IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2015–06–27.
Submitted on 2012–10–31, accepted on 2014–11–25.
ISSN 2105–1232 © 2015 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.2015.49

Obtaining High Quality Photographs of Paintings by Image Fusion

Antoni Buades[1], Gloria Haro[2], Enric Meinhardt-Llopis[3]

[1] CMLA, ENS Cachan, France (toni.buades@cmla.ens-cachan.fr)
[2] UPF, Barcelona, Spain (gloria.haro@upf.edu)
[3] CMLA, ENS Cachan, France (enric.meinhardt@cmla.ens-cachan.fr)

Communicated by Jean-Michel Morel and Julie Delon *Demo edited by* Enric Meinhardt-Llopis

This IPOL article is related to a companion publication in the SIAM Journal on Imaging Sciences:
G. Haro, A. Buades, and J.M. Morel, "Photographing paintings by image fusion", *SIAM Journal on Imaging Sciences, 5 (2012), pp. 1055-1087.*
http://dx.doi.org/10.1137/120873923

Abstract

You walk through a museum taking photographs of some paintings with your commodity camera. You load all these images into the computer. The computer builds a high-quality image of each one of the paintings. This article describes what the computer does. More precisely, we explain a method to produce a single, high-resolution, clean and well-lit image, out of many low-resolution noisy images taken in bad lighting conditions.

Source Code

The source code (ANSI C), its documentation, and the online demo are accessible at the IPOL web page of this article[1].

Keywords: registration; denoising; reconstruction; image fusion; image blending; burst denoising; highlight removal; glare removal; reflection removal; shadow removal

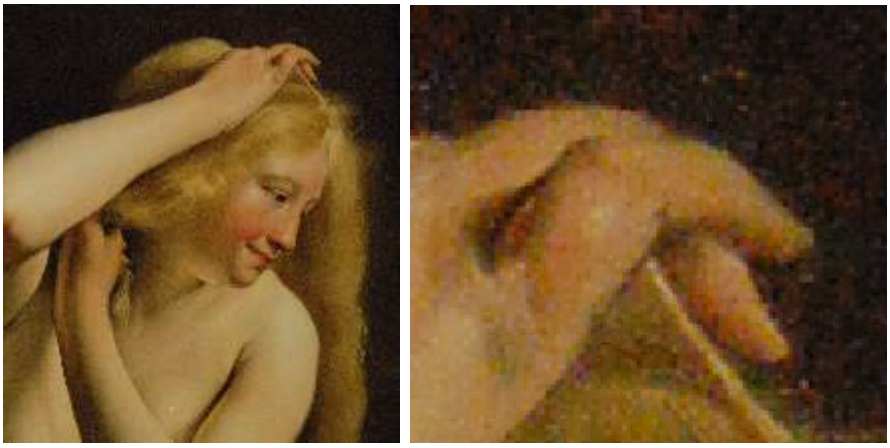


Figure 1: Noisy image taken on a museum (left: whole image, right: detail).

Figure 2: Two photographs of the same painting showing reflections at different places.

Figure 3: An image and a detail. For such large paintings, a single high-resolution photograph has simply not enough resolution to represent all the detail.

1 Introduction

When taking a single photograph of a painting, three problems may occur. The first problem is noise: since museums typically use subdued lighting, the photographs have to be taken at high sensitivity and they are noisy (see Figure 1). The second problem is reflections: since there are non-diffuse light sources, and the surface of the canvas is too bright, glossy, varnished or even covered by glass, highlights and reflections may appear (see Figure 2). The third problem is limited resolution: if the whole painting has to fit in a single photograph, the resolution of the camera may be simply not enough to show the small details of the painting (see Figure 3).

The present article, based on [5], addresses these three problems using an appropriate combination of classical image processing tools. In each case, the problem is solved by combining the information from several images into one. The problem of noise is solved by *burst denoising*, whereby several photographs from the same viewpoint are averaged in order to reduce noise. The problem of reflections is solved by *highlight removal*, whereby several photographs taken from different viewpoints are combined using a robust average, in order to remove the highlights which appear at different places among the images. The problem of resolution is solved by *detail pasting*, by means of which photographs of parts of the picture are seamlessly pasted into a reference image of the whole picture, producing a higher resolution version of it. The present article focuses on burst denoising and highlight removal, leaving the delicate analysis of detail pasting to a forthcoming article.

In Section 2 we give an informal overview of the whole reconstruction pipeline. In Section 3 we point to the basic tools from classical image processing that are needed. In Sections 4 and 5 we describe respectively the complete algorithms for burst denoising and highlight removal. In Section 6 we describe the algorithm for detecting bursts in a sequence of images.

2 Overview of the Whole Pipeline

This article describes a method for constructing a single image of a painting from many photographs of that painting. Thus, the input is a set of images and the output is a single image. The goal is that this combination removes several defects of the individual images.

For simplicity, we assume that all the images are photographs of the same painting and that each subset of images taken from the same point of view (henceforth called *burst*) is consecutive. We also assume that the first photograph is a frontal view of the whole painting, which may have been manually cropped. The pipeline is described in Algorithm 1.

