Tutorial: Tensor Approximation in Visualization and Graphics

Clustering and Sparsity

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

University of Zurich[UZH]

VISUALIZATIONANDMULTIMEDIALAB

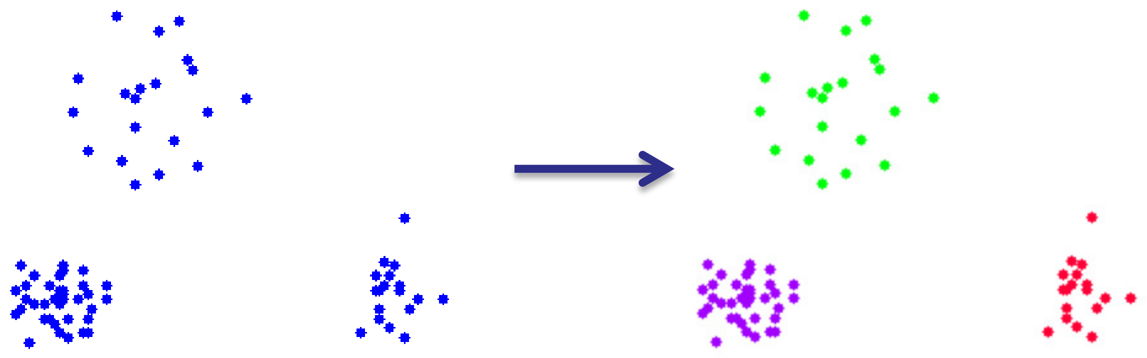
universitätbonn

Institute of Computer Science II
Computer Graphics

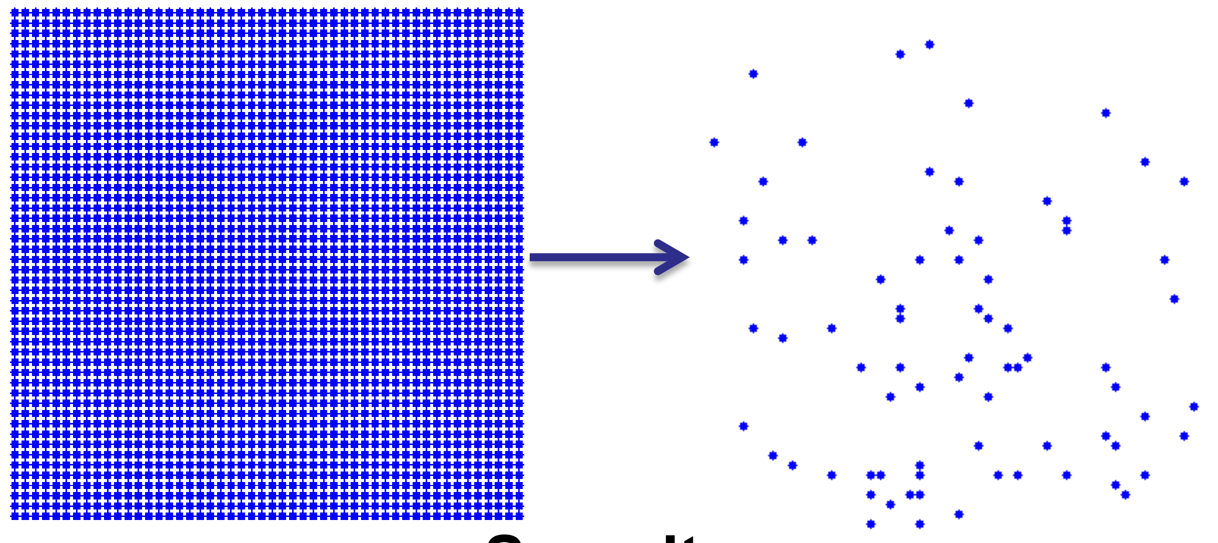
Clustering and Sparsity

Clustered/Sparse output

- Several decomposition techniques utilize either clustering or sparsity to
  - Increase the compression ratio
  - Reduce the decompression time



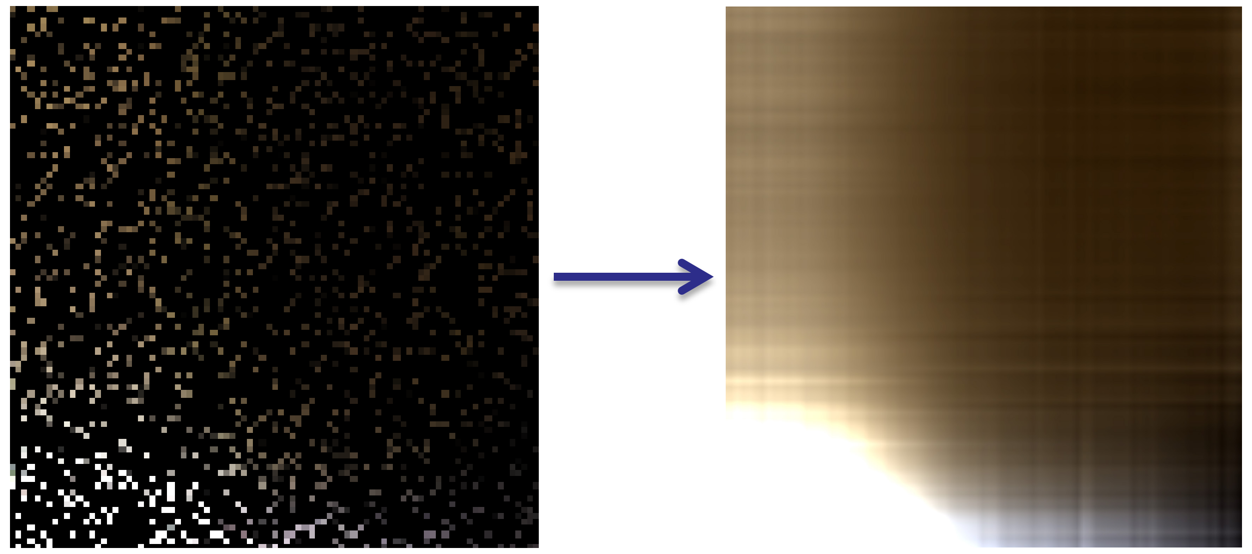
Clustering



Sparsity

Sparse Input

- How to handle missing values?
- How to cope with sparse and irregular input samplings?

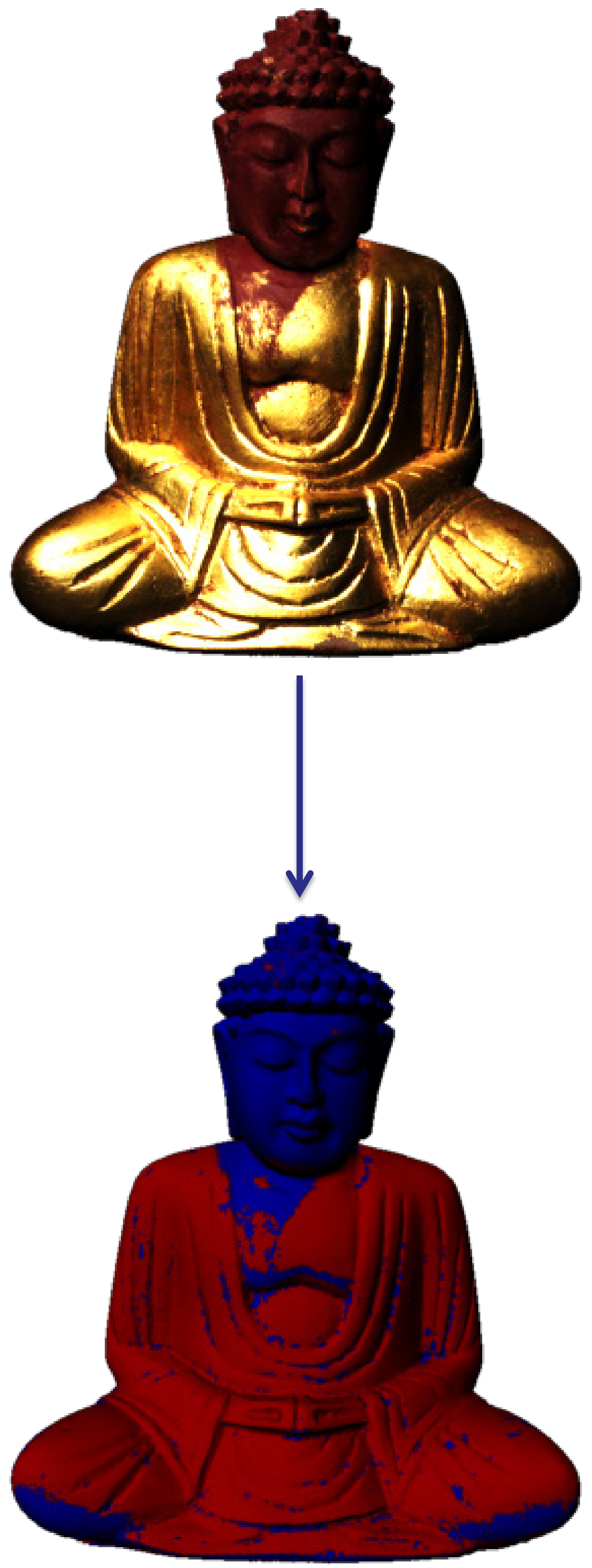


Sparse and irregular input

Clustered Tensor Approximation

- Some datasets are composed of several parts which are mostly independent
  - E.g. a surface composed of several different materials
  - There is no correlation between these parts which can be exploited for compression

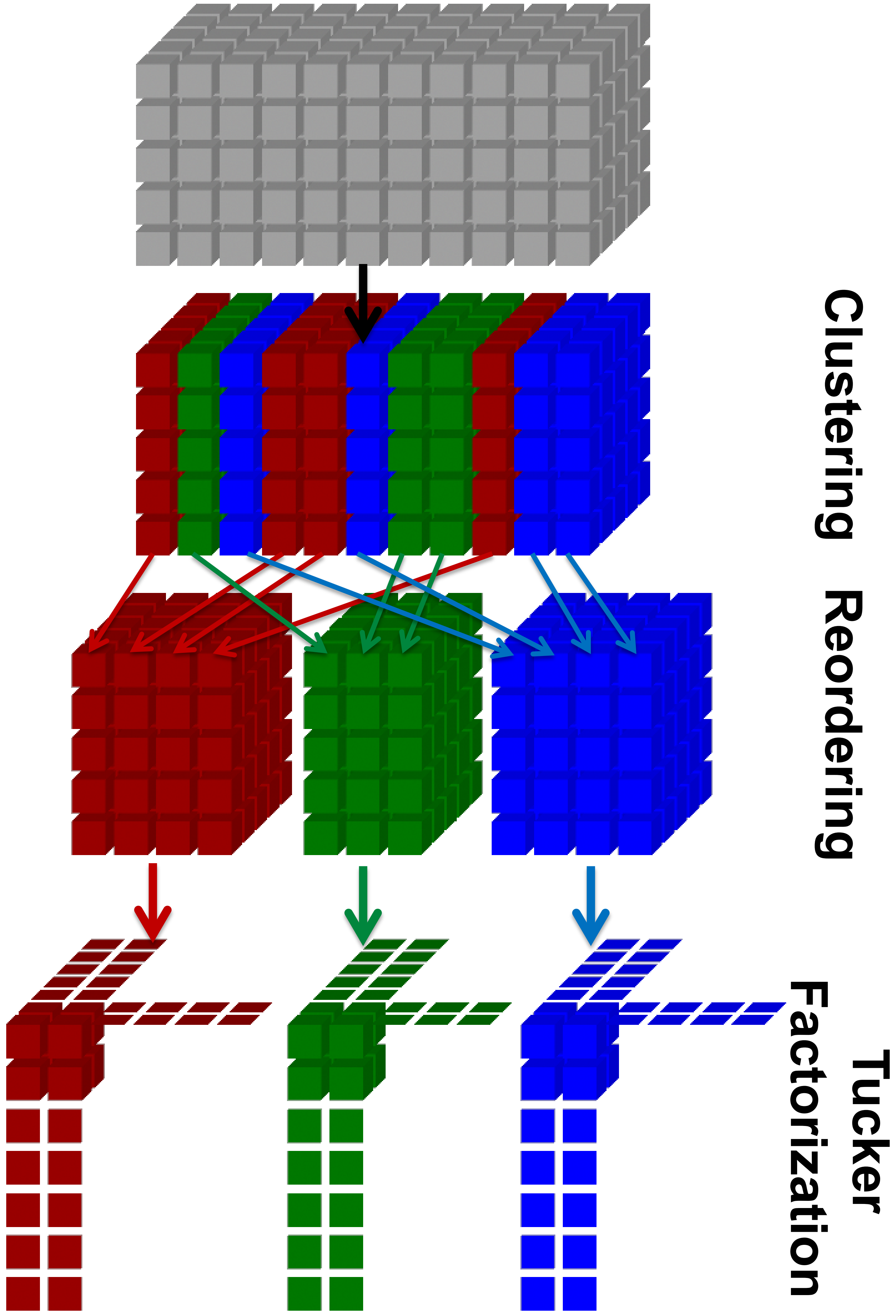
- Combine clustering and tensor approximation
  - Proposed in [Tsai-2006]
  - Extension of Clustered PCA to tensors



Clustered Tensor Approximation

- The tensor is clustered along one of its modes
- All slices corresponding to one cluster are grouped into new tensors
- For each of these tensors a Tucker factorization is performed

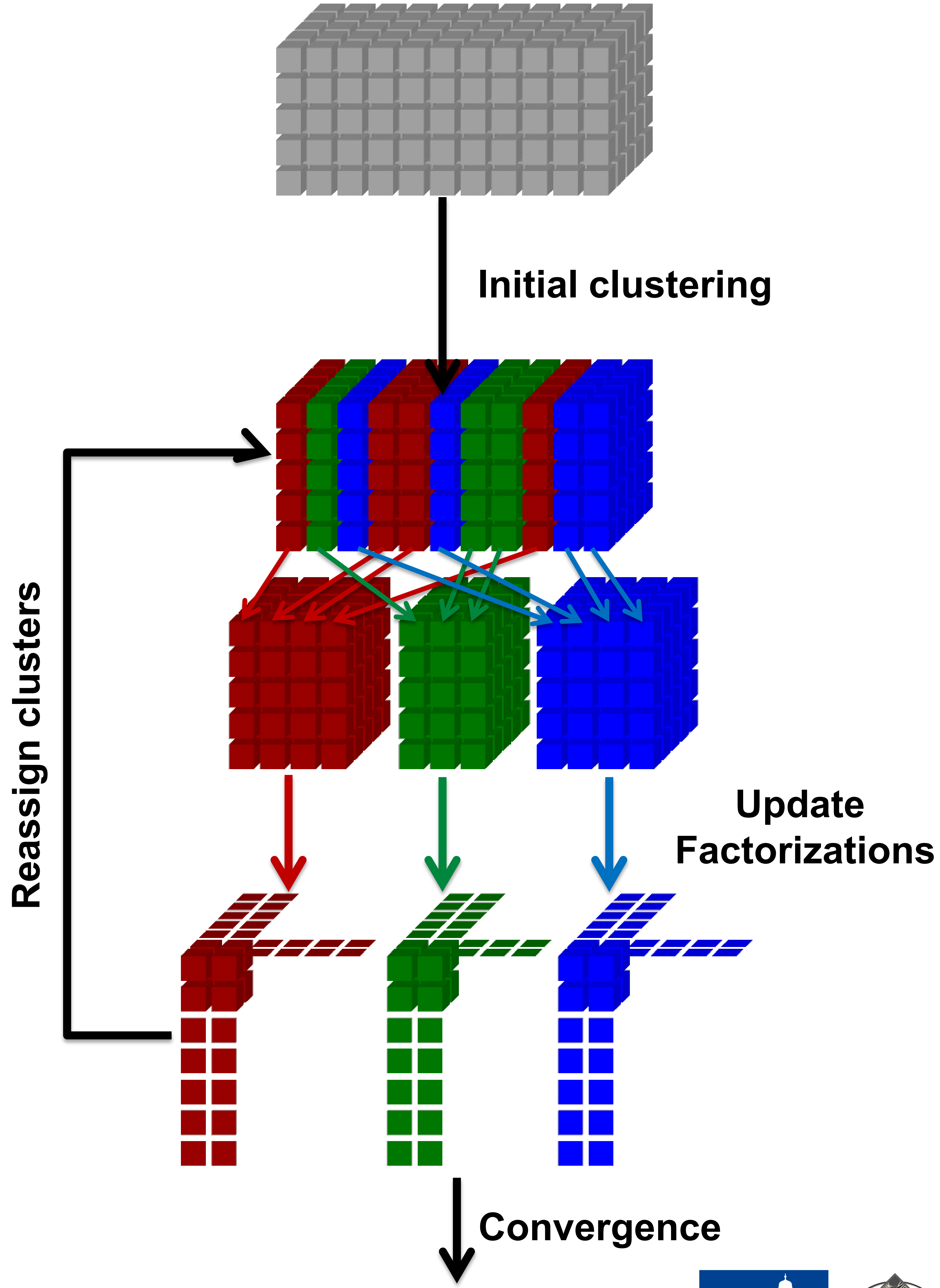
- Each of the individual clusters can be compressed with a smaller core tensor
  - Faster decompression
  - Potentially better compression ratio
    - Only if a good clustering is possible!



Clustered Tensor Approximation

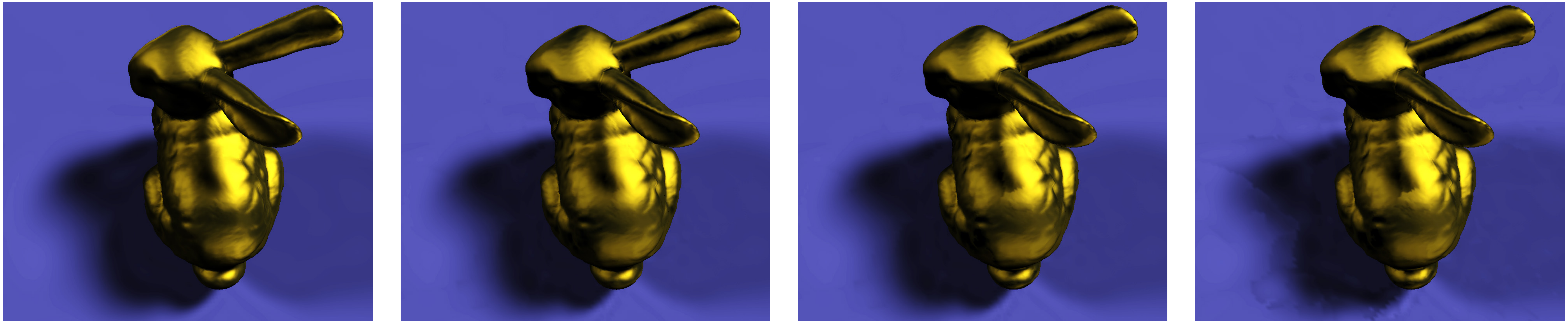
- The clusters should be grouped in such a way, that the compression error is minimized

- Iterative algorithm (similar to k-means)
  - Initialize with clustering on unrolled slices
  - Repeat until convergence
    - Perform Tucker factorization for each cluster
    - Reassign slices into cluster in which they can be represented with the smallest error
      - Using core tensor and factor matrices from the previous step



Clustered Tensor Approximation

- Applications to PRT in [Tsai-2006] and [Sun-2007]



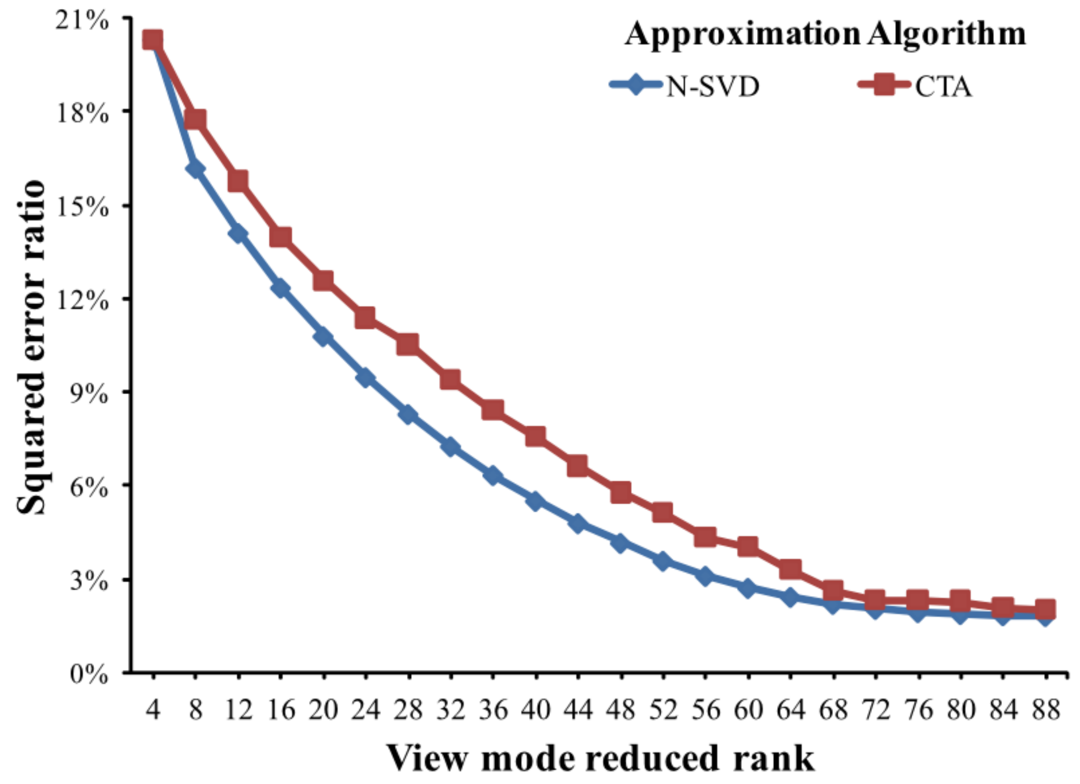
uncompressed | 1:75 | 1:127 | 1:165

[Tsai-2006]

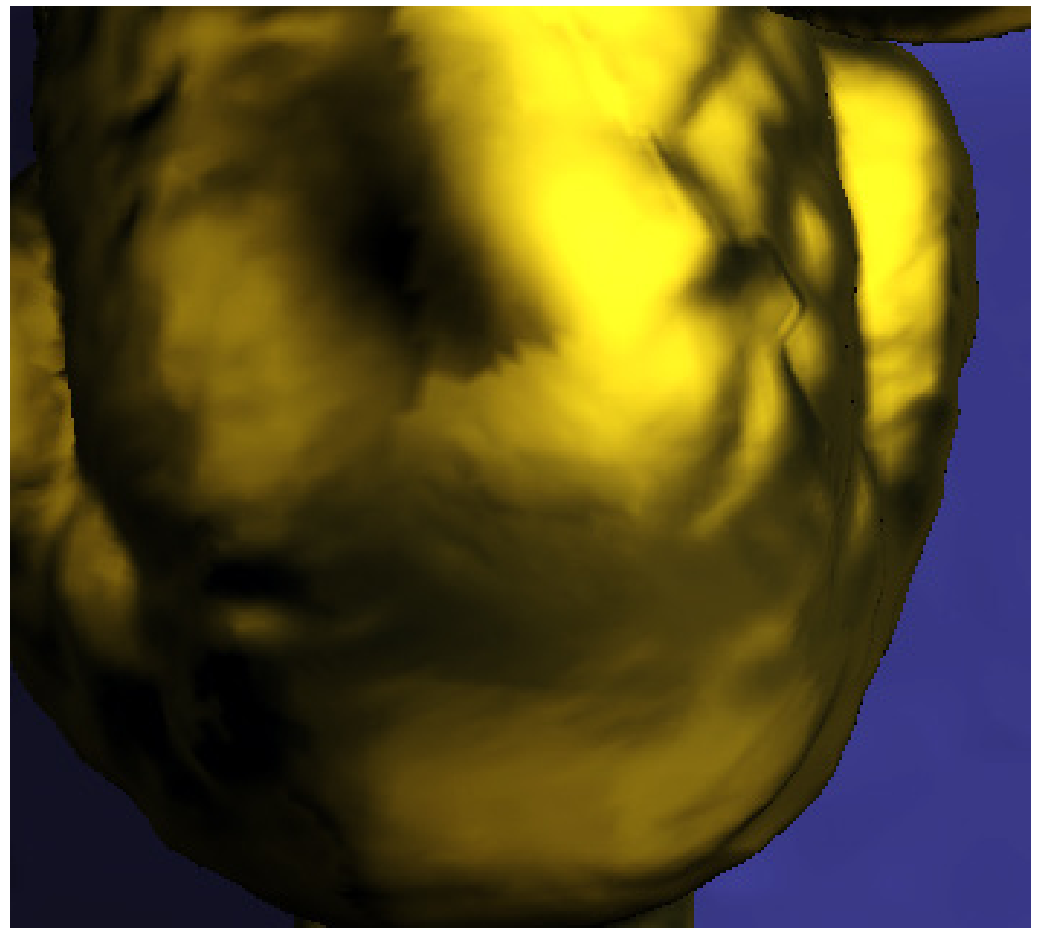
- Good approximation quality at a compression ratio of 1:75
- Interactive rendering possible
- Cluster boundaries visible at higher compression ratios
- No direct comparison to other compression techniques provided

Clustered Tensor Approximation

- Decreased tensor size improves rendering performance
  - 30%-80% higher framerate for BTF rendering compared to Tucker factorization [Tsai-2009]
- Coherence between clusters is not utilized at all
  - Compression ratio on BTF datasets inferior to Tucker factorization
- Clustering can result in visible cluster boundaries
- Linear interpolation in the clustered mode is expensive
  - GPU texture interpolation cannot be used
    - Clustering along view direction for BTFs

