Published in Image Processing On Line on 2021–01–09.
Submitted on 2019–11–18, accepted on 2020–08–19.
ISSN 2105–1232 © 2021 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2021.286

Image Inpainting using Patch Consensus and DCT Priors

Ignacio Ramírez Paulino, Ignacio Hounie

Universidad de la República, Uruguay (nacho,ihounie@fing.edu.uy)

Communicated by Pablo Arias *Demo edited by* Pablo Arias

Abstract

We present an implementation of the PACO-DCT inpainting algorithm. This method is based on maximizing the likelihood of image patches in terms of their DCT coefficients, while requiring consensus on the overlapping patches. The resulting problem is solved as an instance of the PACO framework.

Keywords: PACO; DCT; inpainting

Source Code

The source code for the presented algorithm is available at the associated web page[1].

Supplementary Material

Supplementary material can be found at the project home page[2].

1 Introduction

This paper showcases the application of the Patch Consensus (PACO) framework [12] to the problem of image inpainting. PACO belongs to the recent family of algorithms [4, 8, 10, 17] which aims at reducing undesired blur in the output of patch-based image processing algorithms by seeking solutions where the patches agree at their intersections. Similarly to [4, 8], rather than promoting, PACO *imposes* agreement as a hard constraint in the optimization problem. Contrary to [4, 8], PACO is not restricted to linear and/or sparse patch models. In fact, any energy defined in terms of the patches can be plugged into the PACO framework. Besides its generality, the hard constraint imbued in PACO defines a one-to-one mapping between image and patch spaces, thus allowing for further problem constraints to be defined in either one of these spaces.

The inpainting algorithm described in this work is made up of three building blocks: a weighted ℓ_1 cost function over the DCT transform of patches; the requirement that estimated patches coincide where they overlap in the image (patch consensus); and the inpainting constraint, that is, that the

[1] https://doi.org/10.5201/ipol.2021.286
[2] http://iie.fing.edu.uy/~ihounie/paco/

estimated image coincides exactly with the known samples of the input image. As we will see next, the algorithm is easy to implement, fast, and effective on scratches and holes of moderate size.

This document is organized as follows. Section 2 describes the notation and the main mathematical concepts used thereafter. Section 3 introduces the general PACO framework and its general solution. Section 4 deals with the PACO-DCT problem and the resulting inpainting algorithm. Section 5 provides technical details relevant to understanding the accompanying implementation. Experimental results are shown and discussed in Section 6. The paper is concluded in Section 7.

2 Patch Extraction, Patch Stitching and the Consensus Set

Given a 1D signal $\mathbf{x}$ of length N, the extraction operation is determined by a set $\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$ of linear operators $\mathcal{R}_j : \mathbb{R}^N \to \mathbb{R}^{m_j}$. Each $\mathcal{R}_j$ maps a patch from $\mathbf{x}$ to a vector $\mathbf{y}_j$ of size m_j, so that the total number of patches extracted is n. The operator $\mathcal{R}_j$ can be expressed as $\mathbf{y}_j = \mathbf{R}_j\mathbf{x}$, where each row of $\mathbf{R}_j \in \mathbb{R}^{m_j \times N}$ corresponds to a vector $\mathbf{e}^i$ of the canonical basis of $\mathbb{R}^N$. The concatenated output of all operators, $\mathbf{y} \in \mathbb{R}^{\sum_j m_j}$, can be written as $\mathbf{y} = \mathbf{R}\mathbf{x}$, where $\mathbf{R}^\intercal = [\ \mathbf{R}_1^\intercal \mid \mathbf{R}_2^\intercal \mid \cdots \mid \mathbf{R}_n^\intercal\ ]$. In the common case where $m_j = m$ for all j, the *patches vector* $\mathbf{y} \in \mathbb{R}^{mn}$ can be written as a *patches matrix* $\mathbf{Y} \in \mathbb{R}^{m \times n}$, where each of its columns corresponds to a patch; in this case, we will use both forms interchangeably. In any case, we will denote the space of all possible patches matrices/vectors as $\mathbb{R}^{mn}$.

The operator $\mathcal{R}$ defines a *linear isomorphism* between $\mathbb{R}^N$ and linear subspace $\mathbb{C} \subset \mathbb{R}^{mn}$ which we call the *consensus set*.

By *stitching* we refer to the operation that maps a patches vector $\hat{\mathbf{y}} \in \mathbb{R}^{mn}$ onto a signal $\hat{\mathbf{x}}$ (the hat is used to denote *estimation* hereafter). This operation should coincide with the inverse mapping of $\mathcal{R}$ when applied to $\mathbb{C}$. If we extend the domain of the stitching operator to all patch matrices in $\mathbb{R}^{mn}$, including those outside of $\mathbb{C}$, its definition is not unique. A common choice is to seek for the signal estimate $\hat{\mathbf{x}}$ whose patches, when extracted, are as close as possible to the individually estimated ones in $\hat{\mathbf{y}}$. Using the ℓ_2 norm to measure this distance leads to the least squares estimation

$$\hat{\mathbf{x}}_{ls} = \arg\min_{\mathbf{x}} \|\mathbf{R}\mathbf{x} - \hat{\mathbf{y}}\|_2 = \left(\sum_j \mathbf{R}_j^\intercal \mathbf{R}_j\right)^{-1} \sum_j \mathbf{R}_j^\intercal \hat{\mathbf{y}}_j = (\mathbf{R}^\intercal\mathbf{R})^{-1}\mathbf{R}^\intercal\hat{\mathbf{y}}. \tag{1}$$

Equation (1) is interpreted as follows: it is easy to verify that $\mathbf{R}_j^\intercal\hat{\mathbf{y}}_j$ puts back the patch $\hat{\mathbf{y}}_j$ in its corresponding place in $\hat{\mathbf{x}}_{ls}$; thus, $\mathbf{R}^\intercal\hat{\mathbf{y}}$ produces a vector of length N where the i-th element contains the sum of all the elements of $\hat{\mathbf{y}}$ that are mapped there by one or more $\mathbf{R}_j$. On the other hand $\mathbf{R}_j^\intercal\mathbf{R}_j$ is a diagonal matrix with m ones, and so $\mathbf{R}^\intercal\mathbf{R} = \sum_j \mathbf{R}_j^\intercal\mathbf{R}_j$ is a diagonal matrix whose (i,i)-th element counts how many $\mathbf{R}_j$s have mapped some element of $\hat{\mathbf{y}}$ to the sample i. Thus, the i-th entry of the right hand side of (1) is the average of all the different estimates of the i-th sample in $\hat{\mathbf{y}}$. We refer to the operator defined in (1) as the *average stitching operator* and denote it by $\mathcal{S} : \mathbb{R}^{mn} \to \mathbb{R}^N$, $\hat{\mathbf{x}}_{ls} = \mathcal{S}(\hat{\mathbf{y}})$.

