Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Colorization by Patch-Based Local Low-Rank Matrix Completion

Quanming Yao James T. Kwok
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong
{qyaoaa, jamesk}@cse.ust.hk

Abstract

Colorization aims at recovering the original color of a monochrome image from only a few color pixels. A state-of-the-art approach is based on matrix completion, which assumes that the target color image is low-rank. However, this low-rank assumption is often invalid on natural images. In this paper, we propose a patch-based approach that divides the image into patches and then imposes a low-rank structure only on groups of similar patches. Each local matrix completion problem is solved by an accelerated version of alternating direction method of multipliers (ADMM), and each ADMM subproblem is solved efficiently by divide-and-conquer. Experiments on a number of benchmark images demonstrate that the proposed method outperforms existing approaches.

Introduction

Because of technology limitations, most movies and pictures produced in the last century are monochrome. Colorization is a computer-assisted process that attempts to recover their original colors with some user-provided color pixels (Levin, Lischinski, and Weiss 2004; Luan et al. 2007). However, traditional colorization algorithms are time-consuming, and involve expensive segmentation and tracking of image regions (Markle and Hunt 1987).

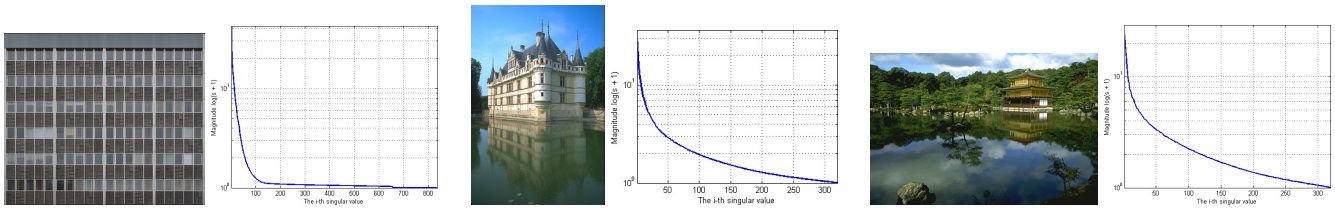
A seminal work that drastically reduces the amount of manual input is proposed in (Levin, Lischinski, and Weiss 2004). It assumes that neighboring pixels with similar intensities should have similar colors. This is formulated as an optimization problem which minimizes the difference between the color at each pixel and the weighted average of colors from its neighboring pixels. Computationally, it leads to a sparse linear system which can be efficiently solved. However, on complex textures, the underlying local color consistency assumption may fail.

In recent years, it is shown that many image objects can be modeled as low-rank matrices. Examples include the dynamic textures (Doretto et al. 2003), and an image set obtained on a convex Lambertian object under different lighting conditions (Basri and Jacobs 2003). Consequently, low-rank modeling has been popularly used in various image analysis tasks, such as face recognition (Candès and Plan 2010; Chen, Wei, and Wang 2012), background removal (Candès et al. 2011), video denoising (Ji et al. 2010), and image restoration (Nguyen et al. 2013).

Recently, low-rank modeling has also been applied to colorization. Wang and Zhang (2012) used the robust principal component analysis (RPCA) model (Candès et al. 2011), which assumed that the target color image can be decomposed as the sum of a low-rank matrix and a sparse error component. Computationally, this is formulated as a convex optimization problem which can be solved with the alternating direction method of multipliers (ADMM) (Boyd et al. 2011). By combining with the local color consistency approach in (Levin, Lischinski, and Weiss 2004), state-of-the-art colorization results are obtained.

However, an image matrix is low-rank only if its columns (or rows) are linear combinations of a small subset of columns (resp. rows). This is the case when the image contains many regular patterns, as is commonly found in man-made objects (such as the *building* in Figure 1(a)). However, it can be a crude approximation on natural images (Figures 1(c) and 1(e)). Often, a significant number of singular values have to be removed before the image is close to low-rank (Liu et al. 2013).

On the other hand, a set of similar images are often low-rank (Cai, Candès, and Shen 2010; Candès et al. 2011; Peng et al. 2012). This motivates the usage of a patch-based approach, which has achieved impressive results on various image processing tasks including denoising (Elad and Aharon 2006; Dabov et al. 2007; Mairal et al. 2009a), super-resolution (Yang et al. 2010), and inpainting (Mairal et al. 2009b). Specifically, the proposed colorization scheme (i) divides the image into patches; (ii) groups similar patches together; and then (iii) performs colorization on each low-rank patch group. By allowing patches in the same group to borrow strength from each other, better performance can be attained. Besides, even when the individual patches are not low-rank, the whole group is likely to contain shared patterns among patches and thus has a low-rank structure. Computationally, this model leads to a convex optimization problem which can be solved by ADMM. Moreover, by using the ℓ_2 loss (instead of the ℓ_1 in (Wang and Zhang 2012)) on the reconstruction error, it will be shown that an accelerated version of ADMM (Goldstein, ODonoghue, and Setzer 2014) can be used. We further develop a novel divide-and-conquer algorithm so that each ADMM subproblem can be



(a) building. (b) spectrum. (c) castle. (d) spectrum. (e) lake. (f) spectrum.

