Recent advances on low-rank and sparse decomposition for moving object detection

Matrix and Tensor-based approaches

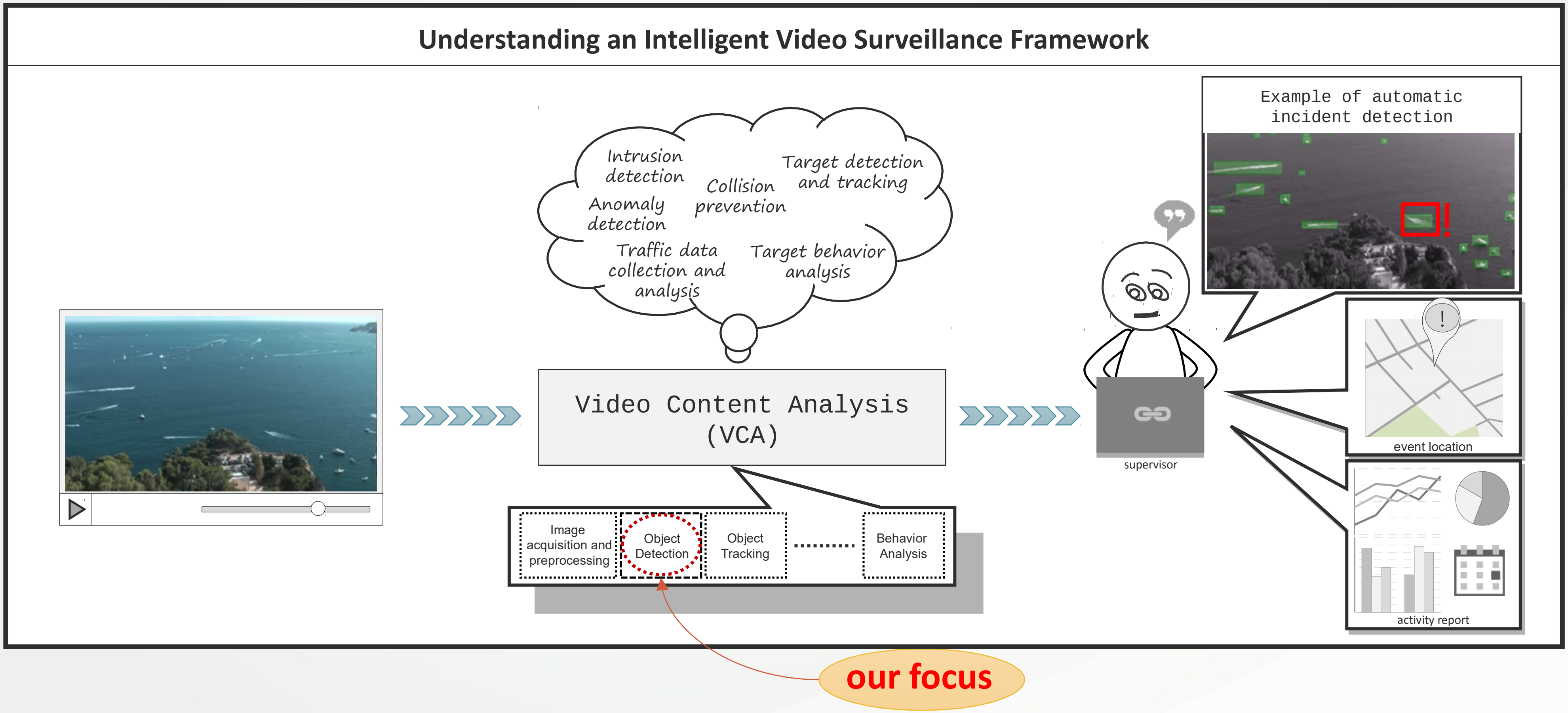
Atelier : Enjeux dans la détection d’objets mobiles par soustraction de fond

Andrews Cordolino Sobral
Ph.D. Student, Computer Vision
L3i / MIA, Université de La Rochelle
http://andrewssobral.wix.com/home

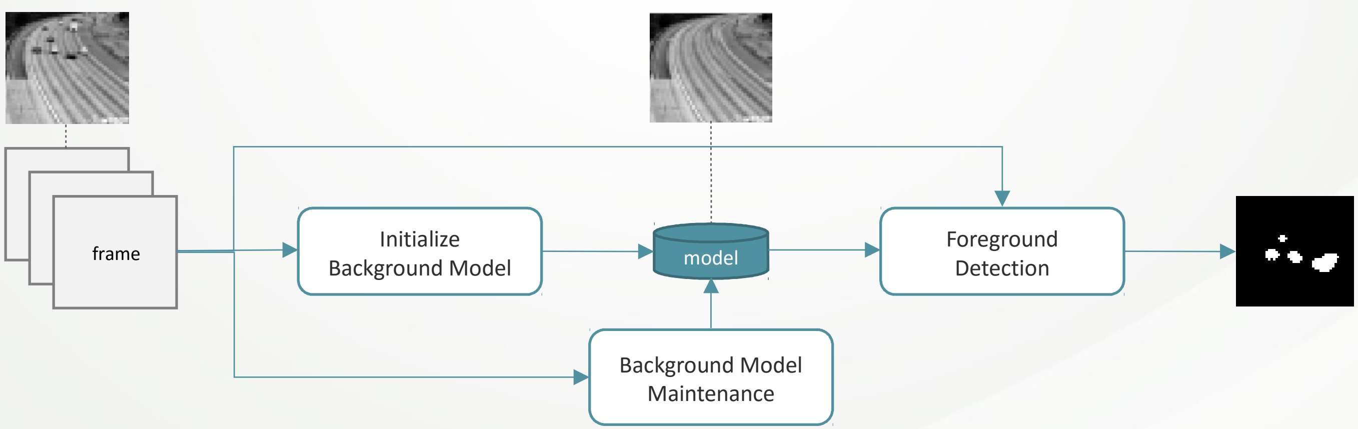
Summary

- Context
  - Understanding an Intelligent Video Surveillance Framework
  - Introduction to Background Subtraction
- Decomposition into Additive Matrices
  - Case 1: Low-rank Approximation and Matrix Completion
  - Case 2: Robust Principal Component Analysis (RPCA)
  - Case 3: Stable decomposition
    - Constrained RPCA
- Introduction to Tensors
  - Tensor Decomposition
    - Tucker/HoSVD
    - CANDECOMP-PARAFAC (CP)
    - Applications to Background Subtraction

Behind the Scenes of an Intelligent Video Surveillance Framework



Introduction to Background Subtraction



Background Subtraction Methods

A large number of algorithms have been proposed for background subtraction over the last few years:

Traditional methods:

- Basic methods, mean and variance over time
- Fuzzy based methods
- Statistical methods
- Non-parametric methods
- Neural and neuro-fuzzy methods

BGSLibrary (C++)
https://github.com/andrewssobral/bgslibrary

Matrix and Tensor Factorization methods:

- Eigenspace-based methods (PCA / SVD)
- RPCA, LRR, NMF, MC, ST, etc.
- Tensor Decomposition, NTF, etc.

LRSLibrary (MATLAB)
https://github.com/andrewssobral/lrslibrary

our focus

Andrews Sobral and Antoine Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. Computer Vision and Image Understanding (CVIU), 2014.

Bouwmans, Thierry; Sobral, Andrews; Javed, Sajid; Ki Jung, Soon; Zahzah, El-Hadi. "Decomposition into Low-rank plus Additive Matrices for Background/Foreground Separation: A Review for a Comparative Evaluation with a Large-Scale Dataset". Submitted to Computer Science Review, 2015.

Background Subtraction Methods

A large number of algorithm
last few years:

Traditional methods:
- Basic methods, mean and vari
- Fuzzy based methods
- Statistical methods
- Non-parametric methods
- Neural and neuro-fuzzy metho

Matrix and Tensor Factorization m
- Eigenspace-based methods (P
- RPCA, LRR, NMF, MC, ST, etc
- Tensor Decomposition, NTF, e

Glossary of terms:

PCA	Principal Component Analysis
SVD	Singular Value Decomposition
LRA	Low-rank Approximation
MC	Matrix Completion
NMF	Non-negative Matrix Factorization
RPCA	Robust Principal Component Analysis
LRR	Low-rank Recovery
RNMF	Robust NMF
ST	Subspace Tracking
Stable RPCA	Stable version of RPCA
TTD	Three-Term Decomposition
TD	Tensor Decomposition
NTF	Non-negative Tensor Factorization

Andrews Sobral and Antoine Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. Computer Vision and Image Understanding (CVIU), 2014.

Bouwmans, Thierry; Sobral, Andrews; Javed, Sajid; Ki Jung, Soon; Zahzah, El-Hadi. "Decomposition into Low-rank plus Additive Matrices for Background/Foreground Separation: A Review for a Comparative Evaluation with a Large-Scale Dataset". Submitted to Computer Science Review, 2015.

Decomposition into Additive Matrices

- The decomposition is represented in a general formulation:

$$A = \sum_{k=1}^{K} M_k$$

- where **K** usually is equal to 1, 2, or 3. For **K** = 3, $\mathbf{M_1} \dots \mathbf{M_3}$ are commonly defined by:

$$A = M_1 + M_2 + M_3 = L + S + E$$

- The characteristics of the matrices $\mathbf{M_K}$ are as follows:
  - The first matrix $\mathbf{M_1 = L}$ is the low-rank component.
  - The second matrix $\mathbf{M_2 = S}$ is the sparse component.
  - The third matrix $\mathbf{M_3 = E}$ is generally the noise component.
- When **K = 1**, the matrix **A ≈ L** and **S** (implicit) can be given by **S = A – L**. e.g.: LRA, MC, NMF, ...
- When **K = 2**, **A = L + S**. This decomposition is called explicit. e.g.: RPCA, LRR, RNMF, …
- When **K = 3**, **A = L + S + E**. This decomposition is called stable. e.g.: Stable RPCA / Stable PCP.

Bouwmans, Thierry; Sobral, Andrews; Javed, Sajid; Ki Jung, Soon; Zahzah, El-Hadi. "Decomposition into Low-rank plus Additive Matrices for Background/Foreground Separation: A Review for a Comparative Evaluation with a Large-Scale Dataset". Submitted to Computer Science Review, 2015.

Decomposition into Additive Matrices

- The decomposition is represented in a general formulation:

$$A = \sum_{k=1}^{K} M_k$$

- where **K** usually is equal to 1, 2, or 3. For **K** = 3, $\mathbf{M_1}$ **…** $\mathbf{M_3}$ are commonly defined by:

$$A = M_1 + M_2 + M_3 = L + S + E$$

- The characteristics of the matrices $\mathbf{M_K}$ are as follows:
  - The first matrix $\mathbf{M_1 = L}$ is the low-rank component.
  - The second matrix $\mathbf{M_2 = S}$ is the sparse component.
  - The third matrix $\mathbf{M_3 = E}$ is generally the noise component.
- When **K = 1**, the matrix **A ≈ L** and **S** (implicit) can be given by **S = A – L**. e.g.: LRA, MC, NMF, ...
- When **K = 2**, **A = L + S**. This decomposition is called explicit. e.g.: RPCA, LRR, RNMF, …
- When **K = 3**, **A = L + S + E**. This decomposition is called stable. e.g.: Stable RPCA / Stable PCP.

Bouwmans, Thierry; Sobral, Andrews; Javed, Sajid; Ki Jung, Soon; Zahzah, El-Hadi. "Decomposition into Low-rank plus Additive Matrices for Background/Foreground Separation: A Review for a Comparative Evaluation with a Large-Scale Dataset". Submitted to Computer Science Review, 2015.

Low Rank Approximation

- Low-rank approximation (LRA) is a minimization problem, in which the cost function measures the fit between a given matrix (the data) and an approximating matrix (the optimization variable), subject to a constraint that the approximating matrix has reduced rank.

$$\begin{aligned} \underset{L}{\text{minimize}} \quad & ||A - L||_F \\ \text{subject to} \quad & \text{rank}(L) < r, \ \text{(desired rank)}. \end{aligned}$$

usually $rank(L)$ is remplaced by $||L||_* = \sum_{i=1}^{r} \sigma_i$ where $\sigma_1, ..., \sigma_r$ are the singular values of L.

Low Rank Approximation

- Low-rank approximation (LRA) is a minimization problem, in which the cost function measures the fit between a given matrix (the data) and an approximating matrix (the optimization variable), subject to a constraint that the approximating matrix has reduced rank.

$$\underset{L}{\text{minimize}} \quad ||A - L||_F$$

$$\text{subject to} \quad \text{rank}(L) < r, \ \text{(desired rank)}.$$

usually $rank(L)$ is remplaced by $||L||_* = \sum_{i=1}^{r} \sigma_i$ where $\sigma_1, ..., \sigma_r$ are the singular values of L.

!!! Singular Value Decomposition !!!

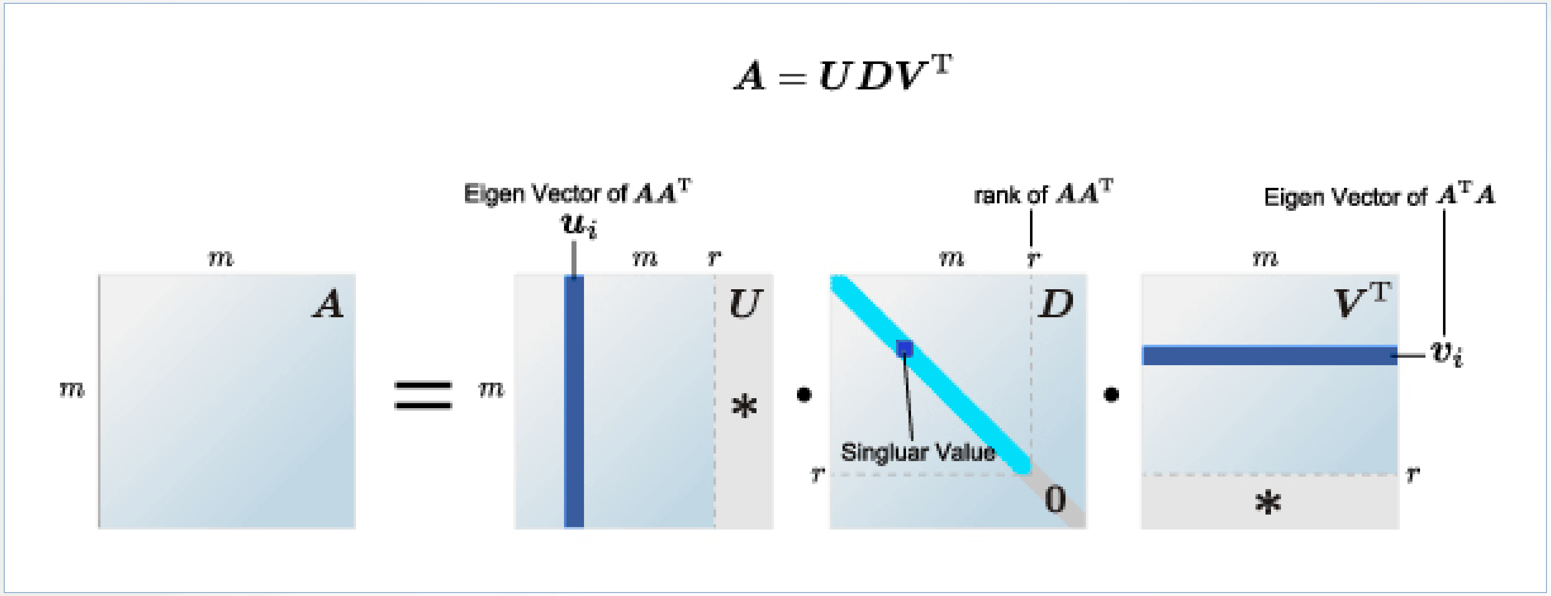
Singular Value Decomposition

- Formally, the singular value decomposition of an **m×n** real or complex matrix **A** is a factorization of the form:

$$A = UDV^T$$

- where **U** is a **m×m** real or complex unitary matrix, **D** is an **m×n** rectangular diagonal matrix with non-negative real numbers on the diagonal, and **V**$^{\mathbf{T}}$ (the transpose of **V** if **V** is real) is an **n×n** real or complex unitary matrix. The diagonal entries **D** are known as the singular values of **A**.

- The **m** columns of **U** and the **n** columns of **V** are called the left-singular vectors and right-singular vectors of **A**, respectively.



