Jan Spirik – Jan Zatyik *

IMAGE EXTRAPOLATION USING SPARSE METHODS

Image extrapolation is the specific application in image processing. You have to extrapolate the image for example when you want to process the given image piecewise. When the border patches are incompleted you must extrapolate them to the given size. Nowadays,some basic extrapolations, e.g. linear, polynomial etc. are used. The advanced methods are presented in this paper. We are using the algorithms that are based on finding the sparse solutions in underdetermined systems of linear equations. Three algorithms are presented for image extrapolation. First one is the K-SVD algorithm. K-SVD is the algorithm that trains a dictionary which allows the optimal sparse representation. Second one is Morphological Component Analysis (MCA) which is based on Independent Component Analysis (ICA). The last is the Expectation Maximization (EM) algorithm. This algorithm is statistics-based. These three algorithms for image extrapolation are compared on the real images.

Keywords: *Image extrapolation, sparse, K-SVD, MCA, EM.*

1. Introduction

Image processing using underdetermined systems of linear equations is a very promising approach in different applications. In the previous years methods for image denoising based on image splitting were used [1]. Today, we could denoise the image via underdetermined systems of linear equations [2]. Except image denoising we use these systems for image compression [3], deblurring [4] or image inpainting [5] and for many other applications.

We will use the algorithms originally designed for image inpainting to image extrapolation. Image inpainting is the framework for filling in the known holes in the input image. Classical inpainting methods assume filling the holes from different directions. If we want to use these algorithms for image extrapolation we must redefine or modify some parameters or parts of the original algorithms.

If we are using underdetermined systems of linear equations we want to find the sparse solution. The sparse solution is the one which contains only a few nonzero coefficients. The basics of sparse signal representations are introduced in [3]. All methods for finding sparse solutions start from the basic problem (P_0) which is defined as follows:

$$(P_0): \min_x \|\boldsymbol{x}\|_0 \quad \text{subject to } \boldsymbol{y} = \boldsymbol{D}\boldsymbol{x} \tag{1}$$

where y is the known signal (e.g. image) we want to reconstruct, $\boldsymbol{D}$ is the dictionary which consists of n atoms ("elementary signals") in columns and m rows which denote the length of atoms. An unknown sparse solution represents amounts of each atom in the original signal. We define the norm of a vector:

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=0}^{N} |\boldsymbol{x}_i|^p\right)^{1/p} \tag{2}$$

and $0 < p < \infty$. If $0 < p < 1$, it is actually not the norm, it is the quasinorm. The quasinorm is similar to the standard norm, but it does not satisfy the triangle inequality.

The optimization problem (1) is defined in ℓ_0. When $\|\boldsymbol{x}\|_0 \ll n$ for $\boldsymbol{x} \in \mathbb{R}^n$, we could say that $\boldsymbol{x}$ is sparse. Searching for the sparse solution without any algorithm in ℓ_0 norm is NP-hard problem. The problem can be redefined to:

$$(P_0): \min_x \|\boldsymbol{x}\|_0 \quad \text{subject to } \|\boldsymbol{D}\boldsymbol{x} - \boldsymbol{y}\|_2 \leq \varepsilon \tag{3}$$

where ε is the error of solution. There are various algorithms dealing with this problem. The overview of these algorithms is presented in [6].

2. Applied algorithms

A) K-SVD algorithm

In most applications via underdetermined systems, we assume fixed dictionary $\boldsymbol{D}$. K-SVD algorithm is the algorithm for adaptive learning the dictionary that allows sparse representation of the input signal. It is called K-SVD because of the SVD (Singular Value Decomposition) algorithm which is always performed K times where K is the count of columns in $\boldsymbol{D}$. K-SVD algorithm is the generalization of the K-means algorithm [7]. K-SVD is the iterative algorithm containing two basic steps. The first one is finding the

* **Jan Spirik, Jan Zatyik**
Department of Telecommunications, Brno University of Technology, Brno, Czech Republic,
E-mail: jan.spirik@phd.feec.vutbr.cz, zatyik.jan@phd.feec.vutbr.cz

sparse solution **x** for the actual dictionary. The second step is the dictionary update. Before application of these steps we must initialize the dictionary. The initialized dictionary must have all atoms (all columns) ℓ_2 normalized. We could use, for example, DCT dictionary or otherwise a random matrix for the initial dictionary.

