Tensor Decomposition in Graphics and Interactive Visualization

Rafael Ballester-Ripoll

rballester@ifi.uzh.ch

CDISE Seminar
24th June 2016

Universität Zürich[UZH]

Table of Contents

Section 1

Motivation

Introduction

- Tensor decomposition: **dozens of applications**
  - Many different domains

Introduction

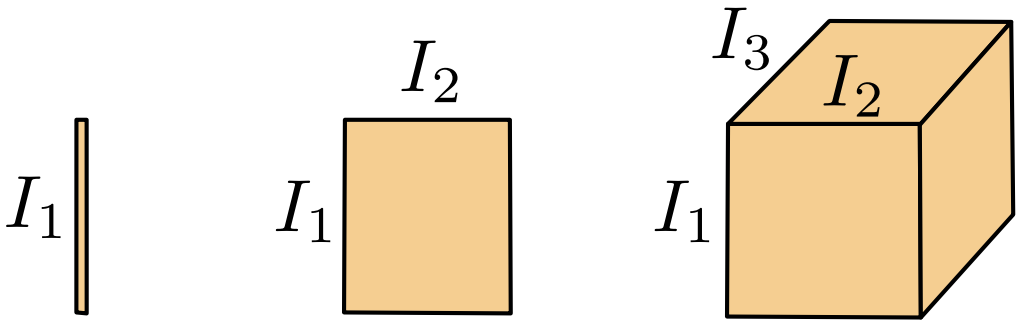
- Tensor decomposition: **dozens of applications**
  - Many different domains
- I will cover a set of related approaches:
  - Methods oriented towards **visualization** and **human interaction**

Introduction

- Tensor decomposition: **dozens of applications**
  - Many different domains
- I will cover a set of related approaches:
  - Methods oriented towards **visualization** and **human interaction**
- Key: spatial data is well-compressible
  - On the ground of low-rank expressions

What is a Tensor?

- For us, a *multidimensional array*:
  - A vector is a 1D tensor
  - A matrix is a 2D tensor
  - Etc...



- Visual data has **redundancy** along **several dimensions**

Why Tensors?

- Tensor decompositions **generalizes**:
  - Singular value decomposition
  - **Frequency-based** transforms
  - Separable **wavelets**

Why Tensors?

- Tensor decompositions **generalizes**:
  - Singular value decomposition
  - **Frequency-based** transforms
  - Separable **wavelets**
- Designed to overcome the **curse of dimensionality**
  - The more dimensions, the better the advantage

Why Tensors?

- Tensor decompositions **generalizes**:
  - Singular value decomposition
  - **Frequency-based** transforms
  - Separable **wavelets**
- Designed to overcome the **curse of dimensionality**
  - The more dimensions, the better the advantage
- Good **compression quality**
- Good at **selecting features**

Tensor Approximation in Computer Graphics

- **Texture synthesis** [VBP+05,CSS08,WXC+08]
- **Multiresolution rendering** [SIM+11,SMP13,BGG+14]
- **Micro-tomography compression** [BSP15, BP15]
- **Bidirectional texture functions** [WWS+05,WXC+08,RK09,TS12,Tsa15]
- **Bidirectional reflectance distribution functions** [RSK12]

Section 2

Tensor Methods for Volume Compression and Visualization

Situation

- Large-scale interactive visualization: **complex data** over regular grids
  - Computer tomography, simulations, etc.
  - We tolerate (and encourage) approximations: $\|\mathscr{A} - \widetilde{\mathscr{A}}\| \leq \epsilon$

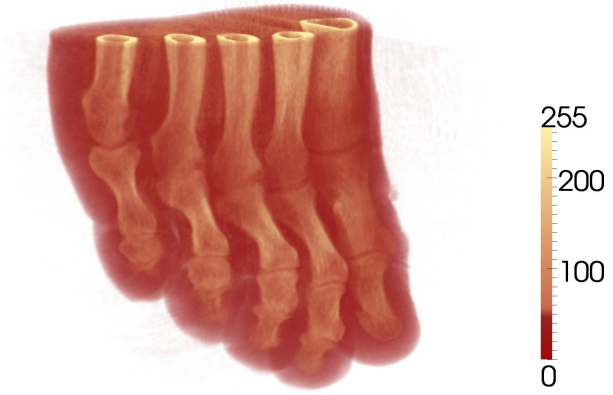
Situation

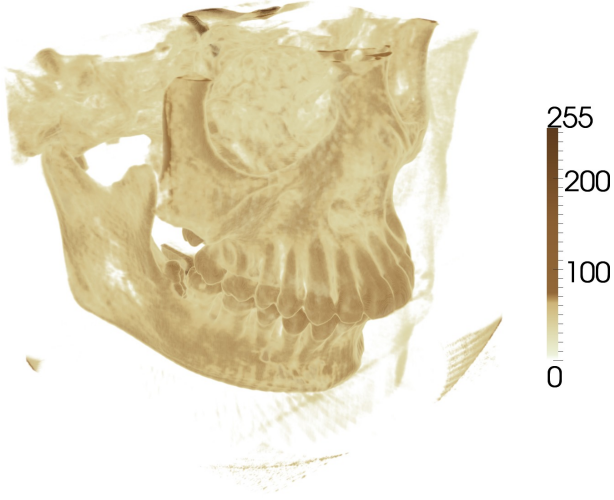
- Large-scale interactive visualization: **complex data** over regular grids
  - Computer tomography, simulations, etc.
  - We tolerate (and encourage) approximations: $\|\mathscr{A} - \widetilde{\mathscr{A}}\| \leq \epsilon$
- Asymmetric pipeline:
  - Slow decomposition is acceptable (offline stage)
  - But **fast reconstruction** is critical (online stage). We use parallel algorithms for graphics cards

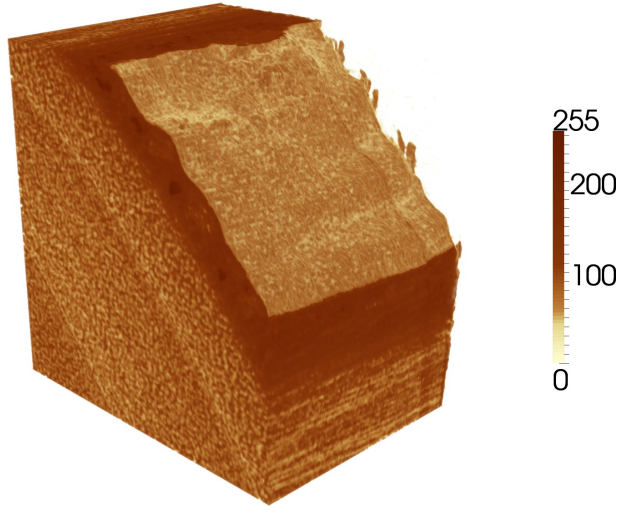
Situation

- Large-scale interactive visualization: **complex data** over regular grids
  - Computer tomography, simulations, etc.
  - We tolerate (and encourage) approximations: $\|\mathscr{A} - \widetilde{\mathscr{A}}\| \leq \epsilon$
- Asymmetric pipeline:
  - Slow decomposition is acceptable (offline stage)
  - But **fast reconstruction** is critical (online stage). We use parallel algorithms for graphics cards
- In volume rendering: data sets of size I^3, with I large (e.g. 2048).
  - Possible added dimension(s): features (RGB color, X-ray density), time, etc.

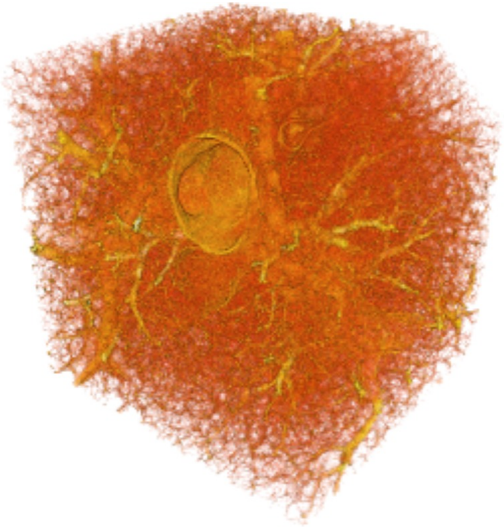
Example Tensors (I)

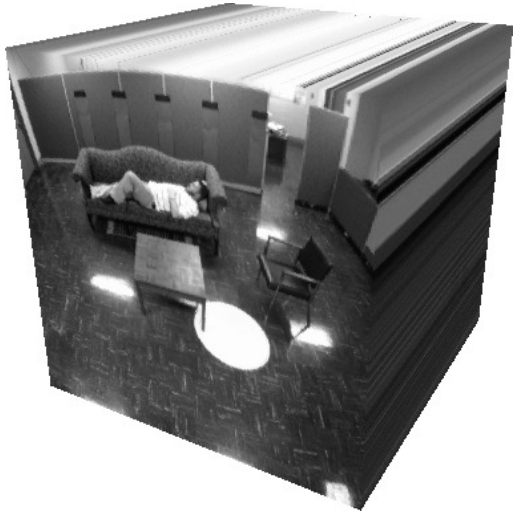


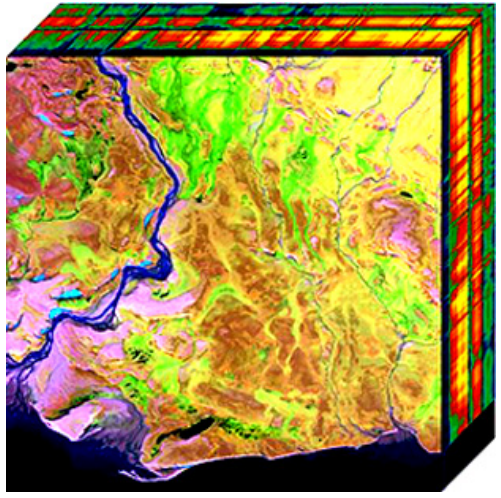




Example Tensors (II)







Bidirectional Texture Functions

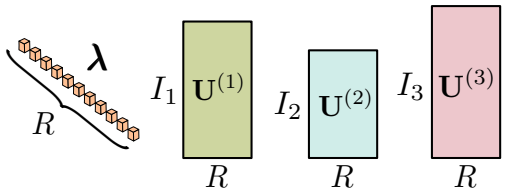
- 7-dimensional tensor

Useful Models

- We use multilinear methods that work for several models

Useful Models

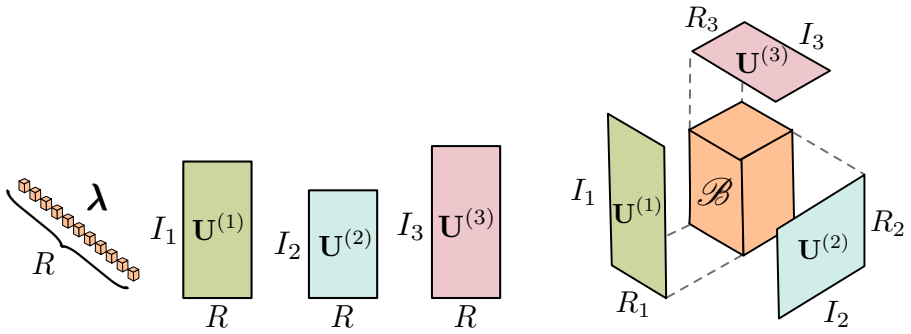
- We use multilinear methods that work for several models



CP

Useful Models

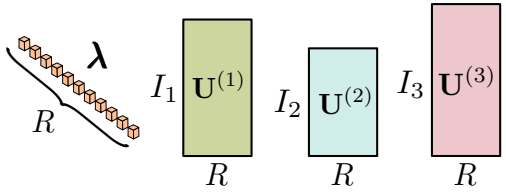
- We use multilinear methods that work for several models



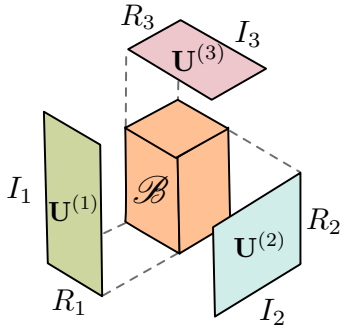
CP Tucker

Useful Models

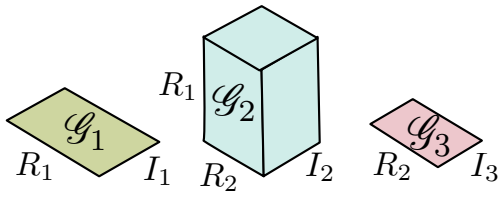
- We use multilinear methods that work for several models



CP



Tucker

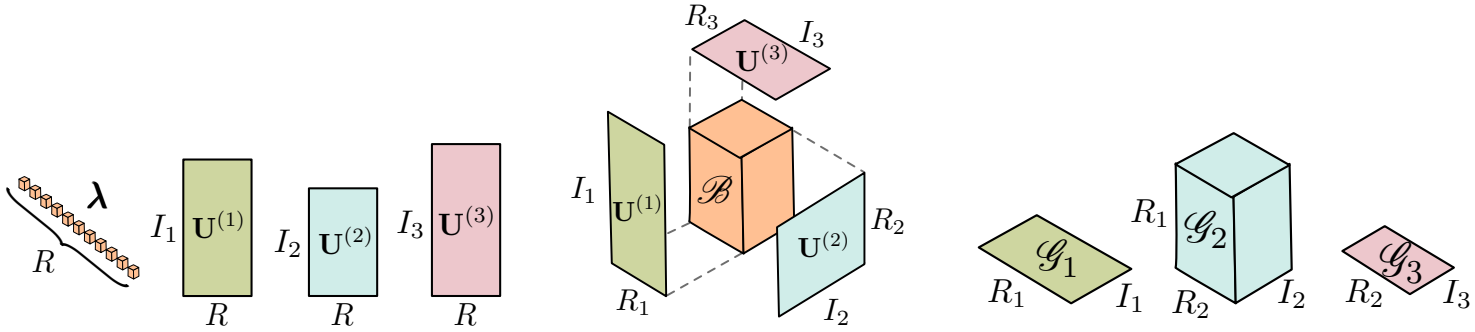


Tensor Train

VISUALIZATIONAND MULTIMEDIALAB

Useful Models

- We use multilinear methods that work for several models



- In the graphics literature, **Tucker** is the **most common**
  - CP has expensive reconstruction (usually high rank $R \gg I$)
  - TT is still quite recent

In a Nutshell

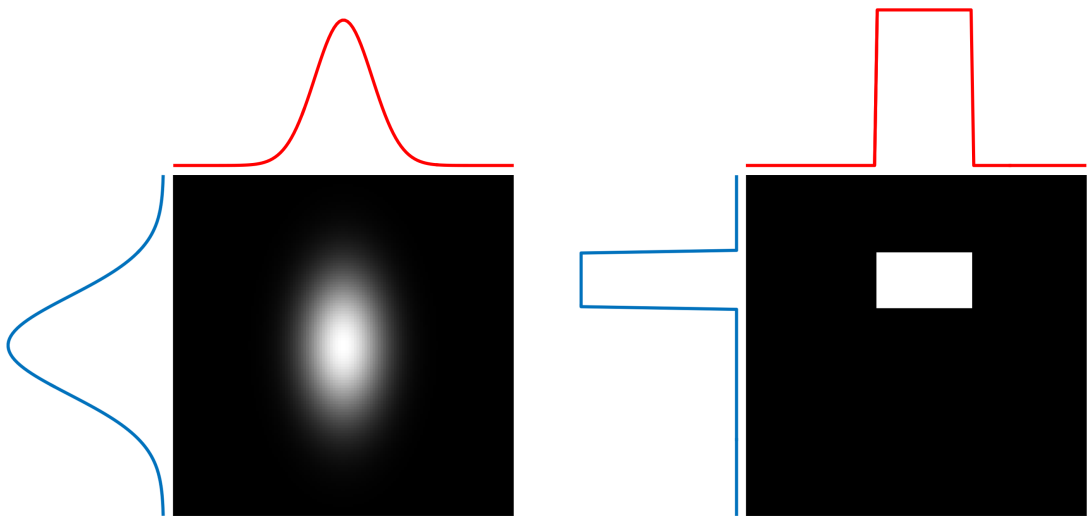
- Let us express a tensor as a **sum of *simpler* terms**

In a Nutshell

- Let us express a tensor as a **sum of *simpler* terms**
- Main ingredient: **separable** (rank-1) components

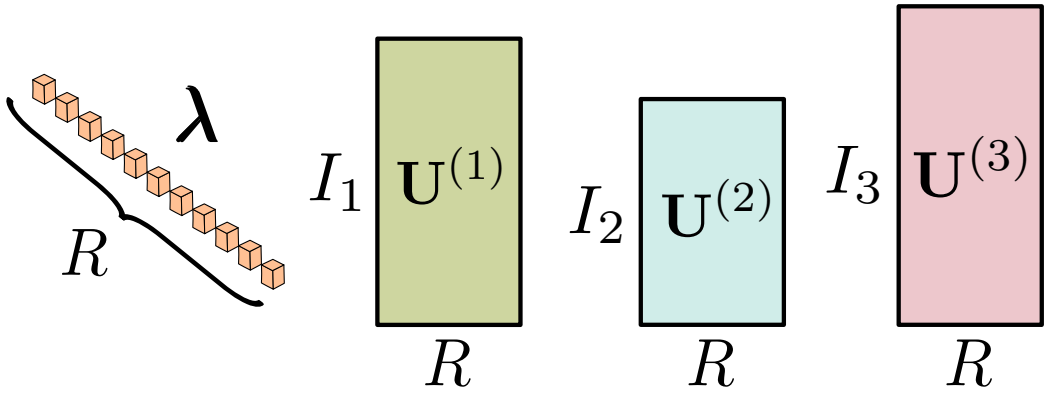
In a Nutshell

- Let us express a tensor as a **sum of *simpler* terms**
- Main ingredient: **separable** (rank-1) components
- Example in 2D (outer product $u \circ v$)



CP Decomposition

- One **factor matrix** per dimension
  - Coefficients in a **diagonal** form



- Formula:

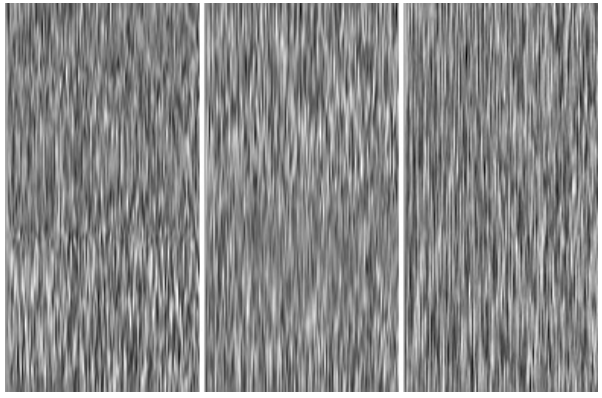
$$\mathscr{A} = \sum_{r=1}^{R} \lambda_r \cdot (u_r^{(1)} \circ_r^{(2)} \circ_r^{(3)})$$

