**FCTN Decomposition and Its Application to Higher Order Tensor Completion**

Arun Prakash. J (CB.EN.U4AIE19014)

Asswin CR(CB.EN.U4AIE19016)

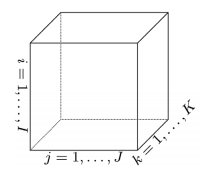
**Abstract**:

Tensor decomposition has always played a crucial role in finding the underlying relations in the tensor. Several algorithms were developed to serve the purpose. CP decomposition and Tucker decomposition have achieved promising results. However, establishment of connection between factors seemed to be missing. Tensor Train and Tensor Ring was developed to solve the issue to an extent. Tensor Train and Tensor Ring has its own limitations of high sensitivity to permutation and establishment of connect only between two adjacent factors. Fully connected Tensor Node Decomposition was thus developed as a generalized tensor decomposition. The superiority of this method lies in the correlation between any two factors. We employ FCTN decomposition to one specific task, the Tensor Completion.

**Keywords**: CP Decomposition, Tucker Decomposition, Tensor Train, Tensor Ring.

**Tensor**

Tensor is an algebraic object that represents the relation between objects. It is a multidimensional array. Rank /order of a tensor is the number of dimensions. The order-N tensor is an element of the tensor product of N vector spaces. An order-0 tensor denotes a scalar. An order-1 tensor denotes a vector. Matrix is an order-2 tensor. An order-3 tensor is a tensor. Order 4 and all ranks above 3 are called high dimensional tensors. Each dimension is called a mode.



**Fig 1.1 An order-3 tensor**

**Notations**

Scalar is represented as x. Vector is represented as **x**. Matrix is represented as **X**. Tensor is represented as . An order N tensor is represented as and denotes the  element of the tensor. The Frobenius norm of the tensor is denoted as . FCTN composition is denoted by 

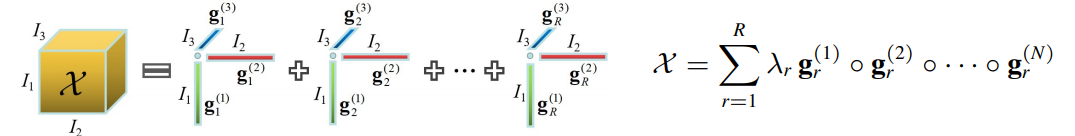
**Tensor Decomposition**

The main goal is to decompose a higher order tensor to a set of low dimensional factors. Tensor decomposition has the capability to capture the global correlations of tensors. This correlation is very much useful for Tensor completion.

**CP Decomposition**

The CP decomposition decomposes the tensor into outer product of order-1 tensors. An order-N tensor is broken down into outer product of N order-1 tensor. The CP decomposition uses to alternating least squares algorithm to decompose the tensor.





**Fig 4.1 CP decomposition**

**Matlab Code:**

clc; clear all; close all;

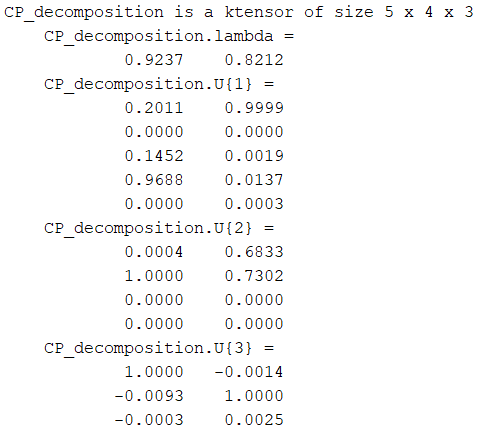
X=sptenrand([5 4 3],10); % creates a Sparse uniformly distributed random tensor

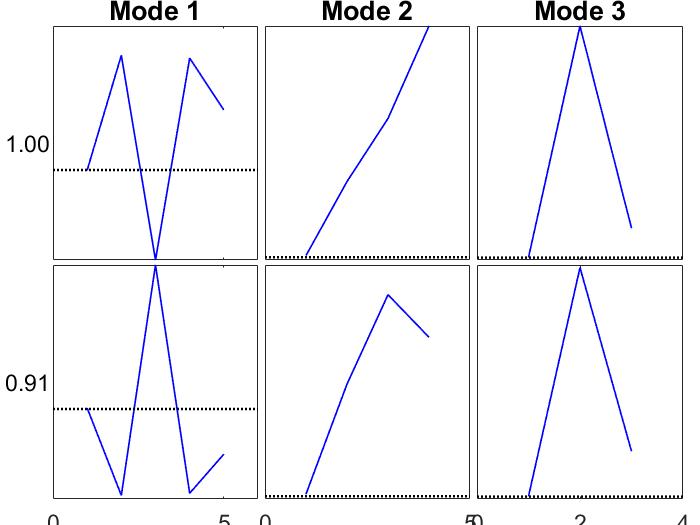
CP\_decomposition=cp\_als(X,2); % computes an estimate of the best rank-2

