Error Concealment using 3D Low-Rank Tensor Completion

Marine Froidevaux
marine.froidevaux@epfl.ch

Supervisors: Dr. Jonas Ballani and Prof. Daniel Kressner

June 4, 2015

Contents

1 Introduction

Error concealment is a field of signal processing which deals with corruption and loss of data and aims at reducing the deterioration of the signal.

As data are typically packetised for transmission over a network or storage on a hardware, any data loss or corruption usually results in errors in the form of missing blocks.

This report presents an error concealment algorithm for images and videos. This method reconstructs the corrupted patches by exploiting the low-rank property of a stack of similar patches. Results obtained by application of this algorithm to movie reconstruction and inpainting are presented here, as well as a comparison with results from another low-rank tensor approximation method.

2 Basics of tensor algebra

Before looking at the error concealment algorithm in details, a few basic tools of tensor algebra need to be defined. This section is based on the paper by Kolda and Bader [3], which should be referred back to for further details.

It is sufficient for our purpose to consider three-dimensional tensors only, but all algebraic concepts presented in this section can be generalised to higher dimensions.

Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. The ith matricisation of $\mathcal{X}$ is a reordering of its entries into a matrix $X_{(i)} \in \mathbb{R}^{I_i \times \prod_{j \neq i} I_j}$ such that the indices of $\mathcal{X}$ in the ith dimension become the row indices of $X_{(i)}$ and the indices of $\mathcal{X}$ in the other dimensions are rearranged in lexicographical order to become column indices of X_i.

The $i-$rank of $\mathcal{X}$ is defined as the rank of its ith matricisation. The multilinear rank of $\mathcal{X}$ is a 3-tuple grouping the i-rank of $\mathcal{X}$ in each of the three dimensions:

$$\text{rank}(\mathcal{X}) := \left[\text{rank}(X_{(1)}), \text{rank}(X_{(2)}), \text{rank}(X_{(3)})\right]$$

The ith-mode product of $\mathcal{X}$ with a matrix $M \in \mathbb{R}^{m \times I_i}$ is a tensor $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$, with $J_j = I_j$ for $j \neq i$, and $J_i = m$. $\mathcal{Y}$ is defined by means of the matricisation as follows:

$$\mathcal{Y} = \mathcal{X} \times_i M \iff Y_{(i)} = M X_{(i)}$$

Note that the matricisation operation is a bijection from $\mathbb{R}^{I_1 \times I_2 \times I_3}$ into $\mathbb{R}^{I_i \times \prod_{j \neq i} I_j}$. $\mathcal{Y}$ is thus uniquely determined by any of its matricisation.

The inner product of two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is the sum of the product of their entries, that is to say:

$$< \mathcal{X}, \mathcal{Y} > := \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \sum_{i_3=1}^{I_3} x_{i_1,i_2,i_3} y_{i_1,i_2,i_3}} \tag{1}$$

The induced norm $\|\mathcal{X}\| := \sqrt{< \mathcal{X}, \mathcal{X} >}$ is thus a generalisation of the matrix Frobenius norm to 3 dimensions.

Any tensor $\mathcal{X}$ of multilinear rank $[r_1, r_2, r_3]$ can be decomposed into a core tensor $\mathcal{C} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ and three orthogonal basis matrices $A_i \in \mathbb{R}^{I_i \times r_i}$, such that

$$\mathcal{X} = \mathcal{C} \times_1 A_1 \times_2 A_2 \times_3 A_3 \tag{2}$$

This is called the Tucker Decomposition and is the basis of the High-Order Singular Value Decomposition (HOSVD) algorithm.