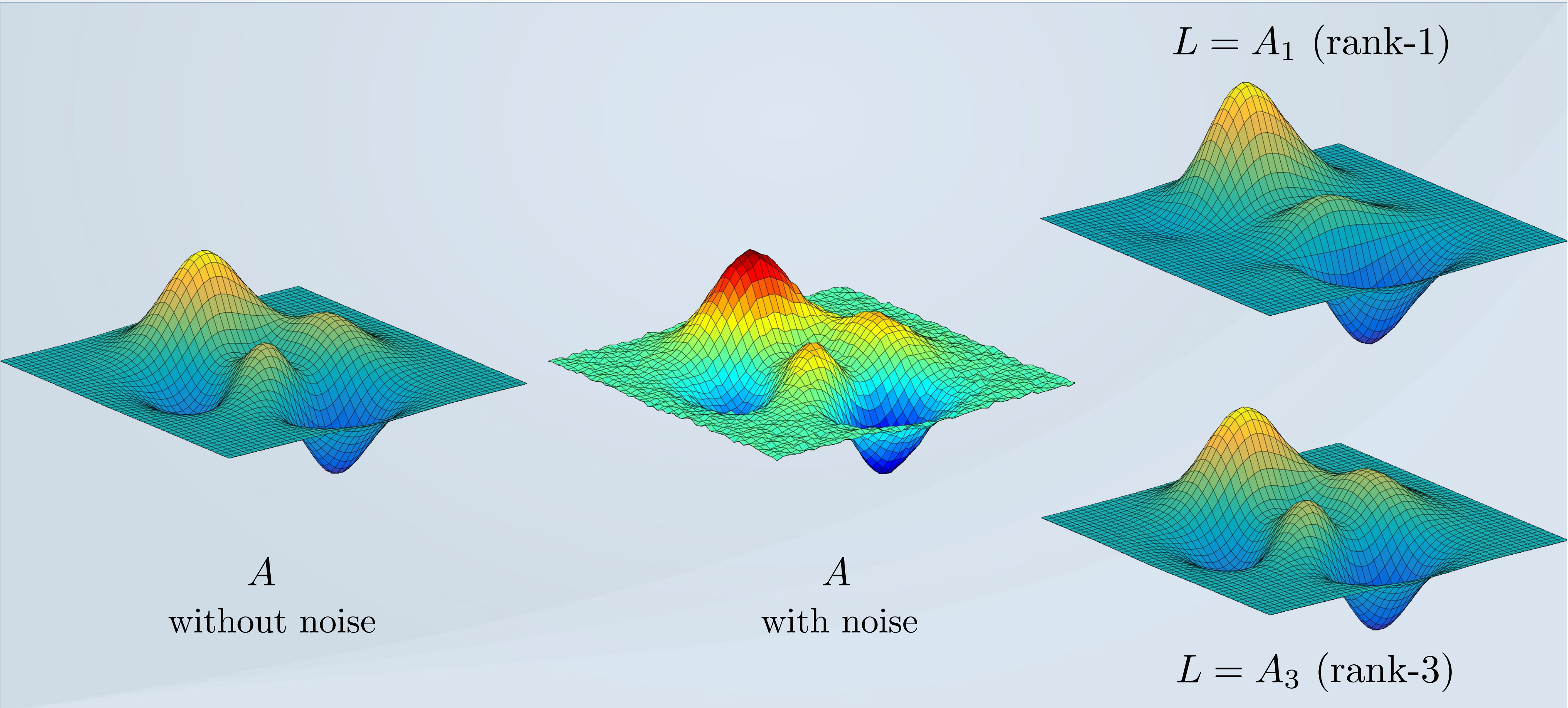
generalization of eigenvalue decomposition

Best rank r Approximation

If $A = UDV^T$ is the SVD of A and the singular values are sorted as $\sigma_1 \geq \sigma_2 \geq \sigma_n$ then for any $r < n$, the best rank-r approximation to A is:

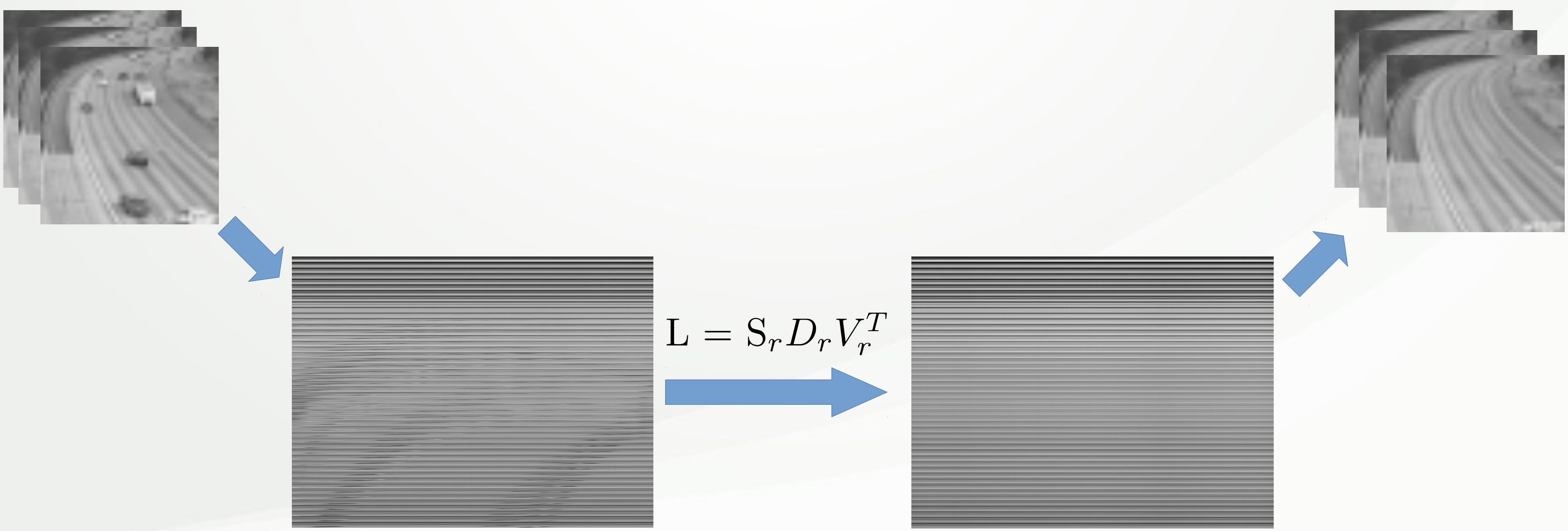
$$L = A_r = U_r D_r V_r^T \text{ or } L = \sum_{i=1}^{r} u_i d_i v_i^T$$

$L = A_r$ minimizes $||A - L||_2$ and $||A - L||_F$ among all rank-r matrices



Background Model Estimation

```matlab
%% Load video
load(fullfile(lrs_conf.lrs_dir,'dataset','trafficdb','traffic_patches.mat'));
V = im2double(imgdb{100});
% convert to 2D matrix
A = convert_video3d_to_2d(V);
%% low-rank appoximation (rank-1)
[m,n] = size(A);
% let us define that rank to be equal to r
r = 1;
% doing it the normal SVD way
[U,S,V] = svd(A);
L = U(1:m,1:r)*S(1:r,1:r)*V(1:n,1:r)';
% ||A - L||_F
norm(abs(A-L),'fro')
```

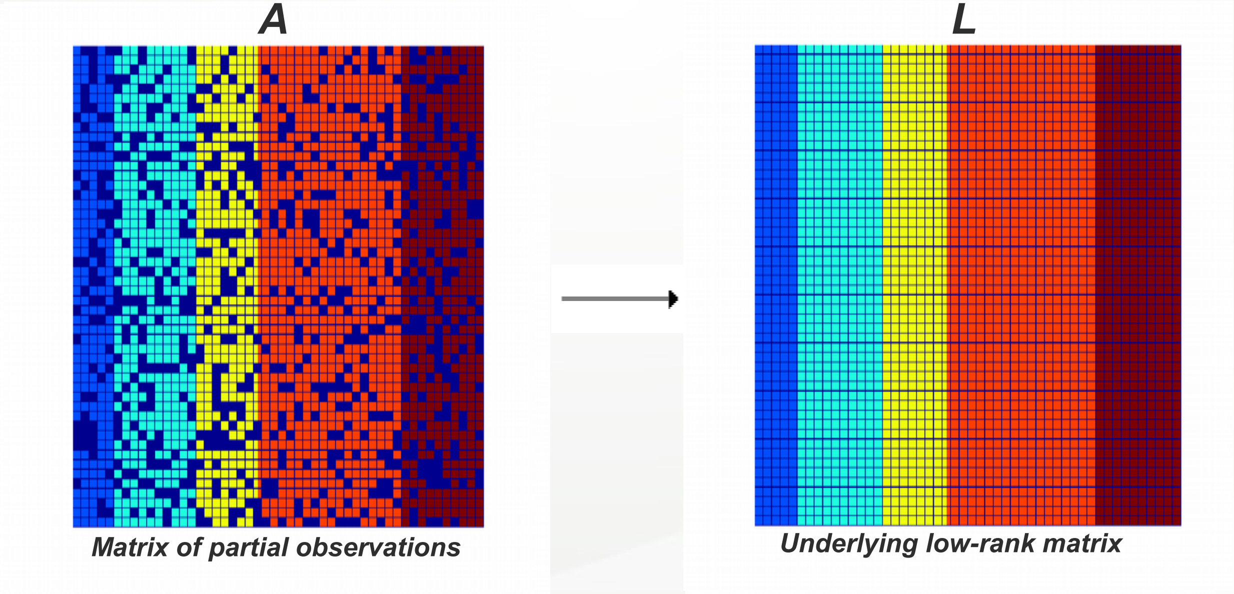


What about LRA for corrupted entries?

Introduction to Matrix Completion (MC)

- Matrix Completion (MC) can be formulated as the problem of o recover a low rank matrix **L** from the partial observations of its entries (represented by **A**):

$$\begin{array}{ll} \underset{L}{\text{minimize}} & \text{rank}(L) \\ \text{subject to} & L_{ij} = A_{ij},\ (i,j) \in \Omega \text{ (set of observed elements).} \end{array}$$



Matrix of partial observations

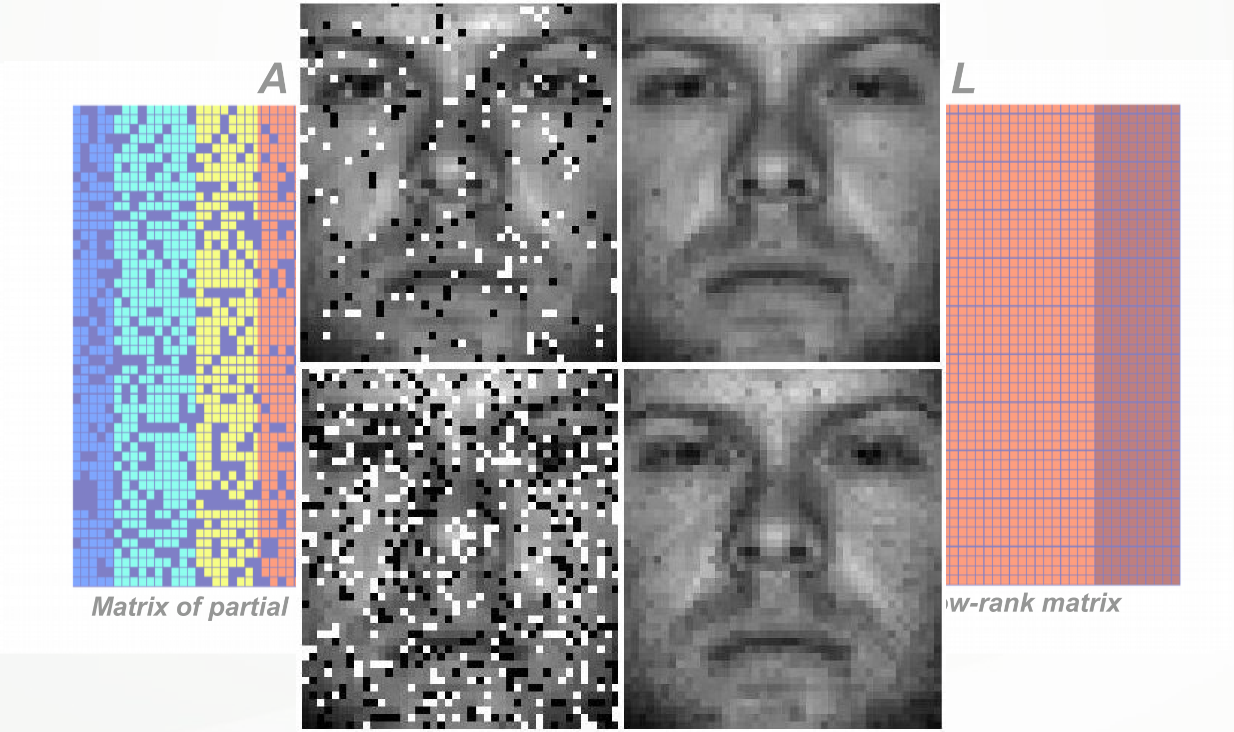
Underlying low-rank matrix

http://perception.csl.illinois.edu/matrix-rank/home.html

Introduction to Matrix Completion (MC)

- Matrix Completion (MC) can be formulated as the problem of o recover a low rank matrix **L** from the partial observations of its entries (represented by **A**):

$$\begin{aligned} \underset{L}{\text{minimize}} \quad & \text{rank}(L) \\ \text{subject to} \quad & L_{ij} = A_{ij},\ (i,j) \in \Omega \text{ (set of observed elements)}. \end{aligned}$$



http://perception.csl.illinois.edu/matrix-rank/home.html

Demo: Matrix Completion

Setup a problem

```
rng(234923);    % for reproducible results
N   = 16;       % the matrix is N x N
r   = 2;        % the rank of the matrix
df  = 2*N*r - r^2;  % degrees of freedom of a N x N rank r matrix
nSamples    = 3*df; % number of observed entries

% For this demo, we will use a matrix with integer entries
% because it will make displaying the matrix easier.
iMax    = 5;
X       = randi(iMax,N,r)*randi(iMax,r,N); % Our target matrix
```

Now suppose we only see a few entries of X. Let "Omega" be the set of observed entries

```
rPerm   = randperm(N^2); % use "randsample" if you have the stats toolbox
omega   = sort( rPerm(1:nSamples) );
```

Print out the observed matrix in a nice format. The "NaN" entries represent unobserved values. The goal of this demo is to find out what those values are!

```
Y = nan(N);
Y(omega) = X(omega);
disp('The "NaN" entries represent unobserved values');
disp(Y)
```

```
The "NaN" entries represent unobserved values
    30   NaN    24     8    12    14    12   NaN    22   NaN   NaN    10    14    24    20   NaN
    30    21   NaN    11    21   NaN    12    21   NaN    15    17   NaN     8    15    23   NaN
   NaN     9   NaN   NaN     9   NaN   NaN   NaN     7     9     8     7     5   NaN    11     6
    35   NaN   NaN    10   NaN   NaN   NaN   NaN    23   NaN    17    13    15    26    24   NaN
   NaN     9     9     5   NaN   NaN   NaN   NaN   NaN     9     8   NaN   NaN     9    11   NaN
   NaN    11    19     7   NaN    13    10    11    17    19    12     9    11    19    17   NaN
    45    24    30   NaN   NaN    29    18   NaN    25    30   NaN    19    17    30    32   NaN
   NaN   NaN    21   NaN    15    18    12    15    18    21    15    12    12    21    21    12
    25    11   NaN     7    11    13    10    11    17    19    12     9   NaN    19   NaN   NaN
   NaN    13    11     7    13    16     8   NaN     8    11    11   NaN   NaN    11    15     8
    45    24    30    14    24    29    18    24    25    30    23    19    17   NaN    32    18
   NaN    15    21     9   NaN    18    12    15    18    21    15    12   NaN    21    21    12
    25   NaN    13     9    17   NaN    10    17   NaN    13    14    13     7    13    19    10
    40    20    28    12    20   NaN    16    20   NaN   NaN   NaN    16    16   NaN    28    16
   NaN   NaN    18    10    18    22    12   NaN    14   NaN    16    14    10   NaN    22    12
    25    11    19     7    11   NaN    10   NaN   NaN   NaN   NaN     9    11    19    17   NaN
```

$$
\begin{array}{ll}
\text{minimize} & ||L||_* \\
\text{subject to} & P_\Omega(L) = P_\Omega(A) \\
& P_\Omega(.) \text{ (is the sampling operator).}
\end{array}
$$

Matrix completion via TFOCS

http://cvxr.com/tfocs/demos/matrixcompletion/

Demo: Matrix Completion

```
% Add TFOCS to your path (modify this line appropriately):
addpath ~/Dropbox/TFOCS/

observations = X(omega);    % the observed entries
mu           = .001;        % smoothing parameter

% The solver runs in seconds
tic
Xk = solver_sNuclearBP( {N,N,omega}, observations, mu );
toc
```

```
Auslender & Teboulle's single-projection method
Iter    Objective   |dx|/|x|    step
----+----------------------------------
100 | +3.54125e+02  1.68e-04  1.32e-03
200 | +3.54125e+02  7.42e-07  1.77e-03*
251 | +3.54125e+02  5.41e-09  2.28e-03*
Finished: Step size tolerance reached
Elapsed time is 1.289716 seconds.
```

```
Recovered matrix (rounding to nearest .0001):
    30    12    24     8    12    14    12    12    22    24    14    10    14    24    20    12
    30    21    15    11    21    26    12    21    10    15    17    16     8    15    23    12
    15     9     9     5     9    11     6     9     7     9     8     7     5     9    11     6
    35    16    26    10    16    19    14    16    23    26    17    13    15    26    24    14
    15     9     9     5     9    11     6     9     7     9     8     7     5     9    11     6
    25    11    19     7    11    13    10    11    17    19    12     9    11    19    17    10
    45    24    30    14    24    29    18    24    25    30    23    19    17    30    32    18
    30    15    21     9    15    18    12    15    18    21    15    12    12    21    21    12
    25    11    19     7    11    13    10    11    17    19    12     9    11    19    17    10
    20    13    11     7    13    16     8    13     8    11    11    10     6    11    15     8
    45    24    30    14    24    29    18    24    25    30    23    19    17    30    32    18
    30    15    21     9    15    18    12    15    18    21    15    12    12    21    21    12
    25    17    13     9    17    21    10    17     9    13    14    13     7    13    19    10
    40    20    28    12    20    24    16    20    24    28    20    16    16    28    28    16
    30    18    18    10    18    22    12    18    14    18    16    14    10    18    22    12
    25    11    19     7    11    13    10    11    17    19    12     9    11    19    17    10

Original matrix:
    30    12    24     8    12    14    12    12    22    24    14    10    14    24    20    12
    30    21    15    11    21    26    12    21    10    15    17    16     8    15    23    12
    15     9     9     5     9    11     6     9     7     9     8     7     5     9    11     6
    35    16    26    10    16    19    14    16    23    26    17    13    15    26    24    14
    15     9     9     5     9    11     6     9     7     9     8     7     5     9    11     6
    25    11    19     7    11    13    10    11    17    19    12     9    11    19    17    10
    45    24    30    14    24    29    18    24    25    30    23    19    17    30    32    18
    30    15    21     9    15    18    12    15    18    21    15    12    12    21    21    12
    25    11    19     7    11    13    10    11    17    19    12     9    11    19    17    10
    20    13    11     7    13    16     8    13     8    11    11    10     6    11    15     8
    45    24    30    14    24    29    18    24    25    30    23    19    17    30    32    18
    30    15    21     9    15    18    12    15    18    21    15    12    12    21    21    12
    25    17    13     9    17    21    10    17     9    13    14    13     7    13    19    10
    40    20    28    12    20    24    16    20    24    28    20    16    16    28    28    16
    30    18    18    10    18    22    12    18    14    18    16    14    10    18    22    12
    25    11    19     7    11    13    10    11    17    19    12     9    11    19    17    10

Relative error, no rounding: 0.00000410%
```

