Livrable D2.2 of the PERSEE project : Analyse/Synthese de Texture

Josselin Gautier, Christine Guillemot, Vincent Jantet, Olivier Le Meur, Luce Morin, Mehmet Turkan

▶ **To cite this version:**

Josselin Gautier, Christine Guillemot, Vincent Jantet, Olivier Le Meur, Luce Morin, et al.. Livrable D2.2 of the PERSEE project : Analyse/Synthese de Texture. 2011, pp.57. hal-00773172

HAL Id: hal-00773172
https://hal.science/hal-00773172

Submitted on 11 Jan 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Projet PERSEE
Texture analysis and synthesis
n ANR-09-BLAN-0170

Livrable **D2.2** *1/11/2011*

Del 2.2: Analyse/Synthese de Texture

Josselin	GAUTIER	INRIA
Christine	GUILLEMOT	INRIA
Vincent	JANTET	INRIA
Olivier	LE MEUR	INRIA
Luce	MORIN	INSA
Mehmet	TURKAN	INRIA

Contents

Chapter 1

Executive Summary

This document describes texture analysis tools developed in the framework of the PERSEE project to be used both for prediction in 2D and 3D video codecs as well as in the rendering step of the 3D coding/transmission/rendering chain. Texture synthesis and image inpainting techniques have shown remarkable progresses over the past years and appear to be promising directions for a number of image processing problems: compression, loss concealment, and hanlding of disocclusions in the context of virtual view synthesis for free viewpoint navigation.

Existing methods include parametric techniques based on a parameterized texture model, non parametric methods, or graph-cut techniques which aim at preserving structural continuity. In parametric and non-parametric synthesis techniques, the Probability Density Function (PDF) of a given texture example is approximated and sampled to generate similar texture samples. Parametric synthesis approaches approximate the PDF of the texture using a compact model. Non-parametric approaches typically formulate the texture synthesis problem based on Markov Random Fields (MRF). Another family of techniques based on sparse approximations has also apperaed recently for texture synthesis in contexts of prediction and inpainting.

In this deliverable, new methods of prediction and inpainting inspired from examplar-based techniques which have been developed in the PERSEE project are described. The key contributions underlying these new methods concern the introduction of new priority terms for processing the patches to be completed as well as the approximation methods which are based on neighbor embedding techniques rather than simple template matching. These methods have been used both for prediction and inpainting. The resulting prediction technique (described in Chapter 2) has been integrated and evaluated in a still image codec. A solution based on a particular neighbor embedding, which can be seen as an average of template mathcing, has also been integrated as in an HEVC-based video codec, and assessed in this context. A new examplar-based inpainting method has been developed and is presented in Chapter 3.

These solutions have then been used for handling disocclusions which occur when extrapolating virtual view from one single 2D+depth input view or when interpolating virtual views from multiple input video plus depth views. Towards this goal, the priority computation has been extended in prder to take into account the depth information available in a 3D (multi-view plus depth) application. The resulting depth-aided inpainting technique, with the renedering results obtained, is described in Chapter 3Dinpainting.

In virtual view synthesis, these methods for disocclusion handling in general follow a step which consists in projecting the input view unto the virtual view point. Both processes - inpainting and projection - can thus benefit from a coupling of the two steps. This was the scope of

another direction axis of the project. This led to the introduction of a joint projection/inpainting technique which is described in Chapter 5.

These results have been published in international journals [1], [2] and in international conferences [3], [4], [5].

Chapter 2

Texture synthesis for image prediction

Closed-loop intra prediction is a key component of image compression algorithms. For example, in H.264/AVC, there are two intra-frame prediction types called Intra-16x16 and Intra-4x4 [6]. Each 4x4 block is predicted from prior encoded pixels of spatially neighboring blocks. In addition to the so-called "DC" mode which consists in predicting the entire 4x4 block from the mean of neighboring pixels, eight directional prediction modes have been specified. The prediction is done by simply "propagating (interpolating)" the pixel values along the specified direction. This approach is suitable in the presence of contours when the directional mode chosen corresponds to the orientation of the contour. However, it fails in more complex textured areas.

An alternative spatial prediction algorithm based on **template matching** (TM) has been described in [7]. A so-called template is formed by previously encoded pixels in the close neighborhood of the block to be predicted. The best match between the template of the block to be predicted and candidate texture patches of same shape as the template, within a causal search window, allows finding the predictor of the block to be predicted. The approach in [7] has later been improved in [8] by averaging multiple template matching predictors, including larger and directional templates, resulting in up to 15% rate saving when included into H.264/AVC intra-prediction.

A prediction algorithm based on sparse approximation techniques has been introduced in [9]. The goal of sparse approximation techniques (e.g., matching pursuit (MP) [10] or orthogonal matching pursuit (OMP) [11]) is to look for a linear expansion approximating the analyzed signal in terms of functions chosen from a large and redundant set (dictionary). In [9], image prediction is regarded as a problem of signal extension from noisy data taken from a causal neighborhood. The sparse signal approximation is run with a set of *masked* basis functions, the masked samples correspond to the location of the pixels to be predicted. The basic principle of the approach is to first search for a linear combination of masked basis functions which best approximates known sample values in a causal neighborhood (template), and keep the same linear combination of basis functions to approximate the unknown sample values in the block to be predicted. The stopping criterion (which is the energy of the residue signal) is computed on the known region. To compute it on the causal neighborhood would lead to a residue of small energy, however, this residue might take large values in the block to be predicted. The number of *atoms* (basis functions) selected in order to minimize a given criterion (i.e., the energy of the residue or a rate-distortion cost function) on the block to be predicted is transmitted to the

decoder. The decoder similarly runs the algorithm with the *masked* basis functions by taking the previously decoded neighborhood as the known support. The number of atoms selected by the encoder can thus be used by the decoder as a stopping criterion.

Dictionaries formed by pre-defined DCT or DFT waveforms are used in [9]. These dictionaries are particularly well suited for predicting periodic texture patches. However, the prediction fails for more complex non-periodic structures with discontinuities. The authors in [12] have considered instead **dynamic** and **locally adaptive** dictionaries formed by atoms derived from texture patches present in a causal neighborhood of the block to be predicted. The principle of the approach is thus to first search for a linear combination of image patches (stacked as columns in a matrix $\mathbf{A}$ called the dictionary) which best approximates the template, and then keep the same linear combination of co-locoted pixel values to approximate the unknown sample values in the block to be predicted. The use of several templates selected by the prediction or compression algorithm according to either a mean squared error (MSE) or a rate-distorion (RD) criterion allows improving the prediction or compression performance. This method can be seen as a generalization of template matching. The TM is indeed a special case when only one iteration is used with a weighting coefficient equal to 1. This method is referred to here as **sparse prediction** (SP).

In [13], the image prediction problem is placed in a context of data dimensionality reduction using the non-negative matrix factorization (NMF) [14] algorithm. Given a fixed non-negative dictionary (or basis functions), the underlying main idea is to first obtain an NMF representation of the support region (template) and keep the same representation parameters to approximate the unknown pixel values in the block to be predicted. This approach leads to very good compression performance (a PSNR gain of up to 3 dB when compared with the TM and the SP methods). Furthermore, it does not require sending extra information (as the iteration number in the SP method) but on the other hand, its high computational complexity, because of the update equations and the size of the fixed dictionary, makes difficult the use of several candidate templates.

In PERSEE, we have pushed further the study of image prediction based on dimensionality reduction algorithms by considering the two methods: NMF and locally linear embedding (LLE) [15]. We first introduce a sparsity constraint, which has already been included in the LLE algorithm implicitly, into the NMF based prediction method. In the classical method [16], the sparseness constraints in the NMF algorithm has been achieved by keeping the ℓ_2 norm unchanged and setting the corresponding ℓ_1 norm to the desired sparsity. Here, we propose an ℓ_0 norm sparsity constraint by initially selecting $k, k = 1...K$, patches to be used in the prediction algorithm. The main idea explored here is again to search for a linear combination to approximate the known pixel values in the template, and then keep the same weighting coefficient to estimate the pixels to be predicted as a linear combination of the co-located pixels in the k neareast neigbouring (k-NN) patches (for both the NMF and the LLE based methods). The parameter k controls the sparsity of the data approximation.

The proposed image prediction methods have been evaluated in a complete image codec, both strict and relaxed sparsity constraints (controlled by the parameter k). A detailed analysis has been carried out on the prediction quality and the encoding efficiency in comparison to the sparse approximation based method [12] and also H.264/AVC intra image prediction. The results obtained show a significant improvement in terms of the rate-distotion gain of the reconstructed image, after coding and decoding the prediction residue (up to 2 dB and up to 1 dB) when compared to H.264/AVC intra prediction and to the sparse prediction respectively. Further RD

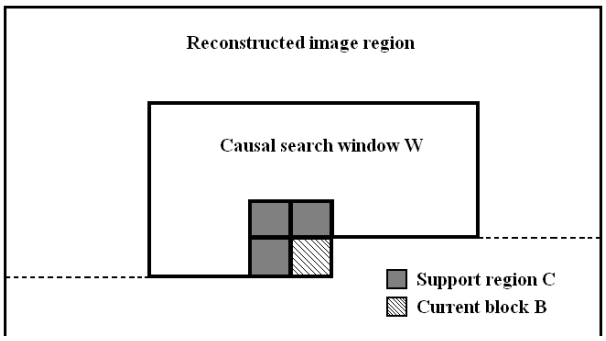


Figure 2.1: $\mathbf{C}$ is the approximation support, $\mathbf{B}$ is the current block to be predicted, $\mathbf{W}$ is the window from which texture patches are taken to construct the dictionary to be used for the prediction of $\mathbf{B}$, and $\mathrm{S} \triangleq \mathbf{C} \cup \mathbf{B}$.

gains have been achieved with LLE based prediction method by relaxing the sparsity constraints and setting the parameter k to 100, or more, at the expense however of extra complexity.

2.1 Neighbor embedding techniques

Let S denote a region in the image containing a block $\mathbf{B}$ of $n \times n$ pixels and its causal neighborhood $\mathbf{C}$ used as approximation support (template) as shown in Fig. 2.1. In Fig. 2.1, the region S contains 4 blocks, hence is of size $\mathrm{N} = 4n^2$ pixels, for running the prediction algorithm. However, we will see later (see Fig. 2.2) that different forms of template $\mathbf{C}$ can be considered. In any case, in the region S, there are known samples (the template $\mathbf{C}$) and unknown samples (the values of the pixels of the block $\mathbf{B}$ to be predicted). The principle of the prediction approach is to first search for a good approximation for the known pixels in $\mathbf{C}$ and keep the same procedure to approximate the unknown pixel values in $\mathbf{B}$.

Let $\mathbf{A}$ denote a so-called dictionary represented by a matrix of dimension $\mathrm{N} \times \mathrm{M}$, where $\mathrm{N} \leq \mathrm{M}$. In all prediction methods presented below (template matching, sparse prediction, NMF, and LLE), the dictionary $\mathbf{A}$ is constructed by stacking the luminance values of all patches (having the same geometric shape as S) in a given causal search window $\mathbf{W}$ in the reconstructed image region as shown in Fig. 2.1. The use of a causal window guarantees that the decoder can construct exactly the same dictionary.

We denote $\mathbf{A}_c$ as the compacted dictionary matrix obtained by masking the rows of the matrix $\mathbf{A}$ which correspond to the spatial location of the pixels of the area $\mathbf{B}$. The compacted matrix $\mathbf{A}_c$ is of size $3n^2 \times \mathrm{M}$. The N sample values of the region S are stacked in a vector $\mathbf{b}$ (by assuming the unknown values in $\mathbf{B}$ are equal to zero). The vector $\mathbf{b}$ is also compacted in the vector $\mathbf{b}_c$ of $3n^2$ values, the vector $\mathbf{b}_c$ thus corresponds to the support region (template) $\mathbf{C}$ organized in a vector form. Let us define another matrix $\mathbf{A}_t$ (of size $n^2 \times \mathrm{M}$) as the corresponding spatial dictionary obtained by masking the rows of $\mathbf{A}$ with respect to the spatial location of the pixels of the area $\mathbf{C}$, and assume that $\mathbf{b}_t$ and $\hat{\mathbf{b}}_t$ represent the actual and the predicted pixel values of the block $\mathbf{B}$ respectively.

2.1.1 Template Matching

The TM algorithm searches for the best match between the template $\mathbf{b}_c$ and possible candidate patches (of the same shape as the template) stored in the columns of $\mathbf{A}_c$ which are taken from the causal search window. The problem of template matching can thus be formulated as the search for the index j of the atom $\mathbf{a}_{c_j}$ in the dictionary $\mathbf{A}_c$ (i.e., that is stacked in the j^{th} column of the matrix $\mathbf{A}_c$) which will minimize the distance d_j as

$$j_{opt} = \underset{j\in\{1...\mathrm{M}\}}{\arg\min}\,\{\mathrm{d}_j\} \quad \text{where} \quad \mathrm{d}_j = ||\mathbf{b}_c - \mathbf{a}_{c_j}||_2^2, \;\; j = 1...\mathrm{M}. \tag{2.1}$$

The signal $\mathbf{b}_t$ is then simply predicted by copying the pixel values of the candidate $\mathbf{a}_{t_{j_{opt}}}$ (i.e., that is stacked in the j^{th} column of the matrix $\mathbf{A}_t$) as $\hat{\mathbf{b}}_t = \mathbf{a}_{t_{j_{opt}}}$.

Sparse Prediction (SP)

The principle of the sparse prediction approach is to first search for a linear combination of basis functions (atoms, or texture patches) taken from the compacted dictionary $\mathbf{A}_c$, which best approximates the template $\mathbf{C}$, and then keep the same linear combination (the same indexes of atoms $\mathbf{a}_{t_j}$ in $\mathbf{A}_t$ and the same weights) to approximate the unknown pixel values in $\mathbf{B}$. Sparse representation algorithm aims at solving the approximate minimization as

$$\mathbf{x}_{opt} = \min_{\mathbf{x}} \|\mathbf{b}_c - \mathbf{A}_c\mathbf{x}\|_2^2 \quad \text{subject to} \quad \min \|\mathbf{x}\|_0. \tag{2.2}$$

In practice, one actually seeks an approximate solution which satisfies

$$\min \left\{ \|\mathbf{x}\|_0 \;:\; \|\mathbf{b}_c - \mathbf{A}_c\mathbf{x}\|_2^2 \leq \rho \right\}, \tag{2.3}$$

for some $\rho \geq 0$ characterizing an admissible reconstruction error. Matching pursuit (MP) [10] and orthogonal matching pursuit (OMP) [11] algorithms have been introduced as heuristic methods to find approximate solutions to the above problem with tractable complexity.

In the sequel, the OMP algorithm is used and it proceeds as follows. At the first iteration $\mathbf{x}_0 = 0$ and an initial residual vector $\mathbf{r}_0 = \mathbf{b}_c - \mathbf{A}_c\mathbf{x}_0 = \mathbf{b}_c$ is computed. At iteration k, the algorithm identifies the atom $\mathbf{a}_{c_{j_k}}$ in $\mathbf{A}_c$ having the maximum correlation with the approximation error. Let $\mathbf{A}_c^k$ denote the compacted matrix containing all the atoms selected in the previous iterations. One then projects $\mathbf{b}_c$ onto the subspace spanned by the columns of $\mathbf{A}_c^k$, i.e., one solves

$$\min_{\mathbf{x}_k} \; \|\mathbf{b}_c - \mathbf{A}_c^k\mathbf{x}_k\|_2^2, \tag{2.4}$$

and the coefficient vector at the k^{th} iteration is given as

$$\mathbf{x}_k = (\mathbf{A}_c^{k\mathrm{T}}\mathbf{A}_c^k)^{-1}\mathbf{A}_c^{k\mathrm{T}}\mathbf{b}_c = \mathbf{A}_c^{k+}\mathbf{b}_c, \tag{2.5}$$

where $\mathbf{A}_c^{k+}$ represents the pseudo-inverse of $\mathbf{A}_c^k$. Notice that here $\mathbf{x}_k$ is a vector of coefficients. All the coefficients assigned to the selected atoms are recomputed at each iteration.

The algorithm at the encoder runs until a pre-specified iteration number K is reached by keeping track of the MSE or the RD cost function values obtained for the block to be predicted $\mathbf{b}_t$, and finally selects the number of *atoms* (i.e., image patches used) which minimizes the considered criterion, leading to an "optimum" sparse vector denoted $\mathbf{x}_{opt}$. The value of the number of atoms

Table 2.1: Sparse Approximation based Image Prediction using OMP.

Input: $\mathbf{A}_c$, $\mathbf{A}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$, K
Output: Predicted values of unknowns $\hat{\mathbf{b}}_t$
Initialization: $k = 0$, $\mathbf{x}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{b}_c$, $\mathbf{A}_c^0 = [\,]$, $\mathbf{A}_t^0 = [\,]$
do until $k = K$
$k = k + 1$;
$j_k = \arg\max_j \left|\mathbf{A}_c^{\mathrm{T}} \mathbf{r}_{k-1}\right|$;
$\mathbf{A}_c^k = \mathbf{A}_c^{k-1} \cup \{\mathbf{a}_{c_{j_k}}\}$ and $\mathbf{A}_t^k = \mathbf{A}_t^{k-1} \cup \{\mathbf{a}_{t_{j_k}}\}$;
$\mathbf{x}_k = \mathbf{A}_c^{k+} \mathbf{b}_c$;
$\mathbf{r}_k = \mathbf{b}_c - \mathbf{A}_c^k \mathbf{x}_k$;
$\mathbf{p}_k = \mathbf{A}_t^k \mathbf{x}_k$;
end **do**
Select the optimum k^* minimizing the selected criterion;
Set $\hat{\mathbf{b}}_t = \mathbf{p}_{k^*}$

used is then transmitted to the decoder which can similarly search for the sparse approximation of the template (support region) with the signalled number of atoms. Note that template matching can be seen as a particular case of the sparse approximation where only one iteration is used and the weighting coefficient is equal to 1.

The prediction signal $\hat{\mathbf{b}}_t$ is then calculated by multiplying the matrix $\mathbf{A}_t$ by $\mathbf{x}_{opt}$ as $\hat{\mathbf{b}}_t = \mathbf{A}_t \mathbf{x}_{opt}$. Note here that the columns (atoms) of $\mathbf{A}_t$ are first to be normalized with the norm of corresponding columns of $\mathbf{A}_c$, i.e., $\mathbf{a}_{t_m} = \mathbf{a}_{t_m} / \|\mathbf{a}_{c_m}\|_2 \ \forall m$, and then the atoms of $\mathbf{A}_c$ are normalized in ℓ_2 norm. The complete sparse approximaton based image prediction algorithm is summarized in Table 2.1.

