Master Mathematical Analysis and Applications
Course M1 - S2

Computer vision

Fitting and alignment

Week 5

Mohammed Hachama
hachamam@gmail.com
http://hachama.github.io/home/

University of Khemis Miliana

-2020-

Plan

1. Fitting and alignment

SVD and QR decomposition

Factorization SVD

- Any matrix X can be decomposed as $X_{n\times p} = U_{n\times n}\,\Sigma_{n\times p}\,V^T_{p\times p}$,
  - U, V are orthogonal and $\sigma_1 \geq ... \geq \sigma_r$ are the singular values.
  - $\Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \\ & \bigcirc & \end{pmatrix}$ or $\Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \bigcirc \\ & & \sigma_r & \end{pmatrix}$.
- Columns of V are normalized eigenvectors of $X^T X$.
- Columns of U are normalized eigenvectors of XX^T.
- $\mathbf{u}_j$ and $\mathbf{v}_j$ share the same eigenvalue λ_j, where $\sigma_j = \sqrt{\lambda_j}$.
- Every matrix X of rank r has exactly r nonzero singular values.

Factorization SVD

- Full and compact (reduced) SVD

$$\overset{U_1\ (m \times r)}{\underset{U\ (m \times m)}{\begin{bmatrix} \boxed{\mathbf{u}_1 \ \cdots \ \mathbf{u}_r} & \mathbf{u}_{r+1} & \cdots & \mathbf{u}_m \end{bmatrix}}}
\overset{\Sigma_1\ (r \times r)}{\underset{\Sigma\ (m \times n)}{\begin{bmatrix} \boxed{\begin{matrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{matrix}} & & \\ & 0 & \\ & & \ddots & \\ & & & 0 \end{bmatrix}}}
\overset{V_1^{\mathsf{H}}\ (r \times n)}{\underset{V^{\mathsf{H}}(n \times n)}{\begin{bmatrix} \boxed{\begin{matrix} \mathbf{v}_1^{\mathsf{H}} \\ \vdots \\ \mathbf{v}_r^{\mathsf{H}} \end{matrix}} \\ \mathbf{v}_{r+1}^{\mathsf{H}} \\ \vdots \\ \mathbf{v}_n^{\mathsf{H}} \end{bmatrix}}}$$

- Full SVD : $X = U\Sigma V^T$
- Compact SVD : $X = U_1\Sigma_1 V_1^T$, i.e., $X = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$.

Computing the compact SVD

procedure $\textsc{compact\_SVD}(A)$
  $\boldsymbol{\lambda}, V \leftarrow \mathrm{eig}(A^{\mathsf{H}}A)$ $\triangleright$ Calculate the eigenvalues and eigenvectors of $A^{\mathsf{H}}A$.
  $\boldsymbol{\sigma} \leftarrow \sqrt{\boldsymbol{\lambda}}$ $\triangleright$ Calculate the singular values of A.
  $\boldsymbol{\sigma} \leftarrow \mathrm{sort}(\boldsymbol{\sigma})$ $\triangleright$ Sort the singular values **from greatest to least**.
  $V \leftarrow \mathrm{sort}(V)$ $\triangleright$ Sort the eigenvectors **the same way**.
  $r \leftarrow \mathrm{count}(\boldsymbol{\sigma} \neq 0)$ $\triangleright$ Count the number of nonzero singular values (the rank of A).
  $\boldsymbol{\sigma}_1 \leftarrow \boldsymbol{\sigma}_{:r}$ $\triangleright$ Keep only the positive singular values.
  $V_1 \leftarrow V_{:,:r}$ $\triangleright$ Keep only the corresponding eigenvectors.
  $U_1 \leftarrow AV_1/\boldsymbol{\sigma}_1$ $\triangleright$ Construct U with array broadcasting.
  return $U_1, \boldsymbol{\sigma}_1, V_1^{\mathsf{H}}$

Factorization SVD

- Intuition : X can be seen as a linear transformation. This transformation can be decomposed in three sub-transformations :
  1. Rotation,
  2. Re-scaling,
  3. Rotation.