Algorithm 1: scheme of the whole pipeline (see Figure 4)

Input : A list of images $I_1, \cdots, I_n$, where $I_i : \Omega \subseteq \mathbf{Z}^2 \to \mathbf{R}^3$, $i = 1, \ldots, n$, and Ω a rectangular image domain.

Output: An image $I_* : \Omega \to \mathbf{R}^3$

1. Apply `BurstSegmentation` to the list of images, partitioning it into bursts.
2. Apply `BurstDenoising` to each burst, obtaining one denoised image per burst
3. Apply `HighlightRemoval` to the set of views of each detail, obtaining an image of each detail without reflections
4. Apply `DetailPasting` to these images, obtaining I_* (not described in this article)

[1] https://doi.org/10.5201/ipol.2015.49

A common particular case of this algorithm arises when all the input images are photographs of the whole painting. In that case, there is no image corresponding to a subpart of the painting, so we can consider that there is only one "detail", the whole image itself. This particular case is suitable when the painting has small dimensions, so that a single shot can have enough resolution. Another particular case arises when all the bursts consist of one single photograph. This case is suitable when the scene is bright enough so that no denoising is required.

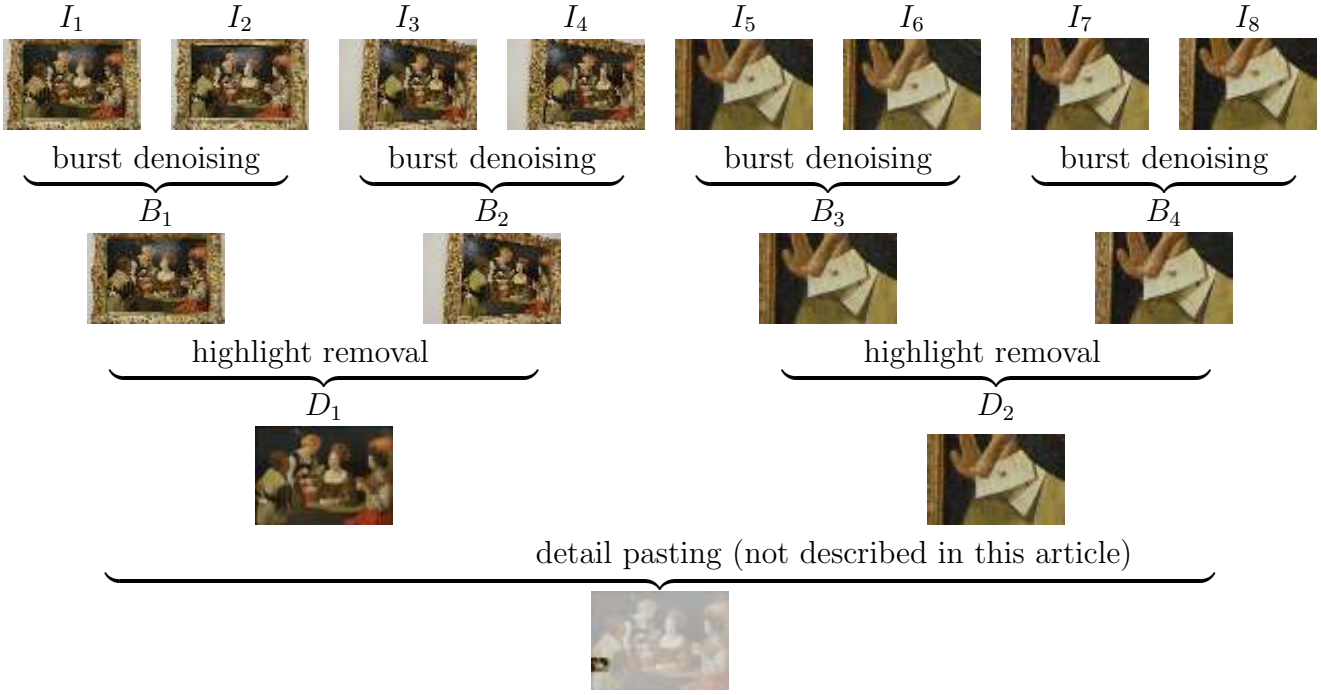


Figure 4: Schematic overview of the whole pipeline. The input sequence is partitioned into bursts. The images from each burst are combined into a single denoised image. These denoised images are partitioned into "details". All the views from each detail are combined into a single image, without highlights. These images may be finally pasted into a single reference frame.

3 Basic Tools

The proposed method is a high-level image processing algorithm. As such, it uses several lower-level methods as its building blocks. We consider these methods as black boxes, and we explain below their input, output and purpose. Here we will not detail their inner workings, instead we refer to other publications where they are explained in detail. However, an optimized implementation of the proposed method may need to break inside these building blocks and fine-tune them to the needs of the larger algorithm.

3.1 SIFT Homographies

We use SIFT [7] as a method for finding a homography that matches two images. This is an appropriate tool for registering different photographs of the same planar object because, under the pinhole camera model, these images differ exactly by a homography. Notice that for non-planar objects, homographies are still a correct model if the camera motion is a rotation around the center. In practice, when taking a burst of images from a fixed position, the camera rotation is a valid approximation

of the real movement; thus there is a registering homography even for three-dimensional scenes (as assessed by the fact that we can denoise photographs of sculptures by using this method as shown in Figure 12).