Figure 1: Singular value spectra of the *building*, *castle* and *lake* images.

solved efficiently.

The rest of this paper is organized as follows. We first provide short reviews on ADMM and RPCA. Next, we present the proposed formulation (and the associated optimization solver) based on patch-based low-rank matrix completion. The last section provides experimental results on a number of benchmark images and some concluding remarks.

Notations: In the sequel, the transpose of a vector / matrix is denoted by the superscript $(\cdot)^\top$, and the identity matrix by I. Moreover, for a matrix $A = [a_{ij}] \in \mathbb{R}^{m\times n}$, $\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2}$ is the Frobenius norm, $\|A\|_* = \sum_i \sigma_i$ (where σ_i's are the singular values of A) is the nuclear norm, and $\mathrm{vec}(A) \in \mathbb{R}^{mn}$ stacks the columns of A to a vector. Besides, for two matrices $A = [a_{ij}], B = [b_{ij}] \in \mathbb{R}^{m\times n}$, $A \odot B = [a_{ij}b_{ij}]$ is the Hadamard product. For $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$, $A \otimes B = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}$ is the Kronecker product. Finally, for $a = [a_i] \in \mathbb{R}^n$, $\mathrm{Diag}(a)$ reshapes it to a diagonal matrix with elements a_i's.

Related Works

Alternating Direction Method of Multipliers

In recent years, the alternating direction method of multipliers (ADMM) has been popularly used in diverse fields such as machine learning, data mining and image processing (Boyd et al. 2011). Consider optimization problems of the form

$$\min_{x,y} \ \phi(x) + \psi(y) \ : \ Ax + By = c, \tag{1}$$

where ϕ, ψ are convex functions, and A, B (resp. c) are constant matrices (resp. vector). ADMM considers the augmented Lagrangian $\mathcal{L}(x, y, \nu) = \phi(x) + \psi(y) + \nu^T(Ax + By - c) + \frac{\rho}{2}\|Ax + By - c\|^2$, where ν is the dual variable, and $\rho > 0$ is a penalty parameter. At the tth iteration, the values of x and y (denoted x_t and y_t) are updated by minimizing $\mathcal{L}(x, y, \nu_{t-1})$ w.r.t. x and y in an alternating manner:

$$\begin{aligned} x_t &= \arg\min_x \mathcal{L}(x, y_{t-1}, \nu_{t-1}), \\ y_t &= \arg\min_y \mathcal{L}(x_t, y, \nu_{t-1}), \end{aligned}$$

and then ν is updated as $\nu_t = \nu_{t-1} + \rho(Ax_t + By_t - c)$. While (1) has only two blocks of variables (x and y), the ADMM has been recently extended to problems with three or more separable blocks (Hong and Luo 2012)

$$\min_{x_1,\ldots,x_K} \sum_{i=1}^K \phi_i(x_i) \text{ s.t. } \sum_{i=1}^K A_i x_i = c_i.$$

A number of interesting ADMM applications with $K \geq 3$ blocks can be found in (Ma 2012).

The standard ADMM algorithm above has a convergence rate of $O(1/T)$, where T is the number of iterations (He and Yuan 2012). Recently, inspired by the acceleration techniques for gradient descent methods (Beck and Teboulle 2009; Nesterov 2004), Goldstein, ODonoghue, and Setzer (2014) proposed an accelerated version of ADMM (Algorithm 1), which has a much faster $O(1/T^2)$ convergence rate. However, this acceleration only works for the two-block ADMM. Extension to multiple blocks is still open.

Algorithm 1 Accelerated ADMM (Goldstein, ODonoghue, and Setzer 2014).

Initialize: $y_0 = \hat{y}_1 = 0$, $\nu_0 = \hat{\nu}_1 = 0$, $\rho > 0$, $\alpha_1 = 1$, and $\eta \in (0, 1)$.
for $t = 1, 2, \ldots$ **do**
  $x_t = \arg\min_x \mathcal{L}(x, \hat{y}_t, \hat{\nu}_t)$;
  $y_t = \arg\min_y \mathcal{L}(x_t, y, \hat{\nu}_t)$;
  $\nu_t = \hat{\nu}_t + \rho\,(Ax_t + By_t - c)$;
  $c_t = \rho^{-1}\|\nu_t - \hat{\nu}_t\|^2 + \rho\|B(y_t - \hat{y}_t)\|^2$;
  if $c_t < \eta c_{t-1}$ **then**
    $\alpha_{t+1} = \frac{1}{2}(1 + \sqrt{1 + 4\alpha_t^2})$;
    $\hat{y}_{t+1} = y_t + \frac{\alpha_t - 1}{\alpha_{t+1}}(y_t - y_{t-1})$;
    $\hat{\nu}_{t+1} = \nu_t + \frac{\alpha_t - 1}{\alpha_{t+1}}(\nu_t - \nu_{t-1})$;
  else
    $\alpha_{t+1} = 1$, $\hat{y}_{t+1} = y_t$, $\hat{\nu}_{t+1} = \nu_t$;
    $c_t = \eta^{-1} c_{t-1}$;
  end if
end for