2.1.2 Average Template Matching (ATM)

ATM can be seen as a simple extension of TM where several patches are combined with uniform weights. The patch selection process generally proceeds by choosing K number of most similar patches to the template in the source image. After obtaining K nearest neighboring patches $\Psi_{\hat{q}_k}, k = 1...\mathrm{K}$, one uniformly combines colocated pixels of these patches in order to estimate the fill-in region values as follows

$$\Psi_{\hat{p}}^u = \frac{1}{K} \sum_{k=1}^{K} \Psi_{\hat{q}_k}^u. \tag{2.6}$$

2.1.3 Locally Linear Embedding

LLE [15] is a constrained optimization algorithm which solves the nonlinear data dimensionality reduction problem. LLE tries to find a global transformation of the high-dimensional coordinates into low-dimensional ones by exploiting the locally-linear characteristics of the high-dimensional data. It aims at preserving the local linear structure of the high-dimensional data in the lower-dimensional space.

Formally, given M high-dimansional data points consisting of N-real valued vectors $\mathbf{X}_i$, the LLE method consists of the following steps:

1. It first identifies K nearest neighbors $\mathbf{X}_j$ per data point $\mathbf{X}_i$ for all $i, i \neq j$. The usual measure is the Euclidean distance;

2. It then searches for the weights $\mathbf{W}_{i,j}$, so that each data point is approximated by its neighbors. The algorithm thus aims at minimizing the cost function,

$$E(\mathbf{W}) = \sum_i \left\| \mathbf{X}_i - \sum_j \mathbf{W}_{i,j} \mathbf{X}_j \right\|_2^2. \tag{2.7}$$

   The weights $\mathbf{W}_{i,j}$ represent the contribution of the j^{th} data point to the reconstruction of the i^{th} point, and they are constrained to sum to one, i.e., $\sum_j \mathbf{W}_{i,j} = 1, \forall i$. Each data point $\mathbf{X}_i$ can only be reconstructed from its neighbors (i.e., $\mathbf{W}_{i,j} = 0$ if the data point $\mathbf{X}_j$ does not belong to the neighbors of $\mathbf{X}_i$). The optimal weights satisfying these constraints are obtained by solving the constrained least squares problem per data point $\mathbf{X}_i$ as

$$E_i(\mathbf{W}_j) = \left\| \sum_j \mathbf{W}_j (\mathbf{X}_i - \mathbf{X}_j) \right\|_2^2 \quad \text{subject to} \quad \sum_j \mathbf{W}_j = 1; \tag{2.8}$$

3. Finally, an embedding cost function is minimized in order to obtain the low-dimensional global internal coordinates $\mathbf{Y}_i$ by fixing the weights $\mathbf{W}_{i,j}$ as

$$\zeta(\mathbf{Y}) = \sum_i \left\| \mathbf{Y}_i - \sum_j \mathbf{W}_{i,j} \mathbf{Y}_j \right\|_2^2. \tag{2.9}$$

   Please see [15] for more information.

The LLE method (steps 1 and 2) is here first applied on the template $\mathbf{C}$. One thus searches for an approximation of the template by a linear combination of its k-NN ($k = 1...K$) patches within the search window and then keeps the same weighting coefficients in the linear combination of the co-located pixels in order to estimate the unknown values of the block to be predicted. In terms of LLE based method, (2.2) can be re-written as

$$\mathbf{x}_{opt} = \min_{\mathbf{x}} \|\mathbf{b}_c - \mathbf{A}_c \mathbf{x}\|_2^2 \quad \text{subject to} \quad \sum_m \mathbf{x}_m = 1 \quad \text{and} \quad \min \|\mathbf{x}\|_0. \tag{2.10}$$

A sparsity constraint has been imposed (also implicitly by the LLE) onto the problem by choosing k closest patches to $\mathbf{b}_c$ in Euclidean space. At iteration k, suppose that $\mathbf{A}_c^k$ denotes the submatrix which contains the selected k atoms (texture patches) in $\mathbf{A}_c$. The algorithm thus tries to solve the constrained minimization as

$$\min_{\mathbf{x}_k} \left\| \mathbf{b}_c - \mathbf{A}_c^k \mathbf{x}_k \right\|_2^2 \quad \text{subject to} \quad \sum_m \mathbf{x}_{k_m} = 1. \tag{2.11}$$

and the optimal weighting coefficients in $\mathbf{x}_k$ are computed as

$$\mathbf{x}_k = \frac{\mathbf{D}_k^{-1} \mathbf{1}}{\mathbf{1}^{\mathrm{T}} \mathbf{D}_k^{-1} \mathbf{1}} \tag{2.12}$$

Table 2.2: Locally Linear Embedding based Image Prediction.

Input: $\mathbf{A}_c$, $\mathbf{A}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$, K
Output: Predicted values of unknowns $\hat{\mathbf{b}}_t$
Initialization: $k = 0$, $\mathbf{A}_c^0 = [\,]$, $\mathbf{A}_t^0 = [\,]$
do until $k = K$
$k = k + 1$;
$j_k = \arg\min_j \{d_j\}$ where $d_j = \left\|\mathbf{b}_c - \mathbf{a}_{c_j}\right\|_2^2$;
$\mathbf{A}_c^k = \mathbf{A}_c^{k-1} \cup \{\mathbf{a}_{c_{j_k}}\}$ and $\mathbf{A}_t^k = \mathbf{A}_t^{k-1} \cup \{\mathbf{a}_{t_{j_k}}\}$;
$\mathbf{A}_c \leftarrow \mathbf{A}_c \backslash \{\mathbf{a}_{c_{j_k}}\}$ and $\mathbf{A}_t \leftarrow \mathbf{A}_t \backslash \{\mathbf{a}_{t_{j_k}}\}$;
Calculate local covariance matrix $\mathbf{D}_k$ of $\mathbf{A}_c^k$;
Solve $\mathbf{D}_k \mathbf{x}_k = \mathbf{1}$ for $\mathbf{x}_k$;
$\mathbf{x}_k = \mathbf{x}_k / sum(\mathbf{x}_k)$;
$\mathbf{p}_k = \mathbf{A}_t^k \mathbf{x}_k$;
end **do**
Select the optimum k^* minimizing the selected criterion;
Set $\hat{\mathbf{b}}_t = \mathbf{p}_{k^*}$

where $\mathbf{D}_k$ denotes the local covariance matrix (i.e., in reference to $\mathbf{b}_c$) of the selected k patches in $\mathbf{A}_c^k$, and $\mathbf{1}$ is the column vector of ones. In practice, instead of an explicit inversion of the matrix $\mathbf{D}_k$, the linear system of equations $\mathbf{D}_k \mathbf{x}_k = \mathbf{1}$ is solved, then the weights are rescaled so that they sum to one.

Here also, the algorithm at the encoder keeps track of the MSE or the RD cost function values obtained for the block to be predicted and finally selects the number k of *used patches* which minimizes the considered criterion, leading to an "optimum" sparse vector denoted $\mathbf{x}_{opt}$. The value of the number k is then transmitted to the decoder which can similarly search for the same LLE approximation of the template with the signalled number. The predicted signal $\hat{\mathbf{b}}_t$ is then calculated by multiplying the dictionary $\mathbf{A}_t$ by $\mathbf{x}_{opt}$ as $\hat{\mathbf{b}}_t = \mathbf{A}_t \mathbf{x}_{opt}$. The complete LLE based image prediction algorithm is summarized in Table 2.2.

2.1.4 Non-negative Matrix Factorization

NMF is a subspace approximation algorithm which finds a suitable low-rank representation of the high-dimensional data. In many data analysis tasks the data to be analysed is non-negative, and classical tools, e.g., principle component analysis (PCA) [17], can not guarantee to maintain the non-negativity property of the original data in the low-dimensional space. NMF is a recent method for obtaining such a non-negative representation, which is indeed helpful for physical interpretation of the results in many data analysis tools such as dimensionality reduction, data mining, and noise removal.

Formally, given a non-negative matrix $\Phi \in \mathbb{R}^{\mathrm{N} \times \mathrm{L}}$ and a positive integer $\mathrm{M} < \min\{\mathrm{N}, \mathrm{L}\}$, the aim is to find non-negative matrix factors $\mathbf{A} \in \mathbb{R}^{\mathrm{N} \times \mathrm{M}}$ and $\Psi \in \mathbb{R}^{\mathrm{M} \times \mathrm{L}}$, such that $\Phi \approx \mathbf{A}\Psi$ where the reconstruction error between Φ and $\mathbf{A}\Psi$ is minimized. The most widely used cost function is the squared Euclidean distance, i.e.,

$$\min_{\mathbf{A}, \Psi} \left[\frac{1}{2} \left\| \Phi - \mathbf{A}\Psi \right\|_{\mathrm{F}}^2 \right] \quad \text{subject to} \quad \mathbf{A} \geq 0 \text{ and } \Psi \geq 0, \tag{2.13}$$

and NMF algorithm is optimized with the *multiplicative* update equations as

$$\Psi_{a\mu} \leftarrow \Psi_{a\mu} \frac{\left(\mathbf{A}^{\mathrm{T}}\Phi\right)_{a\mu}}{\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\Psi\right)_{a\mu}+\varepsilon}, \qquad \mathbf{A}_{ia} \leftarrow \mathbf{A}_{ia} \frac{\left(\Phi\Psi^{\mathrm{T}}\right)_{ia}}{\left(\mathbf{A}\Psi\Psi^{\mathrm{T}}\right)_{ia}+\varepsilon}, \tag{2.14}$$

where ε is a small constant equal to 10^{-9} to avoid divide by zero in the update equations. Here, $a\mu$ (or ia) represents the a^{th} (or i^{th}) row and μ^{th} (or a^{th}) column elements of the corresponding matrices respectively. In the standard algorithm, the matrices $\mathbf{A}$ and Ψ are initialized with random non-negative values, and the Euclidean distance $||\Phi - \mathbf{A}\Psi||_{\mathrm{F}}^2$ is decreasing under the above alternating update rules as proven in [14].

The product $\mathbf{A}\Psi$ is called an NMF of Φ, and the underlying features of Φ are extracted as basis vectors in $\mathbf{A}$, which can then be used in data analysis tools instead of Φ. Note that $\Phi \approx \mathbf{A}\Psi$ can be rewritten as $\mathbf{b} \approx \mathbf{A}\mathbf{x}$ where $\mathbf{b}$ and $\mathbf{x}$ represent the corresponding columns of Φ and Ψ respectively. One can interpret that a column vector $\mathbf{b}$ of Φ is approximated by a linear combination of the columns of dictionary $\mathbf{A}$, weighted by the column vector $\mathbf{x}$ of Ψ.

Let us now come back to the problem of image prediction. Given a fixed non-negative dictionary $\mathbf{A} \in \mathbb{R}^{\mathrm{N}\times\mathrm{M}}$ formed by texture patches as explained above, and the (non-negative) data vector $\mathbf{b} \in \mathbb{R}^{\mathrm{N}}$, the underlying basic idea is to first obtain an NMF representation $\mathbf{x}$ of the support region $\mathbf{C}$ and keep the same representation parameters (i.e., weighting coefficients in $\mathbf{x}$) to approximate the unknown pixel values in the block to be predicted $\mathbf{B}$. The non-negativity constraints are satisfied for both $\mathbf{A}$ and $\mathbf{b}$, similarly for $\mathbf{A}_c$ and $\mathbf{b}_c$, since the values in the spatial domain range between 0 and 255.

Assuming the compacted dictionary $\mathbf{A}_c$ is fixed for the data vector $\mathbf{b}_c$, the NMF formulation for the representation vector $\mathbf{x}$ of $\mathbf{b}_c$ can be written as

$$\min_{\mathbf{x}} \left[\frac{1}{2}\left\|\mathbf{b}_c - \mathbf{A}_c\mathbf{x}\right\|_2^2\right] \quad \text{subject to} \quad \mathbf{x} \geq 0, \tag{2.15}$$

and the *multiplicative* update equation for $\mathbf{x}$ becomes

$$x_a \leftarrow x_a \frac{\left(\mathbf{A}_c^{\mathrm{T}}\mathbf{b}_c\right)_a}{\left(\mathbf{A}_c^{\mathrm{T}}\mathbf{A}_c\mathbf{x}\right)_a+\varepsilon}, \;\; a = 1...\mathrm{M}. \tag{2.16}$$

Here again, a sparsity constraint can be imposed onto the prediction problem as in (2.2) by limiting the number of non-zero coefficients in $\mathbf{x}$ as

$$\mathbf{x}_{opt} = \min_{\mathbf{x}} \left[\frac{1}{2}\left\|\mathbf{b}_c - \mathbf{A}_c\mathbf{x}\right\|_2^2\right] \quad \text{subject to} \quad \mathbf{x} \geq 0 \text{ and } \min\|\mathbf{x}\|_0. \tag{2.17}$$

The selection is done by choosing k-NN ($k = 1...K$) patches which are close to $\mathbf{b}_c$ in Euclidean distance. At iteration k, the algorithm identifies k atoms $\left[\mathbf{a}_{c_{j_1}}, ..., \mathbf{a}_{c_{j_k}}\right]$ in $\mathbf{A}_c$ which are close to $\mathbf{b}_c$, and let $\mathbf{A}_c^k$ denote the compacted matrix containing all the atoms selected in the k^{th} iteration. One then solves,

$$\min_{\mathbf{x}_k} \left[\frac{1}{2}\left\|\mathbf{b}_c - \mathbf{A}_c^k\mathbf{x}_k\right\|_2^2\right] \quad \text{subject to} \quad \mathbf{x}_k \geq 0 \tag{2.18}$$

by updating the non-negative and randomly initialized elements of $\mathbf{x}_k$ as

$$x_{k_a} \leftarrow x_{k_a} \frac{\left(\mathbf{A}_c^{k\,\mathrm{T}}\mathbf{b}_c\right)_a}{\left(\mathbf{A}_c^{k\,\mathrm{T}}\mathbf{A}_c^k\mathbf{x}_k\right)_a+\varepsilon}, \;\; a = 1...k. \tag{2.19}$$

Table 2.3: Non-negative Matrix Factorization based Image Prediction.

Input: $\mathbf{A}_c$, $\mathbf{A}_t$, $\mathbf{b}_c$, $\mathbf{b}_t$, K, T
Output: Predicted values of unknowns $\hat{\mathbf{b}}_t$
Initialization: $k = 0$, $\mathbf{A}_c^0 = [\,]$, $\mathbf{A}_t^0 = [\,]$
do until $k = K$
$k = k + 1$;
$j_k = \arg\min_j \{d_j\}$ where $d_j = \left\|\mathbf{b}_c - \mathbf{a}_{c_j}\right\|_2^2$;
$\mathbf{A}_c^k = \mathbf{A}_c^{k-1} \cup \{\mathbf{a}_{c_{j_k}}\}$ and $\mathbf{A}_t^k = \mathbf{A}_t^{k-1} \cup \{\mathbf{a}_{t_{j_k}}\}$;
$\mathbf{A}_c \leftarrow \mathbf{A}_c \backslash \{\mathbf{a}_{c_{j_k}}\}$ and $\mathbf{A}_t \leftarrow \mathbf{A}_t \backslash \{\mathbf{a}_{t_{j_k}}\}$;
initialize $\mathbf{x}_k$ and $t = 0$;
iterate until $t = T$, or change in $\mathbf{x}_k$ is small
$t = t + 1$;
$\mathbf{x}_k \leftarrow \mathbf{x}_k \otimes \left(\mathbf{A}_c^{k\mathrm{T}} \mathbf{b}_c\right) \oslash \left(\mathbf{A}_c^{k\mathrm{T}} \mathbf{A}_c^k \mathbf{x}_k + 10^{-9}\right)$;
end **iterate**
$\mathbf{p}_k = \mathbf{A}_t^k \mathbf{x}_k$;
end **do**
Select the optimum k^* minimizing the selected criterion;
Set $\hat{\mathbf{b}}_t = \mathbf{p}_{k^*}$

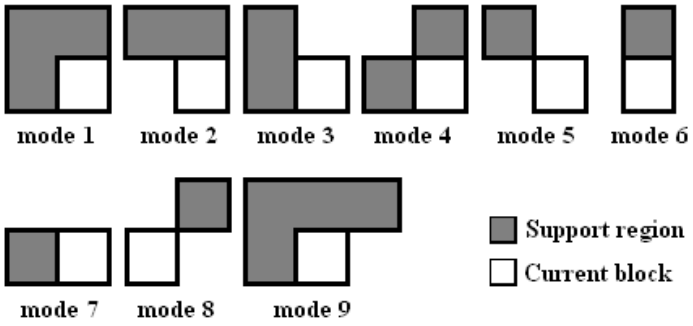


Figure 2.2: Nine possible modes for the MSE/RD optimized support (template) selection.

The algorithm at the encoder, as in the prediction approach based on sparse approximations, keeps track of the MSE or the RD cost function values obtained for the block to be predicted and finally selects the number k of *texture patches* which minimizes the considered criterion, leading to an "optimum" sparse vector denoted $\mathbf{x}_{opt}$. The value of the number k is then transmitted to the decoder which can similarly search for the same NMF approximation of the template with the signalled number of atoms. In order to obtain the optimum weighting vector $\mathbf{x}_{opt}$, the update equation is iterated until a pre-defined iteration number T is reached, or the total change in the elements of the vector $\mathbf{x}$ is very small between two consecutive iterations $t-1$ and t, $t = 1...T$. The predicted signal $\hat{\mathbf{b}}_t$ is then calculated by multiplying the dictionary $\mathbf{A}_t$ by $\mathbf{x}_{opt}$ as $\hat{\mathbf{b}}_t = \mathbf{A}_t \mathbf{x}_{opt}$. The complete NMF based image prediction algorithm is summarized in Table 2.3.

2.2 Performance illustration for still image coding

2.2.1 Encoder Structure

The proposed NMF and LLE based spatial image prediction methods have been assessed in a still image (or intra frame) coding/compression scheme by comparing it to the TM and sparse prediction, on one hand as well as to the intra prediction approach based on H.264/AVC on the other hand. The performance assessment is done both in terms of prediction quality and PSNR/bit-rate efficiency.

In order to initialize the prediction process, the top 4 rows and left 4 columns of blocks of size 4x4 are predicted with H.264/AVC intra modes. Once a block has been predicted with the respective prediction method, the DCT transformed residue is quantized, zig-zag scanned, and encoded with an algorithm similar to JPEG. In this coding structure, a uniform quantization matrix with $\Delta = 16$ is weighted by a quality factor. The quality factor (q_f) is increased from 10 to 90 with a step size of 10, and the corresponding weight $\mathbf{w}_{q_f}$ is calculated by means of the following equation

$$\mathbf{w}_{q_f} = \begin{cases} 50/q_f & \text{if } q_f \leq 50 \\ 2 - 0.02q_f & \text{if } q_f > 50 \end{cases} . \tag{2.20}$$

Image blocks are processed in a raster scan order, and the reconstructed image is obtained by adding the quantized residue to the prediction. A skip mode (the corresponding flag is arithmetically encoded) has also been included to the encoder to avoid coding the blocks of prediction residue in which all the transformed and quantized coefficients are zero.

