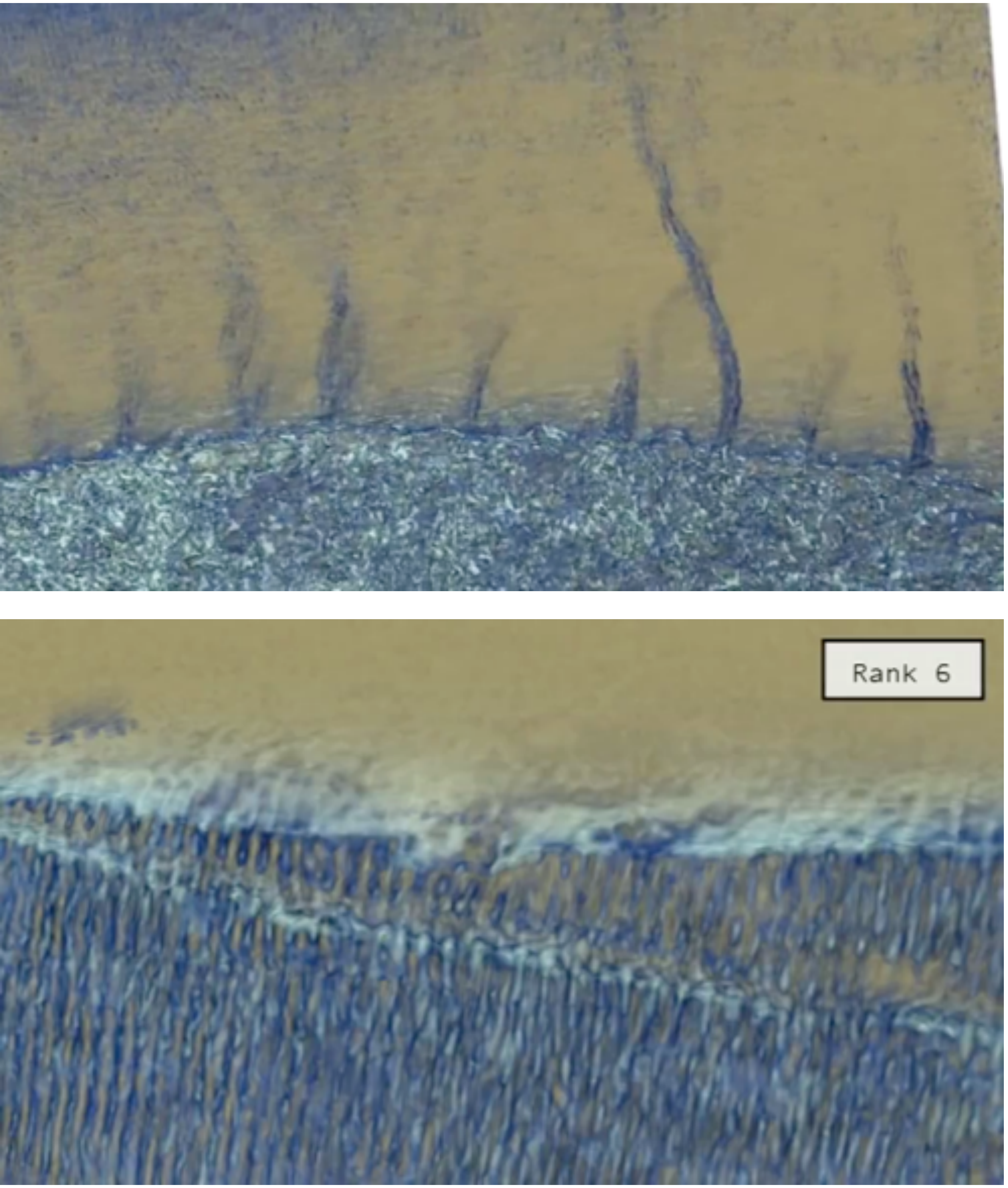
- Intel Core 2 E8500 3.2GHz Linux PC, 4GB memory
- NVIDIA GeForce GTX 480, 1.5GB memory

great ape molar (17GB -> 5.5 GB)