Robust Principal Component Analysis

Recently, Wang and Zhang (2012) proposed a state-of-the-art colorization algorithm based on RPCA. Here, we consider RGB color images. Each $m \times n$ color image has three color components (red, green and blue), and these together can be represented by a matrix of size $m \times 3n$.

Given a monochrome image $G \in \mathbb{R}^{m\times n}$, and a small subset of user-provided color pixels (with values stored in $O \in \mathbb{R}^{m\times 3n}$, and their positions indicated by the binary matrix $\Omega \in \{0, 1\}^{m\times 3n}$). The target color image $L \in \mathbb{R}^{m\times 3n}$

is obtained by the following optimization problem:

$$\min_{L} \frac{1}{2}\|LT - G\|_F^2 + \lambda\|\Omega \odot (L - O)\|_1 + \mu\|L\|_*, \quad (2)$$

where $T = [I, I, I]^\top$ is a linear transform that averages the three color components to form a monochrome image. The first term in (2) measures the discrepancy between the transformed and observed monochrome images; the second term measures the difference with the observed color pixels; while the last term encourages L to be low-rank. For efficient optimization, (2) is rewritten as

$$\begin{aligned} \min_{L,X,E} \quad & \frac{1}{2}\|LT - G\|_F^2 + \lambda\|\Omega \odot E\|_1 + \mu\|X\|_* \\ \text{s.t.} \quad & O = L + E,\ L = X, \end{aligned} \quad (3)$$

which is then solved by ADMM. However, as there are three variable blocks, the accelerated ADMM cannot be used.

Proposed Algorithm

While a set of similar natural images can often be effectively approximated by a low-rank matrix (Cai, Candès, and Shen 2010; Candès et al. 2011; Peng et al. 2012), Wang and Zhang (2012) assumed that a single image (i.e., the target color image L) is low-rank. As discussed in the introduction, this requires L to have regular patterns, which may not be valid on natural images.

In this paper, instead of assuming that the whole L is low-rank, we first extract groups of similar image patches, and then only assume that each group has a low-rank structure. As will be seen, this group-based local low-rank assumption is more appropriate and leads to better colorization performance. The proposed procedure, which will be called **Pa**tch-based **L**ocal **L**ow-**R**ank colorization (PaLLR) in the sequel, is shown in Algorithm 2. Note that in step 2, the uncolored patches are selected as in BM3D (Dabov et al. 2007) (i.e., from top-left to bottom-right of the image). However, the exact order is not important and preliminary results show that random selection yields similar performance.

Algorithm 2 PaLLR procedure.

1: input: monochrome image; a small set of color pixels.
2: **while** there exists a patch P not yet colored **do**
3: find $k-1$ patches that are most similar to P;
4: obtain colorization for the group of k patches (by solving (4) with accelerated ADMM in Algorithm 1);
5: **end while**
6: **for** each patch P **do**
7: perform (weighted) average on the colorization results from all groups containing P;
8: **end for**
9: **for** each pixel in the image **do**
10: average the values from overlapping patches.
11: **end for**

Grouping of Similar Image Patches

Given a monochrome image (of size $m \times n$), we first extract overlapping patches each of size $r \times r$. Let the patch at position (i, j) be $P_{i,j} \in \mathbb{R}^{r\times r}$. The similarity between patches $P_{i,j}$ and $P_{i',j'}$ is defined as $\|P_{i,j} - P_{i',j'}\|_F^2 + \beta\left(\frac{1}{m^2}(i-i')^2 + \frac{1}{n^2}(j-j')^2\right)$. The first term measures similarity in terms of pixel intensity, while the last two measure the physical distance. β is a trade-off parameter, and is set to 1 in the experiments.