Note that a Tucker decomposition is not unique. Indeed, if $\mathcal{X}$ can be decomposed as in Equation 2 then for any invertible $V_1 \in \mathbb{R}^{r_1 \times r_1}, V_2 \in \mathbb{R}^{r_2 \times r_2}, V_3 \in \mathbb{R}^{r_3 \times r_3}$, the following decomposition holds as well:

$$\mathcal{X} = \tilde{\mathcal{C}} \times_1 \tilde{A}_1 \times_2 \tilde{A}_2 \times_3 \tilde{A}_3$$

where $\tilde{\mathcal{C}} = \mathcal{C} \times_1 V_1 \times_2 V_2 \times_3 V_3$ and $\tilde{A}_i = A_i V_i^{-1}$.

HOSVD aims at computing a rank-$[R_1, R_2, R_3]$ approximation of a tensor $\mathcal{X}$ of higher rank in the Tucker format. This is done by matricising $\mathcal{X}$ in each dimension $i = 1, 2, 3$ and using the R_i principal left singular vectors as basis A_i of the Tucker decomposition. The pseudo-code for the HOSVD procedure is given in Algorithm 1.

Algorithm 1 Higher-Order Singular Value Decomposition

```
1: procedure HOSVD ($\mathcal{X}, R_1, R_2, R_3$)
2:     for $i = 1, 2, 3$ do
3:         $A_i \leftarrow R_i$ leading left singular vectors of $X_{(i)}$
4:     $\mathcal{C} \leftarrow \mathcal{X} \times_1 A_1^\top \times_2 A_2^\top \times_3 A_3^\top$
5:     return $\mathcal{C}, A_1, A_2, A_3$
```

This procedure can be viewed as a generalisation of the matrix Singular Value Decomposition (SVD) as a truncated SVD is performed on each matricisation of $\mathcal{X}$ to obtain a lower rank approximation. However, in contrast to the 2-dimensional case, the HOSVD does not lead to the best rank-$[R_1, R_2, R_3]$ approximation of $\mathcal{X}$. Let us denote the subspace of all 3-dimensional tensors of rank $r = [R_1, R_2, R_3]$ by $\mathcal{M}_r := \{\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times I_3} \mid \operatorname{rank}(\mathcal{T}) = [R_1, R_2, R_3]\}$ and let $P_{\mathcal{M}_r}\mathcal{X}$ be a best approximation of $\mathcal{X}$ in the subspace $\mathcal{M}_r$ relatively to the norm defined above. If $P_{HO}\mathcal{X}$ represents the result of the HOSVD procedure applied to $\mathcal{X}$ then the following results holds [2]:

$$\|\mathcal{X} - P_{HO}\mathcal{X}\| \leqslant \sqrt{3}\|\mathcal{X} - P_{\mathcal{M}_r}\mathcal{X}\|$$

This means that a best low-rank approximation of $\mathcal{X}$ is only better by a factor $\frac{1}{\sqrt{3}}$ than what is computed by means of the HOSVD procedure.

3 Statement of the problem

The description is made here for the case of movie reconstruction. Image inpainting can be viewed as a particular case of movie reconstruction where the number of frames is equal to 1.

Firstly, a Macro-Block (MB) grid is defined for each frame. The size of a MB, denoted by N, is typically 8×8 or 16×16 pixels and the MB grid is shifted by half a block size in the vertical and horizontal direction of the frame. A given MB therefore typically overlaps with 8 other MBs from the same frame (see Figure 1) and a natural division of each MB into 4 sub-blocks arises.

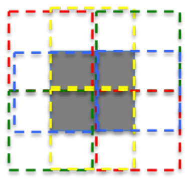

Figure 1: A given MB (grey) typically overlaps with 8 other MBs (red, yellow, blue and green contour) from the same frame.

3.1 Tensor construction via selection of similar patches

In each frame of the movie separately, all MBs containing corrupted entries are put in a list and sorted relatively to the number of clean (i.e. non-corrupted) pixels they contain. We call P^0 a corrupted MB and Ω the set of its clean and concealed pixels. P^0s with a larger set Ω will be reconstructed first.