We could formulate the K-SVD as:

$$\min_{\boldsymbol{D},\boldsymbol{X}} \left\{ \left\| \boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X} \right\|_F^2 \right\} \text{ subject to } \forall i, \left\| \boldsymbol{x}_i \right\|_0 \le T_0 , \tag{4}$$

where $\boldsymbol{Y}$ is a matrix that contains training samples $\{\boldsymbol{y}_i\}_{i=1}^N$ in columns, $\boldsymbol{X}$ is a matrix that contains the corresponding coefficients, T_0 is the error of the representation and $_F$ denotes the Frobenius norm. The Frobenius norm is defined as:

$$\left\| \boldsymbol{D} \right\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} \left| d_{i,j} \right|^2} , \tag{5}$$

In the first stage we assume $\boldsymbol{D}$ fixed. We can express the penalty term as:

$$\left\| \boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X} \right\|_F^2 = \sum_{i=1}^{N} \left\| \boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{x} \right\|_2^2 , \tag{6}$$

Then we could divide (6) into problems:

$$i = 1, 2, ..., N \quad \min_{s_i} \left\{ \left\| \boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{x}_i \right\|_2^2 \right\}$$

$$\text{subject to } \left\| \boldsymbol{x}_i \right\|_0 \le T_0 , \tag{7}$$

These problems could be solved by the known algorithms for finding sparse solutions in underdetermined systems of linear equations.

In the second stage the dictionary $\boldsymbol{D}$ is updated. We assume $\boldsymbol{D}$ and $\boldsymbol{X}$ fixed. In each step only one atom (column in $\boldsymbol{D}$) $\boldsymbol{d}_k$ and corresponding row $\boldsymbol{x}_T^k$ in $\boldsymbol{X}$ will be updated. Based on previous statement we define sets ω_k that consists of indexes of vectors $\{\boldsymbol{y}_i\}$ which use the atom $\boldsymbol{d}_k$, in fact where $\boldsymbol{x}_T^k$ is nonzero:

$$\omega_k = \left\{ i \middle| 1 \le i \le N, \boldsymbol{x}_T^k(i) \ne 0 \right\} . \tag{8}$$

Then we define matrices of errors $\boldsymbol{E}_k$ that express error for all N samples with missing k-atom:

$$\boldsymbol{E}_k = \boldsymbol{Y} - \sum_{j \ne k} \boldsymbol{d}_j \boldsymbol{x}_T^j . \tag{9}$$

With these conditions we could rewrite (4) as:

$$\begin{aligned} &\left\| \boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X} \right\|_F^2 = \left\| \boldsymbol{Y} - \sum_{j \ne k} \boldsymbol{d}_j \boldsymbol{x}_T^j \right\|_F^2 = \\ &= \left\| \left(\boldsymbol{Y} - \sum_{j \ne k} \boldsymbol{d}_j \boldsymbol{x}_T^j \right) - \boldsymbol{d}_j \boldsymbol{x}_T^k \right\|_F^2 = \left\| \boldsymbol{E}_k - \boldsymbol{d}_j \boldsymbol{x}_T^k \right\|_F^2 \end{aligned} \tag{10}$$

For simplification we apply sets ω_k to the matrices $\boldsymbol{E}_k$ in that manner we choose only columns that correspond with ω_k and we get $\boldsymbol{E}_k^R$. Using the algorithm SVD we divide $\boldsymbol{E}_k^R$. into:

$$\boldsymbol{E}_k^R = \boldsymbol{U}\Delta\boldsymbol{V}^T \tag{11}$$

The last step in the dictionary update stage is only replacing the current atom (column) of the dictionary $\boldsymbol{d}_k$ with the first column of the matrix $\boldsymbol{U}$ and the corresponding row of coefficients $\boldsymbol{x}_T^k$ with the first column of matrix $\boldsymbol{V}$ multiplied by .