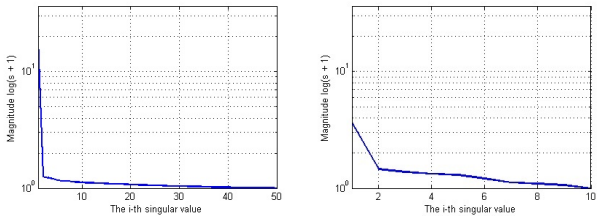
For a given image patch, the standard block matching operation (Dabov et al. 2007) can be used to find its k most similar patches in step 3. However, this takes $O(kr^2D^2)$ time (where D is the maximum search distance), and can be expensive when D is large. In this paper, we perform the search more efficiently using approximate nearest neighbors (Muja and Lowe 2014). Its time complexity is $O(kr^2 \log N)$, where N is the total number of patches. Empirically, this is much faster than block matching.

Local Low-Rank Factorization

For each image patch, let the k most similar patches (including itself) be contained in the matrix $\tilde{G} \in \mathbb{R}^{r^2\times k}$. Similar to (2), these patches can be colorized by solving the following optimization problem

$$\min_{\tilde{L}} \frac{1}{2}\|\tilde{L}\tilde{T} - \tilde{G}\|_F^2 + \frac{\lambda}{2}\|\tilde{\Omega} \odot (\tilde{L} - \tilde{O})\|_F^2 + \mu\|\tilde{L}\|_*, \quad (4)$$

where $\tilde{O}$ and $\tilde{\Omega}$ indicate the values and positions of the color pixels in these k patches, $\tilde{T} \in \mathbb{R}^{3k\times k}$ is the color-to-monochrome linear transform, and $\tilde{L} \in \mathbb{R}^{r^2\times 3k}$ is the target colorization of $\tilde{G}$. While (2) assumes that the whole image is low-rank, here this is assumed only on each group of similar image patches. Figure 2(a) shows the singular value spectrum of a typical patch group. Compared to Figure 1(d), the patch group has a much lower rank.



(a) group of patches. (b) one patch.

Figure 2: Singular value spectra on a patch group and single patch (*castle* image).

Another difference with (2) is that the ℓ_2 loss, instead of the ℓ_1, is used on the low-rank reconstruction error. As will be seen in the next section, this leads to a simpler optimization problem, with fewer optimization variables and allows the use of accelerated ADMM.

Combining the Colorization Results

Since a patch may belong to multiple groups, its colorizations from different groups can be different. These can be combined by weighted averaging, with smaller contributions from the more distant groups. Finally, one has to assemble the whole image from the overlapping colorized patches. As is common in patch-based image processing (Dabov et al.

2007), the color of a pixel is obtained by averaging the color values in patches containing it.

Solving (4) Using Accelerated ADMM

In this section, we discuss how to efficiently solve the local low-rank factorization problem in (4). First, we rewrite it as

$$\min_{\tilde{L},X} \frac{1}{2}\|\tilde{L}\tilde{T}-\tilde{G}\|_F^2+\frac{\lambda}{2}\|\tilde{\Omega}\odot(\tilde{L}-\tilde{O})\|_F^2+\mu\|X\|_* \quad \text{s.t.} \quad X=\tilde{L}. \tag{5}$$

Since we only have two variable blocks, it can be readily solved using the accelerated ADMM algorithm. The first two terms in the objective of (5) together play the role of ϕ, and the last term is ψ. The augmented Lagrangian is $\mathcal{L}(\tilde{L},X,Q)=\frac{1}{2}\|\tilde{L}\tilde{T}-\tilde{G}\|_F^2+\frac{\lambda}{2}\|\tilde{\Omega}\odot(\tilde{L}-\tilde{O})\|_F^2+\mu\|X\|_*+\text{tr}(Q^\top(\tilde{L}-X))+\frac{\rho}{2}\|\tilde{L}-X\|_F^2$, where Q is the dual variable. Thus, in Algorithm 1, variable x becomes $\tilde{L}$, y becomes X, and ν becomes Q. The two core steps are the updates of $\tilde{L}$ and X, which will be discussed in the following.

ADMM Update on $\tilde{L}$

Step 2 in Algorithm 1 becomes $\min_{\tilde{L}} \frac{1}{2}\|\tilde{L}\tilde{T}-\tilde{G}\|_F^2+\frac{\lambda}{2}\left\|\tilde{\Omega}\odot(\tilde{L}-\tilde{O})\right\|_F^2+\text{tr}(\hat{Q}_t^\top(\tilde{L}-\hat{X}_t))+\frac{\rho}{2}\|\tilde{L}-\hat{X}_t\|_F^2$. Setting its derivative w.r.t. $\tilde{L}$ to zero, and using properties of the Hadamard and Kronecker products, it is easy to see that $\tilde{L}_t$ can be obtained as