http://cvxr.com/tfocs/demos/matrixcompletion/

MC Algorithms

- **LRSLibrary**:
  - MC: Matrix Completion (14)
    - FPC: Fixed point and Bregman iterative methods for matrix rank minimization (Ma et al. 2008)
    - GROUSE: Grassmannian Rank-One Update Subspace Estimation (Balzano et al. 2010)
    - IALM-MC: Inexact ALM for Matrix Completion (Lin et al. 2009)
    - LMaFit: Low-Rank Matrix Fitting (Wen et al. 2012)
    - LRGeomCG: Low-rank matrix completion by Riemannian optimization (Bart Vandereycken, 2013)
    - MC_logdet: Top-N Recommender System via Matrix Completion (Kang et al. 2016)
    - MC-NMF: Nonnegative Matrix Completion (Xu et al. 2011)
    - OP-RPCA: Robust PCA via Outlier Pursuit (Xu et al. 2012)
    - OptSpace: Matrix Completion from Noisy Entries (Keshavan et al. 2009)
    - OR1MP: Orthogonal rank-one matrix pursuit for low rank matrix completion (Wang et al. 2015)
    - RPCA-GD: Robust PCA via Gradient Descent (Yi et al. 2016)
    - ScGrassMC: Scaled Gradients on Grassmann Manifolds for Matrix Completion (Ngo and Saad, 2012)
    - SVP: Guaranteed Rank Minimization via Singular Value Projection (Meka et al. 2009)
    - SVT: A singular value thresholding algorithm for matrix completion (Cai et al. 2008)

https://github.com/andrewssobral/lrslibrary

MC Algorithms

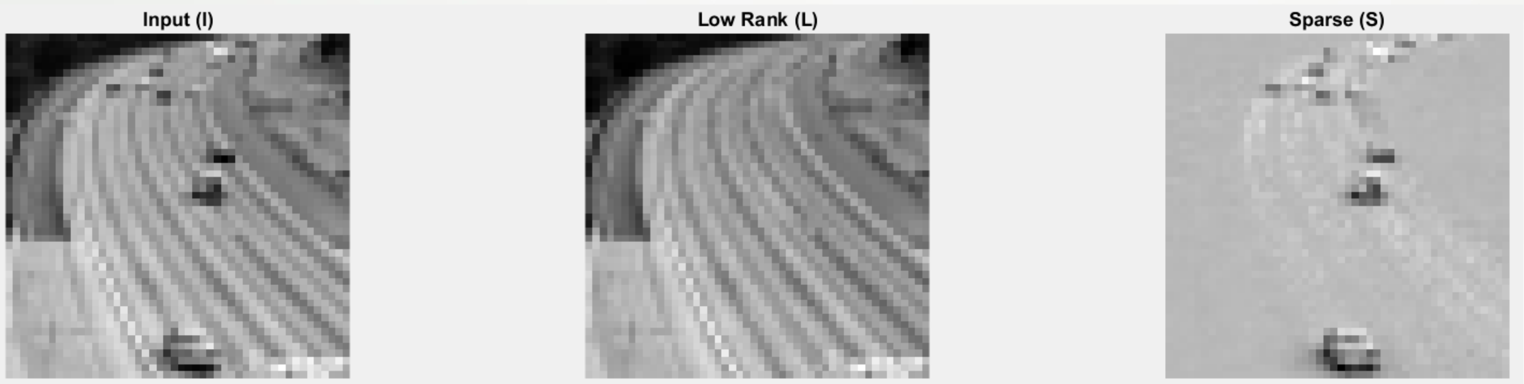
- **LRSLibrary**:
  - MC: Matrix Completion (14)
    - FPC: Fixed point and Bregman iterative methods for matrix rank minimization (Ma et al. 2008)
    - GROUSE: Grassmannian Rank-One Update Subspace Estimation (Balzano et al. 2010)
    - IALM-MC: Inexact ALM for Matrix Completion (Lin et al. 2009)
    - LMaFit: Low-Rank Matrix Fitting (Wen et al. 2012)
    - LRGeomCG: Low-rank matrix completion by Riemannian optimization (Bart Vandereycken, 2013)
    - MC_logdet: Top-N Recommender System via Matrix Completion (Kang et al. 2016)
    - MC-NMF: Nonnegative Matrix Completion (Xu et al. 2011)
    - OP-RPCA: Robust PCA via Outlier Pursuit (Xu et al. 2012)
    - OptSpace: Matrix Completion from Noisy Entries (Keshavan et al. 2009)
    - OR1MP: Orthogonal rank-one matrix pursuit for low rank matrix completion (Wang et al. 2015)
    - RPCA-GD: Robust PCA via Gradient Descent (Yi et al. 2016)
    - ScGrassMC: Scaled Gradients on Grassmann Manifolds for Matrix Completion (Ngo and Saad, 2012)
    - SVP: Guaranteed Rank Minimization via Singular Value Projection (Meka et al. 2009)
    - SVT: A singular value thresholding algorithm for matrix completion (Cai et al. 2008)

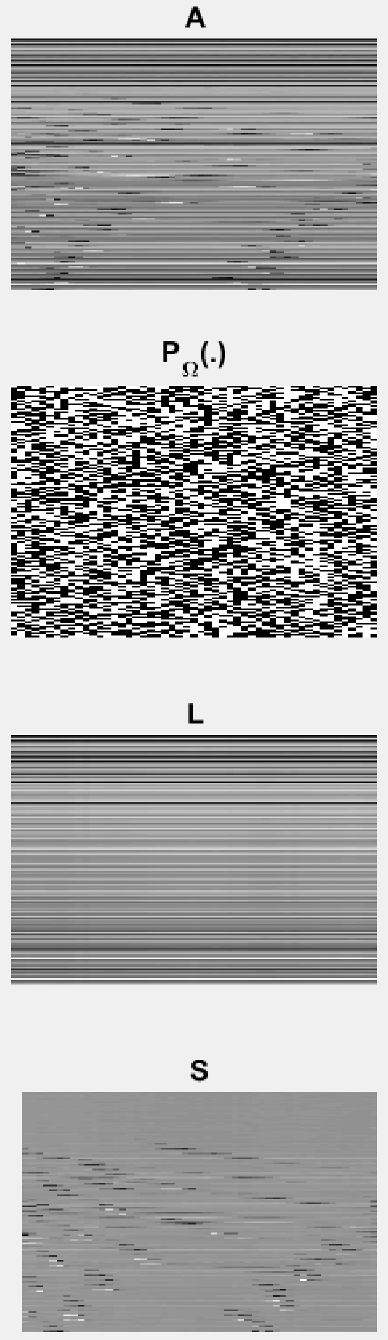
https://github.com/andrewssobral/lrslibrary

Demo: LRSLibrary for MC

https://github.com/andrewssobral/lrslibrary/blob/master/algorithms/mc/GROUSE/run_alg.m

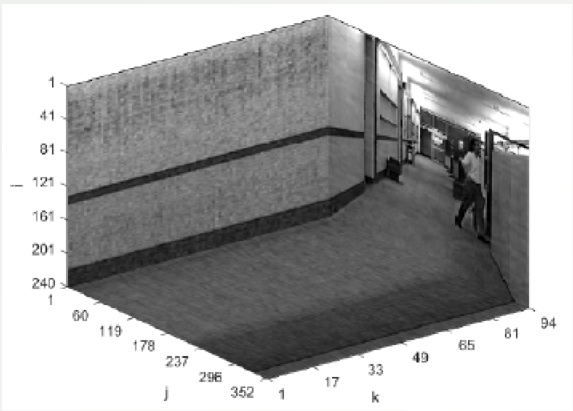
```
% MC | GROUSE | Grassmannian Rank-One Update Subspace Estimation (Balzano et al. 2010)
% process_video('MC', 'GROUSE', 'dataset/demo.avi', 'output/demo_MC-GROUSE.avi');

[numr,numc] = size(M);
I = randi([0 1],numr,numc); % ones(size(M));
maxrank = 1;
maxCycles = 100;
step_size = 0.1;

[Usg, Vsg, err_reg] = grouse(M,I,numr,numc,maxrank,step_size,maxCycles);
L = Usg*Vsg';
S = M - L;

% show_2dvideo(M,m,n);
% show_2dvideo(M.*I,m,n);
% show_2dvideo(L,m,n);
% show_2dvideo(S,m,n);
```

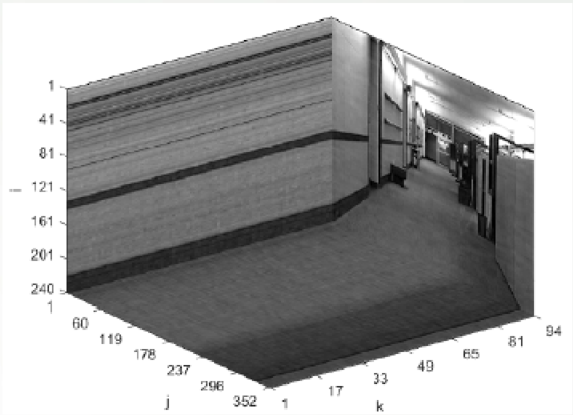


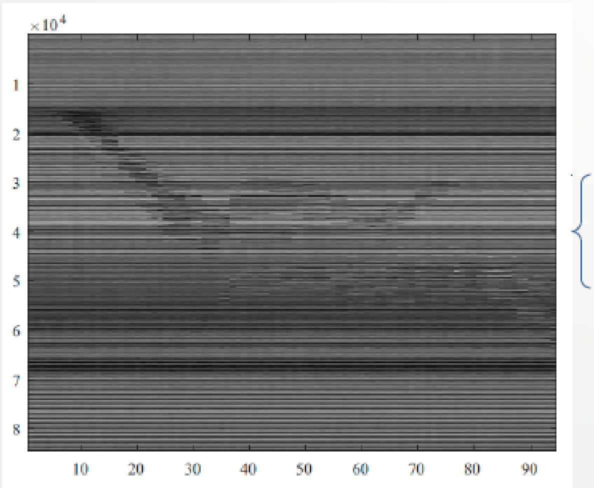


https://github.com/andrewssobral/lrslibrary

MC for Background Model Initialization

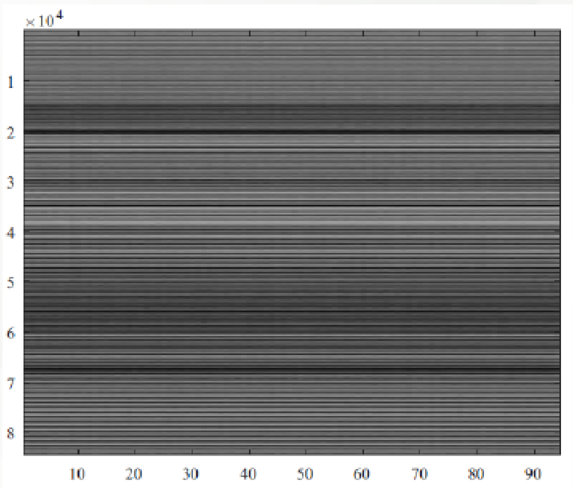






A

$$\left\{\begin{array}{ll} \text{minimize} & \|L\|_* \\ \text{subject to} & P_\Omega(L) = P_\Omega(A) \end{array}\right\}$$



L

where $P_\Omega(.)$ is the sampling operator.

Sobral, Andrews; Bouwmans, Thierry; Zahzah, El-hadi. "Comparison of Matrix Completion Algorithms for Background Initialization in Videos". Scene Background Modeling and Initialization (SBMI), Workshop in conjunction with ICIAP 2015, Genova, Italy, September, 2015.

Decomposition into Additive Matrices

- The decomposition is represented in a general formulation:

$$A = \sum_{k=1}^{K} M_k$$

- where **K** usually is equal to 1, 2, or 3. For **K** = 3, $\mathbf{M_1}$ **…** $\mathbf{M_3}$ are commonly defined by:

$$A = M_1 + M_2 + M_3 = L + S + E$$

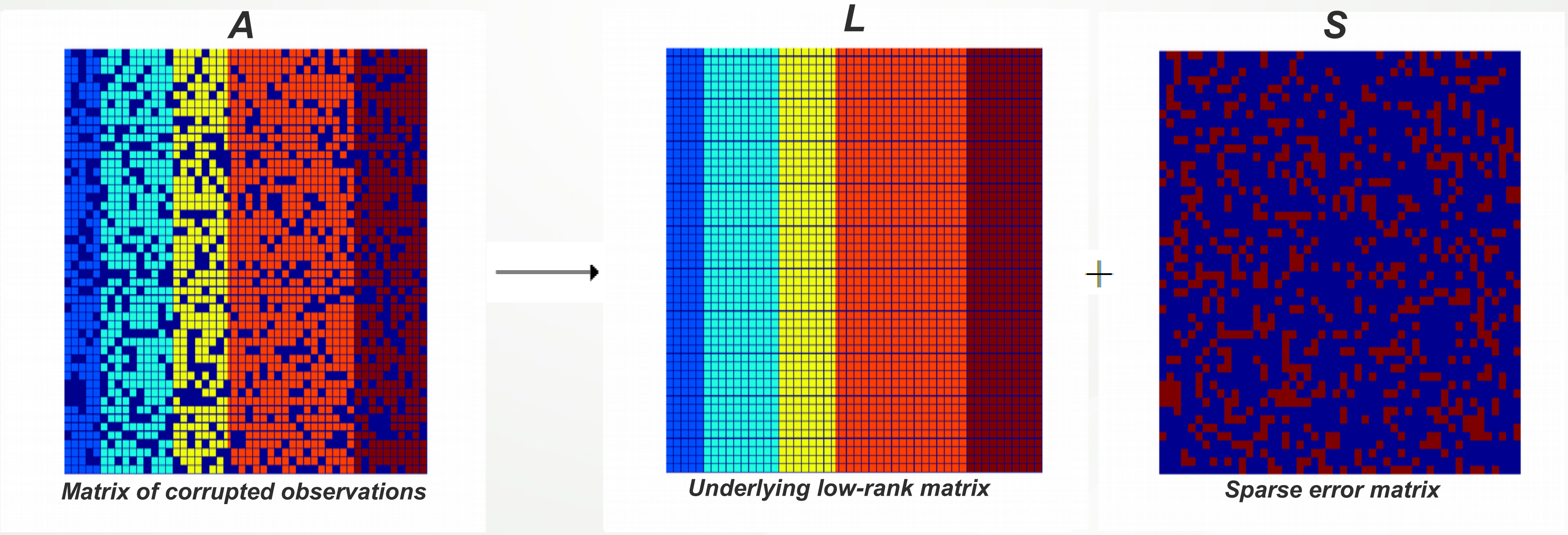
- The characteristics of the matrices $\mathbf{M_K}$ are as follows:
  - The first matrix $\mathbf{M_1 = L}$ is the low-rank component.
  - The second matrix $\mathbf{M_2 = S}$ is the sparse component.
  - The third matrix $\mathbf{M_3 = E}$ is generally the noise component.
- When **K = 1**, the matrix **A ≈ L** and **S** (implicit) can be given by **S = A – L**. e.g.: LRA, MC, NMF, ...
- When **K = 2**, **A = L + S**. This decomposition is called explicit. e.g.: RPCA, LRR, RNMF, …
- When **K = 3**, **A = L + S + E**. This decomposition is called stable. e.g.: Stable RPCA / Stable PCP.

Bouwmans, Thierry; Sobral, Andrews; Javed, Sajid; Ki Jung, Soon; Zahzah, El-Hadi. "Decomposition into Low-rank plus Additive Matrices for Background/Foreground Separation: A Review for a Comparative Evaluation with a Large-Scale Dataset". Submitted to Computer Science Review, 2015.

Robust Principal Component Analysis (RPCA)

- RPCA can be formulated as the problem of decomposing a data matrix **A** into two components **L** and **S**, where **A** is the sum of a low-rank matrix **L** and a sparse matrix **S**:

$$A = L + S$$



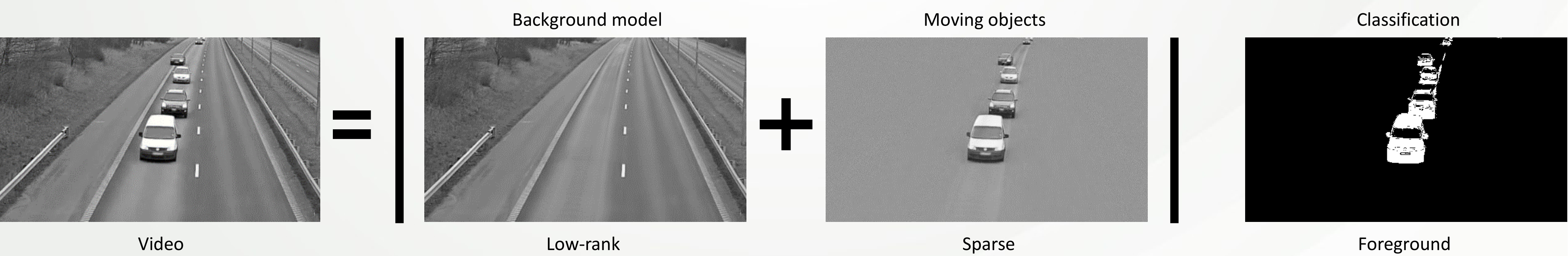
Robust Principal Component Analysis (RPCA)

- Candès et al. (2009) show that **L** and **S** can be recovered by solving a convex optimization problem, named as Principal Component Pursuit (PCP):

$$\begin{aligned} &\text{minimize} \quad ||L||_* + \lambda ||S||_1 \\ &\text{subject to} \quad A = L + S, \text{ where } \lambda \text{ is a weighting parameter.} \end{aligned}$$

$||L||_*$ enforces low rank in L.

$||S||_1$ enforces the sparsity in S.