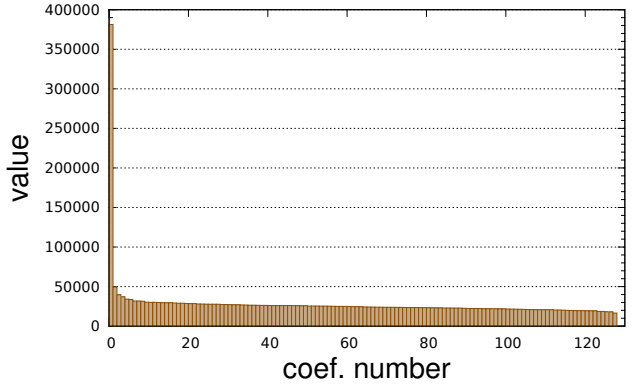
Example: CP

$\mathbf{U}^{(n)}$

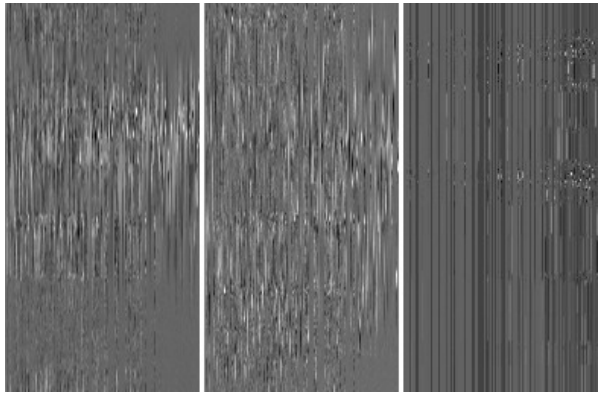
$\boldsymbol{\lambda}$

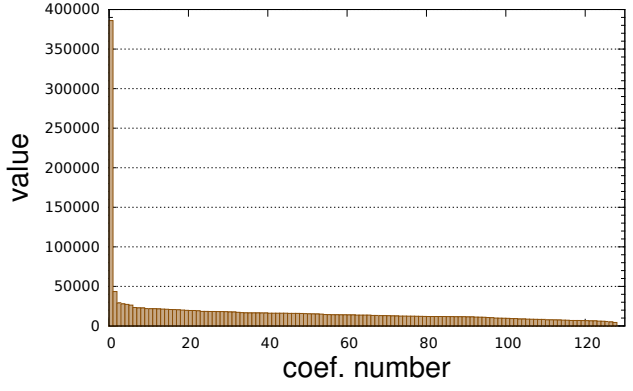
Lung





Video

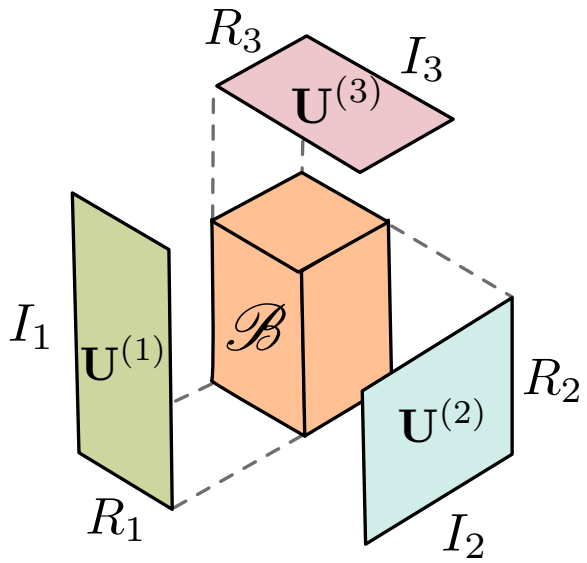




VISUALIZATIONAND MULTIMEDIALAB

Tucker Decomposition

- Generalization of CP
- We can enforce orthonormality
- Here, most space is **used by the core**



Tucker Decomposition

- Formula:

$$\mathscr{A} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \mathscr{B}_{r_1 r_2 r_3} \cdot \left(u_{r_1}^{(1)} \circ u_{r_2}^{(2)} \circ u_{r_3}^{(3)} \right)$$

Tucker Decomposition

- Formula:

$$\mathscr{A} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \mathscr{B}_{r_1 r_2 r_3} \cdot (u_{r_1}^{(1)} \circ u_{r_2}^{(2)} \circ u_{r_3}^{(3)})$$

- **Tensor-times-matrix** notation: $\mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$
  - *Multilinear projection*

Tucker Decomposition

- Formula:

$$\mathscr{A} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \mathscr{B}_{r_1 r_2 r_3} \cdot (u_{r_1}^{(1)} \circ u_{r_2}^{(2)} \circ u_{r_3}^{(3)})$$

- **Tensor-times-matrix** notation: $\mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$
  - *Multilinear projection*
- **Distance preservation**:

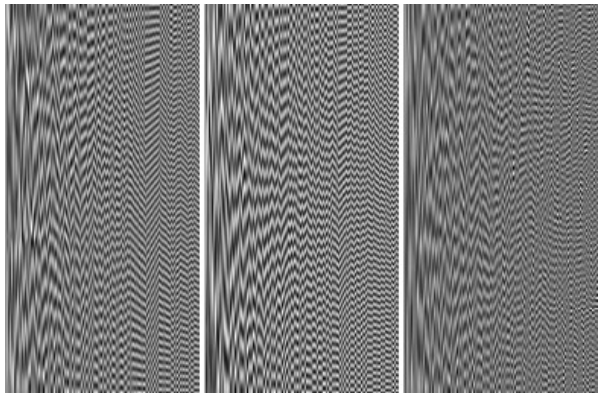
$$\left\{ \begin{array}{l} \mathscr{A}_1 \approx \mathscr{B}_1 \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \\ \mathscr{A}_2 \approx \mathscr{B}_2 \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \end{array} \right\} \Rightarrow \mathscr{B}_1 \approx \mathscr{B}_2$$

Example: Tucker

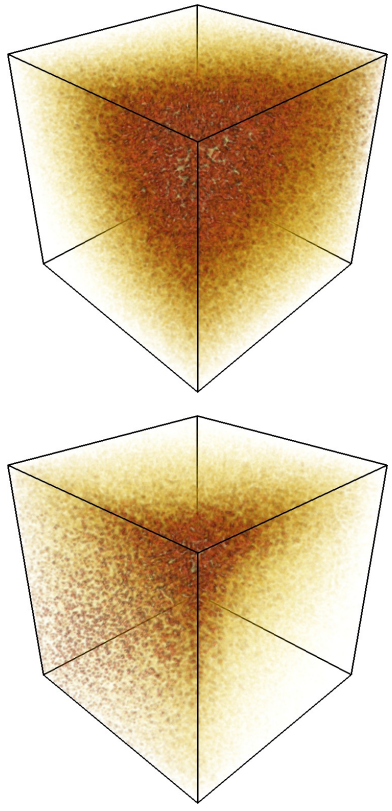
Lung

Video

$\mathbf{U}^{(n)}$



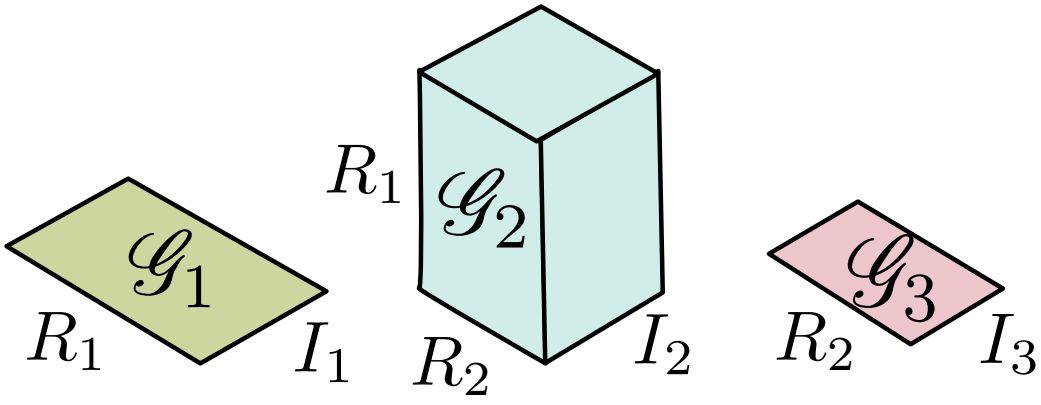
$\mathcal{B}$



VISUALIZATIONAND MULTIMEDIALAB

Tensor Train Decomposition

- One **core** per dimension

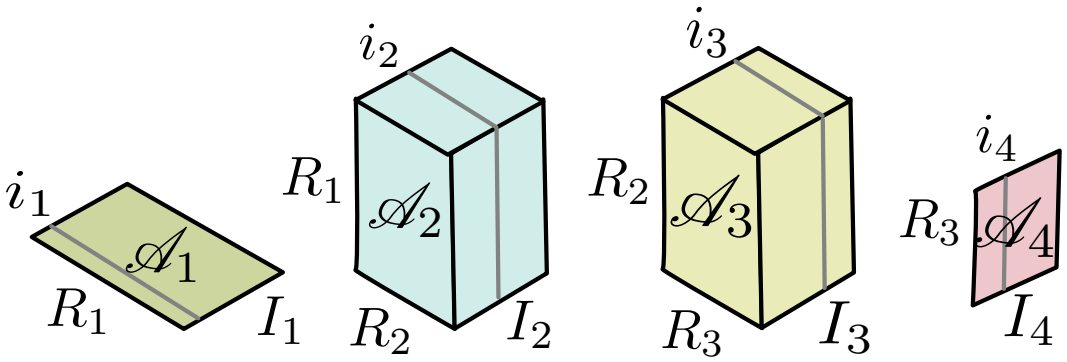


- Formula:

$$\mathscr{A} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \mathscr{G}_1(:, r_1) \circ \mathscr{G}_2(r_1, :, r_2) \circ \mathscr{G}_3(r_2, :)$$

Tensor Train Decomposition

- Reconstruction of 1 element (4D)



- Sequence of vector-matrix products

Decomposition Algorithms

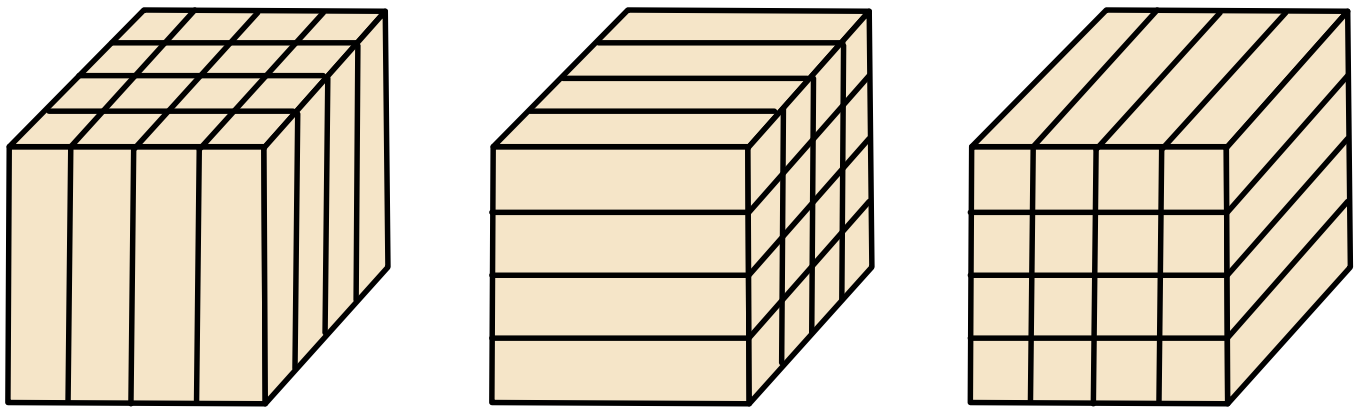
- CP: ill-posed $\rightarrow$ algorithms may only work well *in practice*

Decomposition Algorithms

- CP: ill-posed $\rightarrow$ algorithms may only work well *in practice*
- Tucker and TT:
  - **Algorithms more intuitive**
  - Stable
  - Let us look at Tucker's

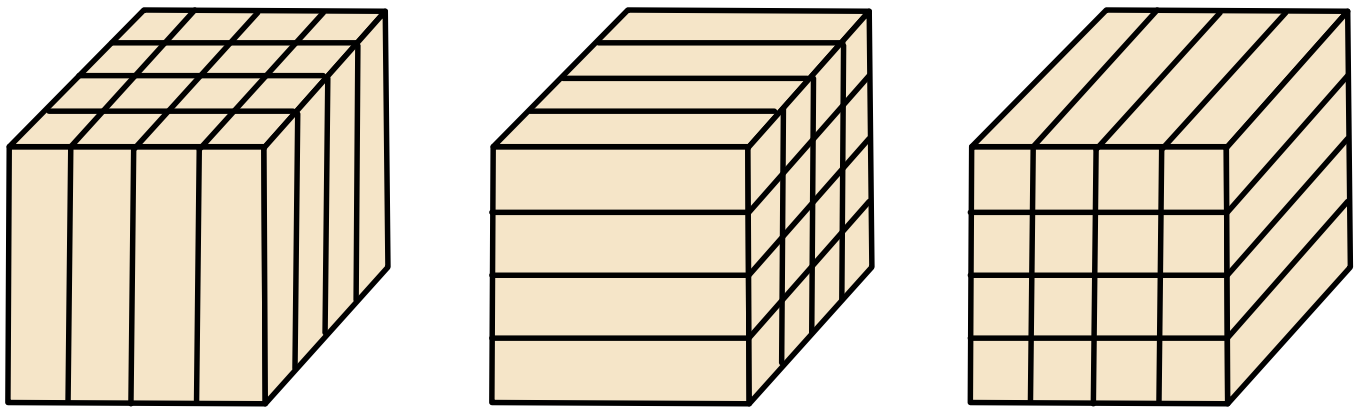
Fibers

- In 3D: **columns**, **rows**, **tubes**



Fibers

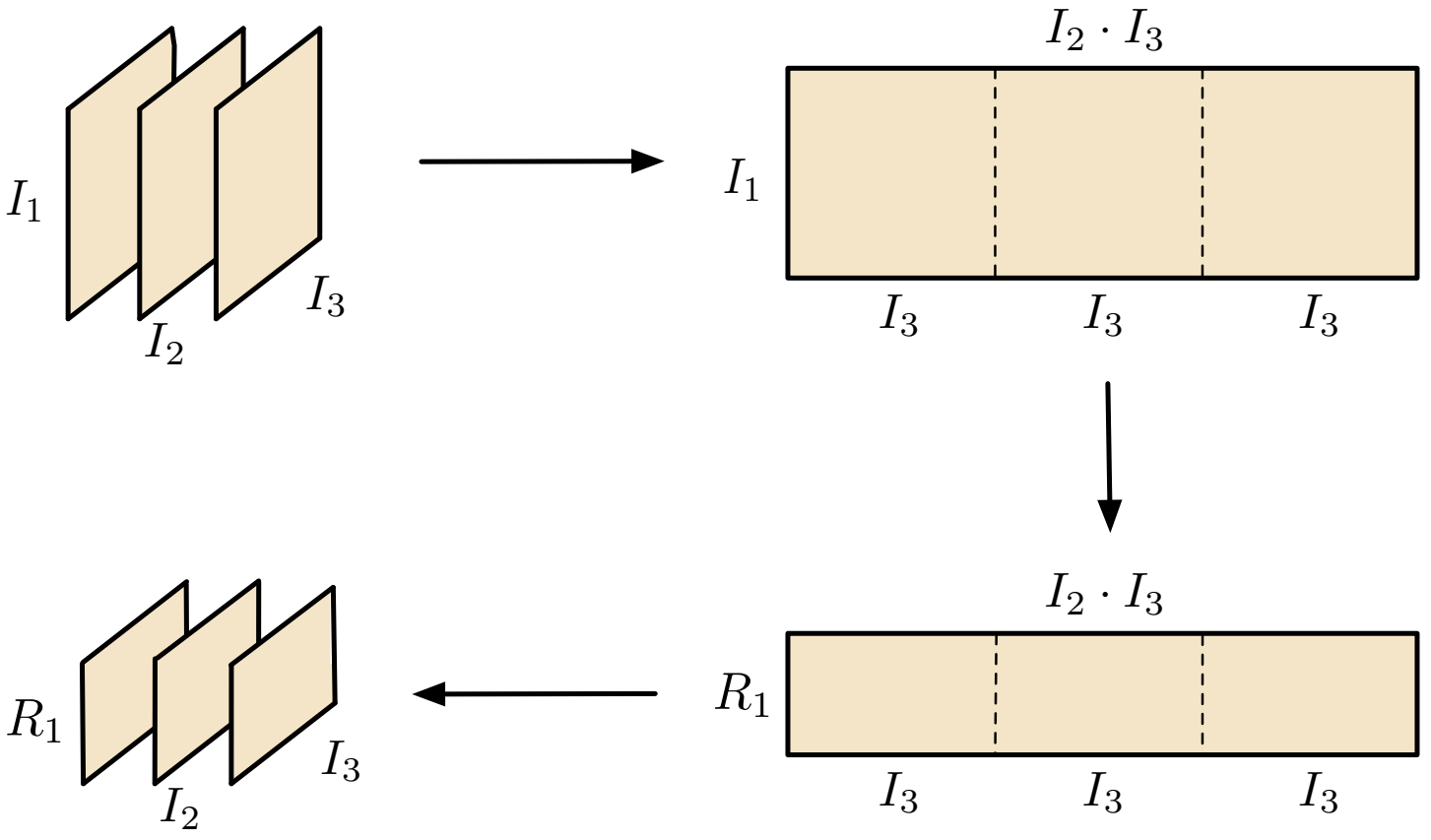
- In 3D: **columns**, **rows**, **tubes**



- Apply **principal component analysis** (PCA)

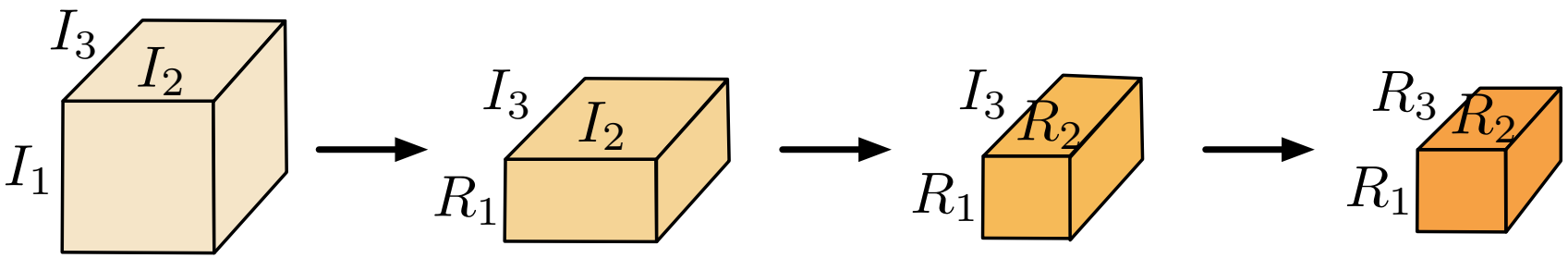
Unfoldings

- We do PCA **per fiber**



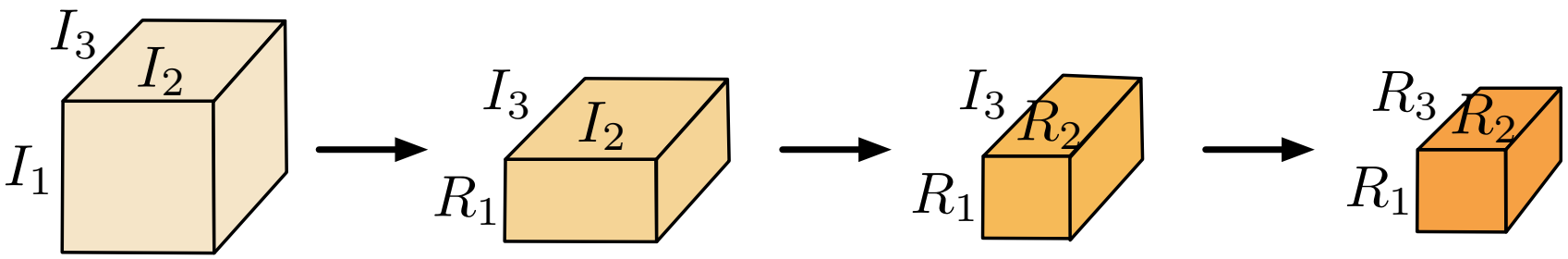
Multilinear Transforms

- **Orthogonal basis** of R vectors
- Fibers are compressed one dimension at a time