CP\_decomposition.lambda % returns the lambda values

CP\_decomposition.U % reutrns the order-1 tensors

viz(CP\_decomposition) % visualizes the components of the tensor





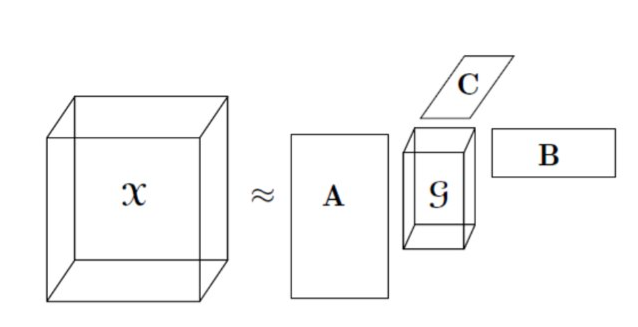
**Limitations of CP decomposition**:

1. There is no connection between the low dimensional factors.
2. Difficulty in finding optimal solution.

**Tucker Decomposition**

The tucker decomposition decomposes a tensor into a core tensor along with its factor matrices. It is the extension of Singular Value Decomposition to a higher dimension. The tensor is approximated by inner product of the factor matrices.





**Fig 5.1 Tucker decomposition**

**Matlab code:**

clc; clear all; close all;

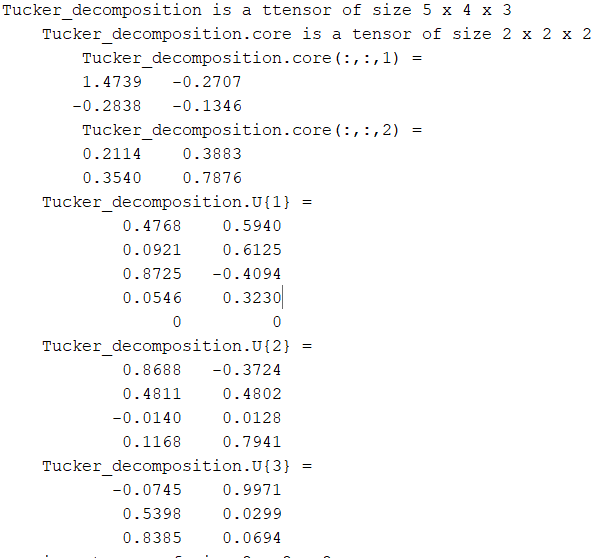
X=sptenrand([5 4 3],10); % creates a Sparse uniformly distributed random tensor

Tucker\_decomposition=tucker\_als(X,2) % computes an estimate of the best rank-2

Tucker\_decomposition.lambda; % returns the core tensor

Tucker\_decomposition.U; % returns the factor matrices

**Output:**



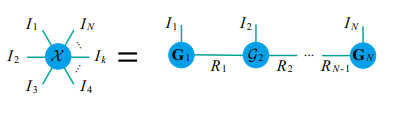
**Limitations of Tucker decomposition:**

1. Only characterizes correlation among one mode and the rest of all other modes rather than any two modes.
2. Needs high storage cost.

