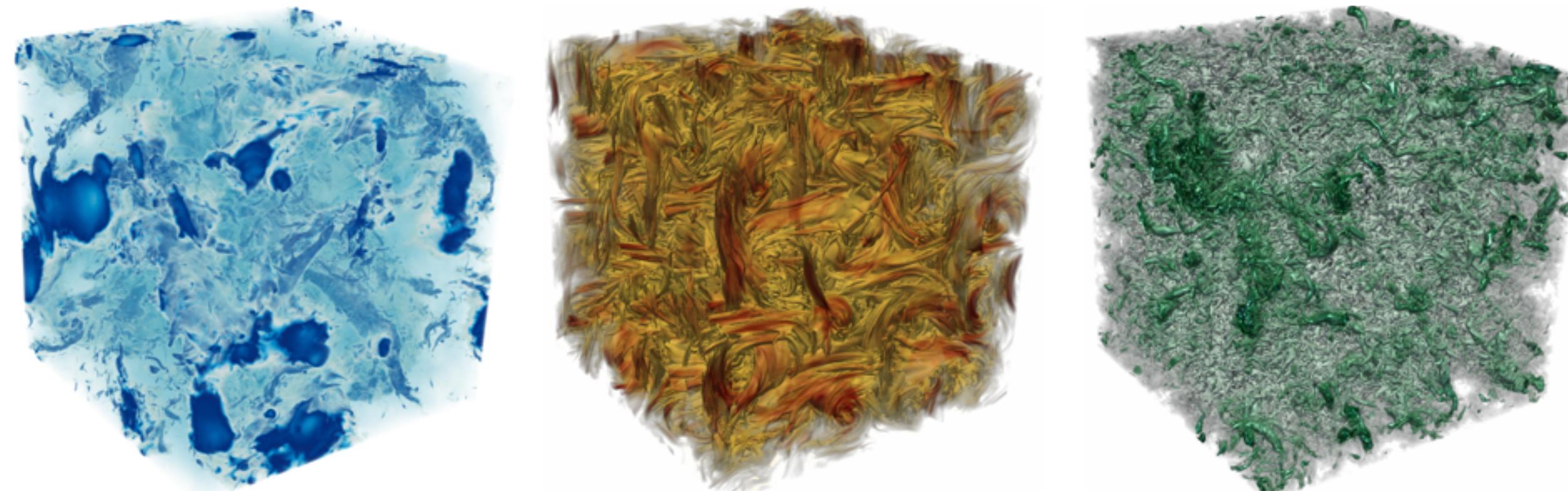


Tensor Decompositions for Integral Histogram Compression and Look-up

Rafael Ballester-Ripoll & Renato Pajarola

IEEE VIS '18 (TVCG paper)

25th October 2018



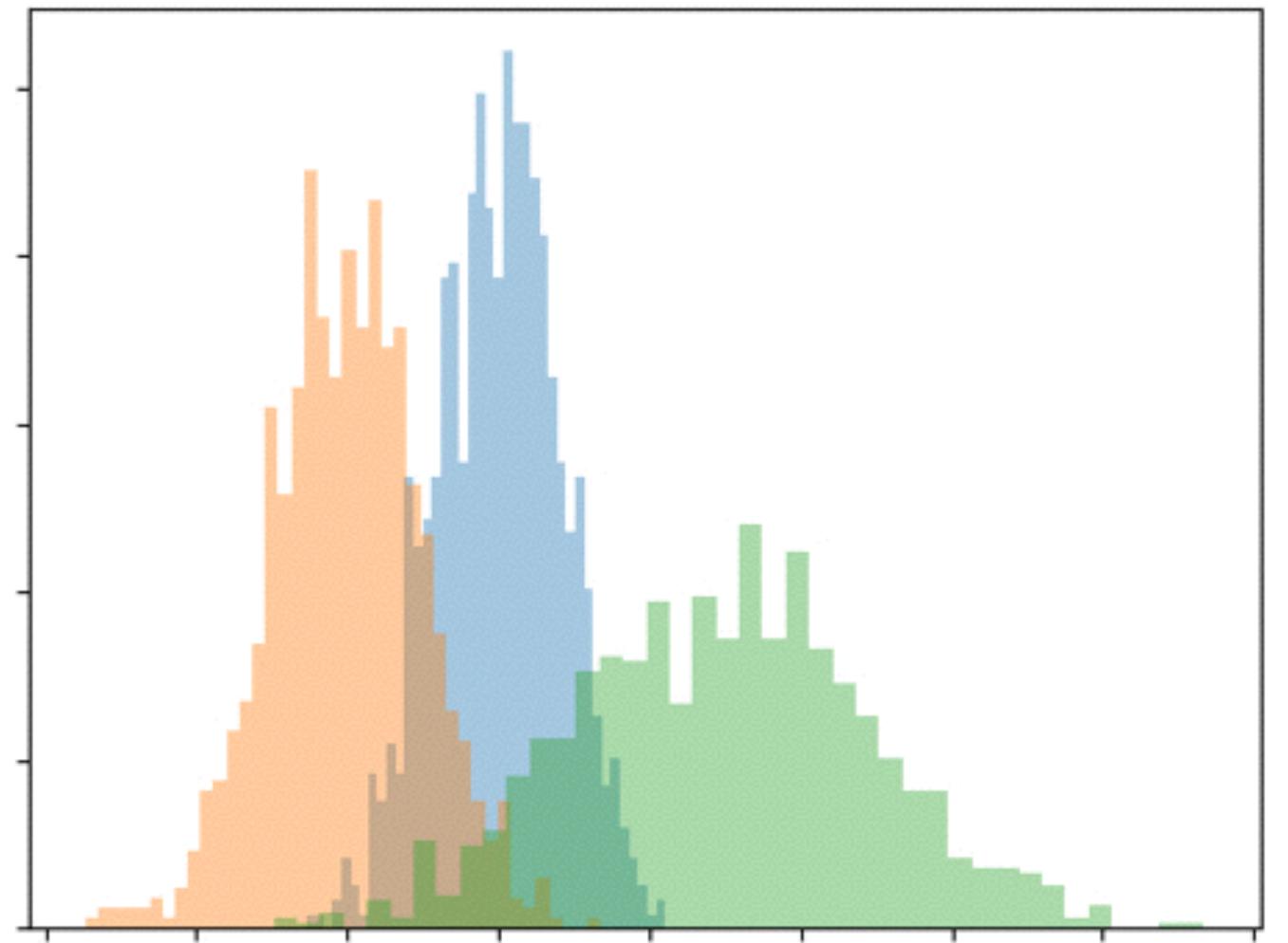
University of
Zurich^{UZH}



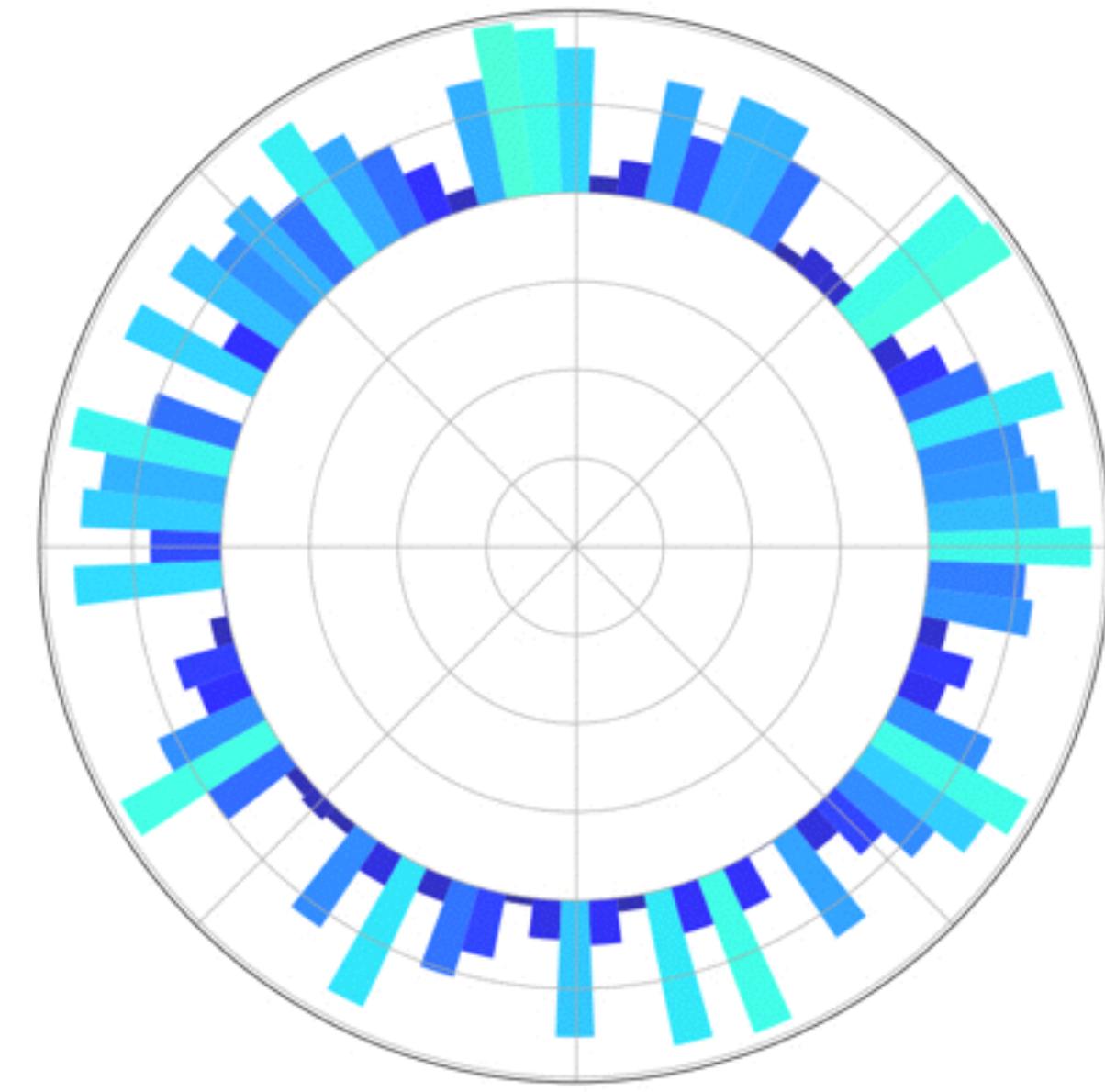
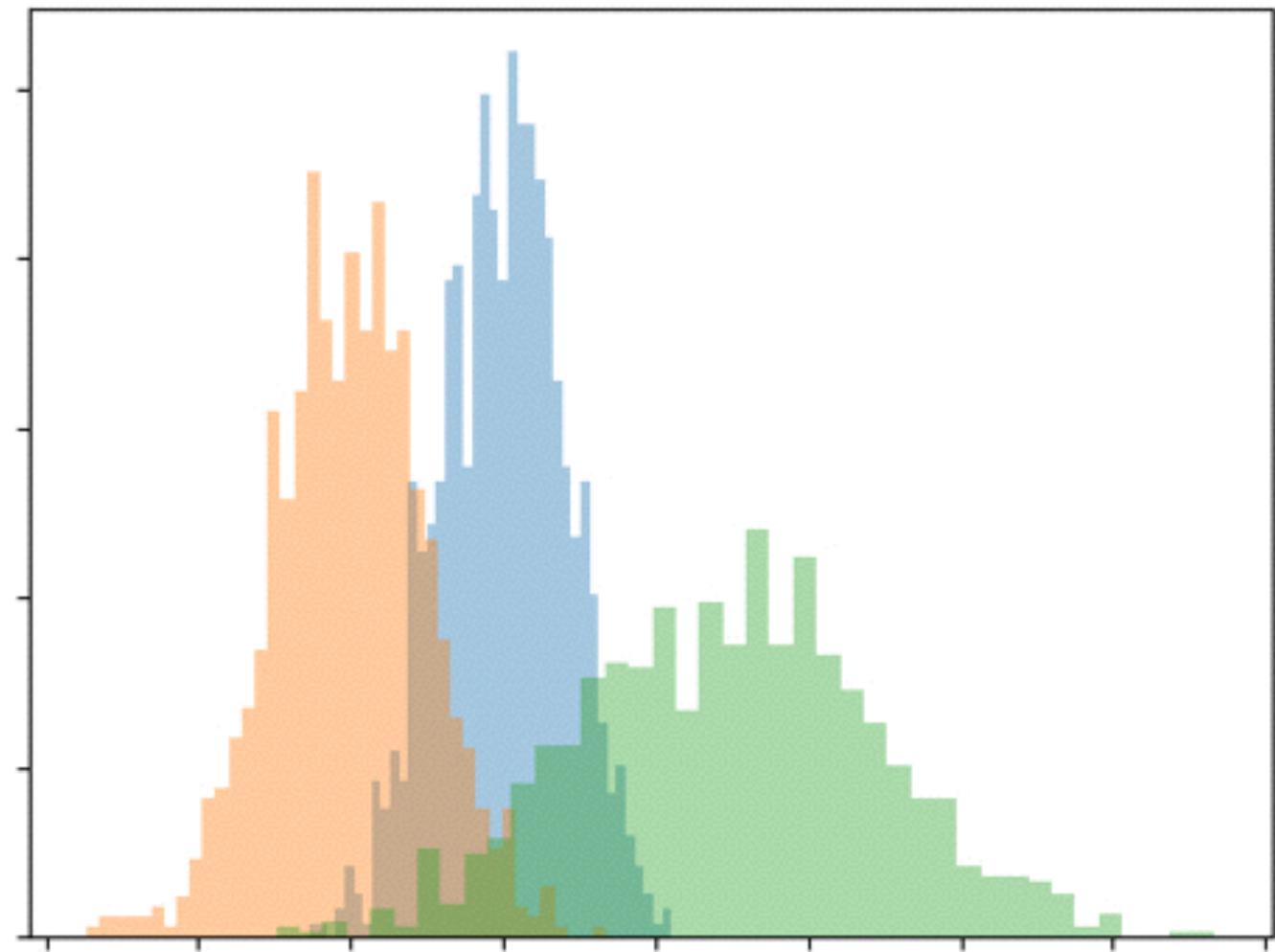
VISUALIZATION AND
MULTIMEDIALAB



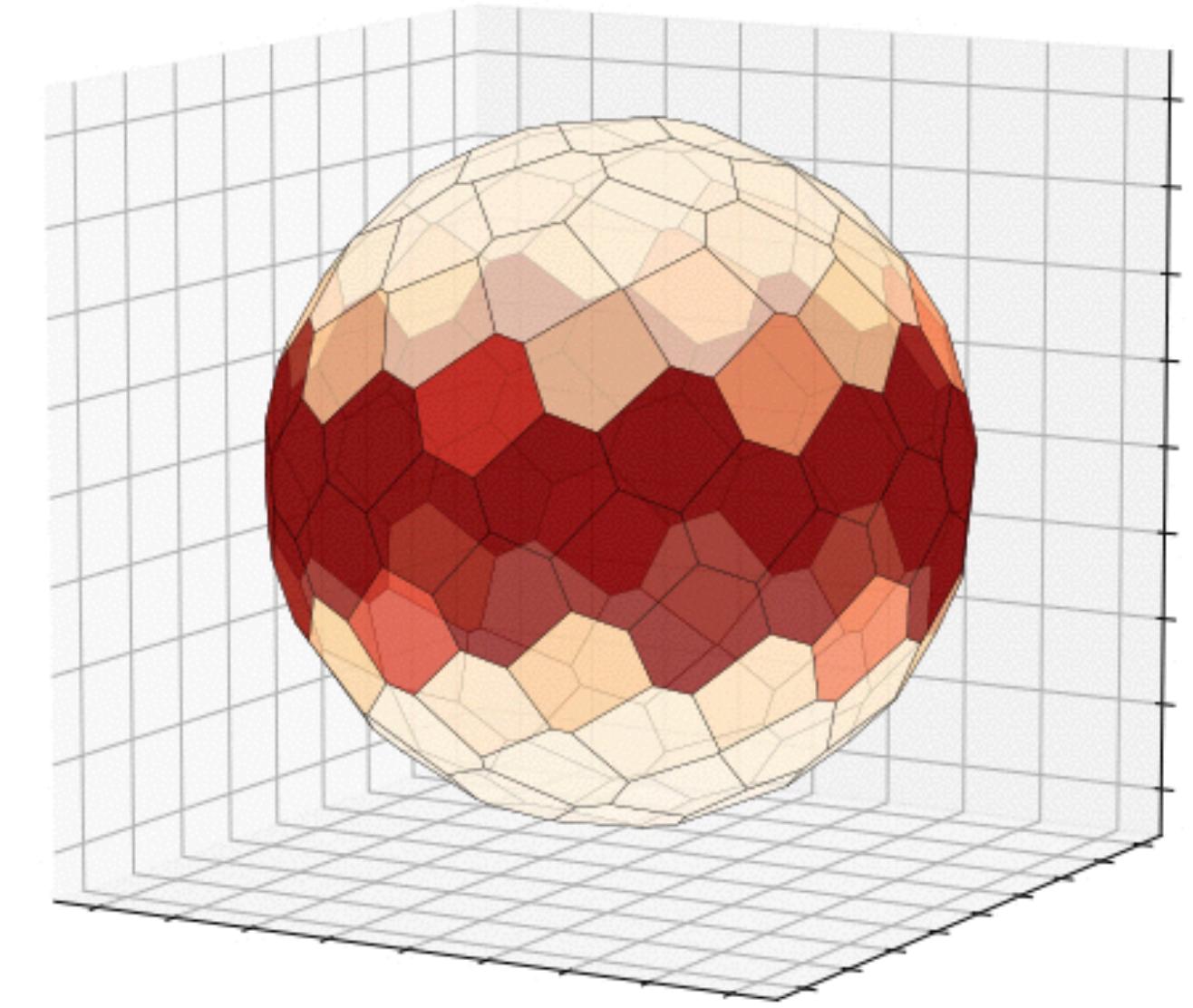
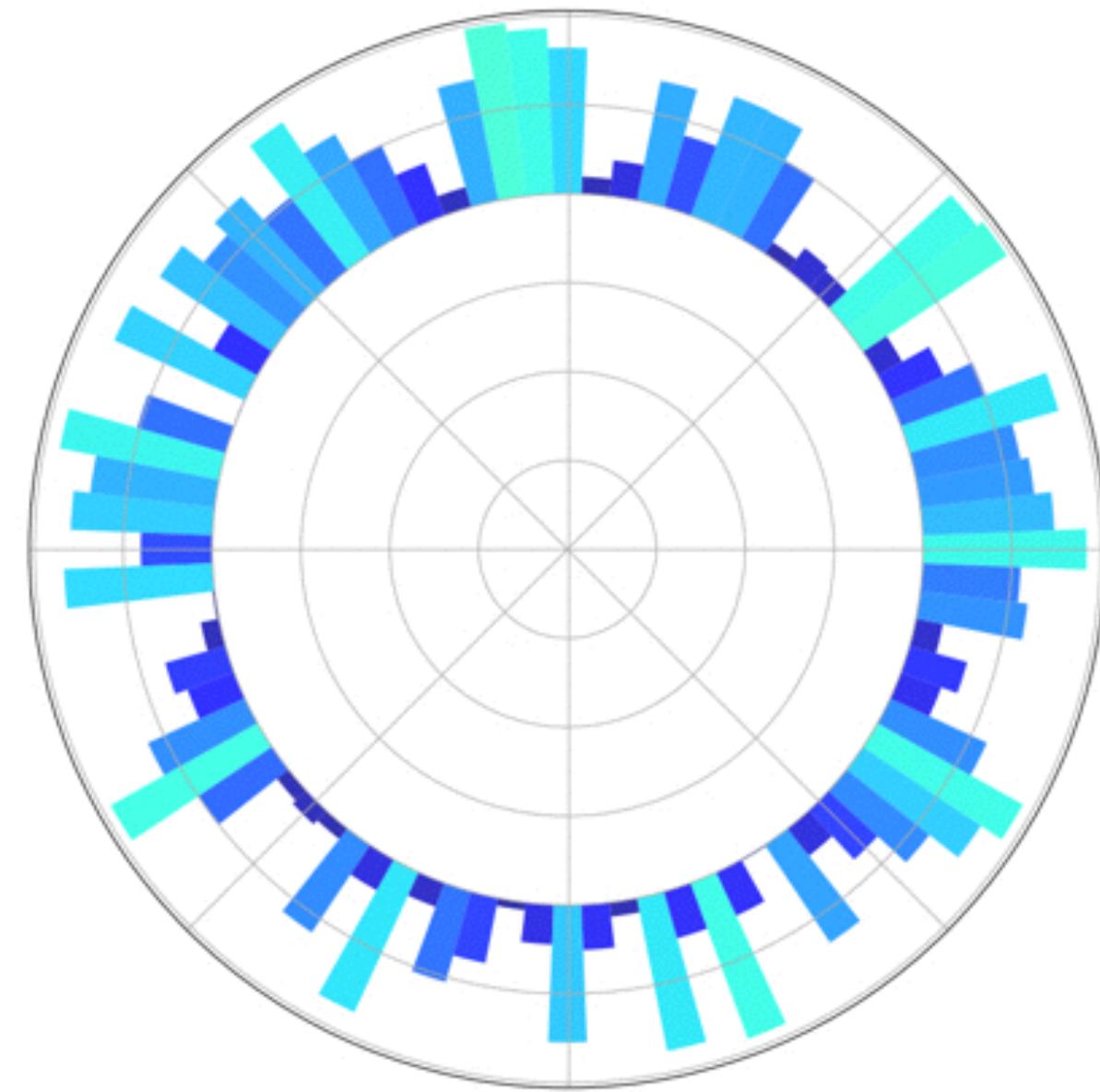
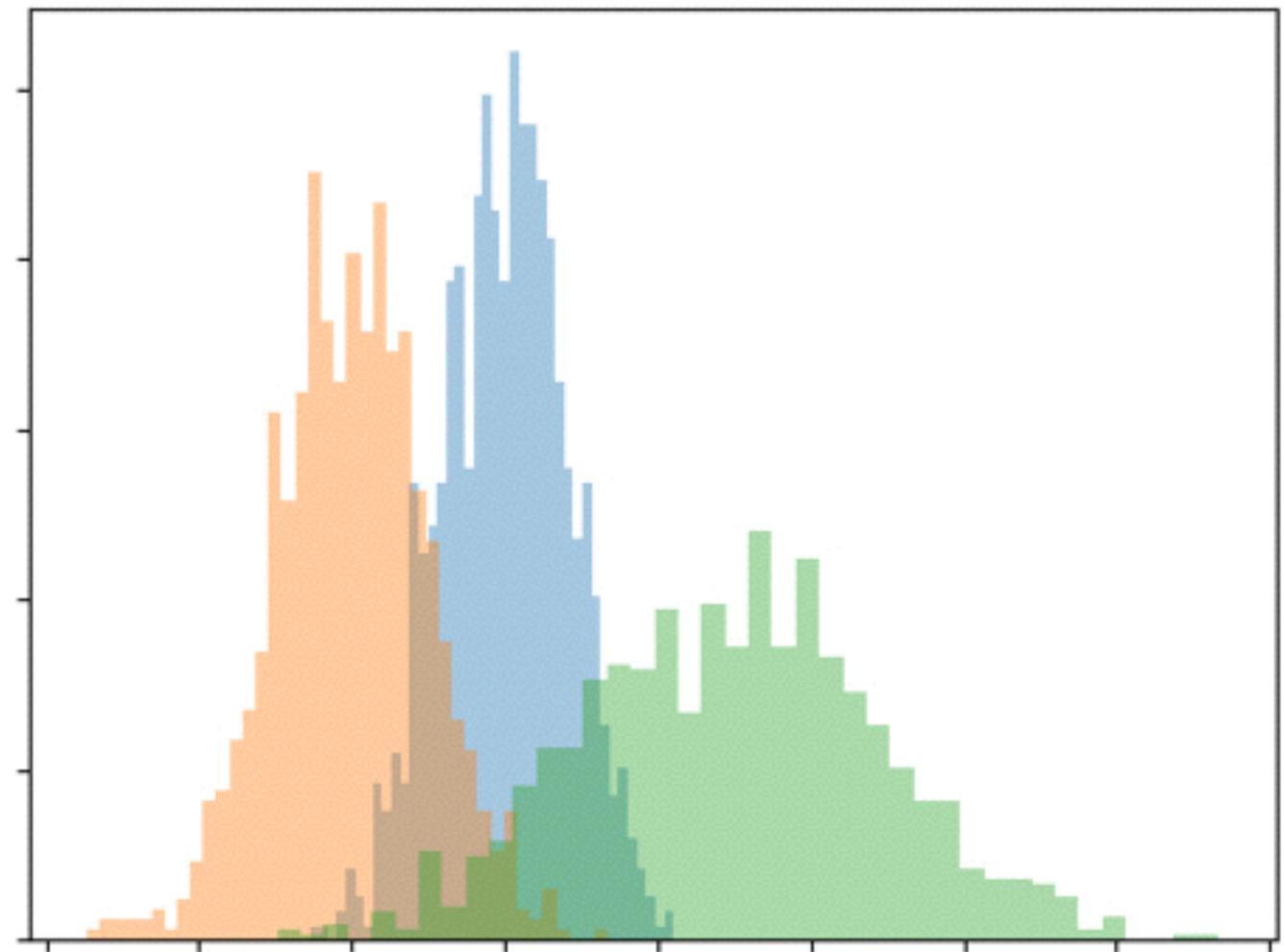
Histograms



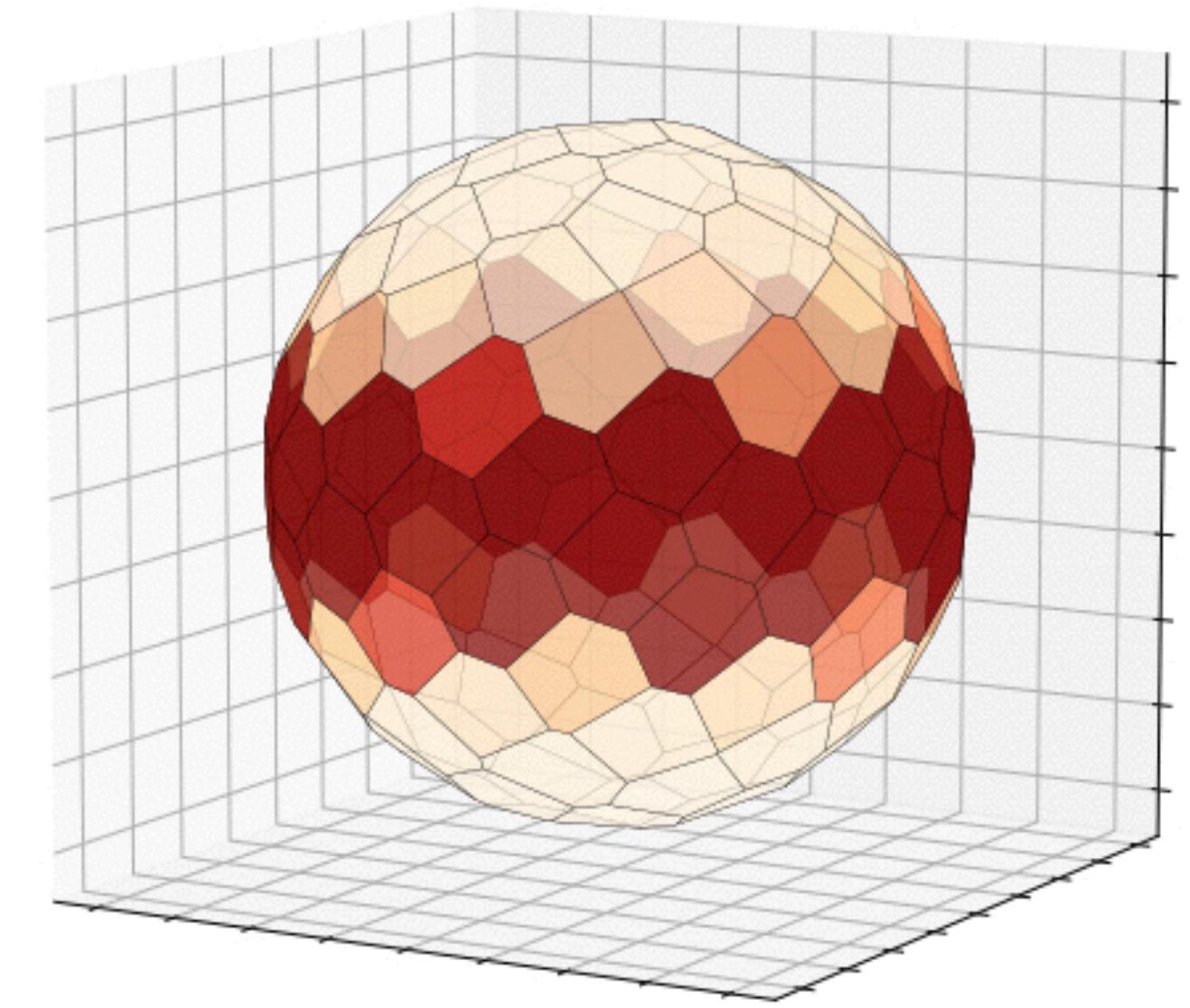
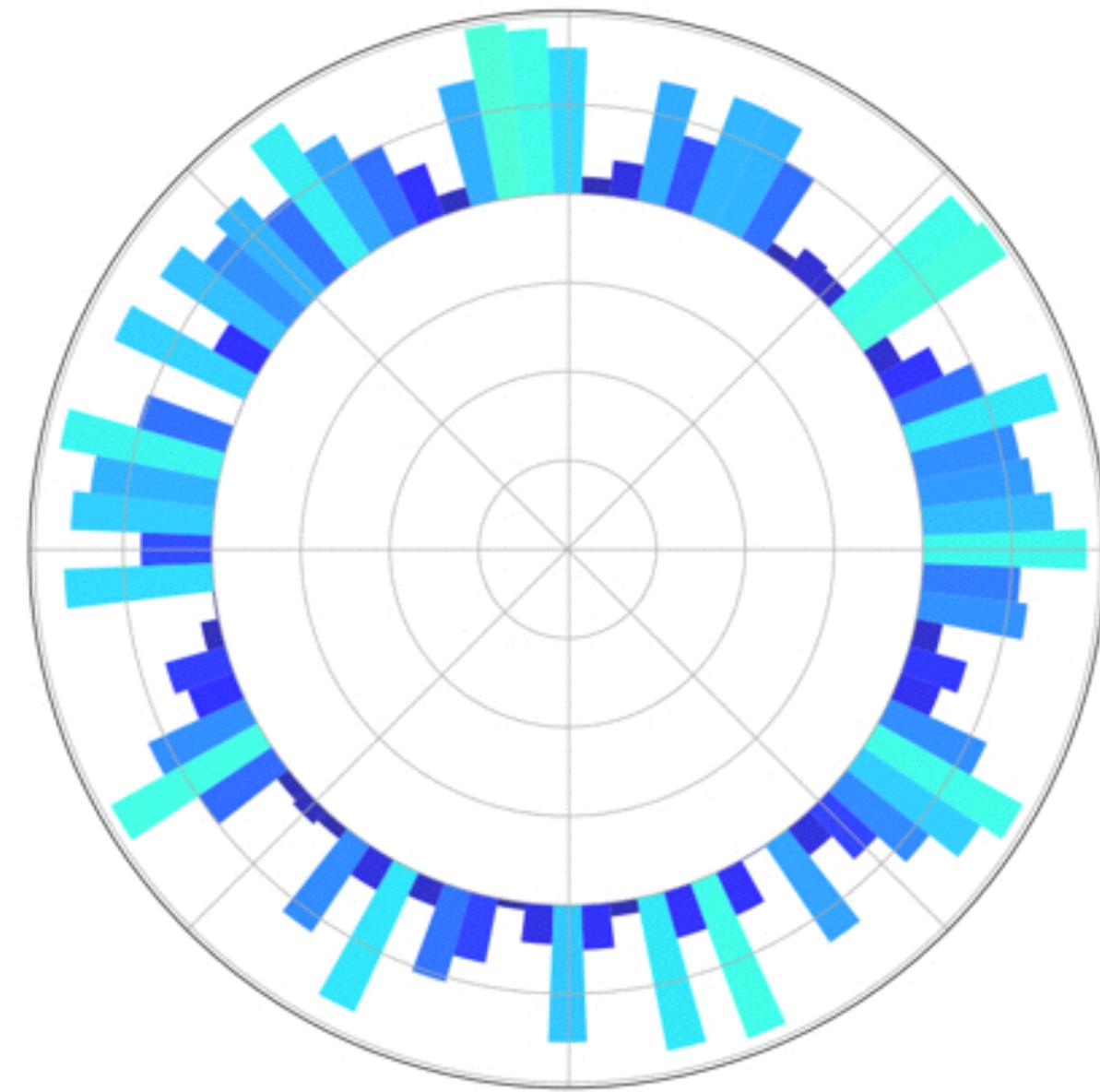
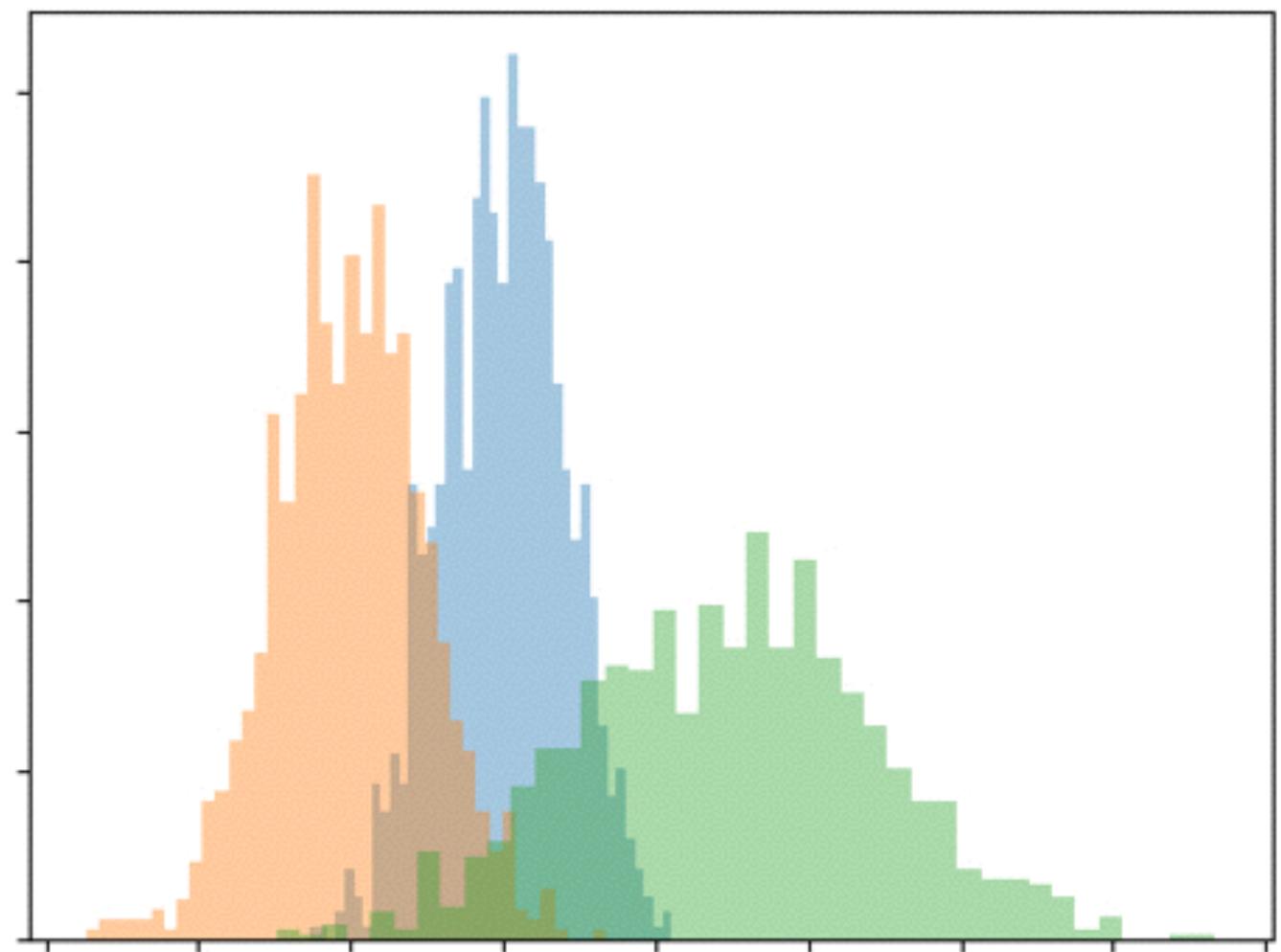
Histograms



Histograms

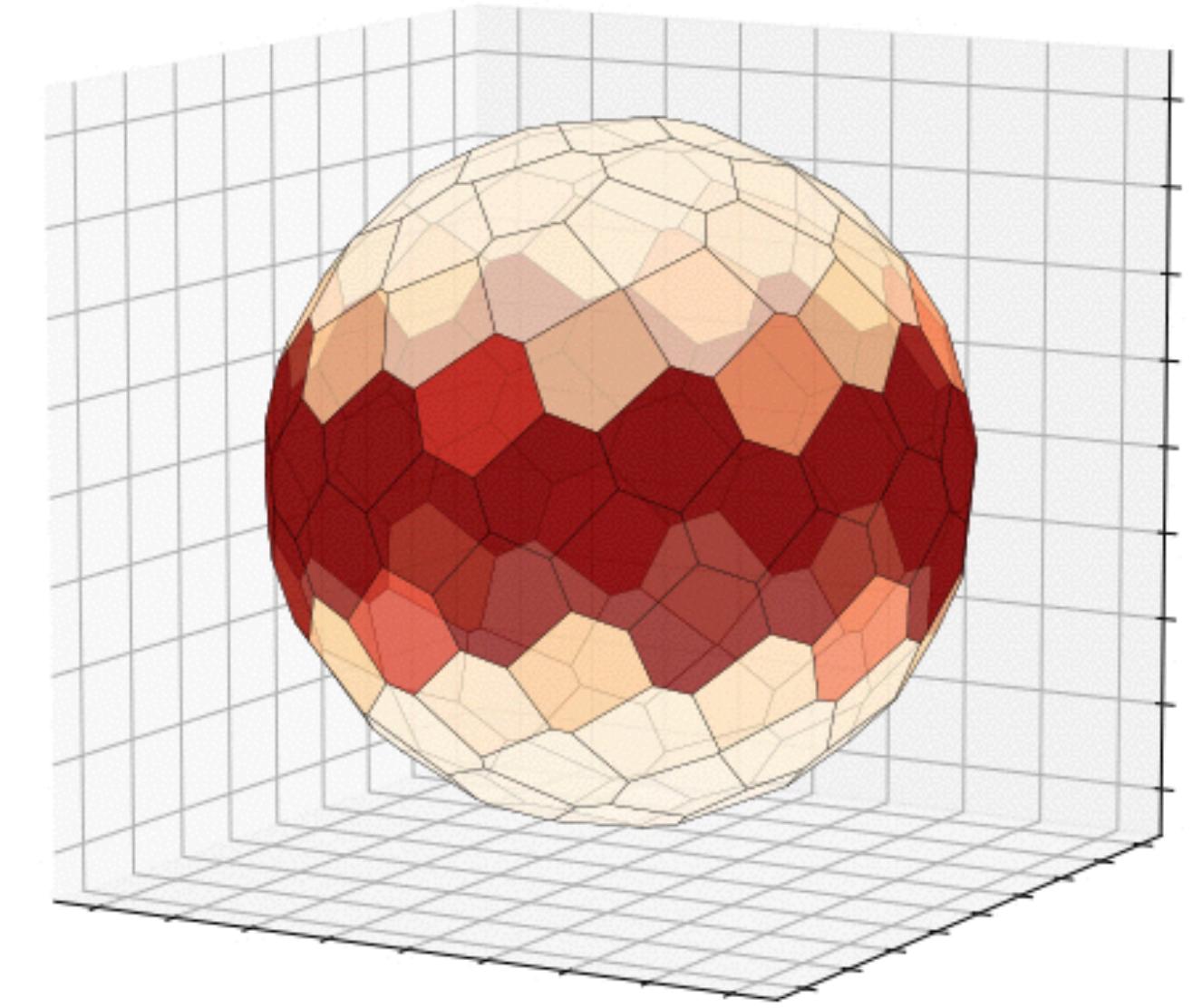
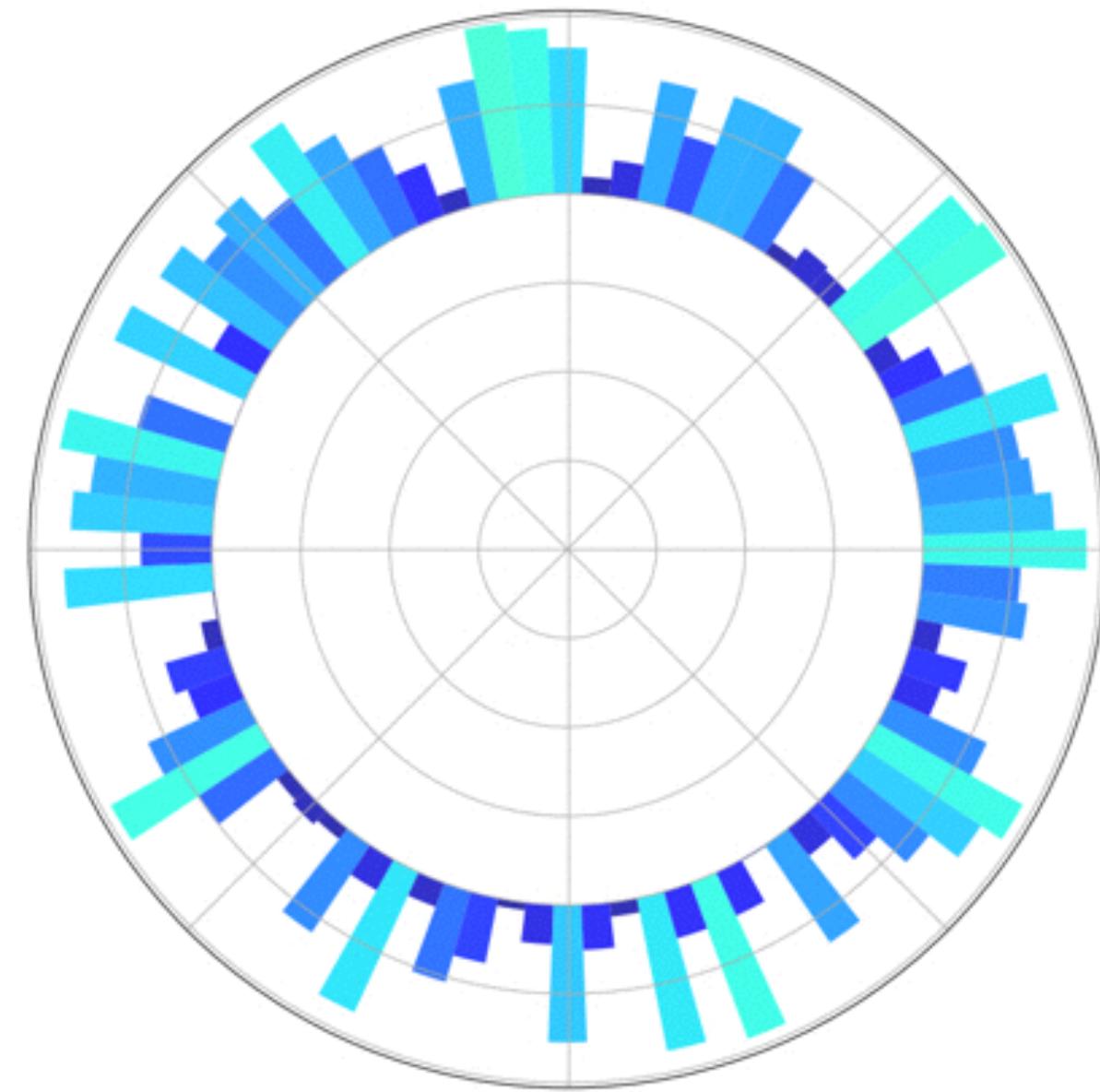
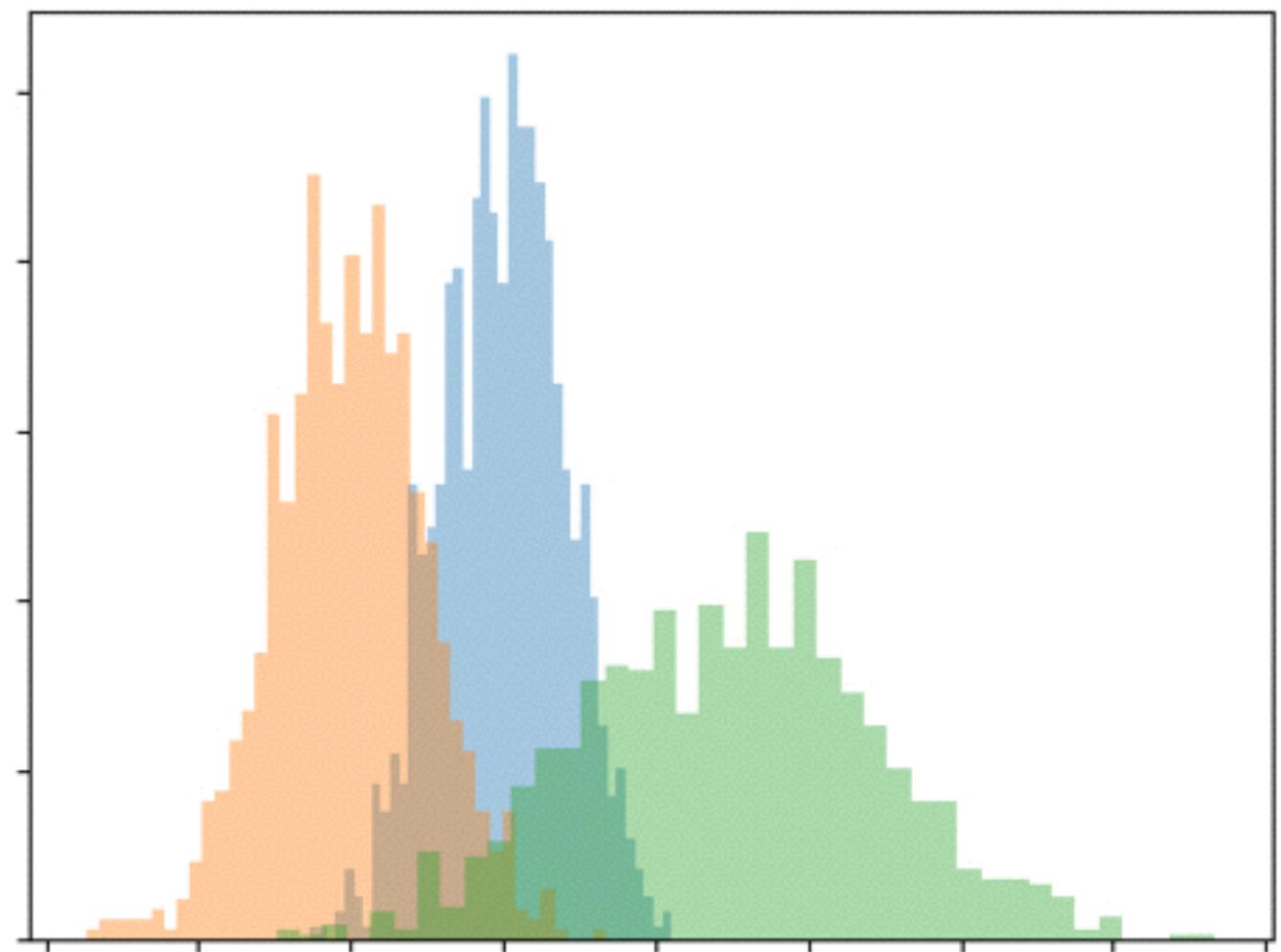


