High-order Tensor Completion for Data Recovery via Sparse Tensor-train Optimization

Longhao Yuan[1, 2], Qibin Zhao[2, 3, *], and Jianting Cao[4, 1, *]

[1] Graduate School of Engineering, Saitama Institute of Technology, Japan
[2] Tensor Learning Unit, RIKEN Center for Advanced Intelligence Project (AIP), Japan
[3] School of Automation, Guangdong University of Technology, China
[4] School of Computer Science and Technology, Hangzhou Dianzi University, China

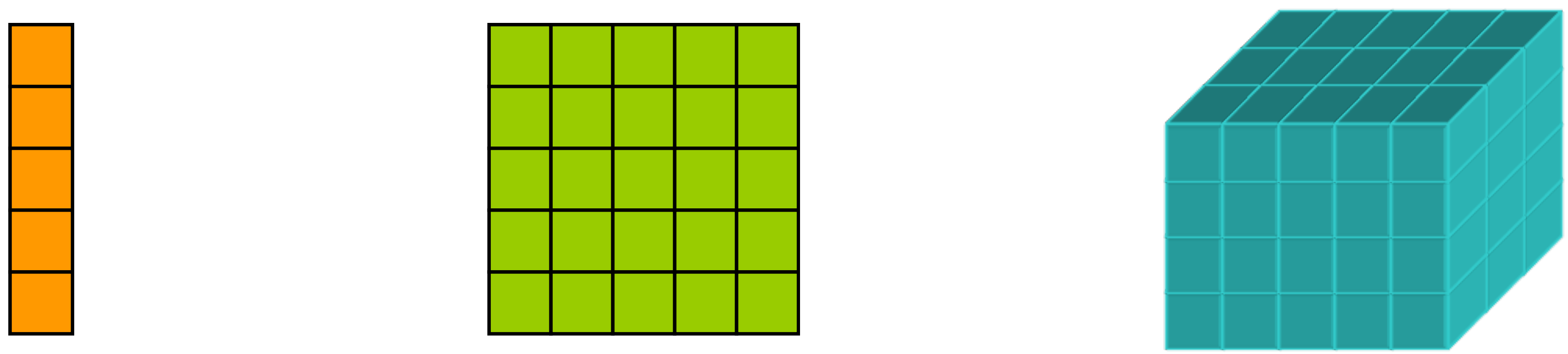
Outline

- 1. Background
- 2. Sparse tensor-train optimization
- 3. Experiments and results
- 4. Conclusion

Background

What is tensor?

Generalization of an n-dimensional array.



vector: 1st-order tensor   matrix: 2nd-order tensor   3rd-order tensor

What can tensor do?

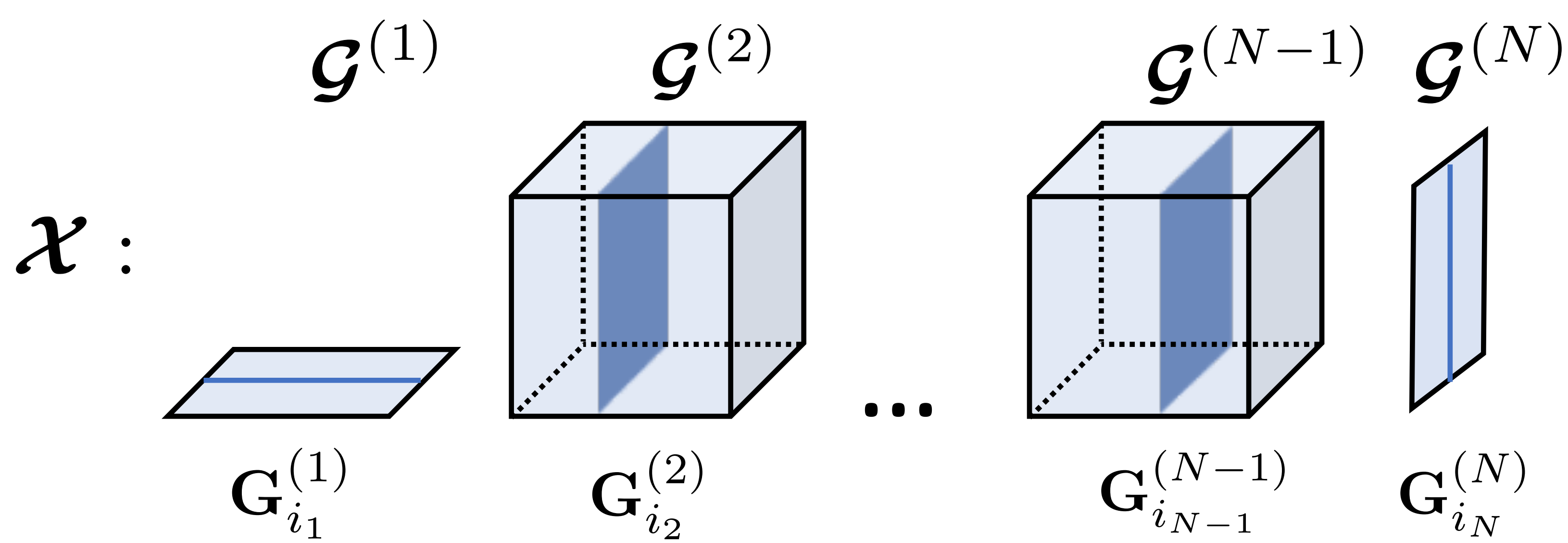
A matrix can represent a 2-order relation.

