universität
innsbruck

Bachelorthesis

Integrating low-rank components into weighted K-SVD for dictionary based inpainting

Markus Tiefenthaler

Date of submission: September 17, 2018

Supervision: Karin Schnass

Department for Mathematics Innsbruck

Contents

1 INTRODUCTION **3**

2 PROBLEM SETUP **5**

2.1 Signal model 6
2.2 Corruption model 6
2.3 Low-rank component 7
2.3.1 What is the low-rank component? 8
2.3.2 Why do we want to consider the low-rank component? 9
2.3.3 How do we find the low-rank component? 10

3 DICTIONARY RECOVERY **11**

3.1 K-SVD 12
3.2 Weighted K-SVD 15
3.3 Weighted K-SVDL 16
3.4 ITKrMM 17

4 DICTIONARY BASED INPAINTING **18**

5 APPLICATION AND RESULTS **18**

6 CONCLUSION AND FUTURE DIRECTIONS **25**

1 INTRODUCTION

The basic problem which is considered in this bachelor thesis is the reconstruction of a damaged image, also called inpainting.

Corrupted Picture Reconstruction

Figure 1: The basic problem is to find the original image.

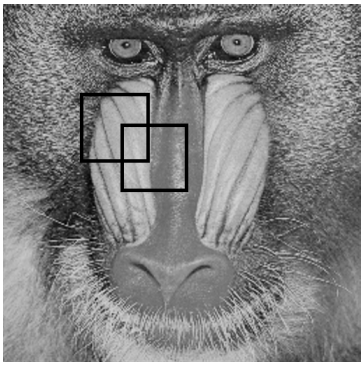
There are many different approaches to obtaining such a reconstruction, but we only address dictionary based inpainting. This approach relies on the assumption, that every $s_1 \times s_2$ patch of a picture (signal) can be approximated by a linear combination of a small number of elementary signals, called atoms. The atoms are drawn from a pre-specified representation system, like a basis or frame, called dictionary. If this assumption holds, we say that the image is sparse in the dictionary. In mathematical terms this means, that every vectorised patch $y_n \in \mathbb{R}^d$, with $d = s_1 \cdot s_2$, can be represented as a matrix vector multiplication

$$y_n = \Phi x_n \tag{1.1}$$

with $\Phi \in \mathbb{R}^{d \times K}$ as our dictionary. The dictionary contains K normalised vectors ϕ_k, called atoms. The atoms are stored as columns in the dictionary $\Phi = (\phi_1, ..., \phi_K)$. The vector $x_n \in \mathbb{R}^d$ has only $S \ll d$ non-zero entries and defines the small number of columns we take from Φ to represent y_n. Since equality is hard to fulfill, we are also satisfied with a good approximation. This model can now be applied on the whole picture by storing all the patches in a matrix $Y = (y_1, ..., y_N) \in \mathbb{R}^{d \times N}$, which leads us to a column wise sparse matrix X, so that

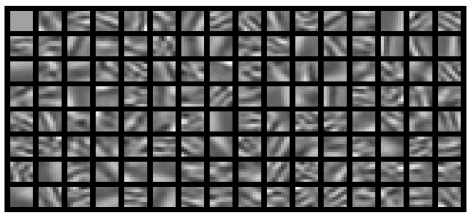
$$Y \approx \Phi X \tag{1.2}$$

The representation of Y in (1.2) has proven to be very useful in signal processing for tasks such as inpainting. For the history of sparse representation see [7].
To get a better understanding how this works for images we take a look at Figure 2. On the left hand side, we see the $s_1 \times s_2$ patches y_n, y_m of the picture *Mandrill*. On the right hand side, we see the dictionary Φ learned from all patches $(y_1, ..., y_N)$.



(a) Patches y_n,y_m in the picture

Learning
Approximating



(b) Dictionary Φ, where each square represents an atoms ϕ_k

Figure 2

Each square represents an atom ϕ_k. The connection between these illustrations is that we learn a dictionary from all patches and approximate each patch by a linear combination of a small amount of dictionary atoms.
There already exist multiple ways to learn the dictionary Φ from masked data, such as the weighted K-SVD algorithm (wKSVD), first introduced by Mairal, Elad and Sapiro [3] and the iterative thresholding and K residual means algorithm for masked data (ITKrMM), introduced by Naumova and Schnass [2]. One of their main differences is, that ITKrMM is able to consider low-rank components of arbitrary sizes, whereas wKSVD is only able to consider one dimensional ones. The low-rank component is a low dimensional subspace which contains most of the signal energy. Looking again at Figure 2 we see a dictionary with a one dimensional low-rank component. The low-rank atom is the grey square in the top left corner. The focus of this work will lie in answering the following questions:

- How can we modify wKSVD to account for arbitrary sized low-rank components?
- How does the consideration of the low-rank component influence the inpainting results?

The updated algorithm, weighted K-SVD low-rank component (wKSVDL), will then be compared to the already established ITKrMM.

Notation: Before we get deeper into the topic, it is necessary to clarify certain basic notations. We denote the transpose of a matrix A by A^* and its Moore-Penrose pseudo inverse by $A^\dagger$. The orthogonal projection onto the column span of the matrix A is denoted by $P(A) = AA^\dagger$. Further, by $Q(A)$ the orthogonal projection onto the orthogonal complement of the column span of A is meant, and it is calculated as $Q(A) = \mathbb{I}_d - P(A)$, where $\mathbb{I}_d$ is the identity operator (matrix) in $\mathbb{R}^d$.
The coherence μ of a dictionary Φ is defined as the maximal absolute inner product between two different atoms, $\mu = \max_{k \neq j} |\langle \phi_k, \phi_j \rangle|$. We call a dictionary incoherent, if

the coherence is small $\mu \ll 1$, meaning that the dictionary atoms are almost orthogonal. Further, the restriction of the dictionary to the atoms indexed by I is denoted by Φ_I, and thus $\Phi_I = (\phi_{i_1}, ..., \phi_{i_S})$, $i_j \in I$.
The representation of a masked signal My is defined as a matrix vector multiplication, where M is a diagonal matrix with ones and zeros in its diagonal, indicating which entries of the original signal y we receive and which are erased. On the other hand, for the whole masked data set, we write $\mathbf{M} \odot Y$. Here $\mathbf{M}$ is an object, in which we save all the masks $(M_1, ..., M_N)$ corresponding to the patches. The multiplication $\mathbf{M} \odot Y$ now is defined as $\mathbf{M} \odot Y = (M_1 y_1, ..., M_N y_N)$ and $\mathbf{M} \odot Y$ represents the whole set of damaged signals.

Orthogonal Matching Pursuit (OMP): After clarifying the basic terminology, we will discuss the Orthogonal Matching Pursuit algorithm (OMP) in more detail, since it is used frequently in this work. OMP, introduced by Pati, Rezaiifar and Krishnaprasad, [8], is a matching pursuit class algorithm. It iteratively selects atoms from the dictionary Φ to use in the representation for the signal y_n. In every step it uses the orthogonal projection onto the space spanned by the atoms selected so far as the current approximation. This orthogonalisation improves the stability and further ensures a fast convergence for this greedy algorithm. Moreover, OMP can find the true support of y_n if the dictionary is incoherent, see [9] for more details. For masked data, OMP has to be modified to take into account that the damaged dictionary is not normalised. Thus in every step a rescaling is done by $1/\|M\phi_k\|_2$. Without considering the mask in OMP less damaged atoms take precedence over better fitting ones.

Algorithm 1 (OMP masked).

Input: a damaged signal My together with the mask M, a dictionary Φ and a sparsity level S.
Initialisation: $r = My$, $I \neq \emptyset$ and while $|I| < S$. Repeat:

- Atom selection: find $j = \arg\max_{M\phi_k \neq 0} |\langle r, M\phi_k \rangle| / \|M\phi_k\|_2$
- Approximation: Set $I = I \cup \{j\}$, $x_I = (M\Phi_I)^\dagger My$ and $r = My - M\Phi_I x_I$

Output: x_I

Note that for $M = \mathbb{I}_d$ Algorithm 1 reduces to normal OMP.