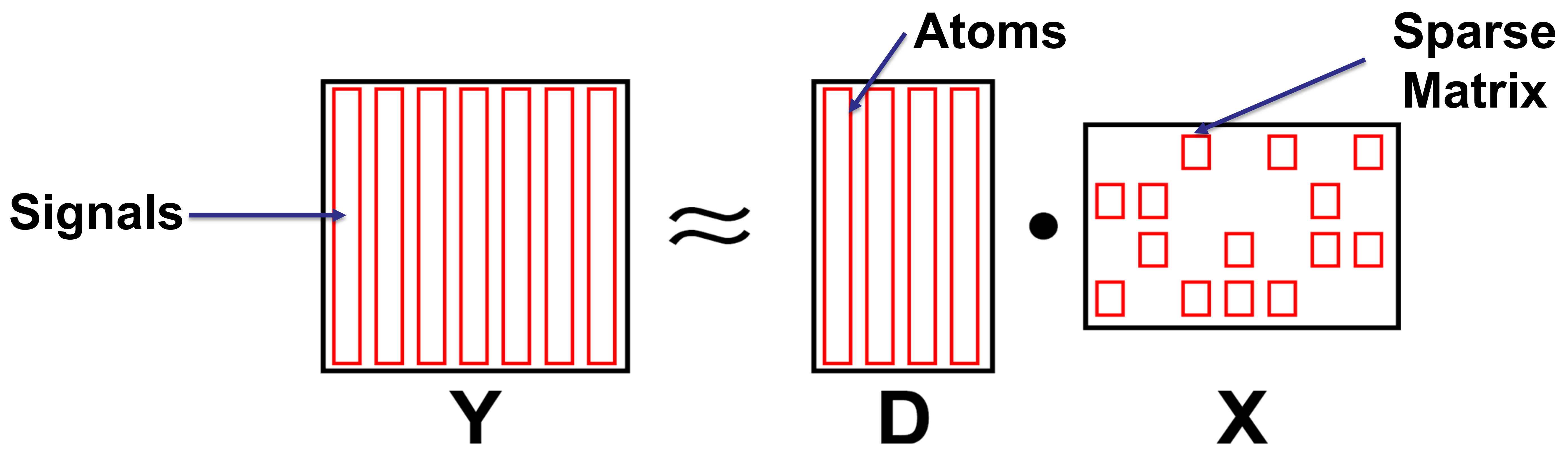


BTF compression errors from [Tsai-2009]



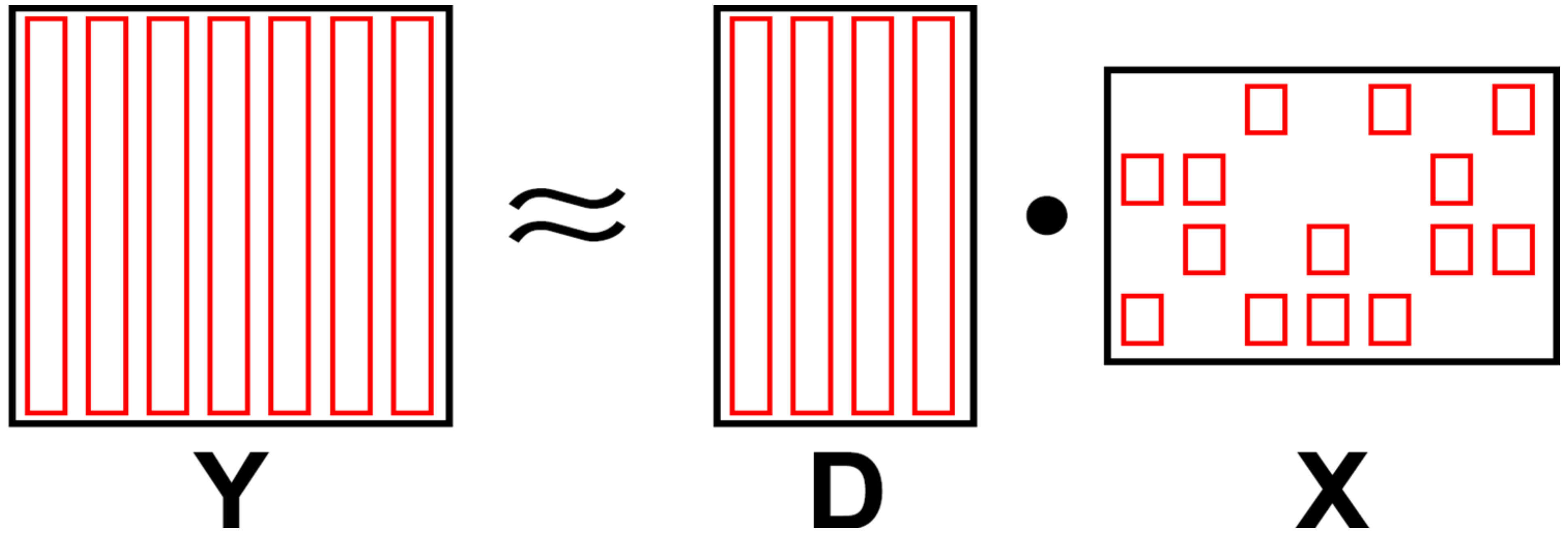
[Tsai-2006]

Sparse Signal Coding



- Represent matrix **Y** as a product of a dictionary matrix **D** and a sparse matrix **X**
  - Each column of **X** contains at most *k* entries
- Each column of **Y** (signal) is thus approximated as linear combination of at most *k* columns from **D** (atom)

Sparse Signal Coding

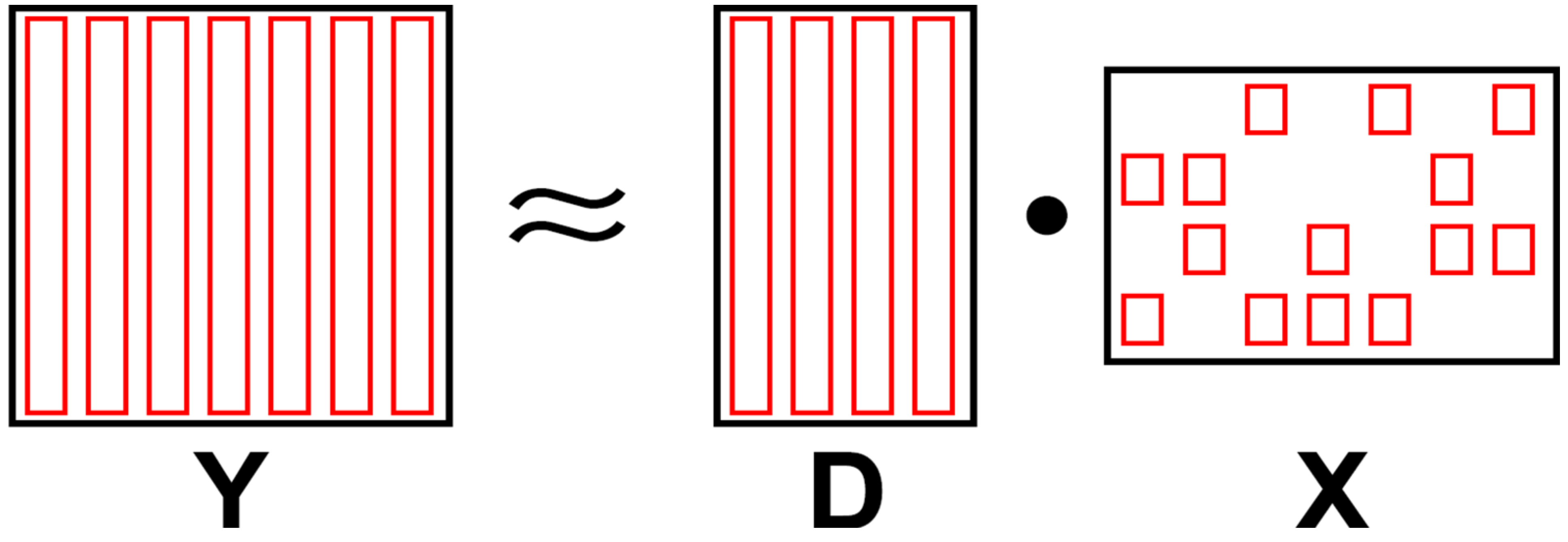


- Given Y this minimization problem has to be solved:

$$\min_{\mathbf{X},\mathbf{D}} \|\mathbf{Y} - \mathbf{DX}\|^2 \quad \text{subject to} \quad \forall i : \|\mathbf{X}_i\|_0 \leq k$$

- K-SVD [Aharon-2006] optimizes this problem
  - Searches for both the dictionary D and the sparse representation X

Sparse Signal Coding

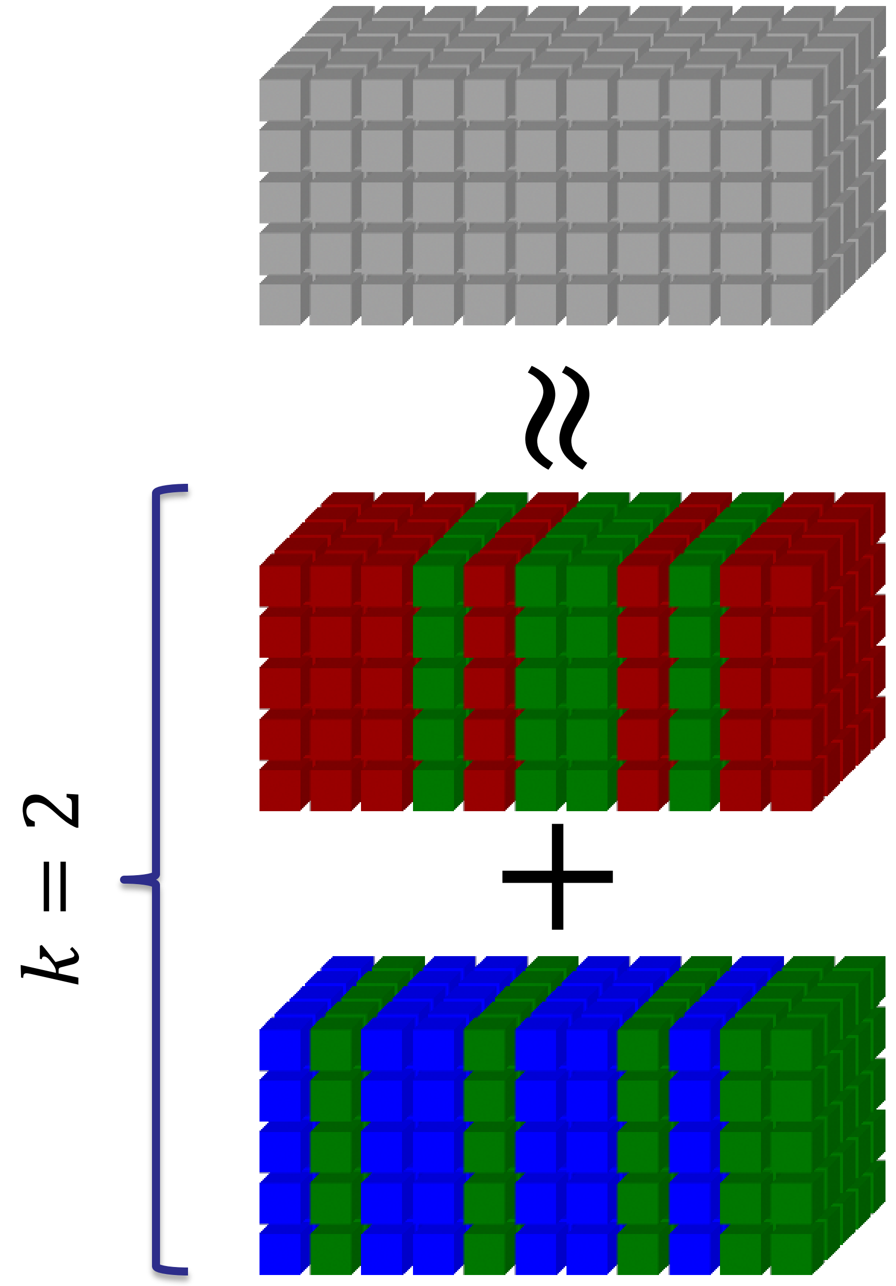


- The sparsity of **X** has two important advantages compared to full matrices
  - It can be represented more compactly
  - The matrix product can be evaluated faster

- Two different applications of K-SVD to tensors have been proposed
  - **K-Clustered Tensor Approximation** [Tsai-2009] and [Tsai-2011]
  - **Sparse Tensor Decomposition** [Ruiters-2009]

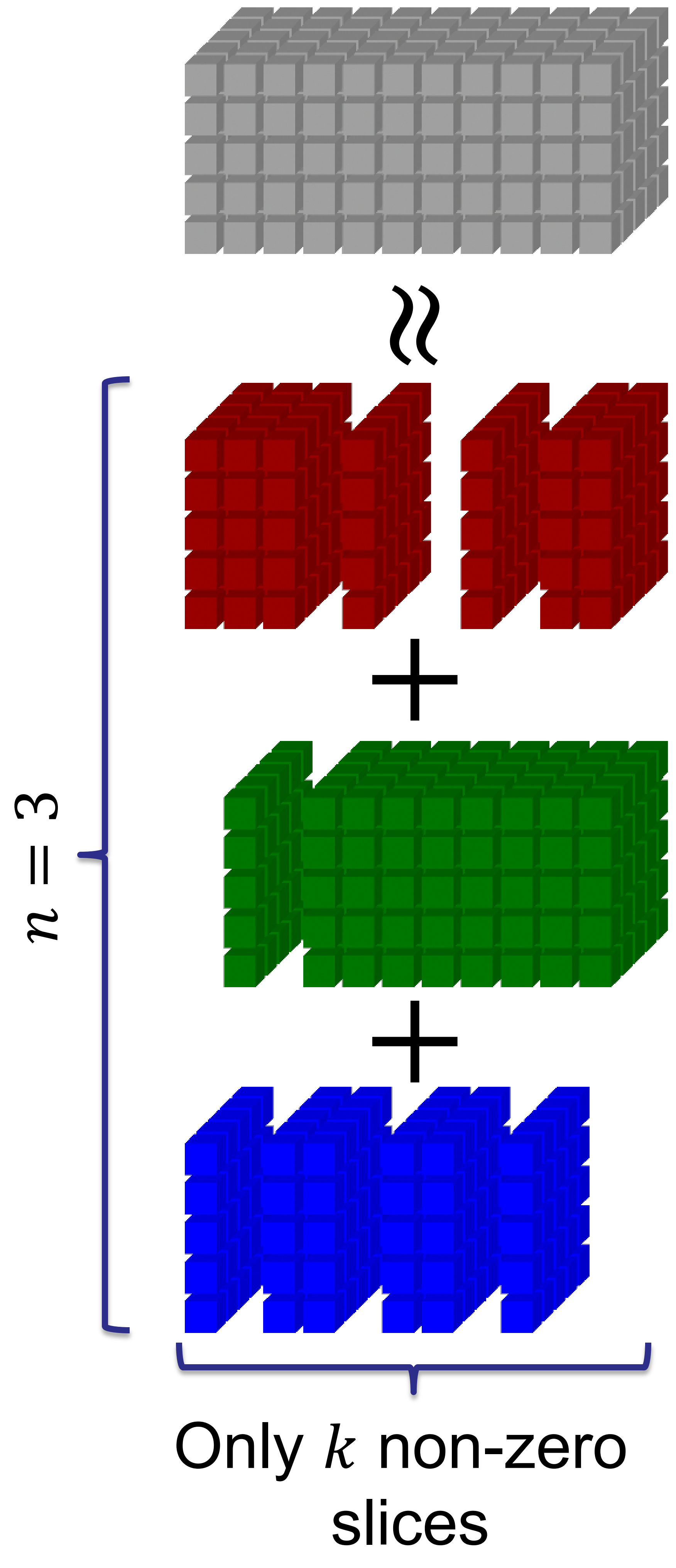
K-Clustered Tensor Approximation

- Utilize inter-cluster correlations by assigning each slice in mode m to k from n clusters
  - Each slice is approximated as the sum of the contribution of k slices each belonging to one of n clusters



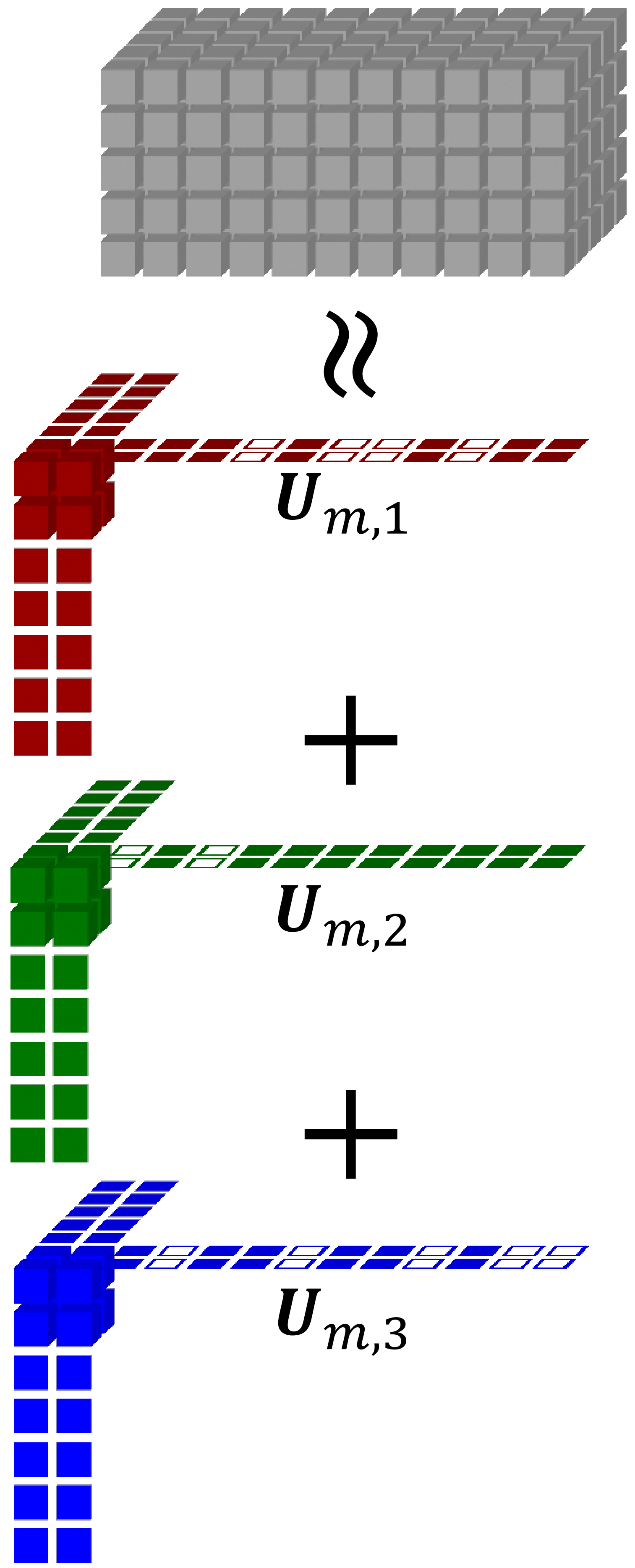
K-Clustered Tensor Approximation

- Utilize inter-cluster correlations by assigning each slice in mode m to k from n clusters
  - Each slice is approximated as the sum of the contribution of k slices each belonging to one of n clusters
  - This can also be considered as a sum of n Tensors in which along mode m only k slices are non-zero.



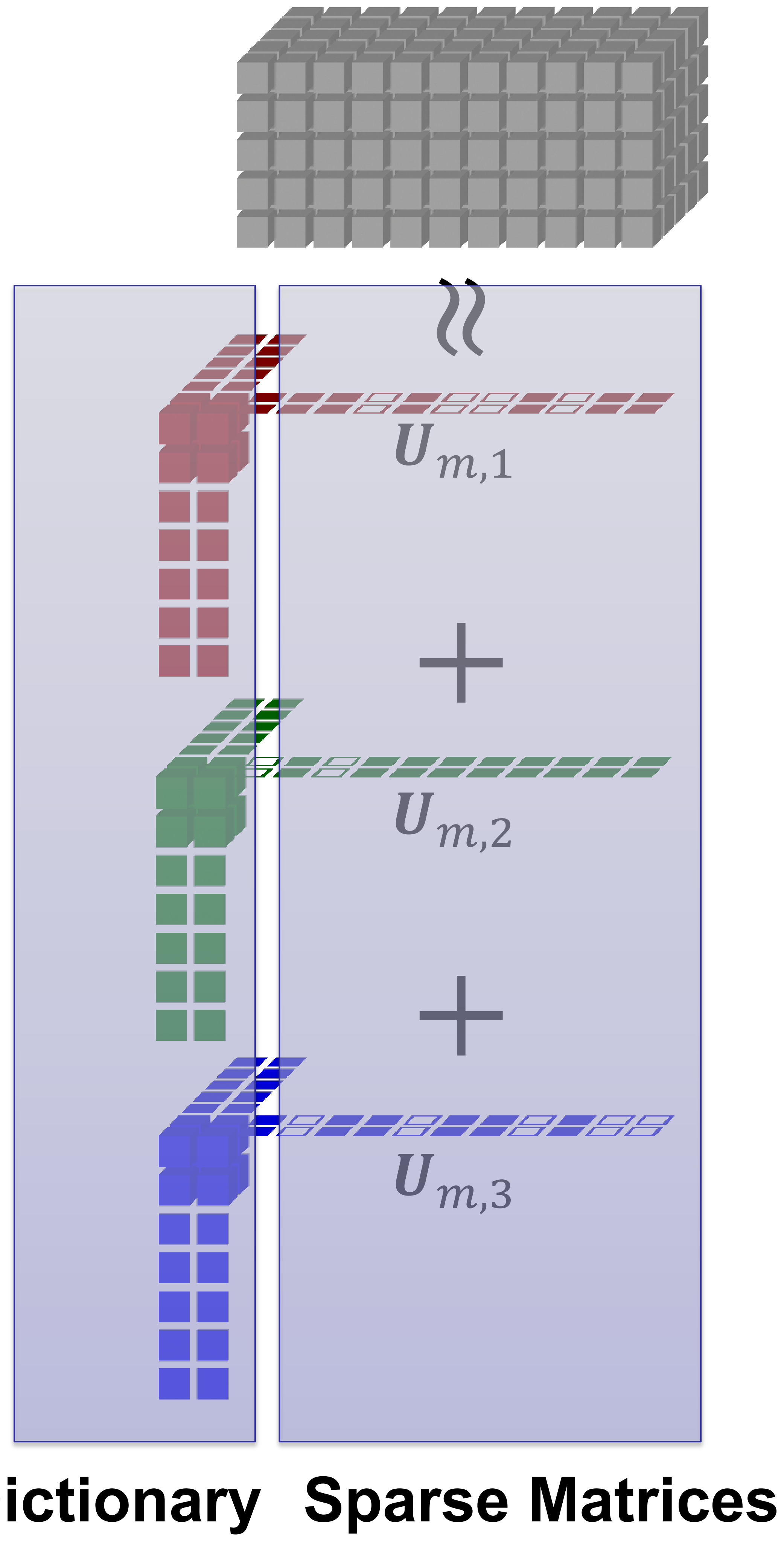
K-Clustered Tensor Approximation

- Utilize inter-cluster correlations by assigning each slice in mode m to k from n clusters
  - Each slice is approximated as the sum of the contribution of k slices each belonging to one of n clusters
  - This can also be considered as a sum of n tensors in which along mode m only k slices are non-zero
  - This results in sparse $\boldsymbol{U}_{m,c}$ matrices in the Tucker factorizations of the per-cluster tensors



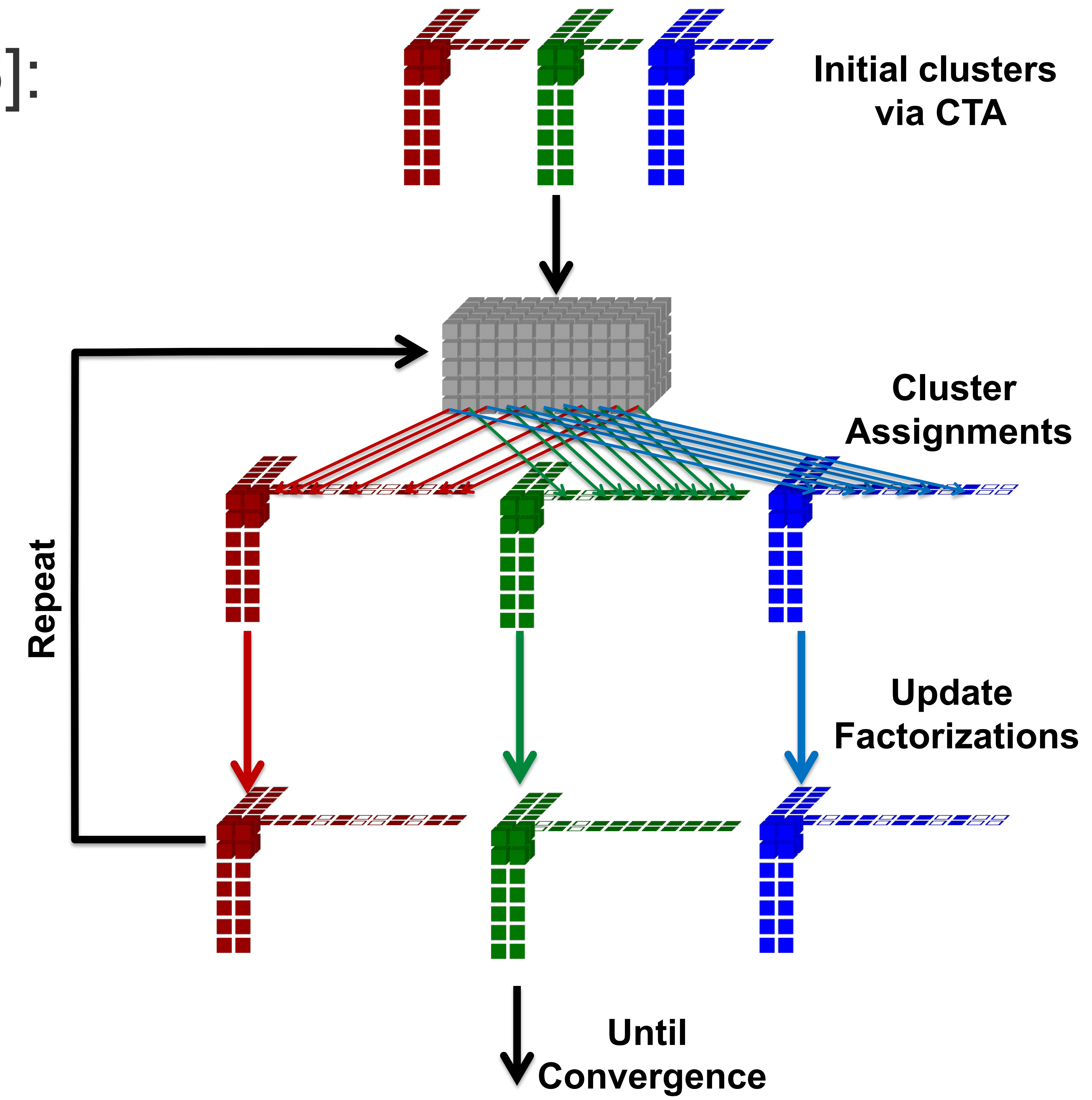
K-Clustered Tensor Approximation

- Utilize inter-cluster correlations by assigning each slice in mode m to k from n clusters
  - Each slice is approximated as the sum of the contribution of k slices each belonging to one of n clusters
  - This can also be considered as a sum of n tensors in which along mode m only k slices are non-zero
  - This results in sparse $\boldsymbol{U}_{m,c}$ matrices in the Tucker factorizations of the per-cluster tensors
- Relation to the K-SVD
  - The core tensor and the other mode matrices correspond to the Dictionary **D**
  - The sparse matrices $\boldsymbol{U}_{m,c}$ correspond to the sparse matrix **X**