A tensor can represent a high-order relation.

Background

Tensor-train Decomposition (TTD)

Decompose a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ to TT format:



Core tensor: $\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$,

Silce: $\mathbf{G}^{(n)} \in \mathbb{R}^{r_{n-1} \times r_n}$,

TT-rank: $\{r_0, r_1, \cdots, r_N\}, r_0 = r_N = 1,$

$n = 1, 2, \cdots, N.$

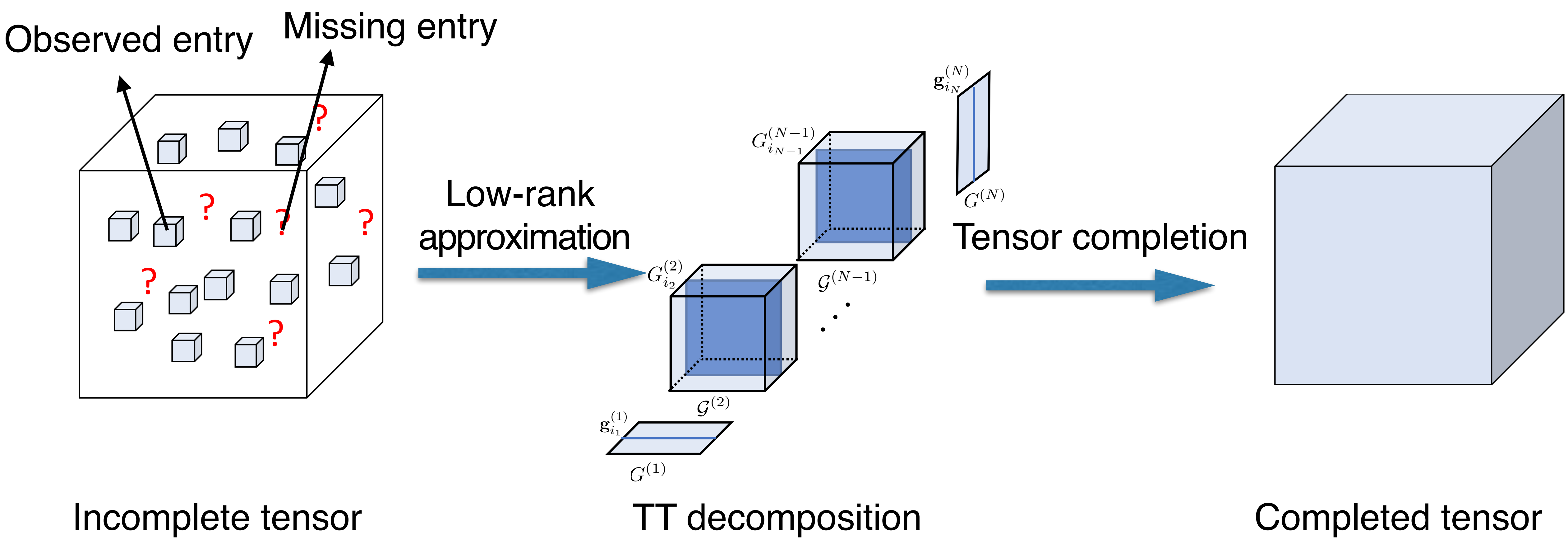
For each element:

$$x_{i_1 \cdots i_N} = \prod_{n=1}^{N} \mathbf{G}^{(n)}_{i_n}$$

[Oseledets 2011]

Background

Tensor completion by TT model

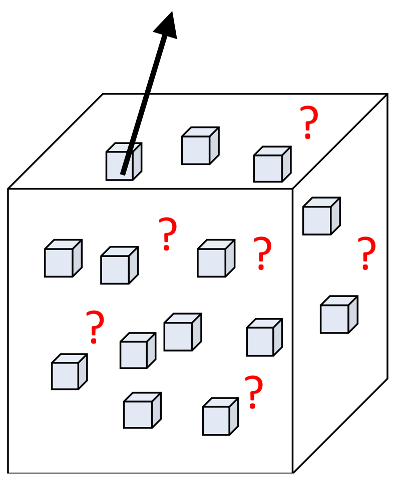


Find the low-rank TT decomposition by observed entries.