Extrapolation via K-SVD is realized as training the dictionary on the image patches with the chosen size (for example 8 × 8, 16 × 16 pixels). The count of atoms in the dictionary is chosen by user too. For the image extrapolation we add random pixels in the place where we want to extrapolate the image. We also define the binary matrix $\boldsymbol{M}$ which extends the original image. Ones in the binary mask represent the known pixels in the image and zeros indicate the missing ones. We can reformulate problem (P_0) as:

$$\min_x \left\| x \right\|_0 \text{ subject to } \boldsymbol{M}\boldsymbol{D}\boldsymbol{x} = \boldsymbol{y}. \tag{12}$$

If we perform K-SVD we apply $\boldsymbol{M}$ to the dictionary $\boldsymbol{D}$, but in the extrapolated part of the image we use the whole trained dictionary. For better results we completely overlap all the patches from the image [3].

One of the biggest disadvantages of K-SVD algorithm is the dependency on the length of extrapolation with the patch size. We must train the dictionary for the patches of minimum (e + 1) × × (e + 1) pixels, where e is the length of extrapolation in pixels. It means that the computational efficiency is dependent quadratically on the length of extrapolation. This patch size is chosen because of the overlapping patches. If we do not fulfill this limit, the extrapolation will be unsuccessful, the end of the extrapolated part will be black.

On the other hand, the big advantage of K-SVD extrapolation is that the training part for one image is done only once. You can extrapolate the image to different directions, but the trained dictionary remains the same. Only the reconstruction will be done. For example, if you train the dictionary for patches 8 × 8 pixels, you should perform extrapolation to any directions with the extrapolation length of maximum 7 pixels.

B) Morphological Component Analysis

The basics of Morphological Component Analysis (MCA) are introduced in [8]. We assume that the input picture is a linear combination of two independent parts: cartoon and texture. This idea comes from Independent Component Analysis (ICA). We use two incoherent dictionaries. The dictionary $\boldsymbol{D}_t$ allows sparse decomposition of the part of the image $\boldsymbol{y}_t$ (we assume the picture as 1-D vector) that contains only texture. On the opposite side, the dictionary $\boldsymbol{D}_c$ allows sparse decomposition of the part of the image $\boldsymbol{y}_c$ that contains only cartoon. We could formulate the problem as:

$$\left\{ \boldsymbol{x}_t^{opt}, \boldsymbol{x}_c^{opt} \right\} = \arg\min_{\{x_t, x_c\}} \left\| \boldsymbol{x}_t \right\|_0 + \left\| \boldsymbol{x}_c \right\|_0 \text{ subject to}$$

$$\boldsymbol{y} = \boldsymbol{D}_t \boldsymbol{x}_t + \boldsymbol{D}_c \boldsymbol{x}_c . \tag{13}$$

Because of the problem for finding the numerical solution we must reformulate equation (13) as:

$$\left\{\boldsymbol{x}_t^{opt},\boldsymbol{x}_c^{opt}\right\} = \arg\min_{\{x_t,x_c\}}\left\|\boldsymbol{x}_t\right\|_1 + \left\|\boldsymbol{x}_c\right\|_1 + \\ + \lambda\left\|\boldsymbol{y} - \boldsymbol{D}_t\boldsymbol{x}_t - \boldsymbol{D}_c\boldsymbol{x}_c\right\|_2^2 + \gamma TV\left\{\boldsymbol{D}_c\boldsymbol{x}_c\right\}, \tag{14}$$

where the third term of the equation reflects the reconstruction error and TV is the abbreviation for total variation penalty. We use the TV for recovering piecewise smooth objects with pronounced edges, when applied to the cartoon layer [9]. The total variation is essentially the ℓ_1 norm of the gradient.