In general, SIFT is a method for extracting keypoints from an image and assigning robust descriptors to them. These descriptors can be matched to those of another image in order to produce a list of pairs of points. This list of matching points can be used to estimate the parameters of a geometric transformation between the two images. Typically, a robust estimation method such as RANSAC [3] is used, which automatically rejects wrong matches before computing the model.

For our purposes, we use this combination of SIFT and RANSAC to register a pair of images by a homography (see Algorithm 2). Thus, from the point of view of this article, SIFT is a method that takes two images and produces as output a homography that registers them. Optionally, it gives the list of matching point pairs that are consistent with the estimated homography. In Figure 5 we show the steps and the result of this registration algorithm.

Algorithm 2: `SIFT` (find a homography by SIFT+RANSAC)

Input : Image $A : \Omega_A \to \mathbf{R}$
Input : Image $B : \Omega_B \to \mathbf{R}$
Input : Parameter m = maximum allowed distance
Input : Parameter ϵ = maximum allowed error
Output: A boolean value `match-found`
Output: A homography $H : \mathbf{R}^2 \to \mathbf{R}^2$, represented by a 3×3 matrix
Output: A list of n pairs of points $(a_i, b_i) \in \Omega_A \times \Omega_B$, for $i = 1, \ldots, n$

1 Compute the SIFT matches between A and B (see [7])
2 Run RANSAC to find the best homography among these matches (see [3])

When `match-found` is true (see Algorithm 2), the output of the SIFT+RANSAC algorithm has the following properties:

1. $|H(a_i) - b_i| < \epsilon$, so that if ϵ is small we can expect images $A \circ H$ and B to be well registered (if the keypoints were representative at all)

2. $|a_i - b_i| < m$, so that if m is small we can expect H to be close to the identity, meaning that the images were already almost registered.

Reasonable values of these parameters are $\epsilon = 1$, meaning that we allow an error of one pixel in the position of the keypoints, and m equal to one tenth of the diameter of the image, meaning that H is a small deformation. Another meaningful value is $m = \infty$, which allows for arbitrarily large deformations.

This `SIFT` algorithm is used by the algorithms `BurstSegmentation` (Section 6), `BurstDenoising` (Section 4) and `HighlightRemoval` (Section 5).

3.2 ASIFT Homographies

The SIFT+RANSAC method works well when the desired transformation is locally isotropic. However, it breaks down when there is a large tilt between the two images. This is due to the fact that the SIFT descriptors are scale-invariant but not invariant under arbitrary affine transformations (e.g., having a different zoom factor in different directions).

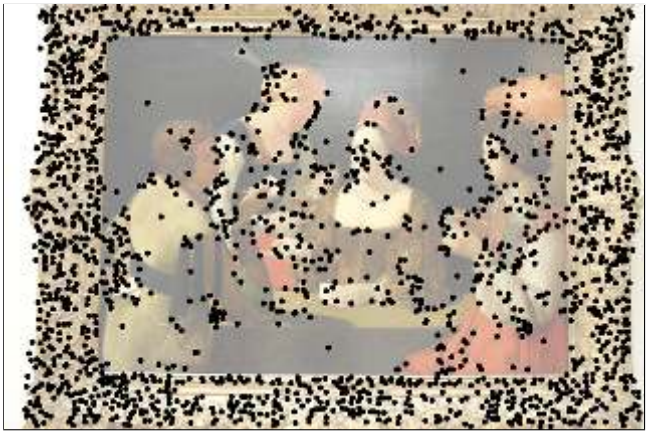


Image A keypoints (2291)

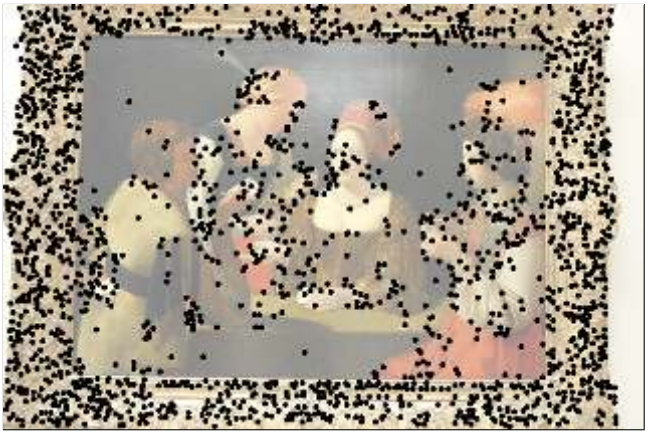
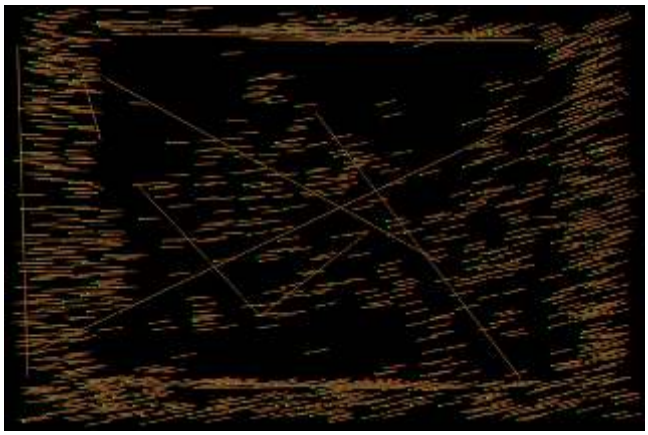
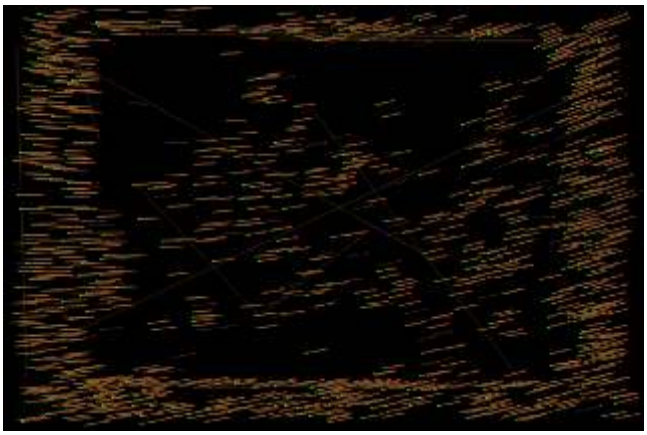