Several forms of support regions (templates) are considered as shown in Fig. 2.2. The optimum template is selected among nine possible modes. The best mode, as well as the iteration number k for SP, and the number of used patches (also referred to here as k) for LLE and NMF, is selected according to two criteria:

- Minimization of the prediction MSE on the unknown block $\mathbf{b}_{\mathrm{t}}$ in order to observe the impact on the prediction quality;
- Minimization of an RD cost function of the form $\mathbf{J}_{RD} = \mathbf{D} + \lambda\mathbf{R}$ when the prediction is used in the coding scheme in order to observe the impact on the encoding efficiency. Here, $\mathbf{D}$ is the distortion (i.e., the sum of squared error (SSE)) of the reconstructed block (after adding the quantized residue to the prediction), and $\mathbf{R}$ is the residue encoding cost which is estimated as $\mathbf{R} = \gamma_0\mathbf{M}'$ for low bit-rate compression [18] with $\mathbf{M}'$ being defined as the number of non-zero quantized DCT coefficients, and for DCT basis $\gamma_0 = 6.5$. By considering a uniform scalar quantizer with a quantization step Δ (where the deadzone is equal to 2Δ), the relation between the optimum Lagrange multiplier λ_{opt} and the quantization step Δ is given by [19]

$$\lambda_{opt} = \frac{3\Delta^2}{4\gamma_0}. \tag{2.21}$$

The optimization is done in two steps, i.e., first for the selection of iteration number k in the case of SP, or the optimal number of used patches, k, in the case of the NMF and LLE method, and then for the selection of template mode type. In terms of encoding, one needs to add the coding cost of side information, here it is the k value and the template type. This information is signalled to the decoder using Huffman codes.

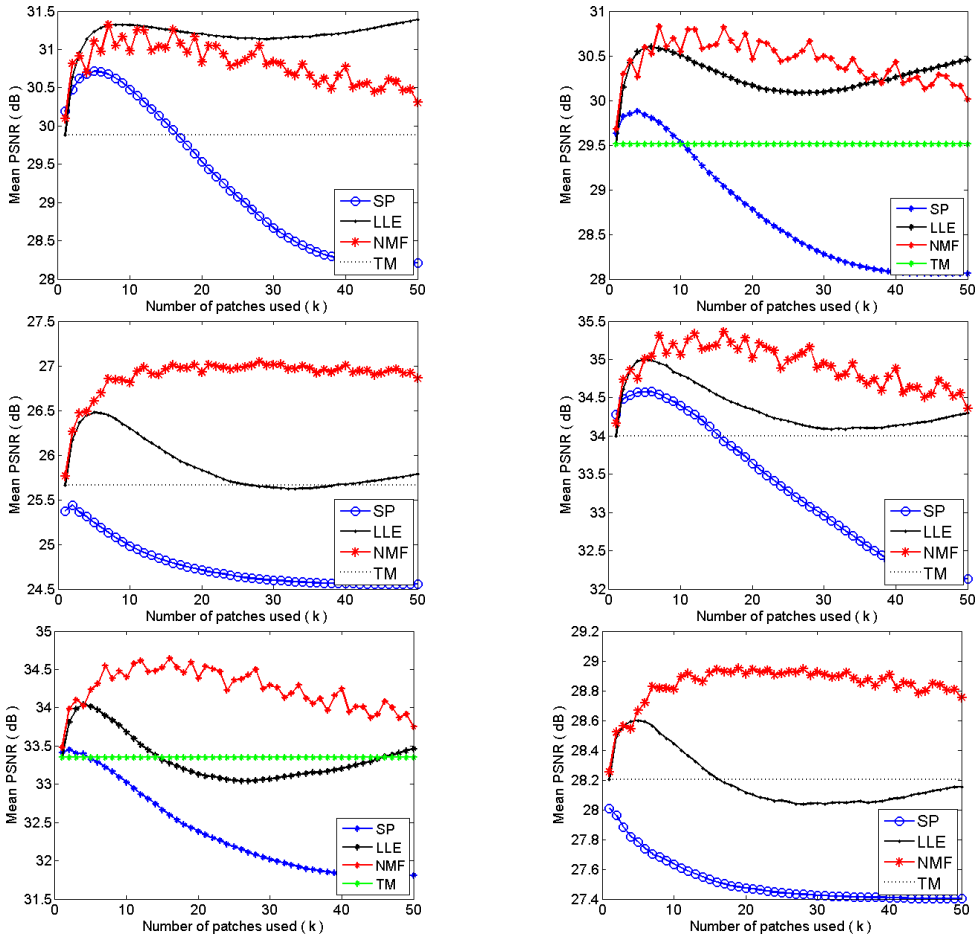


Figure 2.3: Mean approximation PSNR versus sparsity performance of (left) low ($q_f = 90$), (middle) medium ($q_f = 50$), and (right) high ($q_f = 10$) quantization noise corrupted (top-row) Barbara (512x512) and (bottom-row) Foreman (CIF) images using SP, NMF, and LLE based image prediction algorithms with 4x4 block size. Template mode 1 is used as shown in Fig. 2.2 and $k \in [1, 50]$.

2.2.2 Impact of Sparsity Constraint and Quantization Noise

In the prediction methods decsribed above, the dictionary $\mathbf{A}$ is constructed by stacking the luminance values of all patches (having the same geometric shape as S) in a given causal search region $\mathbf{W}$ in the reconstructed image region. When using those methods which are mainly based on the approximations of a template, a good approximation of the template does not necessarily lead to a good approximation of the unknown pixel values as the template and the unknown region pixels may have different characteristics. Furthermore, in an image coding context, at the decoder side the template information $\mathbf{C}$, as well as the patches stored in the dictionary $\mathbf{A}$, may not be clean depending on the prediction quality and the residue signal quantization. Therefore, it is crucial to analyse and to optimize the prediction quality of the unknown pixel values as a function of sparsity and the quantization noise. This sub-section briefly analyses the effect of the sparsity constraint on SP, LLE, and NMF based prediction methods in the case where the image signals are corrupted by various levels of quantization noise.

Fig. 2.3 shows the mean prediction PSNR obtained for Barbara (512x512) and Foreman (CIF) images with varying sparsity contraints in the case where the image signals (i.e., blocks) are simply corrupted by a low ($q_f = 90$), a medium ($q_f = 50$), and a high ($q_f = 10$) quantization noise with the quantization scheme as described in Sec. 2.2.1. The quality of predicted signal, in terms of mean PSNR, is significantly improved (up to 1.3 dB) with the NMF based prediction method when compared to TM and SP, even in the presence of a high quantization noise. In the LLE and SP based prediction methods, the mean prediction PSNR has its maximum when $k \in [1, 10]$ whereas NMF based prediction has its maximum PSNR when $k \in [1, 20]$. Note here that the mean performance curves of NMF based method look noisy because of the randomization of the weighting coefficients. For each simulation point $k, k = 1...50$, the vector $\mathbf{x}_k$ is initialized with the same seed of Mersenne twister random number generator.

Another interesting point might be further relaxing the sparsity constraint, which will increase the computational complexity however, for the sake of a complete analysis of the proposed prediction methods' global characteristics as a function of sparsity and the quantization noise. Fig. 2.4 shows the mean prediction PSNR obtained for Barbara and Foreman images for $k \in [51, 100]$. The mean prediction performance for the LLE based method increases with the number of atoms (patches) used in the algorithm. However, adding more elements into the NMF model has negative effect on the performance as in the SP method. Here SP based method stops iterating for $k > 48$ since the template mode 1 has 48 known pixels in $\mathbf{C}$ for 4x4 block size. At iteration $k = 48$, the OMP algorithm constructs a complete orthogonal basis for the known values in $\mathbf{C}$ and the algorithm stops.

2.2.3 Experimental Setup

Three test images are chosen for the simulations as shown in Fig. 2.5. Foreman (the first frame in the CIF sequence) can be seen as the image which has mainly diagonal edges and smooth regions. Barbara (512×512) contains a combination of smooth and textural regions as well as the edges. Finally, Roof (512×512) contains highly textural regions. Two sets of tests have been carried out. The first one makes use of approximations with sparsity constraints. The number of iterations k, or equally the number of used patches considered in the k-NN search, is varied from 1 to 8, i.e., $K = 8$. The optimal k value in terms of the selected criterion (i.e., either the MSE or the RD) is then signalled to the decoder. The second set of experiments relax the sparsity constraint and take $k = K = 100$.

In the NMF based prediction method, the maximum iteration number for update equation (2.19) is set to 100, i.e., $T = 100$, (see also Table 2.3). However, we have experimentally observed that the total change in the elements of $\mathbf{x}_k$ gets very small for $t > 5$ in the smooth areas, and for $t > 50$ in the edgel areas. In the highly textural areas the algorithm iterates upto 100 iterations.

Fig. 2.6 shows the configuration of the search window for 4x4 block size that is used in the simulations reported in this paper. All possible unique image patches in the search region are extracted to construct the dictionary matrix $\mathbf{A}$.

2.2.4 Prediction performance with MSE criterion

Fig. 2.7 and Fig. 2.8 demonstrate visually the prediction performance for the test images, a textural region in Barbara and an edgel region in Foreman images respectively. The optimization criterion is the minimization of the prediction MSE, and $q_f = 10$ (i.e., at low bit-rates). The sparsity constraint has been used for these experiments, k is varied from 1 to 8 for SP, NMF and

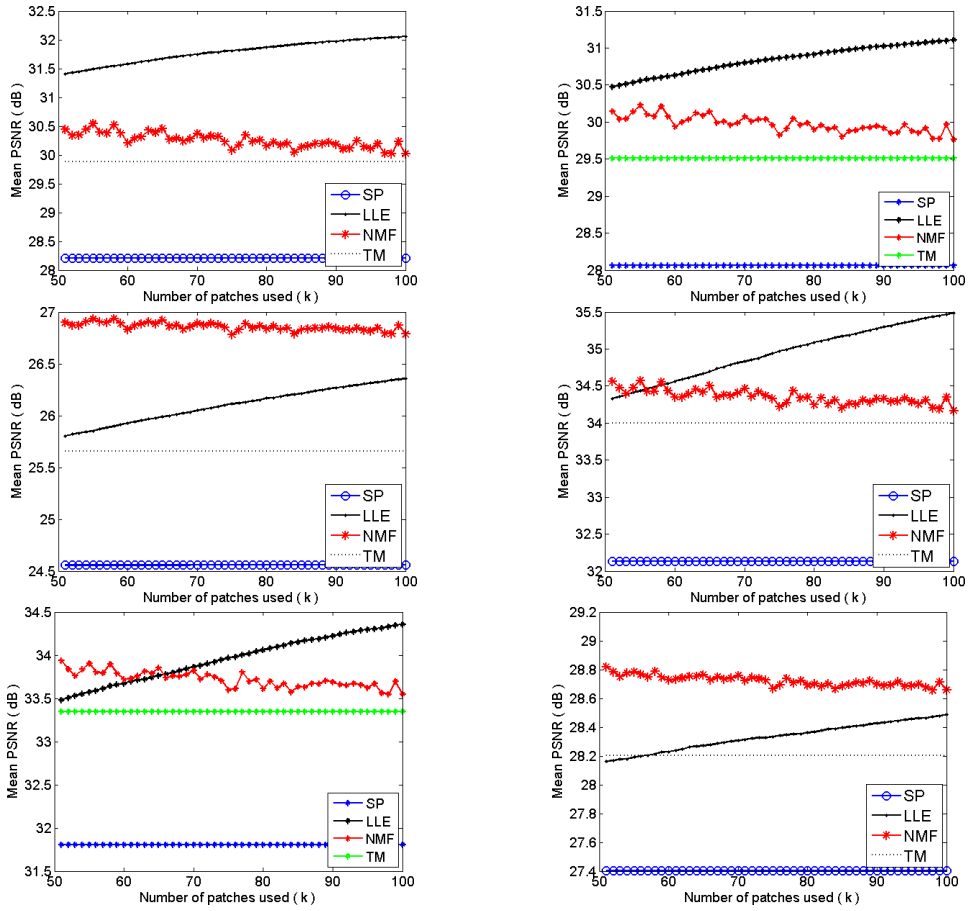


Figure 2.4: Mean approximation PSNR versus sparsity performance of (left) low ($q_f = 90$), (middle) medium ($q_f = 50$), and (right) high ($q_f = 10$) quantization noise corrupted (top-row) Barbara (512x512) and (bottom-row) Foreman (CIF) images using SP, NMF, and LLE based image prediction algorithms with 4x4 block size. Template mode 1 is used as shown in Fig. 2.2 and $k \in [51, 100]$.

Figure 2.5: The test images. (a) Foreman (CIF), (b) Barbara (512x512), and (c) Roof (512x512).

LLE based methods.

The quality of the predicted signal, in terms of visual quality, is significantly improved by the LLE, and even further by the NMF based methods when compared to the SP and H.264/AVC

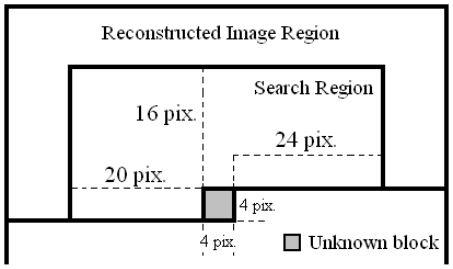


Figure 2.6: The configuration of the search region for 4x4 block size. All possible unique image patches are extracted from the search region in order to construct dictionary **A**.

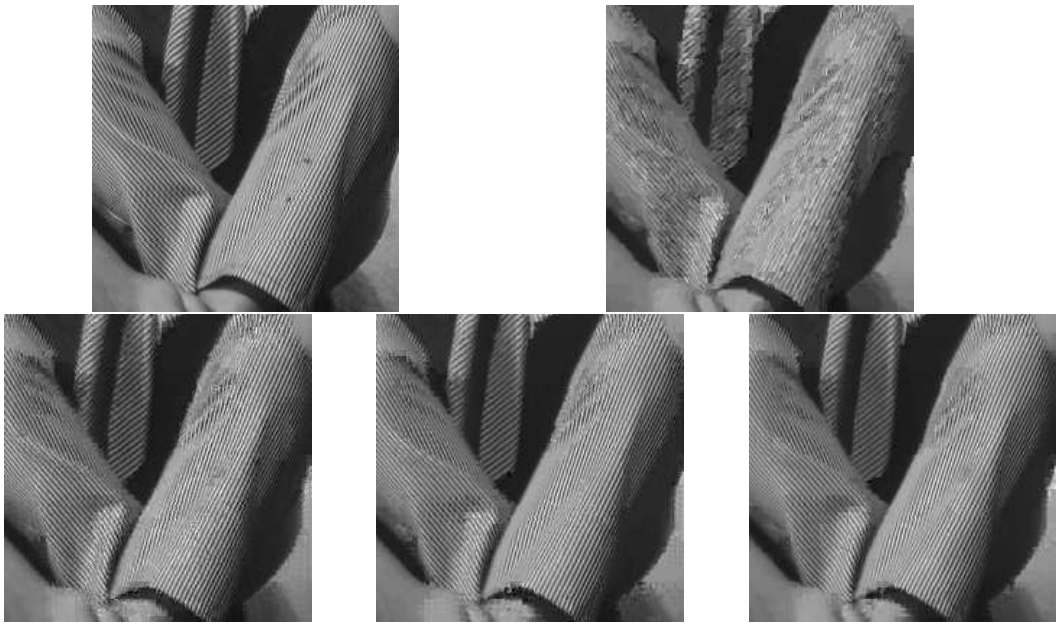


Figure 2.7: Prediction results for a textural region of Barbara image at low bit-rates ($q_f = 10$). (a) Original image, (b) H.264 intra modes, (c) SP, (d) LLE, and (e) NMF based prediction methods with MSE criterion and sparsity constraints ($k \in [1, 8]$).

based intra prediction especially for the image regions which contain complex textural structures and edges. Note here that the performance images shown here do not take the encoding cost of residue and the side information into account but only the prediction quality in terms of MSE. Therefore, it is informative rather than conclusive in terms of rate-distortion performance.

2.2.5 Compression performance with RD criterion

Fig. 2.9 shows the total encoding PSNR/bit-rate performance for the test images where the optimization criterion is the minimization of the RD cost function. One can observe that the proposed prediciton methods with NMF and LLE improve the encoding performance of the images especially containing textural regions when compared to the SP and H.264/AVC intra prediction. A gain up to 2 dB has been achieved by the NMF based method when compared to H.264/AVC, and up to 1 dB in comparison with the sparse prediction method. Furthermore,

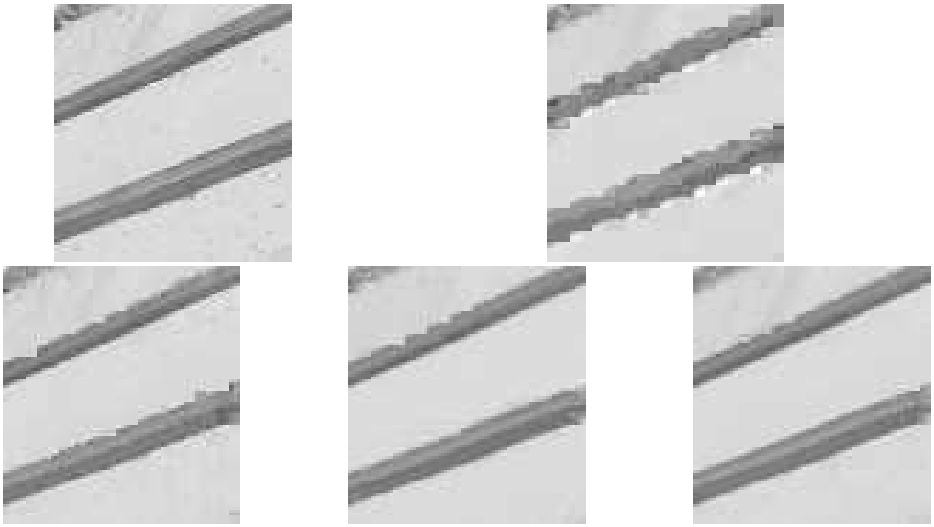


Figure 2.8: Prediction results for an edgel region of Foreman image at low bit-rates ($q_f = 10$). (a) Original image, (b) H.264 intra modes, (c) SP, (d) LLE, and (e) NMF based prediction methods with MSE criterion and sparsity constraints ($k \in [1, 8]$).

