Master SIF - REP (15-16/20)
Advanced tools for Digital Image Processing I

Thomas Maugey
thomas.maugey@inria.fr

Table of Contents

Table of Contents

Morphological Component Analysis

$\mathbf{x}$ is an image that is vectorized.
N is the number of pixels in the image.

The **Morphological Component Analysis** consists in decomposing an image as the sum of a cartoon image and a texture image:

$$\mathbf{x} =$$

[Elad, M., Starck, J. L., Querre, P., and Donoho, D. L. (2005). Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). Applied and Computational Harmonic Analysis, 19(3), 340-358.]

[Starck, J.-L., Murtagh, F. and J. Fadili, A., Sparse Image and Signal Processing: Wavelets, Curvelets, Morphological Diversity , Cambridge University Press, 2010]

Morphological Component Analysis

$\mathbf{x}$ is an image that is vectorized.
N is the number of pixels in the image.

The **Morphological Component Analysis** consists in decomposing an image as the sum of a cartoon image and a texture image:

$$\mathbf{x} = \mathbf{x}_n +$$

[Elad, M., Starck, J. L., Querre, P., and Donoho, D. L. (2005). Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). Applied and Computational Harmonic Analysis, 19(3), 340-358.]

[Starck, J.-L., Murtagh, F. and J. Fadili, A., Sparse Image and Signal Processing: Wavelets, Curvelets, Morphological Diversity , Cambridge University Press, 2010]

Morphological Component Analysis

$\mathbf{x}$ is an image that is vectorized.
N is the number of pixels in the image.

The **Morphological Component Analysis** consists in decomposing an image as the sum of a cartoon image and a texture image:

$$\mathbf{x} = \mathbf{x}_n + \mathbf{x}_t$$

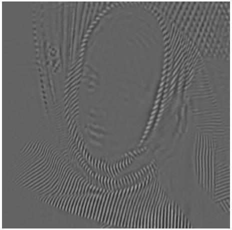
[Elad, M., Starck, J. L., Querre, P., and Donoho, D. L. (2005). Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). Applied and Computational Harmonic Analysis, 19(3), 340-358.]

[Starck, J.-L., Murtagh, F. and J. Fadili, A., Sparse Image and Signal Processing: Wavelets, Curvelets, Morphological Diversity , Cambridge University Press, 2010]

Texture image modeling

Let $\mathbf{T}_t$ of dimension $N \times K$ (with $K >> N$) be a dictionary of texture. $\mathbf{X}_t$ is a texture image if it exists a sparse decomposition:

$$\mathbf{x}_t = \mathbf{T}_t \mathbf{c}_t, \quad \text{with } \mathbf{c}_t \text{ sparse}$$



The sparsity can be modeled with:

- $||\mathbf{c}_t||_0 = |\{k, c_t(k) \neq 0\}|$
- $||\mathbf{c}_t||_p = \left(\sum_{k=1}^{K} c_t(k)^p\right)^{\frac{1}{p}}$

Hypothesis:

- Localization: $\mathbf{T}_t$ should include multi-scale and local of textural information
- Incoherence: cartoon images cannot be sparsely described with $\mathbf{T}_t$

Cartoon image modeling

Let $\mathbf{T}_n$ of dimension $N \times K$ (with $K >> N$) be a dictionary of texture. $\mathbf{X}_n$ is a texture image if it exists a sparse decomposition:

$$\mathbf{x}_n = \mathbf{T}_n \mathbf{c}_n, \quad \text{with } \mathbf{c}_n \text{ sparse}$$

The sparsity can also be modeled with:

- $||\mathbf{c}_n||_0 = |\{k, c_n(k) \neq 0\}|$
- $||\mathbf{c}_n||_p = \left(\sum_{k=1}^{N} c_n(k)^p\right)^{\frac{1}{p}}$

Similar hypothesis:

- Localization: $\mathbf{T}_n$ should include multi-scale and local of textural information
- Incoherence: texture images cannot be sparsely described with $\mathbf{T}_n$

Morphological Component Analysis

The MCA decomposition is thus denoted as:

$$\mathbf{x} = \mathbf{T}_n \mathbf{c}_n + \mathbf{T}_t \mathbf{c}_t$$

Given the dictionaries, the decomposition is found by solving:

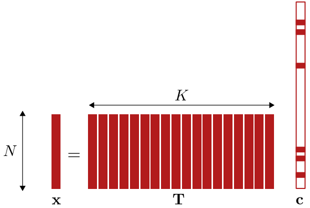
$$\{\mathbf{c}_n^*, \mathbf{c}_t^*\} = \text{arg} \min_{\mathbf{c}_n, \mathbf{c}_t} ||\mathbf{c}_n||_0 + ||\mathbf{c}_t||_0 \quad \text{s.t. } \mathbf{x} = \mathbf{T}_n \mathbf{c}_n + \mathbf{T}_t \mathbf{c}_t$$

This formulation can be relaxed such that the decomposition becomes an approximation (with a small approximation error ε):

$$\{\mathbf{c}_n^*, \mathbf{c}_t^*\} = \text{arg} \min_{\mathbf{c}_n, \mathbf{c}_t} ||\mathbf{c}_n||_0 + ||\mathbf{c}_t||_0 \quad \text{s.t. } ||\mathbf{x} - \mathbf{T}_n \mathbf{c}_n - \mathbf{T}_t \mathbf{c}_t||_2^2 < \varepsilon$$

Matching pursuit

An optimization problem as
$\mathbf{c}^* = \arg\min_{\mathbf{c}} ||\mathbf{c}||_0$ s.t. $||\mathbf{x} - \mathbf{Tc}||_2^2 < \varepsilon$
(with $\mathbf{T}$ being the dictionary) can be solved as using a matching pursuit algorithm



- A residual vector $\mathbf{r}_0$ that is initialized with $\mathbf{x}$
- Initialize $\mathbf{c}$ as a zero vector of size K
- At every iteration $J \geq 0$,
  - Find the column of $\mathbf{T} = \{\mathbf{t}_k\}_{k \leq K}$ for which the inner product $\mathbf{r}_{(J)}^\top \mathbf{t}_k$ is maximal
    $$k^*_{(J)} \leftarrow \arg\max_{k \leq K} \mathbf{r}_{(J)}^\top \mathbf{t}_k$$
  - $c_{k^*_{(J)}} \leftarrow \mathbf{r}_{(J)}^\top \mathbf{t}_{k^*_{(J)}} / ||\mathbf{t}_{k^*_{(J)}}||_2^2$
  - $\mathbf{r}_{(J+1)} \leftarrow \mathbf{r}_{(J)} - c_{k^*_{(J)}} \mathbf{t}_{k^*_{(J)}}$
- Stop when the residue is sufficiently small $||\mathbf{r}_k||_2^2 < \varepsilon$

Orthogonal Matching Pursuit

The main difference from MP is that after every step, all the coefficients extracted so far are updated, by computing the orthogonal projection of the signal onto the set of atoms selected so far. This can lead to better results than standard MP, but requires more computation.

- A residual vector $\mathbf{r}_0$ that is initialized with $\mathbf{x}$
- Initialize $\mathbf{c}$ as a zero vector of size K and $\mathbf{A}_{(0)} = \emptyset$
- At every iteration $J \geq 1$,
  - Find the column of $\mathbf{T} = \{\mathbf{t}_k\}_{k \leq K}$ for which the inner product $\mathbf{r}_{(J)}^\top \mathbf{t}_k$ is maximal
  $$k^*_{(J)} \leftarrow \arg\max_{k \leq K} \mathbf{r}_{(J)}^\top \mathbf{t}_k$$
  - $\mathbf{A}_{(J)} \leftarrow \mathbf{A}_{(J-1)} \cup \{\mathbf{t}_{k^*_{(J)}}\}$
  - $\mathbf{P}_{(J)} \leftarrow \mathbf{A}_{(J)}(\mathbf{A}_{(J)}^\top \mathbf{A}_{(J)})^{-1}\mathbf{A}_{(J)}^\top$
  - $\mathbf{r}_{(J+1)} \leftarrow (\mathbf{I} - \mathbf{P}_{(J)})\mathbf{r}_{(J-1)}$
- Stop when the residue is sufficiently small $||\mathbf{r}_k||_2^2 < \varepsilon$

[Cai, T., and Wang, L. (2011). Orthogonal matching pursuit for sparse signal recovery with noise. IEEE Transactions on Information theory, 57(7), 4680-4688.]