  These three steps correspond to the three matrices U, D, and V.

Applications of SVD

1. Pseudoinverse

- The **pseudoinverse** of a rectangular matrix X is

$$X^{+} = V \begin{pmatrix} \sigma_1^{-1} & & & \\ & \ddots & & \bigcirc \\ & & \sigma_p^{-1} & \end{pmatrix} U^T$$

- X has linearly independent columns ($X^T X$ is invertible) :

$$X^{+} = (X^T X)^{-1} X^T.$$

  In this case, $X^{+} X = I$.

- X has linearly independent rows (matrix XX^T is invertible) :

$$X^{+} = X^T (XX^T)^{-1}.$$

  This is a right inverse, as $XX^{+} = I$.

Applications of SVD

2. Low-Rank Matrix Approximations

- If A is a $m \times n$ matrix of rank $r < min\{m, n\}$, store the matrices U_1, Σ_1 and V_1 instead of A.
  - Store A : mn values
  - Store the matrices U_1, Σ_1 and V_1 : $mr + r + nr$ values.
- Example : If A is 100×200 and has rank 20
  - Store A : $20,000$ values
  - Store the matrices U_1, Σ_1 and V_1 : $6,020$ enteries.

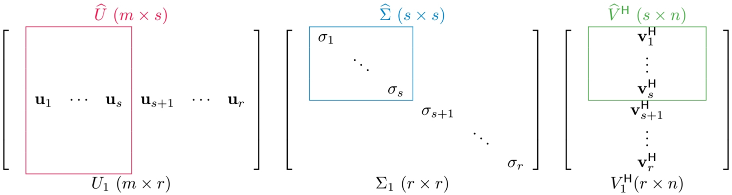
Applications of SVD

2. Low-Rank Matrix Approximations

- The *truncated SVD* keeps only the first $s < r$ singular values, plus the corresponding columns of U and V :

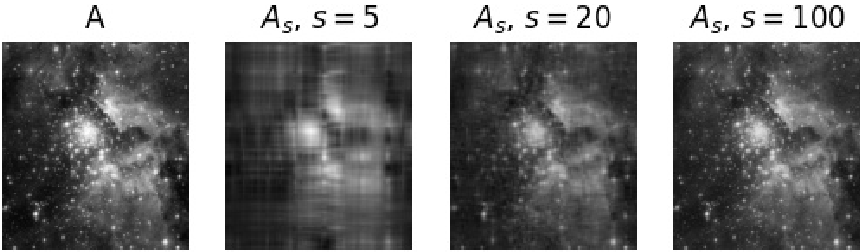
$$A_s = \sum_{i=1}^{s} \sigma_i \mathbf{u}_i \mathbf{v}_i^{\mathsf{H}}.$$

The resulting matrix A_s has rank s and is only an approximation to A, since $r - s$ nonzero singular values are neglected.



Applications of SVD

2. Low-Rank Matrix Approximations - Image compression



QR Factorization

- Any matrix $A_{n\times p}$ with linearly independent columns admits a unique decomposition

$$A_{n\times p} = Q_{n\times p} R_{p\times p}$$

where $Q^T Q = I$ and R is upper triangular.

  - QR is a useful factorization when $n > p$.

- Full QR decomposition

$$\underbrace{A}_{m\times n} = \underbrace{Q}_{m\times m} \underbrace{\begin{bmatrix} R \\ 0 \end{bmatrix}}_{m\times n} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \underbrace{\begin{bmatrix} R \\ 0 \end{bmatrix}}_{m\times n}$$

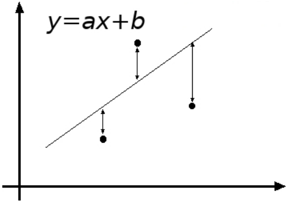
- Reduced QR decomposition

$$\underbrace{A}_{m\times n} = \underbrace{Q_1}_{m\times n} \underbrace{R}_{n\times n}$$