2.1 Patch Re-projection and the Stitching Trick

The extraction operator $\mathcal{R}$ is given by $\mathbf{y} = \mathcal{R}(\mathbf{x}) = \mathbf{R}\mathbf{x}$. If we compose this with the stitching operator $\mathcal{S}$ defined earlier we obtain

$$\mathcal{R}[\mathcal{S}(\hat{\mathbf{y}})] = \mathbf{R}(\mathbf{R}^\intercal\mathbf{R})^{-1}\mathbf{R}^\intercal\hat{\mathbf{y}} = \Pi_{\mathbb{C}}(\hat{\mathbf{y}}), \tag{2}$$

where $\Pi_{\mathbb{C}}(\hat{\mathbf{y}})$ is the orthogonal projection of $\hat{\mathbf{y}} \in \mathbb{R}^{mn}$ onto $\mathbb{C}$. Thus, we conclude that the projector operator (2) from $\mathbb{R}^{mn}$ onto $\mathbb{C}$ can be written as the composition of the extraction operator with its

corresponding average stitching operator. Although (2) is a well known result, to the best of our knowledge, its straightforward implementation as the aforementioned composition has not been fully exploited in the literature. Rather, works such as [4, 8] compute the orthogonal projection matrix $\mathbf{R}(\mathbf{R}^\intercal\mathbf{R})^{-1}\mathbf{R}^\intercal$ explicitly, something that quickly becomes unwieldy as the signal size increases even if the structure of such matrix is exploited (e.g., block sparsity).

Note that the aforementioned result does not impose any particular requirement on the extraction operator $\mathbf{R}$. In particular, it does not require equally-sized patches.

The preceding discussion generalizes to signals $\mathbf{x}$ of any dimension by *vectorizing* $\mathbf{x}$ prior to extraction: $\mathbf{y}_j = \mathbf{R}_j\text{vec}(\mathbf{x})$. Here $\text{vec}(\cdot)$ arranges its argument into a vector by traversing its elements in some predefined order. In our case, for $2D$ signals of size $M\times N$, we follow a row-major ordering, that is, the output vector is the concatenation of the M input rows. The inverse of $\text{vec}(\cdot)$ is called the matrification operator and is denoted by $\text{mat}(\cdot)$.

3 PACO: The PAtch COnsensus Problem

Given a patch extraction operator $\mathcal{R} : \mathbb{R}^N \to \mathbb{R}^{mn}$, a cost function $f(\mathbf{Y}) : \mathbb{R}^{mn} \to \mathbb{R}$, and a constraint set $\Omega \subseteq \mathbb{R}^{mn}$, the PACO problem is stated as follows

$$\hat{\mathbf{Y}} = \arg\min_{\mathbf{Y}} f(\mathbf{Y}) \quad \text{s.t.} \quad \mathbf{Y} \in \Omega \cap \mathbb{C}. \tag{3}$$

In [12] we show that the problem of patch consensus is a particular case of the General Consensus Problem when $\Omega = \mathbb{C}$ (see [11, Chapter 5]). We will see later how to treat the general case $\Omega \neq \mathbb{C}$. At this point, we shall remind the reader that $\mathbb{C}$ is isomorphic to $\mathbb{R}^N$, so that the constraint set can be defined in either $\mathbb{R}^N$ or $\mathbb{C}$. For example, in order to clip the resulting samples to the range $[0 - 255]$ we could define $\Omega = \{\mathbf{Y} \in \mathbb{R}^{mn} : \mathcal{S}(\mathbf{Y}) \in [0, 255]^N\}$. The inpainting constraint that we describe later in Section 4 falls into this category.

3.1 Problem Formulation

We now define the *convex indicator function* associated to the PACO constraint set $\mathbb{C} \cap \Omega$, $g(\cdot) : \mathbb{R}^{mn} \to \mathbb{R} \cup \{+\infty\}$,

$$g(\mathbf{Y}) = \begin{cases} 0 & , \quad \mathbf{Y} \in \mathbb{C} \cap \Omega \\ +\infty & , \quad \mathbf{Y} \notin \mathbb{C} \cap \Omega \end{cases} . \tag{4}$$

This allows us to transform the PACO problem into an unconstrained one,

$$\hat{\mathbf{Y}} = \arg\min_{\mathbf{Y}} f(\mathbf{Y}) + g(\mathbf{Y}). \tag{5}$$

The problem (5) will be convex if the function $f(\cdot)$ and the set Ω are also convex. This allows for convex restoration problems to be defined under consensus constraints. This being said, the PACO framework (5) is not limited to convex problems.

3.2 General Algorithm

The following method is based on the *proximal operator form* [11] of the popular *Alternating Directions Method of Multipliers* (ADMM) [6]. The proximal operator of a function $f(\cdot)$ with parameter λ is given by,

$$\text{prox}_{\lambda f}(y) := \arg\min f(x) + \frac{1}{2\lambda}\|y - x\|^2. \tag{6}$$

The proximal operator has many interesting properties (see [11] and references therein). Among them, if $g(\cdot)$ is the indicator function of a set A we have that

$$\text{prox}_{\lambda g}(\cdot) = \Pi_A(\cdot).$$

This is particularly useful within the PACO framework, as $\Pi_{\mathbb{C}}(\cdot)$ can be computed efficiently using the stitching trick described in Section 2.1, Equation (2). However, in general, this trick cannot be applied directly to (5). In order to do that, we need to reformulate (5) so that projecting onto the constraint set, and minimizing the target cost function $f(\cdot)$, can be done separately. One such reformulation is the so called *splitting*,

$$(\hat{\mathbf{Y}}, \hat{\mathbf{Z}}) = \arg\min_{\mathbf{Y},\mathbf{Z}} f(\mathbf{Y}) + g(\mathbf{Z}) + \frac{1}{2\lambda}\|\mathbf{Y} - \mathbf{Z}\|_F^2 \quad \text{s.t.} \quad \mathbf{Y} = \mathbf{Z}, \tag{7}$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix. The ADMM provides a general way to obtain a solution to (7) which is guaranteed to be global if the problem is convex, and local otherwise. The particular ADMM steps for this case are described in Algorithm 1.

Algorithm 1: General PACO ADMM algorithm

input : $\mathbf{Y}^{(0)}$, $\mathbf{Z}^{(0)}$, $\lambda^{(0)}$, step reduction factor $0 < \kappa \leq 1$, convergence criteria
output: $\mathbf{Y}^*$, $\mathbf{Z}^*$

1 $t \leftarrow 0$, $\mathbf{U}^{(0)} \leftarrow \mathbf{0}$;
2 **repeat**
3 $\quad \mathbf{Y}^{(t+1)} \leftarrow \text{prox}_{\lambda f}\left(\mathbf{Z}^{(t)} - \mathbf{U}^{(t)}\right)$; `// problem dependent: f`
4 $\quad \mathbf{Z}^{(t+1)} \leftarrow \Pi_{\mathbb{C}\cap\Omega}\left(\mathbf{Y}^{(t+1)} + \mathbf{U}^{(t)}\right)$; `// problem dependent: Ω`
5 $\quad \mathbf{U}^{(t+1)} \leftarrow \mathbf{U}^{(t)} + \mathbf{Y}^{(t+1)} - \mathbf{Z}^{(t+1)}$; `// trivial, problem independent`
6 $\quad \lambda^{(t+1)} \leftarrow \kappa\lambda^{(t)}$;
7 $\quad t \leftarrow t + 1$;
8 **until** *convergence criteria are met*;

Steps 3 and 4 of Algorithm 1 are problem dependent. In particular, if $\Omega = \mathbb{R}^{mn}$ then $g = \Pi_{\mathbb{C}}(\cdot)$ and 4 can be solved using the stitching trick (2),

$$\mathbf{Z}^{(t+1)} = \mathcal{R}\left[\mathcal{S}\left(\mathbf{Y}^{(t+1)} + \mathbf{U}^{(t)}\right)\right].$$

If $\Omega \subset \mathbb{C}$ then $\Pi_{\Omega\cap\mathbb{C}}(\cdot) = \Pi_{\Omega}[\Pi_{\mathbb{C}}(\cdot)]$. If $\Omega \neq \mathbb{C}$, a general solution to step 4 can be obtained iteratively using Dykstra's Algorithm 2. Step 3 depends on $f(\cdot)$. In fact, any restoration method which can be formulated as the minimization of a function defined in patch space $f : (\mathbf{Y}) : \mathbb{R}^{mn} \to \mathbb{R}$ can be accommodated into the PACO framework by plugging its corresponding proximal operator into Step 3.