Histograms



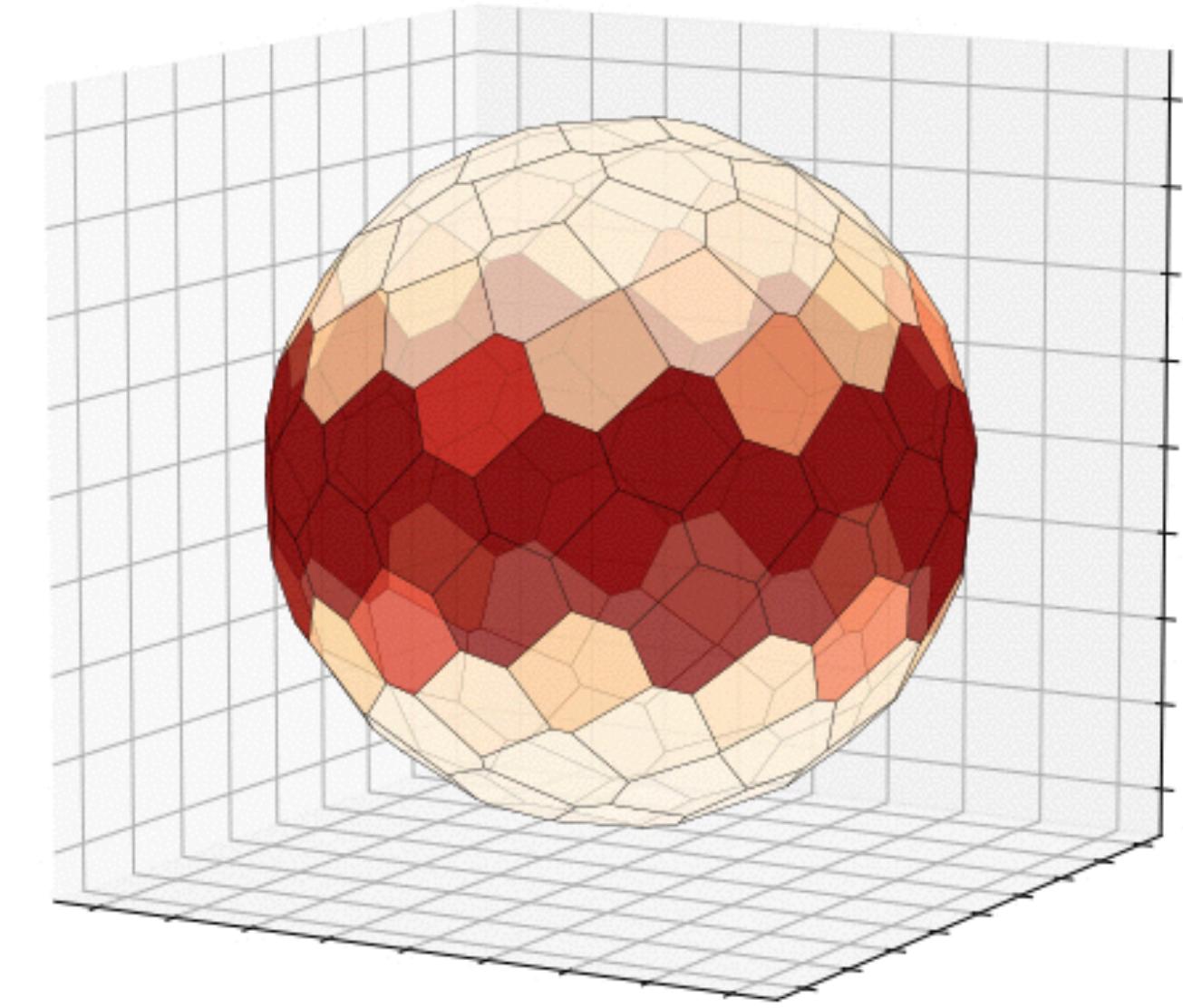
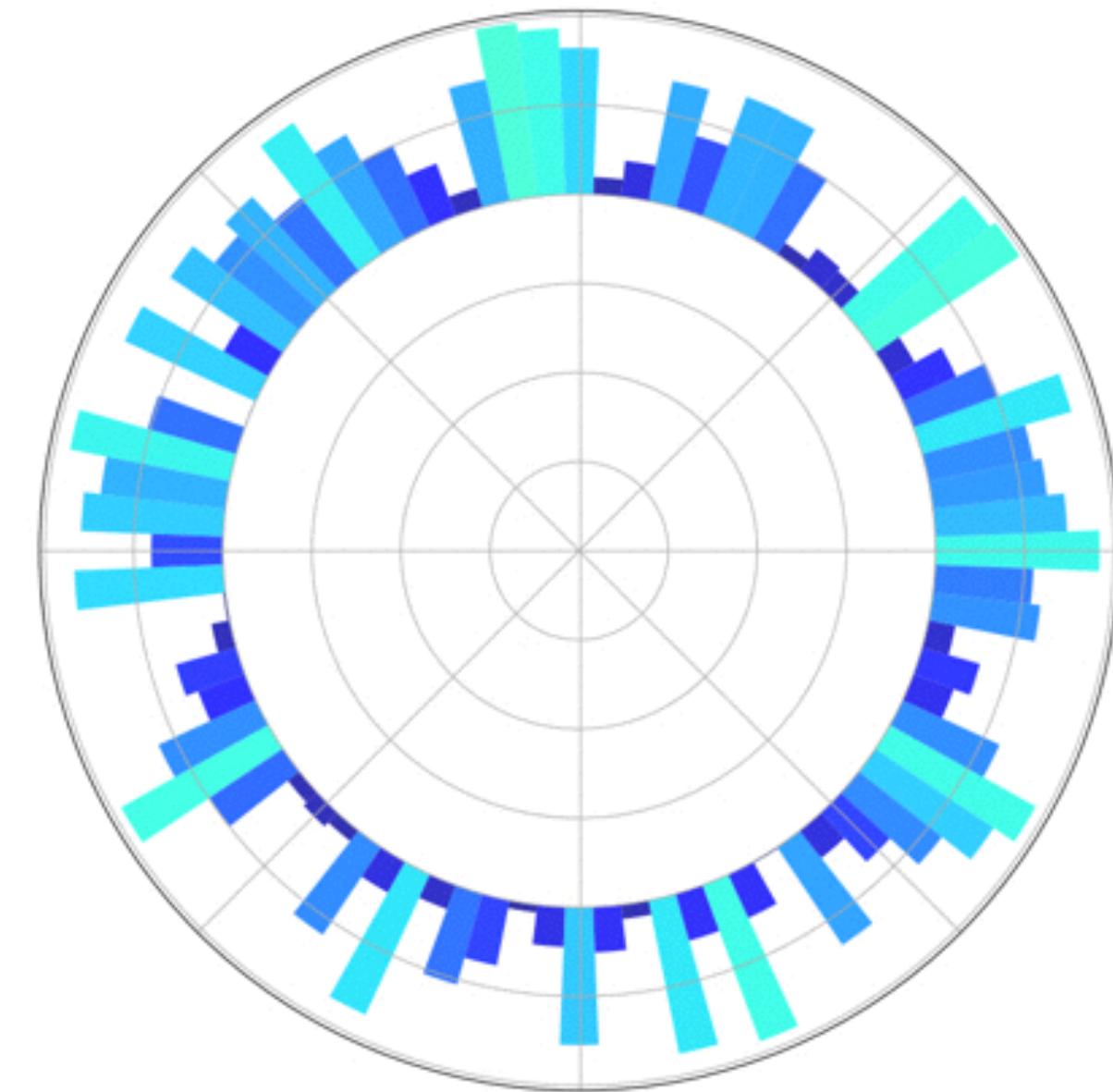
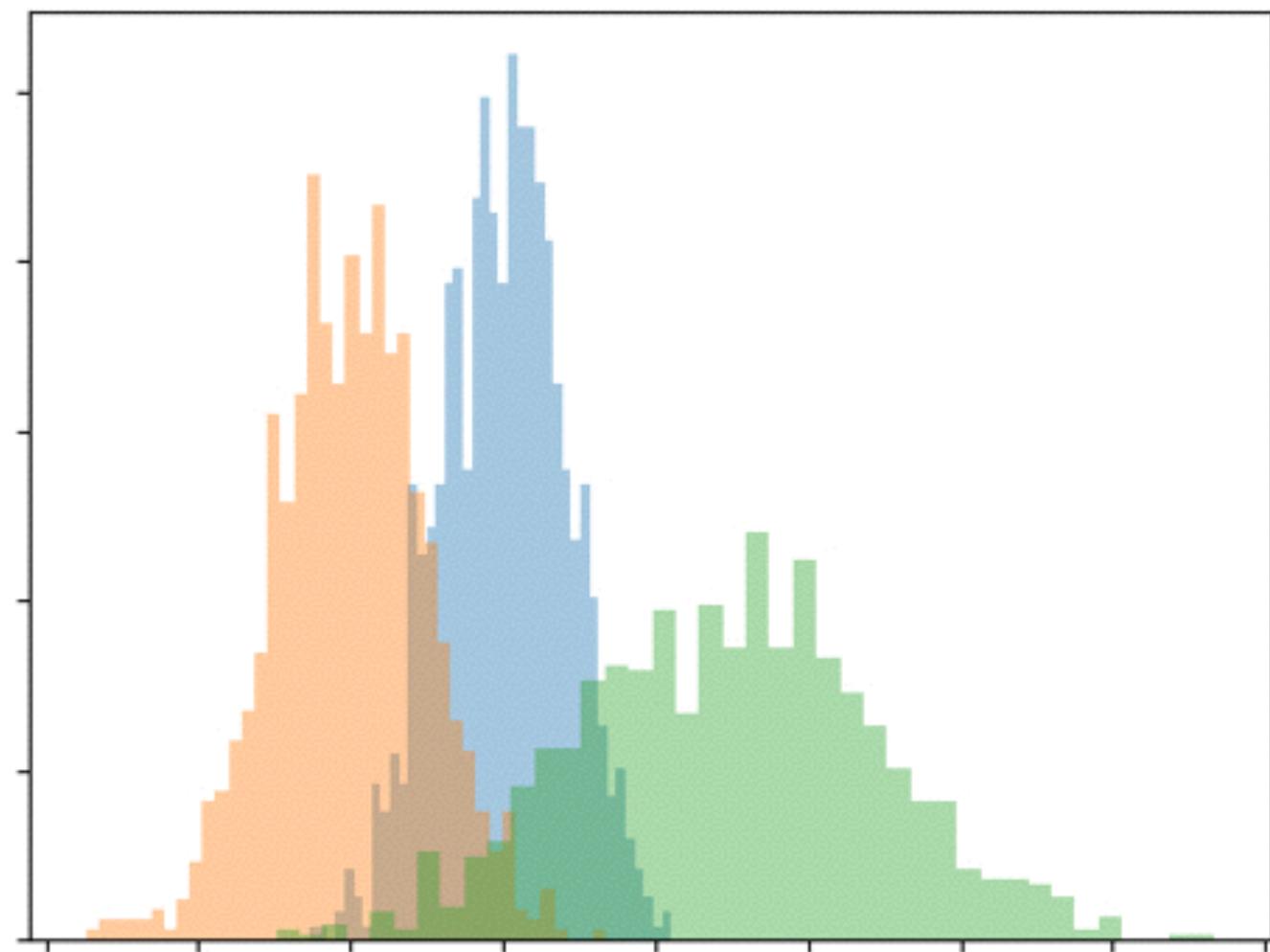
- Image editing

Histograms



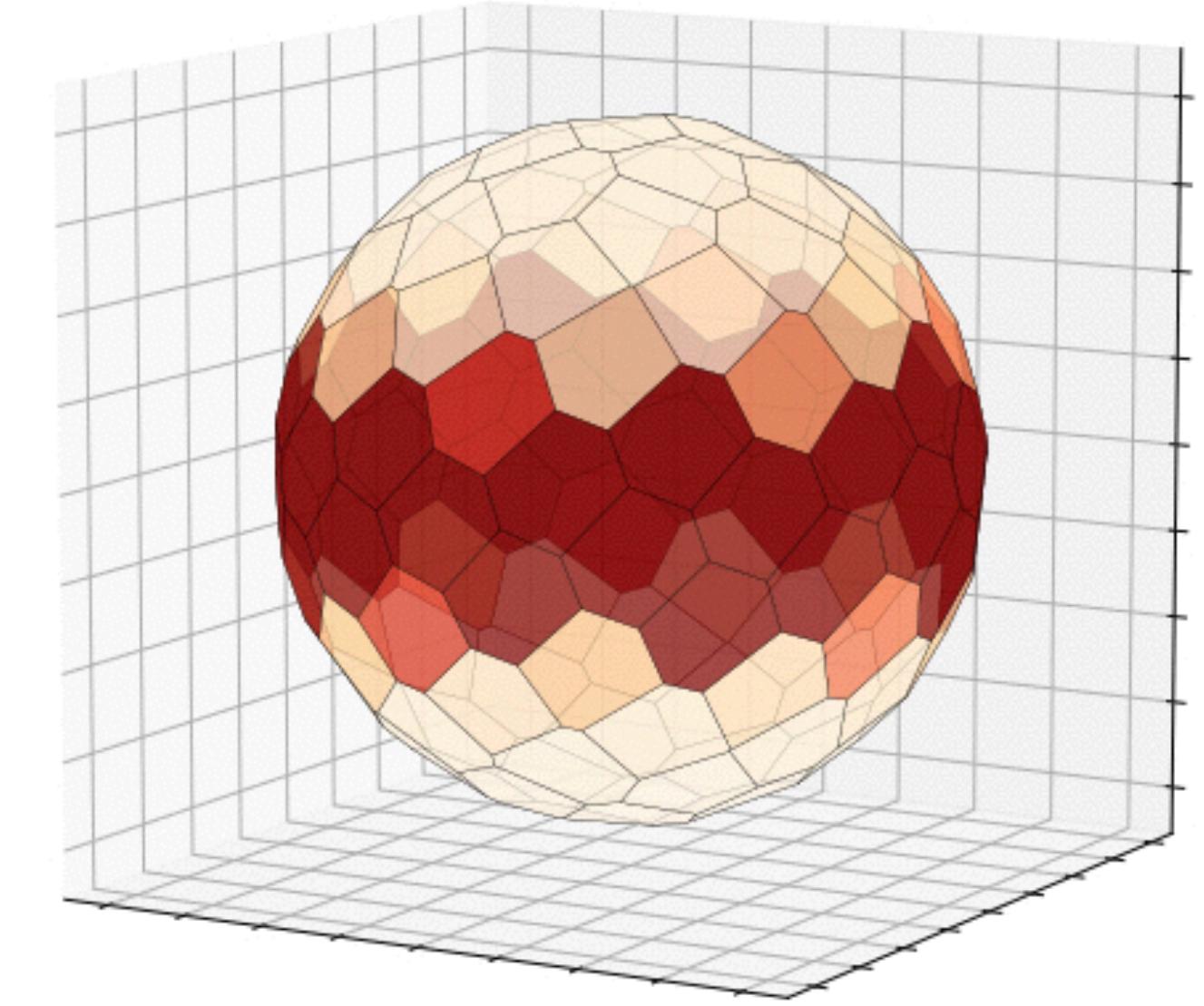
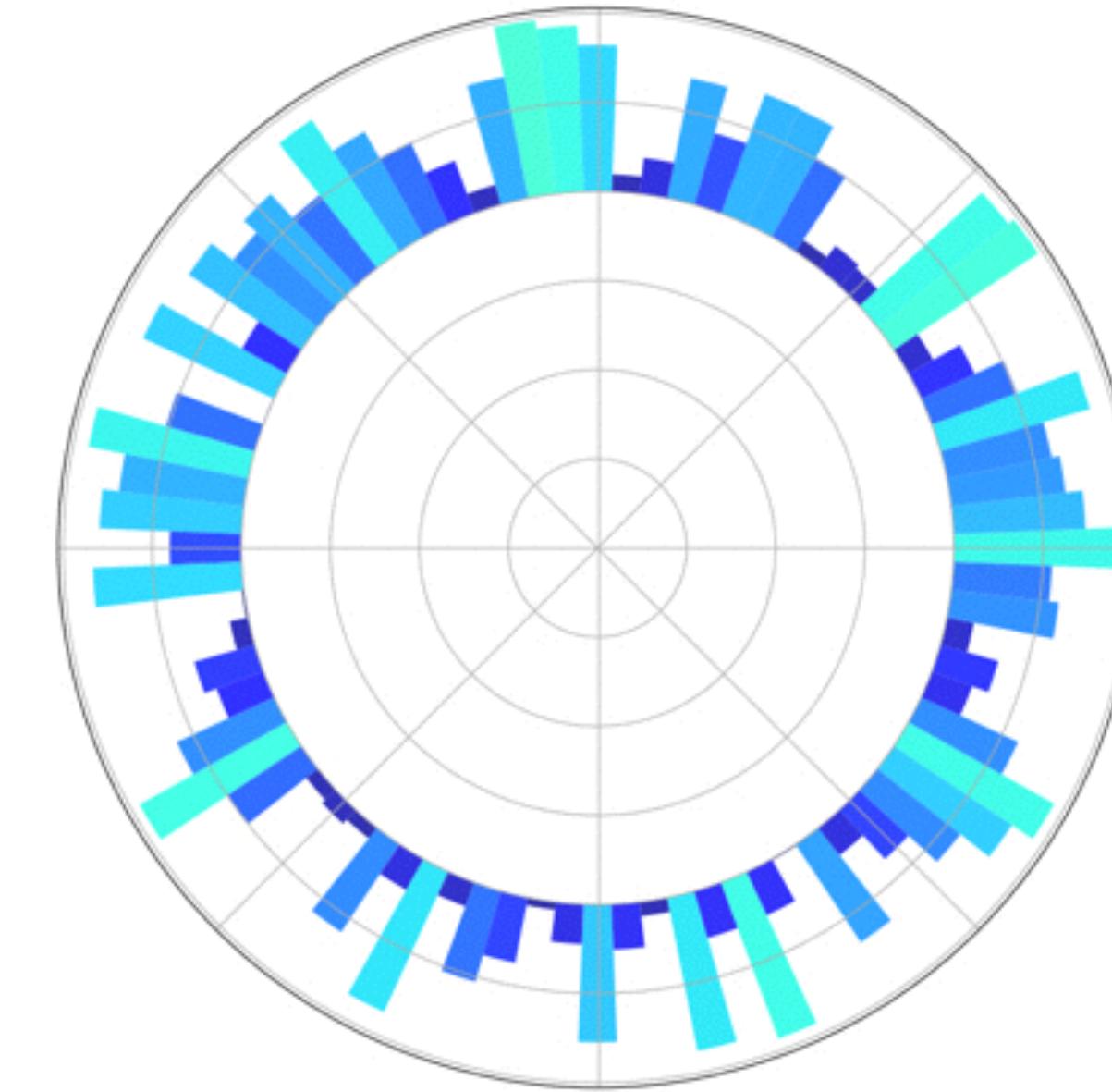
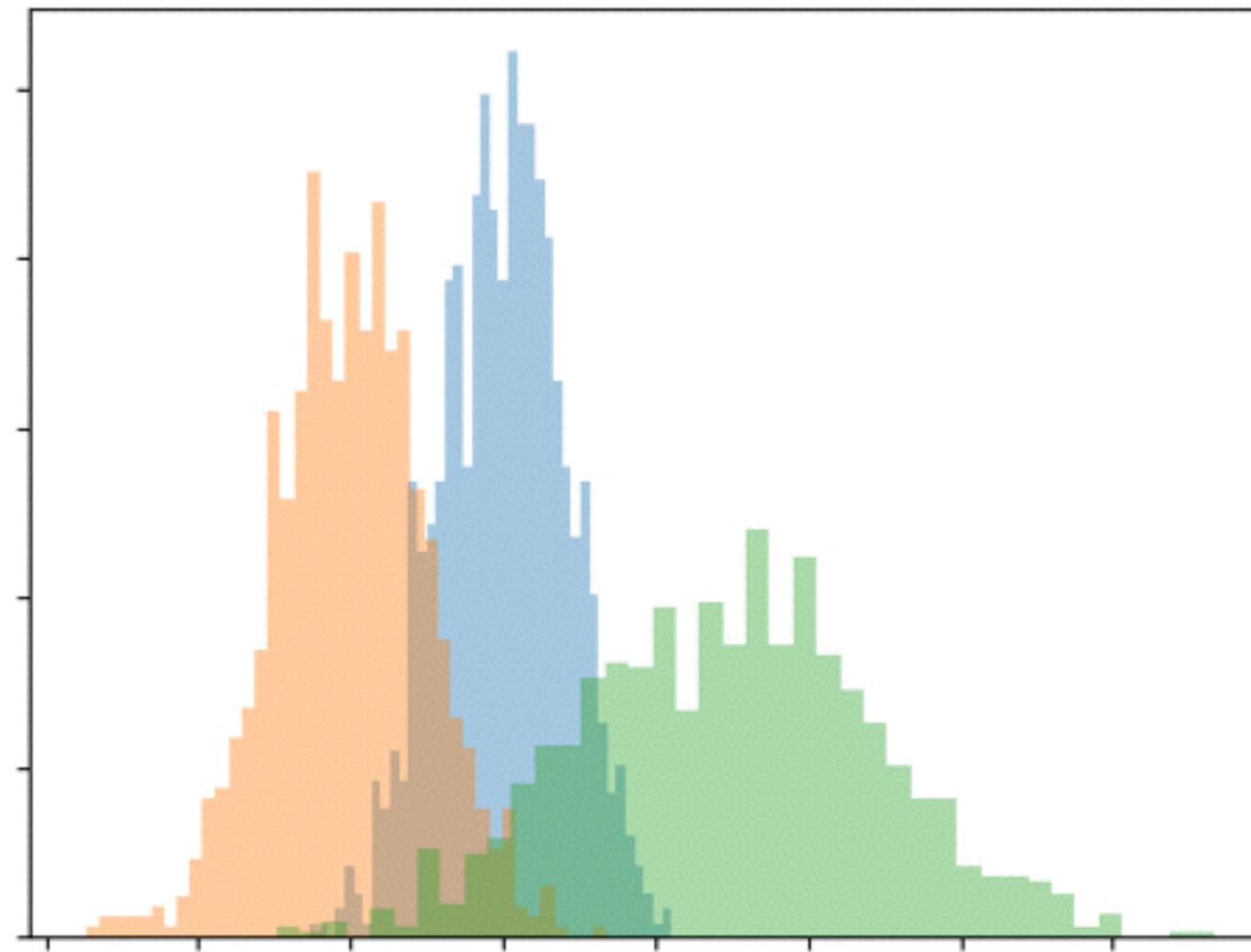
- Image editing
- Feature extraction

Histograms



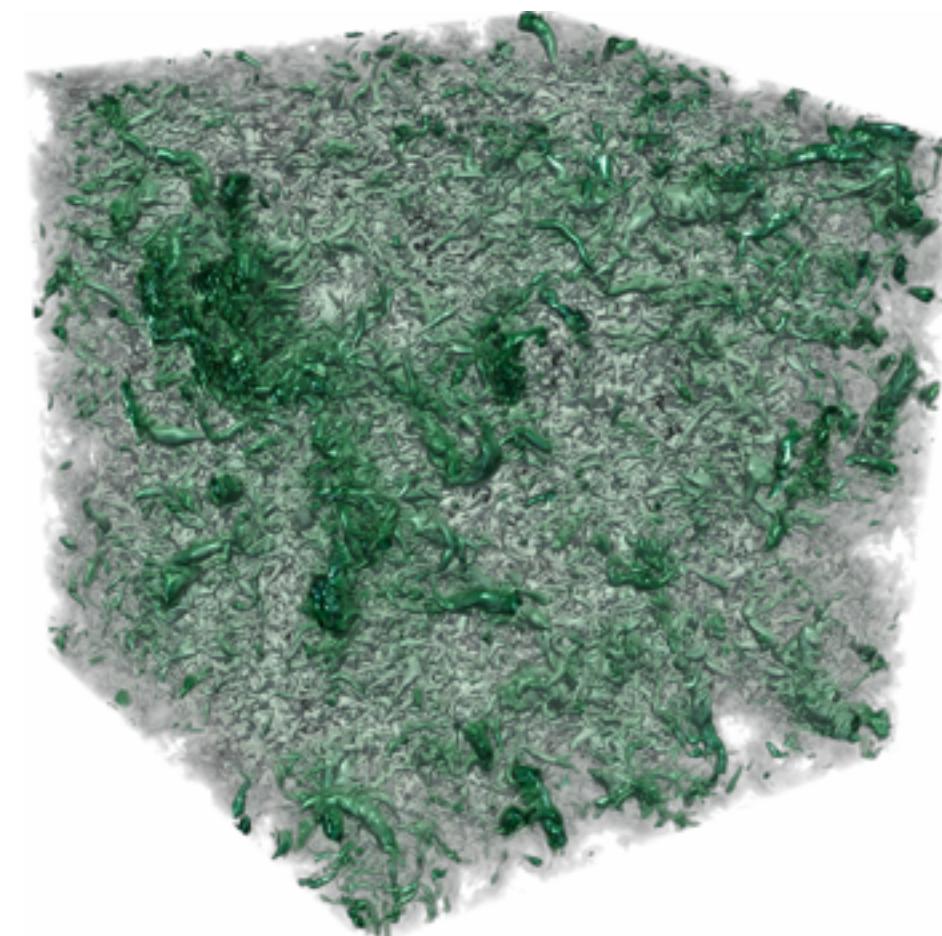
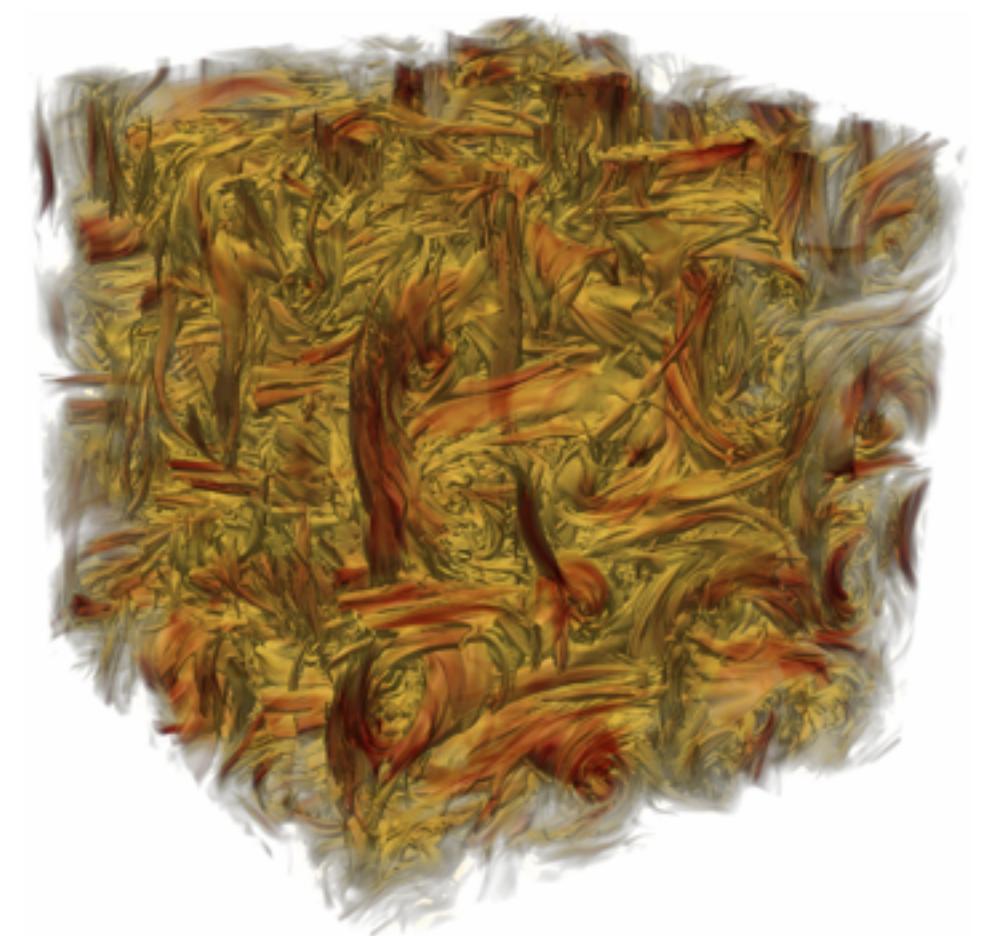
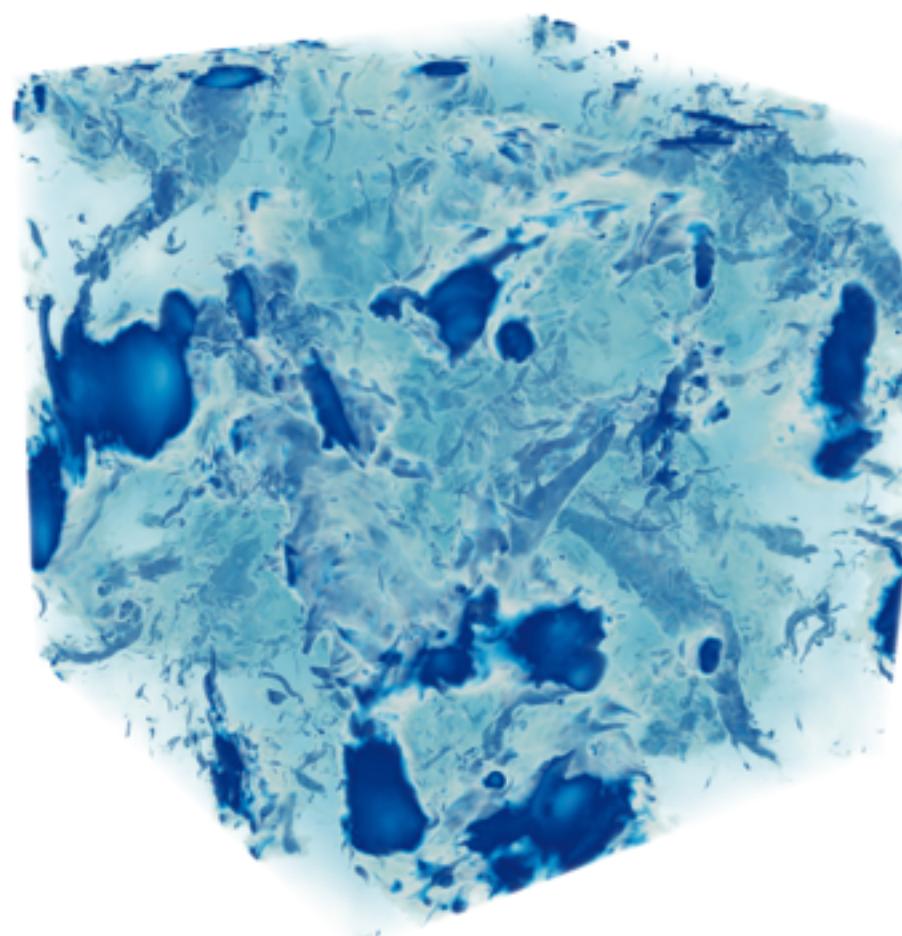
- Image editing
- Feature extraction
- Signal processing

Histograms

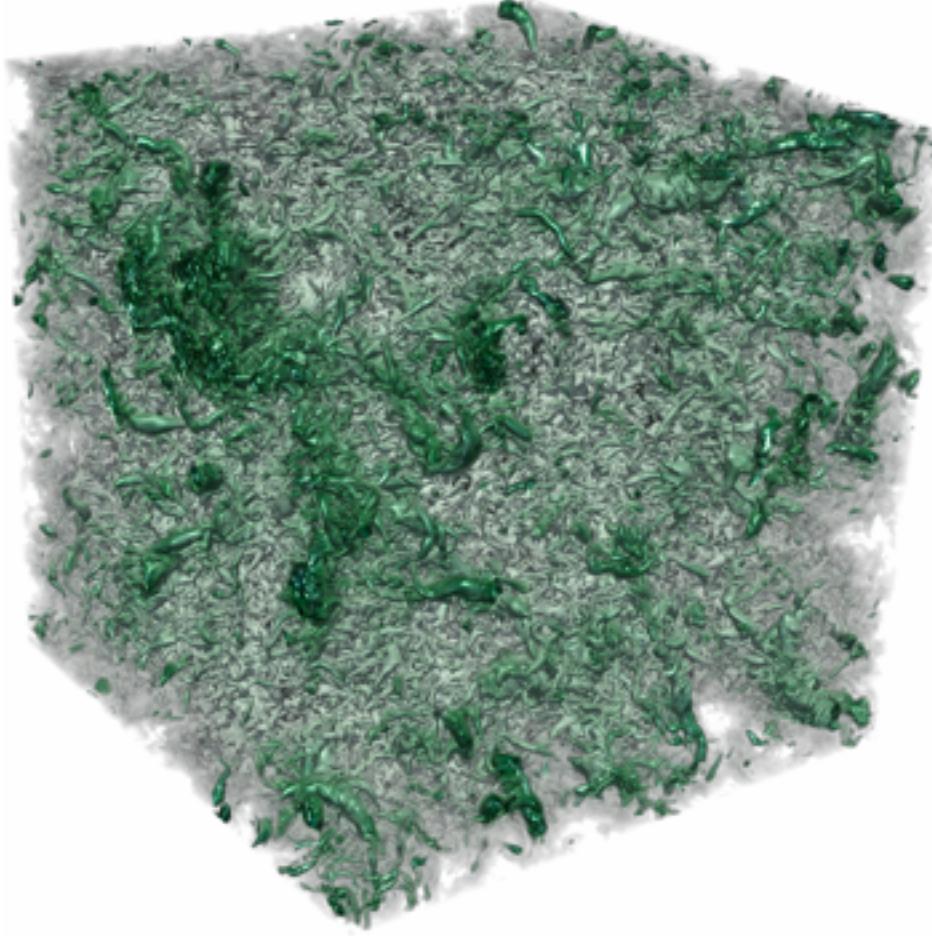
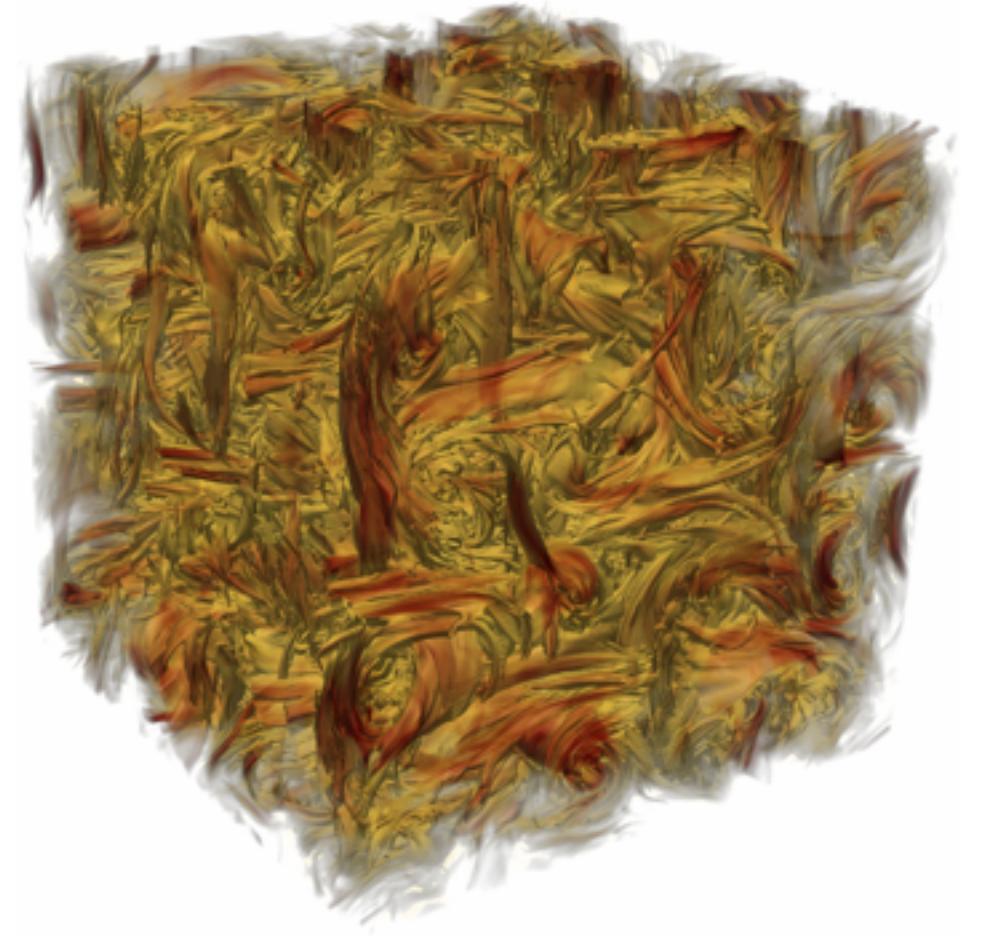
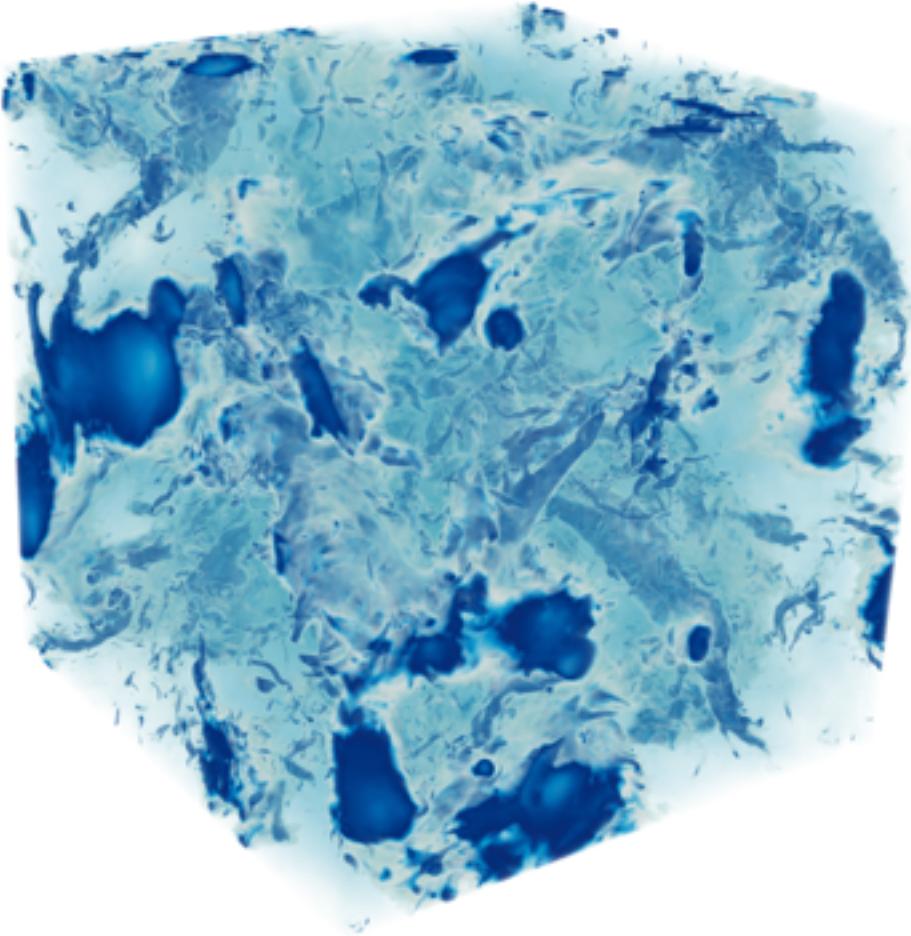


- Image editing
- Feature extraction
- Signal processing
- Computer vision

Large-scale Data Sets

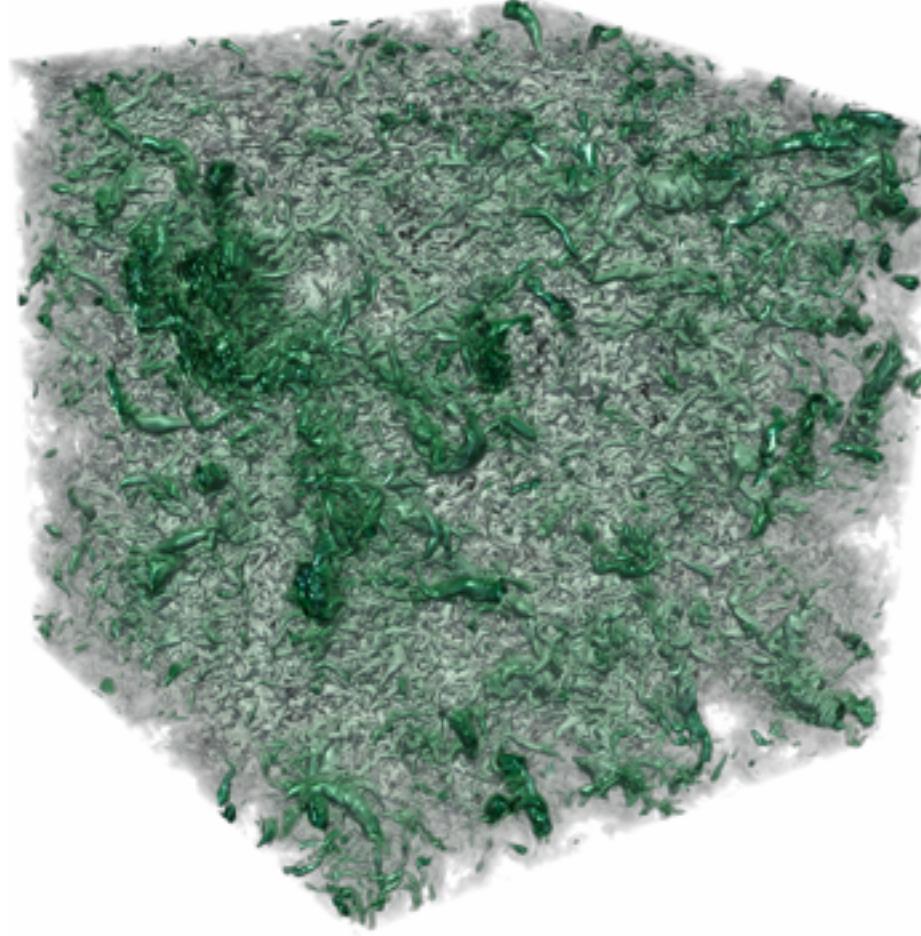
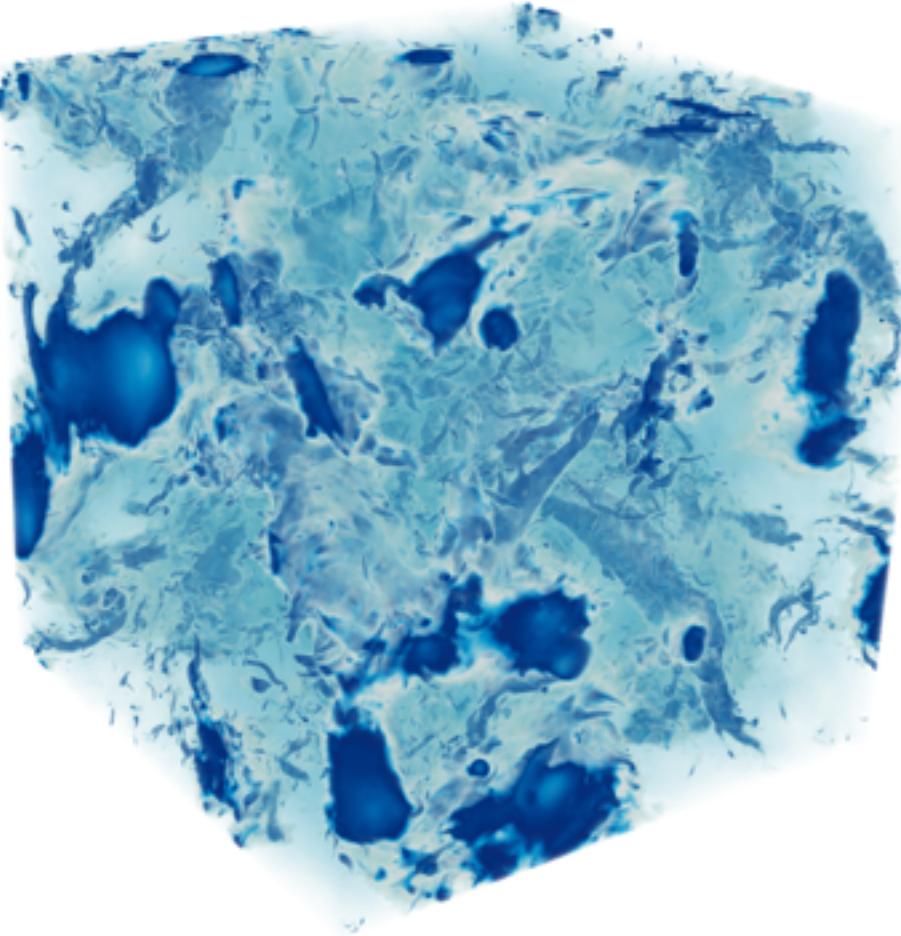