2 PROBLEM SETUP

The goal of this work is to enable wKSVD to consider a low-rank component Γ, which will be extensively discussed in Subsection 2.3. This will be considered since in natural phenomenon the signals are often not perfectly sparse but can be modelled as the orthogonal sum of a low-rank component and a sparse component.
The wKSVD algorithm is able to learn a dictionary Φ from a set of corrupted data

$\mathbf{M} \odot Y$ under the condition that the signals $M_n y_n \in \mathbf{M} \odot Y$ are sparse in the dictionary Φ. To learn appropriately in this setup, we need two assumptions. The first is that the dictionary is robust to corruption. The dictionary is, for instance, not robust to corruption if an atom ϕ_k is in every representation, in which it is used, corrupted by the same mask M_0. Since ϕ_k is never fully observed it cannot be recovered. The consequence is that we need independence between the signal y_n and the corruption, represented by the masks M_n. The second assumption is that the incoherence of the dictionary is robust towards the corruption. Otherwise it will be very difficult to recover the sparse representations and in consequence the dictionary. Thus, we assume that the dictionary Φ as well as the low-rank component $\Gamma = (\gamma_1, ..., \gamma_L)$ consist of 'flat' atoms, where $\|\phi_k\|_\infty \ll 1$ respectively $\|\gamma_l\|_\infty \ll 1$ for $\gamma_l \in \Gamma$. These assumptions now lead us to the following signal model and corruption model.

2.1 Signal model

The basic signal model for natural images is taken from Naumova and Schnass, [2]. Given a $d \times L$ low-rank component Γ with $\Gamma^*\Gamma = \mathbb{I}_L$ and a $d \times K$ dictionary Φ, where $\Gamma^*\Phi = 0$, a sparsity level S and $L \ll K$ the signals are generated as,

$$y = s \cdot \frac{\Gamma v + \Phi x + r}{\sqrt{1 + \|r\|_2^2}} \approx s(\Gamma v + \Phi_I x_I), \tag{2.1}$$

where $\|v\|_2^2 + \|x\|_2^2 = 1$, meaning that the vector (v, x) is normalised. Further, $|I| = S$ holds, r is noise in the signal and s is a scaling parameter that accounts for signals with different energy levels.
For our purpose we will consider no noise and set s constant. Leading to the simplified model

$$y = \Gamma v + \Phi_I x_I \tag{2.2}$$

Since we assume that the low-rank component is present in (almost) every signal, the coefficients v are dense. Further, it should be irreducible, meaning that $\mathbb{E}(vv^*)$ is a diagonal matrix. On the other hand, the coefficients x should be sparse and in addition should be distributed in a way that for every single signal only S entries in x are effectively non-zero. Moreover, the average contribution of a sparse atom should be smaller than that of a low-rank atom, $\mathbb{E}(|x(k)|) \ll \mathbb{E}(|v(l)|), \forall k, l$.
After discussing the signal model, let us summarise the assumptions underlying the masks.

2.2 Corruption model

As already seen, we model the corruption of the signal y by applying a mask M, where we assume that the distribution of the mask is independent of the signal distribution. In the *Peppers* image from Figure 1, for example, this means that the mask is not allowed to destroy one pepper completely, where guessing the shade of grey would be almost impossible. Further, we assume that we have access both to the corrupted signal My,

and the location of the corruption in form of the mask M, meaning we receive the pair (My, M).
In our experiments we will consider two types of mask. On the one hand, we used random erasures. Meaning that the j-th coordinate is received with probability η_j independently of the reception of the other coordinates. Mathematically, this means that $M(j, j) \sim B(\eta_j)$ are independent Bernoulli variables. On the other hand, we used a pre-build structured mask, called *Cracks*, as in Figure 1.
Finally, we can discuss the theoretical core of this work, the low-rank component, extensively.

2.3 Low-rank component

One of the advantages of ITKrMM is that it is able to consider low-rank components of arbitrary sizes, whereas wKSVD is only able to consider one dimensional low-rank components. Thus, in order to compare the two algorithms properly, both should be able to use low-rank components of arbitrary sizes. But before we get to the point, where we will integrate this ability into wKSVD, we first need to answer these three basic questions,

1. What is the low-rank component?
2. Why do we want to consider the low-rank component?
3. How do we find the low-rank component?

which will be done in the following three subsections.

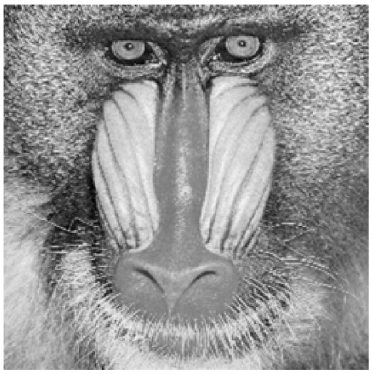
(a) Recovered low-rank component (b) DCT basis functions

Figure 3

2.3.1 What is the low-rank component?

The low-rank component $\Gamma \in \mathbb{R}^{d \times L}$ is a low dimensional subspace, which contains most of the signal energy. This means that in the simplified signal model in (2.2) v is a dense vector and thus almost all low-rank atoms are used to represent the signal. Another understanding is that it is a generalisation of the signal mean. The following visualisation of the first three low-rank atoms from a greyscaled picture should make this understanding observable. In Subfigure 3a, we see the first three low-rank atoms learned from the uncorrupted picture *Peppers*, as the first three left singular vectors of the data matrix. The first atom is used to represent the primary grayscale, underlying each patch or in other words the signals mean. The second and third atoms resemble the energy difference from left to right respectively from bottom to top. Further, notice the resemblance of the recovered low-rank atoms in 3a to the first three basis functions of the discrete cosine transform illustrated in Subfigure 3b. The discrete cosine transform is for example used by JPEG for image compression.

(a) Peppers

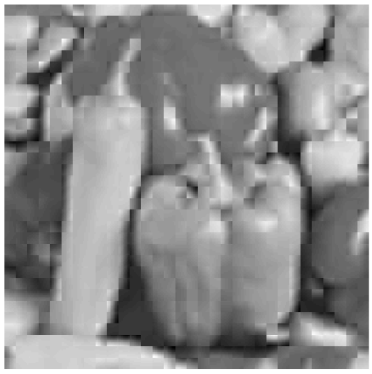


(b) Mandrill

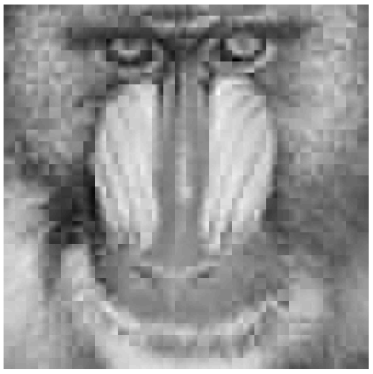
(c) Peppers, $L = 1$

(d) Mandrill, $L = 1$

Figure 4



(e) Peppers, $L = 3$



(f) Mandrill, $L = 3$

Figure 4

To show the power of the low-rank component, we look at Figure 4. To construct the Subfigures 4c respectively 4d we divided the original image in non-overlapping patches to learn the low-rank component and then approximated each patch with the first low-rank atom. We can observe that by using just the signal means for every patch we already get a recognisable reconstruction. The result significantly improves if we use the first three low-rank atom to approximate the original image, as we see in the Subfigures 4e respectively 4f. This leads to the following interpretation that most of the image energy is captured by the low-rank component. Now that we answered the first question, it is time to discuss the benefits we expect to obtain from it.

2.3.2 Why do we want to consider the low-rank component?

The straightforward answer to this question is that if in nature the signals tend to not being perfectly sparse but can be modelled as the orthogonal sum of a low-rank component and a sparse component, we want to model them exactly the same in our algorithm. Further, ignoring the low-rank component leads to a coherent dictionary since the atoms are drawn towards the low-rank component. The explanation for this is that the atoms want to capture a good part of the signal energy and as we have seen in Question 2.3.1 the low-rank component captures most of the signal energy. Further, the sparse approximation gets easier since the first L atoms for the representation are clearly determined. To get a better understanding how the two components in the signal model (2.2) work we visualise the equation. For the following equation we split the image *Peppers* up into its three dimensional low-rank component and the sparse component. Notice that the low-rank component focusses on the contrast, whereas the sparse component represents the contour of the picture.

Besides an incoherent dictionary, we additionally get two benefits. First, we can use the natural signal model and secondly the dictionary recovery splits up in a rough (low-rank component) and a precise (sparse) part, making the sparse approximation easier. In Section 5 we will see that considering the low-rank component leads to an increased performance in inpainting.