Image B keypoints (2324)



SIFT matches (1357)



RANSAC inliers (1203)

difference $B - A$

registered difference $B - A \circ H$

Figure 5: Registration of two images by SIFT+RANSAC. The SIFT matches are run through RANSAC to fit a homographic model H. The homography H registers both images, as can be seen from the fact that the registered difference contains mostly noise.

The ASIFT Algorithm [8] is built upon SIFT and is able to find homographies with a much larger tilt. Internally, it builds an orbit of affinely deformed versions of each image, before running SIFT on each resulting pair of images. By careful sampling on the space of such deformations, it runs efficiently. See Figure 6 for an example of two images related by a tilt of 15.

The signature of the algorithm is the same as for SIFT, so we do not repeat it here. The only difference is in the running time (which is about 10 times slower), and the larger tilt factors which is able to recover (up to 40, according to [8]). Also, in the cases where SIFT was able to find the correct homography, ASIFT finds many more matching keypoints. On the other hand, the precision of the computed homographies is sligthly worse, because the inherent inaccuracy in the location of the keypoints is compound with a larger deformation. Thus, it is common to *refine* a registration produced by ASIFT by post-processing the results using regular SIFT.

The `ASIFT` algorithm is used by the Algorithm `HighlightRemoval` (Section 5). An online implementation of the ASIFT algorithm is separately available on IPOL [10].

Figure 6: ASIFT is able to detect homographies with a huge tilt, as in this image. Here, SIFT does not find any match.

3.3 Irregular Sampling using Splines

Deforming the image domain by a homography, or by another smooth transformation, is a particular case of *irregular sampling* [2]. In general this is a difficult problem. However, for the purpose of this article we assume that both images have roughly the same size and that the deformation is not strongly non-isotropic. Then the sampling can be performed naively by a linear interpolating scheme such as splines (Algorithm 3).

Algorithm 3: irregular sampling using splines

Input : Image $I : \mathbf{Z}^2 \to \mathbf{R}$
Input : Smooth function $F : \mathbf{R}^2 \to \mathbf{R}^2$
Input : Parameter $n \in \{1, 2, 3, -3, 5, 7, 11\}$ order of the spline
Output: Image $J : \mathbf{Z}^2 \to \mathbf{R}$, approximating $I \circ F$
1 Compute the sampling locations $(x, y) = F(i, j) \quad \forall (i, j) \in \mathbf{Z}^2$
2 Evaluate $I(x, y)$ using a spline of order n

The meaning of the parameters, and the method itself, is explained in detail elsewhere (see [4] and MegaWave[2] documentation). Our preferred choices are $n = -3$ (bicubic interpolation) for speed and $n = 7$ for a good compromise between accuracy and ringing artifacts.

[2]MegaWave software. http://megawave.cmla.ens-cachan.fr/

We use this algorithm for deforming an image by a homography, as in Figure 7. This is needed by the `BurstDenoising` algorithm (Section 4, line 4 of the algorithm); and by the `HighlightRemoval` algorithm (Section 5, lines 4 and 8 of the algorithm).

I

$$\begin{pmatrix} 0.803226 & -0.0337366 & 18.7011 \\ 0.0955645 & 0.673925 & 18.7011 \\ 0.000645791 & -0.000855805 & 1.005381 \end{pmatrix}$$

H

$I \circ H$

Figure 7: Irregular sampling allows us to deform the image domain by an arbitrary homography.

3.4 Midway Histogram Equalization

The cumulative normalized histogram of an image A is an increasing function h_A whose values go from 0 to 1 as its argument goes from the minimum to the maximum value of A. Thus, the image $h_A \circ A$ has values on the interval $[0, 1]$, and its histogram is uniform. We can change the contrast of an image A so that it has a prescribed cumulative histogram h by the operation $h^{-1} \circ h_A \circ A$, where h^{-1} is a pseudo-inverse of the function h.