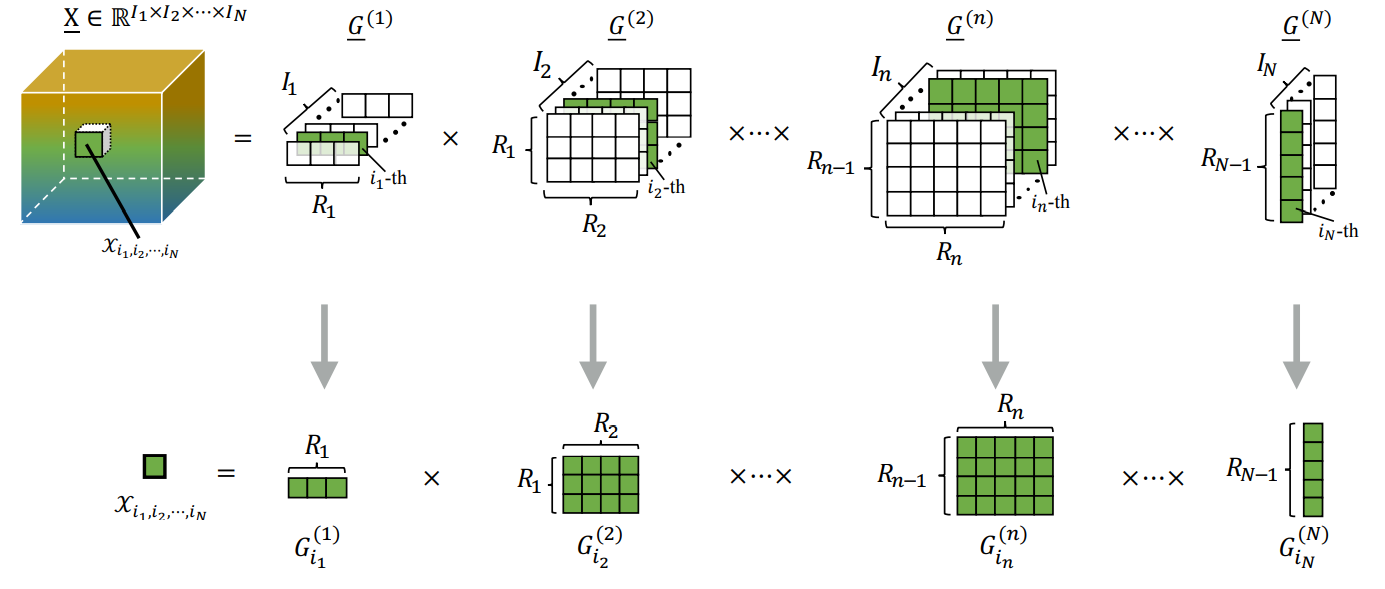
**Tensor Train Decomposition**

Tensor train decomposition, decomposes a tensor to a set of core tensors connected in a sequential manner. It is also called the matrix product state. The TT decomposition, decomposes an Nth order tensor to N-2 third order tensors at the intermediate along with two order-1 tensor at both the tail ends. The principle is to approximate each tensor element by the sequential product of tensors. The vectors in the tail ends ensure that the output is a scalar.





**Fig 6.1 TT decomposition**



**Fig 6.2 TT decomposition explained**

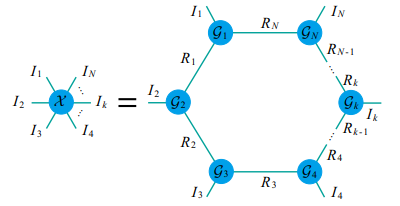
**Limitations of TT:**

1. There is a constraint on the TT-ranks, i.e.  and  =1.
2. TT-ranks always have a fixed pattern, i.e., smaller for the border cores and larger for the middle cores, which might not be the optimum for specific data tensor.
3. Relation is established only between two adjacent factors, which limits the correlation of tensors.
4. ​TT keeps invariance only when the factors make a reverse permutation.