Alternatives formulations

Other formulations are possible, using for example the l_1 norm in order to make the problem convex:

$$\{\mathbf{c}_n^*, \mathbf{c}_t^*\} = \arg\min_{\mathbf{c}_n, \mathbf{c}_t} ||\mathbf{c}_n||_1 + ||\mathbf{c}_t||_1 \quad \text{s.t. } \mathbf{x} = \mathbf{T}_n\mathbf{c}_n + \mathbf{T}_t\mathbf{c}_t$$

Regularization terms might be added to make the convergence easier (*e.g.*, total variation of cartoon image).

$$\{\mathbf{c}_n^*, \mathbf{c}_t^*\} = \arg\min_{\mathbf{c}_n, \mathbf{c}_t} ||\mathbf{c}_n||_1 + ||\mathbf{c}_t||_1 + \lambda||\mathbf{x} - \mathbf{T}_n\mathbf{c}_n - \mathbf{T}_t\mathbf{c}_t||_2^2 + \gamma TV(\mathbf{T}_n\mathbf{c}_n)$$

Possible solvers:

[S.S. Chen, D.L. Donoho, M.A. Saunder, Atomic decomposition by basis pursuit, SIAM J. Sci. Comput. 20 (1998) 33âĂŞ61.]

[D.L. Donoho, M. Elad, V. Temlyakov, Stable recovery of sparse overcomplete representations in the presence of noise, IEEE Trans. Inform. Theory (2004),]

[L.I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation noise removal algorithm, Physica D 60 (1992) 259âĂŞ268.]

[T.A. Tropp, Just relax: Convex programming methods for subset selection and sparse approximation, IEEE Trans. Inform. Theory (2004)]

MCA-based Inpainting

Let us define a mask $\mathbf{M}$, which is a diagonal matrix of dimension $N \times N$, whose i^{th} diagonal element is 1 if the pixel i is visible (and 0 otherwise).

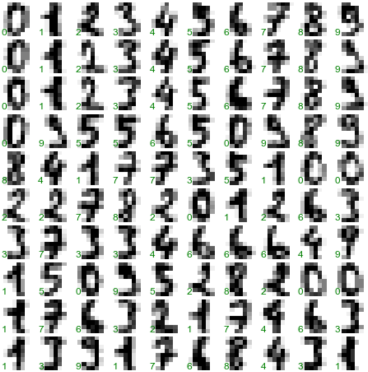
The inpainting formulation becomes

$$\{\mathbf{c}_n^*, \mathbf{c}_t^*\} = \arg \min_{\mathbf{c}_n, \mathbf{c}_t} ||\mathbf{c}_n||_1 + ||\mathbf{c}_t||_1 + \lambda ||\mathbf{M}(\mathbf{x} - \mathbf{T}_n \mathbf{c}_n - \mathbf{T}_t \mathbf{c}_t)||_2^2 + \gamma TV(\mathbf{T}_n \mathbf{c}_n)$$

This formulation is very similar to the image decomposition, and can be solved similarily

Illustrative example

Let us consider the following dictionary of hand-written digits numbers ($N = 64$ and $K = 1790$)



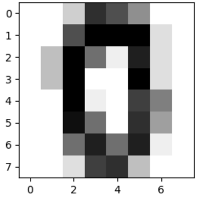
[F. Alimoglu, E. Alpaydin, "Methods of Combining Multiple Classifiers Based on Different Representations for Pen-based Handwriting Recognition," Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96), June 1996, Istanbul, Turkey.]

Illustrative example

We pick one digit from the database

```
from sklearn import datasets
import matplotlib.pyplot as plt

digits = datasets.load_digits()
i_missing = 20
plt.figure(1, figsize=(3, 3))
plt.imshow(digits.images[i_missing], cmap=plt.cm.gray_r,
interpolation='nearest')
```




```
# We take the vectorized version of this image
im = digits.data[i_missing,:]
```

Illustrative example

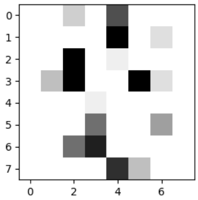
We remove some pixels of the selected digit

```
import numpy as np

# We create a mask with 50% of the pixels
vecRand = np.random.rand(64)
mask_ = vecRand > 0.5

# We mask the vector
im_masked = digits.data[i_missing,:]
im_masked[mask_] = 0

im_masked2d = im_masked.reshape(8,8,order='C').copy()
plt.figure(2, figsize=(3, 3))
plt.imshow(im_masked2d, cmap=plt.cm.gray_r, interpolation='nearest')
```



Illustrative example

We find the best decomposition of the masked vector in the masked dictionary

```
import sklearn.linear_model

# We build the dictionary in its masked version, and without the selected digit
index_item = np.append(np.arange(i_missing), np.arange(i_missing+1, 1790))
dico_m = digits.data[index_item, :]
dico_m[:, mask_] = 0

# We perform the OMP
coeff_ = sklearn.linear_model.orthogonal_mp(dico_m.transpose(),
im_masked.transpose(), n_nonzero_coefs=2, tol=None, precompute=False,
copy_X=True, return_path=False, return_n_iter=False)
```

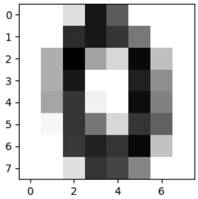
Illustrative example

We retrieve the full image from the complete dictionary

```
# We build the complete dictionary without the selected digit
dico = digits.data[index_item, :]

# We estimate the full digit
im_recon = dico.transpose() @ coeff_.transpose()
im_recon[im_recon < 0] = 0

im_recon2d = im_recon.reshape(8,8,order='C').copy()
plt.figure(3, figsize=(3, 3))
plt.imshow(im_recon2d, cmap=plt.cm.gray_r, interpolation='nearest')
```



More complex Dictionaries

Dictionaries might be designed "by hand", choosing for example:

- known transforms
- fast to compute and to inverse

For texture dictionary $\mathbf{T}_t$

- local DCT
- Oscillatory wavelets
- Gabor transform

For structure dictionary $\mathbf{T}_n$

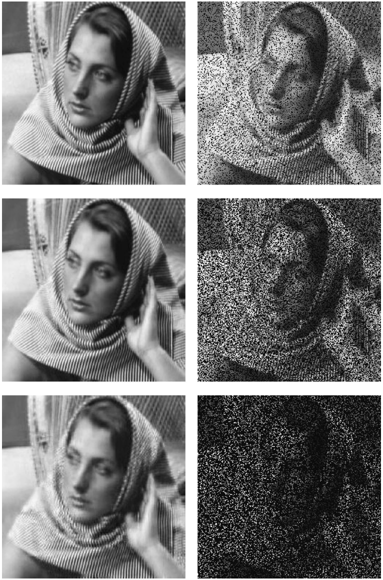
- curvelet
- ridgelet
- contourlet
- wavelet

[Elad, M., Starck, J. L., Querre, P., and Donoho, D. L. (2005). Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). Applied and Computational Harmonic Analysis, 19(3), 340-358.]