Solving PCP

One effective way to solve PCP for the case of large matrices is to use a standard augmented Lagrangian multiplier method (ALM) (Bertsekas, 1982).

$$\ell(\mathbf{L},\mathbf{S},\mathbf{Y}) \triangleq \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 + \langle \mathbf{Y}, \mathbf{M}-\mathbf{L}-\mathbf{S}\rangle + \mu\|\mathbf{M}-\mathbf{L}-\mathbf{S}\|_F^2 \quad (1)$$

and then minimizing it iteratively by setting

$$(\mathbf{L}^{(k)},\mathbf{S}^{(k)}) = \arg\min_{(\mathbf{L},\mathbf{S})} \ell(\mathbf{L},\mathbf{S},\mathbf{Y}^{(k)}) \quad (2)$$

and updating $\mathbf{Y}^{(k+1)} \leftarrow \mathbf{Y}^{(k)} + \mu(\mathbf{M}-\mathbf{L}^{(k)}-\mathbf{S}^{(k)})$.

where:

$$\arg\min_{\mathbf{S}} \ell(\mathbf{L},\mathbf{S},\mathbf{Y}) = \mathcal{S}_{\lambda\mu^{-1}}(\mathbf{M}-\mathbf{L}+\mu^{-1}\mathbf{Y}) \quad (3)$$

$$\arg\min_{\mathbf{L}} \ell(\mathbf{L},\mathbf{S},\mathbf{Y}) = \mathcal{D}_{\mu^{-1}}(\mathbf{M}-\mathbf{S}+\mu^{-1}\mathbf{Y}). \quad (4)$$

$\mathcal{S}_\tau(x) \triangleq \operatorname{sgn}(x)\max\{|x|-\tau,0\}$.

$\mathcal{D}_\tau(\mathbf{A}) \triangleq \mathbf{U}\mathcal{S}_\tau(\mathbf{\Sigma})\mathbf{V}^T$

$\langle\mathbf{A},\mathbf{B}\rangle \triangleq \operatorname{tr}(\mathbf{A}^T\mathbf{B})$, where $\operatorname{tr}(\cdot)$ denotes the trace operator

$\lambda = 1/\sqrt{\max\{n_1,n_2\}}$ and $\mu = (n_1 n_2)/(4\|\mathbf{M}\|_1)$

Algorithm 1 ALM using alternating directions [2], [3], [5]

input: $\mathbf{M} \in \mathbb{R}^{n_1\times n_2}$
initialize: $\mathbf{S}^{(0)} = \mathbf{Y}^{(0)} = \mathbf{0}$, $\lambda = 1/\sqrt{\max\{n_1,n_2\}}$, $\mu = (n_1 n_2)/(4\|\mathbf{M}\|_1)$, $k=0$
while not converged **do**
  $\mathbf{L}^{(k+1)} \leftarrow \mathcal{D}_{\mu^{-1}}(\mathbf{M}-\mathbf{S}^{(k)}+\mu^{-1}\mathbf{Y}^{(k)})$
  $\mathbf{S}^{(k+1)} \leftarrow \mathcal{S}_{\lambda\mu^{-1}}(\mathbf{M}-\mathbf{L}^{(k+1)}+\mu^{-1}\mathbf{Y}^{(k)})$
  $\mathbf{Y}^{(k+1)} \leftarrow \mathbf{Y}^{(k)}+\mu(\mathbf{M}-\mathbf{L}^{(k+1)}-\mathbf{S}^{(k+1)})$
end while
output: $\mathbf{L}^{(k)}, \mathbf{S}^{(k)}$

shrinkage operator $D_\tau(.)$

$$\begin{matrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \cancel{\lambda_3} \end{matrix}$$

More information:
(Qiu and Vaswani, 2011), (Pope et al. 2011), (Rodríguez and Wohlberg, 2013)

RPCA solvers

Algorithm 1 (RPCA via Iterative Thresholding)

Input: Observation matrix $D \in \mathbb{R}^{m \times n}$, weights λ and τ.

1: **while** not converged **do**
2: $(U, S, V) = \mathrm{svd}(Y_{k-1})$,
3: $A_k = U \mathcal{S}_\tau[S] V^T$,
4: $E_k = \mathcal{S}_{\lambda\tau}[Y_{k-1}]$,
5: $Y_k = Y_{k-1} + \delta_k (D - A_k - E_k)$.
6: **end while**

Output: $A \leftarrow A_k$, $E \leftarrow E_k$.

Algorithm 5 (RPCA via the Inexact ALM Method)

Input: Observation matrix $D \in \mathbb{R}^{m \times n}$, λ.

1: $Y_0 = D/J(D)$; $E_0 = 0$; $\mu_0 > 0$; $\rho > 1$; $k = 0$.
2: **while** not converged **do**
3: // Lines 4-5 solve $A_{k+1} = \arg\min_A L(A, E_k, Y_k, \mu_k)$.
4: $(U, S, V) = \mathrm{svd}(D - E_k + \mu_k^{-1} Y_k)$;
5: $A_{k+1} = U \mathcal{S}_{\mu_k^{-1}}[S] V^T$.
6: // Line 7 solves $E_{k+1} = \arg\min_E L(A_{k+1}, E, Y_k, \mu_k)$.
7: $E_{k+1} = \mathcal{S}_{\lambda\mu_k^{-1}}[D - A_{k+1} + \mu_k^{-1} Y_k]$.
8: $Y_{k+1} = Y_k + \mu_k (D - A_{k+1} - E_{k+1})$.
9: Update μ_k to μ_{k+1}.
10: $k \leftarrow k + 1$.
11: **end while**

Output: (A_k, E_k).

Algorithm 2 (RPCA via Accelerated Proximal Gradient)

Input: Observation matrix $D \in \mathbb{R}^{m \times n}$, λ.

1: $A_0 = A_{-1} = 0$; $E_0 = E_{-1} = 0$; $t_0 = t_{-1} = 1$; $\bar{\mu} > 0$; $\eta < 1$.
2: **while** not converged **do**
3: $Y_k^A = A_k + \frac{t_{k-1}-1}{t_k}(A_k - A_{k-1})$, $Y_k^E = E_k + \frac{t_{k-1}-1}{t_k}(E_k - E_{k-1})$.
4: $G_k^A = Y_k^A - \frac{1}{2}\left(Y_k^A + Y_k^E - D\right)$.
5: $(U, S, V) = \mathrm{svd}(G_k^A)$, $A_{k+1} = U \mathcal{S}_{\frac{\mu_k}{2}}[S] V^T$.
6: $G_k^E = Y_k^E - \frac{1}{2}\left(Y_k^A + Y_k^E - D\right)$.
7: $E_{k+1} = \mathcal{S}_{\frac{\lambda\mu_k}{2}}[G_k^E]$.
8: $t_{k+1} = \frac{1+\sqrt{4t_k^2+1}}{2}$; $\mu_{k+1} = \max(\eta\,\mu_k, \bar{\mu})$.
9: $k \leftarrow k + 1$.
10: **end while**

Output: $A \leftarrow A_k$, $E \leftarrow E_k$.

Robust PCA Algorithm Comparison

Algorithm	Rank of estimate	Relative error in estimate of A	Time (s)
Singular Value Thresholding	20	3.4×10^{-4}	877
Accelerated Proximal Gradient	20	2.0×10^{-5}	43
Accelerated Proximal Gradient (with partial SVDs)	20	1.8×10^{-5}	8
Dual Method	20	1.6×10^{-5}	177
Exact ALM	20	7.6×10^{-8}	4
Inexact ALM	20	4.3×10^{-8}	2
Alternating Direction Methods	20	2.2×10^{-5}	5

For more information see: (Lin et al., 2010) http://perception.csl.illinois.edu/matrix-rank/sample_code.html

What about RPCA for very dynamic background?

http://changedetection.net/
http://www.svcl.ucsd.edu/projects/background_subtraction/demo.htm

Decomposition into Additive Matrices

- The decomposition is represented in a general formulation:

$$A = \sum_{k=1}^{K} M_k$$

- where **K** usually is equal to 1, 2, or 3. For **K** = 3, $\mathbf{M_1} \dots \mathbf{M_3}$ are commonly defined by:

$$A = M_1 + M_2 + M_3 = L + S + E$$

- The characteristics of the matrices $\mathbf{M_K}$ are as follows:
  - The first matrix $\mathbf{M_1 = L}$ is the low-rank component.
  - The second matrix $\mathbf{M_2 = S}$ is the sparse component.
  - The third matrix $\mathbf{M_3 = E}$ is generally the noise component.
- When **K = 1**, the matrix **A ≈ L** and **S** (implicit) can be given by **S = A – L**. e.g.: LRA, MC, NMF, ...
- When **K = 2**, **A = L + S**. This decomposition is called explicit. e.g.: RPCA, LRR, RNMF, …
- When **K = 3**, **A = L + S + E**. This decomposition is called stable. e.g.: Stable RPCA / Stable PCP.

Bouwmans, Thierry; Sobral, Andrews; Javed, Sajid; Ki Jung, Soon; Zahzah, El-Hadi. "Decomposition into Low-rank plus Additive Matrices for Background/Foreground Separation: A Review for a Comparative Evaluation with a Large-Scale Dataset". Submitted to Computer Science Review, 2015.

Stable PCP

- The PCP is limited, the low-rank component needs to be exactly low-rank and the sparse component needs to be exactly sparse, but in real applications the observations are often corrupted by noise.

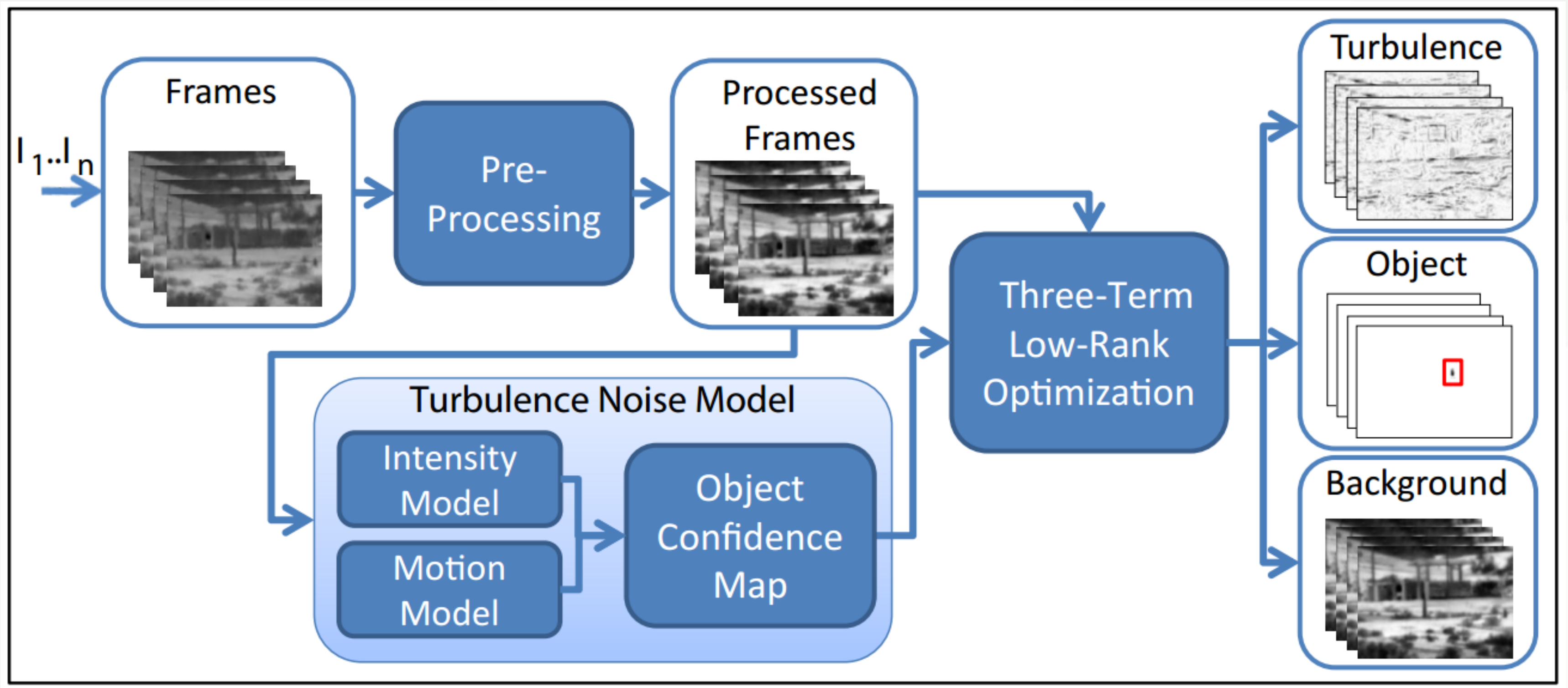
- Zhou et al. (2010) proposed a stable version of PCP, named Stable PCP (SPCP), adding a third component that guarantee stable and accurate recovery in the presence of entry-wise noise. The observation matrix **A** is represented as **A = L + S + E**, where **E** is a noise term.

$$\begin{aligned} \text{minimize} \quad & ||L||_* + \lambda_1 ||S||_1 + \lambda_2 ||E||_F^2 \\ \text{subject to} \quad & A = L + S + E \end{aligned}$$

Constrained RPCA (example 1)

- Some authors added an additional constraint to improve the background/foreground separation:
  - Oreifej et al. (2013) use a turbulance model that quantify the scene's motion in terms of the motion of the particles which are driven by dense optical flow.

$$\begin{array}{ll} \text{minimize} & ||L||_* + \lambda_1 ||\Pi(S)||_1 + \lambda_2 ||E||_F^2 \\ \text{subject to} & A = L + S + E, \text{ where } \Pi(.) \text{ represents the confidence map.} \end{array}$$

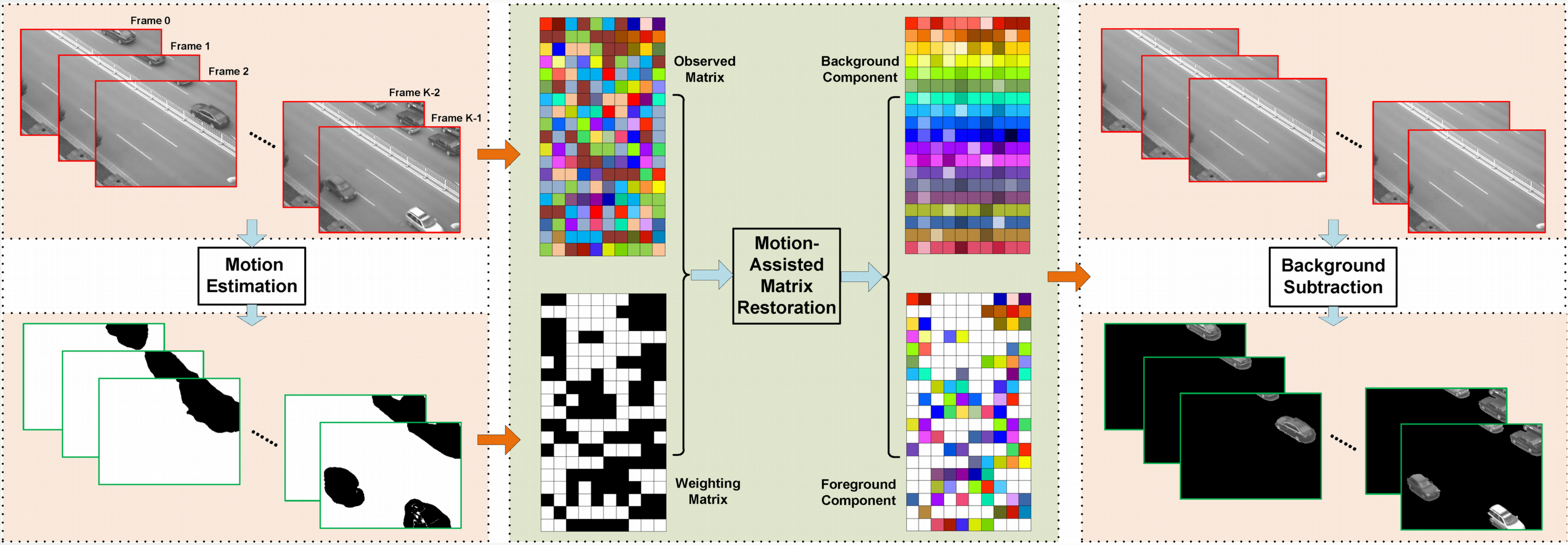


http://www.cs.ucf.edu/~oreifej/papers/3-Way-Decomposition.pdf

Constrained RPCA (example 2)

- Yang et al. (2015) propose a robust motion-assisted matrix restoration (RMAMR) where a dense motion field is estimated for each frame by dense optical flow, and mapped into a weighting matrix which indicates the likelihood that each pixel belongs to the background.

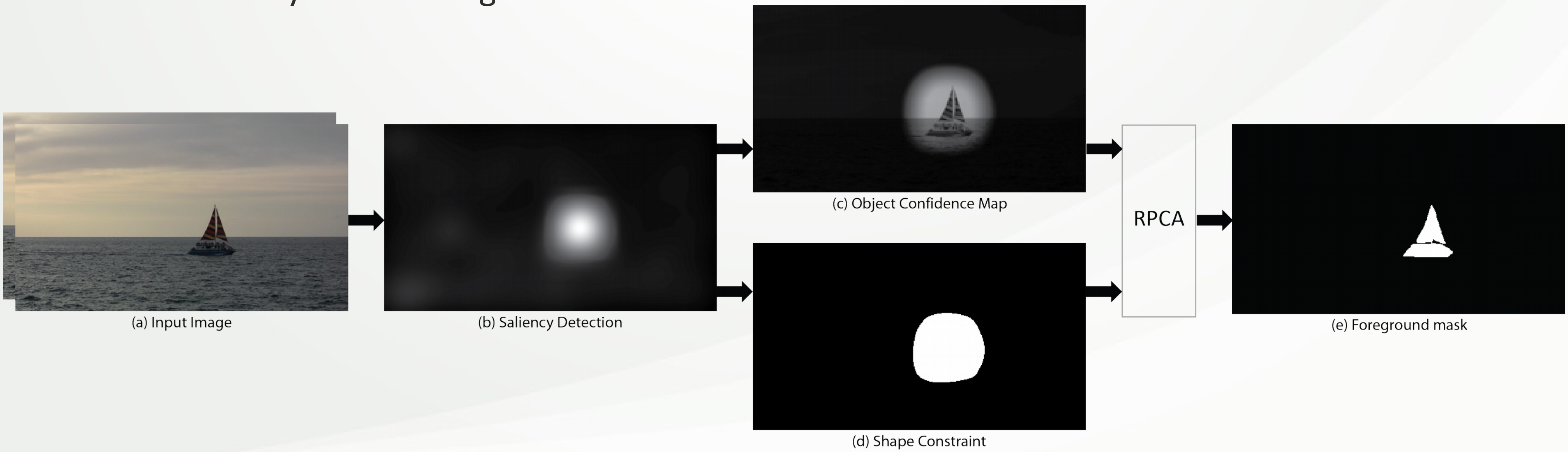
$$\begin{aligned} &\text{minimize} && ||L||_* + \lambda_1 ||S||_1 + \lambda_2 ||E||_F^2 \\ &\text{subject to} && W \circ A = W \circ (L + S + E), \text{ where } W \text{ represents the weighting matrix.} \end{aligned}$$



http://projects.medialab-tju.org/bf_separation/

Double-constrained RPCA?