For the purpose of image extrapolation, we must apply the binary mask $\boldsymbol{M}$ the same way as for K-SVD to the image:

$$\left\{\boldsymbol{x}_t^{opt},\boldsymbol{x}_c^{opt}\right\} = \arg\min_{\{x_t,x_c\}}\left\|\boldsymbol{x}_t\right\|_1 + \left\|\boldsymbol{x}_c\right\|_1 + \\ + \lambda\left\|\boldsymbol{M}\left(\boldsymbol{y} - \boldsymbol{D}_t\boldsymbol{x}_t - \boldsymbol{D}_c\boldsymbol{x}_c\right)\right\|_2^2 + \gamma TV\left\{\boldsymbol{D}_c\boldsymbol{x}_c\right\}. \tag{15}$$

Ones in the binary mask express the known pixels in the image and zeros indicate the missing ones.

If we assume that $\boldsymbol{y}_t = \boldsymbol{D}_t\boldsymbol{x}_t$ and we know the texture part of the image $\boldsymbol{y}_t$, we can calcute the sparse vector $\boldsymbol{x}_t$ as $\boldsymbol{x}_t = \boldsymbol{D}_t^{+}\boldsymbol{x}_t + \boldsymbol{r}_t$, where $\boldsymbol{r}_t$ is a residual vector in the null-space of $\boldsymbol{D}_t$ and $^{+}$ denotes Moore-Penrose pseudoinverse. We apply the same properties to the cartoon part. For simplification we assume $\boldsymbol{r}_t = \boldsymbol{r}_c = 0$. Then we could minimize the problem as:

$$\min_{\{y_t,y_c\}}\left\|\boldsymbol{D}_t^{+}\boldsymbol{y}_t\right\|_1 + \left\|\boldsymbol{D}_c^{+}\boldsymbol{y}_c\right\|_1 + \\ + \lambda\left\|\boldsymbol{M}\left(\boldsymbol{y} - \boldsymbol{y}_t - \boldsymbol{y}_c\right)\right\|_2^2 + \gamma TV\left\{\boldsymbol{y}_c\right\}. \tag{16}$$

When we implement the algorithm we first choose the threshold factor, number of iterations and the parameters λ and γ. The last two parameters should be constant during the iterations or they could be descended. Then we initialize the cartoon part of the image as $\boldsymbol{y}_c = \boldsymbol{y}$ and the texture part as $\boldsymbol{y}_t = 0$. After that we iterate the algorithm to the stopping rule (threshold or number of iterations). In the iteration part of the algorithm we first fix the texture part of the image $\boldsymbol{y}_t$ and we update the cartoon part $\boldsymbol{y}_c$ and then vice-versa. After these two stages we apply the TV penalization.

The important step in the algorithm is the choice of the dictionaries. For texture part we should use local DCT, Gabor or wavelet packets transforms and for the cartoon part wavelet, curvelet, ridgelet, contourlet and many other transforms [9]. MCA extrapolation significantly depends on the choice of dictionaries. For different types of pictures (real-life pictures, cartoons, computer images) a different combination of dictionaries is more vital. The extrapolation is performed for every single direction separately. You cannot do any temporary calculations. You only perform the reconstructions phase, there is no training phase as for K-SVD. All the conditions for successful extrapolation are included in the algorithm.

C) Expectation-Maximization algorithm

Expectation-Maximization algorithm (EM) is based on mathematical statistics. It is iterative method for estimation of maximum-likelihood. In this case we use this method for finding sparse solutions based on penalized maximum-likelihood estimator estimation [5]. In terms of statistics inpainting is a problem of estimation from incomplete data sets. EM algorithm is used for computation of sparse vector x from the previous iteration. We reconstruct the whole image, not only the missing parts. As well as MCA the efficiency of EM depends on the choice of the dictionary. One of the biggest advantages is that we could make a dictionary from several different transformations, i.e. utilize underdetermination of the system.