Results


nage inpainting [2, 10, 20, 38] is the proces
ing data in a designated region of a still or
lications range from removing objects fro
uching damaged paintings and photograp
produce a revised image in which the i
is seamlessly merged into the image in
detectable by a typical viewer. Traditional
been done by professional artists. For pho
, inpainting is used to revert deterioration
otographs or scratches and dust spots in fil
emove elements (e.g., removal of stamped
from photographs, the infamous "airbrushi
enemies [20]). A current active area of r

Results

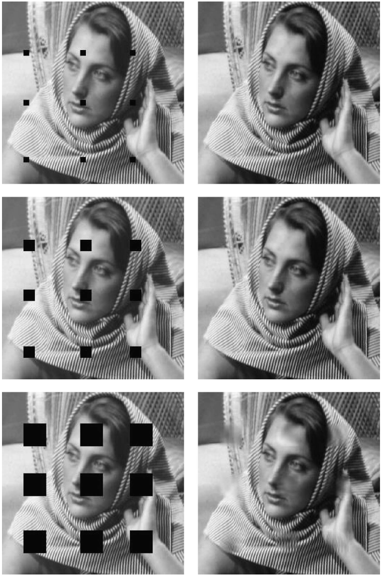


20% of missing pixels

50% of missing pixels

80% of missing pixels

Results



Learn the dictionary

Instead of building manually the dictionary, one can learn it from a set of data that has the same statistical properties than the processed ones.

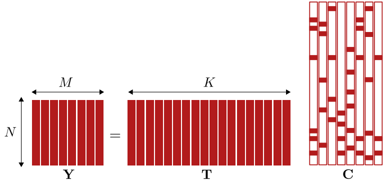
Given a set of M training signals $\mathbf{Y} = \{\mathbf{y}_i\}$, we seek the dictionary $\mathbf{T}$ that leads to the best representation for each $\mathbf{y}_i$:

$$\{\mathbf{T}^*, \mathbf{C}^*\} = \arg\min_{\mathbf{T},\mathbf{C}} \sum_i ||\mathbf{c}_i||_0 \quad \text{s.t. } ||\mathbf{Y} - \mathbf{TC}||_F^2 < \varepsilon$$

or equivalently

$$\{\mathbf{T}^*, \mathbf{C}^*\} = \arg\min_{\mathbf{T},\mathbf{C}} ||\mathbf{Y} - \mathbf{TC}||_F^2 \quad \text{s.t. } \forall i, \; ||\mathbf{c}_i||_0 < \eta$$

where $||.||_F$ is the Frobenius norm: $||\mathbf{A}||_F^2 = \sum_i \sum_j a_{ij}^2$.



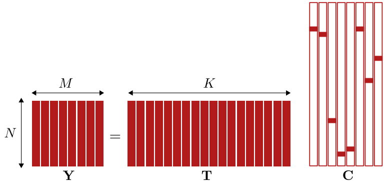
K-means algorithm

K-means algorithm deals with an extreme case of sparsity decomposition where each training signal $\mathbf{y}_i$ is represented by one of the K columns of $\mathbf{T}$:

$$\mathbf{y}_i \approx \mathbf{T}\mathbf{e}_k$$

where

- $\forall k \in [\![1, K]\!]$, $\mathbf{e}_k$ is a vector of dimension L that is 1 at the index k and 0 elsewhere.
- $\forall l \neq k, \; ||\mathbf{y}_i - \mathbf{T}\mathbf{e}_k||_2^2 < ||\mathbf{y}_i - \mathbf{T}\mathbf{e}_l||_2^2$



K-means algorithm

Objective of the K-means algorithm:
Find the best possible codebook $\mathbf{T}$ to represent $\{\mathbf{y}_i\}$:

$$\{\mathbf{T}^*, \mathbf{C}^*\} = \arg\min_{\mathbf{T},\mathbf{C}} ||\mathbf{Y} - \mathbf{TC}||_F^2 \quad \text{s.t. } \forall i, \exists k \in [\![1, K]\!], \; \mathbf{c}_i = \mathbf{e}_k$$

- Initialize $\mathbf{T}^{(0)}$, and $J = 0$
- At each iteration J
  - **Sparse coding stage**: Partition the training set $\{\mathbf{y}_i\}$ into $(R_1^{(J)}, \dots, R_K^{(J)})$, where

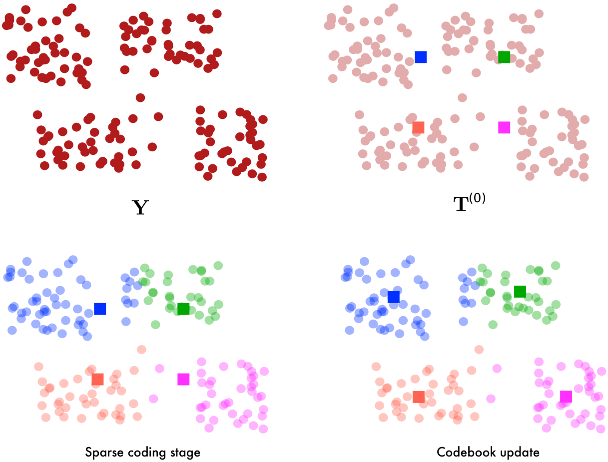
    $$R_k^{(J)} = \left\{ i \mid \forall l \neq k, \; ||\mathbf{y}_i - \mathbf{T}^{(J)}\mathbf{e}_k||_2^2 < ||\mathbf{y}_i - \mathbf{T}^{(J)}\mathbf{e}_l||_2^2 \right\}$$

  - **Codebook update**: for each column k of $\mathbf{T}^{(J)}$, update

    $$\mathbf{t}_k^{(J+1)} = \frac{1}{|R_K^{(J)}|} \sum_{i \in R_K^{(J)}} \mathbf{y}_i$$

- $J \leftarrow J + 1$

K-means illustration



K-SVD algorithm

The sparse representation problem

$$\{\mathbf{T}^*, \mathbf{C}^*\} = \arg\min_{\mathbf{T},\mathbf{C}} ||\mathbf{Y} - \mathbf{TC}||_F^2 \quad \text{s.t. } \forall i, \ ||\mathbf{c}_i||_0 < \eta$$

can be viewed as a generalization of the K-means, in which we allow each input signal $\mathbf{y}_i$ to be represented by a linear combination of columns of $\mathbf{T}$.

As K-means, the algorithm will alternate between

1. Find the best representation $\mathbf{C}$, given a dictionary $\mathbf{T}$
2. Update each column $\mathbf{t}_k$ one after the other, and finding, for each one, a better corresponding coefficients in $\mathbf{C}$ (based on SVD).

[Aharon, M., Elad, M., and Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on signal processing, 54(11), 4311.]

K-SVD algorithm

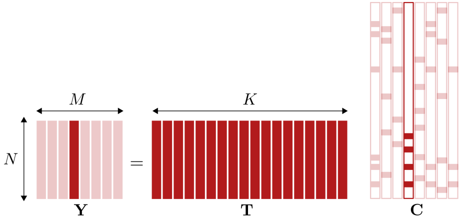
1- Find the best representation $\mathbf{C}$, given a dictionary $\mathbf{T}$

Solve a sparse representation problem for each training signal $\mathbf{y}_i$:

$$\forall i, \quad \mathbf{c}_i^* = \arg\min_{\mathbf{c}_i} ||\mathbf{y}_i - \mathbf{T}\mathbf{c}_i||_2^2 \quad \text{s.t.} \quad ||\mathbf{c}_i||_0 < \eta$$

This is done using the "pursuit algorithms" introduced before.

If η is small enough, their solution is a good approximation of the ideal one.



2- Update each column $\mathbf{t}_k$

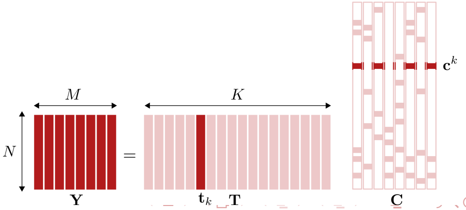
We assume that $\mathbf{T}$ and $\mathbf{C}$ are fixed, and we put in question $\mathbf{t}_k$ and the coefficients that correspond to it $\mathbf{c}^k$ (the k^{th} row of $\mathbf{C}$).