- Sobral et al. (2015) propose a double-constrained Robust Principal Component Analysis (RPCA), named SCM-RPCA (Shape and Confidence Map-based RPCA), is proposed to improve the object foreground detection in maritime scenes. It combine some ideas of Oreifej et al. (2013) and Yang et al. (2015).
    - The weighting matrix proposed by Yang et al. (2015) can be used as a shape constraint (or region constraint), while the confidence map proposed by Oreifej et al. (2013) reinforces the pixels belonging from the moving objects.
- The original 3WD was modified adding the shape constraint as has been done in the RMAMR. We chose to modify the 3WD instead of RMAMR due its capacity to deal more robustly with the multimodality of the background.



https://sites.google.com/site/scmrpca/

Solving the SCM-RPCA

Author(s)	Minimization
Oreifej et al. [12]	$\min_{L,S,E} \|\|L\|\|_* + \lambda \|\|\Pi(S)\|\|_1 + \gamma \|\|E\|\|_F^2$ s.t. $A = L + S + E$
Yang et al. [15]	$\min_{L,S,E} \|\|L\|\|_* + \lambda \|\|S\|\|_1 + \gamma \|\|E\|\|_F^2$ s.t. $W \circ A = W \circ (L + S + E)$
SCM-RPCA	$\min_{L,S,E} \|\|L\|\|_* + \lambda \|\|\Pi(S)\|\|_1 + \gamma \|\|E\|\|_F^2$ s.t. $A = L + W \circ S + E$

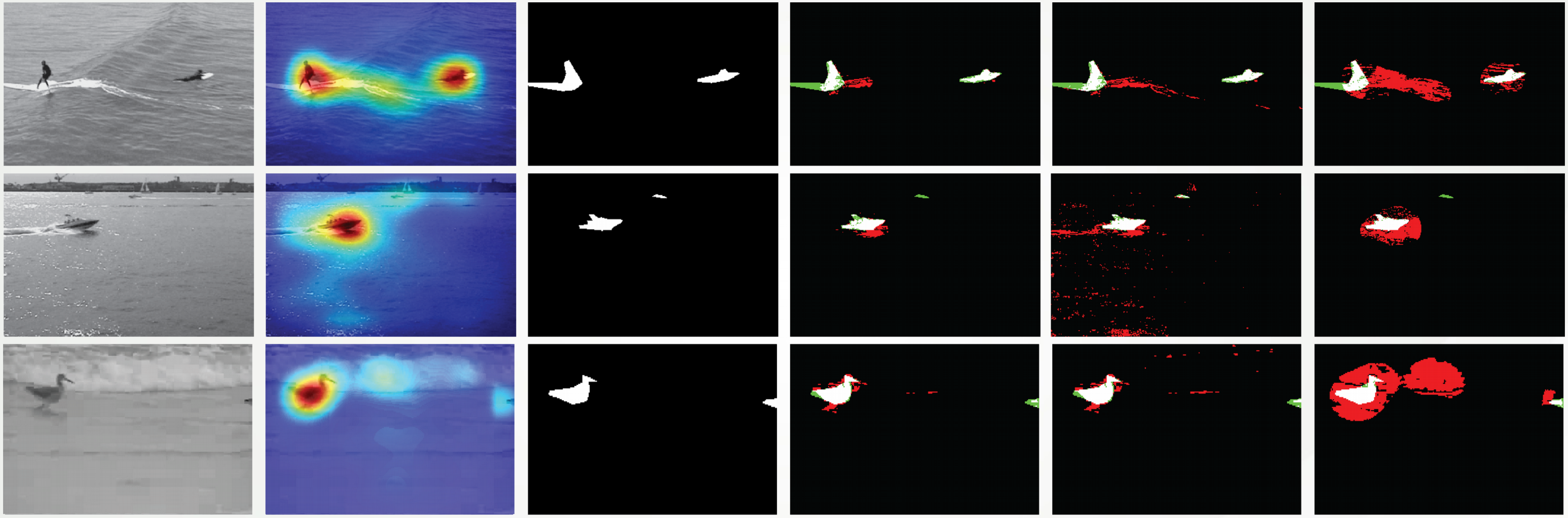
Table 1. Comparison of the proposed method and related works.

Algorithm 1 Algorithm for solving SCM-RPCA.

Input:
Observation $A \in \mathbb{R}^{mn \times k}$
Confidence Map $\Pi \in \mathbb{R}^{mn \times k}$
Shape Constraint $W \in [0, 1]^{mn \times k}$
Output:
Background $L \in \mathbb{R}^{mn \times k}$
Foreground $S \in \mathbb{R}^{mn \times k}$
Noise $E \in \mathbb{R}^{mn \times k}$
while not converged *do*
$\Upsilon = \beta_t^{-1} Y_t$
$URV^T = svd(A - L_t - E_t + \Upsilon)$
$L_{t+1} = U s_{(1/\beta_t)}(R) V^T$
$S_{t+1} = W \circ s_{(\lambda/\beta_t \Pi)}(A - L_{t+1} - E_t + \Upsilon)$
$\kappa = (1 + \frac{2\gamma}{\beta_t})^{-1}$
$E_{t+1} = \kappa(A - L_{t+1} - S_{t+1} + \Upsilon)$
$Z = A_{t+1} - L_{t+1} - S_{t+1} - E_{t+1}$
$Y_{t+1} = Y_t + \beta_t Z$
$\beta_{t+1} = \rho \beta_t$
$t = t + 1$
end

Is important to note that the double constraints (confidence map and shape) can be built from two different types of source (i.e. from spatial, temporal, or spatio-temporal information), but in this work we focus only on spatial saliency maps.

SCM-RPCA - Visual results on UCSD data set



(a)	(b)	(c)	(d)	(e)	(f)

From left to right: (a) input frame, (b) saliency map generated by BMS, (c) ground truth, (d) proposed approach, (e) 3WD, and (f) RMAMR.

[Dataset: http://www.svcl.ucsd.edu/projects/background_subtraction/ucsdbgsub_dataset.htm](http://www.svcl.ucsd.edu/projects/background_subtraction/ucsdbgsub_dataset.htm)

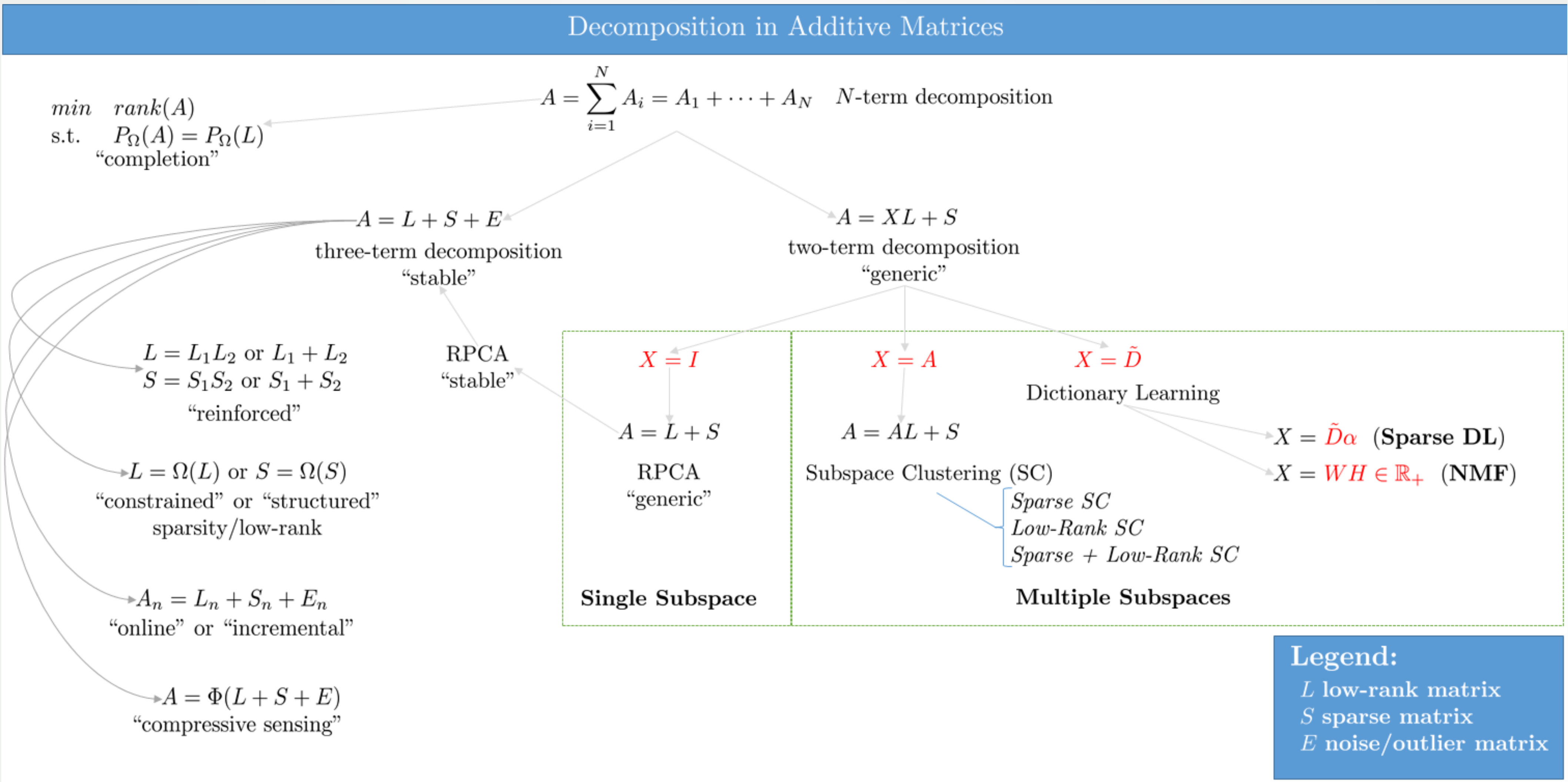
SCM-RPCA - Visual results on MarDT data set

(a) (b) (c) (d) (e) (f)

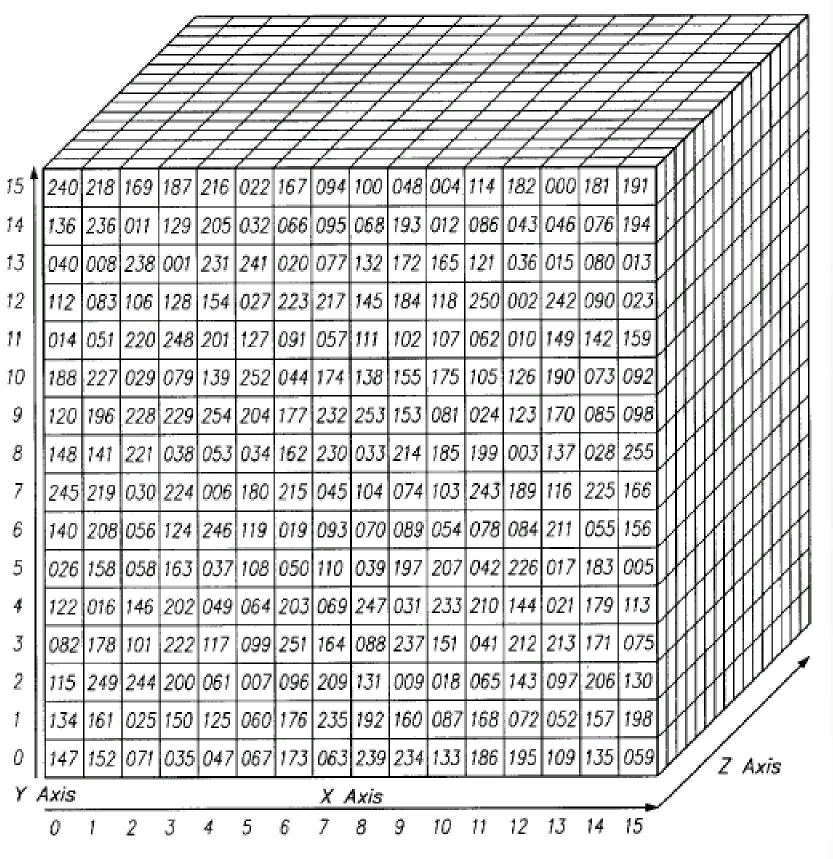
Is important to note that in the UCSD scenes we have used the original spatial saliency map provided by BMS, while for the MarDT scenes we have subtracted its temporal median due to the high saliency from the buildings around the river.

Dataset:
http://www.dis.uniroma1.it/~labrococo/MAR/index.htm

Infinity and beyond



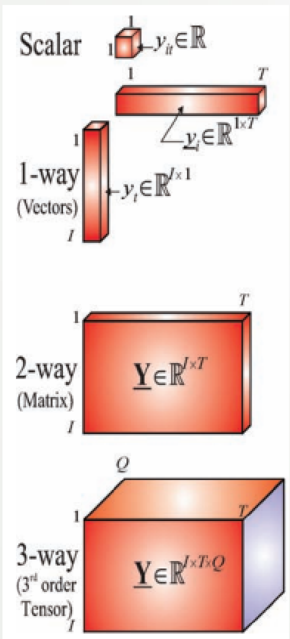
What about multidimensional data?

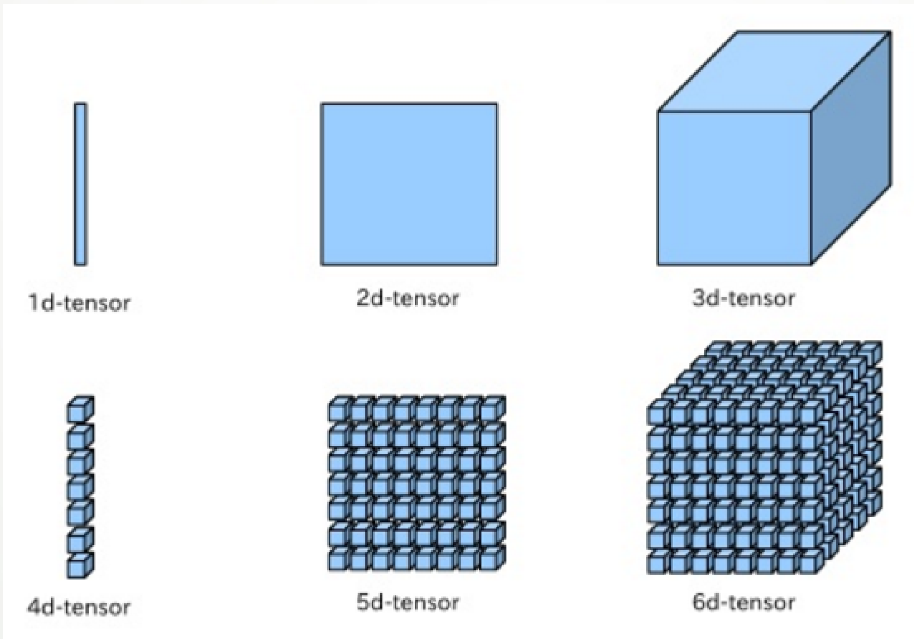


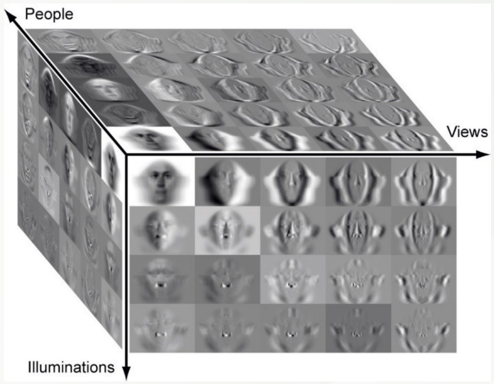
Introduction to tensors

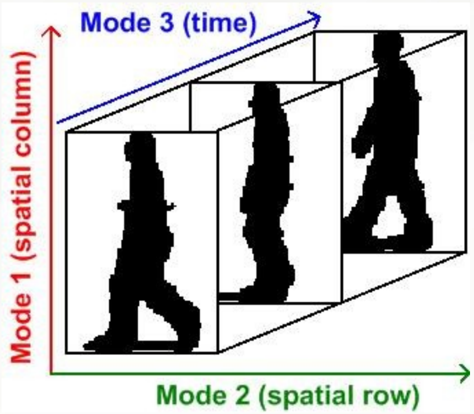
Introduction to tensors

- Tensors are simply mathematical objects that can be used to describe physical properties. In fact tensors are merely a generalization of scalars, vectors and matrices; a scalar is a zero rank tensor, a vector is a first rank tensor and a matrix is the second rank tensor.



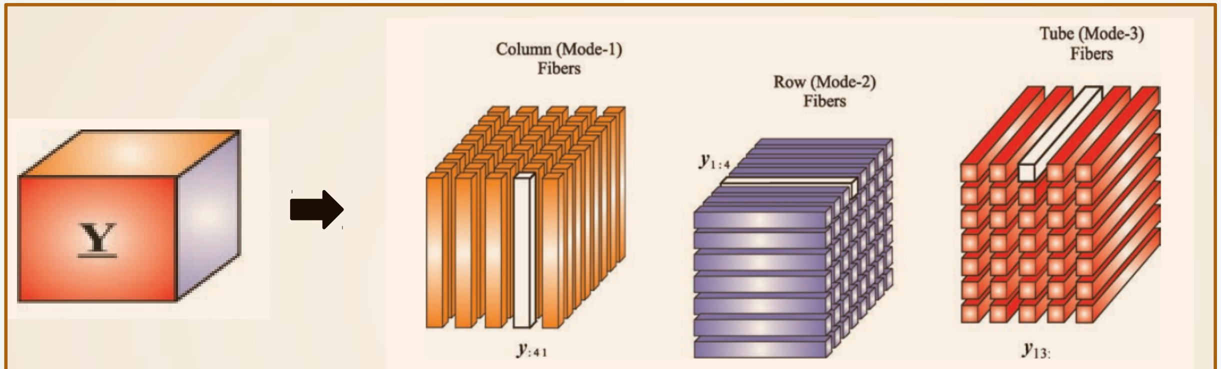






Introduction to tensors

- **Subarrays**, **tubes** and **slices** of a 3rd order tensor.