$$R\ \text{vec}\left(\tilde{L}_t\right)=\text{vec}(C), \tag{6}$$

where $R=\left(\tilde{T}\tilde{T}^\top\right)\otimes I+\lambda\text{Diag}\left(\text{vec}\left(\tilde{\Omega}\right)\right)+\rho I$, and $C=\tilde{G}\tilde{T}^\top+\lambda\left(\tilde{\Omega}\odot\tilde{O}\right)+\rho\hat{X}_t-\hat{Q}_t$. This is a simple linear system. However, as $R\in\mathbb{R}^{\tilde{m}\tilde{n}\times\tilde{m}\tilde{n}}$ (where, for simplicity of notations, we denote $\tilde{m}=r^2$ and $\tilde{n}=3k$), a naive matrix inversion takes $O(\tilde{m}^3\tilde{n}^3)$ time. Alternatively, iterative approaches such as conjugate gradient (CG) can be used. Empirically, this is faster, though each CG iteration takes $O(\tilde{m}\tilde{n})$ time and $O(\min(\tilde{m},\tilde{n}))$ iterations are required for convergence (Nocedal and Wright 2006).

Much faster matrix inversion is possible by exploiting the structure of R. Observe that R can be partitioned into $(3r^2)^2$ blocks (each of size $k\times k$), with the (i,j)th block being

$$B^{ij}=\begin{cases} I & i\neq j \\ \lambda\text{Diag}(\omega_i)+(1+\rho)I & i=j \end{cases},$$

and ω_i is the sub-vector in $\text{vec}(\tilde{\Omega})$ containing its (ik^2+1)th to $(i+1)k^2$th elements. As B^{ij} is diagonal, $\left(B^{ij}\right)^{-1}$ can be easily obtained in $O(k)$ time as

$$\left(B^{ij}\right)^{-1}=\begin{cases} I & i\neq j \\ \text{Diag}\left(1./\left(\lambda\omega_i+(1+\rho)1\right)\right) & i=j \end{cases}. \tag{7}$$

Here, we use the MATLAB notation and the ./ operator performs division on each element of the vector separately. Using divide-and-conquer and the equation for partitioned matrix inverse, R^{-1} can be efficiently obtained as FastInv(R) using Algorithm 3 in $O(\tilde{m}\tilde{n}\log\tilde{m})$ time.

Algorithm 3 FastInv(M): Fast inverse on a sub-matrix M of R using divide-and-conquer.

if M is of size $k\times k$ **then**
  invert M using (7);
else
  partition M to 4 equal-sized sub-matrices $\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$;
  $M_{22}^{-1}=\text{FastInv}(M_{22})$;
  $U^{-1}=\text{FastInv}(M_{11}-M_{12}M_{22}^{-1}M_{21})$;
  $M^{-1}=\begin{bmatrix} U^{-1} & -U^{-1}M_{12}M_{22}^{-1} \\ -M_{22}^{-1}M_{21}U^{-1} & M_{22}^{-1}+M_{22}^{-1}M_{21}U^{-1}M_{12}M_{22}^{-1} \end{bmatrix}$.
end if

It can be easily seen that R^{-1}, like R, is also sparse and has only $O(\tilde{m}\tilde{n})$ nonzero elements. Thus, on recovering $\tilde{L}_t$ from (6) (as $\text{vec}(\tilde{L}_t)=R^{-1}\text{vec}(C)$), this matrix-vector multiplication takes $O(\tilde{m}\tilde{n})$ time. In total, the $\tilde{L}$ update takes $O(\tilde{m}\tilde{n}\log\tilde{m})$ time, which is much faster than both direct matrix inversion and conjugate gradient.

ADMM Update on X

Step 3 in Algorithm 1 can be rewritten as:

$$X_t=\arg\min_X \frac{1}{2}\left\|X-\left(\tilde{L}_t+\frac{1}{\rho}\hat{Q}_t\right)\right\|_F^2+\frac{\mu}{\rho}\|X\|_*.$$

It is well-known that this reduces to the singular value thresholding (SVT) operator $\text{SVT}_{\frac{\mu}{\rho}}\left(\tilde{L}_t+\frac{1}{\rho}\hat{Q}_t\right)$ (Cai, Candès, and Shen 2010). In general, let $U\Sigma V^\top$ be the SVD of a matrix Z. Then, $\text{SVT}_\tau(Z)=U\left(\Sigma-\tau I\right)_+V^\top$. The SVT operation here takes $O\left(\tilde{m}\tilde{n}\min(\tilde{m},\tilde{n})\right)$ time.

Discussion

Recall that if $\|\tilde{\Omega}\odot(\tilde{L}-\tilde{O})\|_F^2$ in (4) is replaced by $\|\tilde{\Omega}\odot(\tilde{L}-\tilde{O})\|_1$, the optimization problem will be of the same form as (2) and the solver in (Wang and Zhang 2012) can be used. While the per-iteration complexity in both solvers are dominated by the SVT step, (3) involves three primal and two dual variables, but (5) only has two primal variables and one dual variable. Hence, ours is faster by a constant factor. More importantly, since (5) has only two variable blocks, the accelerated ADMM can be used, while (3) can only use the standard ADMM. As shown in (Goldstein, ODonoghue, and Setzer 2014), the difference in convergence rates (namely, $O(1/T^2)$ vs $O(1/T)$) can lead to a significant difference in the number of iterations required.