The penalty term becomes

$$\begin{aligned} \|\mathbf{Y}-\mathbf{TC}\|_F^2 &= \left\|\mathbf{Y}-\sum_{l=1}^{K}\mathbf{t}_l\mathbf{c}^l\right\|_F^2 \\ &= \left\|\left(\mathbf{Y}-\sum_{l\neq k}\mathbf{t}_l\mathbf{c}^l\right)-\mathbf{t}_k\mathbf{c}^k\right\|_F^2 \\ &= \left\|\mathbf{E}_k-\mathbf{t}_k\mathbf{c}^k\right\|_F^2 \end{aligned}$$

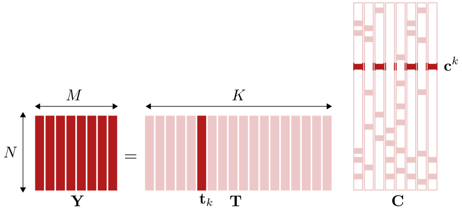
The term $\mathbf{TC}$ is decomposed into a sum of K rank-1 matrices, in which $K-1$ are fixed.

$\mathbf{E}_k$ stands for the errors for the samples when atom k is removed.



2- Update each column $\mathbf{t}_k$

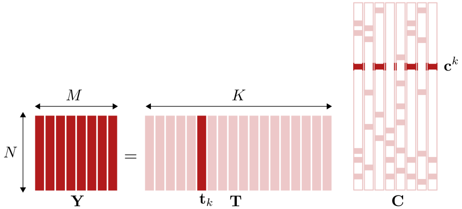
An SVD finds the closest rank-1 matrix that approximate $\mathbf{E}_k$. Use that to update $\mathbf{t}_k$ and $\mathbf{c}^k$?



K-SVD algorithm

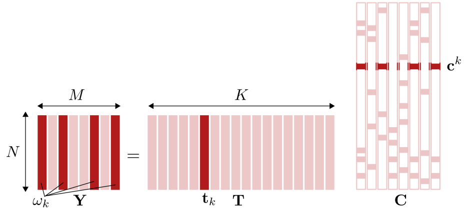
2- Update each column $\mathbf{t}_k$

An SVD finds the closest rank-1 matrix that approximate $\mathbf{E}_k$. Use that to update $\mathbf{t}_k$ and $\mathbf{c}^k$? No because $\mathbf{c}^k$ would not be sparse.



2- Update each column $\mathbf{t}_k$

An SVD finds the closest rank-1 matrix that approximate $\mathbf{E}_k$. Use that to update $\mathbf{t}_k$ and $\mathbf{c}^k$? No because $\mathbf{c}^k$ would not be sparse.



We define ω_k as the group of indices pointing to $\{\mathbf{y}_i\}$ that use the atom $\mathbf{t}_k$:

$$\omega_k = \{m \mid 1 \leq m \leq M,\ \mathbf{c}^k(m) \neq 0\}$$

And $\mathbf{E}_k^R, \mathbf{c}_R^k$ as the respective restrictions of $\mathbf{E}_k$ and $\mathbf{c}^k$ whose columns are in ω_k.

The aim is to minimize $\left|\left|\mathbf{E}_k^R - \mathbf{t}_k \mathbf{c}_R^k\right|\right|_F^2$.

We compute the SVD of $\mathbf{E}_k^R$ leading to $\mathbf{E}_k^R = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top$. We update $\mathbf{t}_k$ with the first column of $\mathbf{U}$ and $\mathbf{c}_R^k$ with the first column of $\mathbf{V}$.

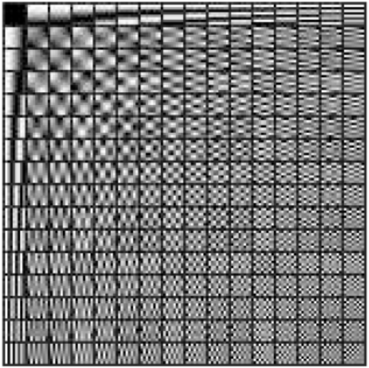
K-SVD algorithm

Objective: $\{\mathbf{T}^*, \mathbf{C}^*\} = \arg\min_{\mathbf{T},\mathbf{C}} ||\mathbf{Y} - \mathbf{T}\mathbf{C}||_F^2 \quad \text{s.t. } \forall i,\ ||\mathbf{c}_i||_0 < \eta$

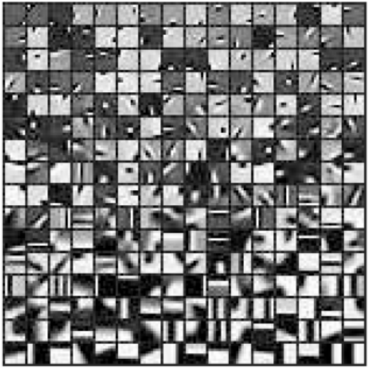
- Initialize $\mathbf{T}^{(0)}$ with l_2 normalized columns, $J = 0$
  - **Sparse Coding stage**: $\forall i$, solve
    $$\mathbf{c}_i^{(J)} = \arg\min_{\mathbf{c}_i} ||\mathbf{y}_i - \mathbf{T}^{(J)}\mathbf{c}_i||_2^2 \quad \text{s.t. } ||\mathbf{c}_i||_0 < \eta$$
    using any for example OMP algorithm.
  - **Code update Stage**: for each column $k \in [\![1, K]\!]$, update $\mathbf{t}_k^{(J)}$ and its corresponding coefficients:
    - Define the group of training signals that use this atom,
      $$\omega_k = \{m \mid 1 \leq m \leq M,\ \mathbf{c}^k_{(J)}(m) \neq 0\}$$
    - Compute the overall representation error matrix
      $$\mathbf{E}_k = \mathbf{Y} - \sum_{l \neq k} \mathbf{t}_l^{(J)} \mathbf{c}^l_{(J)}$$
    - Restrict $\mathbf{E}_k$ and $\mathbf{c}^k_{(J)}$ by choosing the column that belongs to ω_k and obtain $\mathbf{E}_k^R$ and $\mathbf{c}^k_{(J),R}$
    - Apply SVD $\mathbf{E}_k^R = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^\top$, and
      $$\mathbf{t}_k^{(J+1)} \leftarrow \text{first column of } \mathbf{U}$$
      $$\mathbf{c}^k_{(J+1),R} \leftarrow \text{first column of } \mathbf{V}$$
  - $J \leftarrow J + 1$

Example of learned dictionaries

Learning a dictionary on patches (8×8) of several image ($K = 256$)



Overcomplete DCT

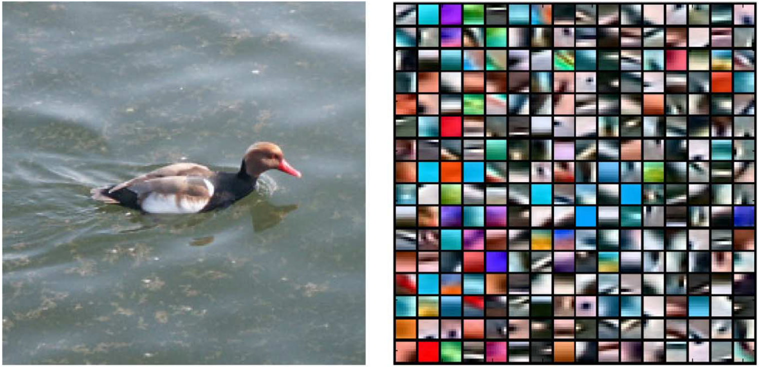


Learned Dictionary

[Elad, M., and Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. IEEE Transactions on Image processing, 15(12), 3736-3745.]