Large-scale Data Sets



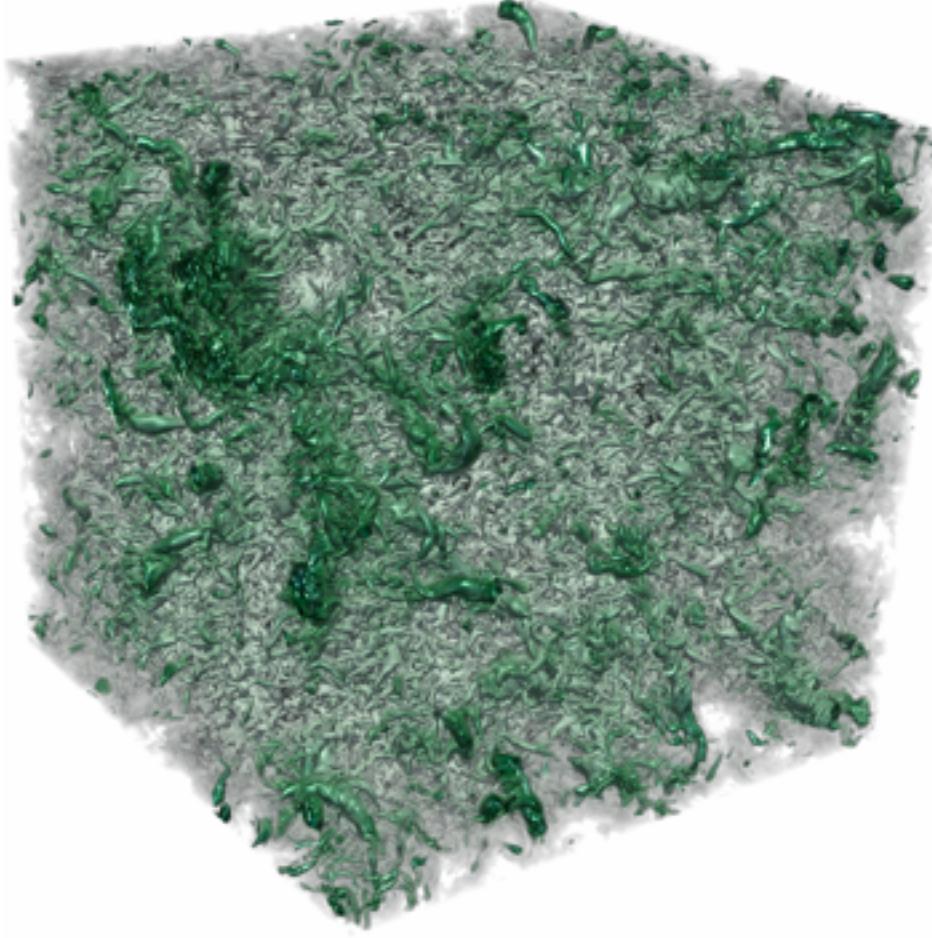
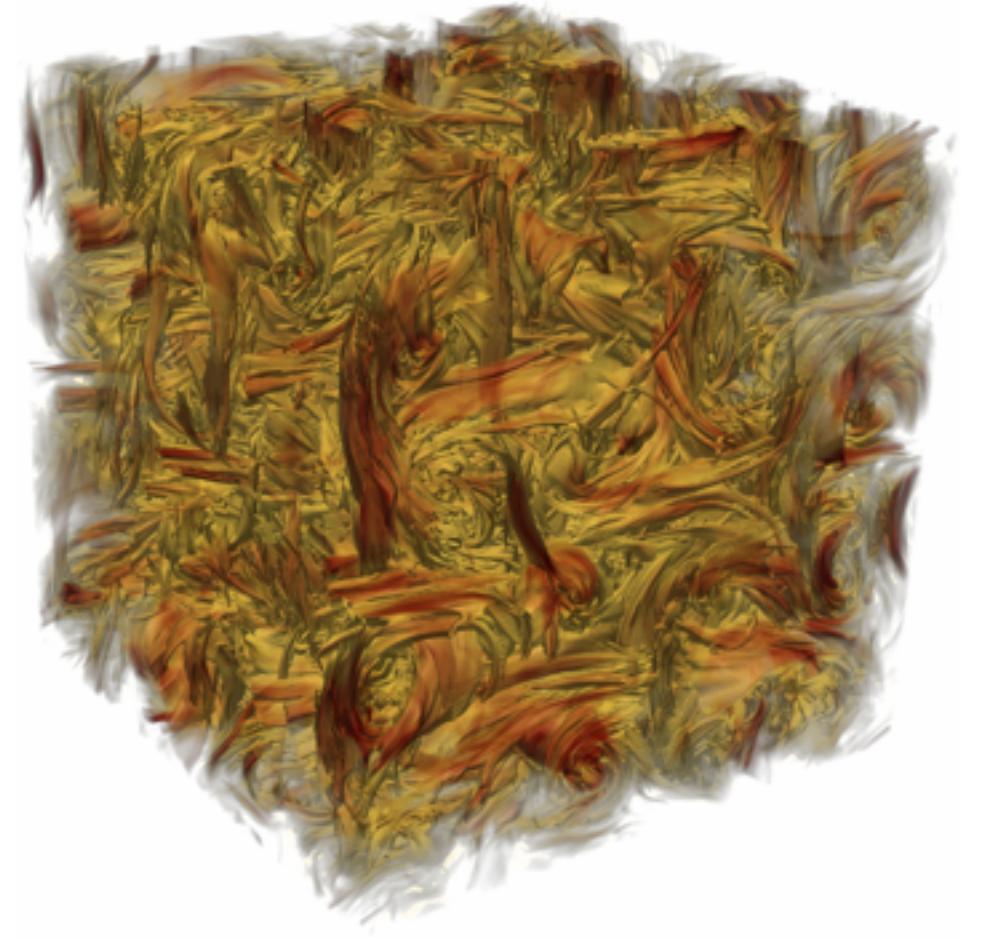
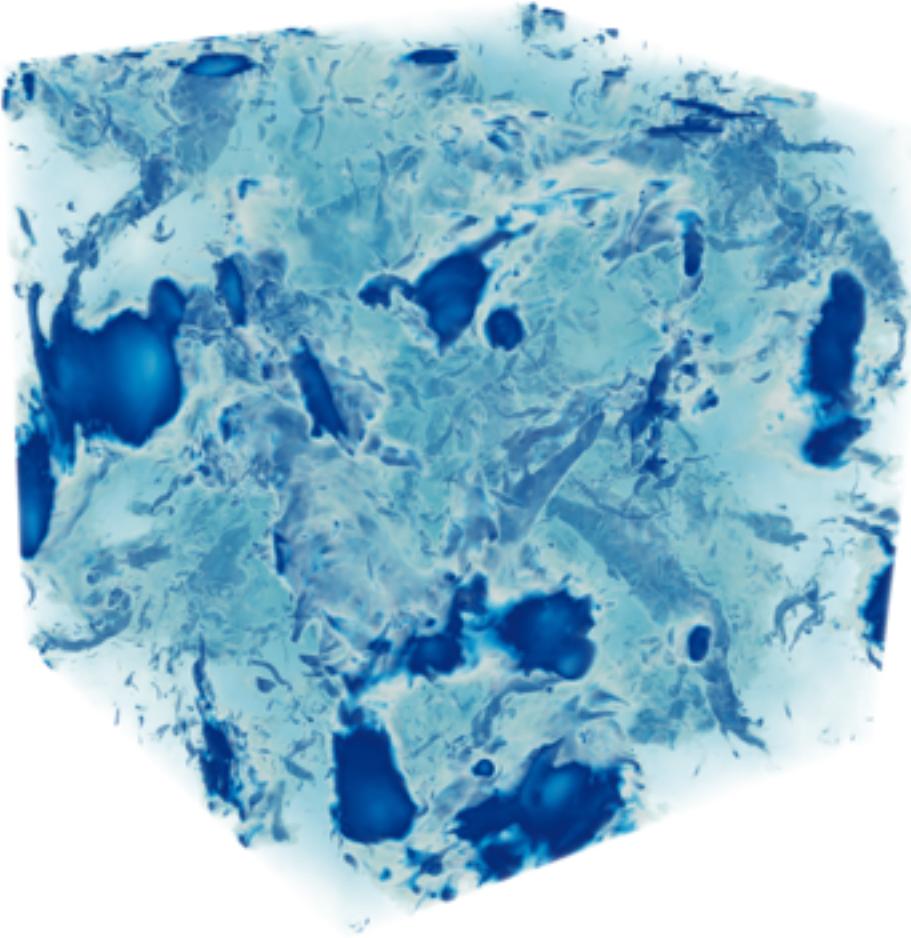
- Growing resolution:
 - ▶ More computing power
 - ▶ Better acquisition technology

Large-scale Data Sets



- **Growing resolution:**
 - ▶ More computing power
 - ▶ Better acquisition technology
- **Multiple dimensions**

Large-scale Data Sets



- Growing **resolution**:
 - ▶ More computing power
 - ▶ Better acquisition technology
- Multiple **dimensions**
- We focus on **regular grid data**

Brute-force Histograms

- CT scan: $1024^3 = 1$ billion voxels



Brute-force Histograms

- CT scan: $1024^3 = 1$ billion voxels
- Computing a **histogram over some region**: up to a billion memory accesses



Brute-force Histograms

- CT scan: $1024^3 = 1$ billion voxels
- Computing a **histogram over some region**: up to a billion memory accesses
- Data set fits in RAM/GPU...
 - ▶ ... but many accesses are **non-sequential**
 - ▶ In this example: up to 1 million



Brute-force Histograms

- CT scan: $1024^3 = 1$ billion voxels
- Computing a **histogram over some region**: up to a billion memory accesses
- Data set fits in RAM/GPU...
 - ▶ ... but many accesses are **non-sequential**
 - ▶ In this example: up to 1 million
- Typical times: 1-50ms



Brute-force Histograms

- CT scan: $1024^3 = 1$ billion voxels
- Computing a **histogram over some region**: up to a billion memory accesses
- Data set fits in RAM/GPU...
 - ▶ ... but many accesses are **non-sequential**
 - ▶ In this example: up to 1 million
- Typical times: 1-50ms
- What if we need **many histograms**?
 - ▶ E.g. over 1K regions → **no longer interactive**



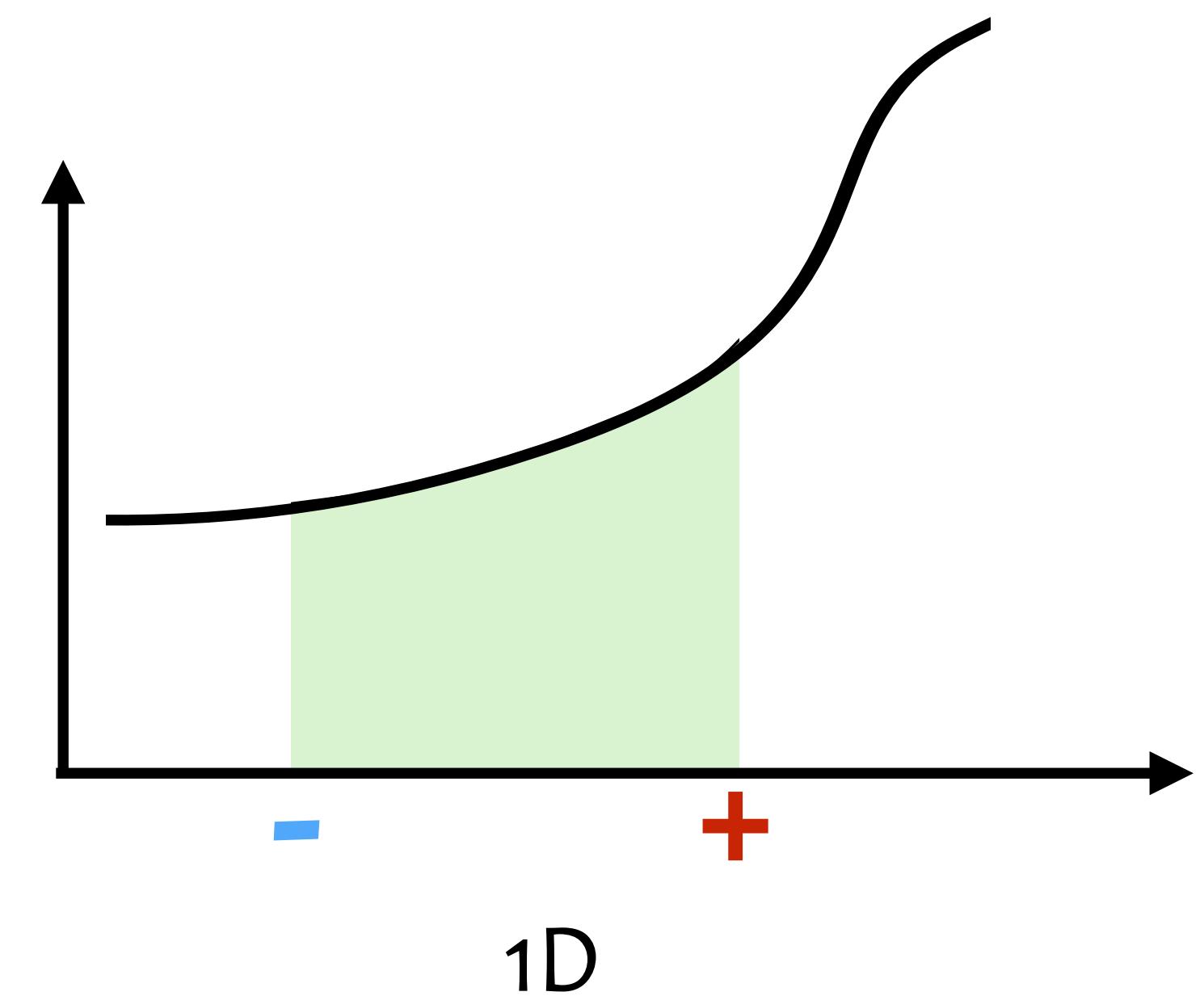
Integral Histogram

- Data structure to **speed-up** histogram computation [Porikli '05]
 - ▶ E.g. million memory accesses → few hundreds

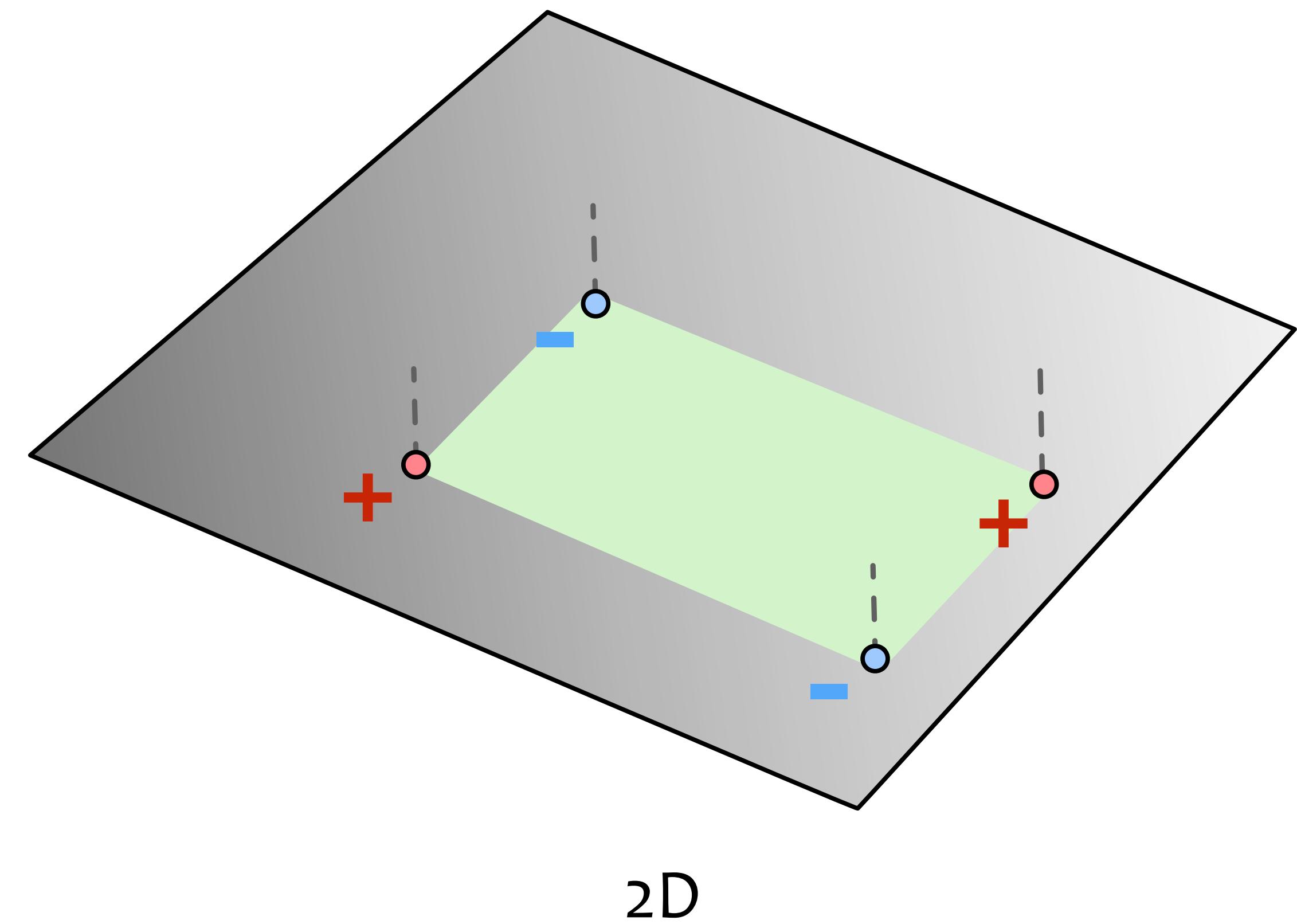
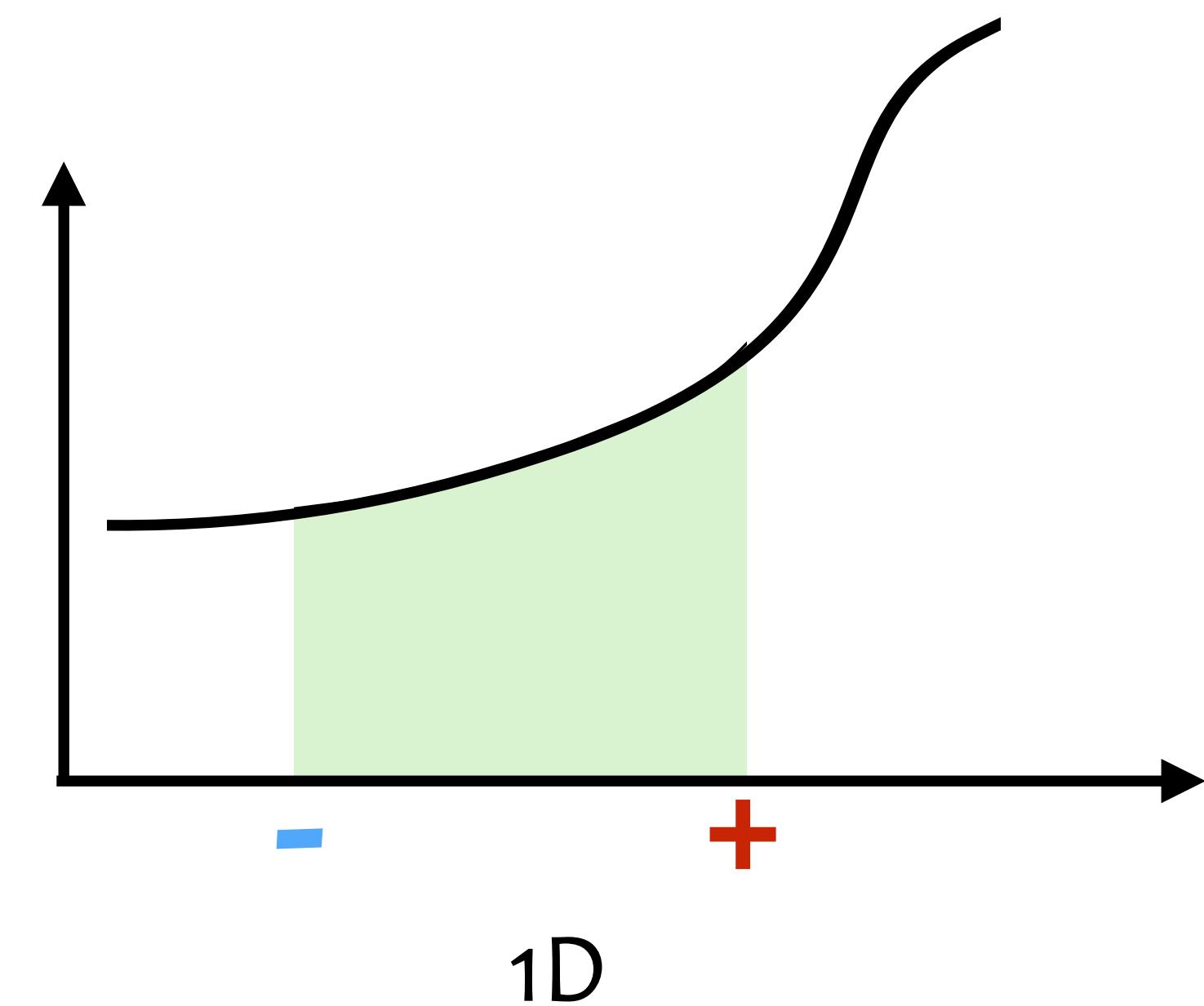
Integral Histogram

- Data structure to **speed-up** histogram computation [Porikli '05]
 - ▶ E.g. million memory accesses → few hundreds
- Based on summed area tables (SAT)
 - ▶ One per bin $b = 1, \dots, B$

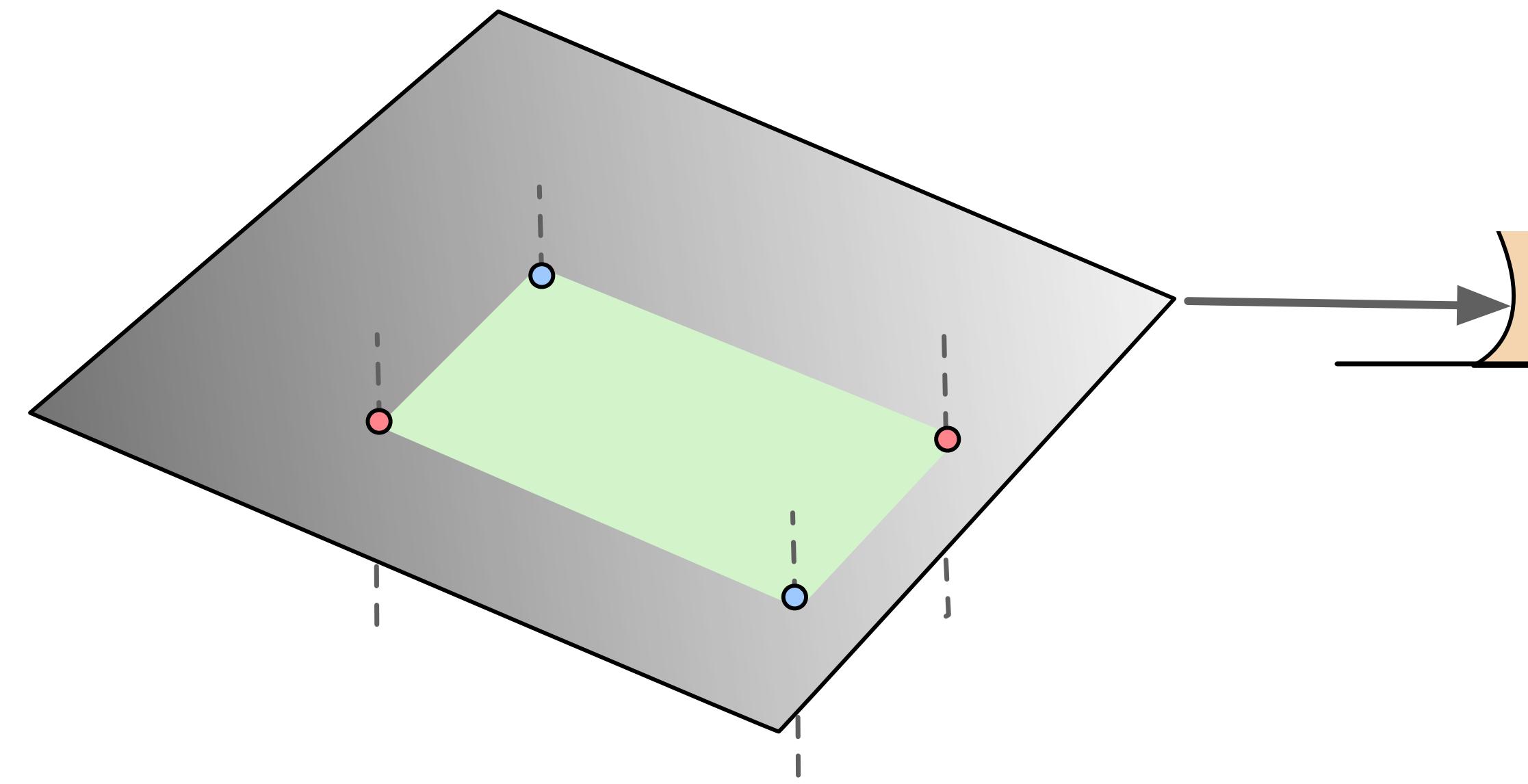
Summed Area Tables



Summed Area Tables

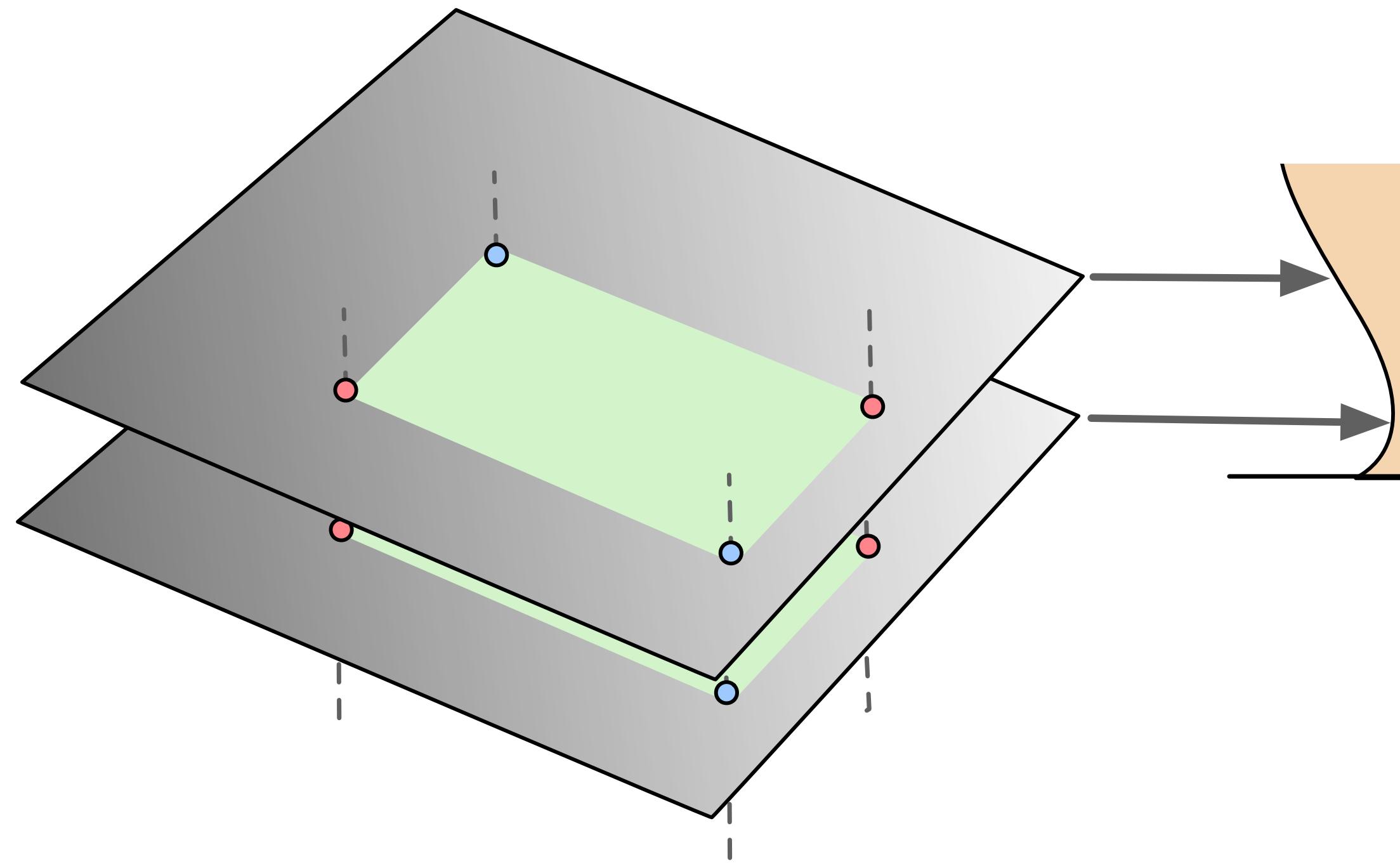


Integral Histogram



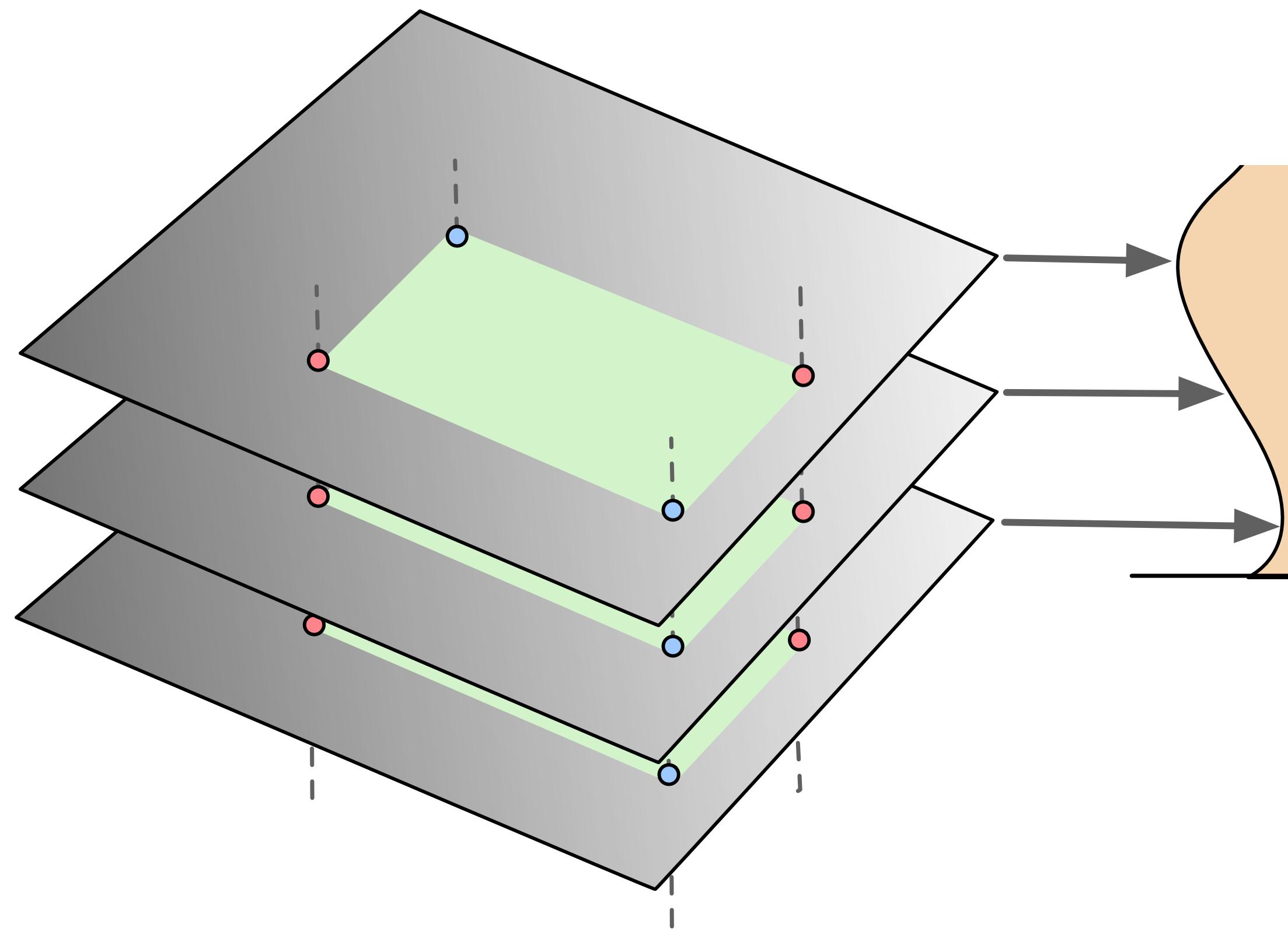
[Porikli '05]

Integral Histogram



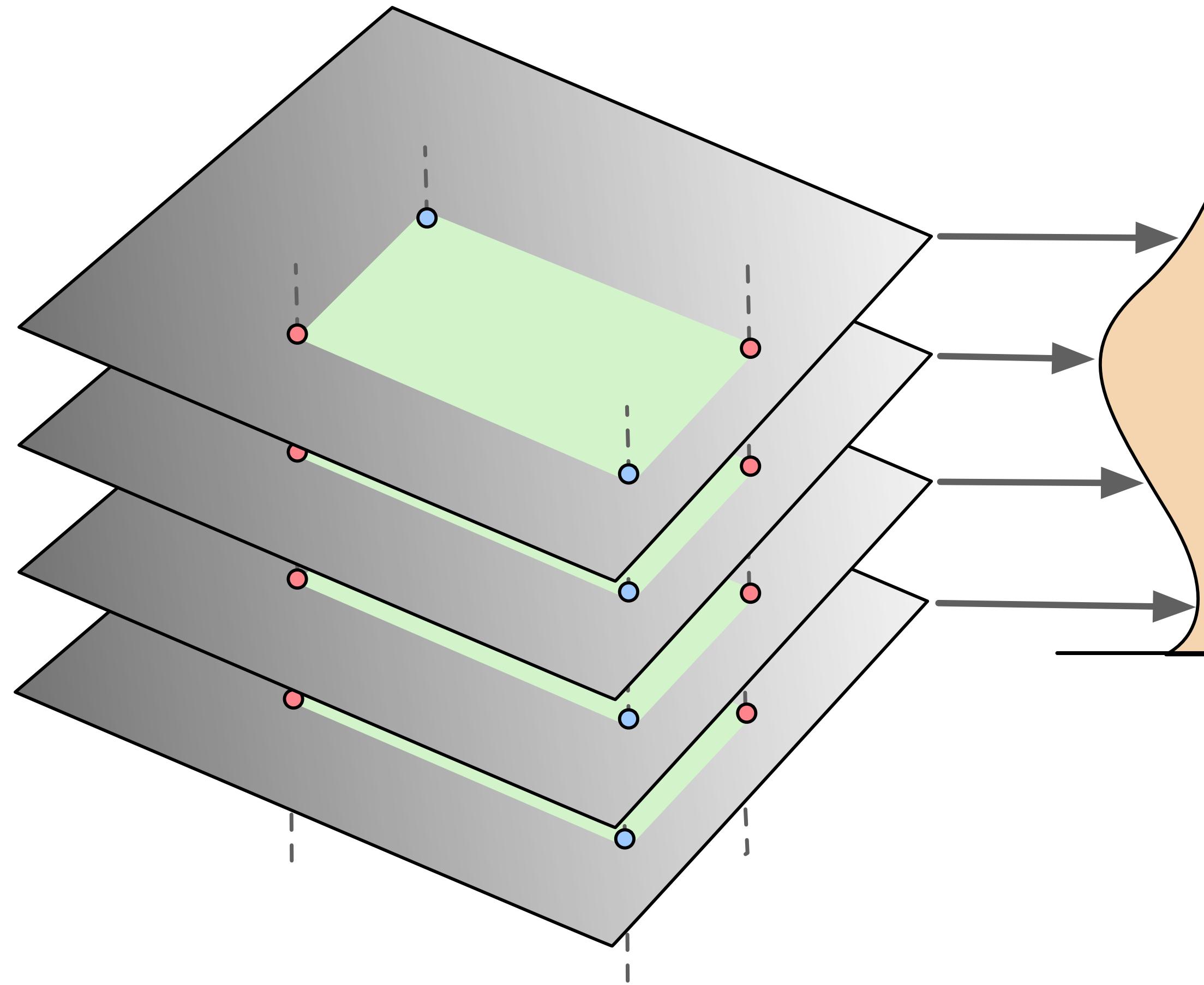
[Porikli '05]

Integral Histogram



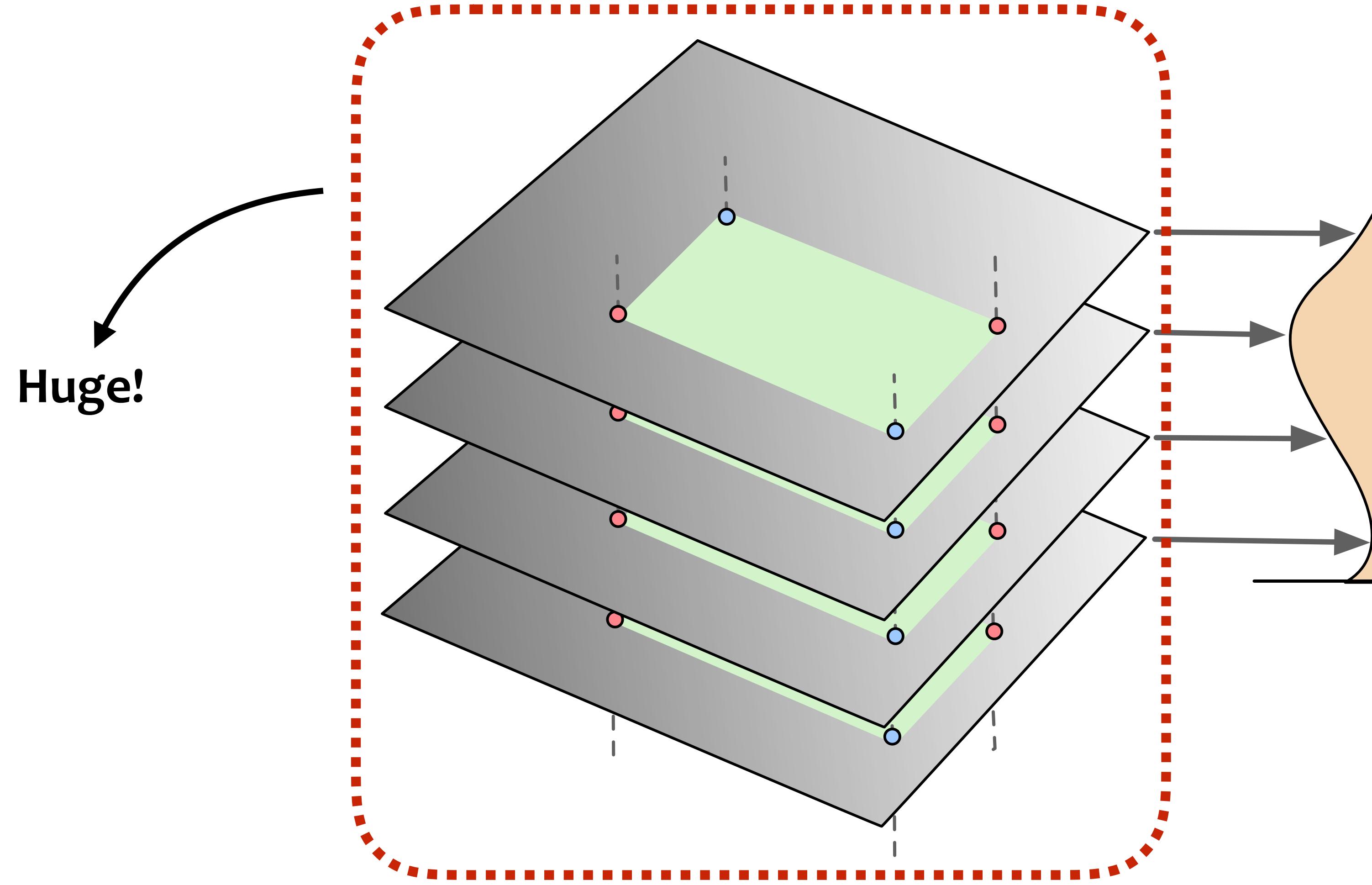
[Porikli '05]

Integral Histogram



[Porikli '05]

Integral Histogram



Integral Histogram: Drawbacks

- Trade-off: higher speed, but **gigantic storage cost**
 - ▶ E.g. 1GB and 128 bins → 512GB
 - ▶ Won't fit in GPU or even main memory
 - ▶ Must use disk, still **prohibitive**

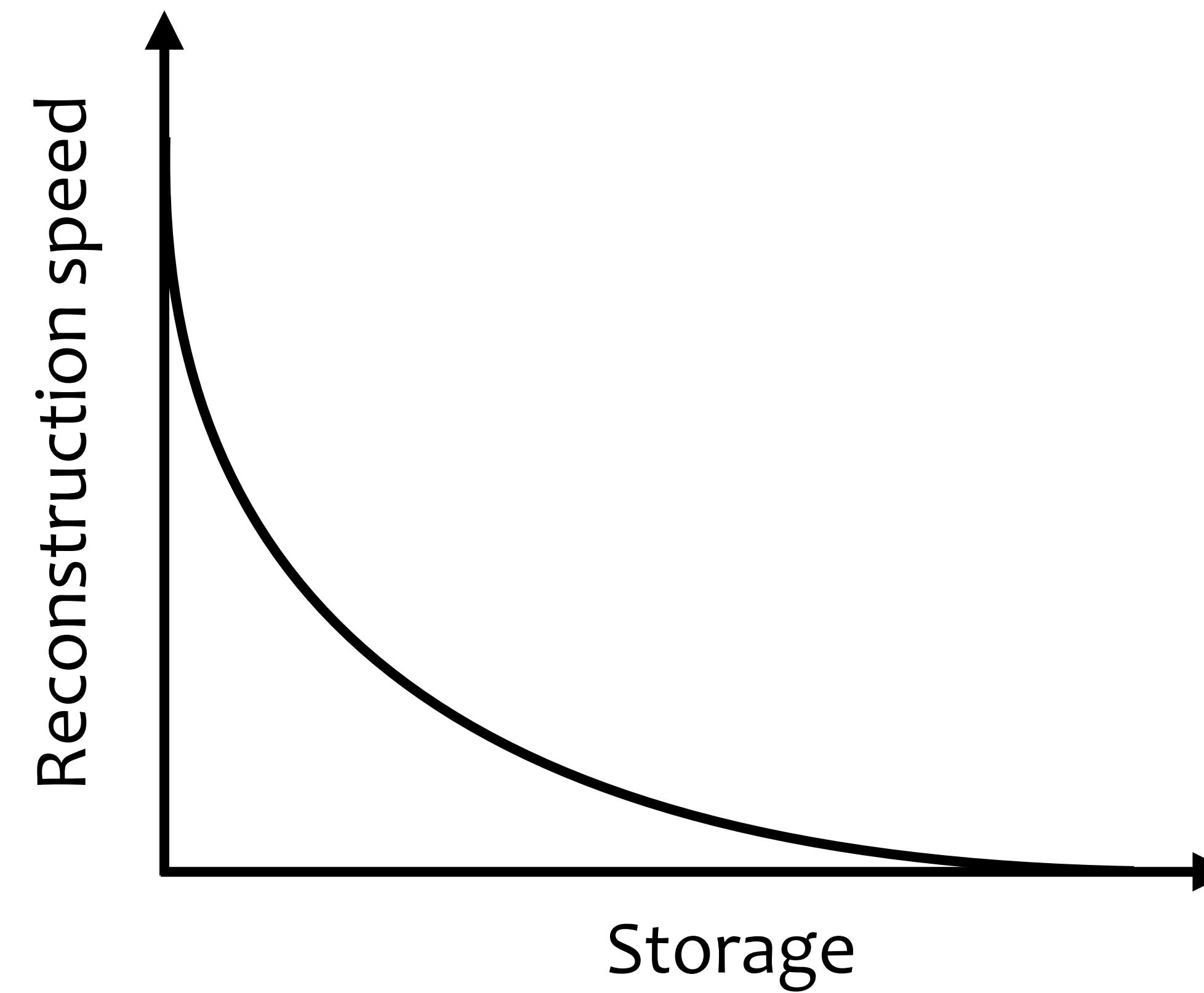
Integral Histogram: Drawbacks

- Trade-off: higher speed, but **gigantic storage cost**
 - ▶ E.g. 1GB and 128 bins → 512GB
 - ▶ Won't fit in GPU or even main memory
 - ▶ Must use disk, still **prohibitive**
- Massively redundant!