Once a P^0 is chosen for reconstruction of its missing entries, similar MBs that are fully clean will be looked for in a some defined reference frames. We call these similar MBs P^i and seek to select the K P^is which best match P^0. The best matching P^i is defined as the one which minimises the error

$$\|P^0_\Omega - P^i_\Omega\|_1 = \sum_j |P^0_j - P^i_j| \tag{3}$$

where j is an index that loops over all pixels of Ω. All MBs are rescaled to have the same L_2-norm as P^0 (on the set Ω only) before the matching error is computed.

Note that K needs to be defined as a user input, but in case less than K "good matching" MBs are actually available in the reference frames, then only $K_{eff} < K$ MBs are effectively selected, in order to avoid taking into account random MBs which would pervert the results. In my particular implementation, any MB with an error more then twice bigger than the best matching P^i was discarded from the selection.

It is important here to realise that, generally, the higher the number of reference frames for the selection of P^is, the better the matching with P^0 and thus the better the reconstruction. However, the scanning of reference frames for detection of matching MBs is a very time-consuming process. The number of reference frames should therefore be a trade-off between computation time and quality of the results.

P^0 and all the P^is are then gathered into a 3-dimensional tensor $\mathcal{X}$ of size $N \times N \times K_{eff}$, where P^0 is the first slice of the tensor. Later in this report, K will be used in place of K_{eff} for the sake of simplicity, but it should be remembered that this K might be smaller than the value given as user input.

3.2 Low-rank property of the tensor

Ideally, if the objects in the movie were not moving and the camera was keeping the same view, then $\mathcal{X}$ would be composed of K times the same patch. It could therefore be represented as a product between a 2D patch P and a constant vector 1 in the 3rd dimension. Allowing some small sparse noise $\mathcal{E}$, it would write:

$$\mathcal{X} = \mathcal{X}_l + \mathcal{E} = P \times_3 1 + \mathcal{E} \tag{4}$$

After decomposition of $\mathcal{X}_l$ in the Tucker format

$$\mathcal{X}_l = \mathcal{C} \times_1 A_1 \times_2 A_2 \times_3 A_3$$

and comparison with Equation 4 we can write

$$P = \mathcal{C} \times_1 A_1 \times_2 A_2 \text{ and } A_3 = 1$$

In practice, $\mathcal{X}_l$ is not composed of K exactly identical patches and its rank can thus not be 1. It is however reasonable to assume that it still has a small 3-rank. In this light, two different algorithms have been used to find a good low-rank approximation of $\mathcal{X}$.

4 Description of the algorithms for low-rank approximation

4.1 Alternating Least Squares Algorithm (ALS)

We call now $\bar{\Omega}$ the set of missing entries in P^0.

The Alternating Least Square algorithm presented in the paper by Nguyen, Dao and Tran [1] starts by filling $P^0_{\bar{\Omega}}$ in the first slice of $\mathcal{X}$ with the mean values of the P^is at these locations (see Algorithm 2, line 2).

Algorithm 2 Alternating Least Square

```
1: Form $\mathcal{X}$ from $P^0, ..., K-1$
2: $(\mathcal{X}(:,:,1))_{\bar{\Omega}} = (\frac{1}{K-1}\sum_{i=1}^{K-1} P^i)_{\bar{\Omega}}$
3: Choose mode ranks $[R_1, R_2, R_3]$ and tolerance $\sigma_{rel}, \sigma_{it}$
4: Initialise $A_1 \in \mathbb{R}^{N\times R_1}, A_2 \in \mathbb{R}^{N\times R_2}, A_3 \in \mathbb{R}^{K\times R_3}$ randomly.
5: $A^3(:,1) = [1, ..., 1]^\top / K$
6: for $i = 1, 2, 3$ do
7:     $\mathcal{Y} = \mathcal{X}$
8:     for $j \neq i$ do
9:         $\mathcal{Y} = \mathcal{Y} \times_j A_j^\top$
10:    $Y_i \leftarrow$ unfold $\mathcal{Y}$ in mode $i$
11:    $A_i$=first $R_i$ principal components of $Y_i$
12: $\mathcal{C} = \mathcal{X} \times_1 A_1^\top \times_2 A_2^\top \times_3 A_3^\top$
13: $\mathcal{X}_l = \mathcal{C} \times_1 A_1 \times_2 A_2 \times_3 A_3$
14: If convergence is reached, stop ; else return to Step 5
15: Recover missing area in $P^0$: $(P^0)_{\bar{\Omega}} = (\mathcal{X}_l(:,:,1))_{\bar{\Omega}}$
```