STTO algorithm

For **one** observed entry:

$$y_m = \mathcal{Y}(i_1^m, i_2^m, \cdots, i_N^m)$$



The approximation of TTD: $x_m = \prod_{n=1}^{N} \mathbf{G}_{i_n^m}^{(n)}$

Loss function: $$f(\mathbf{G}_{i_1^m}^{(1)}, \mathbf{G}_{i_2^m}^{(2)}, \cdots, \mathbf{G}_{i_N^m}^{(N)}) = \frac{1}{2}\left\| y_m - \prod_{n=1}^{N} \mathbf{G}_{i_n^m}^{(n)} \right\|_F^2$$

The gradient for according slice of core tensor:

$$\frac{\partial f}{\partial \mathbf{G}_{i_n^m}^{(n)}} = (x_m - y_m)(\mathbf{G}_{i_n^m}^{>n} \mathbf{G}_{i_n^m}^{<n})^T,$$

Where $$\mathbf{G}_{i_n^m}^{>n} = \prod_{n=n+1}^{N} \mathbf{G}_{i_n^m}^{(n)}, \ \mathbf{G}_{i_n^m}^{<n} = \prod_{n=1}^{n-1} \mathbf{G}_{i_n^m}^{(n)}.$$

STTO algorithm

For **all** the observed entries:

Loss function: $$f(\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \cdots, \mathcal{G}^{(N)}) = \frac{1}{2}\sum_{m=1}^{M} \|y_m - x_m\|_F^2$$

Gradient accumulation: $$\frac{\partial f}{\partial \mathbf{G}_j^{(n)}} = \sum_{\substack{m=1 \\ m: i_n^m = j}}^{M} (x_m - y_m)(\mathbf{G}_{i_n^m}^{>n} \mathbf{G}_{i_n^m}^{<n})^T$$

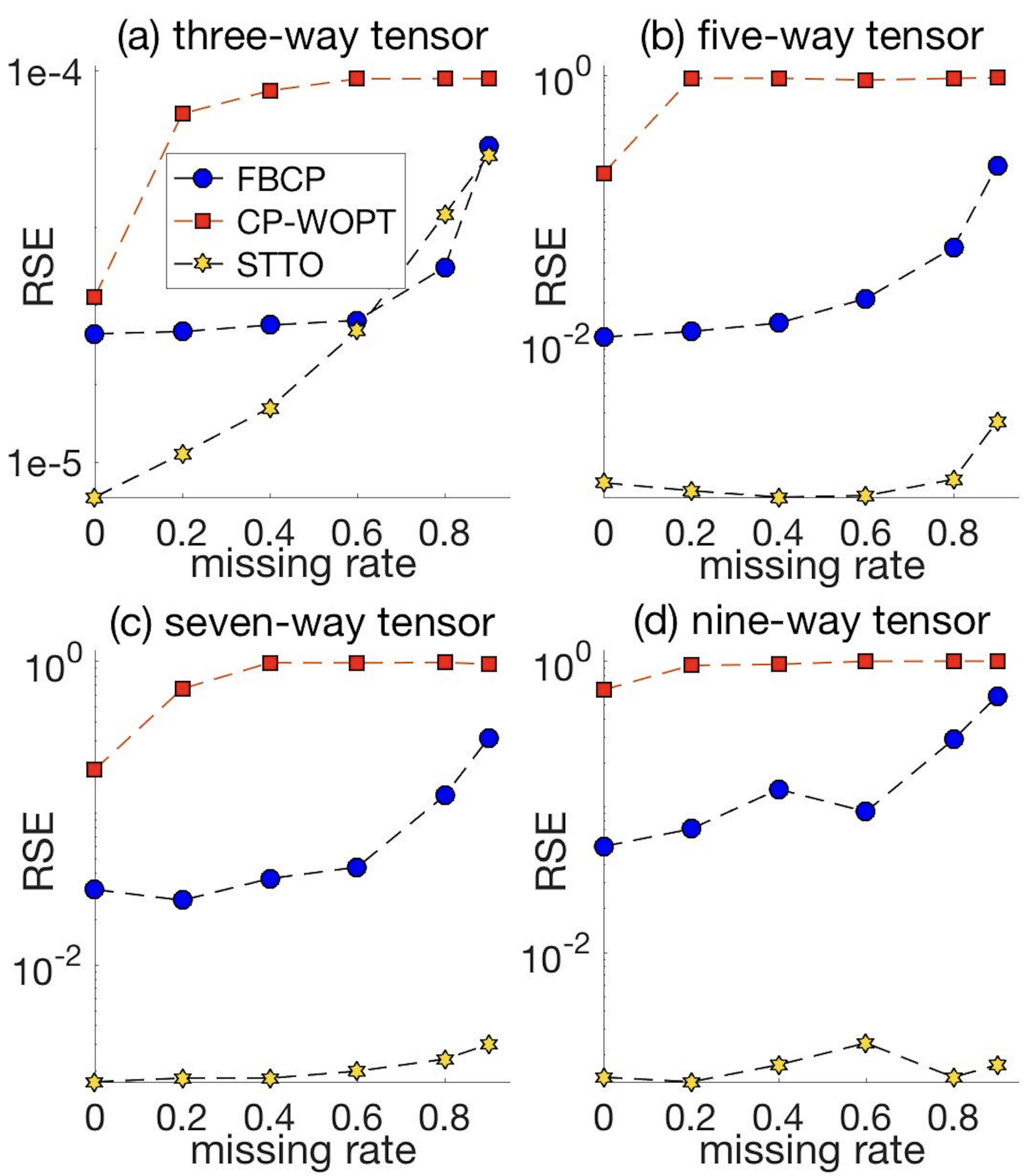
Computational complexity: $\mathcal{O}(MN^2R^3)$