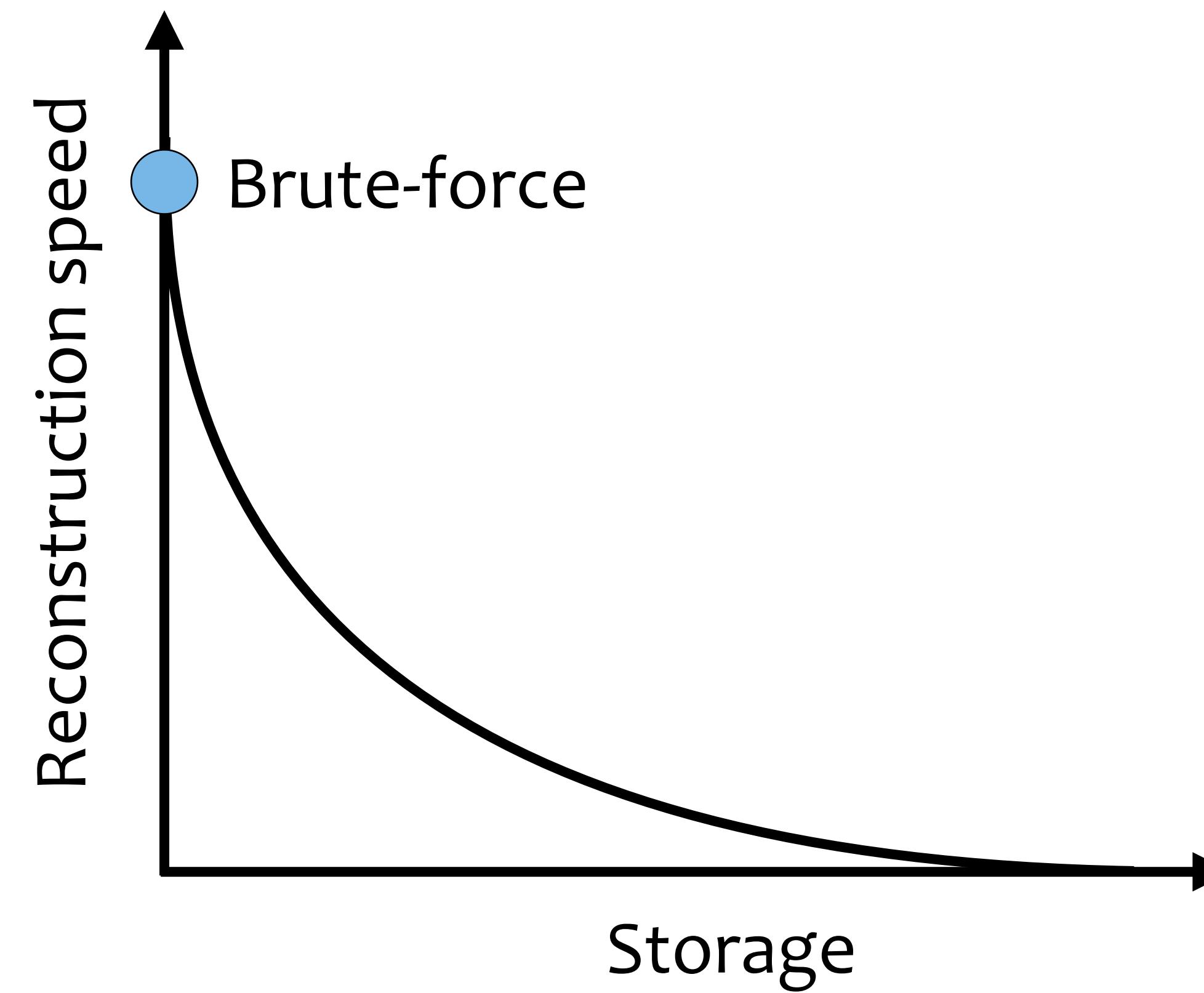
Integral Histogram: Drawbacks

- Trade-off: higher speed, but **gigantic storage cost**
 - ▶ E.g. 1GB and 128 bins → 512GB
 - ▶ Won't fit in GPU or even main memory
 - ▶ Must use disk, still **prohibitive**
- Massively redundant!
- Only good for rectangular regions

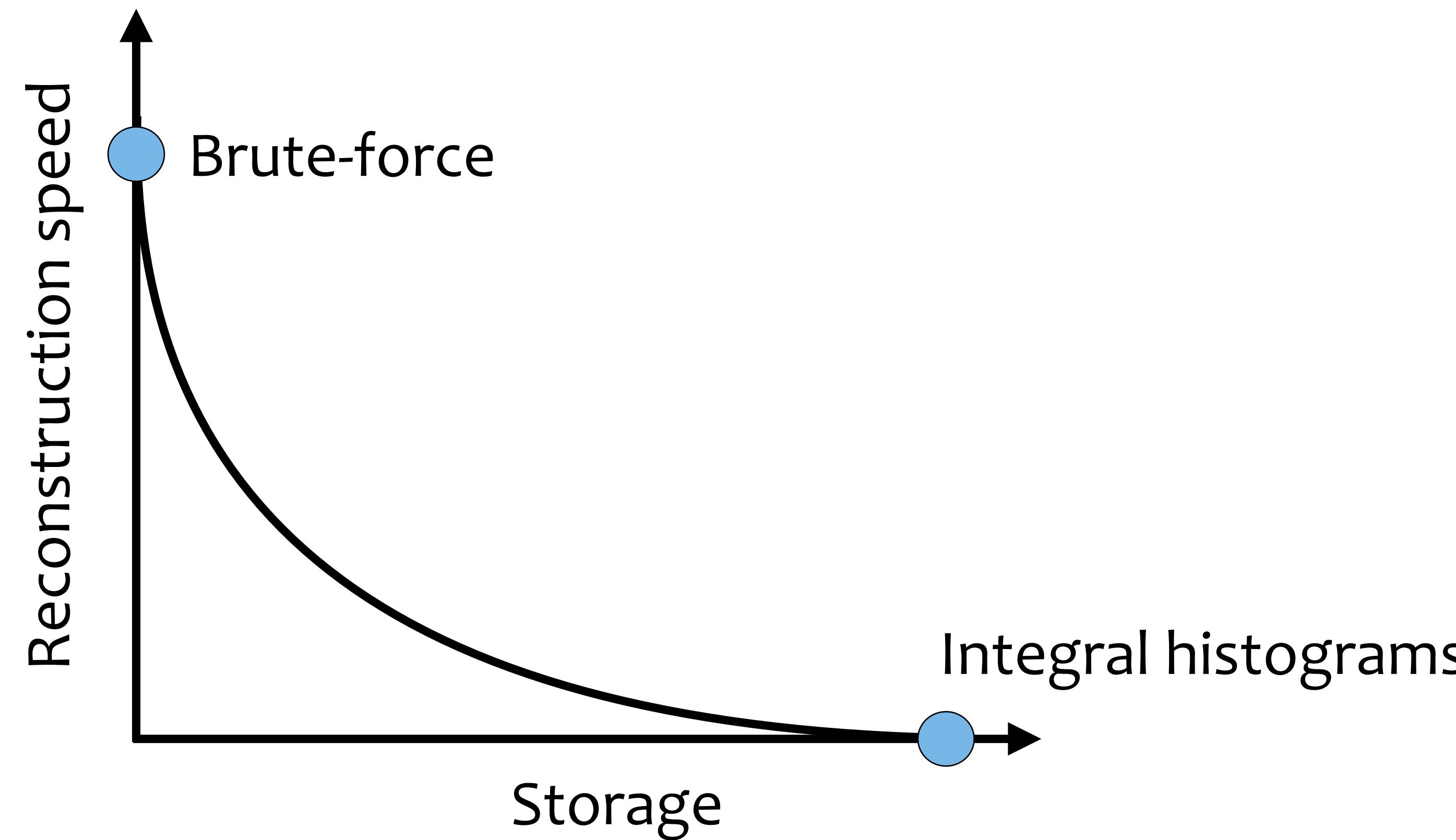
Integral Histogram: Compression



Integral Histogram: Compression

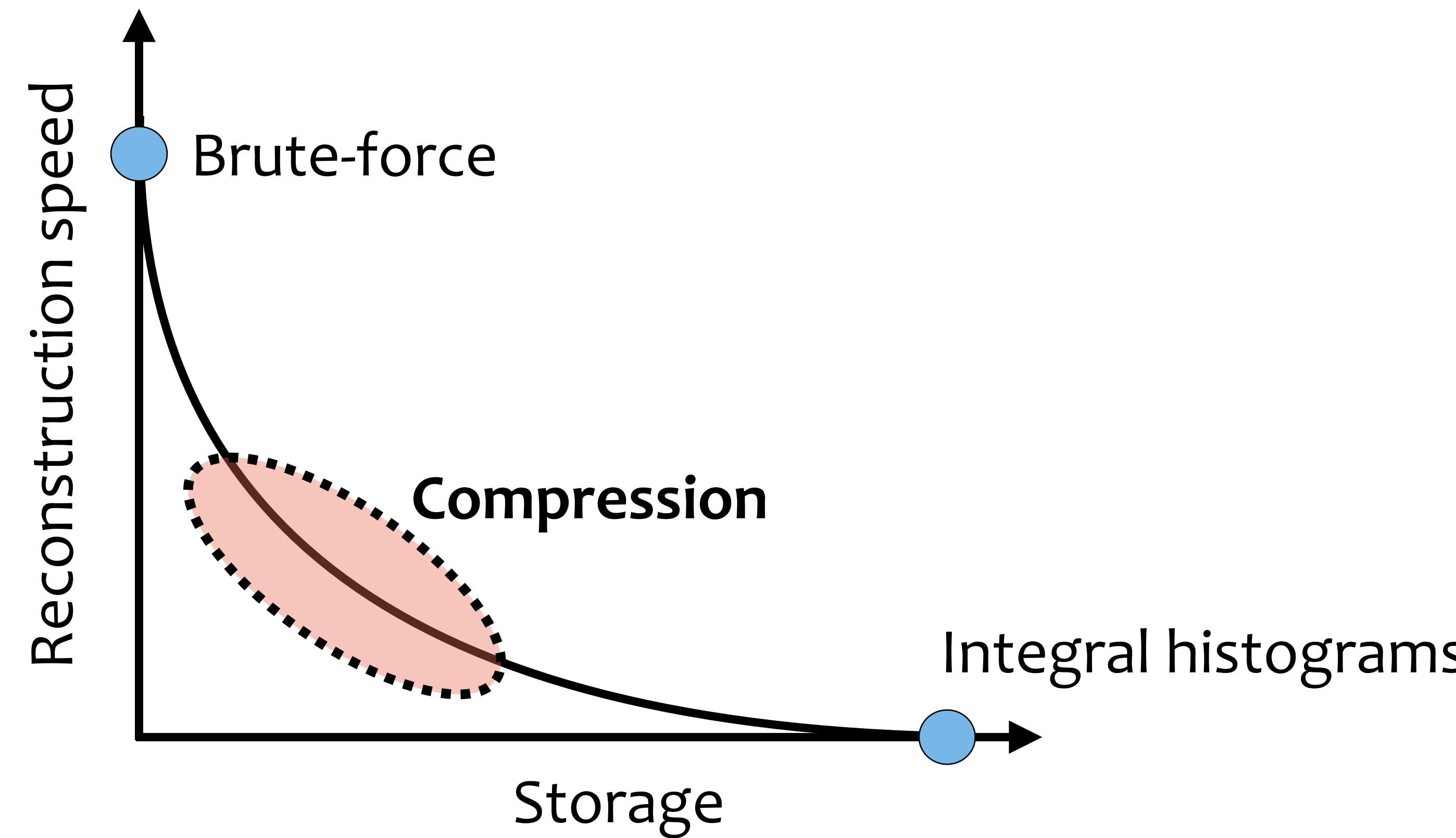


Integral Histogram: Compression



Integral Histogram: Compression

Let's compress the integral histogram



Integral Histogram: Compression

Let's compress the integral histogram

Integral Histogram: Compression

Let's compress the integral histogram

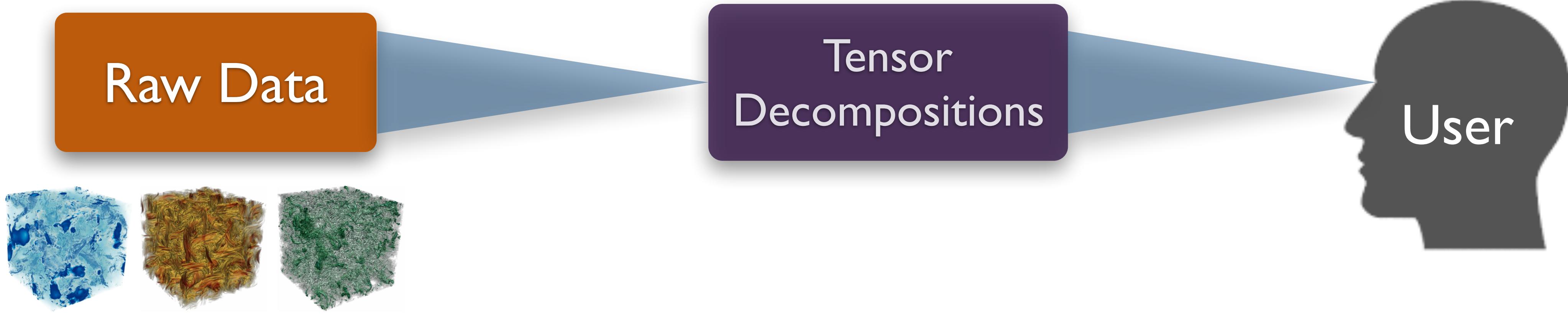
- Challenges:
 - ▶ Compressed IH **should fit in memory**
 - ▶ Decompression should be **faster** than brute-force histogram computation

Integral Histogram: Compression

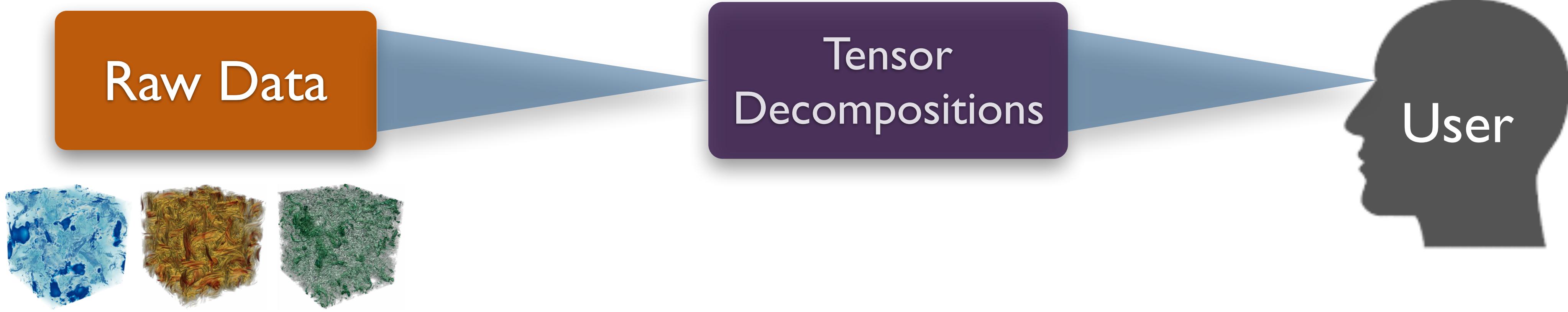
Let's compress the integral histogram

- Challenges:
 - ▶ Compressed IH **should fit in memory**
 - ▶ Decompression should be **faster** than brute-force histogram computation
- Example: Wavelet-SAT [Lee & Shen '13]
 - ▶ Fast IH decompression using wavelets
 - ▶ Lossless → moderate compression ratios
 - ▶ Only rectangular regions

Tensor Decompositions

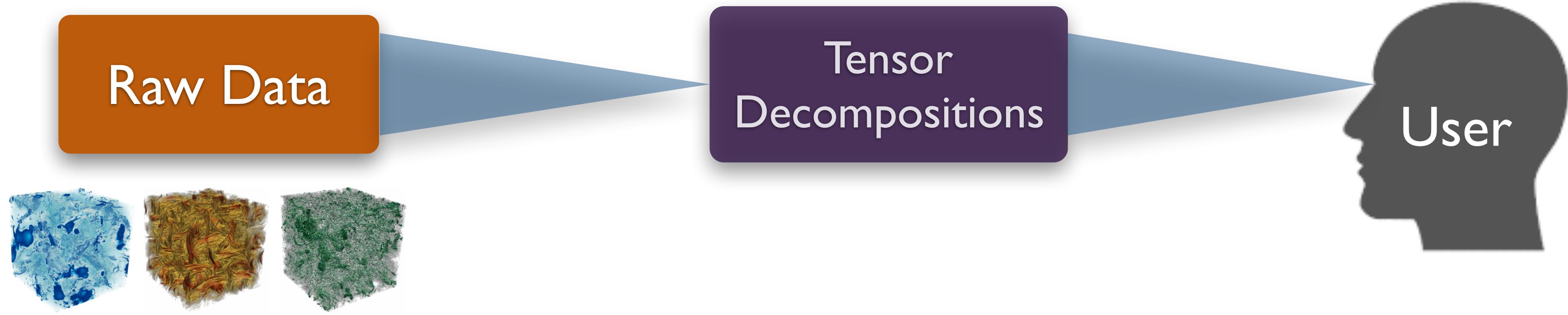


Tensor Decompositions



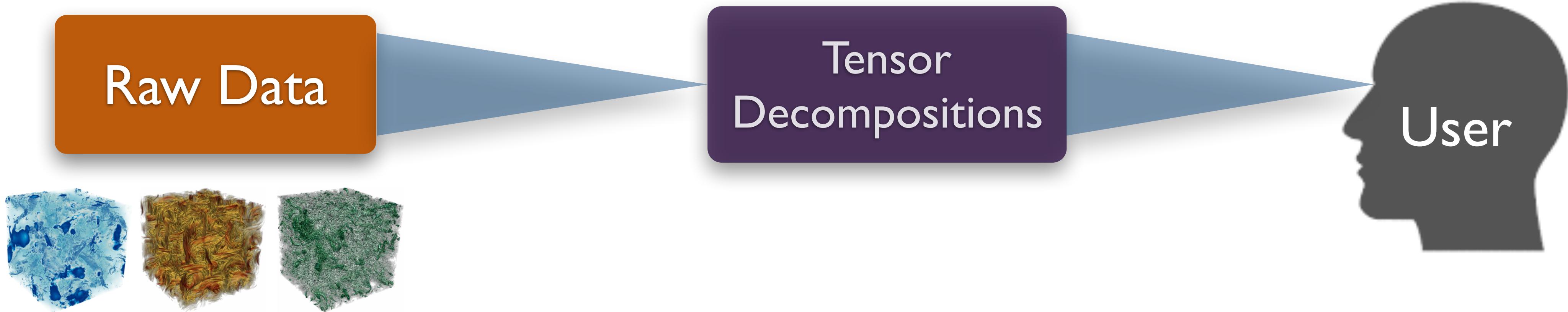
- Support for **non-rectangular** regions

Tensor Decompositions



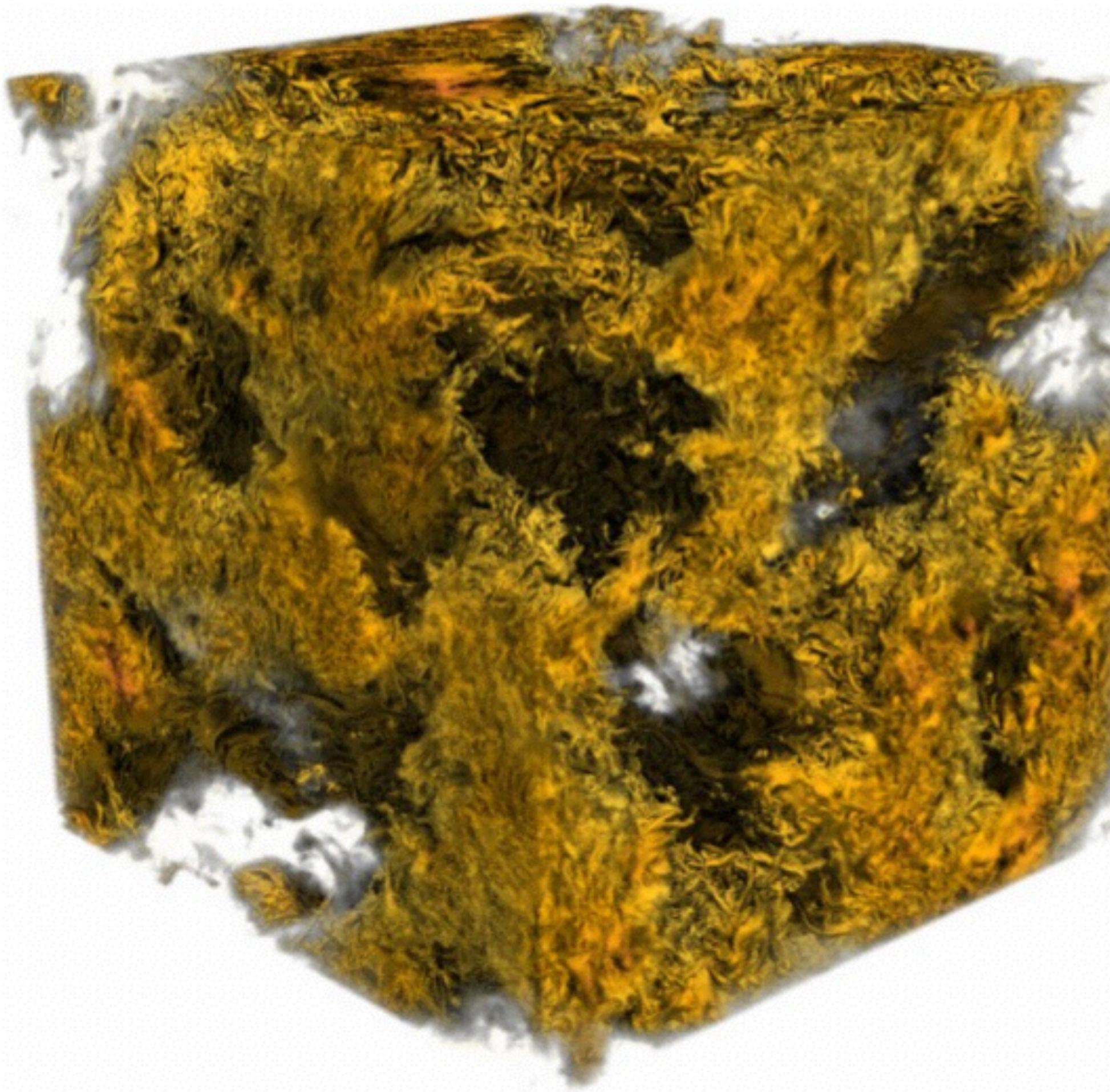
- Support for **non-rectangular** regions
- Reconstruction: matrix/tensor products

Tensor Decompositions

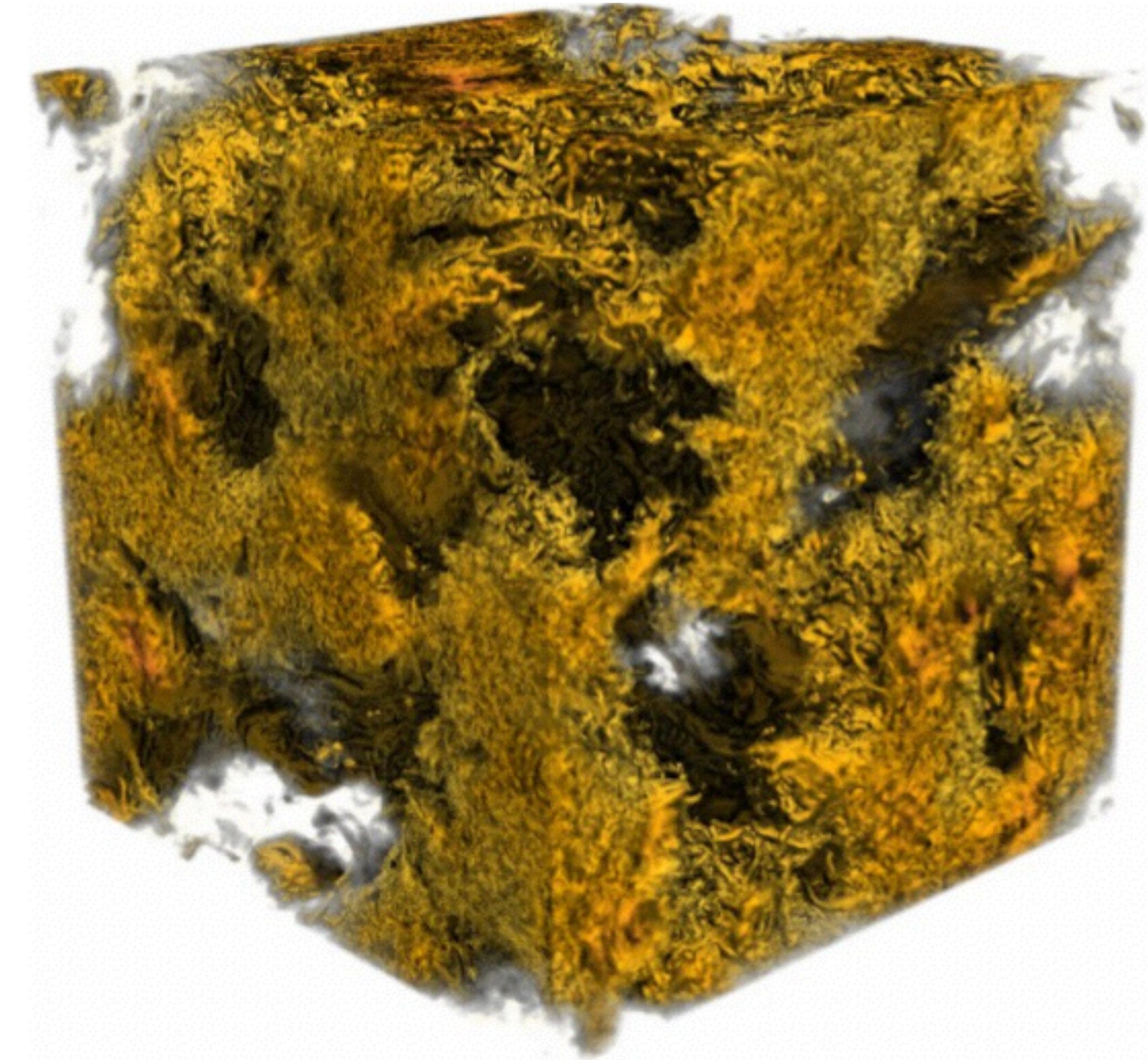


- Support for **non-rectangular** regions
- Reconstruction: matrix/tensor products
- Lossy → very **high compression rates**

Tensor Compression



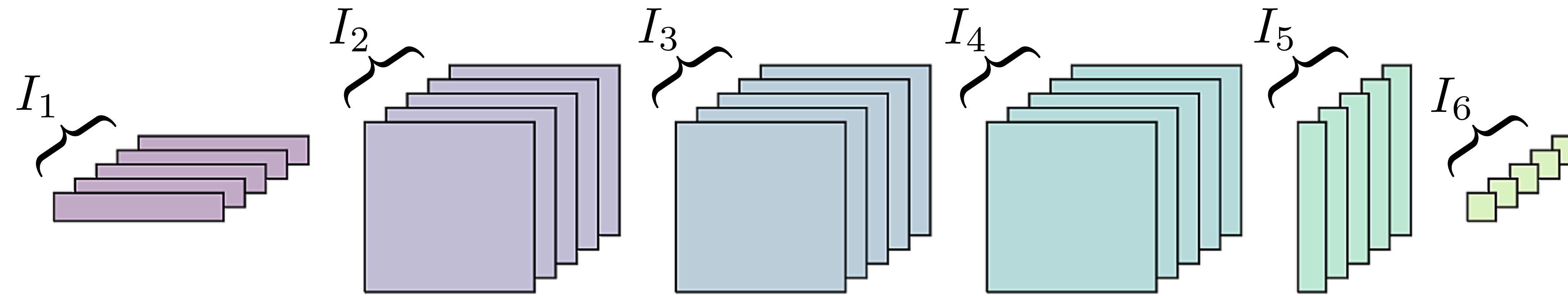
Scalar field (512MB)



Tensor-compressed (1.4MB)

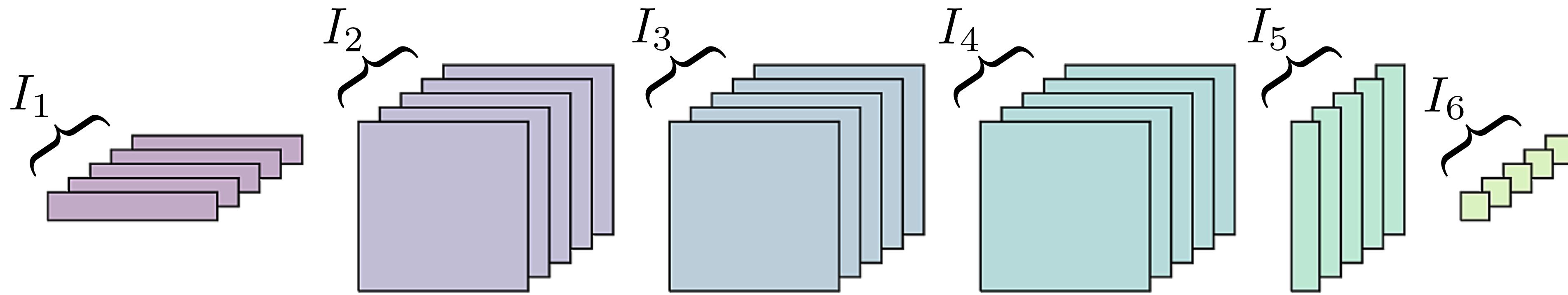
Tensor Train Decomposition

- Sequences of matrix products [Oseledets' 11]



Tensor Train Decomposition

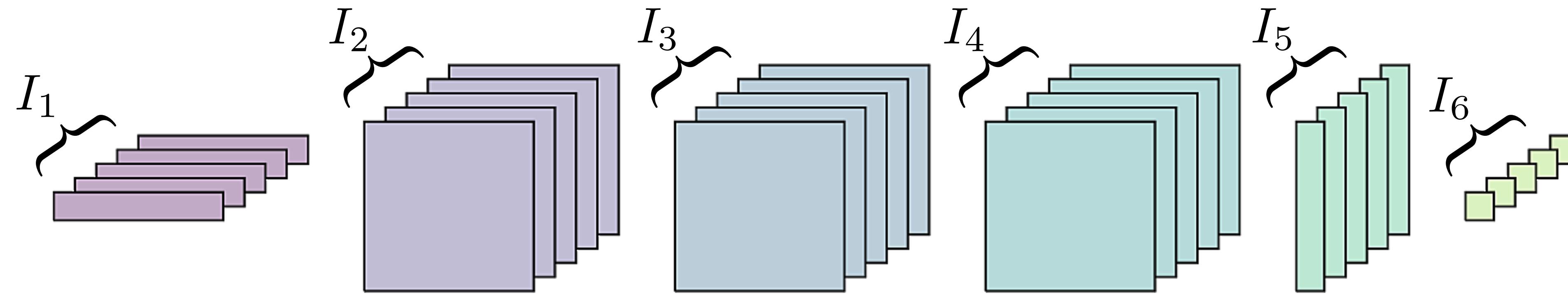
- Sequences of matrix products [Oseledets' 11]



$$\mathcal{T}[i_1, \dots, i_N] \approx \mathcal{T}^{(1)}[:, i_1, :] \cdots \mathcal{T}^{(N)}[:, i_N, :]$$

Tensor Train Decomposition

- Sequences of matrix products [Oseledets' 11]



$$\mathcal{T}[i_1, \dots, i_N] \approx \mathcal{T}^{(1)}[:, i_1, :] \cdots \mathcal{T}^{(N)}[:, i_N, :]$$

- Compact, flexible, scales well with #dimensions, easy to operate with

The Tensor Toolkit

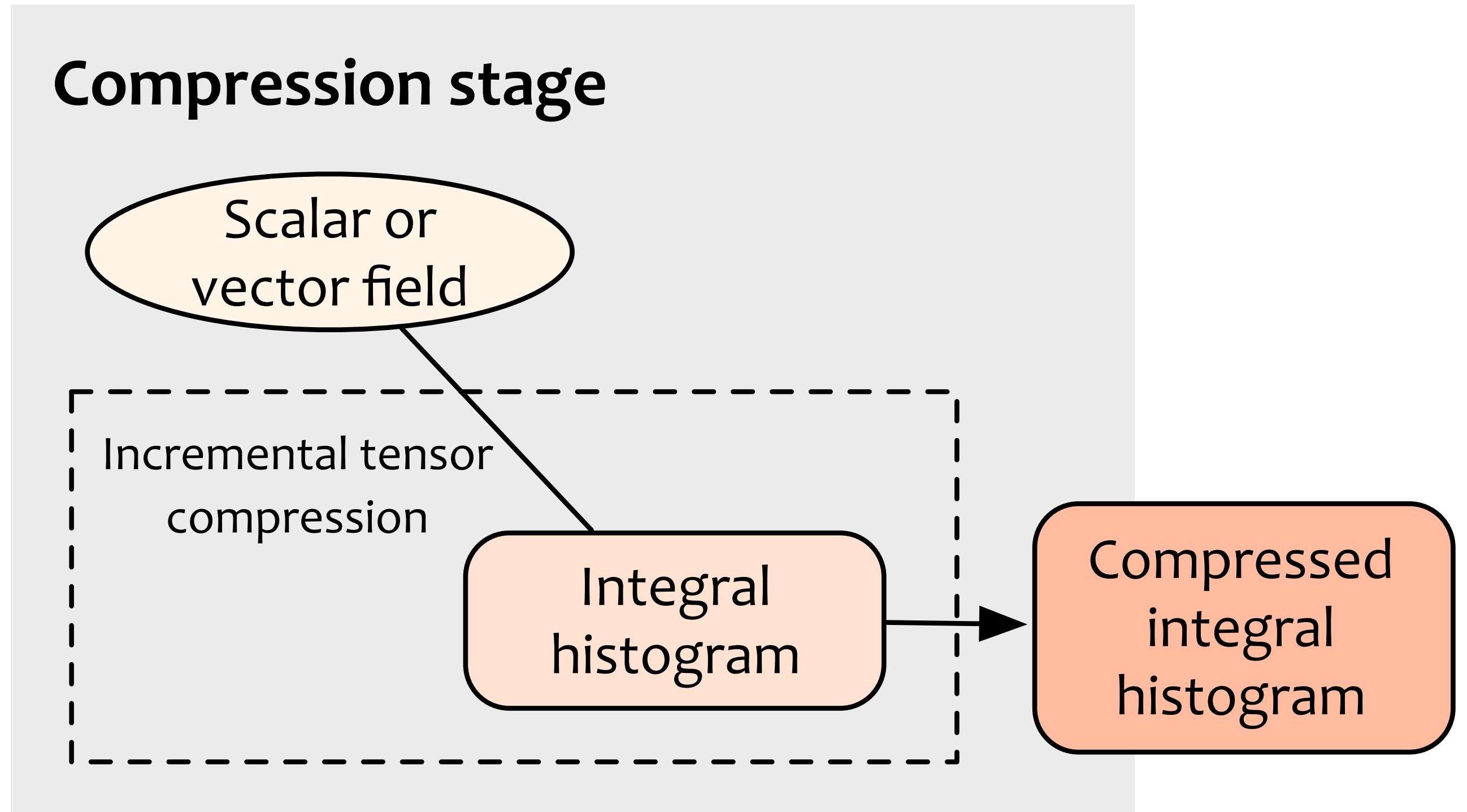
“Easier” (fewer operations)

- Indexing/slicing
- Decompression
- Addition/subtraction
- Editing tensor entries
- Stacking tensors
- DCT, DFT, linear transforms
- Separable convolution
- Differentiation/integration

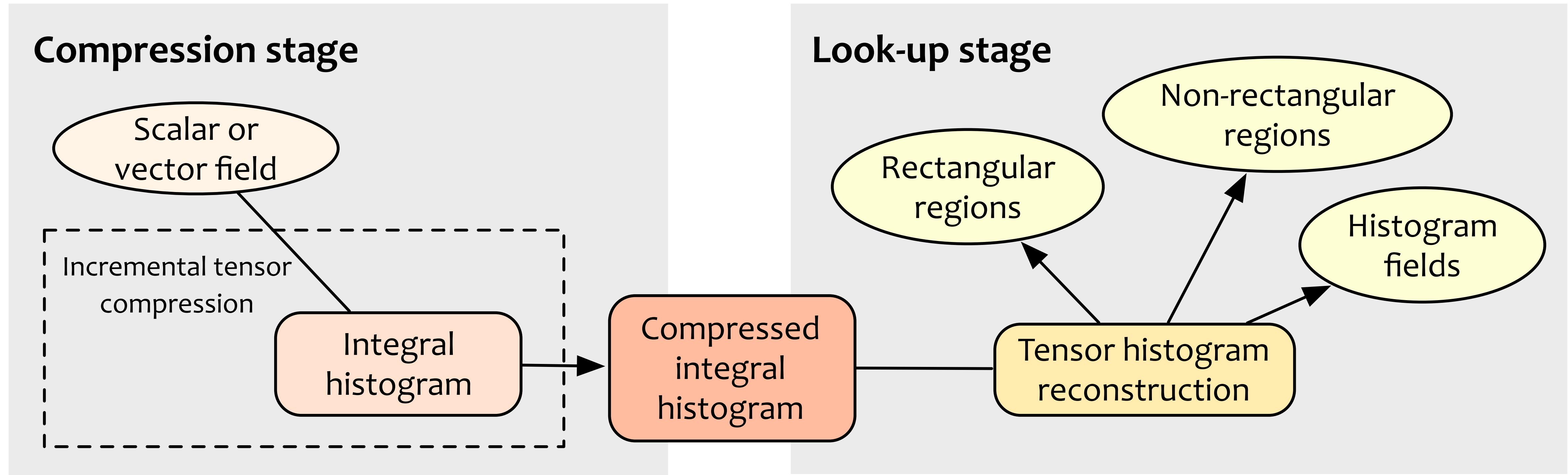
“Harder” (more operations)

- Compression
- Recompression
- Product
- Elementwise functions
- General convolution
- Dot product
- Finding extrema of a tensor
- Interpolation

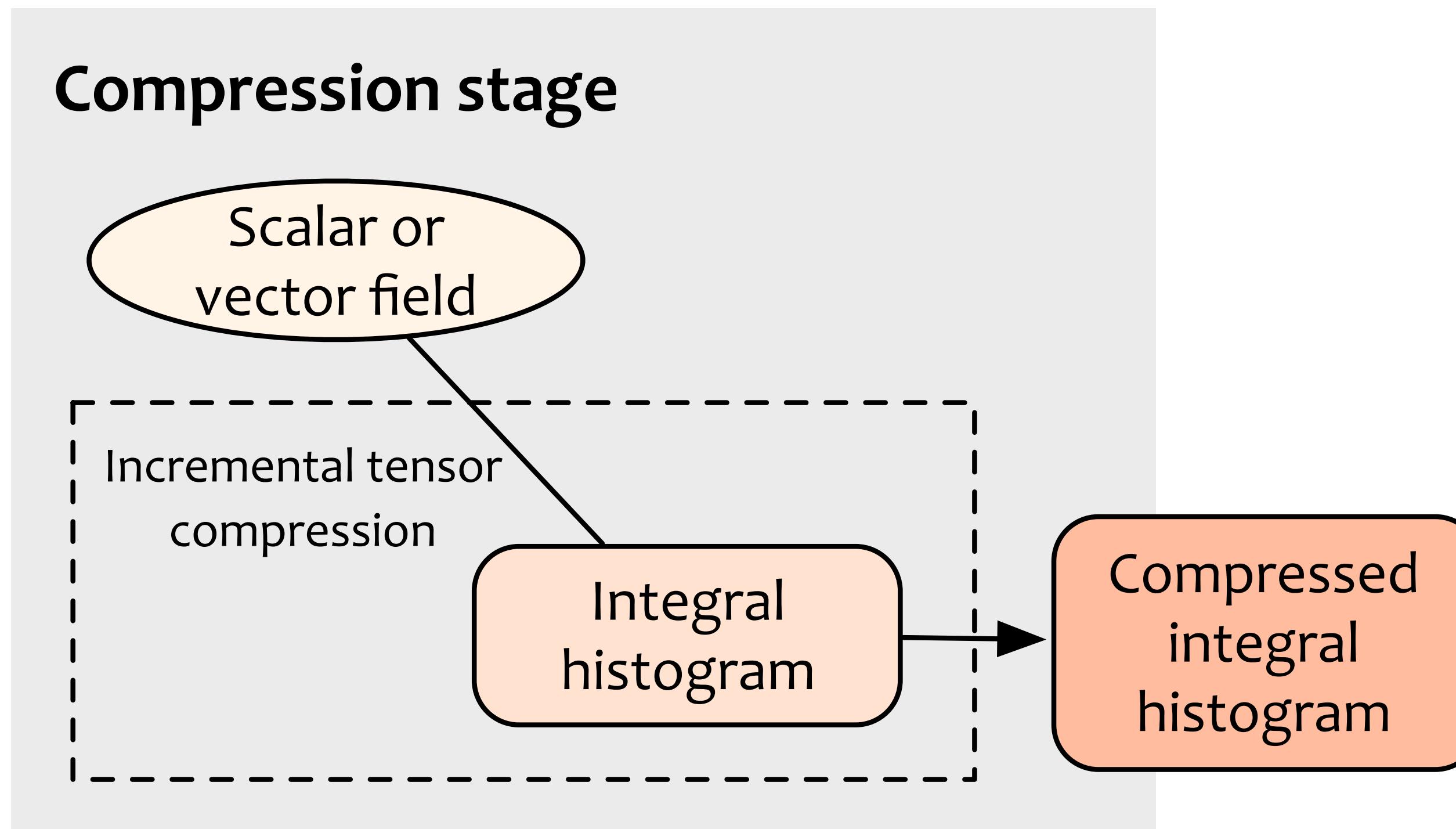
Proposed Solution



Proposed Solution



Compression Stage



The Tensor Toolkit

“Easier” (fewer operations)

- Indexing/slicing
- Decompression
- Addition/subtraction
- Editing tensor entries
- Stacking tensors
- DCT, DFT, linear transforms
- Separable convolution
- Differentiation/integration

“Harder” (more operations)

- Compression
- Recompression
- Product
- Elementwise functions
- General convolution
- Dot product
- Finding extrema of a tensor
- Interpolation