Least squares

Total Least squares

- Fitting a line to a point cloud



- Data : $(x_1, y_1), \cdot, (x_n, y_n)$
- Line equation : $y = ax + b$
- Find (a, b) to minimize

$$E(a, b) = \sum_{i=1}^{n} (y_i - ax_i - b)^2$$

Total Least squares

- Fitting a line to a point cloud

$$\begin{aligned} E(a,b) &= \sum_{i=1}^{n}(y_i - ax_i - b)^2 \\ &= ||A\mathbf{c} - \mathbf{d}||^2 \end{aligned}$$

$$A = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} ; \mathbf{c} = \begin{bmatrix} a \\ b \end{bmatrix} \; ; \; \mathbf{d} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

- Cons
  - Fails for vertical lines
  - Not rotation-invariant

Total Least squares

- Minimize the function

$$\min_{\mathbf{c}\in\mathbb{R}^n} E(\mathbf{c}) \ , \ / \ E(\mathbf{c}) = \|A\mathbf{c} - \mathbf{d}\|^2$$

- The problem always has solution ($size(A) = m \times n$).
- Sol. is unique $\Leftrightarrow rank(A) = n$.
- Resolution
  - Normal equations
  - Orthogonal methods
  - SVD

Total Least squares

- Minimize the function

$$\min_{\mathbf{c}\in\mathbb{R}^n} E(\mathbf{c}) \ , \ / \ E(\mathbf{c}) = \|A\mathbf{c} - \mathbf{d}\|^2$$

- EL :

$$\frac{\partial E}{\partial \mathbf{c}} = 2(A^T A)\mathbf{c} - 2A^T\mathbf{d} = 0.$$

- Normal equation

$$A^T A \, \mathbf{c} \ = \ A^T \mathbf{d}.$$

$$A^T A \, \mathbf{c} \ = \ A^T \mathbf{d} \implies \mathbf{c} = A^+ \mathbf{d}.$$

when A has linearly independent columns and where

$$A^+ = (A^T A)^{-1} A^T$$

Total Least squares

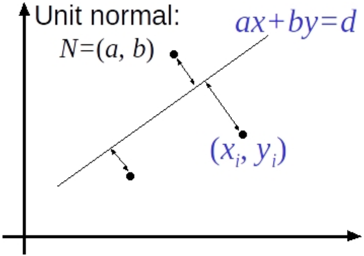
- Solution using QR factorization $A = QR$

$$\begin{aligned}\hat{\mathbf{c}} &= (A^T A)^{-1} A^T \mathbf{d} \\ &= ((QR)^T (QR))^{-1} (QR)^T \mathbf{d} \\ &= (R^T Q^T QR)^{-1} R^T Q^T \mathbf{d} \\ &= (R^T R)^{-1} R^T Q^T \mathbf{d} \\ &= R^{-1} R^{-T} R^T Q^{-1} \mathbf{d} \\ &= R^{-1} Q^T \mathbf{d}\end{aligned}$$

- Algorithm
  - 1. compute QR factorization $A = QR$.
  - 2. matrix-vector product $x = Q^T d$
  - 3. solve $Rc = x$ by back substitution

Orthogonal least squares

- Fitting a line to a point cloud



$$E(a,b) = \sum_{i=1}^{n} (ax_i + by_i - d)^2$$

$$\frac{\partial E}{\partial d} = \sum_{i=1}^{n} -2(ax_i + by_i - d) = 0$$

Orthogonal least squares

- Fitting a line to a point cloud

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \ , \ \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$\begin{aligned} E &= \sum_{i=1}^{n}(a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 \\ &= \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\| \\ &= (UN)^T(UN) \end{aligned}$$