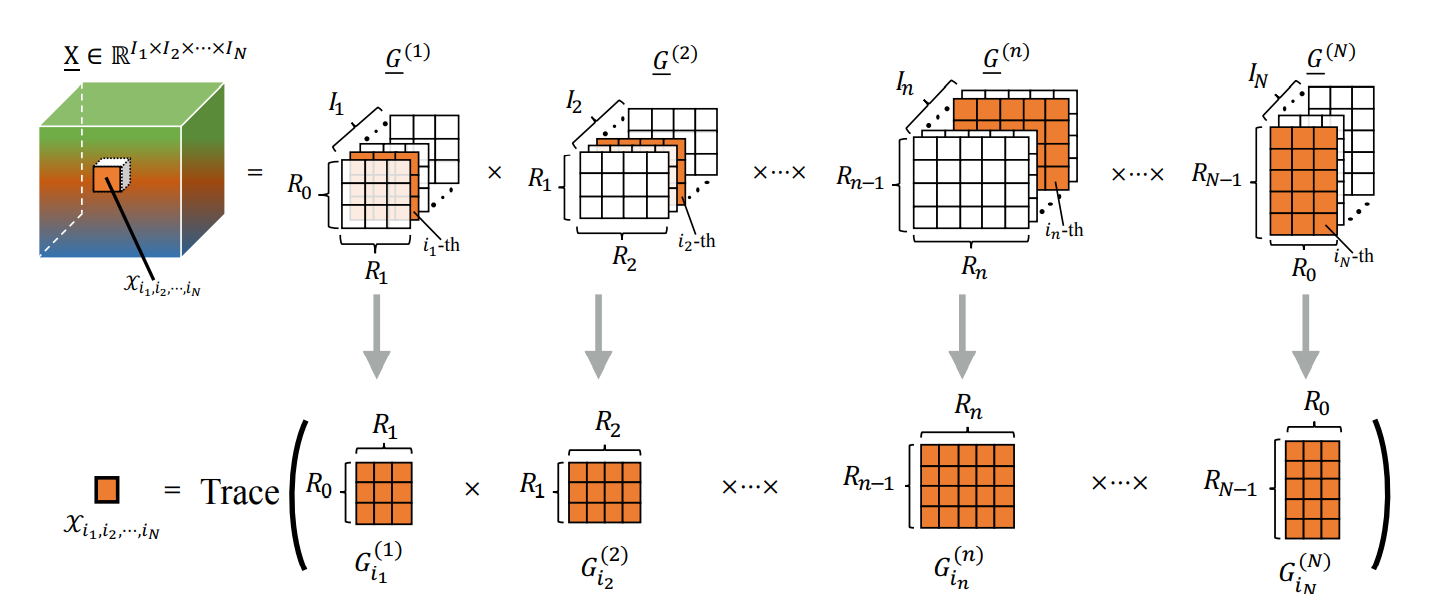
**Tensor Ring Decomposition**

The TR decomposition was built to counter the disadvantages of TT decomposition. The first disadvantage being, constraint of ranks, the TR decomposition, decomposes an order-N tensor into N tensors of order-3. It replaces the first and last order-1 tensor in TT by order-3 tensors. The TR approximated a tensor by taking the trace of the tensor product of the core tensors in the circular manner.





**Fig 7.1 TR decomposition**

**Fig 7.2 TR decomposition explained**

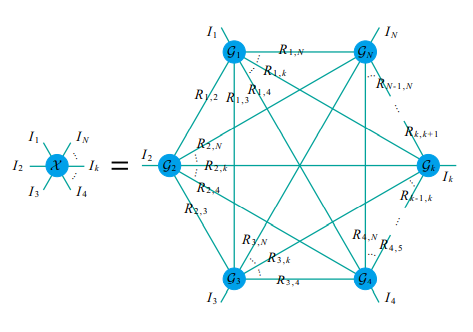
**Limitations of TR:**

1. Correlation only among adjacent cores can be observed.
2. TR preserves invariance only when permuted in a circular order.

**Fully Connected Tensor Network Decomposition**

The TT and TR decomposition, are highly sensitive to permutation of tensor modes making decomposition inflexible. The FCTN decomposition, decomposes an order N tensor into a set of order N tensors establishing a connection between any to factors. Since all the nodes are interconnected it is called a fully connected network. All the factors are of equal size and the correlation is given by (correlation between the 1st mode and kth mode).





**Fig 8.1 FCTN decomposition**

**FCTN Composition**

Composition is the process of generating the tensor  from its factors .

**Tensor Completion**

Tensor Completion is the high dimensional version of matrix completion. The core problem of the missing value estimation lies on how to build relationship between the known and the unknown elements. The rank is what captures the global relations between the knowns and the unknowns. Tensor completion’s applications include image reconstruction, filling in recommendation systems, etc.

Given an incomplete tensor,  of the target tensor , the reconstruction is formulated as:





**Algorithm:**



**Matlab Code:**

**Tucker and HOSVD Decomposition of Micro seismic Data:**

%% HOSVD

close all;

clear all;

clc;

%% Load Data

%Load data tensor

str=load('tensor\_test.mat','tensor');

str\_xyz=load('xyztensor.mat','tensor\_xyz');

%Convert to tensor

clear tensor

a=str.tensor; %Microseismic Data

b=str\_xyz.tensor\_xyz; %XYZ Microseismic Data

X=tensor(a);

X\_doub=double(X);

xs=size(X\_doub);

eps = 0.02; %Tolerance

%% HOSVD estimation

T = hosvd(X,2\*sqrt(eps));