We will leave the convergence criteria undefined for now, as its choice depends on the particular application and practical implementation decisions.

Finally, the ADMM parameter λ is (optionally) decreased at each iteration. While this is not necessary for convergence and is not part of the standard ADMM algorithm, a good choice of κ can improve the practical convergence rate significantly.

The next section describes the implementation of Steps 3 and 4 of Algorithm 1 for the PACO-DCT algorithm implemented in this work.

Algorithm 2: Dykstra's algorithm for computing the projection of a point α onto the intersection of two sets A and B

input : point α, sets A and B
output: $\beta = \Pi_{A\cap B}(\alpha)$

$t \leftarrow 0$, $\alpha^{(0)} \leftarrow \alpha$;
$\mathbf{p}^{(0)} \leftarrow \mathbf{0}$, $\mathbf{q}^{(0)} \leftarrow \mathbf{0}$;
repeat
  $\beta^{(t)} \leftarrow \Pi_B(\alpha^{(t)} + \mathbf{p}^{(t)})$;
  $\mathbf{p}^{(t+1)} \leftarrow \alpha^{(t)} + \mathbf{p}^{(t)} - \beta^{(t)}$;
  $\alpha^{(t+1)} \leftarrow \Pi_A(\beta^{(t)} + \mathbf{q}^{(t)})$;
  $\mathbf{q}^{(t+1)} \leftarrow \beta^{(t)} + \mathbf{q}^{(t)} - \alpha^{(t+1)}$;
  $t \leftarrow t + 1$;
until *convergence*;

4 PACO-DCT Inpainting

We now describe the application of the PACO framework to the problem of filling in missing samples in signals, where such missing samples appear as contiguous, large regions. This problem is commonly known as *inpainting* in the image/video processing literature. Our method seeks to minimize the weighted ℓ_1 norm of the Discrete Cosine Transform (DCT) coefficients of the estimated signal patches $\hat{\mathbf{Y}}$ while keeping up with the consensus constraint and with the known samples of the signal. As shown later in Section 6, the method appears to be effective on a number of interesting scenarios.

4.1 Inpainting Constraint Set

In the following discussion we will be dealing with sets of signal and patch indexes; here $[N]$ will be shorthand for $\{1, \ldots, N\}$. We denote the set of *observed* samples as

$$O = \{i \in [N] : x_i \neq ?\},$$

where the symbol "?" represents an unknown sample value. The inpainting task is to infer the unknown sample values $\{x_i : i \in O^c\}$ from the known samples $\{x_i : i \in O\}$. The sets O and O^c are typically specified using a binary mask of the same size as the input signal, where a 1 at position i indicates that x_i is *unknown*. The masks used in this work are shown in Figure 2.

In general, it is desirable that the estimation $\hat{\mathbf{x}}$ coincides with the signal $\mathbf{x}$ in those places indexed by O. This *inpainting constraint set* is naturally defined in signal space as

$$\Gamma = \{\hat{\mathbf{x}} \in \mathbb{R}^N : \hat{x}_i = x_i, i \in O\}. \tag{8}$$

The corresponding constraint set in patch space is given by,

$$\Omega = \{\hat{\mathbf{Y}} \in \mathbb{R}^{mn} : \mathcal{S}(\hat{\mathbf{Y}})_i = x_i, i \in O\}. \tag{9}$$

Given O and $\mathcal{R}$, we also define the set $J \in [n]$ of the *incomplete* patches as

$$J = \{\, j \in [n] : \exists\, i \in [m] \mid (\mathbf{y}_j)_i = ?\,\}.$$

The *complete* patches are those whose patch indexes do not belong to J. These sets, along with other index sets of practical interest, are depicted in Figure 1.

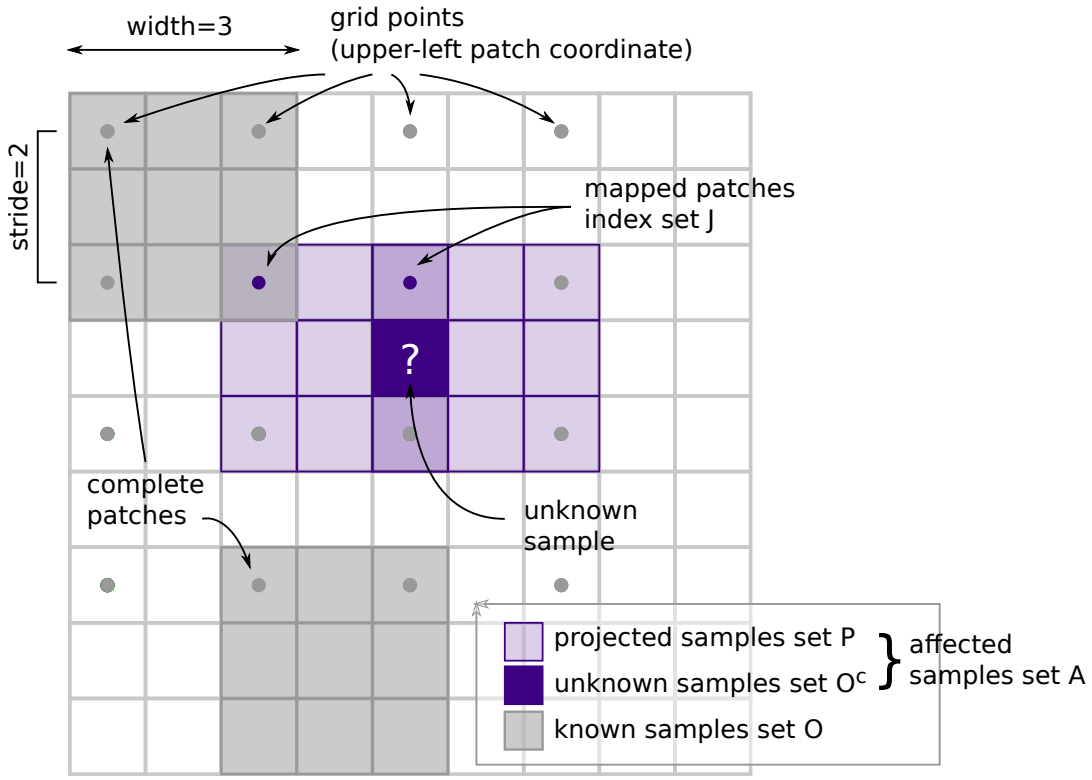

Figure 1: Example of precomputed index sets; linear indexes begin at 1, and image indexes at $(1,1)$. Left: sample 9×9 image decomposed into 3×3 patches separated by a stride $s=2$, leading to 16 different patches. There is one unknown sample at position $(4,5)$; the corresponding set of missing sample linear indices is $O^c=\{32\}$. There are two incomplete patches whose upper-left corners are at image coordinates $(3,3)$ and $(3,5)$, so that their linear indexes are $J=\{21,23\}$. The set A indexes all the pixels contained in the incomplete patches $\mathbf{y}_j, j\in J$. In this example we have $A=\{21,\dots,25,30,\dots,34,39,\dots,43\}$. The index set of projected samples P is given by $A\setminus O=\{1,\dots,31,33,\dots,81\}$.