The proposed PaLLR algorithm is related to the recent local low-rank matrix approximation (LLORMA) algorithm (Lee et al. 2013; 2014), which has demonstrated encouraging results on recommendation systems. However, LLORMA may not be appropriate for colorization. When used in this context, it assumes that all image patches are low-rank (Table 1). For each patch, a low-rank matrix completion problem (similar to (2) and (4)) is solved. However,

method	entity of low-rank structure
(Wang and Zhang 2012)	whole image
(Lee et al. 2013)	each single patch
proposed method	group of similar patches

Table 1: Low-rank assumptions used by various methods.

an image patch is small, and thus unlikely to contain repetitive patterns or regular structures at such a scale. Consequently, its rank may not be low (Figure 2(b)). As will be seen from the experimental results, this low-rank assumption leads to inferior colorization performance.

One potential problem with local algorithms (such as the proposed PaLLR and LLORMA) is that a large number of local models may have to be learned. To alleviate this problem, instead of computing a local model at each possible location, LLORMA only randomly chooses a subset as anchor points for the local models. For PaLLR, we only learn colorizations for patches that have not yet been colorized. As each group has k patches, this reduces the number of local models by about k times. Moreover, the optimization variables in (4) are much smaller than those in (2) ($\tilde{m}\tilde{n}$ vs mn). Besides, as in LLORMA, local low-rank factorizations for multiple image patches can be easily parallelized.

Experiments

Experiments are performed on eight color images from the Berkeley segmentation data set (Figure 3). These have also been used in (Wang and Zhang 2012), and we follow the same setup. For each color image, the monochrome version is obtained by averaging the R, G and B components. Varying numbers of pixels ($1\% - 10\%$) are randomly sampled from the color image as observed labels input to the colorization algorithm. The following methods will be compared:[1]

1. Local color consistency (LCC) (Levin, Lischinski, and Weiss 2004);
2. Global low-rank matrix completion (GLR) (Wang and Zhang 2012): The values of λ and μ are set as suggested in (Wang and Zhang 2012);
3. Local low-rank matrix approximation (LLORMA) (Lee et al. 2013): To be consistent with PaLLR, each local matrix factorization is based on (4), and the resultant local colorization results are combined as in PaLLR.
4. The proposed PaLLR: We experiment with both ℓ_1 and ℓ_2 losses for the reconstruction error. For ℓ_1, we follow the parameter setting in GLR. For ℓ_2, note that the first term in (4) sums over r^2k pixels, while the second term sums over $n_{\tilde{\Omega}}$ pixels (where $n_{\tilde{\Omega}}$ is the number of 1's in $\tilde{\Omega}$). Thus, we set $\lambda \propto r^2k/n_{\tilde{\Omega}}$. In the experiments, we fix the proportionality constant to 5, and $\mu = 0.16$. As will be seen, the performance is not sensitive to these settings.

As suggested by (Wang and Zhang 2012), local color consistency is used as preprocessing for GLR, LLORMA, and PaLLR. This expands the initial set of color pixels, and makes colorization easier. Besides, both LLORMA and PaLLR operate on image patches. We fix the patch size r to 16, and use $k = 50$ patches in each PaLLR group. As will be seen, the performance is again not sensitive to these settings. In (Lee et al. 2013; 2014), the local models of LLORMA are positioned at random locations. To avoid the possible statistical variation, here we construct its local models at all patches. Hence, LLORMA has an unfair advantage over PaLLR as it uses more local models.

[1]Codes for LCC and GLR are obtained from their authors.

(a) castle. (b) koala. (c) mushroom. (d) woman.

(e) couple. (f) lake.

(g) landscape. (h) street.

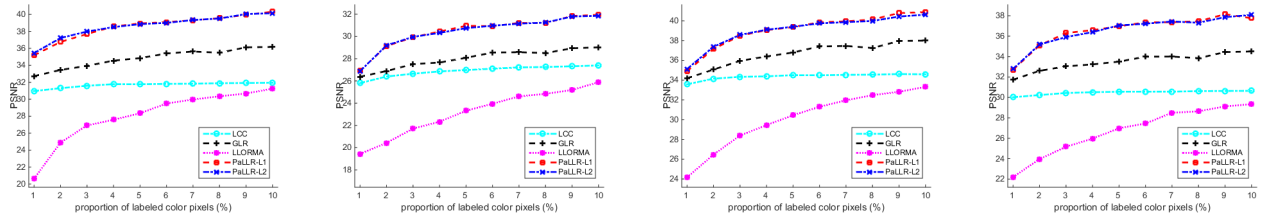
Figure 3: Images used in the experiments. (a)-(d) are of size 481×321, (e)-(h) are of size 321×481.