T\_doub=double(T);

fprintf('Time for 10 runs of ST-HOSVD\_T:\n');

tic, for i =1:10, T = hosvd(X,2\*sqrt(eps),'verbosity',0); end; toc

%Factor matrices (U,V,W) and core tensor

core=T.core;

G=double(core);

U=T.U{1};

V=T.U{2};

W=T.U{3};

%% Compression ratio

s\_core=size(core);

s\_X=size(X);

wh\_X=whos('bytes','X');

siz\_X=wh\_X.bytes\*10e-06;

wh\_T=whos('bytes','T');

siz\_T=wh\_T.bytes\*10e-06;

CR=(1-(wh\_T.bytes/wh\_X.bytes))\*100;

%% Singular values (Frobenius Norm)

sv\_1 = zeros(size(G,1),1);

for i = 1:size(G,1)

sv\_1(i)=sqrt(sum(sum((G(i,:,:).^2))));

end

sv\_1=sv\_1';

sv\_1=(sv\_1./max(sv\_1));

sv\_2 = zeros(size(G,2),1);

for i = 1:size(G,2)

sv\_2(i)=sqrt(sum(sum((G(:,i,:).^2))));

end

sv\_2=sv\_2';

sv\_2=(sv\_2./max(sv\_2));

sv\_3 = zeros(size(G,3),1);

for i = 1:size(G,3)

sv\_3(i)=sqrt(sum(sum((G(:,:,i).^2))));

end

sv\_3=sv\_3';

sv\_3=(sv\_3./max(sv\_3));

sv = {sv\_1 sv\_2 sv\_3};

%% Plots

%% Normalized singular values

figure(1)

for n = 1:3

subplot(1,3,n)

plot(sv{n});

title(sprintf('Mode %d',n));

end

set(gcf,'color','w');

sgtitle('Normalized singular values')

%% Comparison between Recovered and Original

k=20;

figure(2)

subplot(1,2,1)

imagesc(T\_doub(:,:,k))

title('Recovered');

subplot(1,2,2)

imagesc(X\_doub(:,:,k))

title('Original');

set(gcf,'color','w')

sgtitle(sprintf('Reconstruction (X vs Recovered(X))\n Tol: %.4g - CR: %.4g',eps,CR));

%% Number of variables

figure(3)

subplot(1,2,1)

V=100-CR;

pi=[CR V];

pie(pi);

title('Variables');

subplot(1,2,2)

r=(s\_X(1)\*s\_X(2)\*s\_X(3))\*(1-(CR/100));

p=(s\_X(1)\*s\_X(2)\*s\_X(3));

bar(1, p, 'FaceColor',[0 0.4470 0.7410])

hold on

bar(2, r, 'FaceColor',[0.9290 0.6940 0.1250])

legend('Tensor','HOSVD')

title('Number of variables');

set(gca, 'xticklabels', []);

set(gcf,'color','w');

sgtitle('Number of variables')

**Output:**

Computing HOSVD...

Size of core: 181 x 34 x 11

||X-T||/||X|| = 0.0234079 <=0.028284 (tol)

Time for 10 runs of ST-HOSVD\_T:

Elapsed time is 5.345875 seconds.

**Inference from the output:**

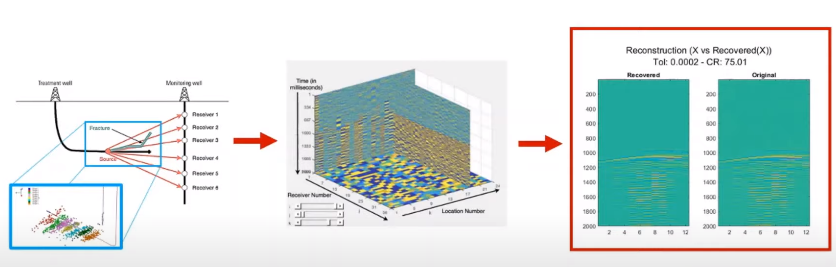
Size of the core that best apply to the epsilon to the tolerance factor.( it decreases as we increase the value of tolerance).

Relative Error is associated with it.

**GitHub link :**

[**https://github.com/AsswinCR/MIS-4\_FCTN.git**](https://github.com/AsswinCR/MIS-4_FCTN.git)

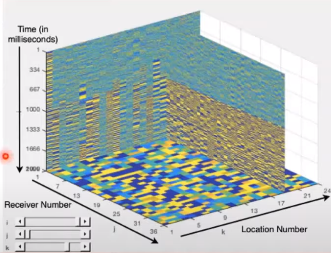
**Working principle of the code:**

****

1)Data used is micro seismic data which is received by receivers.