2.3.3 How do we find the low-rank component?

Now that we know what the low-rank component is and why we want to consider it, we come to the hard part of finding it. To do so, we have to distinguish between two cases, learning from complete data and learning from corrupted data. The first case can be approached straightforwardly by using SVD on the data set and selecting as for example the first L left singular vectors as our low-rank component. After subtracting its contribution from the signals via $\tilde{y}_n = Q(\Gamma)y_n$, where $Q(\Gamma)$ is the orthogonal projection onto the orthogonal complement of the column span of Γ, we can learn the dictionary from the modified signals $\tilde{y}_n$. This results in a dictionary orthogonal to the low-rank component $\Phi^*\Gamma = 0$ and since the columns of Γ are drawn from a unitary matrix $\Gamma^*\Gamma = \mathbb{I}_L$ holds.
Learning the low-rank component from corrupted data, gets more complicated, because we cannot simply use SVD anymore. The problem that prohibits a direct application of SVD is that the corruption will distort the left singular vectors in the direction of the more frequently observed coordinates. This means that it is impossible to recover the correct low-rank component. Since this naive approach does not work, we have to think of something else. If we take another look at the Signal Model (2.2) we see, that the low-rank component works like a dictionary. To understand this, let us assume that there is only one low-rank atom, $L = 1$. In 2.3.1 it is shown that the low-rank component captures a good part of the signals energy. One interpretation of this property is to say that all masked signals are 1-sparse in a dictionary of one masked atom. This atom can be recovered by an adapted masked dictionary learning algorithm and further low-rank atoms can now be iteratively recovered by considering the already estimated low-rank component in the algorithm. The following algorithm was proposed and tested in [2] and is summarised in Algorithm 2.

Algorithm 2 (low-rank atom recovery from corrupted data - one iteration).

Given an estimate of the previously recovered low-rank component $\tilde{\Gamma} = (\tilde{\gamma}_1 ... \tilde{\gamma}_{l-1})$, an input low-rank atom $\hat{\gamma}_l$ and N corrupted training signals $y_n^M = (M_n y_n, M_n)$ do:

- For all n set $M_n \tilde{y}_n = Q(M_n \tilde{\Gamma}) M_n y_n$
- Calculate

$$\bar{\gamma}_l = \sum_n [\mathbb{I}_d - P(M_n(\tilde{\Gamma}, \hat{\gamma}_l)) + P(M_n \hat{\gamma}_l)] M_n \tilde{y}_n \cdot \text{sign}(\langle \hat{\gamma}_l, M_n \tilde{y}_n \rangle)$$

$$\text{and } W = \sum_n M_n$$

- Set $\bar{\bar{\gamma}}_l = Q(\tilde{\Gamma}) W^\dagger \bar{\gamma}_l$ and output $\bar{\bar{\gamma}}_l / \|\bar{\bar{\gamma}}_l\|_2$

In the first step, this algorithm subtracts the contribution of the estimated masked low-rank component from the masked data. In the second step, it updates the input low-rank atom by using K residual means. Finally, it projects the updated atom on the orthogonal complement of the columns span of Γ, ensuring that the recovered atom does not share any contribution with the already recovered low-rank component and renormalises.

3 DICTIONARY RECOVERY

Now that the basics have been settled, we can finally discuss the dictionary recovery algorithms ITKrMM and wKSVD, which will be compared. Both algorithms find the dictionary by approximately solving a minimisation problem. The K-SVD has the following objective function

$$\min_{\Phi, X} \|Y - \Phi X\|_F^2 \quad \text{subject to} \quad \forall i, \|x_i\|_0 \leq S \tag{3.1}$$

Where $\|.\|_F$ defines the Frobenius norm, S defines the sparsity level of the coefficient matrix X and the norm $\|.\|_0$ counts the nonzero entries in the vector x_i. The ITKrMM algorithm, on the other hand, can be interpreted to minimise the following sum

$$\min_{\Phi, X} \sum \|y_n - \Phi x_n\|_2 \quad \text{subject to} \quad \forall i, \|x_i\|_0 \leq S \tag{3.2}$$

The following table provides an overview on how these algorithms work.

wKSVD weighted K singular value decomposition	ITKrMM iterative thresholding and K residual means for masked data
1. iterative algorithm	1. iterative algorithm
2. uses any pursuit algorithm for sparse approximation (often OMP)	2. uses thresholding as sparse approximation algorithm
3. uses the singular value decomposition for the dictionary update	3. uses residual averages for the dictionary update
4. works with one dimensional low-rank component	4. works with arbitrary dimensional low-rank components
5. expensive	5. cheap

To be able to compare those algorithms appropriately, weighted K-SVD should also be able to handle multi dimensional low-rank components. Since there is a lot of work behind wKSVD, we approach it step by step and talk first about the K-Singular Value Decomposition (K-SVD).

3.1 K-SVD

The K-SVD algorithm was introduced by Aharon, Elad and Bruckstein [1]. It approximates a solution of the minimisation problem 3.1. The objective function $\|Y - \Phi X\|_F^2$ is non-convex due to the bi-linearity between the dictionary Φ and the coefficients X. One approach to minimise such a function is the block coordinate descent. Instead of minimising Φ and X together, the block coordinate descent minimises the error over one block coordinate while fixing the other. This is the basic concept of K-SVD. In other words, K-SVD is an iterative algorithm which alternates between two steps:

1. Sparse approximations
2. Dictionary update

To understand, why these steps solve the given minimisation problem 3.1, let us first consider the sparse coding stage. We assume that Φ is fixed, and calculate the support matrix X so that the following penalty term is minimised.

$$\|Y - \Phi X\|_F^2 = \sum_{i=1}^{N} \|y_i - \Phi x_i\|_2^2 \tag{3.3}$$

The second representation of the penalty term shows that we can decouple the problem 3.4 to N distinct problems of the form

$$\min_{x_i} \|y_i - \Phi x_i\|_2^2 \quad \text{subject to} \quad \|x_i\|_0 \leq S, \quad \text{for} \quad i = 1, 2, ..., N. \tag{3.4}$$

Note that sparse approximation is non-convex due to the 0-pseudonorm. Further, finding the optimal solution for this problem is NP hard since our dictionary Φ is redundant. Thus, the approach is finding a suboptimal solution by using a pursuit algorithm. In this work we use OMP, where the masked version was introduced in Algorithm 1. The OMP algorithm selects iteratively atoms from the dictionary to use for the approximation of y_n. Leading to $y_n \approx \Phi x_n$ where only S entries of x_n are non-zero.
In the second step, the dictionary update, we assume that except for one atom ϕ_k and its corresponding coefficients both X and Φ are fixed. The coefficients that correspond to ϕ_k are in the kth row of X which we denote by x_T^k. The multiplication $\phi_k x_T^k$ results in a $d \times N$ matrix, with the dictionary atom in each column j for which $x_T^k(j) \neq 0$. We return to the objective function 3.1 and see that the penalty term can be written as

$$\begin{aligned} \|Y - \Phi X\|_F^2 &= \|Y - \sum_{j=1}^{K} \phi_j x_T^j\|_F^2 \\ &= \|(Y - \sum_{j \neq k} \phi_j x_T^j) - \phi_k x_T^k\|_F^2 = \|E_k - \phi_k x_T^k\|_F^2 \end{aligned} \tag{3.5}$$

Now we have separated one term, which we want to improve, whereas the other $K-1$ terms are assumed fixed. Note that $\phi_k x_T^k \in \mathbb{R}^{d \times N}$ is a rank one matrix. To minimise the new penalty term 3.5, one would suggest to use SVD to update ϕ_k respectively x_T^k, because SVD finds the closest rank one approximation to E_k. The problem with this approach is that we may lose the sparsity of x_T^k. To avoid this, we just look at the signals which use the atom ϕ_k. Thus, we define $\tilde{\omega}_k$ as the group of indices where $x_T^k(i)$ is nonzero. This means that we put the focus on the signals that use the atom ϕ_k for its representation and ignore the rest.

$$\tilde{\omega}_k = \{i | 1 \leq i \leq N, x_T^k(i) \neq 0\}$$

To properly use the new set, we define ω_k as tupel

$$\omega_k = (i_1, ..., i_{|\tilde{\omega}_k|}) \quad \text{with} \quad i_n \in \tilde{\omega}_k \quad \text{and} \quad i_1 < ... < i_{|\tilde{\omega}_k|} \tag{3.6}$$

Now we can define Ω_k as a matrix of size $N \times |\tilde{\omega}_k|$, where the $(w_k(i), i)$th entries are ones and the others are set to zero. Multiplying $x_R^k = x_T^k \Omega_k$ allows us to throw away all the zero entries of the row vector x_T^k, leading to our shrunken row vector x_R^k with length $|\tilde{\omega}_k|$. Applying Ω_k on our data Y, creates a matrix $Y_k^R = Y\Omega_k$ of size $d \times |\tilde{\omega}_k|$. This matrix contains all the signals in Y that use ϕ_k for its representation. Exactly the same happens with $E_k^R = E_k \Omega_k$, where only the error columns are selected that correspond to signals that use the atom ϕ_k. Keeping this in mind, we can return to 3.5, but this time we force our solution to have the same support as x_T^k. This is equivalent to the minimisation of

$$\|E_k \Omega_k - \phi_k x_T^k \Omega_k\|_F^2 = \|E_k^R - \phi_k x_R^k\|_F^2 \tag{3.7}$$

Algorithm 3 (The K-SVD-algorithm).