At first we must define the penalized maximum-likelihood estimator [5]:

$$\hat{\boldsymbol{x}} = \arg\min_{x}\frac{1}{2\sigma^2}\left\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{x}\right\|_2^2 + \lambda\Psi(\boldsymbol{x}) \tag{17}$$

where σ^2 is a variance of zero-mean additive white Gaussian noise, $\lambda > 0$ is regularization parameter and Ψ is a penalty function. The penalty function must be non-negative, continuous, even-symmetric and non-decreasing function. But it must not be necessarily convex in $\mathbb{R}^{+}$. We often use ℓ_1-norm penalty for Ψ. We divide the input image into two sets: $\boldsymbol{y}_0$ which contains the known pixels from the image and $\boldsymbol{y}_m$ which contains the unknown pixels from the image. The incomplete observation makes impossible calculate (17) at once. We use EM algorithm for iterative reconstruction of missing data so that we use (17) for computation of new estimations until the convergence is achieved.

As already mentioned above the EM algorithm is iterative process. It consists of two steps: Expectation (E) step and Maximization (M) step. In the first step the conditional expectation of the penalized log-likelihood of complete data y_0 and actual parameters is calculated. The particular expectations can be expressed as the conditional expected squared values of the missing data:

$$\boldsymbol{y}_i^t = \exists\left[y_i \middle| \boldsymbol{D}, \boldsymbol{y}_0, \boldsymbol{x}^t, \sigma^{2t}\right] = \\ = \begin{cases} \boldsymbol{y}_i, & for\ observed\ data, i \in \boldsymbol{I}_0 \\ \left(\boldsymbol{D}\boldsymbol{x}^t\right)_i & for\ missing\ data, i \in \boldsymbol{I}_m \end{cases} \tag{18}$$

$$\exists\left[\boldsymbol{y}_i^2 \middle| \boldsymbol{D}, \boldsymbol{y}_0, \boldsymbol{x}^t, \sigma^{2t}\right] = \\ = \begin{cases} \boldsymbol{y}_i^2, & for\ observed\ data, i \in \boldsymbol{I}_0 \\ \left(\boldsymbol{D}\boldsymbol{x}^t\right)_i^2 + \sigma^{2t} & for\ missing\ data, i \in \boldsymbol{I}_m \end{cases} \tag{19}$$

We could express the estimation at t iteration as:

$$\boldsymbol{y}^t = \boldsymbol{y}_{act} + (\boldsymbol{I} - \boldsymbol{M})\boldsymbol{D}\boldsymbol{x}^t = \boldsymbol{D}\boldsymbol{x}^t + \left(\boldsymbol{y}_{act} - \boldsymbol{M}\boldsymbol{D}\boldsymbol{x}^t\right) \tag{20}$$

where $\boldsymbol{y}_{act} = \boldsymbol{M}\boldsymbol{y}$ and $\boldsymbol{M}$ is the binary mask with the same properties as above.

In $\boldsymbol{M}$ step we maximize the penalized function with the missing observations using the estimates from the $\boldsymbol{E}$ step at t iteration:

$$\sigma^{2t+1} = \frac{1}{n}\left[\sum_{i\in I_0}\left(\boldsymbol{y}_i - \left(\boldsymbol{D}(\boldsymbol{x})^{\gamma}\right)_i\right)^2 + \left(n - n_0\sigma^{2t}\right)\right], \tag{21}$$

where n_0 is the number of observed pixels.

These two steps are repeated t times until the convergence is reached:

$$\hat{\sigma}^2 \rightarrow \frac{1}{n_0}\sum_{i\in I_0}\left(\boldsymbol{y}_i - \left(\boldsymbol{D}(\hat{\boldsymbol{x}})^t\right)_i\right)^2, \tag{22}$$

that is the maximum-likelihood estimate of the noise variance inside the mask with the observed pixels. The properties of convergence depend on the structure of the dictionary. The algorithm behaves differently when the dictionary is the basis, tight frame or union of several incoherent orthogonal dictionaries [5].

The EM extrapolation is in properties very similar to MCA extrapolation. There is no training phase. The result depends on the dictionary if it is basis or union of basis or etc.

The biggest disadvantage of EM algorithm is that the reconstruction is performed on the whole image, not only on planned parts. It means the extrapolated image is blurred. The blur effect depends on the length of extrapolation. You can improve the EM reconstruction result by some technique presented in [4].