Overcome the curse of dimensionality

Algorithm implementation:

1. Initialize core tensors.
2. Do gradient descent until stopping condition is satisfied.
3. Use optimized tensor cores to approximate missing entries.

Simulations



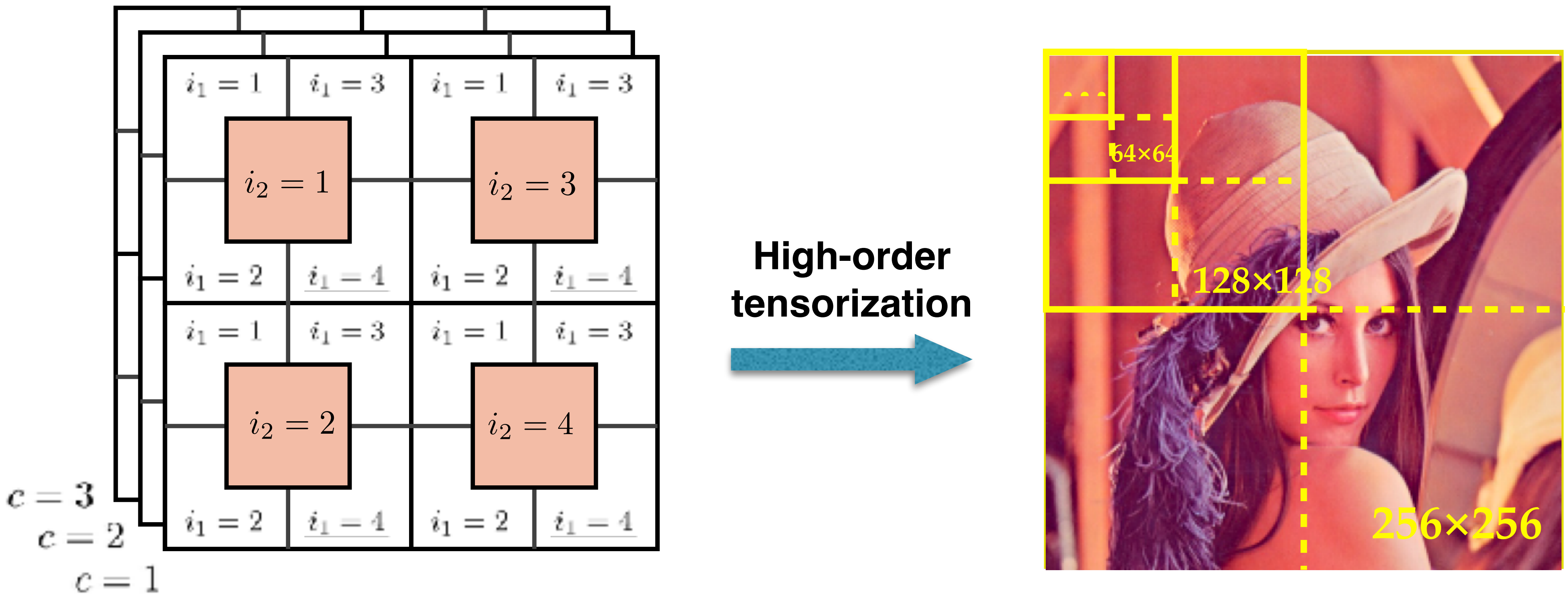
Simulation results

Synthetic data:
1. produced from a highly oscillating function.
2. Experiments by 3D, 5D, 7D, 9D tensors.

Conclusions:
1. STTO performs well in 3D cases.
2. STTO outperforms others in high-order cases.

Image data experiment

High-order tensorization for visual data



Tensorization for a 256×256×3 image

From 3-way to 9-way

1. Reshape 256×256×3 to 2×2×…×2×3 (17-way tensor).
2. Permute by {1 9 2 10 3 11 4 12 5 13 6 14 7 15 8 16 17}.
3. Reshape to 4×4×4×4×4×4×4×4×3 (9-way tensor).