A nice way to define a target histogram h computed from a collection of images is given by the midway histogram algorithm [1] (Algorithm 4). For color images, we apply the midway histogram algorithm channel by channel.

Algorithm 4: `MidwayHistogram`

Input : Images $I_1, \ldots, I_n$, where $I_i : \Omega \to \mathbf{R}$, $I = 1, \ldots, n$.
Output: A cumulative histogram h

1 **for** $i = 1, \ldots, n$ **do**
2 $\quad h_i \leftarrow$ cumulative histogram of image I_i
3 $f(t) \leftarrow \dfrac{1}{n} \displaystyle\sum_{i=1}^{n} h_i^{-1}(t) \quad t \in [0, 1]$
4 $h \leftarrow f^{-1}$

Notice that the actual computations performed on lines 3 and 4, written here in symbolic form, require a careful sampling of the involved functions. See [1] for details.

Algorithm `MidwayHistogram` is used by Algorithm `HighlightRemoval`, described in Section 5.

3.5 Poisson Editing

Poisson's equation

$$\Delta u = f$$

can be used to recover an image I knowing its gradient $\mathbf{F}$. Namely, we need to solve Poisson's equation with $u = I$ and $f = \operatorname{div}(\mathbf{F})$. Without any boundary conditions, there is a family of solutions differing by addition of an arbitrary harmonic function. If we impose that the derivative of this harmonic function must vanish at the boundary of the image, the family of solutions is one-dimensional, parametrized by an additive constant.

Poisson's equation on the whole plane can be solved using the 2D Fourier transform. If we denote the 2D Fourier transform of a function $f(x, y)$ by $\hat{f}(\xi, \eta)$, we have that $\hat{\Delta u} = -(\xi^2 + \eta^2)\hat{u}$. Thus, the solution of the Poisson equation above can be computed as the inverse Fourier transform of the following function

$$\hat{u} = \frac{-1}{\xi^2 + \eta^2}\hat{f}.$$

Notice that $\hat{u}$ is undefined at $(\xi, \eta) = 0$, so that the mean value of the function is lost. In writing this equation, we use the convention that $1/0 = 0$.

For a discrete and rectangular domain, we can use the solution above by periodizing the data to the whole plane. This corresponds to Poisson's equation on the rectangle with vanishing Neumann boundary conditions. For the implementation details see, for example, [6]. In the present article we use Poisson equation as a procedure for recovering an image from its gradient and its average value. This method is also the basis of Poisson Editing [9], a technique for seamlessly pasting images. See Figure 8 for the visual effect of the Δ operator and its inverse, and Figure 9 for the recovery of an image from its gradient.

Algorithm 5: `PoissonEquation`

Input : Vector Field $\mathbf{G} : \Omega \to \mathbf{R}^2$, the desired gradient
Input : Number m, the desired mean value
Output: Image $I : \Omega \to \mathbf{R}$

1 $g \leftarrow \operatorname{div}(\mathbf{G})$ using backward differences
2 $f \leftarrow \texttt{FFT}(g)$
3 **for** $(k, l) \in \Omega$ **do**
4 $\quad f(k, l) \leftarrow \frac{-1}{k^2+l^2} f(k, l)$
5 $I_0 \leftarrow \texttt{IFFT}(f)$
6 $I \leftarrow I_0 + m$

Algorithm `PoissonEquation` (Algorithm 5) is used at the end of Algorithm `HighlightRemoval`, described in Section 5.

3.6 Inverse Heat Equation

A common operation in image processing is *sharpening*, by which blur is removed from an image. In general, this is an ill-posed problem and its solutions are unstable. The worst case happens when the blurry images are also affected by noise. For the purposes of this article, we need sharpening to enhance the slightly blurry images obtained after the whole pipeline. Since these reconstructed images are already denoised, it is relatively safe to try to sharpen them, for a great visual improvement.

We are content with the simplest form of sharpening, given by the inverse heat equation

$$\frac{\partial}{\partial t} I = -\Delta I.$$

Using the blurry image I as initial condition for this equation, we obtain a sequence of iteratively sharpened versions of it by a finite difference scheme. A few iterations are computed until artifacts start to appear (the artifacts must be detected visually). Some iterations of this algorithm are illustrated in Figure 10. In our program, by default we perform 3 iterations which is a safe value.

Algorithm 6: `Sharpening`

Input : Image $\mathbf{I} : \Omega \to \mathbf{R}$
Output: Sequence $I^1, I^2, I^3, \ldots$ of sharpened versions of I

$t \leftarrow 0.1$
$I^0 \leftarrow I$
for $n = 0, 1, 2, \ldots$ until *artifacts appear* **do**
  for $(x, y) \in \Omega$ **do**
    $L(x, y) \leftarrow I^n(x+1, y) + I^n(x-1, y) + I^n(x, y+1) + I^n(x, y-1) - 4I^n(x, y)$
  $I^{n+1} \leftarrow I^n - tL$

The `Sharpening` algorithm (Algorithm 6) is used as the last step of algorithms `BurstDenoising` (Section 4) and `HighlightRemoval` (Section 5).