Find the best dictionary to represent the data samples $\{y_i\}_{i=1}^N$ as sparse composition, by solving

$$\min_{\Phi,X} \|Y - \Phi X\|_F^2 \quad \text{subject to} \quad \forall i, \|x_i\|_0 \leq S \tag{3.8}$$

Initialisation: Set the dictionary matrix $\Phi^{(0)} \in \mathbb{R}^{n\times K}$ with l^2 normalised columns. Set $J = 1$ Repeat until convergence (stopping rule or fixed number of iterations):

- Sparse Coding Stage: Use OMP to compute the representation vectors x_i for each example y_i,
- Dictionary Update Stage: For each column k=1,2,...,K in $\Phi^{(J-1)}$, update it by
  - Define the tupel of examples that use this atom, $\omega_k = (i_1, ..., i_{|\tilde{\omega}_k|})$ as defined in (3.6)
  - Compute the overall representation error matrix, E_k, by

$$E_k = Y - \sum_{j\neq k} \phi_j x_T^j \tag{3.9}$$

  - Restrict E_K by choosing only the columns corresponding to ω_k, and obtain E_k^R.
  - Apply SVD decomposition $E_k^R = U\Lambda V^T$. Choose the updated dictionary column $\tilde{\phi}_k$, to be the first column of U. Update the coefficient vector x_R^k to be the first column of V multiplied by λ_1
- Set J=J+1

This problem can again be addressed directly with SVD. Applying SVD on our restricted error matrix E_k^R, we get $E_k^R = U\Lambda V^T = \sum_{i=1}^d \lambda_i u_i v_i^T$. The Theorem of Schmidt-Mirsky now tells us, that $\lambda_1 u_1 v_1^T$ is the best rank one approximation to E_k^R, with $\lambda_1 = \Lambda(1,1) \geq \lambda_i \quad \text{for} \quad i \in \{1, ..., d\}$. We define the solution for $\tilde{\phi_k}$ as the first column of U, and the coefficient vector x_R^k as the first column of V multiplied by λ_1. Note that $\tilde{\phi_k}$ is normalised since U is a unitary matrix. Further, the support either stays the same or gets smaller by possible nulling of terms. Putting this all together we get to the K-SVD Algorithm 3 proposed by Aharon, Elad and Bruckstein [1]. Since in inpainting we do not have access to the original image Y, but only to its corrupted version MY, it would be unwise to use K-SVD to learn the dictionary. Thus, in the next section we present the sequel, weighted K-SVD (wKSVD), which will be able to learn the dictionary from corrupted signals $M_n y_n$.

3.2 Weighted K-SVD

Now that we know an algorithm that finds a dictionary from uncorrupted data, we want to extend this algorithm and make it applicable for corrupted data. The extension known as weighted K-SVD was introduced by Mairal, Elad and Sapiro [3]. The idea is that if the uncorrupted signal y_i is well approximated by Φx_i, then the masked signal $M_i y_i$ should also be well approximated by the masked dictionary $M_i \Phi x_i$. We already know a OMP version, seen in Algorithm 1, that handles the masked sparse approximation. Thus we can solve the following problem.

$$\min_{x_i} \|M_i y_i - M_i \Phi x_i\|_2^2 \quad \text{subject to} \quad \|x_i\|_0 \leq S$$

In the second step, the learning step, we proceed at first like in K-SVD. This means that we fix X and Φ except for one dictionary atom, and thus we can write the penalty term as

$$\begin{aligned} \|\mathbf{M} \odot (Y - \Phi X)\|_F^2 &= \|\mathbf{M} \odot (Y - \sum_{j=1}^{K} \phi_j x_T^j)\|_F^2 \\ &= \|\mathbf{M} \odot Y - \sum_{j \neq k} \mathbf{M} \odot \phi_j x_T^j - \mathbf{M} \odot \phi_k x_T^k\|_F^2 \\ &= \|\mathbf{M} \odot E_k - \mathbf{M} \odot \phi_k x_T^k\|_F^2 \end{aligned} \tag{3.10}$$

Again we define the group of relevant indices $\tilde{\omega}_k = \{i | 1 \leq i \leq K, x_T^k(i) \neq 0\}$ as well as the tupel $\omega_k = (i_1, ..., i_{|\tilde{\omega}_k|})$ identical to Subsection 3.1. Further, we set Ω_k as a matrix of size $N \times |\tilde{\omega}_k|$, with ones on the $(\omega_k(i), i)th$ entries and zeros elsewhere. This leads to the new minimisation problem

$$\min \|(\mathbf{M} \odot E_k)\Omega_k - (\mathbf{M} \odot \phi_k x_T^k)\Omega_k\|_F^2 = \min \|\mathbf{M}^R \odot E_k^R - \mathbf{M}^R \odot \phi_k x_R^k\|_F^2$$

$\mathbf{M}^R$ denotes the restriction on the masks corresponding to E_k^R. Meaning that, we save the masks $(M_i)_{i \in \omega_k}$ in $\mathbf{M}^R$.
Since the masks prohibit using simple SVD, a truncated SVD is used to solve this weighted rank-one approximation, leading to an iterative algorithm which gives an approximation of a local minimum. This weighted rank-one approximation is shown in Algorithm 4.
In this algorithm an Expectation-Maximisation (EM) procedure to find (ϕ_{new}, x_{new}) is used. In the expectation step we fill in the current estimate of the solution for the missing values in the matrix $\mathbf{M}^R \odot E_k^R$. In the maximisation step we use SVD to re-estimate a rank one approximation (ϕ_{new}, x_{new}). A detailed discussion regarding the convergence of this method can be found in [4]. In the experiments we used 10 iterations to update the dictionary atom.
The weighted rank-one approximation now takes the part of SVD in K-SVD, leading to weighted K-SVD. The final problem is that wKSVD is only able to consider one dimensional low-rank components, even though natural images contain components of arbitrary sizes.

Algorithm 4 (weighted rank-one approximation algorithm).

Input: Penalty term E_k^R, corresponding mask M^R and a number of iterations J
Output: ϕ_{new} ($d \times 1$ vector), x_{new} ($|\tilde{\omega}_k| \times 1$ vector)
Problem: Find a better dictionary atom ϕ_k by solving

$$\arg\min_{\phi_k, x_R^k} \{\|\mathbf{M}^R \odot E_k^R - \mathbf{M}^R \odot \phi_k x_R^k\|_F^2\}$$

Initialization: $\phi_{new} = 0$, $x_{new} = 0$.
Loop: For i from 1 to J, use a truncated SVD to solve:

- $(\phi_{old}, x_{old}) = (\phi_{new}, x_{new})$.
- $(\phi_{new}, x_{new}) = \arg\min_{\phi, x} \|\mathbf{M}^R \odot E_k^R + (1_{d\times|\tilde{\omega}_k|} - \mathbf{M}^R) \odot \phi_{old} x_{old}^T - \phi x^T\|_F^2$

where $1_{d\times|\tilde{\omega}_k|}$ is a $d \times |\tilde{\omega}_k|$ matrix filled with ones.

Thus, we get to the practical core of this work the weighted K-SVD-low-rank-component algorithm (wKSVDL), where we enable wKSVD to consider low-rank components of arbitrary sizes.

3.3 Weighted K-SVDL

The last step and our contribution is to consider the possible existence of a low-rank component $\Gamma \in \mathbb{R}^{d\times L}$. If the signal contains a low-rank component, then the masked signal contains a masked low-rank component, meaning $My_i = M\Gamma v + M\Phi x_i$. In the sparse approximation step, we still use OMP. This is legitimate because we assume that Γ is present in all signals and thus OMP always chooses its atoms for the representation. Further, this means that the first L support indices are fixed, and only $S - L$ are freely chosen from the dictionary for the sparse representation. In the second step, the component atoms are not updated. This is easily done since wKSVD updates atom by atom. For the dictionary update we have to consider the upcoming problem which is that the mask destroys the orthogonality between dictionary and component. However, this problem can be solved by ensuring the orthogonality in every iteration by simply removing its contribution in the updated dictionary atom ϕ_k. This is done by a multiplication with the orthogonal projection onto the orthogonal complement of the column span of the component, $\phi_k \leftarrow Q(\Gamma)\phi_k$. After this we have to renormalise the atom, $\phi_k \leftarrow \phi_k / \|\phi_k\|_2$. These thoughts lead to the following extended algorithm, summarised in Algorithm 5.
This modified algorithm can now be used for dictionary inpainting which also considers a low-rank component. Before we are going to test its performance, we introduce a second algorithm for comparison.