Example of learned dictionaries

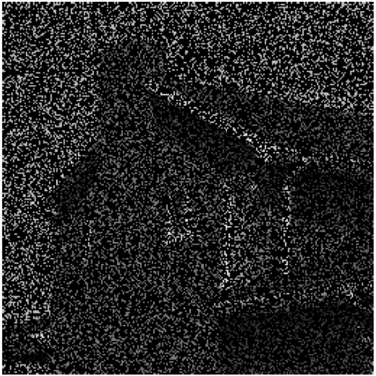
Learning a dictionary on patches ($8 \times 8 \times 3$) of image (a) results in dictionary (b)



(a) (b)

[J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. IEEE Transactions on Image Processing, 17(1):53âĂŞ69, January 2008b.]

Application to inpainting

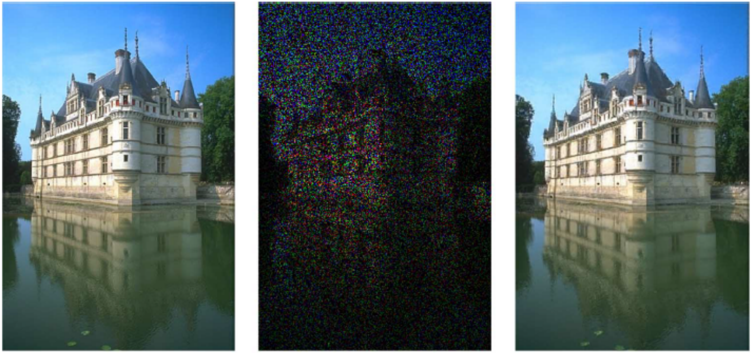


80% missing pixels

Recovered Image

[J. Mairal, G. Sapiro, and M. Elad. Learning multiscale sparse representations for image and video restoration. SIAM Multiscale Modelling and Simulation, 7(1): 214âĂŞ241, April 2008d.]

Application to inpainting



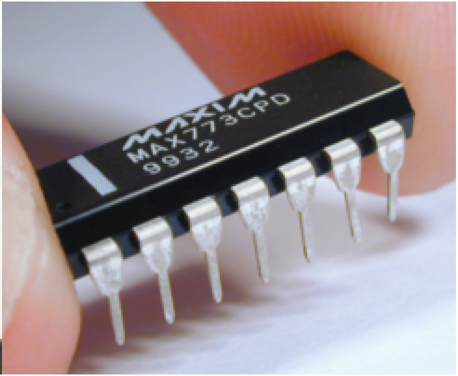
Original image | 80% missing pixels | Recovered Image

[J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. IEEE Transactions on Image Processing, 17(1):53âĂŞ69, January 2008b.]

Super resolution or Digital Zooming

When you increase the size of an image (*e.g.*, by a factor of 4), you may :

Naively increase the size of each pixel

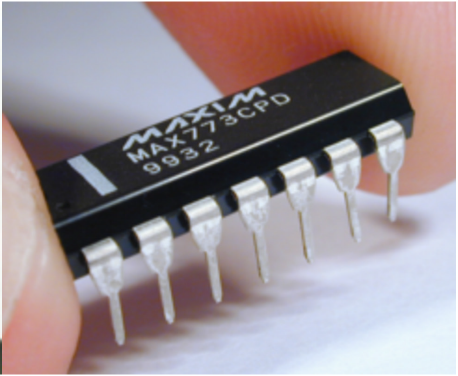


[Couzinie-Devy, F., Mairal, J., Bach, F., and Ponce, J. (2011). Dictionary learning for deblurring and digital zoom. arXiv preprint arXiv:1110.0957.]

Super resolution or Digital Zooming

When you increase the size of an image (*e.g.*, by a factor of 4), you may :

Locally interpolate between pixels

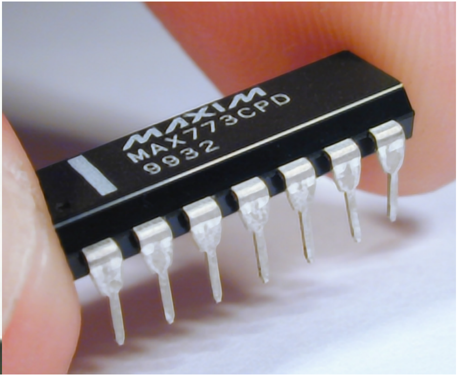


[Couzinie-Devy, F., Mairal, J., Bach, F., and Ponce, J. (2011). Dictionary learning for deblurring and digital zoom. arXiv preprint arXiv:1110.0957.]

Super resolution or Digital Zooming

When you increase the size of an image (*e.g.*, by a factor of 4), you may :

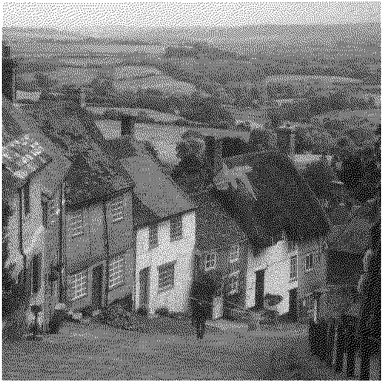
Use dictionary-based formulation



[Couzinie-Devy, F., Mairal, J., Bach, F., and Ponce, J. (2011). Dictionary learning for deblurring and digital zoom. arXiv preprint arXiv:1110.0957.]

Inverse Half-Toning

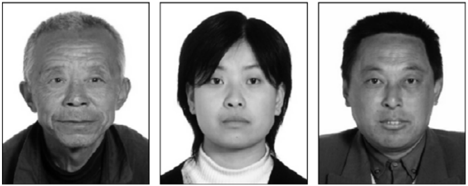
Inverse Half-Toning



and many other applications . . .

Application to compression

Dictionary learning can be useful when the case of study can be **specific**
For example, when compressing faces

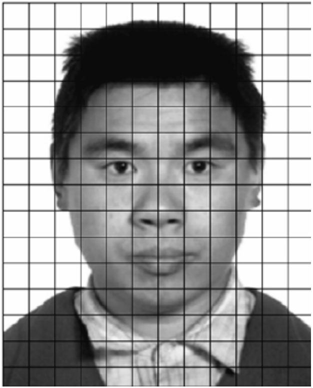


Transforms can be learned for each specific parts of the face

[Bryt, O., and Elad, M. (2008). Compression of facial images using the K-SVD algorithm. Journal of Visual Communication and Image Representation, 19(4), 270-282.]

Face dictionary learning

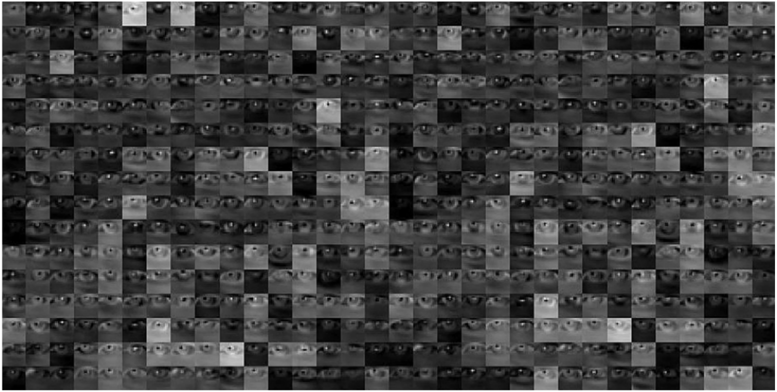
Align the faces and split them into blocks