For performance evaluation, we use the peak signal-to-noise ratio (PSNR) which has been commonly used in image processing (Elad and Aharon 2006). Let the original color image be $\tilde{I} \in \mathbb{R}^{m\times 3n}$ and the colorization result be $\hat{I} \in \mathbb{R}^{m\times 3n}$, PSNR is defined as $10\log_{10}\left(\frac{255}{\Delta}\right)$, where $\Delta = \frac{1}{3mn}\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{3n}(\tilde{I}_{ij} - \hat{I}_{ij})^2}$.

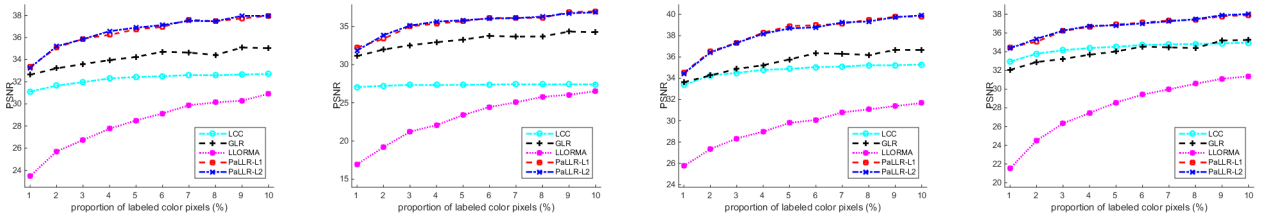
Results

Results on PSNR are shown in Figure 4. As can be seen, PaLLR significantly outperforms the other methods, and the improvement increases with the number of labeled color pixels. Using ℓ_1 or ℓ_2 for the reconstruction error in PaLLR lead to very similar PSNR results, but using ℓ_2 is about 4 times faster. Overall, GLR outperforms LCC (except on the images *street* and *woman*) and is in the second place in terms of PSNR. LLORMA is the worst, even though it uses more patches and thus more local models.

Colorization results and difference images are shown in Figures 5 and 6, respectively. The LLORMA results are in-

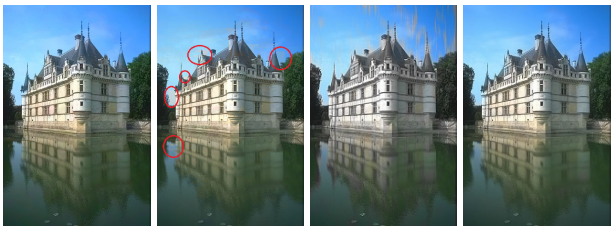


(a) castle. (b) couple. (c) koala. (d) lake.



(e) landscape. (f) mushroom. (g) street. (h) woman.

Figure 4: PSNR results on the various images.

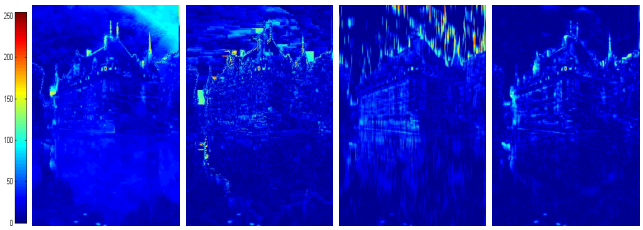


(a) LCC. (b) GLR. (c) LLORMA. (d) PaLLR.

(e) LCC. (f) GLR. (g) LLORMA. (h) PaLLR.

Figure 5: Colorization results on *castle* (1% labels) and *woman* (10% labels). Artifacts for GLR are circled in red.

ferior, which agree with the previous observations on PSNR.

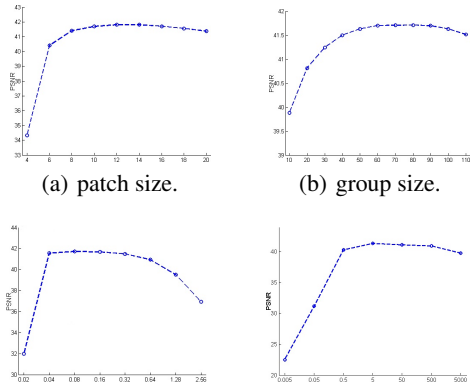


(a) LCC. (b) GLR. (c) LLORMA. (d) PaLLR.

Figure 6: Differences between the original *castle* image and various colorization results in Figure 5.

For GLR, artifacts can be seen. Moreover, while the errors produced by GLR and PaLLR are localized, those by LCC are more diffused.