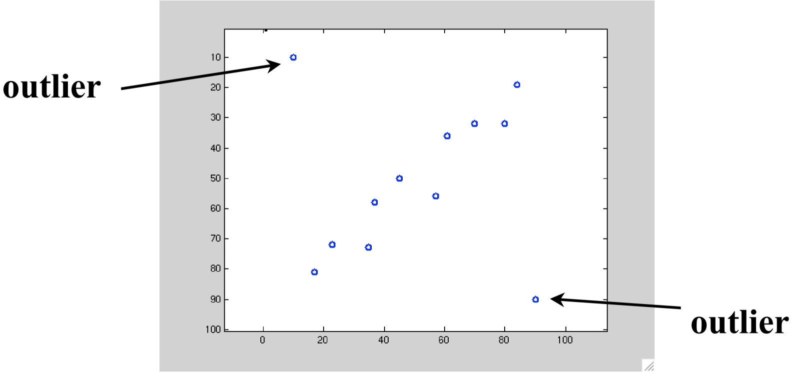
- min E s.t. $||N||^2 = 1$: eigenvector of $U^T U$ associated with the smallest eigenvalue[1].

1. Proof : exercise.

RANSAC

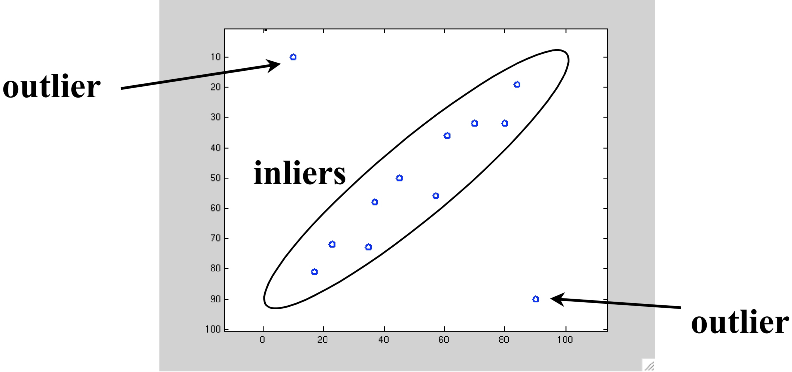
Robust estimation

- Outliers are points that don't "fit" the model.



Robust estimation

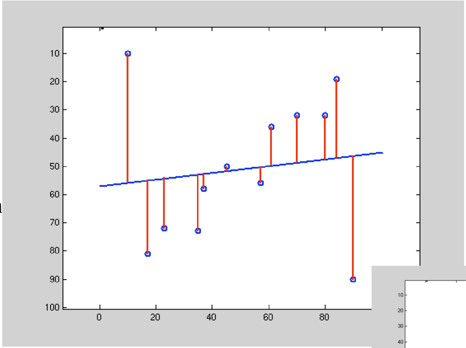
- Outliers are points that don't "fit" the model.



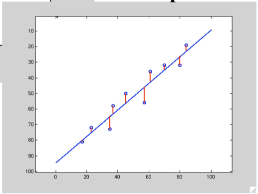
Robust estimation

- Outliers are points that don't "fit" the model.

Least squares regression with outliers



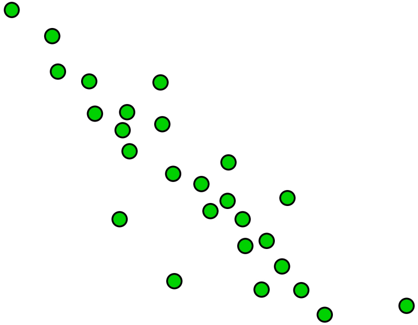
compare



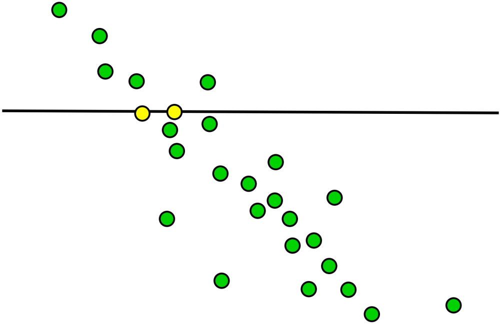
Algorithm

- RANSAC or "RANdom SAmple Consensus" an iterative method to "**robustly**" estimate parameters of a mathematical model from a set of observed data which contains outliers.
- Untill N iterations have occured :
  - Draw a random sample of S points from the data
  - Fit the model to that set of S points
    - Classify data points as outliers or inliers
    - ReFit the model to inliers while ignoring outliers
- Use the best fit from this collection using the fitting error as a criterion.