Multilinear Transforms

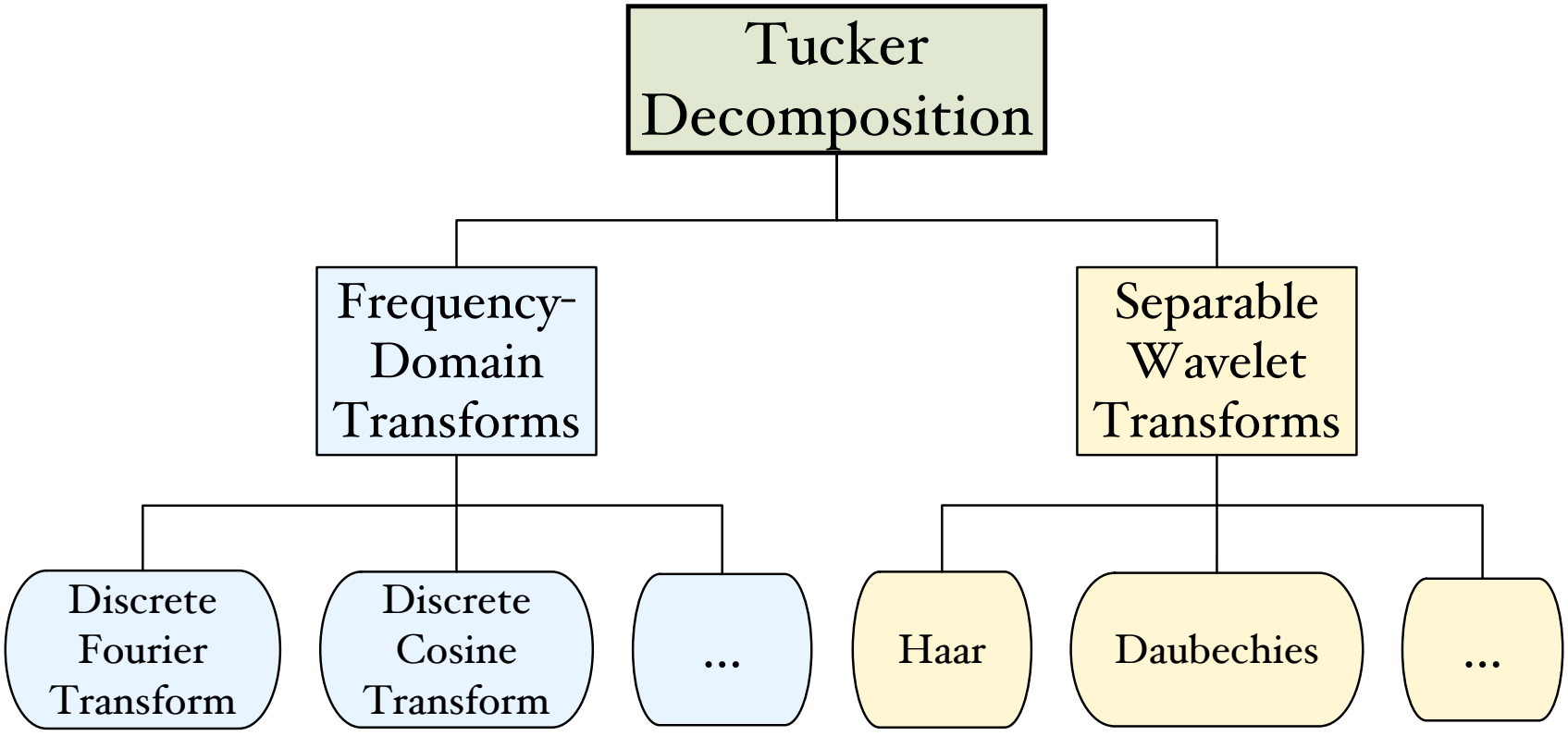
- **Orthogonal basis** of R vectors
- Fibers are compressed one dimension at a time



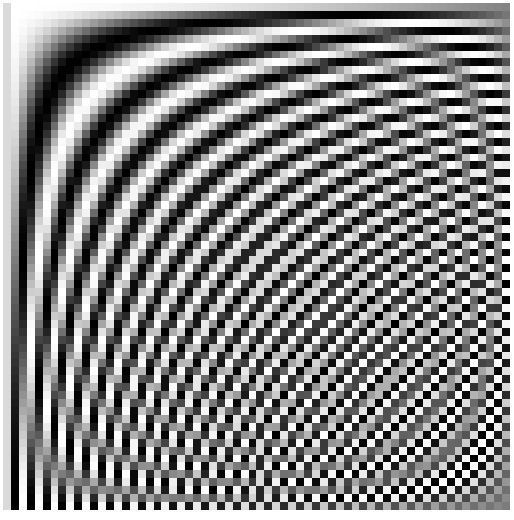
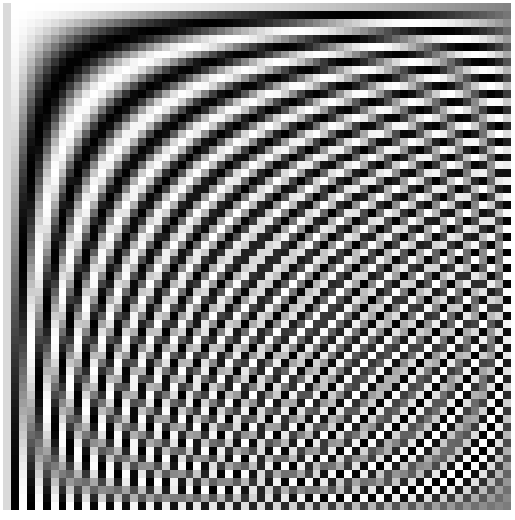
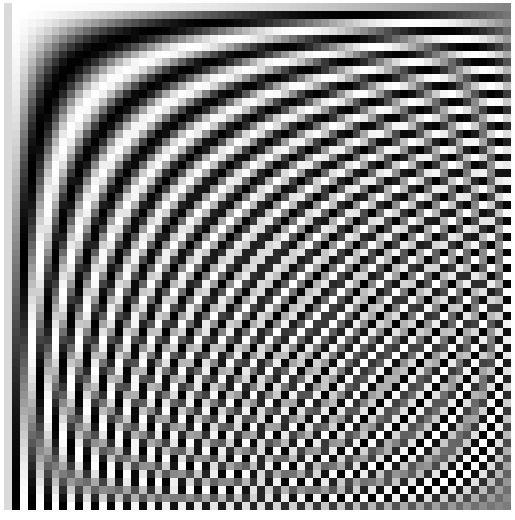
- These bases are precisely the matrices we want

$$\begin{cases} \mathscr{B} = \mathscr{A} \times_1 \mathbf{U}^{(1)^T} \times_2 \mathbf{U}^{(2)^T} \times_3 \mathbf{U}^{(3)^T} \\ \mathscr{A} \approx \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \end{cases}$$

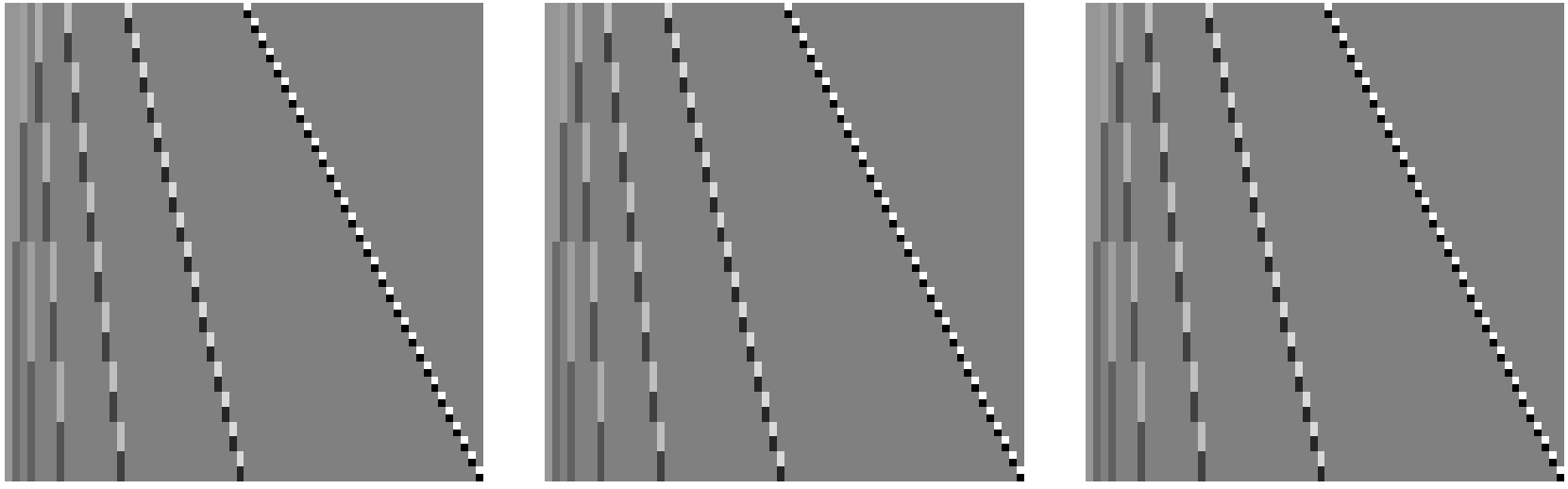
In Context...



Discrete Cosine Transform

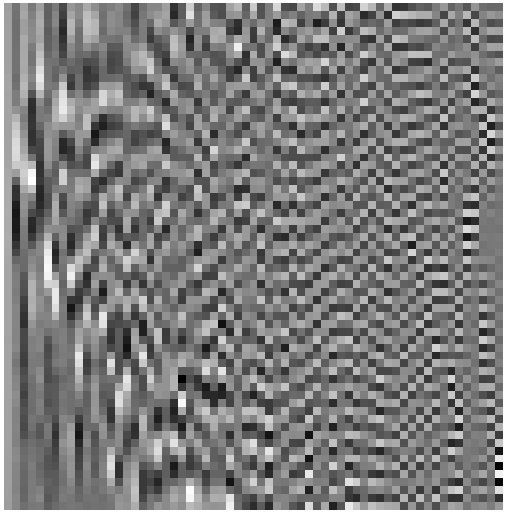
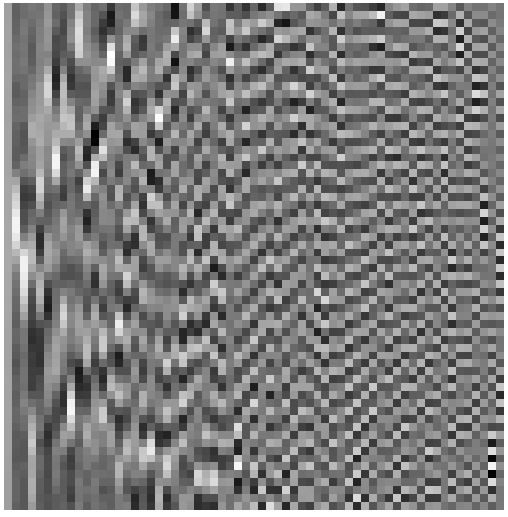
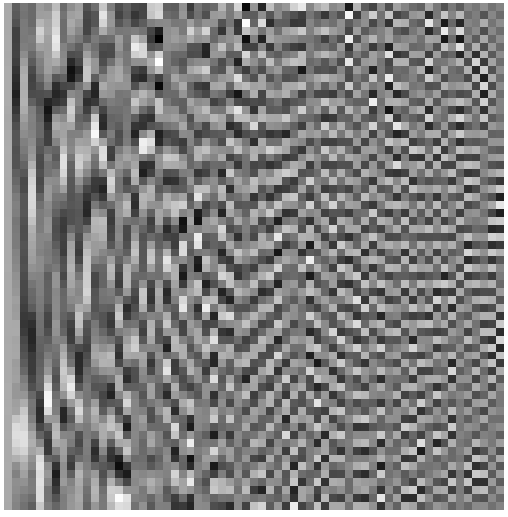




Haar Wavelets



Tucker Decomposition

- We leave it free $\rightarrow$ find **optimal** matrices

Advantages of Tensor Decomposition

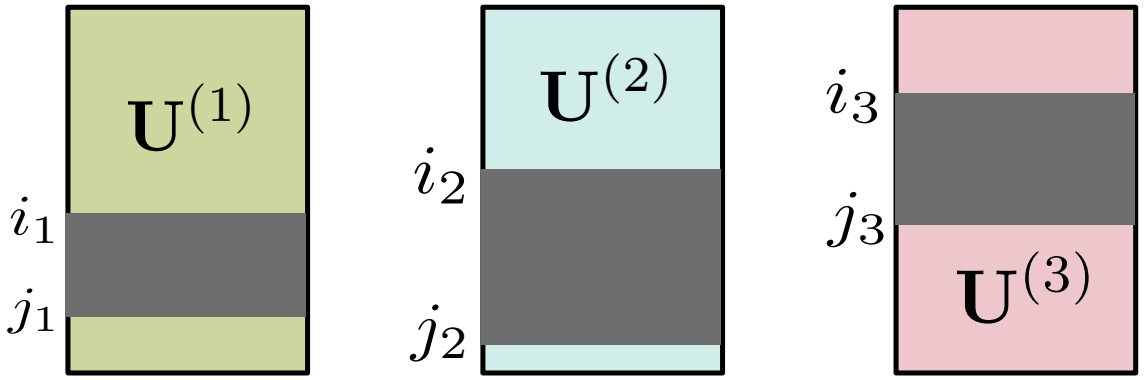
- Optimal bases $\rightarrow$ **competitive compression** rates
  - Good for out-of-core solutions
  - Often, the compressed data fits **entirely in the GPU**

Advantages of Tensor Decomposition

- Optimal bases $\rightarrow$ **competitive compression** rates
  - Good for out-of-core solutions
  - Often, the compressed data fits **entirely in the GPU**
- We can operate on the factor matrices:
  - Translation
  - Stretching
  - Projection
  - Convolution
  - Frequency-domain transforms
- Example: DCT on the factors + Reconstruction = Reconstruction + DCT on the result

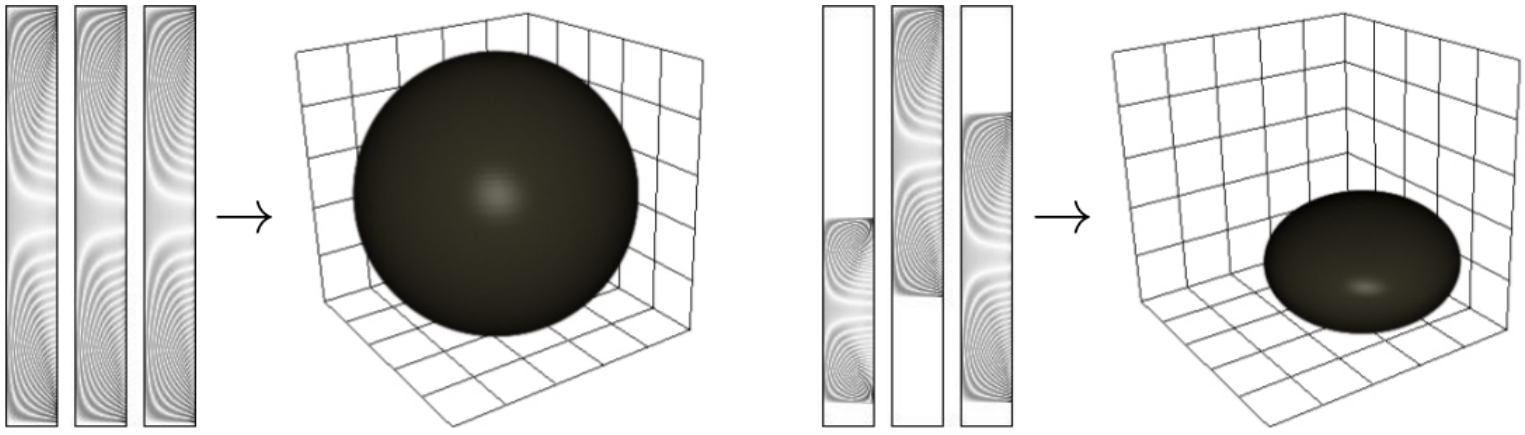
Spatial Selectivity

- To reconstruct the subregion $[i_1, j_1] \times [i_2, j_2] \times [i_3, j_3]$:



- Has equivalents in other tensor models

Example: Stretching + Translation



Costs

	Number of coefficients	$\mathbf{R}$ vs. $\mathbf{I}$
CP	$NIR + R$	$R \gg I$
Tucker	$R^N + NIR$	$R < I$
TT	$(N-2)IR^2 + 2IR$	Same order of magnitude

Costs

	Number of coefficients	$\mathbf{R}$ vs. $\mathbf{I}$
CP	$NIR + R$	$R \gg I$
Tucker	$R^N + NIR$	$R < I$
TT	$(N-2)IR^2 + 2IR$	Same order of magnitude

	Full reconstruction cost	One element
CP	$O(I^N R + NR)$	NR
Tucker	$O(I^N R + R^N)$	$O(R^N)$
TT	$O(I^N R + NR^2)$	$O(NR^2)$

The Potential of TT

- TT **is still unknown** to the graphics and visualization communities

The Potential of TT

- TT **is still unknown** to the graphics and visualization communities
- Generally best for $N \geq 4$

The Potential of TT

- TT **is still unknown** to the graphics and visualization communities
- Generally best for $N \geq 4$
- **Faster random access**

The Potential of TT

- TT **is still unknown** to the graphics and visualization communities
- Generally best for $N \geq 4$
- **Faster random access**
- Most compression approaches using Tucker with $N \geq 4$ can probably be greatly improved just by using TT!