Finally, we study the effects on varying the patch size r, group size r and regularization parameters (μ and λ) in (4). As can be seen from Figure 7, PaLLR is stable over a wide range of parameter settings.



(a) patch size. (b) group size.

(c) μ. (d) λ.

Figure 7: Variation in PSNR with different parameter settings on the *castle* image.

Conclusion

In this paper, we extended the matrix completion approach for colorization. Instead of using a low-rank structure on the whole image, we adopted a patch-based scheme and imposed this assumption only on each group of similar patches. The resultant optimization problem can be solved by accelerated ADMM, with the ADMM subproblem efficiently solved with a novel divide-and-conquer algorithm. Experimental results demonstrate that the proposed method outperforms existing approaches.

Acknowledgments

This research was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region (Grant 614513).

References

Basri, R., and Jacobs, D. W. 2003. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25(2):218–233.

Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences* 2(1):183–202.

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends of Machine Learning* 3(1):1–122.

Cai, J.; Candès, E. J.; and Shen, Z. 2010. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization* 20:1956–1982.

Candès, E. J., and Plan, Y. 2010. Matrix completion with noise. *Proceedings of the IEEE* 98(6):925–936.

Candès, E. J.; Li, X.; Ma, Y.; and Wright, J. 2011. Robust principal component analysis. *Journal of the ACM* 58(3):1–37.

Chen, C.-F.; Wei, C.-P.; and Wang, Y.-C. F. 2012. Low-rank matrix recovery with structural incoherence for robust face recognition. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2618–2625.

Dabov, K.; Foi, A.; Katkovnik, V.; and Egiazarian, K. 2007. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing* 16(8):2080–2095.

Doretto, G.; Chiuso, A.; Wu, Y.; and Soatto, S. 2003. Dynamic textures. *International Journal of Computer Vision* 51(2):91–109.

Elad, M., and Aharon, M. 2006. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing* 15(12):3736–3745.

Goldstein, T.; ODonoghue, B.; and Setzer, S. 2014. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences* 7(3).

He, B., and Yuan, X. 2012. On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method. *SIAM Journal on Numerical Analysis* 50(2):700–709.

Hong, M., and Luo, Z.-Q. 2012. On the linear convergence of the alternating direction method of multipliers. Technical Report arXiv:1208.3922.

Ji, H.; Liu, C.; Shen, Z.; and Xu, Y. 2010. Robust video denoising using low rank matrix completion. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 1791–1798.

Lee, J.; Kim, S.; Lebanon, G.; and Singer, Y. 2013. Local low-rank matrix approximation. In *Proceedings of the 30th International Conference on Machine Learning*, 82–90.

Lee, J.; Bengio, S.; Kim, S.; Lebanon, G.; and Singer, Y. 2014. Local collaborative ranking. In *Proceedings of the 23rd International Conference on World Wide Web*, 85–96.

Levin, A.; Lischinski, D.; and Weiss, Y. 2004. Colorization using optimization. *ACM Transactions on Graphics* 23(3):689–694.

Liu, J.; Musialski, P.; Wonka, P.; and Ye, J. 2013. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(1):208–220.

Luan, Q.; Wen, F.; Cohen-Or, D.; Liang, L.; Xu, Y.-Q.; and Shum, H.-Y. 2007. Natural image colorization. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, 309–320.

Ma, S. 2012. Alternating proximal gradient method for convex minimization. Technical report.

Mairal, J.; Bach, F.; Ponce, J.; Sapiro, G.; and Zisserman, A. 2009a. Non-local sparse models for image restoration. In *Proceedings of the 12th International Conference on Computer Vision*, 2272–2279.

Mairal, J.; Bach, F.; Ponce, J.; and Sapiro, G. 2009b. Online dictionary learning for sparse coding. In *Proceedings of the 26th International Conference on Machine Learning*, 689–696.

Markle, W., and Hunt, B. 1987. Coloring a black and white signal using motion detection. Patent US4755870 A.

Muja, M., and Lowe, D. 2014. Scalable nearest neighbour algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Nesterov, Y. 2004. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer.

Nguyen, H.; Peng, X.; Do, M.; and Liang, Z.-P. 2013. Denoising MR spectroscopic imaging data with low-rank approximations. *IEEE Transactions on Biomedical Engineering* 60(1):78–89.

Nocedal, J., and Wright, S. J. 2006. *Numerical Optimization*. Springer.

Peng, Y.; Ganesh, A.; Wright, J.; Xu, W.; and Ma, Y. 2012. RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(11):2233–2246.

Wang, S., and Zhang, Z. 2012. Colorization by matrix completion. In *Proceedings of the 26th Conference on Artificial Intelligence*.

Yang, J.; Wright, J.; Huang, T. S.; and Ma, Y. 2010. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing* 19(11):2861–2873.