4.2 Cost Function

Given an image patch $\mathbf{y}_j\in\mathbb{R}^m$, and a matrix $\mathbf{D}\in\mathbb{R}^{m\times m}$ whose columns correspond to the basis elements of the (inverse) orthogonal DCT type II, the corresponding DCT coefficients vector is given by

$$\mathbf{a}_j=\mathbf{D}^{-1}\mathbf{y}_j.$$

The DCT has long been used as an energy-concentrating transform for images and image patches. It is well known that DCT coefficients of natural images exhibit a heavy-tailed, Laplacian-like distribution (see e.g. [14] for a detailed discussion). Under this model, the joint distribution of the coefficients vector $\mathbf{a}_j=\mathbf{D}^{-1}\mathbf{y}_j$ is characterized by the following multivariate Laplacian density function

$$p(\mathbf{a}_j)=\prod_{i=1}^{m}\frac{2}{\omega_i}e^{-\omega_i|a_{i,j}|}, \tag{10}$$

where the scale parameters $\omega_i, i=1,\dots,m$ typically vary significantly with i. In order to produce a signal estimate, we seek to minimize the negative log-likelihood of (10) summed over all patches $\mathbf{a}_j, j\in[n]$. After discarding constant additive terms, we obtain the following cost function defined *over the* DCT *coefficients*

$$f'(\mathbf{A})=\sum_{j=1}^{n}\sum_{i=1}^{m}\omega_i|a_{i,j}|. \tag{11}$$

The corresponding cost function over the patches matrix $\mathbf{Y}$ is $f(\mathbf{Y})=f'(\mathbf{D}^{-1}\mathbf{Y})$. Of course, many priors which are more expressive and accurate than the Laplacian DCT exist: Non-Local [3, 7],

Sparse Models [2], Gaussian Mixture Models [16], Bayesian [1], or Patch Manifolds [13], to name a few. Our choice is based on two aspects of the DCT. First, being an orthogonal transform, it allows for a much simpler and faster solution of the resulting optimization problem (more on this later). Second, it is very fast to compute, with specialized, high performance libraries such as the FFTW[3] readily available on many platforms and languages. To fix ideas, computing the DCT of an 8×8 grayscale patch is about two orders of magnitude faster than a matrix-vector product for a matrix of size 64×64.

The weights $\{\omega_i : i \in [m]\}$ are not known a priori. Our implementation computes the Maximum Likelihood Estimator (MLE) of each $\omega_i, i \in [m]$ given the coefficients of the patches $a_{i,j}, j \notin J$. We then scale the resulting MLE estimators so that $\min_{i\in[m]}\{\omega_i\} = 1$; this improves the numerical stability of the method. The resulting weights are given by

$$\omega_i = \frac{\min_{i\in m}\{\hat{w}_i\}}{\hat{w}_i}, \quad \hat{w}_i = \sum_{j\notin J} |a_{i,j}|, \quad \mathbf{a}_j = \mathbf{D}^{-1}\mathbf{y}_j. \tag{12}$$

4.3 Inpainting Algorithm

According to the cost function and constraint sets defined before, the PACO-DCT inpainting problem (splitting included) is given by,

$$\hat{\mathbf{Y}} = \arg\min_{\mathbf{Y}} \sum_{i,j} \omega_i \left|(\mathbf{D}^{-1}\mathbf{y}_j)_i\right| + g(\mathbf{Z}), \quad \text{s.t.} \quad \mathbf{Y} = \mathbf{Z} \tag{13}$$

It is easier to define the problem directly in terms of the corresponding DCT coefficients,

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} \sum_{i,j} \omega_i |a_{i,j}| + g'(\mathbf{B}). \tag{14}$$

where $\mathbf{B} = \mathbf{D}^{-1}\mathbf{Z}$ and $g'(\mathbf{B}) = g(\mathbf{DB})$. The proximal operator of $f'(\mathbf{A}) = \sum_{i,j} \omega_i |a_{i,j}|$ is known as the *soft-thresholding* operator and is given by (15),

$$\mathcal{T}_{\lambda w_i}(a) = \min\{a + \lambda w_i, \max\{0, a - \lambda w_i\}\}. \tag{15}$$

As for $g'(\cdot)$, it is easy to show that $\text{prox}_{\lambda g'}(\cdot) = \mathbf{D}^{-1}\text{prox}_{\lambda g}(\mathbf{D}\cdot)$ when $\mathbf{D}$ is orthogonal (the non-orthogonal case is treated in detail in [12]). In our case, $\text{prox}_{\lambda g}(\cdot) = \Pi_{\mathbb{C}\cap\Omega}(\cdot)$.

From (8) and (9) it is clear that $\Gamma = \mathcal{S}(\Omega)$. As $\mathcal{R}$ and $\mathcal{S}$ define a linear isomorphism between $\mathbb{R}^N$ and $\mathbb{R}^{mn}$, we then have that $\Omega \cap \mathbb{C}$ is the *preimage* of Γ under $\mathcal{S}$. Thus, we can project a patches matrix $\mathbf{Y} \in \mathbb{R}^{mn}$ onto $\Omega \cap \mathbb{C}$ by first applying $\mathcal{S}(\hat{\mathbf{Y}})$, then projecting the resulting signal $\hat{\mathbf{x}}$ onto Γ, and then inverting the mapping using $\mathcal{R}$,

$$\text{prox}_{\lambda g}(\mathbf{Z}) = \Pi_{\mathbb{C}\cap\Omega}(\mathbf{Z}) = \mathcal{R}\left\{\Pi_\Gamma\left[\mathcal{S}(\mathbf{Z})\right]\right\}. \tag{16}$$

Combining (16) with the aforementioned property of the proximal operator of an orthogonal transformation we obtain the following expression for the proximal operator defined in terms of the DCT coefficients

$$\text{prox}_{\lambda g'}(\mathbf{B}) = \mathbf{D}^{-1}\text{prox}_{\lambda g}(\mathbf{DB}) = \mathbf{D}^{-1}\left(\mathcal{R}\left\{\Pi_\Gamma\left[\mathcal{S}(\mathbf{DB})\right]\right\}\right). \tag{17}$$

Algorithm 3 describes the pseudo-code which implements the proximal operator of $g'(\cdot)$ as given by (17). Algorithm 4 provides the complete pseudo-code of PACO-DCT, with Algorithm 3 inlined in steps 4–6.

[3] http://fftw.org/.

Algorithm 3: Proximal operator for the indicator function $g'(\cdot)$.

input : DCT coefficients matrix $\mathbf{B}$, observed signal $\mathbf{x}$, known sample indices set O
output: Projection of $\mathbf{B}$ onto the inpainting constraint set, $\Pi_{\mathbb{C}\cap\Omega}(\mathbf{B})$

$\mathbf{Z} \leftarrow \mathbf{DB}$; // Convert DCT coefficients to patch samples
$\mathbf{v} \leftarrow \mathcal{S}(\mathbf{Z})$; // Stitch patches in signal space
$v_i \leftarrow x_i,\ \forall\, i \in O$; // overwrite estimated samples to known values in x; this is Π_Γ
$\mathbf{Z} \leftarrow \mathcal{R}(\mathbf{v})$; // Go back to patch space
$\Pi_{\mathbb{C}\cap\Omega}(\mathbf{B}) \leftarrow \mathbf{D}^{-1}\mathbf{Z}$; // Get DCT coefficients from samples

Algorithm 4: Complete PACO-DCT algorithm (in DCT space).