Fibers: for a third-order tensor $\underline{\mathbf{Y}} = [y_{itq}] \in \mathbb{R}^{I\times T\times Q}$ (all fibers are treated as column vectors).

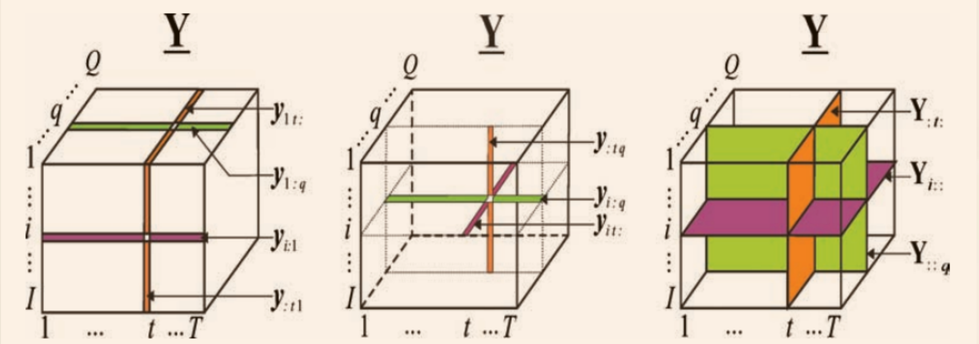
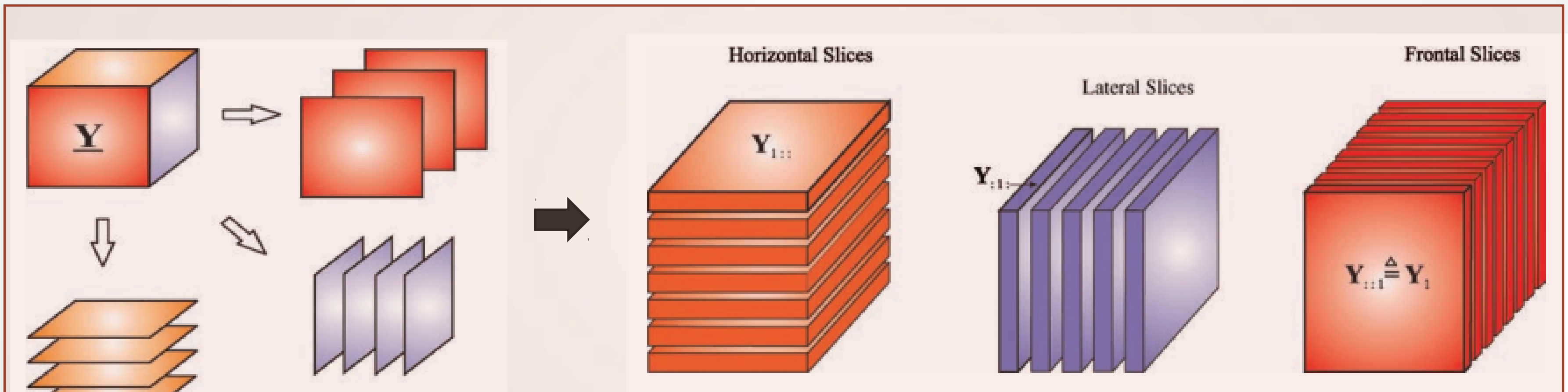


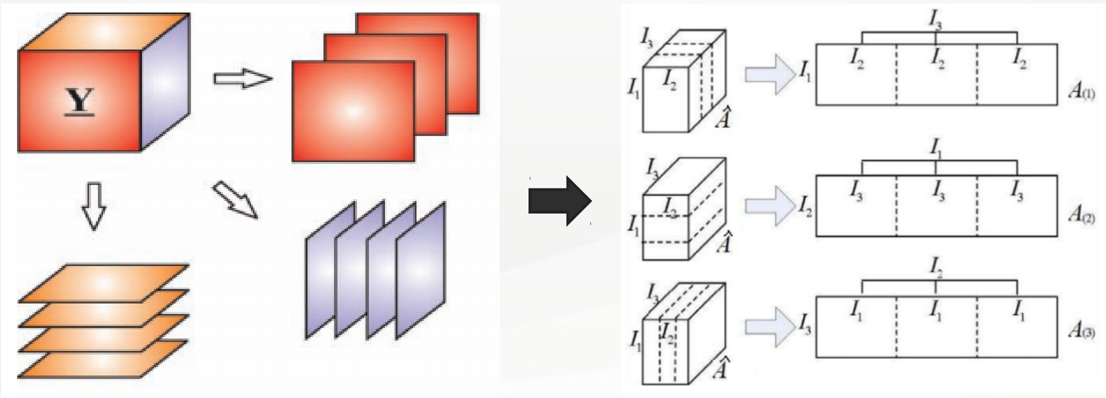
Illustration of subsets (subarrays) of a three-way tensor and basic tensor notations of tubes and slices.

Introduction to tensors

- **Matricization** and **unfolding** a 3rd order tensor.

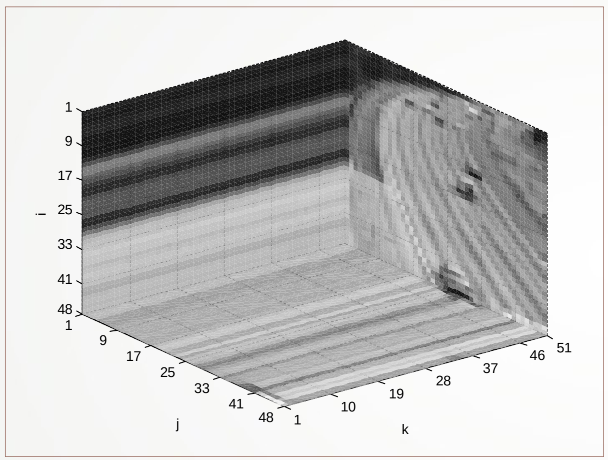


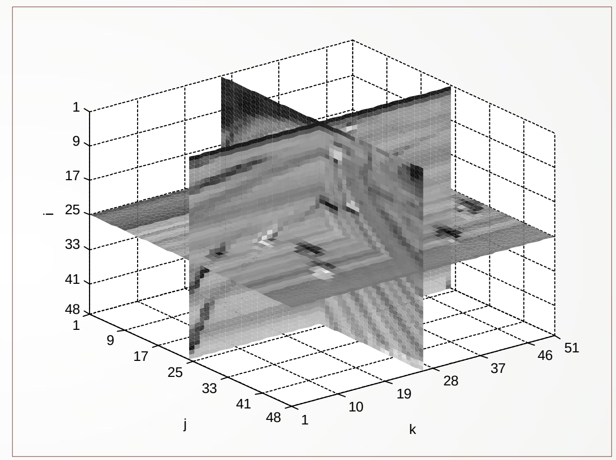
Slices for a third-order tensor $\underline{\mathbf{Y}} = [y_{itq}] \in \mathbb{R}^{I \times T \times Q}$.

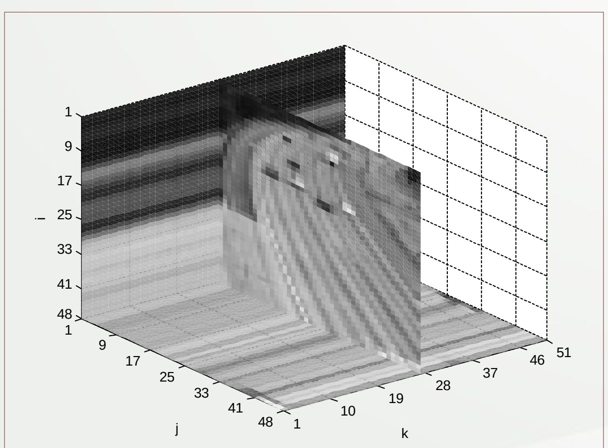


Introduction to tensors

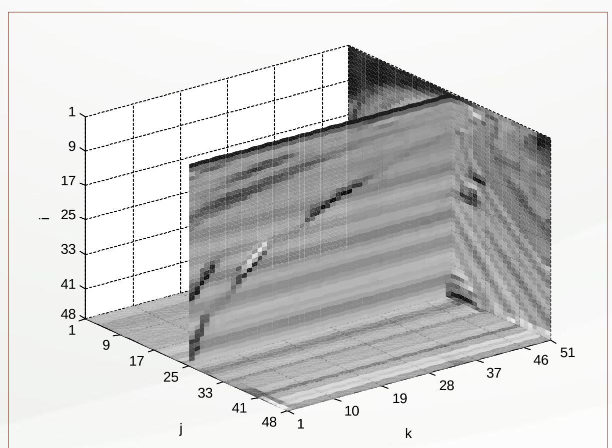
- **Horizontal**, **vertical** and **frontal** slices from a 3rd order tensor.



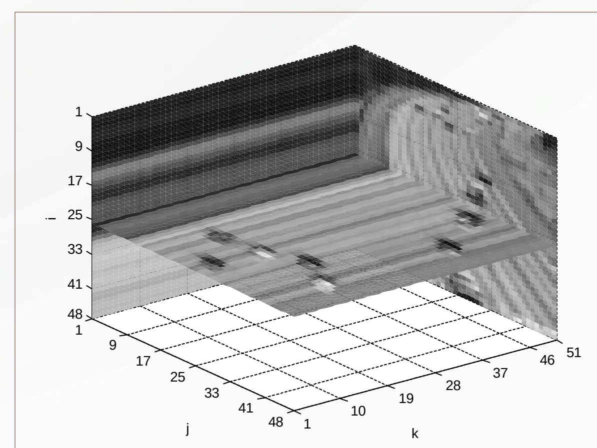




Frontal



Vertical

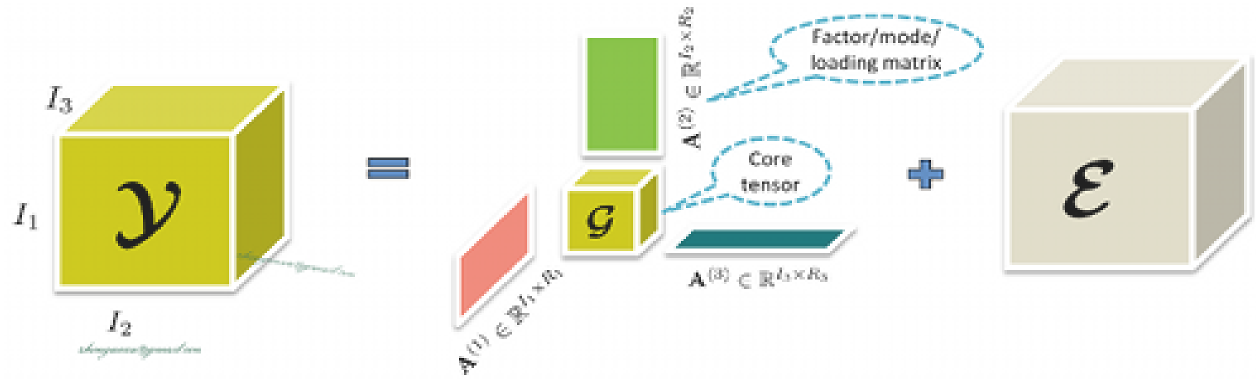


Horizontal

Tensor decomposition methods

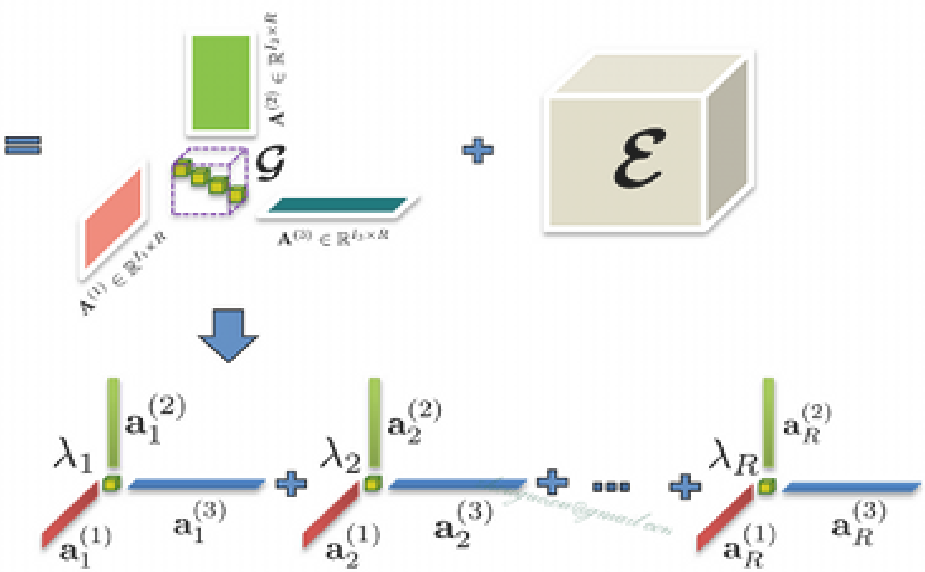
- Approaches:
  - Tucker / HOSVD
  - CANDECOMP-PARAFAC (CP)
  - Hierarchical Tucker (HT)
  - Tensor-Train decomposition (TT)
  - NTF (Non-negative Tensor Factorization)
  - NTD (Non-negative Tucker Decomposition)
  - NCP (Non-negative CP Decomposition)

Tucker Decomposition



$$\mathcal{Y} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} + \mathcal{E}$$

CP Decomposition



- $R_1 = R_2 = R_3 = R$.
- $\mathcal{G}$ is super diagonal.

$$\mathcal{Y} = \sum_r \lambda_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \mathbf{a}_r^{(3)} + \mathcal{E}$$

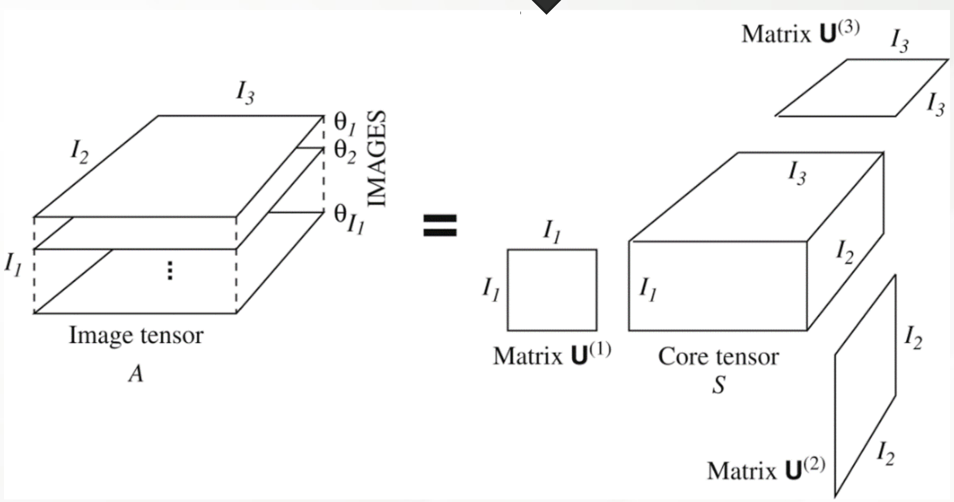
Tucker / HoSVD

The Higher Order Singular Value Decomposition (HOSVD)

The HOSVD of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ involves computing the matrix SVDs of its modal unfoldings $\mathcal{A}_{(1)}, \ldots, \mathcal{A}_{(d)}$. This results in a representation of $\mathcal{A}$ as a sum of rank-1 tensors.

```
function [S,U] = HOSVD(A)
% A is an n(1)-by-...-by-n(d) tensor.
% U is a length-d cell array with the
%   property that U{k} is the left singular
%   vector matrix of A's mode-k unfolding.
% S is an n(1)-by-...-by-n(d) tensor given by
%    A x1 U{1} x2 U{2} ... xd U{d}

S = A;
for k=1:length(A.size)
    C = tenmat(A,k);
    [U{k},Sigma,V] = svd(C.data);
    S = ttm(S,U{k}',k);
end
```

$$\mathcal{A} = \mathcal{S} \times_{n=1}^{N} U^{(n)}$$



CP

- The CP model is a special case of the Tucker model, where the core tensor is superdiagonal and the number of components in the factor matrices is the same.

Solving by ALS (alternating least squares) framework

```
n = [ 5 6 7 ]; rmax = 35;
% Generate a random tensor...
A = tenrand(n);
for r = 1:rmax
   % Find the closest length-r ktensor...
   X = cp_als(A,r);
   % Display the fit...
   E = double(X)-double(A);
   fit = norm(reshape(E,prod(n),1));
   fprintf('r = %1d, fit = %5.3e\n',r,fit);
end
```

The CP Approximation Problem

Given $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and r, determine $\lambda \in \mathbb{R}^r$, $F \in \mathbb{R}^{n_1 \times r}$, $G \in \mathbb{R}^{n_2 \times r}$, and $H \in \mathbb{R}^{n_3 \times r}$ so that

$$\mathcal{A} \approx [[\, \lambda;\ F,\ G,\ H \,]] = \mathcal{X}.$$

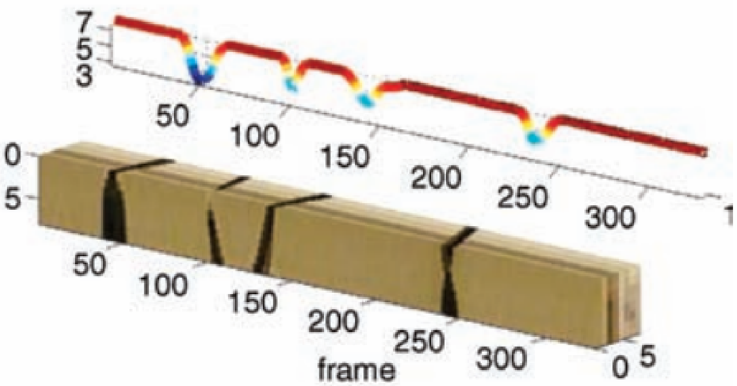
What About r?