Compression Quality

- The Tucker decomposition has competitive compression accuracy
  - Its bases (factor matrices) are learned (**data-dependent**)
  - For 3+ dimensions, they only take a **small fraction of the total memory**

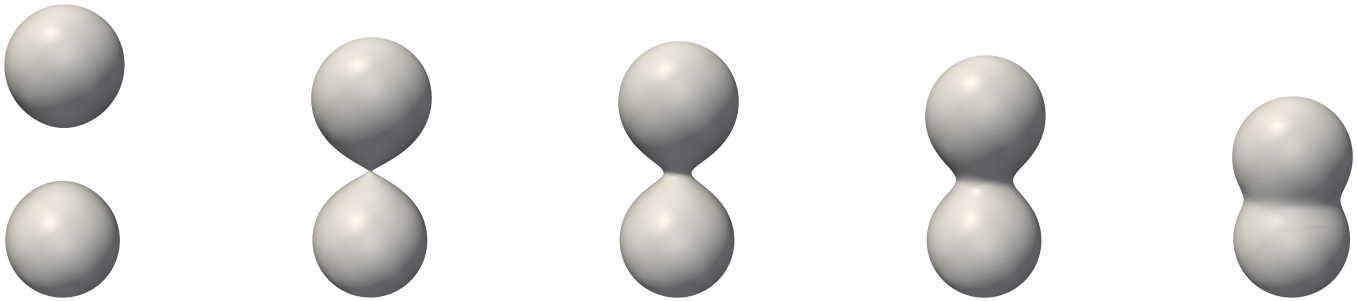
[Wu et al., 2008]
[Suter et al., 2011]
[Ballester-Ripoll and Pajarola, 2015]

Smooth Feature Compression

- At high compression rates, tensor decompositions are good at **preserving visual features**

Smooth Feature Compression

- At high compression rates, tensor decompositions are good at **preserving visual features**
- One way to see it: **isosurfaces**
  - For example, spheres are isosurfaces of multivariate Gaussians (rank-1)



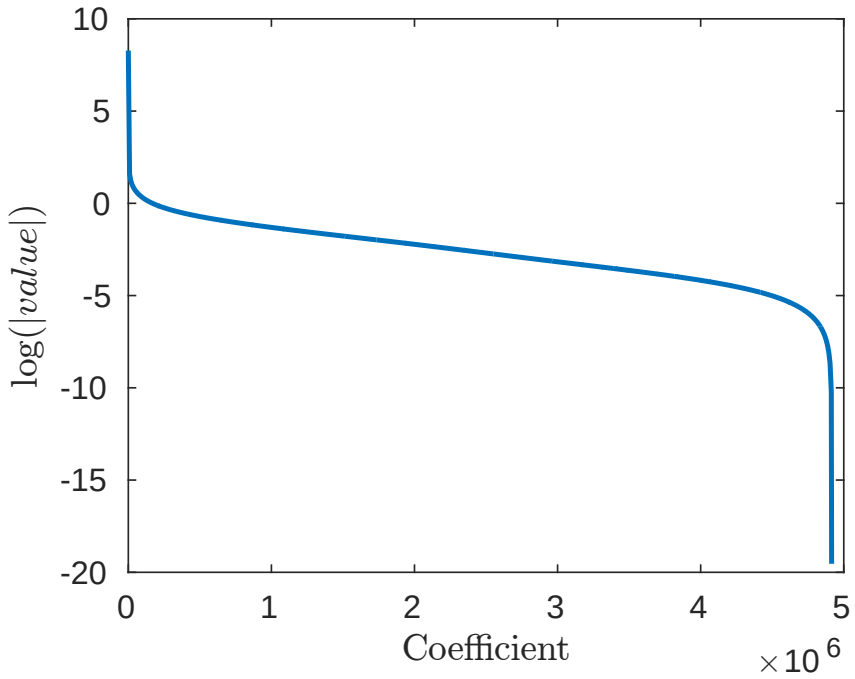
Smooth isosurfaces from a rank-1 function

Tucker Compression

- Quantization [SMP13]
- Truncation [SMP13,BP15,BSP15]
- Thresholding [BP15]

Quantization

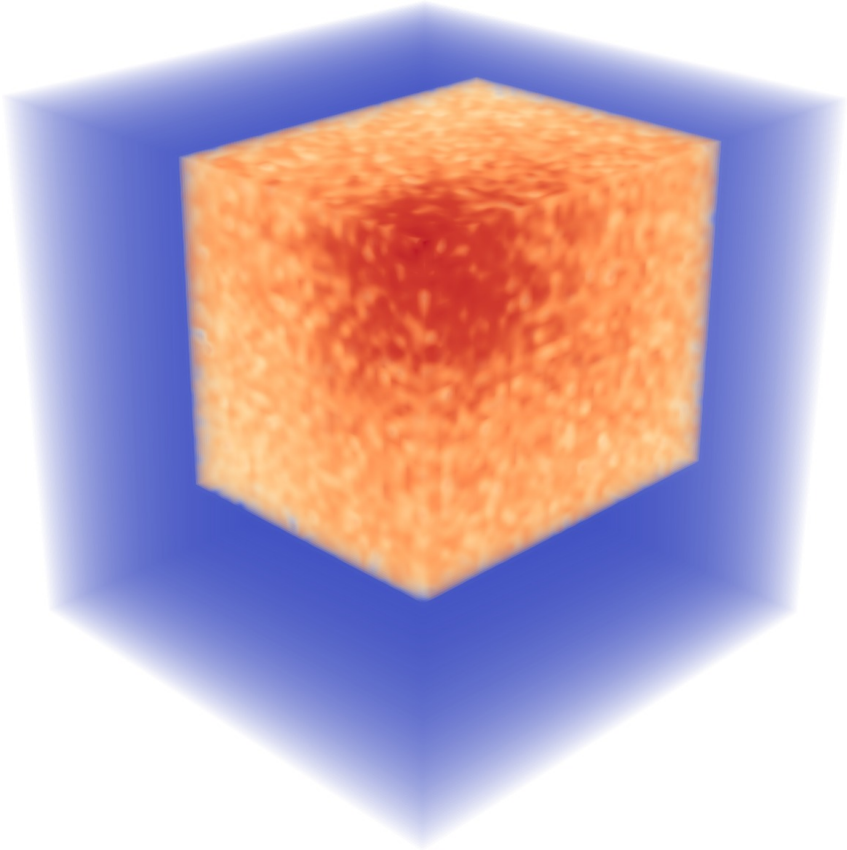
- Core coefficients are roughly **logarithmic**:



- **Logarithmic quantization** works best
  - E.g. 8 bits + 1 sign bit

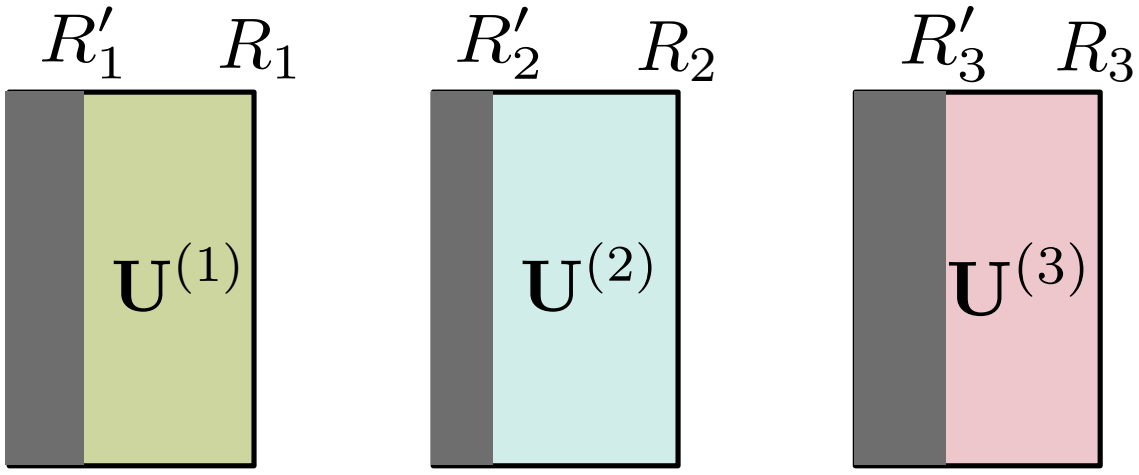
Tucker Truncation

- During visualization, **reduce** ranks as needed
- Very fast to apply



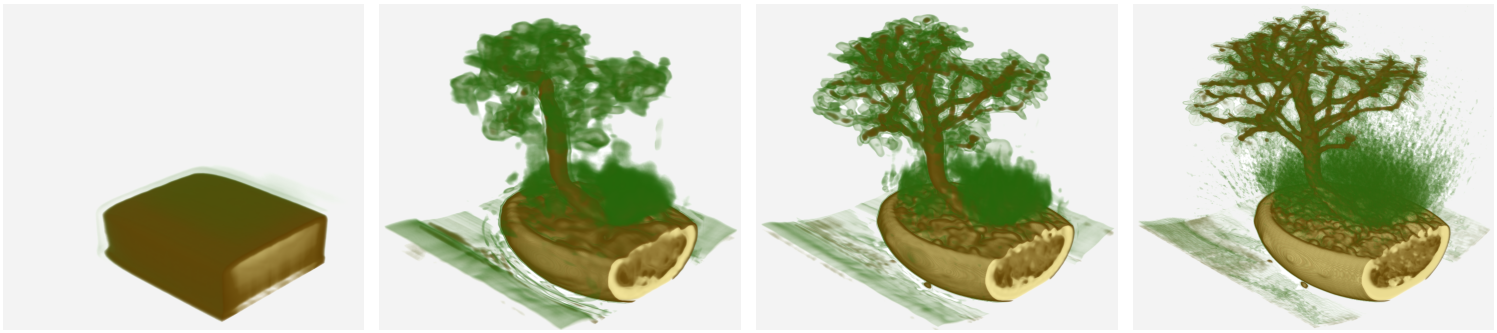
Tucker Truncation

- Rank selection for interactive level-of-detail [Suter et al., 2013]: Tucker core from $\mathbb{R}^{R_1 \times R_2 \times R_3}$ to $\mathbb{R}^{R'_1 \times R'_2 \times R'_3}$

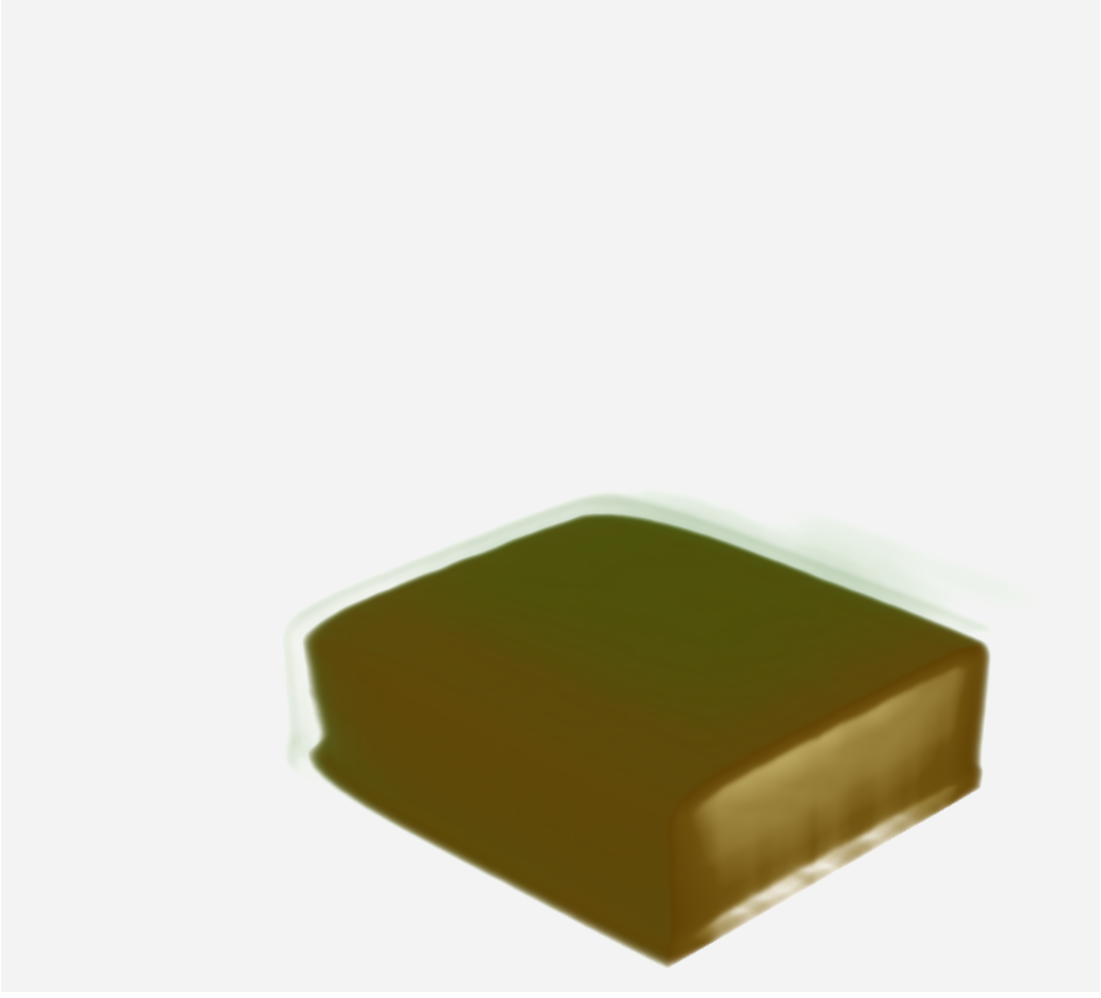


Tucker Truncation

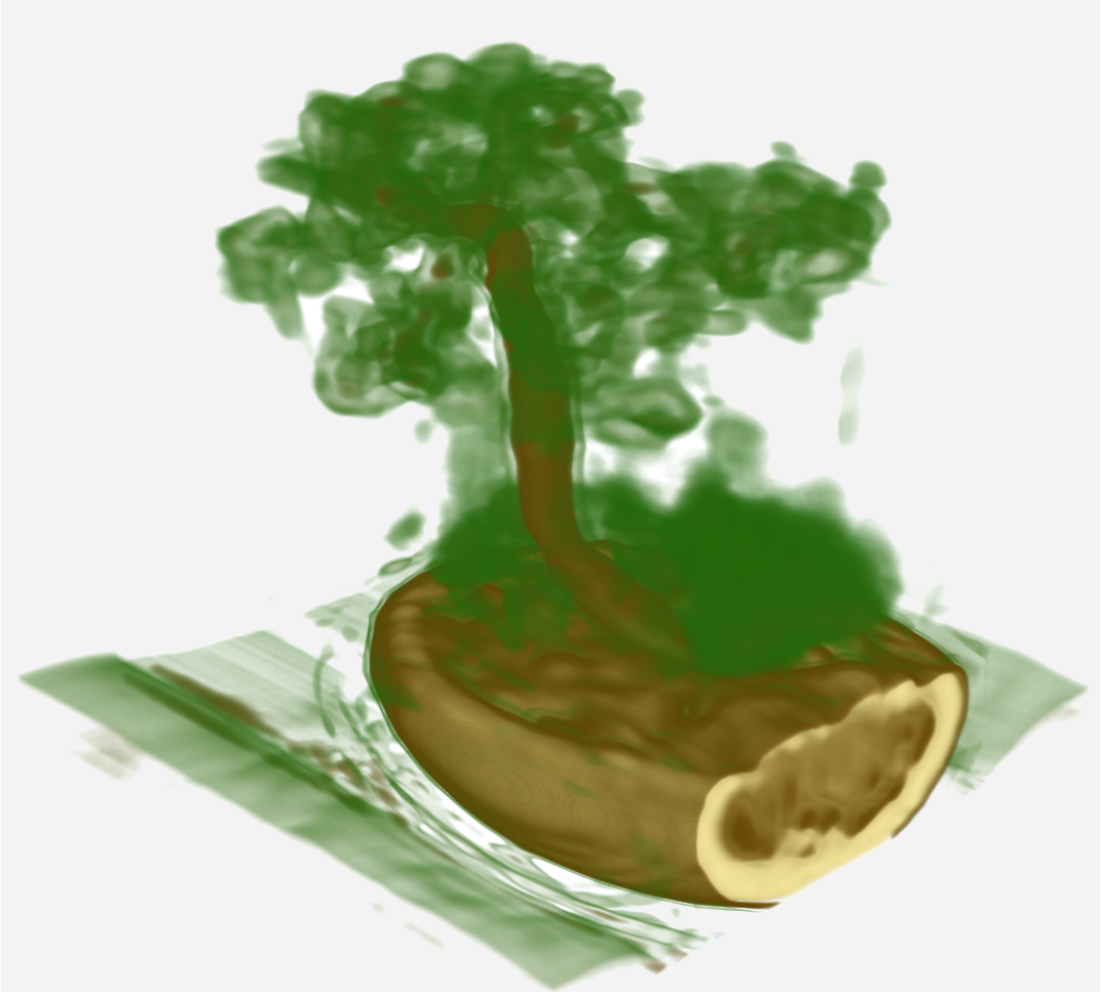
- **Different ranks select different features at different scales**
  - Example: bonsai (256^3), from 1 to 256 Tucker ranks