input : signal $\mathbf{x}$, initial estimation $\hat{\mathbf{x}}^{(0)}$, observed sample indices set O, initial stepsize $\lambda^{(0)}$, step reduction factor $0 < \kappa \leq 1$, convergence criteria
output: inpainted signal $\hat{\mathbf{x}}$

$\mathbf{B}^{(0)} \leftarrow \mathbf{D}\mathcal{R}(\hat{\mathbf{x}}^{(0)})$; $\mathbf{U}^{(0)} \leftarrow \mathbf{0}$;
repeat
  $a_{i,j}^{(t+1)} \leftarrow \mathcal{T}_{\lambda\omega_{i,j}}(b_{i,j}^{(t)} - u_{i,j}^{(t)}),\ \forall\, i,j$; // Soft thresholding
  $\hat{\mathbf{x}}^{(t+1)} \leftarrow \mathcal{S}[\,\mathbf{D}(\,\mathbf{A}^{(t+1)} + \mathbf{U}^{(t)}\,)\,]$; // Convert to patch space and stitch
  $\hat{x}_i^{(t+1)} \leftarrow x_i^{(t+1)},\ \forall\, i \in O$; // Project onto Γ.
  $\mathbf{B}^{(t+1)} \leftarrow \mathbf{D}^{-1}\mathcal{R}(\hat{\mathbf{x}}^{(t+1)})$; // Extract patches and convert them to DCT space
  $\mathbf{U}^{(t+1)} \leftarrow \mathbf{U}^{(t)} + \mathbf{A}^{(t+1)} - \mathbf{B}^{(t+1)}$; // Update multiplier
  $\lambda^{(t+1)} \leftarrow \kappa\lambda^{(t)}$;
  $t \leftarrow t + 1$;
until *convergence criteria are met*;

5 Implementation Details

5.1 Image Padding and Patch Indexation

Absolute pixel coordinates with respect to the whole image start at $(0, 0)$ for the upper-left corner of the image. Pixel coordinates within a patch are relative to the upper-left pixel of the patch (this deviates from the de-facto standard of using the center of the patch as the origin of the relative pixel coordinates, but simplifies some technicalities in our discussion). Depending on the size of the image, the patch size, and the stride, padding might be required to the right and/or bottom of the image, in which case we reflect the borders. If N is the dimension of the original signal and N' that of the padded one, we define $x_{N+k} = x_{N+1-k}$, $1 \leq k \leq N' - N$. For simplicity, we will assume that $N = N'$ in the rest of the document.

5.2 Pre-computation of Indexes and Normalization Factors

To begin with, any patches that are already complete can be ignored altogether. Thus, the optimization works only on the incomplete patches (indexed by $J \subset [n]$) and their respective samples. Additional computational savings can be obtained if we pre-compute the indexes of the samples involved in the optimization, namely those which belong to any incomplete patch $\mathbf{y}_j, j \in J$; we call this the *affected indexes set* A. The set A can then be subdivided into two: the subset of A which corresponds to *known* samples, that is $P = A \cap O$, and the set of missing samples, O^c. Finally, we

Algorithm 5: Pre-computation of indexes and normalization factors. Below we show the simplified algorithm for patches of same size m on a regular grid over a 1D signal.

input : known samples indexes O, extraction operator $\mathcal{R} = \{\mathbf{R}_j, j = 1, \ldots, n\}$.
output: Incomplete patch indexes $J \subset [n]$, affected samples indexes $A \subset [N]$
output: Projected samples indexes $P \subset [N]$, normalization factors $\mathbf{c}$.

$J \leftarrow \emptyset, A \leftarrow \emptyset, P \leftarrow \emptyset$;
$\mathbf{h} = \mathbf{0}_N$; $\mathbf{h}_i \leftarrow 1$, $\forall i \notin O$; `// Pseudo-signal: missing samples mask.`
$\mathbf{s} = \mathbf{0}_N$; `// Pseudo-signal: pixel occurrence counter.`
for $j \leftarrow 1$ **to** n **do**
  $\mathbf{z}_j \leftarrow \mathbf{R}_j\mathbf{h}$; `// Extract j-th pseudo-patch from the missing samples mask.`
  if $\mathbf{z}_j \neq \mathbf{0}_N$ **then** `// There is at least one missing sample in the pseudo-patch`
    $J \leftarrow J \cup j$; `// Thus, the j-th patch is incomplete.`
    $\mathbf{s} \leftarrow \mathbf{s} + \mathbf{R}_j^\intercal \mathbf{1}_m$; `// Increment s where the j-th patch falls.`
  end
end
$A \leftarrow \{i \in [N] : s_i > 0\}$; `// Zeroes in s correspond to non-affected samples.`
$P \leftarrow A \cap O$; `// P is the set of affected and observed samples.`
$\mathbf{c} \leftarrow \{1/s_i : i \in A\}$; `// Normalization factor: inverse of counter.`

can speed up the stitching operation if we pre-compute the normalization factors

$$\mathbf{c} = \mathrm{diag}(\mathbf{R}^\intercal\mathbf{R})^{-1} \in \mathbb{R}^{|A|}.$$

Algorithm 5 shows a method for computing the sets J, A, P and the normalization factors vector $\mathbf{c}$. Figure 1 illustrates these sets on a toy example.

5.3 Color Representation

Color images are first converted to the YUV color space, each channel is processed as a grayscale image, and the final result is mapped back to RGB.

5.4 Initialization

If available, the user may explicitly provide an existing image as the initial signal estimate. Otherwise, the initial values of $\hat{\mathbf{x}}^{(0)}$ at the unknown places are filled with the average values of the known places, that is

$$\hat{x}_i = \bar{x}\,,\ \forall i \in O^c\,, \quad \bar{x} = \frac{1}{|O|}\sum_{r \in O} x_r.$$

5.5 Convergence Criteria

The algorithm will stop if a number of maximum iterations is reached or the relative change between iterations of $f'(\mathbf{B})$, $|f'(\mathbf{B}^{(t+1)}) - f'(\mathbf{B}^{(t)})|/|f'(\mathbf{B}^{(t+1)}| < \epsilon$ for $0 < \epsilon \ll 1$.

6 Results and Discussion

The experiments in this section address three important questions about the proposed PACO-DCT method: i) how good is the convergence of the numerical optimization algorithm? ii) how do the

problem and algorithm parameters affect the quality of the result? iii) overall, is the method competitive compared to the state of the art?

6.1 Experimental Setup

Tests were performed on the Kodak image dataset [4]. This dataset contains 24 portrait (512×768) and landscape (768×512) color images. These images are sharp, high quality, and losslessly compressed; some examples are shown in Figure 2. We apply four different missing pixels masks to each one of these images (also shown in Figure 2).