In the CP approximation problem we have assumed that r, the length of the approximating ktensor, is given:

$$\mathcal{A} \approx \mathcal{X} = \sum_{j=1}^{r} \lambda_j U_1(:,j) \circ \cdots \circ U_d(:,j)$$

We can think of $\mathcal{X}$ as a rank-r approximation to $\mathcal{A}$.

Background Model Estimation via Tensor Factorization



(b) A 8 × 8 × 3 × 330 sub-tensor at pixel (49,65)

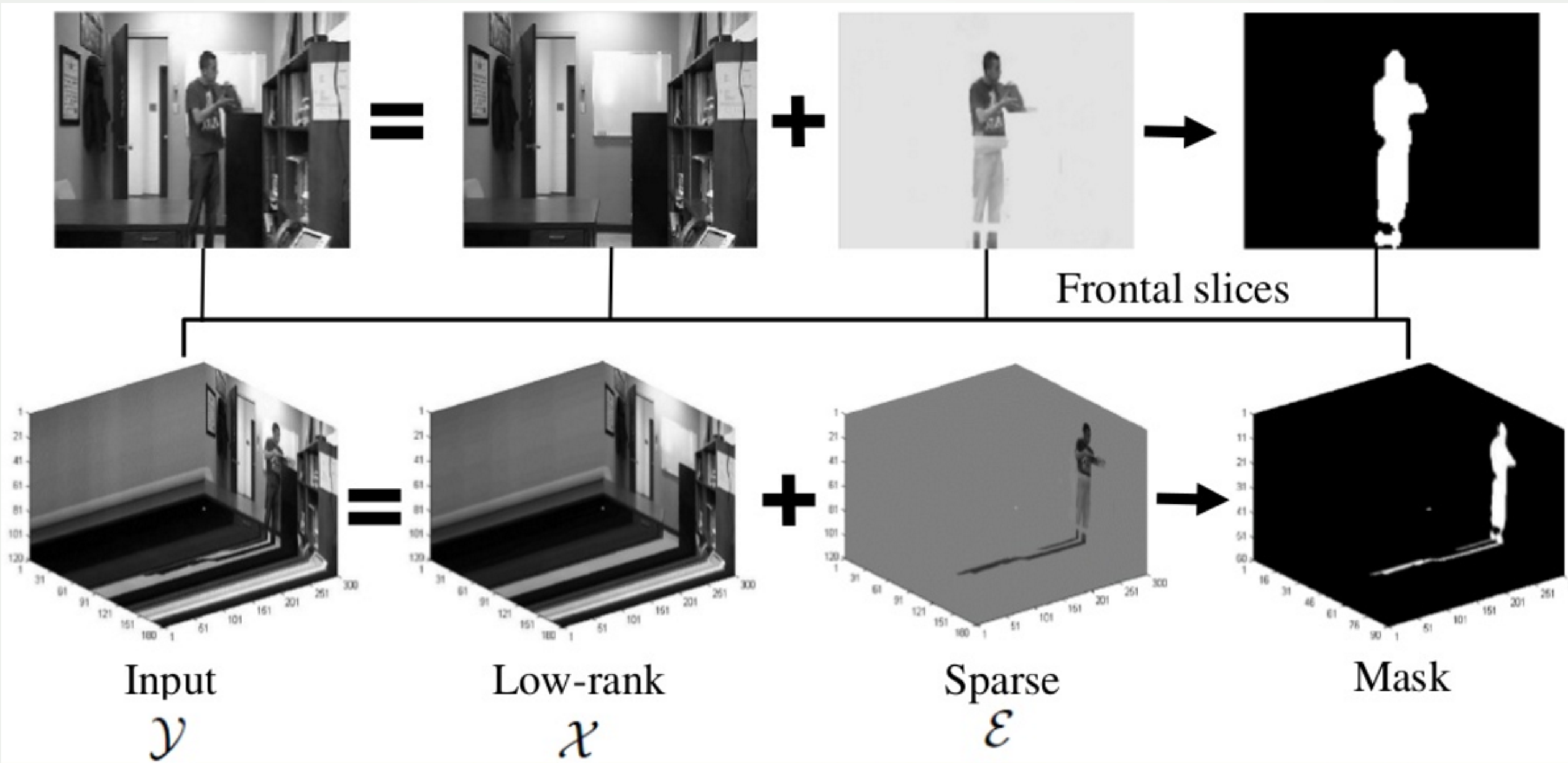
(a) Background of the Hall Monitor sequence

Listing 7.13 Background estimation for the `HallMonitor` sequence.

```
% Copyright 2008 by Anh Huy Phan and Andrzej Cichocki
% Load Hall sequence
clear
load Hall4Dtensor;
%% Process full block
sT = size(T);
blksize = 8;
d = zeros(sT(4),prod(sT(1:2))/blksize^2);
kblk = 0;
xy = zeros(1,2);
for xu = 1:8:sT(1)-7
    for yu = 1:8:sT(2)-7
        kblk = kblk + 1;
        Tblk  = T(xu:xu+blksize-1, yu:yu+blksize-1,:,:);

        %% Factorize subtensor with Parafac algorithms R = 1
        Yblk = permute(tensor(Tblk),[4 1 2 3]);
        options = struct('verbose',1,'tol',1e-6,'maxiters',500,...
            'init',2,'nonlinearproj',1);
        [X_hals,Uinit,A_,ldam,iter] = parafac_hals(Yblk,1,options);
        d(:,kblk) = double(A_{1});
        xy(kblk,:) = [xu yu];
    end
end

%% Find stationary blocks and build background image
maxd = max(d); mind = min(d);
thresh = 0.005;
Imbgr = zeros(sT(1:3));
for k = 1:size(d,2);
    edges = [mind(k):thresh:maxd(k) maxd(k)+eps] ;
    [n,bin] = histc(d(:,k),edges);
    m = mode(bin);
    indbgr = find((d(:,k)≥edges(m)) & (d(:,k)≤ edges(m+1)));
    bgrblk  = median(T(xy(k,1):xy(k,1)+blksize-1,...
        xy(k,2):xy(k,2)+blksize-1,:,indbgr),4);
    Imbgr(xy(k,1):xy(k,1)+blksize-1, xy(k,2):xy(k,2)+blksize-1,:) = bgrblk;
end

%% Display the estimated background image
imshow(Imbgr)
```

https://github.com/andrewssobral/mtt/blob/master/tensor_demo_subtensors_ntf_hals.m

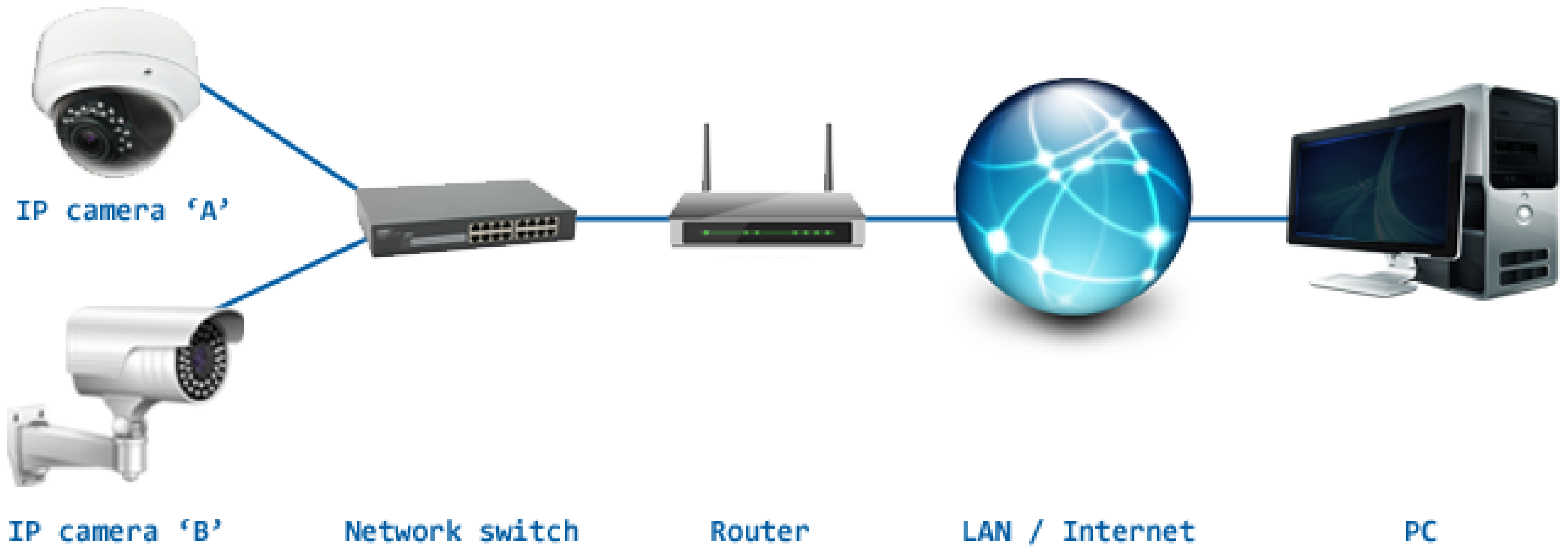
Background Subtraction via Tensor Decomposition



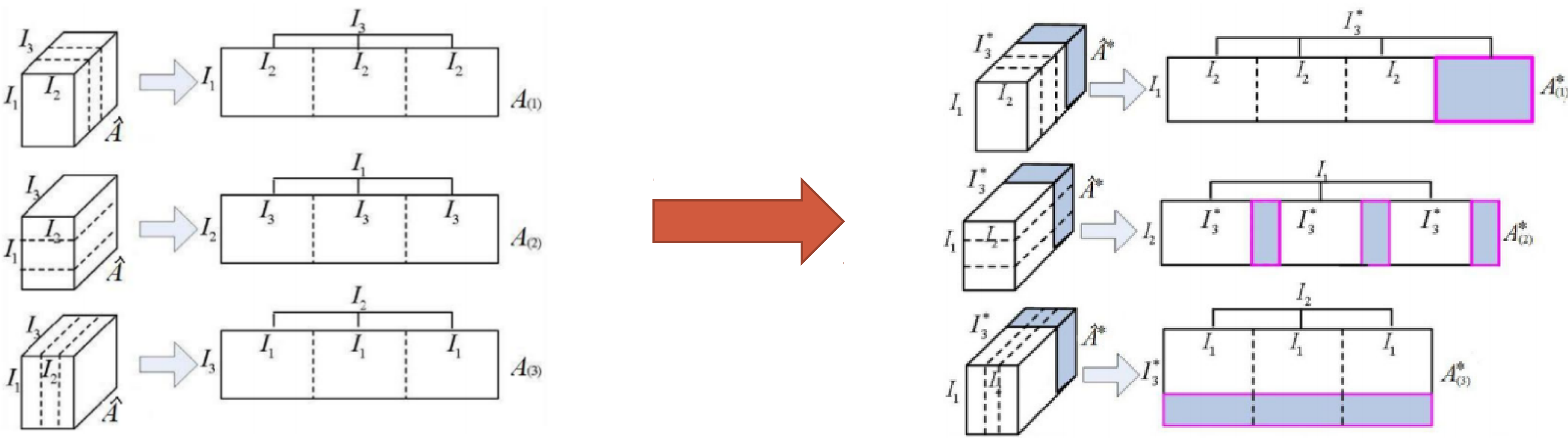
https://github.com/andrewssobral/lrslibrary

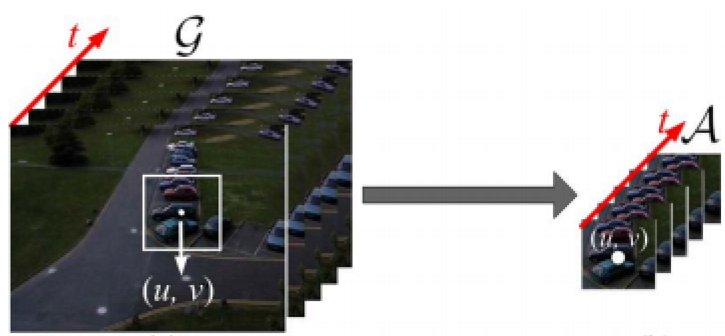
Incremental Tensor Learning

Interested in stream processing?



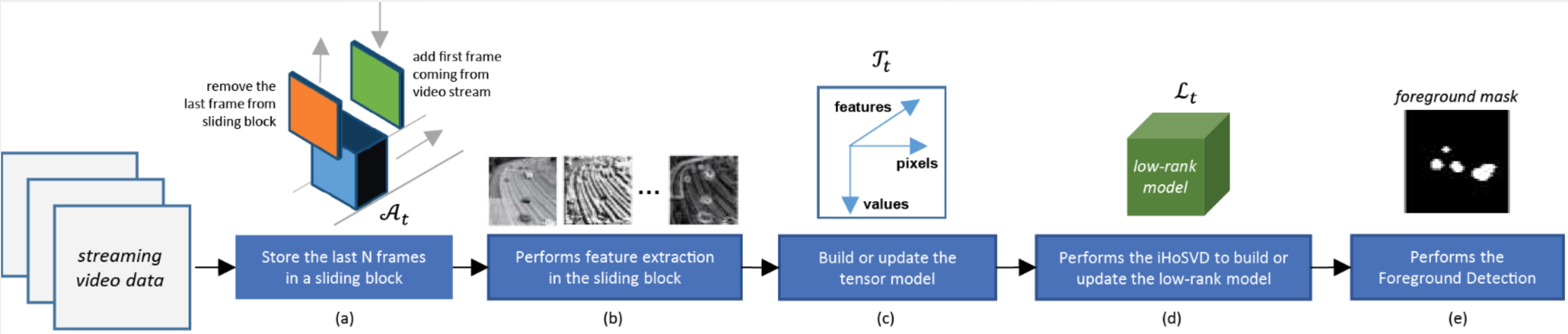
Incremental Tensor Subspace Learing





(a) (b)

Incremental and Multifeature



Algorithm 1 Proposed iHoSVD algorithm.

function INCREMENTALHOSVD($\mathcal{T}_t$, $r^{(n)}$, $t^{(n)}$)
 $\mathcal{S}_t \leftarrow \mathcal{T}_t$
 if $t = 0$ **then** ▷ Performs the standard rank-r SVD
  for $i = 1$ to n **do**
   $[\mathbf{U}_t^{(n)}, \mathbf{\Sigma}_t^{(n)}, \mathbf{V}_t^{(n)}] \leftarrow \text{SVD}(\mathcal{T}_t^{(n)}, r^{(n)}, t^{(n)})$
  end for
 else ▷ Performs the incremental rank-r SVD
  for $i = 1$ to n **do**
   $[\mathbf{U}_t^{(n)}, \mathbf{\Sigma}_t^{(n)}, \mathbf{V}_t^{(n)}] \leftarrow \text{iSVD}(\mathcal{T}_t^{(n)}, r^{(n)}, t^{(n)}, \mathbf{U}_{t-1}^{(n)}, \mathbf{\Sigma}_{t-1}^{(n)}, \mathbf{V}_{t-1}^{(n)})$
  end for
 end if
 $\mathcal{S}_t \leftarrow \mathcal{T}_t \times_1 (\mathbf{U}_t^{(1)})^T \ldots \times_n (\mathbf{U}_t^{(n)})^T$ ▷ $\times_n$ denotes the n-mode tensor times matrix
 return $\mathcal{S}_t$, $\mathbf{U}_t^{(1)}$, ..., $\mathbf{U}_t^{(n)}$
end function

https://github.com/andrewssobral/imtsl

Incremental and Multifeature

A total of 8 features are extracted:

1) red channel,

2) green channel,

3) blue channel,

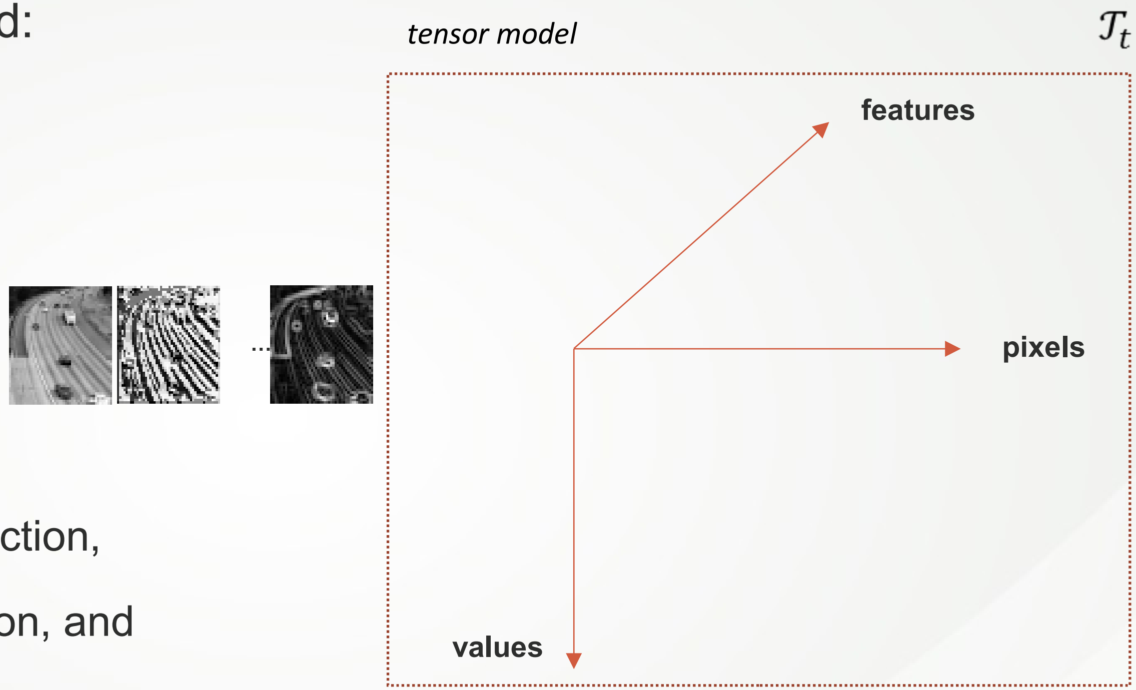
4) gray-scale,

5) local binary patterns (LBP),

6) spatial gradients in horizontal direction,

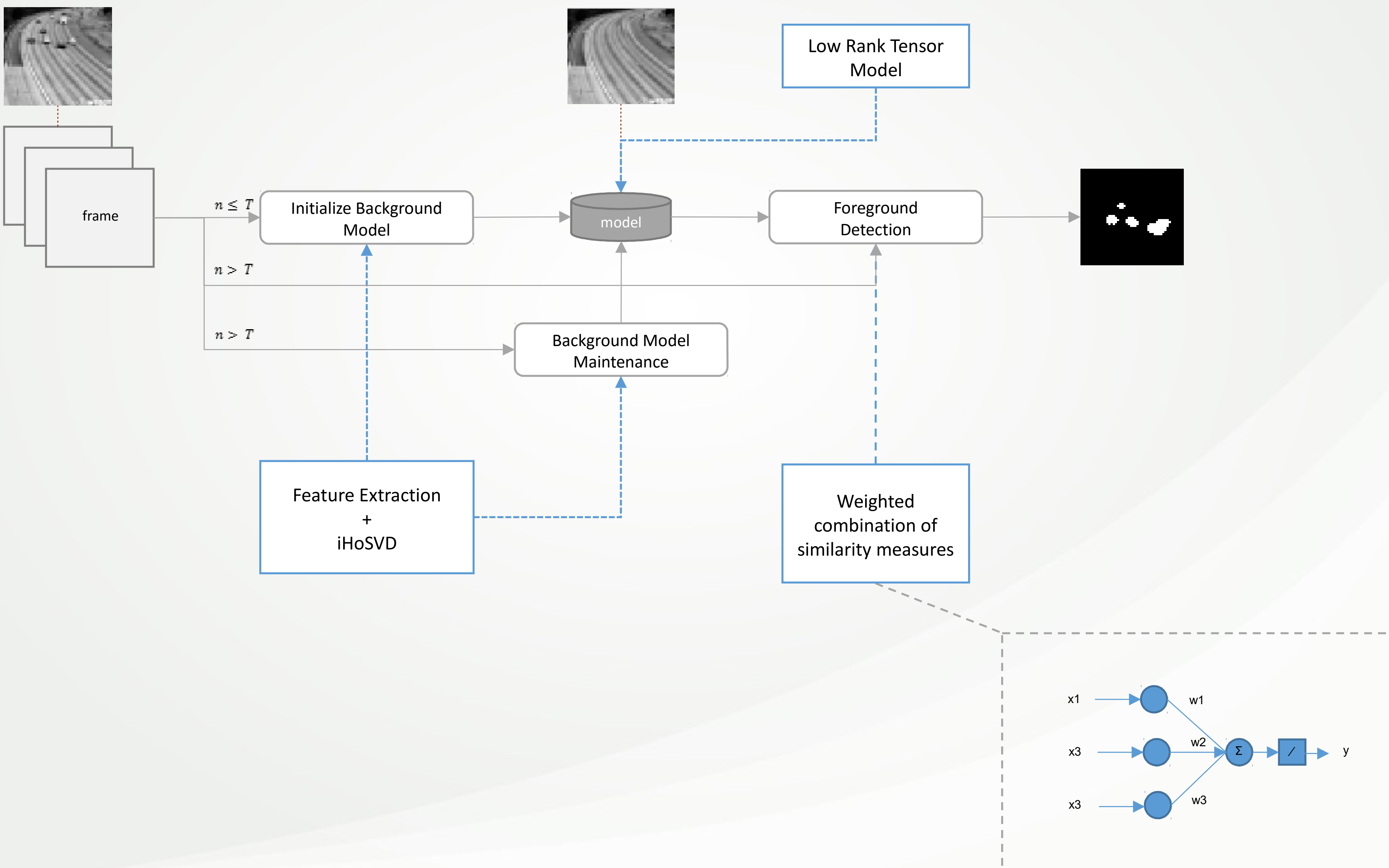
7) spatial gradients in vertical direction, and

8) spatial gradients magnitude.



$$\mathcal{T}_t \in \mathbb{R}^{19200 \times 25 \times 8}$$

Incremental and Multifeature



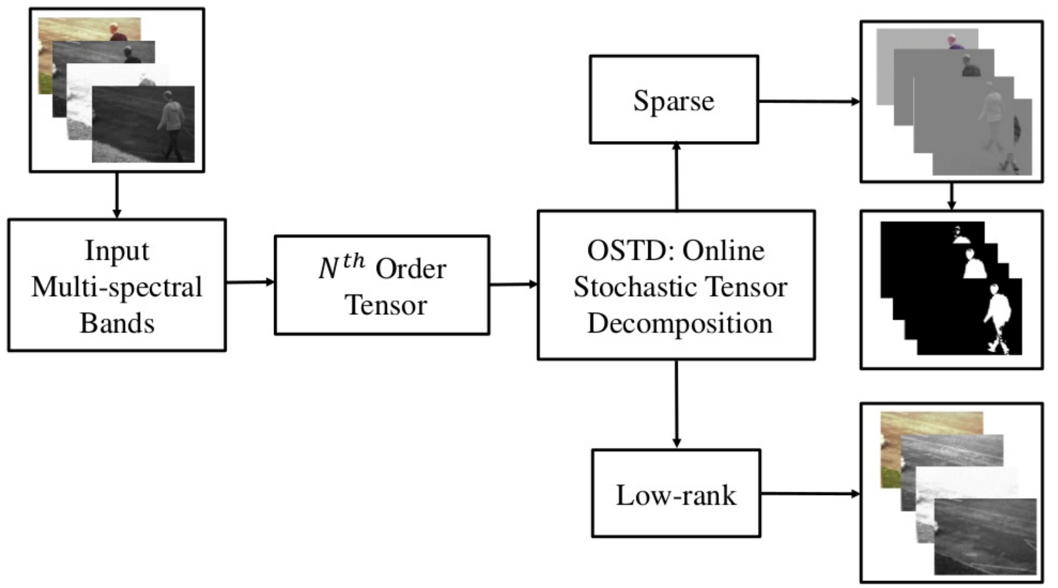
https://github.com/andrewssobral/imtsl

Online Stochastic

- Let say N^{th} order observation tensor $\mathcal{Y}$
  - corrupted by outliers, $\mathcal{E}$
- Main assumption
  - $\mathcal{Y}$ can be reconstructed by the combination of
    - low-rank component, $\mathcal{X}$
    - sparse component, $\mathcal{E}$
  - convex optimization framework

$$\min_{\mathcal{X},\mathcal{E}} \frac{1}{2}\sum_{i=1}^{N}||\mathcal{Y}_i - \mathcal{X}_i - \mathcal{E}_i||_F^2 + \lambda_1||\mathcal{X}_i||_* + \lambda_2||\mathcal{E}_i||_1,$$

    - $||\mathcal{X}_i||_*$ represents the nuclear norm of i^{th} mode
    - $||\mathcal{E}_i||_1$ represents the l_1 norm
- Stochastic/Online optimization proposed by [**Feng** *et.al* 2013]



https://github.com/andrewssobral/ostd

OSTD cont…

- Advantages
  - no batch processing
  - iteratively update the basis
  - used for each i^{th} mode
- Major Processing: 3 Steps
  - **Low-rank** approximation
    - Initialize the basis, L
      - Bilateral Random Projections (BRP) method

$$L = Y_1 (A_1^T Y_1)^{-1} Y_2^T$$

        - L, Y, A are all random matrices
        - speed-up low-rank recovery: fast convergence
          - SVD decay slowly

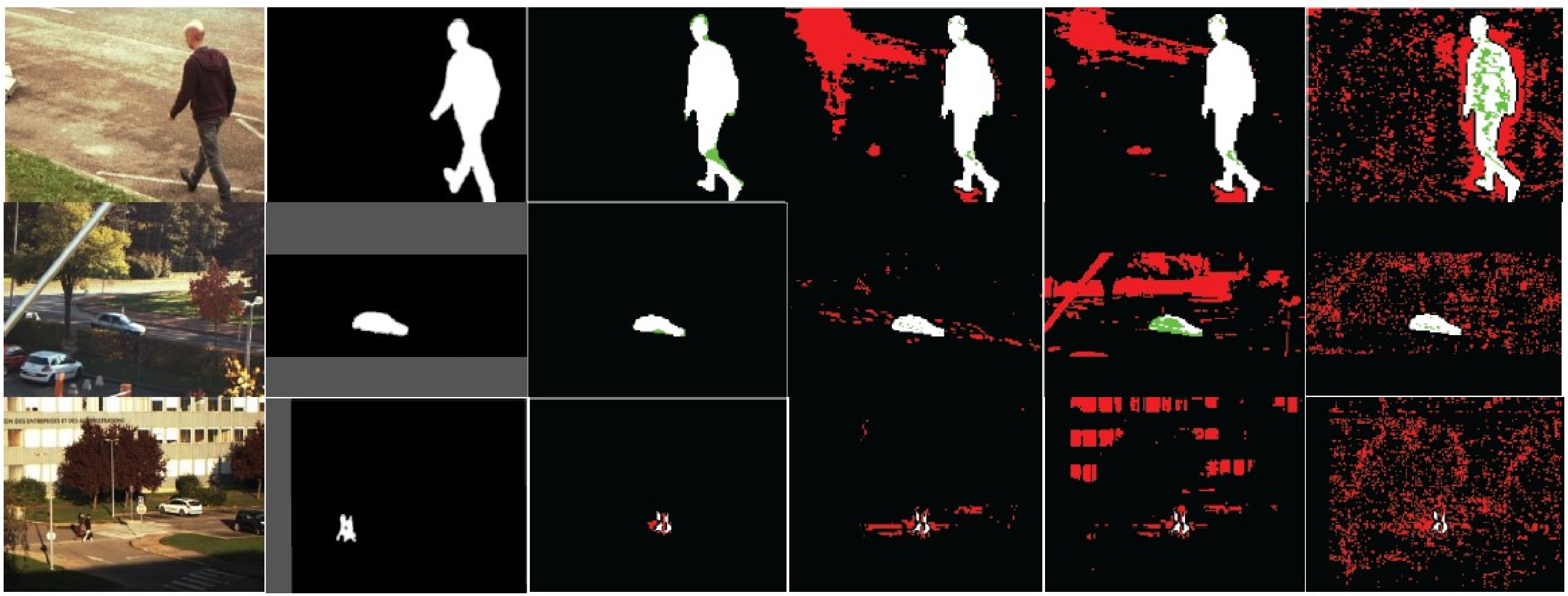
Experimental Evaluations

- Qualitative Comparison
  - White: True positive (TP) pixels
  - Black: True negatives (TN) pixels
  - Red: False positives (FP) pixels
  - Green: False negatives (FN) pixels

Video 2

Video 3

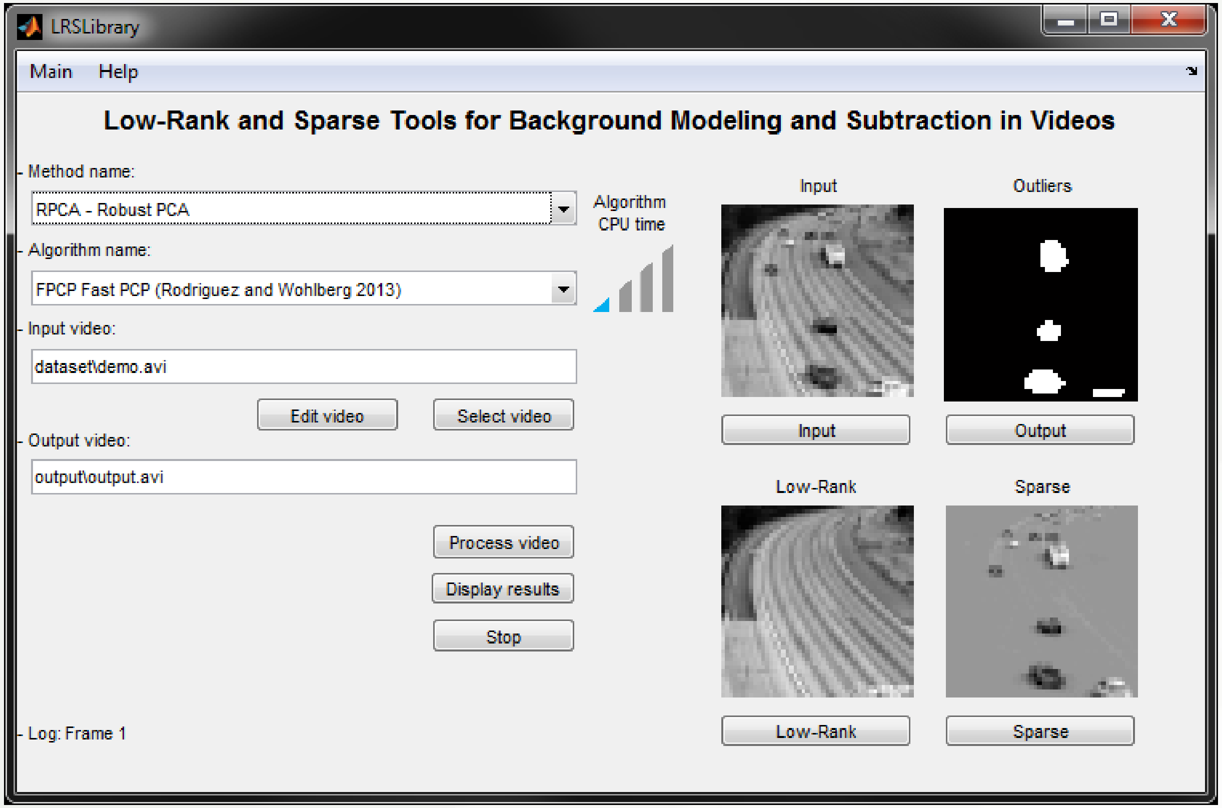
Video 5



Input | Ground Truth | Proposed | BRTF | HORPCA | CP-ALS

LRSLibrary

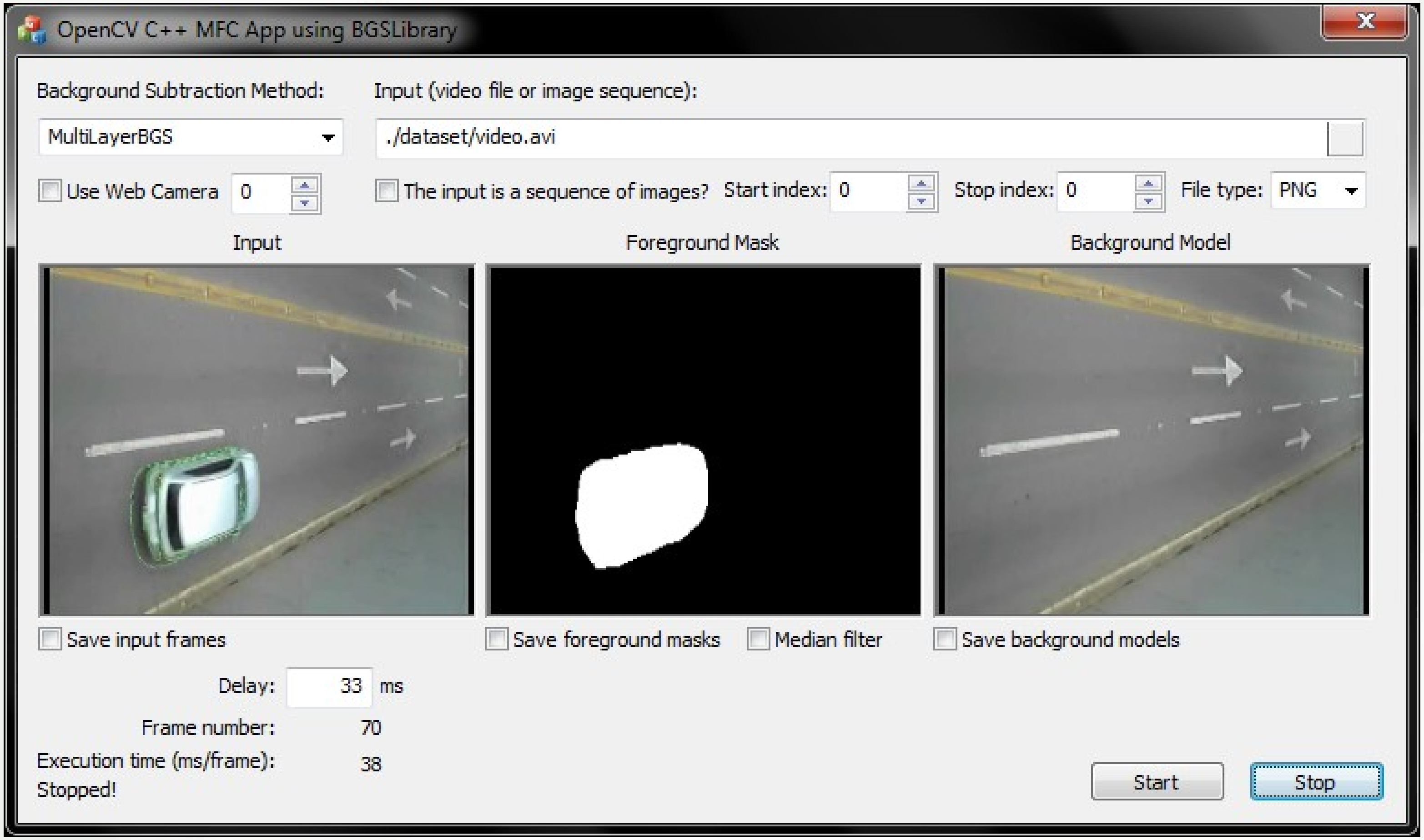
The LRSLibrary provides a collection of low-rank and sparse decomposition algorithms in MATLAB. The library was designed for motion segmentation in videos, but it can be also used or adapted for other computer vision problems. Currently the LRSLibrary contains a total of 103 matrix-based and tensor-based algorithms.



https://github.com/andrewssobral/lrslibrary

BGSLibrary

The BGSLibrary provides an easy-to-use C++ framework based on OpenCV to perform background subtraction (BGS) in videos. The BGSLibrary compiles under Linux, Mac OS X and Windows. Currently the library offers **37** BGS algorithms.



The End