Learn the dictionaries for each block

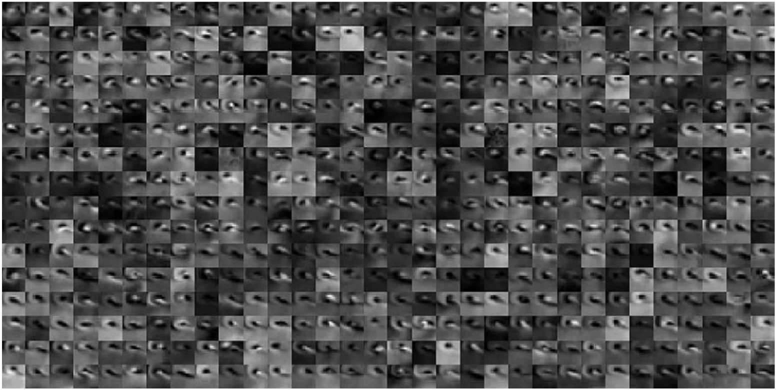
Examples of learned dictionaries

The Dictionary obtained by K-SVD for Patch No. 80 (the left eye)



Examples of learned dictionaries

The Dictionary obtained by K-SVD for Patch No. 87 (the right nostril)



Results

Original and Compressed images (632 bytes, 358×441 pixels)

Results

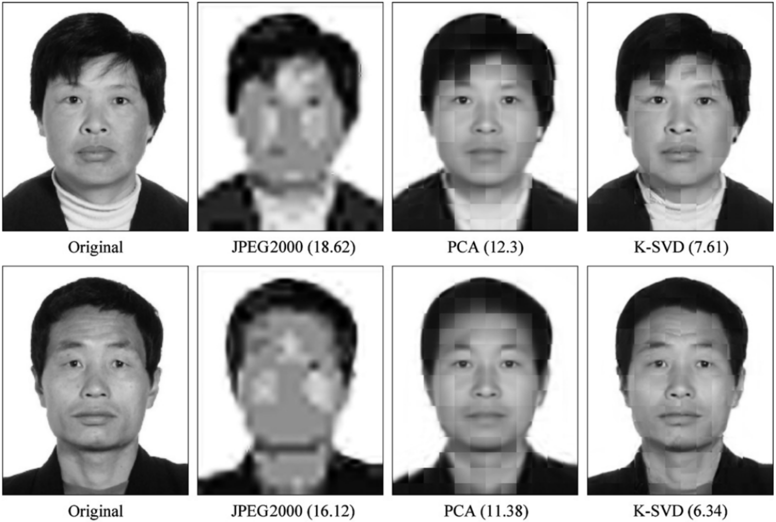
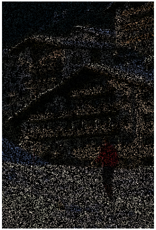

Original
JPEG2000 (18.62)
PCA (12.3)
K-SVD (7.61)
Original
JPEG2000 (16.12)
PCA (11.38)
K-SVD (6.34)

Table of Contents

Object or region removal

Need for a new type of inpainting

Original image



80% of the pixels have been removed.

Sparsity and low-rank methods

damaged portions in black, scratches

Diffusion-based methods

object removal*

Examplar-based methods*

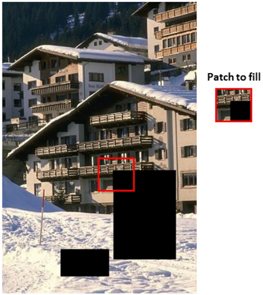
Examplar-based inpainting (1/4)

Texture synthesis

Examplar-based inpainting methods rely on the assumption that the known part of the image provides a good dictionary which could be used efficiently to restore the unknown part.

The recovered texture is therefore inferred from similar regions.

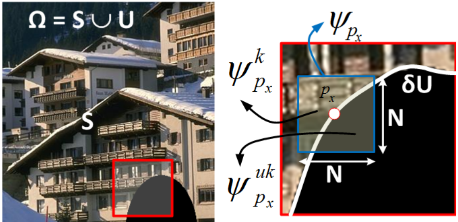
- Simply by sampling, copying or combining patches from the known part of the image;
  Template Matching
- Patches are then stitched together to fill in the missing area.



[A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In IEEE Computer Vision and Pattern Recognition (CVPR), pages 1033âĂŞ1038, 1999.]

Examplar-based inpainting (2/4)

Notations:



$$\psi_{p_x} = \begin{bmatrix} I_1(0,0) \\ I_1(0,1) \\ \vdots \\ I_M(N-2,N-2) \\ I_M(N-1,N-1) \end{bmatrix}$$

- a patch ψ_{p_x} is a discretized $N \times N$ neighborhood centered on the pixel p_x. This patch can be vectorized in a raster-scan order as a N^2-dimensional vector;
- $\psi^{uk}_{p_x}$ denotes the unknown pixels of the patch;
- $\psi^{k}_{p_x}$ denotes its known pixels;
- $\psi_{p_{x(i)}}$ denotes the i^{th} nearest neighbor of ψ_{p_x};
- δU is the front line;

Examplar-based inpainting (3/4)

Criminisi et al.'s algorithm

It has brought a new momentum to inpainting applications and methods. They proposed a new method based on two sequential stages:

1. Filling order computation;
2. Texture synthesis.

1. Filling order computation: $P(p_x) = C(p_x) \times D(p_x)$

Confidence term

$$C(p_x) = \frac{\sum_{q \in \psi^k_{p_x}} C(q)}{|\psi_{p_x}|}$$

where $|\psi_{p_x}|$ is the area of ψ_{p_x}.

Data term

$$D(p_x) = \frac{|\nabla I^{\perp}(p_x) \cdot \overrightarrow{n}_{p_x}|}{\alpha}$$

where α is a normalization constant in order to ensure that $D(p_x)$ is in the range 0 to 1.

[A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by examplar-based image inpainting. IEEE Trans. On Image Processing, 13:1200âĂŞ1212, 2004.]

Examplar-based inpainting (4/4)

2. Texture synthesis:

A template matching is performed within a local neighborhood:

$$p_q^* = \arg\min_{q \in \mathcal{W}} d(\psi_{p_q}^{\mathbf{k}}, \psi_{p_{x^*}}^{\mathbf{k}})$$

- $\mathcal{W} \subseteq S$ is the window search;
- $\psi_{p_{x^*}}^k$ are the known pixels of the patch $\psi_{p_{x^*}}$ with the highest priority;
- $\psi_{p_q^*}^k$ are the known pixels of the nearest patch neighbor;
- $d(a, b)$ is the sum of squared differences between a and b.

The pixels of the patch $\psi_{p_q^*}^{uk}$ are then copied into the unknown pixels of the patch $\psi_{p_{x^*}}$.

Filling order computation (1/4)

$$P(p_x) = C(p_x) \times D(p_x)$$

Two variants are here presented:

- Tensor-based data term

  [O. Le Meur, J. Gautier, and C. Guillemot. Examplar-based inpainting based on local geometry. In ICIP, 2011.]

- Sparsity-based data term

  [Z. Xu and J. Sun. Image inpainting by patch propagation using patch sparsity. IEEE Trans. on Image Processing, 19(5): 1153âĂŞ1165, 2010]

Many others: edge-based data term, transformation of the data term in a nonlinear fashion, entropy-based data term...

[P. Buyssens, M. Daisy, Tschumperlé, and O. LÃŁezoray. Exemplar-based inpainting: Technical review and new heuristics for better geometric reconstructions. IEEE Trans. On Image Processing, 2015.]

Filling order computation (2/4)

Tensor-based data term

Instead of using the gradient, we can used the structure tensor which is more robust:

$$D(p_x) = \alpha + (1-\alpha) exp\left(-\frac{\eta}{(\lambda_1 - \lambda_2)^2}\right)$$

where η is a positive value and $\alpha \in [0,1]$.

The structure tensor is a symmetric, positive semi-definite matrix:

$$J_{\rho,\sigma}[I] = K_\rho * \left(\sum_{i=1}^{m} \nabla(I_i * K_\sigma) \nabla(I_i * K_\sigma)^T\right)$$

where K_a is a Gaussian kernel with a standard deviation a. The parameters ρ and σ are called integration scale and noise scale, respectively.