Tucker Truncation

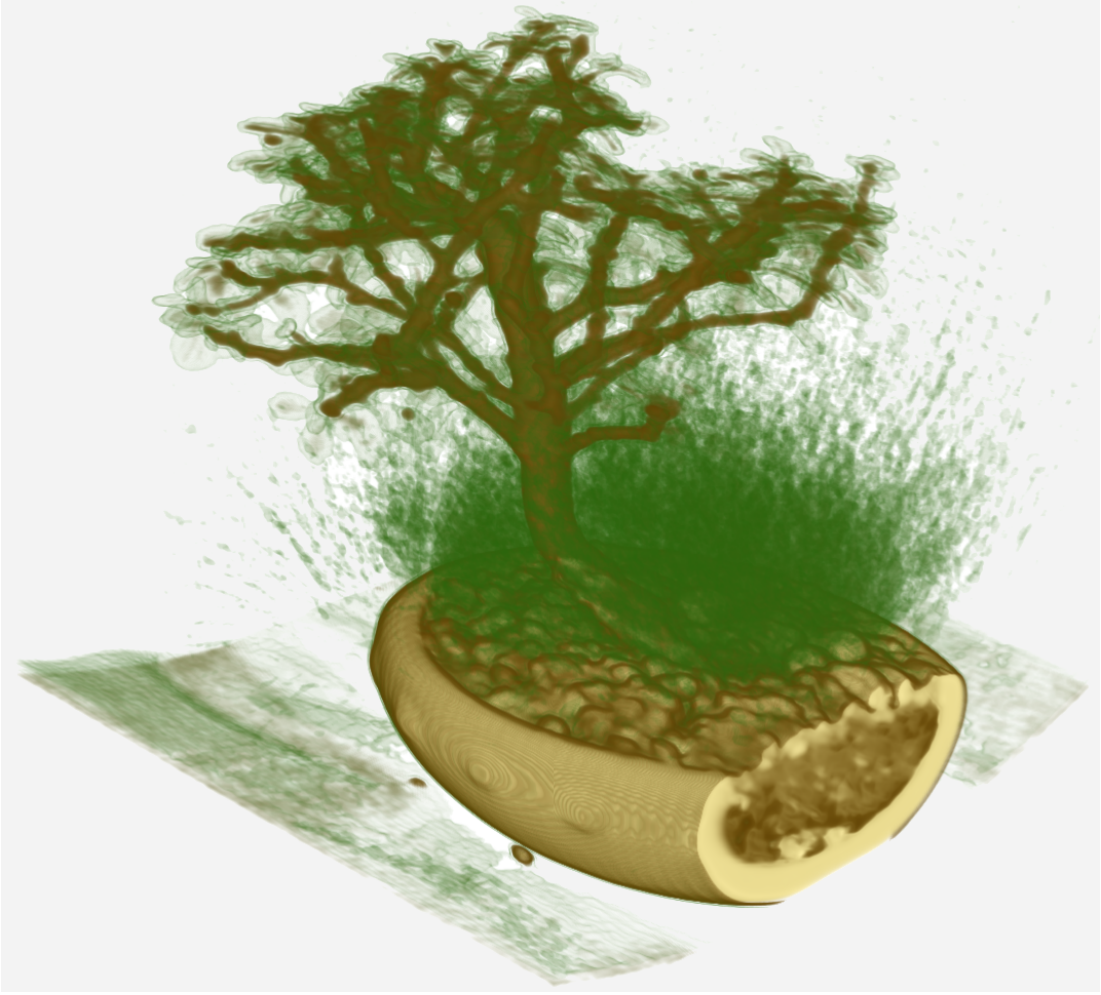


Tucker Truncation



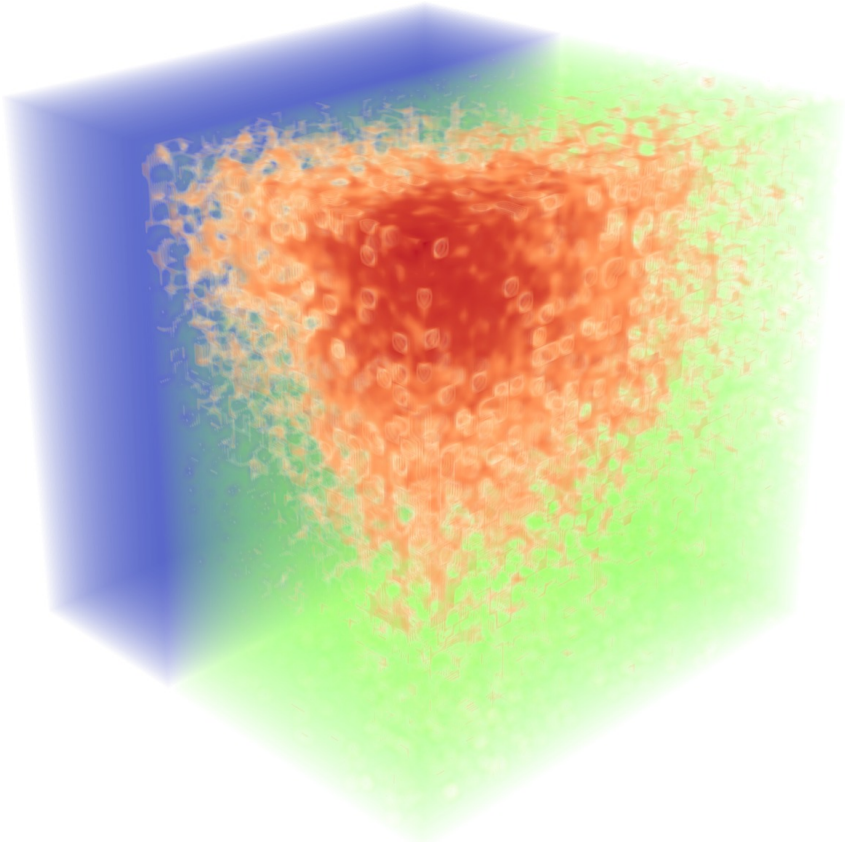
Tucker Truncation

Tucker Truncation



Tucker Thresholding

- Make small elements 0
- Run-length + Entropy encoding

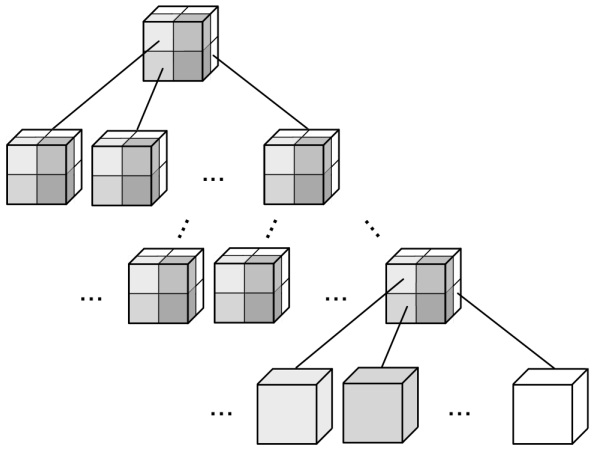


Multiresolution Tucker Compression

- 3D Tucker rank-R reconstruction cost: $I^3R + I^2R^2 + IR^3$
- $O(R)$ operations per voxel result

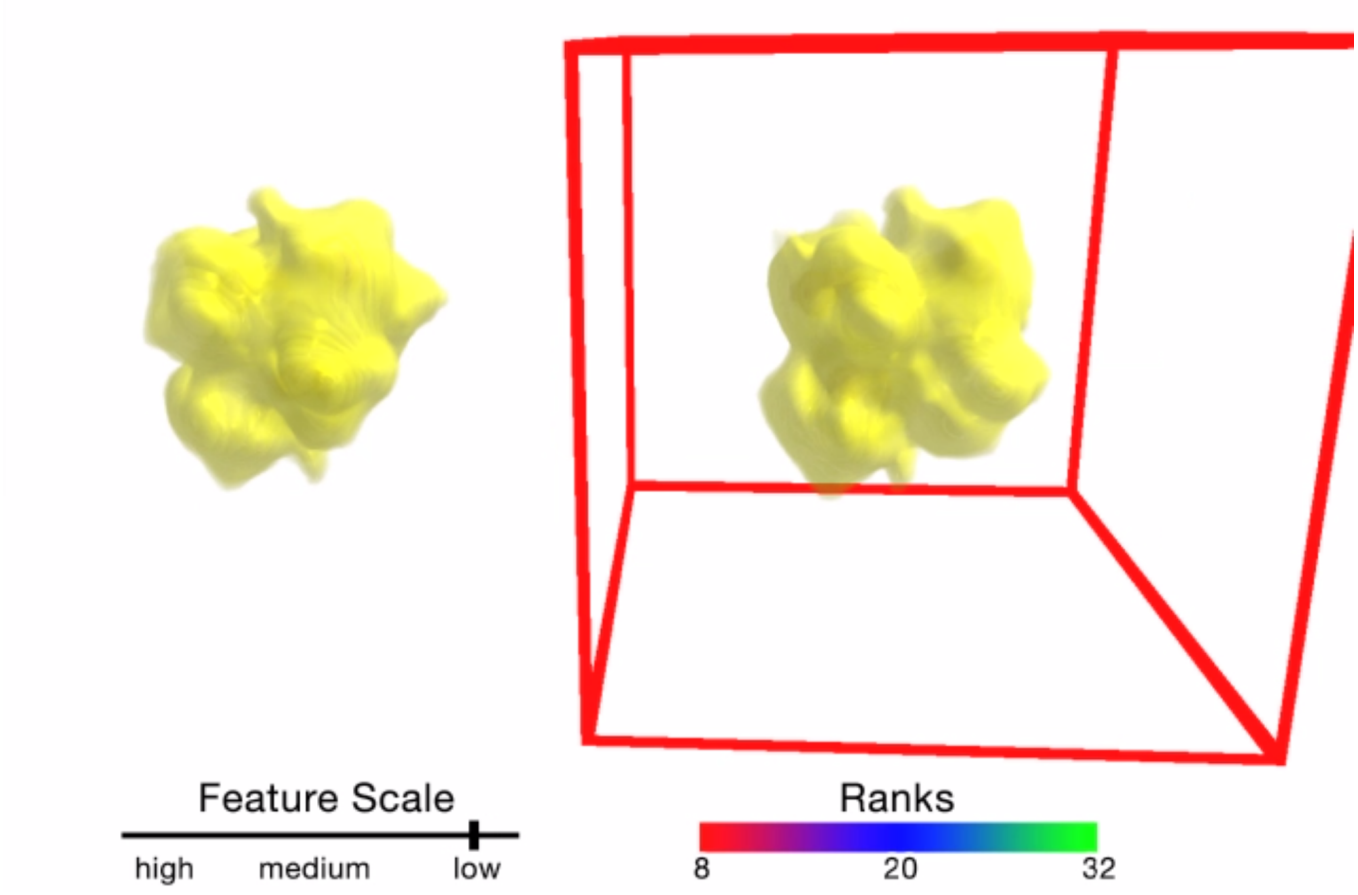
Multiresolution Tucker Compression

- 3D Tucker rank-R reconstruction cost: $I^3R + I^2R^2 + IR^3$
- $O(R)$ operations per voxel result
- Octree: partition I^3 volume into bricks of size B^3 [Suter et al., 2013]:

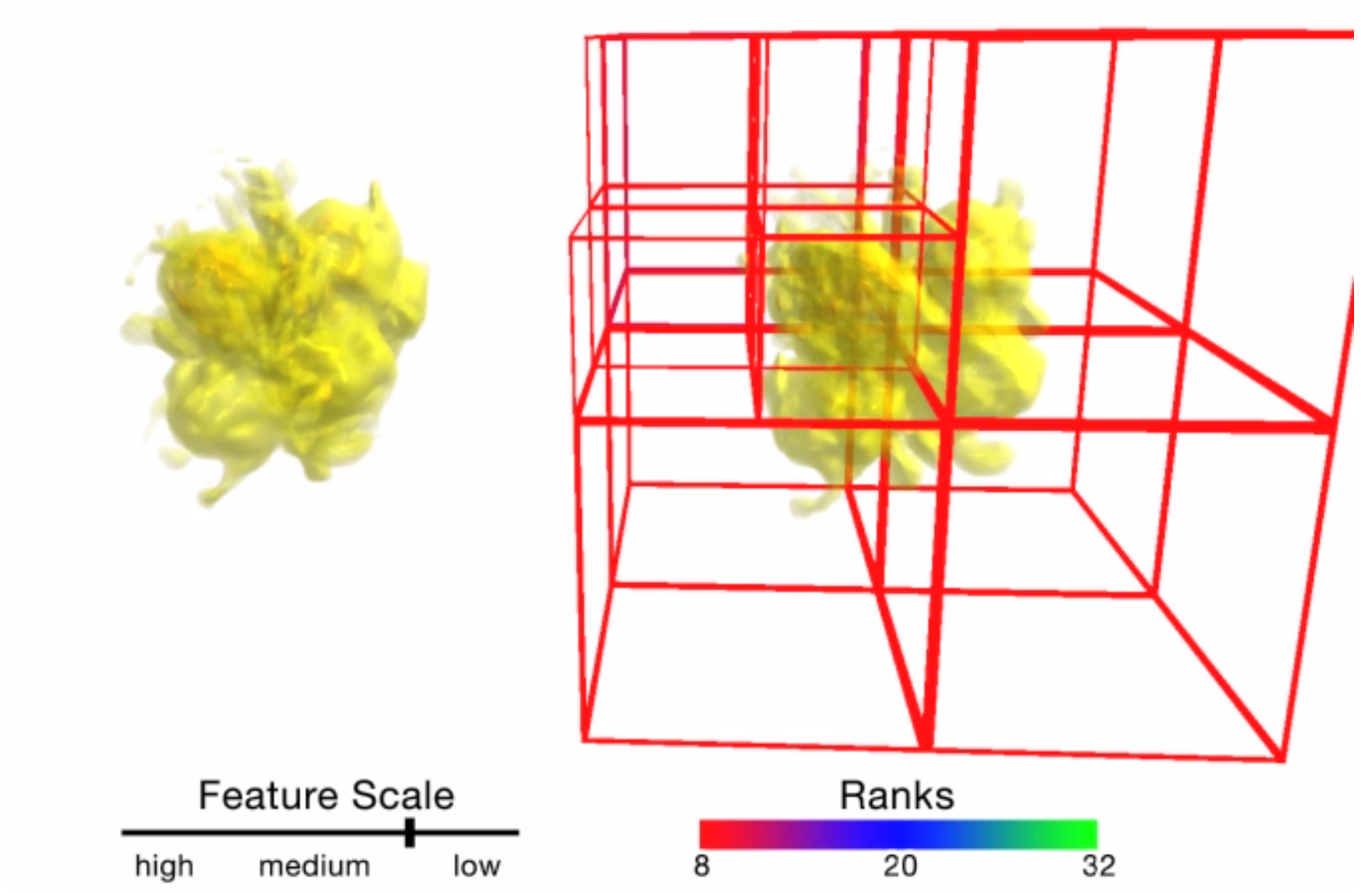


- One Tucker core per brick: **speeds up** reconstruction by a factor $\approx I/B$