Algorithm 5 (The wKSVDL-algorithm).

Input: Estimated $d \times L$ low-rank component $\tilde{\Gamma}$, dictionary Φ with $\Phi^* \tilde{\Gamma} = 0$, $\Psi = (\Gamma, \Phi)$ sparsity level S and N corrupted training signals $y_n^N = (M_n y_n, M_n)$
Repeat until convergence (stopping rule or fixed number of iterations):

- Sparse Coding Stage: Use OMP to compute the representation vectors
- Dictionary Update Stage: For each column k=L+1,...,K in $\Psi^{(J-1)}$, update it by
  - Define the tupel of examples that use this atom $\omega_k = (i_1, ..., i_{|\tilde{\omega}_k|})$ as defined in (3.6)
  - Compute the overall masked representation error matrix, ME_k and restrict it by choosing only the columns corresponding to ω_k, and obtain E_k^R and M^R.
  - Solve the minimisation problem $\min_{\psi_k, x_R^k} \|M^R E_k^R - M^R \psi_k x_R^k\|_F^2$ with weighted rank one approximation.
  - Ensure orthogonality between the updated atom and the component $\psi_k \leftarrow Q(\tilde{\Gamma})\psi_k$ and that the dictionary is normalised $\psi_k \leftarrow \psi_k / \|\psi_k\|_2$.

3.4 ITKrMM

The iterative thresholding and K residual means for masked data algorithm (ITKrMM), introduced by Naumova and Schnass [2], is an extension of ITKrM. It has got the same framework as K-SVD, meaning that both alternate between sparse approximation and updating the dictionary based on it. Nevertheless, the way to do so is significantly different. As the name suggests, ITKrMM uses thresholding to sparsely approximate the signals in the current version of the dictionary. Thresholding is compared to OMP rather simple since it straightforwardly selects the S dictionary atoms with the largest absolute inner product with the signal. In the second step, ITKrMM uses residual averages to update the dictionary. ITKrMM is summerised in Algorithm 6. A detailed discussion on how this algorithm works can be found in the paper of Naumova and Schnass [2].
The approach to handle the low-rank component is slightly different to the one we already discussed. The estimated low-rank component is removed from the signal before learning the dictionary, with a orthogonal projection onto the the orthogonal complement of the column span, $M_n \tilde{y}_n = Q(M_n \tilde{\Gamma}) M_n y_n$. Further, in the update stage, one has to replace the projection onto the current estimate of the corrupted atoms in the support with the projection onto these and the estimated corrupted low-rank component, $P(M_n \Phi_{I_n^t}) \rightarrow P(M_n(\tilde{\Gamma}, \Phi_{I_n^t}))$. After receiving a new dictionary atom, one has (similar to wKSVDL) to ensure the orthogonality between dictionary and component and renormalise the dictionary. This leads to the following algorithm. Since we now know how to recover dictionaries, we want to use those to solve inpainting problems. In the next section we explain the approach of dictionary based inpainting.

Algorithm 6 (ITKrMM-one iteration).

Input: Estimate of the low-rank component $\tilde{\Gamma}$, dictionary Φ with $\Phi^*\tilde{\Gamma} = 0$, a sparsity level S and N corrupted training signals $y_n^M = (M_n y_n, M_n)$ Repeat:

- For all n set $M_n\tilde{y}_n = Q(M_n\tilde{\Gamma})M_n y_n$.
- For all n find $I_n^t = \arg\max_{I:|I|=S} \sum_{i\in I: M_n\phi_i \neq 0} \frac{|\langle M_n\phi_i, M_n\tilde{y}_n\rangle|}{\|M_n\phi_i\|_2}$.
- For all k calculate

$$\bar{\phi}_k \leftarrow \sum_{n:k\in I_n^t} [\mathbb{I}_d - P(M_n(\tilde{\Gamma}, \Phi_{I_n^t})) + P(M_n\phi_k))]M_n\tilde{y}_n \cdot \operatorname{sign}(\langle \phi_k, M_n\tilde{y}_n\rangle)$$
$$\text{and } W_k = \sum_{n:k\in I_n^t} M_n$$

- Set $\bar{\bar{\phi}}_k \leftarrow Q(\tilde{\Gamma})W_k^\dagger\bar{\phi}_k$ and output $\bar{\bar{\Phi}} \leftarrow (\bar{\bar{\phi}}_1/\|\bar{\bar{\phi}}_1\|_2, ..., \bar{\bar{\phi}}_K/\|\bar{\bar{\phi}}_K\|_2)$.

4 DICTIONARY BASED INPAINTING

The idea for dictionary based inpainting is that in case an image patch y is sparse in a dictionary, then the masked/damaged patch My is sparse in the masked/damaged patch dictionary $M\Phi$. For $|I| \leq S$ this means the following

$$y \approx \Phi_I x_I \Longrightarrow My \approx M\Phi_I x_I. \tag{4.1}$$

This implication shows that for the reconstruction of a patch, we simply need to recover the coefficients $\tilde{x}_I \approx x_I$ by sparsely approximating My in $M\Phi$ and to set $\tilde{y} = \Phi\tilde{x}_I$. The flatter the dictionary is, the more stable the atoms are to erasures and the easier it is to recover the correct coefficients. Since we split the damaged image up into overlapping patches y_n, we reconstruct those via sparse approximation and then reconstruct the full image by averaging every pixel over all reconstructed patches in which it is contained. We use OMP as sparse approximation algorithm, which was already used by wKSVD respectively wKSVDL for the sparse approximation.

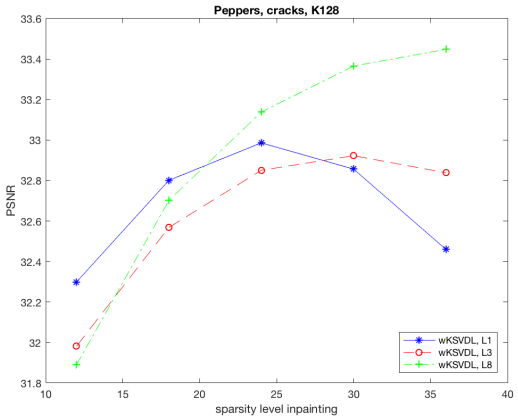
5 APPLICATION AND RESULTS

After a full discussion of the theory, we are ready to test the new developed algorithm in practice. The goal is to compare the algorithm to get optimised results for image inpainting. As already mentioned, inpainting is the process of filling in missing pixels in a damaged image. Such damaged images can appear, in particular, as old analogue images or as images, where one wants to remove objects, like text or date stamps.
As mentioned above, we are comparing wKSVDL to ITKrMM since those two bear several similarities. Further, the effects of considering different sized low-rank compo-

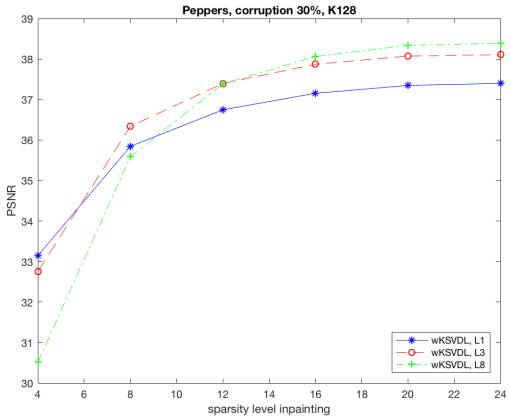
nents will be shown. This leads us to the following setup for dictionary based inpainting.

Data: The images we consider for the experiments are grayscale images of size 256×256 from a standard image database. As mentioned in the corruption odel 2.2, we apply a random mask on the images which erases pixel with probability $0.3, 0.5, 0.7$, leading to images where 30%, 50% ,70% of the pixels are missing on average. Besides the random erasures, we use a structured mask, *Cracks*, to remove the pixels from the image, see Figure 10b. Since we want to learn the dictionary from a set of signals, we extract all possible patches of size $p \times p$ pixels from the corrupted image. This results in the pair $(M_n y_n, M_n)$, where $M_n y_n$ is a vectorised patch and M_n the vectorised patch of the corresponding mask. This pair then is given to the dictionary learning algorithm. For the experiment we set $p = 8$ in case of the random mask and $p = 12$ for the structured mask.