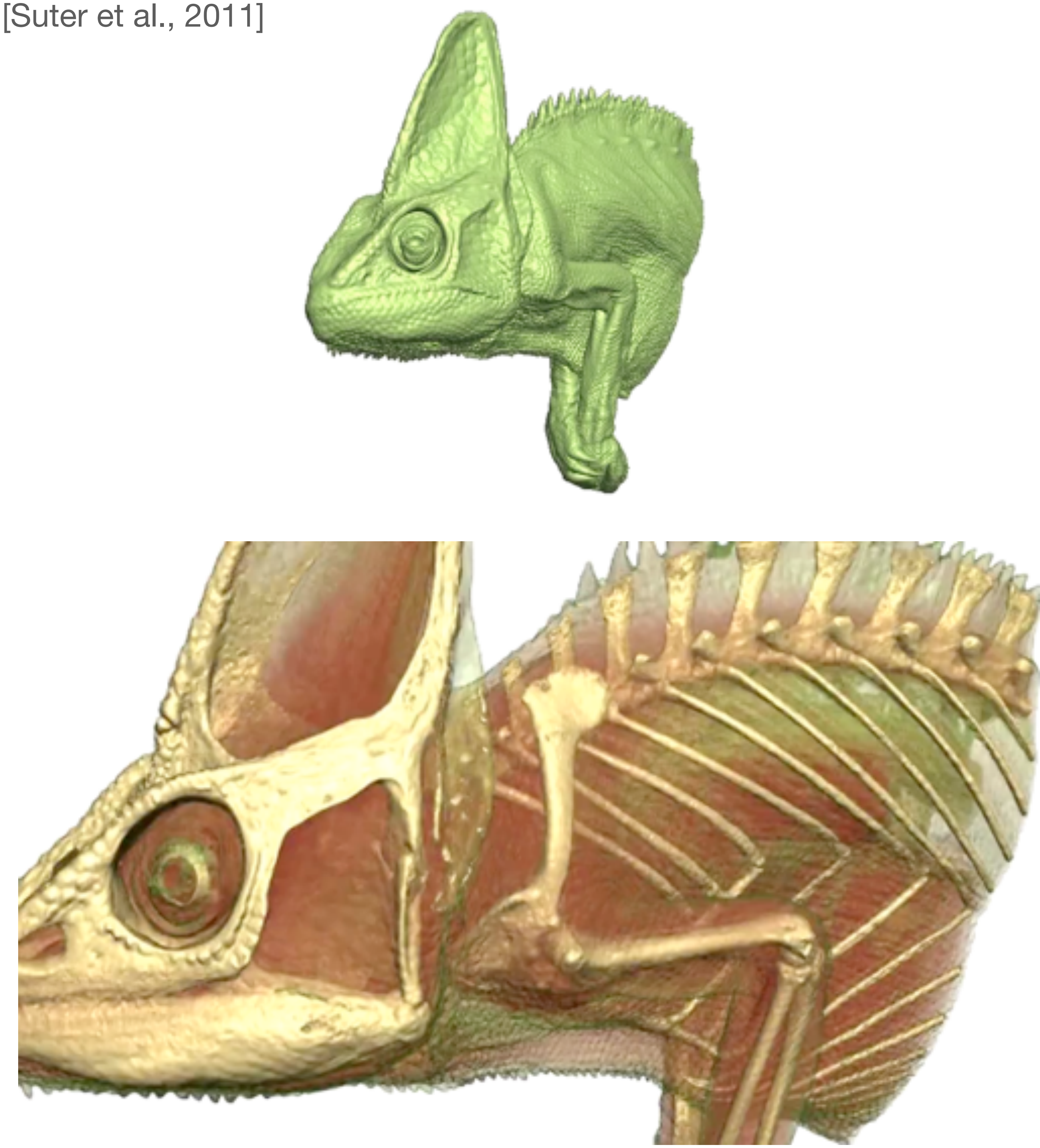
demo videos

http://www.youtube.com/user/VMMLuzh

chameleon (2 GB -> 230 MB)



[Suter et al., 2011]



Conclusion

- Critical implementation steps
- Tensor decomposition
  - initial decomposition or a large input tensor
    - memory mapping
  - tensor times matrix (TTM) multiplications
    - parallel matrix matrix multiplications
  - higher-order SVD
- Tensor reconstruction
  - GPU implementation of TTM