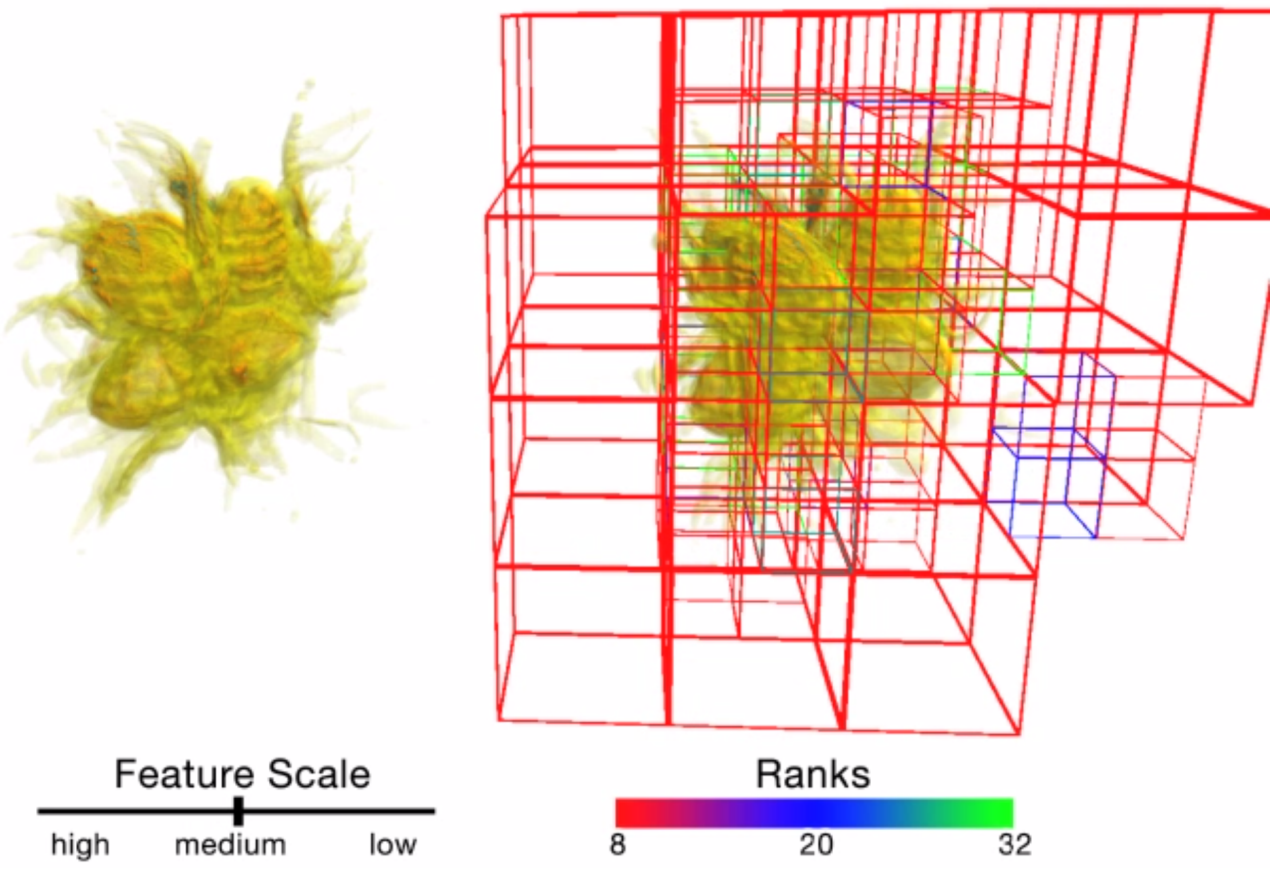
Multiresolution Tucker Visualization



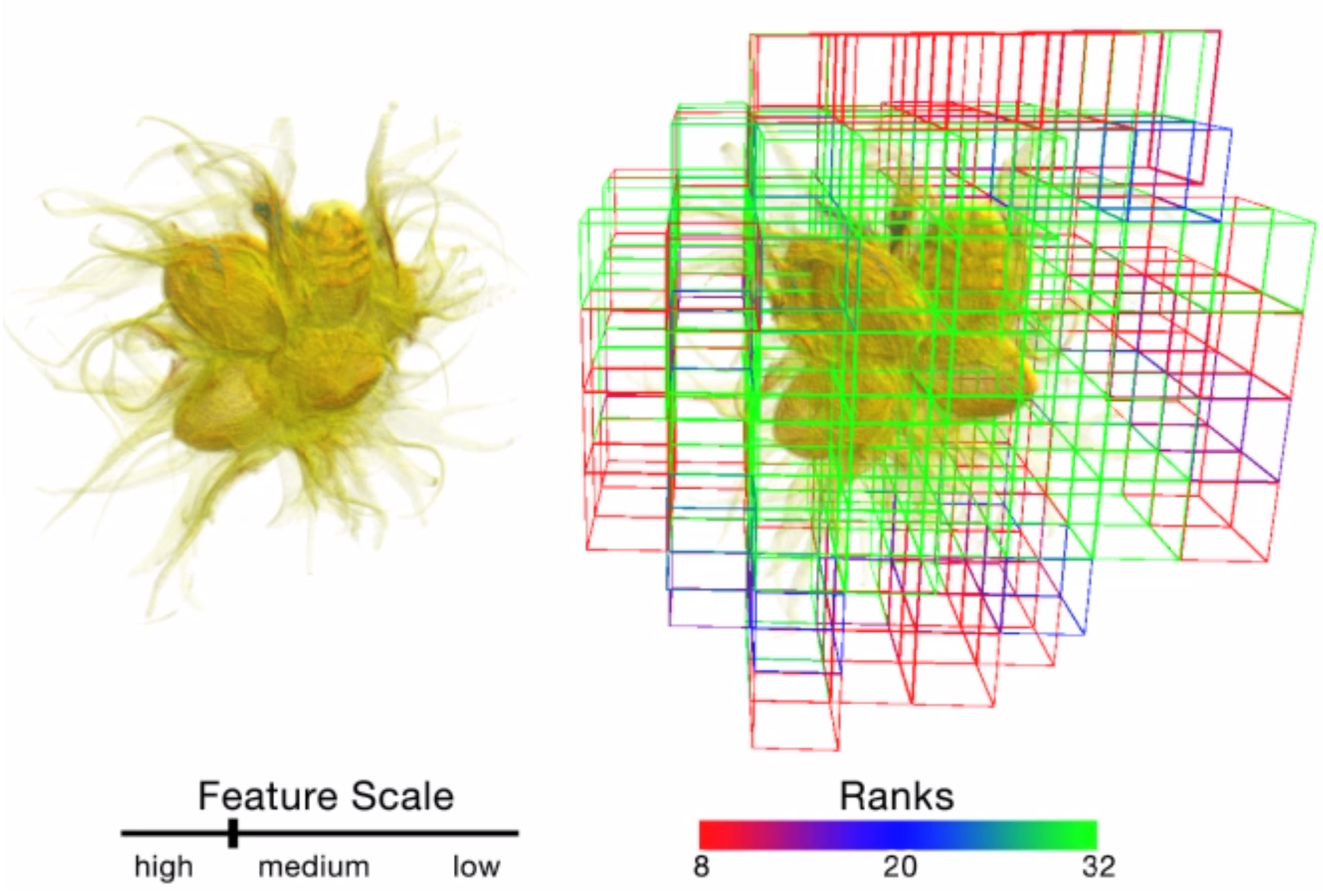
Multiresolution Tucker Visualization



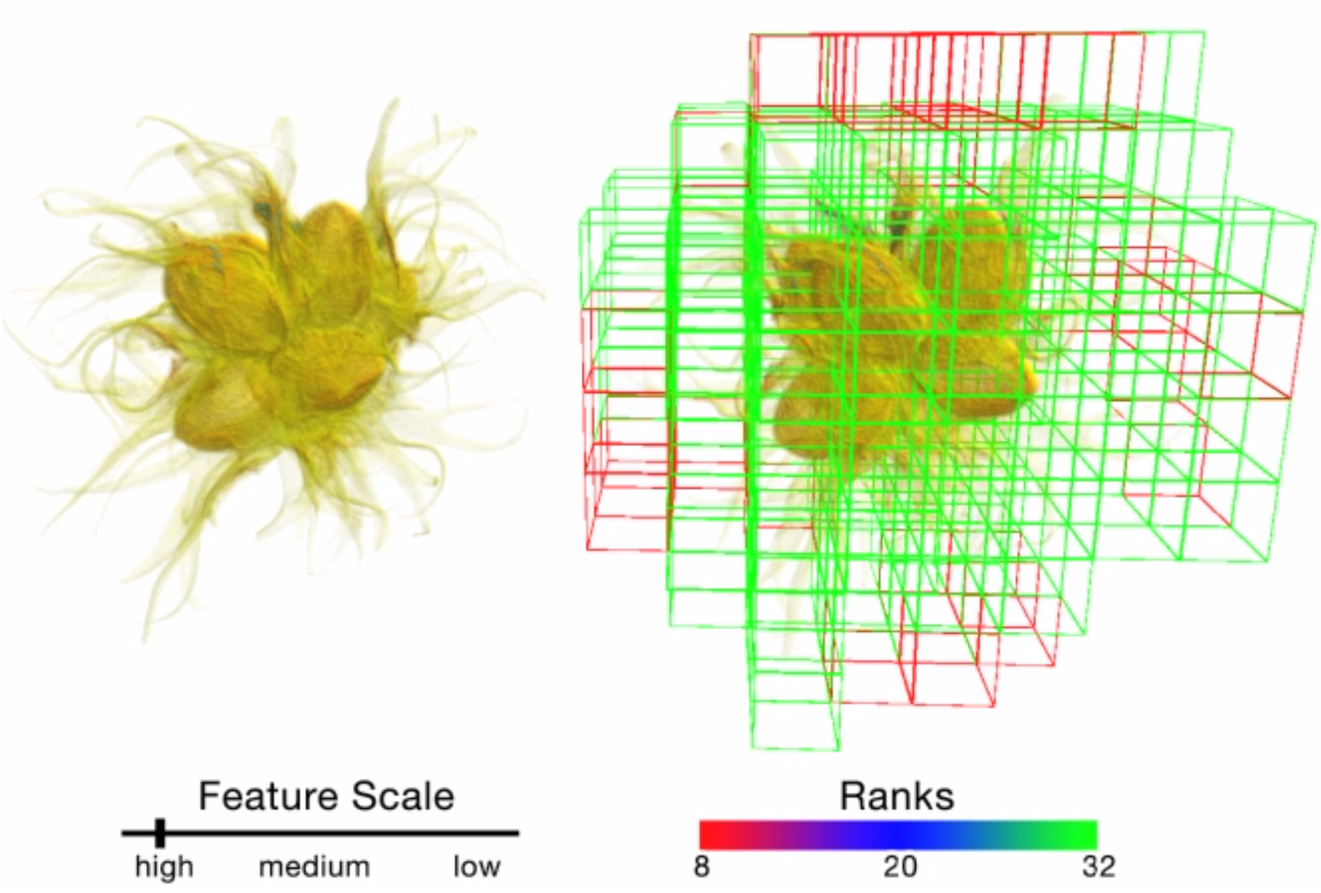
Multiresolution Tucker Visualization



Multiresolution Tucker Visualization



Multiresolution Tucker Visualization



Multiresolution Filtering

- Problem: filter over different resolution levels
  - Lower resolution requires downsampling the filter kernel

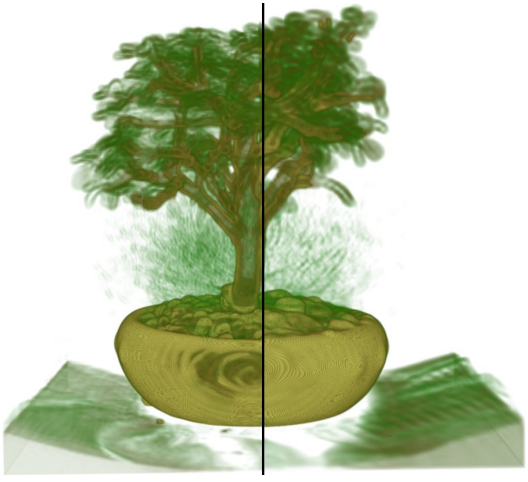
Multiresolution Filtering

- Problem: filter over different resolution levels
  - Lower resolution requires downsampling the filter kernel
- Example: 3D Sobel operator for edge detection
  - Size $3 \times 3 \times 3$, **cannot be downsampled**!

$$h_z = \begin{bmatrix} -1 & -2 & -1 \\ -2 & -4 & -2 \\ -1 & -2 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} +1 & +2 & +1 \\ +2 & +4 & +2 \\ +1 & +2 & +1 \end{bmatrix}$$

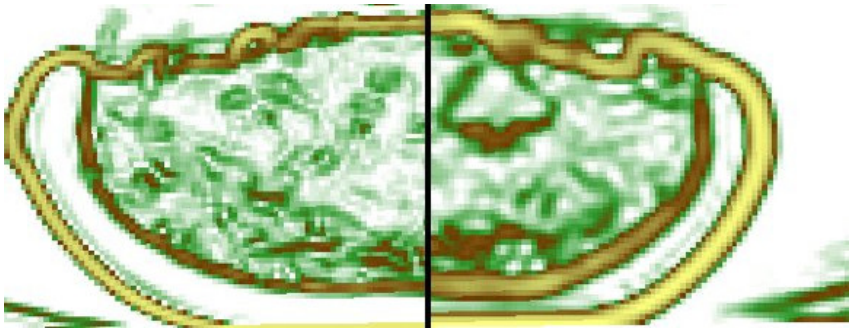
Multiresolution Filtering

- Problem: filter over different resolution levels
  - Lower resolution requires downsampling the filter kernel
- Example: 3D Sobel operator for edge detection
  - Size $3 \times 3 \times 3$, cannot be downsampled!
- With naive filtering, **artifacts appear**



Multiresolution Filtering

- Problem: filter over different resolution levels
  - Lower resolution requires downsampling the filter kernel
- Example: 3D Sobel operator for edge detection
  - Size $3 \times 3 \times 3$, cannot be downsampled!
- With naive filtering, **artifacts appear**



Tensor Convolution

- Separable linear transforms can be computed on the corresponding factors
  - Convolution theorem. Cosine and Fourier transforms, separable wavelets, etc.

Tensor Convolution

- Separable linear transforms can be computed on the corresponding factors
  - Convolution theorem. Cosine and Fourier transforms, separable wavelets, etc.

- **We can filter via the Tucker factors**
  - Long-known property, already in 2D for the SVD
  - Works for other tensor models

Example: Sobel

- The 3D Sobel operator is a combination of 3 rank-1 filters:

$$\widehat{\mathscr{A}}(\boldsymbol{i}) = \sqrt{(\mathscr{A} * h_x)(\boldsymbol{i})^2 + (\mathscr{A} * h_y)(\boldsymbol{i})^2 + (\mathscr{A} * h_z)(\boldsymbol{i})^2}$$

with

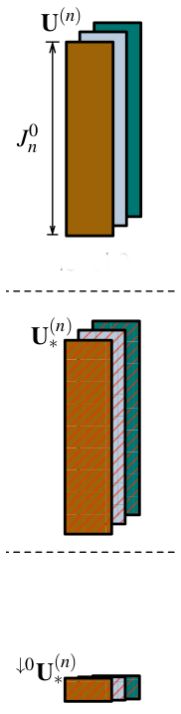
$$\begin{cases} h_x = \boldsymbol{u} \circ \boldsymbol{v} \circ \boldsymbol{v} \\ h_y = \boldsymbol{v} \circ \boldsymbol{u} \circ \boldsymbol{v} \\ h_z = \boldsymbol{v} \circ \boldsymbol{v} \circ \boldsymbol{u} \end{cases}$$

and

$$\boldsymbol{u} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \boldsymbol{v} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

Filter + Decompression

- Decompressing a filtered brick in low resolution:
  - We first convolve the tall global matrices
  - Then we downscale them as needed
  - After reconstruction, we get the desired resolution



Filter + Decompression

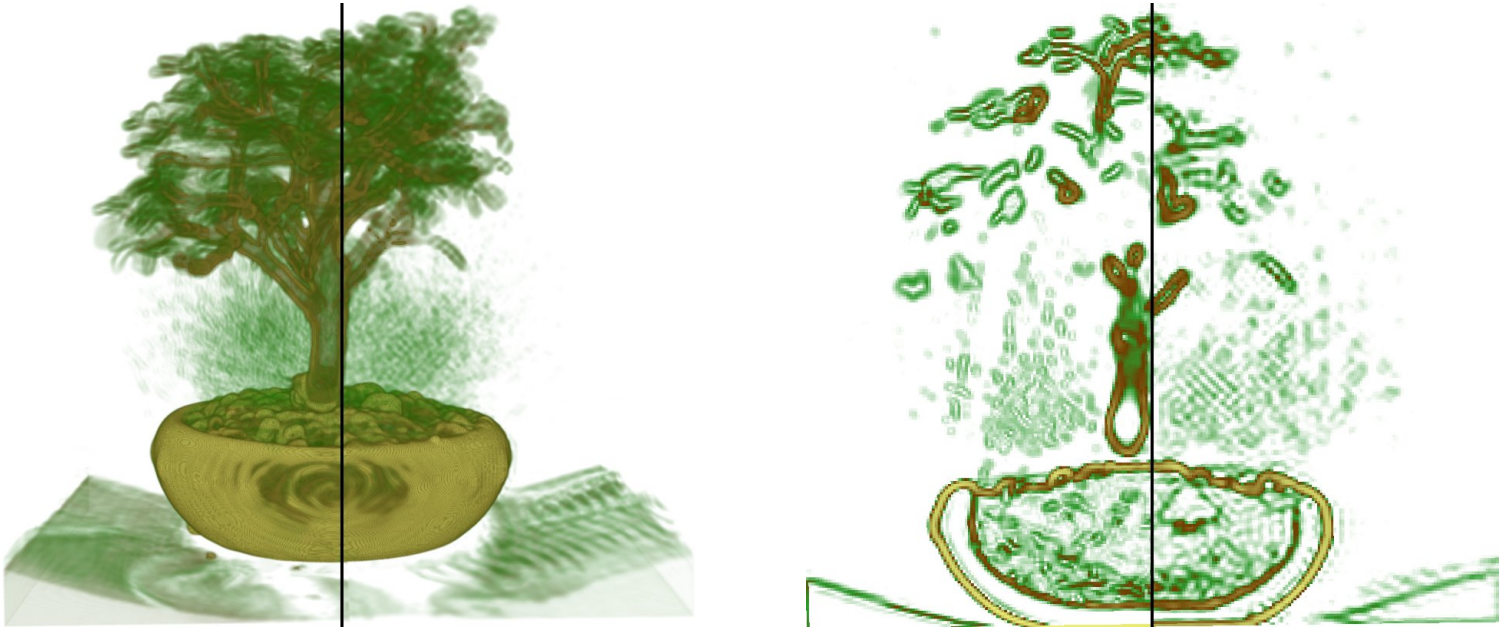
- Filtering cost: $O(NKR)$. **Negligible** compared to reconstruction ($O(I^N R)$)

Filter + Decompression

- Filtering cost: $O(NKR)$. **Negligible** compared to reconstruction ($O(I^N R)$)
- GPU implementation:
  - C++ CUDA filtering and decompression
  - For a 2048^3 volume: under 1 ms for a rank-2 filter (difference of Gaussians) of size 20

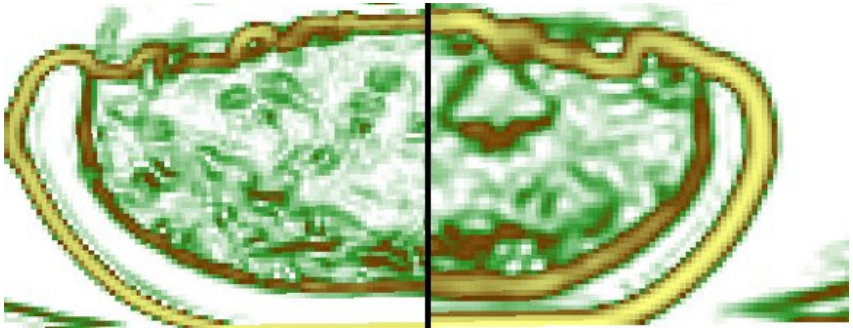
Result

- Smoother response across different levels of detail [Ballester-Ripoll et al., 2016]

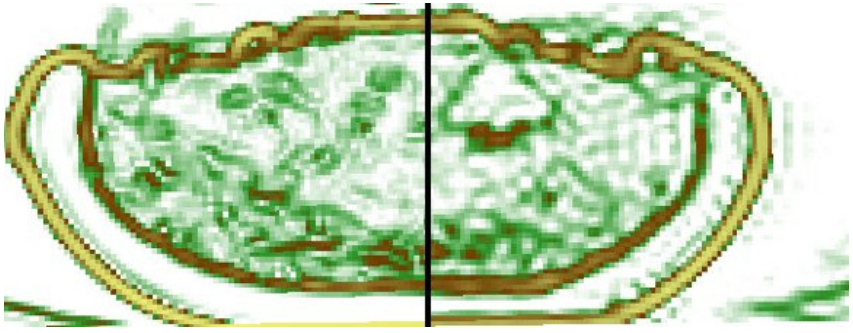


Sobel on two resolution levels

Result (Close-up)



Naive filtering



Multiresolution Tucker

To Sum Up...

- Tensor algorithms in real-time applications need very efficient reconstruction
  - Achieved via parallelization, data partitioning, etc.

To Sum Up...

- Tensor algorithms in real-time applications need very efficient reconstruction
  - Achieved via parallelization, data partitioning, etc.
- Operating in the tensor bases is a powerful trick
  - Usually very fast, flexible, and matching our needs

To Sum Up...

- Tensor algorithms in real-time applications need very efficient reconstruction
  - Achieved via parallelization, data partitioning, etc.
- Operating in the tensor bases is a powerful trick
  - Usually very fast, flexible, and matching our needs
- Tensor decompositions are a promising framework for interactively visualizing data sets
  - Especially, with increasing size and number of dimensions

Section 3

Tensor Sampling and Completion

Motivation

Many methods revolve around **missing values**. Two types of situations:

Motivation

Many methods revolve around **missing values**. Two types of situations:

- Known samples are given
  - Some signal regions were not captured
  - Why? Sensor was damaged, or limited observations
  - Keywords: **tensor recovery**, **texture synthesis**

Motivation

Many methods revolve around **missing values**. Two types of situations:

- Known samples are given
  - Some signal regions were not captured
  - Why? Sensor was damaged, or limited observations
  - Keywords: **tensor recovery**, **texture synthesis**

- We can choose where to sample
  - The signal has size I^N
  - Too expensive to sample completely!
  - Keywords: **sparse and adaptive sampling**, **black-box sampling**

Some Public Implementations

Authors	Format	Sampling	Implementation
Oseledets et al., 2008	Tucker	Adaptive	MATLAB
Caiafa and Cichocki, 2010	Tucker	Adaptive	MATLAB
Oseledets et al., 2010-now	TT	Adaptive	Python, MATLAB
Filipovic and Jukic, 2012	Tucker	Fixed	MATLAB
Kressner et al., 2013	Tucker	Fixed	MATLAB
Savostyanov, 2014	TT	Adaptive	Fortran
Steinlechner, 2015	TT	Fixed	MATLAB (to appear)

Tensor Recovery

Recover a tensor $\mathscr{A}$:

- Over the known samples Ω, $\mathscr{A}$ should be *close* to the groundtruth:
  - $\mathscr{A}_\Omega \approx \mathcal{T}_\Omega$

Tensor Recovery

Recover a tensor $\mathscr{A}$:

- Over the known samples Ω, $\mathscr{A}$ should be *close* to the groundtruth:
  - $\mathscr{A}_\Omega \approx \mathcal{T}_\Omega$
- The rank of $\mathscr{A}$ should be *small*:
  - rank$(A) = r$
  - rank$(A) \leq r$
  - ... $+ \alpha \cdot$ rank$(A) \rightarrow$ min
  - Relax rank$()$ for another function
- The low-rank constraint/objective acts as a regularization term
  - Other terms may be added

CUR Factorization

- A rank-r matrix can be exactly recovered from r linearly independent rows and columns

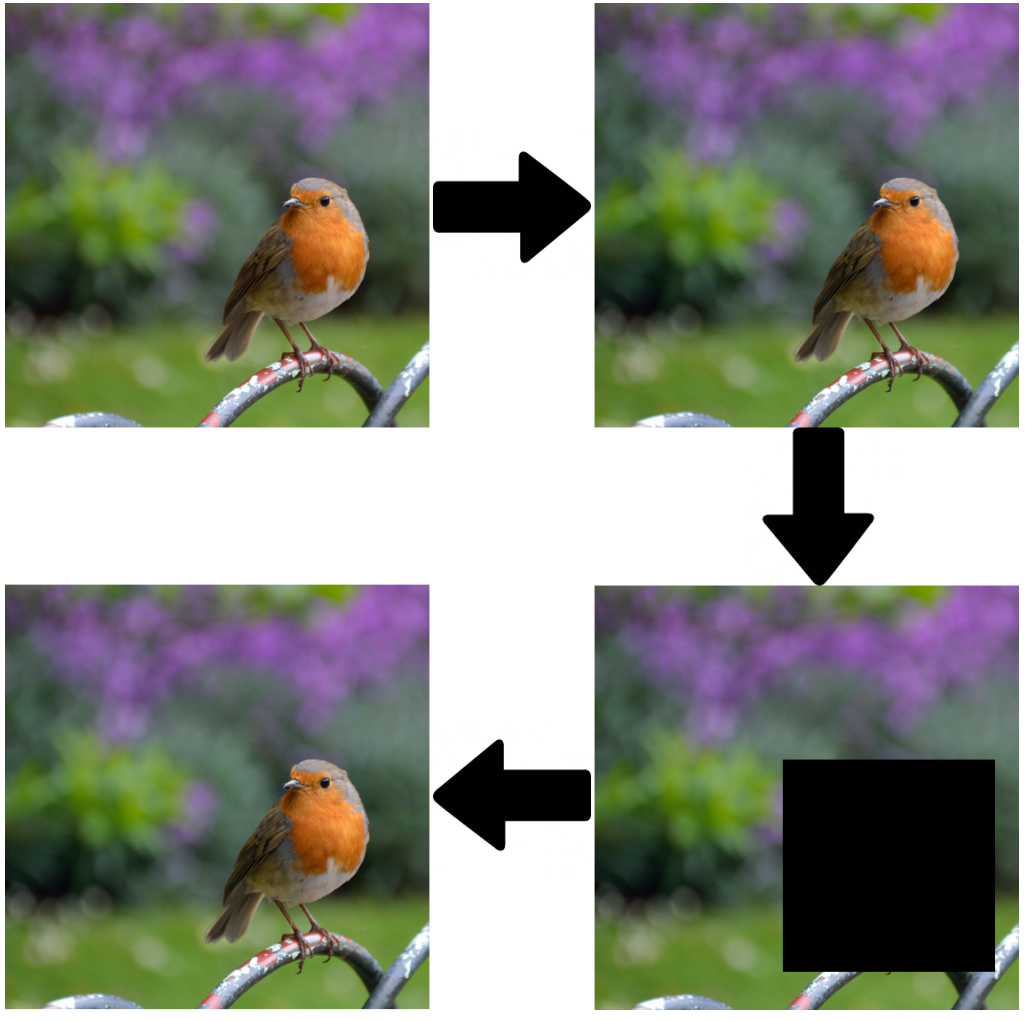
CUR Factorization

- A rank-r matrix can be exactly recovered from r linearly independent rows and columns
- Can work as a robust **error correction code:**
  1. Decrease the rank
  2. Transmit signal
  3. A part is lost
  4. Recover low-rank matrix: $A = CU^{-1}R$