I

ΔI

$\Delta^{-1} I$

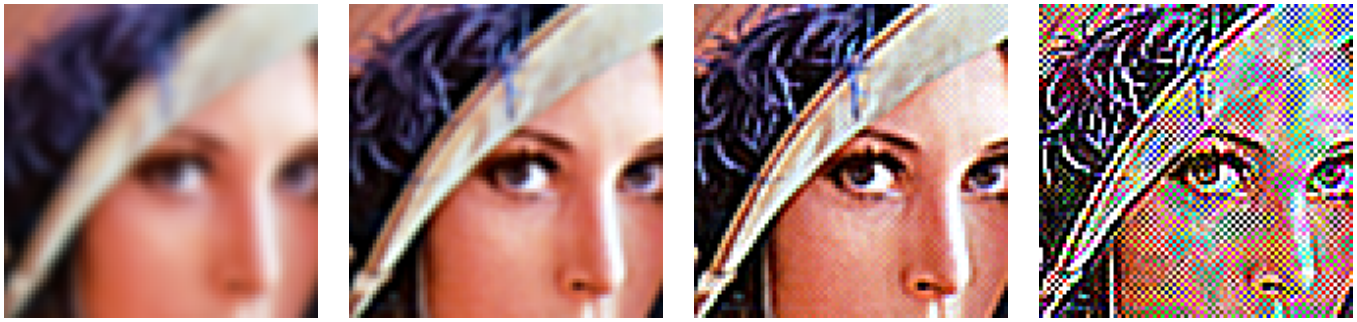
Figure 8: The Laplacian and anti-Laplacian of the same image. Notice that the anti-Laplacian is a very smooth image. The ranges of these three images are respectively $[0, 255]$, $[-5, 5]$ and $[-127, 127]$. For display purposes, these ranges have been re-scaled to fill the interval $[0, 255]$.

4 Burst Denoising

The algorithm for burst denoising combines several images taken from the same viewpoint into a single, cleaner image (see Algorithm 7). This algorithm assumes that all the images on the list can be registered by SIFT to the first one. This assumption is automatically true when this algorithm is run inside the whole pipeline. Since we don't expect large illumination changes, the registered images can be averaged directly. However, some of the images may be blurry due to camera shake. Thus, this average is locally weighted by a local measure of image quality. Finally, this average is

$f = \partial_x I$ $g = \partial_y I$ $\Delta^{-1}(\partial_x f + \partial_y g)$

Figure 9: The Poisson editing method allows to recover an image (up to an additive constant), from its gradient.



I I^3 I^5 I^7

Figure 10: Visual effects of the sharpening algorithm on a blurry image I (detail). At 3 iterations the blur is mostly removed. At 5 iterations the image acquires an artificial appearance. At 7 iterations, the image is completely destroyed by high-frequency artifacts.

sharpened to reduce the blur due to minor inaccuracies in the registration. Figures 11 and 12 show some results of the burst denoising algorithm.

The local weights are defined by the following formula (see [5])

$$\texttt{LocalWeight}(\texttt{I},\ \mathbf{x}) := \int_{R(\mathbf{x})} |\nabla \bar{I}(\mathbf{x}+\mathbf{y})| \mathrm{d}\mathbf{y},$$

where $R(\mathbf{x})$ denotes a square neighborhood of side 100 pixels centered around $\mathbf{x}$, and $\bar{I}$ is the luminance component of the image I.

Notice that, for simplicity, we have written the algorithm so that the first image is registered to itself. In practice, this computation is omitted.

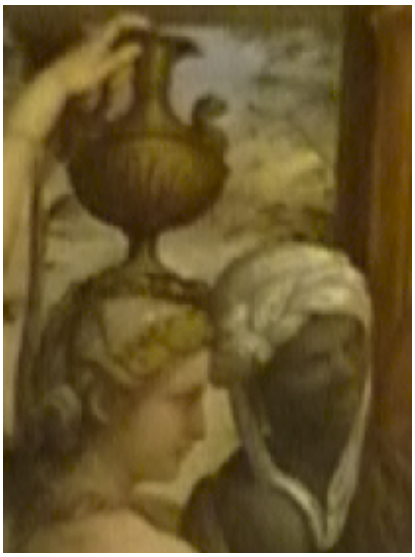
The `BurstDenoising` algorithm is used at step 2 of Algorithm 1 (the whole reconstruction pipeline).

Algorithm 7: `BurstDenoising`

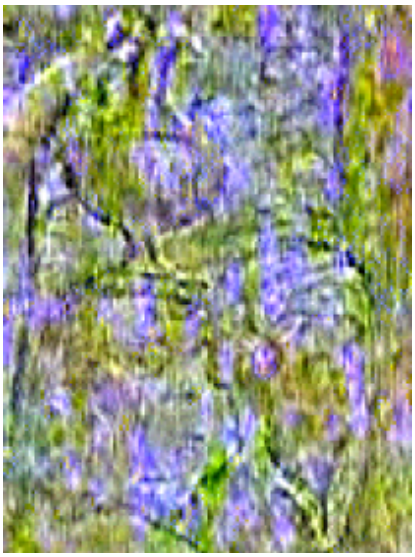
Input : Images $I_1, \dots, I_n$
Input : Parameter $t \geq 0$ (sharpening strength)
Output: Image I_*