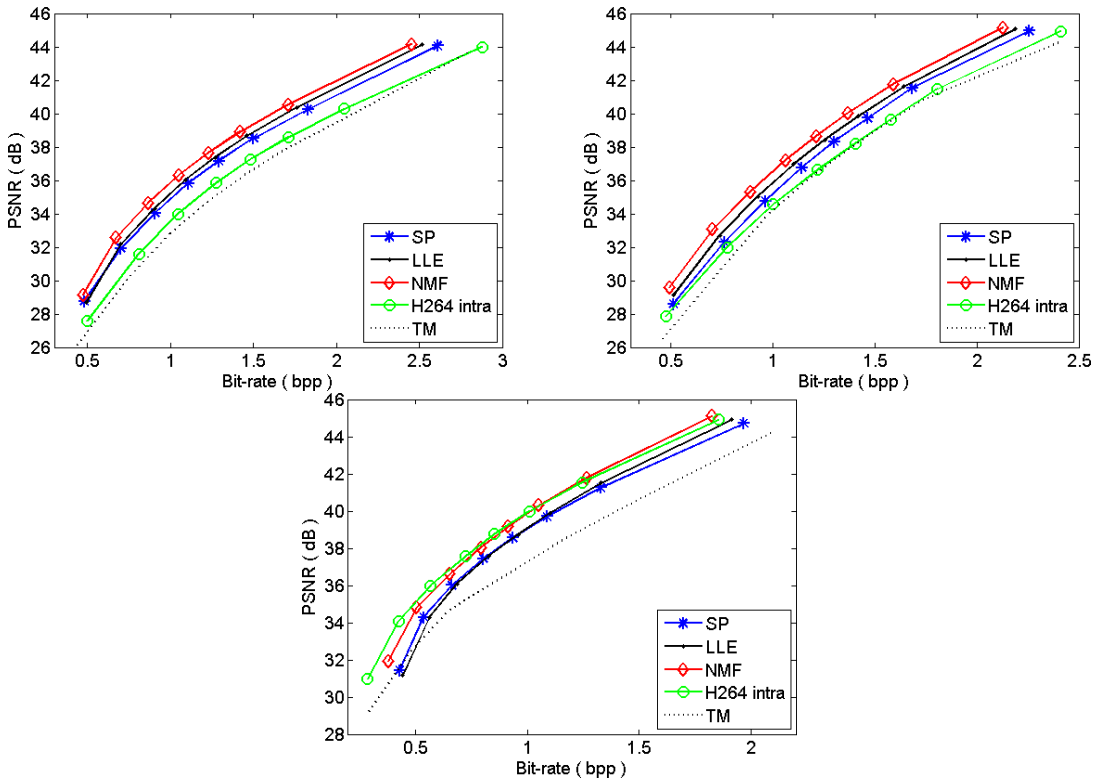


Figure 2.9: PSNR/bit-rate performance for (a) Barbara, (b) Roof, and (c) Foreman images.

the LLE based method outperforms the prediction methods including the SP and H.264/AVC.

Notice that for H.264/AVC intra prediction, only the selected prediction mode number is

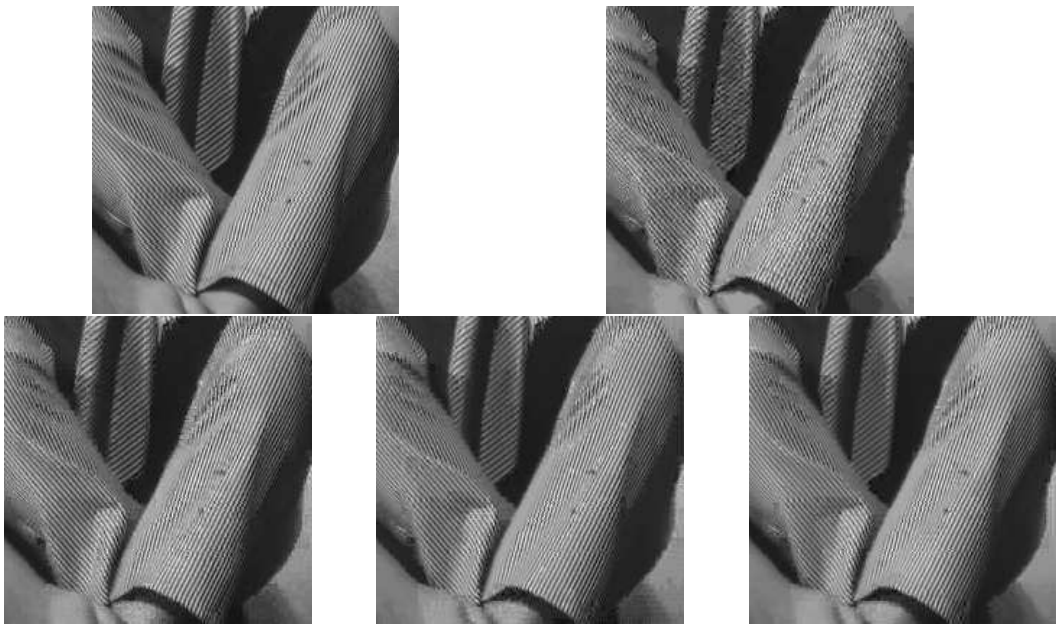


Figure 2.10: Reconstruction results for a textural region of Barbara image. (a) Original image, (b) H.264 intra modes (27.56 dB at 0.50 bpp), (c) SP (28.69 dB at 0.48 bpp), (d) LLE (28.68 dB at 0.49 bpp), and (e) NMF (29.14 dB at 0.48 bpp) based image prediction methods with sparsity constraints ($k \in [1, 8]$).

signalled to the decoder, however, for the other methods (i.e., SP, LLE, and NMF) the optimum number of used patches k (equally the iteration number in SP) is signalled in addition to the optimal template type. Thus, the gain in prediction might not compensate the coding cost of an extra side information for the images (as Foreman) which contain mostly smooth regions, and especially directional contours which are higly alligned with the H.264/AVC intra prediction modes. H.264/AVC intra prediction works relatively well for this particular image but is outperfomed by the other methods for the other test images.

Fig. 2.10 demonstrates a reconstructed textural region of Barbara image with H.264/AVC intra, SP, LLE, and NMF based image prediction methods.

Chapter 3

Texture synthesis for 2D/3D image inpainting

Image inpainting refers to methods which consist in filling-in missing regions (holes) in an image [20]. Inpainting techniques find applications in a number of image processing problems: image editing (e.g. object removal), image restoration, object disocclusion in image based rendering, image coding, loss concealment after impaired transmission. Existing methods can be classified into two main categories. The first category concerns diffusion-based approaches which propagate level lines or linear structures (so-called isophotes) via diffusion based on partial differential equations [20], [21] and variational methods [22]. In other words, they tend to prolong isophotes arriving at the border of the region to be filled. The diffusion-based methods tend to introduce some blur when the hole to be filled in is large.

The second type of approach concerns examplar-based methods which sample and copy best match texture patches from the known image neighborhood [23], [24], [25], [26], [27], [28], [29]. These methods have been inspired from texture synthesis techniques [30] and are known to work well in cases of regular textures. The first attempt to use exemplar-based techniques for object removal has been reported in [26]. The authors in [25], improve the search for similar patches by introducing an a priori rough estimate of the inpainted values using a multi-scale approach which then results in an iterative approximation of the missing regions from coarse to fine levels. In addition, the candidate patches for the match also include rotated, scaled and mirrored version of texture patches taken from the image.

In this section, we describe novel inpainting algorithms based on the examplar-based solution of [24]. The approach is extended along two directions: the first one consists in proposing new methods for defining the patch filling order. Indeed, as in [24], the proposed methods involve two steps: first, a filling order is defined to favor the propagation of structure contained in the patch to be filled in. Second, an approximation of the known samples in the patch to be filled is performed via template matching or with the help of more elaborate approaches (using neighbor embedding techniques) is performed in order to find the best candidates to fill in the hole.

3.1 Overview of examplar-based inpainting

Let I be the image and Ω the region to be filled in. Let $\delta Omega$ be the border of the region to be filled in. Given a patch Ψ_p centered at the point p (unknown pixel) located near the front line, the filling order (also called priority) is defined as the product of three terms: P(p) =

C(p)D(p)E(p). The first term, called confidenece term is given by [24]:

$$C(p) = \frac{\sum_{q \in \Psi_p \cap (I-\Omega)} C(q)}{|\Psi_p|} \tag{3.1}$$

It gives the ratio between the number of known pixels with respect to the total number of pixels in the patch to be filled in. The term $D(p)$, called the data term, is given by [24]

$$D(p) = \frac{|\nabla I_p^{\perp} n_p|}{\alpha} \tag{3.2}$$

where n_p is a unit vector orthogonal ($\perp$) to the filling front $\delta\Omega$ in the point p. This second term increases the priority of the patch having isophotes perpendicular to the filling front. However, it does not really reflect the predominance of an edge within the patch. Therefore, we introduce here a third term $E(p)$ which is the ratio of the amount of kwown pixels of the patch which belong to an edge with respect to the total number of pixels in the patch. Thus $E(p)$ is given by

$$E(p) = \frac{\sum_{q \in \Psi_p \cap (I-\Omega)} \delta(q \in \mathbf{E})}{|\Psi_p|} \tag{3.3}$$

where $delta()$ is a binary function which returns 1 when its argument if true and 0 otherwise. **E** is the set of edge pixels which is determined by using a canny edge detector.

Figure 3.1 compares the inpainted images with simple template matching (as in [24]) when using the priority function augmented with the "edge" term with respect to the original priority function.

Figure 3.1: Effect of the edge term in the priority function: Mask of the inpainted region (left); Inpainting with priority function of [24](middle); Inpainting with the augmented priority function (right).

3.2 New Priority computation

Two new approaches for computing the priority of the patches to be filled have been conceived.

3.2.1 Tensor-based priority computation

Given a patch ψ_p centered at the point $\mathbf{p}$ (unknown pixel) located near the front line, the filling order (also called priority) is defined as the product of two terms: $P(\mathbf{p}) = C(\mathbf{p})D(\mathbf{p})$.
The first term, called the confidence, is the same as in [24]. It is given by:

$$C(\mathbf{p}) = \frac{\sum_{q \in \psi_p \cap (I-\Omega)} C(\mathbf{q})}{|\psi_p|} \tag{3.4}$$

where $|\psi_p|$ is the area of ψ_p. This term is used to favor patches having the highest number of known pixels (At the first iteration, $C(\mathbf{p}) = 1 \ \forall \mathbf{p} \in \Omega$ and $C(\mathbf{p}) = 0 \ \forall \mathbf{p} \in I - \Omega$).
The second term, called the data term, is different from [24]. The definition of this term is inspired by PDE regularization methods acting on multivalued images [31]. The most efficient PDE-based schemes rely on the use of a structure tensor from which the local geometry can be computed. As the input is a multivalued image, the structure tensor, also called Di Zenzo matrix [32], is given by:

$$\mathbf{J} = \sum_{i=1}^{n} \nabla I_i \nabla I_i^T \tag{3.5}$$

$\mathbf{J}$ is the sum of the scalar structure tensors $\nabla I_i \nabla I_i^T$ of each image channel I_i (R,G,B). The structure tensor gives information on orientation and magnitudes of structures of the image, as the gradient would do. However, as stated by Brox et al. [33], there are several advantages to use a structure tensor field rather than a gradient field. The tensor can be smoothed without cancellation effects : $\mathbf{J}_\sigma = \mathbf{J} * G_\sigma$ where $G_\sigma = \frac{1}{2\pi\sigma^2} exp(-\frac{x^2+y^2}{2\sigma^2})$, with standard deviation σ. In this paper, the standard deviation of the Gaussian distribution is equal to 1.0.
The Gaussian convolution of the structure tensor provides more coherent local vector geometry. This smoothing improves the robustness to noise and local orientation singularities. Another benefit of using a structure tensor is that a structure coherence indicator can be deduced from its eigenvalues. Based on the discrepancy of the eigenvalues, this kind of measure indicates the degree of anisotropy of a local region. The local vector geometry is computed from the structure tensor $\mathbf{J}_\sigma$. Its eigenvectors $\mathbf{v}_{1,2}$ ($\mathbf{v}_i \in R^n$) define an oriented orthogonal basis and its eigenvalues $\lambda_{1,2}$ define the amount of structure variation. $\mathbf{v}_1$ is the orientation with the highest fluctuations (orthogonal to the image contours), and $\mathbf{v}_2$ gives the preferred local orientation. This eigenvector (having the smallest eigenvalue) indicates the isophote orientation. A data term D is then defined as [34]:

$$D(\mathbf{p}) \quad = \quad \alpha + (1-\alpha) exp(-\frac{C}{(\lambda_1 - \lambda_2)^2}) \tag{3.6}$$

where C is a positive value and $\alpha \in [0, 1]$ ($C = 8$ and $\alpha = 0.01$). On flat regions ($\lambda_1 \approx \lambda_2$), any direction is favored for the propagation (isotropic filling order). The data term is important in presence of edges ($\lambda_1 >> \lambda_2$).
Figure 3.2 shows the isophote directions (a) and the value of the coherence norm $(\frac{\lambda_1-\lambda_2}{\lambda_1+\lambda_2})^2$ (b). Black areas correspond to areas for which there is no dominant direction.

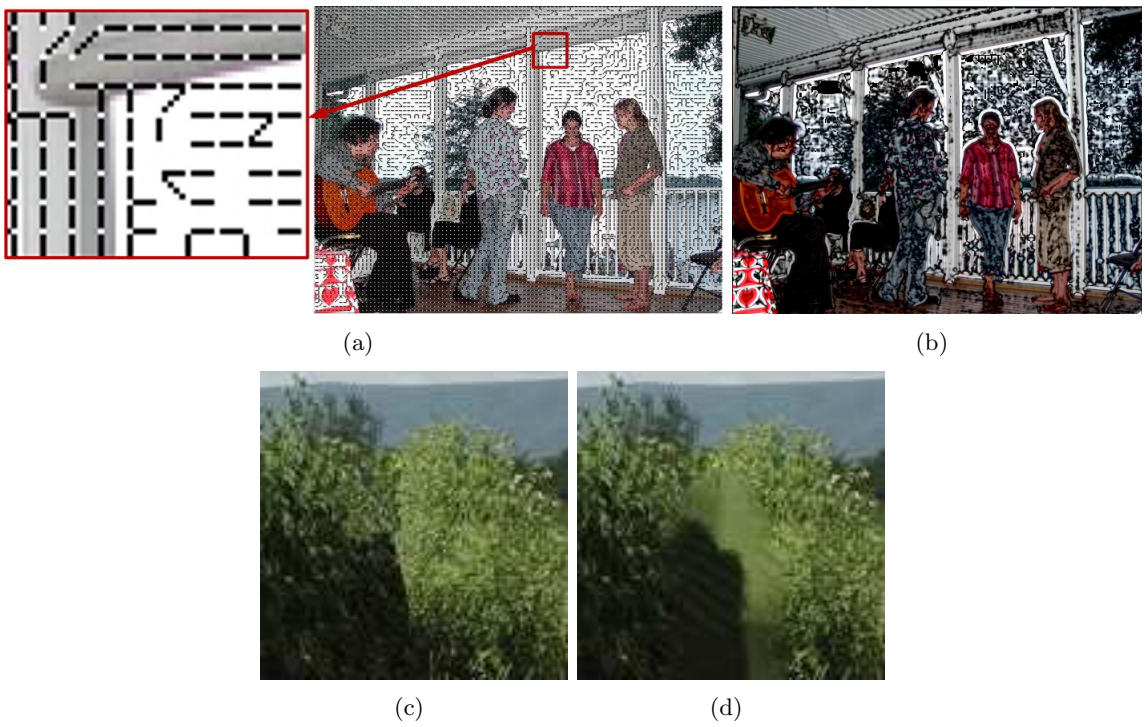


Figure 3.2: (a) direction of the isophotes;(b) coherence norm: black areas correspond to areas for which there is no dominant direction; (c) Filling with the best candidate (K=1); (d) Filling with the best 10 candidates.

3.2.2 Hierarchical tensor-based priority computation

The computation of the gradient ∇I as explained above to define the structure tensor presents some limitations. Indeed, as the pixels belonging to the hole to fill are initialized to a given value (0 for instance), it is required to compute the gradient only on the known part of the patch ψ_p. This constraint can undermine the final quality. To overcome this limitation, a hierarchical decomposition is used in order to propagate throughout the pyramid levels an approximation of the structure tensor. A Gaussian pyramid is then built with successive low-pass filtering and downsampling by 2 in each dimension leading to nL levels. At the coarsest level $\mathcal{L}_0$, the algorithm described in the previous section is applied. For a next pyramid level $\mathcal{L}_n$, a linear combination between the structure tensors of level $\mathcal{L}_n$ and $\mathcal{L}_{n-1}$(after upsampling) is performed:

$$\mathbf{J}_h^{\mathcal{L}_n} = \nu \times \mathbf{J}^{\mathcal{L}_n} + (1-\nu) \times \uparrow 2(\mathbf{J}^{\mathcal{L}_{n-1}}) \tag{3.7}$$

where $\mathbf{J}_h$ is a structure tensor computed from a hierarchical approach. $\uparrow 2$ is the upsampling operator. In our implementation, ν is fixed and set to 0.6. This hierarchical approach makes the inpainting algorithm more robust. At the coarsest level, the local structure tensor is a good approximation of the local dominant direction. Propagating such information throughout the pyramid decreases the sensitivity to local orientation singularities and noise. By default, nL is set to 3.

3.2.3 Edge-based priority computation

Given a patch Ψ_p centered at the point p (known pixel) located near the front line, the filling order (also called priority) is defined as the product of three terms: $P(p) = C(p)D(p)E(p)$. The first term, called confidence term is given by [24]:

$$C(p) = \frac{\sum_{q \in \Psi_p \cap (I-\Omega)} C(q)}{|\Psi_p|} \tag{3.8}$$

It gives the ratio between the number of known pixels with respect to the total number of pixels in the patch to be filled-in. The term $D(p)$, called the data term, is given by [24] as

$$D(p) = \frac{|\nabla I_p^{\perp} n_p|}{\alpha} \tag{3.9}$$

where α is a normalization factor and n_p is a unit vector orthogonal ($\perp$) to the filling front $\delta\Omega$ in the point p. This second term increases the priority of the patch having isophotes perpendicular to the filling front. However, it does not really reflect the predominance of an edge within the patch. Therefore, we introduce here a third term $E(p)$: this is the ratio of the amount of known pixels of the patch which belong to an edge with respect to the total number of known pixels in the patch. Thus $E(p)$ is defined as

$$E(p) = \frac{\sum_{q \in \Psi_p \cap (I-\Omega)} \delta(q \in \mathbf{E})}{|\Psi_p \cap (I - \Omega)|} \tag{3.10}$$

where $\delta()$ is a binary function which returns 1 when its argument is true and 0 otherwise. $\mathbf{E}$ is the set of edge pixels which is determined by using a Canny edge detector.

Figure 3.2.3 compares the inpainted images with simple template matching (as in [24]) when using the priority function augmented with the "edge" term with respect to the original priority function. The proposed priority function with edge term changes the filling order by giving more priority to structural blocks containing edge information. As it can also be seen from Figure 3.2.3, after 100 iteration steps, the shape of the mask is different from the original priority function. Thus, the structures are propagated first to prevent introducing any annoying visible artifacts in the continuation of the edgel areas in the image.

3.3 Propagating texture and structure information

Once the priority P has been computed for all unknown pixels $\mathbf{p}$ located near the front line, pixels are processed in decreasing order of priority. This filling order is called percentile priority-based concentric filling (PPCF). PPCF order is different from Criminisi's approach. Criminisi et al. [24] updated the priority term after filling a patch and systematically used the pixel having the highest priority. The advantage is to propagate the structure throughout the hole to fill. However, this advantage is in a number of cases a weakness. Indeed, the risk, especially when the hole to fill is rather big, is to propagate too much the image structures. The PPCF approach allows us to start filling by the $L\%$ pixels having the highest priority. The propagation of image structures in the isophote direction is still preserved but to a lesser extent than in [24].