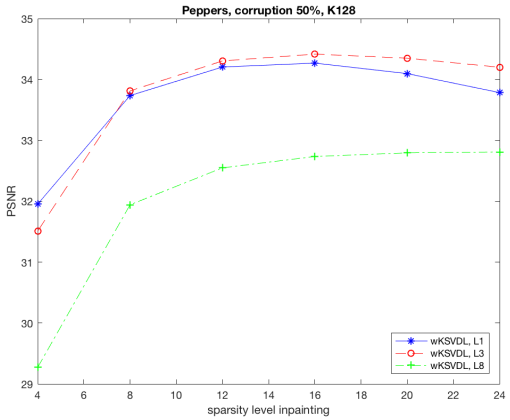
Dictionary & low-rank component: Both algorithms learn dictionaries of size



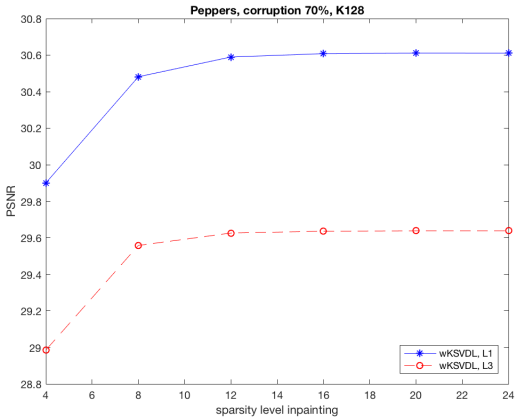
(a) *Cracks*



(b) 30% erasures



(c) 50% erasures



(d) 70% erasures

Figure 5: PSNR for different sizes of the low-rank component in wKSVDL

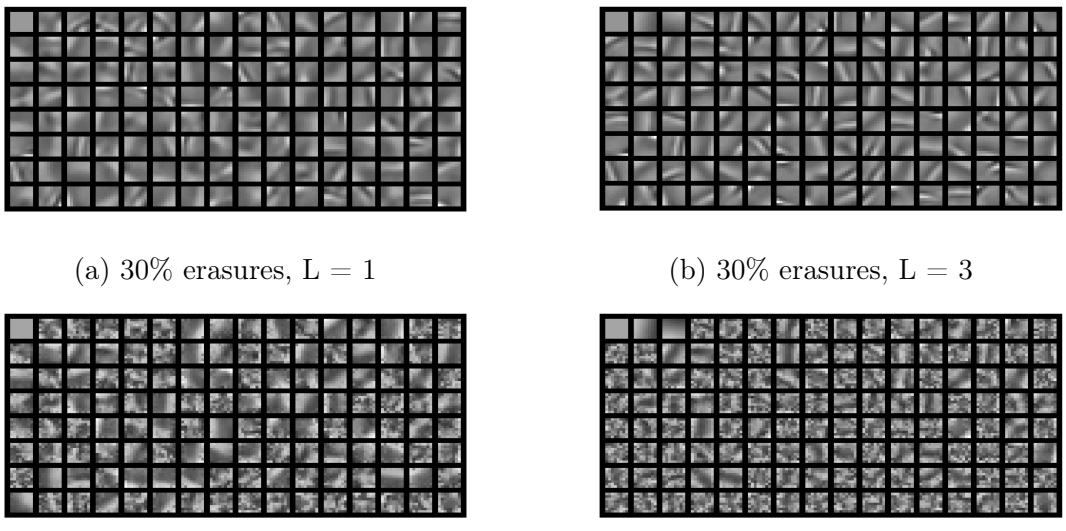
$K = 2d - L$, where $d = p^2$ is the dimension of the patches and L denotes the size of the low-rank component. In our experiment, we consider low-rank components of dimension $1, 3, 8$. We choose the dictionary to be twice the size of the patch dimension, so that the dictionary is over complete. This redundancy allows the dictionary learning algorithm to capture different features, while still being computationally efficient.

Initialisation & sparsity level: For the initial dictionary we create a random $d \times K$ matrix, and ensure its normality and its orthogonality to the low-rank component, that has already been learned. This means that the initial dictionary for ITKrMM and wKSVDL are the same. As sparsity level for wKSVDL we use $S = p$ if $L < p$ holds and in the case $L = 8$ we have to increase the sparsity level, because for $S = L$ the sparse approximation would just select those L low-rank component atoms, which will not be updated. Thus we set $S = 12$.

For ITKrMM we use the sparsity level $S = p - L$ if $L < p$. The difference is, that OMP still has the theoretical choice whether it selects the whole low-rank component or just parts of it in the sparse approximation, whereas in ITKrMM we force the algorithm to use the whole component in the update stage. In case of $L = 8$ we set $S = 4$. This should keep the comparison between the two algorithms as well as between different low-rank components fair.

To learn the low-rank component as well as the dictionary, we act in accordance with Naumova and Schnass [2] and use 10 iterations per low-rank atom and all available patch/mask pairs and 40 iteration on all available patch/mask pairs for both dictionary learning algorithms.

We further test how the sparsity level affects the inpainting results thus doing inpainting with the sparsity levels $(4, 8, 12, 16, 20, 24)$ for the random erasures and $(12, 18, 24, 30, 36)$ for the structured mask.



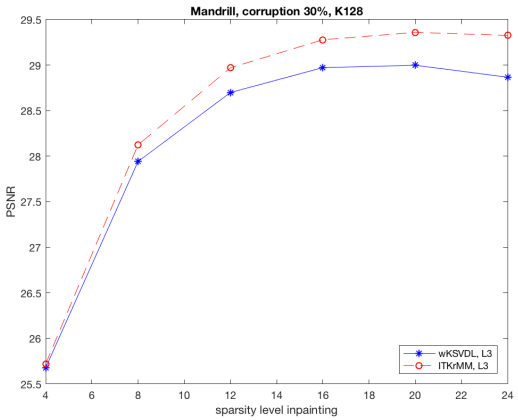
(a) 30% erasures, L = 1

(b) 30% erasures, L = 3

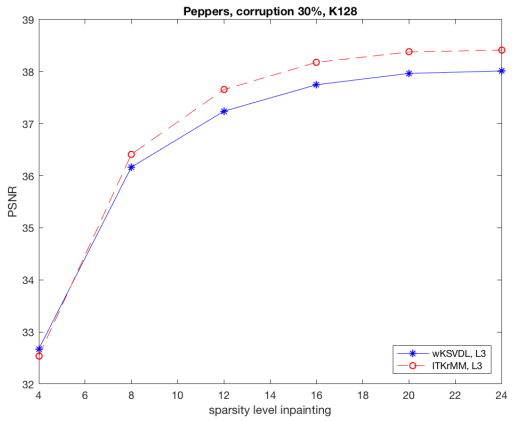
(c) 70% erasures, L = 1

(d) 70% erasures, L = 3

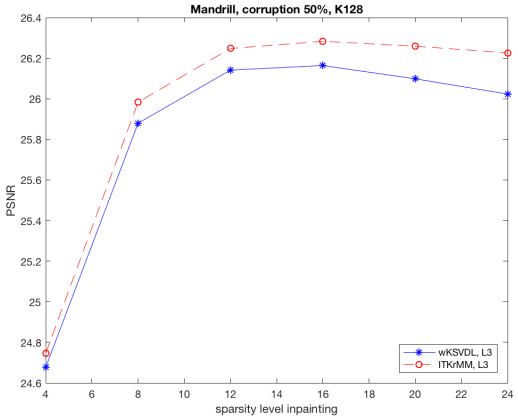
Figure 6: Dictionaries learned from corrupted Peppers image with wKSVDL