Compression stage

Tree-based Compression

Bin 1 Bin 2

Tree-based Compression

Compress

Bin 1

Bin 2

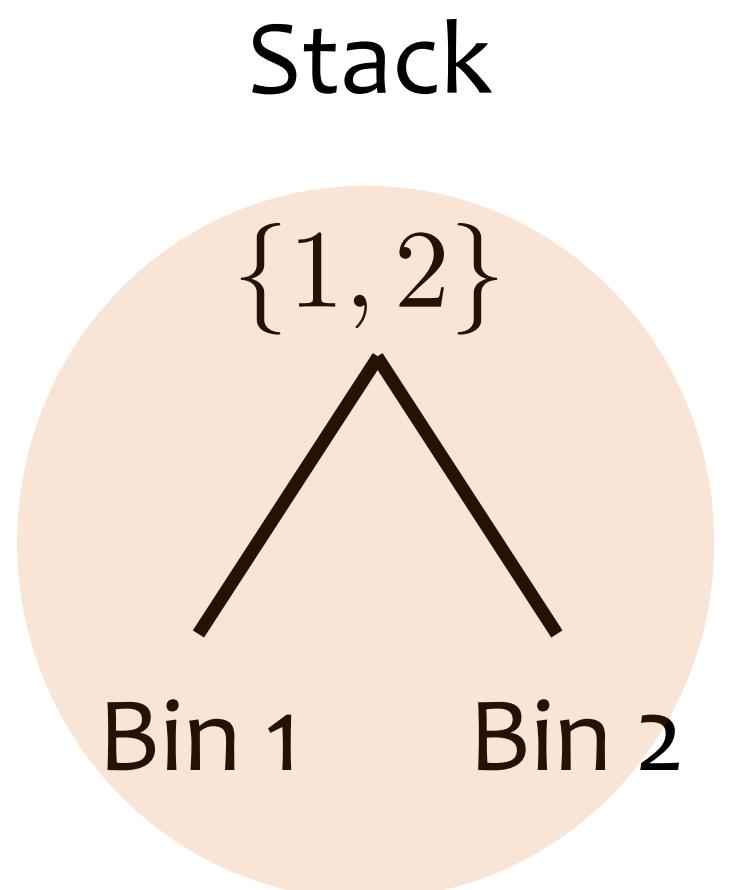
Tree-based Compression

Compress

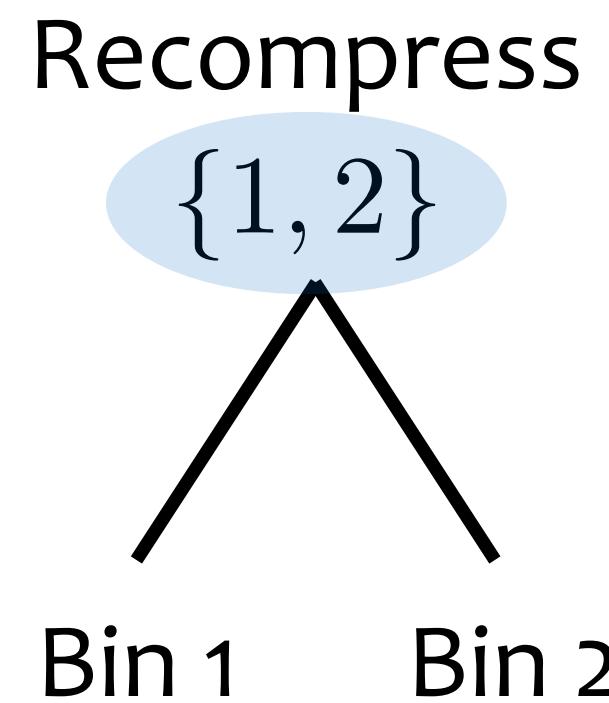
Bin 1

Bin 2

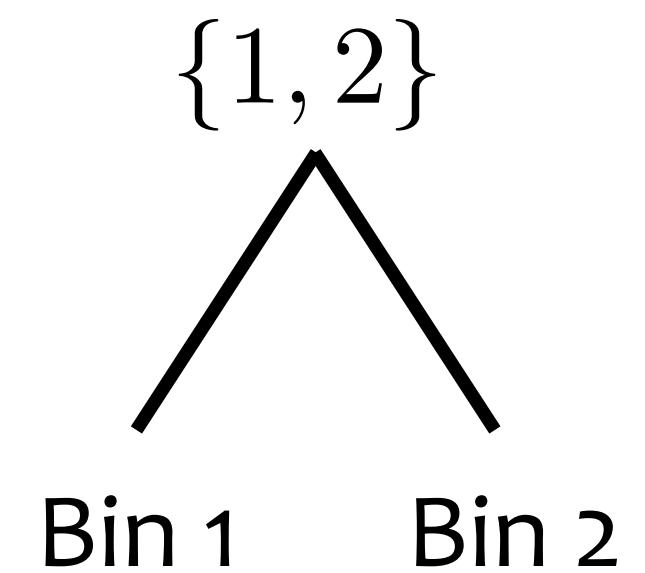
Tree-based Compression



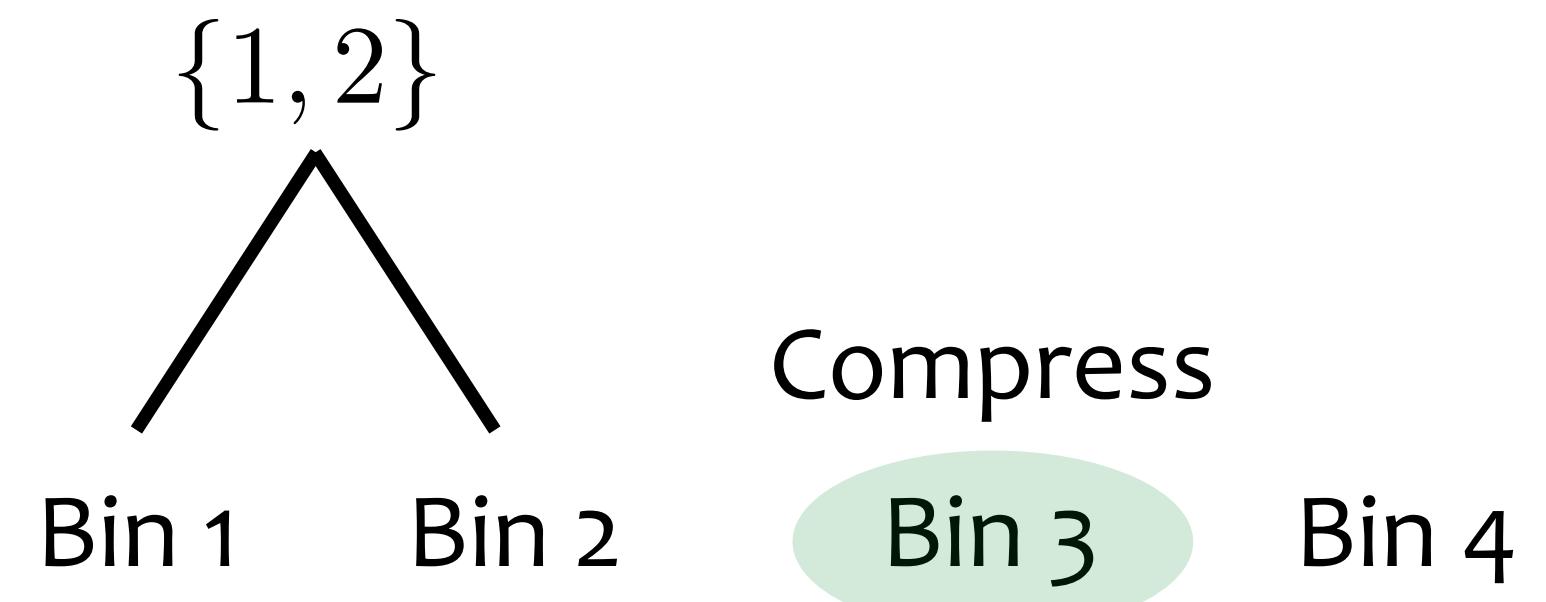
Tree-based Compression



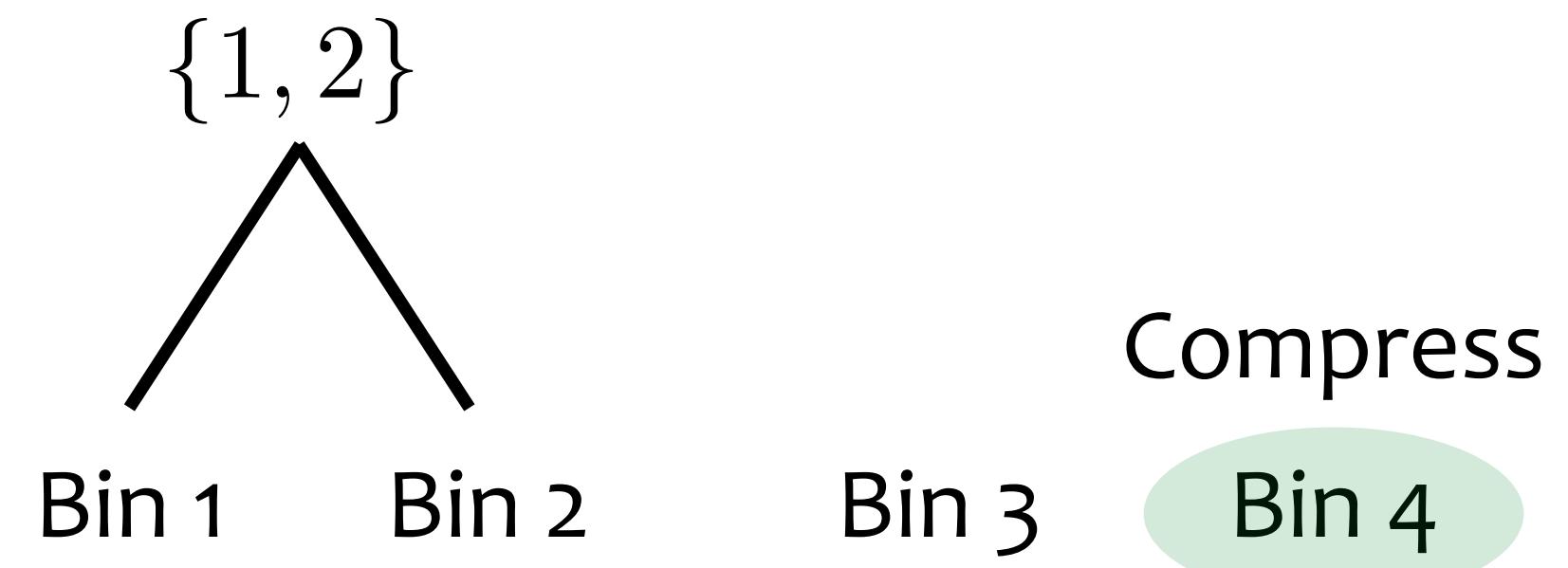
Tree-based Compression



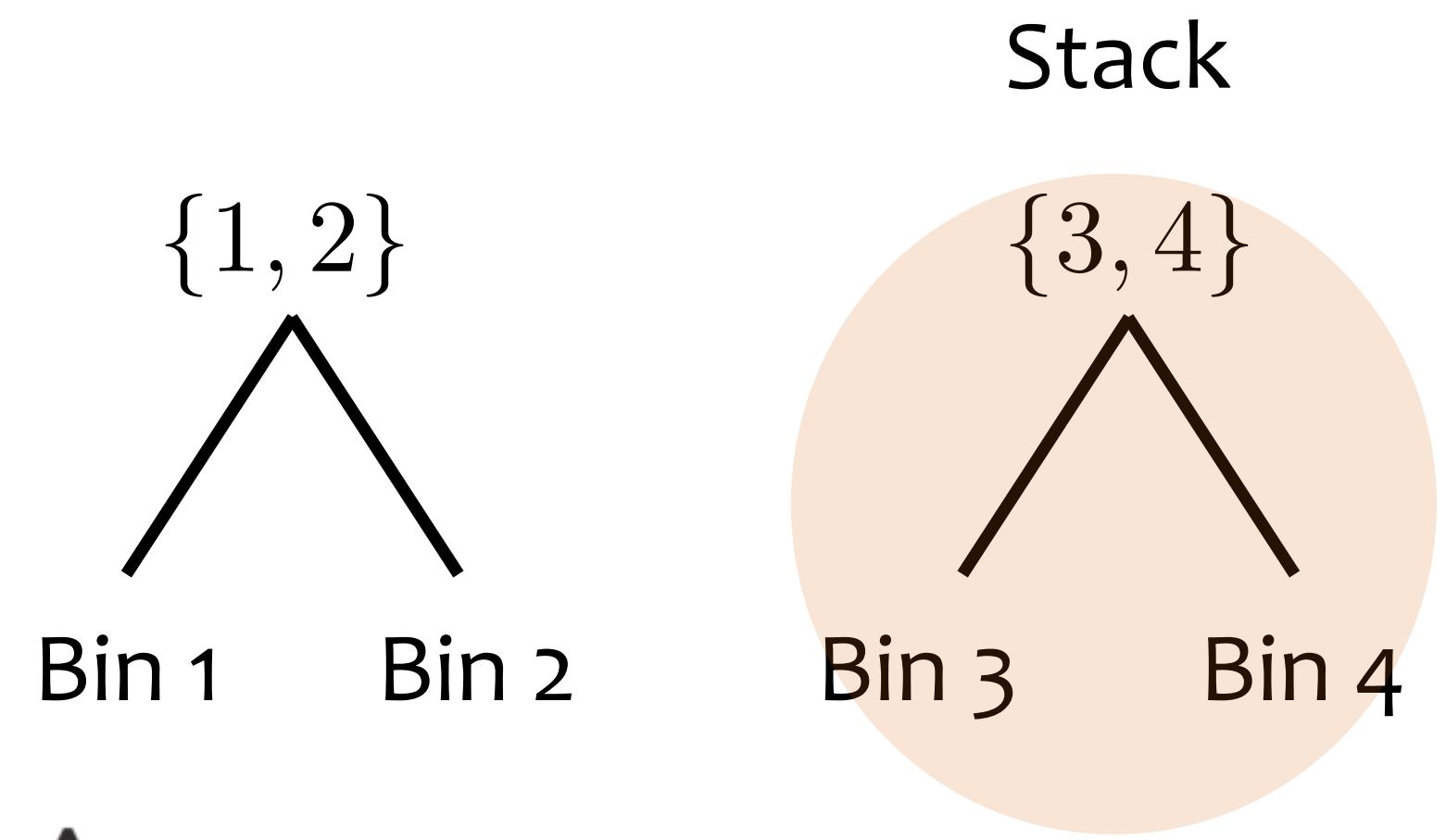
Tree-based Compression



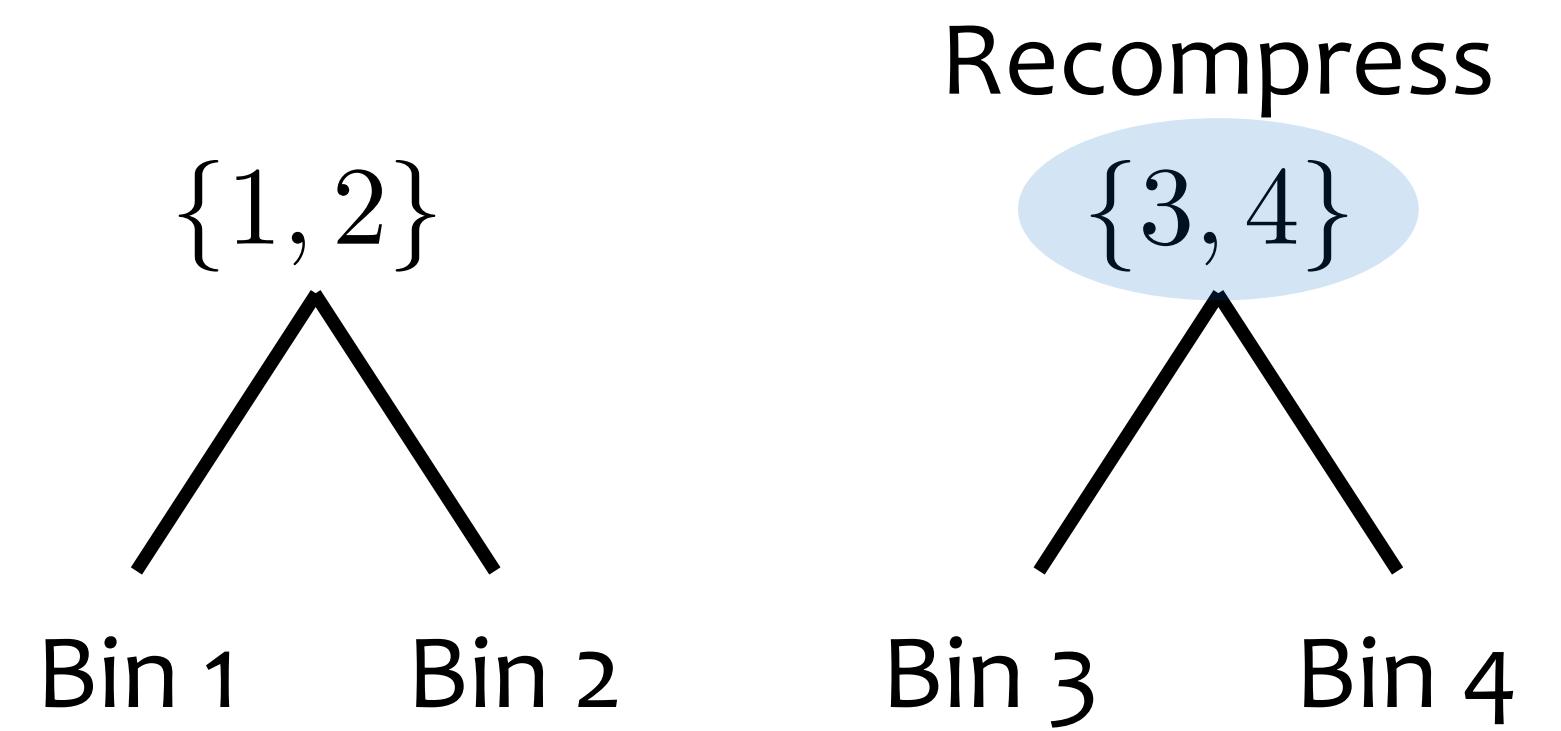
Tree-based Compression



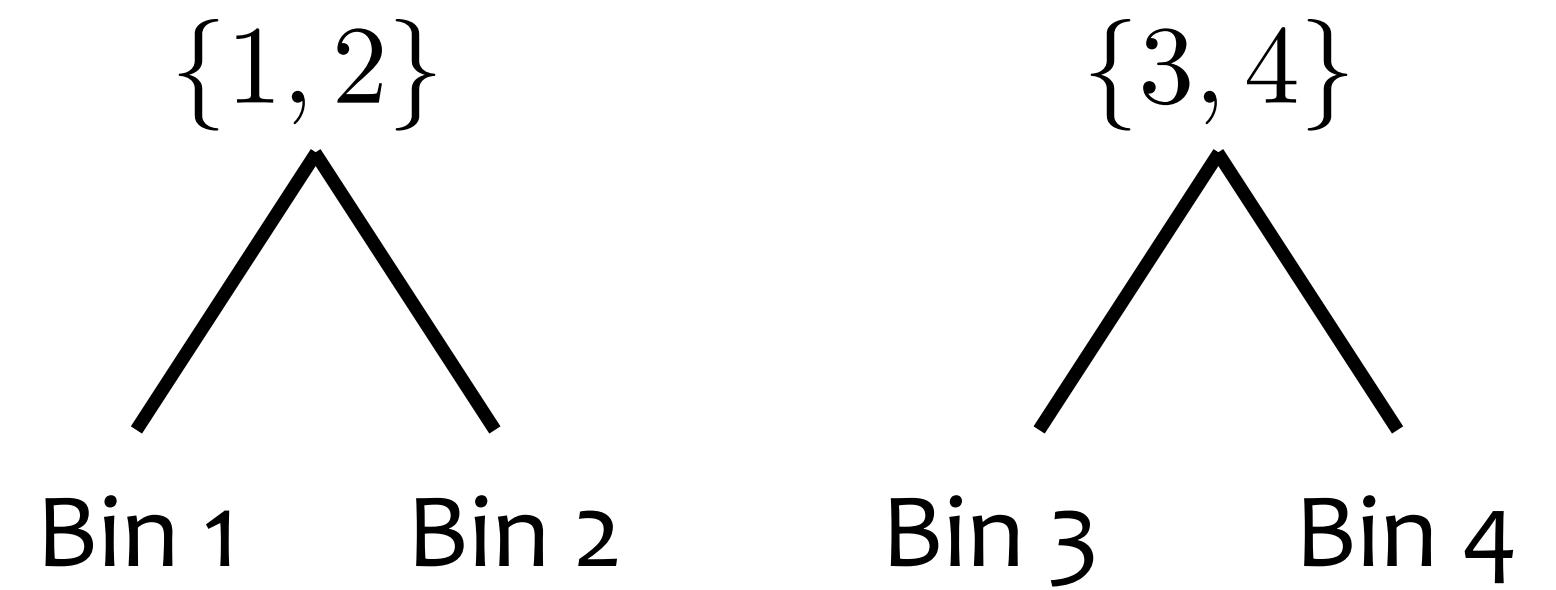
Tree-based Compression



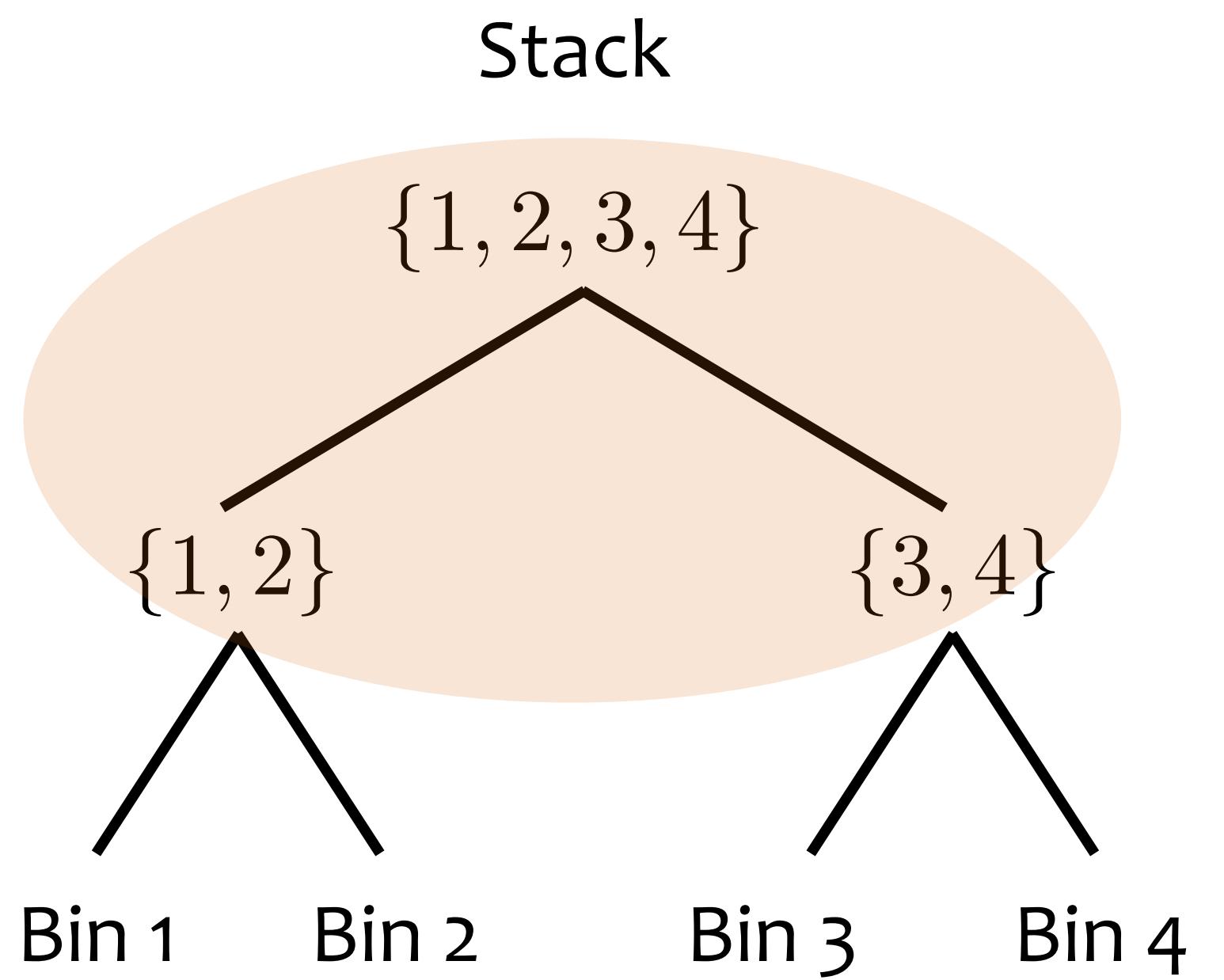
Tree-based Compression



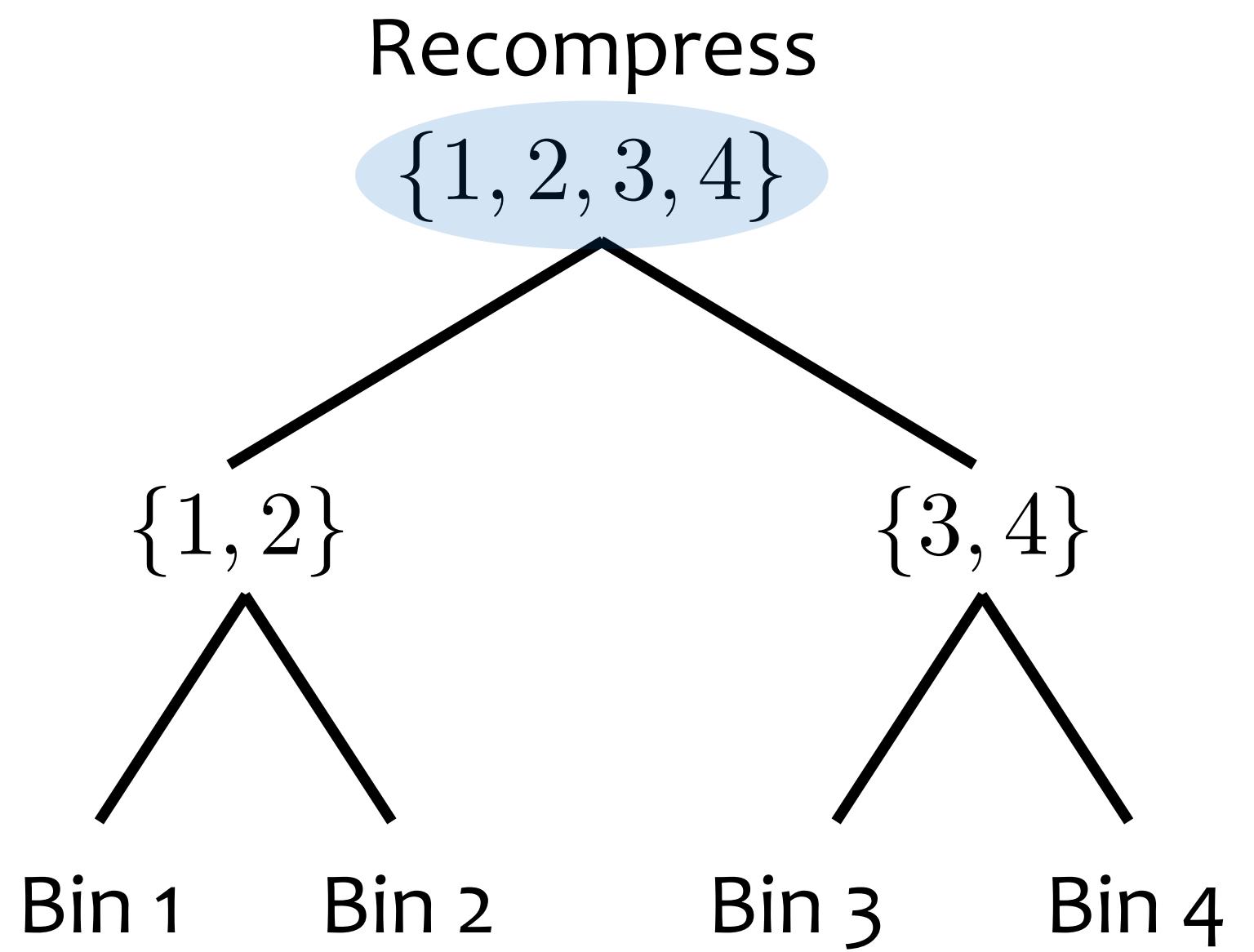
Tree-based Compression



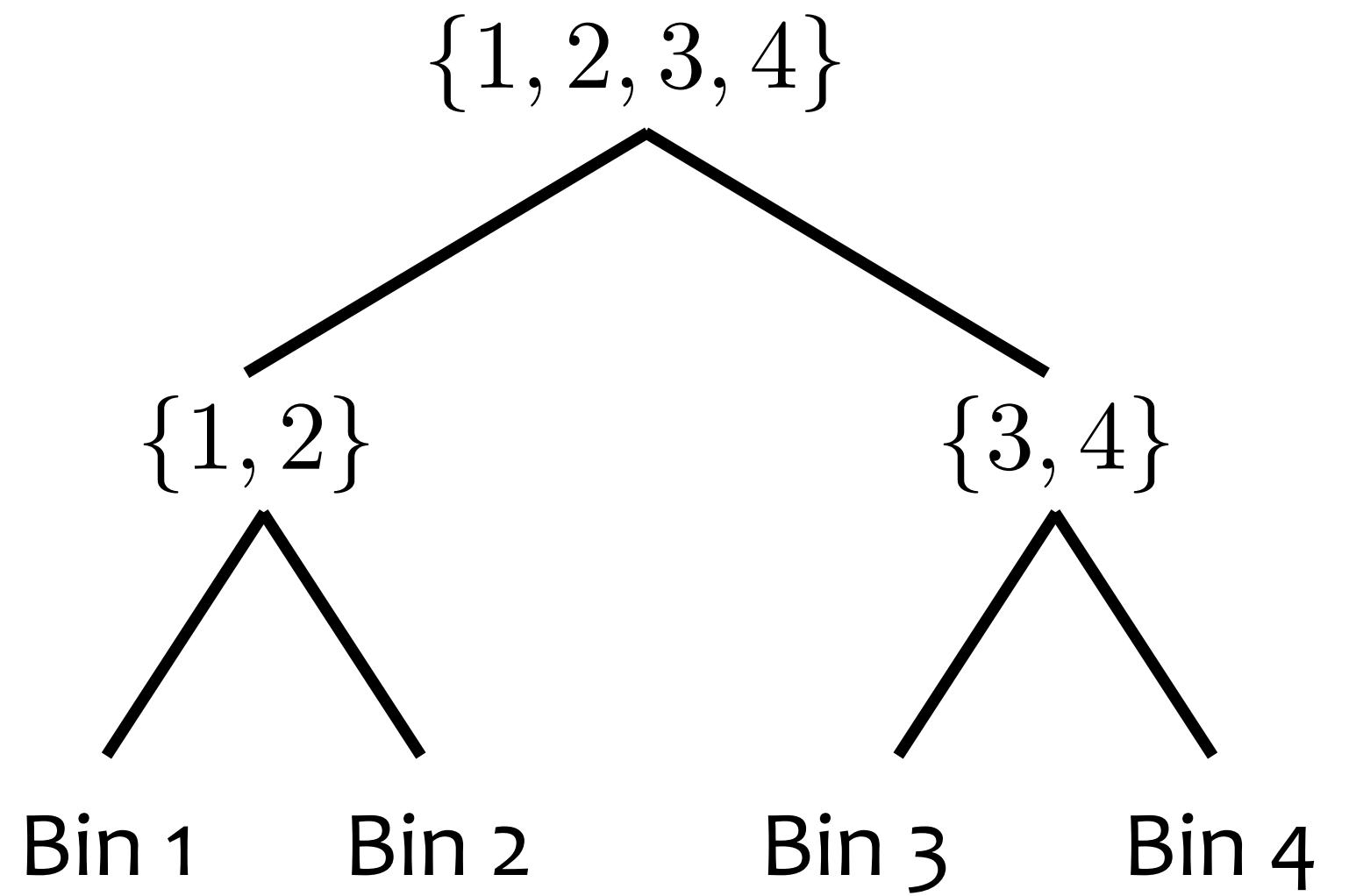
Tree-based Compression



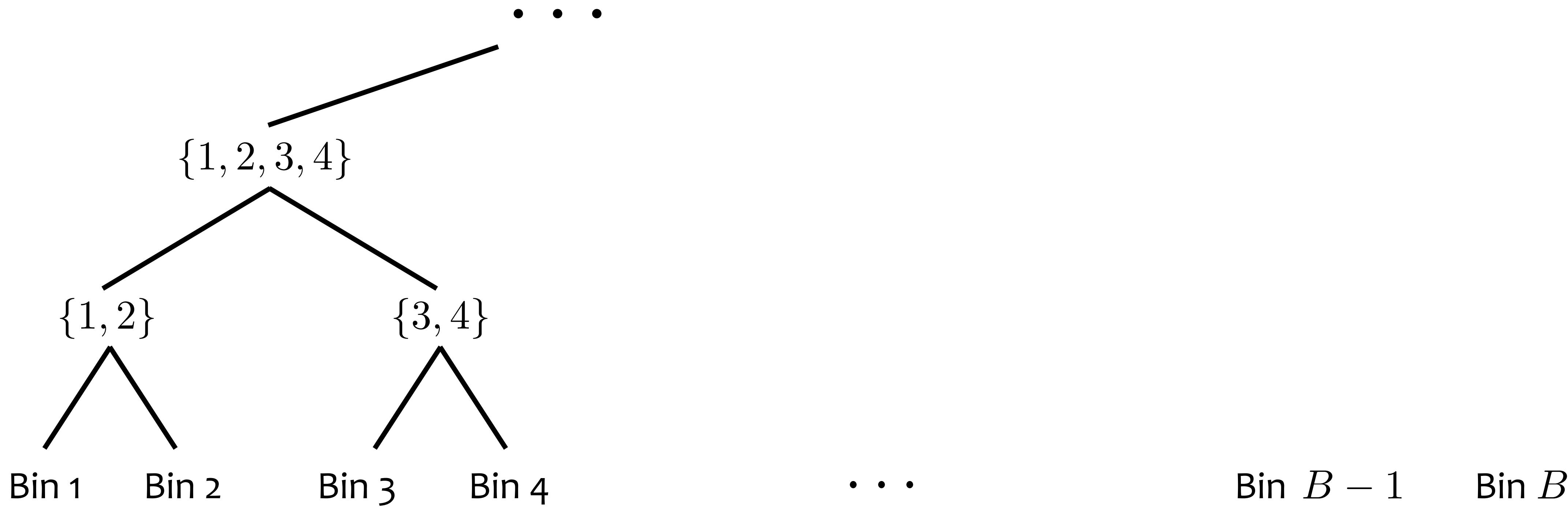
Tree-based Compression



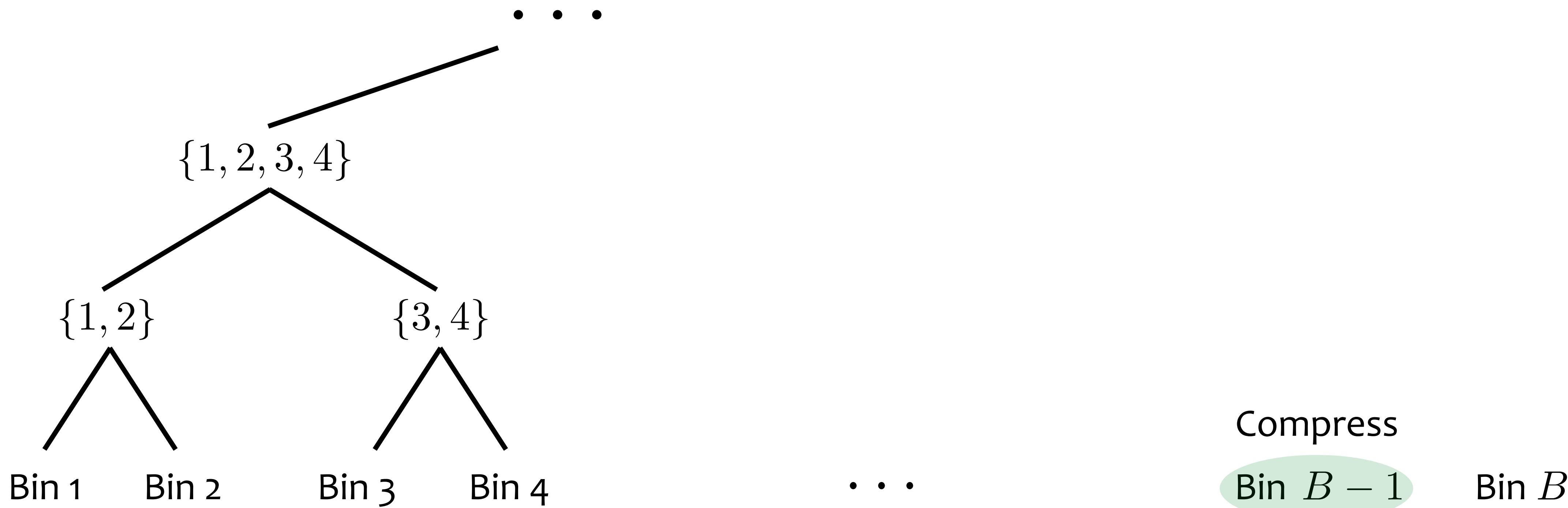
Tree-based Compression



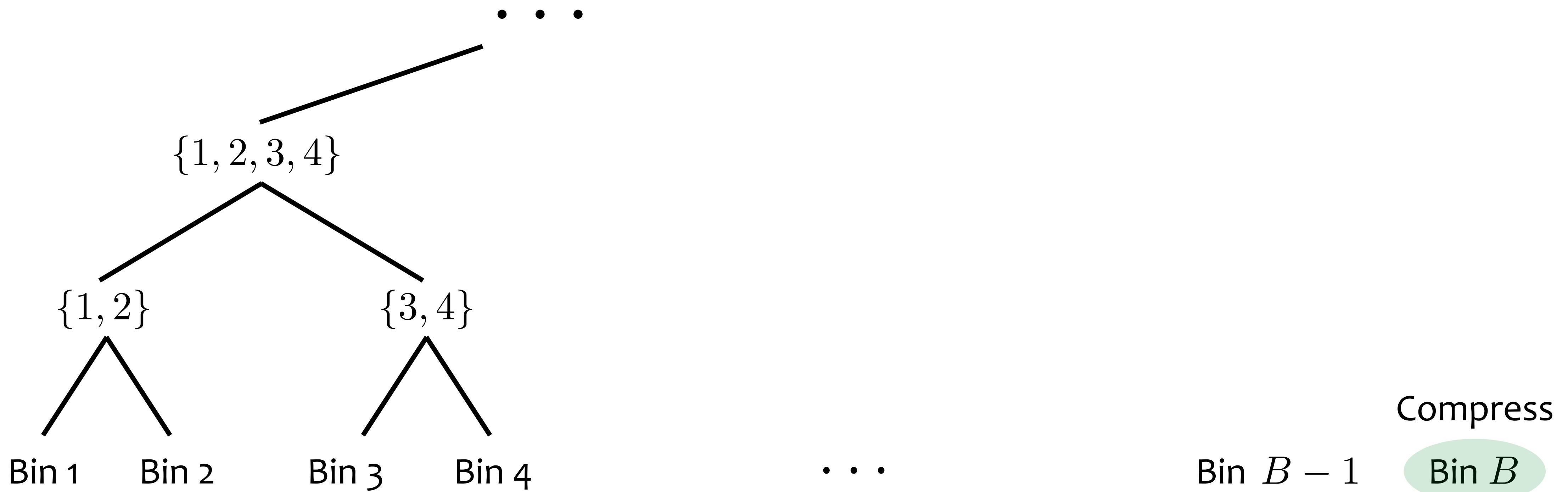
Tree-based Compression



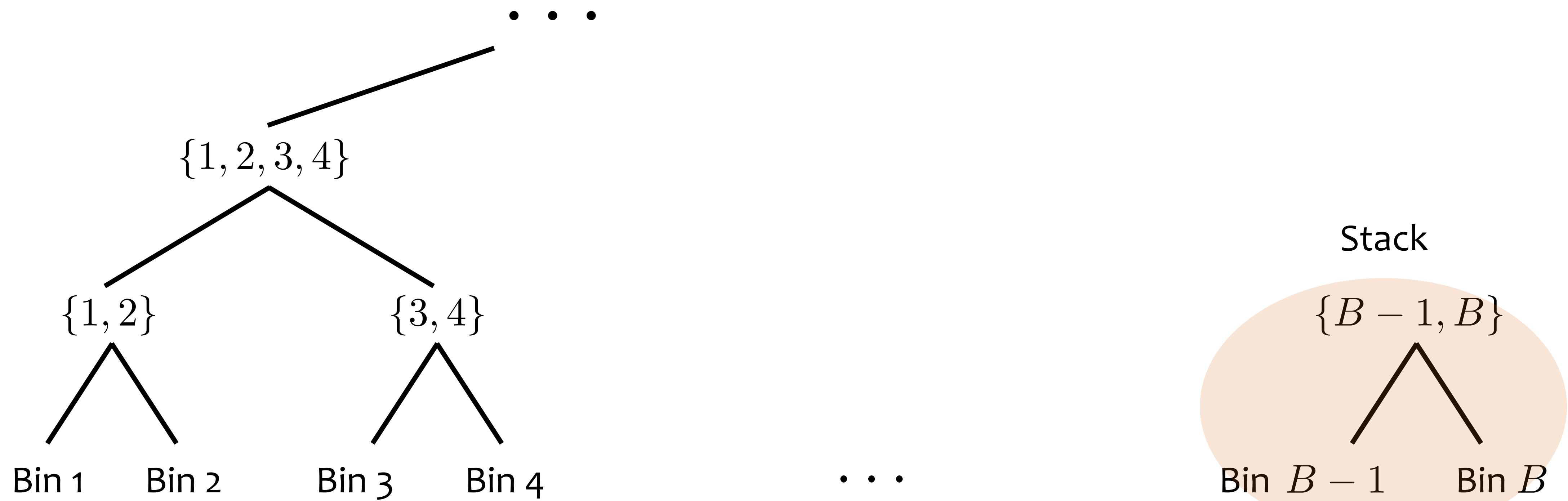
Tree-based Compression



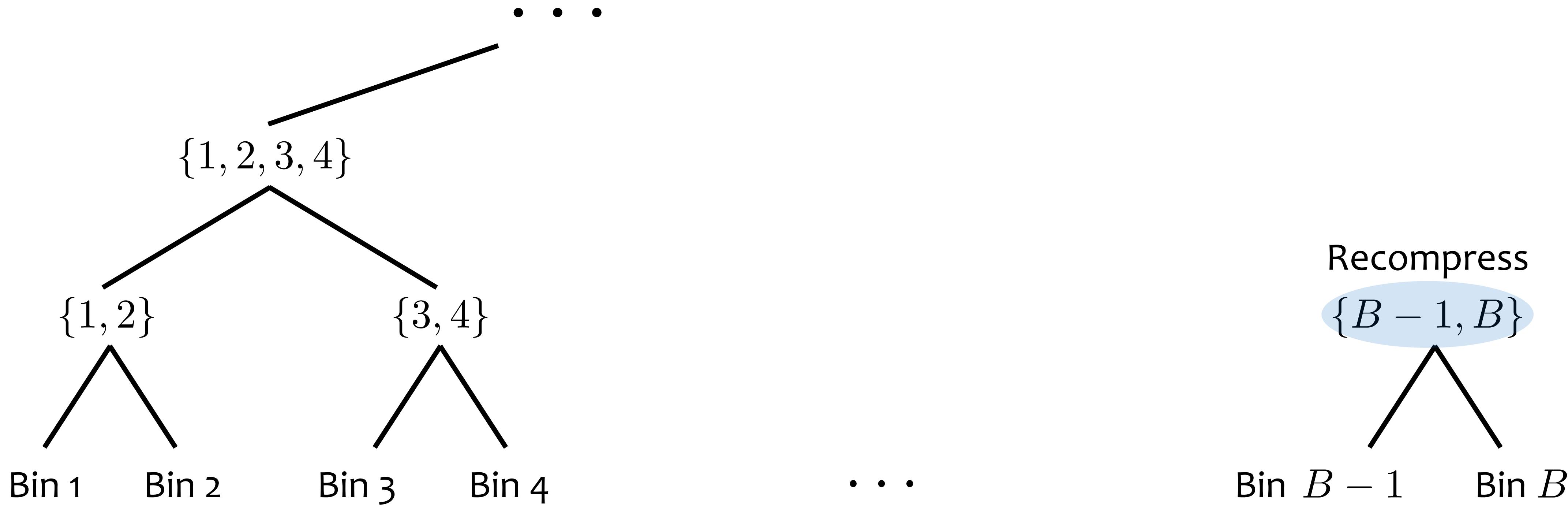
Tree-based Compression



Tree-based Compression



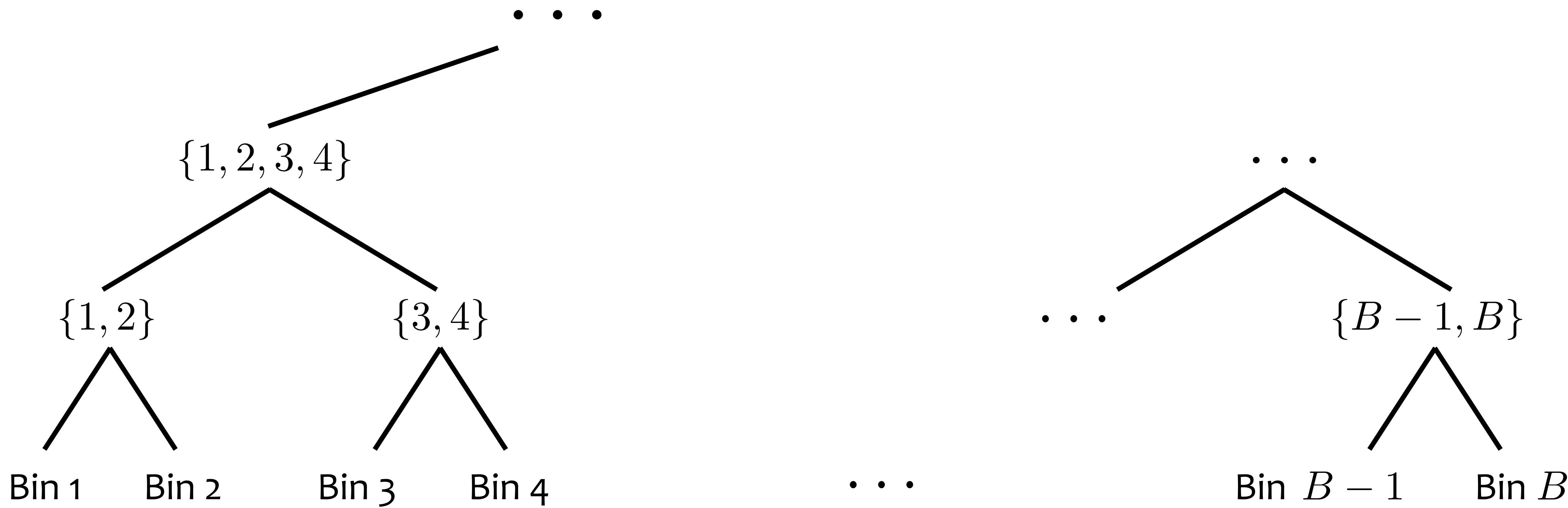
Tree-based Compression



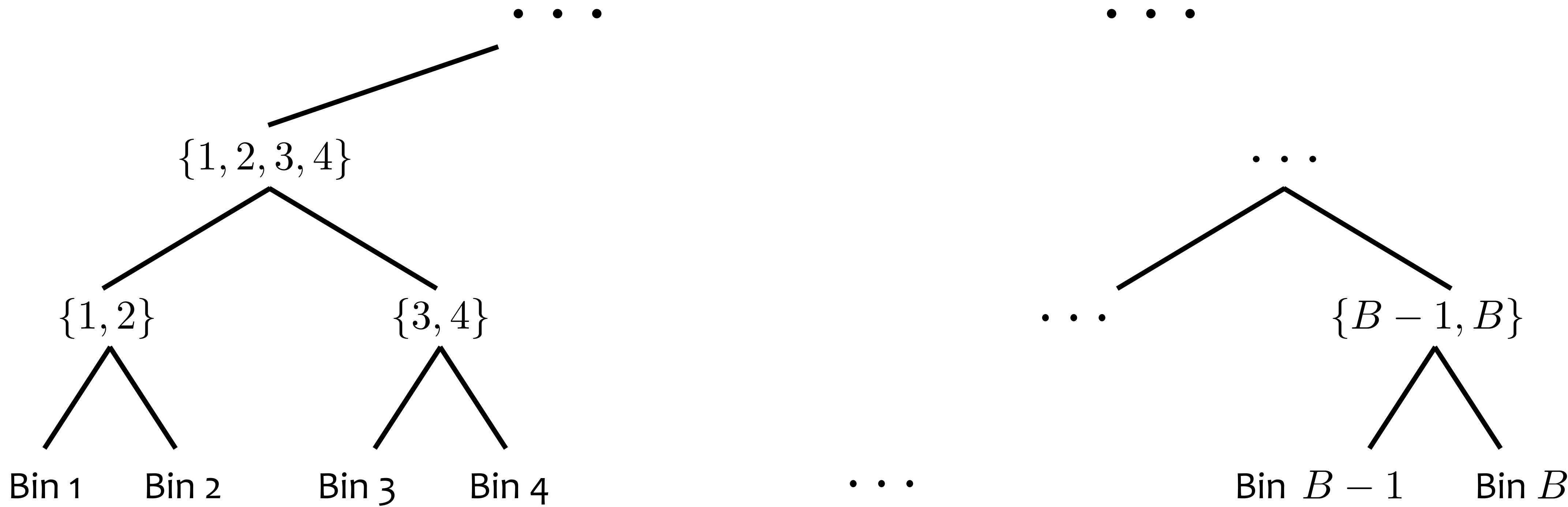
Tree-based Compression



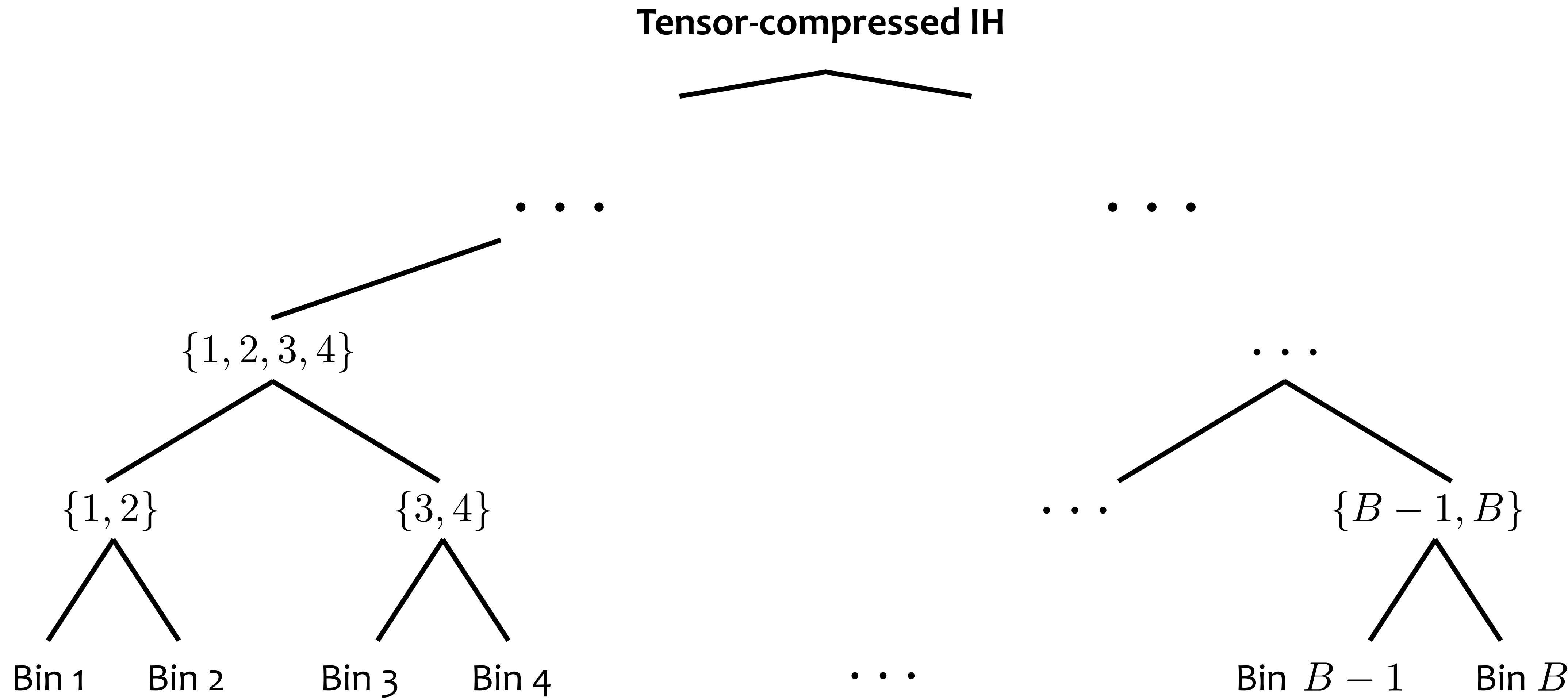
Tree-based Compression



Tree-based Compression

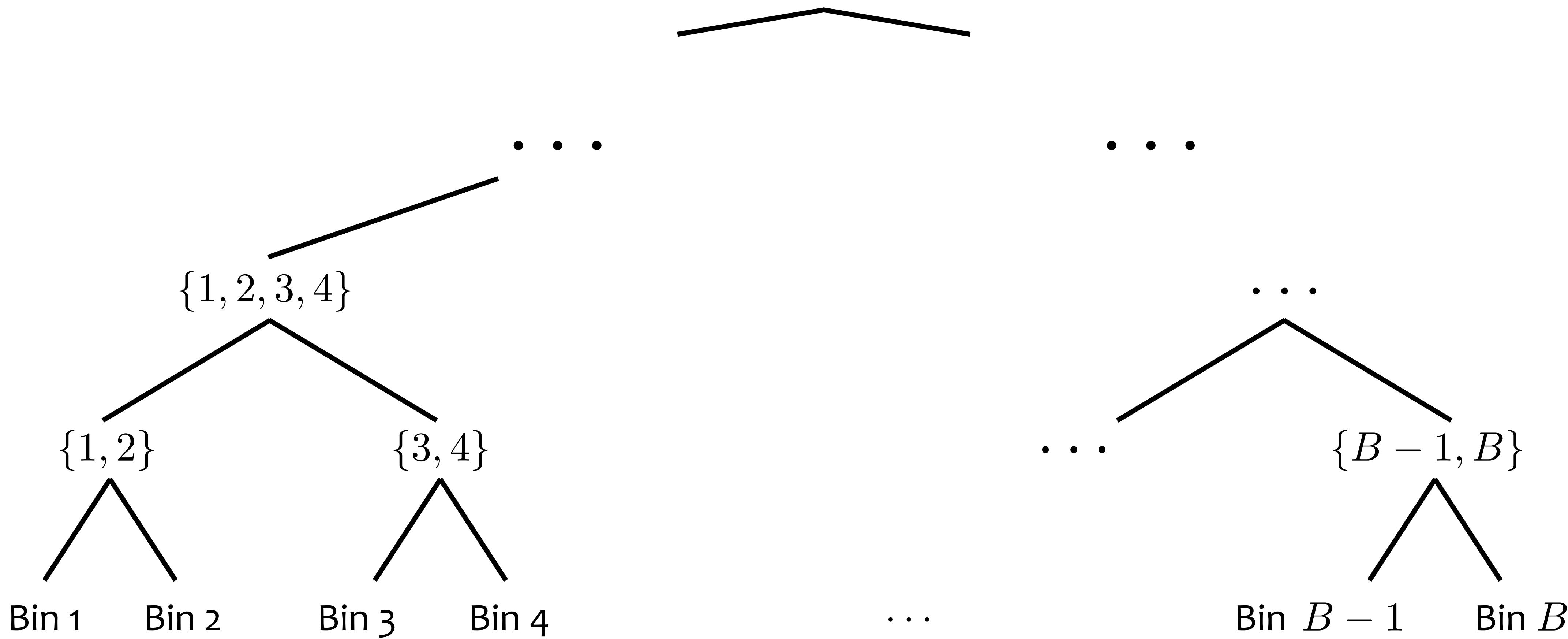


Tree-based Compression

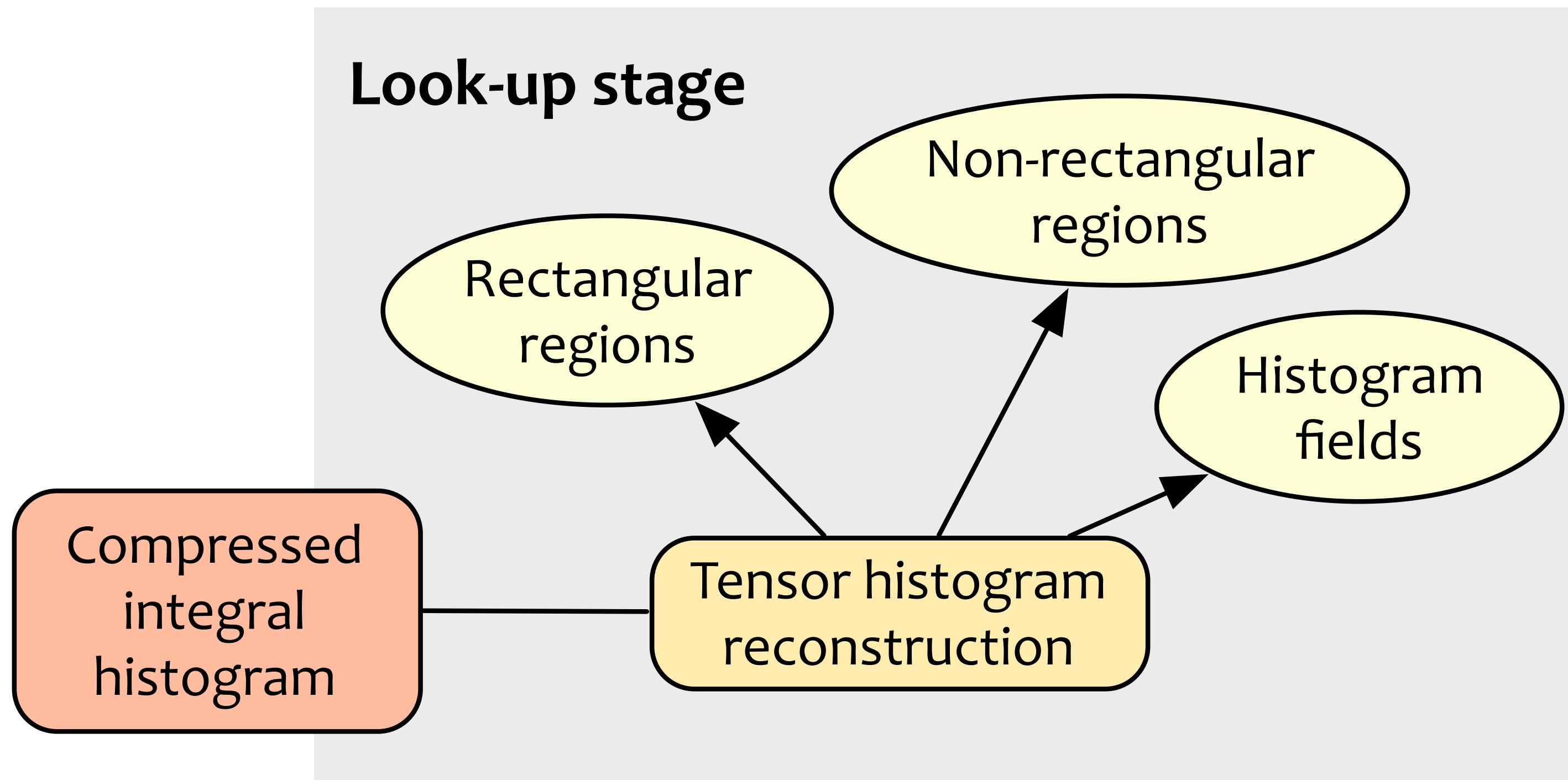


Tree-based Compression

Tensor-compressed IH → x100 - x1000 compression



Look-up Stage



The Tensor Toolkit

“Easier” (fewer operations)