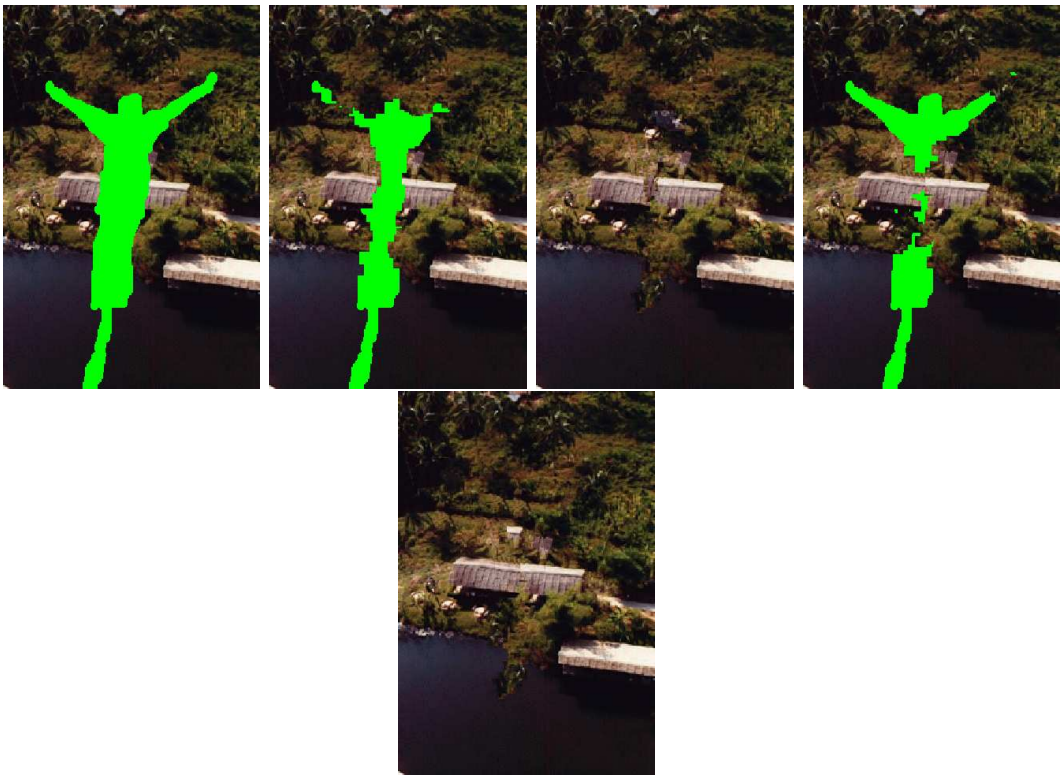


Figure 3.3: Effect of the edge term in the priority function. From left-to-right: The mask for the inpainting algorithm; the inpainting process after 100 patches with priority function of [24] and the final inpainted image; the inpainting process after 100 patches with the augmented priority function and the final inpainted image.

3.3.1 Template Matching along the Isophote direction

Once the pixel having the highest priority is found, a template matching based on the sum of squared differences (SSD) is applied to find a plausible candidate. SSD is computed between this candidate (entirely contained in ϕ) and the already filled or known pixels of ψ_p. Finally, the best candidate is chosen by the following formula:

$$\psi_{\widehat{q}} = \arg \min_{\psi_q \in \mathcal{W}} d(\psi_{\widehat{p}}, \psi_q) \tag{3.11}$$

where $d(.,.)$ is the SSD. Note that a search window $\mathcal{W}$ centered on p is used to perform the matching.
Finding the best candidate is fundamental for different reasons. The filling process must ensure that there is a good matching between the known parts of ψ_p and a similar patch in ϕ in order to fill the unknown parts of ψ_p. The metric used to evaluate the similarity between patches is then important to propagate the texture and the structure in a coherent manner. Moreover, as the algorithm is iterative, the chosen candidate will influence significantly the result that will be obtained at the next iteration. An error leading to the apparition of a new structure can be propagated throughout the image. In order to improve the search for the best candidate, the

previous strategy is modified as follows:

$$\psi_{\widehat{q}} = \arg\min_{\psi_q \in \phi} d(\psi_{\widehat{p}}, \psi_q) + (\frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2})^2 \times f(p,q) \tag{3.12}$$

where the first term $d(.,.)$ is still the SSD and the second term is used to favor candidates in the isophote direction, if any. Indeed, the term $(\frac{\lambda_1-\lambda_2}{\lambda_1+\lambda_2})^2$ is a measure of the anisotropy at a given position. On flat areas, this term tends to 0. The function $f(p,q)$ is given by:

$$f(p,q) = \frac{1}{\epsilon + \frac{|\mathbf{v}_2 \cdot \mathbf{v}_{pq}|}{\|\mathbf{v}_{pq}\|}} \tag{3.13}$$

where $\mathbf{v}_{pq}$ is the vector between the centre p of patch ψ_p and the centre q of a candidate patch ψ_q. ϵ is a small constant value, set to 0.001. If the vector $\mathbf{v}_{pq}$ is not collinear to the isophote direction (assessed by computing the scalar product $\mathbf{v}_2 \cdot \mathbf{v}_{pq}$), this candidate is penalized. In the worst case (the two vectors are orthogonal), the penalization is equal to $1/\epsilon$. When the two directions are collinear, the function $f(p,q)$ tends to one.

3.3.2 Hole filling based on neighbor embedding techniques

A K nearest neighbour search algorithm can also be used to compute the final candidate to improve the robustness. We follow Wexler et al.'s proposition [35] by taking into account that all candidate patches are not equally reliable (see equation 3 of [35]). An inpainting pixel $\widehat{c}$ is given by (c_i are the pixels of the selected candidates):

$$\widehat{c} = \frac{\sum_i s_i c_i}{\sum_i s_i} \tag{3.14}$$

where s_i is the similarity measure deduced from the distance (see equation 2 of [35]). Most of the time, the number of candidates K is fixed. This solution is not well adapted. Indeed, on stochastic or fine textured regions, as soon as K is greater than one, the linear combination systematically induces blur. One solution to deal with that is to locally adapt the value K. In this approach we compute the variance σ_W^2 on the window search. K is given by the function $a + \frac{b}{1+\sigma_W^2/T}$ (in our implementation we use $a = 1$, $b = 9$ and $T = 100$. It means that we can use up to 10 candidates to fill in the holes). Figure 3.2 (c) and (b) shows the rendering of a fine texture with the best and the best ten candidates. For this example, good rendering quality is achieved by taking into account only the best candidate.

Once the K-NN patches are found, the embedding which essentially searches for the best linear combination of these K-NN patches can be done in different ways.

Locally Linear Embedding

A method called locally linear embedding has been introduced in [15] for data dimensionality reduction. It aims at preserving the local linear structure of the high-dimensional data in the lower-dimensional space. The LLE method consists of the following steps:

1. It first identifies K nearest neighbors X^j per data point X^i. The usual measure is the Euclidean distance;

2. It then searches for the weights $W_{i,j}$, so that each data point is approximated by its neighbors. The algorithm thus aims at minimizing the cost function,

$$E(W) = \sum_i |X^i - \sum_j W_{i,j} X^j|^2. \tag{3.15}$$

The weights $W_{i,j}$ represent the contribution of the j^{th} data point to the reconstruction of the i^{th} point and are constrained to sum to one ($\sum_j W_{i,j} = 1$). Each data point X^i can only be reconstructed from its neighbors (i.e., $W_{i,j} = 0$ if the data point X^j does not belong to the neighbors of X^i). The optimal weights satisfying these constraints are obtained by solving a standard least square problem;

3. Finally, an embedding cost function is minimized in order to obtain the low-dimensional global internal coordinates by fixing the weights $W_{i,j}$ calculated in the previous step. (Please see [15] for more information.)

The LLE method (steps 1 and 2) is here directly applied on texture patches. One thus searches to approximate the known pixels of the patch $\Psi_{\hat{p}}$ to be filled-in by a linear combination of the colocated pixels in the K-NN patches $\Psi_j, j = 1 \dots K$, and keep the same linear combination to inpaint the unknown pixels of the patch.

Figure 3 compares the inpainting results obtained by using an LLE based approximation of the patch to be filled-in rather than using a simple template matching with both the priority function of [24] and the augmented priority function with $E(p)$. The LLE based approach using K patches instead of one (as in template matching) leads to a more realistic inpainted image. E.g., in Figure 3, one can see that the bushes are propagated into the sea with template matching whereas the LLE based method prevents this propagation. Furthermore, the quality of the inpainting of the roof is also improved visually. (K is fixed to 10, and patch size is 9×9.)

3.4 Performance illustration of tensor-based priority and isophote constrained TM

Figures 3.5 and 3.6 show the performance of the proposed method. The approach in [36]preserves quite well the images structures but the apparition of blur is annoying when filling large areas. Regarding Criminisi's approach, results of both approaches are similar on the first picture. On the latter two, the proposed approach outperforms it. For instance, the roof as well as the steps of the third picture are much more natural than those obtained by Criminisi's method. The use of tensor and hierarchical approach brings a considerable gain. Figure 3.6 shows results on pictures belonging to Kawai et al.'s database [37]. Compared to previous assessment, these pictures have a smaller resolution (200×200 pixels) than those used previously (512×384). As illustrated by the figure, the unknown regions have been coherently reconstructed. Except for the last picture, structures are well propagated without loss of texture information.
Next studies will focus on stochastic or inhomogeneous textures, for which repetitions of structure are absent. In this case, template matching fails to replicate this kind of texture in a coherent manner. Instead of using an exemplar-based method, it would be probably better to synthesise such texture by using stochastic-based texture models.

Figure 3.4: Inpainting with template matching as in [24] (left-top row); Inpainting with template matching with augmented priority (right-top row); Inpainting with LLE without edge-based priority (left-bottom row); Inpainting with LLE with edge-based priority (right-bottom row).

3.5 Performance illustration of the neighbor embedding techniques

The performance of the entire inpainting algorithm using the new patch priority computation, LLE and the learned mapping has been assessed on several test images, and compared with state-of-the-art diffusion-based approaches [38, 31] and exemplar-based methods [24, 39]. Figure 3.5 illustrates some of the results obtained for the test images. We selected natural images with large holes to be filled-in in order to test our proposed inpainting algorithm for object removal. Since the holes to be filled-in are quite large, diffusion-based approaches introduce blur into the image. When we compare our method with the other exemplar-based methods including [24] and [39], we see the natural looking and structure preserving capacity of the proposed method in this paper.

(a) Mask (b) Proposed (c) Tschumperle (d) Criminisi

(a) Mask (b) Proposed (c) Tschumperle (d) Criminisi

(a) Mask (b) Proposed (c) Tschumperle (d) Criminisi

Figure 3.5: Comparison of the proposed approach with the approaches [36, 24].

(a) Mask (b) Proposed (a) Mask (b) Proposed

(a) Mask (b) Proposed (a) Mask (b) Proposed

Figure 3.6: Results of the proposed approach on pictures proposed by [37].

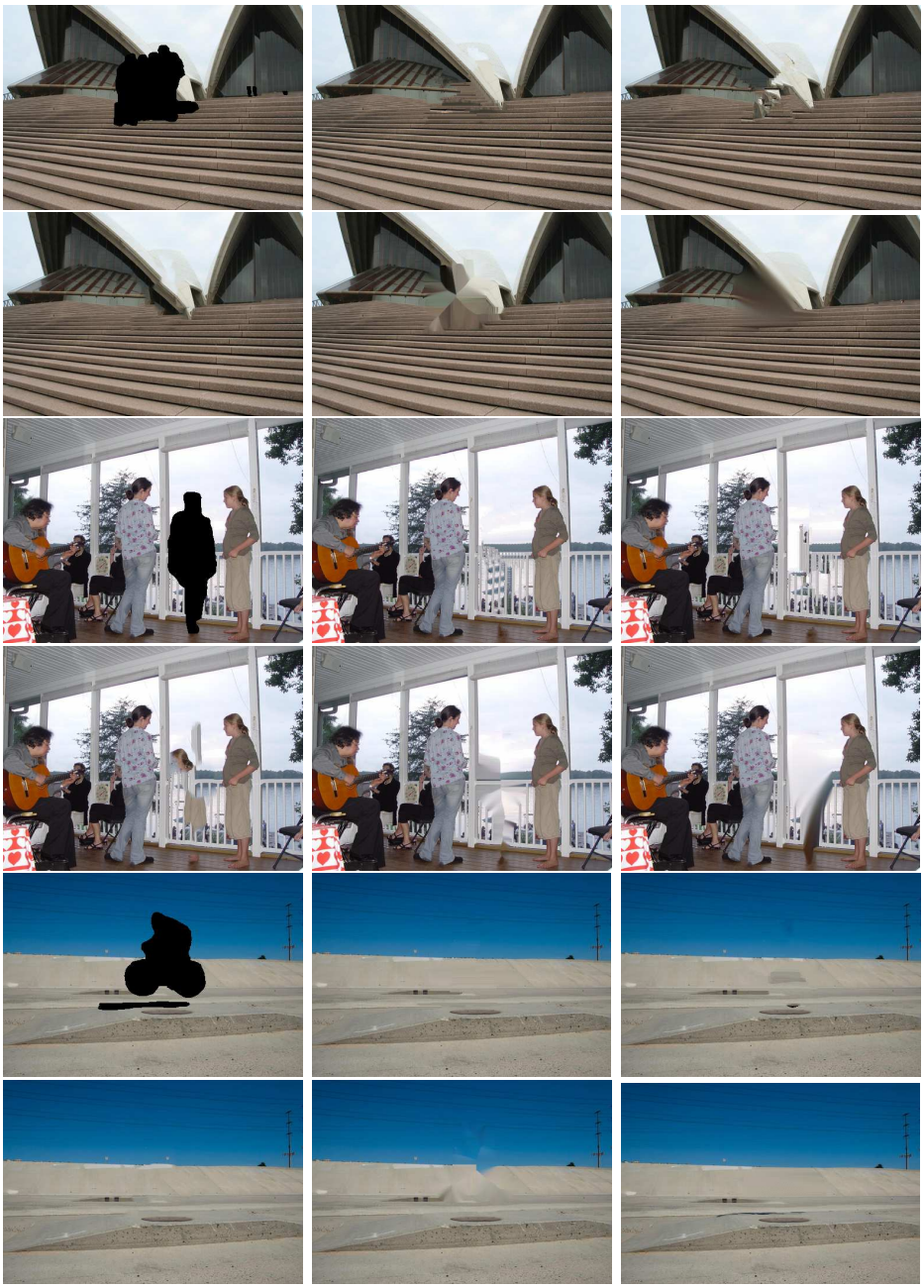


Figure 3.7: Inpainting results for natural test images. From left-to-right top-to-bottom per image: Inpainting mask, our method, method in [24], method in [39], method in [38], and method in [31].

Chapter 4

Texture synthesis for 3D inpainting in virtual view synthesis

3DTV and FTV are promising technologies for the next generation of home and entertainment services. Depth Image Based Rendering (DIBR) are key-solutions for virtual view synthesis on multistereoscopic display from any subset of stereo or multiview plus depth (MVD) videos. Classical methods use depth image based representations (MVD, LDV) to synthesize intermediate views by mutual projection of two views. Then, disoccluded areas due to the projection of the first view to the new one could be filled in with the remaining one. However, in freeviewpoint video (FVV) applications, larger baseline (distance or angle between cameras) involves larger disoccluded areas. Traditional inpainting methods are not sufficient to complete these gaps. To face this issue the depth information can help to guide the completion process. The use of depth to aid the inpainting process has already been considered in the literature. Oh et al. [40] based their method on depth thresholds and boundary region inversion. The foreground boundaries are replaced by the background one located on the opposite side of the hole. Despite the use of two image projections, their algorithm relies on an assumption of connexity between disoccluded and foreground regions, which may not be verified for high camera baseline configurations. Indeed, upon a certain angle and depth, the foreground object does not border the disoccluded part anymore. Daribo et al. [41] proposed an extension to the Criminisi's [24] algorithm by including the depth in a regularization term for priority and patch distance calculation. A prior inpainting of the depth map was performed. Our approach relies on the same idea. However, our contributions are threefold. The relevance of patch prioritization is improved by first using the depth as a coherence cue through a 3D tensor, and then by using a directional term preventing the propagation from the foreground. A combination of the K-nearest neighbor candidates is finally performed to fill in the target patch.

4.1 Algorithm

The motivation to use a Criminisi-based algorithm resides in its capacity to organize the filling process in a deterministic way. As seen in fig.4.1, this technique propagates similar texture elements $\Psi_{\hat{q}}$ to complete patches Ψ_p along the structure directions, namely the isophotes. Their algorithm basically works in two steps. The first step defines the higher order patch priorities along the borders $\delta\Omega$. The idea is to start from where the structure is the strongest (in term of

Figure 4.1: Illustration of principle. On (a) a warped view, (b) a zoom on the disoccluded area behind the person on the right, with the different elements overlaid.

local intensity, with $D(p)$) and from patches containing the highest number of known pixels, $C(p)$. The priority is then expressed as $P(p) = D(p) \times C(p)$. The second step consists in searching for the best candidate in the remaining known image in decreasing priority order.

In the context of view synthesis, some constraints can be added to perform the inpainting and improve the natural aspect of the final rendering. The projection in one view will be along the horizontal direction. For a toward-right camera movement the disoccluded parts will appear on the right of their previously occluding foreground (Figure 4.1a), and oppositely for a toward-left camera movement.

Whatever camera's movement, these disoccluded areas should always be filled in with pixels from the background rather than the foreground. Based on this a priori knowledge, we propose a depth-based image completion method for view synthesis based on robust structure propagation. In the following, $D(p)$ is described.

4.1.1 Depth-aided and direction-aided priority

The priority computation has been further improved by exploiting the depth information, first by defining a 3D tensor product, secondly by constraining the side from where to start inpainting.

3D tensor

The 3D tensor allows the diffusion of structure not only along color but also along depth information. It is critical to jointly favor color structure as well as geometric structure. The classical structure tensor defined in section 3.2.1 is extended with the depth map taken as an additional image component Z:

$$J = \sum_{l=R,G,B,Z} \nabla I_l \nabla I_l^T$$

One side only priority

The second improvement calculates the traditional priority term along the contour in only one direction. Intuitively, for a camera moving to the right, the disocclusion holes will appear to the right of foreground objects, while out-of-field area will be on the left of the former left border (in orange in Figure 4.1a). We then want to prevent structure propagation from foreground by

supporting the directional background propagation, as illustrated in Figure 4.1b with the blue arrows.

The patch priority is calculated along this border, the rest of the top, bottom and left patches being set to zero. Then for disocclused areas, the left border possibly connex to foreground will be filled at the very end of the process. For out-of-field areas, even if left borders are unknown, we will ensure to begin from the right border rather than possible top and bottom ones. These two proposals have been included in the prioritization step.