[J. Weickert. Coherence-enhancing diffusion filtering. International Journal of Computer Vision, 32:111âĂŞ127, 1999.]

Filling order computation (3/4)

$$D(p_x) = \alpha + (1-\alpha)exp\left(-\frac{\eta}{(\lambda_1-\lambda_2)^2}\right)$$

When $\lambda_1 \simeq \lambda_2$, the data term tends to α. It tends to 1 when $\lambda_1 >> \lambda_2$.

Filling order computation (4/4)

Sparsity-based data term

Sparsity-based data term is based on the sparseness of nonzero patch similarities:

$$D(p_x) = \sqrt{\frac{|N_s(p_x)|}{|N(p_x)|} \times \sum_{p_j \in \mathcal{W}_s} w^2_{p_x,p_j}}$$

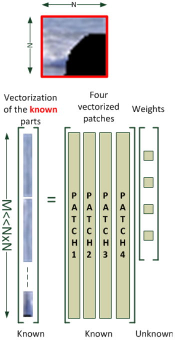
where N_s and N are the numbers of valid and candidate patches in the search window.
Weight w_{p_x,p_j} is proportional to the similarity between the two patches centered on p_x and p_j ($\sum_j w_{p_x,p_j} = 1$).

A large value of the structure sparsity term means sparse similarity with neighboring patches
$\Rightarrow$ a good confidence that the input patch is on some structure.

Texture synthesis (1/4)

Texture synthesis with more than one candidate

From K patches $\psi_{p_{x(i)}}$ which are the most similar to the known part $\psi_{p_x}^k$ of the input patch, the unknown part of the patch to be filled $\widehat{\psi}_{p_x}^{uk}$ is then obtained by a linear combination of the sub-patches $\psi_{p_{x(i)}}^{uk}$.



$$\psi_{p_x}^k = A \times w$$

$$\widehat{\psi}_{p_x}^{uk} = \sum_{i=1}^{K} w_i \psi_{p_{x(i)}}^{uk}$$

How can we compute the weights w_i of this linear combination?

Note: K is locally adjusted by using an ϵ-ball including patches within a certain radius.

Texture synthesis (2/4)

$$\widehat{\psi}^{uk}_{p_x} = \sum_{i=1}^{K} w_i \psi^{uk}_{p_{x(i)}}$$

Different solutions exist:

- Average template matching: $w_i = \frac{1}{K}, \forall i$;
- Non-local means approach:

$$w_i = exp\left(-\frac{d(\psi_{p^k_x}, \psi_{p^k_{x(i)}})}{h^2}\right)$$

[A. Buades, B. Coll, and J.M. Morel. A non local algorithm for image denoising. In IEEE Computer Vision and Pattern Recognition (CVPR), volume 2, pages 60âĂŞ65, 2005.]

- Least-square method minimizing

$$E(w) = \|\psi^k_{p_x} - Aw\|^2_{2,a}$$

$$w^* = \arg\min_w E(w)$$

Texture synthesis (3/4)

$$\widehat{\psi}_{p_x}^{uk} = \sum_{i=1}^{K} w_i \psi_{p_{x(i)}}^{uk}$$

- Constrained Least-square optimization with the sum-to-one constraint of the weight vector ⇒ LLE method

$$E(w) = \|\psi_{p_x}^{k} - Aw\|_{2,a}^2$$

$$w^* = \arg\min_w E(w) \qquad s.t. \qquad w^T \mathbf{1}_K = 1$$

[L.K. Saul and S.T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. Journal of Machine Learning Research, 4:119âĂŞ155, 2003.]

- Constrained Least-square optimization with positive weights ⇒ NMF method

$$w^* = \arg\min_w E(w) \qquad s.t. \qquad w_i \geq 0$$

[D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In In NIPS, pages 556âĂŞ562. MIT Press, 2001.]

Texture synthesis (4/4)

Similarity metrics:

- Using a Gaussian weighted Euclidean distance

$$d_{L^2}(\psi_{p_x}, \psi_{p_y}) = \|\psi_{p_x} - \psi_{p_y}\|_{2,a}^2$$

  where a controls the decay of the Gaussian function $g(k) = e^{-\frac{|k|}{2a^2}}$, $a > 0$;

- A better distance:

$$d(\psi_{p_x}, \psi_{p_y}) = d_{L^2}(\psi_{p_x}, \psi_{p_y}) \times (1 + d_H(\psi_{p_x}, \psi_{p_y}))$$

  where $d_H(\psi_{p_x}, \psi_{p_y})$ is the Hellinger distance

$$d_H(\psi_{p_x}, \psi_{p_y}) = \sqrt{1 - \sum_k \sqrt{p_1(k)p_2(k)}}$$

  where p_1 and p_2 represent the histograms of patches ψ_{p_x}, ψ_{p_y}, respectively.

[A. Bugeau, M. BertalmÂťÄśo, V. Caselles, and G. Sapiro. A comprehensive framework for image inpainting. IEEE Trans. on Image Processing, 19(10):2634âĂŞ2644, 2010]

[O. Le Meur and C. Guillemot. Super-resolution-based inpainting. In ECCV, pages 554âĂŞ567, 2012.]

Some Examples

Inpainted pictures with Criminisi's method (Courtesy of P. Pérez):

[A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by examplar-based image inpainting. IEEE Trans. On Image Processing, 13:1200âĂŞ1212, 2004.]

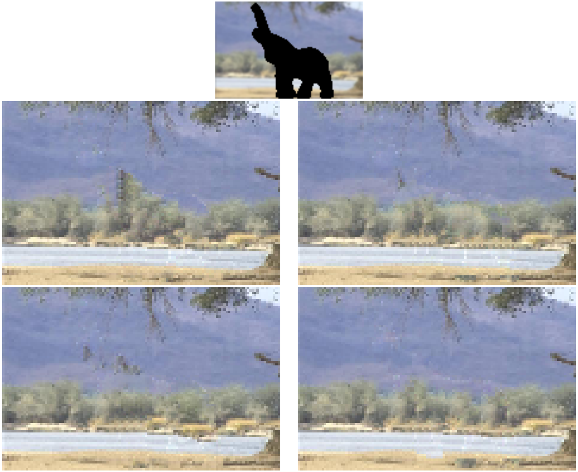
Table of Contents

Problems we want to solve... (1/4)

- The linear combination of several candidates induces blur.

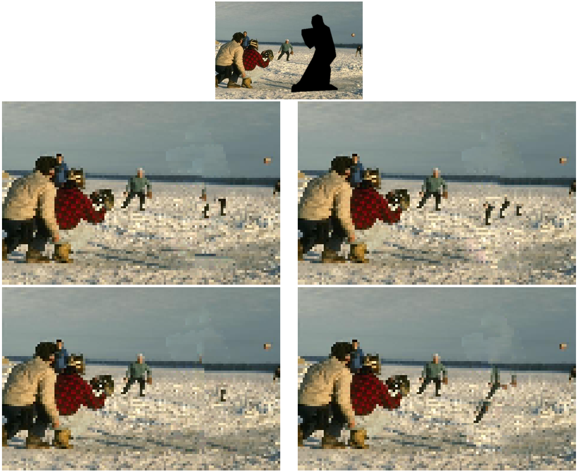
Problems we want to solve... (2/4)

- Very sensitive to the parameter settings such as the filling order and the patch size:



Problems we want to solve... (3/4)

- Very sensitive to the parameter settings such as the filling order and the patch size:



Problems we want to solve... (4/4)

- Examplar-based methods are one-pass greedy algorithms.

A greedy algorithm is an algorithm which makes the locally optimal choice at each stage with the hope of finding a global optimum.