K-Clustered Tensor Approximation

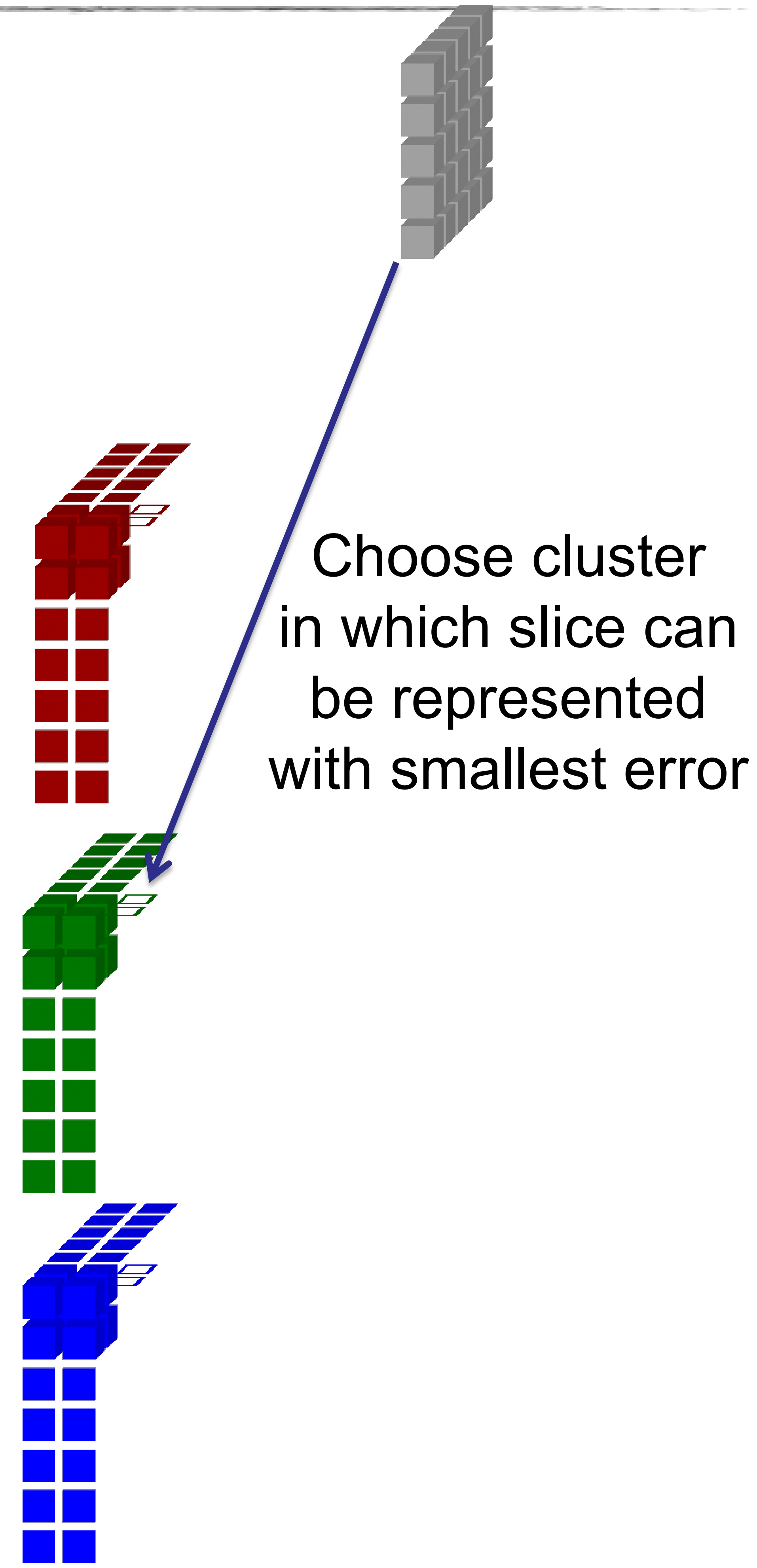
- The algorithm is similar to K-SVD [Aharon-2006]:
  - Initialize the clusters with a single step CTA
  - Repeat until convergence
    - Update the assignment of each slice to k clusters
    - Update the factorizations of each cluster



- Computations can be performed efficiently on the factorization with reduced rank
  - Not shown in the following

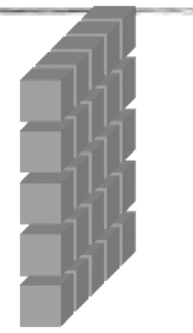
K-Clustered Tensor Approximation

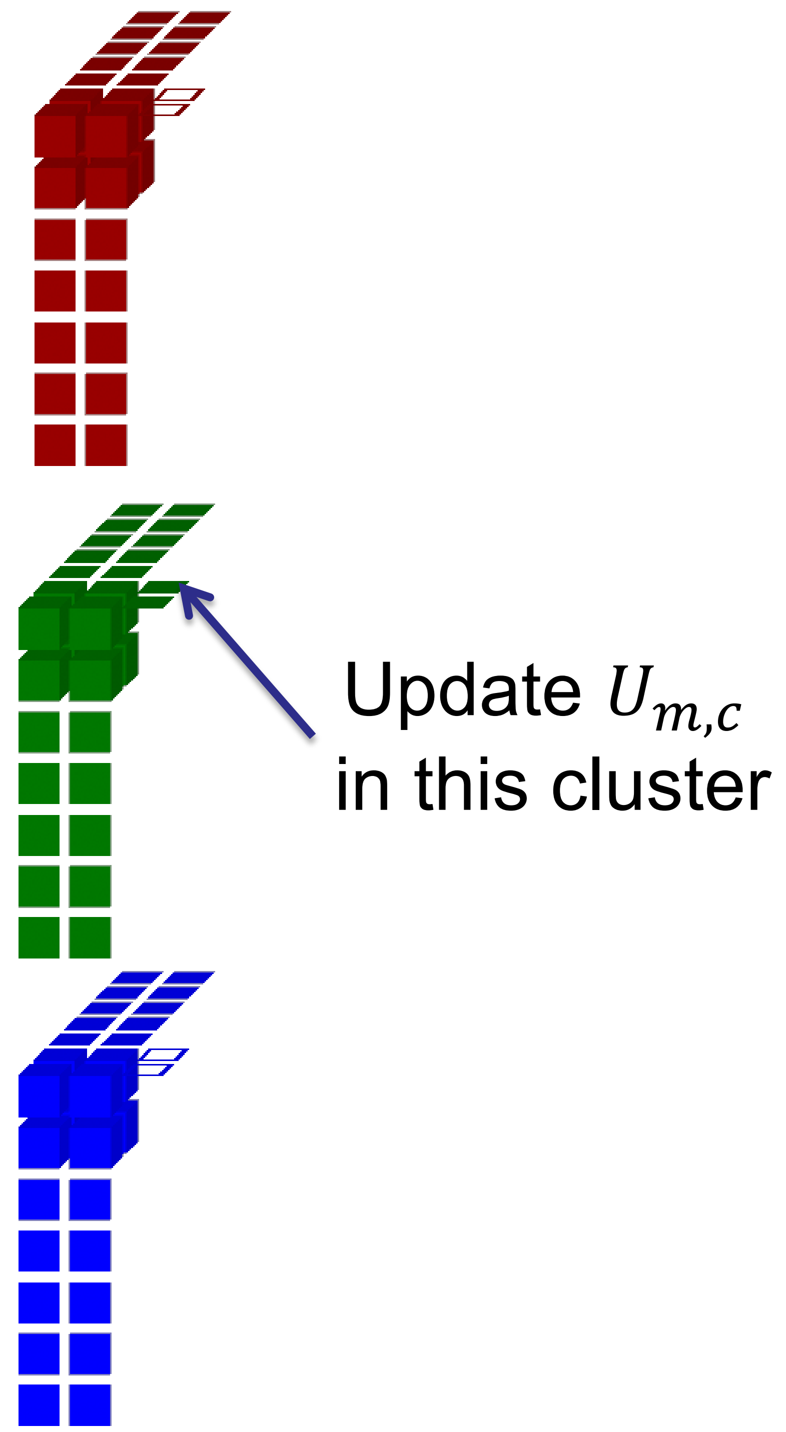
- The algorithm is similar to K-SVD [Aharon-2006]:
  - Initialize the clusters with a single step CTA
  - Repeat until convergence
    - Update the assignment of each slice to k clusters
      - Greedily via orthogonal matching pursuit
        - Compute the residual by subtracting the representation in the already chosen clusters
        - Assign the cluster in which this residual can be represented with the smallest error to the slice
        - Update the representation $U_{m,c}$ in the already assigned clusters



K-Clustered Tensor Approximation

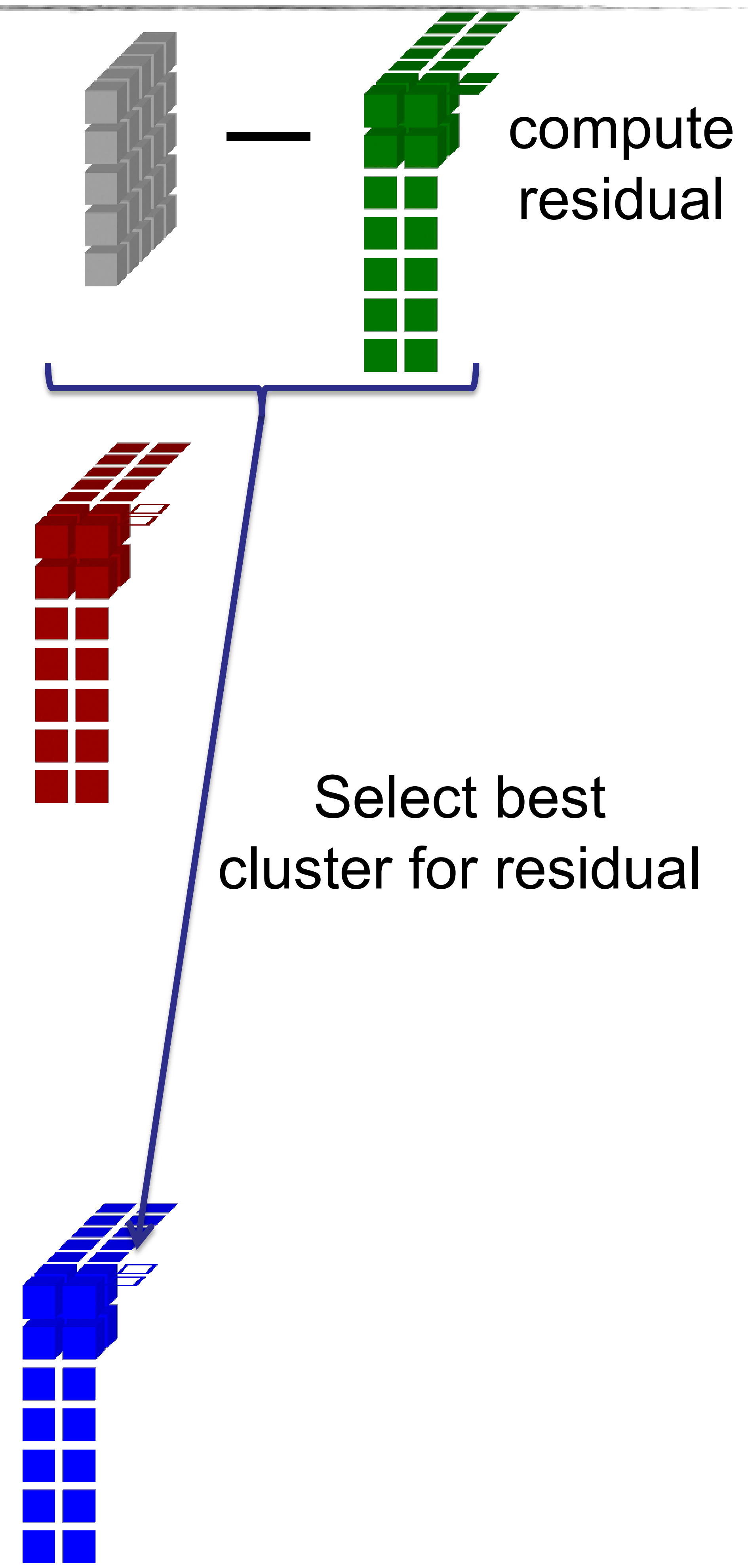
- The algorithm is similar to K-SVD [Aharon-2006]:
  - Initialize the clusters with a single step CTA
  - Repeat until convergence
    - Update the assignment of each slice to k clusters
      - Greedily via orthogonal matching pursuit
        - Compute the residual by subtracting the representation in the already chosen clusters
        - Assign the cluster in which this residual can be represented with the smallest error to the slice
        - Update the representation $U_{m,c}$ in the already assigned clusters





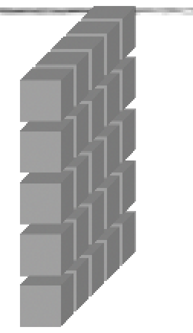
K-Clustered Tensor Approximation

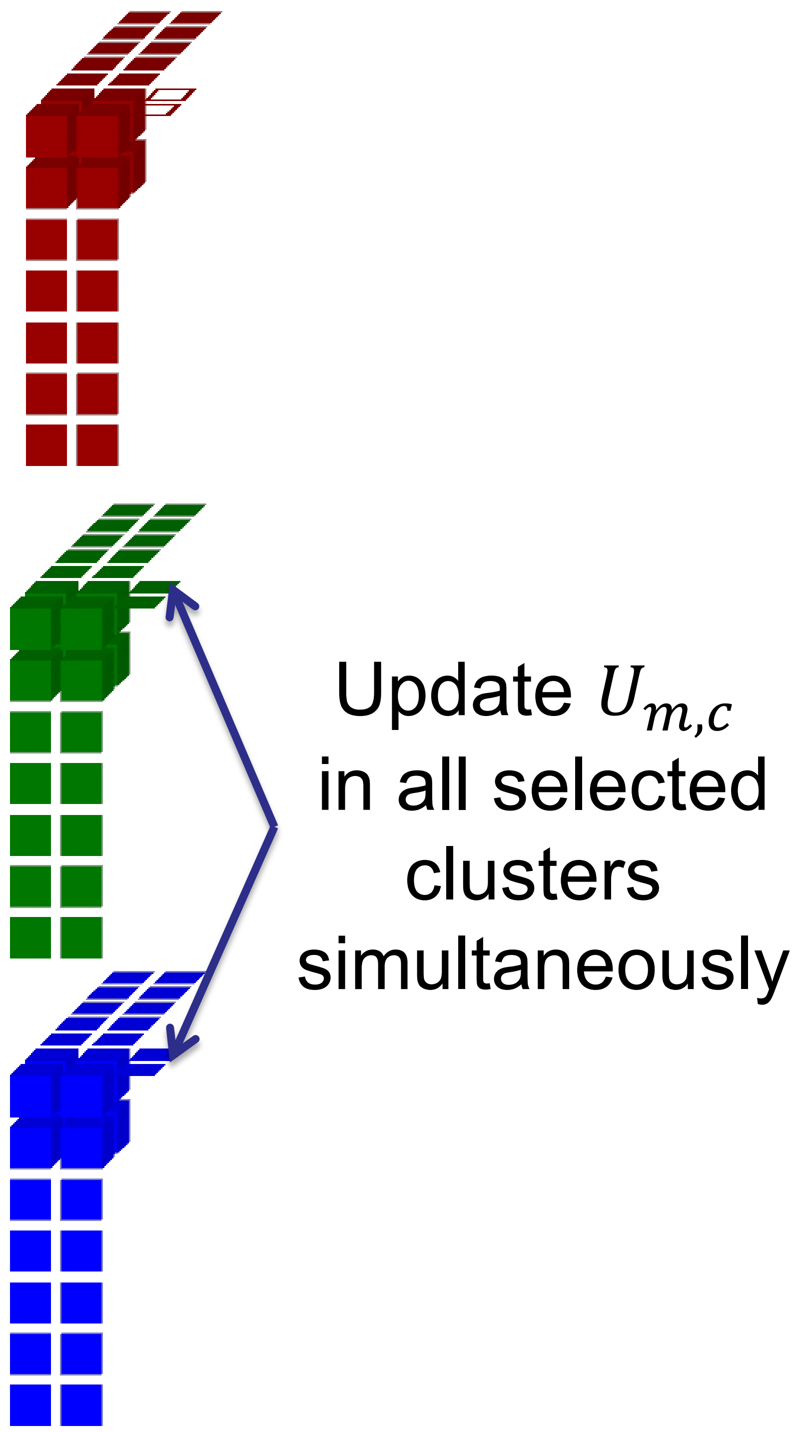
- The algorithm is similar to K-SVD [Aharon-2006]:
  - Initialize the clusters with a single step CTA
  - Repeat until convergence
    - Update the assignment of each slice to k clusters
      - Greedily via orthogonal matching pursuit
        - Compute the residual by subtracting the representation in the already chosen clusters
        - Assign the cluster in which this residual can be represented with the smallest error to the slice
        - Update the representation $U_{m,c}$ in the already assigned clusters



K-Clustered Tensor Approximation

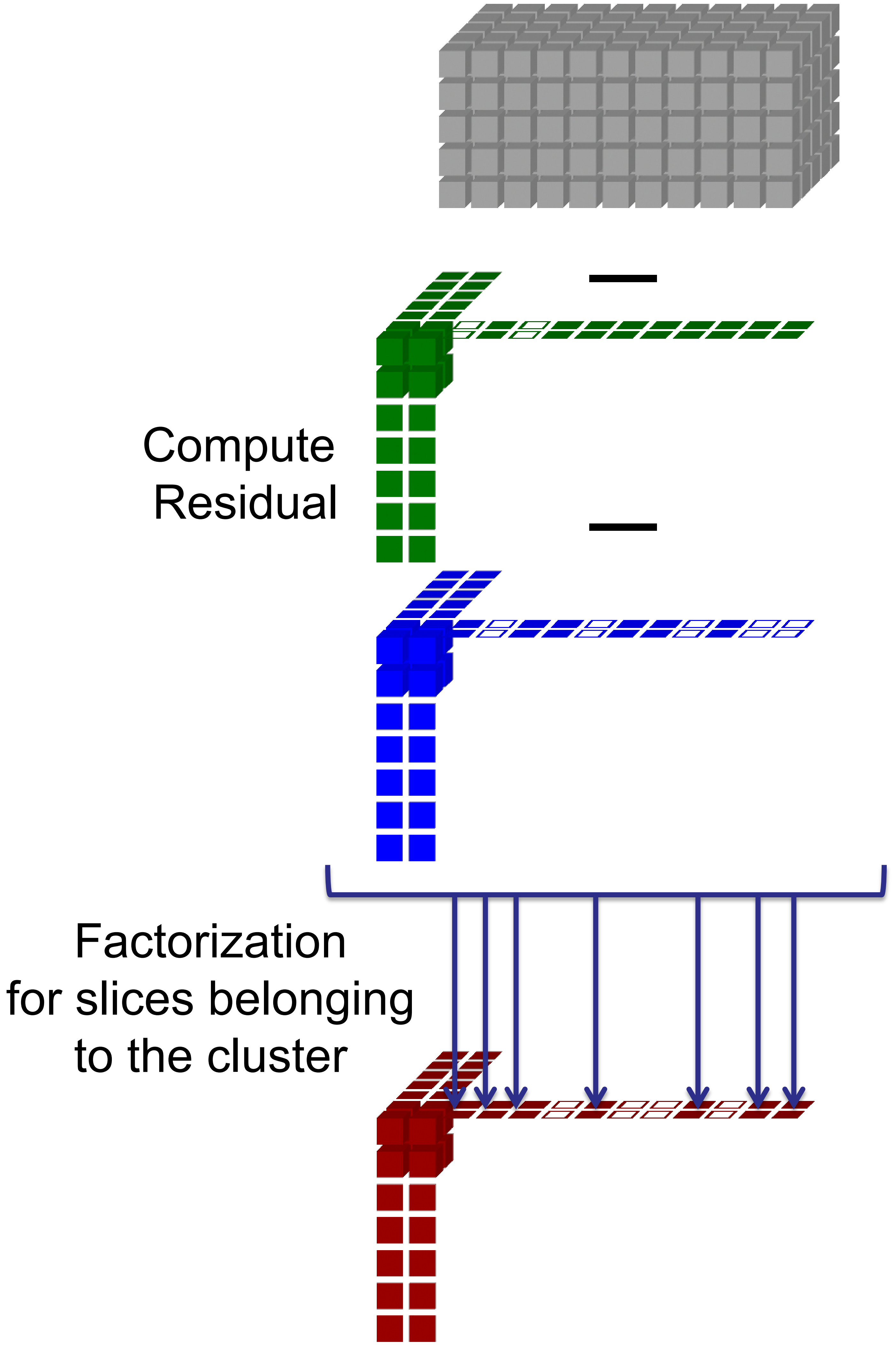
- The algorithm is similar to K-SVD [Aharon-2006]:
  - Initialize the clusters with a single step CTA
  - Repeat until convergence
    - Update the assignment of each slice to k clusters
      - Greedily via orthogonal matching pursuit
        - Compute the residual by subtracting the representation in the already chosen clusters
        - Assign the cluster in which this residual can be represented with the smallest error to the slice
        - Update the representation $U_{m,c}$ in the already assigned clusters





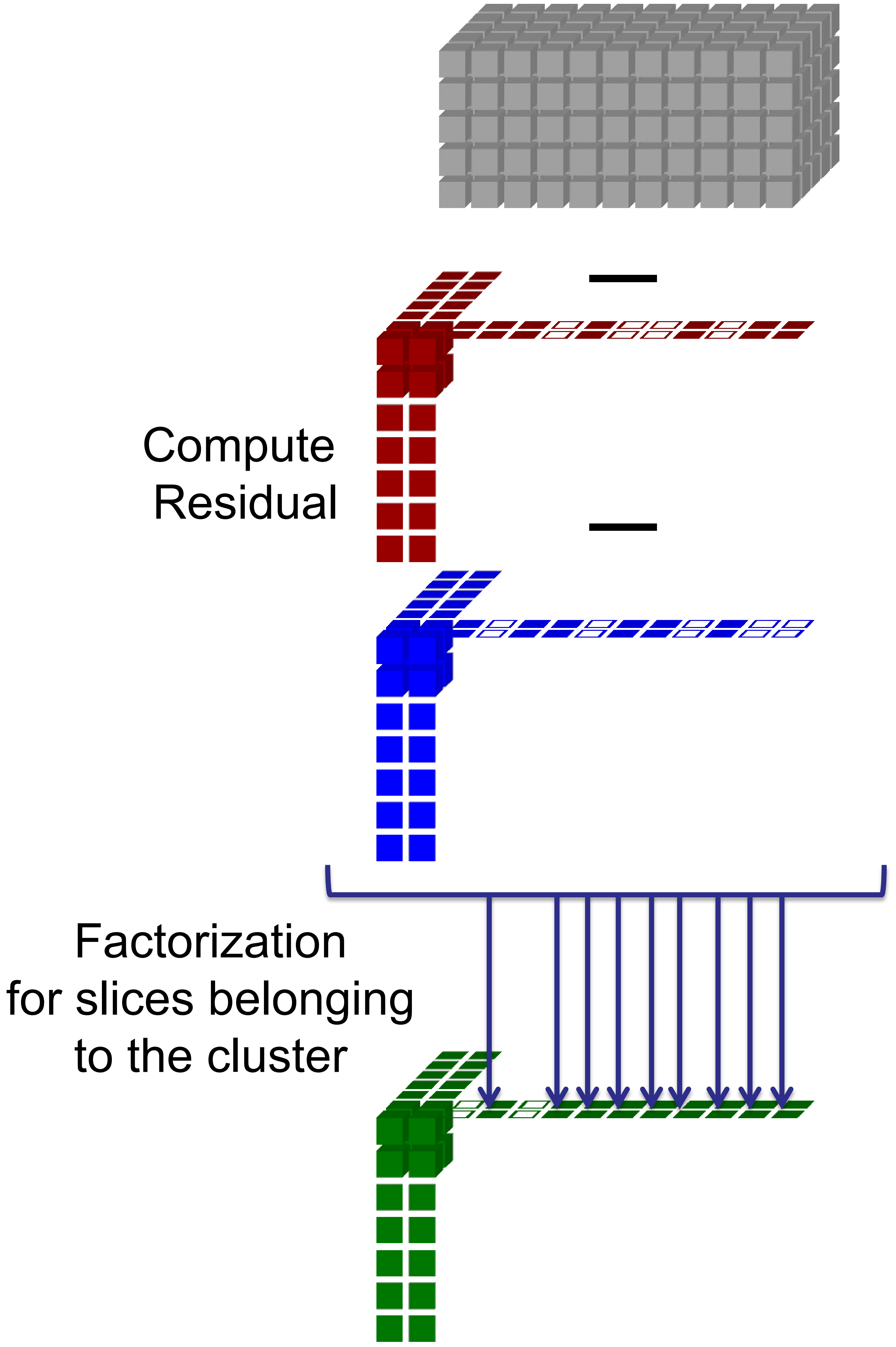
K-Clustered Tensor Approximation

- The algorithm is similar to K-SVD [Aharon-2006]:
  - Initialize the clusters with a single step CTA
  - Repeat until convergence
    - Update the assignment of each slice to k clusters
      - Greedily via orthogonal matching pursuit
        - Compute the residual by subtracting the representation in the already chosen clusters
        - Assign the cluster in which this residual can be represented with the smallest error to the slice
        - Update the representation $U_{m,c}$ in the already assigned clusters
    - Update the tensor factorizations
      - Iteratively for each cluster
        - Compute the residual by subtracting the reconstruction of all other clusters from the data tensor
        - Factorize the residual for the selected slices



