SSA for textured images

Konstantin Usevich

Saint Petersburg State University
Mathematical Department

UK-China workshop on SSA and its applications
September 20-22, 2010, Cardiff
SSA for 2D images: algorithm

Input: image (matrix)

\[ F = \begin{pmatrix} f_{1,1} & \cdots & f_{N_x,1} \\ \vdots & \ddots & \vdots \\ f_{1,N_y} & \cdots & f_{N_x,N_y} \end{pmatrix} \]

Parameters: window sizes \((L_x, L_y)\)

\[
\begin{align*}
K_x &= N_x - L_x + 1, \\
K_y &= N_y - L_y + 1, \\
L &= L_x L_y
\end{align*}
\]

Vectorization

Trajectory matrix

\[
X = \left( \begin{array}{c}
\text{vec } F_{1,1} \\
\vdots \\
\text{vec } F_{K_x,1} \\
\text{vec } F_{1,K_y} \\
\vdots \\
\text{vec } F_{K_x,K_y}
\end{array} \right)
\]

Covariance matrix

\[
S = XX^T \in \mathbb{R}^{L \times L}, \quad \begin{cases}
\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_L \geq 0 \\
U_1, U_2, \ldots, U_L \in \mathbb{R}^L
\end{cases}
\]

SVD

\[
X = \sum_{i=1}^{d} \sqrt{\lambda_i} U_i V_i^T, \quad V_i = (X^T U_i) / \sqrt{\lambda_i}
\]

Grouping, projection, ...
SSA for 2D images: construction of trajectory matrix

\[
F = \begin{pmatrix}
  f_{1,1} & f_{2,1} & f_{3,1} & f_{4,1} & f_{5,1} \\
  f_{1,2} & f_{2,2} & f_{3,2} & f_{4,2} & f_{5,2} \\
  f_{1,3} & f_{2,3} & f_{3,3} & f_{4,3} & f_{5,3} \\
  f_{1,4} & f_{2,4} & f_{3,4} & f_{4,4} & f_{5,4}
\end{pmatrix}
\]

\((N_x, N_y) = (5, 4), (L_x, L_y) = (3, 2), (K_x, K_y) = (3, 3)\)

Trajectory matrix:

\[
X = \begin{pmatrix}
  f_{1,1} & f_{2,1} & f_{3,1} & f_{1,2} & f_{2,2} & f_{3,2} & f_{1,3} & f_{2,3} & f_{3,3} \\
  f_{2,1} & f_{3,1} & f_{4,1} & f_{2,2} & f_{3,2} & f_{4,2} & f_{2,3} & f_{3,3} & f_{4,3} \\
  f_{3,1} & f_{4,1} & f_{5,1} & f_{3,2} & f_{4,2} & f_{5,2} & f_{3,3} & f_{4,3} & f_{5,3} \\
  f_{1,2} & f_{2,2} & f_{3,2} & f_{1,3} & f_{2,3} & f_{3,3} & f_{1,4} & f_{2,4} & f_{3,4} \\
  f_{2,2} & f_{3,2} & f_{4,2} & f_{2,3} & f_{3,3} & f_{4,3} & f_{2,4} & f_{3,4} & f_{4,4} \\
  f_{3,2} & f_{4,2} & f_{5,2} & f_{3,3} & f_{4,3} & f_{5,3} & f_{3,4} & f_{4,4} & f_{5,4}
\end{pmatrix}
\]

Hankel-block-Hankel matrix

Description of algorithm: (Golyandina, Usevich, 2010, Matrix Methods)
SSA for 2D images: details

**Devectorization**

\[
U_i \in \mathbb{R}^{L_x L_y} \rightarrow \Psi_i \in \mathbb{R}^{L_x \times L_y} \quad \text{eigenimage}
\]

\[
V_i \in \mathbb{R}^{L_x L_y} \rightarrow \Phi_i \in \mathbb{R}^{L_x \times L_y} \quad \text{factor image}
\]