4.1.2 Patch matching

Once we precisely know from where to start in a given projected image, it is important to favor the best matching candidates in the background only. Nevertheless, starting from a non-foreground patch does not prevent it from choosing a candidate among the foreground, whatever the distance metric used. Thus, it is crucial to restrict the search to the same depth level in a local window: the background. We simply favor candidates in the same depth range by integrating the depth information in the commonly used similarity metric, the SSD (Square Sum of Differences):

$$\Psi_{\hat{q}} = \underset{\Psi_q \in \Phi}{\arg\min}\, d(\Psi_{\hat{p}}, \Psi_q) \quad \text{with } d = \sum_{p,q \in \Psi_{p,q} \cap \Phi} \alpha_l \left\| \Psi_{\hat{p}} - \Psi_q \right\|^2$$

The depth channel is chosen to be as important as the color one ($l \in R, G, B, Z$ with $\alpha_{R,G,B} = 1$ and $\alpha_Z = 3$). Then it will not prevent the search in foreground patches, but will seriously penalize and unrank the ones having a depth difference above, i.e in front of the background target patch. As proposed by [35], a combination of the best candidates to fill in the target patch shows more robustness than just duplicating one. We use a weighted combination of the K-best patches depending on their exponential SSD distances to the original patch. ($K = 5$ in our experiments).

4.2 Implementation

Experiments are performed on an unrectified Multiview Video-plus-Depth (MVD) sequence "Ballet" from Microsoft [42]. The depth maps are estimated through a color segmentation algorithm [42] and are supplied with their camera parameters. The choice of this sequence is motivated by the wide baseline unrectified camera configuration as well as its highly depth-and-color contrast resulting in distinct foreground-background. This makes the completion even more visible and the issue even more challenging.

First, the central view 5 is warped in different views. Standard cracks (unique vacant pixels) are filled in with an average filter. We then suppress certain ghosting effects present on the borders of disoccluded area in the background: the background ghosting. Indeed, as we start the filling process by searching from the border, it is of importance to delete ghostings containing inadequate foreground color values. A Canny edge detection on the original depth map, followed by a deletion of color pixels located behind that dilated border successfully removes this ghosting.

Finally, our inpainting method is applied on each warped image, using the depth of the final view. The depth inpainting issue is out of the scope of this paper, but encouraging methods are proposed in the literature [41]. In the context of MVD applications, it is realistic to consider a separate transmission of depth information through geometric representation (currently under investigation).

4.3 Results

Figure 4.2 illustrates the results obtained with the proposed method, comparatively with methods from the literature [24], [41], when rendering views located at varying distances from the reference viewpoint. The three versions take in input the same color and depth information, except for the approach in [24] using color only. Our method not only preserves the contour of foreground persons, but also successfully reconstructs the structure of missing elements of the disoccluded area (i.e. edges of the curtains and bars behind the person on the right, background wall behind the left one).

Thanks to our combination term, we can even extend the synthesis to very distant views, without suffering of aliasing effects. As illustrated, the view 5 is projected to view 2 ($V_{5\to 2}$) and the out-of-field blank areas occupying one quarter width of the warped image are reconstructed. The counterpart of the patch combination is the smoothing effect appearing on the bottom part of this area. By taking different numbers of patches for combination, it is possible to limit this effect. We encourage people to refer to additional results available on our webpage[1] with videos illustrating the priority-based progressive inpainting principle. The results can indeed be essentially address visually, as argued by [43].

[1] http://www.irisa.fr/temics/staff/gautier/inpainting

(a) $V_{5\to4}$ after warping and background antighosting

(b) $V_{5\to2}$ after warping and background antighosting

(c) $V_{5\to4}$ inpainted with Criminisi's method

(d) $V_{5\to2}$ inpainted with Criminisi's method

(e) $V_{5\to4}$ inpainted with Daribo's method

(f) $V_{5\to2}$ inpainted with Daribo's method

(g) $V_{5\to4}$ inpainted with our method

(h) $V_{5\to2}$ inpainted with our method

Figure 4.2: Illustration of different methods of inpainting. Our approach relying on 3D tensor and directional prioritization shows efficient filling.

Chapter 5

Joint projection/inpainting method

One classical problem in computer vision applications is the synthesis of virtual views from a single video sequence, accompanied by the corresponding depth map. This problem is encountered in applications such as robot navigation, object recognition, intermediate view rendering in free-viewpoint navigation, or scene visualization with stereoscopic or auto-stereoscopic displays for 3DTV.

Many rendering algorithms have been developed and are classified rather as Image-Based Rendering (IBR) techniques or Geometry-Based Rendering (GBR) techniques, according to the amount of 3D information they use. IBR techniques use multi-view video sequences and some limited geometric information to synthesize intermediate views. These methods allow the generation of photo-realistic virtual views at the expense of virtual camera freedom [44]. GBR techniques require detailed 3D models of the scene to synthesize arbitrary viewpoints (points of view). GBR techniques are sensitive to the accuracy of the 3D model, which is difficult to estimate from real multi-view videos. GBR techniques are thus more suitable for rendering synthetic data.

Depth-Image-Based Rendering (DIBR) techniques [45] include hybrid rendering methods between IBR and GBR techniques. DIBR methods are based on warping equations, which project a reference view onto a virtual viewpoint. Each input view is defined by a "color" (or "texture") map and a "depth" map, which associate a depth value to each image pixel. These depth maps are assumed to be known, or can be estimated from multi-video sequences by using a disparity estimation algorithm [46, 47].

The classical DIBR scheme for virtual view extrapolation from single input view plus depth video sequences is shown in Figure 5.1. The process in classical DIBR schemes is divided in several distinct steps, each one designed to solve a specific problem. First, the input depth map is warped onto the virtual viewpoint. The obtained warped depth map contains disocclusions, cracks and ghosting artifacts (these artifacts are detailed in section 5.1). Second, this virtual depth map is filtered a first time with a median filter, in order to remove the cracks, then a second time to dilate disocclusion areas on the background side, in order to avoid ghosting artifacts during view synthesis. Third, the filtered depth map is involved in a backward warping to compute the color of each pixel of the virtual view. Fourth, this resulting depth map is inpainted, to fill in disocclusion areas. Finally, this complete depth map is used by a depth-aided inpainting algorithm to fill in disocclusions in the color map.

All these steps are inter-dependent, and errors introduced by each one are amplified by the following one. Connectivity information is lost during the first projection step, as shown

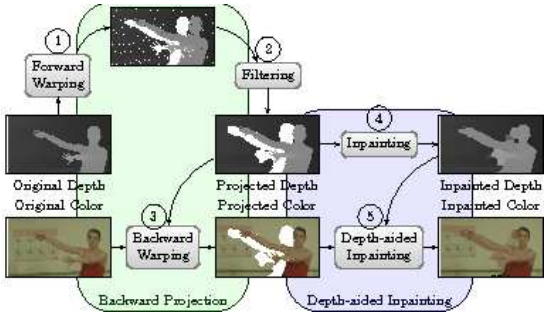


Figure 5.1: Classical scheme for virtual view extrapolation from a single input view plus depth video sequence. First, the input depth map is projected onto the virtual viewpoint. Second, the resulting depth map is filtered to avoid cracks and ghosting artifacts. Third, the filtered depth map is projected back onto the reference viewpoint to find the color of each pixel. Fourth, the depth map is inpainted to fill in disocclusions. Finally, the inpainted depth map is used to conduct disocclusions filling of the color map (synthesized view).

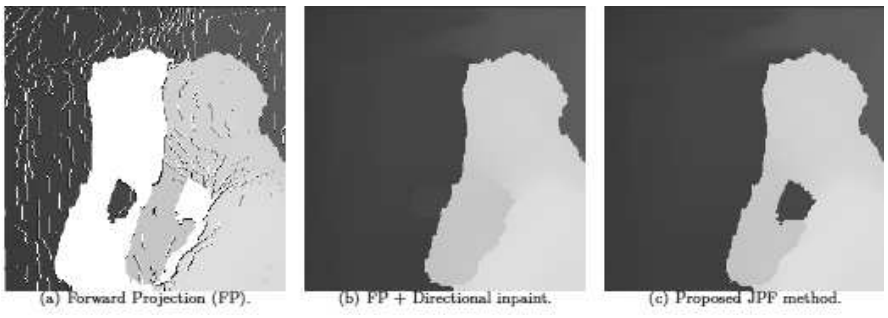


Figure 5.2: Virtual depth map synthesized by three forward projection methods. The point-based projection method generates cracks and disocclusions (left). Median filtering and directional inpainting [48] fills some holes with foreground depth (middle). The proposed JPF method fills cracks and disocclusions with realistic background (right).

in figure 5.2. Without this connectivity information, every inpainting method fails to fill in background disocclusions if the disoccluded area is surrounded by foreground objects. This case may happen each time a foreground object is not convex, and contains holes, as shown in figure 5.2. As a result, depth-aided inpainting uses wrong foreground patches to fill in background disocclusions, producing annoying artifacts, as shown in figure 5.2.

This chapter describes a new DIBR technique based on a novel forward projection technique, called the Joint Projection Filling (JPF) method. The JPF method performs forward projection, using connectivity information to fill in disocclusions in a single step. The JPF method is designed to handle disocclusions in virtual view synthesis, from one or many inputs view plus depth video sequences. The proposed DIBR method, depicted in Fig.5.3 is designed to extrapolate virtual views from a single input view plus depth video sequence. The method differs from the classical scheme by two points: the virtual depth map is synthesized by the JPF method, avoiding the use of dedicated filtering and inpainting processes; the depth-aided inpainting method is revised to take into account the high quality of the synthesized depth map.

The JPF method fills in disocclusion areas during the projection, to ensure that geomet-

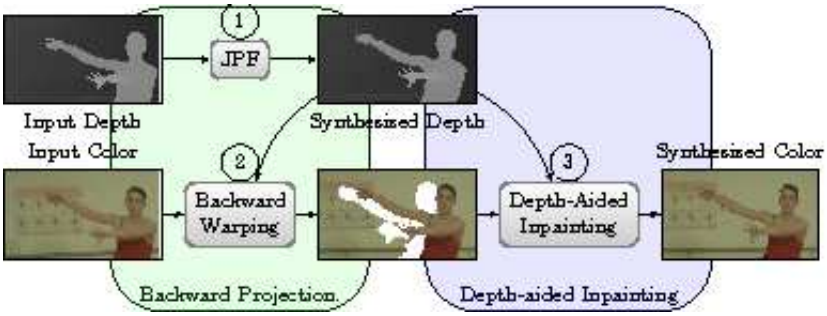


Figure 5.3: Virtual depth map synthesized by three forward projection methods. The point-based projection method generates cracks and disocclusions (left). Median filtering and directional inpainting [48] fills some holes with foreground depth (middle). The proposed JPF method fills cracks and disocclusions with realistic background (right).

rical structures are well preserved. The method uses the occlusion-compatible ordering presented by McMillan in [49], which uses epipolar geometry to select a pixel scanning order. The algorithm was initially introduced to perform the painter's algorithm during the projection without the need of a Z-buffer. Here, not using a Z-buffer is not our purpose (by the way, the constructed depth map is a Z-buffer). The occlusion-compatible ordering is instead used to handle disocclusions gracefully. Cracks are filled in by interpolation of neighboring pixels, whereas disocclusions are only filled in by background pixels. This technique can be used with non-rectified views, avoiding prior creation of parallax maps as done in [50].

In summary, the technique described here improves upon state-of-the-art DIBR methods as described in [51], by introducing the following key contributions:

- A novel forward projection method for DIBR, using occlusion compatible ordering [49] for detecting cracks and disocclusions, for which the unknown depth values are estimated while performing the warping. The resulting projection method thus allows us to handle both depth maps warping and disocclusion filling simultaneously. Small cracks and large disocclusions are handled gracefully, with similar computational cost as simple forward projection, avoiding the use of the filtering step as done in the classical approach.
- A ghost removal method to avoid ghosting artifacts in the rendered views, relying on a depth-based pixel confidence measure.
- A depth-aided inpainting method which takes into account all information given by the depth map to fill in disocclusions with textures at the correct depth.
- A method to handle inaccuracies of cameras calibration and depth map estimation by the use of the Floating Texture approach.

5.1 Background work

DIBR methods are based on warping techniques which project a reference view onto a virtual viewpoint. Directly applying warping equations may cause some visual artifacts in the synthesized view, like disocclusions, cracks and ghosting artifacts. Disocclusions are areas occluded in the reference viewpoint and which become visible in the virtual viewpoint, due to parallax effect.

Cracks are small disocclusions, mostly due to texture re-sampling. Ghosts are artifacts due to projection of pixels that have background depth and mixed foreground/background color. Various methods have been proposed in the literature to avoid these artifacts. This section presents state-of-the-art solutions to avoid each one of these three usual artifacts.

Ghosting artifacts are often avoided by detecting depth discontinuities on the depth map, in order to separate the boundary layer (containing pixels near a boundary) from the main layer (containing pixels far from a boundary) [42]. The main layer is first projected into the virtual viewpoint, then the boundary layer is added everywhere it is visible (i.e. where its depth value is smaller than the main layer's one). In [51], the authors propose to split again the boundary layer into foreground and background boundary layers. The main layer is first projected, the foreground boundaries layer is then added everywhere it is visible, and the background boundaries layer is finally used to fill in remaining holes. Ghosting artifacts can be further avoided by estimating the background and foreground contributions in the rendered view with the help of advanced matting techniques [52, 53, 54].

Cracks and other sampling artifacts are frequently avoided by performing a backward projection [55], which works in three steps. At first, the depth map is warped with a forward projection, resulting in some cracks and disocclusions. Then, this virtual depth map is median filtered to fill cracks, and bilateral filtered to smoothen the depth map while preserving edges. Finally, the filtered depth map is warped back into the reference viewpoint to find the color of the synthesized views. In [56], the authors propose to reduce the complexity by performing backward projection only for pixels labeled as cracks, i.e. pixels whose depth values are significantly modified by the filtering step. In [48], the authors propose an improved occlusion removal algorithm, followed by a depth-color bilateral filtering, in order to handle disocclusions on the depth map. Other improved rendering methods based on surface splatting have been proposed for avoiding cracks and texture re-sampling artifacts [57, 58].

Disocclusions are often filled in with information from some extra views, when they are available. The classical scheme is to synthesize the virtual view from each input view independently, then to blend the resulting synthesized views. In [59], the authors propose to compute an optical flow on intermediate rendered views, and then, with the help of the optical flow, to perform a registration step before the blending step, in order to avoid blurring in the final view due to blending mis-registered views. Note that specific representations such as Layered Depth Videos (LDV) can also be helpful for addressing the problem of occlusion handling since they allow storing texture information seen by other cameras [60, 61, 62, 63].

When extra views are not available, the frequent solution for disocclusion handling is image interpolation with inpainting techniques. Unfortunately, most inpainting techniques use neighboring pixels solely based upon colorimetric distance, while a disocclusion hole should be filled in with background pixels, rather than foreground ones [21, 38, 24]. In [56], the authors estimate each pixel value inside a disocclusion area from nearest known pixels along the eight cardinal directions, after nullifying the weight of foreground pixels. In [40], the authors temporarily replace foreground textures by background texture before inpainting, so that disocclusions are filled in only with background texture.

Advanced depth-aided inpainting methods assume that the depth map of the virtual viewpoint to be rendered is available. In [64], the authors enhance the inpainting method in [24] by reducing the priority of patches containing a depth discontinuity, and by adding a depth comparison in the search for best matches. In [65], the authors use a similar approach but estimate isophotes directions with a more robust tensor computation and constrain the propagation in

the direction of the epipole.

The full depth map from the virtual view is most of the time not available, and must be estimated from the input depth map. In [64], the authors perform a diffusion-based inpainting [21] on the projected depth map, but both foreground and background are diffused to fill disocclusions. In [48], the authors constrain the depth map inpainting in the direction of the epipole, in order that only the background is diffused, but this method fails when a disocclusion is surrounded by foreground depth, as shown in figure 5.2.

As a conclusion, state-of-the-art DIBR methods need a complete depth map at the rendered viewpoint (for backward projection and depth-aided inpainting). However, no fully satisfying method yet exists to obtain a complete and correct depth map, avoiding artifacts generation when used for DIBR. Moreover, most of disocclusions handling methods proposed in the literature work as a post treatment on the projected view. Connectivity information is not preserved during the projection, and inpainting methods fail to fill in background disocclusions when they are surrounded by foreground objects. The proposed JPF method aims at suppressing such drawbacks. As shown in figure 5.2, the JPF method enables to recover correct depth information in critical areas. We also propose a full-Z depth-aided inpainting technique which takes into account the high quality of the computed depth map to fill disocclusions with texture from the correct depth.

5.2 Projection-based disocclusion handling

This section introduces the Joint Projection Filling (JPF) method, which simultaneously handles warping and disocclusion filling, in order to preserve connectivity and fill in disocclusions with background textures.

During warping, there might happen overlapping (several pixels projected at the same position) or disocclusion (no pixels projected at a position). In [49], a pixel scanning order is introduced to perform the painter's algorithm during the projection. In case of overlapping, this pixel scanning order ensures the pixel just projected at a position to be the foreground pixel so that the z-buffer is not needed. A second property, resulting from the first one, is more helpful to handle disocclusions. If two successive pixels are not adjacent, there is a disocclusion, and the pixel just projected is the background pixel. This second property is exploited to ensure only background pixels are used to fill in disocclusion areas.

The JPF algorithm is described in section 5.2.1. It is first introduced for rectified cameras and then generalized for non-rectified cameras. Section 5.2.2 presents a ghosting removal method, based on pixels confidence measure. Finally, section 5.2.3 presents some synthesized textures and depth map, obtained by the JPF method.

5.2.1 Disocclusion detection

In the following, we assume that the epipolar geometry is such that the pixels from the reference image are processed sequentially, from top-left to bottom-right, according to McMillan scanning order [49].

Figure 5.6 presents the principle of the Joint Projection Filling (JPF) method, in the particular case of rectified cameras. Each row is thus independent of the others, reducing the problem to one dimension. Consider a row of pixels from the reference view, and a pixel $p = (p_x, p_y)$ on that row. The pixel p is projected on position $p' = (p'_x, p_y)$ in the synthesized view.