1 **for** $i = 1, \dots n$ **do** `// find homographies`
2 $\quad H_i \leftarrow \texttt{SIFT}(I_1, I_i)$
3 **for** $i = 1, \dots n$ **do** `// register all images to the first one`
4 $\quad I'_i \leftarrow I_i \circ H_i^{-1}$
5 **for** $i = 1, \dots n$ **do** `// compute the weight images`
6 $\quad D_i(\mathbf{x}) \leftarrow \texttt{LocalWeight}(I'_i, \mathbf{x})$
7 $I_0 \leftarrow (\sum_{i=1}^n D_i I'_i) \,/\, (\sum_{i=1}^n D_i)$ `// weighted linear combination`
8 $I_* \leftarrow \texttt{Sharpening}(I_0)$ `// sharpen the result`

Image I_1 from a burst of 10 (detail)



Denoised image I



Difference $I - I_1$

Figure 11: Example of burst denoising. These images show details of larger images of size 1920×1080. Notice that, besides being denoised, the reconstructed image is sharper. This is due to the weights used in the denoising, which tend to reject images in the burst which are blurry. The difference is shown here within the range $[-20, 20]$.

5 Highlight Removal

The algorithm for highlight removal is similar to the one used for burst denoising: first register the images and then combine them. However, the goals and the constraints are different. Namely, here we expect larger deformations and global and local illumination changes. Thus the registration is performed by ASIFT instead of SIFT, and the histograms are normalized before combining the images. Finally, the combination is not a linear average but a gradient median, which is much more robust to some images having extremely different colors due to reflections and highlights.

The method is described in Algorithm 8. Lines 1–8 compute a set of registered images. Lines 9–12 normalize the histograms of the registered images Then, on lines 13–18 the registered and color-

Image I_1 from a burst of 10 (detail)

Denoised image I

Difference $I - I_1$

Figure 12: Example of burst denoising. These images show details of larger images of size 1080×1920 of a three-dimensional sculpture. The third image is shown within the range $[-10, 10]$.

Algorithm 8: `HighlightRemoval`

Input : Images $I_1, \dots, I_n$
Input : Parameter $t \geq 0$ (sharpening strength)
Output: Image I_*

for $i = 1, \dots n$ **do** `// find homographies by ASIFT`
  $H_i \leftarrow \texttt{ASIFT}(I_1, I_i)$
for $i = 1, \dots n$ **do** `// register all images to the first one`
  $I'_i \leftarrow I_i \circ H_i^{-1}$
for $i = 1, \dots n$ **do** `// refine the homographies by SIFT`
  $F_i \leftarrow \texttt{SIFT}(I_1, I'_i)$
for $i = 1, \dots n$ **do** `// resample using the combined transformation`
  $I''_i \leftarrow I_i \circ H_i^{-1} \circ F_i^{-1}$
$h \leftarrow \texttt{MidwayHistogram}(I''_1, \dots, I''_n)$ `// compute the target histogram`
for $i = 1, \dots n$ **do** `// impose this histogram to all images`
  $h_i \leftarrow$ cumulative histogram of image $I_{i''}$
  $J_i \leftarrow h^{-1} \circ h_i \circ I''_i$
Compute the gradients (using forward differences) of the original color images ∇J_i
Compute the gradients (using forward differences) of the luminance components $\nabla \bar{J}_i$.
for *each pixel* $\mathbf{x}$ **do** `// vector medians`
  $k \leftarrow \arg\min_{i=1,\dots,n} \sum_{j \neq i} \|\nabla \bar{J}_i(\mathbf{x}) - \nabla \bar{J}_j(\mathbf{x})\|$
  $\mathbf{M}(\mathbf{x}) \leftarrow \nabla J_k(\mathbf{x})$
$I_0 \leftarrow \texttt{PoissonEquation}(\mathbf{M}, \text{mean}(I_1))$ `// combine the resulting images`
$I_* \leftarrow \texttt{Sharpening}(I_0)$ `// sharpen the result`

normalized images are combined into a single image I_0. This combination is done by computing the vector median of the gradients, and then reconstructing the image from this target gradient by solving a Poisson equation.

As an implementation detail, notice that the ASIFT registrations may be computed over small versions of the images. Since this registration will be refined later, the loss of precision due to the subsampling is not a problem. This results in a significant reduction of the overall computational burden.

See figures 13 to 15 for an example of this algorithm. As a curiosity, notice that since this algorithm removes highlights, the highlights themselves can be recovered by substracting the reconstructed image from the original. Typically, these images contain the lights in the museum, the other pictures, shadows of visitors, and even the reflection of the photographer who takes the pictures. See Figure 16 for an example of this highlight recovery.

The `HighlightRemoval` algorithm is used at step 3 of Algorithm 1 (the whole reconstruction pipeline).

B_1 B_2 B_3 B_4 B_5

Figure 13: Example of highlight removal: list of input images (5 out of 10 are shown).