**Convolution**

\[
\Phi_i = (F \ast U_i) / \sqrt{\lambda_i}
\]

\[
\lambda_i = \| F \ast \Psi_i \|_2^2
\]

Source image:

Eigenimages:

\[\begin{align*}
\Psi_1 & \quad \Psi_2 \\
\Psi_3 & \quad \Psi_4 \\
\Psi_5 & \quad \Psi_6 \\
\Psi_7 & \quad \Psi_8
\end{align*}\]

Factor images:

\[\begin{align*}
\Phi_1 & \\
\Phi_2 & \\
\Phi_3 & \quad \Phi_4 \\
\Phi_5 & \\
\Phi_6
\end{align*}\]

Restored by 3,4:

Residual:
SSA for 2D images: real-life images

Typical image

Eigenvalues (20x20 window)

Reconstructed by 1

Residual

W-correlations

Problems:

- Infinite rank, no periodics
- Too complex $\Rightarrow$ just smoothing, details $\rightarrow$ noise
- Too many components $\Rightarrow$ eigenvalues mixing
Real-life and artificial textures

Sinusoidal  Periodic  Distorted  Roof  Cloth

Wood  Pebbles  Grass  Limestone  Marble

Structure: regular (periodic) $\leftrightarrow$ random.

Texture $=$ pattern looking homogeneous (background).

- Uniform ("stationary")
- Repeating small elements
- "Uncountable" number of elements
Real-life and artificial textures

Sinusoidal  Periodic  Distorted  Roof  Cloth

Wood  Pebbles  Grass  Limestone  Marble

Structure: regular (periodic) $\longleftrightarrow$ random.

Texture = pattern looking homogeneous (background).
- Uniform (“stationary”)
- Repeating small elements
- “Uncountable” number of elements
Real-life and artificial textures

Structure: regular (periodic) \(\leftrightarrow\) random.

Texture = pattern looking homogeneous (background).

- Uniform ("stationary")
- Repeating small elements
- "Uncountable" number of elements
(1) Texture classification

New image $\rightarrow$ Reference images

Class 1  Class 2  ...

(2) Texture segmentation

Supervised:

Unsupervised...

(3) Texture detection

(4) Defect detection (industrial applications)

Textiles  Ceramic tiles  Frozen food
(1) Texture classification

New image ? Reference images
Class 1 Class 2 ...

Dimensionality reduction, Introducing a distance

(2) Texture segmentation

Supervised:
Unsupervised...

Take sliding windows Perform classification

(3) Texture detection

(4) Defect detection (industrial applications)

Textiles Ceramic tiles Frozen food
SSA-like methods: eigenfilters
Filtering approach (Laws, 1980) — HVS

Convolution: \( Y_i = F \ast H_i \leftrightarrow (Y_i)_{k,l} = \sum_{s=1,t=1}^{L_x,L_y} f_{k+s-1,l+t-1} h_{s,t} \)

Different textures \( F^{(1)}, F^{(2)} \) — different distribution of responses \( Y^{(1)}, Y^{(2)} \)

Early approaches: compare \( (DY^{(1)}_1, \ldots, DY^{(1)}_m) \) and \( (DY^{(2)}_1, \ldots, DY^{(2)}_m) \).

Recent approaches: histogram comparison (clustering)
Take folded eigenvectors of $S$ as filters (Ade, 1983) → eigenimages ($H_i = \Psi_i$).

Images $F^{(1)}$, $F^{(2)}$, $\Psi_i$ — eigenimages of $F^{(1)}$.

Responses:
- $Y_i^{(1)} = F^{(1)} \star \Psi_i = \sqrt{\lambda_i} \Phi_i$ — factor image, $DY_i^{(1)} = \lambda_i$.
- $Y_i^{(2)}$ — projection of trajectory space on $\Psi_i$.

Early work (just decomposition, using local variances):
- (Unser, Ade, 1983) — defect detection in textile materials.
- (Patel et al., 1996) — detection of food contaminants.