Better data structure

The first order represent a 2×2 pixel block.
The second order represent four 2×2 pixel block.

…

This can catch more structure relation of data.

Improve performance of STTO.
Able to deal with irregular missing.

Image data experiment

Image completion overview:

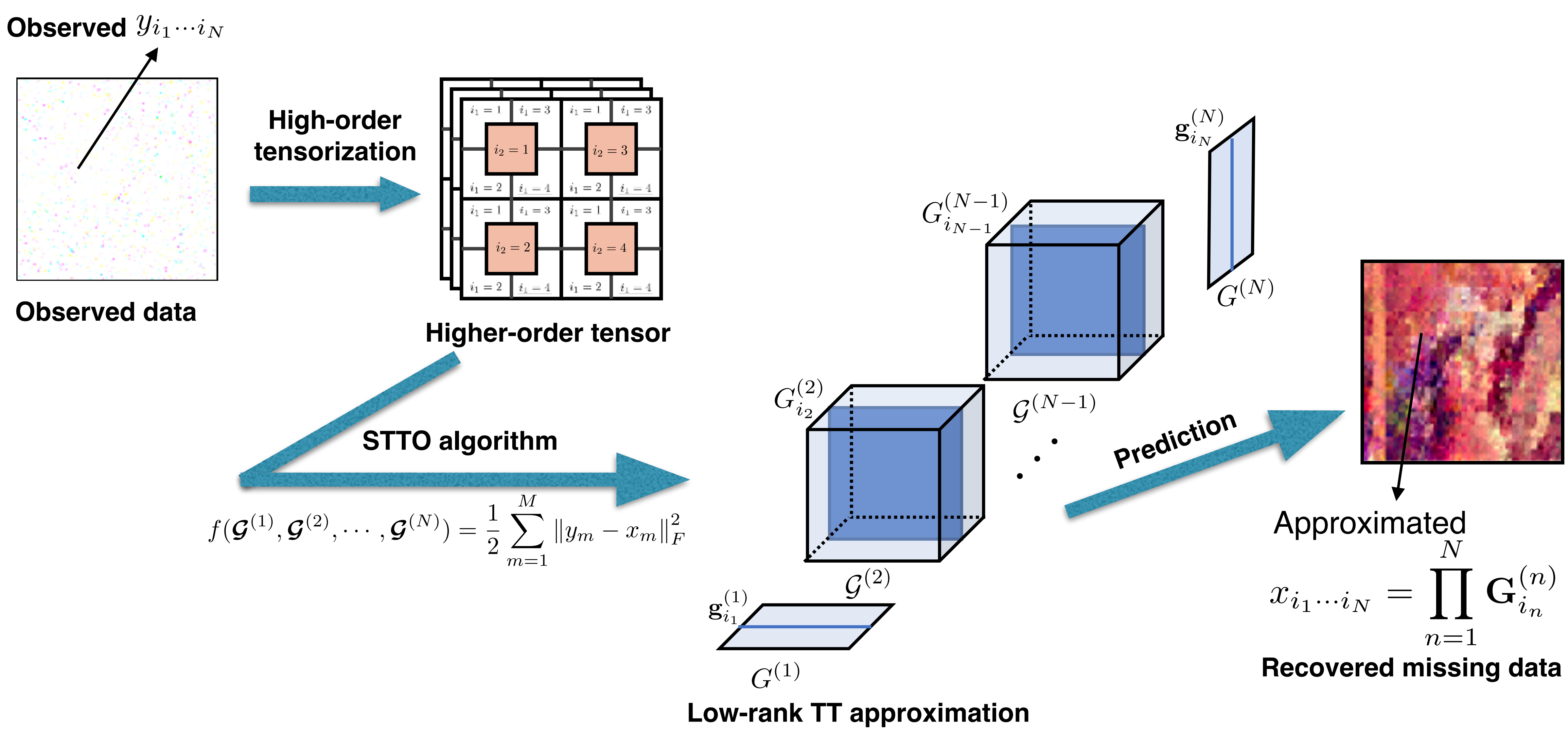
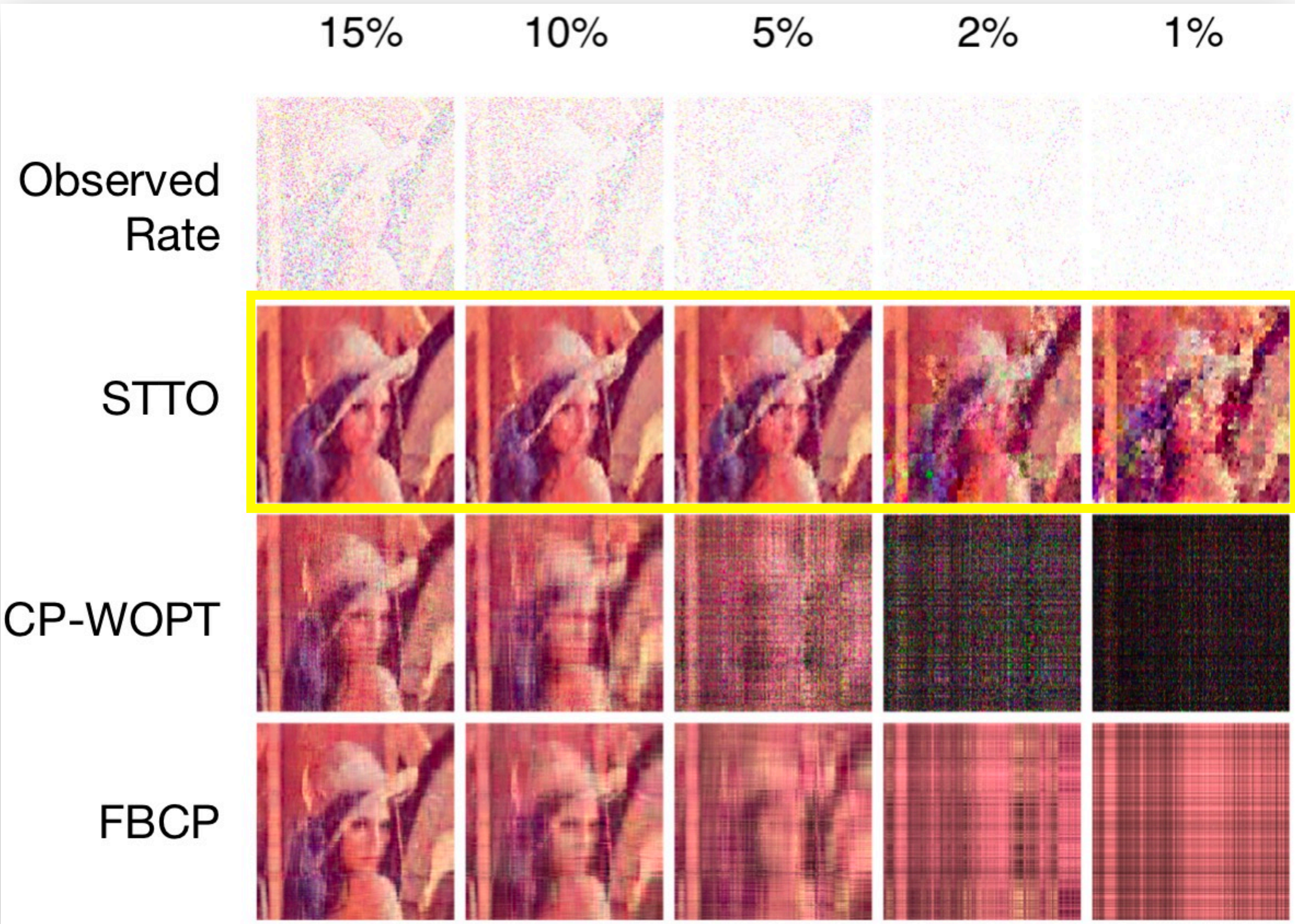