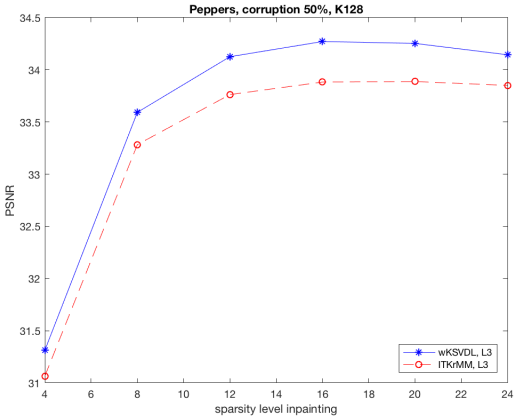
(a) Mandrill, 30% erasures, L =3



(b) Peppers, 30% erasures, L =3



(c) Mandrill, 50% erasures, L =3



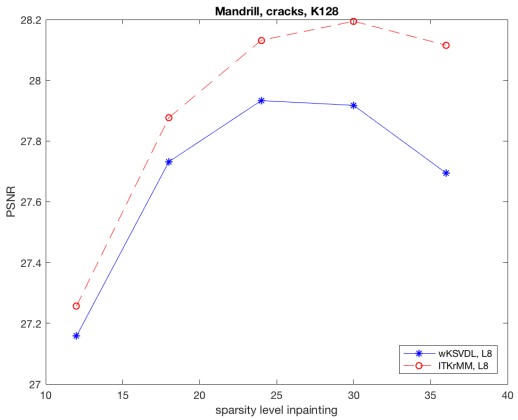
(d) Peppers, 50% erasures, L =3

Figure 7: Comparison of the inpainting success between wKSVD and ITKrMM

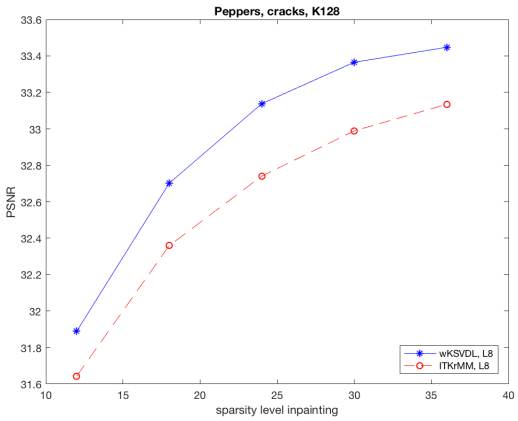
Comparison/Error: After receiving the reconstructed picture $\tilde{Y}$, we want to measure the success of the respective dictionary and low-rank component. This is done by calculating the peak signal-to-noise ratio (PSNR) between the original picture Y and the recovered version $\tilde{Y}$. PSNR is a common value to measure noise in a signal. For images Y and $\tilde{Y}$ of the same size, $d_1 \times d_2$, the PSNR is defined as

$$\text{PSNR in dB} = 10 \cdot \log_{10} \left(\frac{(\max_{i,j} Y(i,j) - \min_{i,j} Y(i,j))^2}{\frac{1}{d_1 d_2} \sum_{i,j} (Y(i,j) - \tilde{Y}(i,j))^2} \right). \tag{5.1}$$

To keep the aspect of randomness in mind we average over 5 runs, each with a different mask and initialisation. This allows to take into account that the PSNR varies in this setup.
In Figure 5 we have a look how different sizes of the low-rank component affect the inpainting results. Thus, we used different masks and corruption levels to show their



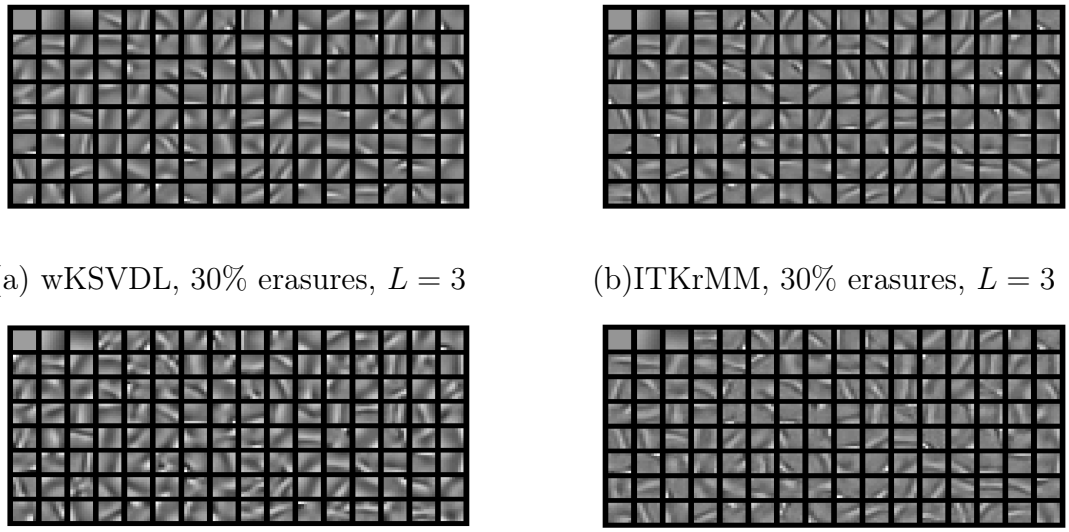
(e) Mandrill, *Cracks*, L = 8



(f) Peppers, *Cracks*, L = 8

Figure 7: Comparison of the inpainting success between wKSVD and ITKrMM

effect in inpainting. Further, we can observe how the inpainting results develops with increasing sparsity level in the inpainting step. We notice that for lower corruption levels, the consideration of the low-rank component improves the inpainting result, whereas for higher corruption levels choosing a low-rank component of high dimension leads to a worse reconstruction. The problem the dictionary learning algorithm has in case of high corruption and a big low-rank component is, that the orthogonality between the dictionary and the low-rank component hinders the algorithm to recover the dictionary atoms correctly, meaning that we end up with noisy atoms. In other words, by considering a high dimensional low-rank component, we extracted too much information from the strongly corrupted picture to learn appropriately from the remaining data.
Considering a larger low-rank component in pictures with low corruption, leads to more textured atoms in the dictionary. This can be noticed when comparing in Figure 6 the dictionaries learned for 30% and 70% erasures. Moreover, for 70% erasures we observe that the dictionary learned with one low-rank atom contains some (noisy) smooth atoms. The remaining atoms are noisy versions of the high frequency atoms. Further, using a three dimensional low-rank component leads to a dictionary which contains hardly any smooth atoms but rather noisy high frequency atoms. The inpainting results then becomes worse because, except for the low-rank atoms, hardly any dictionary atom is of good use in the inpainting step.
Next we compare wKSVDL inpainting results to the results of ITKrMM, and since the low-rank component has a similar effect on both, we always choose one low-rank component size for one corruption level to compare them. For the corruption levels 30% and 50% we use a three dimensional low-rank component, since it performs well in both cases. On the other hand, for the mask pattern *Cracks* we compare them with $L = 8$, since this version outperforms the other two. The pictures used differ in the sense that one (*Peppers*) is smooth, whereas the other (*Mandrill*) is textured. By smooth it is meant that many pixels look similar to those surrounding them, each pepper has one