Recent work (reconstruction attempts):
- (Monadjemi, 2004) — defect detection in ceramic tiles

Related approach: optimal discrimination filter.

Two textures $F^{(1)}$, $F^{(2)}$.

Maximum discrimination of local variances of filter responses $Y^{(1)}$ and $Y^{(2)}$:

Solution of $\lambda S^{(1)} U = S^{(2)} U$. 

9/14 Konstantin Usevich SSA for textured images
Eigenfilters:

- Small window sizes (3x3, 5x5, 7x7)
- Just one-dimensional projections
- Not using eigenvalues

Detection based on eigenvalues:

<table>
<thead>
<tr>
<th>Textured</th>
<th>Non-textured</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Textured Image" /></td>
<td><img src="image2.png" alt="Non-textured Image" /></td>
</tr>
</tbody>
</table>

Measures:

- Sum of jumps: \( \sum_{\Delta \lambda_i > 4 \Delta \lambda_{i-1}} \% \Delta \lambda_i \)
- Share of eigenvalues: \( \sum_{i=1}^{d} \% \lambda_i \)
Texture detection by eigenvalues

Sum of jumps and shares of eigenvalues:
Comparing textures using eigenvalues

Distance based on eigenvalues (Zhigljavsky, 2010, SII)

\( F^{(1)}, F^{(2)} \); trajectory matrices \( X^{(1)} \) and \( X^{(2)} \).

Normalized matrices (sum of singular values = 1):

\[
W_1 = \frac{X^{(1)}}{\sqrt{\text{tr}(X^{(1)}(X^{(1)})^T)}} \\
W_2 = \frac{X^{(2)}}{\sqrt{\text{tr}(X^{(2)}(X^{(2)})^T)}}
\]

Joint trajectory matrix: \( W_{12} = \begin{pmatrix} W_1 \\ W_2 \end{pmatrix} \).

\( \lambda_1, \ldots, \lambda_d \) — eigenvalues of \( S^{(1)} = W_1 W_1^T \),

\( \mu_1, \ldots, \mu_d \) — eigenvalues of \( S^{(2)} = W_2 W_2^T \),

\( \nu_1, \ldots, \nu_d \) — eigenvalues of \( S^{(12)} = \begin{pmatrix} S^{(1)} & W_1 W_1^T \\ W_2 W_1^T & S^{(2)} \end{pmatrix} \).

Eigenvalues \( \rightarrow \) c.d.f.: \( F_1(t) = \sum_{i=1}^{\lfloor t \rfloor} \lambda_i, F_2(t) = \sum_{i=1}^{\lfloor t \rfloor} \mu_i, F_{12}(t) = \sum_{i=1}^{\lfloor t \rfloor} \nu_i \).

\[
\rho_{\infty}(F^{(1)}, F^{(2)}) = \max_{0 < t < k} (F_1(t) + F_2(t) - F_{12}(t)) \\
\rho_1(F^{(1)}, F^{(2)}) = \sum_{0 < t < k} (F_1(t) + F_2(t) - F_{12}(t))
\]
Texture classification tests

- Use centered matrices $S = (X - XEE^T)(X - XEE^T)^T$
- $S_1^{(12)} = S^{(1)} + S^{(2)}$ instead of $S^{(12)} = \begin{pmatrix} S^{(1)} & w_1 w_2^T \\ w_2 w_1^T & S^{(2)} \end{pmatrix}$.

No need to store $W_1$, $W_2$.

Columbia-Utrecht database: textures under different viewing conditions.

- 40 texture, 92 images of each texture $\rightarrow$ training and test sets
- Choose the nearest image out of 1840 reference images.
- Classification accuracy: $\rho_{\infty} = 82.32\%$, $\rho_1 = 86.51\%$

Comparison to (Varma, Zisserman, 2005).
Texture segmentation

Base image

Sliding sub-image: 20x20, window size: 10 x 10.

Barbara

Subspace distance (4 comp.)

Eigenvalues-based distance

- Eigenvalues-based distance looks better
- Computational improvements: (Korobeynikov, 2010, SII).