After having processed pixel p, the next pixel to be processed is $q = (p_x + 1, p_y)$. Its projected position $q' = (q'_x, p_y)$ verifies one out of the three following equations:

$$\begin{cases} q'_x = p'_x + 1 & \text{Pixels } p' \text{ and } q' \text{ are adjacent.} \\ q'_x < p'_x + 1 & \text{There is an overlap.} \\ q'_x > p'_x + 1 & \text{There is a crack or a disocclusion.} \end{cases} \tag{5.1}$$

The first and the second cases do not generate artifacts. In the last case, p' and q' are in same order as p and q, but there is a gap between them. In the proposed method, contrary to classical point-based projection, this gap is filled in immediately, before processing the projection of the next pixel. The method to fill the gap is adapted to its size. If the gap is small enough, it is considered as a crack. p' and q' are thus assumed to be on same layer, and the gap is filled in by interpolating the two pixels p' and q'. If the gap is too large, it is considered as a disocclusion. p' and q' are thus assumed to be on two distinct depth layer and the gap is filled in by background pixel. The McMillan pixel ordering ensures that q' is the background pixel, which is stretched from position p' to q'. The value of each pixel m between p' and q' is thus estimated as follows:

$$m = \begin{cases} (1-\alpha)p' + \alpha q' & \text{if } d \leq K \\ q' & \text{if } d > K \end{cases} \qquad \text{where } \begin{cases} d = q'_x - p'_x \\ \alpha = \frac{1}{d}(m_x - p'_x) \end{cases} \tag{5.2}$$

In the simulation results reported in the paper, the threshold K has been fixed to 5 pixels, to handle cracks and small disocclusions.

The algorithm is generalized for non-rectified cameras, as illustrated in figure 5.6. Pixels p' and q' may no longer be on the same row, thus we define pixel $P^{q'}$ as the last pixel projected on row q'_y. Equation (5.1) is revised, replacing p' with $P^{q'}$, thus q' and $P^{q'}$ are on the same row.

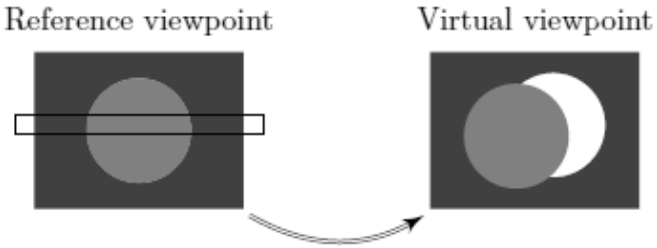
$$\begin{cases} q'_x \leq P_x^{q'} + 1 & \text{There is no artifact.} \\ q'_x > P_x^{q'} + 1 & \text{There is a disocclusion.} \end{cases} \tag{5.3}$$

As previously, the disocclusion handling method depends on the distance between q'_x and $P_x^{q'}$. The value of each pixel m between $P^{q'}$ and q' is thus estimated as follows:

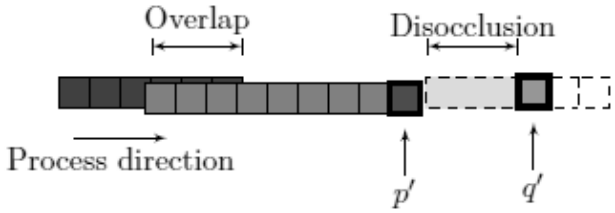
$$m = \begin{cases} (1-\alpha)P^{q'} + \alpha q' & \text{if } d \leq K \\ q' & \text{if } d > K \end{cases} \qquad \text{where } \begin{cases} d = q'_x - P_x^{q'} \\ \alpha = \frac{1}{d}(m_x - P_x^{q'}) \end{cases} \tag{5.4}$$

Figure 5.5 presents the synthesized depth maps obtained with the JPF method, without any ghosting removal technique. Our JPF method has removed all cracks and has filled in the disocclusions with only background pixels. Depth maps from the "Ballet" sequence contain sharp discontinuities, which are preserved by the JPF method (figure 5.5). Depth maps from other sequences contain some blur along depth discontinuities, due to DCT-based compression. This blur produces some sparse pixels inside the disocclusion area, which are stretched to fill the disocclusion, resulting in an annoying ghosting artifact.

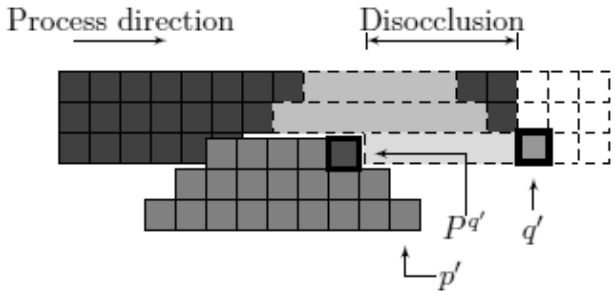
This occlusion-compatible ordering is helpful to detect cracks and disocclusions. Next section explains how to fill in disocclusions while preserving edges sharpness and avoiding ghosting artifacts.



(a) Projection.



(b) Details of a single row of pixels in the synthesized view for a rectified projection.



(c) Details of some rows of pixels in the synthesized view for a non rectified projection.

Figure 5.4: JPF method scheme for rectified and non rectified cameras. q' is a background pixel which is used to fill in the highlighted disocclusion.

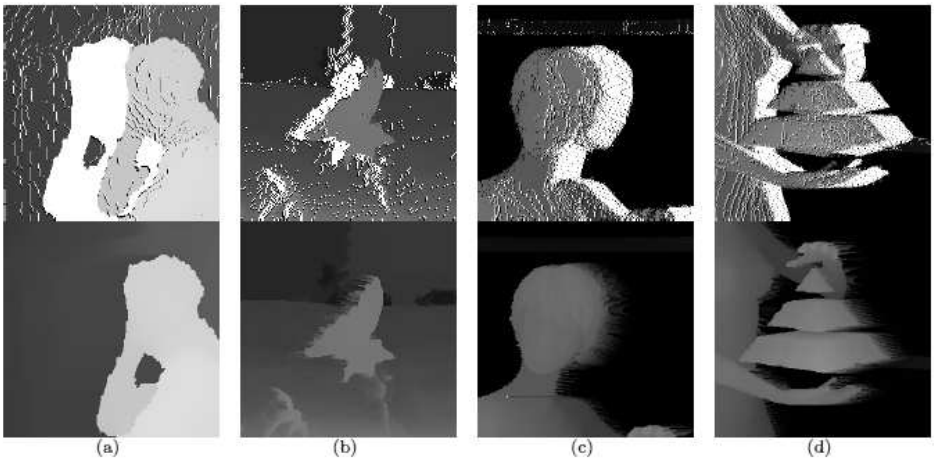


Figure 5.5: Comparison between synthesized depth maps from a forward point-based projection (first row) and from the JPF method (second row). Blurred depth discontinuities in the original depth map produces stretching effects on the synthesized depth maps.

5.2.2 Disocclusion filling

Pixels along objects boundaries are considered unreliable, because they often contain mixed foreground/background information for texture and depth value. Their projection may thus create ghosting artifacts in the synthesized views. The JPF method fills in each row of a disoccluded region using a single pixel. When applied on such a "blended" boundary pixel, this method may result in annoying pixel stretching artifacts, as can be seen in figure 5.5. However, these artifacts can be minimized by adapting the pixel stretching length, according to a pixel confidence measure. The algorithm used to avoid stretching and ghosting artifacts thus proceeds with the following two steps:

In a first step, a confidence measure $\lambda_q \in [0;1]$ is computed for each pixel q by convolving the depth map (Z) with a Difference-Of-Gaussians (DOG) operator as follows:

$$\lambda_q = 1 - (\mathrm{DOG} * \mathrm{Z})(q) \tag{5.5}$$

The DOG operator is built as the difference of two gaussians: the gaussian G of variance σ^2, and the 2D Dirac delta function δ_2.

$$\begin{aligned} \mathrm{DOG} &= G - \delta_2 \\ G(u,v) &= \frac{1}{\sigma^2} \cdot \phi\left(\frac{u}{\sigma}\right) \cdot \phi\left(\frac{v}{\sigma}\right) \end{aligned} \tag{5.6}$$

where ϕ is the standard normal distribution. The value of σ, in the experiments described below, has been fixed to 3.

In a second step, the confidence measure is used during the JPF method, to confine pixel stretching. Reusing the notations introduced in section 5.2.1, suppose that a wide disocclusion is discovered during the projection of pixel q. Instead of filling the whole gap between $P^{q'}$ and q', with color and depth values of q', only a part of the gap is filled in. The rest will be filled with the next pixel which will be projected on that same row j.

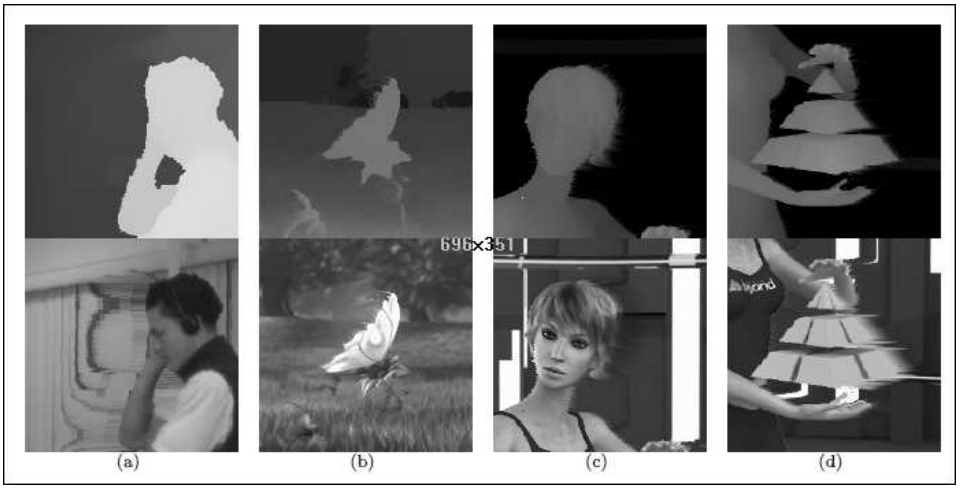


Figure 5.6: Warping results with the JPF method.

Assume M is a point between $P^{q'}$ and q', defined with the following equation:

$$M = (1 - \lambda_q^2)P^{q'} + \lambda_q^2 q' \tag{5.7}$$

The gap between $P^{q'}$ and M is filled in by pixel q', thus pixels on foreground/background boundaries which have low confidence measures are used to fill the disocclusion only for a couple of pixels next to the foreground, where blended pixels are expected to be in the synthesized view.

5.2.3 Results

This confidence-based interpolation method shifts back unreliable pixels near the discontinuities and only uses reliable pixels to fill in disocclusions. Figure 5.6 presents the rendering results of the JPF method with confidence-based interpolation. The projected depth maps, shown on the first row, are to be compared with those presented in figure 5.5. One can see that depth discontinuities are sharpened, producing realistic depth maps. The second row presents the results obtained with the same algorithm applied on texture. Disocclusions are gracefully filled in when the background is uniform, but annoying stretching artifacts appear in case of textured background. This JPF method can be used as a part of a virtual view synthesis algorithm, depending on the application. Two use cases are addressed in section 5.3, either for virtual view extrapolation when only one input view is available, or for intermediate view interpolation when multiple input views are available.

5.3 Virtual view rendering

The JPF method is designed to synthesize virtual views from one or many input view plus depth video sequences, depending on the final application. Section 5.3.1 describes a virtual view extrapolation algorithm, which is used when only one input view plus depth video sequence is available. Section 5.3.2 presents an interpolation algorithm to synthesize intermediate views when multiple video plus depth video sequences are available.

5.3.1 View extrapolation with full-Z depth-aided inpainting

In order to synthesize a virtual view from only one input view plus depth sequence, the developed algorithm proceeds as follows. First, the depth map for the virtual view is synthesized by our JPF method, handling ghosting, cracks and disocclusions. Then, the texture of the virtual view is obtained by a classical backward warping followed by the proposed full-Z depth-aided inpainting algorithm.

Our proposed full-Z depth-aided inpainting algorithm is a modification of the depth-aided inpainting method described in [64], itself based on the exemplar-based inpainting approach, introduced in [24]. Section 5.3.2 describes our proposed modification which takes into account the high quality of the virtual depth map. The importance of the synthesized depth map quality is discussed in section 5.4, for three different depth-aided inpainting methods.

5.3.2 Proposed full-Z depth-aided inpainting

The synthesized depth map does not contain holes, thanks to the JPF method which projects the input depth map onto the virtual viewpoint while filling cracks and disocclusions. The patch $\Psi\hat{p}$ to be filled in contains thus a depth value for each pixel, even for pixels in the hole region Ω. These depth values are close to the ground truth, because disocclusions are only filled in with background depth. The proposed modification is to use the depth value of all pixels in the patch, including those whose color is not known, i.e., the distance between patches is computed as follows:

$$\Psi_{\hat{q}} = \arg\ \min_{\Psi_q \in \Phi} \{\mathrm{SSD}_{\Phi}(\Psi_{\hat{p}}, \Psi_q) + \alpha\, \mathrm{SSD}_{\Phi \cup \Omega}(Z_{\hat{p}}, Z_q)\} \tag{5.8}$$

5.4 Rendering Results

This section compares virtual view synthesis results obtained when using three depth-aided inpainting techniques for occlusion handling. For each inpainting techniques, the virtual depth maps are synthesized either by the classical scheme, shown in figure 5.1, or by the JPF method.

Figure 5.7 shows inpainting results of the algorithm presented in [64]. Figure 5.8 shows inpainting results of the algorithm presented in [65]. Figure 5.9 shows inpainting results of the proposed full-Z depth-aided inpainting algorithm. In each figure, the first column shows a virtual view synthesized by the backward projection, where disocclusions appear in white. The second column shows the virtual depth map where disocclusions are filled in with a Navier-strokes inpainting algorithm [21], whereas the fourth column shows the depth map synthesized with our JPF method. The third and the fifth columns show the results of the depth-aided inpainting method, led by the depth map respectively presented in column 2 and 4.

One can observe that the depth maps shown in column 2 are not realistic because depth discontinuities do not fit with object boundaries. This is due to the depth map inpainting method, which fills disocclusions with both background and foreground values. On the contrary, depth maps presented in column 4 are closer to the ground truth, thanks to the JPF method. Small details are well preserved by the projection, as fingers on row 3 or blades of grass on row 4.

The influence of the virtual depth map can be observed by comparing column 3 and 5 of each figure. Errors in depth map from column 2 are amplified by every depth-aided inpainting method, because some foreground patches are selected to fill in disocclusions. The resulting

images, shown in column 3, contain more artifacts than the ones obtained with a correct depth map.

Depth-aided inpainting methods can be compared with each other by analyzing the fifth column of each figure. Rendering results shown in figures 5.7 and 5.8 still contains blur artifacts along boundaries, even if the correct depth map is used to conduct the inpainting process. The proposed full-Z depth-aided inpainting method preserves small details, as fingers on row 3 or blades of grass on row 4.

As a conclusion, the quality of the rendered view is strongly dependent on the quality of the virtual depth map, no matter the depth-aided inpainting method. Synthesizing high quality virtual depth map is thus an interesting challenge for DIBR techniques. The JPF method is well suited for this purpose, because connectivity information is used during the forward projection. Moreover, the proposed full-Z depth-aided inpainting method improves upon state-of-the-art methods by taking into account the correctness of the synthesized depth map.

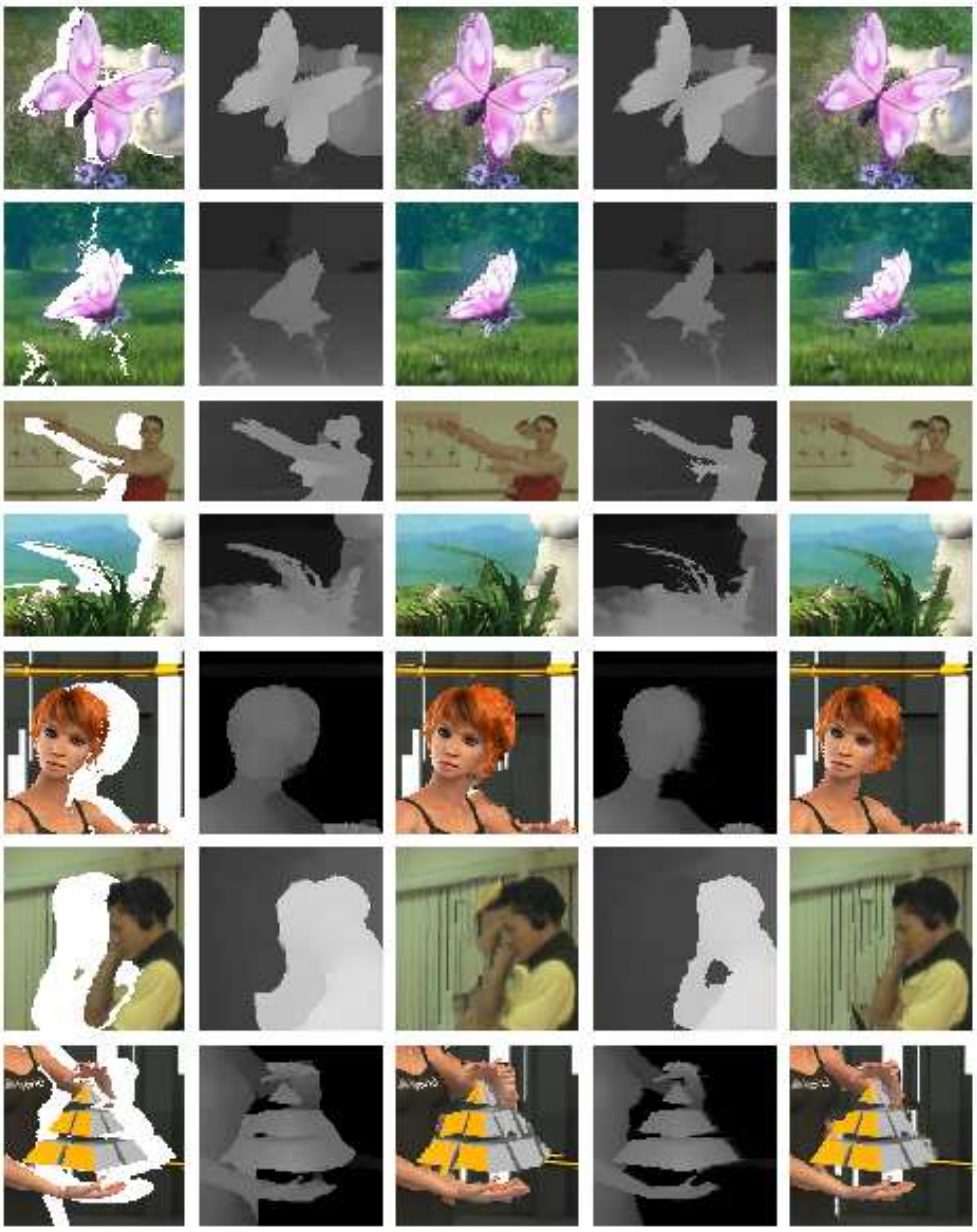


Figure 5.7: Results for Daribo depth-aided inpainting [64]. The first column shows a synthesized view with disocclusions. Columns 2 and 4 present the synthesized depth maps, obtained respectively with a Navier-strokes inpainting algorithm and with our JPF method. Columns 3 and 5 exhibit the results of the inpainting of the texture shown in column 1, guided by the depth map respectively presented in columns 2 and 4.