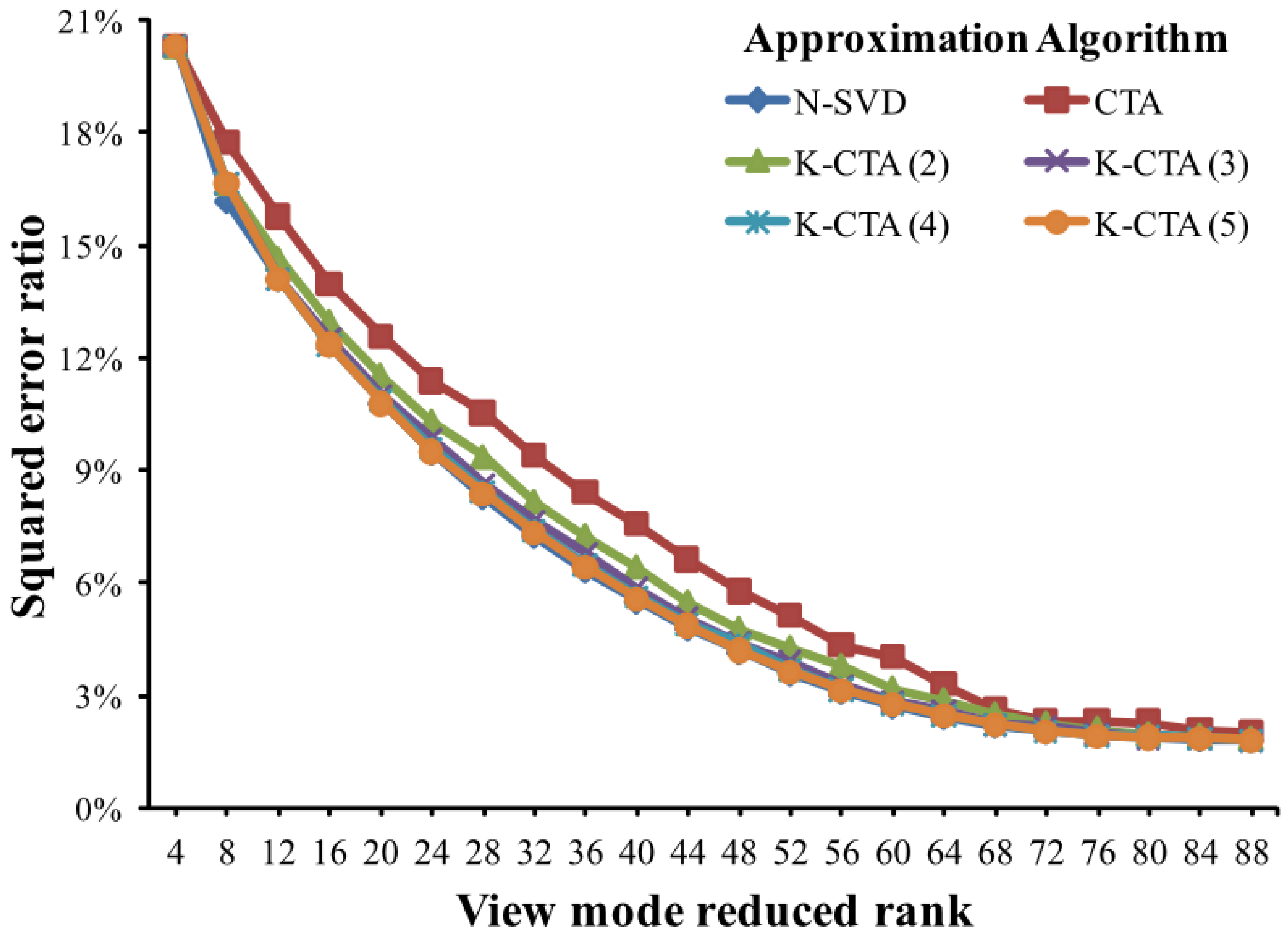
K-Clustered Tensor Approximation

- The algorithm is similar to K-SVD [Aharon-2006]:
  - Initialize the clusters with a single step CTA
  - Repeat until convergence
    - Update the assignment of each slice to k clusters
      - Greedily via orthogonal matching pursuit
        - Compute the residual by subtracting the representation in the already chosen clusters
        - Assign the cluster in which this residual can be represented with the smallest error to the slice
        - Update the representation $U_{m,c}$ in the already assigned clusters
    - Update the tensor factorizations
      - Iteratively for each cluster
        - Compute the residual by subtracting the reconstruction of all other clusters from the data tensor
        - Factorize the residual for the selected slices



K-Clustered Tensor Approximation

- BTF represented as a mode-4 tensor
  - Views × Light × X × Y
  - Clustering along the view mode
  - For GPU rendering the last two modes are premultiplied
- Compression ratio better than CTA
  - For BTF compression approximately equal to Tucker
- Faster decompression than Tucker
  - Since only a small subset of k of the clusters has to be decompressed for each slice
  - 30%-70% higher framerate for BTFs [Tsai-2009]
- Fewer problems with visible cluster boundaries
- Interpolation on GPU remains a problem



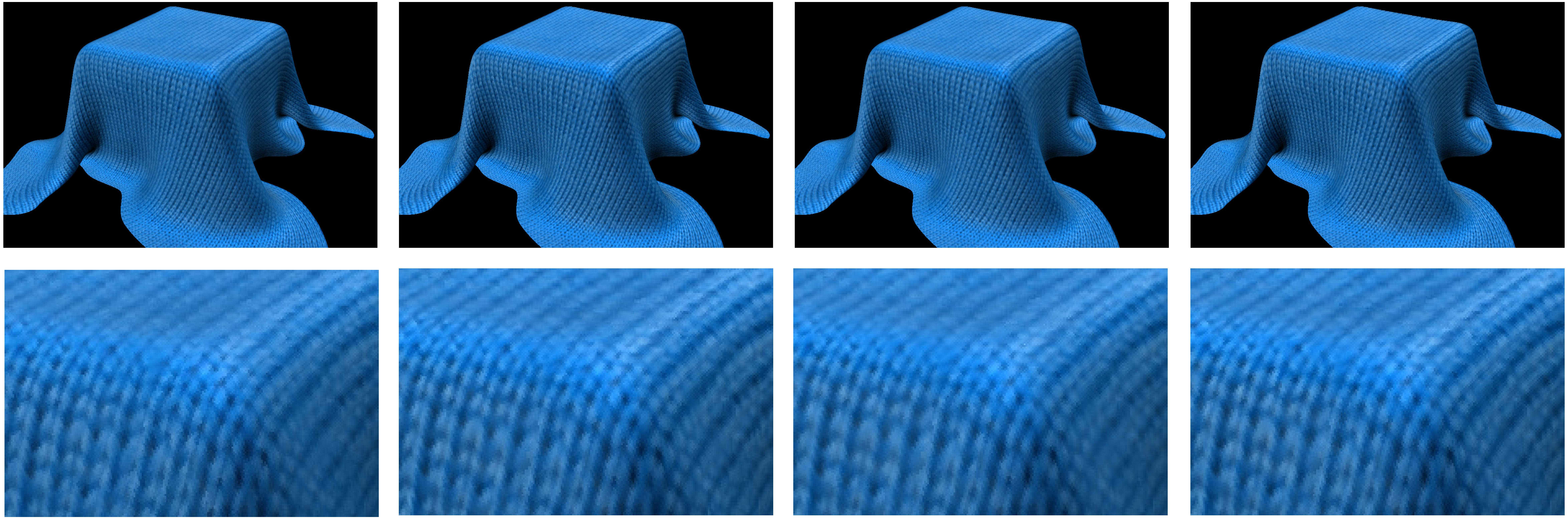
BTF compression errors from [Tsai-2009]

VISUALIZATIONAND MULTIMEDIALAB

universitätbonn

K-Clustered Tensor Approximation

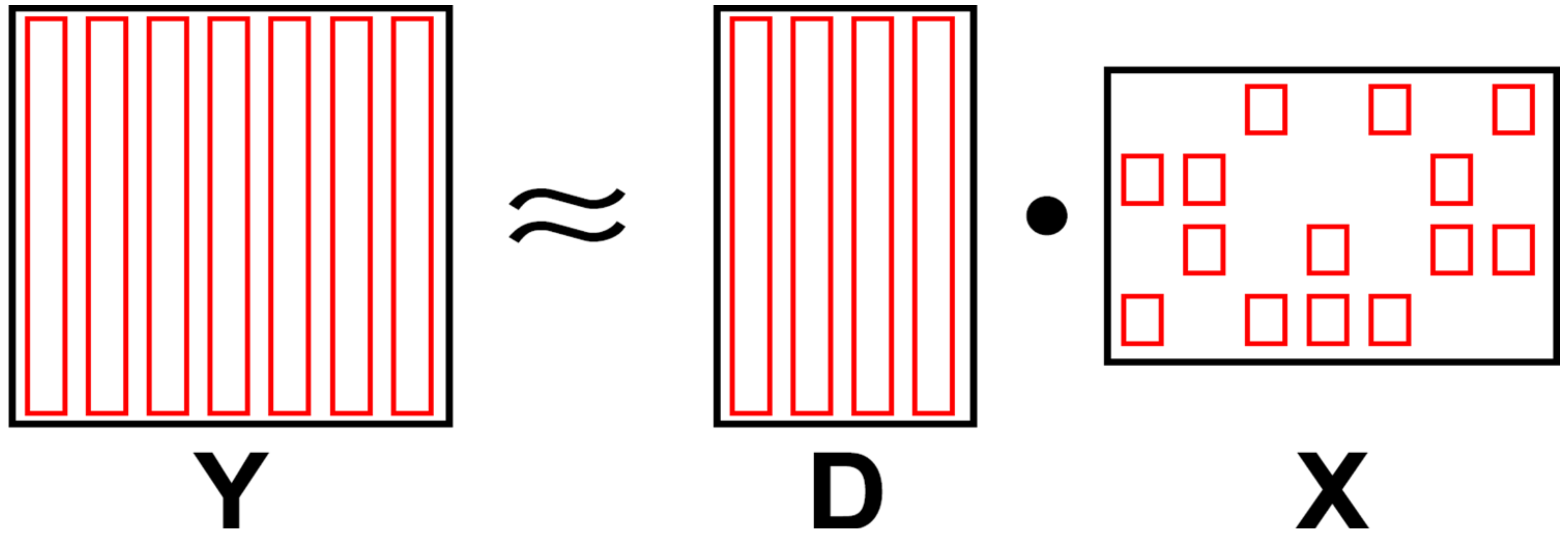
- Applications to BTF Compression in [Tsai-2009]



Uncompressed	**Tucker**	**CTA**	**K-CTA**
	Squared Error Ratio: 0.85%	Squared Error Ratio: 1.06%	Squared Error Ratio: 0.89%

Input: 1.2 GB, Compressed size: ca. 4.6 MB, Compression ratio: 1:267

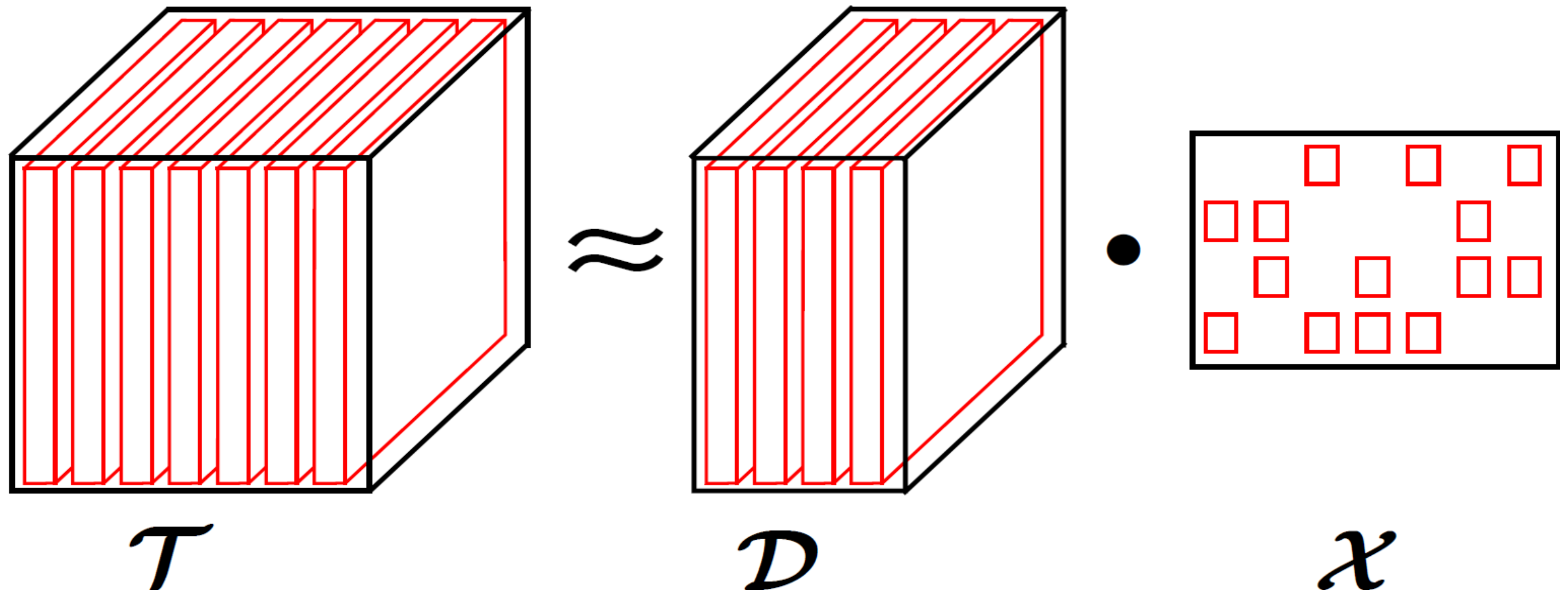
Sparse Signal Coding



- The sparsity of **X** has two important advantages compared to full matrices
  - It can be represented more compactly
  - The matrix product can be evaluated faster

- Two different applications of K-SVD to tensors have been proposed
  - **K-Clustered Tensor Approximation** [Tsai-2009] and [Tsai-2011]
  - **Sparse Tensor Decomposition** [Ruiters-2009]

Sparse Tensor Decomposition



- $\mathcal{T}$ regarded as a collection of Mode-M subtensors
  - Each subtensor is approximated as a combination of at most k dictionary entries

- $\mathcal{D}$ is a dictionary containing mode-M subtensor
- $\mathcal{X}$ is a sparse mode-(N-M+1) tensor

Einstein Summation Convention

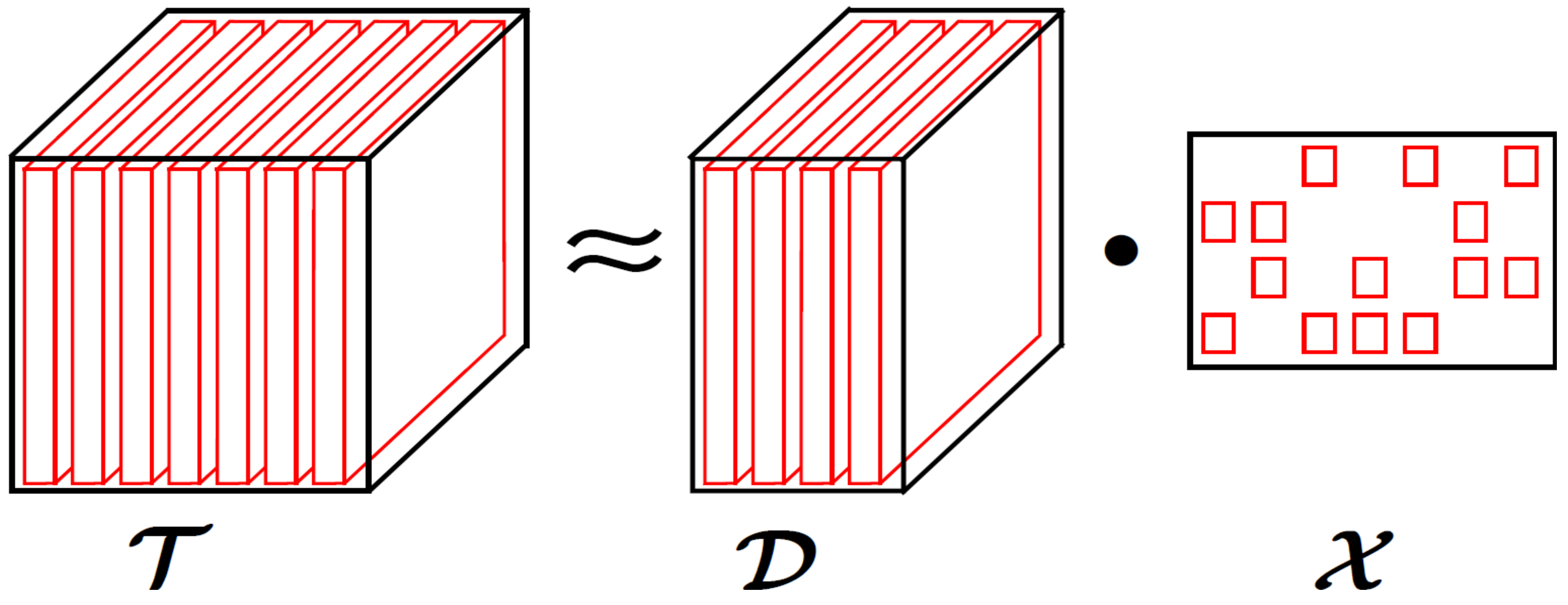
- Implicit summation over repeated indices

$$a_{ij}b_{jk} = \sum_j a_{ij} \cdot b_{jk}$$

- The elements of a tensor $\mathcal{C} = \mathcal{A}_{ij}\mathcal{B}_{jk}$ are thus given by:

$$\mathcal{C}_{i,k} = (\mathcal{A}_{ij}\mathcal{B}_{jk})_{i,k} = \sum_j \mathcal{A}_{i,j} \cdot \mathcal{B}_{j,k}$$

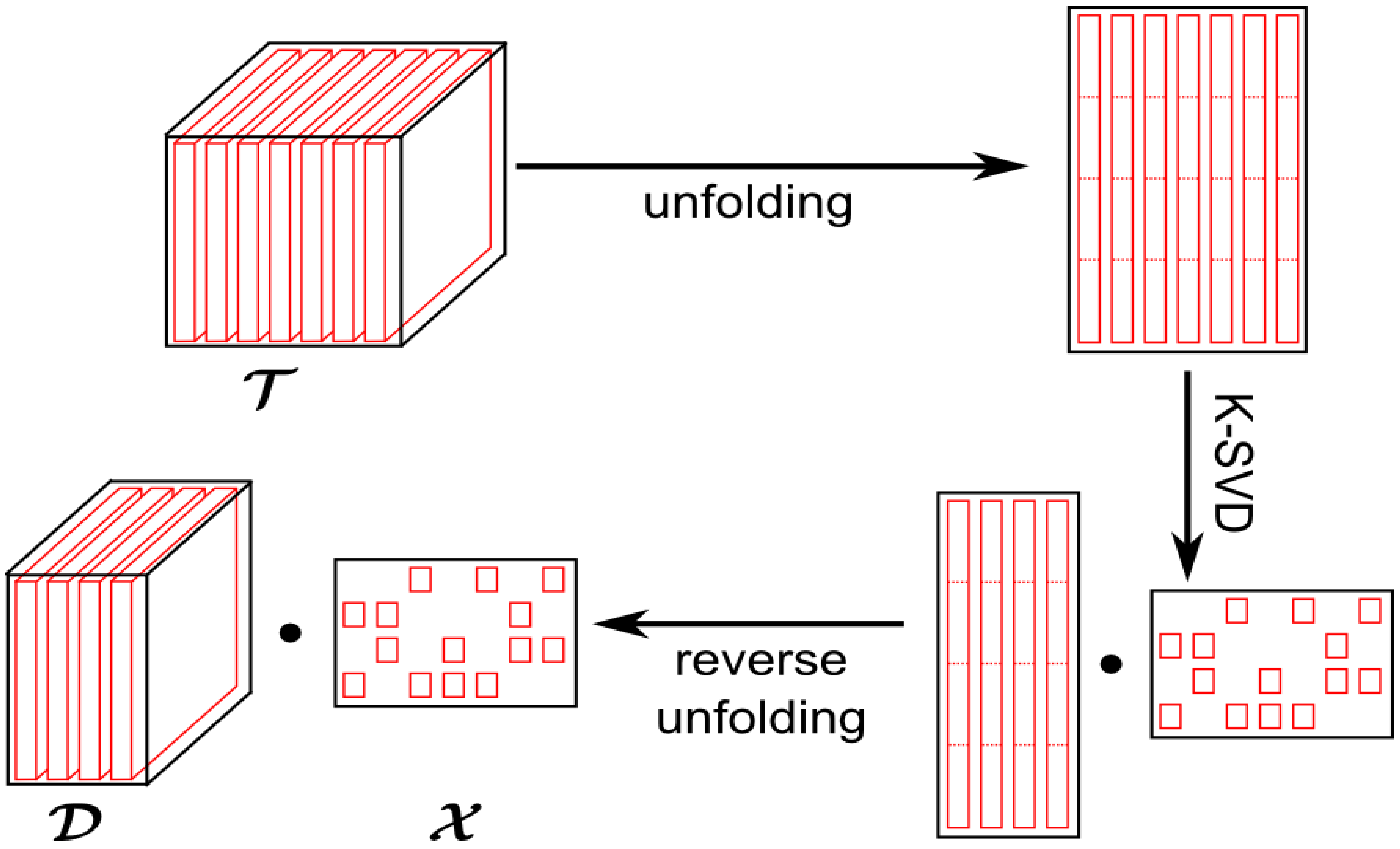
Sparse Tensor Decomposition



- A mode-N tensor $\mathcal{T}$ is decomposed into a mode-(M+1) dictionary $\mathcal{D}$ and a mode-(N-M+1) sparse Tensor $\mathcal{X}$

$$\mathcal{T} \approx \mathcal{D}_{i_1 \cdots i_M j} \mathcal{X}_{j i_{M+1} \cdots i_N}$$

Sparse Tensor Decomposition



- The decomposition is calculated by unfolding the tensor and using K-SVD on the unfolded tensor

Sparse Tensor Decomposition

- Only correlations in one mode have been utilized so far
  - Decomposition can be repeated along a different mode of $\boldsymbol{\mathcal{D}}$

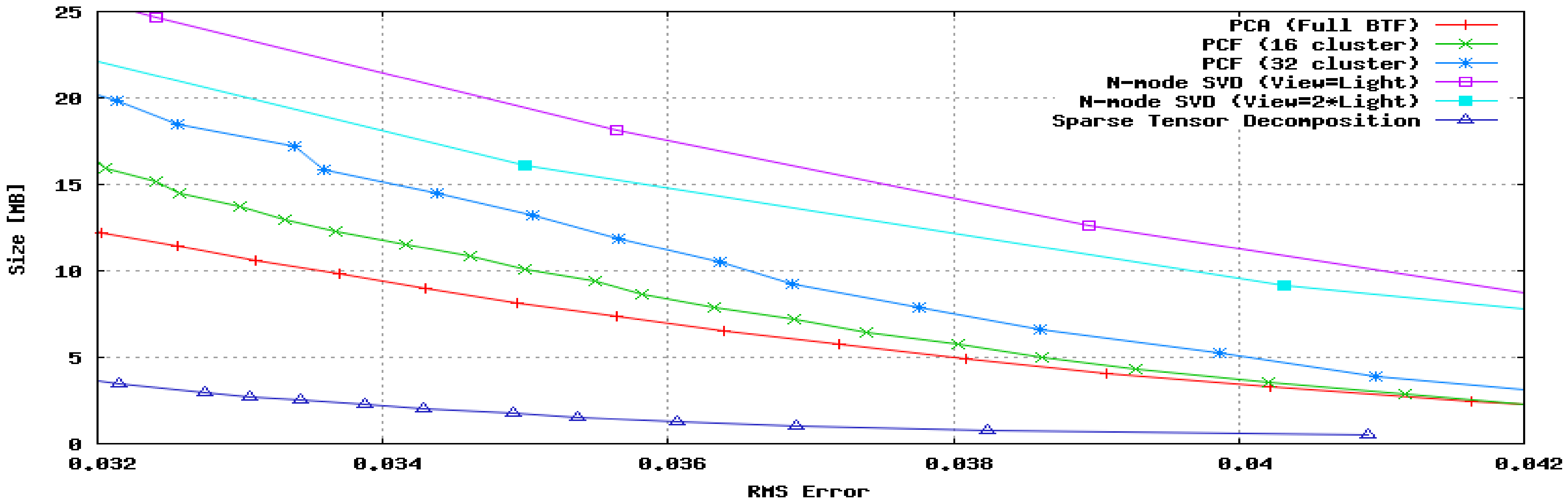
- When performed for all modes, we get a decomposition

$$\mathcal{T} \approx \mathcal{D}_{i_1 j_1} \mathcal{X}^{(1)}_{j_1 i_2 j_2} \mathcal{X}^{(2)}_{j_2 i_3 j_3} \cdots \mathcal{X}^{(N)}_{j_N i_N}$$