2)Next it is transmitted in to Tensor Form(3D tensor).

3)Then apply HOSVD to reduce the size of the data( relation between original(actual data) and the recorded(reduced data))



After converting it into 3rd order tensor :

3 modes:

1)Seismic Event time(z-axis)

2)Receiver Location(x-axis)

3)Event Locations(y-axis).

**Plots:**

**Optimal Solution**

Tolerance value : 0.0002

Compression Ration :75.04

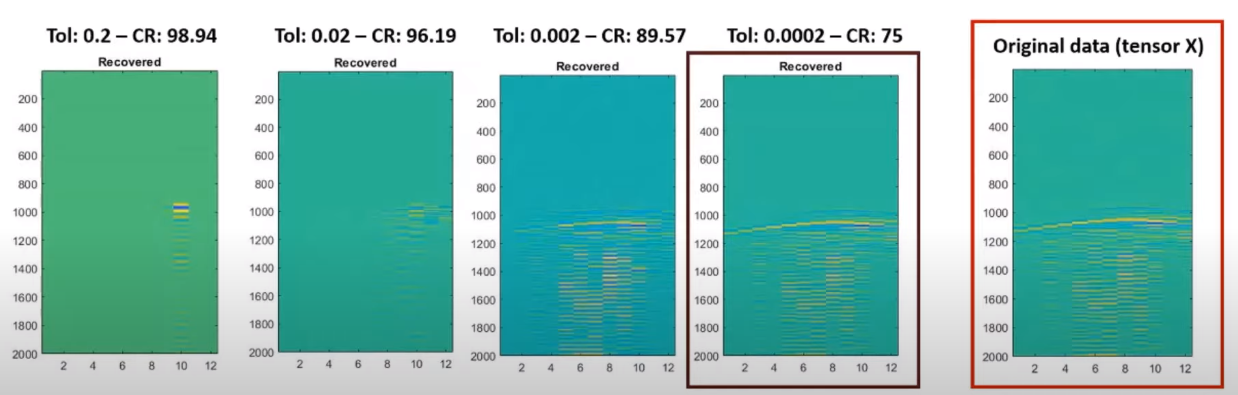
From the diagram it is evident that the reconstructed/recovered X and Original X are almost equal.

**Comparison between original and recovered data**

Higher the value of tolerance, higher is the value of Compression Ratio.

So, we might miss important information about the data, if we use high tolerance value, since the data will be compressed even though we get highly reduced data form.

So it is recommended to choose an optimal Tolerance value to attain a proper Reconstruction.





Variability of each modes is determined by the frobenius norm



Compression Ration=

Bar Graph shows the reduction of variables

**FCTN CODE:**

clc;

clear;

close all;

addpath(genpath('lib'));

addpath(genpath('data'));

%%

methodname = {'Observed', 'FCTN-TC'};

Mnum = length(methodname);

Re\_tensor = cell(Mnum,1);

psnr = zeros(Mnum,1);

ssim = zeros(Mnum,1);

time = zeros(Mnum,1);

%% Load initial data

load('HSV\_test.mat')

if max(X(:))>1

X = X/max(X(:));

end

%% Sampling with random position

sample\_ratio = 0.05;

fprintf('=== The sample ratio is %4.2f ===\n', sample\_ratio);

T = X;

Ndim = ndims(T);

Nway = size(T);

rand('seed',2);

Omega = find(rand(prod(Nway),1)<sample\_ratio);

F = zeros(Nway);

F(Omega) = T(Omega);

%%

i = 1;

Re\_tensor{i} = F;

[psnr(i), ssim(i)] = quality\_ybz(T\*255, Re\_tensor{i}\*255);

enList = 1;

%% Perform algorithms

i = i+1;

% initialization of the parameters

% Please refer to our paper to set the parameters

opts=[];

opts.max\_R = [0, 6, 6, 6;

0, 0, 6, 6;

0, 0, 0, 6;

0, 0, 0, 0];

opts.R = [0, 2, 2, 2;

0, 0, 2, 2;

0, 0, 0, 2;

0, 0, 0, 0];

% R = [0, R\_{1,2}, R\_{1,3}, ..., R\_{1,N};

% 0, 0, R\_{2,3}, ..., R\_{2,N};

% ...