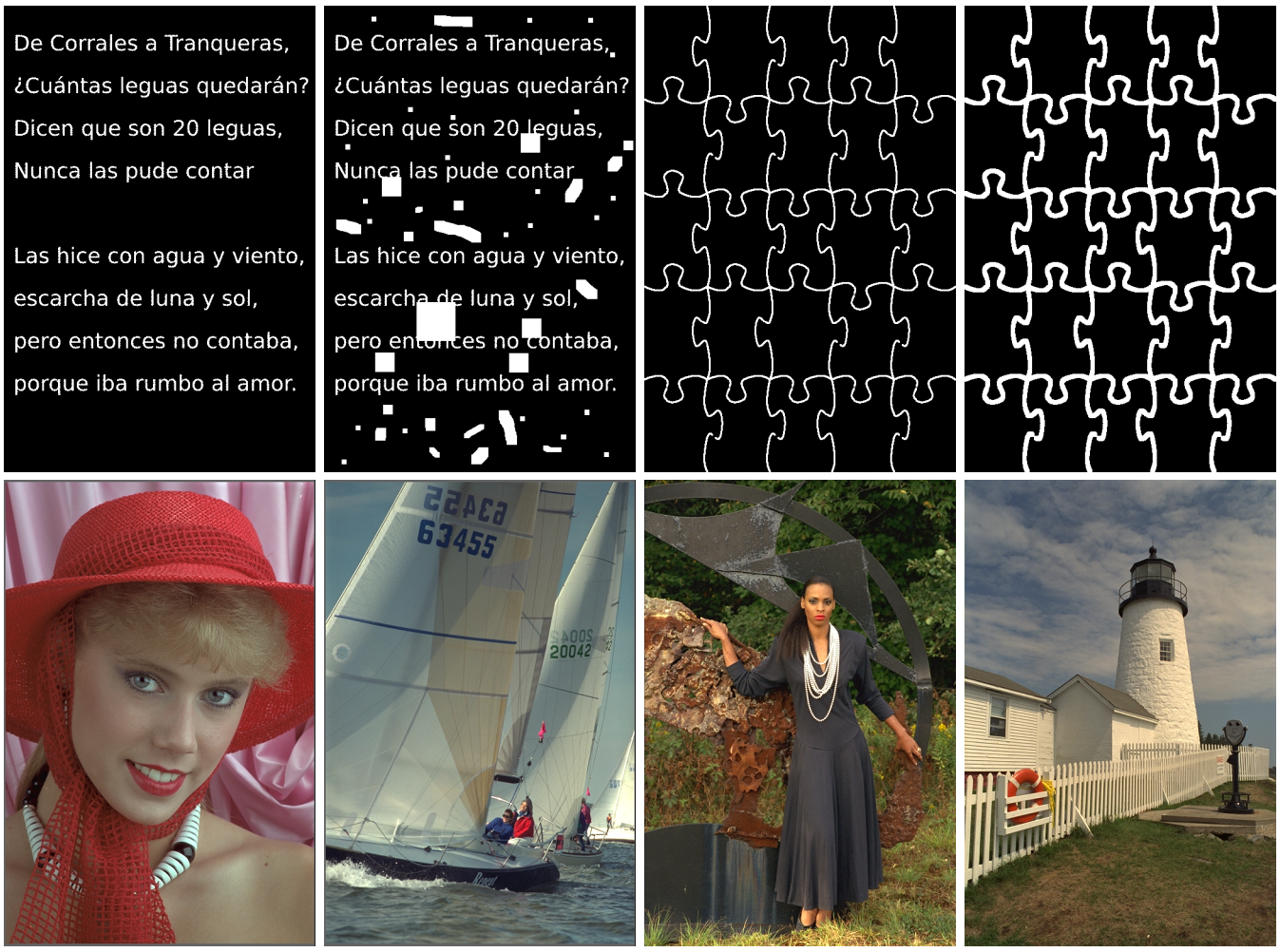

Figure 2: Inpainting masks 1 to 4 and four sample images from the Kodak dataset. Masks 2 and 4 are more challenging due to the size of the erasures.

We evaluate the quality of the results in terms of two measures: the usual Root Mean Squared Error (RMSE), and the more recent *Multiscale Structural Similarity Index Measure* (MS-SSIM) [15]. These metrics are computed *only for the missing pixels*, not on the whole images, as the known pixels are always perfectly recovered. This makes the results more sensitive to the actual performance of the algorithms at the unknown pixels, and less dependent on the number of missing pixels of each different mask.

[4]This is a dataset originally released on a CD by Kodak which was later released to the public. Many sites host a copy of this dataset. We provide our own link at http://iie.fing.edu.uy/~nacho/data/images/kodak_color.7z and our grayscale versions at http://iie.fing.edu.uy/~nacho/data/images/kodak_gray.7z.

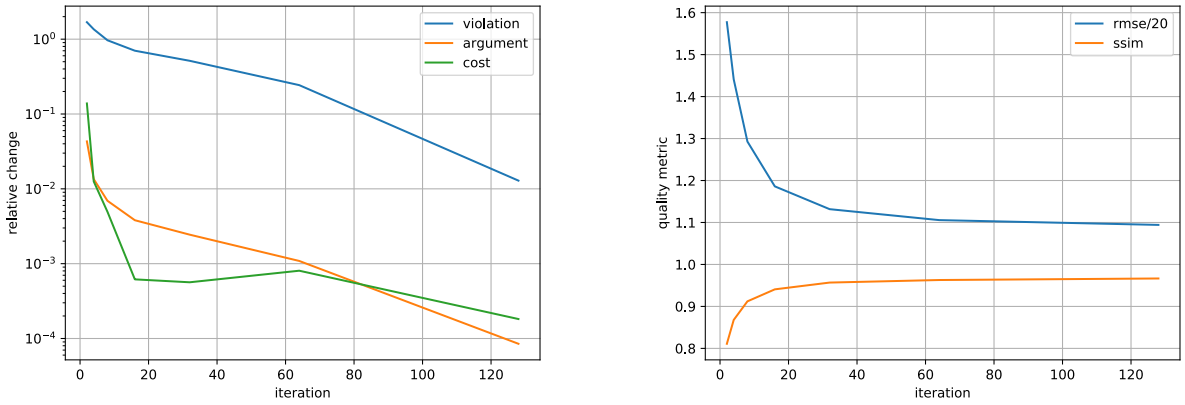

Figure 3: Convergence analysis of the ADMM method (Kodak #19, mask #2, width 16, stride 8). The left figure shows the value of the cost function $f^{(t)}$, constraint violation $\|\mathbf{A}^{(t)} - \mathbf{B}^{(t)}\|_2$ (scaled by $1/(mn)$), cost function change $|f(\mathbf{B}^{(t)}) - f(\mathbf{B}^{(t-1)})|/|f(\mathbf{B}^{(t+1)})|$, and relative argument change $\|\mathbf{B}^{(t)} - \mathbf{B}^{(t-1)}\|_2/\|\mathbf{B}^{(t)}\|_2$. The right figure shows the image quality metrics across iterations.

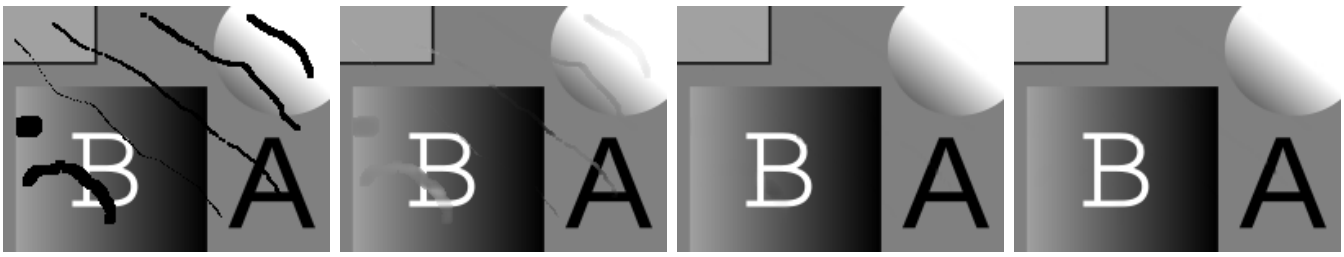

Figure 4: Inpainting results on artificial test image using fully overlapped patches of size 10×10. Left to right: input image; result after first iteration; result after 16 iterations; after 256 iterations. The first iteration is the result of estimating each patch by minimizing our DCT-based cost function once, followed by a plain patch averaging step, that is, with no consensus imposed.