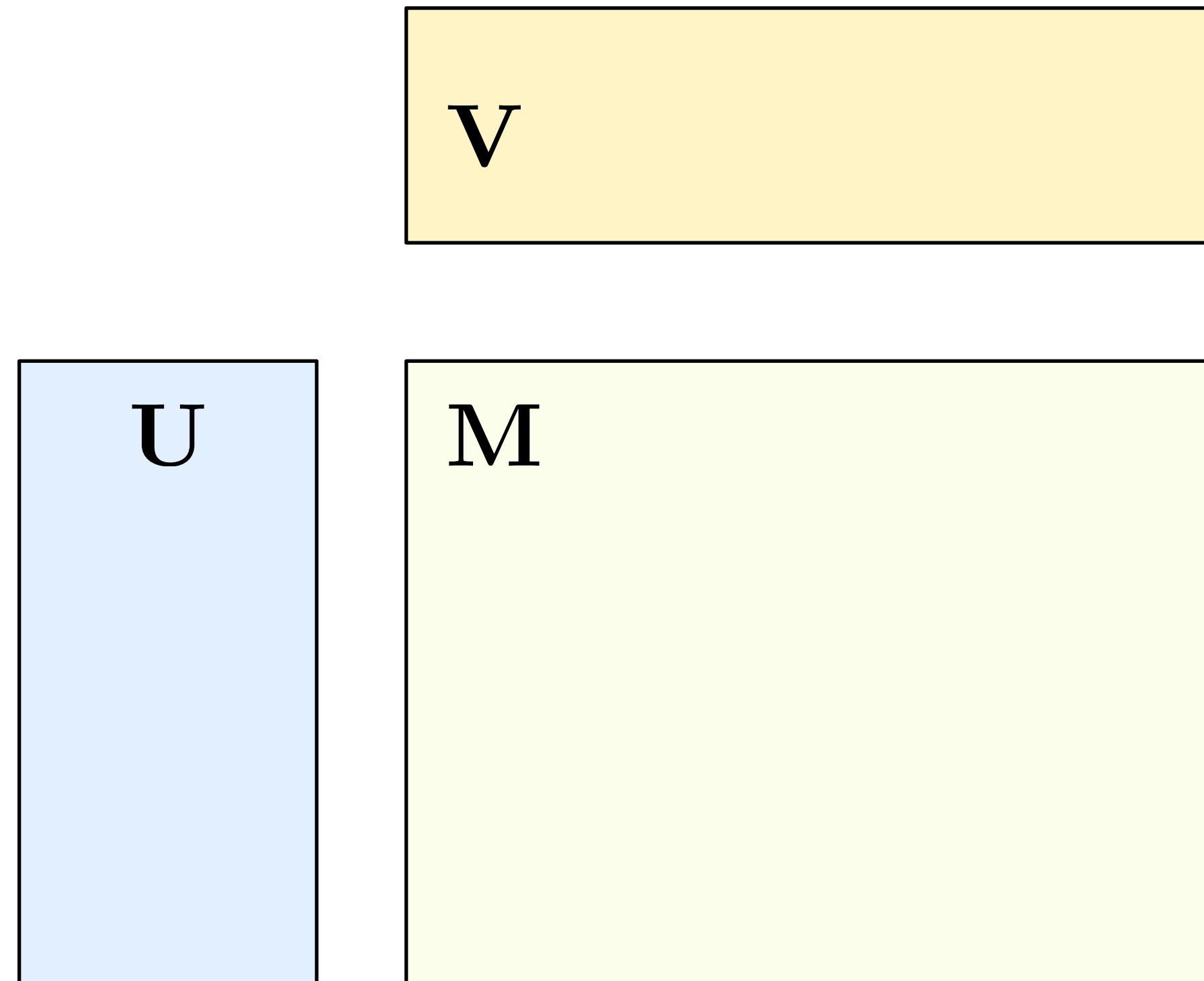
- Indexing/slicing
- Decompression
- Addition/subtraction
- Editing tensor entries
- Stacking tensors
- DCT, DFT, linear transforms
- Separable convolution
- Differentiation/integration

“Harder” (more operations)

- Compression
- Recompression
- Product
- Elementwise functions
- General convolution
- Dot product
- Finding extrema of a tensor
- Interpolation

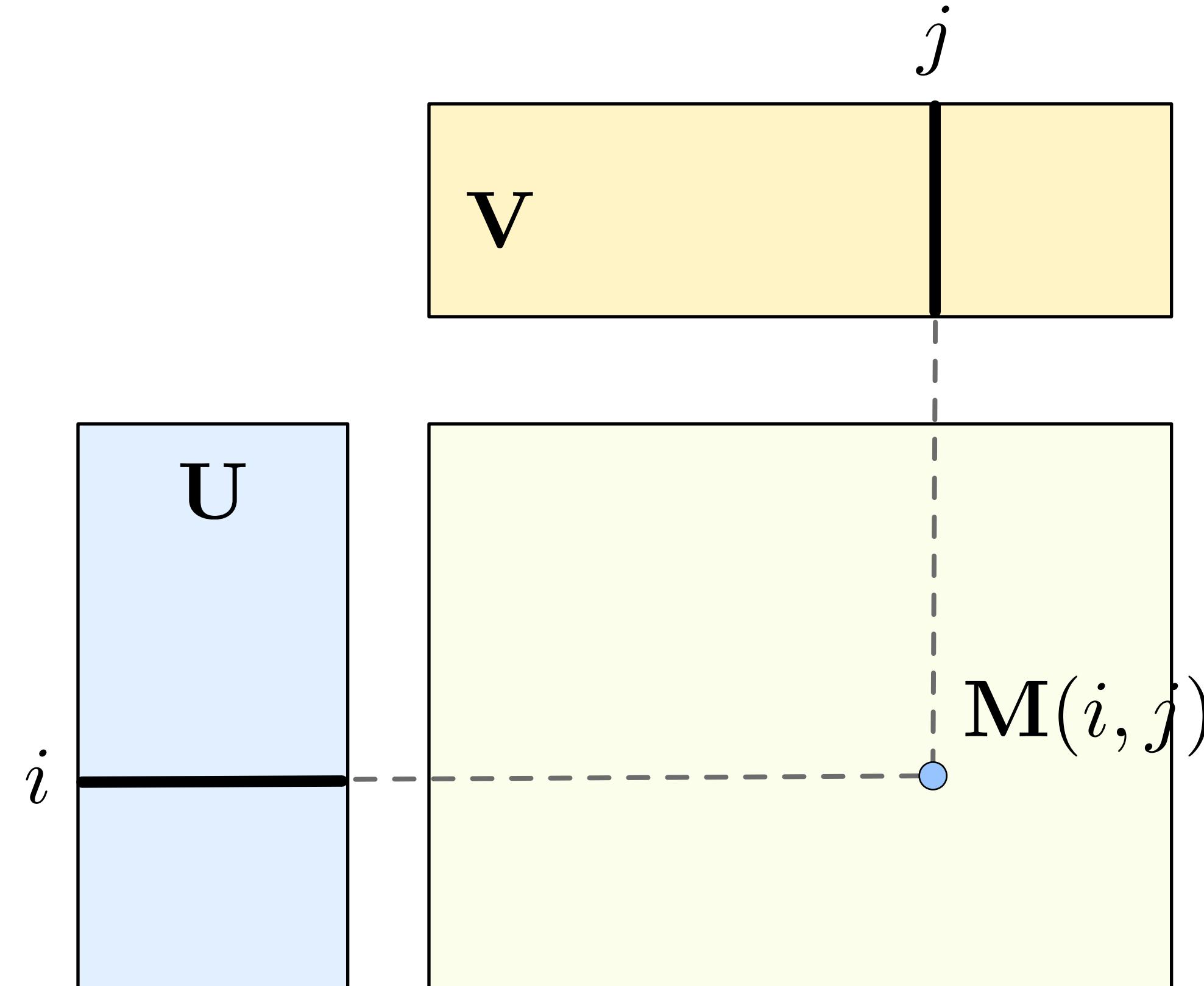
Look-up stage

Tensor SAT Reconstruction



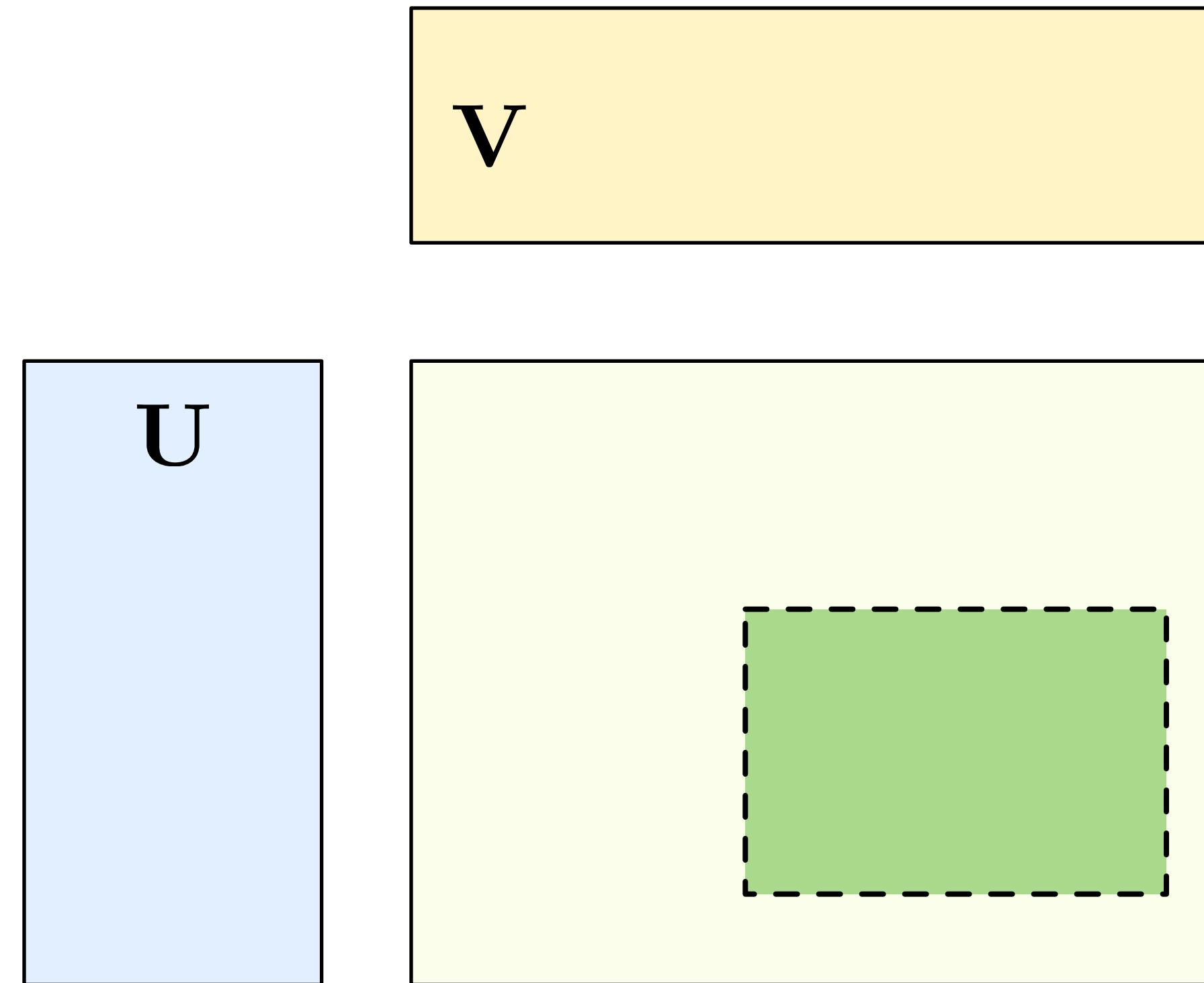
$$M \approx U \cdot V$$

Tensor SAT Reconstruction

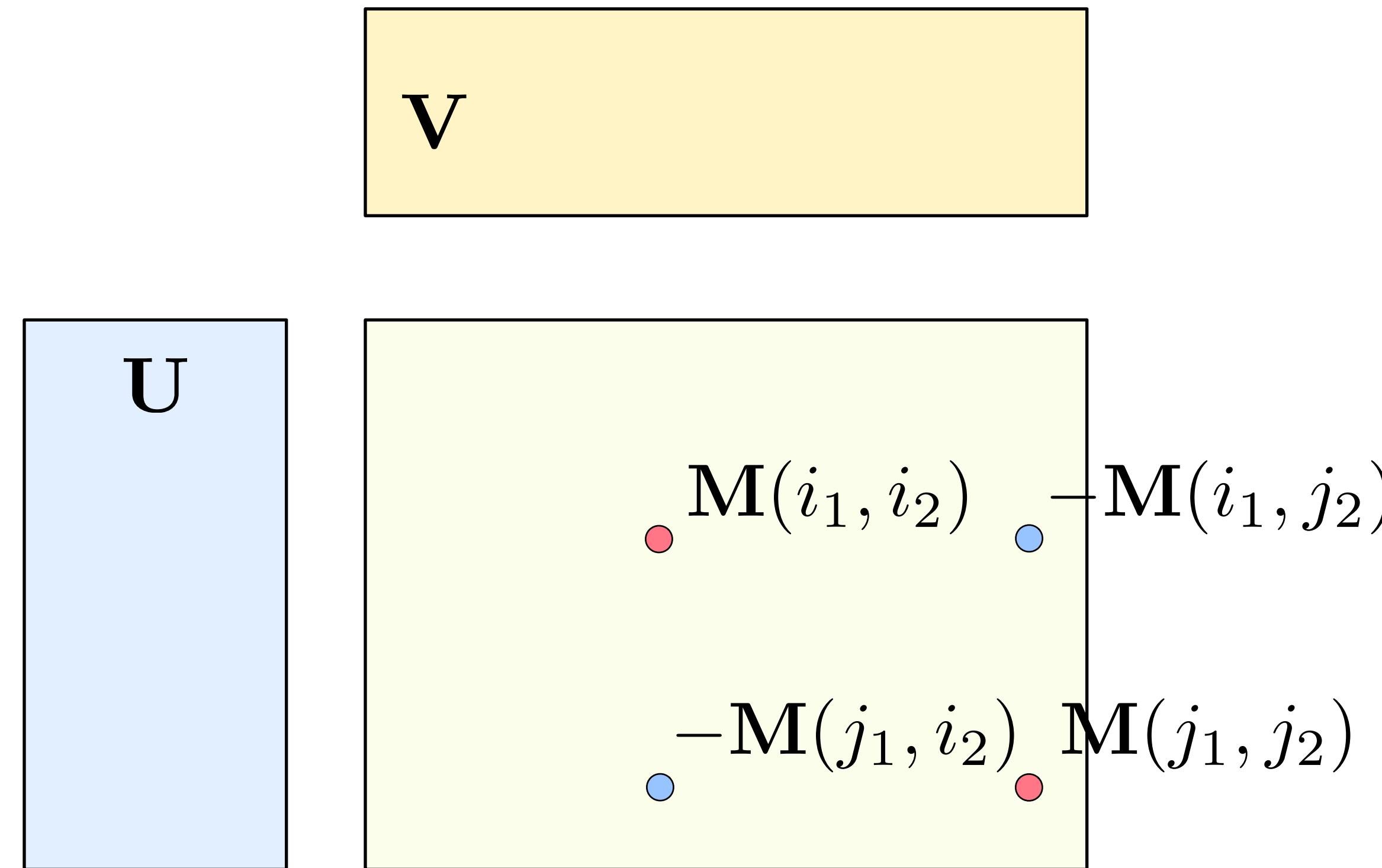


$$\mathbf{M}(i, j) \approx \mathbf{u}_i \cdot \mathbf{v}_j$$

Tensor SAT Reconstruction

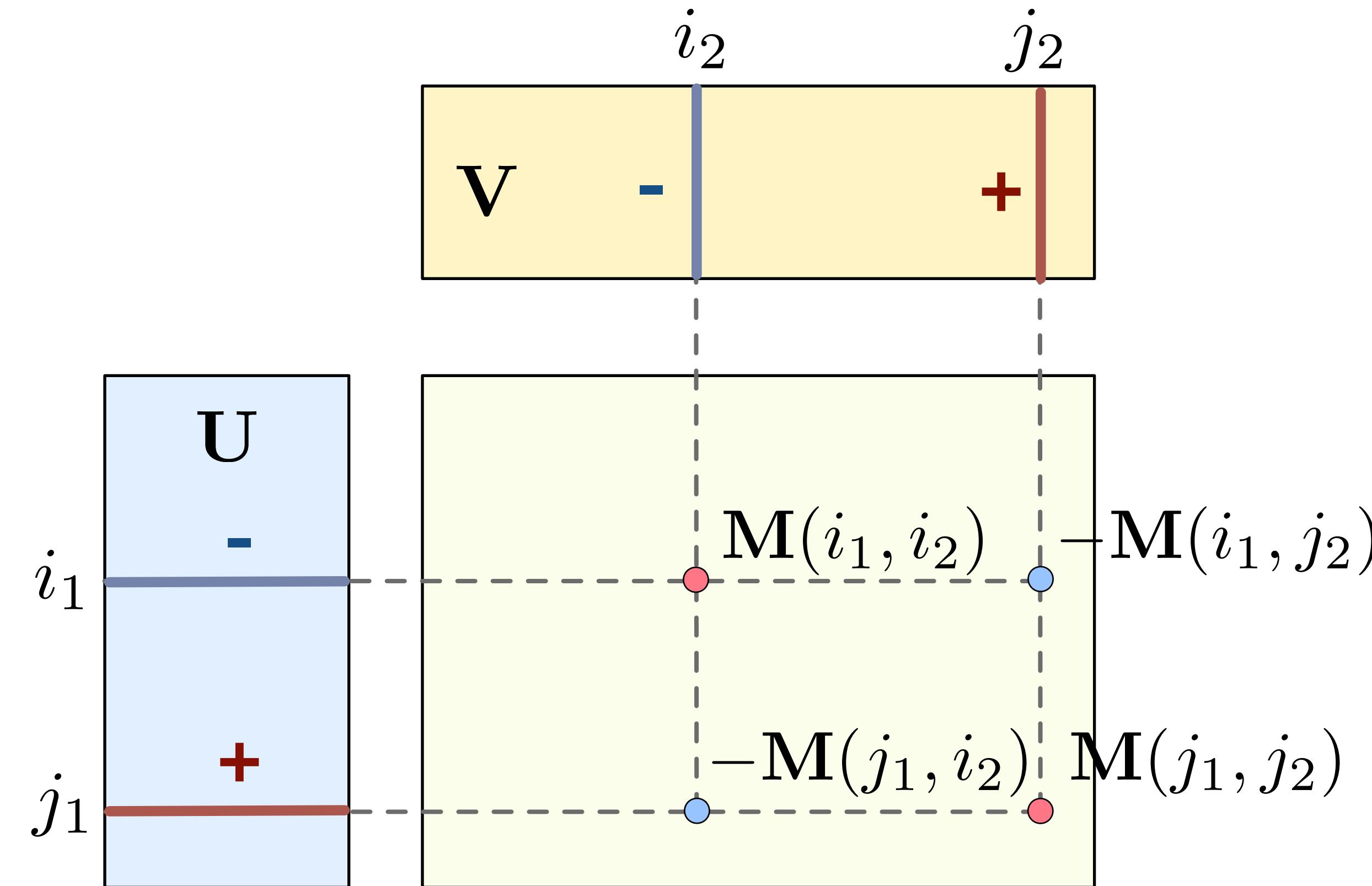


Tensor SAT Reconstruction



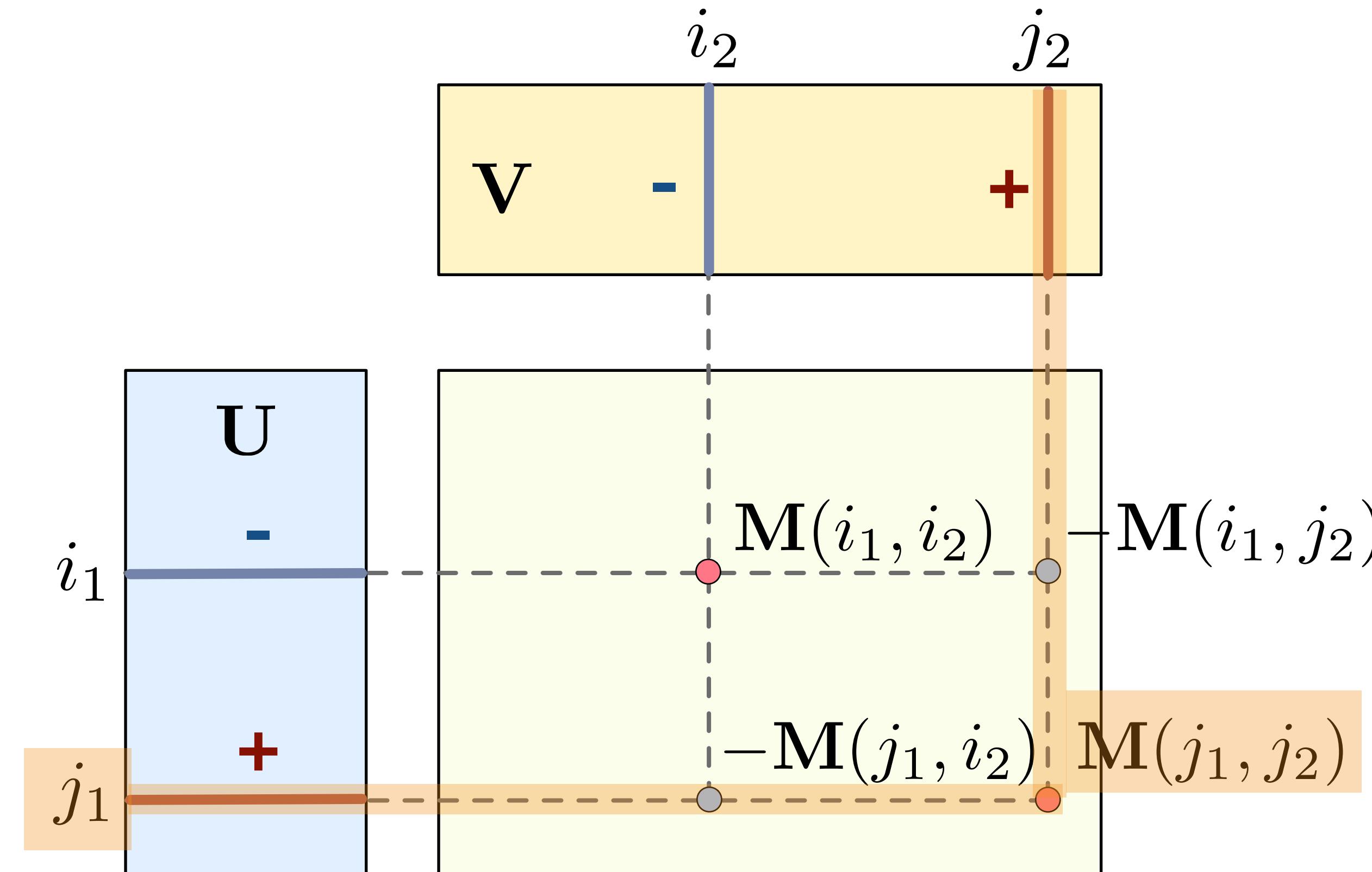
$$\mathbf{M}(j_1, j_2) - \mathbf{M}(j_1, i_2) - \mathbf{M}(i_1, j_2) + \mathbf{M}(i_1, i_2)$$

Tensor SAT Reconstruction



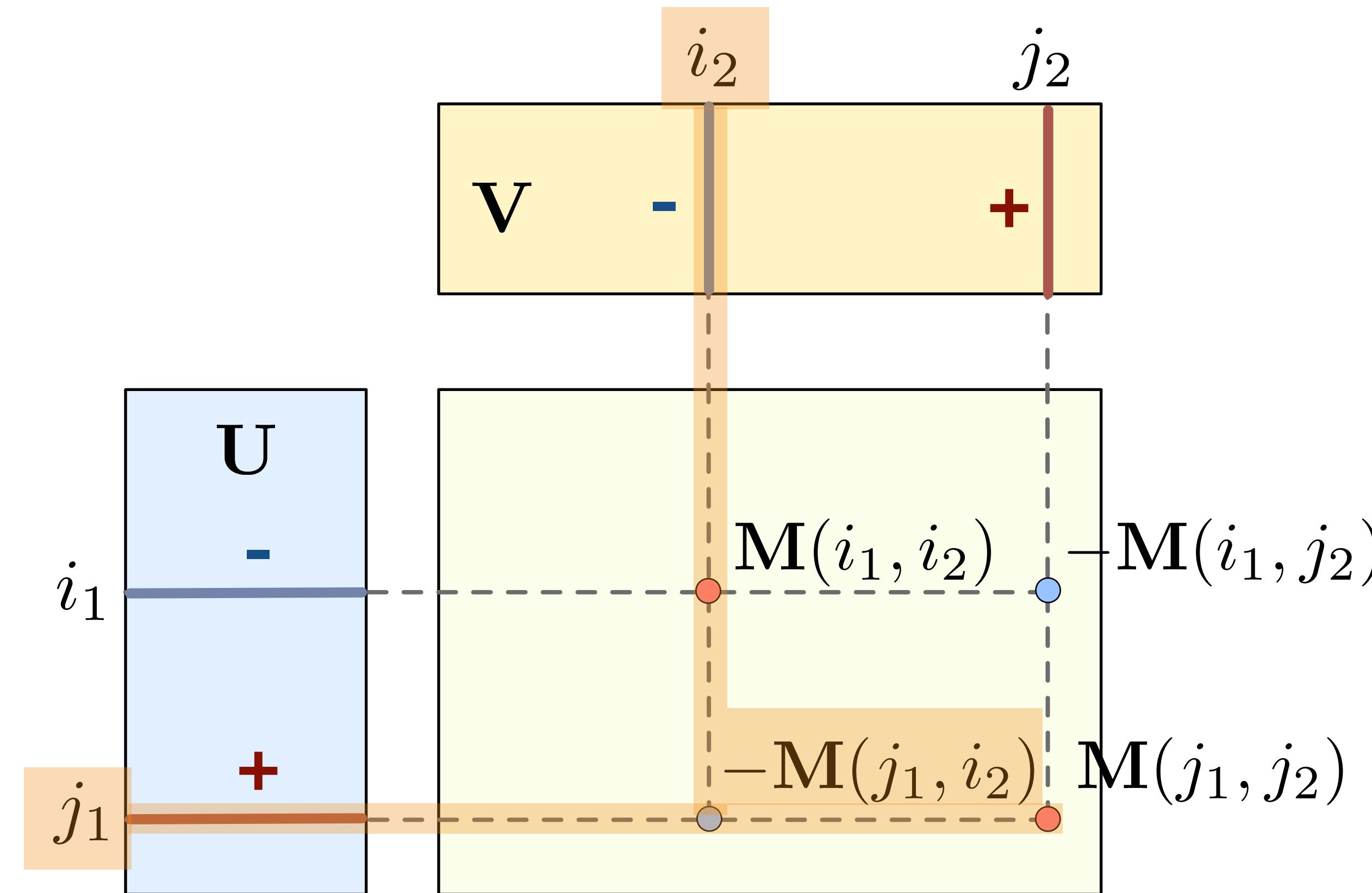
$$(\mathbf{u}_{j_1} - \mathbf{u}_{i_1}) \cdot (\mathbf{v}_{j_2} - \mathbf{v}_{i_2}) \approx \mathbf{M}(j_1, j_2) - \mathbf{M}(j_1, i_2) - \mathbf{M}(i_1, j_2) + \mathbf{M}(i_1, i_2)$$

Tensor SAT Reconstruction



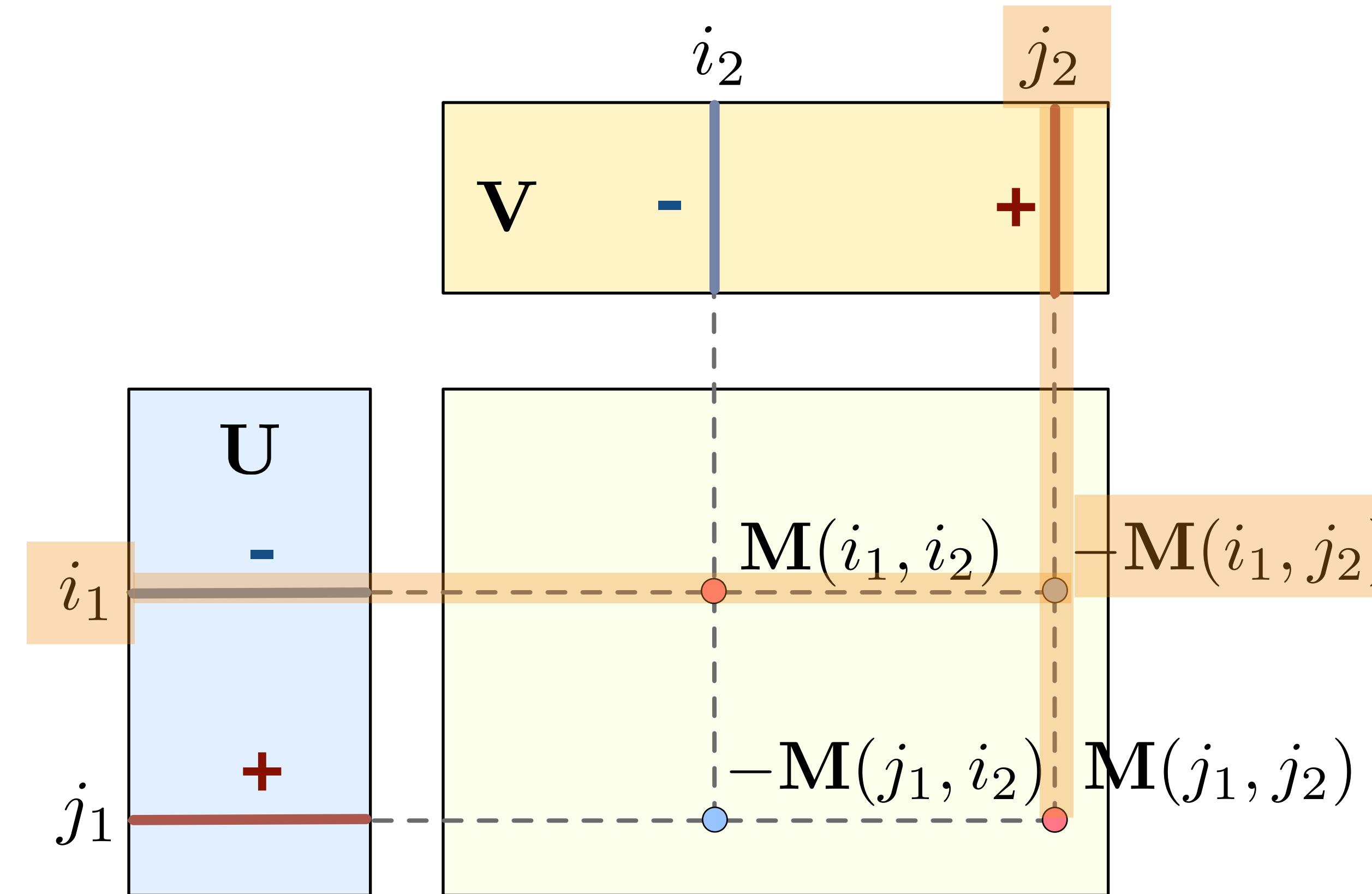
$$(\mathbf{u}_{j_1} - \mathbf{u}_{i_1}) \cdot (\mathbf{v}_{j_2} - \mathbf{v}_{i_2}) \approx \mathbf{M}(j_1, j_2) - \mathbf{M}(j_1, i_2) - \mathbf{M}(i_1, j_2) + \mathbf{M}(i_1, i_2)$$

Tensor SAT Reconstruction



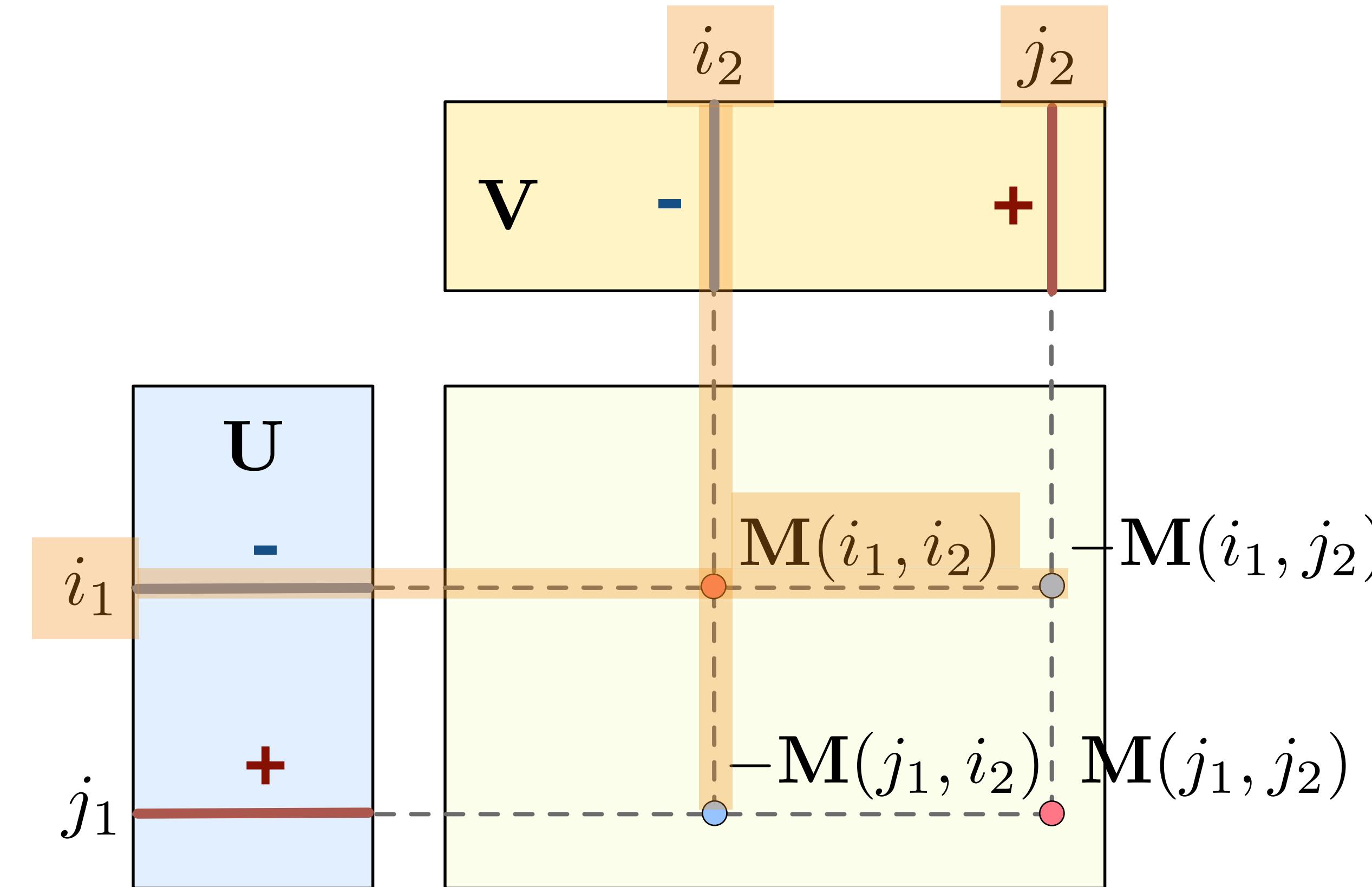
$$(\mathbf{u}_{j_1} - \mathbf{u}_{i_1}) \cdot (\mathbf{v}_{j_2} - \mathbf{v}_{i_2}) \approx \mathbf{M}(j_1, j_2) - \mathbf{M}(j_1, i_2) - \mathbf{M}(i_1, j_2) + \mathbf{M}(i_1, i_2)$$

Tensor SAT Reconstruction



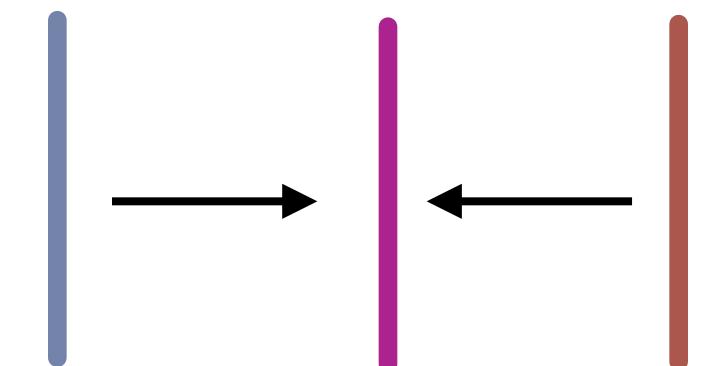
$$(\mathbf{u}_{j_1} - \mathbf{u}_{i_1}) \cdot (\mathbf{v}_{j_2} - \mathbf{v}_{i_2}) \approx \mathbf{M}(j_1, j_2) - \mathbf{M}(j_1, i_2) - \mathbf{M}(i_1, j_2) + \mathbf{M}(i_1, i_2)$$