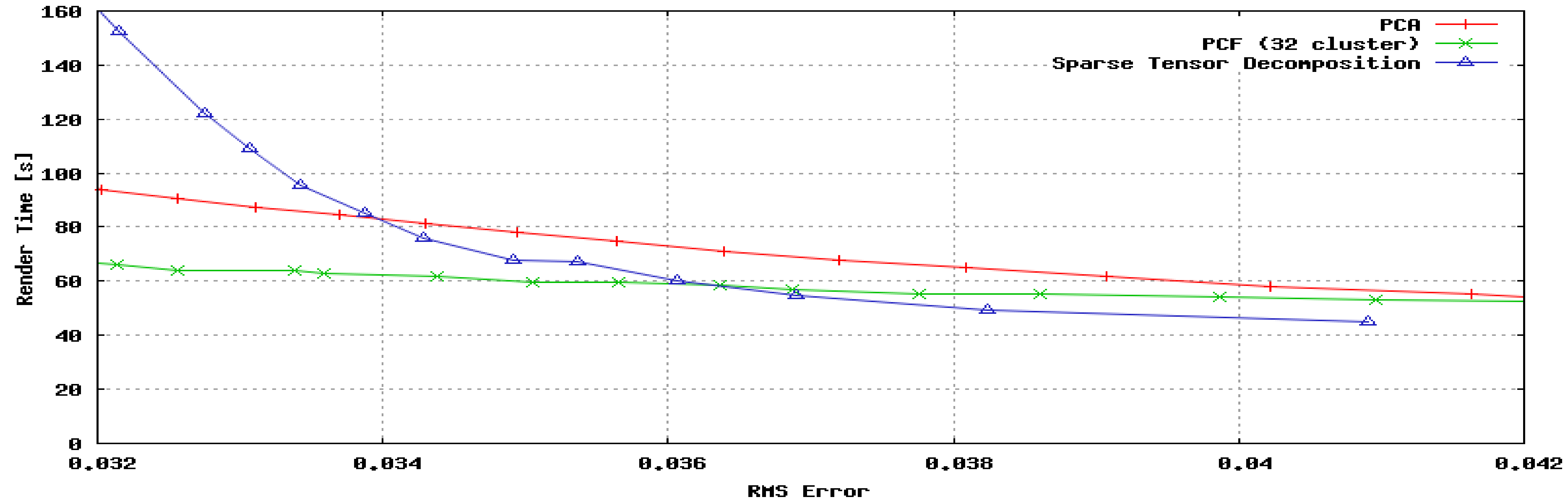
$\boldsymbol{\mathcal{D}}$	Mode-2 dictionary tensor
$\boldsymbol{\mathcal{X}}^{(1)} \ldots \boldsymbol{\mathcal{X}}^{(N-1)}$	Sparse mode-3 tensors
$\boldsymbol{\mathcal{X}}^{(N)}$	Sparse mode-2 tensor

Sparse Tensor Decomposition

- BTF represented as a mode-3 tensor
  - (Color*Light) × Views × Position
- Good compression ratio
  - By a factor of 3-4 better than PCA and by 4-5 times better than Tucker at the same RMS
- Sparsity enables faster rendering
  - Not well suited for GPU rendering
    - Interpolation a problem

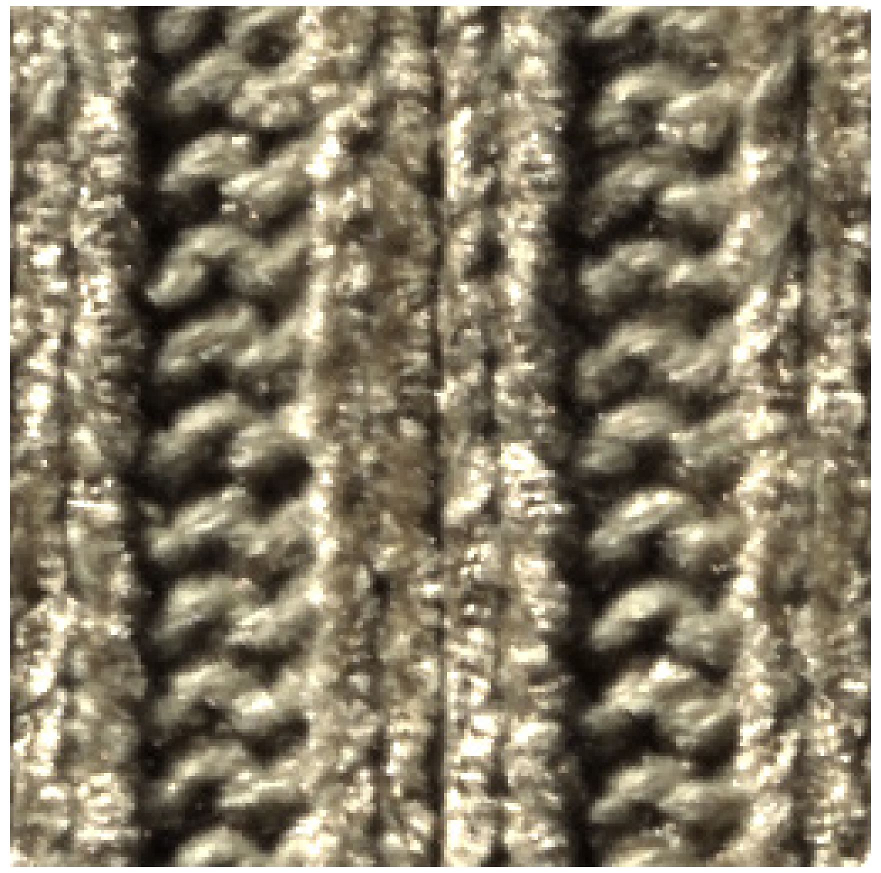


BTF compression errors from [Ruiters-2009]



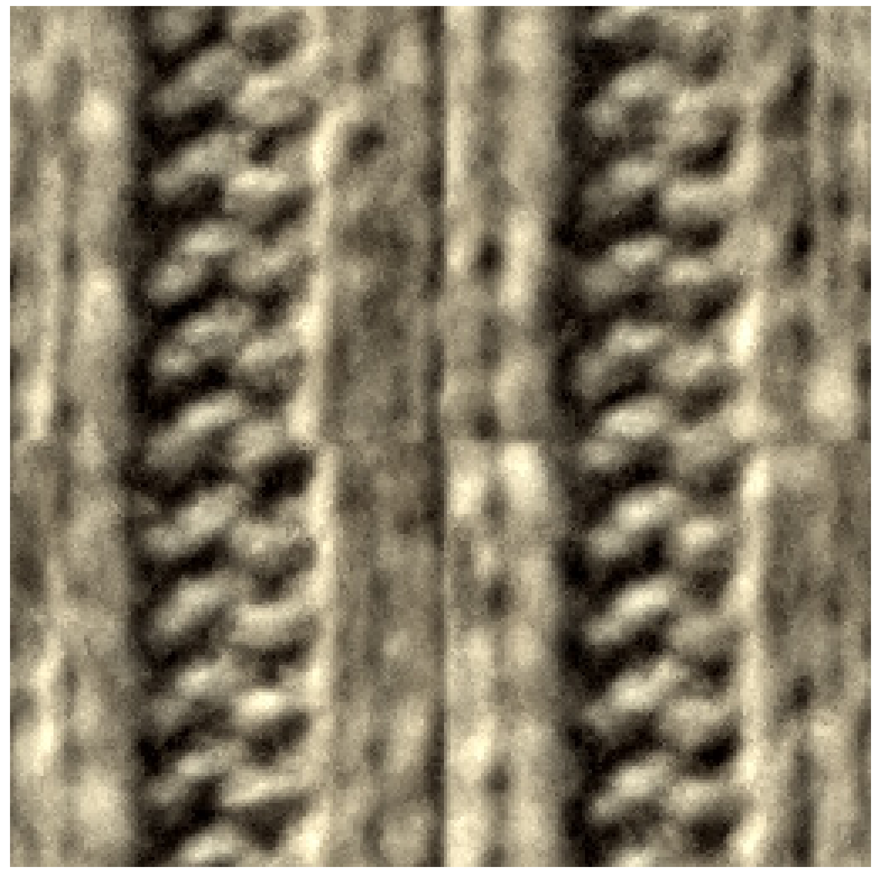
BTF render timings from [Ruiters-2009]

Sparse Tensor Decomposition



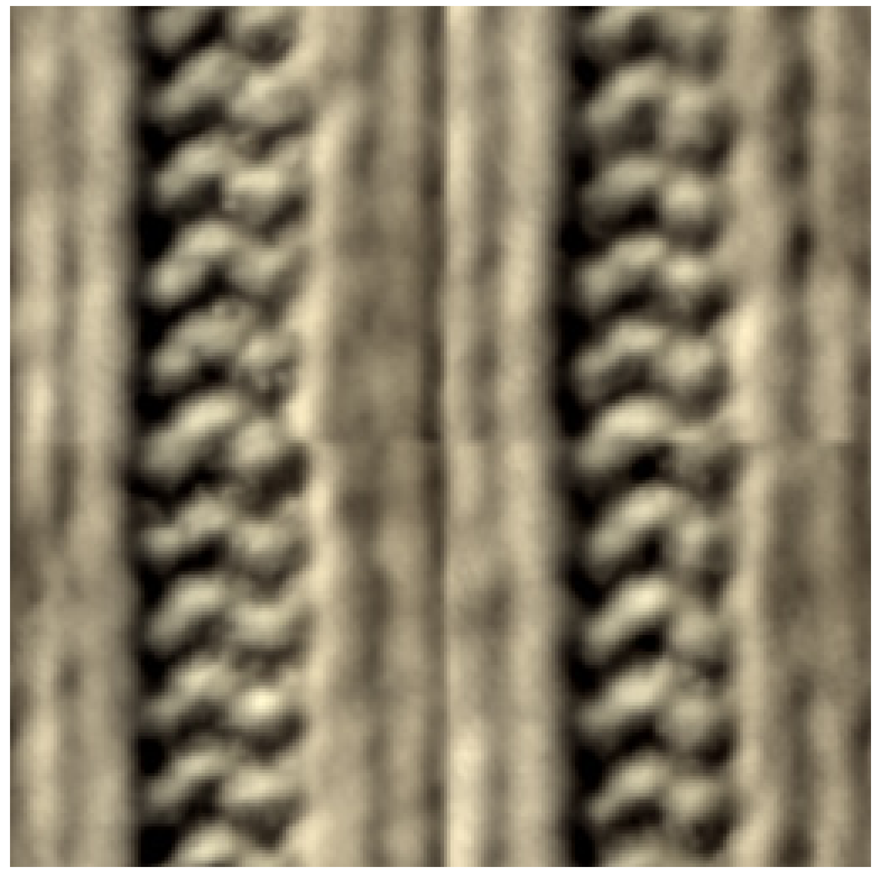
Original

2,4 GB



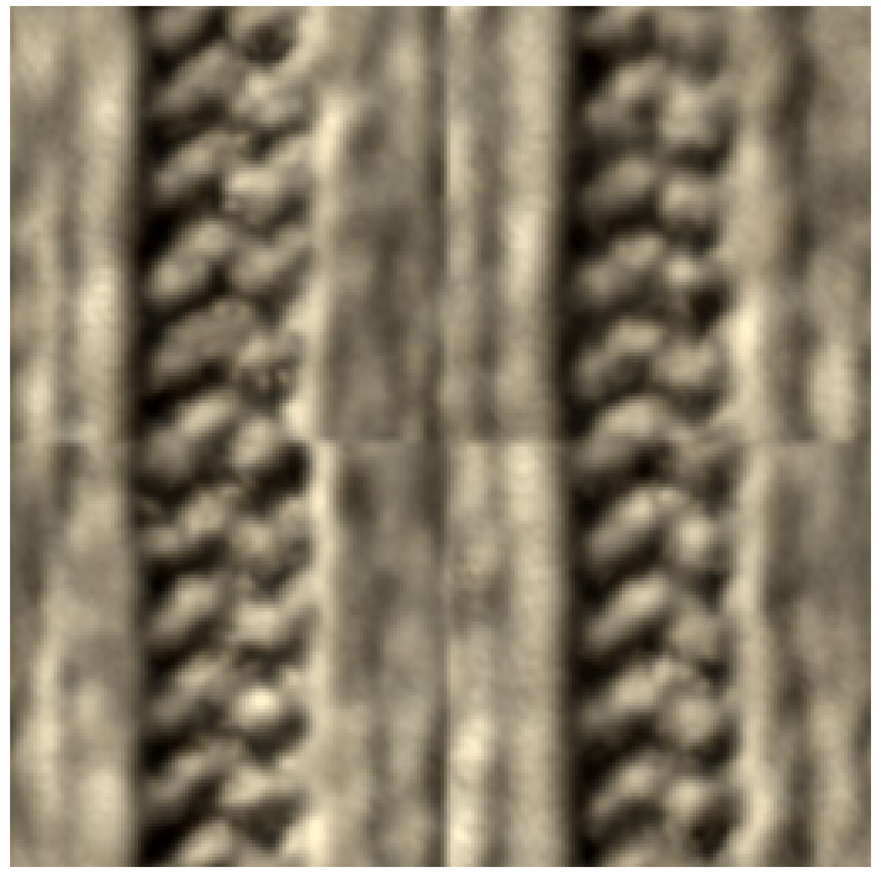
Sparse Tensor Decomposition

3.0 MB, RMS: 0.033



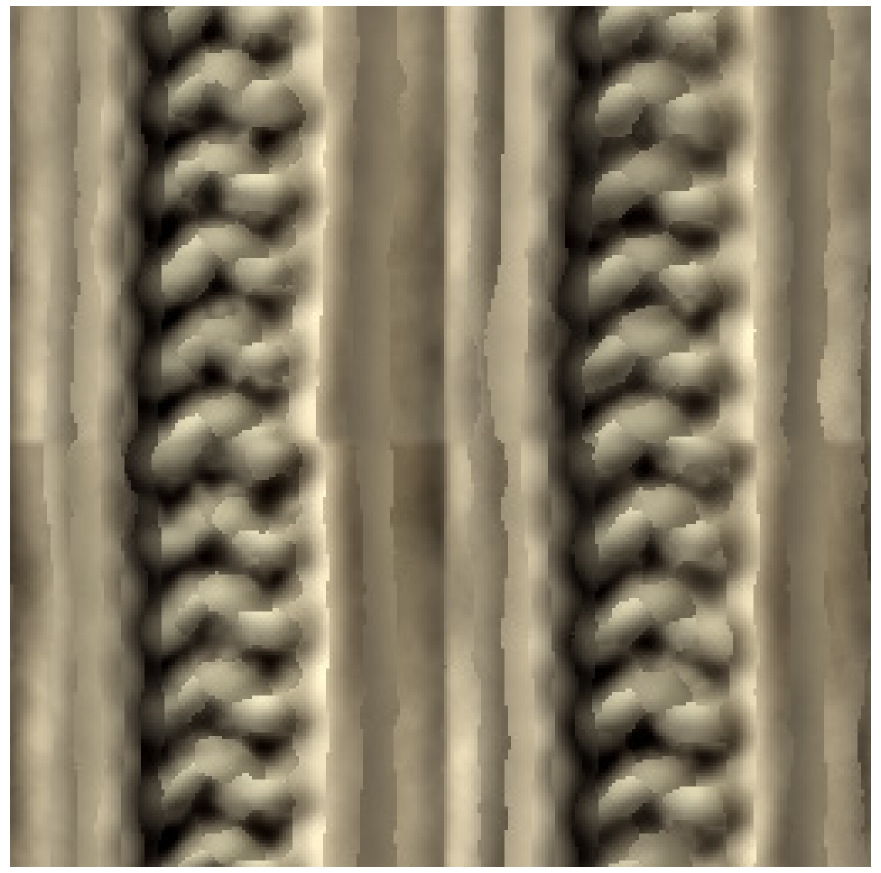
PCA

3.0 MB, RMS: 0.041



N-mode SVD

3.1 MB, RMS: 0.049

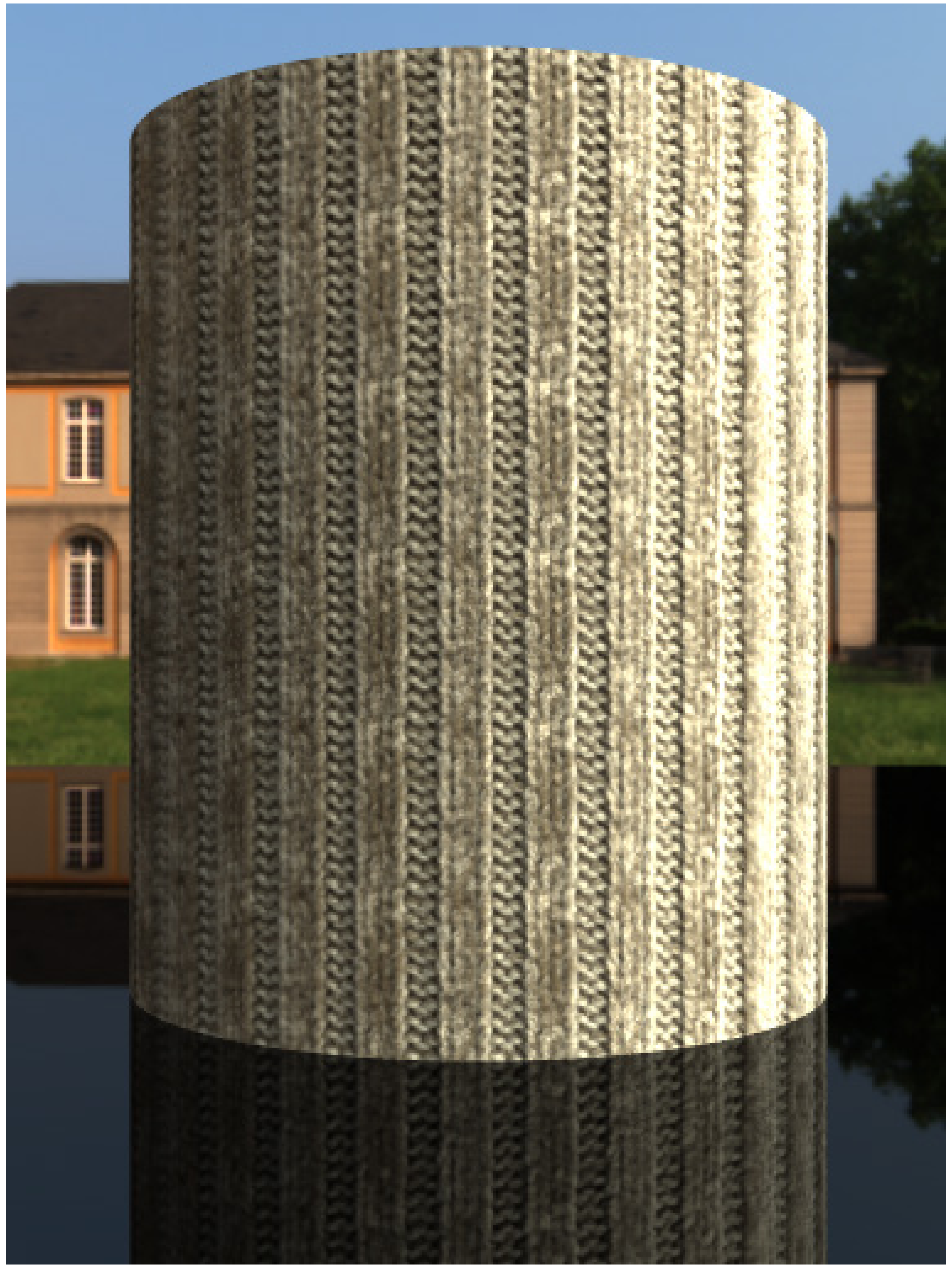


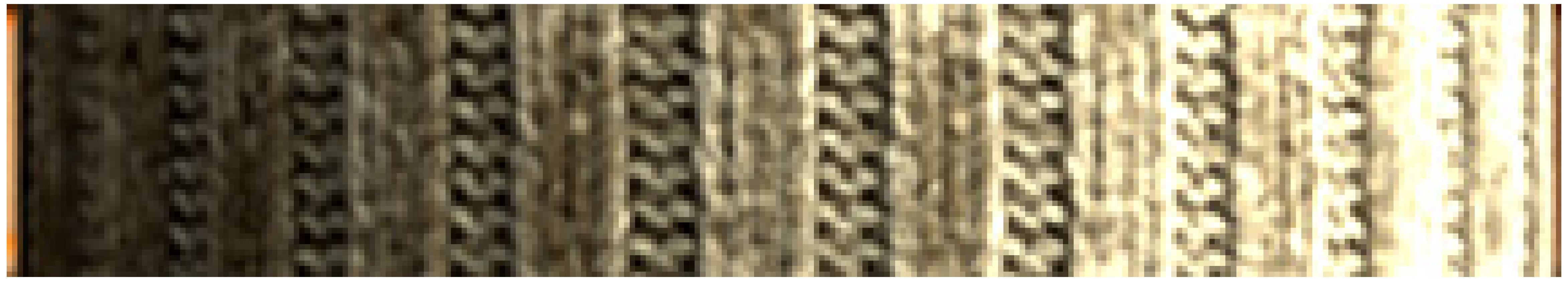
PCF

3.6 MB, RMS: 0.040

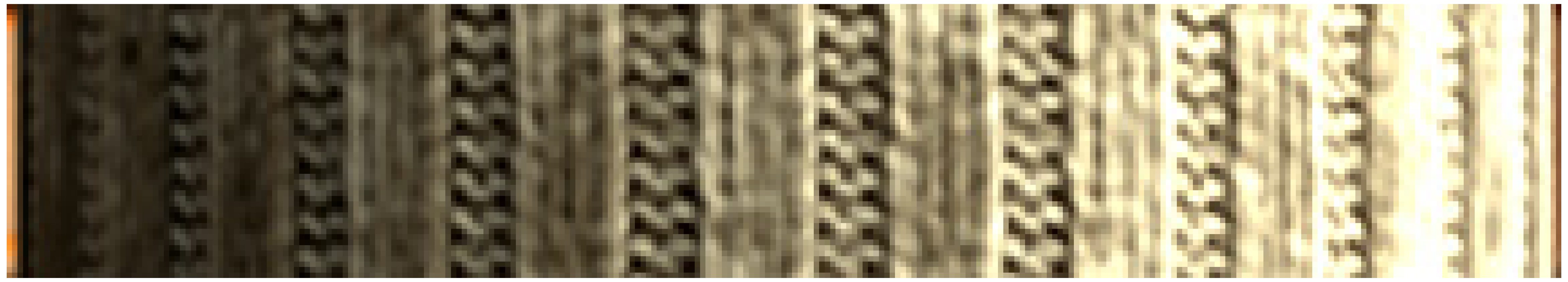
Images from [Ruiters-2009]