CUR Factorization

- A rank-r matrix can be exactly recovered from r linearly independent rows and columns
- Can work as a robust **error correction code:**
  1. Decrease the rank
  2. Transmit signal
  3. A part is lost
  4. Recover low-rank matrix: $A = CU^{-1}R$
- Versions for higher-order tensors

CUR Factorization

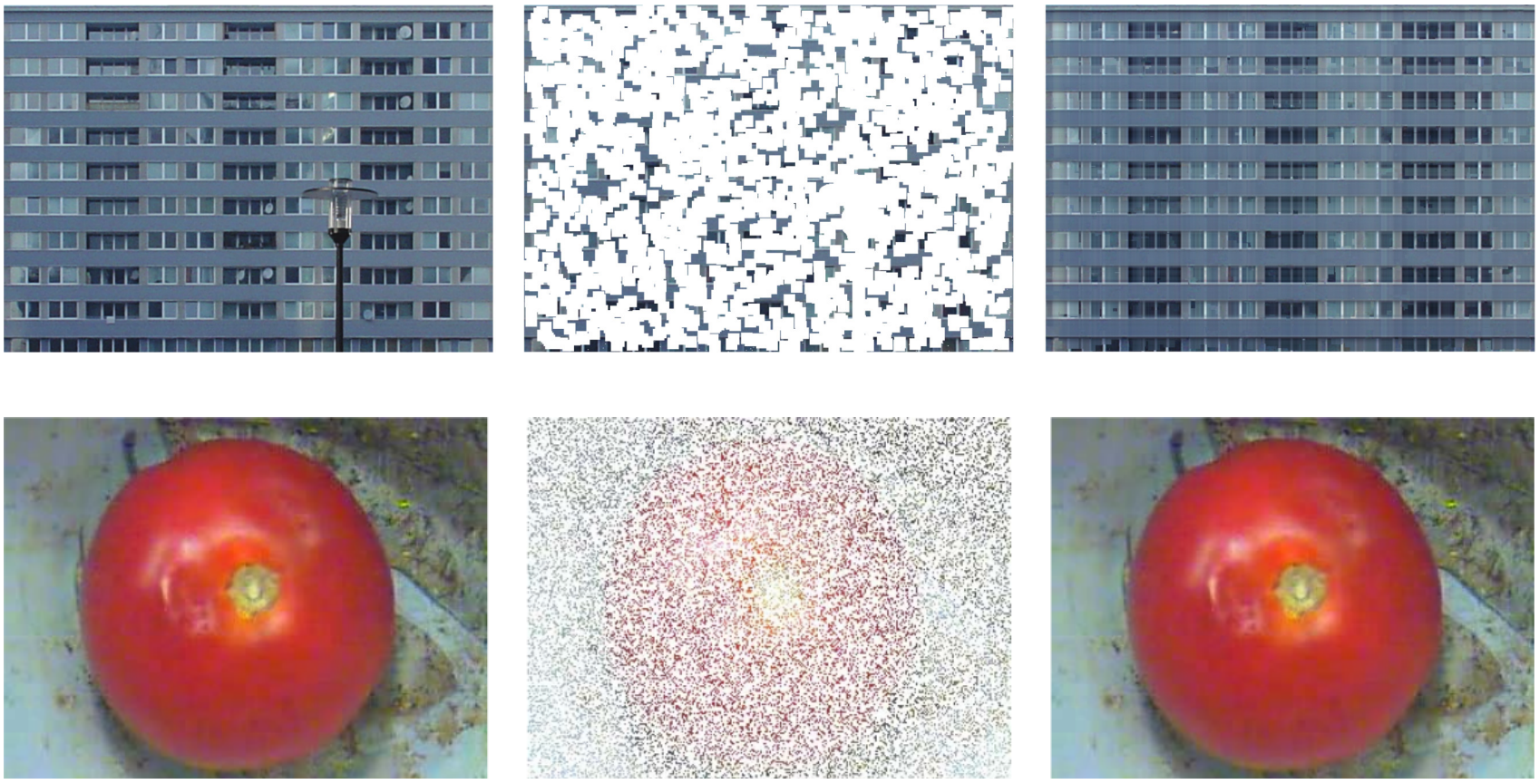


Tensor Recovery

- ”Tensor Completion for Estimating Missing Values in Visual Data (Liu et al.)”

$$\begin{cases} \arg\min_{\mathscr{A},\mathscr{B}} \frac{1}{2}\|\mathscr{A}-\mathscr{B}\|^2 \\ \text{s.t. } \|\mathscr{A}\|_{tr} \leq r \\ \text{and } \mathscr{B}_\Omega = \mathcal{T}_\Omega \end{cases}$$

Tensor Recovery



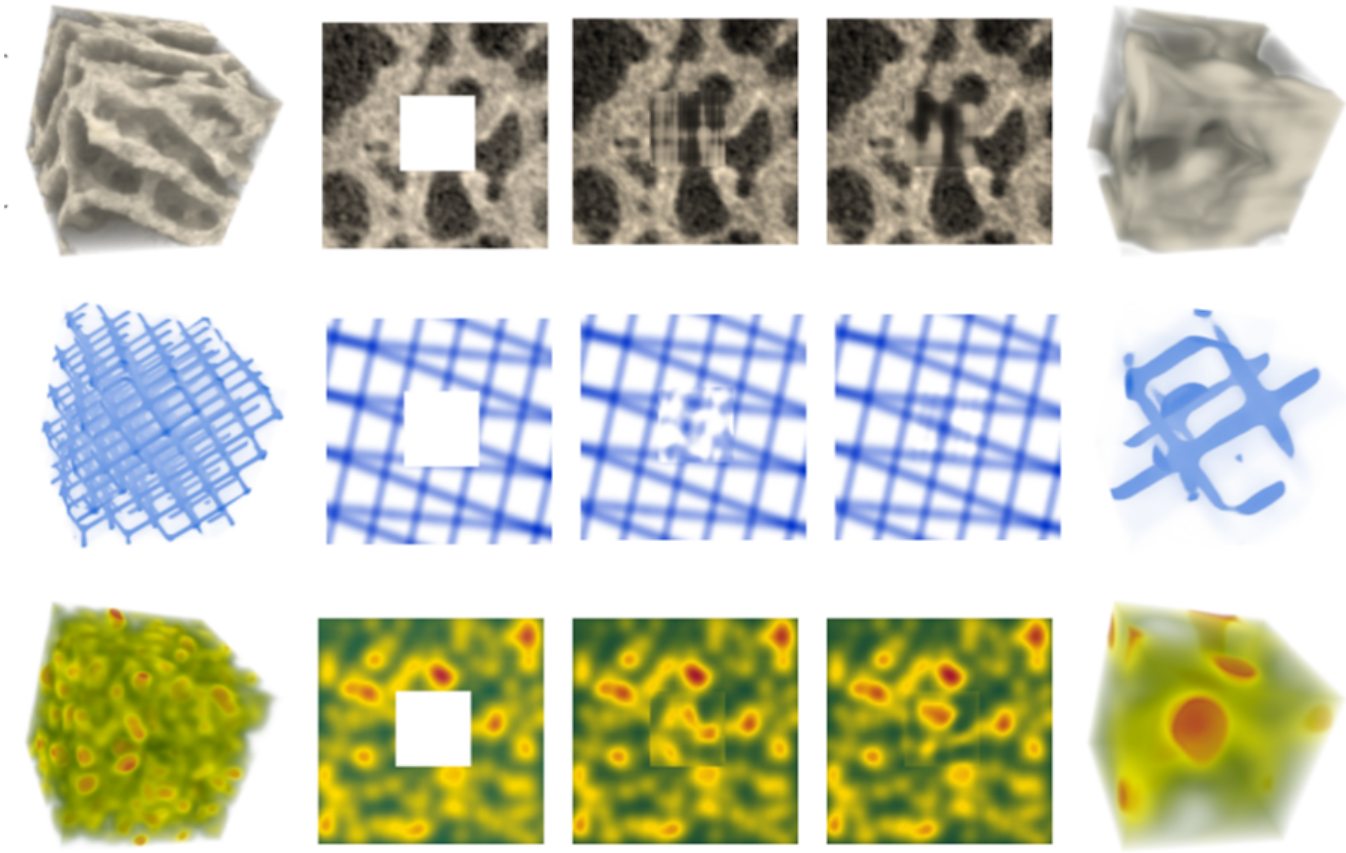
Source: ”Tensor Completion for Estimating Missing Values in Visual Data” ([LMW+09])

Texture Synthesis

- ”Structural Volume Inpainting Via Tucker Dictionary Learning” (Ballester-Ripoll and Pajarola)

$$\left\{ \arg\min_{\mathbf{U}^{(1)},\mathbf{U}^{(2)},\mathbf{U}^{(3)}} \left\| \mathscr{A} - \mathscr{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)} \right\| \right.$$

Texture Synthesis



Source: ”Structural Volume Inpainting Via Tucker Dictionary Learning” ([BP16])

Black-box Sampling

- We have a function $s : \mathbb{R}^N \rightarrow \mathbb{R}$ (or $\mathbb{R}^M$)
  - Example: hyperparameters in a simulation

Black-box Sampling

- We have a function $s : \mathbb{R}^N \rightarrow \mathbb{R}$ (or $\mathbb{R}^M$)
  - Example: hyperparameters in a simulation
- Explicit storage is impossible

Black-box Sampling

- We have a function $s : \mathbb{R}^N \rightarrow \mathbb{R}$ (or $\mathbb{R}^M$)
  - Example: hyperparameters in a simulation
- Explicit storage is impossible
- Freedom to sample $\rightarrow$ **adaptive schemes**

Black-box Sampling

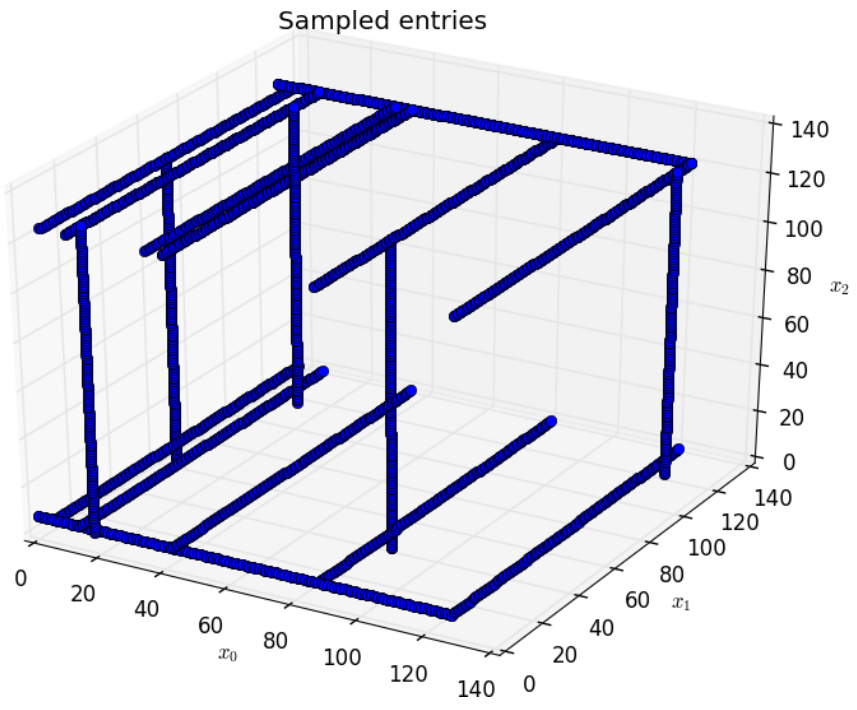
- Example: **cross approximation** [OT10]

Black-box Sampling

- Example: **cross approximation** [OT10]
- Adaptive sampling using fibers
  - Increases progressively the tensor rank
  - Code from the Tensor Train Toolbox

Black-box Sampling

- Example: **cross approximation** [OT10]
- Adaptive sampling using fibers
  - Increases progressively the tensor rank
  - Code from the Tensor Train Toolbox



VISUALIZATIONAND MULTIMEDIALAB

Black-box Sampling

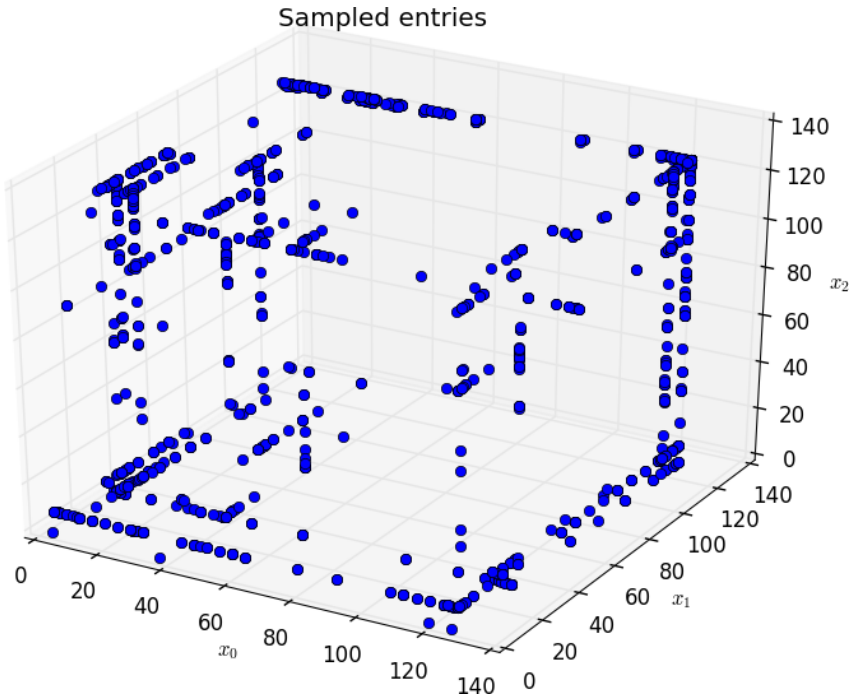
- Quantized tensor train (QTT) format:
  - Reshape axes:

$$[0, 2^d - 1] \text{ becomes } \overbrace{\{0,1\} \times ... \times \{0,1\}}^{d}$$

Black-box Sampling

- Quantized tensor train (QTT) format:
  - Reshape axes:

$$[0, 2^d - 1] \text{ becomes } \overbrace{\{0,1\} \times ... \times \{0,1\}}^{d}$$



VISUALIZATIONAND MULTIMEDIALAB

Surrogate Visualization Models

- Situation:
  - **Expensive sampling**
  - Complex and high-dimensional space

Surrogate Visualization Models

- Situation:
  - **Expensive sampling**
  - Complex and high-dimensional space
- Interactive visualization: important tool to **gain intuition**
  - What is the behavior of s as its parameters are varied?

Surrogate Visualization Models

- Situation:
  - **Expensive sampling**
  - Complex and high-dimensional space
- Interactive visualization: important tool to **gain intuition**
  - What is the behavior of s as its parameters are varied?
- A **surrogate model** approximates an underlying model
  - High-speed reconstruction is critical

Surrogate Visualization Models

- Many visualization diagrams are **axis-aligned subspaces**
  - 1D plot: $\mathscr{A}[i_1, i_2, :]$
  - Surface plot: $\mathscr{A}[i_1, :, :]$
  - Tables, overlayed plots, etc.