Example : Line fitting

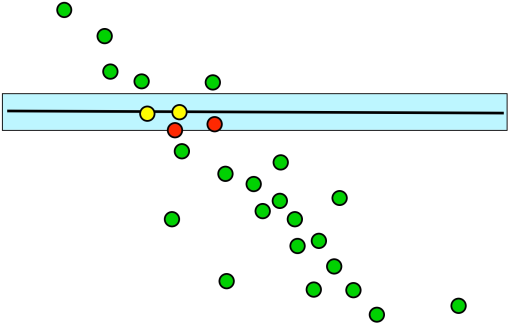


Example : Line fitting

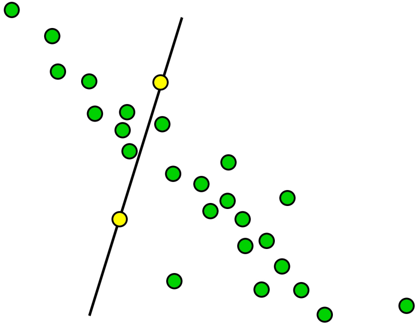


Example : Line fitting

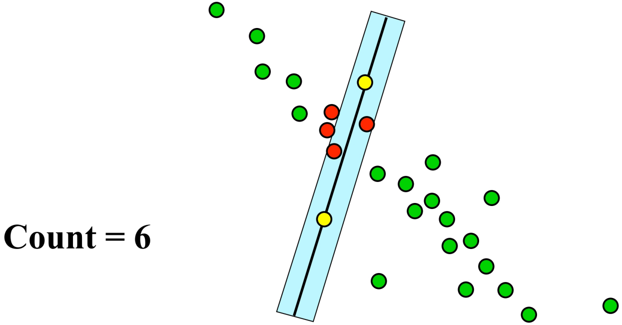
Count = 4



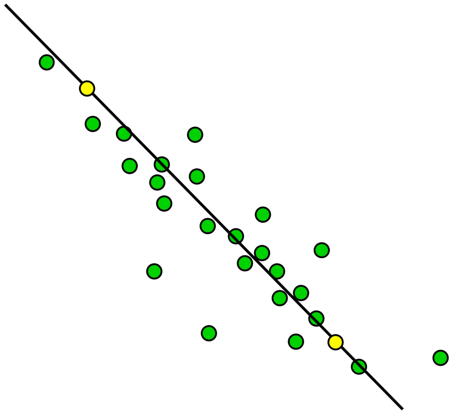
Example : Line fitting



Example : Line fitting

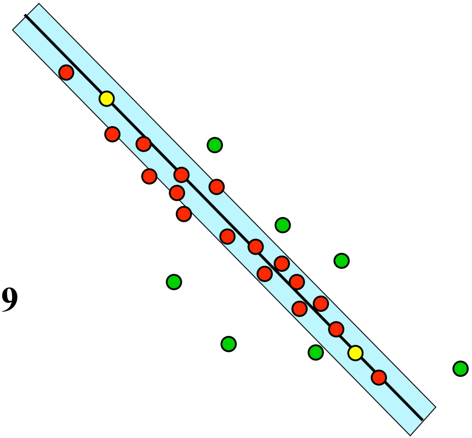


Example : Line fitting

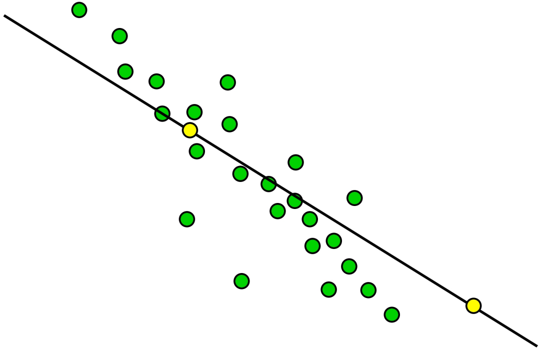


Example : Line fitting

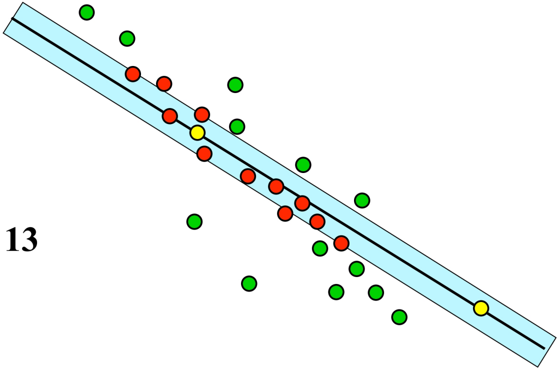
Count = 19



Example : Line fitting

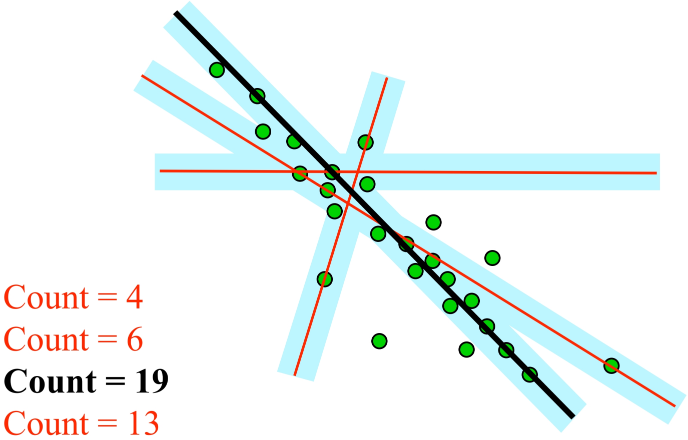


Example : Line fitting



Count = 13

Example : Line fitting

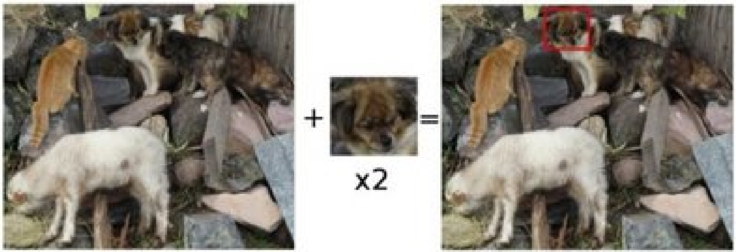


Example : Compute a mapping between two images

- Several hundred key points are extracted from each image and the goal is to match them and compute the transformation which minimises a given criterion.
- There may be outliers (incorrect matches) which will corrupt the estimation.
- RANSAC : Choose a subset of the points from one image, match these to the other image and compute the transformation which minimises the re-projection error. Choose another subset of the points, match them and compute another transformation. Repeat a couple of times using a different subset of key points each time. Then select the transformation which has the minimum re-projection error.

Template matching

- Source image (I) : The image in which we expect to find a match to the template image
- Template image (T) : The patch image which will be compared to the template to detect the highest matching area



- Comparing the template against the source image by sliding it
  - Moving the patch one pixel at a time.
  - At each location, a similarity metric is calculated

- Store the metric in the result matrix (image) (R).

- Metrics
  - Sum of Square differences

$$R(x,y) = \sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2$$

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') - I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

  - Cross-correlation

$$R(x,y) = \sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))$$

$$R(x,y) = \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

- Metrics
  - Correlation coefficient

$$R(x,y)=\sum_{x',y'}(T'(x',y')\cdot I'(x+x',y+y'))$$

where

$$T'(x',y')=T(x',y')-1/(w\cdot h)\cdot\sum_{x'',y''}T(x'',y'')$$

$$R(x,y)=\frac{\sum_{x',y'}(T'(x',y')\cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'}T'(x',y')^2\cdot\sum_{x',y'}I'(x+x',y+y')^2}}$$