% 0, 0, 0, ..., R\_{N-1,N};

% 0, 0, 0, ..., 0 ];

opts.tol = 1e-5;

opts.maxit = 1000;

opts.rho = 0.1;

%%%%%

fprintf('\n');

disp(['performing ',methodname{i}, ' ... ']);

t0= tic;

%[Re\_tensor{i},G,Out] = inc\_FCTN\_TC(F,Omega,opts);

[Re\_tensor{i},G,Out] = inc\_FCTN\_TC\_end(F,Omega,opts);

time(i) = toc(t0);

[psnr(i), ssim(i)] = quality\_ybz(T\*255, Re\_tensor{i}\*255);

enList = [enList,i];

%% Show result

fprintf('\n');

fprintf('================== Result =====================\n');

fprintf(' %8.8s %5.4s %5.4s \n','method','PSNR', 'SSIM' );

for i = 1:length(enList)

fprintf(' %8.8s %5.3f %5.3f \n',...

methodname{enList(i)},psnr(enList(i)), ssim(enList(i)));

end

fprintf('================== Result =====================\n');

figure,

show\_figResult(Re\_tensor,T,min(T(:)),max(T(:)),methodname,enList,1,prod(Nway(3:end)))

**GitHub link :**

[**https://github.com/AsswinCR/MIS-4\_FCTN.git**](https://github.com/AsswinCR/MIS-4_FCTN.git)