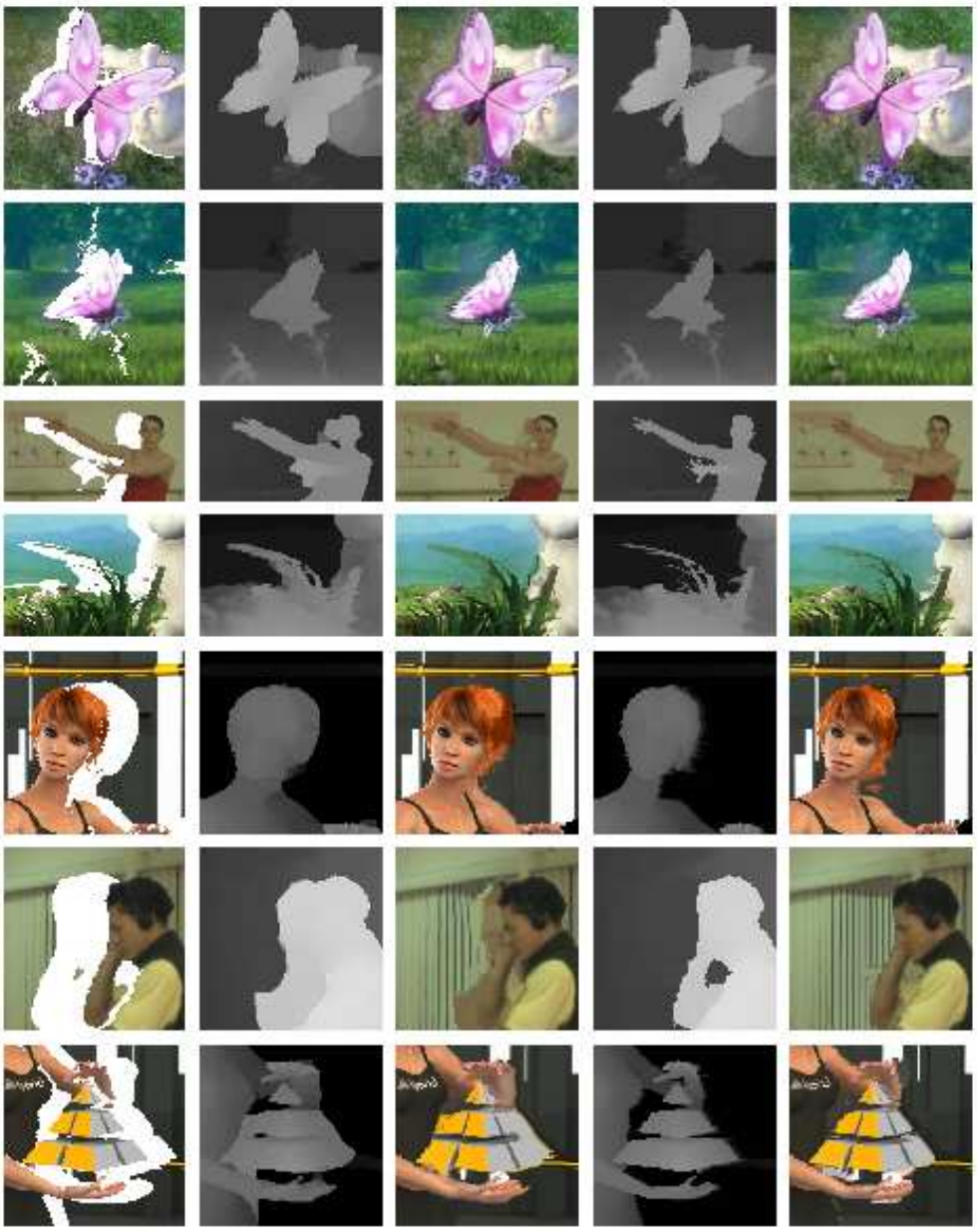


Figure 5.8: Results for Gautier depth-aided inpainting [65]. The first column shows a synthesized view with disocclusions. Columns 2 and 4 present the synthesized depth maps, obtained respectively with a Navier-strokes inpainting algorithm and with our JPF method. Columns 3 and 5 exhibit the results of the inpainting of the texture shown in column 1, guided by the depth map respectively presented in columns 2 and 4.

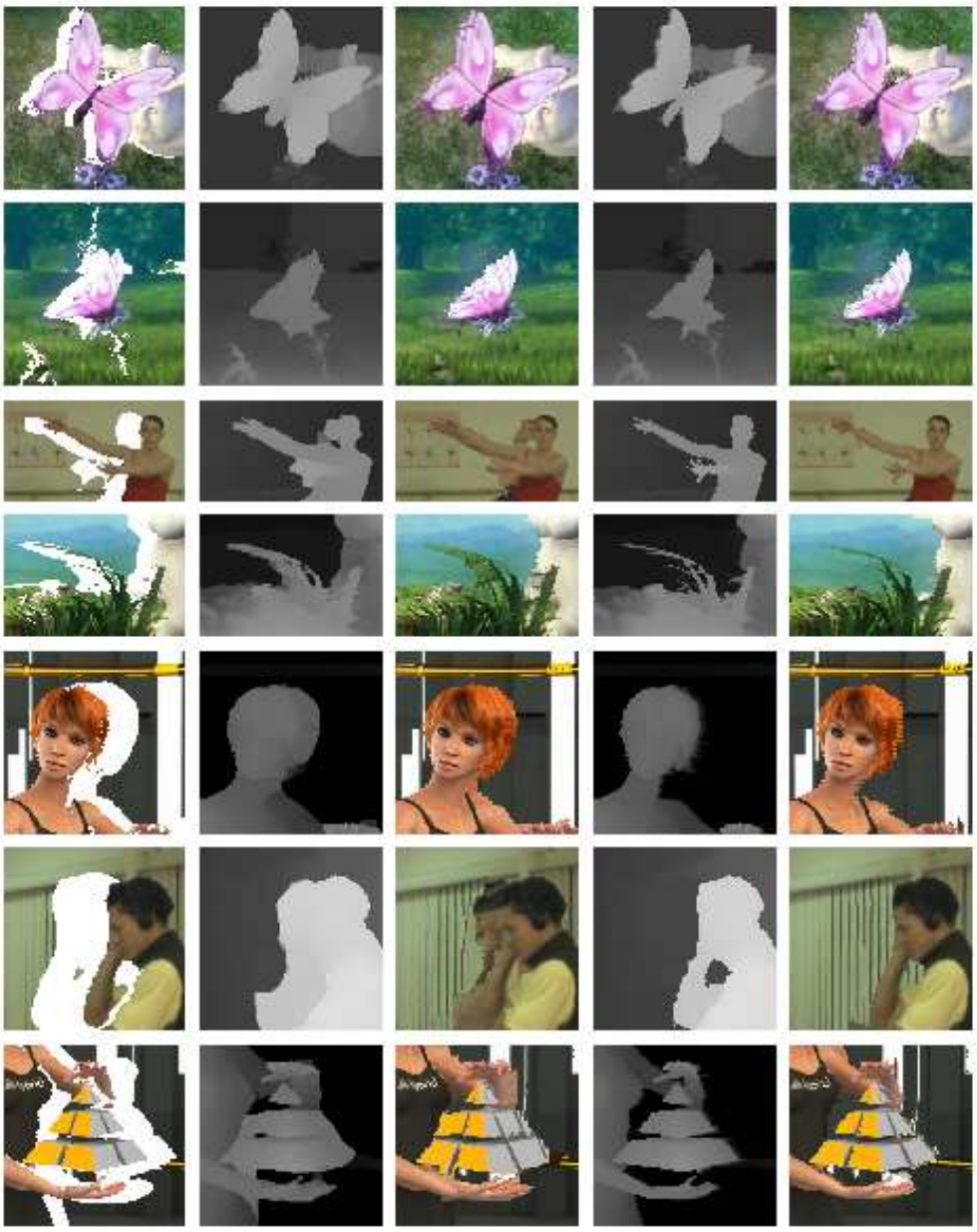


Figure 5.9: Results for proposed full-Z depth-aided inpainting. The first column shows a synthesized view with disocclusions. Columns 2 and 4 present the synthesized depth maps, obtained respectively with a Navier-strokes inpainting algorithm and with our JPF method. Columns 3 and 5 exhibit the results of the inpainting of the texture shown in column 1, guided by the depth map respectively presented in columns 2 and 4.

Bibliography

[1] M. Turkan and C. Guillemot, "Image prediction based on neighbor embedding methods," *IEEE Trans. On Image Processing*, 2011, accepted.

[2] V. Jantet, C. Guillemot, and L. Morin, "Joint projection filling method for occlusion handling in depth-image-based rendering," *International journal on 3D research, special issue on "3DTV"*, 2011, accepted.

[3] J. Gautier, O. Le Meur, and C. Guillemot, "DEPTH-BASED IMAGE COMPLETION FOR VIEW SYNTHESIS," in *3DTVConf*, ANTALYA, Turquie, 2011.

[4] O. Le Meur, J. Gautier, and C. Guillemot, "EXAMPLAR-BASED INPAINTING BASED ON LOCAL GEOMETRY," in *ICIP*, Brussel, Belgique, 2011.

[5] M. Turkan and C. Guillemot, "Image prediction based on non-negative matrix factorization," in *IEEE Int. Conf. on Acous. Speech and Signal Process. (ICASSP)*, Prague, May 2011.

[6] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. on Circuits and Systems for Video technology*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[7] T. K. Tan, C. S. Boon, and Y. Suzuki, "Intra prediction by template matching," in *Proc. IEEE Int. Conf. Image Process.*, 2006, pp. 1693–1696.

[8] ——, "Intra prediction by averaged template matching predictors," in *Proc. IEEE Consumer Comm. Network. Conf.*, 2007, pp. 405–409.

[9] A. Martin, J.-J. Fuchs, C. Guillemot, and D. Thoreau, "Sparse representation for image prediction," in *European Signal Process. Conf.*, 2007.

[10] S. Mallat and Z. Zhang, "Matching pursuit with time-frequency dictionaries," *IEEE Trans. on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec. 1993.

[11] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Proc. Asimolar Conf. Signals Systems Compt.*, 1993, pp. 40–44.

[12] M. Turkan and C. Guillemot, "Sparse approximation with adaptive dictionary for image prediction," in *Proc. IEEE Int. Conf. Image Process.*, 2009, pp. 25–28.

[13] ——, "Image prediction based on non-negative matrix factorization," in *Proc. IEEE Int. Conf. Acous. Speech Signal Process.*, 2011, accepted for publication.

[14] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Process. Syst. (NIPS)*, 2000.

[15] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[16] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *J. Machine Learning Research*, vol. 5, pp. 1457–1469, 2004.

[17] I. T. Jolliffe, *Principle Component Analysis*, 2nd ed. Springer, 2002.

[18] S. Mallat and F. Falzon, "Analysis of low bit rate image transform coding," *IEEE Trans. on Signal Processing*, vol. 46, no. 4, pp. 1027–1042, Apr. 1998.

[19] E. L. Pennec and S. Mallat, "Sparse geometric image representations with bandelets," *IEEE Trans. on Image Processing*, vol. 14, no. 4, pp. 423–438, Apr. 2005.

[20] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *SIGGRPAH 2000*, 2000.

[21] M. Bertalmio, A. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society Conference on*, vol. 1, Los Alamitos, CA, USA, Dec. 2001, pp. 355–362.

[22] T. Chan and J. Shen, "Local inpainting models and tv inpainting," *SIAM J. Appl. Math.*, vol. 62, no. 3, pp. 1019–1043, 2001.

[23] R. Bornard, E. Lecan, L. Laborelli, and J. Chenot, "Missing data correction in still images and image sequences," in *ACM Int. Conf. Multimedia*, Dec. 2002.

[24] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by examplar(based image inpainting," *IEEE Trans. On Image Processing*, vol. 13, pp. 1200–1212, 2004.

[25] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion," *ACM Trans. Graphics*, vol. 22, no. 2003, pp. 303–312, 2005.

[26] P. Harrison, "A non-hierarchical procedure for re-synthesis of complex texture," in *Proc. Int. Conf. Central Europe Comp. Graphics, Visua. and Comp. Vis.*, Feb. 2001.

[27] J. Jia and C. Tang, "Image repairing: Robust image synthesis by adaptive tensor voting," in *Proc. CVPR*, Jun. 2003, pp. 643–650.

[28] Y. Zhang, J. Xiao, and M. Shah, "Region completion in a single image," in *EUROGRAPHICS*, 2004.

[29] J. Sun, L. Yuan, J. Jia, and H. Shum, "Image completion with structure propagation," *ACM Trans. Graphics*, vol. 24, no. 3, pp. 861–868, 2005.

[30] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," in *Proc. ICCV*, Sep. 1999.

[31] D. Tschumperlé, "Fast anisotropic smoothing of multi-valued images using curvature-preserving pde's," *Int. Jounral of Comp. Vision*, vol. 68, no. 1, pp. 65–82, 2006.

[32] S. Di Zenzo, "A note on the gradient of a multi-image," *Computer Vision, Graphics, and Image Processing*, vol. 33, pp. 116–125, 1986.

[33] T. Brox, J. Weickert, B. Burgeth, and P. Mrázk, "Nonlinear structure tensors," *Image and Vision Computing*, vol. 24, pp. 41–55, 2006.

[34] J. Weickert, "Coherence-enhancing diffusion filtering," *International Journal of Computer Vision*, vol. 32, pp. 111–127, 1999.

[35] Y. Wexler, E. Shechtman, and E. Irani, "Space-time completion of video," *IEEE Trans. On PAMI*, vol. 29, no. 3, pp. 463–476, 2007.

[36] D. Tschumperlé and R. Deriche, "Vector-valued image regularization with pdes: a common framework for different applications," *IEEE Trans. on PAMI*, vol. 27, no. 4, pp. 506–517, April 2005.

[37] N. Kawai, T. Sato, and N. Yokoya, "Image inpainting considering brightness change and spatial locality of textures and its evaluation," in *PSIVT2009*, 2009, pp. 271–282.

[38] A. Telea, "An image inpainting technique based on the fast marching method," *Journal of Graphics, GPU, and Game Tools*, vol. 9, no. 1, pp. 23–34, 2004.

[39] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 28, no. 3, 2009.

[40] K.-J. Oh, S. Yea, and Y.-S. Ho, "Hole filling method using depth based in-painting for view synthesis in free viewpoint television and 3-d video," in *Picture Coding Symposium (PCS)*, Piscataway, NJ, USA, May 2009, pp. 233–236.

[41] I. Daribo and P.-P. B., "Depth-aided image inpainting for novel view synthesis," 2010.

[42] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 600–608, Aug. 2004.

[43] N. Kawai, T. Sato, and N. Yokoya, "Image inpainting considering brightness change and spatial locality of textures," in *Proc. Int. Conf. on Computer Vision Theory and Applications (VISAPP)*, vol. 1, 2008, pp. 66–73.

[44] S. Chan, H.-Y. Shum, and K.-T. Ng, "Image-based rendering and synthesis," *Signal Processing Magazine, IEEE*, vol. 24, no. 6, pp. 22–33, Nov. 2007.

[45] C. Zhang and T. Chen, "A survey on image-based rendering–representation, sampling and compression," *Signal Processing: Image Communication*, vol. 19, no. 1, pp. 1–28, Jan. 2004.

[46] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, Mar. 2004.

[47] G. Sourimant, "Depth maps estimation and use for 3dtv," INRIA Rennes Bretagne Atlantique, Rennes, France, Technical Report 0379, Feb. 2010.

[48] Q. H. Nguyen, M. N. Do, and S. J. Patel, "Depth image-based rendering from multiple cameras with 3d propagation algorithm," in *Proceedings of the 2nd International Conference on Immersive Telecommunications*, ser. IMMERSCOM '09, vol. 6. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 1–6.

[49] L. McMillan, "A list-priority rendering algorithm for redisplaying projected surfaces," University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, Tech. Rep. 95-005, 1995.

[50] P. Kauff, N. Atzpadin, C. Fehn, M. Müller, O. Schreer, A. Smolic, and R. Tanger, "Depth map creation and image-based rendering for advanced 3dtv services providing interoperability and scalability," *Signal Processing: Image Communication*, vol. 22, pp. 217–234, Feb. 2007.

[51] K. Müller, A. Smolic, K. Dix, P. Merkle, P. Kauff, and T. Wiegand, "View synthesis for advanced 3d video systems," *EURASIP Journal on Image and Video Processing*, p. 11, Nov. 2008.

[52] S. W. Hasinoff, S. B. Kang, and R. Szeliski, "Boundary matting for view synthesis," *Comput. Vis. Image Underst.*, vol. 103, pp. 22–32, Jul. 2006.

[53] M. Sarim, A. Hilton, and J.-Y. Guillemaut, "Wide-baseline matte propagation for indoor scenes," in *Conference Visual Media Production (CVMP), Proceedings of*, ser. CVMP '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 195–204.

[54] J. Wang and M. F. Cohen, "Image and video matting: a survey," *Found. Trends. Comput. Graph. Vis.*, vol. 3, pp. 97–175, Jan. 2007.

[55] Y. Mori, N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto, "View generation with 3d warping using depth information for ftv," *Image Commun.*, vol. 24, pp. 65–72, Jan. 2009.

[56] L. Do, S. Zinger, Y. Morvan, and P. H. N. de With, "Quality improving techniques in dibr for free-viewpoint video," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, May 2009, pp. 1 –4.

[57] S. Rusinkiewicz and M. Levoy, "Qsplat: a multiresolution point rendering system for large meshes," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., Jul. 2000, pp. 343–352.

[58] M. Zwicker, H. Pfister, J. van Baar, and M. Gross, "Ewa splatting," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 8, pp. 223–238, Jul. 2002.

[59] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent, "Floating textures," *Computer Graphics Forum (Proc. of Eurographics)*, vol. 27, no. 2, pp. 409–418, 2008, received the Best Student Paper Award at Eurographics 2008.

[60] J. Shade, S. Gortler, L.-w. He, and R. Szeliski, "Layered depth images," in *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, Jul. 1998, pp. 231–242.

[61] S.-U. Yoon, E.-K. Lee, S.-Y. Kim, and Y.-S. Ho, "A framework for representation and processing of multi-view video using the concept of layered depth image," *Journal of VLSI Signal Processing Systems for Signal Image and Video Technology*, vol. 46, pp. 87–102, Mar. 2007.

[62] K. Müller, A. Smolic, K. Dix, P. Kauff, and T. Wiegand, "Reliability-based generation and view synthesis in layered depth video," *Multimedia Signal Processing (MMSP), IEEE International 10th Workshop on*, pp. 34–39, Oct. 2008.

[63] V. Jantet, L. Morin, and C. Guillemot, "Incremental-ldi for multi-view coding," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, Potsdam, Germany, May 2009, pp. 1–4.

[64] I. Daribo and B. Pesquet, "Depth-aided image inpainting for novel view synthesis," *Multimedia Signal Processing (MMSP), IEEE International Workshop on*, pp. 167–170, Oct. 2010.

[65] J. Gautier, O. Le Meur, and C. Guillemot, "Depth-based image completion for view synthesis," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, May 2011.