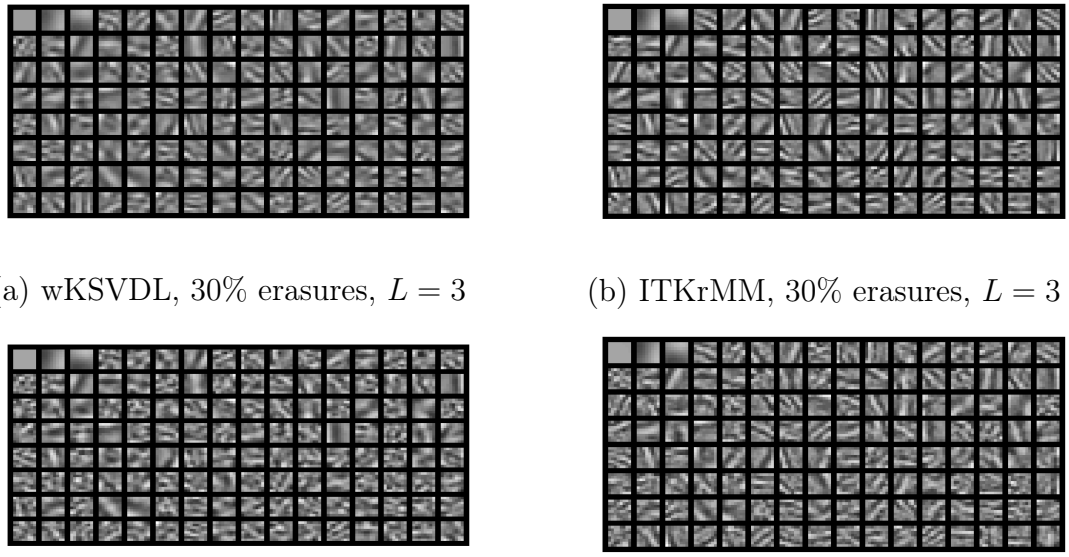


(a) wKSVDL, 30% erasures, $L = 3$ (b)ITKrMM, 30% erasures, $L = 3$

(c)wKSVDL, 50% erasures, $L = 3$ (d) ITKrMM, 50% erasures, $L = 3$

Figure 8: Dictionaries learned from Peppers

colour scheme, whereas in textured images the pixels close to each other differ a lot, as seen in the pelt of the monkey.
The first thing to observe in Figure 7 is that for low corruption levels ITKrMM performs slightly better than wKSVDL. The second thing is that for higher corruption as well as for the *Cracks*, the performance success depends on the corrupted image. For smooth images wKSVDL tends to be the better choice, whereas for textured images it is the other way around.
Further, we can observe that in case of the random mask the growth of PSNR slows down at a sparsity level around 12 and in some cases starts to slowly decrease around $S = 20$. Besides the fact that higher sparsity levels prolong the computational runtime, we see that using to many dictionary atoms can even lead to worse reconstructions. Very much in keeping with the saying: "Too many cooks spoil the broth". The more accurate interpretation is that after a certain sparsity level, we just add more noise to the reconstruction by using more atoms. Further, note that for highly corrupted and more textured images this effect of adding noise is increased. For the structured mask *Cracks* using a sparsity level between 24 and 30 gives the best performance.
To understand why the performance of the algorithms depends on the corrupted image properties, we need to take a look again at the dictionaries recovered by the algorithms, see Figure 8 and 9. In Figure 8 we compare the dictionaries learned on *Peppers* and in Figure 9 those learned on *Mandrill*.
The dictionaries learned on *Peppers* are very similar, and thus to see the difference between the two algorithms we take a look at the dictionaries learned on *Mandrill*. For 30% erasures, we can recognise that wKSVDL produces more smooth (low frequency) atoms, whereas ITKrMM prefers more textured (high frequency) atoms. With more high



(a) wKSVDL, 30% erasures, $L = 3$

(b) ITKrMM, 30% erasures, $L = 3$

(c) wKSVDL, 50% erasures, $L = 3$

(d) ITKrMM, 50% erasures, $L = 3$

Figure 9: Dictionaries learned from Mandrill

frequency atoms, ITKrMM is able to inpaint finer details, leading to better results for more textured images as well as for low corrupted images. On the other side, wKSVDL has an advantage for highly corrupted smooth images since its atoms are noisy versions of the smooth atoms from their 30% counterparts, meaning that it prevents inpainting with noisy details.

The fact that wKSVDL prefers smooth atoms which capture more energy, whereas ITKrMM prefers textured atoms which capture less energy, comes from the different norms considered in the objective function. wKSVDL tries to minimise $\sum_n \|y_n - \phi x_n\|_2^2$, whereas ITKrMM can be interpreted to minimise $\sum_n \|y_n - \phi x_n\|_2$.

After we have seen the numerical results from the inpainting algorithms in Figure 7, we can finally witness its success with our own eyes in Figure 10 and11. In Figure 10, we used wKSVDL to reconstruct *Peppers* with *Cracks* and show the inpainting results for different low-rank component sizes. In Subfigure 10c we used one low-rank atom to learn the dictionary and in Subfigure 10d we used an eight dimensional low-rank component. In Figure 11 we reconstructed *Mandrill* with 50% erasures. We contrast the result of wKSVDL (11c) to the result of ITKrMM (11d) using a three dimensional low-rank component.

In the end, both algorithms operate almost equally good, each with its own advantage. The main difference between wKSVDL and ITKrMM is therefore found in the respective running time. As already addressed above, wKSVDL uses more expensive algorithms in each step to recover the dictionary. Thus, ITKrMM is in the case of the structured mask and patch size 8×8 about 8 times faster, and for the random mask with patch size 12×12 about 12 times faster.

(a) Peppers

(b) Peppers with Cracks

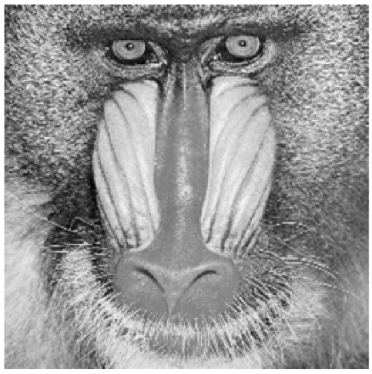
(c) reconstruction with wKSVDL, L=1

(d) reconstruction with wKSVDL, L=8

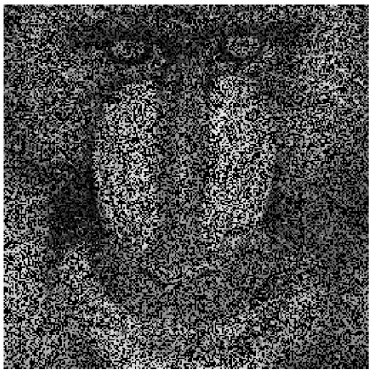
Figure 10

6 CONCLUSION AND FUTURE DIRECTIONS

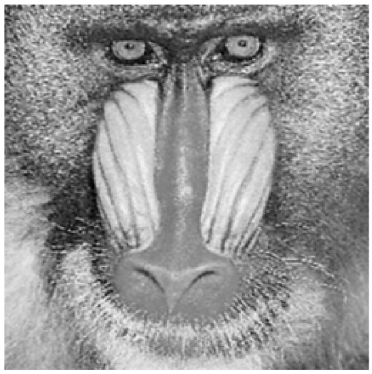
Motivated by the model of natural images, we discussed the presence of low-rank components in images and why we should consider it for inpainting. Thus, we extended the weighted K-SVD (wKSVD) algorithm for dictionary learning to account for the presence of arbitrary sized low-rank components in the data and introduced the weighted K-SVD-low-rank-component (wKSVDL) algorithm. Further, we have shown that in case of low corruption as well as for smooth images the improvement due to higher low-rank components is significant; but also, that the usage of high dimensional low-rank components worsens the inpainting results if the image is textured or strongly corrupted. Therefore, before learning the dictionary, one should carefully consider the dimension of the low-rank component.
Moreover, we compared the inpainting results of wKSVDL to the results of ITKrMM, which is also able to account for arbitrary sized low-rank components. The experiments showed that the algorithms work on par. Each algorithm has its own advantage.



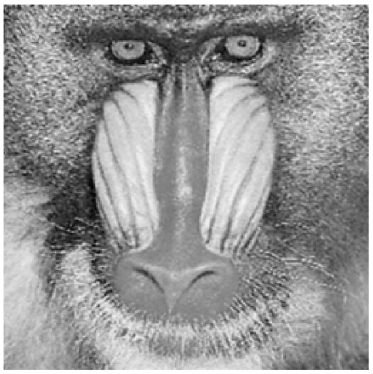
(a) Mandrill



(b) 50% erasures



(c) reconstruction with wKSVDL, L=3



(d) reconstruction with ITKrMM, L=3

Figure 11

The wKSVDL algorithm provides better results for smooth images and ITKrMM outperforms wKSVDL for textured and highly corrupted images. However, ITKrMM is computationally lighter.
Further improvement in inpainting and computing time can be achieved by optimising the dictionary size. For our experiments we chose an overcomplete dictionary with $K = 2d$. Looking back at the learned dictionaries, especially at Subfigure 6c, we see that for high corruption we learn a lot of very noisy atoms. Since, these atoms are not of good use in inpainting we do not necessarily need to learn them. And thus, one suggestion to prevent us form learning many noisy atoms is to simply reduce the dictionary size. Though, testing different dictionary sizes would be interesting to do it is beyond scope of this thesis. Another approach is to adaptively choose the dictionary size. There are works on adaptive choice of the dictionary size, in particular for ITKrM and K-SVD, [5], [6]. These strategies need to be extended to corruption to be applicable for inpainting.
Although, we just considered the application of inpainting, there are many more fields

to use those algorithms. In particular, they can be used to restore any corrupted data given, eg. [2] the ITKrMM algorithm was motivated by a real-life problem with corrupted data in diabetes therapy management.

References

[1] K-SVD:An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation, M. Aharon, M. Elad and A. Bruckstein

[2] Dictionary Learning from Incomplete Data, V. Naumova and K. Schnass

[3] Sparse Representation for Color Image Restoration, J. Mairal, M. Elad and G. Sapiro

[4] Weighted low-rank approximations, N. Srebro, T. Jaakkola

[5] Dictionary learning - from local towards global and adaptive, K. Schnass

[6] Learning an Adaptive Dictionary Structure for Effcient Image Sparse Coding, J. Mazaheri, C. Guillemot, C. Labit

[7] Dictionaries for Sparse Representation Modeling, R. Rubinstein, A. Bruckstein, M. Elad

[8] Orthogonal matching pursuit, Y. C. Pati, R. Rezaiifar, P. S. Krishnaprasad

[9] Greed is Good: Algorithmic Results for Sparse Approximation, J. Tropp