**Output:**

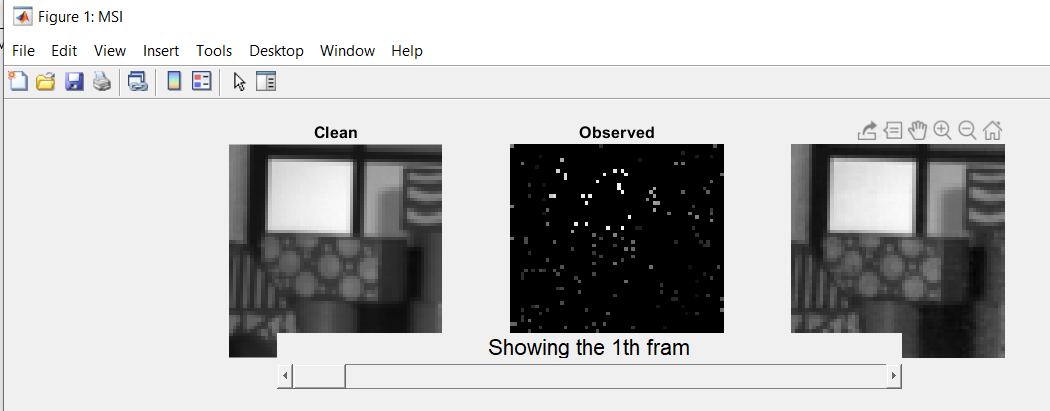
=== The sample ratio is 0.05 ===

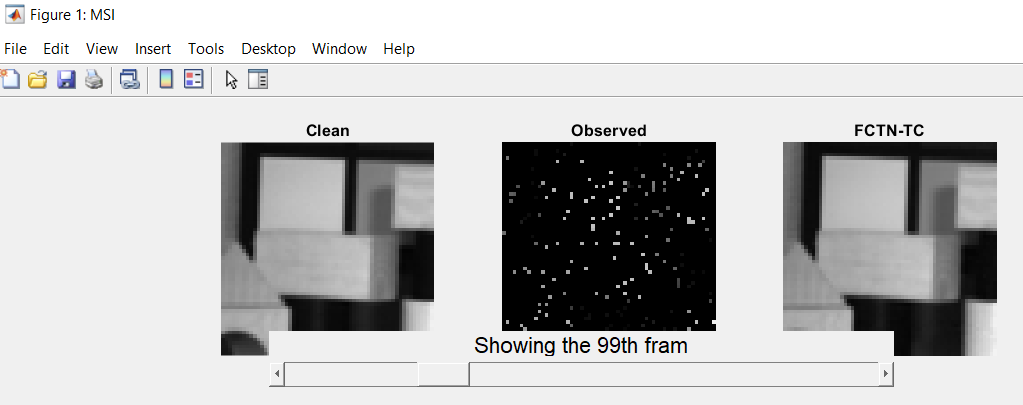
1000 iteration

inc\_FCTN-TC: iter = 990 RSE = 0.000012

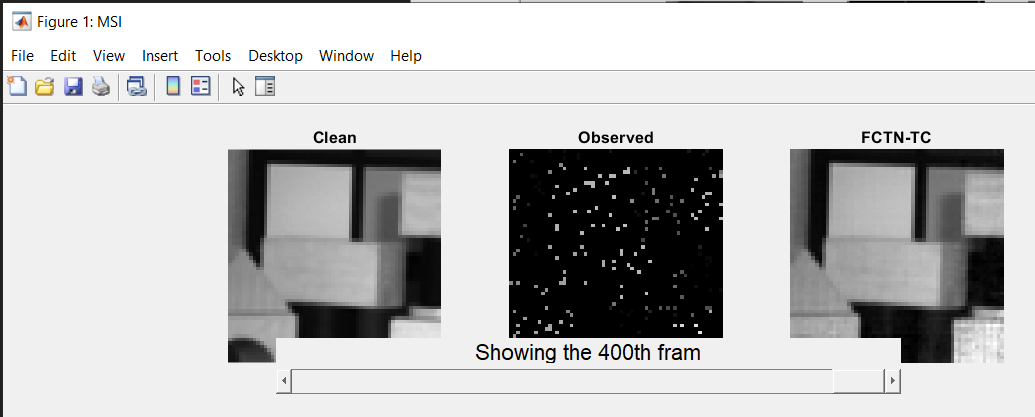
inc\_FCTN-TC: iter = 995 RSE = 0.000012

inc\_FCTN-TC: iter = 1000 RSE = 0.000012









================== Result =====================

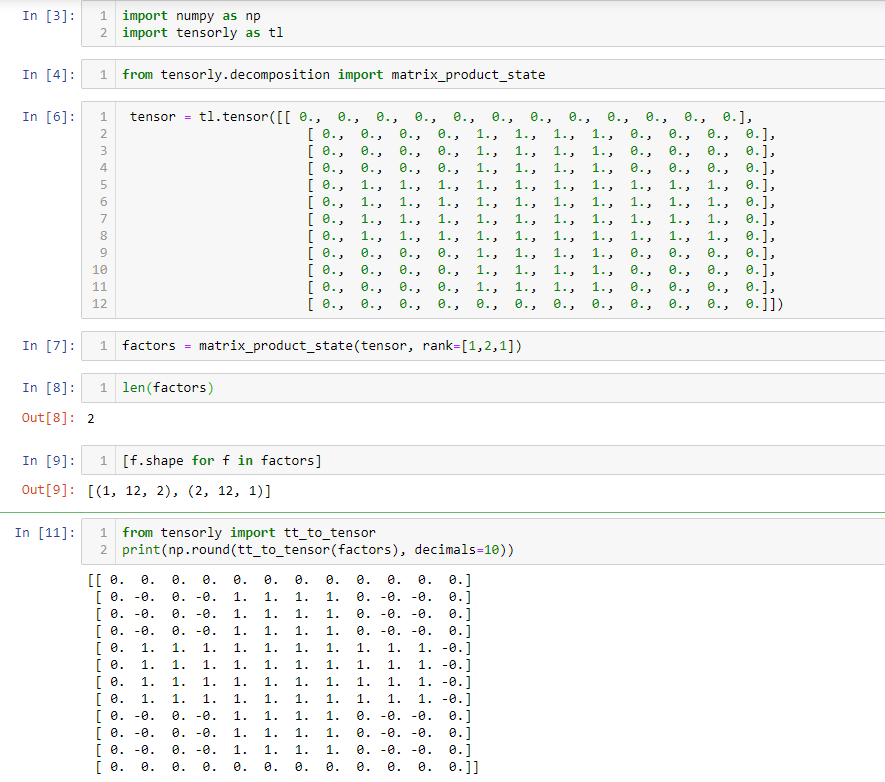
method PSNR SSIM

Observed 8.416 0.016

FCTN-TC 43.912 0.993

================== Result =====================

**Python code for TT:**

****

**Reference:**

*Fully-Connected Tensor Network Decomposition and Its Application to Higher-Order Tensor Completion* Y. Zheng, T. Huang, X. Zhao, Q. Zhao, T. Jiang. [C] AAAI 2021

*Tensor Decompositions and Applications* Tamara G. Kolda, Brett W. Bader

*Tensor Factorization and Tensor Networks for Machine Learning* Qibin Zhao

IFT 6760A - Lecture 11 Tensor Train Decomposition

*On algorithms for and computing with the tensor ring decomposition* Oscar Mickelin Sertac Karaman

*TENSOR-TRAIN DECOMPOSITION* . OSELEDETS