Sparse Tensor Decomposition

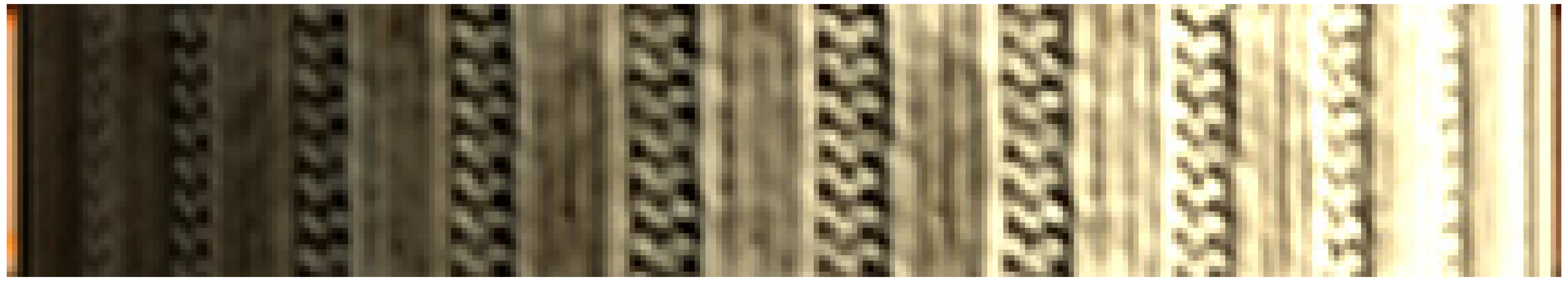




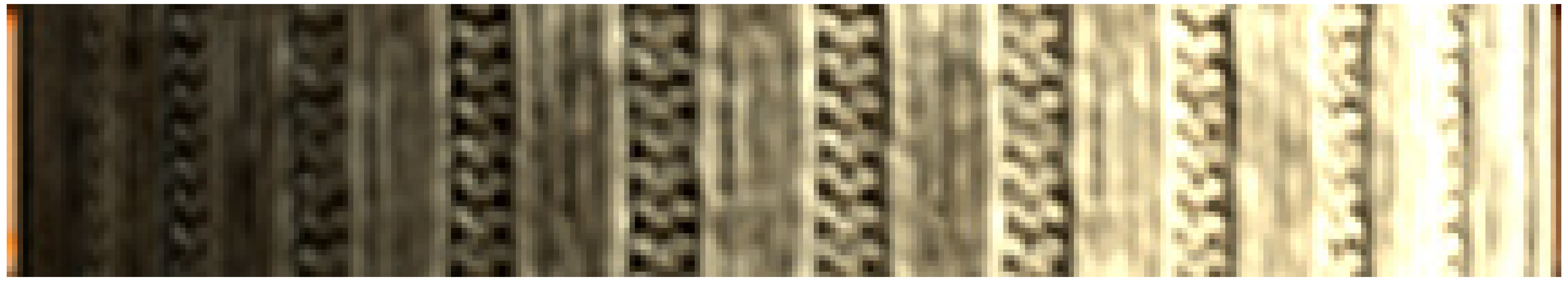
Original



Sparse Tensor Decomposition



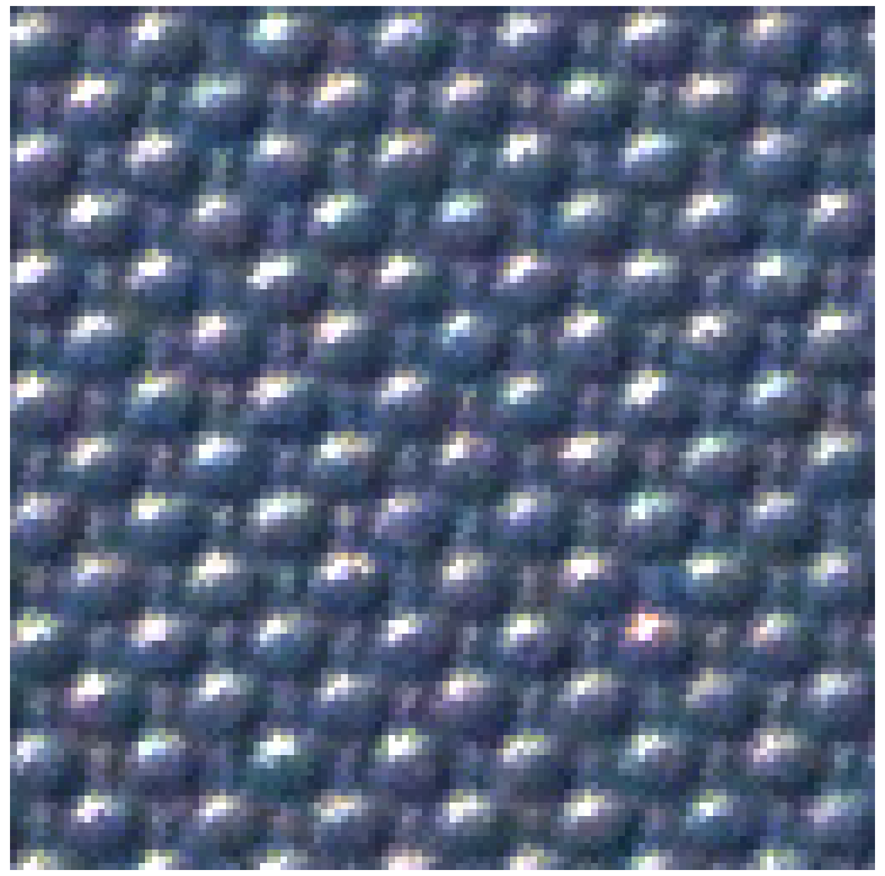
PCA



N-Mode SVD

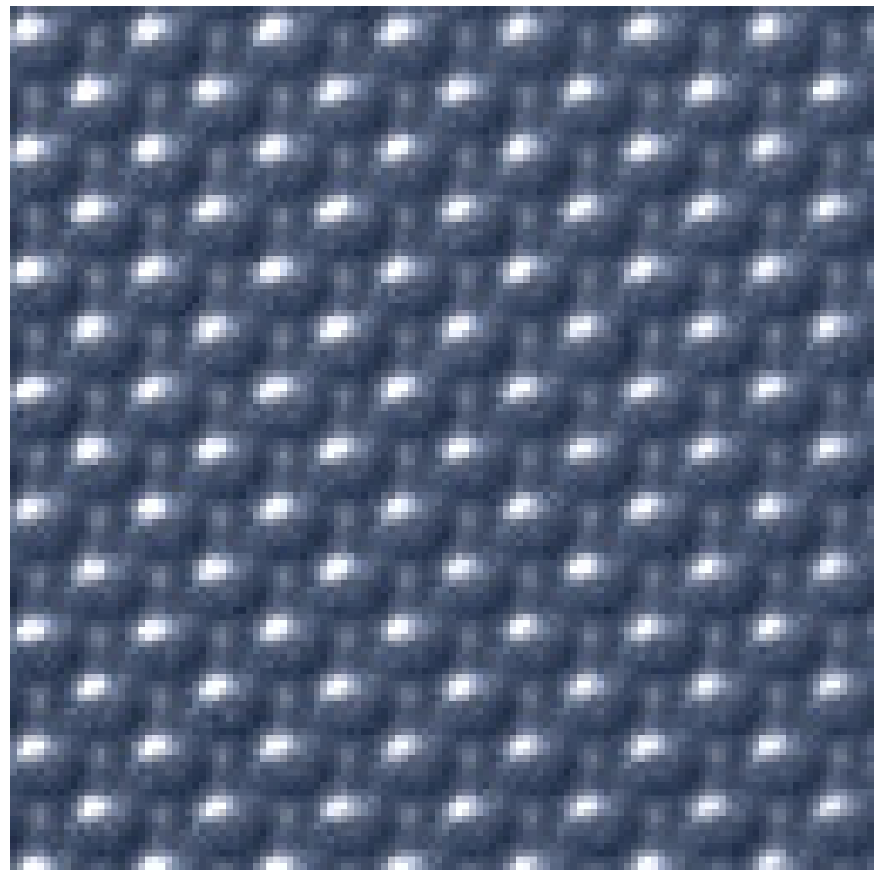
Images from [Ruiters-2009]

Sparse Tensor Decomposition



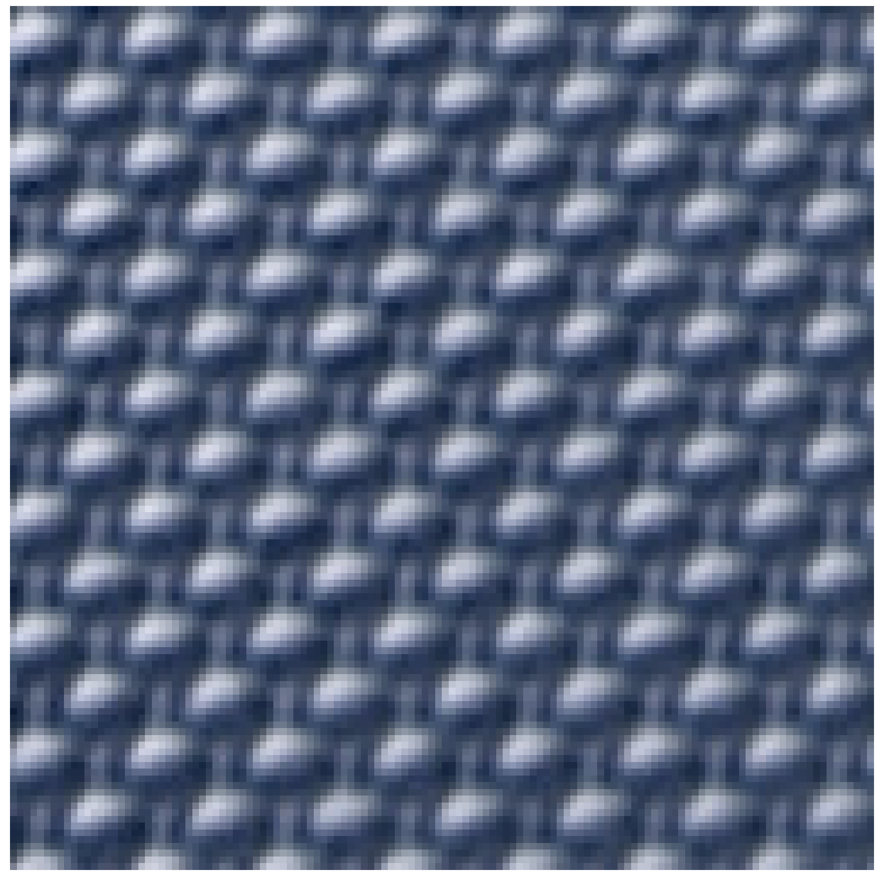
Original

2,1 GB



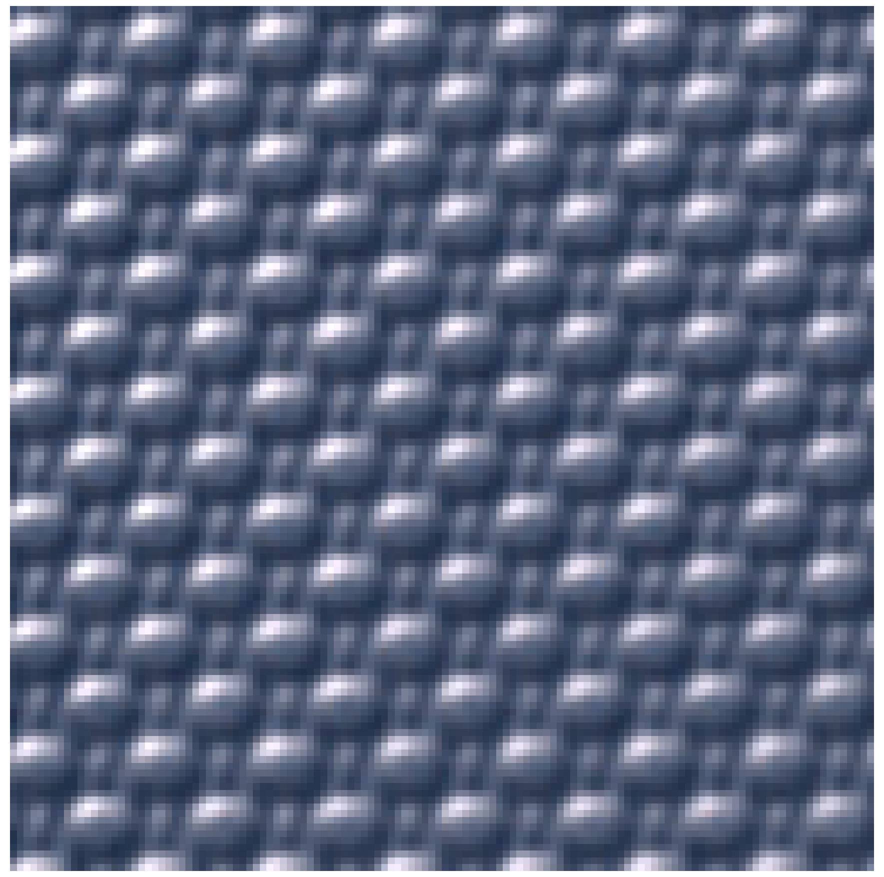
Sparse Tensor Decomposition

1.6 MB, RMS: 0.024



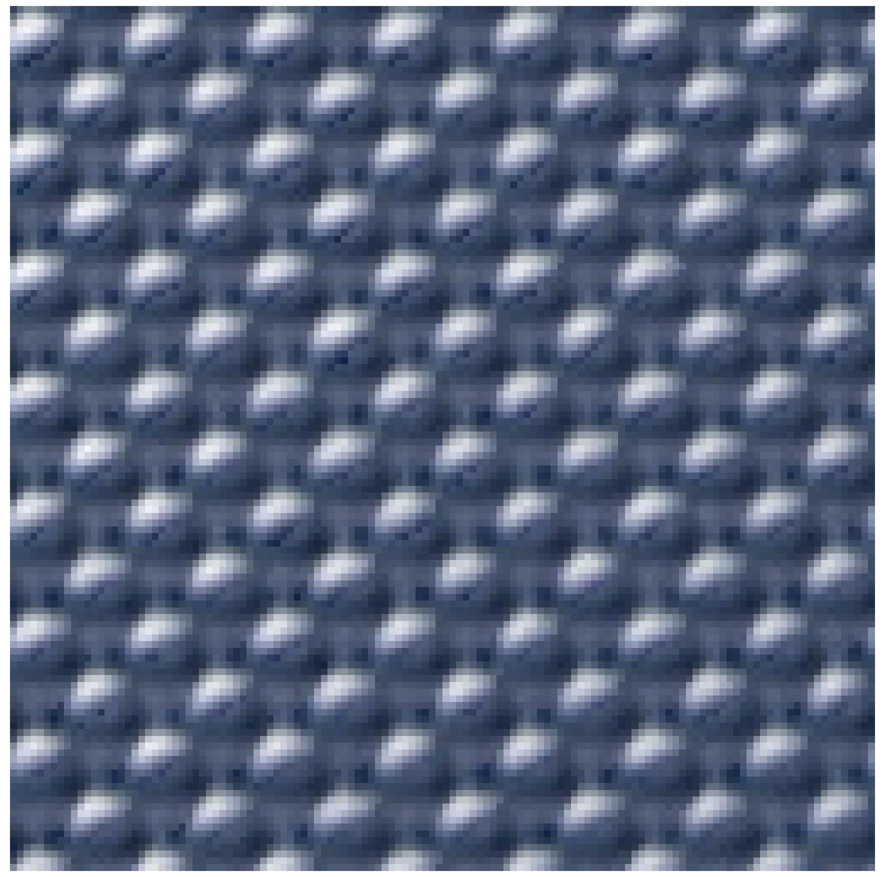
PCA

1.6 MB, RMS: 0.034



N-mode SVD

1.6 MB, RMS: 0.040

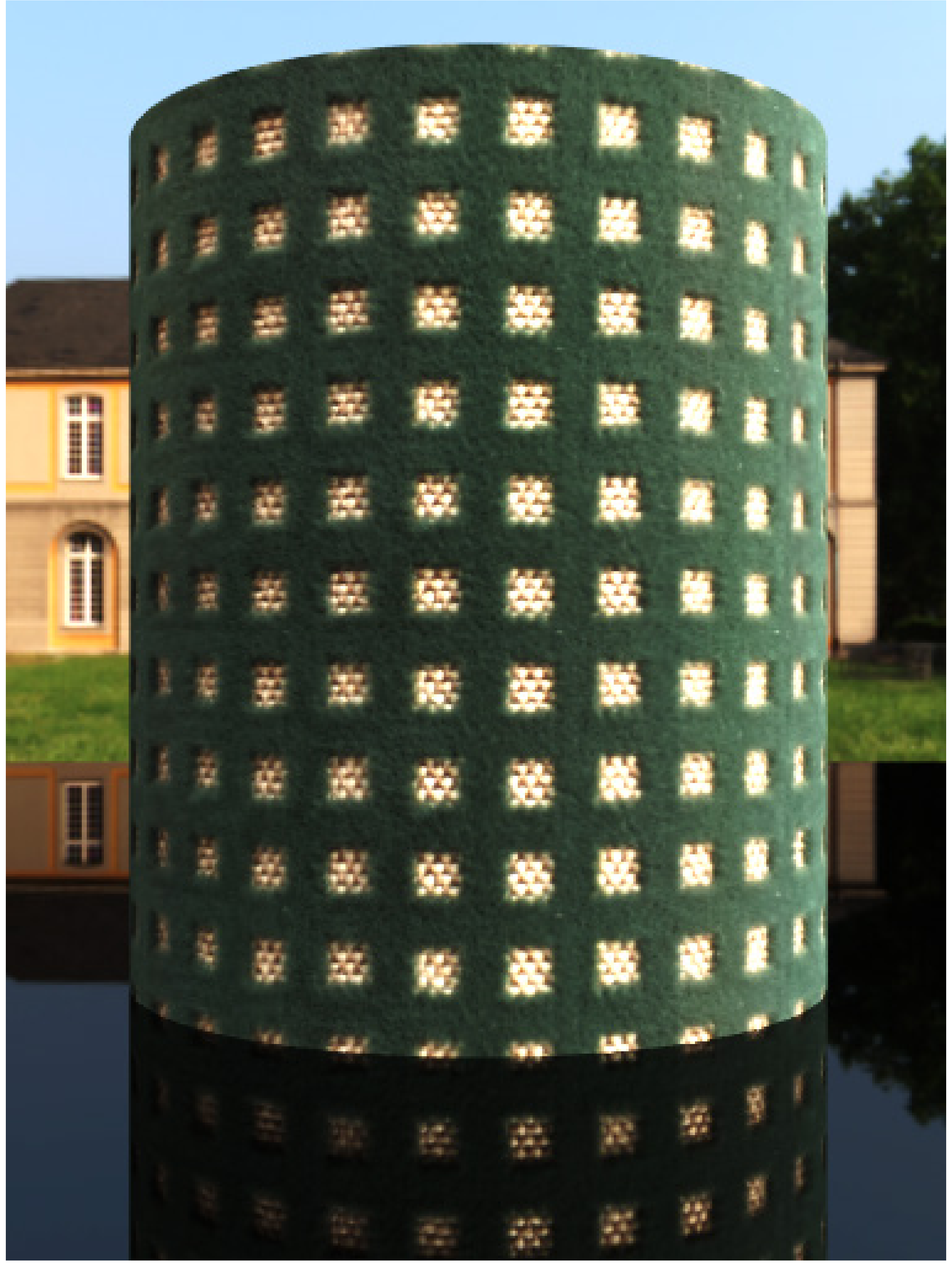


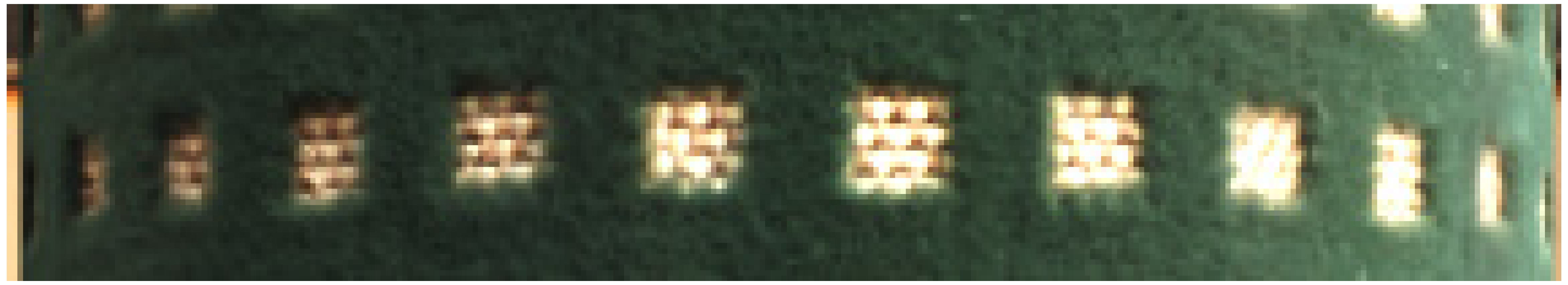
PCF

1.7 MB, RMS: 0.036

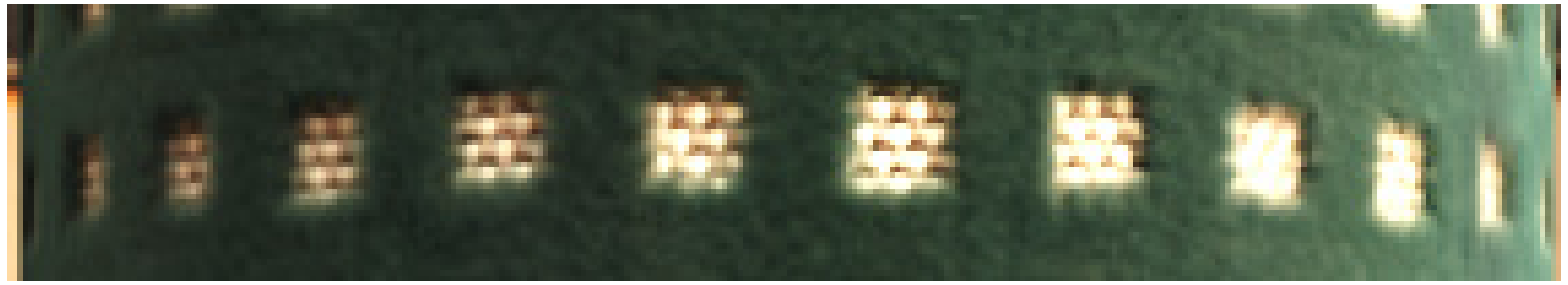
Images from [Ruiters-2009]

Sparse Tensor Decomposition

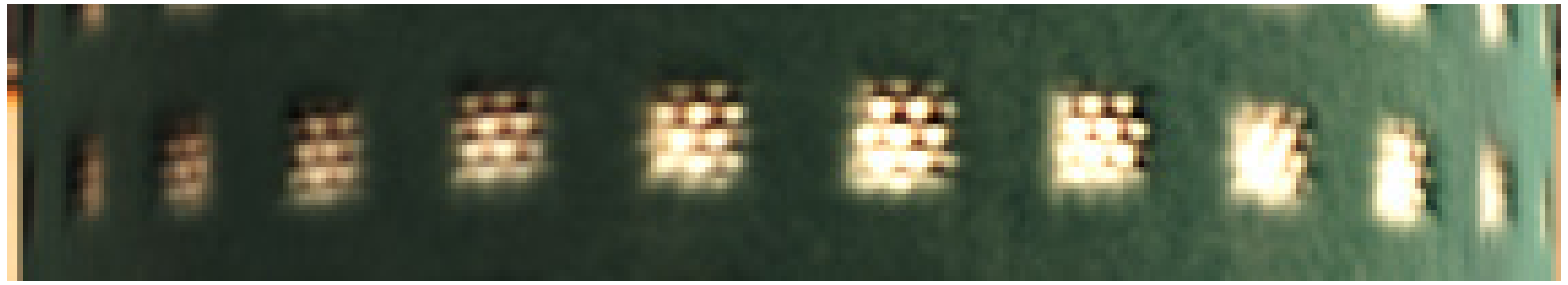




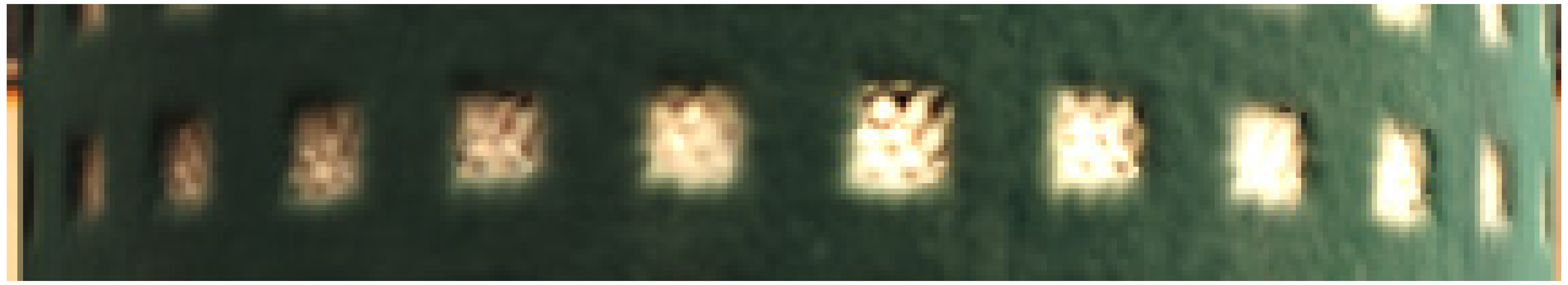
Original 14.77 GB



Sparse Tensor Decomposition 3.9MB, RMS: 0.0058



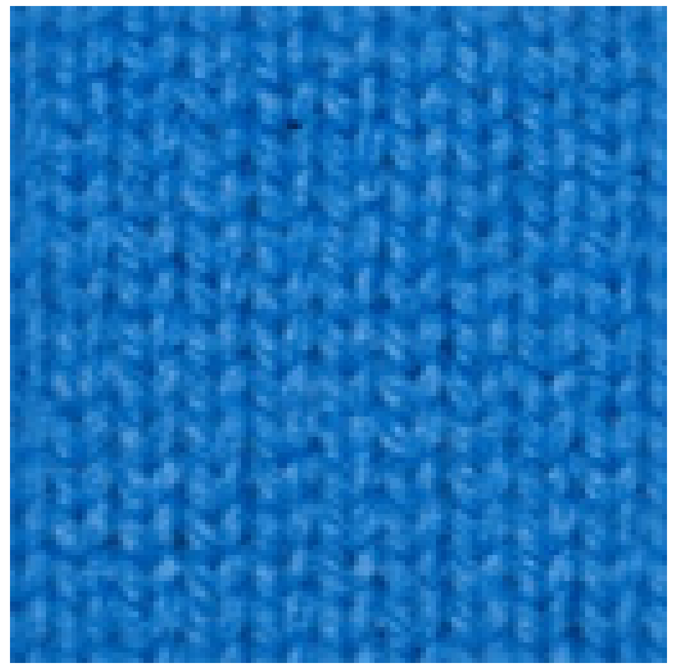
PCA 4.0MB, RMS: 0.0074



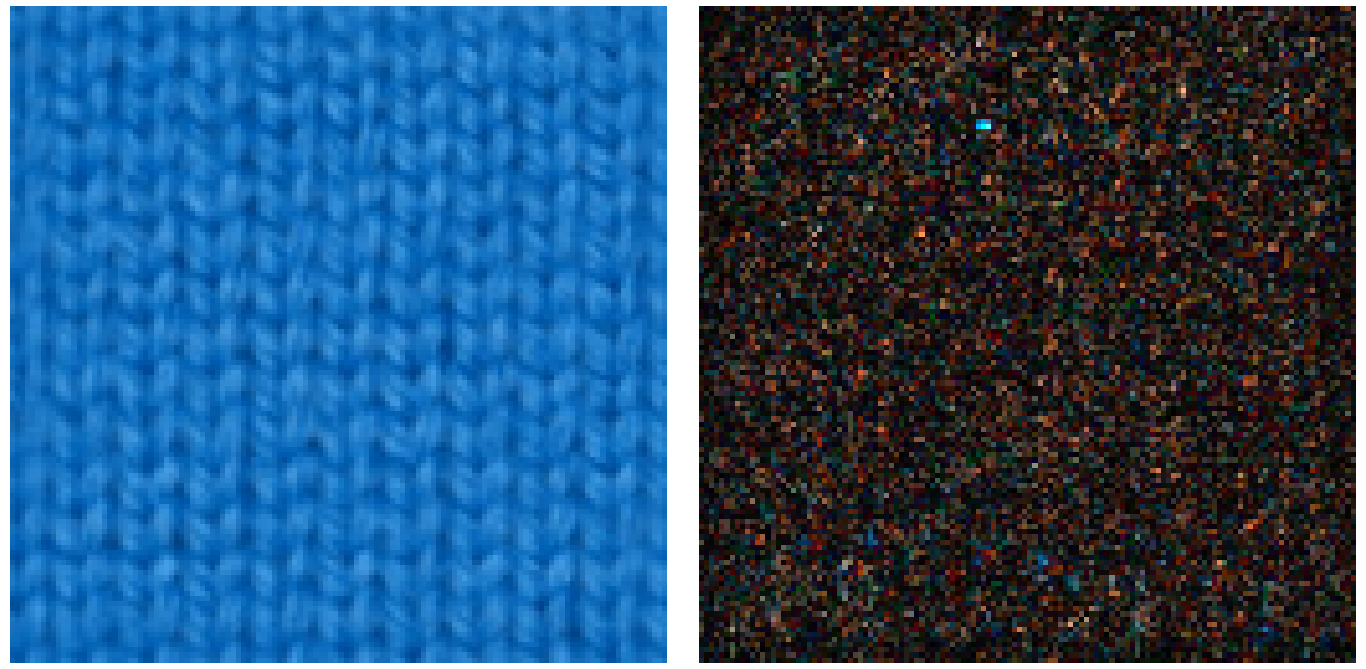
PCF 3.6 MB, RMS: 0.0082

Images from [Ruiters-2009]