Surrogate Visualization Models

- Many visualization diagrams are **axis-aligned subspaces**
  - 1D plot: $\mathscr{A}[i_1, i_2, :]$
  - Surface plot: $\mathscr{A}[i_1, :, :]$
  - Tables, overlayed plots, etc.
- Others:
  - Dimensional stacking
  - Projections
  - Parallel coordinates

Surrogate Visualization Models

- How many axis-aligned slices can we take? $\binom{\mathbf{N}}{\mathbf{2}} \cdot \mathbf{I}^{\mathbf{N-2}}$
- Example: (4D):

$$\mathscr{A}[i_1, i_2, :, :]$$

$$\mathscr{A}[i_1, :, i_3, :]$$

$$\mathscr{A}[i_1, :, :, i_4]$$

$$\ldots$$

  - Precomputing is unfeasible (space needed: $\binom{N}{2} \cdot I^N$)

Surrogate Visualization Models

- How many axis-aligned slices can we take? $\binom{\mathbf{N}}{\mathbf{2}} \cdot \mathbf{I}^{\mathbf{N-2}}$
- Example: (4D):

$$\mathscr{A}[i_1, i_2, :, :]$$

$$\mathscr{A}[i_1, :, i_3, :]$$

$$\mathscr{A}[i_1, :, :, i_4]$$

$$\ldots$$

  - Precomputing is unfeasible (space needed: $\binom{N}{2} \cdot I^N$)
- But: **very efficient** to reconstruct **from a tensor!**

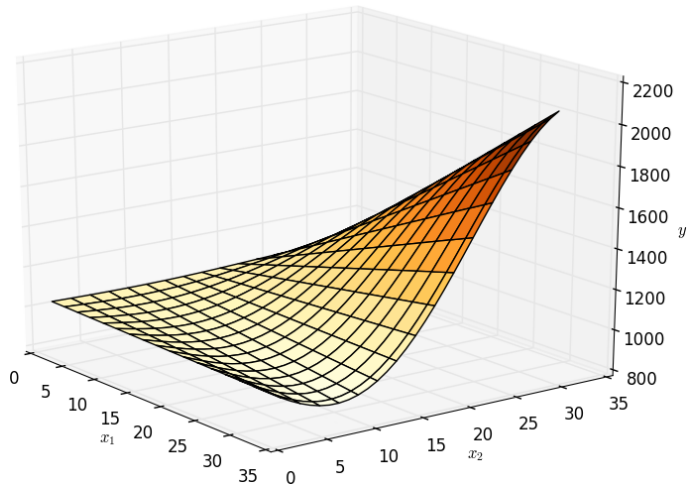
Surrogate Visualization Models

- A subspace of dimension M can be reconstructed in:
  - $O(I^M R + NR)$ ops. (CP)
  - $O(I^M R + R^N)$ ops. (Tucker)
  - $O(I^M R + NR^2)$ ops. (TT)

Surrogate Visualization Models

- A subspace of dimension M can be reconstructed in:
  - $O(I^M R + NR)$ ops. (CP)
  - $O(I^M R + R^N)$ ops. (Tucker)
  - $O(I^M R + NR^2)$ ops. (TT)

- 1D and surface plots: often **a few milliseconds**



TT Optimization

- Global search:

$$\arg\max_{i_1,\ldots,i_N} \mathscr{A}[i_1, \ldots, i_N]$$

  - Available in the Tensor Train Toolbox

TT Optimization

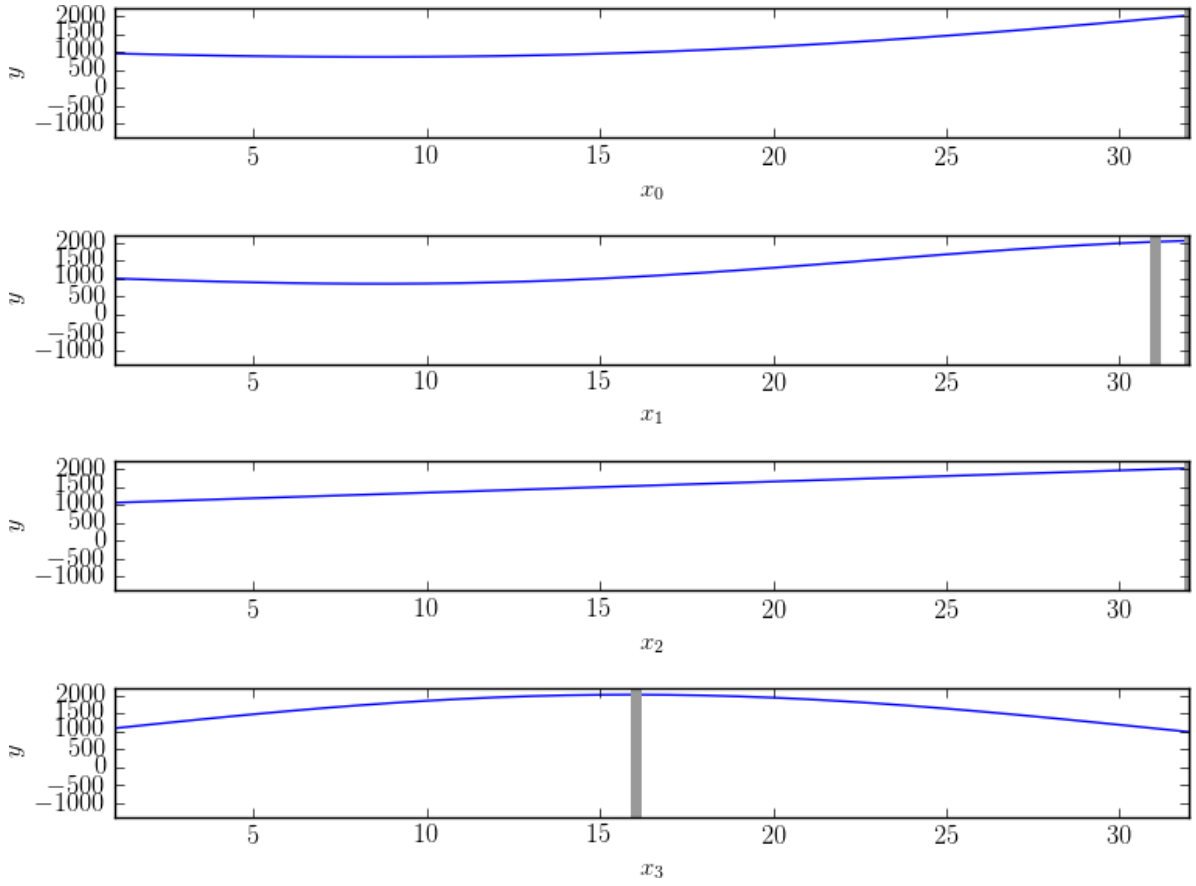
- Global search:

$$\arg\max_{i_1,\ldots,i_N} \mathscr{A}[i_1, \ldots, i_N]$$

  - Available in the Tensor Train Toolbox
- Find the **best hyperparameters**
- Find **good subspaces**:
  - Slice with the most variance
  - Slice with the most curvature
  - Etc.

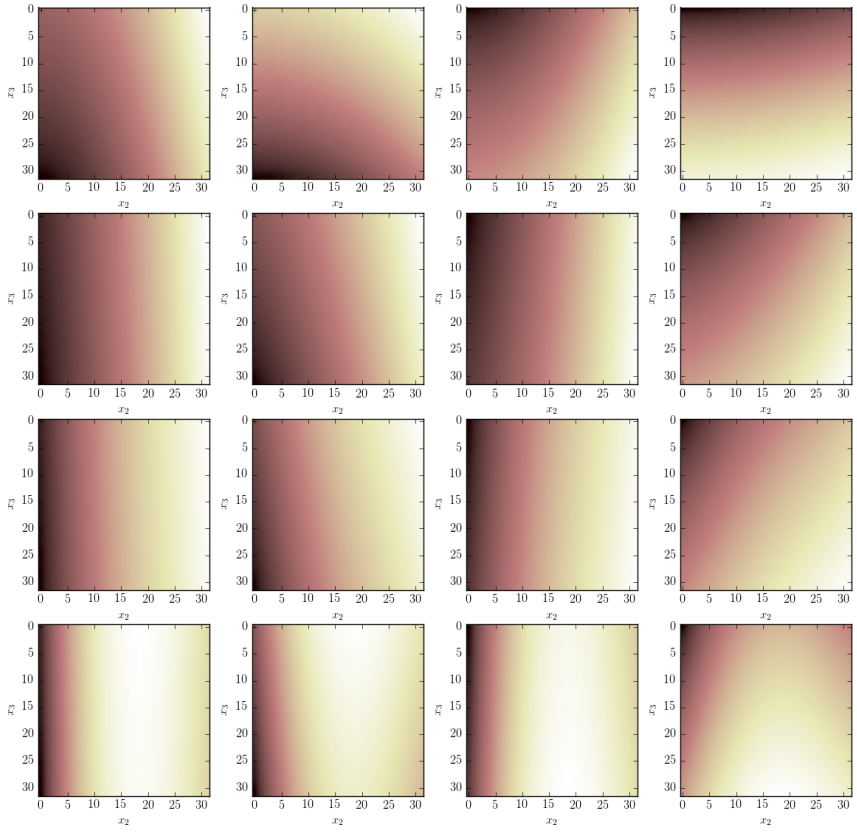
TT Optimization

- Example: fibers passing through the maximum



Dimensional Stacking

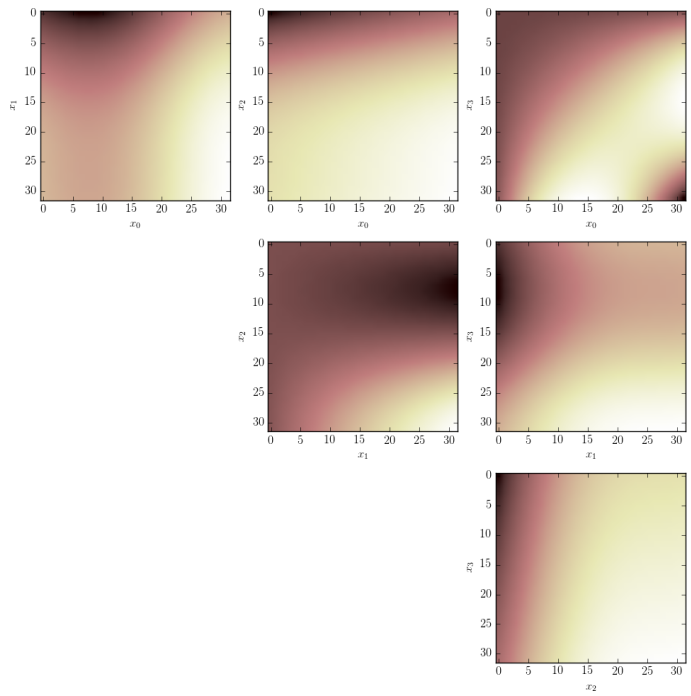
- 2 indices move freely, 2 are discretized
- Example: $\mathscr{A}[::8, ::8, :, :]$ ($\mathscr{A}$ is 32^4)



Bivariate Projections

- Upper triangular matrix of images. At entry (a, b):

$$\sum_{\widehat{i_a}, \widehat{i_b}} \mathscr{A}$$

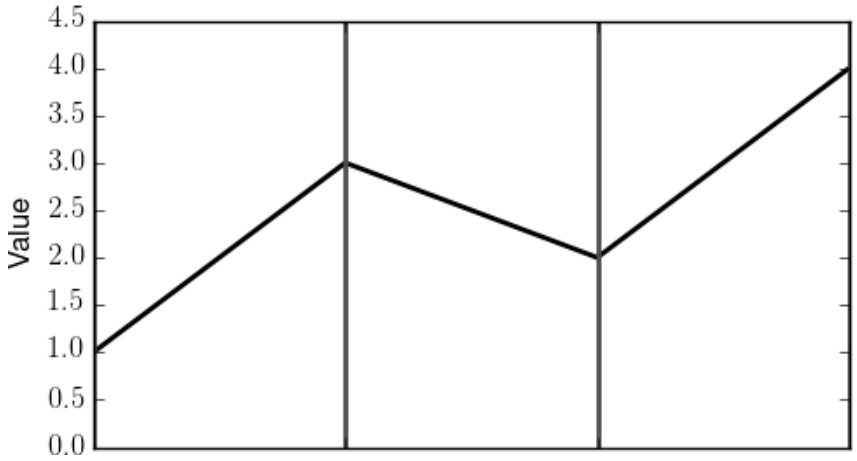


Parallel Coordinates

- A point $(i_1, ..., i_N)$ is represented by a polyline:

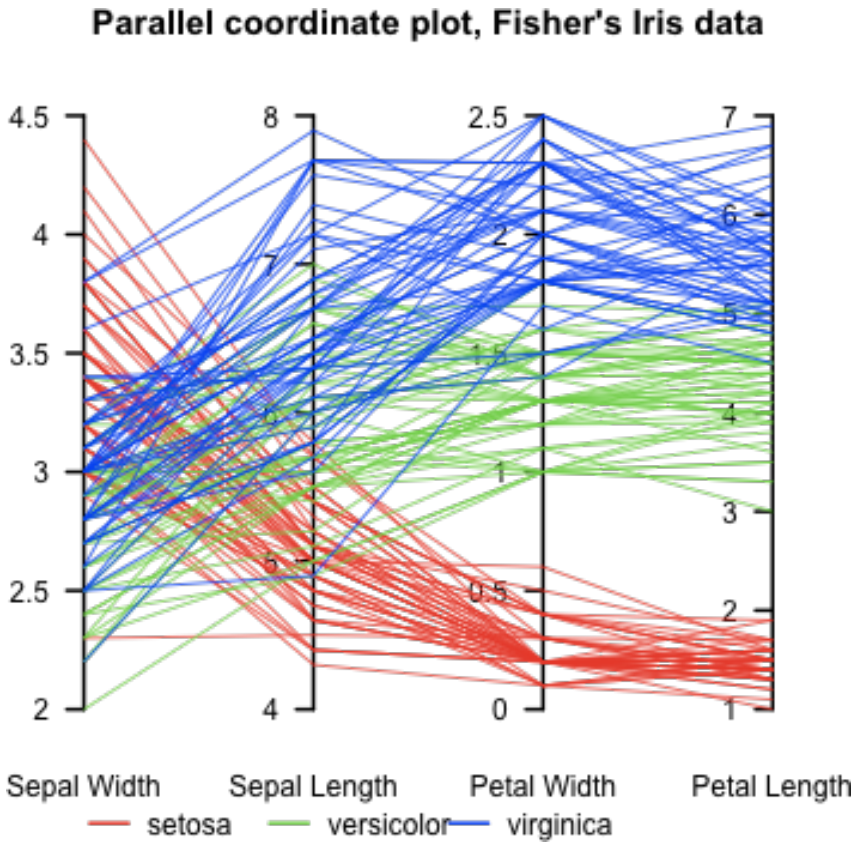
$$(0, i_1) \rightarrow (1, i_2) \rightarrow ... \rightarrow (N-1, i_N)$$

- One vertical axis per dimension
- Example: point $(1, 3, 2, 4)$



Parallel Coordinates

- Real-world example



Source: *en.wikipedia.org*

Parallel Coordinates (Continuous Version)

- $\forall i_1, ..., i_N$ there is a polyline connecting:

$$(0, i_1) \rightarrow (1, i_2) \rightarrow ... \rightarrow (N-1, i_N)$$

with opacity $\mathscr{A}[i_1, ..., i_N]$

Parallel Coordinates (Continuous Version)

- $\forall i_1, ..., i_N$ there is a polyline connecting:

$$(0, i_1) \rightarrow (1, i_2) \rightarrow ... \rightarrow (N-1, i_N)$$

with opacity $\mathscr{A}[i_1, ..., i_N]$

- Therefore $\forall a, \widehat{i_a}, \widehat{i_{a+1}}$, there is a segment

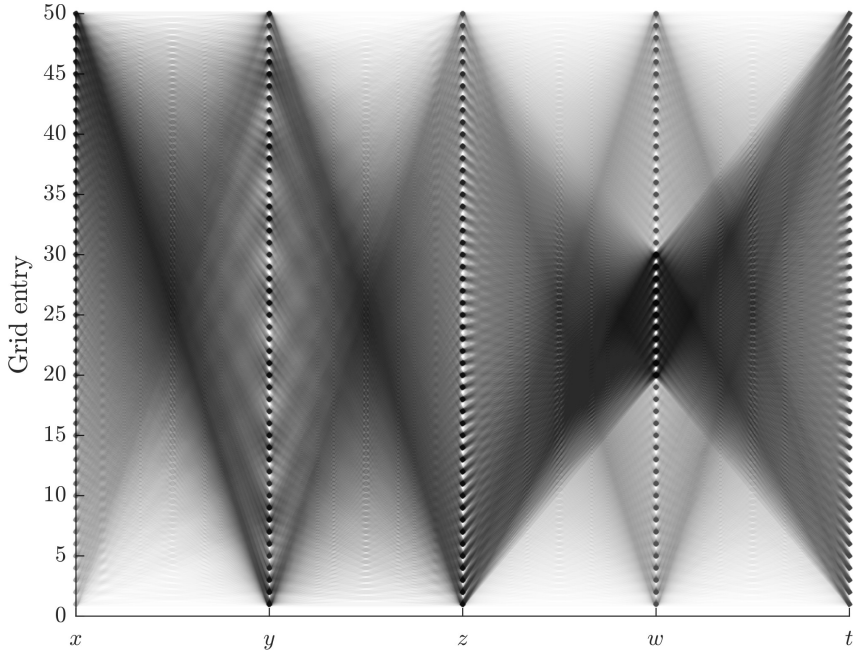
$$(a, \widehat{i_a}) \rightarrow (a+1, \widehat{i_{a+1}})$$

with opacity

$$\sum_{i_a = \widehat{i_a},\ i_{a+1} = \widehat{i_{a+1}}} \mathscr{A}[i_1, ..., i_N]$$

- Can be computed from a **bivariate projection**!

Parallel Coordinates (Continuous Version)



- Computed from a TT in $\approx$ 5ms
- Dimension reordering is trivial

To Sum Up...

Tensors as a **framework for visualization**

Crucial properties combined:

- Methods for sampling and completion
  - Fixed
  - Adaptive

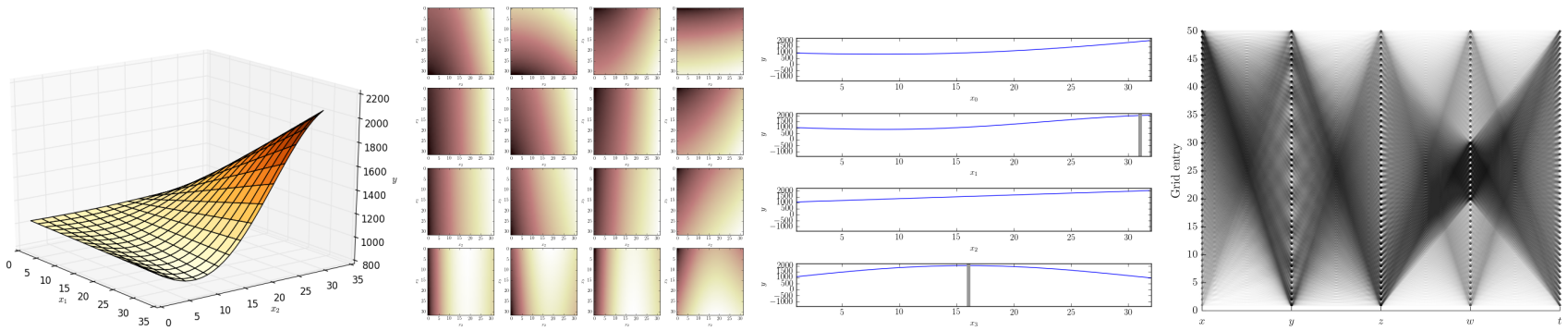
To Sum Up...

Tensors as a **framework for visualization**

Crucial properties combined:

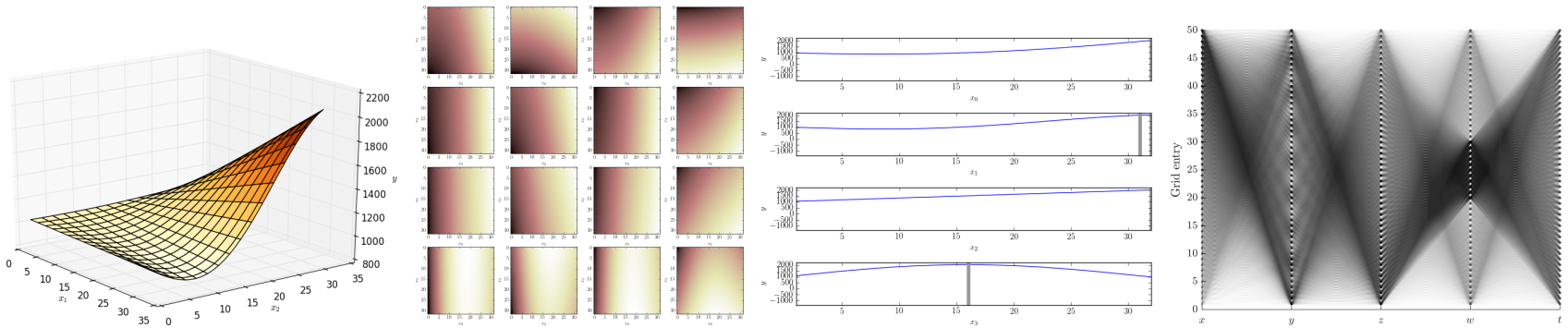
- Methods for sampling and completion
  - Fixed
  - Adaptive
- Fast navigation
  - Statistical moments
  - Optimization
  - Interactive reconstruction

Tensorplot



- Python project to **sample and visualize tensors**
- Collection of wrappers

Tensorplot



- Goals:
  - Support several completion algorithms
  - Support CP, Tucker, TT, QTT
- **Interactive GUI** is in the works
  - Code will be released soon

Rafael Ballester-Ripoll
rballester@ifi.uzh.ch

http://www.ifi.uzh.ch/vmml.html

Universität Zürich[UZH]