3. Experimental results

We choose the excerpt of the Barbara image (Fig. 1) to compare the efficiency of the presented algorithms. The algorithms were compared for different length and type of extrapolation. For the purposes of experimental measurements, the binary mask $\boldsymbol{M}$ to the known part of the image was applied. It is because of possibility to

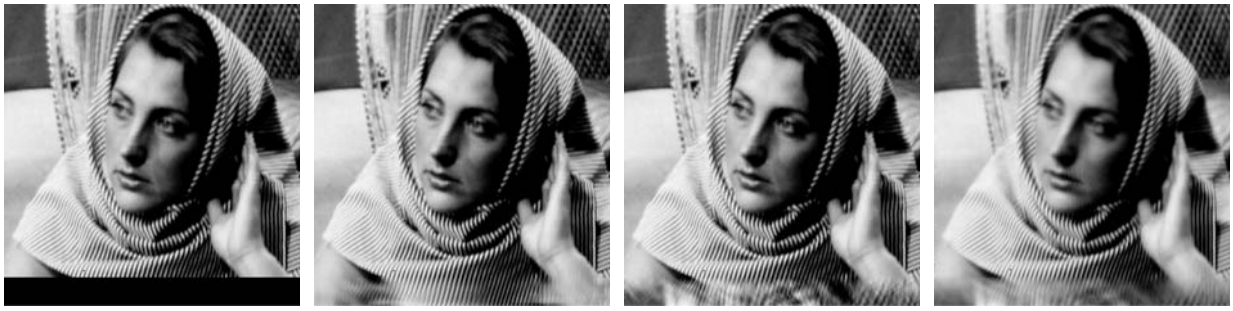
Fig. 1 Used part of Barbara image 256 × 256 pixels

(a) *(b)* *(c)* *(d)*

Fig. 2 Image extrapolation of 7 pixels: (a) original image with applied mask, (b) reconstruction via K-SVD: PSNR 36.3 dB, SSIM 0.9981, (c) reconstruction via MCA: PSNR 29.0 dB, SSIM 0.9946, (d) reconstruction via EM: PSNR 27.7 dB, SSIM 0.9445

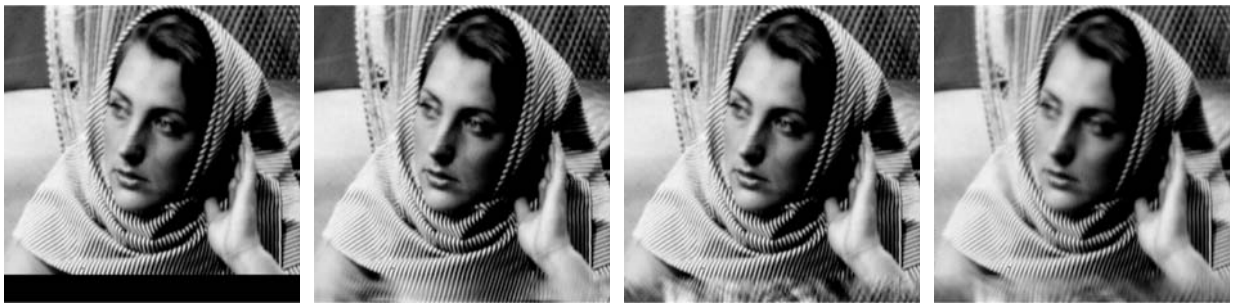
(a) *(b)* *(c)* *(d)*

Fig. 3 Image extrapolation of 15 pixels: (a) original image with applied mask, (b) reconstruction via K-SVD: PSNR 29.5 dB, SSIM 0.9846, (c) reconstruction via MCA: PSNR 25.3 dB, SSIM 0.9765, (d) reconstruction via EM: PSNR 24.4 dB, SSIM 0.9264



(a) (b) (c) (d)

Fig. 4 Image extrapolation of 25 pixels: (a) original image with applied mask, (b) reconstruction via K-SVD: PSNR 25.6 dB, SSIM 0.9612, (c) reconstruction via MCA: PSNR 21.9 dB, SSIM 0.9428, (d) reconstruction via EM: PSNR 21.9 dB, SSIM 0.8984