The main idea (1/1)

Objectives of the proposed method

We apply an examplar-based inpainting algorithm several times and fuse together the inpainted results.

- less sensitive to the inpainting setting;
- relax the greedy constraint.

The main idea (1/1)

Objectives of the proposed method

We apply an examplar-based inpainting algorithm several times and fuse together the inpainted results.

- less sensitive to the inpainting setting;
- relax the greedy constraint.

The inpainting method is applied on a coarse version of the input picture:

- less demanding of computational resources;
- less sensitive to noise;
- K candidates for the texture synthesis without introducing blur.

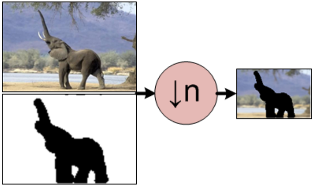
Need to fuse the inpainted images and to retrieve the highest frequencies

Loopy Belief Propagation and Super-Resolution algorithms.

More than one inpainting (1/1)

The baseline algorithm is an examplar-based method:

- Filling order computation;
- Texture synthesis.



- Decimation factor $n = 3$
- 13 sets of parameters

Table: Thirteen inpainting configurations.

Setting	Parameters
1	Patch's size 5×5 Decimation factor $n = 3$ Search window 80×80 Sparsity-based filling order
2	default + rotation by 180 degrees
3	default + patch's size 7×7
4	default + rotation by 180 degrees + patch's size 7×7
5	default + patch's size 11×11
6	default + rotation by 180 degrees + patch's size 11×11
7	default + patch's size 9×9
8	default + rotation by 180 degrees + patch's size 9×9
9	default + patch's size 9×9 + Tensor-based filling order
10	default + patch's size 7×7 + Tensor-based filling order
11	default + patch's size 5×5 + Tensor-based filling order
12	default + patch's size 11×11 + Tensor-based filling order
13	default + rotation by 180 degrees + patch's size 9×9 + Tensor-based filling order

Loopy Belief Propagation (1/4)

Loopy Belief Propagation is used to fuse together the 13 inpainted images.

Let be a finite set of labels $\mathbf{L}$ composed of $M = 13$ values.

$$E(l) = \sum_{p_x \in \mathbf{U}} V_d(l_{p_x}) + \lambda \sum_{(n,m) \in N_4} V_s(l_n, l_m)$$

where,

- l_{p_x} is the label of pixel p_x;
- N_4 is a neighbourhood system;
- λ is a weighting factor.

,

Loopy Belief Propagation (2/4)

$$E(l) = \sum_{p_x \in \mathbf{U}} V_d(l_{p_x}) + \lambda \sum_{(n,m) \in N_4} V_s(l_n, l_m)$$

- $V_d(l_{p_x})$ represents the cost of assigning a label l_{p_x} to a pixel p_x:

$$V_d(l_{p_x}) = \sum_{n \in \mathbf{L}} \sum_{u \in \upsilon} \left\{ \widehat{I}^{(l_{p_x})}(x+u) - \widehat{I}^{(n)}(x+u) \right\}^2$$

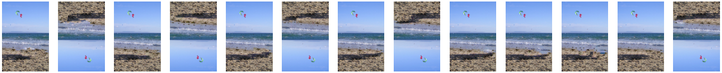
where, $\widehat{I}^{(n)}$ is an inpainted image ($n \in \{1, \ldots, M\}$).

- $V_s(l_n, l_m)$ is the discontinuity cost:

$$V_s(l_n, l_m) = (l_n - l_m)^2$$

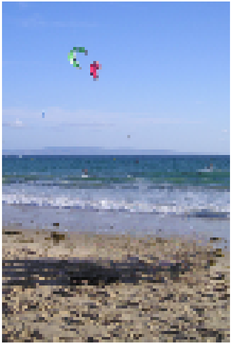
The minimization is performed iteratively (less than 15 iterations)

Loopy Belief Propagation (3/4)

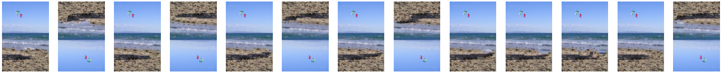


LBP convergence:

- Random initialization;
- 13 inpainted image in input;
- 25 iterations;
- resolution$=80 \times 120$.

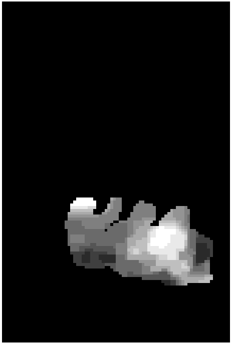


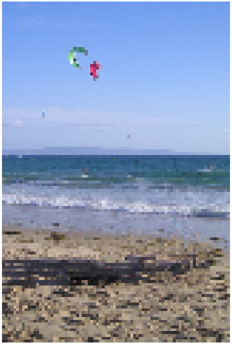
Loopy Belief Propagation (3/4)



LBP convergence:

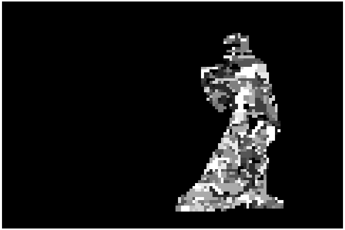
- Random initialization;
- 13 inpainted image in input;
- 25 iterations;
- resolution=80×120.



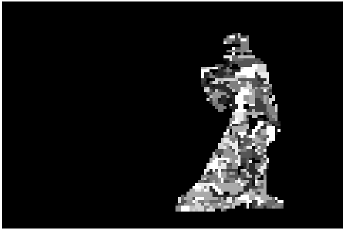


Loopy Belief Propagation (4/4)

Initialization



Iterated minimization



Loopy Belief Propagation (4/4)

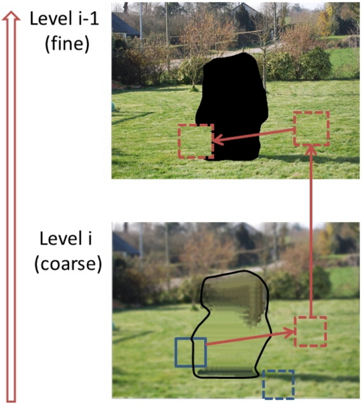
Initialization

Iterated minimization

Super-resolution (1/1)

From the LR patch corresponding to the HR patch having the highest priority:

- We look for its best neighbour in the LR inpainted image;
- Only the best candidate is kept;
- The corresponding HR patch is simply deduced.
- Its pixel values are then copied into the unknown parts of the current HR patch.



Results (1/6)

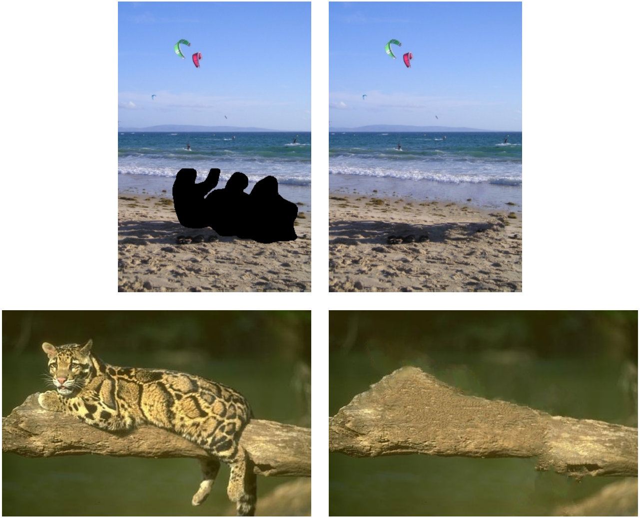
Resolution=440 × 600

Results (1/6)

Resolution=440 × 600

Results (2/6)

Results (3/6)



Results (4/6)

Results (5/6)

Results (6/6)





Much more results on the link:

http://people.irisa.fr/Olivier.Le_Meur/publi/2013_TIP/index.html