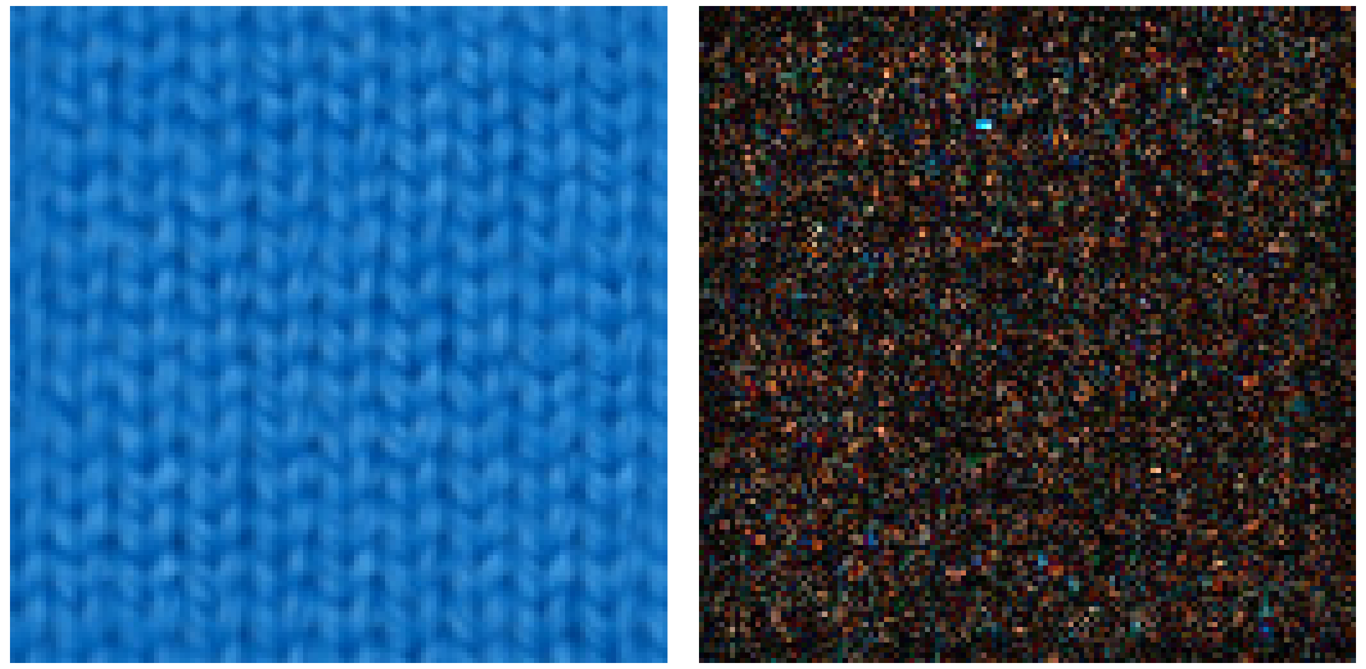
BTF Compression Results



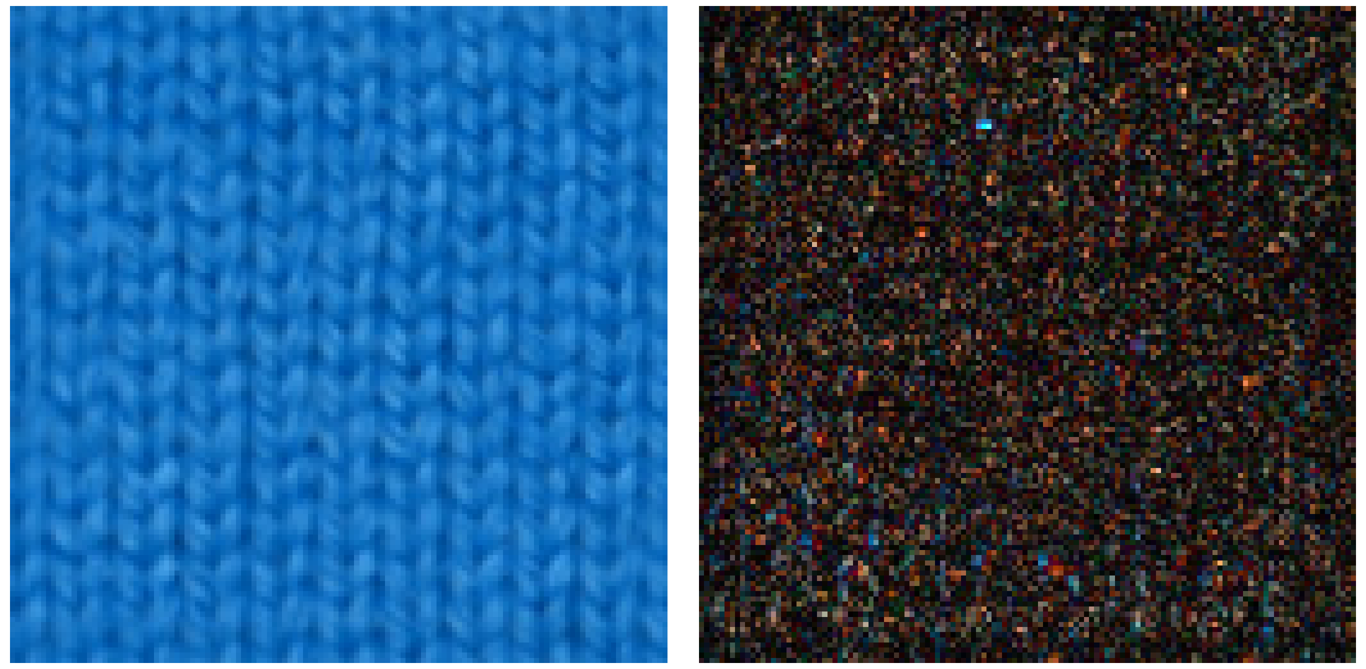
Original



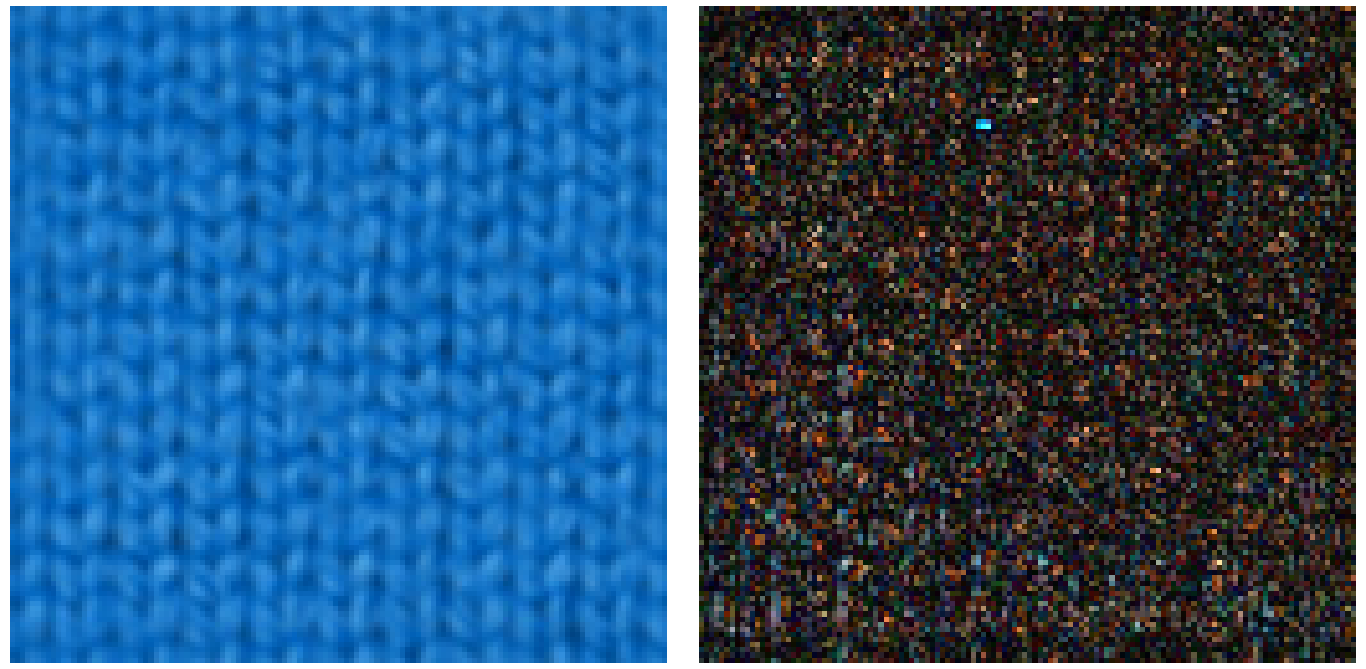
STD, $k_1 = 28, k_2 = 60$
SER 0.55%



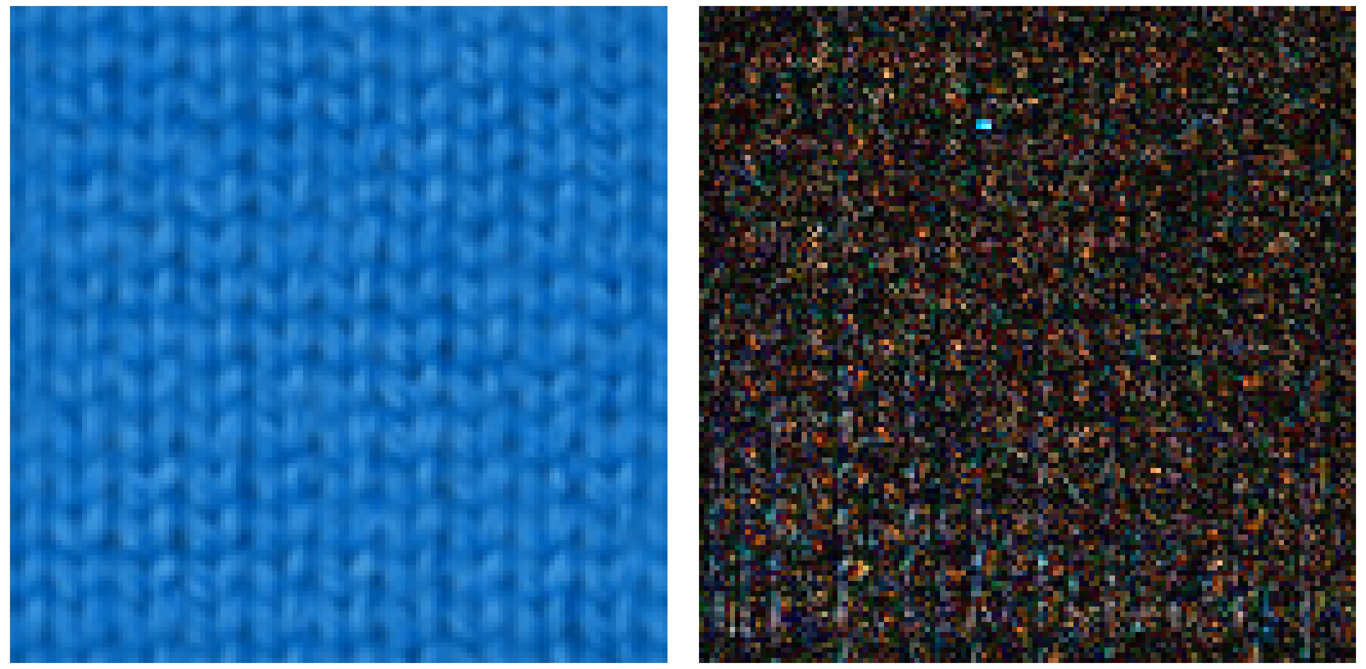
PCA, 66 components
SER 0.64%



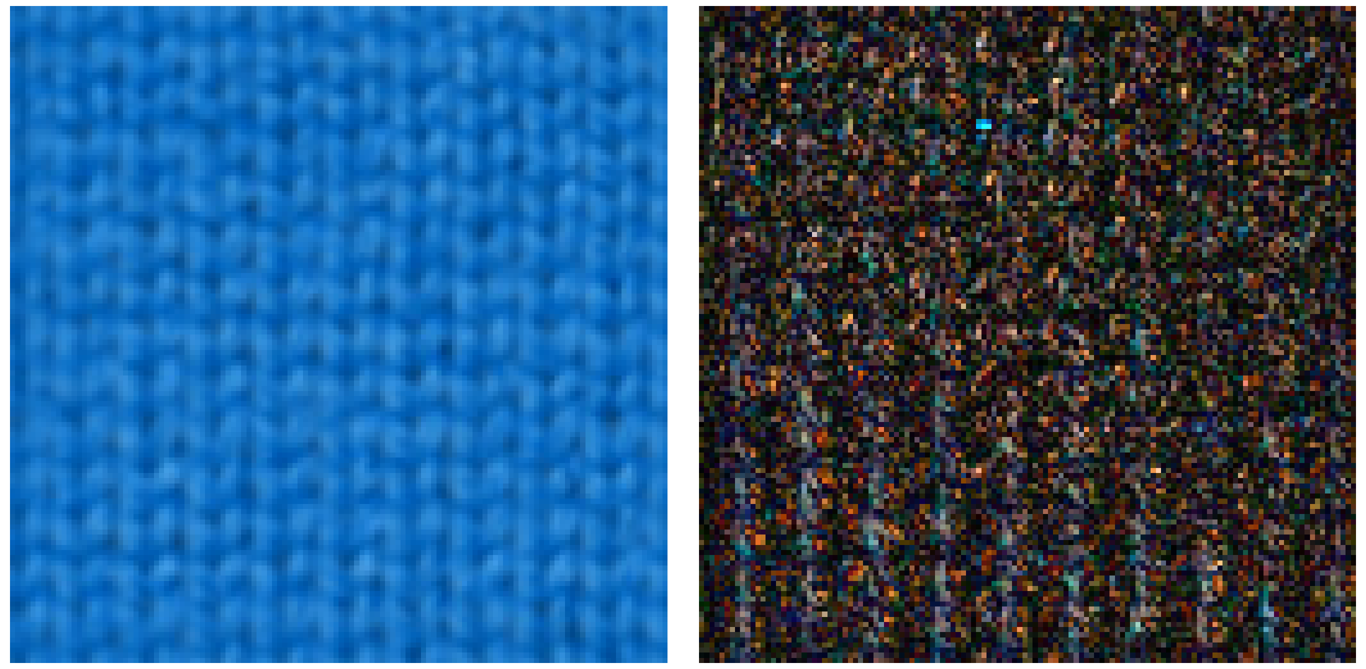
CP, 145 components
SER 0.75%



Tucker, $20 \times 28 \times 64 \times 64$ core
SER 0.85%



K-CTA [Tsai-2009]
SER 0.89%



CTA [Tsai-2009]
SER 1.06%

All BTFs were compressed to ca. 4.6 MB

SER between X and approximation $\tilde{X}$: $\frac{\overline{(X-\tilde{X})^2}}{\overline{X^2}}$

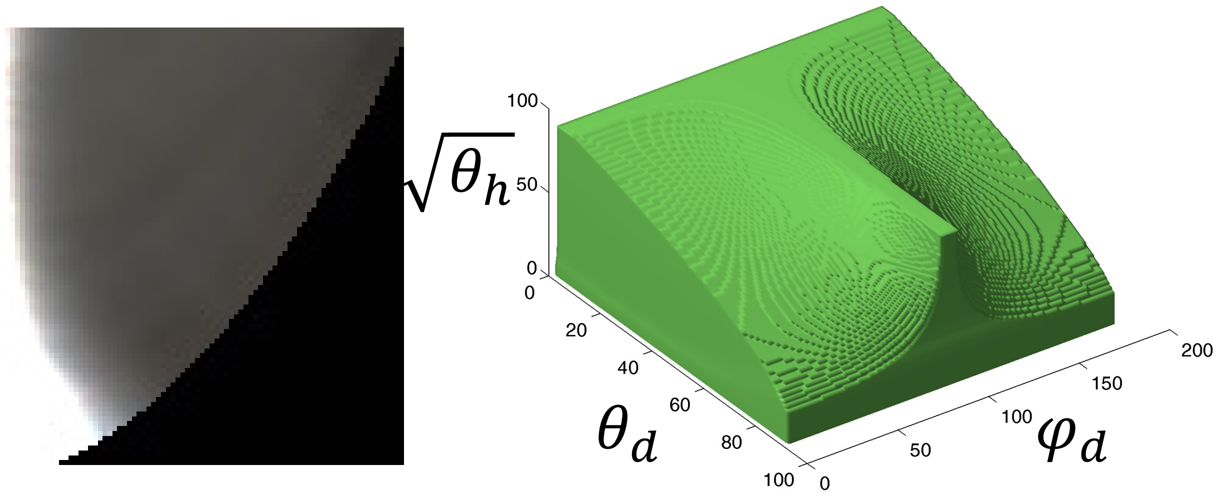
VISUALIZATION AND MULTIMEDIA LAB

universität bonn

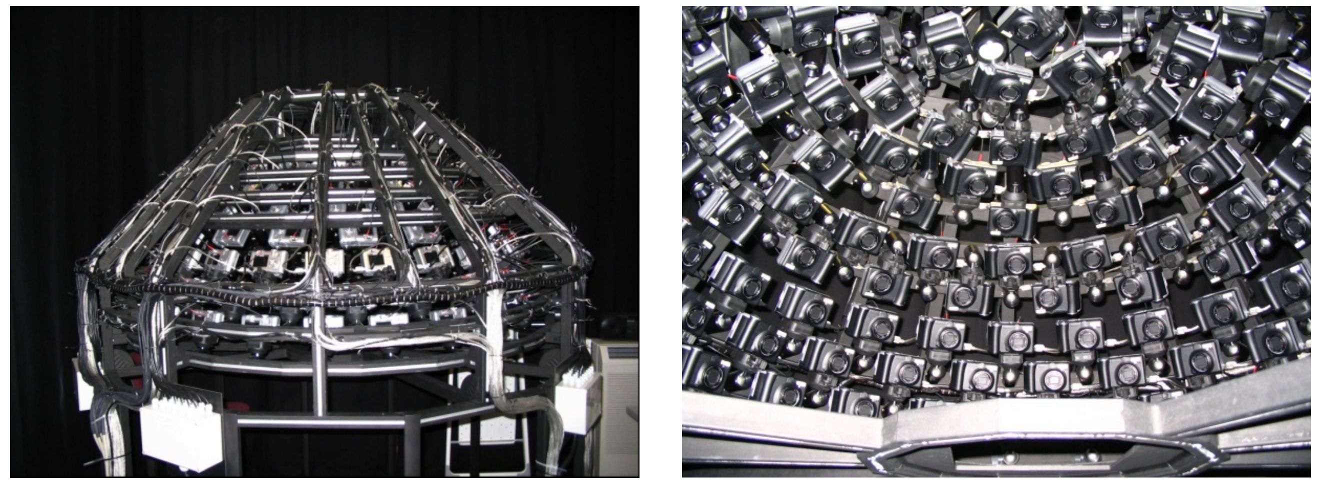
Sparse and Irregularly Sampled Input

- There are several reasons, why the input data might be incomplete and irregularly sampled
  - Not all data have been acquired
    - E.g. for some actors not all styles, actions, etc. are available
  - The domain of the parameterization is not rectangular
    - E.g. when using the Half/Diff parameterization for BRDFs
  - The measurement results in an irregular and sparse sampling
    - Might result from restrictions of the measurement device

Imputation of missing measurements from [Vlasic-2005]



Irregular domain of Half/Diff parameterization



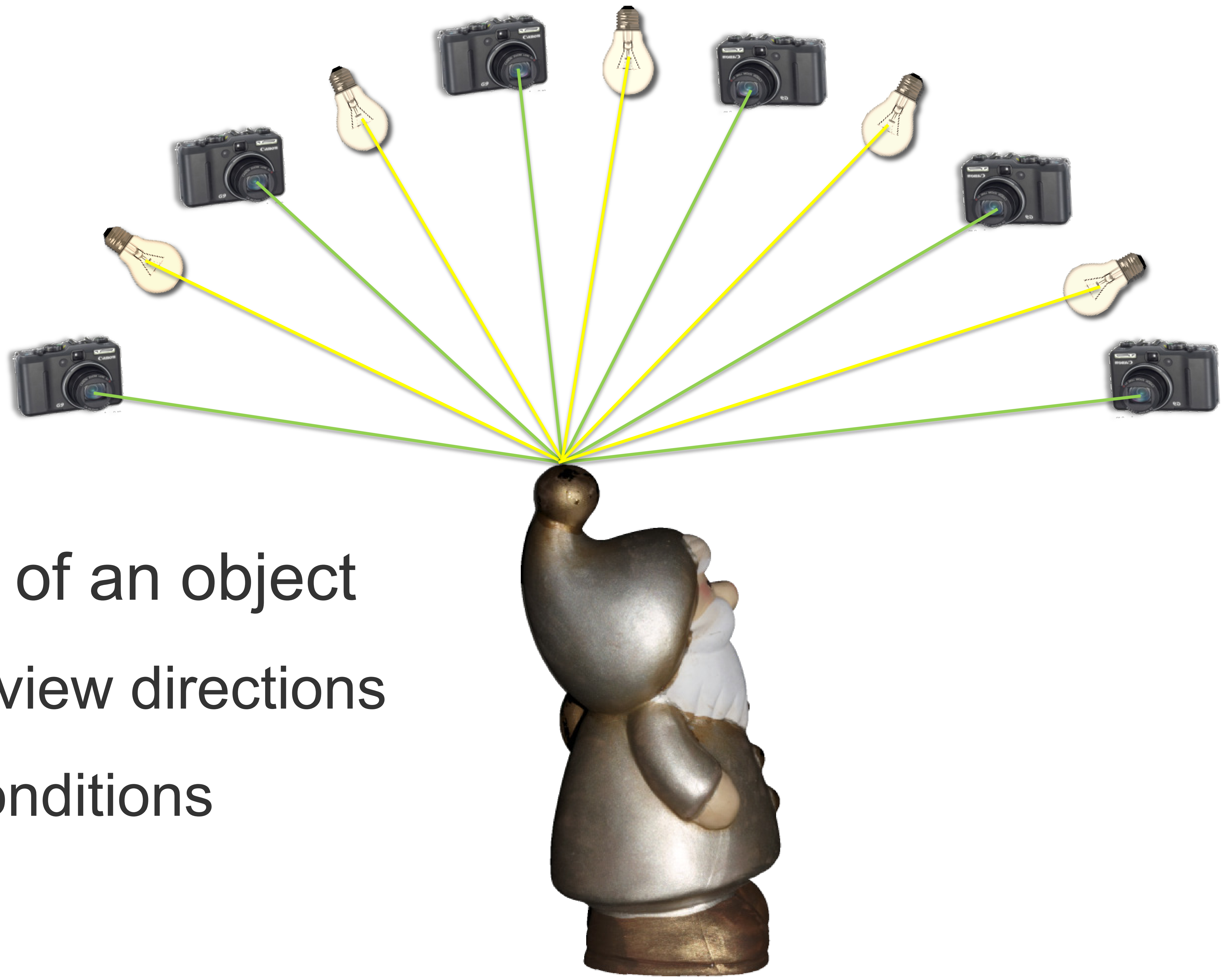
BTF Measurement device at the University of Bonn