(a) (b) (c) (d)

Fig. 5 Image extrapolation of 25 pixels into 2 directions: (a) original image with applied mask, (b) reconstruction via K-SVD: PSNR 22.1 dB, SSIM 0.9045, (c) reconstruction via MCA: PSNR 20.9 dB, SSIM 0.8981, (d) reconstruction via EM: PSNR 21.4 dB, SSIM 0.8541

measure the efficiency of the algorithms. The binary mask will be always rectangular. We choose the length e of extrapolation and then which direction we want to extrapolate. We could make extrapolation for more directions at once. We fill in e columns or rows (it depends on the directions of extrapolation) with zeros in $\boldsymbol{M}$.

The standard PSNR and SSIM are applied for measurement reconstruction quality. We are introducing some examples of extrapolation in the following figures (Figs. 2 – 5). For all extrapolations using MCA and EM algorithms two incoherent dictionaries were used: curvelets [10] and undecimated discrete wavelet transform [11].

4. Conclusion

The principles of three different algorithms for image extrapolation that are using sparse solutions have been presented. The results for different lengths and directions of extrapolation were shown in the figures. Our subjective perception of the quality of the reconstructed images corresponded with the objective measurement of quality: PSNR and SSIM. The best algorithm from the presented algorithms was K-SVD algorithm. The quality was especially in good reconstruction of the texture parts in the image. There is also only a small blur effect. Some modifications can be made in the original K-SVD, for example, non-square patches and so on. These and more modifications will be presented in the future work.

Acknowledgement

The described research was performed in laboratories supported by the SIX project; the registration number CZ.1.05/2.1.00/03.0072, the operational program Research and Development for Innovation.

References

[1] SIMAK, B., KARPF, M.: Salt & Pepper Noise Impact on Image Coding Based on Image Splitting, *Communications - Scientific Letters of the University of Zilina*, vol. 2, pp. 16–20, 2000.

[2] ELAD, M., AHARON, M.: Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries, *Image Processing, IEEE Transactions on,* vol. 15, no. 12, pp. 3736–3745, dec. 2006.

[3] ELAD, M.: *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing, Springer,* 2010.

[4] ZHANG, H., ZHANG, Y.: Sparse Representation Based Iterative Incremental Image Deblurring, in *Proceedings of the 16th IEEE intern. conference on Image processing,* Piscataway, 2009, ICIP'09, pp. 1285–1288, IEEE Press.

[5] FADILI, M. J., STARCK, J. L., MURTAGH, F.: Inpainting and Zooming Using Sparse Representations, *The Computer Journal,* 2007.

[6] SPIRIK, J.: Algorithms for Computing Sparse Solutions, *in Proc. of the 17th conference STUDENT EEICT vol. 3,* 2011, pp. 123–127.

[7] AHARON, M., ELAD, M., BRUCKSTEIN, A. M.: K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representations, *IEEE Transactions on Signal Processing,* vol. 54, pp. 4311–4322, 2006.

[8] STARCK, J. L., ELAD, M., DONOHO, D. L.: Redundant Multiscale Transforms and Their Application for Morphological Component Analysis, *Advances in Imaging and Electron Physics,* vol. 132, 2004.

[9] ELAD, M., STARCK, J. L., QUERRE P., DONOHO, D. L.: Simultaneous Cartoon and Texture Image Inpainting Using Morphological Component Analysis (MCA), *Applied and Computational Harmonic Analysis,* vol. 19, no. 3, pp. 340–358, 2005.

[10] CANDES, E., DEMANET, L., DONOHO, D. L., YING, L.: Fast Discrete Curvelet Transforms, *Multiscla Modeling & Simulation,* vol. 5, no. 3, pp. 861–899, 2006.

[11] GYAOUROVA, A., KAMATH, CH., FODOR, I. K.: Undecimated Wavelet Transforms for Image De-Noising, *Tech. Rep.*, 2002.