While the multilinear rank R_3 is supposed to be small (it was set to 3 in the following examples), no assumption can be made on R_1 and R_2. These two variables are thus generally kept to their maximal possible value N.

The ALS procedure is a modified version of the HOSVD presented in Algorithm 1. It is an iterative process which updates the basis matrices A_i one by one, keeping the other two constant. In this way, A_3 can be enforced at each iteration to fulfil an additional condition. Indeed, in the ideal case described in subsection 3.2, A_3 was a constant vector 1. Since we look for a solution which is known to be close to the ideal case, convergence can be helped by forcing the first vector of A_3 to be constant and normalised to 1.

The procedure is considered to have reached convergence at iteration n if one of these two conditions is fulfilled:

1. The low-rank tensor $\mathcal{X}_l^{(n)}$ has become close enough to $\mathcal{X}$, i.e.

$$\epsilon_{rel}^{(n)} := \|\mathcal{X}_l^{(n)} - \mathcal{X}\| / \|\mathcal{X}\| < \sigma_{rel}$$

2. The change made during in 1 iteration has become negligible, i.e.

$$|\epsilon_{rel}^{(n)} - \epsilon_{rel}^{(n-1)}| < \sigma_{it}$$

4.2 GeomCG

The second algorithm, GeomCG, is the one presented in the paper written by Kressner, Steinlechner, Vandereycken [2]. Their implementation of the algorithm is available on their website [4] and is the one which was used here.

The aim of GeomCG is to minimise the cost function

$$f_{\mathcal{X}}(\mathcal{Y}) = \|P_\Omega \mathcal{Y} - P_\Omega \mathcal{X}\|^2$$

under the constraint $\mathcal{Y} \in \mathcal{M}_r$.

The algorithm is based on the fact that $\mathcal{M}_r$ is a smooth subspace of $\mathbb{R}^{I_1 \times I_2 \times I_3}$ and that, together with the inner product of Equation 1, it forms a Riemannian manifold. An extension of the non-linear Conjugate Gradient method is used to minimise the cost function $f_{\mathcal{X}}(\mathcal{Y}_k)$. The direction of optimisation and the step size are computed by projecting the euclidian gradient of $f_{\mathcal{X}}(\mathcal{Y}_k)$ into the tangent plan of $\mathcal{M}_r$ at position $\mathcal{X}$. After each optimisation the resulting tensor is retracted back onto $\mathcal{M}_r$, which ensures that the low-rank condition on $\mathcal{Y}$ remains satisfied.

5 Results

5.1 Bus Sequence

5.2 Inpainting

5.3 Effect of the corruption ratio

6 Conclusion

7 Suggestion for further research

References

[1] D.T. Nguyen, M.D. Dao and T.D. Tran. The John Hopkins University, 2011. *Error Concealment Via 3-Mode Tensor Approximation.* 18th IEEE Conference on Image Processing

[2] D.Kressner, M. Steinlechner and B.Vandereycken. École Polytechnique Fédérale de Lausanne, 2013. *Low-Rank Tensor Completion by Riemannian Optimization*

[3] T.G. Kolda and B.W. Bader. Sandia National Laboratories, 2009. *Tensor Decomposition and Applications.* SIAM Review, Vol.51, No.3, pp. 455-500

[4] http://anchp.epfl.ch/geomCG