Sparse and Irregularly Sampled Input

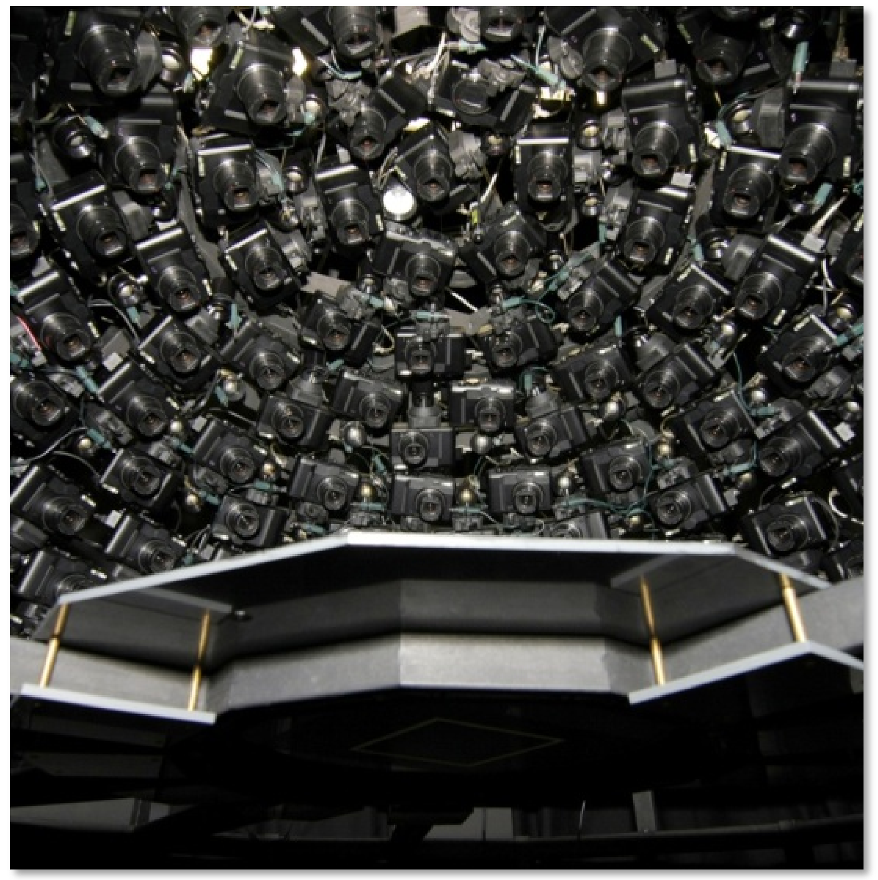
- Several strategies to cope with missing data exist
  - **Weighted Tensor Approximation**
    - Set weights on the missing data to 0 and compute weighted TA
      - The weights can be integrated into the Least Squares Problems during ALS
  - **Expectation Maximization**
    - Initialize the missing elements (e.g. with mean values)
    - In each iteration of the ALS set the missing values to the tensor decomposition
  - **Convex Optimization [Liu-2009]**
    - Solve convex optimization problem which minimizes trace norm as approximation of the tensor rank
- All of these techniques operate on the dense tensor as input
  - This can be a problem if the tensor is very large
  - E.g. a SVBRDF at the angular sampling of the MERL BRDFs and 512x512 spatial resolution
    - $3 \times 180 \times 90 \times 90 \times (512 * 512)$ tensor, ca. 4 TB

Sparse and Irregularly Sampled Input

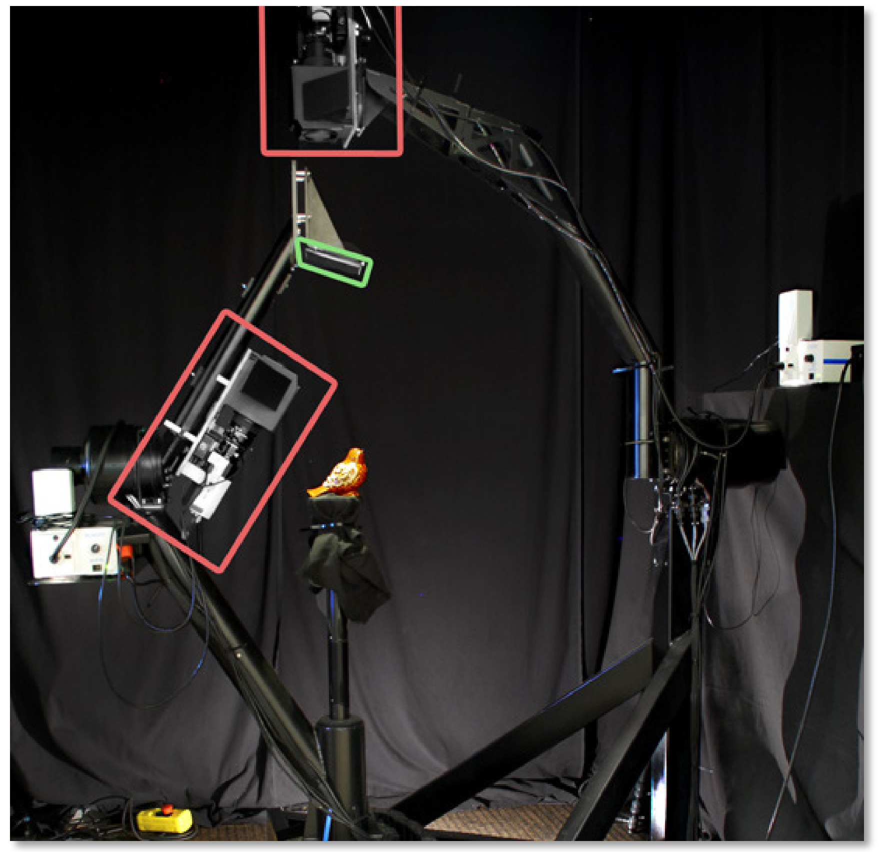
- Measurement of the reflectance of an object
  - Samples are taken from different view directions
  - and under different illumination conditions



3D Geometry

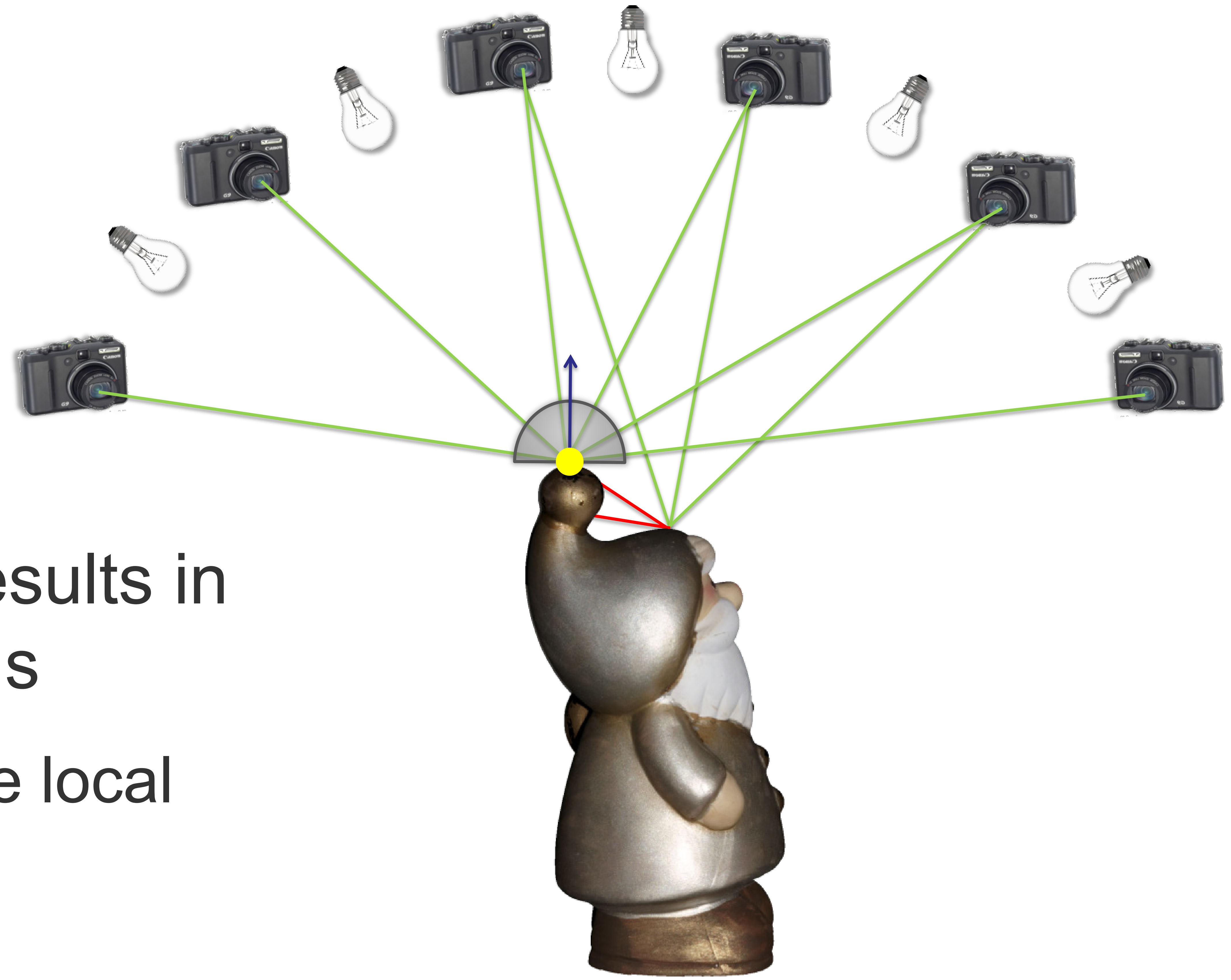


Bonn Multi-View Dome

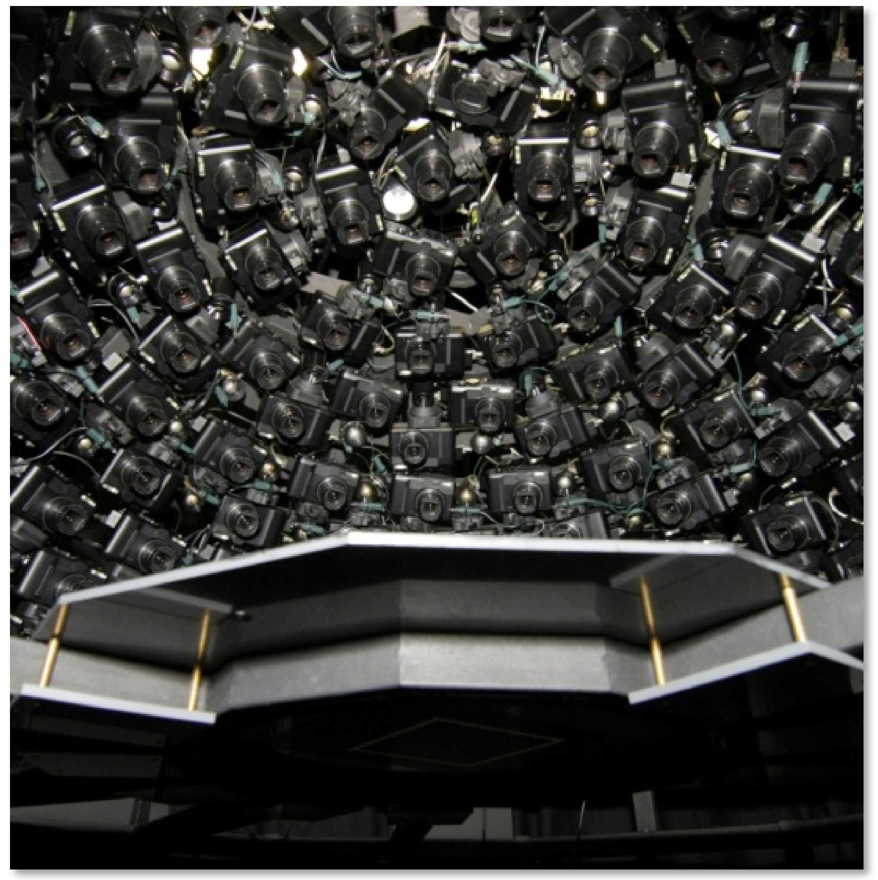


UVa Coaxial Scanner [Holroyd-2010]

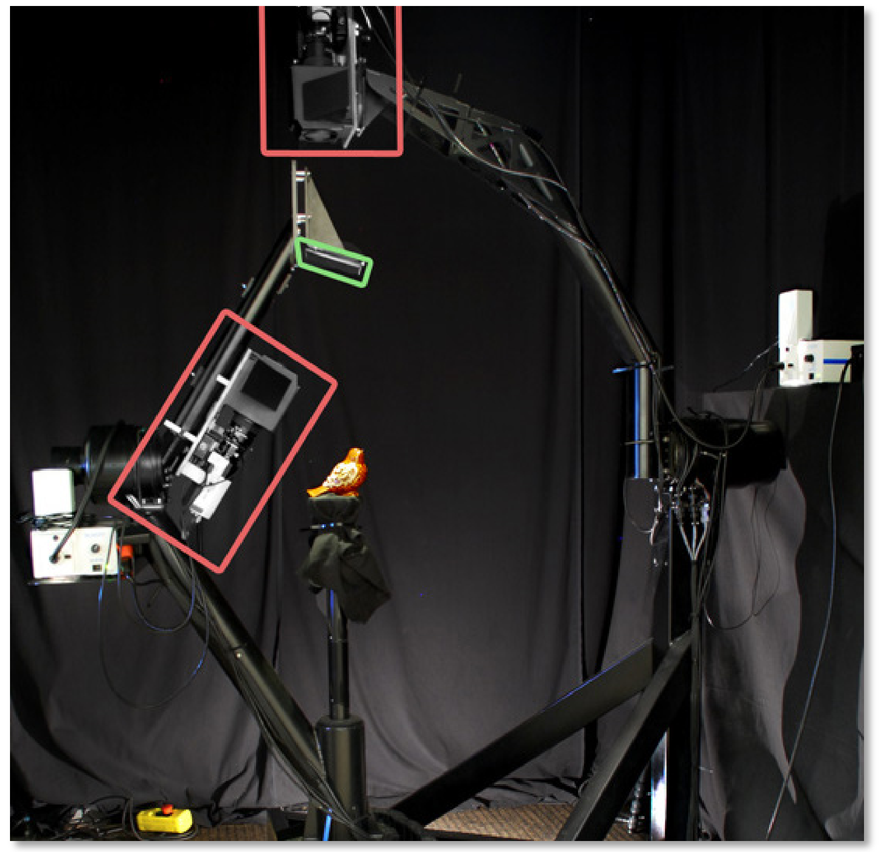
Sparse and Irregularly Sampled Input



3D Geometry



Bonn Multi-View Dome



UVa Coaxial Scanner [Holryd-2010]

- For complex geometry, this results in irregular and sparse samplings
  - Irregular due to variations of the local coordinate system
  - Sparse due to occlusions

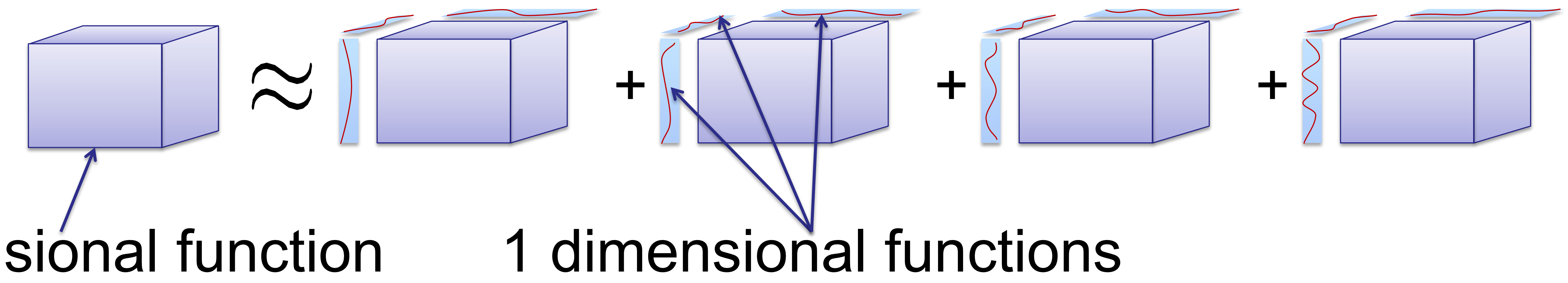
Sparse and Irregularly Sampled Input

- A continuous analogue of the CP factorization can be utilized [Ruiters-2012]
  - Model the SVBRDF as a Sum of Separable Functions (SSF)

$$\rho(\mathbf{x}) \approx \tilde{\rho}(x_1, \cdots, x_5) = \sum_{c=1}^{C} \prod_{d=1}^{5} f^{(c,d)}(x_d)$$

C Separation rank

$f^{(c,d)}$ One dimensional piecewise linear functions for each component *c* and dimension *d*



Objective Function

- To fit this representation to a given set of sample, an objective function with two terms is minimized:

$$E\left(f^{(1,1)},\cdots,f^{(C,5)}\right) = E_{\mathrm{Fit}} + E_{\mathrm{Reg}}$$

E_{Fit}

Fitting Term

- Penalizes deviations from the measured samples
  - Weighted squared error

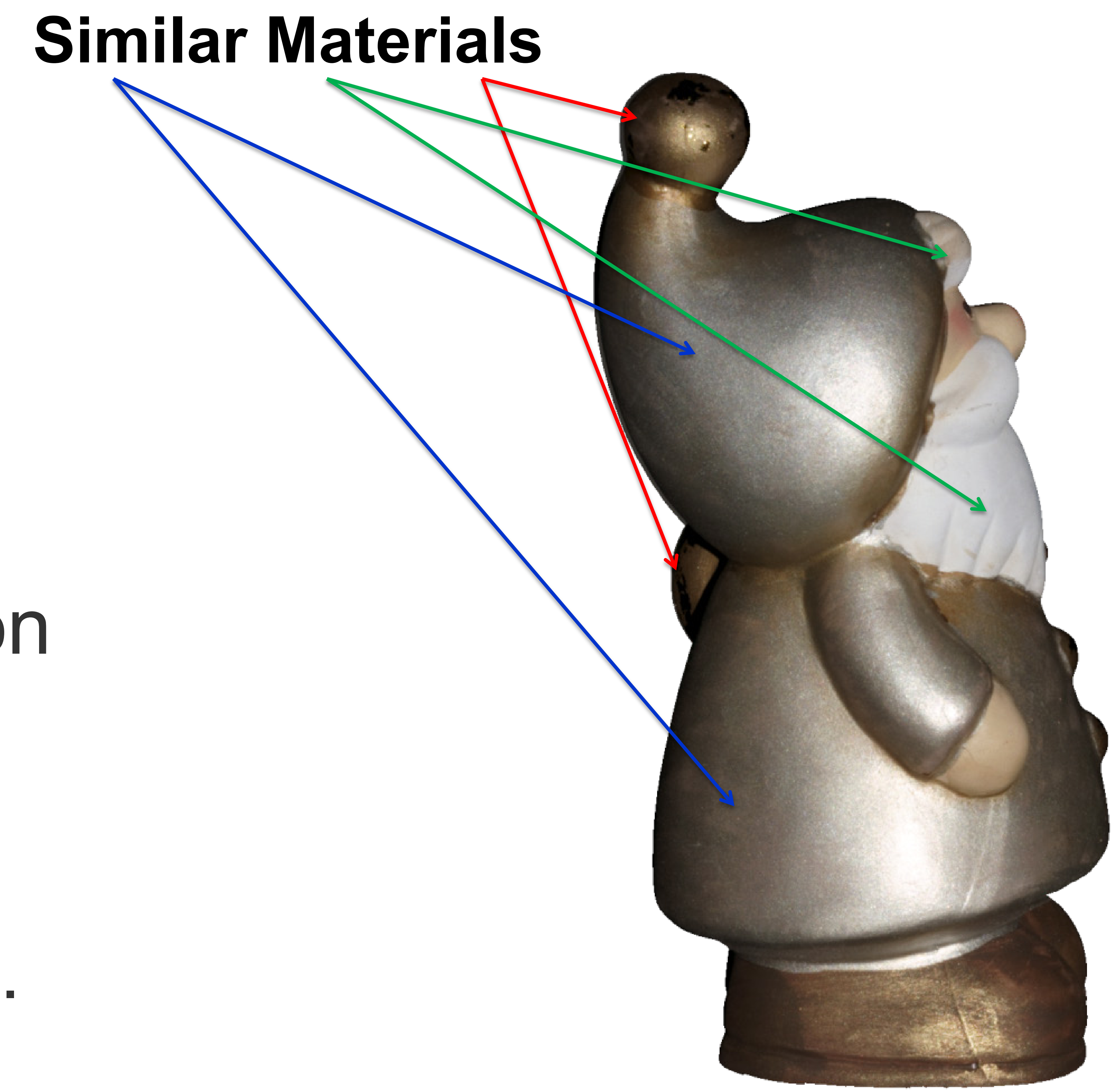
E_{Reg}

Regularization Term

- Enforces angular smoothness
  - Square of the second derivative in the angular parameter domains
- Includes non-local spatial regularization

Spatial Regularization

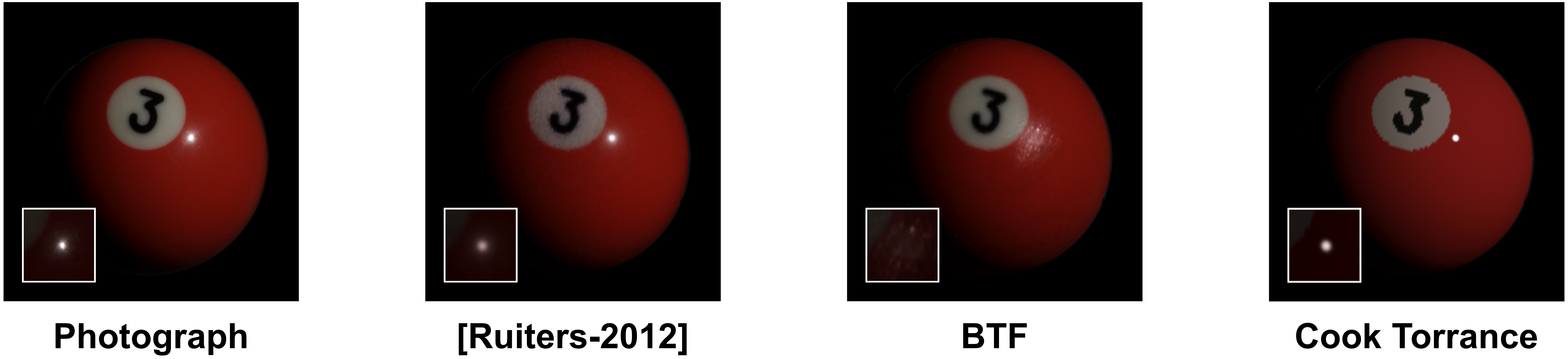
- Not enough samples available to compute material everywhere independently
- Most objects contain many regions with similar materials
- Not always in connected uniform regions
  - Smoothness regularization not adequate
- Non-local, appearance neighborhood based regularization
  - Enforces texels which are near in the appearance space to have similar materials
  - Based on the low-rank approximation from AppProp [An et al. 2008]



Optimization Algorithm

- Optimization strategy very similar to *Alternating Least Squares*
  - Iterate until convergence
    - Update the $f^{(c,d)}$ functions one at a time
      - Keeping all the other functions fixed
        - This results in a linear least squares problem
        - Linear interpolation for continuous samples not on the grid can be taken into account
        - Regularization operations can also be included into the optimization

Results



- 151x151 Views x Lights, 256 x 256 texture resolution
- Cook-Torrance was fitted with ideal distribution map
  - Tensor approximation preserves the highlight shape well, but underestimates brightness
  - BTF fails to resolve the highlight shape due to insufficient angular resolution
  - Brightness for Cook-Torrance better, shape not well preserved

Images from [Ruiters-2012]

Summary

- Clustered Tensor Approximation and K-Clustered Tensor Approximation
  - Fast decoding due to clustering/sparsity
  - Compression ratio inferior (CTA) or comparable (K-CTA) than Tucker
- Sparse Tensor Decomposition
  - Very high compression ratios (for BTFs)
    - Higher than PCA
  - Decompression faster than Tucker but linear interpolation a problem
- Sparse and irregular input
  - Can be treated as missing values
  - Alternatively, a tensor model can be fitted directly to the sparse samples
    - Integrating additional regularization constraints allows for even sparser samplings

References

Aharon-2006	AHARON M., ELAD M., BRUCKSTEIN A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. In *IEEE Transactions on Signal Processing*, 54, 11 (Nov. 2006), 4311–4322
Holroyd-2010	HOLROYD M., LAWRENCE J., ZICKLER T.: A coaxial optical scanner for synchronous acquisition of 3D geometry and surface reflectance. In *ACM Transactions on Graphics* 29, 4 (2010), 99.
Liu-2009	LIU J., MUSIALSKI P., WONKA P., YE J.: Tensor completion for estimating missing values in visual data. In *International Conference on Computer Vision*, (2009), pp. 2114 –2121.
Ruiters-2009	RUITERS R., KLEIN R.: BTF compression via sparse tensor decomposition. In *Computer Graphics Forum* 28, 4 (July 2009), 1181–1188.
Ruiters-2012	RUITERS R., SCHWARTZ C., KLEIN R.: Data driven surface reflectance from sparse and irregular samples. In *Computer Graphics Forum* 31, 2 (May 2012), 315–324.
Sun-2007	SUN X., ZHOU K., CHEN Y., LIN S., SHI J., GUO B.: Interactive relighting with dynamic BRDFs. In *ACM Transactions on Graphics* 26, 3 (2007), 27.
Tsai-2006	TSAI Y.-T., SHIH Z.-C.: All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. In *ACM Transactions on Graphics* 25, 3 (2006), pp. 967–976.
Tsai-2009	TSAI Y.-T.: Parametric Representations and Tensor Approximation Algorithms for Real-Time Data-Driven Rendering. *Ph.D. Dissertation, National Chiao Tung University*, May 2009.
Tsai-2012	TSAI Y.-T., SHIH Z.-C.: K-clustered tensor approximation: A sparse multilinear model for real-time rendering. In *ACM Transactions on Graphics* 31, 3 (2012), 19.
Vlasic-2005	VLASIC D., BRAND M., PFISTER H., POPOVIĆ J.: Face transfer with multilinear models. In *ACM Transactions on Graphics* 24, 3 (2005), pp. 426-433