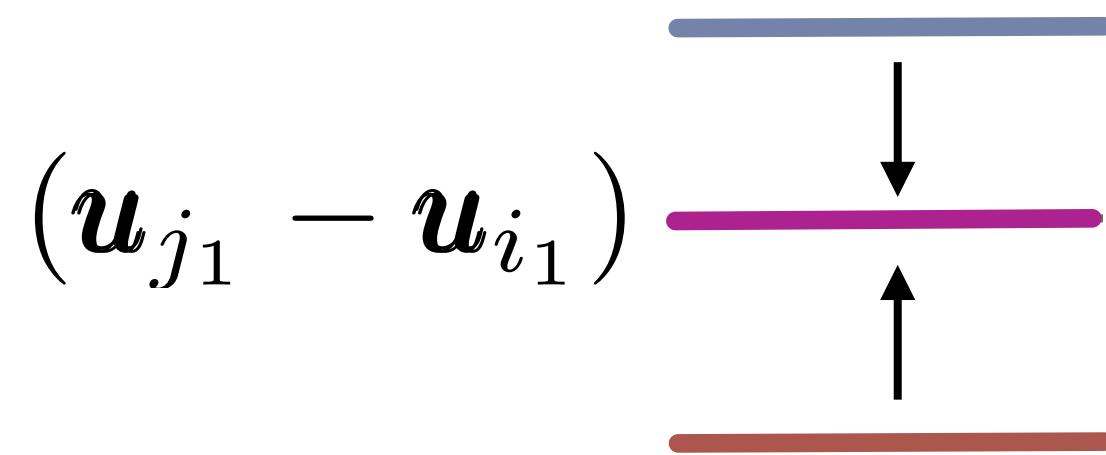
Tensor SAT Reconstruction



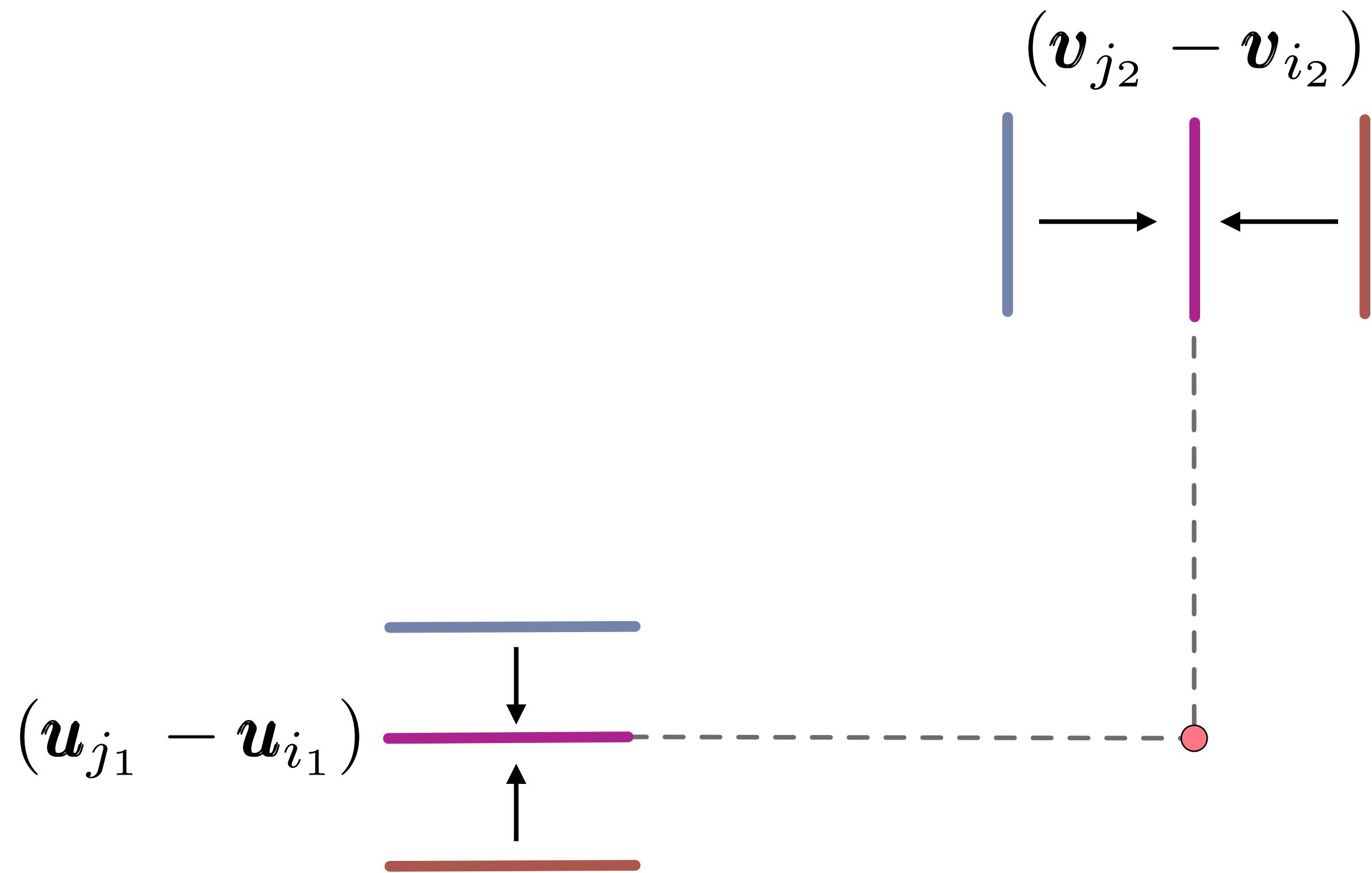
$$(\mathbf{u}_{j_1} - \mathbf{u}_{i_1}) \cdot (\mathbf{v}_{j_2} - \mathbf{v}_{i_2}) \approx \mathbf{M}(j_1, j_2) - \mathbf{M}(j_1, i_2) - \mathbf{M}(i_1, j_2) + \mathbf{M}(i_1, i_2)$$

Tensor SAT Reconstruction

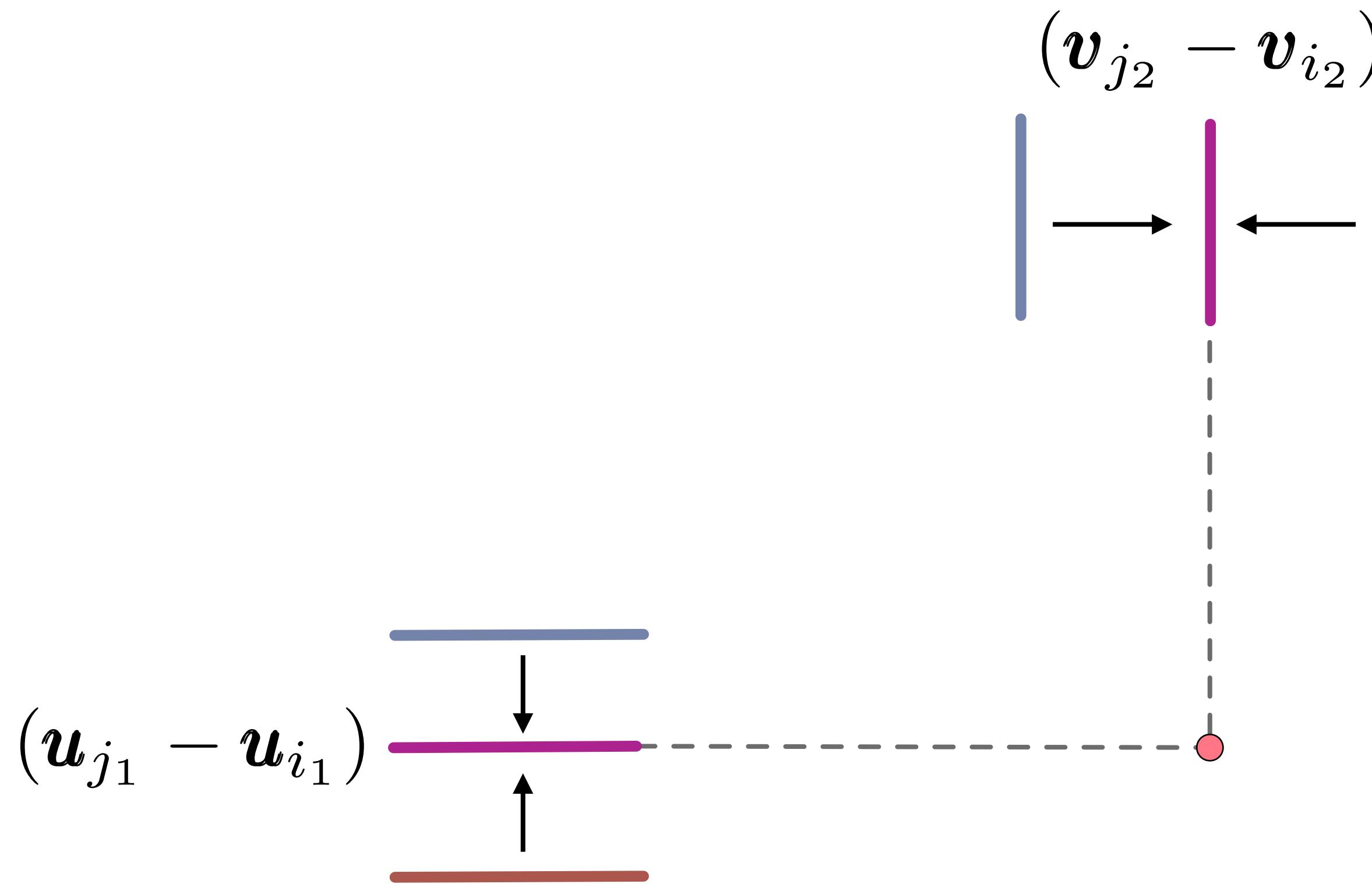
$$(v_{j_2} - v_{i_2})$$


$$(u_{j_1} - u_{i_1})$$


Tensor SAT Reconstruction



Tensor SAT Reconstruction



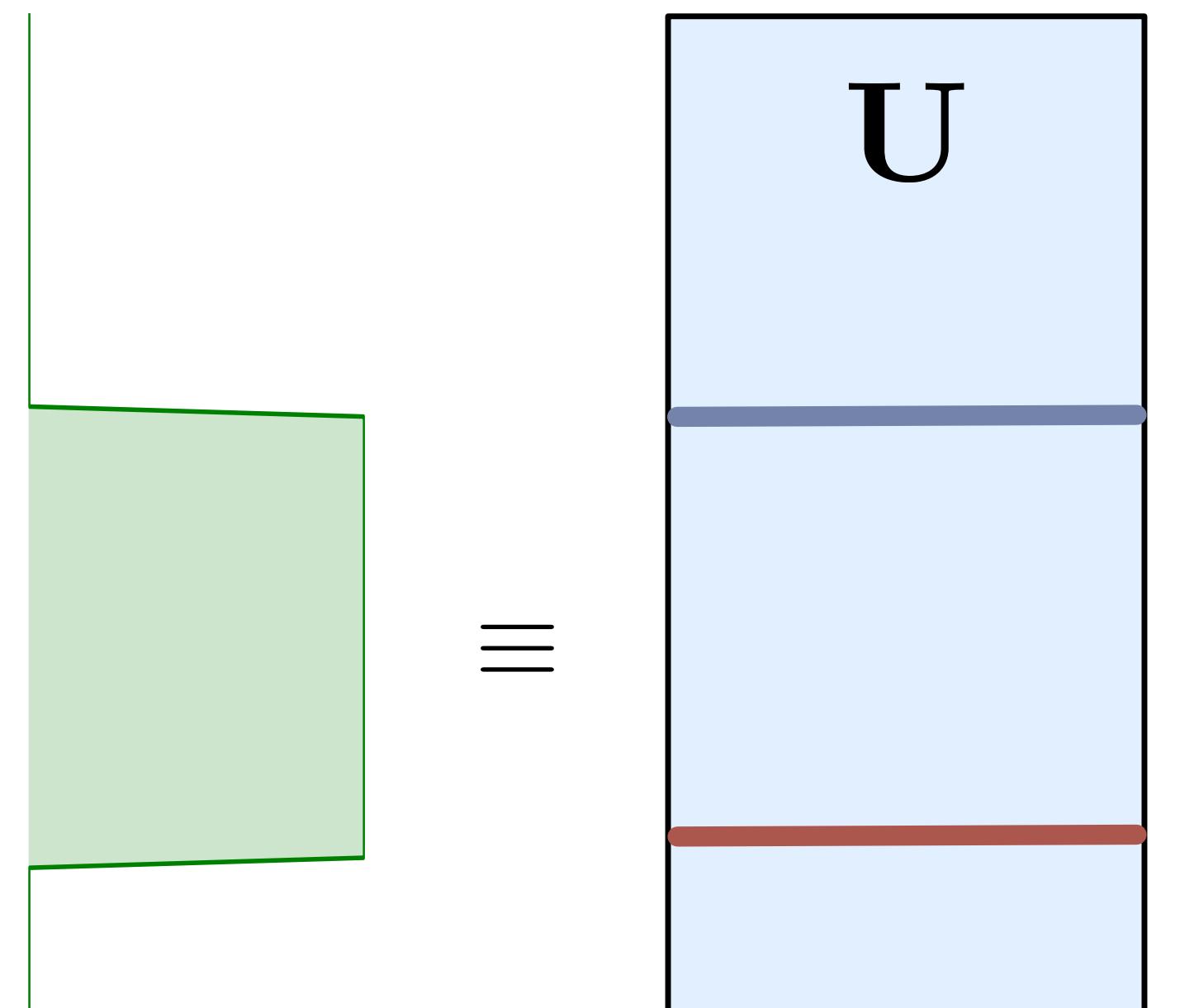
- Higher dimensions: matrix-vector (instead of vector-vector) products

Non-rectangular ROIs

- Convolve rows together, prior to subtraction

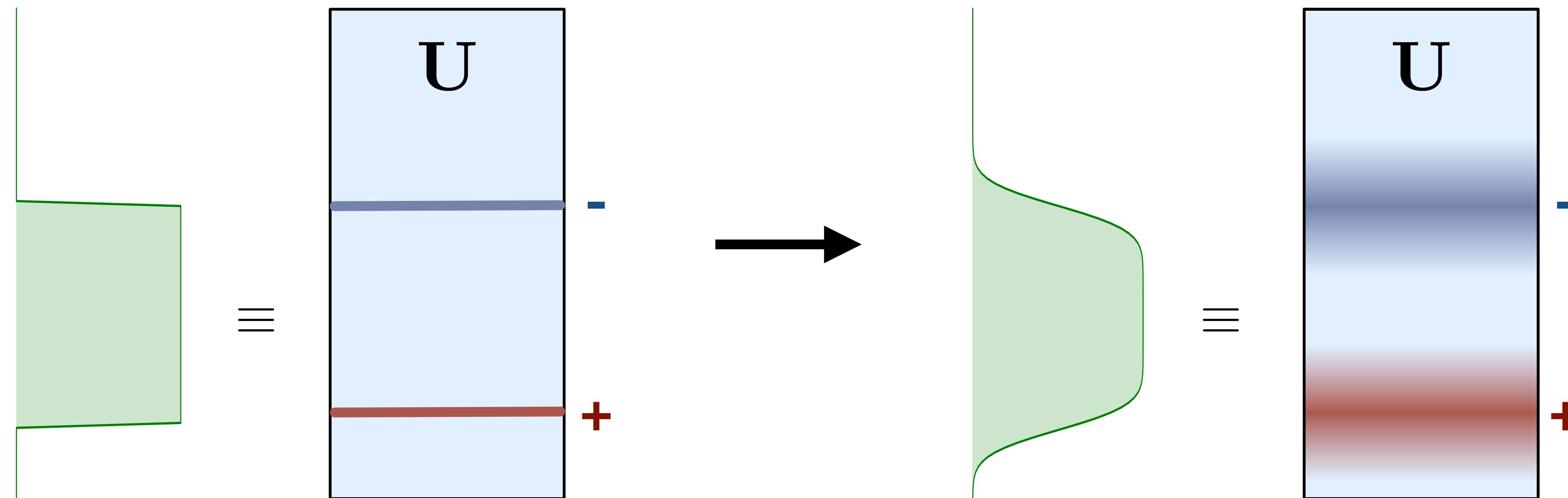
Non-rectangular ROIs

- Convolve rows together, prior to subtraction



Non-rectangular ROIs

- Convolve rows together, prior to subtraction



Histogram Fields

- We were doing: matrix-vector products

Histogram Fields

- We were doing: matrix-vector products
- What if we want to reconstruct **many histograms** at once?

Histogram Fields

- We were doing: matrix-vector products
- What if we want to reconstruct **many histograms** at once?
- Tensor-tensor products (**tensor contractions**)
 - ▶ Generalize matrix-matrix products
 - ▶ Just as parallelizable

Histogram Fields

- We were doing: matrix-vector products
- What if we want to reconstruct **many histograms** at once?
- Tensor-tensor products (**tensor contractions**)
 - ▶ Generalize matrix-matrix products
 - ▶ Just as parallelizable

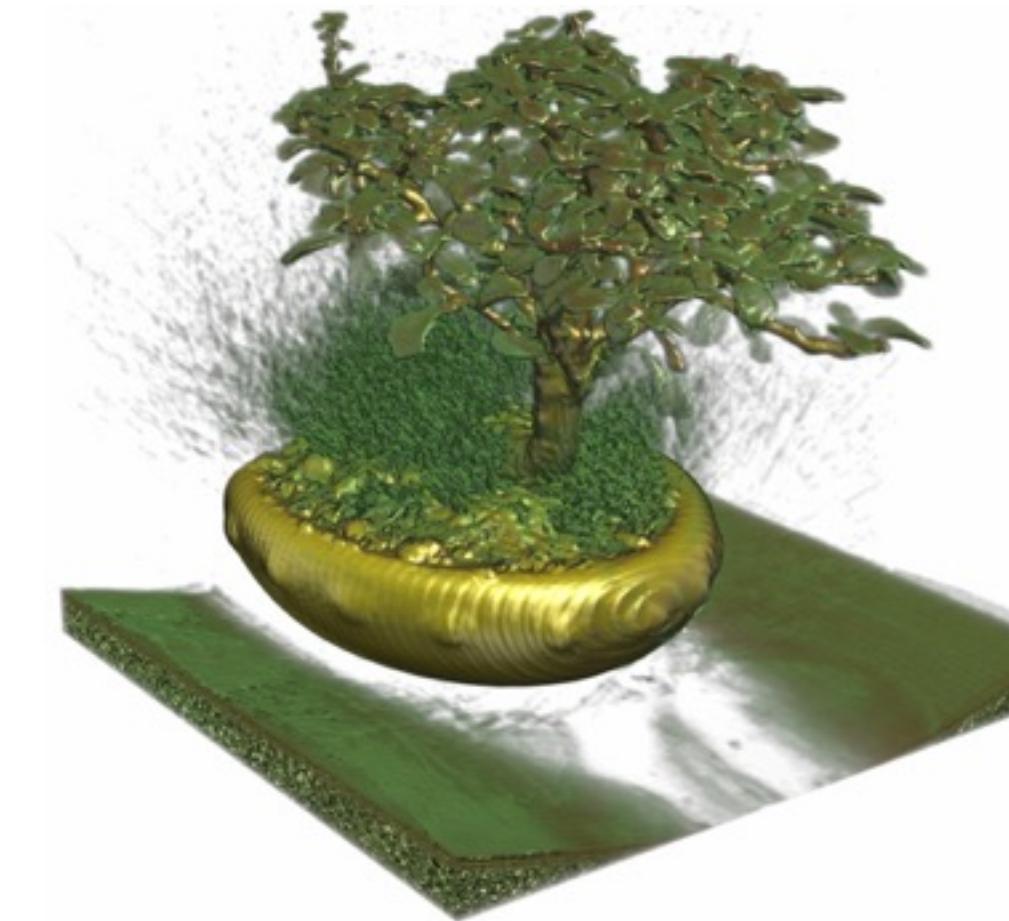
`tensorflow.einsum()` or `torch.einsum()` is all about tensor contraction!

Results

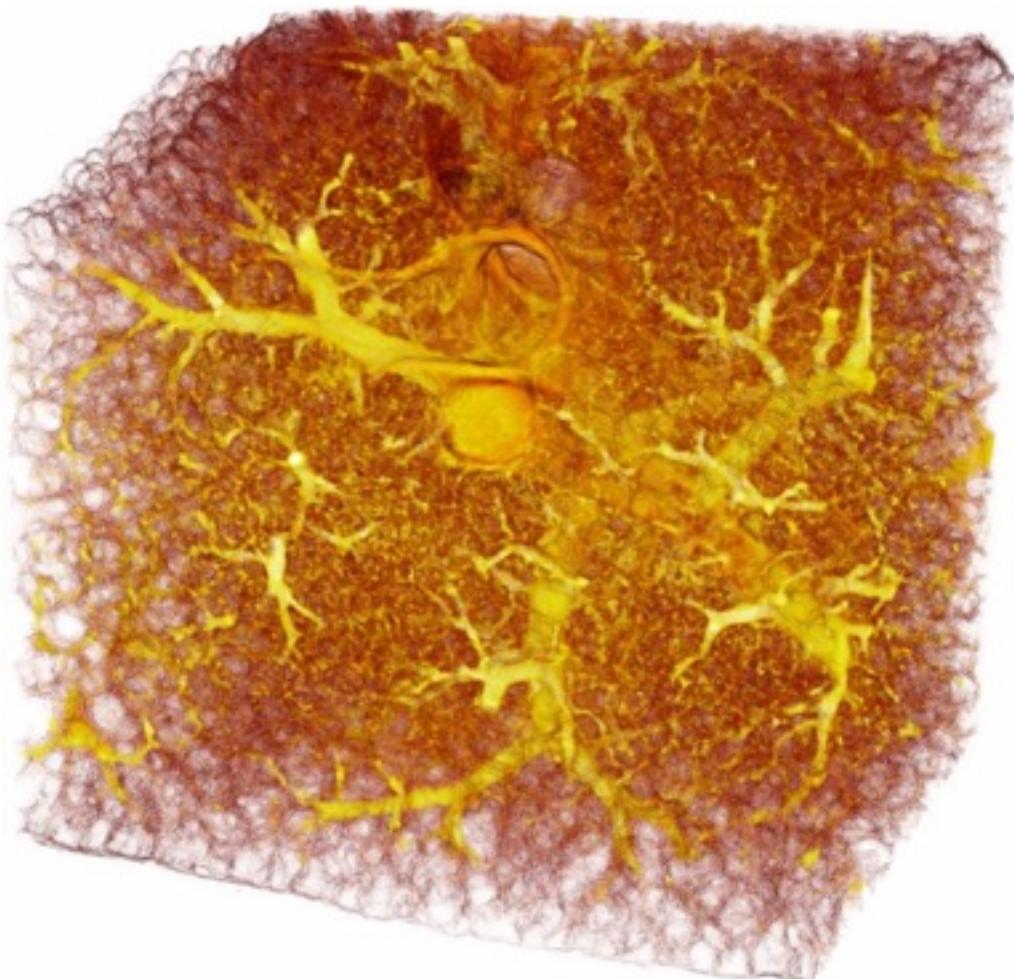
Scalar Fields



Waterfall, 4096^2 , 16MB
IH: 4GB, compressed 22:1



Bonsai, 256^3 , 256MB
IH: 8GB, compressed 214:1

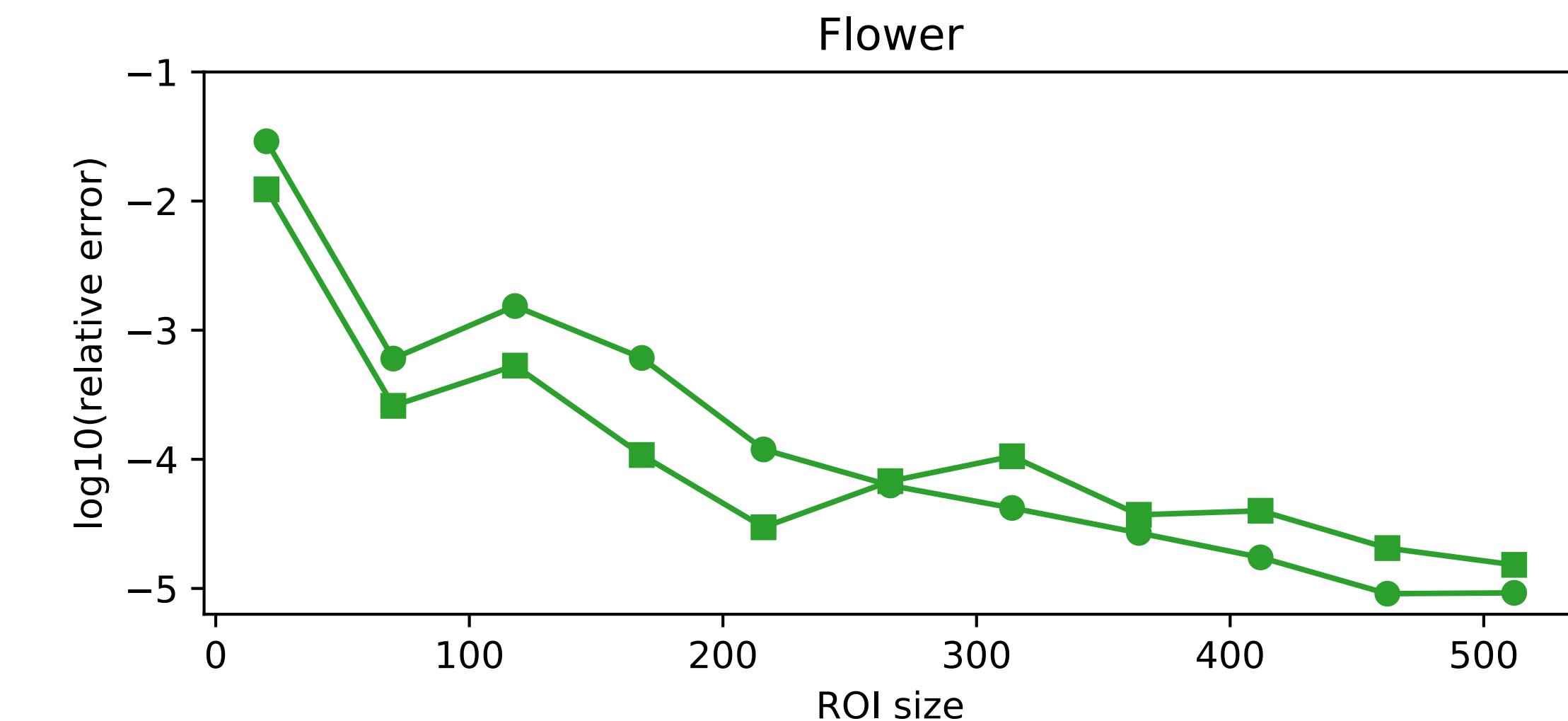
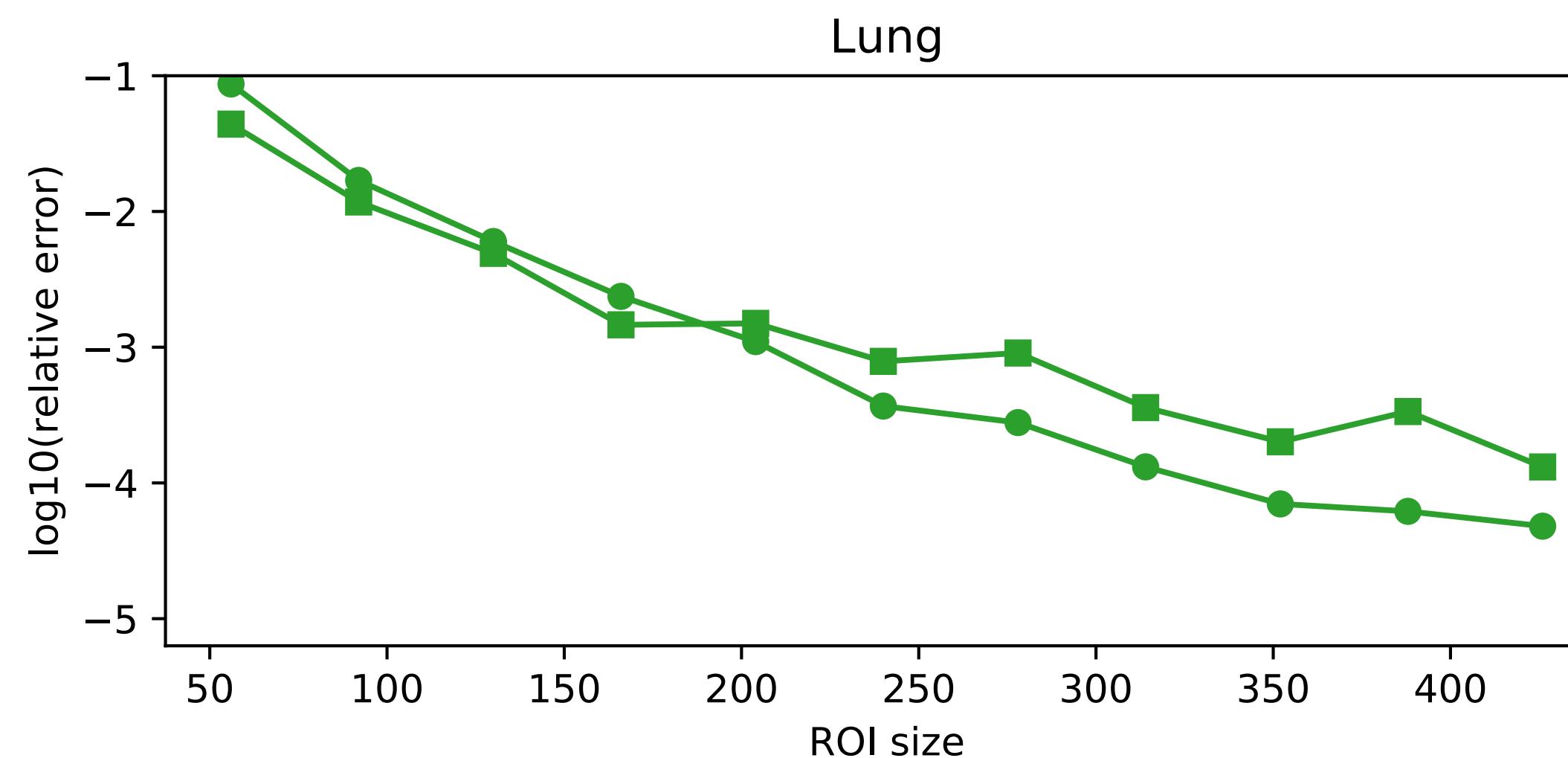
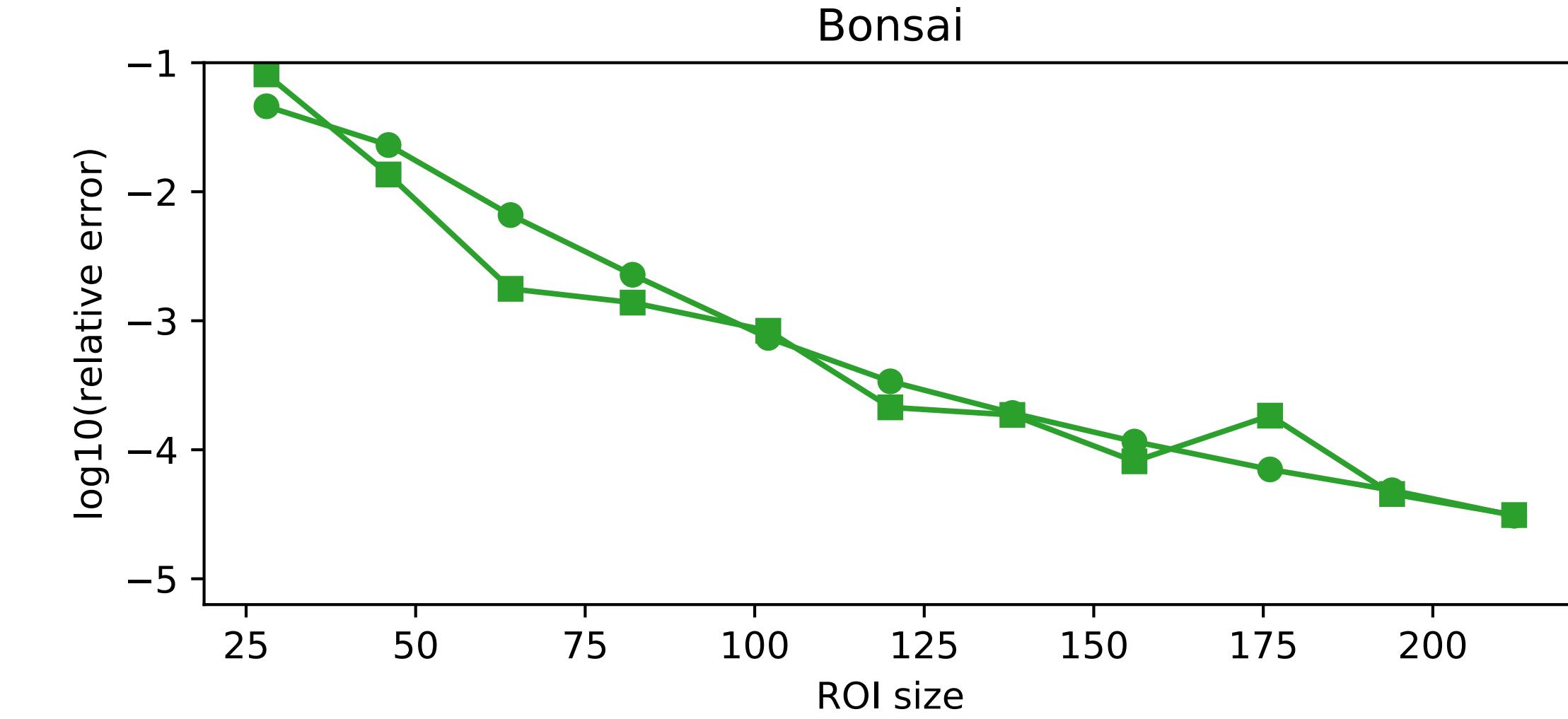
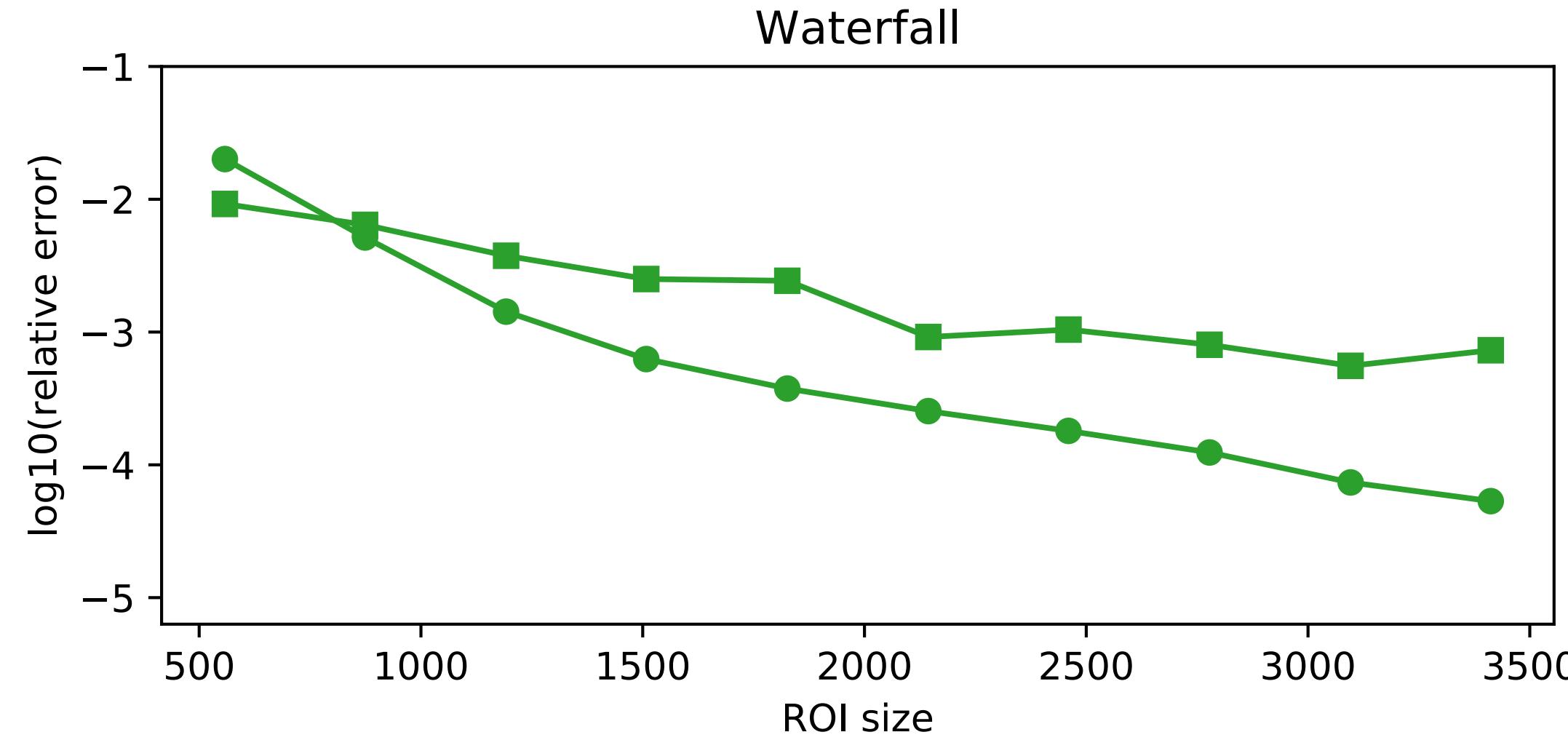


Lung, 512^3 , 128MB
IH: 64GB, compressed 319:1



Flower, 1024^3 , 1GB
IH: 512GB, compressed 766:1

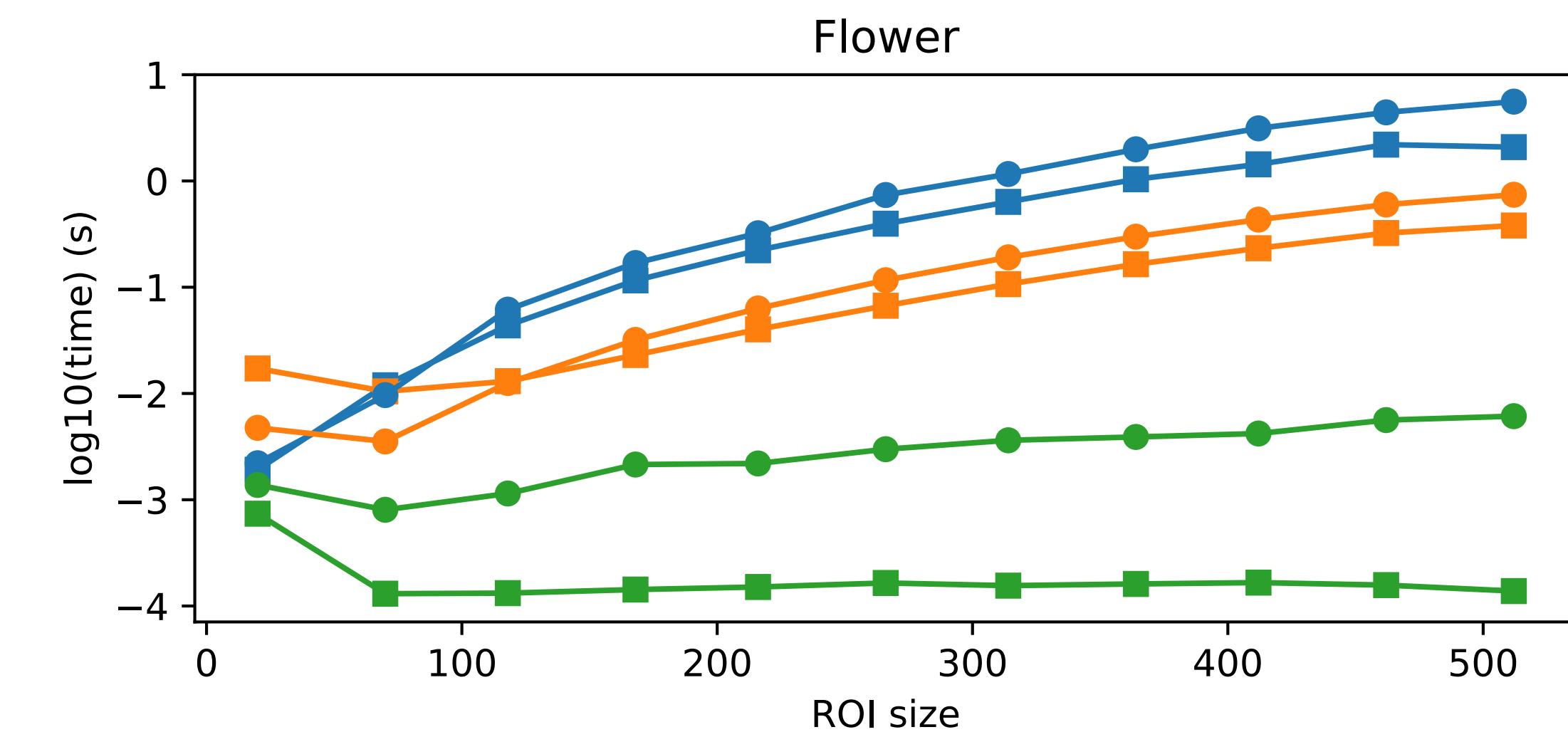
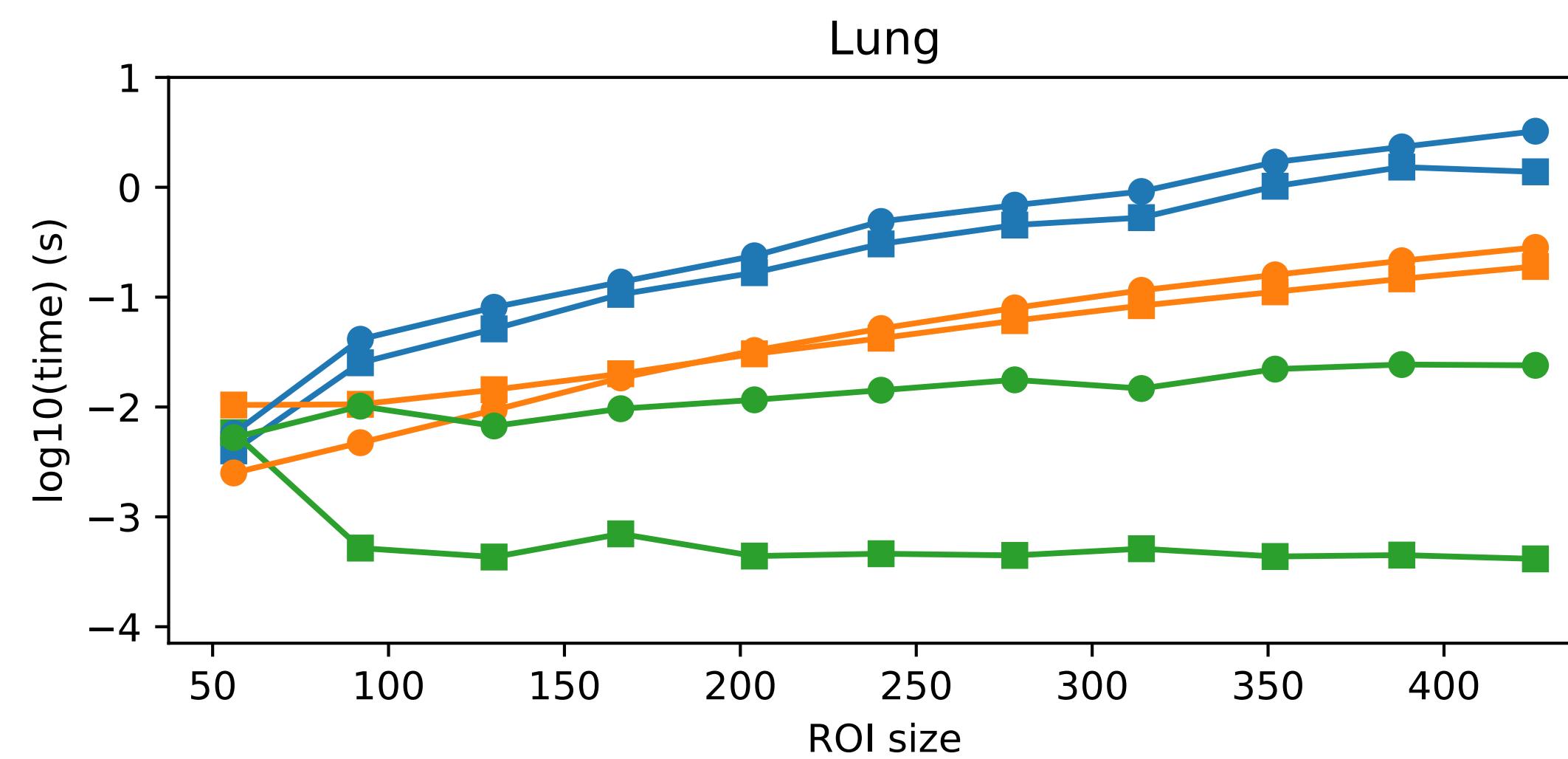
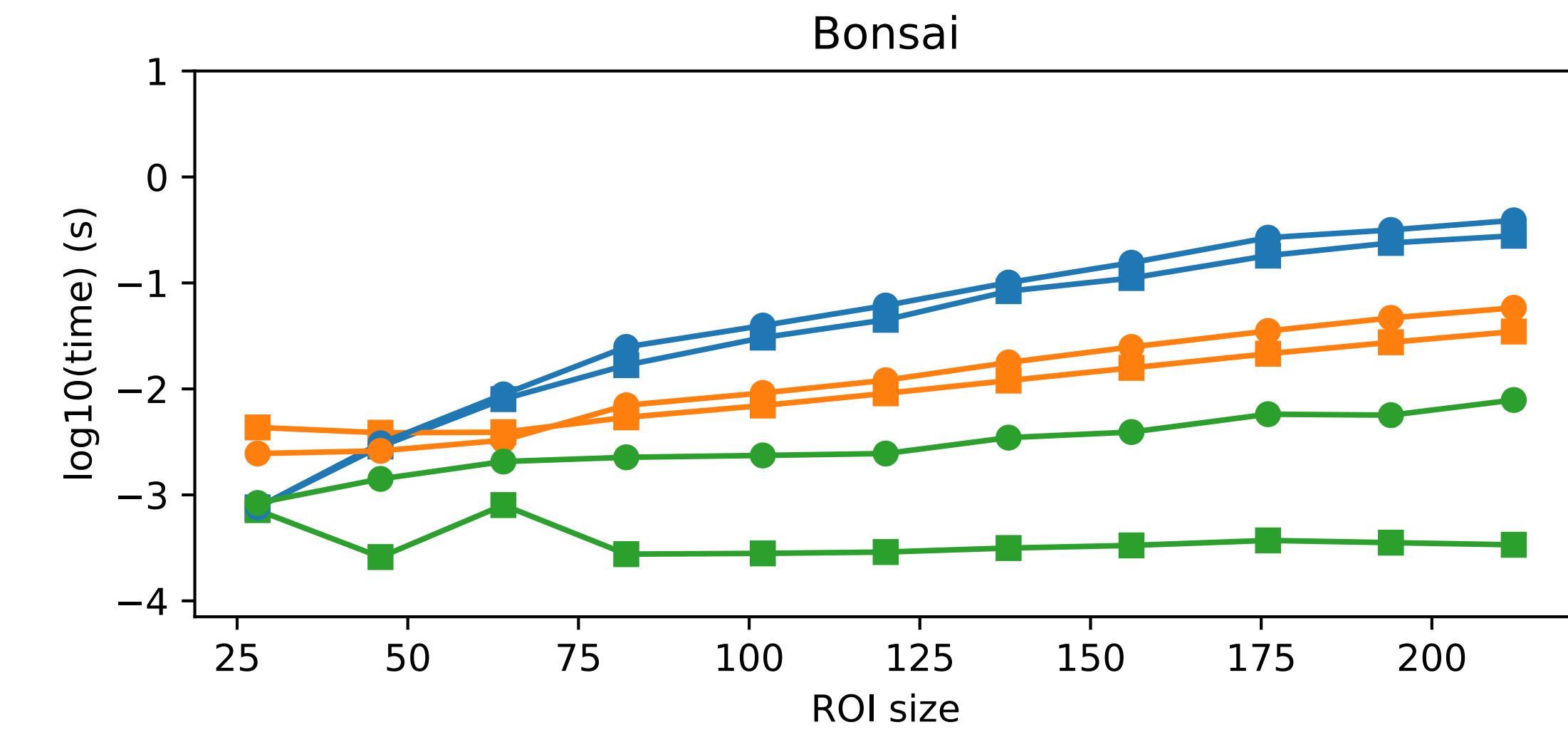
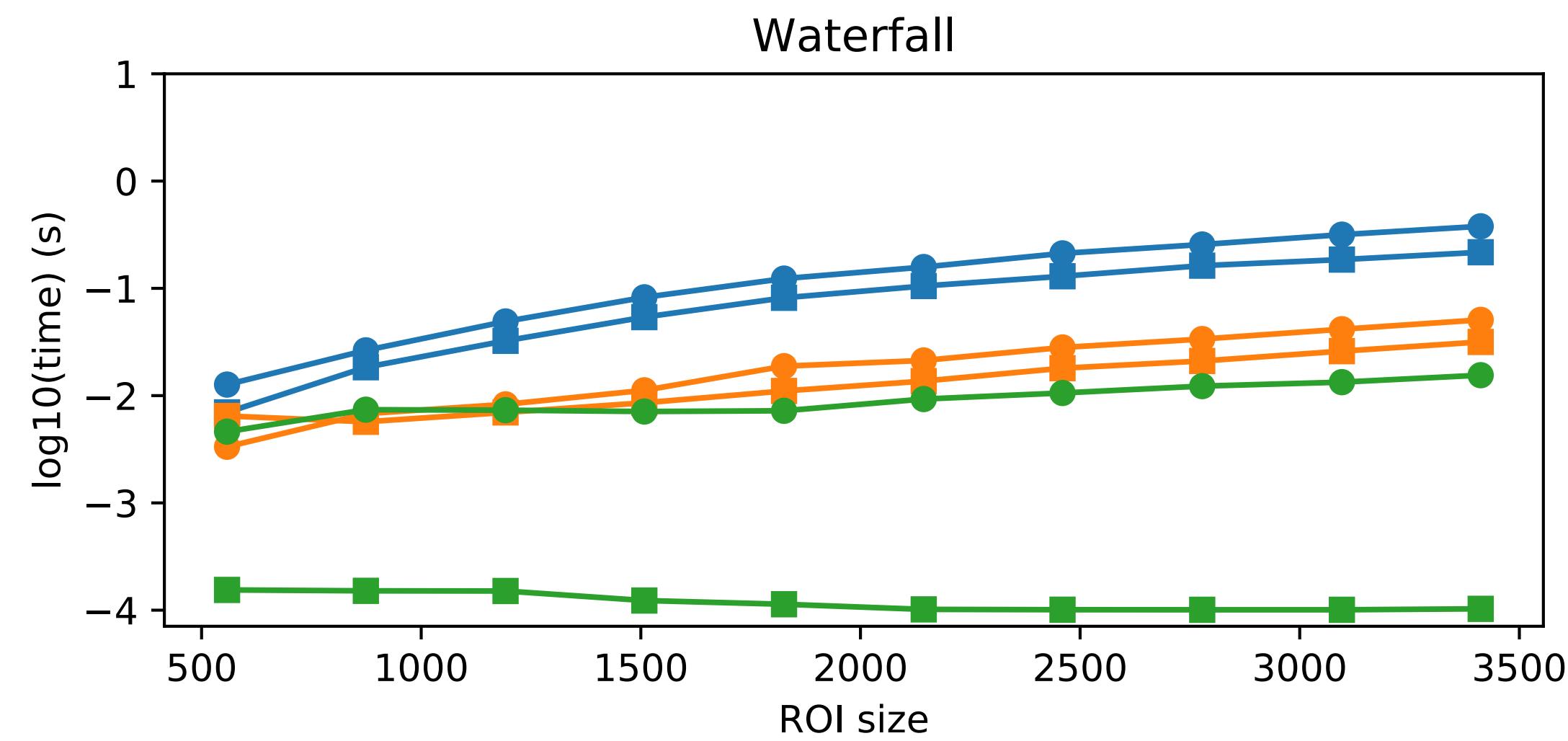
Scalar Fields: Errors