6.2 Problem and Algorithm Parameters

The PACO-DCT problem depends on two parameters: the size of the patches m, and the stride s between positions in the patches grid. The computational complexity of each iteration of the PACO-DCT algorithm is linear in the number of patches, n, which in turn decreases quadratically with s. For fixed n, the complexity grows with the (linear) size of the patch m as $O(m \log_2 m)$, or as $O(w^2 \log_2 w)$ for square patches of size $m = w\times w$, where w is the width of the patch. The convergence of the optimization algorithm further depends on three additional parameters: the initial *stepsize* λ, the stepsize scaling factor κ, and the minimum allowed change in the cost function ϵ.

Numerical optimization parameters The two main parameters that affect convergence are the initial value of λ and the diminishing factor κ. For the results in this section we used $\lambda_0 = 10$ and $\kappa = 0.95$. Figure 3 shows the convergence of various magnitudes of interest using these parameters on the test example shown in Figure 4. The convergence is essentially linear (with the exception of the abrupt fall in the cost function just before iteration 20).

Problem parameters Intuitively, higher quality results should be obtained with larger patches, smaller strides and more iterations. As mentioned before, this comes at a price in computational speed. Figure 5 shows this trade-off. The good news is that the quality of the resulting images using both the objective (RMSE) and subjective (SSIM) metric do not seem to deteriorate significantly even

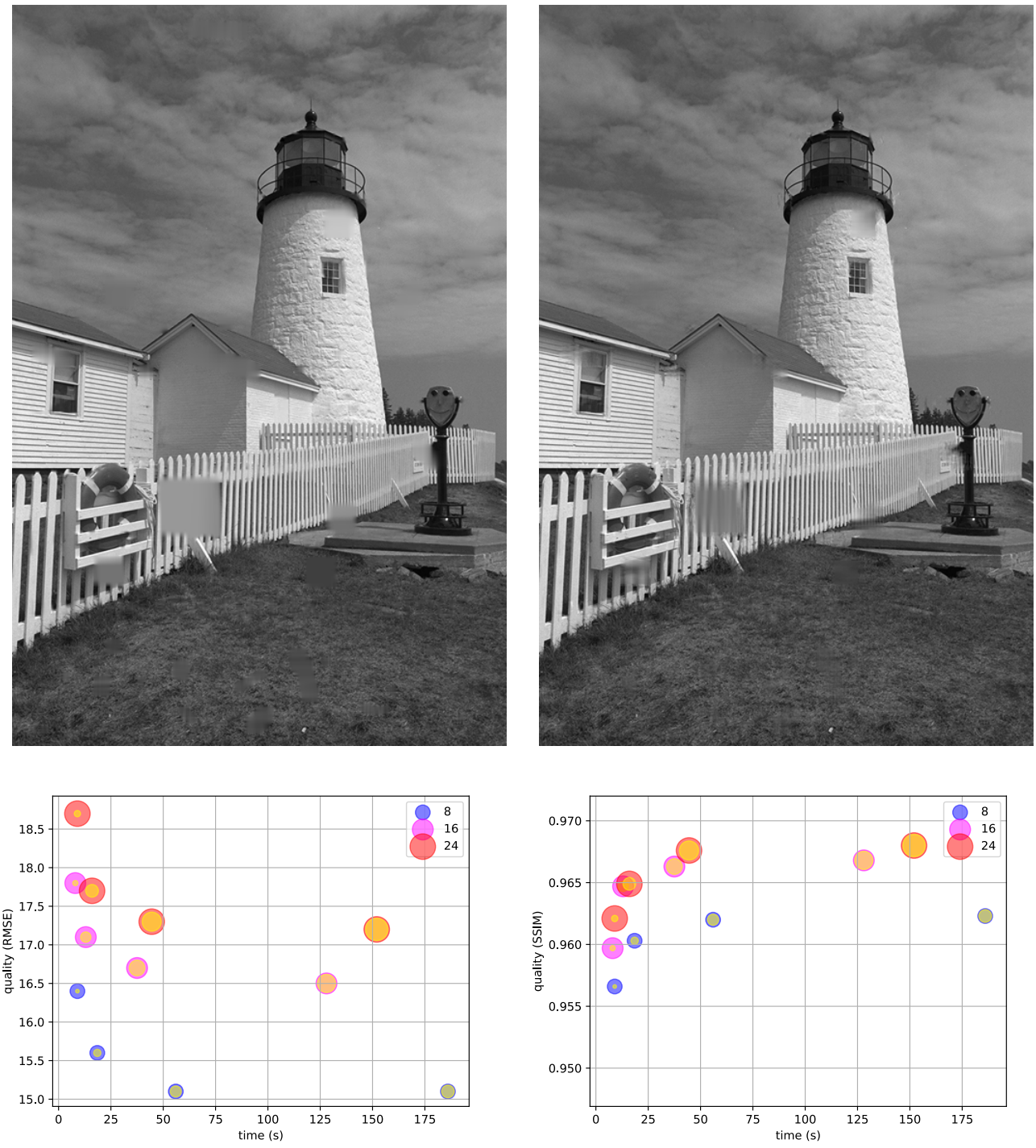

Figure 5: Quality-Time trade-off. **BOTTOM**: median RMSE and SSIM vs. the median execution time in seconds for mask #2. Optimization parameters used: $\lambda = 10$, $\kappa = 0.95$, max. iterations 512, min. change $\epsilon = 1e^{-5}$. The outer circle size corresponds to the patch width $w = 8, 16, 24$ and the inner circle indicates the relative overlap (smallest is $1/4$, largest is $7/8$). For any fixed patch size, as expected, the execution time decreases quickly with the overlap. However, the quality does not decrease significantly up to an overlap of $1/2$! Interestingly, the patch size affects the RMSE and SSIM metrics in complementary ways: larger patches improve the visual quality (SSIM) but increase the overall RMSE. **TOP**: left: best RMSE (width 8 overlap $7/8$); right: best SSIM (width 24 overlap $7/8$); notice that, accordingly with the metrics, textures are better recovered in the best SSIM case.

METRIC →	RMSE				SSIM			
MASK →	1	2	4a	4c	1	2	4a	4c
METHOD ↓	PERCENTILE 25							
PACO	**8.38**	15.68	**8.15**	**11.56**	**0.9886**	**0.9562**	**0.9879**	**0.9513**
[7]	10.44	**15.07**	9.80	12.92	0.9868	0.9556	0.9863	0.9392
[9]	12.12	19.64	10.88	14.77	0.9825	0.9418	0.9826	0.9299
[5]	19.84	23.67	20.68	23.90	0.8791	0.8548	0.8628	0.8403
METHOD	PERCENTILE 50							
PACO	**10.48**	17.32	**10.05**	**14.09**	**0.9906**	**0.9615**	**0.9900**	**0.9567**
[7]	12.46	**16.31**	12.08	15.94	0.9886	0.9607	0.9888	0.9508
[9]	14.75	22.48	13.91	17.96	0.9856	0.9463	0.9846	0.9389
[5]	24.15	29.25	26.21	29.23	0.9103	0.8854	0.9084	0.8777
METHOD	PERCENTILE 75							
PACO	**13.43**	**20.73**	**15.20**	**19.66**	**0.9931**	0.9646	**0.9918**	**0.9618**
[7]	15.95	21.20	17.65	20.90	0.9914	**0.9660**	0.9910	0.9571
[9]	18.92	27.20	20.87	24.72	0.9887	0.9494	0.9886	0.9499
[5]	31.05	37.91	32.12	36.17	0.9414	0.9131	0.9378	0.9087

Table 1: Summary of inpainting results on the whole Kodak dataset (luminance channel) in terms of RMSE (smaller is better) and SSIM (higher is better); best results are in bold blue.

at $s = m/2$, that is, 50% overlap, while being significantly faster to compute (about an order of magnitude). However, there is no single best choice for the patch size: higher patch sizes lead to better SSIM scores but worse RMSE, whereas smaller patches have the opposite effect. This being said, in our opinion, it is better to rely on SSIM for real world applications.