Image data experiment

Random missing results



Ground truth

		85%	90%	95%	98%	99%
TT-WOPT	RSE	**0.1233**	**0.1297**	**0.1416**	**0.2202**	**0.2638**
	PSNR	**23.4877**	**22.6076**	**21.5282**	**18.9396**	**17.0029**
CP-WOPT	RSE	0.1891	0.3169	0.5348	1.0918	1.1309
	PSNR	18.8578	18.0389	12.5649	7.8015	6.4971
FBCP	RSE	0.1440	0.1867	0.2432	0.3052	0.3372
	PSNR	22.2853	19.9410	17.5166	15.4784	14.5841

Image data experiment

Special missing cases results

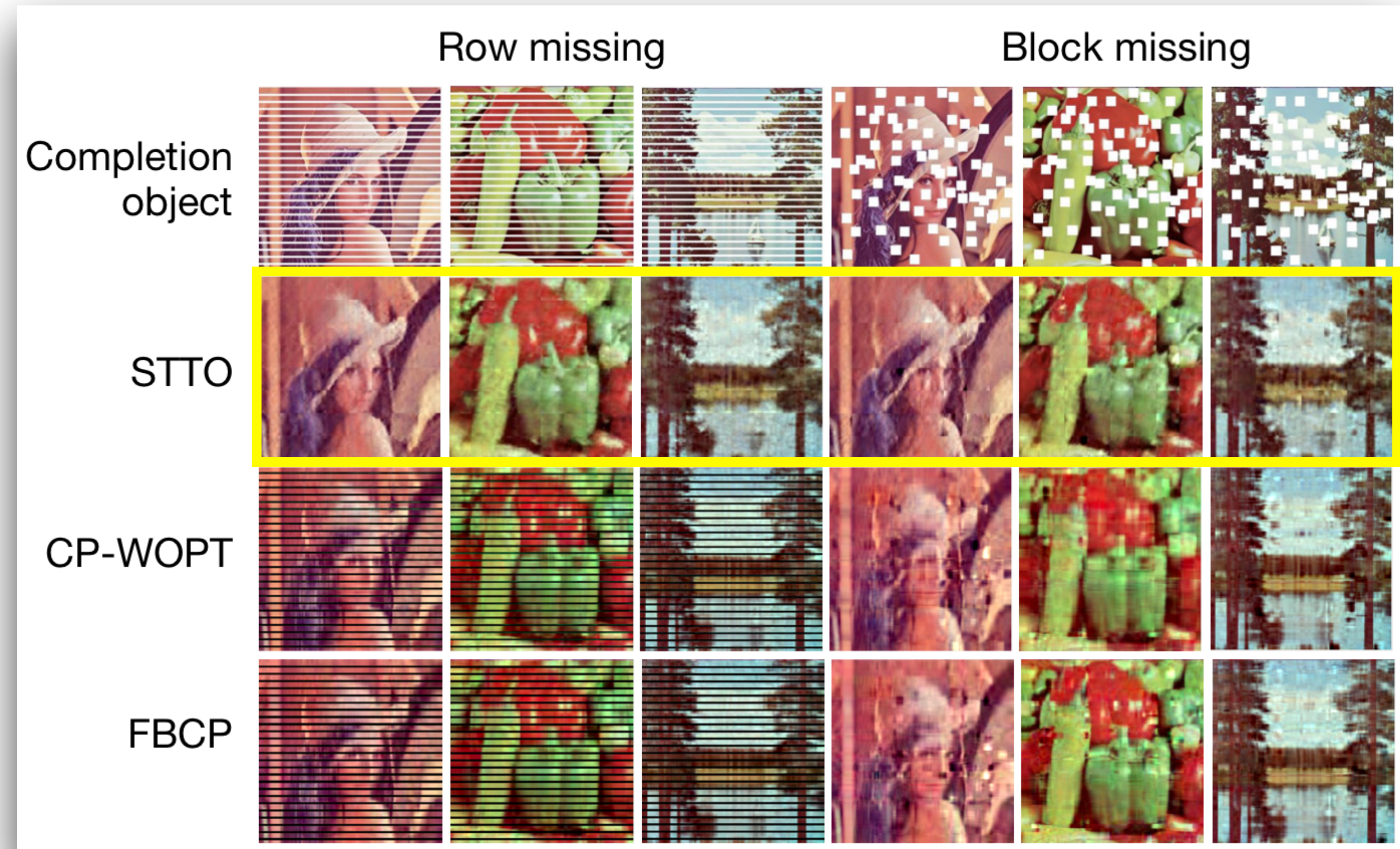


image		row missing			block missing		
		lena	peppers	sailboat	lena	peppers	sailboat
STTO	RSE	**0.1138**	**0.1661**	**0.1767**	**0.1323**	**0.1611**	**0.1704**
	PSNR	**24.00**	**20.80**	**19.93**	**22.69**	**21.06**	**20.25**
CP-WOPT	RSE	0.5401	0.5546	0.5545	0.1746	0.2252	0.2082
	PSNR	10.86	10.85	10.34	20.61	18.27	19.00
FBCP	RSE	0.5503	0.5594	0.5586	0.1498	0.1671	0.1764
	PSNR	10.46	10.58	10.18	21.66	20.79	20.01

Conclusions

Contributions:

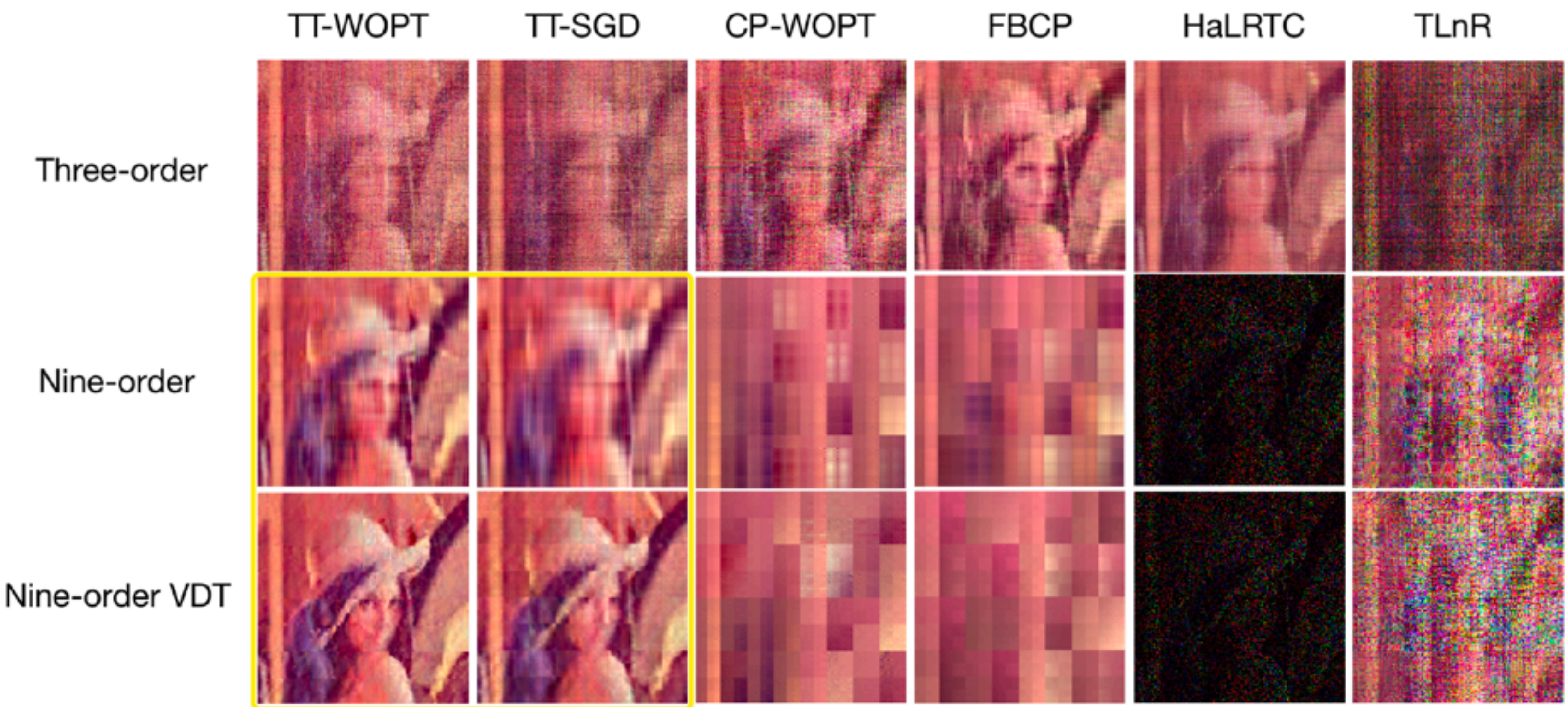
1. Propose STTO algorithm with low computational complexity.
2. Provide tensorization method to transform low-order tensor visual data to high-order.
3. Obtain superior results in simulation and image data completion.

Future works:

1. Develop more scalable completion algorithm based on TTD.
2. Automatically determine TT-rank.

Thank you for your attention!

Comparison of applying tensorization



	Loss function Gradient	Computational complexity
TT-WOPT	$f(\boldsymbol{\mathcal{G}}^{(1)}, \boldsymbol{\mathcal{G}}^{(2)}, \cdots, \boldsymbol{\mathcal{G}}^{(N)}) = \frac{1}{2} \lVert \boldsymbol{\mathcal{Y}}_w - \boldsymbol{\mathcal{X}}_w \rVert_F^2$ $\frac{\partial f}{\partial \mathbf{G}_{(2)}^{(n)}} = (\mathbf{X}_{w(n)} - \mathbf{Y}_{w(n)})(\mathbf{G}_{(1)}^{>n} \otimes \mathbf{G}_{(n)}^{<n})^{\mathrm{T}}$	$\mathcal{O}(NI^N + NI^{N-1}R^2)$
STTO	$f(\boldsymbol{\mathcal{G}}^{(1)}, \boldsymbol{\mathcal{G}}^{(2)}, \cdots, \boldsymbol{\mathcal{G}}^{(N)}) = \frac{1}{2} \sum_{m=1}^{M} \lVert y_m - x_m \rVert_F^2$ $\frac{\partial f}{\partial \mathbf{G}_j^{(n)}} = \sum_{\substack{m=1 \\ m: i_n^m = j}}^{M} (x_m - y_m)(\mathbf{G}_{i_n^m}^{>n} \mathbf{G}_{i_n^m}^{<n})^T$	$\mathcal{O}(MN^2R^3)$
TT-SGD	$f(\mathbf{G}_{i_1^m}^{(1)}, \mathbf{G}_{i_2^m}^{(2)}, \cdots, \mathbf{G}_{i_N^m}^{(N)}) = \frac{1}{2} \left\lVert y_m - \prod_{n=1}^{N} \mathbf{G}_{i_n^m}^{(n)} \right\rVert_F^2$ $\frac{\partial f}{\partial \mathbf{G}_{i_n^m}^{(n)}} = (x_m - y_m)(\mathbf{G}_{i_n^m}^{>n} \mathbf{G}_{i_n^m}^{<n})^T,$	$\mathcal{O}(N^2R^3)$