Box ROI

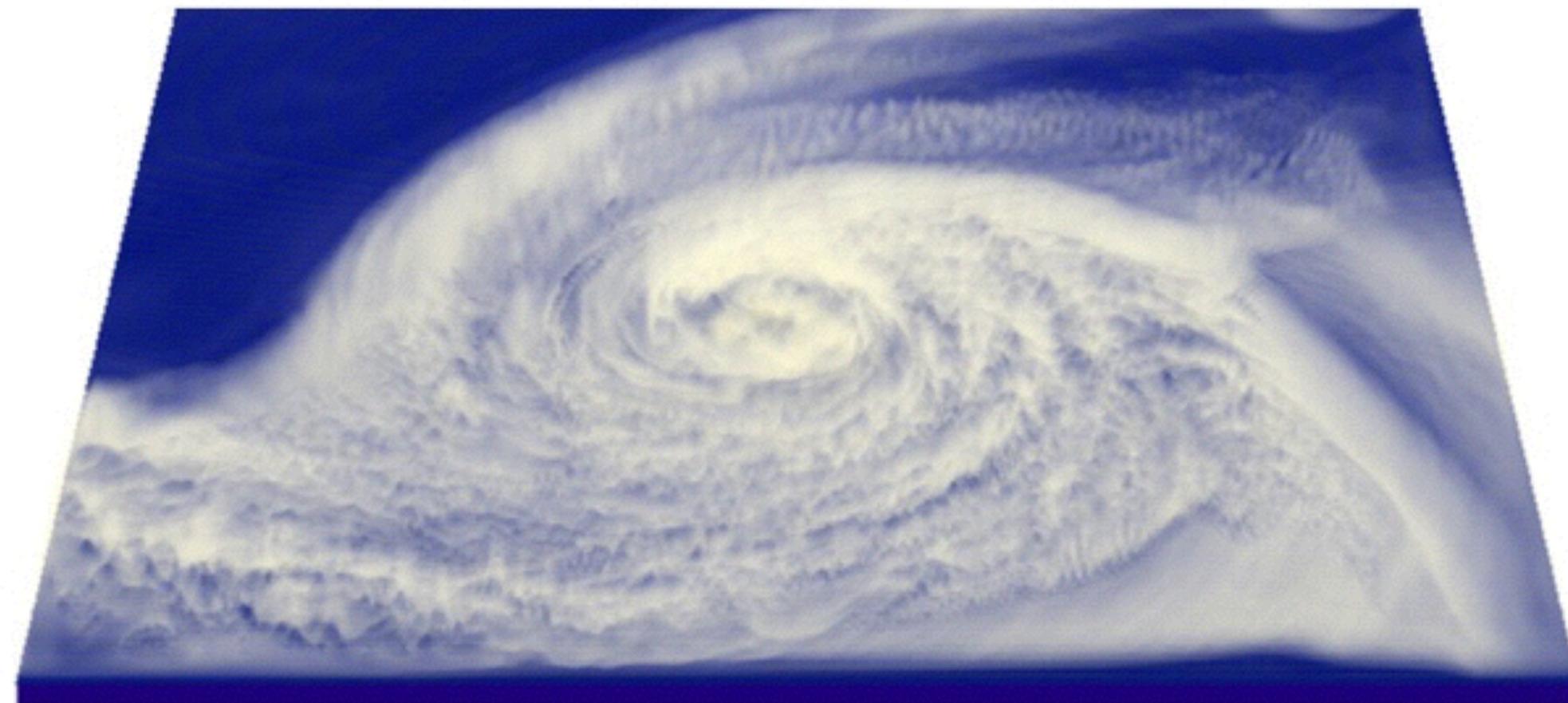
Gaussian ROI

Scalar Fields: Timings



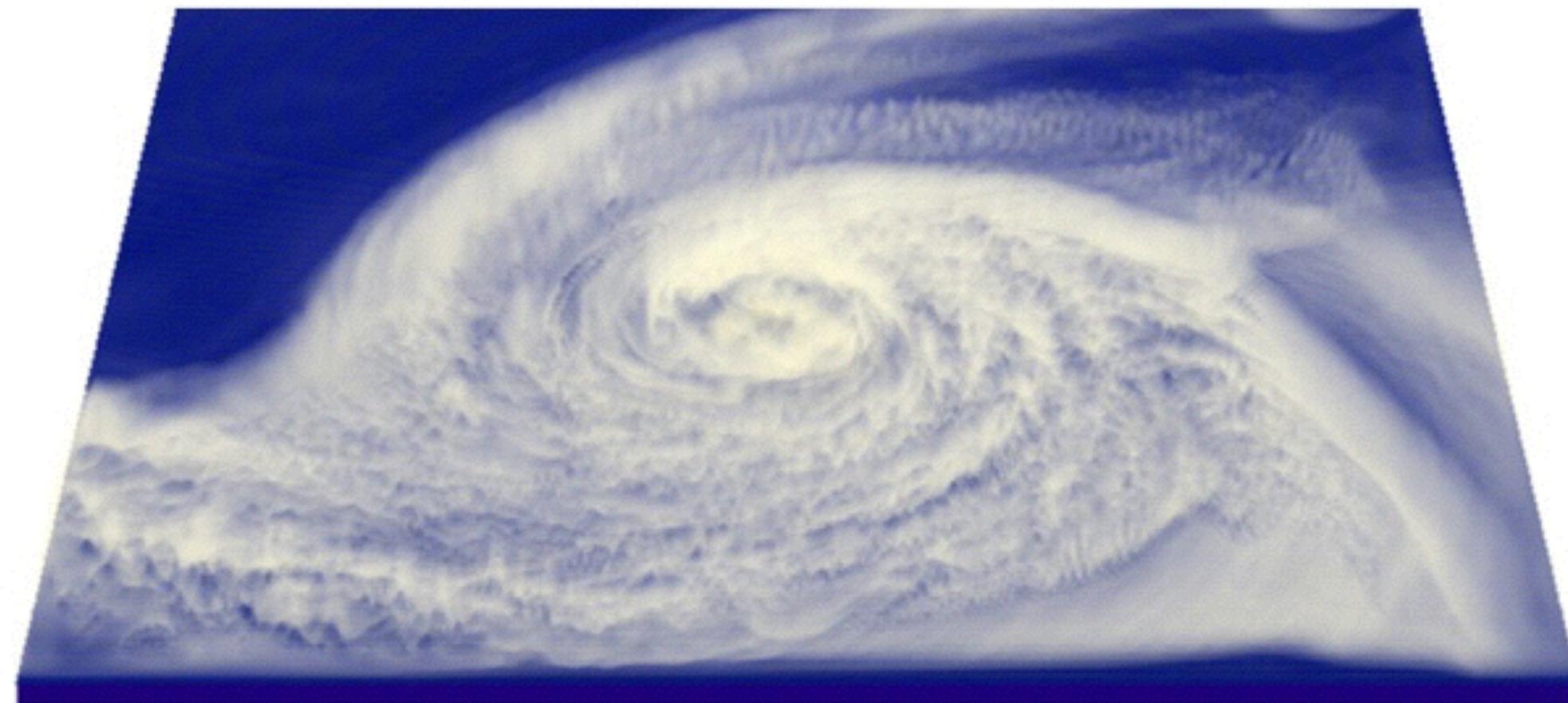
Vector Field Histograms

- Use case: **wind direction** in hurricane Isabel



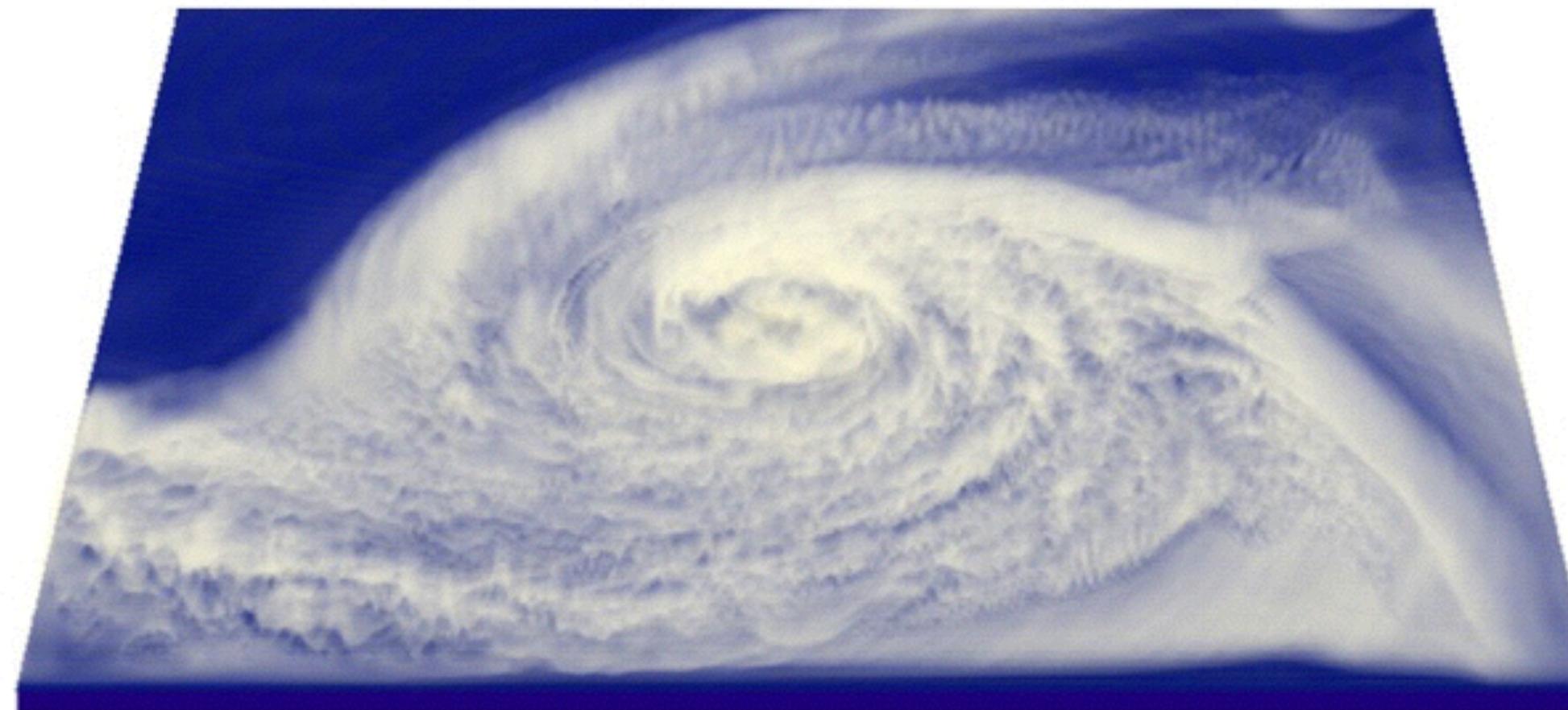
Vector Field Histograms

- Use case: **wind direction** in hurricane Isabel
- Dense and complex vector field — how to visualize?



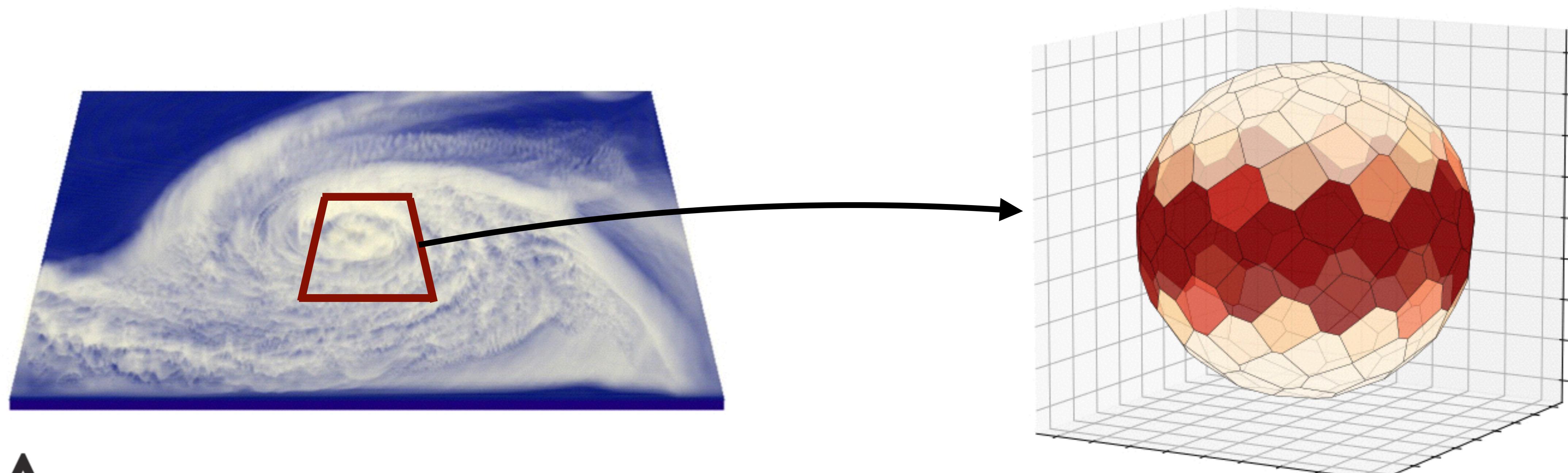
Vector Field Histograms

- Use case: **wind direction** in hurricane Isabel
- Dense and complex vector field — how to visualize?
- We quantize wind direction into 128 bins → **directional histograms**



Vector Field Histograms

- Use case: **wind direction** in hurricane Isabel
- Dense and complex vector field — how to visualize?
- We quantize wind direction into 128 bins → **directional histograms**
- Do that for each window → feature vectors capturing **local wind structure**



Normalized Cross-Correlation (NCC)

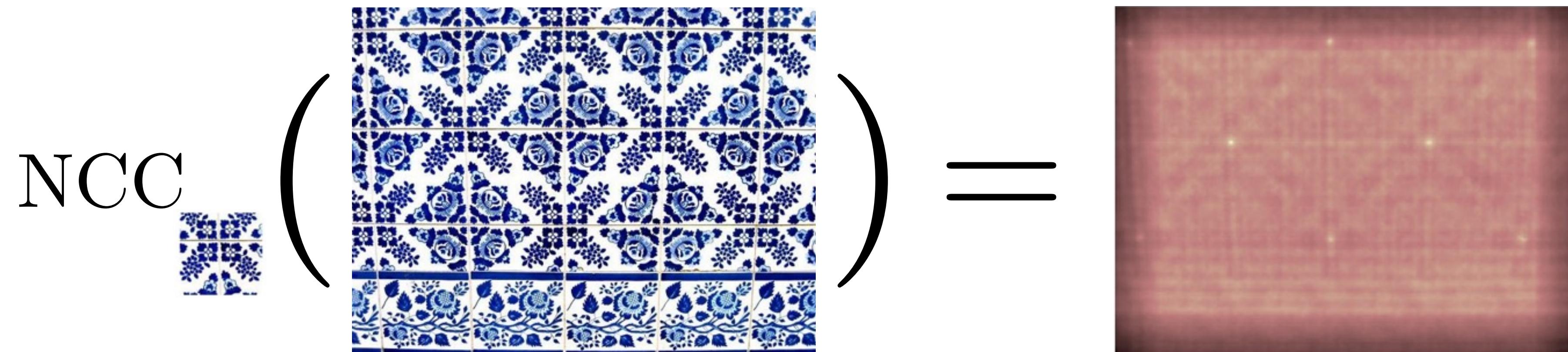
- Defined as cosine similarity between a template t and all feature vectors v :

$$\text{NCC}_t(v) = \frac{t \cdot v}{\|t\| \cdot \|v\|}$$

Normalized Cross-Correlation (NCC)

- Defined as cosine similarity between a template t and all feature vectors v :

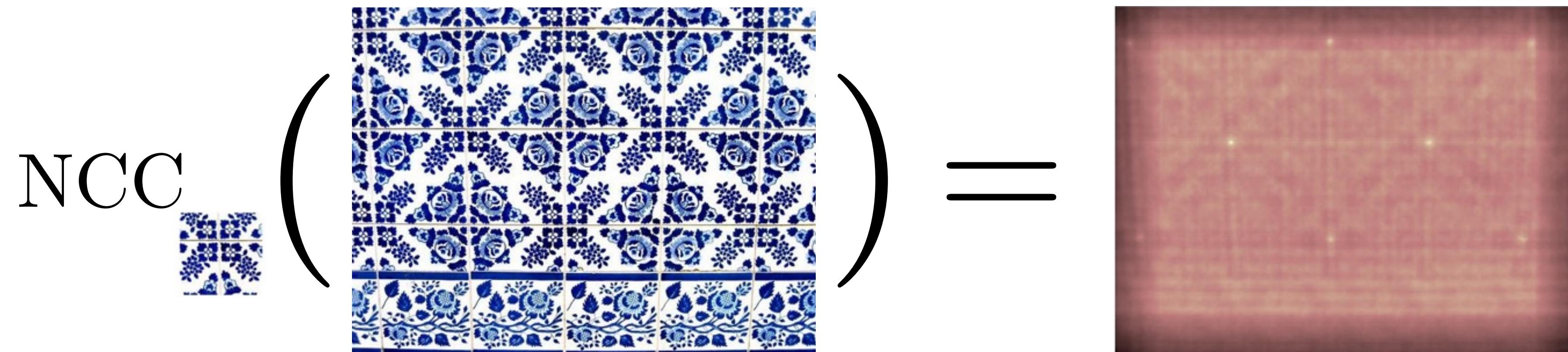
$$\text{NCC}_t(v) = \frac{\mathbf{t} \cdot \mathbf{v}}{\|\mathbf{t}\| \cdot \|\mathbf{v}\|}$$



Normalized Cross-Correlation (NCC)

- Defined as cosine similarity between a template t and all feature vectors v :

$$\text{NCC}_t(v) = \frac{t \cdot v}{\|t\| \cdot \|v\|}$$

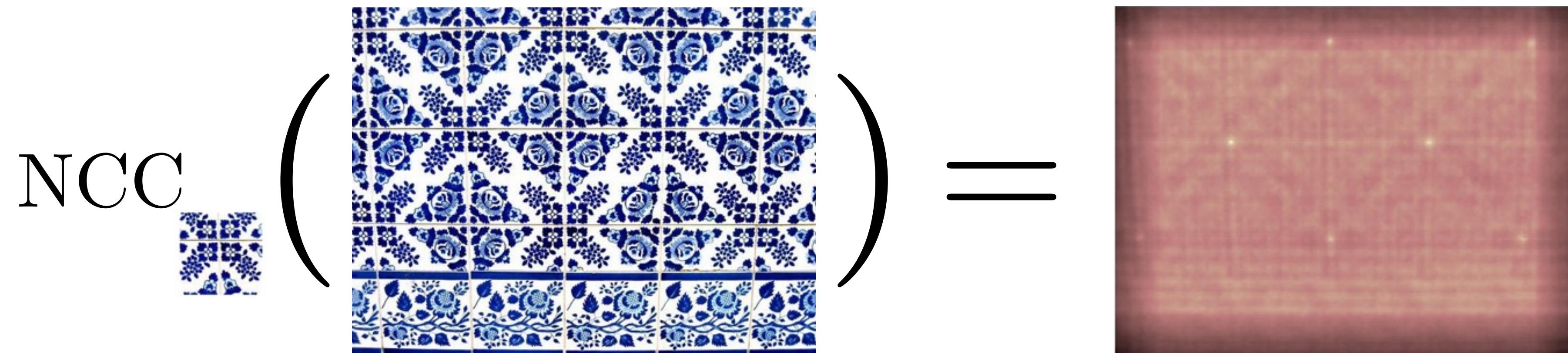


- For us: the feature vectors are directional histograms

Normalized Cross-Correlation (NCC)

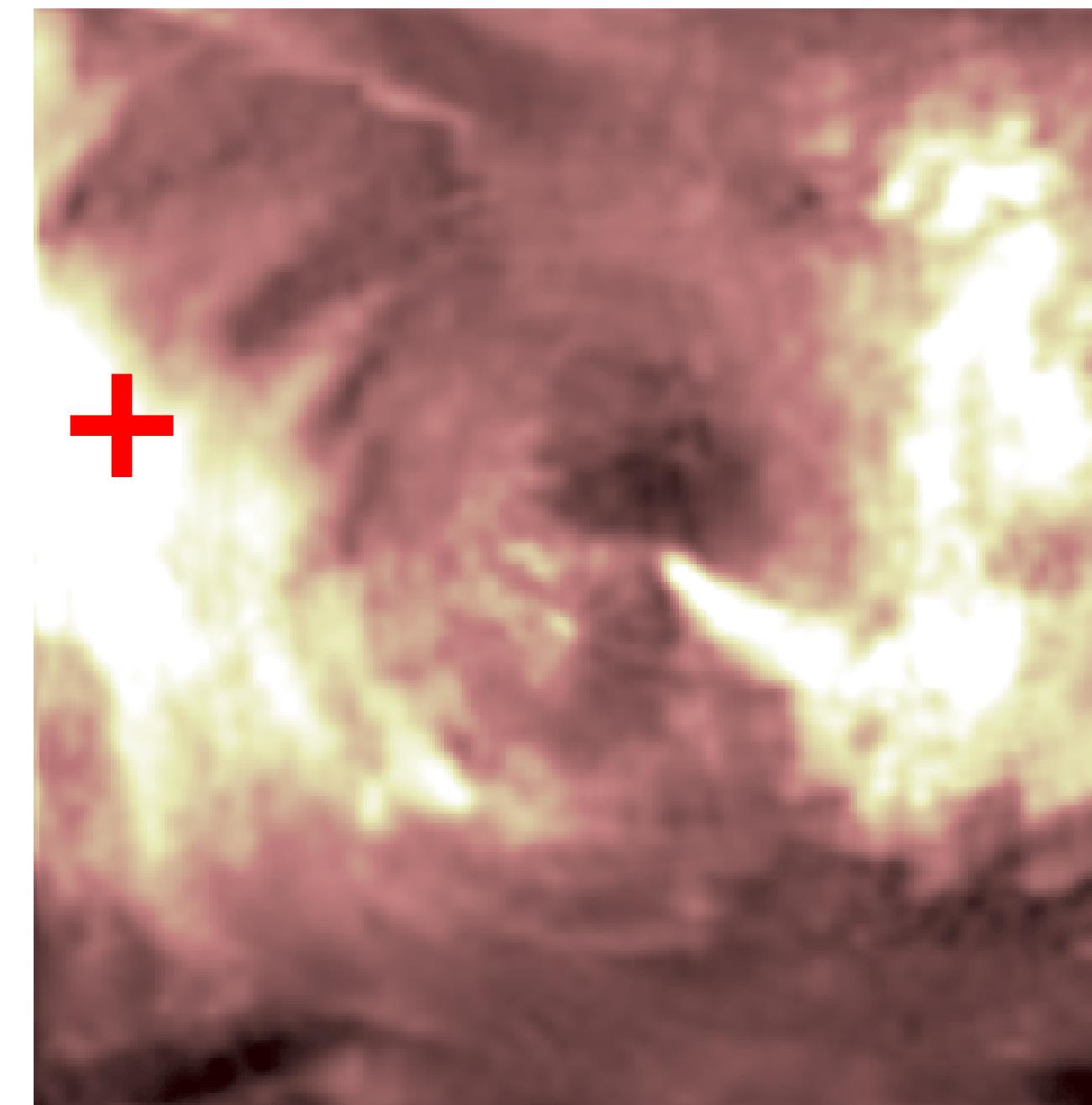
- Defined as cosine similarity between a template t and all feature vectors v :

$$\text{NCC}_t(v) = \frac{t \cdot v}{\|t\| \cdot \|v\|}$$

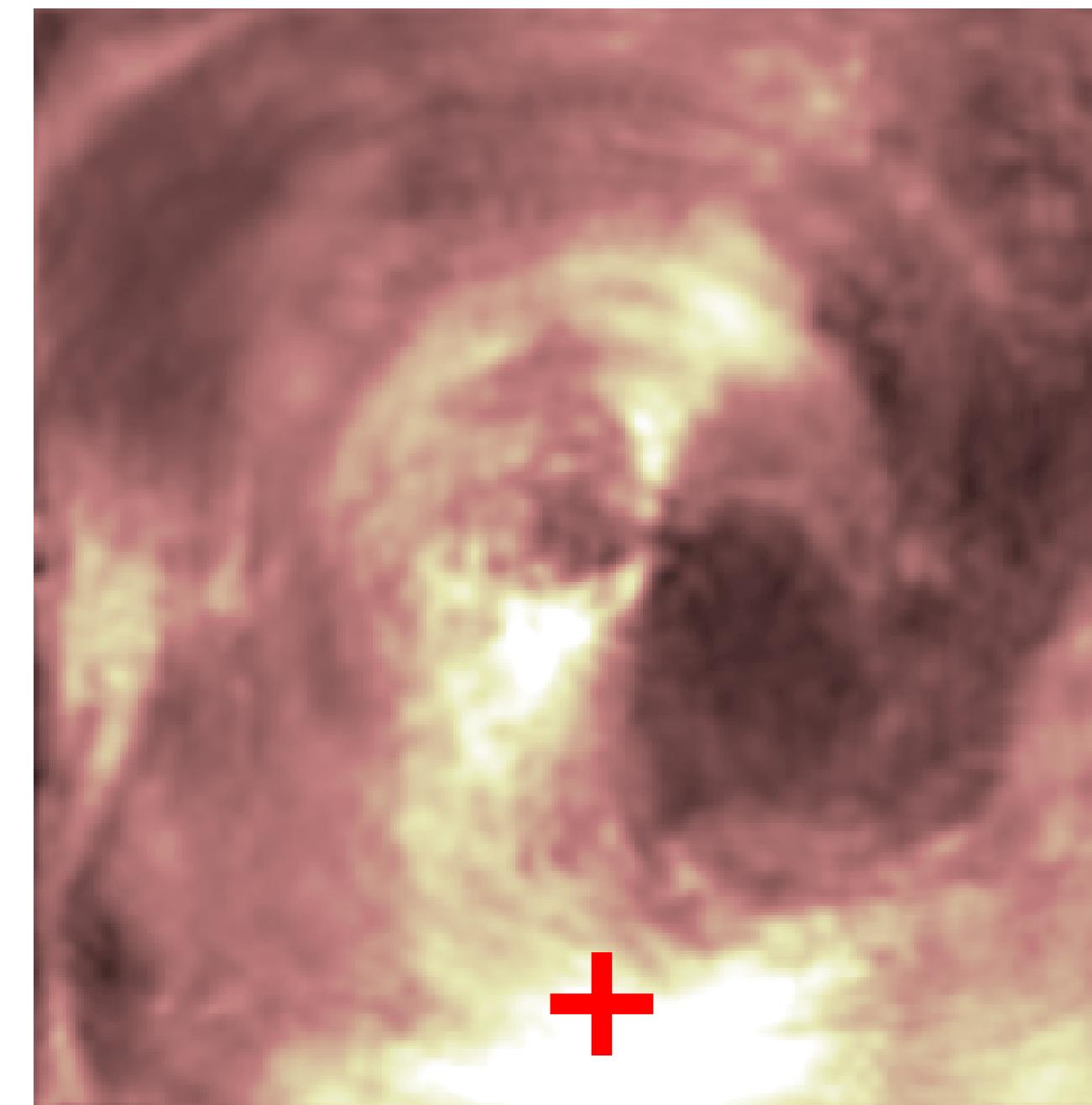


- For us: the feature vectors are directional histograms
- Template: user-defined by clicking on a neighborhood

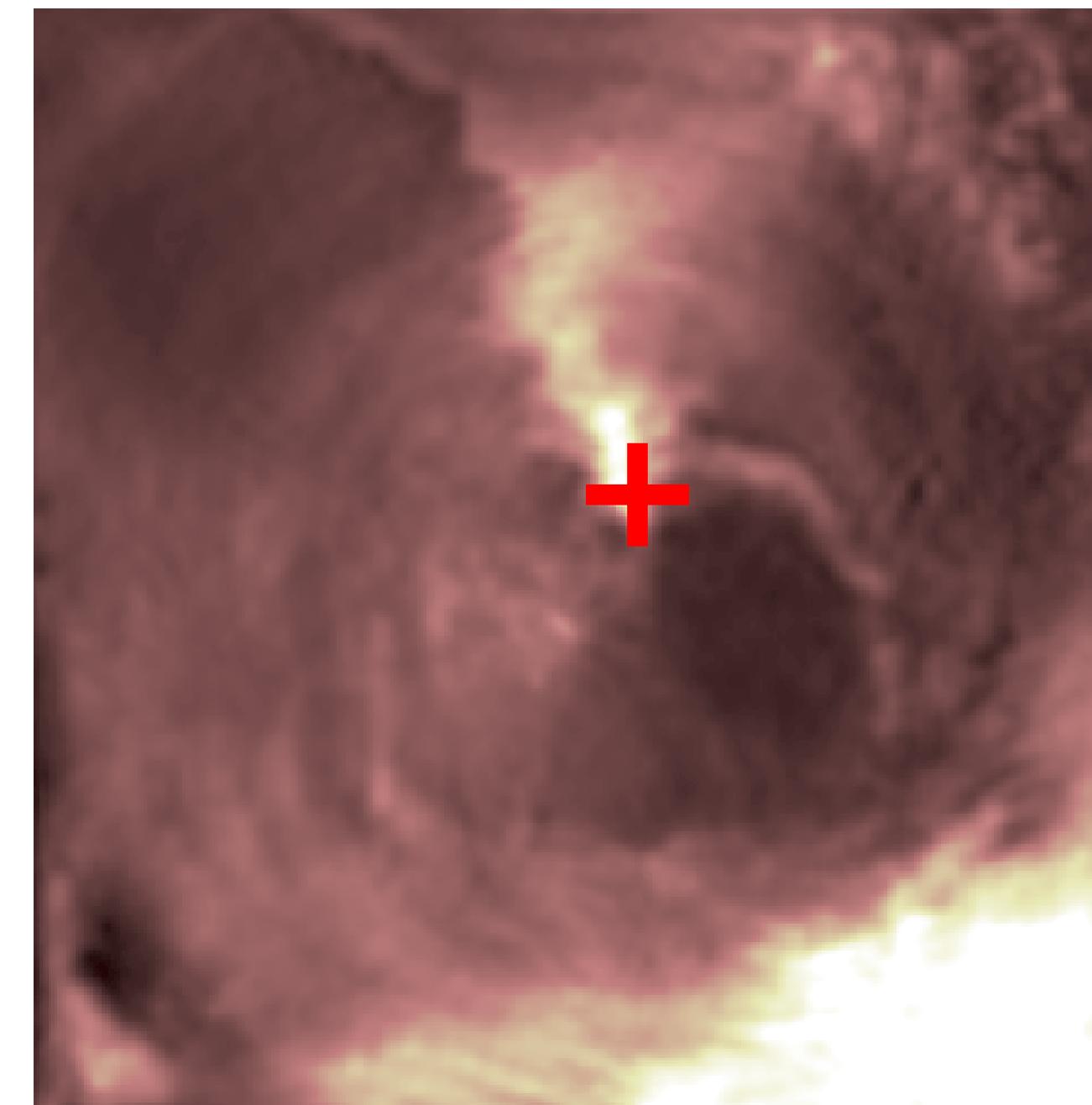
Normalized Cross-Correlation (NCC)



Normalized Cross-Correlation (NCC)

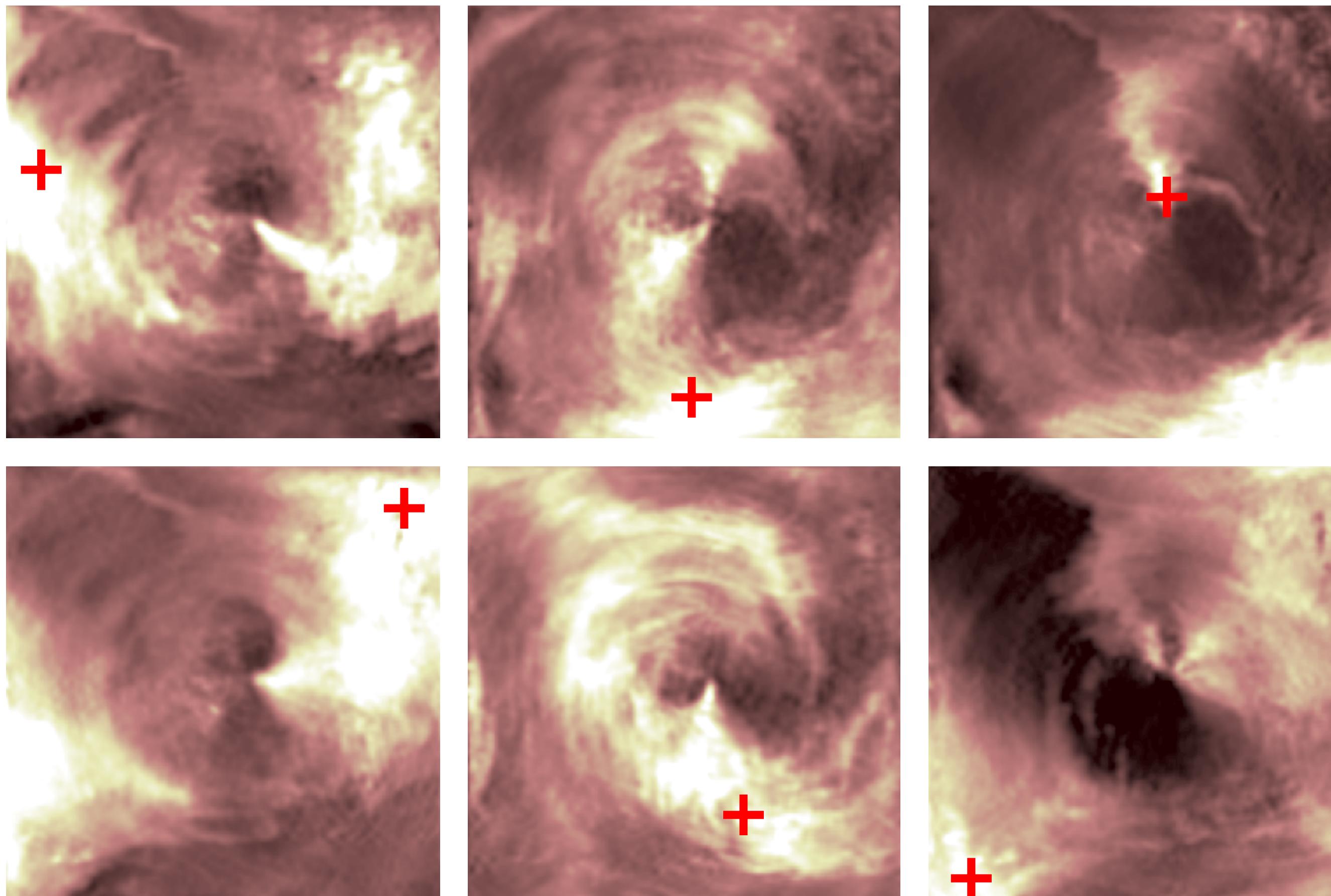


Normalized Cross-Correlation (NCC)



Normalized Cross-Correlation (NCC)

- Responsive interaction (~1s)



Conclusions

Conclusions

- Compression/decompression for **integral histograms**

Conclusions

- Compression/decompression for **integral histograms**
- Using **tensors** (first of such kind)

Conclusions

- Compression/decompression for **integral histograms**
- Using **tensors** (first of such kind)
- **Lossy** compression → **huge reduction ratios**

Conclusions

- Compression/decompression for **integral histograms**
- Using **tensors** (first of such kind)
- **Lossy** compression → **huge reduction ratios**
- Ideal for **medium to large** ROIs

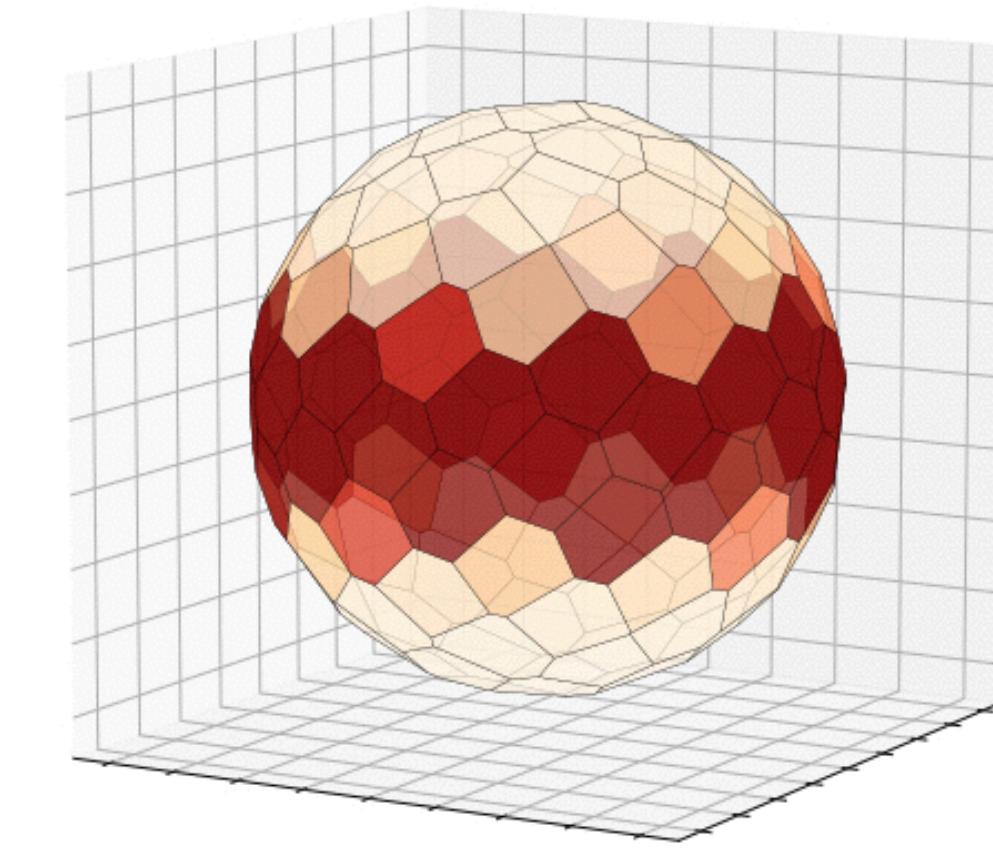
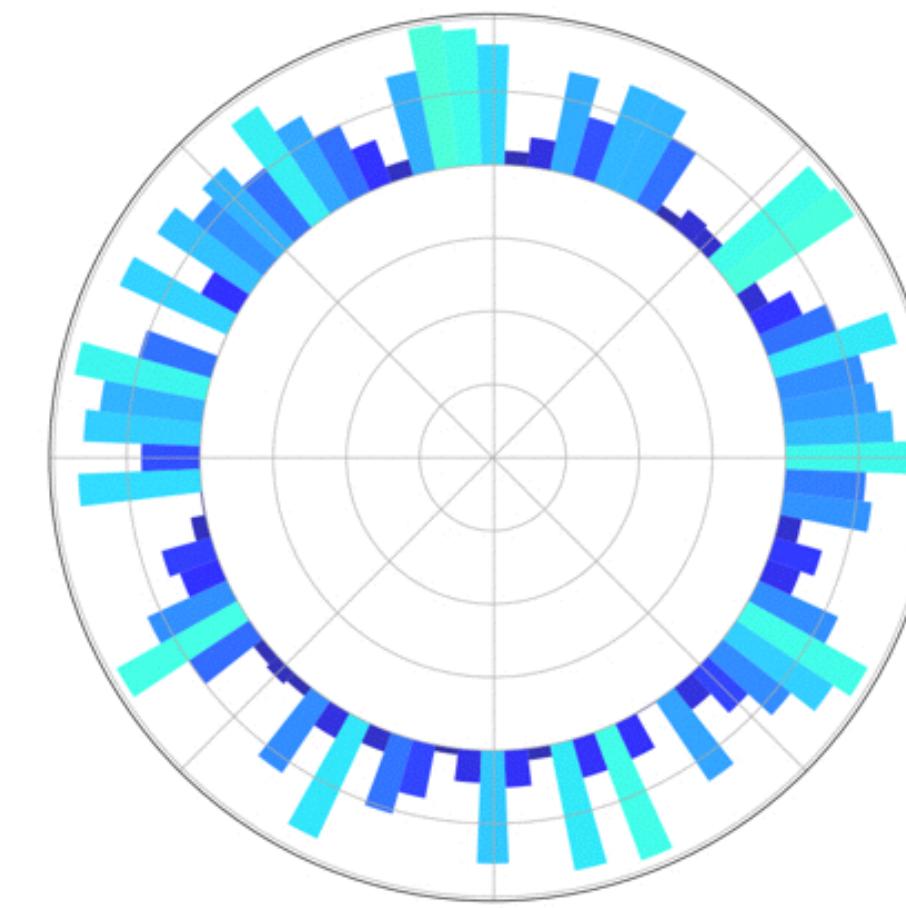
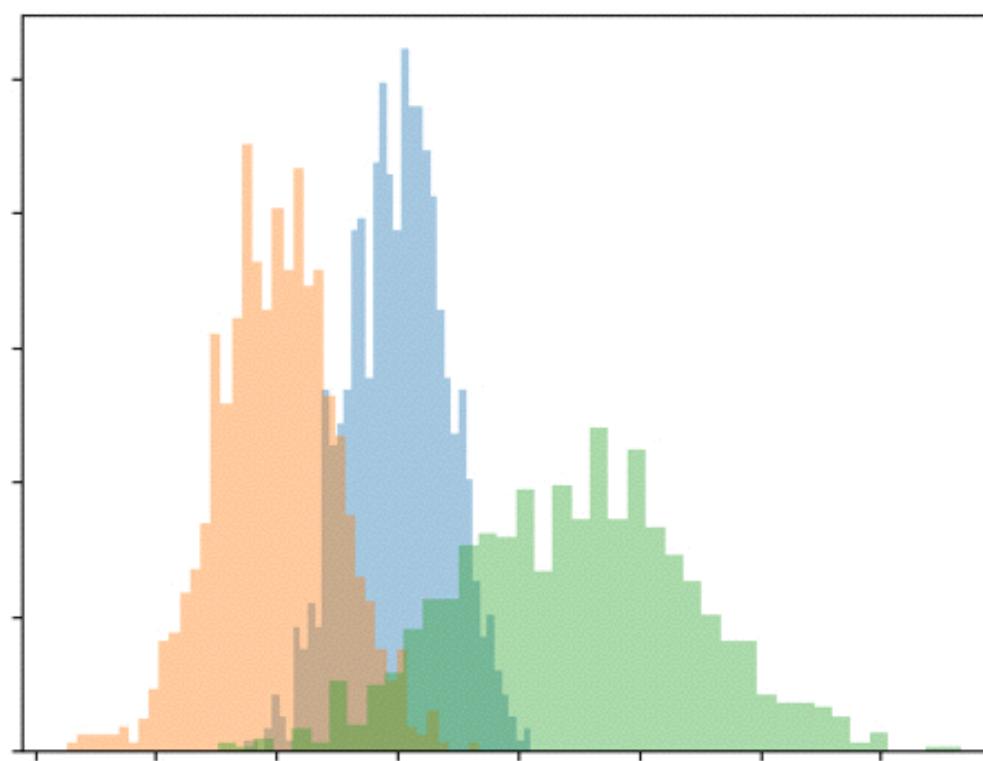
Conclusions

- Compression/decompression for **integral histograms**
- Using **tensors** (first of such kind)
- **Lossy** compression → **huge reduction ratios**
- Ideal for **medium to large** ROIs
- Can handle **non-rectangular** ROIs

Conclusions

- Compression/decompression for **integral histograms**
- Using **tensors** (first of such kind)
- **Lossy** compression → **huge reduction ratios**
- Ideal for **medium to large** ROIs
- Can handle **non-rectangular** ROIs
- Efficient **histogram field** reconstruction

Thank You!



Paper: “Tensor Decompositions for Integral Histogram Compression and Look-up”

Code: github.com/rballester/tthistograms

Slides: github.com/rballester/slides

Why Compress SATs, not the Original Table