6.3 Performance Comparison

We compare the results of PACO-DCT with those of [5, 7, 9]. Both [7, 9] are recent, state-of-the-art methods based on the non-local patch-based estimations paradigm which produce high quality results on large holes at a significant computational cost. On the other hand, [5] is a very fast, non iterative method from the family of signal restoration based on partial differential equations which is suitable for small regions over piecewise smooth signals; this should give us a comparison on the other extreme of the quality-vs-speed spectrum.

According to the preceding section, we use the PACO-DCT problem parameters that yield the best compromise between RMSE and SSIM: $w = 16$ ($m = 256$) and $s = 2$. The optimization parameters are set to $\lambda = 10$, $\kappa = 0.95$ a maximum of 1024 iterations and a minimum allowable cost function decrease of $\epsilon = 1e^{-5}$. When comparing to other algorithms, we set the parameters of such algorithms to the defaults used in their publicly available implementations.

Table 1 summarizes the results of the four methods for the 24 Kodak images in terms of the median, 25 and 75 percentiles of the RMSE and SSIM measures, again computed *only* over the missing pixel locations. For simplicity, we only show the results on the Luminance channel of these images; similar results are obtained for the other channels. Figure 6 shows the results obtained with these four methods for a particular case, again on the Luminance channel. Figure 7 shows the color result obtained on Kodak image #19 and mask #2.

Figure 6: Visual comparison of the four methods on Kodak #19 and mask #2. Top to bottom, left to right: PACO, [7], [9], [5]. All four methods perform well on narrow gaps. For wider gaps, the results of [7] and PACO-DCT are similar, the major difference being the fence next to the life saver; [7] produces sharper results than PACO, but also more artifacts (e.g., large gap in fence or above the lighthouse window); [9] and [5] produce significant artifacts in these regions (we recall the reader that [5] is not suited for such large gaps by design.)

Figure 7: Color image inpainting: sample result for Kodak image #19 and mask #2. Left: overlay of original image and missing pixels mask. Right: Result of applying PACO-DCT with patches of size 64×64 and a stride of 4 pixels (please zoom in on digital document for details). According to Figure 5. Overall, the restored image is visually similar to the original in most regions; the metric SSIM $=0.9678$ is consistent with this. On the other hand, the result is not as good on regions which exceed the patch size by a large margin; this explains the relatively high RMSE $= 19.85$.

Despite the simplicity of the DCT prior, PACO-DCT compares favorably to the other methods in most cases. Although Table 1 should not be considered a thorough comparative study of image inpainting methods, which is clearly outside of the scope of this paper, it provides encouraging evidence on the competitiveness of the proposed PACO-DCT inpainting method.

7 Concluding Remarks

We have presented the PACO-DCT inpainting algorithm: a simple and efficient method based on the PACO framework which produces good results in a number of inpainting scenarios which are of practical interest, for example the removal of small and medium-sized scratches and erasures in pictures and film frames. The method is not well suited for very large inpainting areas; it remains to be seen whether this problem can be alleviated by using more complex formulations (such as multi-scale) or other (convex or non-convex) patch priors.

Image Credits

All the images used in the experiments come from the Kodak image dataset[5].

References

[1] C. Aguerrebere, A. Almansa, J. Delon, Y. Gousseau, and P. Musé, *A Bayesian hyperprior approach for joint image denoising and interpolation, with an application to HDR imaging*, IEEE Transactions on Computational Imaging, 3 (2017).

[2] M. Aharon, M. Elad, and A. Bruckstein, *The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representations*, IEEE Transactions on Signal Processing, 54 (2006), pp. 4311–4322. https://doi.org/10.1109/TSP.2006.881199.

[3] P. Arias, G. Facciolo, V. Caselles, and G. Sapiro, *A variational framework for exemplar-based image inpainting*, International Journal on Computer Vision, 93 (2011), pp. 319–347. http://dx.doi.org/10.1007/s11263-010-0418-7.

[4] D. Batenkov, Y. Romano, and M. Elad, *On the global-local dichotomy in sparsity modeling*, in Compressed Sensing and its Applications: 2nd. International MATHEON Conference, Cham, 2015, pp. 1–53. https://doi.org/10.1007/978-3-319-69802-1_1.

[5] F. Bornemann and T. März, *Fast image inpainting based on coherence transport*, Journal of Mathematical Imaging and Vision, (2007), pp. 259–278. https://doi.org/10.1007/s10851-007-0017-6.

[6] J. Eckstein and D.P. Bertsekas, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Mathematical Programming, 55 (1992), pp. 293–318. https://doi.org/10.1007/BF01581204.

[7] V. Fedorov, G. Facciolo, and P. Arias, *Variational Framework for Non-Local Inpainting*, Image Processing On Line, 5 (2015), pp. 362–386. https://doi.org/10.5201/ipol.2015.136.

[5] http://www.cs.albany.edu/~xypan/research/snr/Kodak.html

[8] M. A. T. Figueiredo, *Synthesis versus analysis in patch-based image priors*, in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), March 2017, pp. 1338–1342. https://doi.org/10.1109/ICASSP.2017.7952374.

[9] A. Newson, A. Almansa, Y. Gousseau, and P. Pérez, *Non-Local Patch-Based Image Inpainting*, Image Processing On Line, 7 (2017), pp. 373–385. https://doi.org/10.5201/ipol.2017.189.

[10] V. Papyan and M. Elad, *Multi-scale patch-based image restoration*, IEEE Transactions on Image Processing, 25 (2016), pp. 249–261. https://doi.org/10.1109/TIP.2015.2499698.

[11] N. Parikh and S. P. Boyd, *Proximal algorithms*, Foundations and Trends in Optimization, 1 (2014), pp. 127–239. https://doi.org/10.1561/2400000003.

[12] I. Ramírez Paulino, *PACO: Global Signal Restoration via PAtch COnsensus*, ArXiv e-prints, (2018). https://arxiv.org/abs/1808.06942.

[13] G. Peyré, *Manifold models for signals and images*, Computer Vision and Image Understanding, 113 (2009), pp. 249–260. https://doi.org/10.1016/j.cviu.2008.09.003.

[14] I. Ramirez and G. Sapiro, *Universal regularizers for robust sparse coding and modeling*, IEEE Transactions on Image Processing, 21 (2012), pp. 3850–3864. https://doi.org/10.1109/TIP.2012.2197006.

[15] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, *Image quality assessment: from error measurement to structural similarity*, IEEE Transactions on Image Processing, 13 (2004), pp. 600–612. https://doi.org/10.1109/TIP.2003.819861.

[16] G. Yu, G. Sapiro, and S. Mallat, *Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity*, IEEE Transactions on Image Processing, 21 (2012), pp. 2481–2499. https://doi.org/10.1109/TIP.2011.2176743.

[17] D. Zoran and Y. Weiss, *From learning models of natural image patches to whole image restoration*, in International Conference on Computer Vision (ICCV), Nov 2011, pp. 479–486. https://doi.org/10.1109/ICCV.2011.6126278.