B_1 $B_2 \circ H_2$ $B_3 \circ H_3$ $B_4 \circ H_4$ $B_5 \circ H_5$

Figure 14: Example of highlight removal: registered images. Notice that the highlights are at different positions on each registered image.

average(B_i) min(B_i) max(B_i) med(B_i) $B_* = \Delta^{-1}\left[\mathrm{div}(\mathrm{med}(\nabla B_i))\right]$

Figure 15: Example of highlight removal: different combinations of the registered images. The last one is the one proposed by this article. A detailed comparison of all these methods can be found in [5].

$B_1 - B_*$ $B_2 \circ H_2 - B_*$ $B_3 \circ H_3 - B_*$ $B_4 \circ H_4 - B_*$ $B_5 \circ B_5 - B_*$

Figure 16: Visualization of highlights. Once we have recovered the image without highlights, we can subtract this image from each of the input images, thus recovering the highlights themselves. In this case, the highlights contain reflections of other objects in the room where the photo was taken, including the photographer herself. All these images are shown in the range $[-50, 50]$.

6 Burst Segmentation

Burst segmentation is the algorithm that takes a list of consecutive photographs and partitions it into a collection of sub-lists, called *bursts*. The idea is that all the images of each burst are photographs taken from the same viewpoint. The bursts are detected by registering the images consecutively by SIFT. When no match is found, or the resulting homography is a large deformation, this signals the beginning of a new burst. The method is described in Algorithm 9.

Algorithm 9: `BurstSegmentation`

Input : A list of images $I_1, \dots, I_n$, where $I_i : \Omega \to \mathbf{R}$, $i = 1, \dots, n$.
Output: An increasing list of indices $b_1, \dots, b_k$

$k \leftarrow 1$
$b_k \leftarrow 1$
for $i = 2, \dots, n$ **do**
    $H = \texttt{SIFT}(I_{b_k}, I_i)$
    if $H = \emptyset$ **or** `IsLargeDeformation`(H) **then**
        $k \leftarrow k + 1$
        $b_k \leftarrow i$

At the end of this algorithm, the indices b_i point to the first image of each burst. See Figure 17 for an example.

The test $H = \emptyset$ means that SIFT found no match. The test `IsLargeDeformation(H)` combines several measures over the 3×3 matrix H:

1. Each of the four corners of the image must be displaced by H a distance less than 10% of the total size.
2. The tilt factor of H must be less than 1.03.
3. $H(3,1) < 10^{-4}$
4. $H(3,2) < 10^{-4}$

If either of these conditions fails, then H is considered a large deformation and a new burst is started.

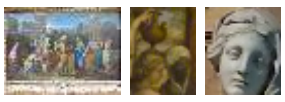
The `BurstSegmentation` algorithm is used at step 1 of Algorithm 1 (the whole reconstruction pipeline).

Acknowledgements

Work partially supported by the European Research Council (Advanced Grant Twelve Labours).

Image Credits


Photographs by Gloria Haro


Photographs by Jean-Michel Morel


Photographs by Guoshen Yu

Figure 17: Example of burst segmentation in a real-life setting. The photographer took 32 images from four different points of view. The output of the algorithm is the set of indices $\{1, 11, 18, 24\}$, correctly identifying the beginning of each burst.

References

[1] J. Delon, *Midway image equalization*, Journal of Mathematical Imaging and Vision, 21 (2004), pp. 119–134. http://dx.doi.org/10.1023/B:JMIV.0000035178.72139.2d.

[2] H.G. Feichtinger and K. Gröchenig, *Wavelets: mathematics and applications*, CRC Press, 1994, ch. Theory and practice of irregular sampling, pp. 305–363. ISBN 978-0849382710.

[3] M.A. Fischler and R.C. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM, 24 (1981), pp. 381–395. http://dx.doi.org/10.1145/358669.358692.

[4] P. Getreuer, *Linear Methods for Image Interpolation*, Image Processing On Line, 1 (2011). http://dx.doi.org/10.5201/ipol.2011.g_lmii.

[5] G. Haro, A. Buades, and J.M. Morel, *Photographing paintings by image fusion*, SIAM Journal on Imaging Sciences, 5 (2012), pp. 1055–1087. http://dx.doi.org/10.1137/120873923.

[6] N. Limare, A.B. Petro, C. Sbert, and J-M. Morel, *Retinex Poisson Equation: a Model for Color Perception*, Image Processing On Line, 1 (2011). http://dx.doi.org/10.5201/ipol.2011.lmps_rpe.

[7] D.G. Lowe, *Object recognition from local scale-invariant features*, in Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, 1999, pp. 1150–1157. http://dx.doi.org/10.1109/ICCV.1999.790410.

[8] J-M. Morel and G. Yu, *ASIFT: A new framework for fully affine invariant image comparison*, SIAM Journal on Imaging Sciences, 2 (2009), pp. 438–469. http://dx.doi.org/10.1137/080732730.

[9] P. Pérez, M. Gangnet, and A. Blake, *Poisson image editing*, in ACM Transactions on Graphics (TOG), vol. 22, ACM, 2003, pp. 313–318. http://dx.doi.org/10.1145/882262.882269.

[10] G. Yu and J-M. Morel, *ASIFT: An Algorithm for Fully Affine Invariant Comparison*, Image Processing On Line, 1 (2011). http://dx.doi.org/10.5201/ipol.2011.my-asift.